

中学生电脑乐园丛书

快乐程序随我编

——Visual Basic 编程小实例

林强 编著

人民邮电出版社

图书在版编目 (C I P) 数据

快乐程序随我编: Visual Basic 编程小实例 / 林强编著. —北京: 人民邮电出版社, 2002.11
(中学生电脑乐园丛书)

ISBN 7 - 115 - 10694 - 0

. 快... . 林... . BASIC 语言 - 程序设计 - 中学—课外读物 IV . G634.673

中国版本图书馆 CIP 数据核字 (2002) 第 079952 号

中学生电脑乐园丛书

快乐程序随我编

——Visual Basic 编程小实例

◆ 编 著 林 强

责任编辑 邓革浩

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号

邮编 100061 电子函件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

读者热线 010-67180876

北京汉魂图文设计有限公司制作

印刷厂印刷

新华书店总店北京发行所经销

◆ 开本: 787×1092 1/16

印张: 11

字数: 262 千字 2002 年 3 月第 1 版

印数: 1 - 0 000 册 2002 年 3 月北京第 1 次印刷

ISBN 7-115-10694-7/TP · 2617

定价: 00.00 元

本书如有印装质量问题, 请与本社联系 电话: (010) 67129223

内容提要

本书是以 Visual Basic 6.0 为工具，以实例为主的学习编程的辅导用书。因此本书从各种趣味性与实用性较强的实例出发，让读者通过不断地实践，编写出各种实用程序，在实践中不断地体会 Visual Basic 的易用性和高效性。

全书共有 7 章，分别是 VB 初探、用户界面设计、图形处理、多媒体、数据库、系统控制、网络应用程序、自给自足。每一章都提供了丰富的实例，通过这些实例较完整地讲解了 Visual Basic 6.0 程序开发的方法和技巧，在此基础上通过各章节的“试一试”为读者提供了一个思考与提高的机会。

本书可以作为在校高、初中学生学习编程语言的自学辅导用书，同时可以作为高中信息技术教材的参考用书。对于有兴趣学习编程的初学者来说，这也是一本非常好的入门与提高类的书。

丛 书 序

目前，全国各地的中小学已普遍开设了信息技术课，使学生有机会了解和掌握信息技术的基本知识和基本技能。但是信息技术的内涵是丰富的，为了配合学校的信息技术课的开展，让学生能掌握更多的电脑技能，提高动手制作的能力，我们特邀请经验丰富的信息技术课教师编写了这套《中学生电脑乐园》丛书。

本丛书包括以下 5 本：

《轻松上网任我行》

《精彩网站由我建》

《趣味动画一点通》

《快乐程序随我编——Visual Basic 编程小实例》

《动感多媒体自己制——Authorware 制作小实例》

本丛书定位为中学生课外电脑辅导和扩展知识的书籍，使中学生在完成课堂内的信息技术教学后，通过对本丛书的学习，能在尽可能短的时间内，达到较高的电脑制作水平，并精通各种信息技术。

本丛书打破传统教科书的讲授模式，而是贯穿了以下原则：

1. 精心设计了若干个能调动学生学习的有趣的小例子，通过完成一个个小例子，带动了应用软件的学习，较好地体现了“任务驱动”的教学模式和“寓教于乐”的教学思想。
2. 注重了循序渐进的教学原则。全书由浅入深，由易到难，结构合理，脉络清晰。
3. 内容的实用性、可模仿性强。书中的许多例子，可以直接模仿，或稍做修改就能用于学生学习和在校园内使用。
4. 动手操作与知识讲解相辅相成，书中以制作实例为主线，中间穿插知识讲解。
5. 全书图文并茂，语言生动活泼，通过实例操作与练习题的相互配合，起到了举一反三、触类旁通的效果。

相信本丛书一定能成为中学生学习电脑的良师益友。

编者

前 言

本书是《中学生电脑乐园》系列丛书中的一本。

Visual Basic 6.0 是微软公司推出的可视化编程工具。它为编程提供了一条便捷之路，让我们远离了编程时的枯燥和复杂，同时也使编程不再神秘，即便是初学者也能轻松上手。但是学好编程毕竟不是一件简单的事情，Visual Basic 是很容易入门，但要精通也实属不易。当今市场上关于 Visual Basic 编程的书有很多，但是有些书厚而全且缺少实例，适合作为编程手册而不太适合自学，而许多教科书又因为系统性、严谨性及层次性的限制，往往过分强调语法、结构等，提供的实例较少，特别是一些实用性强的例子（比如系统控制、游戏等）。因此，本书从各种实例出发，让读者通过不断地实践，学习编写各种实用程序，在实践中逐渐体会 Visual Basic 的易用性及高效性。

本书通过大量的趣味性和实用性较强的实例，系统地讲解了 Visual Basic 编程的多种技巧。全书共有 8 章：第 1 章主要讲解 Visual Basic 的一些历史、基本操作以及学习的方法；第 2 章讲解用户界面设计，通过大量的实例讲述了窗体、菜单的制作方法和美化的技巧以及一些公共对话框的应用方法；第 3 章讲解图形处理实例，主要涉及颜色处理及图形处理的方法和技巧；第 4 章讲解多媒体制作实例，主要是动画、音频和视频的处理方法以及音视频播放器的制作；第 5 章讲解数据库应用实例，主要内容是数据库的建立以及数据的浏览、编辑、追加和删除；第 6 章讲解系统控制实例，主要内容是在应用程序中如何调用系统功能来控制操作系统；第 7 章讲解网络应用程序实例，主要介绍如何通过调用其他应用程序及由它们提供的控件来构建网络应用程序；第 8 章主要讲解游戏的编程实例，其中有几个小游戏的完整代码，旨在提高读者的编程水平。

本书中的所有程序都经过了严格的调试，调试环境为 Visual Basic 6.0，操作系统为 Windows 98、Windows 2000 及 Windows XP。由于编者学识有限，书中难免有疏漏之处，欢迎广大读者批评指正。

编者

目 录

第1章 VB 初探

1.1 Visual Basic 的“身世”	1
1.1.1 Basic 语言	1
1.1.2 可视化的编程工具——Visual Basic.....	1
1.2 初试身手	2
1.2.1 启动 Visual Basic	2
1.2.2 Visual Basic 集成环境	2
1.2.3 和大家打个招呼	4
1.3 找个好老师	12
1.3.1 使用 F1 键快速获得上下文相关的帮助	12
1.3.2 获得联机帮助	13
1.3.3 请教“大虾”	14
1.4 小结	14

第2章 用户界面设计

2.1 窗体设计	15
2.1.1 美化我的窗体	15
2.1.2 一块精美的电子表	20
2.1.3 我可不想挡住别人	22
2.2 菜单设计	24
2.2.1 我也能做菜单	24
2.2.2 弹出式菜单	26
2.2.3 好看的图标菜单	28
2.2.4 伸缩自如的动态菜单	30
2.3 对话框	32
2.3.1 应用程序的信使	32
2.3.2 应用程序的眼睛和耳朵	33
2.3.3 名片——制作自己的“关于对话框”	34
2.3.4 拿来主义	36
2.4 小结	38

第3章 图形处理

3.1 颜色处理	39
----------------	----

3.1.1	一条特殊的彩虹	39
3.1.2	调色板	41
3.2	做一幅动画	43
3.2.1	走马灯	43
3.2.2	百叶窗	45
3.2.3	开动的汽车	46
3.2.4	旋转的图形	47
3.2.5	不知疲倦的小球	51
3.3	做一个放大镜	54
3.3.1	放大和缩小文字	54
3.3.2	简单的图形放大镜	56
3.3.3	放大和缩小屏幕的内容	58
3.4	小结	60
第 4 章 多媒体制作		
4.1	制作自己的动画光标	61
4.1.1	自制一个简单的动画光标	61
4.1.2	调用系统的动画光标	64
4.2	简单的动画	67
4.2.1	天降瑞雪	67
4.2.2	节日彩灯	69
4.3	制作播放器	71
4.3.1	来一段美妙的音乐	71
4.3.2	我也能放电影	72
4.3.3	做一个自己的“超级解霸”	76
4.4	小结	80
第 5 章 数据库应用		
5.1	建立数据库	81
5.1.1	制作电子卡片	81
5.1.2	资料存档	83
5.2	我的“电子同学录”	84
5.2.1	浏览好友资料	89
5.2.2	加入好友	92
5.2.3	编辑好友资料	94
5.2.4	整理好友资料	96
5.3	小结	98
第 6 章 系统控制		

6.1 使用程序修改系统时间	99
6.2 改变任务栏的状态	103
6.3 自动关闭计算机	104
6.4 模拟“关闭程序”	106
6.5 小结	110
第 7 章 网络应用	
7.1 拨号上网	111
7.2 超级链接	113
7.3 自己的 Foxmail	114
7.4 做个浏览器	119
7.5 小结	126
第 8 章 编程实例	
8.1 俄罗斯方块	127
8.2 跳跳球	148
8.3 打蟑螂	155
8.4 小结	165



第 1 章 VB 初探

我们知道程序设计语言有很多种，如 PASCAL、BASIC、Visual Basic、JavaScript 等，其中 Visual Basic（简称为 VB）是微软公司（Microsoft）于 1991 年推出的可视化 BASIC 语言，在语言功能方面它与 BASIC 语言基本兼容，是众多的编程语言中的佼佼者。它是 Windows 环境下最简单易学的编程语言，为我们提供了一个所见即所得的界面设计和面向对象的良好环境。



本章的主要内容如下：

- (1) Visual Basic 的历史；
- (2) Visual Basic 的集成环境及应用；
- (3) Visual Basic 中的帮助功能。

1.1 Visual Basic 的“身世”

1.1.1 Basic 语言

Basic 是美国 Dartmouth 学院的两位学者发明的，其含义为“初学者通用语言”。20 世纪 70 年代中期，比尔·盖茨为微型机配置了 BASIC 语言，并由此开设了大名鼎鼎的微软公司（Microsoft）。

1.1.2 可视化的编程工具——Visual Basic

随着 Windows 操作系统的出现，广大用户拥有了一个直观并且操作简便的工作环境。但是这种出色的环境，导致程序员们编程的难度增加了很多，因为他们需要自己编制程序去实现窗口、菜单、对话框等用户界面。这个问题一直到微软公司于 1991 年推出 Visual Basic 软件才得到了解决。

所谓“Visual”，指的是可视化，也就是指此软件是用来开发图形用户界面的。“Visual Basic”就是一种图形化的程序设计语言，可以用来开发各种应用程序。它将窗口、菜单、对话框等都做成了一些工具、编辑器以及函数等，使得无论是初学编程者，还是经验丰富的程序员都无需自己去编制内部代码，只要用几条简单的控制语句就可以了。

在 Visual Basic 中首次采用了事件驱动来进行编程，这是一种适用于图形用户界面的



开发方法，它能够让用户自己控制程序什么时候该做什么事情，而且还不必按执行顺序一步步指定精确步骤，只需在相应控件的代码窗口中编写出应该进行的操作就可以了，软件会自动按顺序执行。

1.2 初试身手

前面我们已经和 Visual Basic 有了初步接触，了解了一些它的身世和它的一些长处，我们不妨编写一个小程序来进一步熟悉它。

1.2.1 启动 Visual Basic

单击“开始/程序/Microsoft Visual Basic 6.0 中文版/Microsoft Visual Basic 6.0 中文版”启动 Visual Basic，我们将看到如图 1-1 所示的“新建工程”对话框。



图 1-1 新建工程对话框

这里是让我们选择要建立的程序的类型，一般我们常用的是“标准 EXE”。

1.2.2 Visual Basic 集成环境

单击新建工程对话框中的“标准 EXE”图标，会出现一个如图 1-2 所示的 Visual Basic 集成开发环境窗口，它主要包括以下几个部分。

(1) 标题栏

位于窗口的顶部，显示窗口的标题，同时也显示出当前的工作状态，即“设计”、“中断”和“运行”。



(2) 菜单栏

位于标题栏的下方，共包括 13 个下拉式菜单。它的操作方法与其他 Windows 应用程序中的菜单完全相同。单击菜单，再单击相应命令便可执行相应操作。

(3) 工具栏

标准工具栏包含了 VB 中最常用的一些命令，如打开、保存、新建、复制、粘贴、窗口等。使用工具栏中的按钮，不必记忆多级联的菜单，可快速地执行操作。

(4) 工具箱

位于窗体的左部，使用这些工具可以在窗体上构造出所需的各种程序界面。

(5) 窗体窗口

位于屏幕正中，相当于一块画板。可以根据需要放入工具箱中的各种工具，在窗体上书写文字、绘图，使程序界面更美观。

(6) 工程窗口

工程窗口列出的是当前应用程序所包含的文件清单。

(7) 属性窗口

属性窗口列出了所选对象的所有属性，这些属性可按字母顺序排列，也可按性质分类排列。属性窗口可通过标签来切换。

(8) 窗体布局窗口

主要用于窗体位置的调整。工程中所有的窗体都会在窗体布局窗口中显示出来，欲调整窗体位置，用鼠标直接拖动窗体至适当的位置即可。

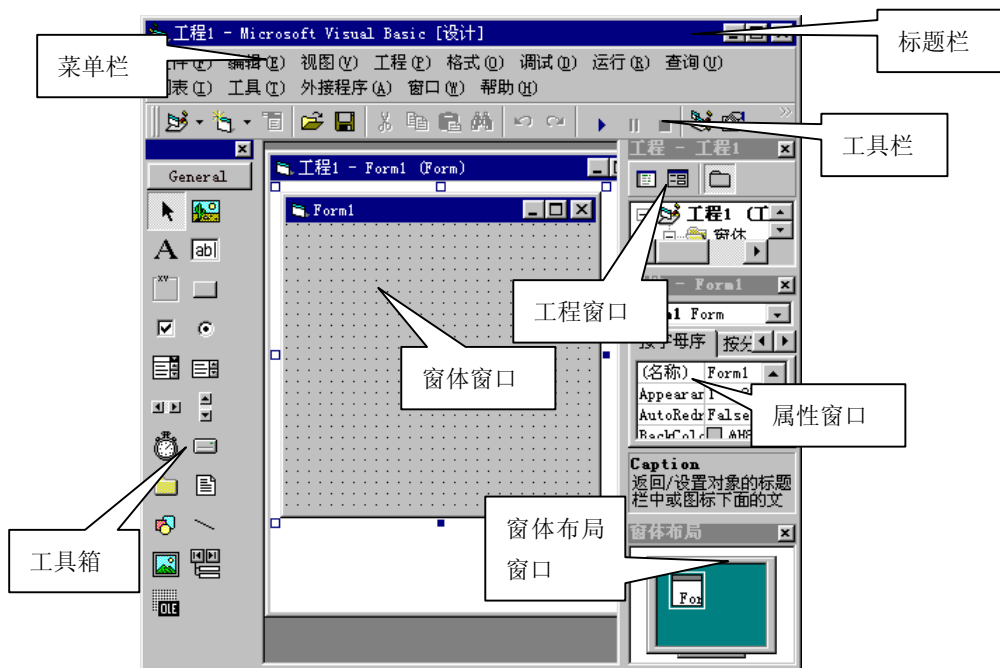


图 1-2 集成开发环境



1.2.3 和大家打个招呼

接下来我们就编写一个“打招呼”的程序吧！简单一点，就从“Hello world!”入手。首先我们应该看到，在 Visual Basic 的集成开发环境中有一个标题为“Form1”的窗体，如图 1-3 所示。

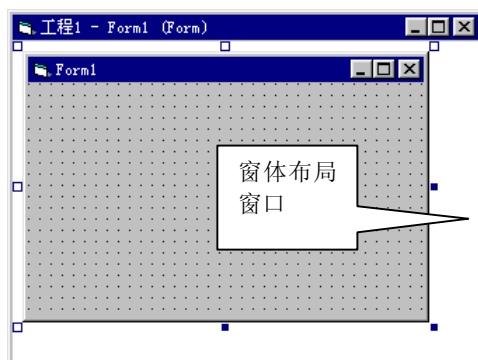


图 1-3 窗体窗口

该窗体是我们用来创建应用程序界面的，它所展示的内容我们将在程序运行时看到。这就是所谓的所见即所得的可视化程序设计。那么我们究竟怎样进行这种设计呢？首先可以根据程序的功能先来确定人机交互的界面，也即应用程序的界面，然后根据界面从工具箱中选取适当的控件，最后把控件放入窗体之中。如图 1-4 所示为包括 Label 控件的工具箱。



图 1-4 工具箱

好，既然我们要在程序窗口上显示“Hello world!”，那就需要在窗体上放一个标签框（Label），具体的步骤如下。

（1）双击工具箱的标签框（Label）控件，窗体中央会自动建立一个标签框对象，如图 1-5 所示。

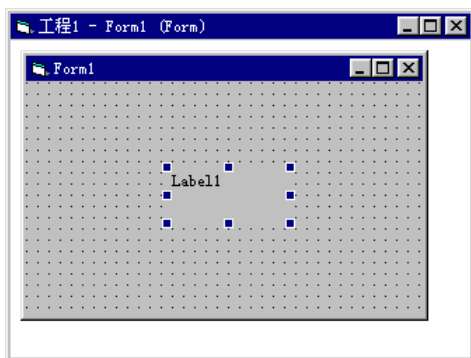


图 1-5 加入标签框

**小技巧：**

在工具箱上单击标签框控件，然后在窗体中拖拽鼠标光标，就可以建立一个自定义大小的控件。

(2) 接下来我们改变对象的各种属性以使之符合我们的需要，本实例中我们将修改标签框的四个属性。

① 选中要改变属性的控件，这时属性对话框的标题将显示为控件对象的名称，在本例中应该是“Label1”，如图 1-6 所示。然后根据需要修改 Label1 的 Caption 属性，该属性将决定标签框在运行时所显示的内容。单击 Caption 属性右边的属性值后输入“Hello world!”

② 改变 AutoSize 属性为 True，如图 1-6 所示。该属性将使标签框的大小自动适应 Caption 的内容，当 Caption 值的长度或字号变化时，标签框的大小也将随之变化。我们单击 AutoSize 属性右边的属性值时，将弹出一个下拉列表框供我们选择属性，如图 1-7 所示。

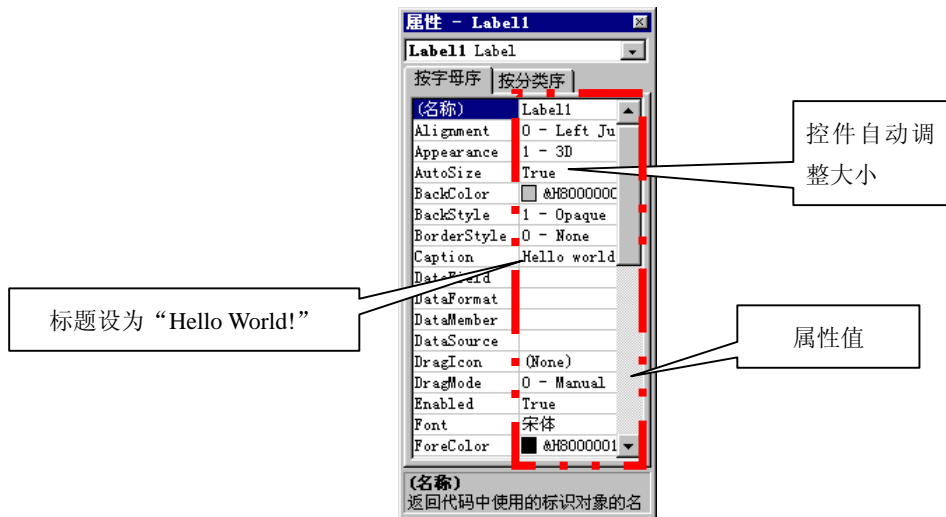


图 1-6 设置 Caption 及 AutoSize 属性

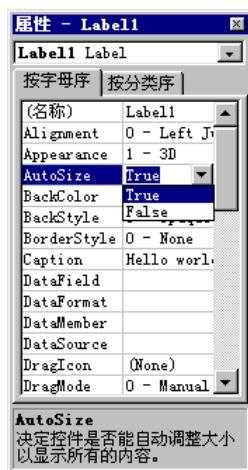


图 1-7 属性值的下拉列表框

当然随着属性值的不同，具体情况也各不相同。如图 1-8 所示，单击 **BackColor** 的属性值将弹出颜色选择对话框。一般情况下当鼠标单击属性值时，如果该行出现“**I**”型光标，表示要求输入一个值；如果该行的右边出现“**▼**”，表示可以在下拉列表框中选择一个值；如果该行的右边出现“**...**”，表示可以单击该按钮来打开一个对话框以确定属性值。



图 1-8 属性值的弹出对话框

③ 改变 **Left** 和 **Top** 属性，这两个属性决定了控件在窗体中的位置（通常确定控件的左上角）。本实例中我们把标签框左上角的坐标定为（0，0），如图 1-9 所示。

（3）现在我们只要单击“运行/启动”菜单或按 **F5** 键就可以运行程序了，结果如图 1-10 所示。

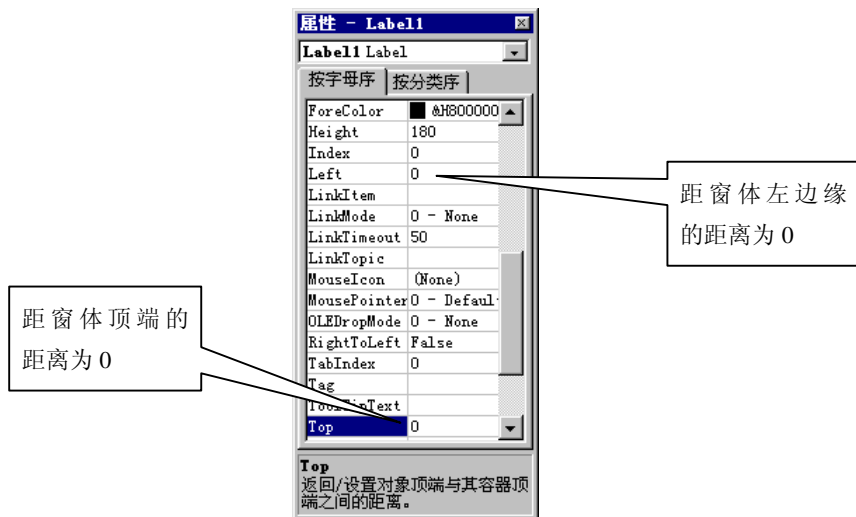


图 1-9 设置 Left 及 Top 属性

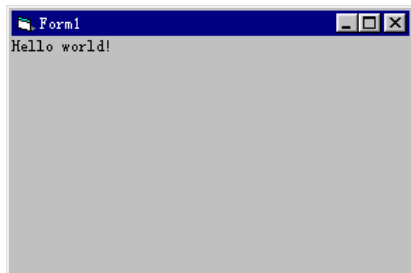


图 1-10 运行结果



小技巧：

单击快捷工具栏上的“▶”按钮就可以运行程序。

到现在为止，我们没有写任何一句代码，一个应用程序就可以完整地运行了，也就是说，Visual Basic 已经帮我们做好了许多事情，比如在程序运行过程中窗口的建立、打开和关闭等，我们不用把过多的精力放在程序界面的设计上，而可以把主要的精力都集中到程序功能的实现上。这是与 C 语言等其他非可视化语言的区别。大家可能已经注意到了上面的程序很简单，运行起来也不美观，那么我们就给它做个“手术”吧！

(4) 现在我们向程序里添加代码以增加一些的功能。在窗体上单击鼠标右键，将弹出一个快捷菜单，如图 1-11 所示，单击“查看代码”菜单将打开一个代码窗口，如图 1-12 所示。



图 1-11 窗体弹出菜单

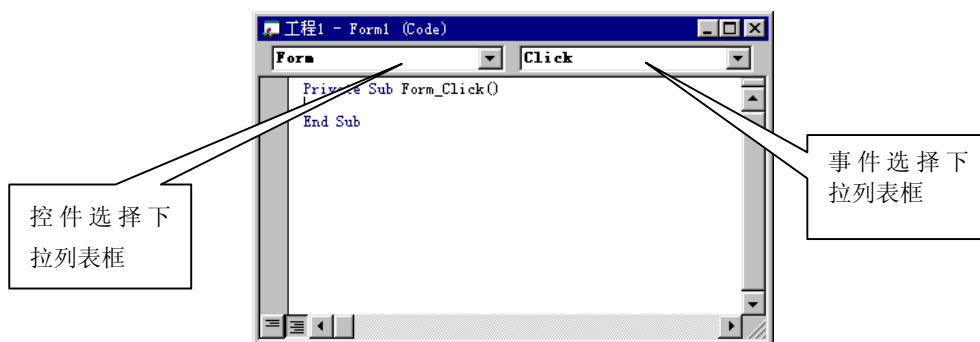


图 1-12 控件及过程的选择

这里是让我们编写控制窗体行为的实际代码。由于 Visual Basic 应用程序是按照事件驱动来编写的，即每个具体发生的事件都有一个相应的处理程序来解决，如果没有相应的处理程序，即表示该事件对窗体没有任何影响。我们必须在窗口中选择一个具体的事件来编写代码，那么我们不妨设定在鼠标单击窗体时产生一些动作。我们首先应该在控件选择下拉列表框中选取 Form，然后在过程选择下拉列表框中选取 Click 事件来编写代码，具体情况如图 1-13 所示。

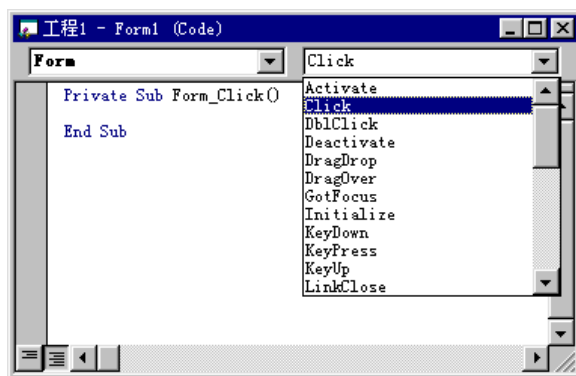


图 1-13 选取 Click 过程

Visual Basic 中的窗体对一个单击事件的发生作出认定时，将自动用相应事件的名字来调用该事件的过程，也就是说事件过程是附加在窗体和控件上的。一个控件的事



件过程的名称是将控件的实际名字（即在 Name 属性中规定的名字）、下划线（_）和事件名组合起来，例如上面的 Form 窗体的单击事件，图 1-13 中显示其名称为 Form_Click。

接下来我们添加如下代码（其中的黑体部分）：

```
Private Sub Form_Click()  
    For i = 1 To 100  
        Load Label1(i)  
        Label1(i).Move Label1(i - 1).Left + 150, Label1(i - 1).Top_  
+ 150  
        Label1(i).Caption = "Hello world!"  
        Label1(i).Visible = True  
    Next i  
End Sub
```

其中 Load 是一个过程，它的作用是在程序运行过程中，动态地向窗体中加入一个控件，在本实例中用该过程装载了 Label 控件数组的一个元素。而 Move 过程则是 Label 控件的一个方法，它用来移动 Label 控件的位置，在本实例中用它来实现相邻两个控件的错位，使新生成的控件的左上角坐标相对于原控件的左上角坐标分别向右和向下移动 150 个缇（一个与屏幕无关的单位）。

为了使该程序能正常运行，我们必须把原来的 Label1 控件改为一个控件数组。具体操作如图 1-14 所示，把 Label1 的 Index 属性设置为 0。



试一试：

观察控件的属性窗口的标题有什么变化，想一想为什么？

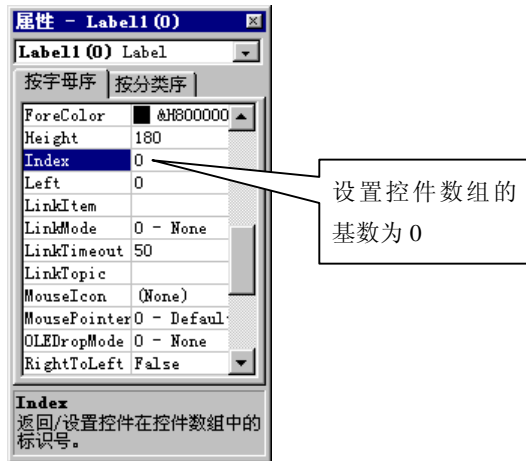


图 1-14 设置控件数组

现在运行该程序并在窗体中单击鼠标左键，其结果将如图 1-15 所示。

（5）图 1-15 所示的结果看上去是要好一些了，但还欠美观。不妨加入字号变化，此时代码必须作如下修改（其中的黑体部分）：



图 1-15 运行结果

```
Private Sub Form_Click()  
For i = 1 To 100  
    Load Label1(i)  
    Label1(i).Move Label1(i - 1).Left + 150, Label1(i - 1).Top_  
+ 150  
    Label1(i).Caption = "Hello word!"  
    Label1(i).FontSize = Label1(i - 1).FontSize + 2  
    Label1(i).Visible = True  
Next i  
End Sub
```

其运行结果如图 1-16 所示。



图 1-16 运行结果

(6) 好，问题出来了，在图 1-16 中随着字号的不断增大，到后来字出现重叠现象，导致无法看清。怎样解决这个问题呢？我们只要对代码作进一步修改即可。

```
Private Sub Form_Click()  
For i = 1 To 100  
    Load Label1(i)  
    Label1(i).Move Label1(i - 1).Left + 150, Label1(i - 1).Top_  
+ Label1(i - 1).Height  
    Label1(i).Caption = "Hello word!"  
    Label1(i).FontSize = Label1(i - 1).FontSize + 2  
    Label1(i).Visible = True  
Next i  
End Sub
```



改“Label1(i).Move Label1(i - 1).Left + 150, Label1(i - 1).Top + 150”为“Label1(i).Move Label1(i - 1).Left + 150, Label1(i - 1).Top + Label1(i - 1).Height”，这样可以使每一个新生成的 Label1 元素的坐标随着前一个元素的高度变化而变化。其运行结果如图 1-17 所示。



图 1-17 运行结果

(7) 现在的问题是图 1-17 的最后一行还不行，看来还得修改代码：

```
Private Sub Form_Click()  
    Dim i As Integer  
    i = 0  
    Do While True  
        i = i + 1  
        Load Label1(i)  
        Label1(i).Move Label1(i - 1).Left + 150, Label1(i - 1).Top_  
+ Label1(i - 1).Height  
        Label1(i).Caption = "Hello word!"  
        Label1(i).FontSize = Label1(i - 1).FontSize + 2  
        If Label1(i).Top + Label1(i).Height >=_  
Form1.ScaleHeight Then  
            Exit Do  
        End If  
        Label1(i).Visible = True  
    Loop  
End Sub
```

首先改 For 循环为 Do 循环，然后判断当前控件是否超出窗体，当控件超出窗体时就不再显示该控件并退出循环。这样得到的最终结果如图 1-18 所示。



图 1-18 运行结果



1.3 找个好老师

在我们实际的编程过程中，也会像制作上面的实例一样，遇到各种各样的问题，所以要求我们要多次修改程序，因为有可能会遇到我们无法解决的问题，这就需要找一个好老师。其实 Visual Basic 的帮助内容十分丰富，它包含在“MSDN library VisualStudio 6.0”中，在安装 Visual Basic 时可以根据提示选择安装，也可以以后单独安装 MSDN。在以后使用 Visual Basic 时，如果遇到了困难，可以使用帮助。下面介绍获取帮助的几种方法。

1.3.1 使用 F1 键快速获得上下文相关的帮助

Visual Basic 的许多部分是上下文相关的 (Context Sensitive)，这意味着你可以直接得到这部分的帮助，而不必通过菜单查找。例如，要想得到 Form 的 Caption 属性的帮助，可以在属性窗口中选中 Caption 属性，然后按 F1 键，就会显示如图 1-19 所示的帮助，并且帮助内容定位于该关键字。

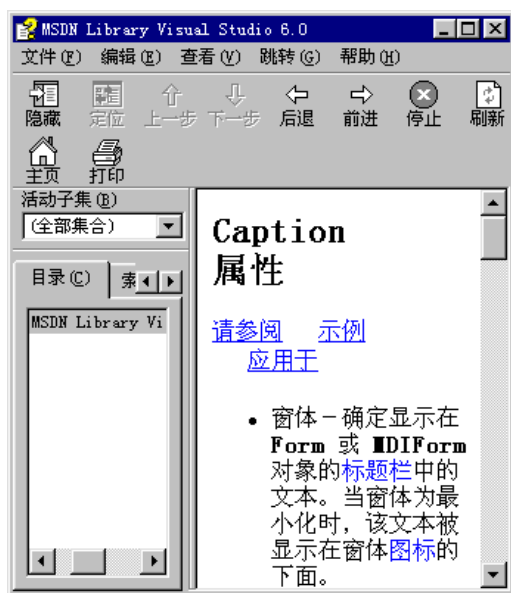


图 1-19 上下文相关帮助

要了解 Timer 控件的属性、事件和方法，可以在工具箱中选中 Timer 控件，然后按 F1 键，会显示如图 1-20 所示的帮助。

在 Visual Basic 中上下文相关的地方有许多，主要有以下几个部分：

- (1) Visual Basic 中的每一个窗口（如属性窗口、代码编辑窗口等）；



- (2) 工具箱中的控件;
- (3) 窗体中的对象或文档对象;
- (4) 属性窗口中的属性项;
- (5) VB 的关键字 (声明、函数、属性、方法、事件及特殊对象);
- (6) 错误信息。

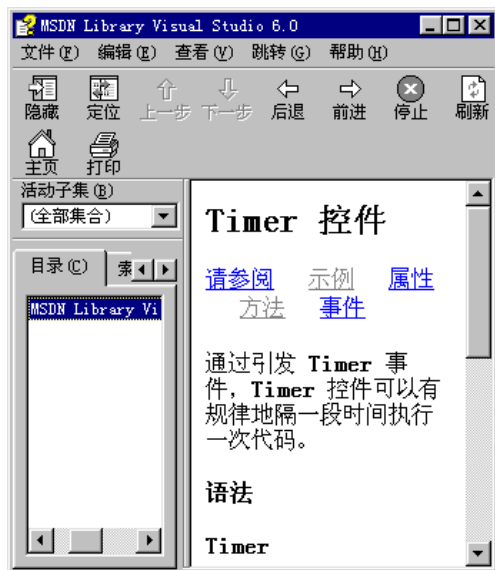


图 1-20 上下文相关帮助

1.3.2 获得联机帮助

这是获得帮助的常用手段。使用 Visual Basic 的帮助菜单可以打开联机帮助，方法如下。

- (1) 在菜单栏上选择“帮助” / “内容”，如图 1-21 所示。



图 1-21 帮助菜单

- (2) 屏幕出现帮助窗口，如图 1-22 所示。

该窗口是按浏览器风格打开的，可以按 [目录]、[索引] 方式来查找帮助。按照目录方式使用帮助就像翻阅一本参考书，如果你明确了要查找的内容，可以使用索引方式，键



入要查找的关键字，然后单击窗口下面的〔显示〕按钮，便可查看帮助内容。



图 1-22 帮助窗口

1.3.3 请教“大虾”

现今 Internet 高速发展，也为我们获取信息、获得知识带来了方便。当我们遇到问题时，就可以通过这个好工具来解决问题：

- (1) 通过 QQ 或 E-mail 向远方的“大虾”请教；
- (2) 通过搜索引擎查找有相关知识的站点；
- (3) 通过论坛来向所有的 Visual Basic 爱好者请教，如 <http://www.csdn.net> 的 Visual Basic 论坛。

1.4 小结

在这一章中，我们学习了 Visual Basic 的一些历史、基本操作以及学习的方法。

在 Visual Basic 的“身世”一节讲述了 BASIC 语言的由来及 Visual Basic 的历史。

在“初试身手”中讲解了 Visual Basic 的集成环境，并通过“Hello world!”程序的编写演示了具体的窗体、工具箱、工具条及菜单的使用方法。

在“找个好老师”一节中则介绍了 Visual Basic 的帮助功能及一些学习的方法。



第 2 章 用户界面设计

任何 Windows 应用程序的界面都非常重要，因为应用程序的界面是人机交流的途径，直接关系一个软件的可操作性及亲和性。



本章的主要内容如下：

- (1) 窗体的设计、美化；
- (2) 应用程序菜单的设计与美化；
- (3) 对话框的设计及公共对话框的使用。

2.1 窗体设计

2.1.1 美化我的窗体

大家都应该看到过如图 2-1 所示的图片吧，它是颜色自上而下逐渐过渡形成的一种效果。怎么样，没想过用编程来实现吧？那么现在就让我们用程序来实现该效果。

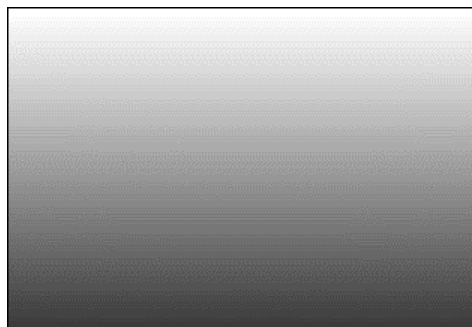


图 2-1 效果图

- (1) 新建“标准 EXE”工程。



(2) 在 Form1 上单击鼠标右键弹出快捷菜单，选择“查看代码”菜单项，打开代码窗口，如图 2-2 所示。



图 2-2 代码窗口

(3) 在代码窗口中输入下列代码：

```
Sub SetBGColor(formhdc As Object, Rval As Integer, Gval_
As Integer, Bval As Integer)
    Dim s, r, topv, leftv, rightv, totalh As Integer
    s = (formhdc.Height / 50)
    topv = 0
    leftv = 0
    rightv = formhdc.Width
    totalh = topv + s
    For r = 1 To 50
        formhdc.Line(leftv,topv)-(rightv,totalh),RGB(Rval,_
Gval, Bval), BF
        Rval = Rval - 4
        Gval = Gval - 4
        Bval = Bval - 4
        If Rval <= 0 Then
            Rval = 0
        End If
        If Gval <= 0 Then
            Gval = 0
        End If
        If Bval <= 0 Then
            Bval = 0
        End If
        topv = totalh
        totalh = topv + s
    Next
End Sub
```

这段代码是一个用户自定义过程，过程名为 SetBGColor。该过程的作用是绘制指定对象的渐变背景色，它的调用格式如下所示：

```
SetBGColor formhdc, Rval, Gval, Bval
formhdc      要设置渐变背景色的对象
Rval         用于合成颜色的红色量
```




Gval 用于合成颜色的绿色量

Bval 用于合成颜色的蓝色量

在该段程序中我们用到了一个方法和一个函数，其中 formhdc.Line 是调用 formhdc 的 Line 方法，其调用格式如下：

object.Line (x1,y1)-(x2,y2),color,[B][F]

object 对象

(x1,y1) 直线或矩形的起点坐标

(x2,y2) 直线或矩形的终点坐标

color 画线时的 RGB 颜色

B 可选，利用对角坐标画出矩形

F 可选，必须与 B 同用表示以矩形边框的颜色来填充矩形

而另外一个 RGB 用于合成颜色的函数，其调用格式如下：

RGB (Rval,Gval,Bval)

Rval 用于合成颜色的红色量

Gval 用于合成颜色的绿色量

Bval 用于合成颜色的蓝色量

(4) 在代码窗口的控件选择下拉列表框中选择“Form”，在“事件下拉列表框”中选择“Activate”事件。输入下列代码：

```
Private Sub Form_Activate()  
SetBGColor Me, 255, 255, 255  
End Sub
```

(5) 按 F5 键运行，效果如图 2-3 所示。

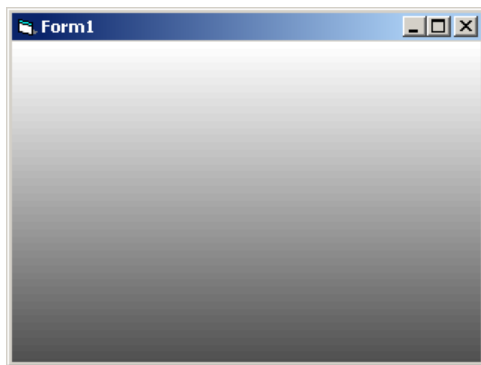


图 2-3 运行效果图

(0, 0, 255) 是蓝色、(255, 0, 0) 是红色、(255, 255, 255) 是白色……这一大堆数字记起来是不是有点勉为其难？那么我们怎么来解决颜色 RGB 值难记的问题呢？我们该记得画图里的颜色编辑对话框（如图 2-4 所示）吧，如果我们能把它利用起来，问题不就迎刃而解了吗？

(1) 在工具箱上单击鼠标右键弹出快捷菜单，选取“部件”菜单项，如图 2-5 所示。

(2) 在打开的部件对话框中选择“控件”页。在“控件”页左边的列表框中找到



“Microsoft Common Dialog Control 6.0 (SP3)”项，并在该项的前面打上勾，如图 2-6 所示，然后单击“确定”按钮。此时，在工具箱上将出现“CommonDialog”按钮，如图 2-7 所示。接下来我们就可以像使用工具箱中的标准控件一样来使用它了。



图 2-4 颜色编辑对话框

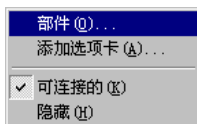


图 2-5 工具箱快捷菜单



图 2-6 部件对话框

(3) 双击“CommonDialog”按钮，在 Form1 窗体中加入 CommonDialog 控件。

(4) 在代码窗口的控件选择下拉列表框中选择“(通用)”，在“事件下拉列表框”中选择“(声明)”事件，在代码窗口中输入下列代码：

```
Function GetRGBval(color As Long) As Integer()
```



```
Dim valR, valG, valB As Integer
Dim val(3) As Integer
valB = color \ 65536
valG = (color - valB * 65536) \ 256
valR = color - valB * 65536 - valG * 256
val(0) = valR
val(1) = valG
val(2) = valB
GetRGBval = val
End Function
```

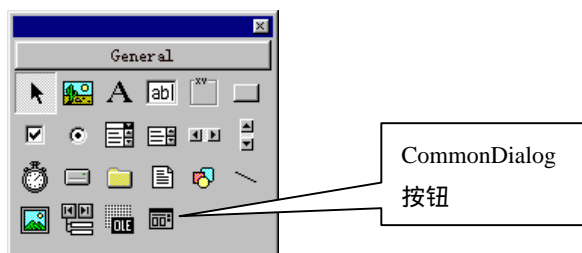


图 2-7 工具箱

这段代码是一个用户自定义函数，函数名为 GetRGBval。该函数的作用是提取合成一个颜色值的红、绿、蓝三原色的各自分量，该函数的调用格式如下所示：

reval=GetRGBval(color)

color 要提取三原色分量的颜色值

reval 返回值，该返回值是一个整型数组。reval(0)存储红色分量、reval(1)存储绿色分量、reval(2)存储蓝色分量

(5) 为 Form1 窗体建立 MouseDown 事件，在代码窗口的控件选择下拉列表框中选择“Form”，在“事件下拉列表框”中选择“MouseDown”事件，并输入下列代码：

```
Private Sub Form_MouseDown(Button As Integer, Shift As_
Integer, X As Single, Y As Single)
    If (Button = vbLeftButton) Then
        Dim sz() As Integer
        CommonDialog1.ShowColor
        sz() = GetRGBval(CommonDialog1.color)
        SetBGColor Form1, sz(0), sz(1), sz(2)
    Else
        End
    End If
End Sub
```

(6) 按 F5 键运行程序。单击鼠标左键，将弹出如图 2-4 所示的颜色编辑对话框，在其中选择一种颜色即可看到该颜色的渐变效果，单击鼠标右键将退出程序。



试一试：

1. 使颜色过渡次序倒过来（黑色在上）。
2. 修改过程 SetBGColor，使它的颜色过渡次序可以随参数变化而变化（再



加入一个参数)。

2.1.2 一块精美的电子表

在 Windows 操作系统中，四四方方、中规中矩的窗口是不是已经让你无法体会美感，而 Windows XP 的华丽窗口又对你有致命的吸引？应该来包装一下我们的应用程序窗口。

(1) 新建“标准 EXE”工程。

(2) 在 Form1 上添加一个 Label1 控件并使之居中，另外添加一个 Timer1 控件，如图 2-8 所示。

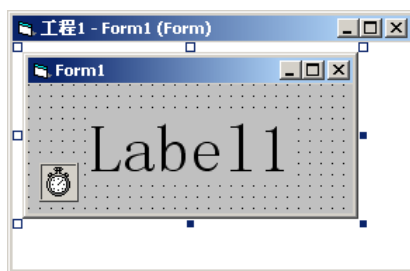


图 2-8 工程窗体

(3) 设置 Timer1 的 Interval 属性值为 1000 毫秒，如图 2-9 所示。

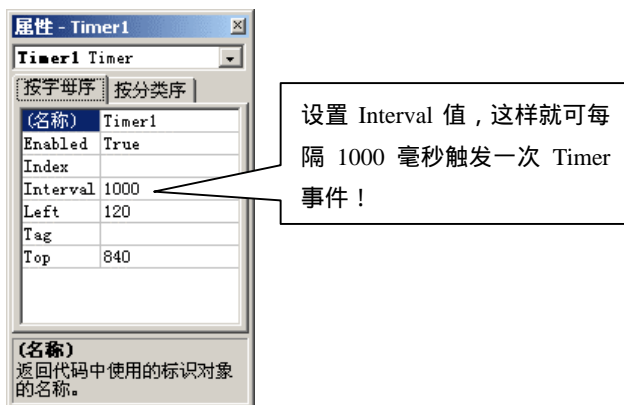


图 2-9 Timer1 属性窗口

(4) 双击窗体，打开代码编辑窗口，在其中添加代码：

```
Private Sub Form_Load()  
    Label1.Caption = Time()  
End Sub
```

Time 函数返回系统当前时间。

(5) 在代码窗口的控件选择下拉列表框中选择“Timer1”，在“事件下拉列表框”中



选择“Timer”事件，在代码窗口中输入下列代码：

```
Private Sub Timer1_Timer()  
    Label1.Caption = Time()  
End Sub
```

(6) 按 F5 键运行程序，效果如图 2-10 所示。

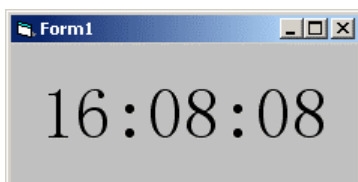


图 2-10 运行效果图

Visual Basic 的窗体都是如此，四四方方的显得太呆板。现在我们就用 Windows API 来对其进行改造，为此我们必须对原工程进行一定的修改。

(1) 设置窗体的 BackColor 属性为 &H00C0C000&，BorderStyle 属性为 0-None。

(2) 设置 Label1 的 BackStyle 属性为 0-Transparent (透明)。

(3) 在代码窗口的控件选择下拉列表框中选择“(通用)”，在“事件下拉列表框”中选择“(声明)”事件，在代码窗口中输入下列代码：

```
Option Explicit  
Private Declare Function CreateEllipticRgn Lib "gdi32" _  
    (ByVal nLeftRect As Long, ByVal nTopRect As Long, ByVal _  
    nRightRect As Long, ByVal nBottomRect As Long) As Long  
Private Declare Function SetWindowRgn Lib "user32" (ByVal _  
    hWnd As Long, ByVal hRgn As Long, ByVal bRedraw As Long) As _  
    Long
```

在这段代码中我们定义了两个函数：CreateEllipticRgn 和 SetWindowRgn，但没有给出具体代码。其实它们是 Windows 操作系统提供的，分别位于 GDI32.DLL 和 USER32.DLL 这两个动态链接库中。

其中 CreateEllipticRgn 用于建立一个椭圆或圆形区域，其格式如下所示：

```
CreateEllipticRgn ( nLeftRect, nTopRect, nRightRect, nBottomRect )  
nLeftRect          区域左上角的横坐标  
nTopRect            区域左上角的纵坐标  
nRightRect          区域右上角的横坐标  
nBottomRect         区域右上角的纵坐标
```

而 SetWindowRgn 则用于设置窗口的区域形状，其格式如下所示：

```
SetWindowRgn ( hWnd, hRgn, bRedraw )  
HWnd              要设置形状的窗口的句柄  
hRgn               区域句柄，由 CreateEllipticRgn 等函数返回  
bRedraw            窗口设置形状后是否刷新
```

(4) 修改 Form1 的 Load 事件代码：

```
Private Sub Form_Load()
```



```
Dim h, d As Long
Dim scrw, scrh As Long
scrw = Me.Width / Screen.TwipsPerPixelX
scrh = Me.Height / Screen.TwipsPerPixelY
h = CreateEllipticRgn(0, 0, scrw, scrh)
d = SetWindowRgn(Me.hWnd, h, True)
Label1.Caption = Time()
End Sub
```

该程序段的作用是把窗口设置为椭圆。

因为 Windows API 一般需要以像素为度量单位,所以必须把 scrw,scrh 的值转化为以像素为单位的值,而 Screen 对象的 TwipsPerPixelX,TwipsPerPixelY 属性是返回水平或垂直度量的对象的每一个像素中的缇数(一个与屏幕无关的单位)。

(5) 为 Form1 窗体建立 DblClick 事件,在代码窗口的控件选择下拉列表框中选择“Form”,在“事件下拉列表框”中选择“DblClick”事件,并输入下列代码:

```
Private Sub Label1_DblClick()
End
End Sub
```

因为本窗体为无标题窗体,所以必须为程序加入退出代码,否则程序无法关闭。

(6) 按 F5 键运行程序,效果如图 2-11 所示。



图 2-11 运行效果图



试一试:

够酷吧?那么就来做一圆形窗口如何?

2.1.3 我可不想挡住别人

我想能做隐形人是很多人的梦想,不过至今无法实现。现在就让我们来做个隐形窗口,以便让它背后的图形能显示出来。

(1) 新建“标准 EXE”工程。

(2) 为工程添加模块,单击“工程”菜单,选择“添加模块”菜单项,如图 2-12 所示。

(3) 在模块代码窗口中输入代码:

```
Declare Function CreateCompatibleBitmap Lib "gdi32" _
    (ByVal hdc As Long, ByVal nWidth As Long, ByVal nHeight As _
    Long) As Long
```



```
Declare Function SelectObject Lib "gdi32" (ByVal hdc As_  
Long, ByVal hgdiobj As Long) As Long
```



图 2-12 工程菜单

在这段代码中我们定义了两个函数：CreateCompatibleBitmap 和 SelectObject，这两个函数由动态链接库 GDI32.DLL 提供。

其中 CreateCompatibleBitmap 用于创建一个与当前窗体相关联的兼容位图句柄，其格式如下所示：

```
CreateCompatibleBitmap ( hdc,nWidth,nHeight )
```

hdc 图形设备句柄

nWidth 位图宽度

nHeight 位图高度

而 SelectObject 则用于更换当前设备句柄，其格式如下所示：

```
SelectObject ( hdc,hgdiobj )
```

hdc 图形设备句柄的窗口句柄

hgdiobj 设置成为当前句柄的图形对象句柄，由 CreateCompatibleBitmap 等函数返回

(4) 设置窗体的 BorderStyle 属性为 0-None。

在窗体中放置一 TextBox 控件，其 Text 属性设置为：“透明窗体的实心控件”，其 Alignment 属性设置为：“2-Center”。

在窗体中放置一 CommandButton 控件，其 Caption 属性设置为“退出”，如图 2-13 所示。

(5) 在代码窗口的控件选择下拉列表框中选择“(通用)”，在“事件下拉列表框”中选择“(声明)”事件，在代码窗口中输入下列代码：

```
Dim obj
```

(6) 在代码窗口的控件选择下拉列表框中选择“Form”，在“事件下拉列表框”中选择“Load”事件，在代码窗口中输入下列代码：



```
Private Sub Form_Load()  
    Dim hBitmap  
    Me.AutoRedraw = True  
    hBitmap = CreateCompatibleBitmap(Me.hdc, 0, 0)  
    obj = SelectObject(Me.hdc, hBitmap)  
    Me.Refresh  
End Sub
```

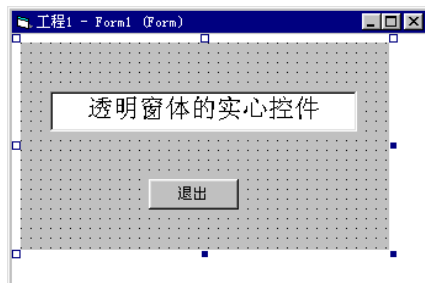


图 2-13 工程窗体

其中 `hBitmap = CreateCompatibleBitmap(Me.hdc, 0, 0)` 用来创建一个与当前窗体相关联的兼容位图句柄，只要更改该函数的 `nWidth` 和 `nHeight` 两个参数就可以控制窗体透明的范围。而 `obj = SelectObject(Me.hdc, hBitmap)` 则用来将当前窗口的图形设备句柄更换为 `hBitmap`，并返回原设备句柄至 `obj`。

(7) 在代码窗口的控件选择下拉列表框中选择“Command1”，在“事件下拉列表框”中选择“Click”事件，在代码窗口中输入下列代码：

```
Private Sub Command1_Click()  
    End  
End Sub
```

(8) 按 F5 键运行程序，效果如图 2-14 所示。



图 2-14 运行效果图



2.2 菜单设计

2.2.1 我也能做菜单

Windows 应用程序与以前的 DOS 应用程序最大的区别，就是 Windows 应用程序使用了图形化界面，从而大大地方便了使用者，其中的菜单集中体现了应用程序的功能，让人一目了然，而这个好东西做起也并不难。

(1) 新建“标准 EXE”工程，并把工程保存为 MyApp.vbp。

(2) 单击“工具”菜单，选取菜单项“菜单编辑器”，如图 2-15 所示。



图 2-15 菜单

(3) 在菜单编辑器（如图 2-16 所示）中建立菜单。

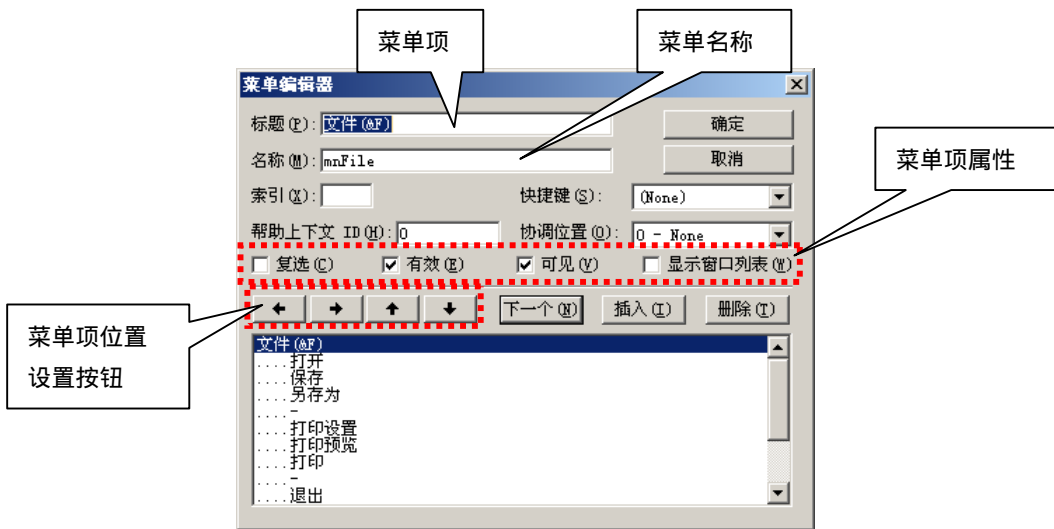


图 2-16 菜单编辑器

标题：每一个菜单项所要显示的文本，即菜单上的“菜名”。

名称：菜单项控件的名称。

复选、有效、可见、显示窗口列表：菜单项的具体属性，由它们决定了菜单的形状、



样式等。

菜单项的具体属性见表 2-1。

表 2-1 菜单项属性表

标题	名称	复选	有效性	可见性
文件(&F)	mnFile	否	有效	可见
打开	mnOpen	否	有效	可见
保存	mnSave	否	有效	可见
续表				
标题	名称	复选	有效性	可见性
另存为	mnSaveAs	否	有效	可见
-	mnBar0	否	有效	可见
打印设置	mnSetup	否	有效	可见
打印预览	mnView	否	有效	可见
打印	mnPrint	否	有效	可见
-	mnBar1	否	有效	可见
退出	mnExit	否	有效	可见

(4) 按 F5 键运行程序，单击“文件”菜单后，将得到如图 2-17 所示的界面。

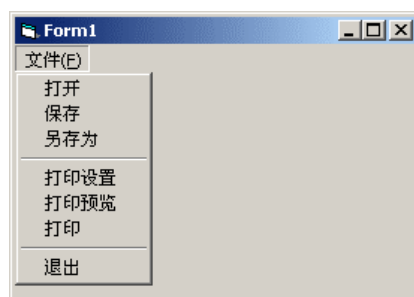


图 2-17 运行效果图



试一试：

在上面的工程中加入“编辑(E)”菜单，该菜单包含两个菜单项，如“拷贝”和“粘贴”。

2.2.2 弹出式菜单

单击鼠标右键，打开快捷菜单。这是 Windows 95 出现以后才广为应用的一种功能，但它确实给我们带来了莫大的方便，好东西自然要把它做出来。

(1) 新建“标准 EXE”工程。

(2) 在窗体上单击鼠标右键弹出快捷菜单，选取“菜单编辑器”菜单项，如图 2-18 所示。

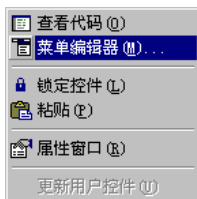


图 2-18 快捷菜单

(3) 在菜单编辑器中建立如图 2-19 所示的菜单，菜单项的具体属性见表 2-2。

表 2-2 菜单项属性表

标题	名称	复选	有效性	可见性
_PopupMenu	mn_PopMenu	否	有效	不可见
菜单项 1	mnMenuItem1	否	有效	可见
菜单项 2	mnMenuItem2	否	有效	可见



图 2-19 菜单编辑器

注意：

因为是弹出菜单，所以菜单 mn_PopMenu 的可见属性应该是 Disable 的，即该菜单在通常情况下应该不可见，只有当我们需要时才会弹出。那么怎样让菜单弹出呢？

(4) 为 Form1 窗体建立 MouseDown 事件，在代码窗口的控件选择下拉列表框中选择“Form”，在“事件下拉列表框”中选择“MouseDown”事件，并输入下列代码：

```
Private Sub Form_MouseDown(Button As Integer, Shift As_  
Integer, X As Single, Y As Single)  
    If Button = vbRightButton Then  
        PopupMenu mn_PopMenu  
    End If  
End Sub
```

其中，Button = vbRightButton 用于判断按下的是否为鼠标右键，而 PopupMenu 方法则用于在窗体上弹出菜单。其格式如下：

PopupMenu menuname
menuname 所要弹出的菜单的名称



(5) 按 F5 键运行程序，并在窗体上单击鼠标右键，效果如图 2-20 所示。



图 2-20 运行效果图

2.2.3 好看的图标菜单

Windows 的“开始”菜单给我们的印象非常深刻，它用不同的图标和文本表示了不同的菜单项，让人一目了然。现在也该包装一下我们的菜单了。

(1) 打开工程 MyApp.vbp。

(2) 在窗体中加入 Image 控件 picOpen，设置它的 Picture 属性，如图 2-21 所示，单击属性值右边的“...”按钮即可导入图片，并设置它的 Visible 属性为 False，即不在窗体中显示这张图片。

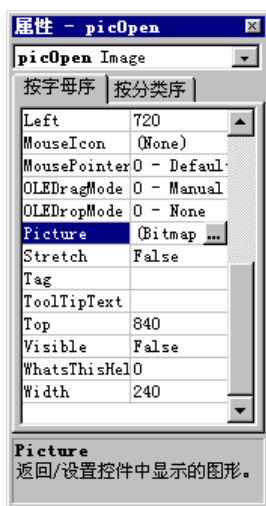


图 2-21 图片属性对话框

(3) 以同样方法再在窗体中加入 3 个 Image 控件：picSave、picCopy 和 picPrint，如图 2-22 所示。



图 2-22 程序界面

(4) 为了使加载的图片出现在菜单上,我们要声明几个函数。在代码窗口的控件选择下拉列表框中选择“(通用)”,在“事件下拉列表框”中选择“(声明)”事件,在代码窗口中输入下列代码:

```
Option Explicit
Private Declare Function GetMenu Lib "user32" (ByVal hwnd_
As Long) As Long
Private Declare Function GetSubMenu Lib "user32" (ByVal_
hMenu As Long, ByVal nPos As Long) As Long
Private Declare Function SetMenuItemBitmaps Lib "user32"_
(ByVal hMenu As Long, ByVal nPosition As Long, ByVal wFlags_
As Long, ByVal hBitmapUnchecked As Long, ByVal hBitmapChecked_
As Long) As Long
Const MF_BYPOSITION = &H400&
```

在这段代码中我们定义了3个函数: GetMenu、GetSubMenu 和 SetMenuItemBitmaps, 以及一个常量 MF_BYPOSITION。这3个函数由动态链接库 USER32.DLL 提供。

其中函数 GetMenu 用于得到与窗体相关联的菜单句柄,其格式如下所示:

```
hMenu=GetMenu(hWnd)
hWnd      要得到菜单句柄的窗口的句柄
hMenu     返回值,获得的菜单句柄
```

函数 GetSubMenu 的作用是得到某菜单项所对应的子菜单的句柄,其格式如下所示:

```
rehMenu=GetSubMenu(hMenu, nPos)
hMenu     主菜单的句柄
nPos      菜单项的索引值
rehMenu   返回值,获得的子菜单的句柄
```

函数 SetMenuItemBitmaps 的作用是使图片与菜单项相关联,其格式如下所示:

```
SetMenuItemBitmaps(hMenu,nPosition,wFlags,hBitmapUnchecked,hBitmapChecked)
hMenu      菜单句柄
nPosition  菜单项的索引值
wFlags     标志值,可以是 MF_BYCOMMAND 或 MF_BYPOSITION
hBitmapUnchecked 菜单项没有被选中时的图片
hBitmapChecked  菜单项被选中时的图片
```



(5) 双击窗体打开代码编辑窗口，并在其中添加 Load 事件的代码：

```
Private Sub Form_Load()  
    Dim m As Long, l As Long  
    Dim sHandle As Long, sHandle2 As Long, sHandle1 As Long  
    m = GetMenu(hwnd)  
    sHandle1 = GetSubMenu(m, 0)  
    l = SetMenuItemBitmaps(sHandle1, 0, MF_BYPOSITION, _  
        picOpen.Picture, uncheck.Picture)  
    l = SetMenuItemBitmaps(sHandle1, 1, MF_BYPOSITION, _  
        picSave.Picture, picSave.Picture)  
    l = SetMenuItemBitmaps(sHandle1, 2, MF_BYPOSITION, _  
        picPrint.Picture, picPrint.Picture)  
    sHandle = GetSubMenu(m, 1)  
    sHandle2 = GetSubMenu(sHandle, 0)  
    l = SetMenuItemBitmaps(sHandle, 0, MF_BYPOSITION, _  
        picCopy.Picture, picCopy.Picture)  
End Sub
```

(6) 按 F5 键运行程序，并单击“文件”菜单，效果如图 2-23 所示。

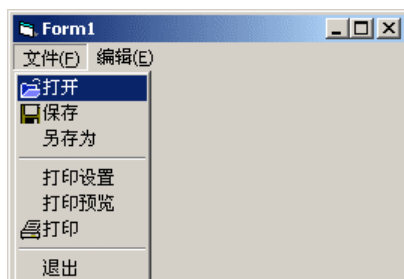


图 2-23 运行效果图



试一试：

为弹出菜单加上图标。

2.2.4 伸缩自如的动态菜单

在 Office 2000 中，不常用的菜单会自动收缩起来，这给用户提供了很大的方便，我们应该学习一下这个优点。那么我们能不能实现这一功能，菜单设计好之后是不是一成不变的呢？

(1) 打开工程 MyApp.vbp。

(2) 打开菜单编辑器，并修改“文件”菜单。在“文件”菜单中增加菜单项“显示全部菜单”和菜单分隔条，并设置菜单项 mnBar0、mnSetup、mnView 和 mnPrint 的 Visible 属性值为 False，即让这几个菜单项不可见，如图 2-24 所示。

(3) 在窗体中加入 Image 控件 picCheck，设置它的 Visible 属性为 False，如图 2-25



所示。

(4) 如下所示修改 Load 事件的代码：

```
Private Sub Form_Load()  
    Dim m As Long, l As Long  
    Dim sHandle As Long, sHandle2 As Long, sHandle1 As Long  
    m = GetMenu(hwnd)  
    sHandle1 = GetSubMenu(m, 0)  
    l = SetMenuItemBitmaps(sHandle1, 0, MF_BYPOSITION, _  
        picOpen.Picture, picOpen.Picture)  
    l = SetMenuItemBitmaps(sHandle1, 1, MF_BYPOSITION, _  
        picSave.Picture, picSave.Picture)  
    'l = SetMenuItemBitmaps(sHandle1, 6, MF_BYPOSITION, _  
        picPrint.Picture, picPrint.Picture)  
    sHandle = GetSubMenu(m, 1)  
    sHandle2 = GetSubMenu(sHandle, 0)  
    l = SetMenuItemBitmaps(sHandle, 0, MF_BYPOSITION, _  
        picCopy.Picture, picCopy.Picture)  
End Sub
```

因为当菜单项不可见时就相当于该菜单项不存在，所以语句

`l = SetMenuItemBitmaps(sHandle1, 6, MF_BYPOSITION, _
 picPrint.Picture, picPrint.Picture)`被注释（即在语句前加上注释号'）。



图 2-24 菜单编辑器

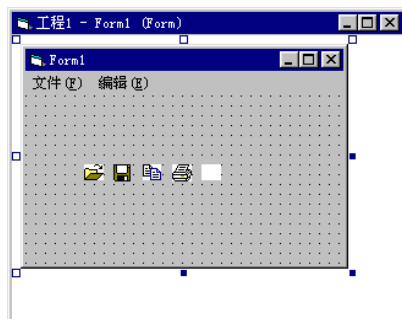




图 2-25 程序界面



试一试：

去掉注释符号“'”运行该程序，观察程序运行结果，指出错误。

(5) 在窗体中单击“文件”菜单，选择菜单项“显示全部菜单”，这时将自动打开代码窗口且光标定位于 mnSs_Click 事件，在其中输入下列代码：

```
Private Sub mnSs_Click()  
    Dim m As Long, l As Long, sHandle1 As Long  
    mnBar0.Visible = Not mnBar0.Visible  
    mnSetup.Visible = Not mnSetup.Visible  
    mnView.Visible = Not mnView.Visible  
    mnPrint.Visible = Not mnPrint.Visible  
    mnSs.Checked = Not mnSs.Checked  
    m = GetMenu(hwnd)  
    sHandle1 = GetSubMenu(m, 0)  
    If mnSs.Checked Then  
        l = SetMenuItemBitmaps(sHandle1, 6, MF_BYPOSITION, _  
picPrint.Picture, picPrint.Picture)  
    Else  
        l = SetMenuItemBitmaps(sHandle1, 6, MF_BYPOSITION, _  
picCheck.Picture, picCheck.Picture)  
    End If  
End Sub
```

上面代码的作用是当我们单击菜单项“显示全部菜单”时，设置菜单项 mnBar0、mnSetup、mnView 和 mnPrint 的 Visible 属性值为 True，即让这几个菜单项可见，并根据 mnSs 的 Visible 属性值来设置“打印”菜单项的图片，效果如图 2-26 所示。

(6) 按 F5 键运行程序，并单击“文件”菜单，选择菜单项“显示全部菜单”，如图 2-27 所示。

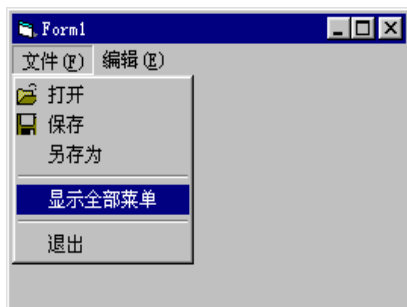


图 2-26 运行效果图



图 2-27 运行效果图



2.3 对话框

2.3.1 应用程序的信使

菜单是做完了，可怎么让它产生动作呢？最起码也得告诉我们选了哪一道“菜”。

(1) 打开工程 MyApp.vbp。

(2) 在窗体中单击“编辑”菜单，选择菜单项“拷贝”，这时代码窗口将自动打开，且光标定位于 `mnCopy_Click` 事件，在其中输入下列代码：

```
Private Sub mnCopy_Click()  
    MsgBox "你单击了" + mnCopy.Caption + "菜单项", vbOKOnly  
End Sub
```

其中 MsgBox 方法的作用是在对话框中显示消息，等待用户单击按钮，其格式如下所示：

MsgBox prompt[, buttons] [, title]

prompt 所要显示的信息

buttons 可选，在对话框中所要显示的按钮和图标。如果省略，则 buttons 的缺省值为 0。这些图标和按钮值是 Visual Basic 系统定义的一些常数，具体见表 2-3。

表 2-3 系统定义的常数表

常数	值	描述
VbOKOnly	0	只显示 OK 按钮
VbOKCancel	1	显示 OK 及 Cancel 按钮
VbAbortRetryIgnore	2	显示 Abort、Retry 及 Ignore 按钮
VbYesNoCancel	3	显示 Yes、No 及 Cancel 按钮
VbYesNo	4	显示 Yes 及 No 按钮
VbRetryCancel	5	显示 Retry 及 Cancel 按钮
VbCritical	16	显示 Critical Message 图标
VbQuestion	32	显示 Warning Query 图标
VbExclamation	48	显示 Warning Message 图标
VbInformation	64	显示 Information Message 图标

title 可选，对话框的标题。缺省则以程序的名字作为对话框的标题。

(3) 按 F5 键运行程序，单击“编辑”菜单并选择“拷贝”菜单项，如图 2-28 所示。





图 2-28 运行效果图



试一试：
为其他几个菜单项也加上产生提示信息的作用。

2.3.2 应用程序的眼睛和耳朵

交流应该是双向的，既然能发出信息，当然也得能接收信息。

(1) 打开工程 MyApp.vbp。

(2) 在窗体中单击“文件”菜单，选择菜单项“打开”，这时代码窗口将自动打开，且光标定位于 mnOpen_Click 事件，在其中输入下列代码：

```
Private Sub mnOpen_Click()  
    Dim FileName As String  
    FileName = InputBox("请输入一个文件名", "文件名输入")  
End Sub
```

其中 InputBox 函数的作用是：在一对话框中显示提示，等待用户输入正文或按下按钮，并返回包含文本框的内容。其格式如下：

InputBox(prompt[, title])
prompt 所要显示的提示信息等
title 可选，对话框的标题，缺省则把程序名放入标题栏中

(3) 按 F5 键运行程序，单击“文件”菜单并选取“打开”菜单项，如图 2-29 所示。

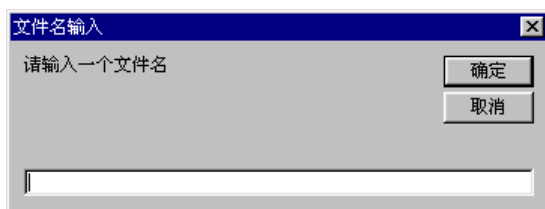


图 2-29 运行效果图

2.3.3 名片——制作自己的“关于对话框”

要推销自己，就得大胆地亮出你自己，让大家了解你、记住你，做张名片是应该的。

(1) 打开工程 MyApp.vbp。

(2) 打开菜单编辑器，并添加菜单项“帮助”和子菜单项“关于...”，如图 2-30 所示。

(3) 在代码窗口的控件选择下拉列表框中选择“(通用)”，在“事件下拉列表框”中选择“(声明)”事件，在代码窗口中追加下列代码：

```
Private Declare Function ShellAbout Lib "shell32.dll"  
    Alias "ShellAboutA" (ByVal hwnd As Long, ByVal szApp As String, _  
    ByVal szOtherStuff As String, ByVal hIcon As Long) As Long
```



图 2-30 菜单编辑器

在这段代码中我们定义了一个函数 ShellAbout, 这个函数由动态链接库 SHELL32.DLL 提供。

该函数的作用是显示 Shell 程序的关于对话框, 其格式如下所示:

ShellAbout (hWnd, szApp, szOtherStuff, hIcon)

hWnd 窗口句柄

szApp 关于窗口的标题文本

szOtherStuff 关于窗口的窗口内文本, 一般用于说明应用程序

hIcon 关于窗口的图标


(4) 在窗体中单击“帮助”菜单, 选择菜单项“关于...”, 这时代码窗口将自动打开, 且光标定位于 mnAbout_Click 事件, 在其中输入下列代码:

```
Private Sub mnAbout_Click()  
    Call ShellAbout(hwnd, "我的 VB 程序", "这是一个用 Visual  
Basic 编写的应用程序" & vbCrLf & "作者:XXX", Me.Icon)  
End Sub
```

(5) 按 F5 键运行程序, 单击“帮助”菜单并选取“关于...”菜单项, 效果如图 2-31 所示。

上面是公用模板制作出来的名片, 风格千篇一律, 该来制作一张个性化的名片。

(1) 打开工程 MyApp.vbp。

(2) 单击工具栏上的“”按钮, 在“添加窗体”对话框中选取“关于”对话框, 如图 2-32 所示。

(3) 在窗体中单击“帮助”菜单, 选择菜单项“关于...”, 这时代码窗口将自动打开, 且光标定位于 mnAbout_Click 事件, 如下修改代码:

```
Private Sub mnAbout_Click()  
    frmAbout.Show  
End Sub
```

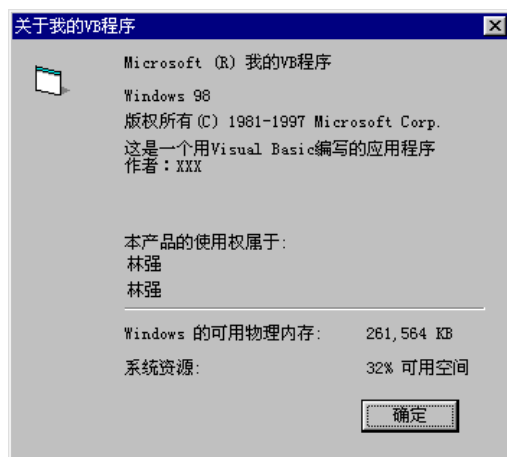


图 2-31 运行效果图

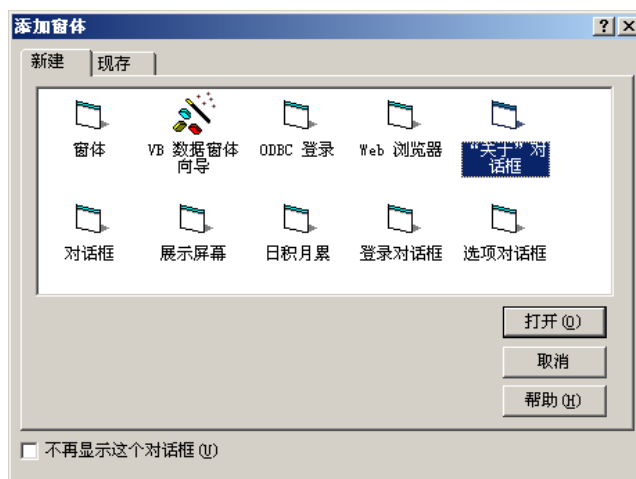


图 2-32 添加窗体对话框

(4) 按 F5 键运行程序，单击“帮助”菜单并选取“关于...”菜单项，如图 2-33 所示。





图 2-33 运行效果图

2.3.4 拿来主义

在很多的应用程序中，它们打开文件的操作很类似，简直是双胞胎，其实这都是继承自 Windows 操作系统的公共对话框。那么现在就让我们来完成“打开”、“另存为”菜单的功能。

- (1) 打开工程 MyApp.vbp。
- (2) 在工具箱上单击鼠标右键弹出快捷菜单, 选取“ 部件 ”菜单项, 打开部件对话框。
- (3) 在部件对话框中找到“ Microsoft Common Dialog Control 6.0 (SP3) ”项, 并把该控件加入到工具箱中。
- (4) 在 Form1 窗体中加入 CommonDialog 控件。

在窗体中选中 CommandDialog1 控件，然后单击鼠标右键弹出快捷菜单，并选取“属性”菜单项，如图 2-34 所示。



图 2-34 快捷菜单

- (5) 在弹出的属性页对话框的过滤器项中输入“文档 *.Doc|*.Doc|文本文件 *.Txt|*.Txt”，完成后按“确定”按钮返回，如图 2-35 所示。



图 2-35 属性页对话框

- (6) 在窗体中单击“文件”菜单，选择菜单项“打开”，这时代码窗口将自动打开，且光标定位于 `mnOpen_Click` 事件，如下修改代码：



```
Private Sub mnOpen_Click()  
    CommonDialog1.ShowOpen  
End Sub
```

(7) 按 F5 键运行程序,单击“文件”菜单,并选取“打开”菜单项,如图 2-36 所示。



图 2-36 运行效果图

(8) 在窗体中单击“文件”菜单,选择菜单项“另存为”,这时代码窗口将自动打开,且光标定位于 mnSaveAs_Click 事件,输入下列代码:

```
Private Sub mnSaveAs_Click()  
    CommonDialog1.ShowSave  
End Sub
```

(9) 按 F5 键运行程序,单击“文件”菜单,并选取“另存为”菜单项,打开如图 2-37 所示的对话框。

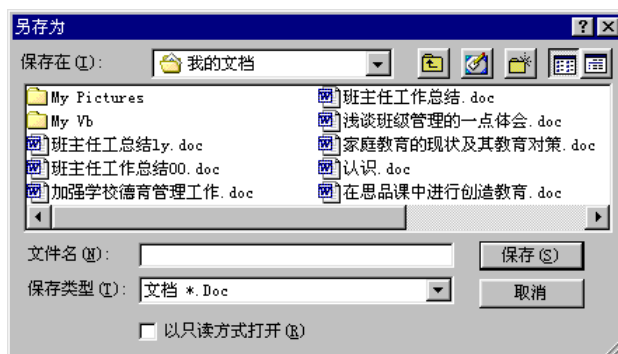


图 2-37 “另存为”对话框



试一试：

其实 CommonDialog 控件除了可显示“打开”、“另存为”和“调色板”对话框外,还可以显示“字体”、“打印”等对话框。请完成“打印”菜单(调用 ShowPrinter 方法)。



2.4 小结

在这一章中，我们通过大量的实例演示了窗体、菜单的制作方法和美化的技巧，以及一些公共对话框的运用。

给出了窗体背景的处理及不规则窗体和特殊窗体的构造方法。

分别给出了普通菜单、弹出菜单及图标菜单的制作方法，并以伸缩菜单为例说明了菜单项属性的处理方法。

讲述了 Visual Basic 的输入、输出语句及公共对话框的应用。



第 3 章 图形处理

在应用程序中对图形进行处理，可以使程序更加生动，更加有吸引力。Visual Basic 具备了极其丰富的图形、图像处理功能，通过它自身携带的控件和 Windows 所提供的丰富的函数，几乎能完成所有的图形操作。



本章的主要内容如下：

- (1) 在 Visual Basic 中的色彩处理；
- (2) 制作简单的动画，包括平移、滚动、镜像、旋转和反射等；
- (3) 图像的放大与缩小。

3.1 颜色处理

3.1.1 一条特殊的彩虹

天上的彩虹是非常美丽的，但遗憾的是不常见，现在我们就在计算机中让美丽重现。

(1) 新建“标准 EXE”工程。

(2) 在代码窗口的控件选择下拉列表框中选择“Form”，在“事件下拉列表框”中选择“Paint”事件。输入下列代码：

```
Private Sub Form_Paint()  
    '画彩色条纹  
    Dim i As Integer  
    n = 0  
    For i = 0 To 255  
        '灰色带  
        Line (300 + n, 500)-(300 + n, 1500), RGB(i, i, i)  
        n = n + 2  
        DoEvents  
    Next i
```




```

For i = 0 To 255
    '红色带
        Line (300 + n, 500)-(300 + n, 1500), RGB(i, 0, 0)
        n = n + 2
        DoEvents
    Next i
For i = 0 To 255
    '绿色带
        Line (300 + n, 500)-(300 + n, 1500), RGB(0, i, 0)
        n = n + 2
        DoEvents
    Next i
For i = 0 To 255
    '蓝色带
        Line (300 + n, 500)-(300 + n, 1500), RGB(0, 0, i)
        n = n + 2
        DoEvents
    Next i
For i = 0 To 255
        Line (300 + n, 500)-(300 + n, 1500), RGB(0, i, i)
        n = n + 2
        DoEvents
    Next i
For i = 0 To 255
        Line (300 + n, 500)-(300 + n, 1500), RGB(i, 0, i)
        n = n + 2
        DoEvents
    Next i
For i = 0 To 255
        Line (300 + n, 500)-(300 + n, 1500), RGB(i, i, 0)
        n = n + 2
        DoEvents
    Next i
For i = 0 To 255
        Line (300 + n, 500)-(300 + n, 1500), RGB(192, 192, _
i)
        n = n + 2
        DoEvents
    Next i
For i = 0 To 255
        Line (300 + n, 500)-(300 + n, 1500), RGB(i, 192, 192)
        n = n + 2
        DoEvents
    Next i
For i = 0 To 255
        Line (300 + n, 500)-(300 + n, 1500), RGB(192, i, 192)
        n = n + 2
        DoEvents
    Next i

```



End Sub

其中 Line 函数我们已经在第二章中学过，它是用来画矩形或线条的。而 DoEvents 语句的作用则是将应用程序对资源的控制切换到操作系统（即 Windows），只要是此环境中的应用程序，都有机会响应待处理事件，然后应用程序又恢复控制。这样就不会因为应用程序使用过多次数的循环，而使应用程序本身或其他应用程序所产生的事件在循环结束前无法得到处理。

而之所以在 Paint 事件中加入代码，是因为当应用程序在重画窗口时就会激发该事件。

（3）按 F5 键运行，如图 3-1 所示。

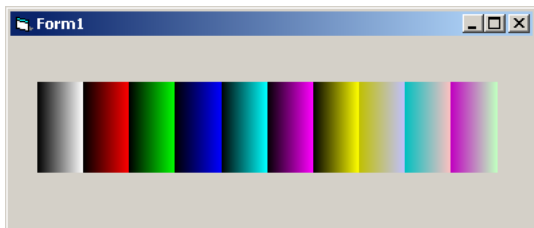


图 3-1 运行效果图



试一试：

把这段代码放入 Load 事件，并取消 Paint 事件。运行该程序并观察它与原来的程序有什么区别。

3.1.2 调色板

在图形处理软件中，我们经常需要调节图形的颜色，这时我们就需要由调色板来帮忙，正所谓“工欲善其事，必先利其器”。

- （1）新建“标准 EXE”工程。
- （2）建立如图 3-2 所示的窗体。

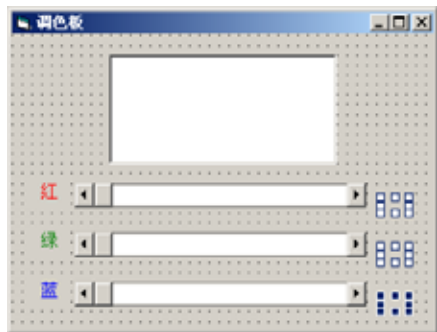


图 3-2 程序界面

该窗体所包含的控件及其属性具体见表 3-1。

表 3-1

控件及其属性值表



控件	名称	部分属性值	
TextBox	Text1	Text	(空)
Label	Label1	Caption	红
续表			
控件	名称	部分属性值	
	Label2	Caption	绿
	Label3	Caption	蓝
	Label4	Caption	(空)
	Label5	Caption	(空)
	Label6	Caption	(空)
HScrollBar	HScroll1	Max	255
		Min	0
		LargeChange	20
	HScroll2	与 HScroll1 相同	
	HScroll3	与 HScroll1 相同	

(3) 在代码窗口的控件选择下拉列表框中选择“(通用)”，在“事件下拉列表框”中选择“(声明)”事件，在代码窗口中输入下列代码：

```
Sub txtBackColor()  
    Dim r As Integer, g As Integer, b As Integer  
    r = HScroll1.Value  
    g = HScroll2.Value  
    b = HScroll3.Value  
    Text1.BackColor = RGB(r, g, b)  
End Sub
```

这是一个自定义方法，它的作用是把 HScroll1、HScroll2 和 HScroll3 的 Value 属性作为 RGB 函数的参数来生成 Text1 的背景色，这样就可以通过滚动条的滚动来改变颜色，从而模拟出调色板。

(4) 为 Form 的 Load 事件输入如下代码：

```
Private Sub Form_Load()  
    txtBackColor  
    Label4.Caption = Str(HScroll1.Value)  
    Label5.Caption = Str(HScroll2.Value)  
    Label6.Caption = Str(HScroll2.Value)  
End Sub
```

(5) 分别为 HScroll1、HScroll2 及 HScroll3 三个控件建立如下的 Change 事件代码：

```
Private Sub HScroll1_Change()  
    txtBackColor  
    Label4.Caption = Str(HScroll1.Value)  
End Sub  
  
Private Sub HScroll2_Change()  
    txtBackColor  
    Label5.Caption = Str(HScroll2.Value)  
End Sub
```



```
Private Sub HScroll13_Change()  
    txtBackColor  
    Label6.Caption = Str(HScroll13.Value)  
End Sub
```

(6) 按 F5 键运行, 效果如图 3-3 所示。

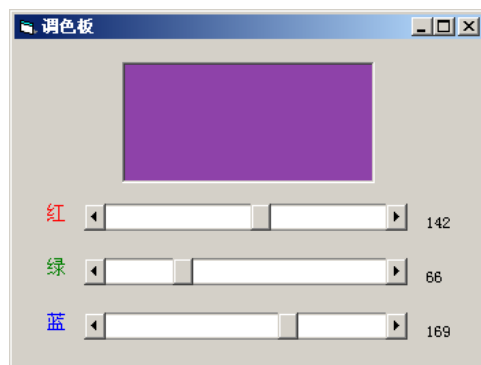


图 3-3 运行效果图

3.2 做一幅动画

3.2.1 走马灯

电视上的滚动字幕可见得多了, 但怎样来实现它呢?

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 3-4 所示的窗体。

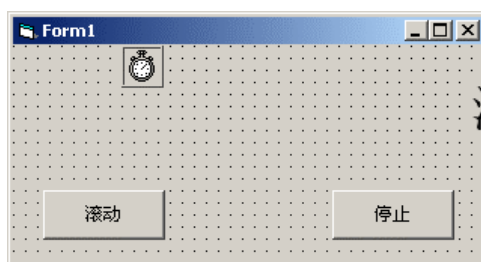


图 3-4 程序界面

该窗体所包含的控件及其属性具体见表 3-2。

表 3-2 控件及其属性值表

控件	名称	部分属性值	
		Caption	属性值
CommandButton	Command1	Caption	滚动
	Command2	Caption	停止



Timer	Timer1	Enabled	False
		Interval	10
Label	Label1	Caption	演示程序

(3) 在代码窗口的“控件选择下拉列表框”中选择“(通用)”，在“事件下拉列表框”中选择“(声明)”事件，在代码窗口中输入下列代码：

```
Dim oLeft
```

以上的代码在窗体中定义了一个模块级变量，用于存放 Label1 的 Left 属性值的初始值。它可以被窗体内的所有过程存取。

(4) 为 Form 的 Load 事件输入如下代码：

```
Private Sub Form_Load()  
    oLeft = Label1.Left  
End Sub
```

(5) 分别为 Command1、Command2 控件建立如下的 Click 事件代码：

```
Private Sub Command1_Click()  
    Timer1.Enabled = True  
End Sub
```

```
Private Sub Command2_Click()  
    Timer1.Enabled = False  
End Sub
```

(6) 为 Timer1 控件建立如下的 Timer 事件代码：

```
Private Sub Timer1_Timer()  
    Label1.Left = Label1.Left - 20  
    If (Label1.Left + Label1.Width < 0) Then  
        Label1.Left = oLeft  
    End If  
End Sub
```

该段代码的作用是当 Label1 移动时，其位置超出窗体时使其回到原来位置，从而实现文字的滚动。

(7) 按 F5 键运行，如图 3-5 所示。

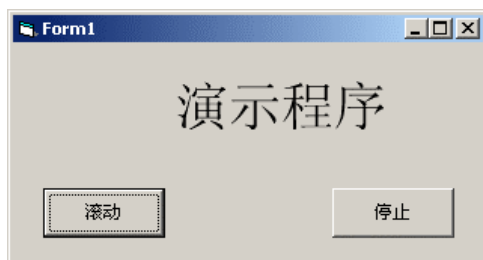


图 3-5 运行效果图



试一试：

(1) 使文本自左向右滚动；



- (2) 使文本自上而下滚动；
- (3) 使文本自下而上滚动。

3.2.2 百叶窗

电视镜头切换时的特效非常多，有淡入淡出、雾化、飞入、卷曲等，我们来做一个简单的效果——百叶窗。

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 3-6 所示的窗体，其中 Form1 的 BackColor 属性值为 &H00000000。

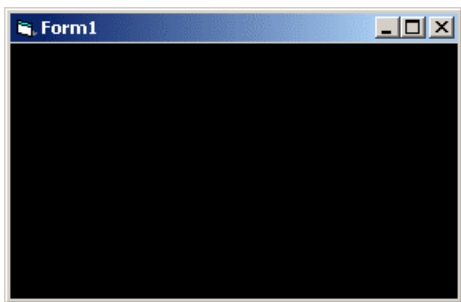


图 3-6 程序界面

- (3) 在代码窗口的“控件选择下拉列表框”中选择“(通用)”，在“事件下拉列表框”中选择“(声明)”事件，在代码窗口中输入下列代码：

```
Private Declare Function SetPixel Lib "gdi32" (ByVal hdc_
As Long, ByVal x As Long, ByVal y As Long, ByVal crColor As_
Long) As Long
```

在这里我们定义了函数 SetPixel，该函数由动态链接库 GDI32.DLL 提供，其作用是设置指定像素的颜色，其格式如下：

reColor=SetPixel(hdc, X, Y, crColor)	
hdc	窗口句柄
X	像素点的横坐标
Y	像素点的纵坐标
crColor	像素点的颜色
reColor	返回值

- (4) 为 Form 的 Load 事件输入如下代码：

```
Private Sub Form_Load()
    If (Not (Me.ScaleHeight Mod 10) = 0) Then
        Me.ScaleHeight = Int(Me.ScaleHeight / 10) * 10
    End If
End Sub
```

以上代码使窗体能刚好被 10 等分。

- (5) 为 Form1 控件建立如下的 Click 事件代码：



```
Private Sub Form_Click()  
    Dim tmp_y As Long  
    For y = 0 To Me.ScaleHeight / 10 / Screen.TwipsPerPixelY  
        For i = 0 To 10  
            tmp_y = i * Me.ScaleHeight / 10 / _  
Screen.TwipsPerPixelY + y  
            For x = 0 To Me.ScaleWidth  
                b = SetPixel(Me.hdc, x, tmp_y, RGB(255, 255, 255))  
            Next x  
            DoEvents  
        Next i  
        DoEvents  
    Next y  
End Sub
```

(6) 按 F5 键运行，效果如图 3-7 所示。

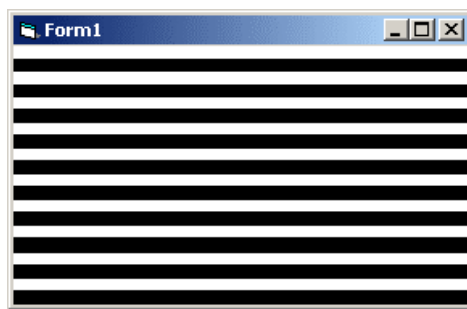


图 3-7 运行效果图



试一试：

以白色作为底色，黑色作为滚动色来构造百叶窗，即从白色向黑色过渡。

3.2.3 开动的汽车

汽车太贵，驾照又太难考，还是在电脑中过把瘾吧！

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 3-8 所示的窗体。




图 3-8 程序界面

该窗体所包含的控件及其属性，具体见表 3-3。

表 3-3 控件及其属性值表

控件	名称	部分属性值	
Image	Image1	见操作(3)	
Label	Label1	Caption	慢
	Label2	Caption	快
Timer	Timer1	Interval	10
CommandButton	Command1	Caption	再来一次
HScroll	HScroll1	Max	50

(3) 设置控件 Image1 的 Picture 属性，单击属性值右边的 “” 按钮导入图片。

(4) 为控件 Command1 建立 Click 事件，具体代码如下所示：

```
Private Sub Command1_Click()  
    Image1.Left = 0  
    Command1.Enabled = False  
End Sub
```

(5) 为控件 Timer1 建立 Timer 事件，其代码如下所示：

```
Private Sub Timer1_Timer()  
    If Image1.Left <= Form1.ScaleWidth Then  
        Image1.Left = Image1.Left + HScroll1.Value + 1  
    Else  
        Command1.Enabled = True  
    End If  
End Sub
```

(6) 按 F5 键运行，其效果如图 3-9 所示。





图 3-9 运行效果图



试一试：

在窗体上加入“倒车”按钮，并实现倒车功能。

3.2.4 旋转的图形

图像处理软件中有一些必不可少的功能，那就是图像的镜像、翻转和旋转。现在也让我们的程序来“折腾”一下“可怜”的图片。

(1) 新建“标准 EXE”工程。

(2) 在工具箱上单击鼠标右键弹出快捷菜单，选取“部件”菜单项，打开部件对话框。在部件对话框中找到“Microsoft Windows Common Controls 6.0 (SP4)”项，并把该控件集加入到工具箱中。

(3) 建立如图 3-10 所示的窗体。



图 3-10 程序界面

该窗体所包含的控件及其属性，具体见表 3-4。

表 3-4 控件及其属性值表

控件	名称	部分属性值	
PictureBox	Picture1	Picture	Bitmap
	Picture2	AutoRedraw	False
		Picture	(空)
Slider	Slider1	LargeChange	10
		Max	360
		Min	0
		SmallChange	1
CommandButton	Command1	Caption	镜像
	Command2	Caption	垂直翻转
	Command3	Caption	旋转



(4) 在代码窗口中加入函数声明等,代码如下所示:

```
Const SRCCOPY = &HCC0020
Const Pi = 3.14159265359
Private Declare Function SetPixel Lib "gdi32" (ByVal hdc_
As Integer, ByVal x As Integer, ByVal y As Integer, ByVal crColor_
As Long) As Long
Private Declare Function GetPixel Lib "gdi32" (ByVal hdc_
As Integer, ByVal nXPos As Integer, ByVal nYPos As Integer)_
As Long
Private Declare Function StretchBlt Lib "gdi32" (ByVal_
hdcDest As Long, ByVal nXOriginDest As Long, ByVal nYOriginDest_
As Long, ByVal nWidthDest As Long, ByVal nHeightDest As Long,_
ByVal hdcSrc As Long, ByVal nXOriginSrc As Long, ByVal_
nYOriginSrc As Long, ByVal nWidthSrc As Long, ByVal nHeightSrc_
As Long, ByVal dwRop As Long) As Long

Subpicrotate(picSrc As PictureBox, picDec As PictureBox, _
ByVal theta As Single)
Dim clx As Integer, cly As Integer
Dim c2x As Integer, c2y As Integer
Dim a As Single
Dim plx As Integer, ply As Integer
Dim p2x As Integer, p2y As Integer
Dim n As Integer, r As Integer
Dim picSrchDC As Long, picDechDC As Long
Dim c0 As Long, c1 As Long, c2 As Long, c3 As Long
Dim returnval As Long
clx = picSrc.ScaleWidth \ 2
cly = picSrc.ScaleHeight \ 2
c2x = picDec.ScaleWidth \ 2
c2y = picDec.ScaleHeight \ 2
If c2x < c2y Then
    n = c2y
Else
    n = c2x
End If
n = n - 1
picSrchDC = picSrc.hdc
picDechDC = picDec.hdc
For p2x = 0 To n
    For p2y = 0 To n
        If p2x = 0 Then
            a = Pi / 2
        Else
            a = Atn(p2y / p2x)
        End If
        r = Sqr(1& * p2x * p2x + 1& * p2y * p2y)
        plx = r * Cos(a + theta)
        ply = r * Sin(a + theta)
```



```

        c0 = GetPixel(picSrchDC, clx + plx, cly + ply)
        c1 = GetPixel(picSrchDC, clx - plx, cly - ply)
        c2 = GetPixel(picSrchDC, clx + ply, cly - plx)
        c3 = GetPixel(picSrchDC, clx - ply, cly + plx)
        If c0 <> -1 Then
            returnval = SetPixel(picDechDC, c2x + p2x, c2y + _
p2y, c0)
        End If
        If c1 <> -1 Then
            returnval = SetPixel(picDechDC, c2x - p2x, c2y - _
p2y, c1)
        End If
        If c2 <> -1 Then
            returnval = SetPixel(picDechDC, c2x + p2y, c2y - _
p2x, c2)
        End If
        If c3 <> -1 Then
            returnval = SetPixel(picDechDC, c2x - p2y, c2y + _
p2x, c3)
        End If
    Next
    DoEvents
Next
End Sub

```

在这段代码中我们声明了两个常数、三个函数和一个自定义函数，其中两个常数 SRCCOPY 及 Pi 分别代表绘图的操作方式和 ，而函数 SetPixel、GetPixel 和 StretchBlt 则都是由动态链接库 GDI32.DLL 提供的，其中函数 SetPixel 的作用及用法上一节已经提过。

函数 GetPixel 的作用是指定像素点的颜色，其格式如下：

```

reColor=GetPixel(hdc, nXPos, nYPos)

```

hdc	窗口句柄
nXPos	像素点的横坐标
nYPos	像素点的纵坐标
reColor	返回值，返回像素点的颜色

函数 StretchBlt 的作用是把一幅位图图像由源区域复制至目的区域，其格式如下：

```

StretchBlt(hdcDest, nXOriginDest, nYOriginDest, nWidthDest, nHeightDest,
hdcSrc, nXOriginSrc, nYOriginSrc, nWidthSrc, nHeightSrc, dwRop)

```

hdcDest	目的设备句柄
(nXOriginDest,nYOriginDest)	目的区域的左上角坐标
nWidthDest,nHeightDest	目的区域的宽和高
hdcSrc	源设备句柄
(nXOriginSrc,nYOriginSrc)	源区域的左上角坐标
nWidthSrc,nHeightSrc	源区域的宽和高



dwRop

复制时的绘图方式

自定义函数 picrotate 是用来对图片进行旋转操作的，其格式如下：

picrotate(picSrc, picDec, theta)

picSrc 源图片

picDec 目的图片

theta 旋转角度

(5) 为 Form 的 Load 事件输入如下代码：

```
Private Sub Form_Load()  
    Picture1.ScaleMode = 3  
    Picture2.ScaleMode = 3  
    Picture2.Width = Picture1.Width  
    Picture2.Height = Picture1.Height  
    Picture2.AutoRedraw = False  
End Sub
```

该代码段保证了两个 PictureBox 有相同的大小，并且都以像素为单位来操作。同时保证了 Picture2 的 AutoRedraw 的属性值为 False。

(6) 分别为 Command1、Command2 及 Command3 三个控件建立如下的 Click 事件代码：

```
Private Sub Command1_Click()  
    Picture2.Cls  
    px% = Picture1.ScaleWidth  
    py% = Picture1.ScaleHeight  
    returnval = StretchBlt(Picture2.hdc, px%, 0, -px%, py%, _  
Picture1.hdc, 0, 0, px%, py%, SRCCOPY)  
End Sub  
  
Private Sub Command2_Click()  
    Picture2.Cls  
    px% = Picture1.ScaleWidth  
    py% = Picture1.ScaleHeight  
    returnval = StretchBlt(Picture2.hdc, 0, py%, px%, -py%, _  
Picture1.hdc, 0, 0, px%, py%, SRCCOPY)  
End Sub  
  
Private Sub Command3_Click()  
    Picture2.Cls  
    Call picrotate(Picture1, Picture2, (Pi * Slider1.Value) _  
/ 180)  
End Sub
```

(7) 按 F5 键运行，其效果如图 3-11 所示。



图 3-11 运行效果图

3.2.5 不知疲倦的小球

我们就用电脑来模拟一个碰撞的小球吧！啊，可怜的小球……

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 3-12 所示的窗体。

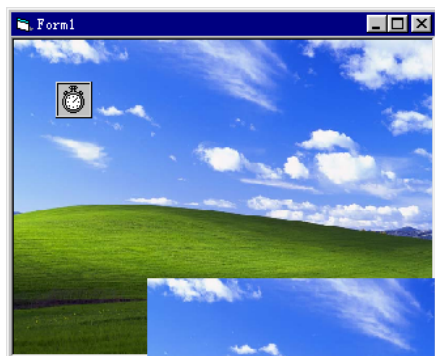


图 3-12 程序界面

该窗体所包含的控件及其属性，具体见表 3-5。

表 3-5 控件及其属性值表

控件	名称	部分属性值	
PictureBox	Picture1	Picture	Bitmap
	Picture2	Picture	Bitmap
		位置与 Picture1 不同，位于窗体底部	
Timer	Timer1	Interval	100

- (3) 在代码窗口中加入函数声明等，代码如下所示：

```
Option Explicit
```

```
Private Const SRCCOPY = &HCC0020
```



```
Private Declare Function BitBlt Lib "gdi32" (ByVal hdcDest_
As Long, ByVal nXDest As Long, ByVal nYDest As Long, ByVal_
nWidth As Long, ByVal nHeight As Long, ByVal hdcSrc As Long,_
ByVal nXSrc As Long, ByVal nySrc As Long, ByVal dwRop As Long)_
As Long
    Private Const BallR = 10
    Private Const BallD = 2 * BallR + 1
    Private CurX As Single
    Private CurY As Single
    Private OldX As Single
    Private OldY As Single
    Private VelX As Single
    Private Vely As Single
    Private Xmax As Single
    Private Ymax As Single

    Private Sub DrawBall()
        BitBlt Picture1.hDC, OldX - BallR, OldY - BallR, BallD,_
        BallD, Picture2.hDC, OldX - BallR, OldY - BallR, SRCCOPY
        OldX = CurX
        OldY = CurY
        Picture1.Circle (CurX, CurY), BallR
        Picture1.Refresh
    End Sub
```

在上面的代码中，我们定义三个常量、一个函数、一些模块级变量及一个自定义方法，其中常量 SRCCOPY 我们已在前面描述过，而函数 BitBlt 则是由动态链接库 GDI32.DLL 提供的，其作用是快速复制位图，其格式如下所示：

BitBlt(hdcDest, nXDest, nYDest, nWidth, nHeight, hdcSrc, nXSrc, nYSrc, dwRop)	
hdcDest	目的设备句柄
(nXDest,nYDest)	目的区域的左上角坐标
nWidth,nHeight	目的区域的宽和高
hdcSrc	源设备句柄
(nXSrc,nYSrc)	目的区域的左上角坐标
dwRop	复制时的绘图方式

常量 BallR 及 BallD 分别代表小球的半径和直径，(CurX, CurY) 代表小球的当前位置，(OldX, OldY) 代表小球的原来位置，包括其他的几个变量在内，它们都是模块级变量，可以被窗体内的所有过程存取。

自定义方法 DrawBall 用来清除原来位置上的小球以及画出新位置上的小球。因为在 Picture1 上用方法 Circle 画出小球后，原来的画面将被破坏，所以必须用包含相同图片的 Picture2 上的相同区域内容来覆盖（用函数 BitBlt 来实现）。PictureBox 控件的 Circle 方法的格式如下：

[object.]Circle (x, y), radius	
object	对象，如：PictureBox
(x,y)	圆心坐标



radius 圆的半径

(4) 为 Form 的 Load 事件输入如下代码：

```
Private Sub Form_Load()  
    Width = (Width - ScaleWidth) + Picture1.Width  
    Height = (Height - ScaleHeight) + Picture1.Height  
    Xmax = Picture1.ScaleWidth - BallR  
    Ymax = Picture1.ScaleHeight - BallR  
    Randomize  
    CurX = Int((Xmax - BallR + 1) * Rnd + BallR)  
    CurY = Int((Ymax - BallR + 1) * Rnd + BallR)  
    OldX = CurX  
    OldY = CurY  
    VelX = Int((10 - 5 + 1) * Rnd + 5)  
    Vely = Int((10 - 5 + 1) * Rnd + 5)
```

DrawBall

End Sub

(5) 为 Timer1 控件建立 Timer 事件的代码，具体如下：

```
Private Sub Timer1_Timer()  
    CurX = CurX + VelX  
    If (CurX > Xmax) Then  
        CurX = Xmax  
        VelX = -VelX  
    ElseIf (CurX < BallR) Then  
        CurX = BallR  
        VelX = -VelX  
    End If  
    CurY = CurY + Vely  
    If (CurY > Ymax) Then  
        CurY = Ymax  
        Vely = -Vely  
    ElseIf (CurY < BallR) Then  
        CurY = BallR  
        Vely = -Vely  
    End If
```

DrawBall

End Sub

(6) 按 F5 键运行，效果如图 3-13 所示。



图 3-13 运行效果图



试一试：

把上面例子中的小球改为椭圆或方块。

3.3 做一个放大镜

3.3.1 放大和缩小文字

眼睛疲劳了，做一个放大镜如何？

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 3-14 所示的窗体。

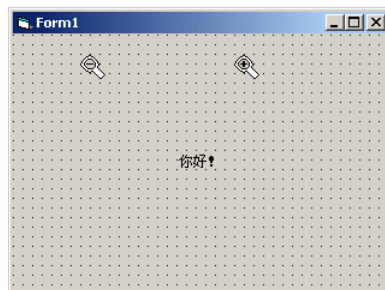


图 3-14 程序界面

该窗体所包含的控件及其属性，具体见表 3-6。

表 3-6 控件及其属性值表

控件	名称	部分属性值	
Image	imgZoomIn	Visible	False
	imgZoomOut	Visible	False
Label	Label1	Caption	你好！
		AutoSize	True



(3) 设置控件 imgZoomIn 和 imgZoomOut 的 Picture 属性, 单击属性值右边的 “...” 按钮, 导入各自的图片。

(4) 为 Form1 建立 Load 事件, 具体代码如下所示:

```
Private Sub Form_Load()  
    Label1.AutoSize=True  
    Label1.Top = Form1.ScaleHeight / 2 - Label1.Height / 2  
    Label1.Left = Form1.ScaleWidth / 2 - Label1.Width / 2  
End Sub
```

本段代码的作用是把 Label1 控件移动到窗体的中间。

(5) 为控件 Label1 建立 MouseDown 事件和 MouseMove 事件, 具体代码如下所示:

```
Private Sub Label1_MouseDown(Button As Integer, Shift As_  
Integer, X As Single, Y As Single)  
    Select Case Shift  
        Case 0  
        Case 1  
            If Label1.Width <= Form1.ScaleWidth And Label1._  
Height <= Form1.ScaleHeight Then  
                Label1.FontSize = Label1.FontSize + 10  
                Label1.Top = Form1.ScaleHeight / 2 - Label1.Height / 2  
                Label1.Left = Form1.ScaleWidth / 2 - Label1.Width / 2  
            End If  
        Case 2  
            If Label1.FontSize >= 10 Then  
                Label1.FontSize = Label1.FontSize - 10  
                Label1.Top = Form1.ScaleHeight / 2 - Label1.Height / 2  
                Label1.Left = Form1.ScaleWidth / 2 - Label1.Width / 2  
            End If  
    End Select  
End Sub  
  
Private Sub Label1_MouseMove(Button As Integer, Shift As_  
Integer, X As Single, Y As Single)  
    Select Case Shift  
        Case 0  
            Label1.MousePointer = vbArrow  
        Case 1  
            Label1.MousePointer = vbCustom  
            Label1.MouseIcon = imgZoomIn.Picture  
        Case 2  
            Label1.MousePointer = vbCustom  
            Label1.MouseIcon = imgZoomOut.Picture  
    End Select  
End Sub
```

其中 MouseDown 事件用于缩放字体的大小并显示在窗体的中间。而 MouseMove 事件则用于更改光标形状, 当我们按下 Shift 键并移动光标时, 将显示放大光标; 按下 Ctrl 键并移动光标时, 将显示缩小光标。



(6) 按 F5 键运行, 如图 3-15 所示。



图 3-15 运行效果图

3.3.2 简单的图形放大镜

有了文字放大镜还得有图形放大镜, 电脑的分工可真是一绝。

(1) 新建“标准 EXE”工程。

(2) 建立如图 3-16 所示的窗体。

该窗体所包含的控件及其属性, 具体见表 3-7。

表 3-7 控件及其属性值表

控件	名称	部分属性值	
Image	imgZoomIn	Visible	False
	imgZoomOut	Visible	False
	Image1	Stretch	True

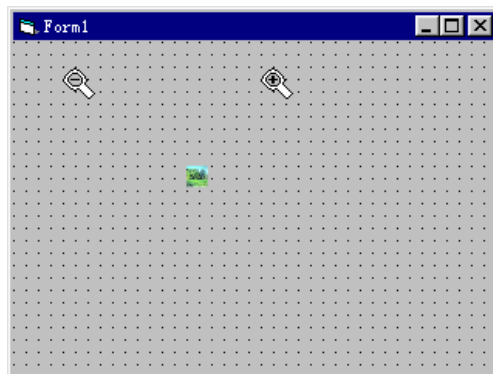


图 3-16 程序界面

(3) 设置控件 imgZoomIn、imgZoomOut 和 Image1 的 Picture 属性, 单击属性值右边的“...”按钮, 导入各自的图片。

(4) 在代码窗口中声明模块级变量 oldHeight 和 oldWidth, 用于保存图片原始的高度和宽度, 具体如下所示:



```
Dim oldHeight, oldWidth
```

(5) 为 Form1 建立 Load 事件，具体代码如下所示：

```
Private Sub Form_Load()  
    Image1.Stretch = True  
    Image1.Top = Form1.ScaleHeight / 2 - Image1.Height / 2  
    Image1.Left = Form1.ScaleWidth / 2 - Image1.Width / 2  
    oldHeight = Image1.Height  
    oldWidth = Image1.Width  
End Sub
```

本段代码的作用是把 Image1 控件移动到窗体的中间，并保存图片的原始高度和宽度。

(6) 为控件 Image1 建立 MouseDown 事件和 MouseMove 事件，具体代码如下所示：

```
Private Sub Label1_MouseDown(Button As Integer, Shift As_  
Integer, X As Single, Y As Single)  
    Select Case Shift  
        Case 0  
        Case 1  
            If Image1.Width <= Form1.ScaleWidth And Image1._  
Height <= Form1.ScaleHeight Then  
                Image1.Height = 1.5 * Image1.Height  
                Image1.Width = 1.5 * Image1.Width  
                Image1.Top = Form1.ScaleHeight / 2 - Image1.Height / 2  
                Image1.Left = Form1.ScaleWidth / 2 - Image1.Width / 2  
            End If  
        Case 2  
            If Image1.Height > oldHeight Or Image1.Width >_  
oldWidth Then  
                Image1.Height = Image1.Height / 1.5  
                Image1.Width = Image1.Width / 1.5  
                Image1.Top = Form1.ScaleHeight / 2 - Image1.Height / 2  
                Image1.Left = Form1.ScaleWidth / 2 - Image1.Width / 2  
            End If  
    End Select  
End Sub  
  
Private Sub Label1_MouseMove(Button As Integer, Shift As_  
Integer, X As Single, Y As Single)  
    Select Case Shift  
        Case 0  
            Image1.MousePointer = vbArrow  
        Case 1  
            Image1.MousePointer = vbCustom  
            Image1.MouseIcon = imgZoomIn.Picture  
        Case 2  
            Image1.MousePointer = vbCustom  
            Image1.MouseIcon = imgZoomOut.Picture  
    End Select  
End Sub
```

其中 MouseDown 事件用于缩放图片的大小并显示在窗体的中间。而 MouseMove 事件



则用于更改光标形状,当我们按下 Shift 键并移动光标时显示放大光标,按下 Ctrl 键并移动光标时显示缩小光标。

(7) 按 F5 键运行,如图 3-17 所示。

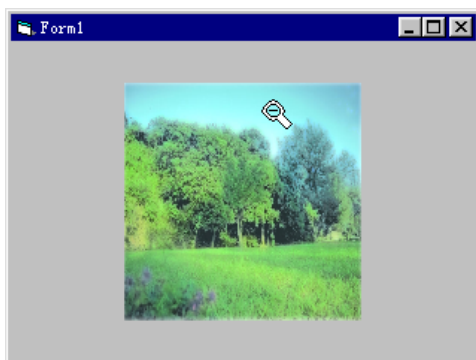


图 3-17 运行效果图

3.3.3 放大和缩小屏幕的内容

应该想个办法,让文字放大镜和图形放大镜合而为一。

(1) 新建“标准 EXE”工程。

(2) 建立如图 3-18 所示的窗体。

在该窗体中只有一个 Timer 控件 Timer1, Timer1 的 Interval 属性值为 50。

(3) 在代码窗口中加入函数声明等代码,具体如下所示:

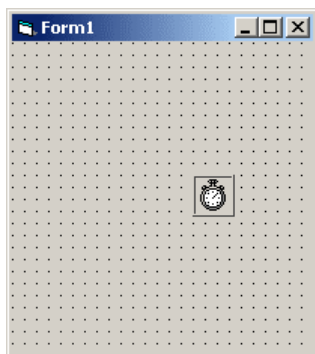


图 3-18 程序界面

```
Option Explicit
Private Type POINTAPI
    x As Long
    y As Long
End Type
Private Const SRCCOPY = &HCC0020
```



```
Private Declare Function GetCursorPos Lib "user32" _
(lpPoint As POINTAPI) As Long
Private Declare Function GetDC Lib "user32" (ByVal hwnd_
As Long) As Long
Private Declare Function StretchBlt Lib "gdi32" (ByVal_
hdcDest As Long, ByVal nXOriginDest As Long, ByVal nYOriginDest_
As Long, ByVal nWidthDest As Long, ByVal nHeightDest As Long, _
ByVal hdcSrc As Long, ByVal nXOriginSrc As Long, ByVal_
nYOriginSrc As Long, ByVal nWidthSrc As Long, ByVal nHeightSrc_
As Long, ByVal dwRop As Long) As Long
Private MyPoint As POINTAPI

Sub zoom()
Dim dx As Long
Dim dy As Long
Dim dl As Long
dl = GetCursorPos(MyPoint)
dx = MyPoint.x - 50
dy = MyPoint.y - 50
dl = StretchBlt(Me.hdc, 0, 0, 200, 200, GetDC(0), dx, _
dy, 100, 100, SRCCOPY)
End Sub
```

在本代码段中我们定义了一个自定义数据类型、一个常量、三个函数和一个自定义方法。其中用自定义数据类型 POINTAPI 说明的变量 MyPoint 是用于记录光标所在位置坐标值的，至于常量 SRCCOPY 及函数 StretchBlt，前面已经有它们的详细描述，而函数 GetCursorPos 和 GetDC 则是由动态链接库 USER32.DLL 提供的。

函数 GetDC 的作用是获取窗口的显示设备句柄，其格式如下：

```
rehdc=GetDC(hwnd)
hwnd          窗口句柄
rehdc          返回值，返回窗口的显示设备句柄
```

函数 GetCursorPos 的作用是获得鼠标的坐标值，其格式如下：

```
reval=GetCursorPos(lpPoint)
lpPoint        POINTAPI 类型的变量，获得的坐标值就保存于该变量
reval          返回值，是一个布尔值，当函数成功执行时返回真
```

自定义方法 zoom 用于把光标所在位置周围的图像放大后填入 Form1。

(4) 为 Timer1 控件建立 Timer 事件，具体代码如下：

```
Private Sub Timer1_Timer()
zoom
End Sub
```

(5) 按 F5 键运行，其效果如图 3-19 所示。



图 3-19 运行效果图



试一试：

给程序加入可以自行调节放大倍率的功能。

3.4 小结

在这一章中，我们讲解了颜色处理及图形处理的方法和技巧。

其中颜色的处理包括彩色的合成及调色板的构造，这里集中反映了电脑处理颜色的机理以及 RGB 函数的运用。而图像的处理则主要包括平移、滚动、镜像、翻转、旋转和反射。其中平移和滚动是通过修改控件的位置属性来实现的，而镜像、翻转、旋转和反射则是通过 GetPixel、SetPixel、StretchBlt 及 BitBlt 等 Windows API 函数来实现的。

文字、图形等的放大、缩小也是以控件的大小属性来实现的，而屏幕的放大、缩小则是以 StretchBlt 等 Windows API 函数来实现的。



第 4 章 多媒体制作

自从具备多媒体处理能力以来,计算机就得到了更进一步的应用,现在可以说多媒体功能几乎已经是计算机的必备功能之一。而 Visual Basic 是一套多媒体处理功能非常强大的语言,它包含了一套控制音频和视频设备但又与设备无关的多媒体控制指令 MCI,所以说用 Visual Basic 来开发多媒体是非常方便的。



本章的主要内容如下:

- (1) 动画光标的应用;
- (2) 制作简单的动画,以程序代码来控制对象运动;
- (3) 制作多媒体播放器,包括音频播放器及视频播放器;

4.1 制作自己的动画光标

4.1.1 自制一个简单的动画光标

自从微软公司推出 Windows 95 以来,整个操作系统都体现出一种人性化的趋势,大大加强了软件的可操作性。就以光标为例吧,无论应用程序处于执行还是处于等待状态,系统都会显示不同的光标,以便让我们随时了解当前程序的执行情况,其中尤以动画光标更为形象和生动。现在就让我们来自制一个动画光标吧。

(1) 新建“标准 EXE”工程。

(2) 在工具箱上单击鼠标右键弹出快捷菜单,选取“部件”菜单项,打开部件对话框。

在部件对话框中找到“Microsoft Windows Common Controls 6.0 (SP4)”项,并把该控件集加入到工具箱中。

(3) 建立如图 4-1 所示的窗体。该窗体所包含的控件及其属性,具体见表 4-1。

表 4-1 控件及其属性值表

控件	名称	部分属性值	
CommandButton	Command1	Caption	启动动画光标
	Command2	Caption	退出
Timer	Timer1	Enabled	False



		Interval	150
ImageList	ImageList1	(自定义)	见操作 (4) (5)

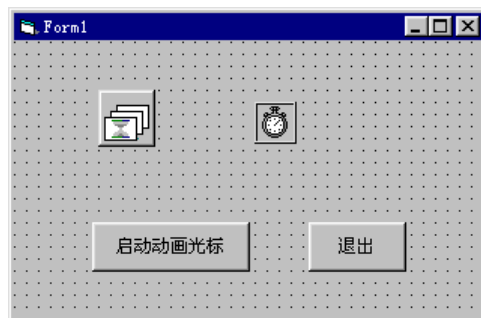


图 4-1 程序界面

接下来要在 ImageList1 控件中插入构成动画光标的每一幅 Ico 图片。

(4) 在窗体中选中 ImageList1 控件，然后单击属性对话框中“(自定义)”属性值右边的“...”按钮，如图 4-2 所示。

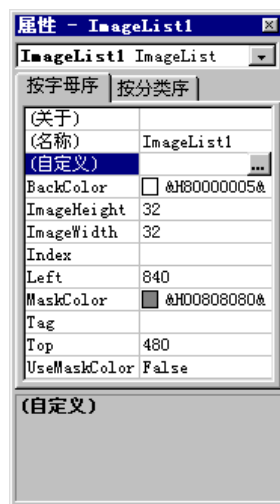


图 4-2 属性窗口

(5) 在弹出的属性页对话框中单击“插入图片”按钮来插入 Ico 图片，完成后按“确定”按钮返回，如图 4-3 所示。

(6) 在代码窗口中定义一模块级变量，以保存动画光标所显示的图片的索引值，具体代码如下：

```
Dim i As Integer
```

(7) 分别为 Command1 和 Command2 控件建立 Click 事件代码，具体如下：

```
Private Sub Command1_Click()  
    If Not Timer1.Enabled Then  
        i = 1
```




```
Timer1.Enabled = True
Me.MousePointer = vbCustom
Command1.Caption = "停止动画光标"
Else
Timer1.Enabled = False
Me.MousePointer = vbDefault
Command1.Caption = "停止动画光标"
i = 1
End If
End Sub

Private Sub Command2_Click()
End
End Sub
```



图 4-3 属性页对话框

(8) 为 Timer1 控件建立 Timer 事件代码，具体如下：

```
Private Sub Timer1_Timer()
Form1.MouseIcon = ImageList1.ListImages.Item(i).Picture
i = i + 1
If i = 16 Then
i = 1
End If
End Sub
```

这段代码的作用是每隔 150 毫秒（Timer 事件 150 毫秒触发一次）就在 ImageList1 控件中取下一张 Ico 图片作为光标，当索引数等于 16 时，使索引数等于 1（即回到第一张 Ico 图片），这就模拟出一个动画光标来。

(9) 按 F5 键运行程序，并单击“启动动画光标”按钮，如图 4-4 所示。



图 4-4 运行效果图

4.1.2 调用系统的动画光标

在 Windows 的系统目录下 (Windows 98 和 Windows XP 一般为 C:\Windows, Windows 2000 一般为 C:\WinNT) 有一个名为 CURSORS 的子目录, 其中有许多扩展名为 ANI 的动画光标文件, 这样的资源我们不能白白浪费, 该合理利用一下。

我们可以使用 LoadCursorFromFile、DestroyCursor、GetClassLong 及 SetClassLong 这 4 个 API 函数来达到目的。首先用 LoadCursorFromFile 函数把 ANI 动画光标文件装入内存, 然后用 GetClassLong 来得到窗口类的光标句柄属性值, 再用 SetClassLong 函数把 LoadCursorFromFile 函数装入动画光标句柄写入窗口类的光标句柄属性值, 最后当光标不再使用时用 DestroyCursor 函数删除动画光标。这里有必要说明一下窗口类的概念, 窗口类是一组定义窗口形状与操作的属性的集合, 在 Windows 操作系统中的任何一个窗口都有一个决定其特征及行为的窗口类, 而用同一个窗口类生成的窗口则都具有相同的特征及行为。

(1) 新建“标准 EXE”工程。

(2) 在工具箱上单击鼠标右键弹出快捷菜单, 选取“部件”菜单项, 打开部件对话框。

在部件对话框中找到“Microsoft Common Dialog Controls 6.0 (SP3)”项, 并把该控件集加入到工具箱中。

(3) 建立如图 4-5 所示的窗体。该窗体所包含的控件及其属性, 具体见表 4-2。

表 4-2 控件及其属性值表

控件	名称	部分属性值	
CommandButton	Command1	Caption	浏览
	Command2	Caption	启动动画光标
	Command3	Caption	退出
TextBox	Text1	Caption	(空)
CommonDialog	CommonDialog1	Filter	动画光标(*.Ani) *.Ani
		InitDir	C:\WINDOWS\CURSORS

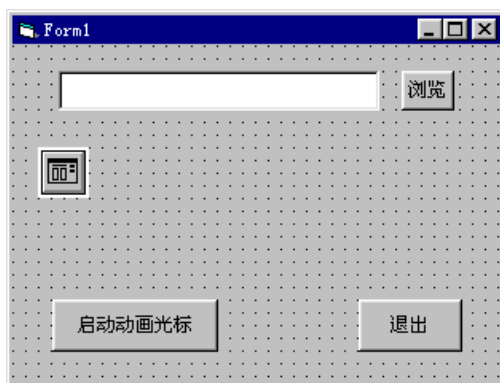


图 4-5 程序界面

(4) 在代码窗口中定义函数及变量等，具体代码如下：

```
Const GCL_HCURSOR = -12
Dim oldCursor, cursor
Dim isAni As Boolean
Private Declare Function LoadCursorFromFile Lib "user32" _
Alias "LoadCursorFromFileA" (ByVal lpFileName As String) As _
Long
Private Declare Function DestroyCursor Lib "user32" (ByVal _
hCursor As Long) As Long
Private Declare Function GetClassLong Lib "user32" Alias _
"GetClassLongA" (ByVal hWnd As Long, ByVal nIndex As Long) _
As Long
Private Declare Function SetClassLong Lib "user32" Alias _
"SetClassLongA" (ByVal hWnd As Long, ByVal nIndex As Long, _
ByVal dwNewLong As Long) As Long
```

在本代码段中我们定义了 4 个函数：LoadCursorFromFile、DestroyCursor、GetClassLong 及 SetClassLong，它们都是由动态链接库 USER32.DLL 提供的。函数 LoadCursorFromFile 的作用是由动画光标文件装入动画光标，其格式如下：

```
recur=LoadCursorFromFile(lpFileName)
lpFileName          动画光标的文件名
recur               返回值，返回装入的光标的句柄
```

函数 DestroyCursor 的作用是销毁光标。其格式如下：

```
reval=DestroyCursor(hCursor)
hCursor            要销毁的光标的光标句柄
reval              返回值
```

函数 GetClassLong 的作用是得到窗口类的某一成员变量的值，在本程序中我们将用它来得到与窗口类相关联的光标的句柄，其格式如下：

```
reval=GetClassLong(hWnd, nIndex)
hWnd               窗口句柄
nIndex             窗口类成员变量的索引值
```



reval 返回值，返回窗口类成员变量的值

函数 SetClassLong 的作用是设置窗口类的成员变量的值，在本程序中我们将用它来设置与窗口类相关联的光标的句柄，其格式如下：

reval=SetClassLong(hWnd, nIndex, dwNewLong)
hWnd 窗口句柄
nIndex 窗口类成员变量的索引值
dwNewLong 要设置的窗口类成员变量的新值
reval 返回值，返回窗口类成员变量的原来值

(5) 为 Form1 建立 Load 及 Unload 代码，具体代码如下：

```
Private Sub Form_Load()  
    oldCursor = GetClassLong((Me.hWnd), GCL_HCURSOR)  
    isAni = False  
End Sub  
  
Private Sub Form_Unload(Cancel As Integer)  
    If isAni Then  
        result = SetClassLong(Me.hWnd, GCL_HCURSOR, _  
oldCursor)  
        result = DestroyCursor(cursor)  
    End If  
End Sub
```

在窗体装入时触发 Load 事件，把与窗口类相关联的光标句柄保存在模块级变量 oldCursor 中，当窗体被销毁时再恢复原来的光标句柄。

(6) 为 Command1、Command2 及 Command3 分别建立 Click 事件代码。

```
Private Sub Command1_Click()  
    CommonDialog1.ShowOpen  
    If CommonDialog1.FileName = "" Then  
        MsgBox "请选择一光标文件", vbOKOnly  
    Else  
        Text1.Text = CommonDialog1.FileName  
    End If  
End Sub  
  
Private Sub Command2_Click()  
    If Text1.Text <> "" Then  
        If Not isAni Then  
            cursor = LoadCursorFromFile(Text1.Text)  
            result = SetClassLong(Me.hWnd, GCL_HCURSOR, cursor)  
            isAni = True  
            Command2.Caption = "停止动画光标"  
        Else  
            result = SetClassLong(Me.hWnd, GCL_HCURSOR, _  
oldCursor)  
            result = DestroyCursor(cursor)  
            isAni = False
```



```
Command2.Caption = "启动动画光标"  
End If  
Else  
MsgBox "请先选择一光标文件", vbOKOnly  
End If  
End Sub  
  
Private Sub Command3_Click()  
Call Form_Unload(0)  
End  
End Sub
```

(7) 按 F5 键运行程序，并打开一个动画光标文件，如图 4-6 所示。



试一试：

去掉窗体的 Unload 事件，并在“退出”按钮中直接加入退出程序的代码，观察两个程序运行的不同之处。



图 4-6 运行效果图

4.2 简单的动画

4.2.1 天降瑞雪

随着全球的气候转暖，近年来特别是在南方，就是冬天也很少有大雪纷飞的日子，现在就让我们在电脑里模拟一下，过把瘾如何？

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 4-7 所示的窗体。

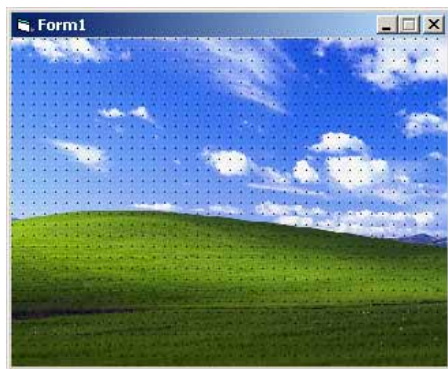



图 4-7 程序界面

本窗体包含了一个背景图片,这是通过设置 Form1 的 Picture 属性来实现的,只要单击属性对话框中“Picture”属性值右边的“”按钮,然后导入图片即可。

(3) 为 Form1 建立 Load 事件代码,具体如下所示:

```
Private Sub Form_Load()  
    Dim Snow(1000, 2), Dx As Integer  
    Form1.Show  
    DoEvents  
    Randomize  
    Dx = 500  
    For J = 1 To Dx  
        Snow(J, 0) = Int(Rnd * Form1.Width)  
        Snow(J, 1) = Int(Rnd * Form1.Height)  
        Snow(J, 2) = 10 + (Rnd * 20)  
    Next J  
    Do While Not (DoEvents = 0)  
        For Temp = 1 To 10  
            For I = 1 To Dx  
                Snow(I, 1) = Snow(I, 1) + Snow(I, 2)  
                If Snow(I, 1) > Form1.Height Then  
                    Snow(I, 1) = 0: Snow(I, 2) = 2 + (Rnd * 30)  
                    Snow(I, 0) = Int(Rnd * Form1.Width)  
                End If  
                Circle (Snow(I, 0), Snow(I, 1)), 20 * Rnd, RGB(255, _  
255, 255)  
            Next I  
            Form1.Cls  
        Next Temp  
    Loop  
End  
End Sub
```

其中 Randomize 语句用于初始化随机数生成器,即以系统计时器返回的值作为新的随机序列产生的种子值(使得随机函数每次都产生不同的序列),而函数 Rnd 的作用是产生一个随机数,在本程序段中随机函数用于生成雪花的坐标,而变量 Dx 则决定了雪下的紧密程度。



(4) 按 F5 键运行程序，效果如图 4-8 所示。



图 4-8 运行效果图



试一试：
在程序中加入可以自行调节雪下的紧密程度的功能。

4.2.2 节日彩灯

按一定时间间隔不同颜色不停转动的霓虹灯 相信大家都看过了吧 ,是不是很吸引人？
让我们也来做一个怎样？

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 4-9 所示的窗体。

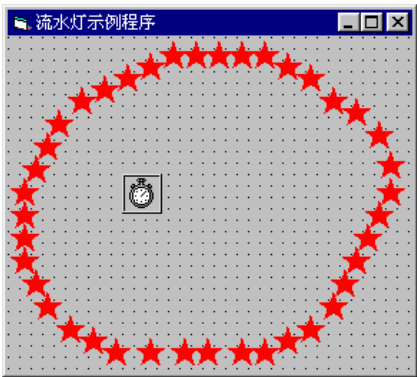


图 4-9 程序界面

该窗体所包含的控件及其属性，具体见表 4-3。

表 4-3 控件及其属性值表

控件	名称	部分属性值	
Label	Label1(0)~Label1(39)	Caption	
		ForeColor	&H000000FF&(红色)



Timer	Timer1	Interval	200
-------	--------	----------	-----

(3) 在代码窗口中两模块级变量 I, J 用于计数, 具体如下所示:

```
Option Explicit
Private I As Integer, J As Integer
```

(4) 为 Form1 建立 Load 事件代码, 具体如下所示:

```
Private Sub Form_Load()
    I = 0
    For J = 0 To 9
        Label1(J * 4).ForeColor = &HFF&
        Label1(J * 4 + 1).ForeColor = &HFF00&
        Label1(J * 4 + 2).ForeColor = &HFFFF&
        Label1(J * 4 + 3).ForeColor = &HFF0000
    Next J
    For J = 0 To 39
        Label1(J).Visible = False
    Next J
End Sub
```

在该代码段中我们对控件数组 Label1(0)~Label1(39)进行了初始化, 对于下标为 4 的倍数的控件, 设置其前景色为红色; 对于下标为 4 的倍数加 1 的控件, 设置其前景色为绿色; 而对于下标为 4 的倍数加 2 的控件, 设置其前景色为黄色; 最后对于下标为 4 的倍数加 3 的控件, 设置其前景色为蓝色。这样 40 个 被分成不同颜色的 4 个组, 并把每个 Label 控件的 Visible 属性值设置为 False。

(5) 为 Timer1 控件建立 Timer 事件的代码, 具体如下:

```
Private Sub Timer1_Timer()
    If I = 40 Then
        I = 0
    End If
    I = I + 1
    For J = 0 To 39
        Label1(J).Visible = False
    Next J
    Select Case I Mod 4
        Case 3
            For J = 0 To 9
                Label1(J * 4 + 2).Visible = True
            Next J
        Case 2
            For J = 0 To 9
                Label1(J * 4 + 1).Visible = True
            Next J
        Case 1
            For J = 0 To 9
                Label1(J * 4).Visible = True
            Next J
        Case 0
```




```
For J = 0 To 9
    Label1(J * 4 + 3).Visible = True
Next J
End Select
End Sub
```

这样，随着时间的不同显示不同颜色组的 Label，就模拟出了节日彩灯。

(6) 按 F5 键运行程序，效果如图 4-10 所示。

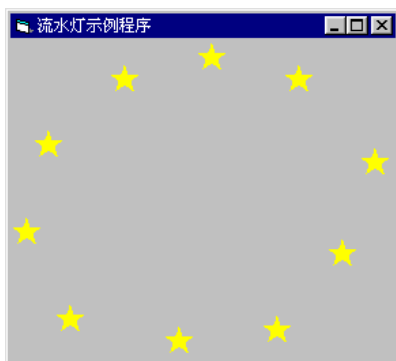


图 4-10 运行效果图



试一试：

让彩灯以顺时针方向转动。

4.3 制作播放器

4.3.1 来一段美妙的音乐

一段美妙的音乐能使人全身心地放松。是不是编程编得很累了，来一段音乐怎么样？

(1) 新建“标准 EXE”工程。

(2) 在工具箱上单击鼠标右键弹出快捷菜单，选取“部件”菜单项，打开部件对话框。在部件对话框中找到“Microsoft Common Dialog Controls 6.0 (SP3)”项，并把该控件集加入到工具箱中。

(3) 建立如图 4-11 所示的窗体。该窗体所包含的控件及其属性，具体见表 4-4。

表 4-4 控件及其属性值表

控件	名称	部分属性值	
CommandButton	Command1	Caption	播放
	Command2	Caption	退出
CommonDialog	CommonDialog1	Filter	音频文件(*.Wav) *.wav

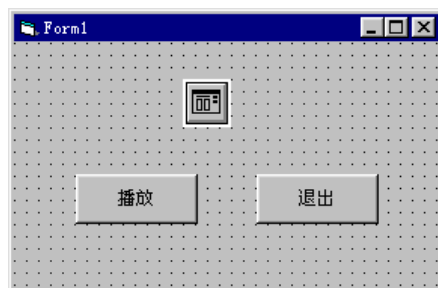


图 4-11 程序界面

(4) 在代码窗口中定义函数和常量，具体代码如下：

```
Option Explicit
Private Declare Function sndPlaySound Lib "winmm.dll" _
Alias "sndPlaySoundA" (ByVal lpszSound As String, ByVal _
fuSound As Long) As Long
Const SND_SYNC = &H0
Const SND_ASYNC = &H1
Const SND_NODEFAULT = &H2
Const SND_LOOP = &H8
Const SND_NOSTOP = &H10
```

在该代码段中我们定义了一个函数和一些常量。其中函数 `sndPlaySound` 由动态链接库 `WINMM.DLL` 提供，该函数用于播放 WAV 格式的音乐文件，其格式如下：

```
reval=sndPlaySound(lpszSound, fuSound)
lpszSound      要播放的 WAV 文件
fuSound        播放属性，其值有：SND_SYNC、SND_ASYNC、
                SND_NODEFAULT、SND_LOOP、SND_NOSTOP
reval          返回值
```

(5) 分别为 `Command1` 和 `Command2` 建立 Click 事件代码，具体代码如下所示：

```
Private Sub Command1_Click()
    CommonDialog1.ShowOpen
    Dim soundfile As String, uFlags As Long, returnval As _
Long
    soundfile = CommonDialog1.FileName
    If soundfile <> "" Then
        uFlags = SND_ASYNC Or SND_NODEFAULT
        returnval = sndPlaySound(soundfile, uFlags)
    End If
End Sub

Private Sub Command2_Click()
    End
End Sub
```

(6) 按 F5 键运行程序。



4.3.2 我也能放电影

有了音乐还不够，当然最好再来一段电影，现在就让我们来播放一段电影。

(1) 新建“标准 EXE”工程。

(2) 在工具箱上单击鼠标右键弹出快捷菜单，选取“部件”菜单项，打开部件对话框。在部件对话框中找到“Microsoft Common Dialog Controls 6.0 (SP3)”项，并把该控件集加入到工具箱中。

(3) 建立如图 4-12 所示的窗体。



图 4-12 程序界面

该窗体所包含的控件及其属性，具体见表 4-5。

表 4-5 控件及其属性值表

控件	名称	部分属性值	
CommandButton	Command1	Caption	播放
	Command2	Caption	循环播放
	Command3	Caption	退出
Timer	Timer1	Enabled	False
		Interval	50
PictureBox	Picture1	默认	
CommonDialog	CommonDialog1	Filter	视频文件(*.avi) *.avi

(4) 在代码窗口中声明两个函数，具体代码如下：

```
Private Declare Function mciSendString Lib "winmm.dll" _
Alias "mciSendStringA" (ByVal lpszCommand As String, ByVal _
lpszReturnString As String, ByVal cchReturn As Long, ByVal _
hwndCallback As Long) As Long

Private Declare Function mciGetErrorString Lib _
"winmm.dll" Alias "mciGetErrorStringA" (ByVal fdwError As _
Long, ByVal lpszErrorText As String, ByVal cchErrorText As _
Long) As Long

Private Function fSendMCICommand(ltCmd As String)
Dim ltResult As String
```



```

        Dim liStatus As String
        Dim liI As Integer
        ltResult = String$(256, 0)
        liStatus = mciSendString(ltCmd, ltResult, Len(ltResult), _
0)
        If (liStatus <> 0) Then
            liStatus = mciGetErrorString(liStatus, ltResult, _
Len(ltResult))
        End If
        liI = InStr(ltResult, Chr$(0))
        If liI <> 0 Then
            fSendMCICommand = Left$(ltResult, liI - 1)
        Else
            fSendMCICommand = ltResult
        End If
    End Function

    Private Function playavi(aviname As String, whichavi As _
Integer) As Integer
        Dim ltreturnval As String
        Dim llFocusofAttention As Long
        Dim ltfile2play As String
        Dim liX As Integer
        Dim liY As Integer
        Dim k As String
        ltreturnval = fSendMCICommand("Close AVIVideo")
        ltreturnval = fSendMCICommand("Close all")
        Picture1.Cls
        Picture1.Refresh
        ltfile2play = aviname
        ltreturnval = fSendMCICommand("Open " & ltfile2play & _
" Alias Video1 Wait")
        ltreturnval = fSendMCICommand("sysinfo Video1_
installname")
        ltreturnval = fSendMCICommand("realize Video1_
Background")
        Dim returnstring As String * 256
        llFocusofAttention = Picture1.hWnd
        ltreturnval$ = fSendMCICommand("Window Video1 handle"_
+ Str$(llFocusofAttention))
        liX = Picture1.ScaleWidth
        liY = Picture1.ScaleHeight
        ltreturnval = fSendMCICommand("Put Video1 Destination_
at 0 0 " & liX & " " & liY)
        Select Case whichavi
            Case 1
                ltreturnval = fSendMCICommand("Play Video1 ")
            Case 0
                ltreturnval = fSendMCICommand("Play Video1 repeat")
        End Select
    End Function

```



```
End Select  
End Function
```

在此我们定义了两个函数：mciSendString 和 mciGetErrorString，它们都是由动态链接库 WINMM.DLL 提供的。

函数 mciSendString 的作用是向多媒体设备发送字符串命令，从而控制多媒体设备工作。其格式如下：

```
reval=mciSendString(lpszCommand,lpszReturnString, cchReturn, hwndCallback)  
lpszCommand          命令字符串，比如：Play...表示播放  
lpszReturnString      返回信息字符串  
cchReturn             返回字符串的长度  
hwndCallback          回调函数，可设为空  
reval                 返回值，如正确执行命令则返回 0
```

函数 mciGetErrorString 的作用是获得多媒体设备的出错信息。其格式如下：

```
reval=mciGetErrorString(fdwError,lpszErrorText, cchErrorText)  
fdwError              错误号，由 mciSendString 或 mciSendCommand 产生  
lpszErrorText         出错信息  
cchErrorText          出错信息长度  
reval                 返回值
```

另外，本代码段还定义了两个自定义函数：fSendMCICommand 和 playavi，其中 fSendMCICommand 是用于向多媒体设备发送 MCI 命令的，而 playavi 是用来播放 AVI 格式的视频文件的。

(5) 分别为 Command1、Command2 和 Command3 控件建立 Click 事件，其代码如下所示：

```
Private Sub Command1_Click()  
    Dim fs  
    Dim i As Integer  
    Dim filename1 As String, filename2 As String  
    CommonDialog1.ShowOpen  
    filename1 = CommonDialog1.FileName  
    If filename1 <> "" Then  
        Set fs = CreateObject("Scripting.FileSystemObject")  
        Set f = fs.GetFile(filename1)  
        filename2 = f.ShortPath  
        Picture1_Click  
        i = playavi(filename2, 1)  
        Timer1.Enabled = True  
    End If  
End Sub  
  
Private Sub Command2_Click()  
    Dim fs  
    Dim i As Integer  
    Dim filename1 As String, filename2 As String  
    CommonDialog1.ShowOpen
```



```

filename1 = CommonDialog1.FileName
If filename1 <> "" Then
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(filename1)
    filename2 = f.ShortPath
    i = playavi(filename2, 0)
End If
End Sub

Private Sub Command3_Click()
    Timer1.Enabled = False
    ltreturnval = fSendMCICCommand("Close")
    ltreturnval = fSendMCICCommand("Close All")
End
End Sub

```

在本段代码中的下列语句需作一下说明：

```

Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(filename1)
filename2 = f.ShortPath

```

其中语句 Set fs= CreateObject("Scripting.FileSystemObject")的作用是建立一个文件系统对象，并把它赋值给 fs，而语句 Set f=fs.GetFile(filename1)的作用是用 filename1 文件建立一个文件对象，最后语句 filename2=f.ShortPath 的作用是得到 filename1 的短文件名（所谓短文件名，即文件名由文件主名、分隔符和扩展名组成，其中文件主名长度不能超过 8 个字符，扩展名不能超过 3 个字符，分隔符为“.”，也就是 8.3 文件名格式。这是从 DOS 操作系统中遗留下来的，相对于 Windows 95 等操作系统所支持的长文件名命名方式来说叫做“短文件名”。在 Windows 操作系统中任意一个长文件名都对应有一个短文件名，比如长文件名“频道屏幕保护程序.SCR”就对应短文件名“频道屏~1.SCR”），然后将得到的短文件名赋值给 filename2。为什么一定要使用短文件名格式呢？因为我们使用函数 mciSendString 向多媒体设备发送字符串命令，如果文件名内含有空格（长文件名），将使该字符串命令不能被正确识别。

（6）为 Picture1 控件建立 Click 事件代码，具体如下：

```

Private Sub Picture1_Click()
    Dim ltreturnval As String
    ltreturnval = fSendMCICCommand("Close")
    ltreturnval = fSendMCICCommand("Close All")
    Timer1.Enabled = False
    Show
End Sub

```

（7）为 Timer1 控件建立 Timer 事件代码，具体如下：

```

Dim ltreturnval As String
Dim filename2 As String
On Error Resume Next
liStatus = fSendMCICCommand("Status Video1 mode")
If liStatus = "stopped" Then
    Timer1.Enabled = False

```



```
ltreturnval = fSendMCICommand("Close")
ltreturnval = fSendMCICommand("Close all")
Picture1_Click
End If
End Sub
```

(8) 按 F5 键运行程序，并打开一个 AVI 文件，如图 4-13 所示。

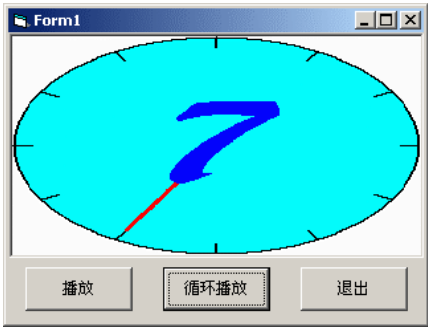


图 4-13 运行效果图

4.3.3 做一个自己的“超级解霸”

上面我们做的播放器有很大的局限性，它只能用于播放 AVI 文件。如果能像“超级解霸”一样播放多种格式的视频文件，使用起来又方便，那就酷毙了。

- (1) 新建“标准 EXE”工程。
- (2) 打开菜单编辑器，建立如图 4-14 所示的菜单，其属性见表 4-6。

表 4-6 菜单项属性表

标题	名称	复选	有效性	可见性
文件	mnFile	否	有效	可见
打开	mnOpen	否	有效	可见
播放	mnPlay	否	无效	可见
退出	mnExit	否	有效	可见





图 4-14 菜单编辑器

(3) 在工具箱上单击鼠标右键弹出快捷菜单, 选取“部件”菜单项, 打开部件对话框。在部件对话框中找到“Microsoft Common Dialog Controls 6.0 (SP3)”及“Windows Media Player”项, 并把这两个控件集加入到工具箱中。

(4) 建立如图 4-15 所示的窗体。该窗体所包含的控件及其属性, 具体见表 4-7。

表 4-7 控件及其属性表

控件	名称	部分属性值	
		AutoRewind	False
MediaPlayer	MediaPlayer1	DisplaySize	0
		Shape	0
CommonDialog	CommonDialog1	Filter	视频文件(*.avi;*.mpg) *.avi;*.mpg

其中 MediaPlayer1 控件的部分属性还可以通过该控件的属性表来设置。首先在 MediaPlayer1 属性窗口中选择“(自定义)”, 然后单击该属性值右边的“...”按钮, 如图 4-16 所示。

这样就会弹出一个“选项”对话框, 设置完选项后按“确定”按钮返回, 如图 4-17 所示。



图 4-15 程序界面



图 4-16 属性窗口

(5) 分别为“打开”、“播放”和“退出”菜单项建立 Click 事件，具体代码如下所示：

```
Private Sub mnOpen_Click()  
    CommonDialog1.ShowOpen  
    If CommonDialog1.FileName <> "" Then  
        MediaPlayer1.FileName = CommonDialog1.FileName  
        MediaPlayer1.Pause  
        mnPlay.Enabled = True  
    End If  
End Sub  
  
Private Sub mnPlay_Click()  
    If MediaPlayer1.FileName <> "" Then  
        If MediaPlayer1.PlayState = mpPlaying Then  
            MediaPlayer1.Pause  
            mnPlay.Caption = "播放"  
        Else  
            MediaPlayer1.Play  
            mnPlay.Caption = "暂停"  
        End If  
    End If  
End Sub  
  
Private Sub mnExit_Click()  
    End  
End Sub
```



图 4-17 选项对话框

(6) 为 MediaPlayer1 控件建立 PlayStateChange 事件代码，具体如下所示：

```
Private Sub MediaPlayer1_PlayStateChange(ByVal OldState_
As Long, ByVal NewState As Long)
    Select Case NewState
        Case mpStopped, mpPaused
            mnPlay.Caption = "播放"
        Case mpPlaying
            mnPlay.Caption = "暂停"
    End Select
End Sub
```

这段代码的作用是当 MediaPlayer1 控件的状态变化时，使菜单项 mnPlay 的标题也跟着一起变化。

(7) 按 F5 键运行程序，并打开一个视频文件，如图 4-18 所示。



图 4-18 运行效果图

4.4 小结

在这一章中，我们讲解了动画、音频和视频等多媒体的处理方法以及音、视频播放器的制作。

其中动画光标是通过多幅图片合成的，按次序来显示不同图片从而达到动画效果，而系统动画光标文件 ANI 文件也是由多幅图片合成，因此本例的动画光标制作方法，与调用系统动画光标文件其实际的最终实现原理还是相同的。

对于“天降瑞雪”和“节日彩灯”则是通过程序代码控制对象的显示与否，来实现动画效果，前一个以随机函数实现雪花出现的任意性，而后一个则以时间间隔来控制“彩灯”的显示。

在制作播放器一节中则是用 `sndPlaySound` 及 `mciSendString` 等 Windows API 函数来实现 Wave 文件与 AVI 文件的播放，另外又以 MediaPlayer 控件实现了播放多种视频格式的“超级解霸”。



第 5 章 数据库应用

数据库是信息的集合，这种集合与特定的主题或目标相联系，它可以说是向用户和应用程序提供数据的基地。随着计算机技术的不断发展，数据库的强大的数据存储和数据管理功能在应用中被不断地体现出来，其应用也随之逐渐广泛。



本章的主要内容如下：

- (1) 数据的创建及数据的录入；
- (2) 在应用程序中实现对数据库的操纵，主要包括数据的浏览、编辑、追加和删除。

5.1 建立数据库

5.1.1 制作电子卡片

制作电子卡片，让电脑充分发挥其快速的检索与处理信息功能。

(1) 单击“开始”菜单并选择“程序”子菜单，在“程序”菜单中选取“Microsoft Access”菜单项来启动 Microsoft Access。

(2) 在 Microsoft Access 窗口的右侧，有一个如图 5-1 所示的新建文件菜单。



图 5-1 新建文件菜单

(3) 在新建文件菜单中选取“空数据库”项，将打开如图 5-2 所示的“文件新建数据库”的文件创建窗口，在选择好正确的文件夹并对文件进行命名后，按“创建”按钮结束。



图 5-2 文件创建窗口

(4) 这时屏幕上出现如图 5-3 所示的数据库管理窗口。

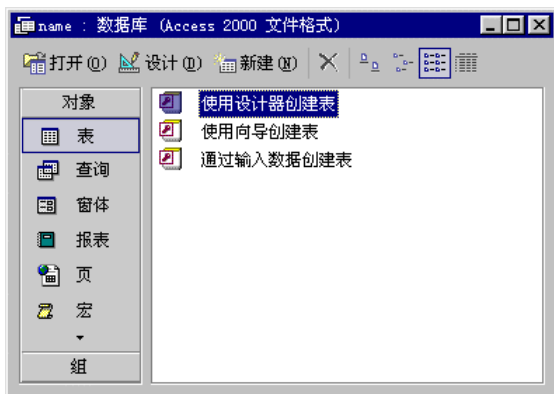


图 5-3 数据库管理窗口

(5) 现在我们可以建立表格来存放数据了。双击“使用设计器创建表”项，打开如图 5-4 所示的表设计窗口。



图 5-4 表设计窗口

表的字段及其属性具体见表 5-1。

表 5-1 表的字段及其属性

字段名称	数据类型	字段大小	必填字段
bf_Name	文本	8	是
bf_Sex	文本	2	是
bf_HomeTel	文本	20	否
bf_HomeAddress	文本	100	否



(6) 当我们设计完表后关闭设计窗口时, 会弹出如图 5-5 所示的“另存为”窗口, 我们将表命名为 Address, 到此为止我们已经设计了一个数据表。

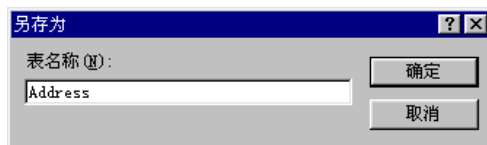


图 5-5 表的另存为窗口

5.1.2 资料存档

(1) 打开 name.mdb 数据库, 将出现如图 5-6 所示的界面。

(2) 双击 Address 表就会打开如图 5-7 所示的数据输入窗口, 我们把同学的记录数据输入到里面。

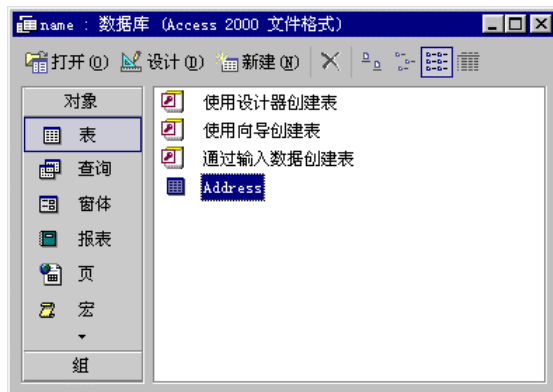


图 5-6 数据库管理窗口



Address : 表				
	bf_Name	bf_Sex	bf_HomeTel	bf_HomeAddress
▶	孙剑	男	0574-88387959	东钱湖镇殷湾村
	汤凌峰	男	0574-88477771	章水镇
	吴雪娜	女	0574-88436064	鄞县龙观乡桓村
	应超男	女	0574-88356252	邱隘镇青年路35号
	张文君	男	13008962253	姜山镇励江岸村上河塘
	张卡卡	男	0574-88434440	鄞县洞桥镇市场西弄12号
	李弘烈	男	0574-88441183	石碇镇栎社村
	杨优	女	0574-88024935	集仕港镇新庙跟村
	汪东达	男	0574-88420389	横街镇大雷村
	沈超	男	0574-88022762	集仕港镇银杏新村93号
	陈玉林	男	0574-88298923	古林布政藕池村
	陈吉云	女	0574-88017834	高桥镇长乐村轴承厂工房1幢30
	周莹莹	女	0574-88314680	塘溪镇大碧浦村
	姜静	女	0574-88350058	梅圩钱家村
	屠一	女	0574-87837378	宁波江东区白鹤新村162幢304
记录: 1 共有记录数: 105				

图 5-7 数据编辑窗口

5.2 我的“电子同学录”

- (1) 新建“标准 EXE”工程，并保存为 Address.vbp。
- (2) 在工具箱上单击鼠标右键弹出快捷菜单，选取“部件”菜单项，打开部件对话框。在部件对话框中找到“Microsoft DataGrid Control 6.0 (SP5)”项，并把该控件集加入到工具箱中。
- (3) 打开菜单编辑器，建立如图 5-8 所示的菜单。菜单项的具体属性见表 5-2。

表 5-2 菜单项属性表

标题	名称	复选	有效性	可见性
地址簿	mnAdress	否	有效	可见
浏览	mnBrow	否	有效	可见
添加	mnAdd	否	有效	可见
续表				
标题	名称	复选	有效性	可见性
编辑	mnEdit	否	有效	可见
删除	mnDel	否	有效	可见
-	mnBar	否	有效	可见
退出	mnExit	否	有效	可见
帮助	mnHelp	否	有效	可见
关于	mnAbout	否	有效	可见



图 5-8 菜单编辑器

(4) 建立如图 5-9 所示的窗体，并保存为 mainForm.frm。



图 5-9 程序界面


(5) 为工程建立数据环境以便在程序中使用数据库，单击“视图”菜单并选取“数据视图窗口”菜单项，这时将弹出如图 5-10 所示的“数据视图”窗口。选中窗口中的“数据环境连接”，并单击“”按钮，将打开如图 5-11 所示的“数据环境设计器”窗口，并在属性窗口中把数据环境改名为“dbDataEnvironment”，连接 Connection1 改名为“dbConnection”。



图 5-10 数据视图窗口

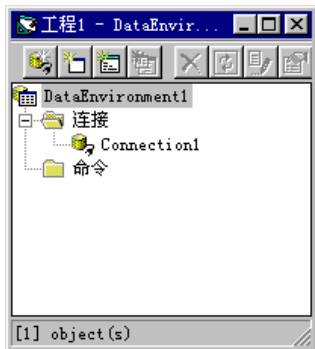


图 5-11 数据环境设计器窗口

(6) 为 dbConnection 设置链接属性, 选取 “dbConnection” 并单击鼠标右键, 在弹出的快捷菜单中选取 “属性” 菜单项, 这时将弹出 “数据链接属性” 窗口, 如图 5-12 所示。



图 5-12 数据链接属性窗口

在图中选取 “Microsoft Jet 4.0 OLE DB Provider” 项 (因为我们使用的是 Access 数据库), 并单击 “下一步” 按钮进入 “连接” 设置页, 具体如图 5-13 所示。在输入数据库名、用户名和密码等设置后可单击 “测试连接” 按钮, 以检验连接是否正常。如果测试返回正常信息 (如图 5-14 所示), 则我们就可以使用该连接了, 按 “确定” 按钮返回。

(7) 在图 5-11 的 “数据环境设计器” 窗口中选取 “命令” 项, 单击鼠标右键, 在弹出的快捷菜单中选取 “添加命令” 菜单项加入命令, 并把其重命名为 “dbCommand”, 如图 5-15 所示。

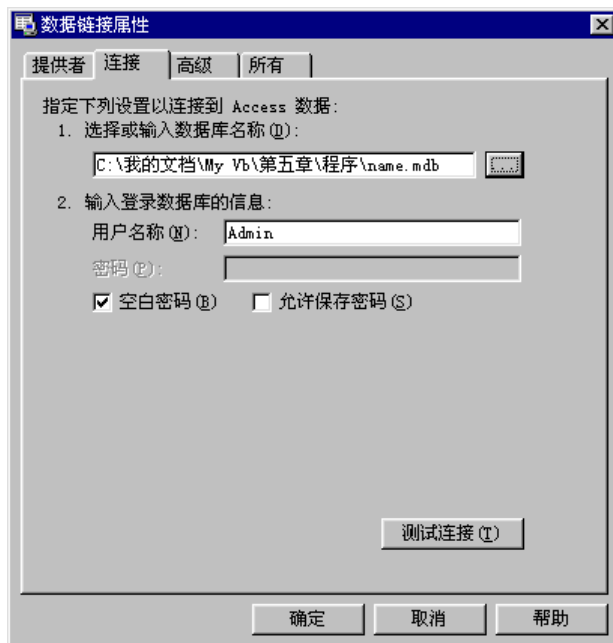


图 5-13 数据链接属性窗口



图 5-14 测试信息

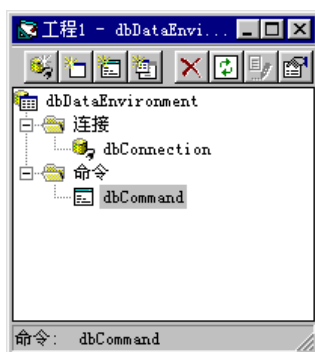


图 5-15 数据环境设计器窗口

(8) 接下来设置 dbCommand 的属性, 选取“dbCommand”菜单项, 单击鼠标右键, 在弹出的快捷菜单中选取“属性”菜单项, 将出现如图 5-16 所示的属性窗口。



图 5-16 dbCommand 属性窗口

设置完成后单击“高级”标签，打开“高级”选项卡设置返回的记录集属性，具体如图 5-17 所示，设定完毕后按“确定”按钮返回，这样我们就建立了一个完整的数据环境。



图 5-17 dbCommand 属性窗口

(9) 在代码窗口中声明一个模块级的公用变量 myRecordSet，用来保存数据环境的记录集，以便在不同的窗体模块中调用。具体代码如下：

```
Public myRecordSet As Recordset
```

(10) 为 Form 建立 Load 事件，具体代码如下所示：

```
Private Sub Form_Load()  
    Set myRecordSet = dbDataEnvironment.Recordsets.Item(1)  
End Sub
```

这段代码的作用是在窗体被装入时保存数据环境的记录集至 myRecordSet。

(11) 为 mnExit 菜单项建立 Click 事件，具体代码如下所示：

```
Private Sub mnExit_Click()
```



```
End
End Sub
```

5.2.1 浏览好友资料


- (1) 打开 Address.vbp 工程。
- (2) 单击工具栏上的“”按钮，在“添加窗体”对话框中选取“对话框”。
- (3) 设计对话框窗体，使之如图 5-18 所示。



图 5-18 浏览窗口界面

该窗体所包含的控件及其属性，具体见表 5-3。

表 5-3 控件及其属性值表

控件	名称	部分属性值	
CommandButton	OKButton	Caption	确定
DataGrid	browDataGrid	DataSource	dbDataEnvironment
		DataMember	dbCommand
		AllowAddNew	False
		AllowArrows	True
		AllowDelete	False
		AllowUpdate	False

(4) 设置控件 browDataGrid 的外观及部分属性值。首先在窗体中选中 brpwDataGrid 控件，单击鼠标右键，在弹出的快捷菜单中选取“编辑”菜单项，如图 5-19 所示。接下来马上在 brpwDataGrid 控件上单击鼠标右键，就会弹出与刚才不一样的快捷菜单，如图 5-20 所示。这样我们就可以选取“插入”或“追加”菜单项来加入 brpwDataGrid 控件的显示列，



同样可以选取“删除”菜单项来删除 brpwDataGrid 控件的显示列。



图 5-19 快捷菜单

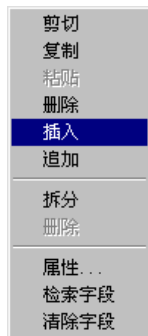


图 5-20 快捷菜单

(5) 在窗体中选中 brpwDataGrid 控件, 单击鼠标右键, 在弹出的快捷菜单中选取“属性”菜单项, 将会弹出属性设置窗口, 如图 5-21 所示。



图 5-21 属性页窗口

在设置好 AllowAddNew、AllowDelete 和 AllowUpdate 属性值后, 我们就可以选取“列”页来设置 browDataGrid 控件的显示列与数据库字段的对应关系, 具体如图 5-22 所示。

同样我们可以选取“布局”页来调整控件 browDataGrid 的外观, 具体如图 5-23 所示。

(6) 为 Form 建立 Activate 事件, 具体代码如下所示:

```
Private Sub Form_Activate()  
    mainForm.myRecordSet.MoveFirst  
End Sub
```



过程 Form_Activate 的作用是当窗体被激活时，自动调用记录集 myRecordSet 的 MoveFirst 方法，把当前记录设置为第一条记录，即记录指针移动到第一条记录。

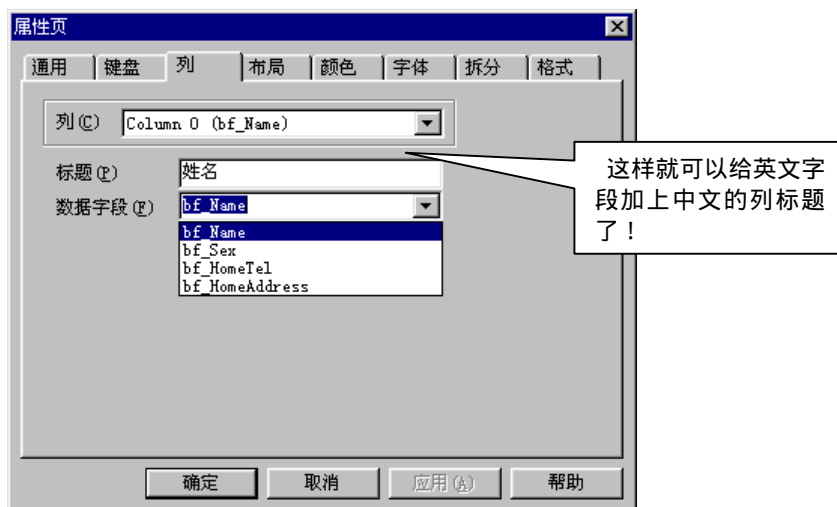


图 5-22 属性页窗口



图 5-23 属性页窗口

(7) 为控件 OKButton 建立 Click 事件，具体代码如下所示：

```
Private Sub OKButton_Click()  
    Unload Me  
End Sub
```

其作用是单击 OKButton 按钮后撤消当前窗口。

(8) 为 mainForm 窗体菜单的 mnBrow 菜单项建立 Click 事件，具体代码如下所示：

```
Private Sub mnBrow_Click()
```



```
browDialog.Show 1  
End Sub
```


该代码的作用是当用户单击 mnBrow 菜单项时，以模态对话框的形式打开窗体 browDialog，即当 browDialog 打开时，应用程序的其他窗口都失去响应，直到 browDialog 窗体被关闭。

(9) 按 F5 键运行程序，并单击“地址簿”菜单，选取其中的“浏览”菜单项，效果如图 5-24 所示。



图 5-24 运行效果图

5.2.2 加入好友

- (1) 打开 Address.vbp 工程。
- (2) 单击工具栏上的“”按钮，在“添加窗体”对话框中选取“对话框”。
- (3) 设计对话框窗体，使之如图 5-25 所示。

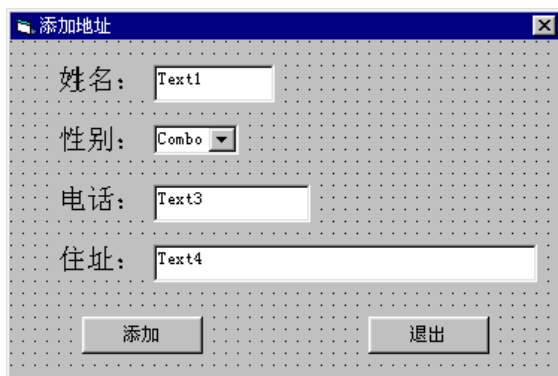


图 5-25 添加窗口界面

该窗体所包含的控件及其属性，具体见表 5-4。

表 5-4

控件及其属性值表



控件	名称	部分属性值	
CommandButton	OKButton	Caption	添加
	CancelButton	Caption	退出
Label	Label1	Caption	姓名：
	Label2	Caption	性别：
	Label3	Caption	电话：
	Label4	Caption	住址：
TextBox	Text1	Caption	Text1
		DataField	bf_Name
		DataMember	dbCommand
		DataSource	dbDataEnvironment
	Text3	Caption	Text3
		DataField	bf_HomeTel
		DataMember	dbCommand
		DataSource	dbDataEnvironment
	Text4	Caption	Text4
		DataField	bf_HomeAddress
		DataMember	dbCommand
		DataSource	dbDataEnvironment
ComboBox	Combo1	DataField	bf_Sex
		DataMember	dbCommand
		DataSource	dbDataEnvironment

(4) 为 Form 建立 Activate 事件和 UnLoad 事件，具体如下所示：

```
Private Sub Form_Activate()  
    mainForm.myRecordSet.AddNew  
End Sub  
  
Private Sub Form_Unload(Cancel As Integer)  
    If Text1.Text = "" Or Combo1.Text = "" Then  
        mainForm.myRecordSet.Delete  
        mainForm.myRecordSet.MoveFirst  
    End If  
    mainForm.myRecordSet.UpdateBatch (adAffectCurrent)  
End Sub
```

Form_Activate 内的代码的作用是当窗体被激活时自动调用记录集 myRecordSet 的 AddNew 方法，在表的最后插入一个空记录，以便我们输入新的记录。

而 Form_Unload 内的代码的作用是当窗体被撤消时，先判断 Text1 和 Combo1 的 Text 属性值是否为空，如果其中的任意一个为空，则调用记录集 myRecordSet 的 Delete 方法，以删除该空记录并把记录指针移动到第一条记录（因为数据字段的要求是姓名与性别不可为空），最后调用 UpdateBatch 方法更新记录。

(5) 为控件 OKButton 建立 Click 事件，具体代码如下所示：

```
Private Sub OKButton_Click()
```



```
If Text1.Text = "" Or Combo1.Text = "" Then  
    MsgBox "姓名与性别必须输入！", vbOKOnly  
Else  
    mainForm.myRecordSet.UpdateBatch (adAffectCurrent)  
    mainForm.myRecordSet.AddNew  
End If  
End Sub
```

这里首先判断 Text1 和 Combo1 的 Text 属性值是否为空,如果其中的任意一个为空则给出出错信息,因为数据字段的要求是姓名与性别不可为空。如果输入数据符合要求,则调用记录集 myRecordSet 的 UpdateBatch 方法,更新由 AddNew 方法产生的新记录,然后再调用 AddNew 方法生成新的记录,以便我们可以接着输入新的记录数据。

(6) 为控件 CancelButton 建立 Click 事件,具体代码如下所示:

```
Private Sub CancelButton_Click()  
    Unload Me  
End Sub
```

(7) 为 mainForm 窗体菜单的 mnAdd 菜单项建立 Click 事件,具体代码如下所示:

```
Private Sub mnAdd_Click()  
    addDialog.Show 1  
End Sub
```

(8) 按 F5 键运行程序,并单击“地址簿”菜单,选取其中的“添加”菜单项,如图 5-26 所示。

图 5-26 运行效果图

5.2.3 编辑好友资料


- (1) 打开 Address.vbp 工程。
- (2) 单击工具栏上的“”按钮,在“添加窗体”对话框中选取“对话框”。
- (3) 设计对话框窗体,使之如图 5-27 所示。



图 5-27 编辑窗体界面

该窗体所包含的控件及其属性，具体见表 5-5。

表 5-5 控件及其属性值表

控件	名称	部分属性值	
CommandButton	OKButton	Caption	确定
	CancelButton	Caption	退出
DataGrid	editDataGrid	DataSource	dbDataEnvironment
		DataMember	dbCommand
		AllowAddNew	False
		AllowArrows	True
		AllowDelete	False
		AllowUpdate	True

(4) 为 Form 建立 Activate 事件代码，具体如下所示：

```
Private Sub Form_Activate()  
    mainForm.myRecordSet.MoveFirst  
End Sub
```

(5) 为控件 OKButton 建立 Click 事件，具体代码如下所示：

```
Private Sub OKButton_Click()  
    mainForm.myRecordSet.UpdateBatch  
    mainForm.myRecordSet.MoveFirst  
End Sub
```

本段代码的作用是调用记录集 myRecordSet 的 UpdateBatch 方法来更新记录，即用修改后的内容来替换原记录，并调用记录集 myRecordSet 的 MoveFirst 方法，把当前记录设置为第一条记录。

(6) 控件 CancelButton 建立 Click 事件，具体代码如下所示：

```
Private Sub CancelButton_Click()  
    Unload Me  
End Sub
```

(7) 为 mainForm 窗体菜单的 mnEdit 菜单项建立 Click 事件，具体代码如下所示：



```
Private Sub mnEdit_Click()  
    editDialog.Show 1  
End Sub
```

(8) 按 F5 键运行程序，并单击“地址簿”菜单，选取其中的“编辑”菜单项，如图 5-28 所示。



图 5-28 运行效果图

5.2.4 整理好友资料

- (1) 打开 Address.vbp 工程。
- (2) 单击工具栏上的“ ”按钮，在“添加窗体”对话框中选取“对话框”。
- (3) 设计对话框窗体，使之如图 5-29 所示。



图 5-29 删除窗体界面

该窗体所包含的控件及其属性，具体见表 5-6。

表 5-6 控件及其属性值表

控件	名称	部分属性值	
CommandButton	delButton	Caption	删除
	OKButton	Caption	确定
	CancelButton	Caption	取消



续表

控件	名称	部分属性值	
DataGrid	delDataGrid	DataSource	dbDataEnvironment
		DataMember	dbCommand
		AllowAddNew	False
		AllowArrows	True
		AllowDelete	True
		AllowUpdate	False

(4) 为 Form 建立 Activate 事件代码，具体所下所示：

```
Private Sub Form_Activate()  
    mainForm.myRecordSet.MoveFirst  
End Sub
```

(5) 为控件 delButton 建立 Click 事件，具体代码如下所示：

```
Private Sub delButton_Click()  
    mainForm.myRecordSet.Delete  
End Sub
```

本段代码的作用是调用记录集 myRecordSet 的 Delete 方法来删除当前记录。由于提供记录集的数据环境的 dbCommand 命令，其 LockType 属性值为 4-adLockBatchOptimistic，即开放式批处理，因此没有调用记录集的 UpdateBatch 方法前，Delete 方法不会被真正实现。下面的 OKButton_Click 事件就是用来真正实现删除操作的。

(6) 为控件 OKButton 建立 Click 事件，具体代码如下所示：

```
Private Sub OKButton_Click()  
    mainForm.myRecordSet.UpdateBatch  
    Unload Me  
End Sub
```

(7) 为控件 CancelButton 建立 Click 事件，具体代码如下所示：

```
Private Sub CancelButton_Click()  
    mainForm.myRecordSet.CancelBatch  
    mainForm.myRecordSet.MoveFirst  
End Sub
```

本段代码的作用是调用记录集 myRecordSet 的 CancelBatch 方法来取消原来的批操作，这里当然指的是删除操作。因为 Delete 方法并没有被真正实现，所以可被取消。而调用记录集 myRecordSet 的 MoveFirst 方法的作用是把当前记录设置为第一条记录。

(8) 为 mainForm 窗体菜单的 mnDel 菜单项建立 Click 事件，具体代码如下所示：

```
Private Sub mnDel_Click()  
    delDialog.Show 1  
End Sub
```

(9) 按 F5 键运行程序，并单击“地址簿”菜单，选取其中的“删除”菜单项，如图



5-30 所示。



图 5-30 运行效果图



试一试：
建立一个通讯录。

5.3 小结

在这一章中，主要讲解数据的建立以及数据的浏览、编辑、追加和删除。

其中数据库的建立是以 Access 数据库的建立为蓝本的，讲述了数据库的建立等数据库基本操作。

在制作“电子同学录”的过程中则采用了 ADO (ActiveX Data Object) 的技术，建立了应用程序的数据使用者类与数据源 Access 数据的绑定。在设计 browDialog、editDialog 及 delDialog 窗体时，则使用了 DataGrid 控件来作为数据使用者，从而大大简化了应用程序的代码编写难度。而在 add Dialog 窗体中由于要处理空白数据的插入问题，所以没有使用 DataGrid 控件，而是采用 TextBox 控件和 ComboBox 控件来接收输入数据。



第 6 章 系统控制

Visual Basic 的功能很强，它几乎能编写 Windows 下的各种各样的应用程序。Windows 操作系统提供了一个巨大的函数库，它的 Kernel32.dll、User32.dll 和 Gdi32.dll 等动态链接库为我们提供了一系列快捷的、功能强大的函数，熟悉 Windows API 的应用可极大地提高编程的技巧与水平。



本章的主要内容如下：

- (1) 通过程序修改系统的属性；
- (2) 通过程序控制系统。

6.1 使用程序修改系统时间

俗话说得好，“一寸光阴一寸金”，时间是宝贵的，我们应该把它牢牢地掌握在自己的手里。

(1) 新建“标准 EXE”工程。

(2) 在工具箱上单击鼠标右键，弹出快捷菜单，选取“部件”菜单项，打开部件对话框。在部件对话框中找到“Microsoft Masked Edit Control 6.0 (SP3)”和“Microsoft Windows Common Controls-2 6.0 (SP4)”项，并把该控件集加入到工具箱中。

(3) 建立如图 6-1 所示的窗体。该窗体所包含的控件及其属性，具体见表 6-1。

(4) 在窗体上选中 MaskedTextBox1 控件，单击鼠标右键弹出快捷菜单，选取“属性”菜单项，将弹出如图 6-2 所示的“属性页”对话框。设置完各项属性后按“确定”按钮返回。

表 6-1 控件及其属性值

控件	名称	部分属性值	
CommandButton	Command1	Caption	调用
	Command2	Caption	设置
Timer	Timer1	Interval	1000
MaskedTextBox	MaskedTextBox1	见操作(4)	
MonthView	MonthView1	见操作(5)	



图 6-1 程序界面

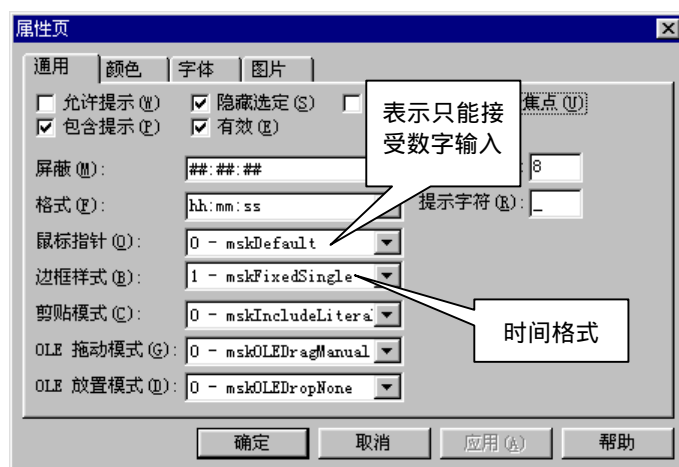


图 6-2 MaskedTextBox 属性窗口

(5) 在窗体上选中 MonthView1 控件，单击鼠标右键弹出快捷菜单，选取“属性”菜单项，将弹出如图 6-3 所示的“属性页”对话框。设置完各项属性后按“确定”按钮返回。

(6) 为 Form1 建立 Load 事件代码，具体如下所示：

```
Private Sub Form_Load()  
    Dim curTime As String  
    MonthView1.Value = Date  
    curTime = Time  
    If Hour(Time) < 10 Then  
        curTime = "0" + curTime  
    End If  
    MaskedTextBox1.Text = curTimeEnd Sub
```

(7) 分别为 Command1 和 Command2 建立 Click 事件，具体代码如下：

```
Private Sub Command1_Click()  
    Dim rtn As Long
```




```
    rtn = Shell("control.exe timedate.cpl", 0)
End Sub

Private Sub Command2_Click()
    Date = MonthView1.Value
    Time = MaskedTextBox1.Text
    Timer1.Enabled = True
    MonthView1.Day = Day(Date)
    MonthView1.Refresh
End Sub
```



图 6-3 MonthView 属性窗口

在 Command1 的 Click 事件中,我们使用了 Shell 函数来执行一个可执行文件,其格式为:

Shell(pathname[,windowstyle])

pathname 要执行的程序名,以及任何必需的参数或命令行变量,可能还包括目录或文件夹,以及驱动器

windowstyle 可选参数,表示在程序运行时窗口的样式

在本程序中,我们使用 Shell 函数执行了控制面板中的调整日期/时间程序。

(8) 为 MaskedTextBox1 控件建立 KeyDown 事件和 Validate 事件,具体代码如下:

```
Private Sub MaskedTextBox1_KeyDown(KeyCode As Integer, Shift_
As Integer)
    Timer1.Enabled = False
End Sub
```

```
Private Sub MaskedTextBox1_Validate(Cancel As Boolean)
    Dim s As String
    s = Left$(MaskedTextBox1.Text, 2)
    If Val(s) > 23 Or Val(s) < 0 Then
        Cancel = True
    End If
    s = Mid$(MaskedTextBox1.Text, 4, 2)
    If Val(s) < 0 Or Val(s) > 59 Then
```



```

Cancel = True
End If
s = Mid$(MaskedTextBox1.Text, 7, 2)
If Val(s) < 0 Or Val(s) > 59 Then
    Cancel = True
End If
If Cancel Then
    MsgBox "输入的时间错误!", vbOKOnly
End If
MaskedTextBox1.Text = Replace(MaskedTextBox1.Text, "_", "0")
End Sub

```

其中 KeyDown 事件的作用是当我们编辑 MaskedTextBox1 控件的 Text 属性时, 暂时停止 Timer 控件的运行, 以免对我们的编辑过程产生干扰。而 Validate 事件的作用是验证用户输入的数据是否符合时间值的格式。

(9) 为控件 MonthView1 建立 SelChange 事件, 具体代码如下所示:

```

Private Sub MonthView1_SelChange(ByVal StartDate As Date, _
    ByVal EndDate As Date, Cancel As Boolean)
    Command2.SetFocus
End Sub

```

(10) 为控件 Timer1 建立 Timer 事件, 具体代码如下所示:

```

Private Sub Timer1_Timer()
    Dim curTime As String
    curTime = Time
    If Hour(Time) < 10 Then
        curTime = "0" + curTime
    End If
    MaskedTextBox1.Text = curTime
End Sub

```

本事件的作用是每隔 1 秒钟刷新一次 MaskedTextBox1 控件的 Text 属性值。

(11) 按 F5 键运行程序, 效果如图 6-4 所示。



图 6-4 运行效果图

(12) 单击“调用”按钮, 将看到如图 6-5 所示的窗口。



图 6-5 运行效果图

6.2 改变任务栏的状态

Windows 的任务栏是用来显示和管理应用程序的，那么应用程序是否能反过来管理它呢？

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 6-6 所示的窗体。

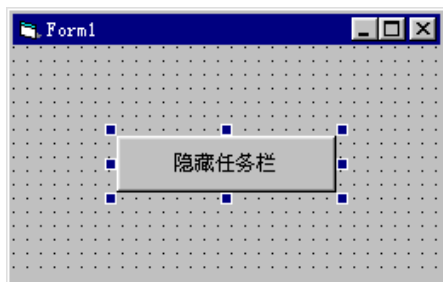


图 6-6 程序界面

本窗体只包含一个 CommandButton 控件 Command1，它的 Caption 属性为“隐藏任务栏”。

- (3) 在代码窗口中声明两个常数和两个函数，具体如下：

```
Const SWP_HIDEWINDOW = &H80
Const SWP_SHOWWINDOW = &H40
Private Declare Function FindWindow Lib "user32" Alias _
    "FindWindowA" (ByVal lpClassName As String, ByVal _
    lpWindowName As String) As Long
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
```



uFlags As Long) As Long

本代码段中我们声明了两个函数：FindWindow 和 SetWindowPos，它们都是由动态链接库 USER32.DLL 提供的。其中函数 FindWindow 用于查找 Windows 窗口，其格式如下所示：

rehWnd=FindWindow(lpClassName, lpWindowName)

lpClassName 窗口类的名称

lpWindowName 窗口名称或窗口标题

rehWnd 返回值，返回窗口句柄

函数 SetWindowPos 是用于设置窗口大小、位置等窗口属性的，其格式如下：

SetWindowPos(hWnd, hWndInsertAfter, X, Y, cx, cy, uFlags)

hWnd 窗口句柄

hWndInsertAfter 窗口在 Z 轴上的位置

(X, Y) 窗口的左上角坐标

cx 窗口的新宽度

cy 窗口的新高度

uFlags 窗口的风格属性，其取值可以是：SWP_HIDEWINDOW
或 SWP_SHOWWINDOW

(4) 为控件 Command1 建立 Click 事件，具体代码如下所示：

```
Private Sub Command1_Click()  
    Dim Taskbarhwn As Long  
    Taskbarhwn = FindWindow("Shell_traywnd", "")  
    If Command1.Caption = "隐藏任务栏" Then  
        Command1.Caption = "显示任务栏"  
        Call SetWindowPos(Taskbarhwn, 0, 0, 0, 0, 0, _  
SWP_HIDEWINDOW)  
    Else  
        Command1.Caption = "隐藏任务栏"  
        Call SetWindowPos(Taskbarhwn, 0, 0, 0, 0, 0, _  
SWP_SHOWWINDOW)  
    End If  
End Sub
```

在本代码段中先用 FindWindow 函数找到任务栏窗口，再利用 SetWindowPos 函数来设置任务栏为可见或隐藏。

(5) 按 F5 键运行程序。

6.3 自动关闭计算机

除了 Windows 自身，我们能否通过其他方法来控制系统的运行呢？下面就让我们来尝试一下。

(1) 新建“标准 EXE”工程。



(2) 建立如图 6-7 所示的窗体。

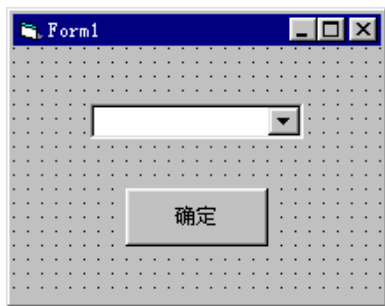


图 6-7 程序界面

在该窗体中包含了一个 CommandButton 控件 Command1 和一个 ComboBox 控件 Combo1，其中 Command1 的 Caption 属性值为“确定”，而 Combo1 的 Text 属性值设置为空，它的 ItemData 和 List 属性值都是一个 List，具体如图 6-8 和图 6-9 所示。

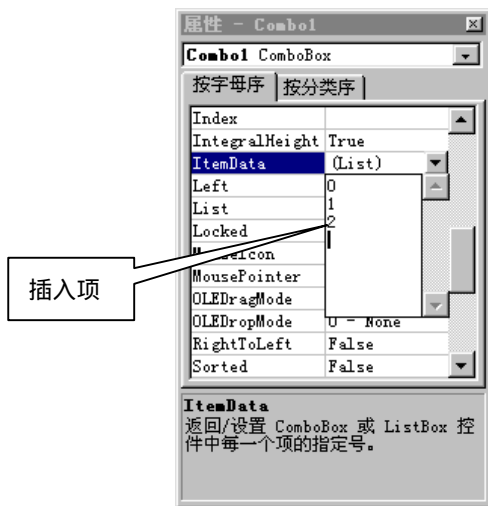


图 6-8 ItemData 属性



图 6-9 List 属性

其实 ComboBox 的 ItemData 属性与 List 属性是相互对应的，我们可以用 ItemData 值来表示被选择的 List 项。

(3) 在代码窗口中声明常量和函数，具体代码如下所示：

```
Const EWX_LOGOFF = &H0Const EWX_SHUTDOWN = &H1
Const EWX_REBOOT = &H2
Const EWX_POWEROFF = &H8
Private Declare Function ExitWindowsEx Lib "user32.dll" _
    (ByVal uFlags As Long, ByVal dwReserved As Long) As Long
```

其中常量 EWX_LOGOFF 表示注销，EWX_REBOOT 表示重新启动，EWX_SHUTDOWN 表示关闭系统，EWX_POWEROFF 则表示关闭电源。而函数 ExitWindowsEx 则是由动态链接库 USER.DLL 提供的，它的作用是退出 Windows 系统，其



格式如下所示：

ExitWindowsEx(uFlags, dwReserved)
uFlags 运行标志，可以是 EWX_LOGOFF 等
dwReserved 应用程序退出码，可以忽略

(4) 为 Form1 建立 Load 事件，代码如下所示：

```
Private Sub Form_Load()  
    Combo1.ListIndex = 0  
End Sub
```

(5) 为控件 Command1 建立 Click 事件，具体代码如下所示：

```
Private Sub Command1_Click()  
    Dim R As Integer, uFlags As Long, dwReserver As Integer  
    Select Case Combo1.ItemData(Combo1.ListIndex)  
        Case 0  
            uFlags = EWX_LOGOFF  
        Case 1  
            uFlags = EWX_REBOOT  
        Case 2  
            uFlags = EWX_SHUTDOWN + EWX_POWEROFF  
        Case Else  
            uFlags = EWX_LOGOFF  
    End Select  
    R = ExitWindowsEx(uFlags, dwReserver)  
End Sub
```

(6) 按 F5 键运行程序，如图 6-10 所示。

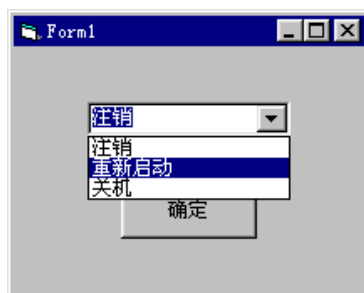


图 6-10 运行效果图

6.4 模拟“关闭程序”

对于“关闭程序”窗口，大家都应该印象深刻吧？当我们需要了解系统正在运行的程序或系统出问题时，只要同时按下 Ctrl、Alt 和 Del 这 3 个键，就能打开该窗口。那么我们能通过其他方法来实现这一效果呢？就让我们一试身手吧。

(1) 新建“标准 EXE”工程。



(2) 建立如图 6-11 所示的窗体。



图 6-11 程序界面

该窗体所包含的控件及其属性，具体见表 6-2。

表 6-2 控件及其属性值表

控件	名称	部分属性值	
CommandButton	Command1	Caption	结束任务
	Command2	Caption	关机
	Command3	Caption	取消
ListBox	List1	默认	

(3) 在代码窗口中声明常量和函数，具体代码如下所示：

```
Const WM_CLOSE = &H10
Const GW_HWNDNEXT = 2
Const GW_HWNDFIRST = 0
Const EWX_SHUTDOWN = &H1
Const EWX_POWEROFF = &H8
Dim WinTitle As String * 256
Dim hWnds(200) As Integer
Private Declare Function GetWindowText Lib "user32" Alias _
"GetWindowTextA" (ByVal hWnd As Long, ByVal lpString As String, _
ByVal nMaxCount As Long) As Long
Private Declare Function GetWindow Lib "user32" (ByVal _
hWnd As Long, ByVal uCmd As Long) As Long
Private Declare Function GetWindowTextLength Lib "user32" _
Alias "GetWindowTextLengthA" (ByVal hWnd As Long) As Long
Private Declare Function SendMessage Lib "user32" Alias _
"SendMessageA" (ByVal hWnd As Long, ByVal Msg As Long, ByVal _
wParam As Long, ByVal lParam As Long) As Long
```



```
Private Declare Function ExitWindowsEx Lib "user32.dll" _
    (ByVal uFlags As Long, ByVal dwReserved As Long) As Long
```

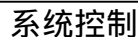
```
Sub GetRunPrg()
    Dim ReturnhWnd As Integer
    Dim Count As Integer
    Dim ReturnValue As Long
    Dim WinTitleLen As Long
    Dim Flag As Boolean
    Count = 0
    Form1.List1.Clear
    ReturnhWnd = GetWindow(Form1.hWnd, GW_HWNDFIRST)
    Do
        ReturnValue = GetWindowText(ReturnhWnd, WinTitle, _
256)
        If ReturnValue <> 0 Then
            WinTitleLen = GetWindowTextLength(ReturnhWnd)
            Flag = Left$(WinTitle, WinTitleLen) <> Form1.Caption _
And Left$(WinTitle, WinTitleLen) <> App.Title
            If Flag Then
                Form1.List1.AddItem WinTitle
                hWnds(Count) = ReturnhWnd
                Count = Count + 1
            End If
        End If
        ReturnhWnd = GetWindow(ReturnhWnd, GW_HWNDNEXT)
    Loop Until ReturnhWnd = 0
    Form1.List1.ListIndex = 0
End Sub
```

其中常量 WM_CLOSE 是一条消息，它表示一个窗口或一个应用程序将要终止的信号。常量 GW_HWNDFIRST 和 GW_HWNDNEXT 分别代表让 GetWindow 取得某一窗口的第一个兄弟窗口（同级窗口）和让 GetWindow 取得列表中在给定窗口之后的兄弟窗口（下一个同级窗口），而函数 GetWindowText、GetWindow、GetWindowTextLength 则都是由动态链接库 USER.DLL 提供的，其中函数 GetWindowText 的作用是获得指定窗口的标题，其格式如下所示：

```
reval=GetWindowText(hWnd, lpString, nMaxCount)
(hWnd)          要得到标题的窗口的句柄。
(lpString)       存放窗口标题的缓冲区。
(nMaxCount)      缓冲区能容纳的最大字节数。
(reval)          返回值，获得的标题的字节数（即长度）。
```

函数 GetWindow 的作用是获得与给定窗口有指定关系的窗口的句柄，其格式如下所示：

```
reval=GetWindow(hWnd, uCmd)
(hWnd)          窗口句柄。
(uCmd)          关系标志，可以取 GW_HWNDFIRST、GW_HWNDNEXT
                等标志。
(reval)          返回值，得到的与窗口 hWnd 有 uCmd 指定关系的窗
                口句柄。
```

hWnd	窗口句柄。
------	-------

reval	返回值，得到的给定窗口标题栏中文字的长度。
-------	-----------------------

而自定义过程 GetRunPrq 的作用是在列表框中填写当前正在运行的程序的标题。

(4) 为 Form1 建立 Load 事件，代码如下所示：

```
Private Sub Form_Load()
```

Call GetRunPrg

End Sub

(5) 为控件 Command1 建立 Click 事件，具体代码如下所示：

```
Private Sub Command1_Click()
```

Dim ReturnValue As Long

```
ReturnValue = SendMessage(hWnds(Form1.List1.ListIndex), WM_CLOSE, 0,
```

0)

```
If ReturnValue <> 0 Then
```

MsaBox "结束任务不成功！"

Else

MsgBox "成功结束任务！"

End If

Call GetRunPrq

End Sub

(6) 为控件 Command2 建立 Click 事件，具体代码如下所示：

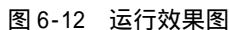
```
Private Sub Command2_Click()
```

Dim ReturnValue As Long, dwReserver As Long

```
ReturnValue = ExitWindowsEx(EWX_SHUTDOWN + EWX_POWEROFF,
```

dwReserver)

End Sub



(7) 为控件 Command3 建立 Click 事件，具体代码如下所示：



```
Private Sub Command3_Click()  
    End  
End Sub
```

(8) 按 F5 键运行程序，效果如图 6-12 所示。

6.5 小结

这一章主要讲解在应用程序中如何通过 Windows API 来调用系统功能，从而实现对系统的控制。

其中“使用程序修改系统时间”是用控件 MonthView 和控件 MaskedTextBox 共同实现的，在该应用程序中同时给出一种不同的方法，即调用控制面板来实现系统时间的设置。

“改变任务栏的状态”和“自动关闭计算机”则是以调用 FindWindow、SetWindowPos 及 ExitWindowsEx 等 Windows API 函数来实现的。

“模拟‘关闭程序’”中以 Windows API 函数 GetWindow 得到当前窗口的同级窗口，然后使用 GetWindowText 函数得到窗口标题，从而在列表框中显示该程序，最后再调用 GetWindow 得到下一个同级窗口，从而得到当前正在运行的所有程序。



第 7 章 网络应用

当今社会已经进入了信息时代，互联网飞速发展，电子商务等网络新兴产业方兴未艾，这使网络成为我们生活的一部分。Visual Basic 也加入这一潮流之中，为我们提供了一系列实现 Internet 或 Intranet 应用程序的途径。



本章的主要内容如下：

编写 Internet 应用程序，包括拨号上网、电子邮件和浏览器等。

7.1 拨号上网

虽然现在有宽带上网方式，但对于大多数的电脑用户来说，“拨号上网”还是上网的主要途径，因此拨号上网也成了许多 Internet 应用程序的一个重要组成部分。

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 7-1 所示的窗体。



图 7-1 程序界面

在本窗体中包含了两个 CommandButton 控件：Command1 和 Command2，它们的 Caption 属性分别是“调用”和“退出”。

- (3) 在代码窗口中声明常数和函数，具体代码如下所示：

```
Private Type OSVERSIONINFO
    dwOSVersionInfoSize As Long
    dwMajorVersion As Long
    dwMinorVersion As Long
    dwBuildNumber As Long
```



```

        dwPlatformId As Long
        szCSDVersion As String * 128
    End Type
    Const VER_PLATFORM_WIN32_NT = 2
    Private Declare Function GetVersionEx Lib "kernel32.dll" _
    Alias "GetVersionExA" (lpVersionInformation As OSVERSIONINFO) _
    As Long

```

在本段代码中我们声明了一种数据类型 OSVERSIONINFO, 该类型的变量是用于记录操作系统版本信息的, 其中 dwOSVersionInfoSize 是用于记录该类型变量的长度的, dwMajorVersion 是用于记录主版本号的, dwMinorVersion 是用于记录次版本号的, dwBuildNumber 是用于记录编译次数的, dwPlatformId 是用于记录平台信息的, 如果 dwPlatformId 值是 VER_PLATFORM_WIN32_NT, 则表示是 Windows NT 或 Windows 2000 平台。

函数 GetVersionEx 由动态链接库 KERNEL32.DLL 提供, 它的作用是得到操作系统的版本信息, 其格式如下:

```

reval=GetVersionEx(lpVersionInformation)
lpVersionInformation    OSVERSIONINFO 类型的变量, 用于存储得到的版本信息
reval                    返回值, 当函数成功执行时返回真

```

(4) 分别为 Command1 和 Command2 控件建立 Click 事件, 代码如下所示:

```

Private Sub Command1_Click()
    Dim rtn, returnval As Long
    Dim winver As OSVERSIONINFO
    winver.dwOSVersionInfoSize = 148
    returnval = GetVersionEx(winver)
    If winver.dwPlatformId = VER_PLATFORM_WIN32_NT Then
        rtn = Shell("rasphone.exe", 0)
    Else
        Dim lname As String
        lname = InputBox("请输入一个连接的名称:", "连接输入框", "")
        rtn = Shell("rundll32.exe rnaui.dll,RnaDial " + lname, 0)
    End If
End Sub

Private Sub Command2_Click()
    End
End Sub

```

之所以在 Command1 的 Click 事件中使用 GetVersionEx 函数, 是因为 Windows 98 操作系统与 Windows 2000 操作系统在拨号网络中使用了不同的技术来实现拨号过程。

(5) 按 F5 键运行程序, 并单击“调用”按钮。我们将看到如图 7-2 和图 7-3 所示的效果图。



图 7-2 Windows 2000 下运行效果图



图 7-3 Windows 98 下运行效果图

7.2 超级链接

对于网虫而言“超级链接”就是他的生命线，现在就让我们来完成“生命线”的联络。

- (1) 新建“标准 EXE”工程。
- (2) 建立如图 7-4 所示的窗体。

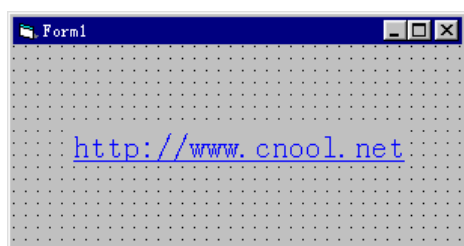


图 7-4 程序界面

该窗体所包含的控件及其属性，具体见表 7-1。

表 7-1 控件及其属性值表



控件	名称	部分属性值	
Label	Label1	Caption	http://www.cnool.net
		Font	宋体、规则、三号、下划线、蓝色
		MousePointer	99-Custom
		MouseIcon	Icon (C:\Windows\Cursor\Hand-m.cur)

(3) 在代码窗口中声明一个函数，具体代码如下所示：

```
Private Declare Function ShellExecute Lib "shell32.dll" _
Alias "ShellExecuteA" (ByVal hWnd As Long, ByVal lpOperation _
As String, ByVal lpFile As String, ByVal lpParameters As _
String, ByVal lpDirectory As String, ByVal nShowCmd As Long) _
As Long
```

在这里我们声明了函数 ShellExecute，它是由动态链接库提供的。它的作用是打开或打印一个文件，其格式如下：

```
reval=ShellExecute (hWnd, lpOperation, lpFile, lpParameters, lpDirectory,_
nShowCmd)
```

hWnd	窗口句柄
lpOperation	操作字符串
lpFile	被操作文件
lpParameters	当被操作文件为可执行文件时，程序的命令行参数
lpDirectory	默认的工作目录
nShowCmd	当被操作文件为可执行文件时，文件执行后所打开的主窗口的显示属性
reval	返回值

(4) 为控件 Label1 建立 Click 事件，具体代码如下所示：

```
Private Sub Label1_Click()
Dim HyperJump
Dim w
w = Label1.Caption
HyperJump = ShellExecute(0&, vbNullString, w,_
vbNullString, vbNullString, vbNormalFocus)
End Sub
```

(5) 按 F5 键运行程序，并单击 <http://www.cnool.net>。

7.3 自己的 Foxmail

电子邮件已经成为人们很常用的一种信息交流的手段，它的方便性、快捷性和高效性已经深入人心，而发送和接收电子邮件自然也就成了部分 Internet 应用程序的十分有用的



功能。

- (1) 新建“标准 EXE”工程。
- (2) 在工具箱上单击鼠标右键弹出快捷菜单,选取“部件”菜单项,打开部件对话框。在部件对话框中找到“Microsoft MAPI Control 6.0”和“Microsoft Tabbed Dialog Control 6.0 (SP5)”项,并把该控件集加入到工具箱中。
- (3) 建立如图 7-5 和图 7-6 所示的窗体。



图 7-5 程序界面



图 7-6 程序界面

本窗体比较特殊,一个窗体包含两个不同的界面。其实这是由 SSTab 控件所构造出来的,从总体的来说该窗体包含了一个 SSTab 控件 SSTab1、一个 MAPISession 控件



MAPISession1 和一个 MAPIMessages 控件 MAPIMessages1。其中 SSTab1 控件有两个页，分别是“接收邮件”和“发送邮件”页。该控件可以这样设置，先选中 SSTab1 控件，单击鼠标右键弹出快捷菜单，选取“属性”菜单项，将打开如图 6-17 所示的对话框。设定完成后按“确定”按钮返回。



图 7-7 SSTab1 的属性对话框

在“接收邮件”页中又包含了一些控件，这些控件及其属性具体见表 7-2。

表 7-2 控件及其属性值表

控件	名称	部分属性值	
Label	lblMsgCount	Caption	第 0 封邮件，总计 0 封邮件
	Label1	Caption	日期
	lblMsgDateReceived	Caption	Label1
	Label2	Caption	发件人
	lblMsgOrigDisplayName	Caption	Label2
	Label3	Caption	主题
	lblMsgSubject	Caption	Label3
	Label4	Caption	内容
CommandButton	cmdPrevious	Caption	上一封
	cmdNext	Caption	下一封
TextBox	txtMsgNoteText	默认	

在“发送邮件”页中也包含了一些控件，这些控件及其属性具体见表 7-3。

表 7-3 控件及其属性值表



控件	名称	部分属性值	
Label	Label7	Caption	收件人
	Label6	Caption	主题
	Label5	Caption	内容
TextBox	Text1	Caption	Text1
	Text2	Caption	Text2
	Text3	Caption	Text3
CommandButton	cmdSend	Caption	发送

(4) 在代码窗口中定义两个过程, 具体代码如下所示:

```
Public Sub FetchNewMail()  
    MAPIMessages1.FetchUnreadOnly = True  
    MAPIMessages1.Fetch  
End Sub  
Public Sub DisplayMessage()  
    lblMsgCount.Caption = "第 " & Ltrim(Str_  
        (MAPIMessages1.MsgIndex + 1)) & "封邮件, 总计"&  
        LTrim(Str(MAPIMessages1.MsgCount)) & " 封邮件"  
    lblMsgDateReceived.Caption =_  
        MAPIMessages1.MsgDateReceived  
    txtMsgNoteText.Text = MAPIMessages1.MsgNoteText  
    lblMsgOrigDisplayName.Caption =_  
        MAPIMessages1.MsgOrigDisplayName  
    lblMsgSubject.Caption = MAPIMessages1.MsgSubject  
End Sub
```

其中函数 Str 的作用是将数值转化为字符串, 而 Ltrim 函数的作用是去掉字符串左边的空白字符。

(5) 为 Form1 建立 Load 事件和 Unload 事件, 具体代码如下所示:

```
Private Sub Form_Load()  
    MAPISession1.SignOn  
    MAPIMessages1.SessionID = MAPISession1.SessionID  
    FetchNewMail  
    DisplayMessage  
End Sub  
  
Private Sub Form_Unload(Cancel As Integer)  
    MAPISession1.SignOff  
End Sub
```

(6) 分别为 cmdNext 控件、cmdPrevious 控件和 cmdSend 控件建立 Click 事件, 具体代码如下所示:

```
Private Sub cmdNext_Click()  
    If MAPIMessages1.MsgIndex < MAPIMessages1.MsgCount -_  
1 Then  
        MAPIMessages1.MsgIndex = MAPIMessages1.MsgIndex_  
+ 1  
        DisplayMessage
```



```

Else
    Beep
End If
End Sub

Private Sub cmdPrevious_Click()
    If MAPIMessages1.MsgIndex > 0 Then
        MAPIMessages1.MsgIndex = MAPIMessages1.MsgIndex_
- 1
        DisplayMessage
    Else
        Beep
    End If
End Sub

Private Sub cmdSend_Click()
    With MAPIMessages1
        .MsgIndex = -1
        .RecipAddress = Text1.Text
        .MsgSubject = Text2.Text
        .MsgNoteText = Text3.Text
        .SessionID = MAPISession1.SessionID
        .Send
    End With
    MsgBox "邮件发送完毕！", , "发送邮件"
End Sub

```

(7) 按 F5 键运行程序，分别单击“接收邮件”和“发送邮件”标签，将看到如图 7-8 和图 7-9 所示的运行结果。

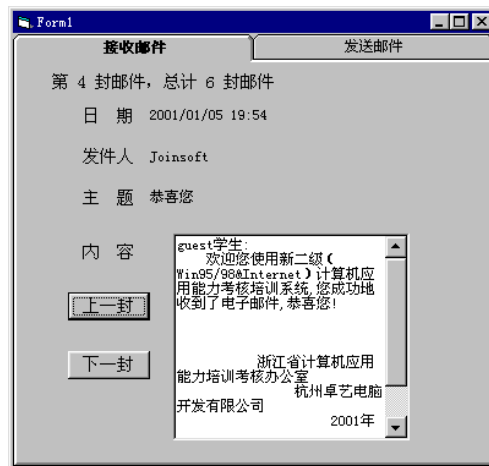


图 7-8 运行效果图



图 7-9 运行效果图

7.4 做个浏览器

随着信息技术的飞速发展，浏览器软件可以说已成为最受网民们欢迎的 Internet 软件之一。现在就让我们充分发扬 DIY 的精神，来装一台奔驰于信息高速公路上的小轿车。

(1) 新建“标准 EXE”工程。

(2) 在部件对话框中找到“Microsoft Common Dialog Control 6.0 (SP3)”、“Microsoft Internet Controls”及“Microsoft Common Controls 6.0(SP4)”项，并把这些控件集加入到工具箱中。

(3) 建立如图 7-10 所示的窗体，该窗体所包含的控件及其部分属性具体见表 7-4。

表 7-4 控件及其属性值表

控件	名称	部分属性值	
ComboBox	Combo1	默认	
CommonDialog	CommonDialog1	Filter	HTML 文件 *.html;*.htm 所有文件 *.*
Timer	Timer1	Interval	10
Label	Label1	Caption	地址：
ImageList	ImageList1	见操作(4)	
ToolBar	ToolBar1	见操作(5)	
StatusBar	StatusBar1	见操作(6)	
WebBrowser	WebBrowser1	默认	



图 7-10 程序界面

(4) 选中 ImageList1 控件，单击鼠标右键弹出快捷菜单，选取“属性”菜单项，将弹出如图 7-111 所示的属性页对话框。打开其中的“图像”选项卡，插入所需图片后按“确定”按钮返回。



图 7-11 属性页对话框

(5) 选中 Toolbar1 控件，单击鼠标右键弹出快捷菜单，选取“属性”菜单项，将打开如图 7-12 所示的属性页对话框。在“通用”选项卡中的“图像列表”、“禁用图像列表”和“热图像列表”3 个下拉列表框中选取“ImageList1”控件，使 Toolbar1 控件与 ImageList1 控件绑定。

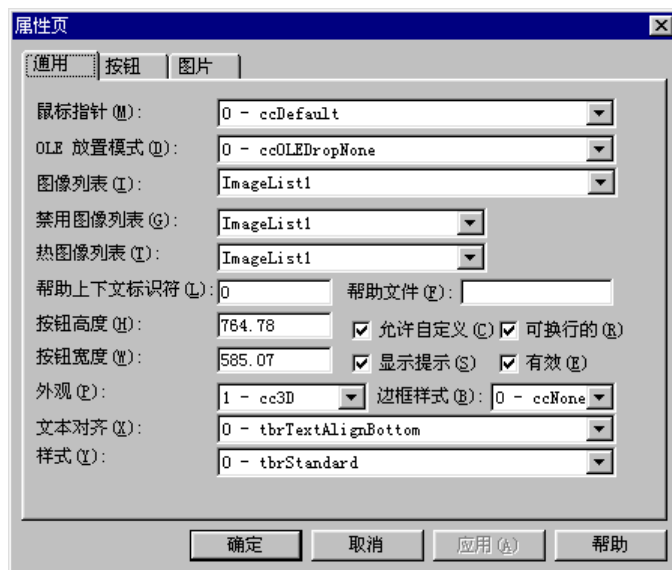


图 7-12 属性对话框

选取“按钮”页，将显示如图 7-13 所示的操作界面，按钮的设置属性见表 7-5。

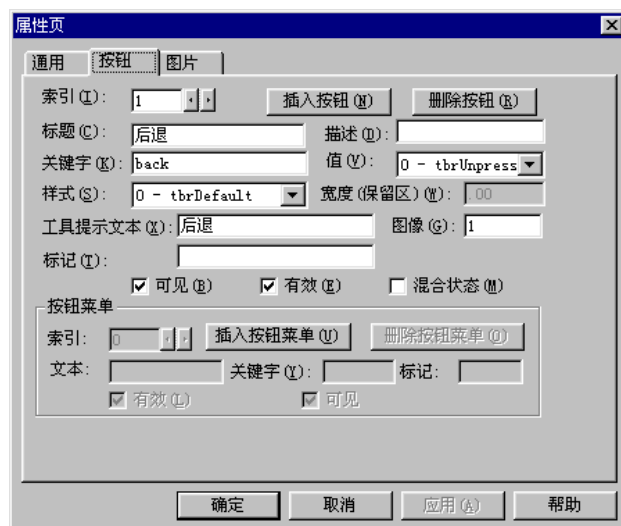


图 7-13 属性页窗口

表 7-5

按钮的属性表

按钮	索引	标题	关键字	式具提文本	图像	可见性	有效性
按钮 1	1	后退	Back	后退	1	可见	有效
按钮 2	2	前进	forward	前进	2	可见	有效
按钮 3	3	停止	Stop	停止	3	可见	有效
按钮 4	4	刷新	refresh	刷新	4	可见	有效
按钮 5	5	主页	home	主页	5	可见	有效



续表

按钮	索引	标题	关键字	式具提文本	图像	可见性	有效性
按钮 7	7	退出	Exit	退出	7	可见	有效
按钮 8	8	打开	Open	打开	8	可见	有效

(6) 选中 StatusBar1 控件，单击鼠标右键弹出快捷菜单，选取“属性”菜单项，将弹出如图 7-14 所示的属性页对话框。选取其中的“窗格”页，插入所需窗格后按“确定”按钮返回（本程序需 3 个窗格）。



图 7-14 属性页窗口

(7) 在代码窗口中声明两个模块级变量，具体代码如下：

```
Public StartingAddress As String
Dim flag As Boolean
```

(8) 为 Form1 建立 Load 事件及 ReSize 事件的代码，具体如下所示：

```
Private Sub Form_Load()
    Label1.Left = 60
    Label1.Top = Toolbar1.Height
    Combo1.Top = Toolbar1.Height
    Label1.Height = 315
    Combo1.Left = Label1.Left + Label1.Width
    Combo1.Width = Form1.ScaleWidth - Combo1.Left
    WebBrowser1.Width = Form1.ScaleWidth
    WebBrowser1.Top = Combo1.Top + 500
    WebBrowser1.Height = Form1.ScaleHeight - Combo1.Height - _
    StatusBar1.Height - 200
    WebBrowser1.Left = 0
    StatusBar1.Panels(1).Width = Form1.Width / 2
    StatusBar1.Panels(2).Width = Form1.Width / 4
    StatusBar1.Panels(3).Width = Form1.Width / 4
    StatusBar1.Panels(2).Text = "我的电脑"
```



```
StatusBar1.Panels(3).Text = Str$(Time)
On Error Resume Next
Me.Show
ToolBar1.Refresh
Open App.Path & "\mybrow.ini" For Input As #2
'打开上次的 WEB 地址
Input #2, StartingAddress
Close #2
If Len(StartingAddress) > 0 Then
    Combol.Text = StartingAddress
    Combol.AddItem StartingAddress
    WebBrowser1.Navigate StartingAddress '打开 Web 页
End If
End Sub

Private Sub Form_Resize()
    Label1.Left = 60
    Label1.Top = ToolBar1.Height
    Combol.Top = ToolBar1.Height
    Label1.Height = 315
    Combol.Left = Label1.Left + Label1.Width
    Combol.Width = Form1.ScaleWidth - Combol.Left
    WebBrowser1.Width = Form1.ScaleWidth
    WebBrowser1.Top = Combol.Top + 500
    WebBrowser1.Height = Form1.ScaleHeight - Combol.Top - _
Combol.Height - StatusBar1.Height - 200
    WebBrowser1.Left = 0

    StatusBar1.Panels(1).Width = Form1.Width / 2
    StatusBar1.Panels(2).Width = Form1.Width / 4
    StatusBar1.Panels(3).Width = Form1.Width / 4
End Sub
```

其中 On Error Resume Next 是一个启动错误处理程序的语句，说明当一个运行错误发生时，控件转到紧接着发生错误的语句之后的语句，并在此继续运行。

Open App.Path & "\mybrow.ini" For Input As #2 是用于打开该应用程序所在目录内的 mybrow.ini 文件。Open 语句的格式如下所示：

Open pathname For mode As [#]filenumber

pathname 要打开的文件

mode 打开文件的方式，具体有：Append、Binary、Input 或 Output

filenumber 文件号，用于唯一地表示已打开的文件

Input #2, StartingAddress 的作用是把从文件 mybrow.ini 中读入的行放入内存变量 StartingAddress 中。Input 语句的格式如下所示：

Input #filenumber, varlist

#filenumber 文件号，由 Open 语句提供

varlist 内存变量列表

Close #2 语句用于关闭已打开的文件号为 2 的文件，Len(StartingAddress)函数用于



返回字符串变量所包含的字符串的长度。

(9) 为控件 Webbrowser1 建立 DownloadComplete 事件和 NavigateComplete 事件，具体代码如下所示：

```
Private Sub webbrowser1_DownloadComplete()  
    On Error Resume Next  
    Me.Caption = WebBrowser1.LocationName '下载  
End Sub  
  
Private Sub webbrowser1_NavigateComplete(ByVal URL As_  
String)  
    '下载完成  
    Dim i As Integer  
    Dim bFound As Boolean  
    Me.Caption = WebBrowser1.LocationName  
    For i = 0 To Combol.ListCount - 1  
        If Combol.List(i) = WebBrowser1.LocationURL Then  
            bFound = True  
            Exit For  
        End If  
    Next i  
    flag = True  
    If bFound Then  
        Combol.RemoveItem i  
    End If  
    Combol.AddItem WebBrowser1.LocationURL, 0  
    Combol.ListIndex = 0  
    MousePointer = 0  
    WebBrowser1.Navigate Combol.Text  
    StatusBar1.Panels(1).Text = "当前页面:" & Combol.Text  
    flag = False  
End Sub
```

(10) 为控件 Combol 建立 Click 事件和 KeyPress 事件，具体代码如下所示：

```
Private Sub combol_Click()  
    If flag Then Exit Sub  
    Timer1.Enabled = True  
    MousePointer = 11  
    StatusBar1.Panels(1).Text = 正在连接 + Combol.Text +_  
"....."  
    WebBrowser1.Navigate Combol.Text  
End Sub  
Private Sub combol_KeyPress(KeyAscii As Integer)  
    On Error Resume Next  
    If KeyAscii = vbKeyReturn Then  
        If flag Then Exit Sub  
        Timer1.Enabled = True  
        MousePointer = 11  
        StatusBar1.Panels(1).Text = 正在连接 +
```




```
Combol.Text_ + "....."  
WebBrowser1.Navigate Combol.Text  
End If  
End Sub
```

(11) 为控件 Timer1 建立 Timer 事件，具体代码如下所示：

```
Private Sub Timer1_Timer()  
    StatusBar1.Panels(3).Text = Str$(Time)  
    If WebBrowser1.LocationURL <> "" Then  
        If WebBrowser1.Busy = False Then  
            Timer1.Enabled = False  
            Me.Caption = WebBrowser1.LocationName  
            StatusBar1.Panels(1).Text = "完成"  
        Else  
            Timer1.Enabled = True  
            Combol.Text = WebBrowser1.LocationURL  
            Me.Caption = WebBrowser1.LocationName  
            StatusBar1.Panels(1).Text = "正在下载: " +_  
WebBrowser1.LocationURL + "....."  
        End If  
    End If  
End Sub
```

(12) 为控件 Toolbar1 建立 ButtonClick 事件，具体代码如下所示：

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)  
    On Error Resume Next  
    Timer1.Enabled = True  
    Select Case Button.Key  
        Case "exit"  
            Open App.Path & "\my.ini" For Output As #2  
            '保存的 WEB 地址  
            Write #2, Combol.Text  
            Close #2  
            Unload Me  
        Case "back"  
            WebBrowser1.GoBack  
        Case "forward"  
            WebBrowser1.GoForward  
        Case "refresh"  
            WebBrowser1.Refresh  
        Case "home"  
            WebBrowser1.GoHome  
        Case "search"  
            WebBrowser1.GoSearch  
        Case "open"  
            On Error GoTo Handle  
            CommonDialog1.ShowOpen  
            Combol.Text = CommonDialog1.FileName  
            WebBrowser1.Navigate CommonDialog1.FileName  
            GoTo ExitCase  
        ExitCase:  
    End Select  
Handle:  
End Sub
```



```

        MsgBox "Can't Open :" + CommonDialog1.FileName, _
vbOKOnly
ExitCase:
Case "stop"
    Timer1.Enabled = False
    MousePointer = 0
    WebBrowser1.Stop
    Me.Caption = WebBrowser1.LocationName
End Select
End Sub

```

(13) 按 F5 键运行程序，效果如图 7-15 所示。



图 7-15 运行效果图

7.5 小结

在这一章中，主要讲解在应用程序中如何调用系统功能和其他应用程序（主要是 IE）的功能。

在“拨号上网”中，是以 Windows API 函数 GetVersionEx 来区分 Windows 版本后，分别用 Shell 函数调用不同程序来实现的，而“超级链接”则完全以 Windows API 函数 ShellExecute 来实现。

在“自己的 Foxmail”和“做个浏览器”中，则分别以控件 MAPIMessages、MAPISession 和 WebBrowser 来实现所有功能。



第 8 章 编程实例

游戏可以说是计算机软件中最有吸引力的软件，游戏的不断发展也推动了计算机软硬件的发展，从这一角度来说，游戏对计算机的发展有着不可忽略的作用。但在所有的软件开发中，游戏可以说是最难开发的，尤其是当我们用 Visual Basic 来开发游戏。



本章的主要内容如下：

- (1) 编写游戏“俄罗斯方块”。
- (2) 编写游戏“跳跳球”。
- (3) 编写游戏“打蟑螂”。

8.1 俄罗斯方块

游戏俄罗斯方块曾经风靡一时，可以说凡是玩过该游戏的人没有不为之着迷的，现在就让我们用 Visual Basic 来实现它。

首先我们来分析一下俄罗斯方块的图形结构。俄罗斯方块的图形并不复杂，有利于我们用 Visual Basic 来实现。它主要有“长条”、“Z 字形”、“反 Z 字形”、“7 字形”、“反 7 字形”、“T 字形”和“田字形”共 7 种图形及由这 7 种图形旋转而成的图形构成的，具体图形如图 8-4 所示。在编写程序的过程中，为了方便起见，我们就用这 19 种图形按规律地替换来模拟图形的旋转。

其次要说的是俄罗斯方块的操作方法。它的操作方法非常简单，只需用 4 个按钮分别负责图形向左右移动及向下移动。我们可以用“←”键控制图形向左移动，“→”键控制图形向右移动，“↓”键控制图形向下快速移动，“↻”键控制图形按顺时针方向旋转。因此我们必须把程序的主要代码放在 Form 的 KeyDown 事件中，以判断用户当前按下的键是否为“←”、“→”、“↓”及“↻”4 个键之一，从而来实现不同的操作，而且必须加入一个时钟控件，以便在其 Timer 事件中控制图形的自由下落。

(1) 新建“标准 EXE”工程。

(2) 打开菜单编辑器，如图 8-1 所示建立菜单。菜单项的具体属性见表 8-1。

表 8-1

菜单项属性值表

标题	名称	复选	有效性	可见性
文件	mnFile	否	有效	可见
打开	mnOpen	否	有效	可见

续表



标题	名称	复选	有效性	可见性
播放	mnPlay	否	无效	可见
退出	mnExit	否	有效	可见



图 8-1 菜单编辑器

(3) 建立如图 8-2 所示的窗体。

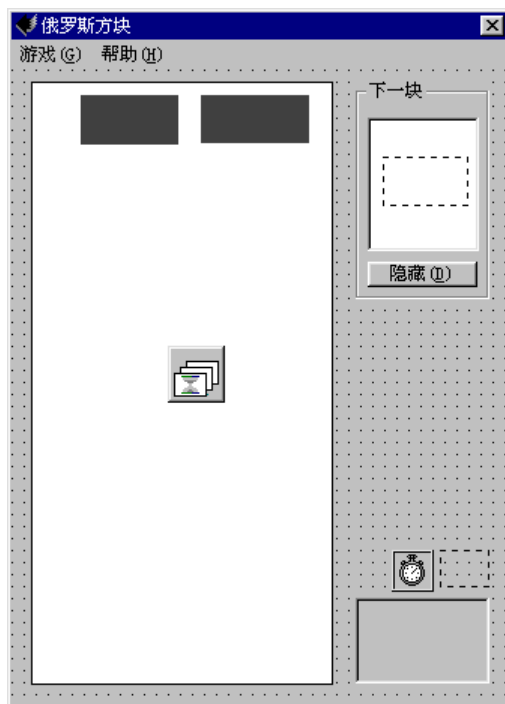


图 8-2 程序界面

该窗体所包含的控件及其属性，具体见表 8-2。



表 8-2

控件及其属性值表

控件	名称	部分属性值	
CommandButton	Comman1	Caption	隐藏
Frame	Frame1	Caption	下一块
Image	imgNext	默认	
	imgNowBackup	默认	
Label	Label1	BorderStyle	1
		ForeColor	&H00C00000&
PictureBox	picBackGround	默认	
	picNextBackGround	默认	
	picNow	默认	
	picTemp	默认	
Timer	tmrDrop	Enabled	False
		Interval	800
ImageList	imgPic	见操作(4)	

(4) 选中“imgPic”控件,单击鼠标右键,在弹出的快捷菜单中选取“属性”菜单项,打开如图 8-3 所示的对话框,插入如图 8-4 所示的 19 张图片后按“确定”按钮返回。

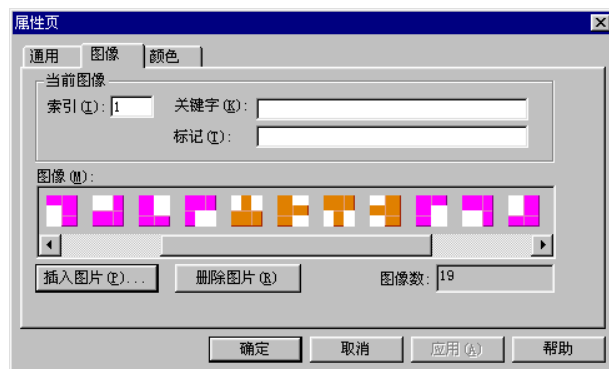


图 8-3 属性页窗口

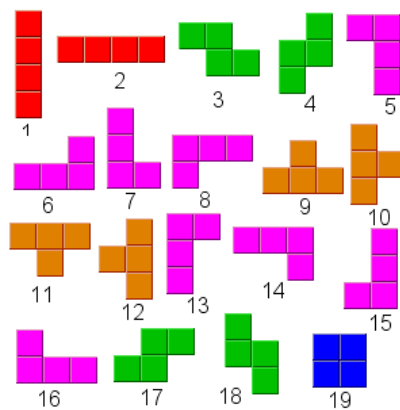


图 8-4 方块面



(5) 按下列代码建立源程序：

```
Dim CurX As Integer           '目前 x 坐标
Dim Total(10, 20) As Boolean   '总体坐标布局 10x20
Dim MinX As Integer, MaxX As Integer, MinY As Integer, _
MaxY As Integer
'一个方块的最小 x 坐标,最大 x 坐标,最小 y 坐标,最大 y 坐标
Dim Score As Integer

Private Type cXs
    cX As Integer 'x 坐标
    cY As Integer 'y 坐标
    cZ As Boolean '判断一个点下面是否是空的
End Type
Dim Xs(4) As cXs

Dim Adjust_Left As Integer '翻转后向左方调整的位置
Dim Adjust_Top As Integer '翻转后向上方调整的位置
```

BitBlt 函数作用：位操作位图，实现不规则的方块的动作

```
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
```

```
Dim Type_Now As Integer '目前方块的类型
Dim Type_Next As Integer '下个方块的类型
Dim intRotate As Integer '方块旋转的状态
```

```
Function Get_X_Value()
    If GetValue(1, 2) Then 'Get X Value
        If MaxX - MinX >= 2 Then
            If MaxX - CurX <= 1 Then
                Adjust_Left = MaxX - 2 - 1
            Else
                Adjust_Left = CurX - 1
            End If
            Get_X_Value = True
            Exit Function
        End If
    End If
    Get_X_Value = False
End Function
```

```
Function GetValue(nType As Integer, nWid As Integer)
    GetCoor
    On Error Resume Next
    Dim OKCount, EmptyCount As Integer
```



```
MinX = Xs(1).cX
MaxX = Xs(1).cX
MinY = Xs(1).cY
MaxY = Xs(1).cY
For i = 2 To 4
    If MinX > Xs(i).cX Then MinX = Xs(i).cX
    If MaxX < Xs(i).cX Then MaxX = Xs(i).cX
    If MinY > Xs(i).cY Then MinY = Xs(i).cY
    If MaxY < Xs(i).cY Then MaxY = Xs(i).cY
Next
For i = MinX To MaxX
    For j = MinY To MaxY
        If Total(i, j) Then
            GetValue = False
            Exit Function
        End If
    Next
Next

If nType = 0 Then 'Get Y Value
    EmptyCount = 0 'Get MinY
    OKCount = 0
    For i = MinY - 1 To MinY - (nWid - 1) Step -1

        For j = MinX To MaxX
            If Total(j, i) = False Then OKCount = OKCount_
+ 1
        Next
        If OKCount >= picNow.Width And OKCount >= _
picNow.Height Then
            EmptyCount = EmptyCount + 1
            OKCount = 0
        Else
            Exit For
        End If
    Next
    MinY = MinY - EmptyCount
    If MinY < 1 Then MinY = 1

    EmptyCount = 0 'GetMaxY
    OKCount = 0
    For i = MaxY + 1 To MaxY + nWid - 1
        For j = MinX To MaxX
            If Total(j, i) = False Then OKCount = OKCount_
+ 1
        Next
        If OKCount >= picNow.Width And OKCount >= _
picNow.Height Then
            EmptyCount = EmptyCount + 1
            OKCount = 0
```



```

        Else
            Exit For
        End If
    Next
    MaxY = MaxY + EmptyCount
    If MaxY > 20 Then MaxY = 20

Else 'Get X Value
    EmptyCount = 0 'Get MinX
    OKCount = 0
    For i = MinX - 1 To MinX - (nWid - 1) Step -1

        For j = MinY To MaxY
            If Total(i, j) = False Then OKCount = OKCount_
+ 1
        Next
        If OKCount >= picNow.Width And OKCount >= _
picNow.Height Then
            EmptyCount = EmptyCount + 1
            OKCount = 0
        Else
            Exit For
        End If
    Next
    MinX = MinX - EmptyCount
    If MinX < 1 Then MinX = 1

    EmptyCount = 0 'GetMaxX
    OKCount = 0
    For i = MaxX + 1 To MaxX + (nWid - 1)
        For j = MinY To MaxY
            If Total(i, j) = False Then OKCount = OKCount_
+ 1
        Next
        If OKCount >= picNow.Width And OKCount >= _
picNow.Height Then
            EmptyCount = EmptyCount + 1
            OKCount = 0
        Else
            Exit For
        End If
    Next
    MaxX = MaxX + EmptyCount
    If MaxX > 10 Then MaxX = 10
End If
GetValue = True
End Function

Function Get_Y_Value()

```




```
If GetValue(0, 2) Then 'Get Y Value
    If MaxY - MinY >= 2 Then
        If MaxY - (picNow.Top + 1) <= 1 Then
            Adjust_Top = MinY - 1
        Else
            Adjust_Top = picNow.Top
        End If
        Get_Y_Value = True
        Exit Function
    End If
End If
Get_Y_Value = False
End Function
```

过程 Global_Init 是用于做重新开始游戏而进行的全局初始化工作的

```
Sub Global_Init()
    '全局初始化
    picBackGround.Cls
    imgNext.Picture = LoadPicture("")
    picNow.Visible = False
    tmrDrop.Enabled = False
End Sub
```

过程 Init 是用于做每个方块被生成后的初始化工作的 Sub Init()

```
'每个方块的初始化过程
picNow.Visible = False
tmrDrop.Enabled = False
Type_Now = Type_Next
picNow.Picture = imgNext.Picture
imgNowBackup.Picture = picNow.Picture
Sel_Next
intRotate = 0
picNow.Left = 4
picNow.Top = 0
picNow.Visible = True
tmrDrop.Enabled = True
End Sub
```

过程 GetCoor 是用于获取一个方块的 4 个点的坐标，由于共有“长条”、“Z 字形”、“反 Z 字形”、“7 字形”、“反 7 字形”、“T 字形”和“田字形”7 种图形，所以本过程是一个由 7 个 Case 项组成的 Select Case 结构 Sub GetCoor()

```
'获取一个方块的 4 个点的坐标
For i = 1 To 4 'init
    Xs(i).cX = 0
    Xs(i).cY = 0
    Xs(i).cZ = False
Next
CurX = picNow.Left + 1
```



```

Select Case Type_Now
Case 1 '长条, 旋转后有横竖两种状态
    If intRotate Mod 2 = 1 Then
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 1
        Xs(1).cZ = True
        For i = 2 To 4
            Xs(i).cX = CurX + i - 1
            Xs(i).cY = picNow.Top + 1
            Xs(i).cZ = True
        Next
    Else
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 4
        Xs(1).cZ = True
        For i = 2 To 4
            Xs(i).cX = CurX
            Xs(i).cY = picNow.Top + i - 1
            Xs(i).cZ = False
        Next
    End If
Case 2 'Z字形, 旋转后有横竖两种状态
    If intRotate Mod 2 = 1 Then
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 3
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 2
        Xs(2).cZ = True
        For i = 3 To 4
            Xs(i).cX = CurX + i - 3
            Xs(i).cY = picNow.Top + 5 - i
            Xs(i).cZ = False
        Next
    Else
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 1
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 2
        Xs(2).cZ = True
        Xs(3).cX = CurX + 2
        Xs(3).cY = picNow.Top + 2
        Xs(3).cZ = True
        Xs(4).cX = CurX + 1
        Xs(4).cY = picNow.Top + 1
        Xs(4).cZ = False
    End If
Case 3 '7字形, 旋转后有4种状态

```



```
Select Case intRotate Mod 4
  Case 0
    Xs(1).cX = CurX
    Xs(1).cY = picNow.Top + 1
    Xs(1).cZ = True
    Xs(2).cX = CurX + 1
    Xs(2).cY = picNow.Top + 3
    Xs(2).cZ = True
    For i = 3 To 4
      Xs(i).cX = CurX + 1
      Xs(i).cY = picNow.Top + i - 2
      Xs(i).cZ = False
    Next
  Case 1
    Xs(1).cX = CurX
    Xs(1).cY = picNow.Top + 2
    Xs(1).cZ = True
    Xs(2).cX = CurX + 1
    Xs(2).cY = picNow.Top + 2
    Xs(2).cZ = True
    Xs(3).cX = CurX + 2
    Xs(3).cY = picNow.Top + 2
    Xs(3).cZ = True
    Xs(4).cX = CurX + 2
    Xs(4).cY = picNow.Top + 1
    Xs(4).cZ = False
  Case 2
    Xs(1).cX = CurX
    Xs(1).cY = picNow.Top + 3
    Xs(1).cZ = True
    Xs(2).cX = CurX + 1
    Xs(2).cY = picNow.Top + 3
    Xs(2).cZ = True
    For i = 3 To 4
      Xs(i).cX = CurX
      Xs(i).cY = picNow.Top + i - 2
      Xs(i).cZ = False
    Next
  Case 3
    Xs(1).cX = CurX
    Xs(1).cY = picNow.Top + 2
    Xs(1).cZ = True
    Xs(2).cX = CurX + 1
    Xs(2).cY = picNow.Top + 1
    Xs(2).cZ = True
    Xs(3).cX = CurX + 2
    Xs(3).cY = picNow.Top + 1
    Xs(3).cZ = True
    Xs(4).cX = CurX
    Xs(4).cY = picNow.Top + 1
```



```

        Xs(4).cZ = False
    End Select
Case 4 'T 字形，旋转后有 4 种状态
    Select Case intRotate Mod 4
    Case 0
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 2
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 2
        Xs(2).cZ = True
        Xs(3).cX = CurX + 2
        Xs(3).cY = picNow.Top + 2
        Xs(3).cZ = True
        Xs(4).cX = CurX + 1
        Xs(4).cY = picNow.Top + 1
        Xs(4).cZ = False
    Case 1
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 3
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 2
        Xs(2).cZ = True
        For i = 3 To 4
            Xs(i).cX = CurX
            Xs(i).cY = picNow.Top + i - 2
            Xs(i).cZ = False
        Next
    Case 2
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 1
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 2
        Xs(2).cZ = True
        Xs(3).cX = CurX + 2
        Xs(3).cY = picNow.Top + 1
        Xs(3).cZ = True
        Xs(4).cX = CurX + 1
        Xs(4).cY = picNow.Top + 1
        Xs(4).cZ = False
    Case 3
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 2
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 3
        Xs(2).cZ = True
    
```



```
For i = 3 To 4
    Xs(i).cX = CurX + 1
    Xs(i).cY = picNow.Top + i - 2
    Xs(i).cZ = False
Next
End Select
Case 5 '反 7 字形, 旋转后有 4 种状态
Select Case intRotate Mod 4
    Case 0
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 3
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 1
        Xs(2).cZ = True
        For i = 3 To 4
            Xs(i).cX = CurX
            Xs(i).cY = picNow.Top + i - 2
            Xs(i).cZ = False
        Next
    Case 1
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 1
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 1
        Xs(2).cZ = True
        Xs(3).cX = CurX + 2
        Xs(3).cY = picNow.Top + 2
        Xs(3).cZ = True
        Xs(4).cX = CurX + 2
        Xs(4).cY = picNow.Top + 1
        Xs(4).cZ = False
    Case 2
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 3
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 3
        Xs(2).cZ = True
        For i = 3 To 4
            Xs(i).cX = CurX + 1
            Xs(i).cY = picNow.Top + i - 2
            Xs(i).cZ = False
        Next
    Case 3
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 2
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
```



```

        Xs(2).cY = picNow.Top + 2
        Xs(2).cZ = True
        Xs(3).cX = CurX + 2
        Xs(3).cY = picNow.Top + 2
        Xs(3).cZ = True
        Xs(4).cX = CurX
        Xs(4).cY = picNow.Top + 1
        Xs(4).cZ = False
    End Select
Case 6 '反 Z 字形，旋转后有横竖两种状态
    If intRotate Mod 2 = 1 Then
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 2
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 3
        Xs(2).cZ = True
        For i = 3 To 4
            Xs(i).cX = CurX + i - 3
            Xs(i).cY = picNow.Top + i - 2
            Xs(i).cZ = False
        Next
    Else
        Xs(1).cX = CurX
        Xs(1).cY = picNow.Top + 2
        Xs(1).cZ = True
        Xs(2).cX = CurX + 1
        Xs(2).cY = picNow.Top + 2
        Xs(2).cZ = True
        Xs(3).cX = CurX + 2
        Xs(3).cY = picNow.Top + 1
        Xs(3).cZ = True
        Xs(4).cX = CurX + 1
        Xs(4).cY = picNow.Top + 1
        Xs(4).cZ = False
    End If
Case 7 '田字形，只有一种形状
    Xs(1).cX = CurX
    Xs(1).cY = picNow.Top + 2
    Xs(1).cZ = True
    Xs(2).cX = CurX + 1
    Xs(2).cY = picNow.Top + 2
    Xs(2).cZ = True
    For i = 3 To 4
        Xs(i).cX = CurX + i - 3
        Xs(i).cY = picNow.Top + 1
        Xs(i).cZ = False
    Next
End Select

```



End Sub

过程 Judge_Full 主要做了两件事。一、判断一行是否被填满，如填满则图像向下移动一行；二、图像的高度是否填满窗口高度，如填满则结束游戏给出所得分数 Sub Judge_Full()

```
'判断是否堆满
R_Value = picNow.Top + 1 'MinY
rx_value = picNow.Top + picNow.Height 'MaxY
For i = rx_value To R_Value Step -1
    If Total(1, i) And Total(2, i) And Total(3, i) And Total_
(4, i) And Total(5, i) And Total(6, i) And Total(7, i) And_
Total(8, i) And Total(9, i) And Total(10, i) Then
        '如果一行已经堆满，则将此行上面的图像全部向下移动一点
        k = BitBlt(picBackGround.hDC, 0, 20, 200, (i - 1)_
* 20, picBackGround.hDC, 0, 0, vbSrcCopy)
        Score = Score + 1
        For j = i To 1 Step -1
            For k = 1 To 10
                Total(k, j) = Total(k, j - 1)
            Next k
        Next j
        i = i + 1
    End If
Next i
'如果目前方块的顶点位置为 1，则表示全部堆满
If picNow.Top <= 1 Then
    msgString = "你玩完了！总共得了" + Trim(Str(10 * Score))
    msgString = msgString + "分。" + Chr(13) + "想再试试
身手吗？"
    reVal = MsgBox(msgString, 4 + 32)
    Score = 0
    Select Case reVal
        Case vbYes
            mnuGameNew_Click
        Case Else
            Global_Init
    End Select
End If
End Sub
```

函数 Judge_Rotate 用于判断方块能否翻转，由于“田字型”旋转后形状不变，所以本函数是一个由 6 个 Case 项组成的 Select Case 结构 Function Judge_Rotate()

```
Select Case Type_Now
Case 1 '长条，旋转后有两种状态
    If intRotate Mod 2 = 1 Then
        If GetValue(0, 4) Then 'Get Y Value
            If MaxY - MinY >= 3 Then
                Adjust_Top = MinY - 1
                Judge_Rotate = True
```



```

        Exit Function
    End If
End If
Judge_Rotate = False
Exit Function
Else
    If GetValue(1, 4) Then 'Get X Value
        If MaxX - MinX >= 3 Then
            If MaxX - CurX <= 2 Then
                Adjust_Left = MaxX - 3 - 1
            Else
                If CurX = MinX Then
                    Adjust_Left = CurX - 1
                Else
                    Adjust_Left = CurX - 1 - 1
                End If
            End If
            Judge_Rotate = True
            Exit Function
        End If
    End If
    Judge_Rotate = False
    Exit Function
End If
Case 2 'Z 字, 旋转后有两种状态
    If intRotate Mod 2 = 0 Then
        Judge_Rotate = Get_Y_Value
        Exit Function
    Else
        Judge_Rotate = Get_X_Value
        Exit Function
    End If
Case 3 '7 字, 旋转后有 4 种状态
    Select Case intRotate Mod 4
        Case 0
            Judge_Rotate = Get_X_Value
            Exit Function
        Case 1
            Judge_Rotate = Get_Y_Value
            Exit Function
        Case 2
            Judge_Rotate = Get_X_Value
            Exit Function
        Case 3
            Judge_Rotate = Get_Y_Value
            Exit Function
    End Select
Case 4 'T 字, 旋转后有 4 种状态
    Select Case intRotate Mod 4

```




```
Case 0
    Judge_Rotate = Get_Y_Value
    Exit Function
Case 1
    Judge_Rotate = Get_X_Value
    Exit Function
Case 2
    Judge_Rotate = Get_Y_Value
    Exit Function
Case 3
    Judge_Rotate = Get_X_Value
    Exit Function
End Select
Case 5 '反 7 字, 旋转后有 4 种状态
Select Case intRotate Mod 4
Case 0
    Judge_Rotate = Get_X_Value
    Exit Function
Case 1
    Judge_Rotate = Get_Y_Value
    Exit Function
Case 2
    Judge_Rotate = Get_X_Value
    Exit Function
Case 3
    Judge_Rotate = Get_Y_Value
    Exit Function
End Select
Case 6 '反 Z 字, 旋转后有两种状态
If intRotate Mod 2 = 0 Then
    Judge_Rotate = Get_Y_Value
    Exit Function
Else
    Judge_Rotate = Get_X_Value
    Exit Function
End If
End Select
End Function
```

函数 JudgeX_Left 用于判断方块能否向左移动

```
Function JudgeX_Left()
    GetCoor
    For i = 1 To 4
        On Error Resume Next
        If Xs(i).cY > 0 Then
            If Total(Xs(i).cX - 1, Xs(i).cY) Or Xs(i).cX_
= 0 Then
                JudgeX_Left = False
                Exit Function
            End If
```



```

        End If
    Next
    JudgeX_Left = True
End Function

```

函数 JudgeX_Reft 用于判断方块能否向右移动

```

Function JudgeX_Right()
    GetCoor
    For i = 1 To 4
        On Error Resume Next
        If Xs(i).cY > 0 Then
            If Total(Xs(i).cX + 1, Xs(i).cY) Or Xs(i).cX_
= 10 Then
                JudgeX_Right = False
                Exit Function
            End If
        End If
    Next
    JudgeX_Right = True
End Function

```

函数 JudgeY 用于判断方块能否向下移动

```

Sub JudgeY()
    GetCoor
    For i = 1 To 4
        If Xs(i).cZ Then
            On Error Resume Next
            If Xs(i).cY > 0 Then
                If Total(Xs(i).cX, Xs(i).cY + 1) Or Xs(i).cY_
= 20 Then
                    '如果不能移动, 将 4 点位置的坐标设置为 True, 并将图形固定下来
                    For j = 1 To 4
                        Total(Xs(j).cX, Xs(j).cY) = True
                    Next j
                    picBackGround.PaintPicture picNow.Picture, _
picNow.Left, picNow.Top, picNow.Width, picNow.Height, , , , _
vbSrcAnd
                    Judge_Full
                    If picNow.Visible Then Init
                    Exit Sub
                End If
            End If
        End If
    Next
End Sub

```

过程 Sel_Next 用于随机的从“长条”、“Z 字形”、“反 Z 字形”、“7 字形”、“反 7 字形”、“T 字形”和“田字形” 7 个方块中选择一个



```
Sub Sel_Next()  
    '随机从 7 个方块中选择一个  
    Randomize  
    Type_Next = Int((7 * Rnd) + 1)  
    Select Case Type_Next  
        Case 1  
            imgNext.Picture =_  
imgPic.ListImages.Item(1).Picture  
        Case 2  
            imgNext.Picture =_  
imgPic.ListImages.Item(3).Picture  
        Case 3  
            imgNext.Picture =_  
imgPic.ListImages.Item(5).Picture  
        Case 4  
            imgNext.Picture =_  
imgPic.ListImages.Item(9).Picture  
        Case 5  
            imgNext.Picture =_  
imgPic.ListImages.Item(13).Picture  
        Case 6  
            imgNext.Picture =_  
imgPic.ListImages.Item(17).Picture  
        Case 7  
            imgNext.Picture =_  
imgPic.ListImages.Item(19).Picture  
    End Select  
    imgNext.Move(picNextBackGround.Width - imgNext.Width) _  
    \ 2 - 30, (picNextBackGround.Height - imgNext.Height) \ 2 - _  
    30  
End Sub
```

'Command1 控件的 Click 事件用于控制下一个方块是否预示

```
Private Sub Commannd1_Click()  
    imgNext.Visible = Not (imgNext.Visible)  
    If imgNext.Visible Then  
        Commannd1.Caption = "隐藏"  
    Else  
        Commannd1.Caption = "显示"  
    End If  
End Sub
```

Form 的 KeyDown 事件控制了用户对 “ ”、“ ”、“ ”及 “ ” 4 个键的击键并根据用户的不同操作做出不同的反应。

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As_  
Integer)  
    '改变 Case 的 KeyCode 值就可以改变键盘控制按钮  
    Select Case KeyCode  
        Case vbKeyLeft '方块向左移动
```



```

        If picNow.Left - 1 >= 0 Then
            J_Value = JudgeX_Left
            If J_Value Then
                picNow.Picture = imgNowBackup.Picture
                r = BitBlt(picTemp.hDC, 0, 0, picNow.Width_
* 20, picNow.Height * 20, picBackGround.hDC, (picNow.Left -
1) * 20, picNow.Top * 20, vbSrcCopy)
                picNow.Left = picNow.Left - 1
                r = BitBlt(picNow.hDC, 0, 0, picNow.Width_
* 20, picNow.Height * 20, picTemp.hDC, 0, 0, vbSrcAnd)
            End If
        End If
        Case vbKeyRight '方块向右移动
            If picNow.Left + picNow.Width <_
picBackGround.ScaleWidth Then
                J_Value = JudgeX_Right
                If J_Value Then
                    picNow.Picture = imgNowBackup.Picture
                    r = BitBlt(picTemp.hDC, 0, 0, picNow.Width_
* 20, picNow.Height * 20, picBackGround.hDC, (picNow.Left +
1) * 20, picNow.Top * 20, vbSrcCopy)
                    picNow.Left = picNow.Left + 1
                    r = BitBlt(picNow.hDC, 0, 0, picNow.Width_
* 20, picNow.Height * 20, picTemp.hDC, 0, 0, vbSrcAnd)
                End If
            End If
        Case vbKeyDown '方块向下移动
            tmrDrop_Timer
        Case vbKeyUp '方块旋转
            If Judge_Rotate Then
                intRotate = intRotate + 1
                Select Case Type_Now
                    Case 1 '长条，旋转后有两种状态
                        If intRotate Mod 2 = 1 Then
                            picNow.Picture =_
imgPic.ListImages.Item(2).Picture
                            picNow.Top = picNow.Top + 3
                            picNow.Left = Adjust_Left
                        Else
                            picNow.Picture =_
imgPic.ListImages.Item(1).Picture
                            picNow.Top = Adjust_Top
                            picNow.Left = picNow.Left + 1
                        End If
                    Case 2 'Z字形，旋转后有两种状态
                        If intRotate Mod 2 = 1 Then
                            picNow.Picture =_
imgPic.ListImages.Item(4).Picture
                            picNow.Top = Adjust_Top

```



```
Else
    picNow.Picture =_
imgPic.ListImages.Item(3).Picture
    picNow.Top = picNow.Top + 1
    picNow.Left = Adjust_Left
End If
Case 3 '7 字形, 旋转后有 4 种状态
    Select Case intRotate Mod 4
        Case 0
            picNow.Picture =_
imgPic.ListImages.Item(5).Picture
            picNow.Top = Adjust_Top
        Case 1
            picNow.Picture =_
imgPic.ListImages.Item(6).Picture
            picNow.Top = picNow.Top + 1
            picNow.Left = Adjust_Left
        Case 2
            picNow.Picture =_
imgPic.ListImages.Item(7).Picture
            picNow.Top = Adjust_Top
        Case 3
            picNow.Picture =_
imgPic.ListImages.Item(8).Picture
            picNow.Top = picNow.Top + 1
            picNow.Left = Adjust_Left
    End Select
Case 4 'T 字形, 旋转后有 4 种状态
    Select Case intRotate Mod 4
        Case 0
            picNow.Picture =_
imgPic.ListImages.Item(9).Picture
            picNow.Top = picNow.Top + 1
            picNow.Left = Adjust_Left
        Case 1
            picNow.Picture =_
imgPic.ListImages.Item(10).Picture
            picNow.Top = Adjust_Top
        Case 2
            picNow.Picture =_
imgPic.ListImages.Item(11).Picture
            picNow.Top = picNow.Top + 1
            picNow.Left = Adjust_Left
        Case 3
            picNow.Picture =_
imgPic.ListImages.Item(12).Picture
            picNow.Top = Adjust_Top
    End Select
Case 5 '反 7 字形, 旋转后有 4 种状态
    Select Case intRotate Mod 4
```



```

        Case 0
            picNow.Picture =_
imgPic.ListImages.Item(13).Picture
            picNow.Top = Adjust_Top
        Case 1
            picNow.Picture =_
imgPic.ListImages.Item(14).Picture
            picNow.Top = picNow.Top + 1
            picNow.Left = Adjust_Left
        Case 2
            picNow.Picture =_
imgPic.ListImages.Item(15).Picture
            picNow.Top = Adjust_Top
        Case 3
            picNow.Picture =_
imgPic.ListImages.Item(16).Picture
            picNow.Top = picNow.Top + 1
            picNow.Left = Adjust_Left
    End Select
    Case 6 '反 Z 字形，旋转后有两种状态
        If intRotate Mod 2 = 1 Then
            picNow.Picture =_
imgPic.ListImages.Item(18).Picture
            picNow.Top = Adjust_Top
        Else
            picNow.Picture =_
imgPic.ListImages.Item(17).Picture
            picNow.Top = picNow.Top + 1
            picNow.Left = Adjust_Left
        End If
    End Select
    imgNowBackup.Picture = picNow.Picture
End If
End Select
End Sub

Private Sub Form_Load()
    Labell1.Caption = Space(14) + "积分：" + Space(6) + Str(10_
* Score)
End Sub

Private Sub mnuGameAbout_Click()
    MsgBox "俄罗斯方块 1.0 Demo", vbInformation
End Sub

Private Sub mnuGameExit_Click()
    End
End Sub

```



```
Private Sub mnuGameNew_Click()  
    '将 10×20 的坐标全部设置为空  
    For i = 1 To 10  
        For j = 0 To 20  
            Total(i, j) = False  
        Next j  
    Next i  
    CurX = 0  
    picBackground.Cls  
    '改变 tmrDrop 的 Interval 值即可改变游戏速度  
    tmrDrop.Interval = 1000  
    Sel_Next  
    Init  
End Sub
```

```
Private Sub mnuHelpKey_Click()  
    MsgBox "键盘控制方法：" + vbCrLf + "    控制方块向左移动；" _  
        + vbCrLf + "    控制方块向右移动；" _  
        + vbCrLf + "    控制方块向下快速移动；" _  
        + vbCrLf + "    控制方块的顺时针方向的翻转。", 64, " _  
    旋转俄罗斯 1.0 键盘操作帮助"  
End Sub
```

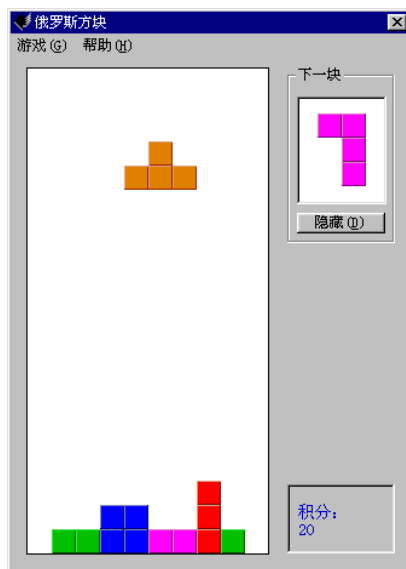


图 8-5 运行效果图

'定时器控件的 Timer 事件，用于控制方块的自由下落

```
Private Sub tmrDrop_Timer()  
    '方块下落  
    JudgeY  
    picNow.Picture = imgNowBackup.Picture
```



```

        r = BitBlt(picTemp.hDC, 0, 0, picNow.Width * 20, _
picNow.Height * 20, picBackGround.hDC, picNow.Left * 20, _
(picNow.Top + 1) * 20, vbSrcCopy)
        picNow.Top = picNow.Top + 1
        r = BitBlt(picNow.hDC, 0, 0, picNow.Width * 20, _
picNow.Height * 20, picTemp.hDC, 0, 0, vbSrcAnd)
        DoEvents
        If picNow.Top + picNow.Height > _
picBackGround.ScaleHeight Then Init
        Labell.Caption = Space(14) + "积分：" + Space(6) + Str(10_
* Score)
    End Sub

```

(6) 按 F5 键运行程序，效果图如图 8-5 所示。

8.2 跳跳球

在第 3 章中我们用一个不知疲倦的小球来模拟碰撞规律，当小球飞行至窗口边框时与边框发生弹性正碰，从而改变了飞行的方向。对该程序只要稍作修改，就能成为跳跳球（接球）游戏，其原理是这样的：当小球与窗口的左边框、右边框及上边框相碰时，还是服从原来的碰撞规律，而当小球碰到下边框时就结束游戏。如果小球到达下边框时碰到接球板，则小球与接球板发生弹性正碰。

由上面的描述我们可以确定，本程序的主要代码应当集中在定时器控件的 Timer 事件中，以该事件来控制小球的飞行，同时必须把控制接球板移动的代码放入到背景图片的 MouseMove 事件中。

(1) 新建“标准 EXE”工程。

(2) 打开菜单编辑器，如图 8-6 所示建立菜单，其属性值见表 8-3。



图 8-6 菜单编辑器



表 8-3

菜单项属性表

标题	名称	复选	有效性	可见性
游戏	mnGame	否	有效	可见
新游戏	mnNew	否	有效	可见
选项...	mnOption	否	无效	可见
结束	mnStop	否	有效	可见
-	mnBar	否	有效	可见
退出	mnExit	否	有效	可见
暂停	mnPause	否	有效	可见
关于	mnAbout	否	有效	可见

(3) 在工具箱上单击鼠标右键弹出快捷菜单, 选取“部件”菜单项, 打开部件对话框。在部件对话框中找到“Microsoft Windows Common Controls 6.0 (SP3)”, 并把这两个控件集加入到工具箱中。

(4) 建立如图 8-7 所示的窗体。



图 8-7 程序界面

该窗体所包含的控件及其属性, 具体见表 8-4。

表 8-4

控件及其属性值表

控件	名称	部分属性值	
PictureBox	Picture1	装入图片	
	Picture2	装入与 Picture1 相同的图片	
Timer	Timer1	Enabled	False
		Interval	50

(5) 按下列代码建立源程序:

```
Option Explicit
```

```
Private Const SWP_HIDEWINDOW = &H80
```

```
Private Const SWP_SHOWWINDOW = &H40
```



```

Private Const MERGEPAINT = &HBB0226
Private Const SRCAND = &H8800C6
Private Const SRCCOPY = &HCC0020
Private Const BallR = 10
Private Const BallD = 2 * BallR + 1
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
Private Declare Function FindWindow Lib "user32.dll" _
Alias "FindWindowA" (ByVal lpClassName As String, ByVal _
lpWindowName As String) As Long
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
Private CurX As Single, CurY As Single
Private OldX As Single, OldY As Single
Private Xmax As Single, Ymax As Single
Private MouseX As Single, oldMouseX As Single
Private isPause As Boolean, GameNotStart As Boolean, _
Taskbarhwn As Long
Public VelX As Single, VelY As Single
Public Speed As Integer, PaddleLen As Integer

Private Sub DrawBall()
    BitBlt Picture1.hDC,
        OldX - BallR, OldY - BallR, BallD, BallD,
        Picture2.hDC, OldX - BallR, OldY - BallR, SRCCOPY
    OldX = CurX
    OldY = CurY
    Picture1.Circle (CurX, CurY), BallR
    Picture1.Refresh
End Sub

```

上述代码中，定义的常量 SWP_HIDEWINDOW 和 SWP_SHOWWINDOW 用于控制窗口的“隐藏”或“显示”，而 BallR 及 BallD 则分别用于记录小球的半径和直径。其中变量 (CurX, CurY) 用于记录小球的当前坐标，(OldX, OldY) 用于记录小球的原坐标，Xmax 用于记录窗口横坐标的最大值，Ymax 用于记录窗口纵坐标的最大值，MouseX 和 oldMouseX 则分别用于记录光标的当前横坐标及原横坐标，isPause 用于记录游戏是否暂停，GameNotStart 用于记录游戏是否开始，Taskbarhwn 用于记录任务栏的窗口句柄，VelX 和 VelY 分别用于记录小球在横向和纵向的运动速度，公共变量 Speed 及 PaddleLen 分别用于记录设置对话框中设置的小球运动速度及接球板的长度。调用系统动态链接库的 3 个函数 BitBlt、FindWindow 和 SetWindowPos，我们已经在前面的章节中讲解过。而自定义过程 DrawBall 的作用，则是先清除在原坐标上的小球（以背景图片覆盖小球），然后在当前坐标处画出小球，从而制作出小球运动的效果。



(6) 为窗体 Form 建立 Load 事件的代码：

```
Private Sub Form_Load()  
    Speed = 10  
    PaddleLen = 40  
    GameNotStart = True  
    Width = (Width - ScaleWidth) + Picture1.Width  
    Height = (Height - ScaleHeight) + Picture1.Height  
    Xmax = Picture1.ScaleWidth - BallR  
    Ymax = Picture1.ScaleHeight - BallR  
    Randomize  
    CurX = Int((Xmax - BallR + 1) * Rnd + BallR)  
    CurY = Int((Ymax - BallR + 1) * Rnd + BallR)  
    OldX = CurX  
    OldY = CurY  
    VelX = Int((10 - 5 + 1) * Rnd + Speed)  
    Vely = Int((10 - 5 + 1) * Rnd + Speed)  
  
    Taskbarhwn = FindWindow("Shell_traywnd", "")  
    Call SetWindowPos(Taskbarhwn, 0, 0, 0, 0, 0, _  
SWP_HIDEWINDOW)  
    Move (Screen.Width - Width) / 2, (Screen.Height - _  
Height) / 2  
End Sub
```

在以上的代码中，首先用随机函数初始化了小球的出现位置及小球的运动速度，然后找到任务栏的窗口句柄并把任务栏隐藏起来

(7) 为窗体 Form 建立 UNload 事件的代码，用于显示任务栏：

```
Private Sub Form_Unload(Cancel As Integer)  
    Call SetWindowPos(Taskbarhwn, 0, 0, 0, 0, 0, _  
SWP_SHOWWINDOW)  
End Sub
```

(8) 为菜单项“mnNew”、“mnOption”、“mnStop”、“mnExit”、“mnPause”和“mnAbout”分别建立 Click 事件的代码：

```
Private Sub mnAbout_Click()  
    frmAbout.Show 1  
End Sub  
  
Private Sub mnExit_Click()  
    Call Form_Unload(0)  
End  
End Sub  
  
Private Sub mnNew_Click()  
    Timer1.Enabled = True  
    If (isPause) Then  
        isPause = False  
        mnPause.Caption = "暂停"  
    End If  
    mnNew.Enabled = False
```



```

mnOption.Enabled = False
mnStop.Enabled = True
GameNotStart = False
End Sub

Private Sub mnOption_Click()
    Dialog.Show 1
    VelX = Int((10 - 5 + 1) * Rnd + Speed)
    Vely = Int((10 - 5 + 1) * Rnd + Speed)
End Sub

Private Sub mnPause_Click()
    If Not GameNotStart Then
        If isPause Then
            isPause = False
            mnPause.Caption = "暂停"
            mnStop.Enabled = True
            Timer1.Enabled = True
        Else
            isPause = True
            mnPause.Caption = "继续"
            mnStop.Enabled = False
            Timer1.Enabled = False
        End If
    End If
End Sub

Private Sub mnStop_Click()
    Timer1.Enabled = False
    BitBlt Picture1.hDC, _
        OldX - BallR, OldY - BallR, BallD, BallD, _
        Picture2.hDC, OldX - BallR, OldY - BallR, SRCCOPY
    mnNew.Enabled = True
    mnOption.Enabled = True
    mnStop.Enabled = False
    GameNotStart = True
End Sub

```

(9) 为 Picture1 控件建立 MouseMove 事件代码：

```

Private Sub Picture1_MouseMove(Button As Integer, Shift_
As Integer, X As Single, Y As Single)
    BitBlt Picture1.hDC,
        oldMouseX - 10, Picture1.ScaleHeight - 10, _
PaddleLen + 2, 10,
        Picture2.hDC, oldMouseX - 10, Picture1.ScaleHeight_
- 10, SRCCOPY
    MouseX = X
    oldMouseX = X
    Picture1.Line (X - 10, Picture1.ScaleHeight - 10)-(X_
+ PaddleLen - 10,

```



```
Picture1.ScaleHeight), &HFFFF&, BF  
End Sub
```

在上面的代码中，先是调用 BitBlt 函数来清除光标原坐标处的接球板（用背景图片来覆盖接球板），然后调用 Picture1 的 Line 方法，在光标的原坐标处画出新的接球板，并把光标的当前坐标记录为原坐标。

（10）为 Timer1 控件建立 Timer 事件代码，在其中调用 DrawBall 过程来模拟小球的运动，并判断小球是否越过窗口的下边框，从而结束当前游戏：

```
Private Sub Timer1_Timer()  
    CurX = CurX + VelX  
    If (CurX > Xmax) Then  
        CurX = Xmax  
        VelX = -VelX  
    ElseIf (CurX < BallR) Then  
        CurX = BallR  
        VelX = -VelX  
    End If  
    CurY = CurY + Vely  
    If (CurY > Ymax - 11 - BallR) Then  
        If CurX + BallR >= MouseX - 10 And CurX - BallR <= _  
MouseX + PaddleLen - 10 Then  
            CurY = Ymax - 11  
            Vely = -Vely  
        Else  
            If (CurY > Ymax + 2 * BallD) Then  
                Dim returnval As Integer  
                returnval = MsgBox("重新开始游戏?", vbRetryCancel, _  
"游戏")  
                If returnval = vbRetry Then  
                    CurY = 0  
                    Vely = -Vely  
                Else  
                    Call Form_Unload(0)  
                    End  
                End If  
            End If  
        ElseIf (CurY < BallR) Then  
            CurY = BallR  
            Vely = -Vely  
        End If  
  
        DrawBall  
    End Sub
```

（11）为工程增加一个“对话框”（Dialog）窗体，具体如图 8-8 所示。

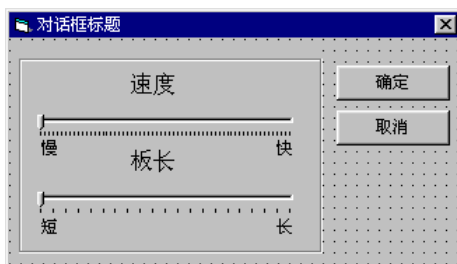


图 8-8 程序界面

该窗体所包含的控件及其属性，具体见表 8-5。

表 8-5 控件及其属性值表

控件	名称	部分属性值	
CommandButton	OKButton	Caption	确定
	CancelButton	Caption	取消
Slider	Slider1	LargeChange	2
		Max	60
		Min	0
		TickFrequency	1
	Slider2	LargeChange	2
		Max	200
		Min	0
		TickFrequency	10
Label	Label1	Caption	速度
	Label2	Caption	板长
	Label3	Caption	慢
	Label4	Caption	快
	Label5	Caption	短
	Label6	Caption	长
Frame	Frame1	Caption	(空)

(12) 按下列代码建立源程序：

```
Option Explicit

Private Sub OKButton_Click()
    Form1.Speed = Slider1.Value
    Form1.PaddleLen = Slider2.Value
    Unload Me
End Sub

Private Sub CancelButton_Click()
    Unload Me
End Sub
```



```
Private Sub Form_Load()  
    Slider1.Value = Form1.Speed  
    Slider2.Value = Form1.PaddleLen  
End Sub
```

(13) 为工程增加一个“关于对话框”(AboutDialog)窗体，具体如图 8-9 所示。

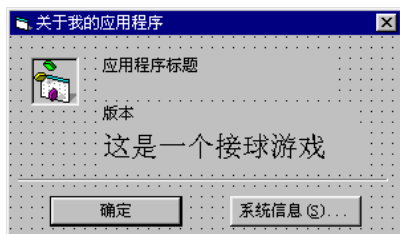


图 8-9 程序界面

对于“关于对话框”窗体，我们无需建立任何代码就可直接应用，因为在生成该对话框时代码已自动建立。

(14) 按 F5 键运行程序，效果如图 8-10 所示。



图 8-10 运行效果图

8.3 打蟑螂

首先我们分析一下打蟑螂游戏的图形结构。打蟑螂游戏的图形非常简单，只有“活蟑螂”和“死蟑螂”以及构造光标的两个“榔头”的两幅图片，这更加有利于我们用 Visual



Basic 来实现。

其次要介绍的是打蟑螂游戏的操作方法。这个游戏的操作方法非常简单，只要当窗口中出现蟑螂时，就在蟑螂上单击鼠标右键以打死蟑螂即可。由此我们可以在 Form 的 MouseDown 事件中判断是否打中蟑螂，同时在定时器控件的 Timer 事件中控制蟑螂的随机显示。

(1) 新建“标准 EXE”工程。

(2) 打开菜单编辑器，如图 8-11 所示建立菜单，其属性值见表 8-6。



图 8-11 菜单编辑器

表 8-6

菜单项属性表

标题	名称	复选	有效性	可见性
游戏	mnGame	否	有效	可见
新游戏	mnNew	否	有效	可见
选项...	mnOption	否	无效	可见
结束	mnStop	否	有效	可见
暂停	mnPause	否	有效	可见
关于	mnAbout	否	有效	可见

(3) 在工具箱上单击鼠标右键弹出快捷菜单，选取“部件”菜单项，打开部件对话框。在部件对话框中找到“Microsoft Windows Common Controls 6.0 (SP3)”，并把这两个控件集加入到工具箱中。

(4) 建立如图 8-12 所示的窗体。

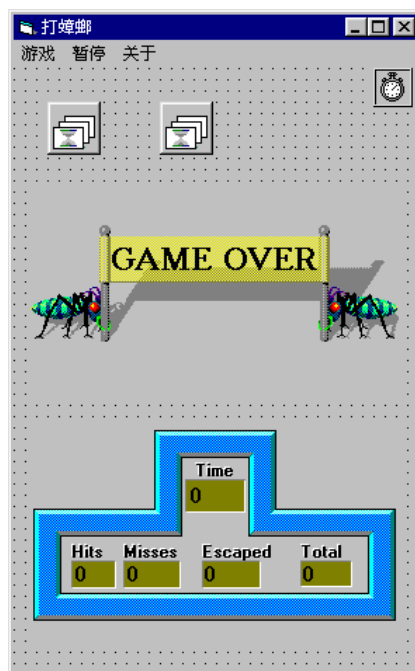


图 8-12 程序界面

该窗体所包含的控件及其属性，具体见表 8-7。

表 8-7 控件及其属性值表

控件	名称	部分属性值	
Label	TimeLabel	Caption	0
	HitsLabel	Caption	0
	MissLabel	Caption	0
	EscapedLabel	Caption	0
	ScoreLabel	Caption	0
PictureBox	Picture1	装入图片	
	picGameOver	装入图片	
Timer	Timer1	Enabled	False
		Interval	100
ImageList	picImageList	如图 8-13 所示	
	curImageList	如图 8-14 所示	

(5) 在代码窗口中声明常数和函数，具体如下：

```
Const GCL_HCURSOR = -12
Const MissedCritter = -1
Const MissedPoints = -2
Const HitPoints = 5
Const CritterSize = 72
Private Type Hole
    Time As Integer
```



```

        Dead As Boolean
    End Type
    Dim HoleInfo(0 To 4) As Hole
    Private Type Point
        x As Long
        y As Long
    End Type
    Dim Holes(0 To 4) As Point
    Dim IsGameOver As Boolean, IsPause As Boolean
    Dim Hits As Integer, Miss As Integer, Escaped As Integer
    Dim nWidth As Long, nHeight As Long
    Public GameTime As Integer, LiveTime As Integer, Score_
As Integer, Frequence As Integer
    Private Declare Function LoadCursorFromFile Lib "user32" _
Alias "LoadCursorFromFileA" (ByVal lpFileName As String) As _
Long
    Private Declare Function DestroyCursor Lib "user32" _
(ByVal hCursor As Long) As Long
    Private Declare Function GetClassLong Lib "user32" Alias _
"GetClassLongA" (ByVal hWnd As Long, ByVal nIndex As Long) _
As Long
    Private Declare Function SetClassLong Lib "user32" Alias _
"SetClassLongA" (ByVal hWnd As Long, ByVal nIndex As Long, _
ByVal dwNewLong As Long) As Long

Sub WriteScore()
    TimeLabel.Caption = GameTime - Timer1.Tag
    HitsLabel.Caption = Hits
    MissLabel.Caption = Miss
    EscapedLabel.Caption = Escaped
    ScoreLabel.Caption = Score
End Sub

```

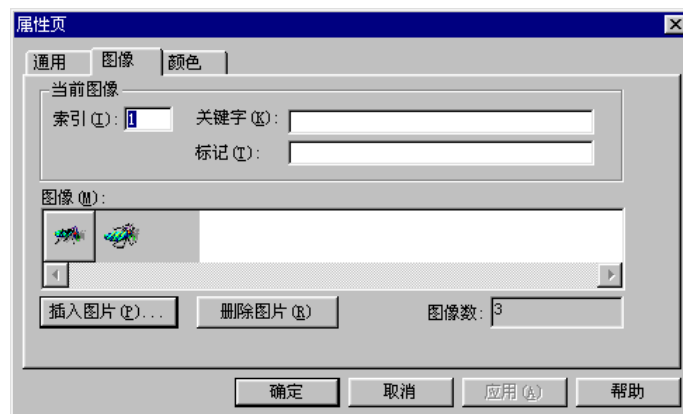


图 8-13 属性页对话框



图 8-14 属性页对话框

在上述代码中，数组 HoleInfo 记录了 5 个蟑螂的生存情况，该数组是 Hole 类型的。Hole 类型是一个用户自定义类型，其成员 Time 记录了蟑螂出现的时间，而成员 Dead 记录了蟑螂是否已死亡。数组 Holes 记录了 5 个蟑螂出现的位置。变量 isGameOver 和 isPause 分别记录了游戏是否结束和游戏是否暂停，其中的自定义过程 WriteScore 用于显示游戏的运行时间、打死的蟑螂数、打空的次数、漏打的次数和获得的分数。

(6) 为 Form 建立 Load 事件代码：

```
Private Sub Form_Load()  
    Me.MouseIcon =_  
    curImageList.ListImages.Item(1).Picture
```

```
Randomize
```

```
IsGameOver = True  
IsPause = False  
LiveTime = 10  
Frequency = 20  
GameTime = 150  
Hits = 0  
Miss = 0  
Escaped = 0  
Score = 0
```

```
Holes(0).x = 120  
Holes(0).y = 120  
Holes(1).x = 3120  
Holes(1).y = 120  
Holes(2).x = 120  
Holes(2).y = 2640  
Holes(3).x = 3120  
Holes(3).y = 2640  
Holes(4).x = 1560  
Holes(4).y = 1320
```



```
nWidth = 1140
'picImageList.ListImages.Item(1).Picture.Width
nHeight = 1140
'picImageList.ListImages.Item(1).Picture.Height

End Sub
```

在上述代码中设置了光标的形状,进行了游戏运行时间等变量的初始化,同时对 Holes 数组进行了初始化,确定了 5 个蟑螂的显示位置(在游戏过程中随机出现)。

(7) 为 Form 建立 MouseDown 及 MouseUp 事件代码,以确定是否打到蟑螂以及进行图片刷新和记分等工作。

```
Private Sub Form_MouseDown(Button As Integer, Shift As _
Integer, x As Single, y As Single)
    Me.MouseIcon = _
    curImageList.ListImages.Item(2).Picture

    If Not (IsGameOver Or IsPause) Then

        Dim hit As Boolean, i As Integer
        hit = False
        For i = 0 To 4
            If (Not HoleInfo(i).Dead) And (HoleInfo(i).Time <>_
0) Then
                If (x > Holes(i).x) And (x < (Holes(i).x + _
picImageList.ListImages.Item(1).Picture.Width)) And _
(y > Holes(i).y) And (y < (Holes(i).y + _
picImageList.ListImages.Item(1).Picture.Height)) Then
                    Score = Score + HitPoints
                    HoleInfo(i).Dead = True
                    HoleInfo(i).Time = Timer1.Tag + 2 * LiveTime
                    Hits = Hits + 1
                    hit = True
                    Me.Line (Holes(i).x, Holes(i).y)-(Holes(i).x+_
nWidth, Holes(i).y + nHeight), Me.BackColor, BF
                    picImageList.ListImages.Item(2).Draw Me.hDC,_
Holes(i).x, Holes(i).y, imlTransparent
                End If
            End If
        Next i
        If Not hit Then
            Score = Score + MissedPoints
            Miss = Miss + 1
        End If
        Call WriteScore
    End If
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As _
```



```
Integer, x As Single, y As Single)
    Me.MouseIcon =_
    curImageList.ListImages.Item(1).Picture
End Sub
```

(8) 为菜单项 “mnNew”、“mnOption”、“mnStop”、“mnPause” 和 “mnAbout” 分别建立 Click 事件的代码：

```
Private Sub mnAbout_Click()
    frmAbout.Show 1
End Sub

Private Sub mnNew_Click()
    Timer1.Enabled = True
    Timer1.Tag = 0
    Score = 0
    Hits = 0
    Miss = 0
    Escaped = 0
    If (IsPause) Then
        IsPause = False
        mnPause.Caption = "暂停"
    End If
    picGameOver.Visible = False
    IsGameOver = False
    mnNew.Enabled = False
    mnOption.Enabled = False
    mnStop.Enabled = True
End Sub

Private Sub mnOption_Click()
    Dialog.Show
End Sub

Private Sub mnPause_Click()
    If Not IsGameOver Then
        If IsPause Then
            IsPause = False
            mnPause.Caption = "暂停"
            mnStop.Enabled = True
            Timer1.Enabled = True
        Else
            IsPause = True
            mnPause.Caption = "继续"
            mnStop.Enabled = False
            Timer1.Enabled = False
        End If
    End If
End Sub

Private Sub mnStop_Click()
```



```

Timer1.Enabled = False
IsPause = False
picGameOver.Visible = True
IsGameOver = True
Timer1.Tag = GameTime
mnNew.Enabled = True
mnOption.Enabled = True
mnStop.Enabled = False
For i = 0 To 4
    If (HoleInfo(i).Time <> 0) Then
        Me.Line (Holes(i).x, Holes(i).y)-(Holes(i).x +_
nWidth, Holes(i).y + nHeight), Me.BackColor, BF
    End If
Next i
End Sub

```

(9) 为 Timer1 控件建立 Timer 事件的代码：

```

Private Sub Timer1_Timer()
    Timer1.Tag = Timer1.Tag + 1
    Dim i As Integer
    i = Int((Frequence + 1) * Rnd)

    If i < 5 Then
        If HoleInfo(i).Time = 0 Then
            HoleInfo(i).Time = Timer1.Tag + LiveTime
            HoleInfo(i).Dead = False
            picImageList.ListImages.Item(1).Draw Me.hDC,_
Holes(i).x, Holes(i).y, imlTransparent
        End If
    End If
    For i = 0 To 4
        If (Timer1.Tag > HoleInfo(i).Time) And_
(HoleInfo(i).Time <> 0) Then
            HoleInfo(i).Time = 0
            If (Not HoleInfo(i).Dead) Then
                Score = Score + MissedCitter
                Escaped = Escaped + 1
            End If
            Me.Line (Holes(i).x, Holes(i).y)-(Holes(i).x +_
nWidth, Holes(i).y + nHeight), Me.BackColor, BF
        End If
    Next i
    Call WriteScore
    If Timer1.Tag >= GameTime Then
        Call mnStop_Click
    End If
End Sub

```

上述代码进行了图片清除及统计显示分数等工作。

(10) 为工程增加一个“对话框”窗体，具体如图 8-15 所示。



图 8-15 程序界面

该窗体所包含的控件及其属性，具体见表 8-8。

表 8-8 控件及其属性值表

控件	名称	部分属性值	
CommandButton	OKButton	Caption	确定
	CancelButton	Caption	取消
Slider	Slider1	LargeChange	2
		Max	60
		Min	0
		TickFrequency	1
	Slider2	LargeChange	2
		Max	200
		Min	0
		TickFrequency	10
Label	Label1	Caption	速度
	Label2	Caption	数量
	Label3	Caption	慢
	Label4	Caption	快
	Label5	Caption	低
	Label6	Caption	高
	Label7	Caption	游戏时间
TextBox	Text1	Text	0
UpDown	UpDown1	如图 8-16 和图 8-17 所示	
Frame	Frame1	Caption	(空)



图 8-16 属性页对话框



图 8-17 属性页对话框

(11) 按下列代码建立源程序：

```
Option Explicit

Private Sub CancelButton_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    Slider1.Value = Slider1.Max + 1 - Form1.LiveTime
    Slider2.Value = Form1.Frequence
    Text1.Text = Form1.GameTime
End Sub

Private Sub OKButton_Click()
    Form1.LiveTime = Slider1.Max + 1 - Slider1.Value
    Form1.Frequence = Slider2.Value
    Form1.GameTime = Text1.Text
    Unload Me
End Sub
```




```
End Sub
```

(12) 为工程增加一个“关于对话框”(AboutDialog)窗体,具体如图 8-18 所示。

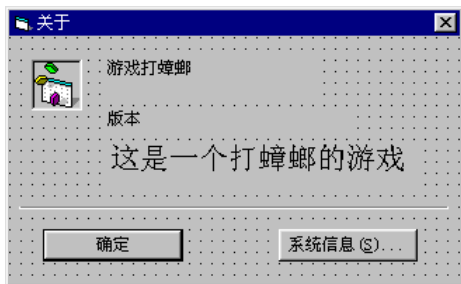


图 8-18 程序界面

(13) 按 F5 键运行程序,其效果如图 8-19 所示。

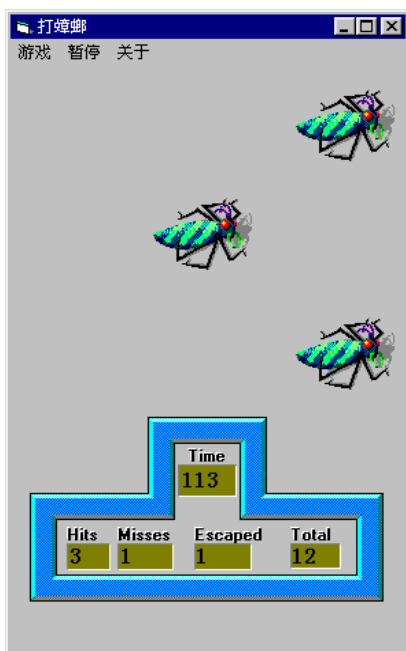


图 8-19 运行效果图

8.4 小结

在这一章中,讲解了“俄罗斯方块”、“跳跳球”和“打蟑螂”3个游戏的编程。

其中游戏“俄罗斯方块”中,主要是通过时钟对象的时钟事件来实现图片对象的平移,以及根据窗体的 KeyDown 事件来实现对应不同的光标产生不同动作,从而选择不同的图片。



游戏“跳跳球”是利用在“不知疲倦的小球”这一例子改变而来的，在时钟事件中加入了判别小球是否在下边界处碰到“接板”来决定游戏是否继续。

游戏“打蟑螂”是以时钟事件和随机函数来决定几个图片显示与否，再以窗体的MouseDown 来确定光标是否停在图片之上，即蟑螂是否被打到。