小游戏开发时尚编程百例

网冠科技　编著

机械工业出版社

本书是一本以实例的方式讲解如何使用开发工具、制作小游戏的书籍。本书使用了现今最流行的开发工具，如 Visual Basic 6.0、Delphi 6.0、C++ Builder 6.0 等，制作出几十个小游戏，包括智力游戏、经典游戏、对抗游戏、控制类游戏和趣味游戏。

通过本书的学习，读者将会掌握基本的游戏制作技巧，能够独立编写出一些小游戏。

本书适合于编程爱好者和广大游戏制作人员。

图书在版编目（CIP）数据

小游戏开发时尚编程百例 / 网冠科技编著.
–北京：机械工业出版社，2002.6
（时尚百例丛书）
ISBN 7-111-10409-9

Ⅰ. 小 … Ⅱ. 网… Ⅲ. 游戏–应用程序–程序设计 Ⅳ. G899

中国版本图书馆 CIP 数据核字（2002）第 037695 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策 划：胡毓坚
责任编辑：王 虹
责任印制：

印刷 · 新华书店北京发行所发行

2002 年 6 月第 1 版 · 第 1 次印刷
787mm × 1092mm 1/16 · 19.5 印张 · 482 千字
0001-5000 册
定价：35.00 元（含 1CD）

前　言

《小游戏开发时尚编程百例》是“时尚百例丛书”中的一本。

本书使用了现今最流行的开发工具，如 Visual Basic、Delphi、C++ Builder 等工具。这些工具都是一些软件公司常用的开发平台，一些大型的游戏制作公司也是以这些工具为开发平台的。本书还讲述了如何使用 DirectX、OpenGL 等进行图形图像编程。

本书以 100 个实例的形式向读者讲解如何制作小游戏。其中用到了游戏制作的基本方法和基本技巧，读完本书，读者将会对游戏制作有一个全面的了解。同时还会掌握使用 Visual Basic、Delphi、C++ Builder 等软件的一些高级技巧。本书包括一个简单的人工智能的游戏（智能黑白棋），在该游戏中玩家可以和电脑对战，电脑可以自己判断在哪里下棋子最为有利。人工智能几乎是每个大型游戏必须具备的部分，往往都有玩家和电脑对战的游戏设置，这些涉及到复杂的算法，需要一些数学知识才能完成。本书详细地说明了一个智能游戏是怎么被设计出来的。

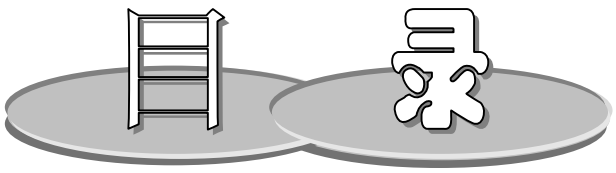
本书共分六篇。第一篇制作了几个简单的小游戏，使读者对游戏制作有一个简单的了解；第二篇讲述如何制作智力游戏；第三篇讲述了一些经典小游戏是如何制作出来的，包括扫雷、华容道、捡金豆等；第四篇讲述了如何制作对抗类游戏；第五

篇讲述了如何制作控制类游戏;第六篇讲述了如何制作趣味游戏。

相信通过本书 100 个实例的学习,可以提高程序员开发小游戏的效率。真诚希望我们的讲解对读者有所帮助。

网冠科技

本书光盘含配套素材，技术支持请点击网冠科技站点 http://netking.163.com。E-mail：netking_@yeah.net。



第三篇　经典游戏

第四篇　对抗游戏

第五篇 控制类游戏

第六篇 趣味游戏

第一篇

小游戏初步

本篇总览

小游戏制作不仅需要有广博的编程知识，还要有丰富的想象力，一个成功的游戏往往通过策划、总体设计、美工制作界面、代码实现等多个步骤来完成。其中，策划是非常关键，又是很容易让人忽略的一步。

本篇以几个简单的实例说明游戏制作的常用手段。主要制作了“数字游戏”、“自动随机出题”、“调用控制面板”、“框架”、“中奖刮刮乐”、“倒计时”、“小日历”等多个游戏，这些游戏有的只是定义了几个变量，然后对玩家的响应作出判断即可，比如“石头、剪刀、布”游戏。

实例 1　“皇后”排列

实例说明

本例制作“皇后”排列小游戏，效果如图 1-1 所示。

程序运行后显示 8X8 的棋盘，游戏者的目的是将 4 个“皇后”和 4 个“武士”放置到棋盘上，让它们互相之间不能吃掉对方。游戏开始时，单击鼠标左键，放置一个“武士”，单击鼠标右键，放置一个“皇后”。游戏界面的右侧显示游戏的状态信息，从这里可看到已放置了几个“皇后”和几个“武士”及单击鼠标的次数。

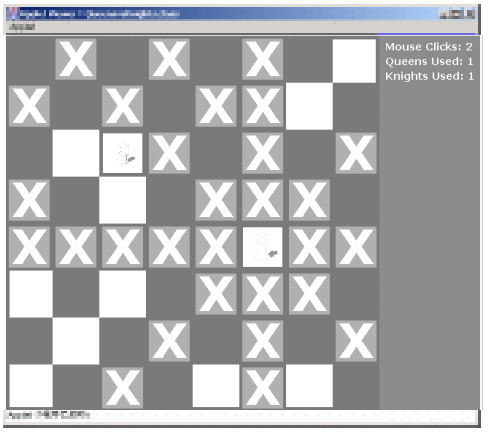


图 1-1　效果图

编程思路

本游戏的“皇后”和“武士”的放置规则是：“皇后”能威胁其所在的直线和斜线上的其他棋子；“武士”能威胁“马”形方格上的其他“武士”和周围两步之内的“皇后”。

程序中将整个棋盘的状态记录在 8×8 的数组中，初始化为 0。如果某个方格放置了“皇后”，则值为 3；放置了“武士”，则值为 2，当前不能放置的，记录为 1。每次用户单击鼠标后，程序即首先判断用户点击的是左键还是右键。如果左键，则判断当前方格是否在威胁之下，并且是空的，然后将值改为 3，标记此处为“皇后”；如果是右键，则类似的值改为 2，放置“武士”。并根据当前的放置，进行数组的重新设置。以便为下一次点击作准备，并重新描绘整个方格棋盘。

创作步骤

1．本游戏实现起来比较简单，只有一个类 QueensvsKnights。它的主要属性有：

整个棋盘的数组，初始化为 0。

```
int array[][] = new int[8][8];                    // 记录“武士”威胁方格的数组
int bj[] = {-2,-2,-1,1,2,2,1,-1};
int bi[] = {-1,1,2,2,1,-1,-2,-2};
String str1,str2,str3;
 Image imageQueen,imageBoard,imageKnight,imageX;
```

2．主要方法描述和实现如下：

```
 public void init()
 {                  // 调入图像，并初始化棋盘数组为 0
...}
 public void paint(Graphics g)
 {            // 显示当前游戏的状态和统计信息，包括一共点击的鼠标次数，“皇后”和“武士”的数目等
.....}
 public boolean mouseDown(Event evt, int   x, int   y)
 {            // 核心方法，处理鼠标点击事件，判断是鼠标左键和鼠标右键
check = checkKnight(arrx,arry);                  // 判断武士的威胁情况
 empty = checkEmpty(arrx,arry);                       // 是否本方格为空白
              if(empty !=1 && check != 1)
              {
                   turn = 1 ;
                   array[arrx][arry] = 3;                  // 数组值置为 1，代表“皇后”
                   queen++;                  // “皇后”数目加 1
                   queenmark(arrx,arry);              // 绘制“皇后”
                   repaint();
                   return true;
              }
                   else return false;

         }

    public void NewMarkKnight(int arrx, int arry)
     {                        // 对武士的“马”形威胁的格子进行判断和置位。这里用到了主要属性中的 bi
和 bj 数组
          int i,xi,yi;
         for (i=0;i<8;i++)
          {
               xi = arrx + bi[i];
               yi = arry + bj[i];
               if (xi >= 0 && xi < 8 && yi >=0 && yi < 8)
                    array[xi][yi] = 1;
          }
                        }

 public void queenmark(int arrx,int arry)
 {                  // 当要放置“皇后”时，对“皇后”的威胁方向逐一地进行分析，将被威胁位置置为 1
       if (arrx >= 0 && arrx <= 7 && arry >=0 && arry <= 7)
```

```
            {
                    mark1(arrx,arry);
                    mark2(arrx,arry);
                    mark3(arrx,arry);
                    mark4(arrx,arry);
                    mark5(arrx,arry);
                    mark6(arrx,arry);
                    mark7(arrx,arry);
                    mark8(arrx,arry);
            }               }

    public int checkKnight(int x,int y)
    {               // 对“武士”的威胁情况判断，即离“武士”在两步之内的格子都被认为是受到了
“武士”的威胁
        if(checkKnight1(x,y) == true )return 1;
        if(checkKnight2(x,y) == true )return 1;
        if(checkKnight3(x,y) == true )return 1;
        if(checkKnight4(x,y) == true )return 1;
        if(checkKnight5(x,y) == true )return 1;
        if(checkKnight6(x,y) == true )return 1;
        if(checkKnight7(x,y) == true )return 1;
        if(checkKnight8(x,y) == true )return 1;

        return(0);
    }

    public int checkEmpty(int column, int row)
    {           // 检查本格子是否为空的方法
        if(array[column][row] != 0 ) return 1;
        else return 0;

    }
```

实例 2　数 字 游 戏

实例说明

本例制作猜数字小游戏，效果如图 2-1 **所示。**

游戏一开始，计算机随机产生了一个不重复的四位数，要输入四位不重复的数与计算机给出的数作对比，如果与计算机给出的数的位置及数字相同，那么将会是 1A**。反之，则显示** 1B**。如计算机的随机数字为** 1234**，猜的数字为：**1356**，那么计算机提示为：**1A1B**，也就是说所猜数字中，有一位数字猜对了，而且数字位置都对，所以显示为** 1A**，而还有一个数字也猜对了，但是位置不对，所以显示为** 1B**。**

图 2-1　**效果图**

编程思路

本例使用随机数生成器随机产生一个 4 **位数，再用命令按钮和菜单实现与用户的交互，从而完成小游戏的制作。**

本例使用的语句及语法如下：

Randomize **语句：初始化随机数生成器。**

语法：Randomize [number]

可选的 number **参数的类型是** Variant **或任何有效的数值表达式。**

说明：Randomize **用** number **将** Rnd **函数的随机数生成器初始化，该随机数生成器给** number **一个新的种子值。如果省略** number**，则用系统计时器返回的值作为新的种子值。如果没有使用** Randomize**，则（无参数的）**Rnd **函数使用第一次调用** Rnd **函数的种子值。若想得到重复的随机数序列，在使用具有数值参数的** Randomize **之前，直接调用具有负参数值的** Rnd **函数。使用具有同样** number **值的** Randomize **是不会得到重复的随机数序列的。**

创作步骤

1**．启动** Visual Basic**，打开一个新的标准工程。如果** Visual Basic **已经运行，那么可在“ 文件 ”菜单中，单击“ 新建工程 ”菜单项，打开一个新的标准工程。**

2**．首先，往窗体中加入一个命令按钮，使用默认的名字** Command1**，如图** 2-2 **所示。**

3．用鼠标选中命令按钮，按鼠标右键，从弹出的菜单中选择复制并粘贴，生成一个控件数组，如图 2-3 所示。

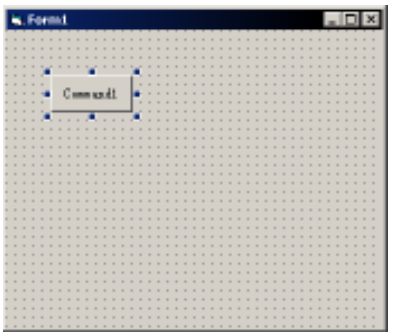


图 2-2

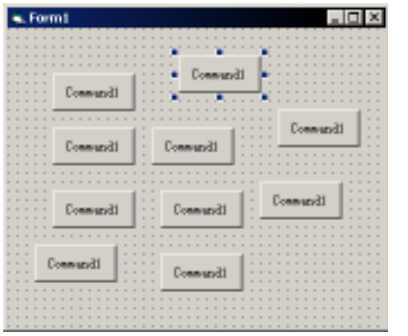


图 2-3

4．设置控件数组的属性，使每个命令按钮的 index 和 caption 的值一样，如图 2-4 所示。

5．选中这些控件，选择菜单中的格式，把它们调整成合适的大小，如图 2-5 所示。

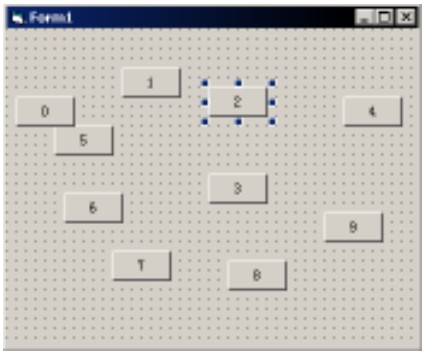


图 2-4

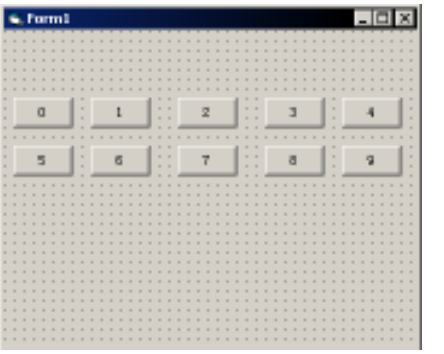


图 2-5

6．加入 Frame 控件，并设置属性，caption 为“提示”，如图 2-6 所示。

7．在 Frame 控件中加入一个 Listbox 控件，如图 2-7 所示。

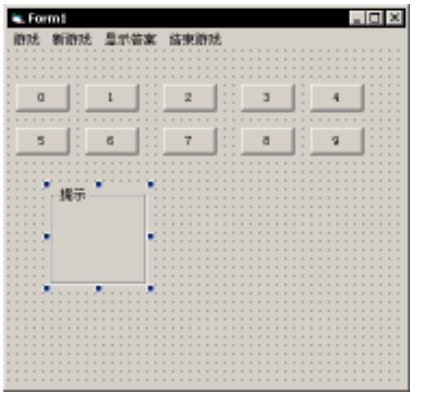


图 2-6

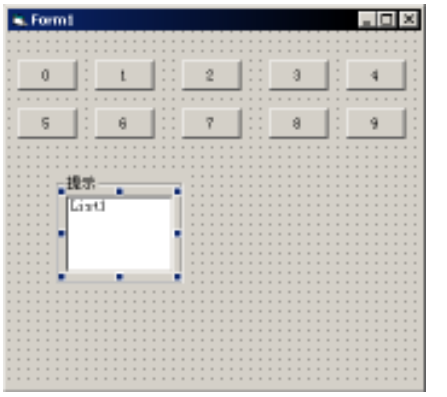


图 2-7

8．加入一个 label 控件，如图 2-8 所示。

9．加入两个命令按钮，caption 分别为“确定”、“取消”，如图 2-9 所示。

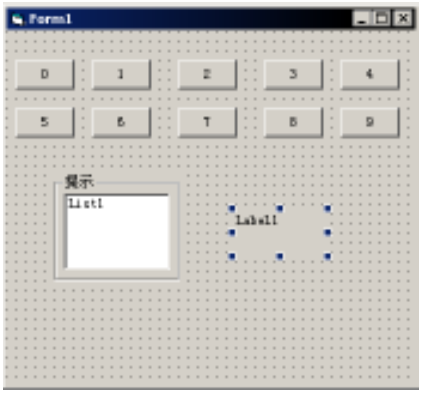


图　2-8

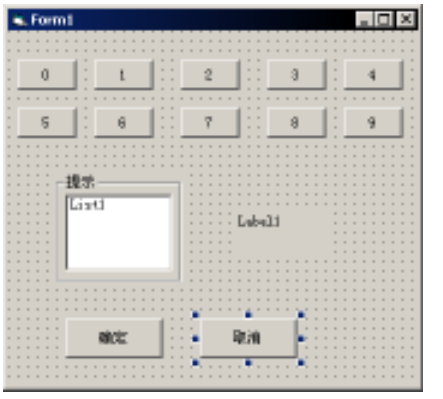


图　2-9

10．然后，按<Ctrl>+<E>键，打开菜单编辑窗口。用菜单编辑器创建 4 个菜单，分别为游戏、新游戏、显示答案、结束游戏。名称分别为 Game、New、View、End，如图 2-10 所示。

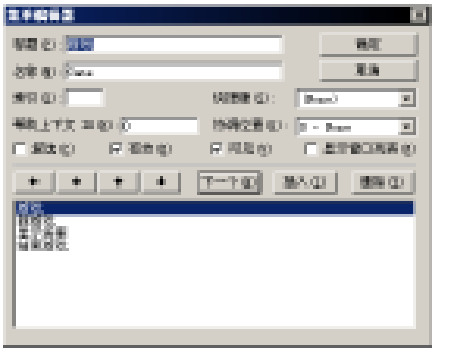


图　2-10

本程序代码详见光盘。

实例 3　石头、剪刀、布

实例说明

本例编写“石头、剪刀、布”游戏，效果如图 3-1 所示。

本例由玩家和电脑对战，具体规则是：“剪刀”赢“布”，“布”赢“石头”，“石头”赢“剪刀”。玩家可以选择任何一种，同时电脑也选择一种，再对电脑和玩家的选择进行比较，看哪一方获胜。最后，在窗体的下方记录成绩。

图 3-1　效果图

编程思路

本实例使用 Delphi 6.0 **来实现。利用随机数产生计算机的选择，然后和玩家的选择进行比较，根据游戏规则，判断结果。**

创作步骤

1．**启动** Delphi 6.0，**打开一个新的标准工程。如果** Delphi **已经运行，则执行** File→New→Application **菜单项，打开一个新的标准工程。向其中添加第一个** Panel **控件，在该控件中，添加三个** Image **控件、一个** GroupBox **控件，三个** RadioButton **控件和一个** Panel **控件；接下来再添加第二个** Panel **控件，在该控件中添加一个** Label **控件和两个** BitBtn **控件。它们的属性设置很简单，这里不再介绍。**

2．**本实例的源代码如下（这里只列出部分关键代码，其他代码详见配套光盘）：**

```
else if nYourChoice = 1 then   // “石头”：“剪刀”
begin
   Label1.Caption := '刀剑无眼,嘿嘿,下次小心了'
   Image1.Picture := picHand;
   Image2.Picture := picBlade;
   Image3.Picture := picFace[3];
   nWin := nWin+1;
end
else if nYourChoice = 2 then          // “布”：“剪刀”
begin
```

```
        Label1.Caption := '我就剪了你的小太阳旗,看你怎么样,哼哼'
        Image1.Picture := picFlag;
        Image2.Picture := picBlade;
        Image3.Picture := picFace[4];
        nLose := nLose + 1;
    end;
end
else if nComChoice = 1 then
begin
    Panel3.Caption := '电脑的选择：“ 石头 ”';
    if nYourChoice = 0 then          // “ 剪刀 ”：“ 石头 ”
    begin
        Label1.Caption := '人总是最厉害的,你就认栽了吧,嘿嘿'
        Image1.Picture := picBlade;
        Image2.Picture := picHand;
        Image3.Picture := picFace[1];
        nLose := nLose + 1;
    end
    else if nYourChoice = 1 then          // “ 石头 ”：“ 石头 ”
    begin
        Label1.Caption := '我怎么还是觉得我的拳头比你的大嘛'
        Image1.Picture := picHand;
        Image2.Picture := picHand;
        Image3.Picture := picFace[2];
        nEqu := nEqu+1;
    end
    else if nYourChoice = 2 then          // “ 布 ”：“ 石头 ”
    begin
        Label1.Caption := '嚣张什么啊，下次砸烂你这块破布'
        Image1.Picture := picFlag;
        Image2.Picture := picHand;
        Image3.Picture := picFace[4];
        nWin := nWin + 1;
    end;
end
else if nComChoice = 2 then
begin
    Panel3.Caption := '电脑的选择：“ 布 ”';        // “ 剪刀 ”：“ 布 ”
    if nYourChoice = 0 then
```

```
    begin
      Label1.Caption := '剪烂我吧，我觉得自己太没脸没羞了';
      Image1.Picture := picBlade;
      Image2.Picture := picFlag;
      Image3.Picture := picFace[4];
      nWin := nWin + 1;
    end
    else if nYourChoice = 1 then          // “石头”：“布”
    begin
      Label1.Caption := '是我不对,我不该赢你,下次我让你好了';
      Image1.Picture := picHand;
      Image2.Picture := picFlag;
      Image3.Picture := picFace[4];
      nLose := nLose + 1;
    end
    else if nYourChoice = 2 then          // “布”：“布”
    begin
      Label1.Caption := '咱们一起去跳太平洋,你觉得如何?';
      Image1.Picture := picFlag;
      Image2.Picture := picFlag;
      Image3.Picture := picFace[4];
      nEqu := nEqu + 1;
    end;
  end;
  Label2.Caption := '当前战绩：'+IntToStr(nWin)+'胜'+IntToStr(nEqu)+'平'+IntToStr(nLose)+'负';
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  nWin := 0;              // 清除记录
  nEqu := 0;
  nLose := 0;
  Label2.Caption := '当前战绩：'+IntToStr(nWin)+'胜'+IntToStr(nEqu)+'平'+IntToStr(nLose)+'负';
end;
end.
```

实例 4　自动随机出题

实例说明

本例编写自动随机出题游戏，效果如图 4-1 **所示。**

假如数学老师要求家长每天给孩子出 10 **道或者** 20 **道算术题作为家庭作业，本例通过游戏来实现这个目标。**

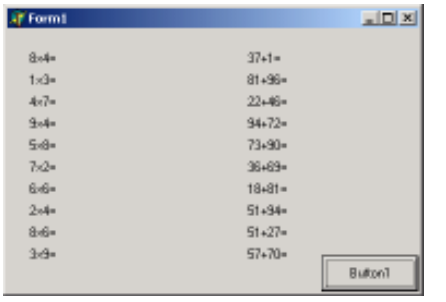


图 4-1　**效果图**

编程思路

在 Delphi **中，有一随机函数，是这样定义的：**function Random [(Range：Integer)]**；其中，参数** Range **为一整数，该函数返回值也为整数，其范围为：**0<=Random(Range)< Range (**指定** Range) 0<="Random<" 1 (**不带参数** Range) **，下面的过程中，**for **循环里第一条语句在屏幕上输出九九表内乘法的随机题；第二条语句在屏幕上输出** 100 **以内的加法随机题，如图** 4-1 **所示。稍加修改，增加一些条件语句即可得到减法、除法以及四则混合运算的随机题。当然，也可以直接将结果输出在打印机的画布（**Canvas**）上。**

创作步骤

1．**启动** Delphi 6.0，**打开一个新的标准工程。如果** Delphi **已经运行，则执行** File→New→Application **菜单项，打开一个新的标准工程。向其中添加一个** Button **控件，如图** 4-2 **所示。**

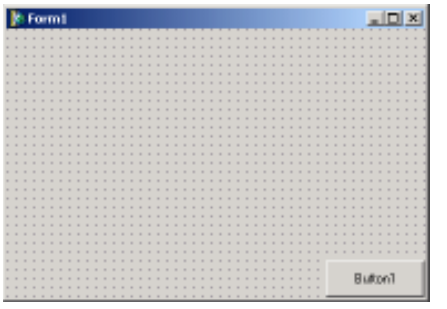


图　4-2　Form **界面图**

2．**设置窗体的属性如下：**

Left = 192

Top = 107

Width = 385

```
    Height = 268
    Caption = 'Form1'
    Color = clBtnFace
    PixelsPerInch = 96
    TextHeight = 13
```

3．本实例源代码如下：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
     I: Integer;
begin
 Randomize;
   for I := 1 to 10 do
    begin
     Canvas.TextOut(20,I*20,
     IntToStr(Random(9)+1)+'×'+IntToStr(Random(9)+1)+'=');
     Canvas.TextOut(220,I*20,
     IntToStr(Random(100))+'+'+IntToStr(Random(100))+'=');
    end;
end;
end.
```

实例 5　火　　炮

实例说明

本例编写的程序是个小动画和声音演示的例子，效果如图 5-1 所示。

在单击按钮的时候，会出现一幅火炮正开炮的动画，同时还伴随着火炮的声响。

本程序中用到了一个 BitBtn 控件，后面所编写的所有代码都是针对它的 Glyph 属性来进行处理的。

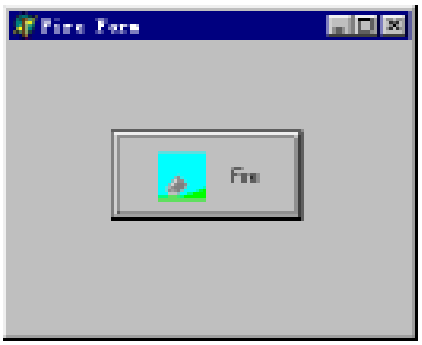


图 5-1　效果图

编程思路

这个程序主要是通过设置 BitBtn 控件的 Glyph 属性来实现的。将它的 Glyph 属性设置为某一幅位图之后，然后就可以动态显示了。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，新建一个工程。向窗体中添加一个 BitBtn 控件。在 BitBtn 控件中添加一张图片。

2．编写相应的事件处理过程，具体代码如下：

```
unit FireForm;
interface
uses Windows, Classes, Graphics, Forms, Controls, Buttons, Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    BitBtnFire: TBitBtn;
    procedure BitBtnFireMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtnFireMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtnFireClick(Sender: TObject);
    private
    { Private declarations }
    public
```

```
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM}
uses
MmSystem;
// 为了使用 PlaySound 必须加入此行
procedure TForm1.BitBtnFireMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
//  载入开火的位图
if Button = mbLeft then
BitBtnFire.Glyph.LoadFromFile ('fire2.bmp');
end;
procedure TForm1.BitBtnFireMouseUp(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
//  载入默认的位图
if Button = mbLeft then
BitBtnFire.Glyph.LoadFromFile ('fire.bmp');
end;
procedure TForm1.BitBtnFireClick(Sender: TObject);
begin
PlaySound ('Boom.wav', 0, snd_Async);
MessageDlg ('Boom!', mtWarning, [mbOk], 0);
end;
end.
```

实例 6　控制 Power 键

实例说明

本程序实现了在 Windows 2000 中按 Power 键快速关机的功能，如图 6-1 所示。

本例解决了没有任何警告就关机，或是有的键盘上 Power 键就在 Delete 键下面，经常在按删除键时误按 Power 键，导致工作中经常关机的问题。

本实例提供了一个程序，当按下 Power 键后，会发出一个提示，等待用户单击“确定”按钮后才关机。

图 6-1　效果图

编程思路

本实例利用 Delphi 中的 API 函数捕获用户的按键信息，如果用户按下了 Power 键，则弹出一个对话框，提示是否关闭计算机。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。

2．向程序中添加如下代码：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
  procedure WMPowerBroadcast(var message: TMessage); message WM_POWERBROADCAST;
  private
    { Private declarations }
  public
    { Public declarations }
```

```
    end;

  var
    Form1: TForm1;

  implementation

  {$R *.dfm}
  procedure TForm1.WMPowerBroadcast(var message: TMessage);
  var
    SkipNextPowerMsg:boolean;
  begin
    if SkipNextPowerMsg then
      begin
        SetForegroundWindow(Self.Handle);
        if  Application.MessageBox('是否关闭系统？','警告',MB_OKCANCEL  +  MB_DEFBUTTON2)<>IDOK
then
        begin
          message.Result := BROADCAST_QUERY_DENY;
          SkipNextPowerMsg:=not SkipNextPowerMsg;
        end
        else
          Close;
      end
    else
      SkipNextPowerMsg:=not SkipNextPowerMsg;
  end;
  end.
```

实例 7　调用控制面板

实例说明

本实例演示了如何调用控制面板中的内容，如图 7-1 所示。

本例运行时，用户单击“启动控制面板”按钮之后，将会弹出 Windows 2000 中的大部分控制程序。

图 7-1　效果图

编程思路

本实例利用 Delphi 中的 API 函数 WinExec，启动 Windows 下 rundll32.exe 和 shell32.dll 文件中的内容。该函数的原型如下：

```
UINT WinExec(
    LPCSTR lpCmdLine,     // 地址命令行
    UINT uCmdShow         // 窗口样式定义
   );
```

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Button 控件。

2．向程序中添加如下代码：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
 x:cardinal;
begin {启动控制面板}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL',9); {辅助选项 属性-键盘}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL access.cpl,,1',9); {辅助选项 属性-声音}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL access.cpl,,2',9); {辅助选项 属性-显示}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL access.cpl,,3',9); {辅助选项 属性-鼠标}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL access.cpl,,4',9); {辅助选项 属性-常规}
 x:=winexec('rundll32.exe  shell32.dll,Control_RunDLL  access.cpl,,5',9);  {添加/删除程序 属性-安装/卸
                                                                          载}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Appwiz.cpl,,1',9); {添加/删除程序 属性-Windows
                                                                          安装程序}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Appwiz.cpl,,2',9); {添加/删除程序 属性-启动盘}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Appwiz.cpl,,3',9); {显示 属性-背景}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL desk.cpl,,0',9); {显示 属性-屏幕保护程序}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL desk.cpl,,1',9); {显示 属性-外观}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL desk.cpl,,2',9); {显示 属性-设置}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL desk.cpl,,3',9); {Internet 属性-常规}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Inetcpl.cpl,,0',9); {Internet 属性-安全}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Inetcpl.cpl,,1',9); {Internet 属性-内容}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Inetcpl.cpl,,2',9); {Internet 属性-连接}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Inetcpl.cpl,,3',9); {Internet 属性-程序}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Inetcpl.cpl,,4',9); {Internet 属性-高级}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Inetcpl.cpl,,5',9); {区域设置 属性-区域设置}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Intl.cpl,,0',9); {区域设置 属性-数字}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Intl.cpl,,1',9); {区域设置 属性-货币}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Intl.cpl,,2',9); {区域设置 属性-时间}
 x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Intl.cpl,,3',9); {区域设置 属性-日期}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Intl.cpl,,4',9); {游戏控制器-一般}
```

```
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Joy.cpl,,0',9); {游戏控制器-高级}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Joy.cpl,,1',9); {鼠标 属性}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Main.cpl',9); {多媒体 属性-音频}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Mmsys.cpl,,0',9); {多媒体 属性-视频}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Mmsys.cpl,,1',9); {多媒体 属性-MIDI}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Mmsys.cpl,,2',9); {多媒体 属性-CD 音乐}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Mmsys.cpl,,3',9); {多媒体 属性-设备}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Mmsys.cpl,,4',9); {调制解调器 属性}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Modem.cpl',9); {网络}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Netcpl.cpl',9); {密码 属性}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Password.cpl',9); {扫描仪与数字相机 属性}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Sticpl.cpl',9); {系统 属性-常规}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Sysdm.cpl,,0',9); {系统 属性-设备管理器}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Sysdm.cpl,,1',9); {系统 属性-硬件配置文件}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Sysdm.cpl,,2',9); {系统 属性-性能}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Sysdm.cpl,,3',9); {日期/时间 属性}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL timedate.cpl',9); {电源管理 属性}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Powercfg.cpl',9); {拨号属性}
  x:=winexec('rundll32.exe shell32.dll,Control_RunDLL Telephon.cpl',9); {调用错误}
  if x=0 then messagebox(0,'程序超出内存','错误',0);
  if x=ERROR_BAD_FORMAT then messagebox(0,'该程序不是一个合法的 Win32.EXE 程序).','错误',0);
  if x=ERROR_FILE_NOT_FOUND then messagebox(0,'指定文件没找到','错误',0);
  if x=ERROR_PATH_NOT_FOUND then messagebox(0,'指定路径没找到','错误',0);
end;
end.
```

实例 8　框　　架

实例说明

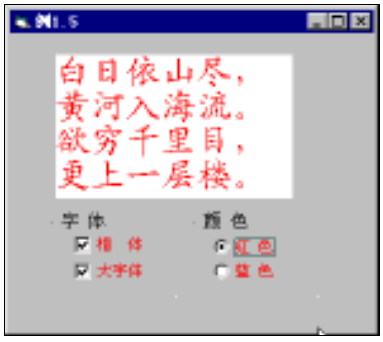


图 8-1　效果图

本例制作框架和复选框程序。效果如图 8-1 所示。

程序运行时，通过单击左边框架内的复选框可改变字体，单击右边框架内的单选按钮可改变字体的颜色。

许多游戏都要编写框架程序，本例作为一个示范实例用 Visual Basic 6.0 来实现。

编程思路

在 Windows 应用程序中，一个窗体窗口内可能会存在许多控件。通常为了能将控件适当功能分类，可将性能相近的控件放在框架内。要将各控件放在框架内，首先必须建立框架，然后再建立控件。按这种方法操作后，以后如果要移动框架，则框架内的控件也将随着移动，以避免分别移动框架和各控件的麻烦。复选框和单选按钮相似，也是选择类控件。在运行时，如果用户用鼠标单击复选框前的方框，则方框中会出现一个“√”字符，表示已选取了这一项功能。复选框的功能是独立的，用户可根据需要选取一个或多个复选框。

创作步骤

1．启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中，单击“新建工程”菜单项，打开一个新的标准工程。

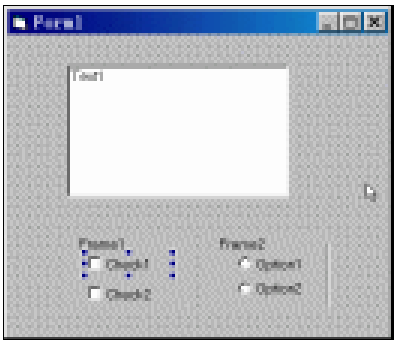


图　8-2

2．在新窗体中创建一个文本框（Text Box）、两个框架（Frame）、两个复选框（Check Box）、两个单选按钮（Option）。复选框和单选按钮的主要区别是，多个复选框可同时选中，而单选按钮仅能选中一个。本例中出现了一个 Frame1 控件。它是一种框架，可以用来把窗体分成多个部分。首先应该在窗体上画出 Frame1 控件，然后再选择相应的控件在 Frame1 上描出轮廓，使其出现在 Frame1 之上。其中复选框的图标是 ，整个窗体界面如图 8-2 所示。

3．设置窗体及各个控件的属性，其中要注意 Text1 的宽度（Width）及高度（Height），保证大字体

时刚好显示两行，小字体时显示四行，如得不到满意的效果，可结合程序进行调整。其效果如图 8-3 所示。

各控件属性设置如下：

Frame1　Name = Frame1
Caption = “字体”
Font = “宋体”（字号为 5 号字）
Width = 1215
Height = 1000
Left = 480

Frame2　Name = Frame2
Caption = “颜色”
Font = “宋体”（字号为 5 号字）
Width = 1215
Height = 1000
Left = 2160

Check1　Name =Ckeck1
Caption = “楷体”
Font = “宋体”（字号为 5 号字）
ForeColor = &H000000FF&
Value = 0 - Unchecked
Width = 852
Height = 192
Left = 280
Top = 280

Check2　Name =Ckeck2
Caption = “大字体”
Font = “宋体”（字号为 5 号字）
ForeColor = &H000000FF&
Value = 0 - Unchecked
Width = 852
Height = 192
Left = 280
Top = 600

Option1　Name =Option1
Caption = “红色”
Font = “宋体”（字号为 5 号字）
ForeColor = &H000000FF&
Value = False
Width = 852
Height = 252
Left = 240
Top = 280

Option2　Name = Option2
Caption = “蓝色”
Font = “宋体”（字号为 5 号字）
ForeColor = &H000000FF&
Value = False
Width = 852
Height = 252
Left = 240
Top = 280

编写代码详见光盘。

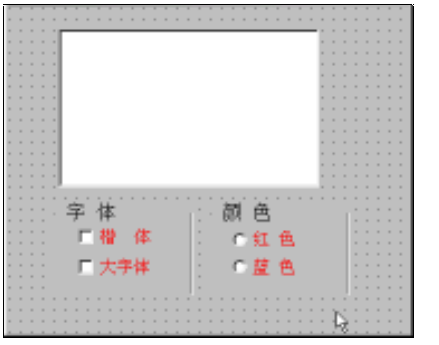


图　8-3

实例 9　设置全局热键

实例说明

本例是一个响应用户按键的小游戏。效果如图 9-1 **所示。**

本例运行过程中，不管当前窗口中正在运行什么程序，只要按下 Alt+F12 **键，该程序窗口就会以全屏显示出来。**

图 9-1　**效果图**

编程思路

本实例使用 Delphi 6.0 **来实现。这里调用了一个** API **函数** RegisterHotKey**。该函数定义一个系统范围的热键。其原型如下：**

```
BOOL RegisterHotKey(
  HWND hWnd,
  int id,
  UINT fsModifiers,
  UINT vk
```

参数 id **是用户自己定义的一个** ID **值。对一个线程来讲其值必须在** 0x0000～0xBFFF **范围之内；对** DLL **来讲其值必须在** 0xC000～0xFFFF **范围之内，在同一进程内该值必须是惟一的。**

参数 fsModifiers **指明与热键联合使用按键，可取值为：**MOD_ALT、MOD_CONTROL、MOD_WIN、MOD_SHIFT**。**

参数 vk **指明热键的虚拟键码。**

创作步骤

1．**启动** Delphi 6.0**，打开一个新的标准工程。如果** Delphi **已经运行，则执行** File→New→Application **菜单项，打开一个新的标准工程。向其中添加一个** Image **控件。在** Image **控件中添加一张图片。**

2．**本实例的源代码如下（这里只列出部分关键代码，其他代码详见配套光盘）：**

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;
```

```
type
  TForm1 = class(TForm)
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
      aatom:atom;
      procedure hotkey(var msg:tmessage);message wm_hotkey;
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation

{$R *.dfm}
procedure TForm1.hotkey(var msg:tmessage);
begin
  if (msg.LParamHi=VK_F12) and (msg.LParamLo=MOD_ALT) then
  begin
   form1.show;
   SetForegroundWindow(handle);
   form1.WindowState:=wsMaximized;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  aatom:=globaladdatom('hot key');
  RegisterHotKey(handle,aatom,MOD_ALT,vk_f12);
end;
procedure TForm1.FormDestroy(Sender: TObject);
begin
 globalDeleteatom(aatom);
end;end.
```

实例 10　中奖刮刮乐

实例说明

本例编写中奖刮刮乐游戏程序，运行后效果如图 10-1 **所示。**

本例可以使用鼠标将标签前面的覆盖层擦去，就像抽奖时刮开中奖号码一样。

图 10-1　**效果图**

编程思路

本实例使用 Visual Basic 6.0 **来实现。利用两个** PictureBox **控件，其中，第一个作为覆盖层，第二个控件用来存放数字。然后响应鼠标的** MouseMove **事件，不断将覆盖层的内容变为第二个** PictureBox **控件中的内容就可以了。**

创作步骤

1．**启动** Visual Basic 6.0，**打开一个新的标准工程。如果** Visual Basic **已经运行，则执行** File→New→Project **菜单项，打开一个新的标准工程。向其中添加两个** PictureBox **控件，如图** 10-2 **所示。**

图　10-2

2．**设置两个** PictureBox **控件的属性如下：**

```
Begin VB.PictureBox Picture2
    Height = 450
    Left = 600
    Picture = "Form1.frx":0000
    ScaleHeight = 390
    ScaleWidth = 1875
    TabIndex = 1
    Top = 792
```

```
    Width = 1935
End
Begin VB.PictureBox Picture1
    BackColor = &H8000000D&
    Height = 468
    Left = 600
    ScaleHeight = 405
    ScaleWidth = 1245
    TabIndex = 0
    Top = 144
    Width = 1308
End
```

3．本实例源代码如下：

```
Option Explicit
Private Const size = 10 代表每次刮掉 10*10 个图形单位 (pixel)

Private Sub Form_Load()
    BackColor = &H80FFFF
    Picture2.BorderStyle = 0
    Picture2.AutoSize = True
    Picture2.AutoRedraw = True
    Picture2.ScaleMode = vbPixels
    Picture2.Visible = False
    Picture1.AutoRedraw = True
    Picture1.BorderStyle = 0
    Picture1.ScaleMode = vbPixels
    Picture1.Move Picture1.Left, Picture1.Top, Picture2.Width, Picture2.Height
End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, _
                               X As Single, Y As Single)
    If Not Button = vbLeftButton Then Exit Sub
    Dim i As Long, j As Long
    For i = 0 To size
        For j = 0 To size
            Picture1.PSet (X + i, Y + j), Picture2.Point(X + i, Y + j)
        Next
    Next
End Sub
```

实例 11　小　屏　保

实例说明

本实例制作了一个简单的屏幕保护程序。效果如图 11-1 **所示。**

程序运行之后，屏幕中将出现几条随机变化的折线。移动鼠标、单击鼠标或者按下键盘时，程序将会自动退出。

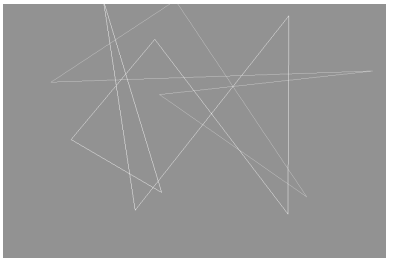


图 11-1　**效果图**

编程思路

本实例使用 Visual Basic 6.0 **来实现。使用一个** Timer **控件随机点来变换折线的颜色，再使用另一个** Timer **控件逐渐移动线条的位置。同时响应** KeyDown、MouseDown、MouseMove **事件，当这些事件发生时，退出程序。**

创作步骤

1．**启动** Visual Basic 6.0，**打开一个新的标准工程。如果** Visual Basic **已经运行，则执行** File→New→Project **菜单项，打开一个新的标准工程。向其中添加两个** Timer **控件。**

2．**设置两个** Timer **控件的关键属性如下：**

Timer1：

Interval = 1000

Timer2：

Interval = 3000

3．**程序源代码如下：**

```
Private Declare Function ShowCursor Lib "user32" (ByVal bShow As Long) As Long
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Dim Cor(3)

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
Unload Me
End Sub

Private Sub Form_Load()
If App.PrevInstance = True Then End          如果该程序在运行，则结束当前程序
```

```
Me.Move 0, 0, Screen.Width, Screen.Height
ShowCursor False                                    隐藏鼠标
End Sub

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Unload Me
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Timer1.Enabled = True Then Exit Sub
Unload Me
End Sub
```

……**（此处代码略，详见光盘）**

```
Next
Do
DoEvents
For n = 1 To 3
Line (X1(n), Y1(n))-(X2(n), Y2(n)), Cor(n)
Line (X2(n), Y2(n))-(X3(n), Y3(n)), Cor(n)
Line (X3(n), Y3(n))-(X4(n), Y4(n)), Cor(n)
Line (X4(n), Y4(n))-(X1(n), Y1(n)), Cor(n)
Next
For n = 1 To 3
If X1(n) > Me.ScaleWidth Or X1(n) < 0 Then a1(n) = -a1(n)
If Y1(n) > Me.ScaleHeight Or Y1(n) < 0 Then a2(n) = -a2(n)
If X2(n) > Me.ScaleWidth Or X2(n) < 0 Then b1(n) = -b1(n)
If Y2(n) > Me.ScaleHeight Or Y2(n) < 0 Then b2(n) = -b2(n)
If X3(n) > Me.ScaleWidth Or X3(n) < 0 Then c1(n) = -c1(n)
If Y3(n) > Me.ScaleHeight Or Y3(n) < 0 Then c2(n) = -c2(n)
If X4(n) > Me.ScaleWidth Or X4(n) < 0 Then d1(n) = -d1(n)
If Y4(n) > Me.ScaleHeight Or Y4(n) < 0 Then d2(n) = -d2(n)
Next
Sleep 0.51
Cls
For n = 1 To 3
X1(n) = X1(n) + a1(n)
Y1(n) = Y1(n) - a2(n)
X2(n) = X2(n) + b1(n)
```

```
Y2(n) = Y2(n) + b2(n)
X3(n) = X3(n) + c1(n)
Y3(n) = Y3(n) - c2(n)
X4(n) = X4(n) + d1(n)
Y4(n) = Y4(n) = d2(n)
Next
DoEvents
Loop
End Sub

Private Sub Timer2_Timer()
For j = 1 To 3
DoEvents
Cor(j) = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
DoEvents
Next
End Sub
```

4．将程序打包成可执行文件之后，将可执行文件的扩展名改为 scr**，放到** C:\WINNT\system32 **目录下，然后在桌面中右击鼠标，在快捷菜单中选择“属性”菜单项。在出现的窗口中选择“屏幕保护程序”标签页，此时就可以看到刚才做的屏保程序了，如图** 11-2 **所示。**

图 11-2

实例 12　无限星空人机交互

实例说明

在本实例中制作了一个绚丽的星空画面，如图 12-1 所示。

本例程序运行后，将以全屏显示，窗口中有一个按钮，单击该按钮后，就开始进行星空之旅了，在窗口中间有一个一闪一闪的小亮点，同时在亮点的四周有许多星星飞过。最后，按下 ESC 键将退出程序。

在本实例中主要讲述窗体的属性设置以及程序公共模块的内容。

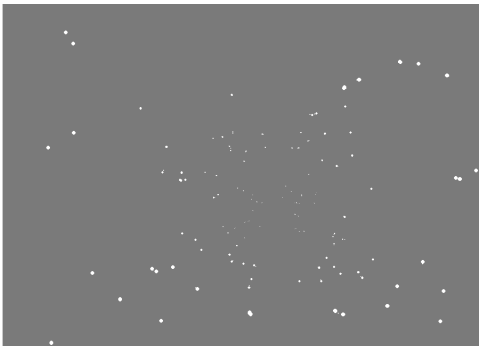


图 12-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。利用 Timer 控件生成随机点，然后利用 SetPixel 函数绘制点，同时不断变换点的位置，就可以实现星空的效果了。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加一个 Timer 控件和一个 CommandButton 控件。

2．设定 Timer 控件的关键属性如下：

Interval = 1

3．在项目导航窗口中右击鼠标，在弹出的快捷菜单中选择 Add→Module 菜单项，向程序中添加一个模块，在该模块中添加如下代码：

```
Public Type Stars
  x As Double
  y As Integer
  AddX As Integer
  AddY As Integer
End Type
Public Star(1000) As Stars
Public Accelarate As Boolean
Declare Function SetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal
```

crColor As Long) As Long

4．**本实例源代码如下：**

```
Private W As Integer
Private H As Integer
Private Sub Command1_Click()
```

响应“开始星空之旅”按钮

```
MoveTo = move_forward
Command1.Visible = False
WindowState = 2
W = ScaleWidth
H = ScaleHeight
For i = 1 To 150
   Star(i).x = W / 2
   Star(i).y = H / 2
RandomX:
   Randomize
   Star(i).AddX = Int(Rnd * 29) - Int(Rnd * 29)
   If Star(i).AddX = 0 Then GoTo RandomX
RandomY:
   Star(i).AddY = Int(Rnd * 19) - Int(Rnd * 19)
   If Star(i).AddY = 0 Then GoTo RandomY
 Next
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = vbKeyEscape Then
```

退出程序

```
End
End If
End Sub

Private Sub Form_Load()
```

确定窗口的大小

```
Move Screen.Width / 2 - Width / 2, Screen.Height / 2 - Height / 2
Command1.Move ScaleWidth / 2 - Command1.Width / 2, ScaleHeight / 4 - Command1.Height / 2
End Sub
```

实例 13　无限星空星星闪

实例说明

续前面的实例，在本实例中讲述“无限星空”的核心部分。效果如图 13-1 所示。只有通过 Timer 控件的 OnTimer 事件处理函数，才能实现绚丽多彩的星空画面。

图 13-1　效果图

编程思路

在 Timer 控件的 OnTimer 事件处理函数中，随机地产生一些点，然后使这些点沿着不同的路径向窗体四周扩散，最后从整体上形成星空的画面。

创作步骤

程序设计步骤续前面的例子。本程序的 Timer 事件处理函数如下：

```
Private Sub Timer1_Timer()
产生星空画面
If Command1.Visible = True Then Exit Sub
For i = 1 To 150
   SetPixel hdc, W / 2, H / 2, &H404040
   Select Case Abs(W / 2 - (Star(i).x))
   Case Is < 20
     col = &H0&
     Size = 1
   Case Is < 80
     col = &H404040
     Size = 1
   Case Is < 150
     col = &H808080
     Size = 2
   Case Is < 200
     col = &HC0C0C0
     Size = 3
```

```
  Case Is < 250
    col = &HFFFFFF
    Size = 4
  Case Else
    col = &HFFFFFF
    Size = 5
End Select

Select Case Abs(H / 2 - (Star(i).y))
  Case Is < 20
    If Size = 0 Then
      Size = 1
      col = back5
    End If
  Case Is < 80
    If Size = 0 Then
      col = &H404040
      Size = 1
    End If
  Case Is < 150
    If Size < 2 Then
      Size = 2
      col = &H808080
    End If
  Case Is < 200
    If Size < 3 Then
      Size = 3
      col = &HC0C0C0
    End If
  Case Is < 250
    If Size < 4 Then
      Size = 4
      col = &HFFFFFF
    End If
  Case Else
    If Size < 5 Then
      Size = 5
      col = &HFFFFFF
    End If
```

```
    End Select
SetPixel hdc, W / 2, H / 2, col
Select Case Size
  Case 1
    SetPixel Me.hdc, Star(i).x, Star(i).y, &H0&
    SetPixel Me.hdc, Star(i).x + Star(i).AddX, Star(i).y + Star(i).AddY, col
……（此处代码略，详见光盘）
    SetPixel Me.hdc, Star(i).x - 1 + Star(i).AddX, Star(i).y - 2 + Star(i).AddY, col
End Select
Star(i).x = Star(i).x + Star(i).AddX
Star(i).y = Star(i).y + Star(i).AddY
Star(i).AddX = Star(i).AddX + Sgn(Star(i).AddX) * (Size / 5)
Star(i).AddY = Star(i).AddY + Sgn(Star(i).AddY) * (Size / 5)

If Star(i).x < 0 Or Star(i).x > ScaleWidth Or Star(i).y < 0 Or Star(i).y > ScaleHeight Then
    Star(i).x = W / 2
    Star(i).y = H / 2
RandomX:
    Randomize
    Star(i).AddX = Int(Rnd * 29) - Int(Rnd * 29)
    If Star(i).AddX = 0 Then GoTo RandomX
RandomY:
    Star(i).AddY = Int(Rnd * 19) - Int(Rnd * 19)
    If Star(i).AddY = 0 Then GoTo RandomY
End If
Next
End Sub
```

实例 14　倒　计　时

实例说明

本例实现了一个倒计时牌的功能。效果如图 14-1 所示。

在“设定时间”下面的三个文本输入框中可以输入倒计时的时间，单击“开始”按钮，则倒计时开始，在“剩余时间”下面将显示剩余的时间。单击“暂停”按钮，则倒计时暂停，此时“置零”按钮变为可用状态，单击该按钮可以将“设定时间”和“剩余时间”下面的数字清零。单击“退出”按钮，将退出程序。

图 14-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。当用户在“设定时间”下面的三个文本输入框中输入数字时，使用三个变量把它们保存起来，然后使用 TimeSerial 函数将三个数字转换为时间类型的变量。当用户单击“开始”按钮后，将激活 Timer 控件，该控件使设定的时间逐渐减少。当单击“暂停”按钮时，将 Timer 控件的 Enabled 属性变为 False。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加六个 Label 控件、三个 TextBox 控件、四个 Command 控件和一个 Timer 控件，并布置控件位置，如图 14-2 所示。

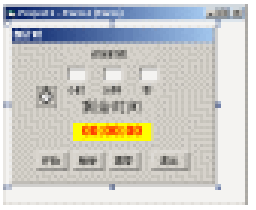


图　14-2

2．设置窗体的关键属性如下：

AutoRedraw = True

BorderStyle = Fixed Single

Caption = "**倒计时**"

PaletteMode = UseZOrder

3．**设置** Timer **控件的关键属性如下：**

Enabled= False

其他控件的属性设置比较简单，这里不再介绍。

4．**本实例的源代码如下：**

```
Dim Hours As Integer
Dim Minutes As Integer
Dim Seconds As Integer
Dim Time As Date
Private Sub Mydisplay()记录用户的输入
    Hours = Val(Text1.Text)
    Minutes = Val(Text2.Text)
    Seconds = Val(Text3.Text)
将数字转变为时间
    Time = TimeSerial(Hours, Minutes, Seconds)
在 Label1 中显示时间
    Label1.Caption = Format$(Time, "hh") & ":" & Format$(Time, "nn") & ":" & Format$(Time, "ss")
End Sub
Private Sub Command1_Click()
激活 Timer 控件
    Timer1.Enabled = True
    Command3.Enabled = False
End Sub
Private Sub Command2_Click()
暂停时间记录
    Timer1.Enabled = False
    Command3.Enabled = True
End Sub
Private Sub Command3_Click()
重置时间
    Hours = 0
    Minutes = 0
    Seconds = 0
    Time = 0
    Text1.Text = " "
    Text2.Text = " "
    Text3.Text = " "
    Text1.SetFocus 'put curser in the hour text box
End Sub
```

```
Private Sub Command4_Click()
退出程序
    End
End Sub
Private Sub Form_Load()
使窗口居中
Form1.Top = (Screen.Height - Form1.Height) / 2
Form1.Left = (Screen.Width - Form1.Width) / 2
    Timer1.Interval = 1000
    Hours = 0
    Minutes = 0
    Seconds = 0
    Time = 0
End Sub
Private Sub Text1_Change()
    Mydisplay
End Sub
Private Sub Text2_Change()
    Mydisplay
End Sub
Private Sub Text3_Change()
    Mydisplay
End Sub
Private Sub Timer1_Timer()
动态改变时间
    Timer1.Enabled = False
    If (Format$(Time, "hh") & ":" & Format$(Time, "nn") & ":" & Format$(Time, "ss")) <> "00:00:00"
    Then 'Counter to continue loop until 0
        Time = DateAdd("s", -1, Time)
        Label1.Visible = False
        Label1.Caption = Format$(Time, "hh") & ":" & Format$(Time, "nn") & ":" & Format$(Time, "ss")
        Label1.Visible = True
        Timer1.Enabled = True
    Else
        Timer1.Enabled = False
        Beep
        Beep
        Command3.Enabled = True
    End If
End Sub
```

实例 15　小　日　历

实例说明

本例制作的是一个小日历，程序运行结果如图 15-1 所示。

程序启动后，将取得系统时间，并显示出来，用户可以在窗口中改变当前的年份、月份和日期。同时在窗口的右侧显示相应时间是星期几。

本例使用 Visual Basic 语言的各种常用控件实现了和 Windows 日历相类似的功能。

图 15-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。在窗体中使用两个 ComboBox 控件来显示年份和月份。使用了 31 个 Label 控件用来显示日期。单击“确定”按钮，将更改日期，单击“取消”按钮，将退出程序。另外采用了一种算法，来取得指定日期对应的是星期几。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向窗体中添加两个 ComboBox 控件（在它们的上方分别添加两个 Lable 控件进行说明）、两个 CommandButton 控件，在“确定”按钮的上方添加两个 Label 控件，名称分别为 lblday 和 lbldate，另外再添加 31 个 Label 控件用来显示时间。

2．设置窗体的关键属性如下：

```
StartUpPosition = 3    'Windows Default
LinkTopic =     "Form1"
```

3．设置窗体的关键属性如下：

```
BorderStyle      =    1    'Fixed Single
Caption          =    "小日历"
ClientHeight     =    2865
ClientLeft       =    930
ClientTop        =    1485
ClientWidth      =    4290
ControlBox       =    0     'False
```

4．本实例的源代码如下：

……（此处代码略，详见光盘）

```
Private Sub cmdok_Click() '“确定”按钮响应
Dim month1%, day1%, year1%, date1$
```

```
day1% = selectedate%
month1% = cbomonth.ListIndex + 1
year1% = cboyear.ListIndex + 1960
date1$ = (Str$(month1%) + "/" + Str$(day1%) + "/" + Str$(year1%))
date1$ = Format$(date1$, "general date")
MsgBox Format$(date1$, "long date")
End Sub

Private Function determinemonth%() 判断选择了哪个月
Dim i%
i% = cbomonth.ListIndex
determinemonth% = i% + 1
End Function

Private Function determineyear%() 判断选择了哪一年
Dim i%
i% = cboyear.ListIndex
If i% = -1 Then Exit Function
determineyear% = CInt(Trim(cboyear.List(i%)))
End Function

Private Sub displaynumbers(number%) 显示日期
Dim i%
For i% = 28 To 30
    lblnumber(i%).Visible = False
Next i%
For i% = 28 To number% - 1
    lblnumber(i%).Visible = True
Next i%
End Sub

Private Sub fillcbomonth() 读入当前的月份
cbomonth.AddItem "January"
cbomonth.AddItem "February"
cbomonth.AddItem "March"
cbomonth.AddItem "April"
cbomonth.AddItem "May"
cbomonth.AddItem "June"
cbomonth.AddItem "July"
cbomonth.AddItem "August"
cbomonth.AddItem "September"
cbomonth.AddItem "October"
cbomonth.AddItem "November"
```

```
cbomonth.AddItem "December"
End Sub

Private Sub fillcboyear() 读入当前的年份
Dim i%
For i% = 1960 To 2060
    cboyear.AddItem Str$(i%)
Next i%
End Sub

Private Sub Form_Load() 初始化各控件的属性
selectedate% = CInt(Format$(Now, "dd"))
Call fillcbomonth
Call fillcboyear
Call setdate
Dim r%, caption1$
r% = Weekday(Format$(Now, "general date"))
If r% = 1 Then
    caption1$ = "Sunday"
ElseIf r% = 2 Then
    caption1 = "Monday"
ElseIf r% = 3 Then
    caption1 = "Tuesday"
ElseIf r% = 4 Then
    caption1 = "Wednesday"
ElseIf r% = 5 Then
    caption1 = "Thursday"
ElseIf r% = 6 Then
    caption1 = "Friday"
Else
    caption1 = "Saturday"
End If
lblday.Caption = caption1$
End Sub

Private Sub lblnumber_click(Index As Integer) 选择日期
Dim i%
……（此处代码略，详见光盘）
year1% = determineyear()
Call checkdate(month1%, year1%)
End Sub
```

第二篇

智力游戏

本篇总览

智力游戏是人们喜闻乐见的一种游戏。这种类型的游戏通常是先定义一个游戏规则，然后玩家在符合规则的情况下，达到某个目标。

本篇主要制作了智力魔方、拼图游戏、点灯、单人跳棋、小迷宫等游戏。每个游戏都依据游戏规则对用户的操作进行判断，如果符合规则就允许其操作，反之取消操作或者给出提示信息。

程序通常定义多个变量用来描述游戏的当前状态，比如小迷宫游戏就是通过一个.dat 文件来保存迷宫的信息，然后当用户操作键盘时，判断玩家所指向的方向是否可以通过。

实例 16　智力魔方人机界面

实例说明

本例编写一个魔方类型的小游戏。游戏运行之后，效果如图 16-1 所示。

开始游戏前，窗口中的小球都是很规则地排列着，每列的小球图案相同。单击“开始新游戏”按钮之后，小球的位置被打乱，随机地出现在每一位置上。玩家可以单击“↑”、“↓”、“←”、“→”的按钮来移动每一行或列。移动一行（或列）时，要触动 5 个小球，可能为了把一个球放到合适位置，而改变了其他球的位置。当玩家把相同图案小球全部移动到同一行或者列的时候，游戏就成功了。

本例主要介绍了窗体中控件的布置以及对一些公共变量的说明。

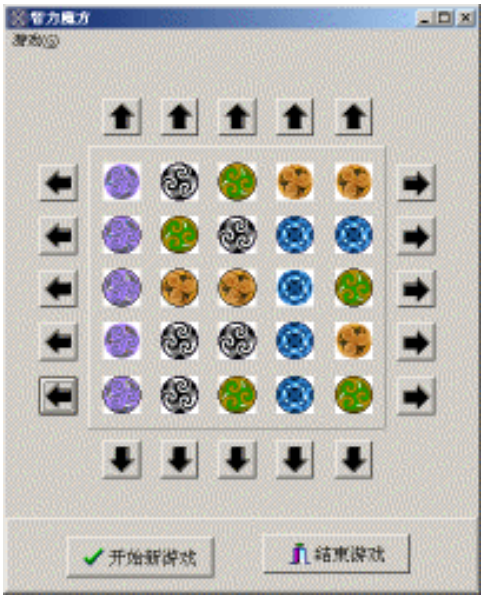


图 16-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。在本实例中利用常见的 Image 控件和 BitBtn 控件制作了一个精美的程序界面。使用 BitBtn 控件而不用 Button 控件，就是利用该控件的 Glyph 属性，向 BitBtn 控件中添加按钮图形。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。首先向其中添加一个 Panel 控件，在该控件中添加 25（5×5）个 Image 控件。然后在每一行或者列的两端添加两个 BitBtn 控件，再向窗体中添加第二个 Panel 控件，在该控件中添加两个 BitBtn 控件，最后向窗体中添加一个 MainMenu 控件。

2．设置窗体的关键属性如下：

BorderStyle = bsSingle
Caption = '智力魔方'
ClientHeight = 458
ClientWidth = 400

Menu = MainMenu1

3．设置 MainMenu 控件的菜单如图 16-2 所示。

图　16-2

4．设置第一个 Panel 1 控件的关键属性如下：

Left = 69

Top = 80

Width = 253

Height = 240

TabOrder = 2

5．设置其中第一个 Image 1 控件的关键属性如下，其他 Image 控件的属性和该控件的类似，这里不再介绍。

Left = 13

Top = 14

Width = 32

Height = 32

6．设置第一个 BitBtn 1 控件（该控件是主窗口左面一列按钮的第一个）的关键属性如下：

Tag = 11

Left = 28

Top = 95

Width = 32

Height = 32

TabOrder = 1

OnClick = DirClick

选中该按钮，然后在对象监视器中单击 Glyph 属性后面的…按钮，将弹出 Picture Editor 对话框，单击其中的 Load 按钮，然后选择一个箭头作为按钮的图标，如图 16-3 所示。其他按钮的属性设置和它类似，这里不再介绍。

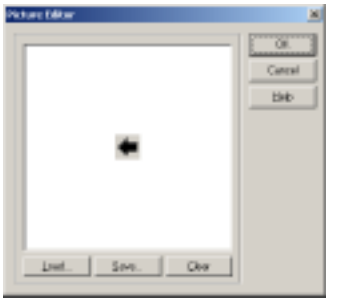


图　16-3

7．另外 Panel2 及其内部的 BitBtn 控件的布置如图 16-1 所示，这里不再介绍。

8．本实例中定义了多个变量和过程，各过程的含义如下：

```
private
   strCurPath: string;                    // 当前可执行文件路径
   nBlock: array [0..4,0..4] of integer;  // 25 个位置上的小球值
   picBall: array [0..4] of TPicture;     // 5 个小球
   bGameStart: boolean;                   // 游戏是否开始
   procedure MyRandom;                    // 产生随机数
   procedure MoveIcon(nFlag: integer);    // 移动小球
   function JudgeWin:boolean;             // 判断是否成功
public
end;
```

9．双击用来移动小球的按钮（任何一个），然后将其事件名称改为 DirClick，向其中添加如下代码：

```
procedure TForm1.DirClick(Sender: TObject);
begin
  if bGameStart then
  begin
    MoveIcon(TBitBtn(Sender).Tag);          // 移动小球
    if JudgeWin then          // 判断是否结束游戏,所有的行或者所有的列的小球相同
    begin
      Application.MessageBox('你太厉害了!我玩了 3 个小时也没有玩出来!','游戏结束',MB_OK);
      bGameStart := false;
    end;
  end;
end;
```

10．在其他移动小球按钮的 OnClick 事件的下拉菜单项中都选择 DirClick。

实例 17　智力魔方组件

实例说明

续前面的实例，本例主要讲述程序代码的实现，如图 17-1 所示。

首先利用 Myrandom（）函数产生 25 个随机数，布置 25 个小球的位置。单击“开始游戏”按钮时，调用 Myrandom（）函数，等待四周按钮的响应，利用上一个实例创建的 DirClick 函数移动小球。

图 17-1　效果图

编程思路

本实例代码部分的基本编程思路如下：在窗口的 OnShow 事件中显示小球。当玩家单击“开始新游戏”按钮时，首先调用 MyRandom 过程产生 25 个随机数，然后利用这 25 个数随机地布置小球的位置，接下来创建了 DirClick 过程，该过程用来响应移动小球的按钮，以达到移动指定行或者列中的小球的目的。

创作步骤

程序设计步骤续前面的实例。本实例的源代码如下（这里只列出了关键代码，其他代码详见配套光盘）：

```
unit Unit1;
interface
uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,ExtCtrls, Menus, StdCtrls,
    Buttons;
type
  TForm1 = class(TForm)
……（此处代码略，详见光盘）
  private
    strCurPath: string;                    // 当前可执行文件路径
    nBlock: array [0..4,0..4] of integer;  // 25 个位置上的小球值
    picBall: array [0..4] of TPicture;     // 5 个小球
    bGameStart: boolean;                   // 游戏是否开始
    procedure MyRandom;                    // 产生随机数
```

```
    procedure MoveIcon(nFlag: integer);        // 移动小球
    function JudgeWin:boolean;                 // 判断是否成功
  public
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.MyRandom;      // 产生 25 个不重复随机数,方法同拼图游戏产生随机数算法
……（此处代码略，详见光盘）
end;
// 根据按钮的 Tag 格式移动小球,其 tag 格式为 xy
procedure TForm1.MoveIcon(nFlag: integer);
var nDirFlag, nLine, nFlag2, nTmpValue, cx: integer;
begin
  if nFlag > 0 then       // 如果是正,则是向上或者向左移
    nDirFlag := 1
  else
……（此处代码略，详见光盘）
  begin
    MoveIcon(TBitBtn(Sender).Tag);    // 移动小球
    if JudgeWin then                  // 判断是否结束游戏,所有的行或者所有的列的小球相同
    begin
      Application.MessageBox('你太厉害了!我玩了 3 个小时也没有玩出来!','游戏结束',MB_OK);
      bGameStart := false;
    end;
  end;
end;
procedure TForm1.MenuNewGameClick(Sender: TObject);
var cx,cy: integer;
begin
  bGameStart := true;
  MyRandom;             // 打乱小球排列顺序
  for cy := 0 to 4 do
    for cx := 0 to 4 do          // 显示打乱以后的小球
        TImage(FindComponent('Image'+IntToStr(cx+1)+IntToStr(cy+1))).Canvas.Draw(0,0,picBall[nBlo
     ck[cx][cy] mod 5].Icon);
end;
```

```
procedure TForm1.FormShow(Sender: TObject);
var cx,cy: integer;
begin
   bGameStart := false;
   strCurPath := ExtractFilePath(Application.ExeName);
   for cx := 0 to 4 do
   begin
      picBall[cx] := TPicture.Create;
      picBall[cx].Icon.LoadFromFile(strCurPath+'RES\Ball'+IntToStr(cx+1)+'.ico');
   end;
   for cy := 0 to 4 do
      for cx := 0 to 4 do            // 初始化,显示小球
TImage(FindComponent('Image'+IntToStr(cx+1)+IntToStr(cy+1))).Canvas.Draw(0,0,picBall[cy].Icon);
end;
function TForm1.JudgeWin:boolean;
var cx,cy: integer;
begin
   Result := TRUE;
   for cx := 0 to 4 do
      for cy := 0 to 4 do            // 如果所有的行或者所有的列小球图案相同,则为赢,否则为输
         if (nBlock[cx,cy] <> cx*5+cy) and (nBlock[cx,cy] <> cy*5+cx) then
         begin
            Result := false;
            exit;
         end;
end;
procedure TForm1.BitBtn21Click(Sender: TObject);
begin
   MenuNewGame.Click;
end;
procedure TForm1.BitBtn22Click(Sender: TObject);
begin
   Close;
end;
end.
```

实例 18　拼图人机界面

实例说明

图 18-1　效果图

本例编写拼图游戏。如图 18-1 所示。

单击“调入图像”按钮，将会导入一张图片，然后单击“重新开始”按钮，将把图片分成九份，重新打乱。玩家可以使用“←”、“↑”、“→”和“↓”键来移动小图片，直到将九张小图片拼成一张大图片。在窗口的下方记录了玩家所移动的步数。

在本实例中将主要讲述程序的窗体设计以及 Myrandom()函数的实现方法。

编程思路

本实例利用 C++ Builder 6.0 来实现。首先设计程序的窗体部分。接下来是 Myrandom 函数的实现方法。它的原理如下：

Myrandom 函数的算法是按顺序从 0 到 7，找这八个数的位置。先利用普通的方法产生一个随机数，比如产生一个 15，然后把该随机数除以还未产生的随机数数目取余，比如已经产生了两个随机数，其中 0 在位置 7，1 在位置 2（这里一个位置对应于 nRandom 数组中的一个元素）。还有六个随机数未产生，因为已经产生了 0 和 1，现在需要产生 2，于是用 15%6 = 3，也就是要从剩下的 0，1，3，4，5，6 这六个位置中找到第四个位置（因为从 0 开始，随机数 3 表示第四个数），然后把该位置填充成 2。接下来开始从第一个位置开始找，找到 0 位置时，发现该位置上数值为-1（说明了该位置还没有被填充上随机数），所以计数器加 1（1）（计数器表示剩下的还没有产生随机数的位置中，从最小的开始记 空的位置 的数量），于是继续找下一个，下一个位置 1 的数值为–1，表示该位置还没有被填充上一个随机数，计数器被加 1（2）。继续找下一个，下一个位置 2 的数值是 1，说明了该位置已经有了随机数 1，继续找下一个，下一个位置 3 的数值是-1，说明了该位置还没有随机数，计数器加 1。这样继续下去，一直找到位置 4 的数值还是–1，计数器被加+1（4）。这时，已经在剩下的 6 个位置的位置 4 的地方记数到了 4，也就是找到了第四个空位，于是该位置的值 nRandom[4] = 2。这样一直继续下去，一直到产生出所有的八个位置为止。

创作步骤

1．启动 C++ Builder，打开一个新的标准工程。如果 C++ Builder 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加八个 Panel 控件、三个 SpeedButton 控件、一个 OpenDialog 控件和一个 Image 控件。具体的布置是，在每个 SpeedButton 控件下面放置一个 Panel 控件；

图 18-1 中窗体左下方放置三个 Panel 控件；在窗体的上方放置两个 Panel 控件和一个 Image 控件。

2．本实例使用了多个公共变量和公共函数，在头文件中声明如下：

```
private:       //    User declarations
        TPanel *MyPanel[9];     //  可移动的九个块，动态产生
        TImage *MyImage[9];     //  用来显示九个图像的 Image 对象，动态产生
        TRect rectSrc[9];       //  定义背景图的九个块的矩形坐标值，用来把背景图分割成为九块
        int nBlock[3][3];       //  记录 cx,cy 位置的块的编号
        TPoint ptCurNull;       //  当前的空块坐标
        Graphics::TBitmap *bmpSrc;//  背景图对象
        int nTotalStep;         //  当前总的步骤数
        bool bInitOK;           //  判断初始化是否完成，0 表示未完成，1 表示完成
        void CalculateRect();   //  计算背景图被分割以后的九个矩形的坐标
        void InitImage();       //  初始化 Image 图像
        void MyRandom(int nValue);  //  产生随机数，用来把各块随机分配到相应的位置
        int JudgeFinish();          //  判断移动是否已经完成
public:                 //    User declarations
            __fastcall TForm1(TComponent* Owner);
};
```

3．用来产生随机数的函数 myrandom 的代码如下：

```
void TForm1::MyRandom(int nValue)
{
   srand(time(NULL));          //  利用当前的时间初始化随机数
   int nRandom[8];             //  用来保存 0～7 这八个数，八个数是由下面的函数随机放置的
   for (int j=0;j<8;j++)
   {
      nRandom[j] = -1;
   }
   for (int j=0;j<8;j++)
   {
      int nCount = 0;              //  用来记录当前的已经产生随机数的位置
      int nTmp = rand()%(8-j);       //  产生一个 8-i 以内的随机数
      for (int i=0;i<8;i++)
      {
         if (nRandom[i] == -1)
         {
            nCount++;
            //  从 0 到 7 开始查找 nRandom[i]，如果当前的值为-1（也就是还没有在该位置产生随机数），
      则计数器加 1
            if (nCount == nTmp+1)          //  如果计数器已经到了执行下面语句
```

```
                {
                    nRandom[i] = j;
                    break;
                }
            }
        }
    }
    for (int cx=0;cx<3;cx++)
        for (int cy=0;cy<3;cy++)
            nBlock[cx][cy] = nRandom[cy*3+cx];          // 在初始化八个块的位置上产生的八个随机数
    nBlock[2][2] = -1;
}
```

实例 19　拼 图 组 件

实例说明

续前面的实例，本例主要讲述程序代码的实现。如图 19-1 所示。

本例通过响应玩家的按键“→”、“←”、“↑”和“↓”来移动图片。在移动之前先要判断图片要移动的位置是否已经有了图片，如果是，则取消玩家的操作；反之，将图片移动到空位置。在每次玩家移动完图片之后，判断是否所有的块都被放到正确的位置，如果是，则显示成功的提示。

图 19-1　效果图

编程思路

本实例在程序的初始化阶段对图片进行分割，将其分割成 3×3 的九个部分。然后响应用户的按键操作以实现交互功能。最后在程序的销毁阶段释放变量和内存。

创作步骤

程序设计步骤续前面的实例。

1．程序的 OnKeyDown 事件响应玩家的按键，其代码如下：

```
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
    if (bInitOK)          // 初始化成功才对按键进行处理
    {
        switch (Key)
        {
            case VK_UP:     // 如果按键是方向键
                if ( (ptCurNull.y+1 <= 2))
                // 如果空白处允许块上移，也就是它下边的位置是还在 3×3 的矩阵范围内
                {
                    nBlock[ptCurNull.x][ptCurNull.y] = nBlock[ptCurNull.x][ptCurNull.y+1];
                // 空白块被其下边的一个块填充上，所以空白块的数字被置成它下边的一个块的数
```

值

```
                nBlock[ptCurNull.x][ptCurNull.y+1] = -1;
                // 原空白块下边的一个块被上移，所以它的值被置-1，表示它成了空白块
                ptCurNull.y += 1;     // 空白块相应向下移，所以 y 坐标+1
                MyPanel[nBlock[ptCurNull.x][ptCurNull.y-1]]->Top -= 102;
                // 显示的 Panel 对象上移一格
                nTotalStep++;          // 步数加 1
                if (JudgeFinish())
                    Panel4->Caption = "你成功了！";
                else
                    Panel4->Caption = "加油！";
                    // 判断是否所有的块都被放到正确的位置，如果是，则显示成功的提示
            }
            break;
        case VK_DOWN:       // 如果按键是方向键下
            if ( (ptCurNull.y-1 >= 0))
            // 如果空白处允许块下移，也就是它上边的位置还是在 3×3 的矩阵范围内
            {
              nBlock[ptCurNull.x][ptCurNull.y] = nBlock[ptCurNull.x][ptCurNull.y-1];
               // 空白块被其上边的一个块填充上，所以空白块的数字被置成它上边的一个块
                的数值
              nBlock[ptCurNull.x][ptCurNull.y-1] =–1;
                // 原空白块上边的一个块被下移，所以它的值被置–1，表示它成了空白块
              ptCurNull.y -= 1;     // 空白块相应向上移，所以 y 坐标–1
              MyPanel[nBlock[ptCurNull.x][ptCurNull.y+1]]->Top += 102;
                // 显示的 Panel 对象下移一格
              nTotalStep++;          // 步数加 1
              if (JudgeFinish())
                  Panel4->Caption = "你成功了！";
              else
                  Panel4->Caption = "加油！";
                  // 判断是否所有的块都被放到正确的位置，如果是，则显示成功的提示
            }
            break;
        case VK_LEFT:
            if ( (ptCurNull.x+1 <= 2))
            {
              nBlock[ptCurNull.x][ptCurNull.y] = nBlock[ptCurNull.x+1][ptCurNull.y];
              nBlock[ptCurNull.x+1][ptCurNull.y] = –1;
```

```
                    ptCurNull.x += 1;
                    MyPanel[nBlock[ptCurNull.x-1][ptCurNull.y]]->Left -= 102;
                    nTotalStep++;
                    if (JudgeFinish())
                        Panel4->Caption = "你成功了！";
                    else
                        Panel4->Caption = "加油！";
                }
                break;
            case VK_RIGHT:
                if ( (ptCurNull.x-1 >= 0))
                {
                    nBlock[ptCurNull.x][ptCurNull.y] = nBlock[ptCurNull.x-1][ptCurNull.y];
                    nBlock[ptCurNull.x-1][ptCurNull.y] = -1;
                    ptCurNull.x -= 1;
                    MyPanel[nBlock[ptCurNull.x+1][ptCurNull.y]]->Left += 102;
                    nTotalStep++;
                    if (JudgeFinish())
                        Panel4->Caption = "你成功了！";
                    else
                        Panel4->Caption = "加油！";
                }
                break;
        }
        Panel3->Caption = "移动步数："+IntToStr(nTotalStep);
        // 提示已经移动的步数
    }
}
```

2．在窗体的 OnCreate **事件中对图像进行分割，其代码如下：**

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    Panel3->Caption = "移动步数：0";
    Panel4->Caption = "";
    for (int cy=0;cy<3;cy++)
        for (int cx=0;cx<3;cx++)
        {
            MyPanel[cy*3+cx] = new TPanel(this);
            MyPanel[cy*3+cx]->Name = "MyPanel"+IntToStr(cy*3+cx+1);
            MyPanel[cy*3+cx]->Parent = Panel1;
```

```
            MyPanel[cy*3+cx]->Left = cx*102;
            MyPanel[cy*3+cx]->Top = cy*102;
            }// 该 for 循环用来动态产生九个 Panel 对象，同时指定相应的 Panel 在窗口中的位置以及父窗口的位置
    for (int i=0;i<9;i++)
    {
        MyPanel[i]->Width = 100;
        MyPanel[i]->Height = 100;
        MyPanel[i]->Caption = "";        // 指定九个 Panel 的宽、高，以及设置其标题为空
        MyImage[i] = new TImage(this);
        MyImage[i]->Parent = MyPanel[i];
        MyImage[i]->Top = 2;
        MyImage[i]->Left = 2;
        MyImage[i]->Height = 96;
            MyImage[i]->Width = 96;
          // 动态产生九个 Image 对象，将其父窗口指定为响应的九个 Panel，并指定其位置和宽高
    }
    MyPanel[8]->Left += 102;         // 把最下脚(2,2)的 Panel 位置右移
    try
    {
        bmpSrc = new Graphics::TBitmap;
        bmpSrc->LoadFromFile(ExtractFilePath(Application->ExeName)+"01.bmp");
        Image1->Canvas->StretchDraw(TRect(0,0,Image1->Width,Image1->Height),bmpSrc);
    }
    catch (...)
    {
            Application->MessageBox("打开初始图像失败，请单击调入图像指定背景图案","警告",MB_OK);
        bInitOK = false;
        return;
        }// 尝试打开默认的该可执行程序所在目录的 bg.bmp 作为背景文件，如果打开失败的话则初始化失败，同时返回
     // 如果初始化成功，则在右边的 Image 对象上显示出该图像的缩放图
    InitImage();         // 如果初始化成功，则初始化动态产生九个 Image 对象的图像内容
    bInitOK = true;
}
```

3．利用 InitImage 函数初始化 Image 对象的图像内容，其代码如下：

```
void TForm1::InitImage()
{
```

```
        TRect rectTmp = TRect(0,0,96,96);          // Image 对象的大小
        CalculateRect();                 // 计算九个 Image 在原图上对应的矩形块
        for (int cy=0;cy<3;cy++)
            for (int cx=0;cx<3;cx++)
            {
                MyImage[cy*3+cx]->Canvas->CopyMode = cmSrcCopy;
                MyImage[cy*3+cx]->Canvas->CopyRect(rectTmp,bmpSrc->Canvas,rectSrc[cy*3+cx]);
            }// 该 for 循环把原背景图的九个相应的块拷贝到对应的 Image 的画布上
    }
```

4．销毁窗口时，释放资源，其代码如下：

```
    void __fastcall TForm1::FormDestroy(TObject *Sender)
    {
        for (int i=0;i<9;i++)
        {
            delete MyImage[i];
            delete MyPanel[i];
        }
        delete bmpSrc;          // 在 Form 销毁的时候，删除所产生的 Image、Panel 以及 Bitmap 对象，释放资源
    }
```

其他源代码详见光盘。

实例 20　趣 味 绘 图

实例说明

本例制作一个利用鼠标绘图的小游戏。效果如图 20-1 **所示。**

本程序运行后可以直接用鼠标绘制直线、圆及椭圆等曲线。

通过本程序可以学会通过 VB **编写绘图程序的知识及处理鼠标事件的知识。**

图 20-1　**效果图**

编程思路

Visual Basic **提供了许多图形处理控件，但有时还需自己绘制一些图形，这就需要用到** VB **的绘图方法来实现。本例讲解几种常用的绘图方法。**

Cls：**清除所有图形和** Print **输出。**

Pset：**设置各个像素的颜色。**

Line：**画线、矩形或填充框。**

Circle：**画圆、椭圆或圆弧。**

Print：**显示一个字符串。**

创作步骤

1．**启动** Visual Basic 6.0，**打开一个新的标准工程。如果** Visual Basic 6.0 **已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程。**

2．**在新窗体（**Form1**）中创建一个** Label **标签，六个** Command **命令按钮。整个窗体界面如图** 20-2 **所示。**

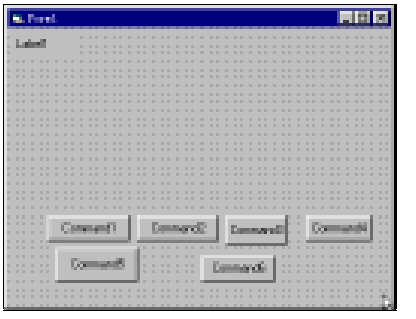


图　20-2

3．设置窗体及各个控件的属性。其效果图如图 20-3 所示。

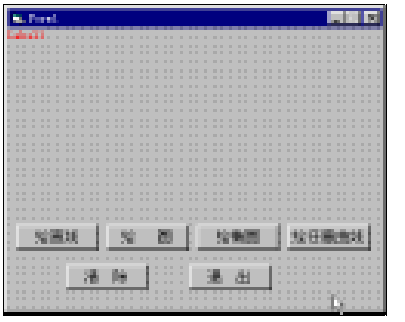


图　20-3

各控件主要属性设置如下：

Label1　Name　=　Label1
　　Caption　=　“Label1”
　　ForeClolor　=　&H000000FF& ‘红色

Command1　Name　=　Command1
　　Caption　=　”绘直线”
　　Font　=　“宋体”（字号为 5 号字）

Command2　Name　=　Command2
　　Caption　=　”绘　圆”
　　Font　=　“宋体”（字号为 5 号字）

Command3　Name　=　Command3
　　Caption　=　”绘椭圆”
　　Font　=　“宋体”（字号为 5 号字）

Command4　Name　=　Command4
　　Caption　=　”绘任意曲线”
　　Font　=　“宋体”（字号为 5 号字）

Command5　Name　=　Command5
　　Caption　=　”清除”
　　Font　=　“宋体”（字号为 5 号字）

Command6　Name　=　Command6
　　Caption　=　”退出”
　　Font　=　“宋体”（字号为 5 号字）

4．程序中最重要的是处理鼠标的事件，即拖动和按下鼠标键。

源代码详见光盘。

实例 21　点灯人机交互

实例说明

本例编写一个挑战智力的游戏，如图 21-1 所示。

开始游戏后，整个画面显现为灰色，表示所有的灯都是关闭的。玩家可以单击窗口中的任何一处，使单击处以及其四周的灯变成和原来相反的状态（也就是，关闭的变成打开的，打开的变成关闭的）。如果玩家把画面中所有的灯都打开，则玩家胜利。

本例主要介绍了程序的窗体设计以及程序所使用的公共变量和公共过程。

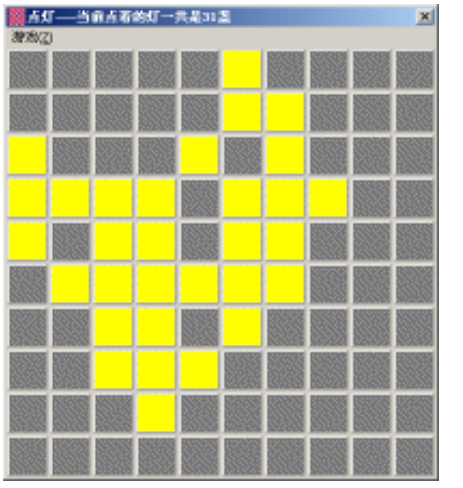


图 21-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。在本实例中介绍了窗体的属性设置，将其 BorderStyle 属性设置成了 bsSingle，表示程序运行后窗体的大小不能改变；将其 BorderIcons 属性设置为 biSystemMenu，表示程序的右上方标题栏上只有一个关闭按钮。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。在新建的界面上加入一个 MainMenu 控件。

2．设置窗体的关键属性如下：

Left = 192

Top = 107

BorderIcons = [biSystemMenu]

BorderStyle = bsSingle

Caption = '点灯'

ClientHeight = 381

ClientWidth = 400

Color = clBtnFace

Menu = MainMenu1

OldCreateOrder = False

OnCreate = FormCreate

OnDestroy = FormDestroy

PixelsPerInch = 96

TextHeight = 13

3．双击 MainMenu 控件，向其中添加菜单项如图 21-2 所示。

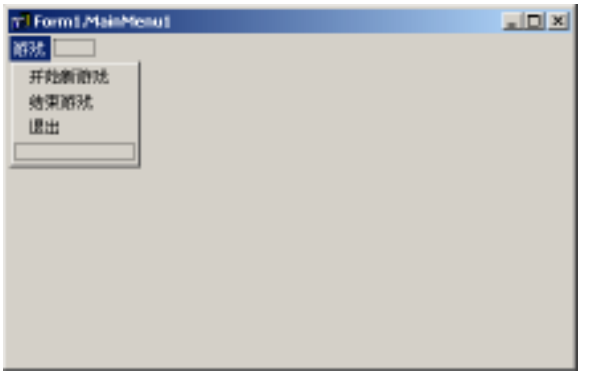


图 21-2

4．本程序中定义了四个公共变量，它们的作用如下：

bGameStart: boolean; // 判断游戏的状态:1 表示开始;0 表示停止

nTotalBright: integer; // 总共被点亮的灯的数量

MyPanel: array [0..9,0..9] of TPanel; // 100 个 Panel 对象

nState: array [0..9,0..9] of integer; // 每一个 Panel 对应一个状态,表示它当前是灭(0)还是亮(1)

5．本程序定义了三个公共过程，它们的功能如下：

procedure MyPanelClick(Sender: TObject); // Panel 对象的 Click 事件

procedure ChangeColor(X,Y: integer); // 更改 X,Y 点的 Panel 的颜色,并把相应的 nState 变反

procedure JudgeWin; // 判断 100 个灯是否全部被点亮

实例 22　点 灯 组 件

实例说明

续前面的实例。本例主要讲述程序的代码实现，如图 22-1 所示。

通过一个 ChangeColor 函数来不断改变 Panel 控件中各相关部分的颜色，然后当玩家单击鼠标时，判断其单击的位置，如果是在窗体的中部，则利用 ChangeColor 函数改变其四周 Panel 控件的颜色，如果单击的是窗体边界上的灯，则只改变窗体内部的灯的状态。

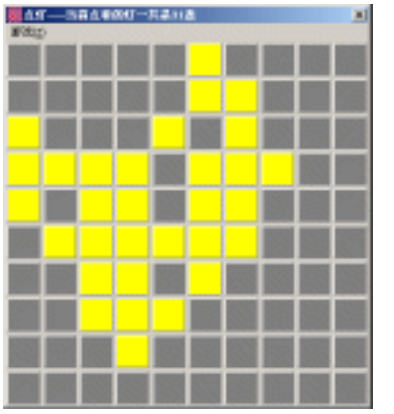


图 22-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。在窗体中只放置一个 Mainmenu 控件，用来控制游戏的进行。当程序运行后，将创建 100 个 Panel 控件（10×10），用来表示 100 盏灯，开始时它们都是关闭的（灰色）。当用户单击“开始新游戏”按钮后，光标变为小手形状，表示可以点灯了。当用户单击窗体中的 Panel 控件时，通过一个 nState 变量来改变每盏灯的状态，同时把 Panel 控件变为黄色。

创作步骤

程序设计步骤续前面的实例。本实例源代码如下：

```
unit Unit1;
interface
uses
……（此处代码略，详见光盘）
    bGameStart: boolean;     // 判断游戏的状态:1 表示开始;0 表示停止
    nTotalBright: integer;   // 总共被点亮的灯的数量
    MyPanel: array [0..9,0..9] of TPanel;        // 100 个 Panel 对象
    nState: array [0..9,0..9] of integer;        // 每一个 Panel 对应一个状态,表示它当前是灭(0)还是亮
(1)
    procedure MyPanelClick(Sender: TObject);     // Panel 对象的 Click 事件
    procedure ChangeColor(X,Y: integer);         // 更改 X,Y 点的 Panel 的颜色,并把相应的 nState 变反
    procedure JudgeWin;                          // 判断 100 个灯是否全部被点亮
```

```
  public
    { Public declarations }
  end;
……（此处代码略，详见光盘）
// 简化思路,把四条边和四个角分别进行处理
procedure TForm1.MyPanelClick(Sender: TObject);
var
  nMyTag,nCurX,nCurY: integer;
begin
  nMyTag := TPanel(Sender).Tag - 1000;        // 利用这 100 个 Panel 的 Tag 来区分每一个 Panel
  nCurX := nMyTag mod 10;                     // 根据每一个 Tag 计算该 Panel 的坐标
  nCurY := nMyTag div 10;
  if nMyTag in [1..8] then                    // 对应上面的边,其左、右、下颜色发生变化
  begin
……（此处代码略，详见光盘）
  ChangeColor(nCurX,nCurY);     // 自身颜色发生变化
  JudgeWin();
end;
// 更改 X,Y 点的 Panel 的颜色,并把相应的 nState 变反
procedure TForm1.ChangeColor(X,Y: integer);
begin
  if nState[X,Y] = 1 then
  begin
    nState[X,Y] := 0;
    MyPanel[X,Y].Color := clGray;
  end
  else begin
    nState[X,Y] := 1;
    MyPanel[X,Y].Color := clYellow;
  end;
end;
// 判断 100 个灯是否全部被点亮
procedure TForm1.JudgeWin;
var
  cx,cy: integer;
begin
  nTotalBright := 0;
  for cy := 0 to 9 do
    for cx := 0 to 9 do
```

```
            begin
              if nState[cx,cy] = 1 then
                nTotalBright := nTotalBright + 1;      // 循环计算当前被点亮的灯的数量
            end;
        Caption := '点灯——当前点着的灯一共是'+IntToStr(nTotalBright)+'盏';
        if nTotalBright = 100 then
        begin
          Application.MessageBox('你太厉害了,简直是人中龙凤啊!','佩服佩服!',MB_OK);
          bGameStart := false;
        end;
      end;
      procedure TForm1.MenuEndGameClick(Sender: TObject); // 停止游戏
      var
        cx,cy: integer;
      begin
        Application.MessageBox(PChar('不错不错'#13'你一共点了'+IntToStr(nTotalBright)+'盏灯'#13'继续努力
吧'),'恭喜',MB_OK);
      ……（此处代码略，详见光盘）
      procedure TForm1.MenuNewGameClick(Sender: TObject);       // 开始游戏
      var
        cx,cy: integer;
      begin
        bGameStart := true;
        Cursor := crHandPoint;
        for cy := 0 to 9 do
          for cx := 0 to 9 do
          begin
            nState[cx,cy] := 0;
            MyPanel[cx,cy].Cursor := crHandPoint;
            MyPanel[cx,cy].OnClick := MyPanelClick;
          end;
      end;
      procedure TForm1.MenuExitClick(Sender: TObject);        // 退出游戏
      begin
        Close;
      end;
      end.
```

实例 23　单人跳棋人机交互

实例说明

本例编写单人跳棋的游戏，游戏画面如图 23-1 所示。

程序运行之后，玩家可以拖动小球，正确的移动方法是在水平或者竖直方向跳过一个小球，同时被移动的小球的最终位置是一个空位，在这种情况下，被越过的小球将被移走。接下来玩家依照同样的规则再移动其他小球，直到不能移动为止，最后剩下一个小球即为胜利。程序还有“撤销”以及“恢复”等功能，以方便玩家思考。

在本实例中主要讲述程序的窗体设计。在窗体中使用了 Image 控件、Shape 控件以及菜单。

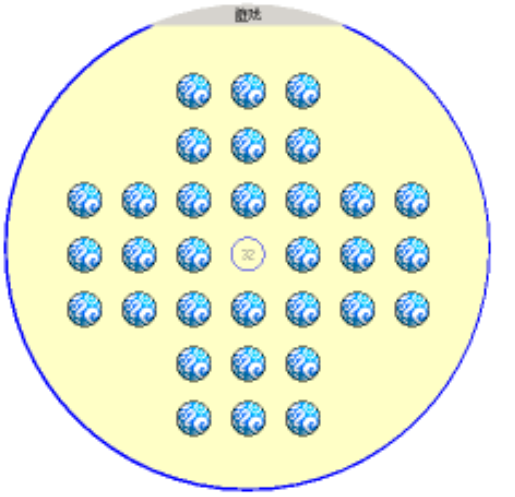


图 23-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。主要使用了 Image 控件和 Shape 控件，通过改变 Shape 控件的 Shape 属性可以在窗体中显示不同的图形，包括圆形、矩形等。利用 VB 的菜单编辑器可以快速地制作出专业的程序菜单，在本实例的第 4 步有详细介绍。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加两个 Image 控件、一个 Label 控件和一个 Shape 控件。如图 23-2 所示。

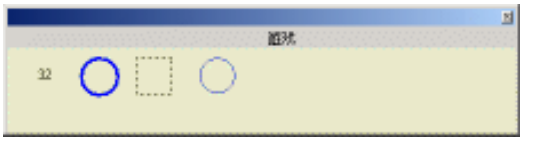


图　23-2

2．设置两个 Image 控件的关键属性如下：

Begin VB.Image imgHole

```
    Height = 480
    Index = 0
    Left = 2520
    Picture = "SolitaireRound.frx":08CA
    Top = 120
    Visible = 0       'False
    Width = 480
  End
  Begin VB.Image imgMarble
     DragMode = 1    'Automatic
     Height = 480
     Index = 0
     Left = 1680
     Top = 120
     Width = 480
  End
```

3．**设置** Shape **控件的关键属性如下：**

```
  Begin VB.Shape shpBorder
    BorderColor = &H00FF0000&
    BorderWidth = 3
    Height = 495
    Left = 960
    Shape = 3    'Circle
    Top = 120
    Width = 495
  End
```

4．**单击工具栏上的菜单编辑器按钮，向其中添加一个菜单，如图** 23-3 **所示。**

图　23-3

实例 24　单人跳棋组件

实例说明

续前面的实例，本例主要讲述程序的代码实现。如图 24-1 所示。

通过响应玩家的鼠标拖动，来移动棋子，如果符合游戏规则，则跳过中间的棋子，并将其拿出棋盘；反之则此次操作无效。

在程序中创建了一个 NewGame()函数，通过该函数可以使所有的棋子显示在棋盘中。当每次玩家重新开始游戏时，都要初始化变量。

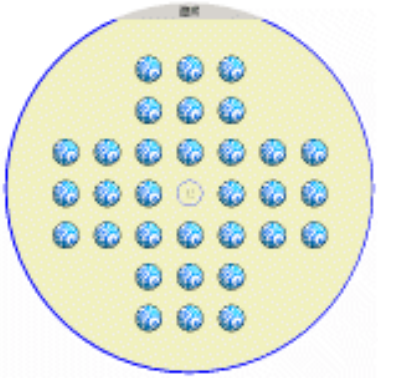


图 24-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。程序运行后，利用 Windows API 函数来实现圆形窗体，接下来响应鼠标的 MouseDown 和 MouseMove 事件，实现对小球的拖动。然后判断玩家拖动的位置是否符合规则。如果是，则将被跳过的小球的 Visible 属性变为 False，实现对小球的隐藏；如果不是，则取消玩家对小球的操作。

创作步骤

程序设计步骤续前面的实例。本实例源代码如下：

```
Option Explicit
Private m_iDragIndex As Integer
Private m_iMarblesLeft As Integer
Private m_iOldMovesIndex As Integer
Private m_vOldMoves(1 To 2, 1 To 35) As Integer
Private Const SIZE As Integer = 7
Private Const HOLE_WIDTH As Integer = 50
Private Const BORDER_TOP As Integer = -19
Private Const BORDER_LEFT As Integer = 17
Private Const BORDER_DIAMETER As Integer = 445
Private Const FORM_DIAMETER As Integer = 7000
Private Const TEXT_SOLITAIRE As String = "单人跳棋"
```

```
Private Const TEXT_INSTRUCTIONS As String = "单人跳棋游戏规则"
Private Const TEXT_INSTRUCTION_1 As String = "尽量多的移去小球"
Private Const TEXT_INSTRUCTION_2 As String = "合法的移动是在水平或者竖直方向跳过一个小球"
Private Const TEXT_INSTRUCTION_3 As String = "被跳过的小球将被移走"
Private Const TEXT_INSTRUCTION_4 As String = "将一个小球跳过另一个小球，并放到一个空地上"
Private Const TEXT_INSTRUCTION_5 As String = "被跳过的小球将被移走"
Private Const TEXT_INSTRUCTION_6 As String = "最后剩下的小球越少越好"
Private Const TEXT_INSTRUCTION_7 As String = "最开始只有一个空位"
Private Declare Function CreateEllipticRgn Lib "gdi32" _
    (ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
Private Declare Function SetWindowRgn Lib "user32" _
    (ByVal hwnd As Long, ByVal hRgn As Long, ByVal bRedraw As Long) As Long
Private Static Sub Form_Load()
    Dim i%, j%
    Dim hr&, dl&
    Dim usew&, useh&
    '创建圆形窗体
    Me.Width = FORM_DIAMETER
    Me.Height = FORM_DIAMETER
    usew& = Me.Width / Screen.TwipsPerPixelX
    useh& = Me.Height / Screen.TwipsPerPixelY
    hr& = CreateEllipticRgn(20, 19, usew, useh)
    dl& = SetWindowRgn(Me.hwnd, hr, True)
    Me.Height = FORM_DIAMETER + 50
    Me.Width = FORM_DIAMETER + 50
    '设置边界属性
        shpBorder.Move BORDER_LEFT, BORDER_TOP, BORDER_DIAMETER,
            BORDER_DIAMETER
    lblMarblesLeft.Move 100, 70
    '导入图像
    ……（此处代码略，详见光盘）

Private Sub Undo()
'撤销操作
    If Not mnuUndo.Enabled Then Exit Sub
    m_iMarblesLeft = m_iMarblesLeft + 1
    lblMarblesLeft = m_iMarblesLeft
    imgMarble(m_vOldMoves(1, m_iOldMovesIndex)).Visible = True
    imgMarble((m_vOldMoves(1, m_iOldMovesIndex) + _
```

```
            m_vOldMoves(2, m_iOldMovesIndex)) / 2).Visible = True
        lblMarblesLeft.Move imgMarble(m_vOldMoves(2, m_iOldMovesIndex)).Left + 7, _
            imgMarble(m_vOldMoves(2, m_iOldMovesIndex)).Top + 9
        imgMarble(m_vOldMoves(2, m_iOldMovesIndex)).Visible = False
        m_iOldMovesIndex = m_iOldMovesIndex - 1
        If m_iOldMovesIndex = 0 Then mnuUndo.Enabled = False
        mnuRedo.Enabled = True
    End Sub

    Private Sub Redo()
        重复操作
        If Not mnuRedo.Enabled Then Exit Sub
        更新窗体信息
        m_iMarblesLeft = m_iMarblesLeft - 1
        lblMarblesLeft = m_iMarblesLeft
        更新小球的显示
        m_iOldMovesIndex = m_iOldMovesIndex + 1
        imgMarble(m_vOldMoves(1, m_iOldMovesIndex)).Visible = False
        imgMarble((m_vOldMoves(1, m_iOldMovesIndex) + _
            m_vOldMoves(2, m_iOldMovesIndex)) / 2).Visible = False
        imgMarble(m_vOldMoves(2, m_iOldMovesIndex)).Visible = True
        lblMarblesLeft.Move imgMarble((m_vOldMoves(1, m_iOldMovesIndex) + _
            m_vOldMoves(2, m_iOldMovesIndex)) / 2).Left + 7, _
            imgMarble((m_vOldMoves(1, m_iOldMovesIndex) + _
            m_vOldMoves(2, m_iOldMovesIndex)) / 2).Top + 9
        If m_vOldMoves(1, m_iOldMovesIndex + 1) = 0 Then mnuRedo.Enabled = False
        mnuUndo.Enabled = True
    End Sub
```

实例 25　小　迷　宫

实例说明

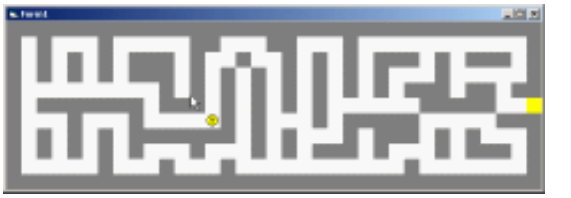


图 25-1　效果图

本例编写迷宫小游戏，如图 25-1 所示。

本例绘制了一个迷宫路径，玩家以一个笑脸图像显示。游戏的起点在左上角，终点在右方，玩家只需使用“↑”、“↓”、“←”、“→”箭头就可以控制头像的移动，将其移动到终点，并且所使用的时间越少越好。

编程思路

本实例使用 Visual Basic 6.0 来实现。把迷宫的地图记录在一个 dat 文件中，然后程序运行时，把图像显示出来。开始游戏后，程序响应玩家的按键，判断头像是否可以按照玩家的控制向指定的方向移动；当玩家把头像移动到目的地时，出现提示信息，告诉玩家已经成功地到达终点，并显示玩家所用的时间。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加一个 PictureBox 控件，如图 25-2 所示。

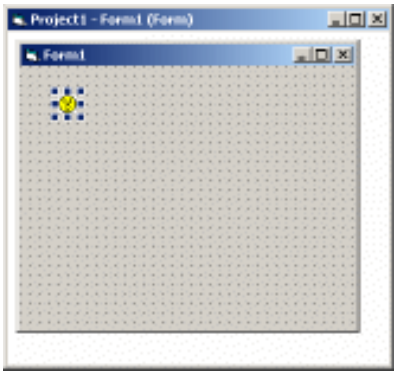


图　25-2

2．新建一个 dat 文件，使用记事本打开它，并且向其中输入如下文字：

```
11, 35
####################################
#s#    #       ##    ##    #              #
```

```
# # # # ### #   #   # # # ### # ### #
# # # # ### # # # # # # #    #       #
#            # # # # # # # # ### ## ##
# ####### ### # # # #        #####   f
#    #    #       # #    # #####       ###
# # # # ####### # #    # ### # #    #
# # # # #    #    # # ### #    # ### #
#       #    #    #             #         #
###################################
```

3．**设置** PictureBox **控件的关键属性如下：**

```
AutoSize = -1    'True
BorderStyle = 0    'None
Height = 225
Left = 480
Picture = "Form1.frx":0000
ScaleHeight = 225
ScaleWidth = 225
TabIndex = 0
Top = 360
Width = 225
```

4．**本实例源代码如下：**

```
Option Explicit
'迷宫信息
Private NumRows As Integer
Private NumCols As Integer
Private LegalMove() As Boolean
'正方形的面积
Private Const SQUARE_WID = 20
Private Const SQUARE_HGT = 20
'玩家的位置
Private PlayerR As Integer
Private PlayerC As Integer
'目标位置
Private RFinish As Integer
Private CFinish As Integer
Private StartTime As Single
'响应玩家的按键
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
Dim r As Integer
Dim c As Integer
    r = PlayerR
```

```
    c = PlayerC
    Select Case KeyCode
        Case vbKeyLeft
            c = PlayerC - 1
        Case vbKeyRight
            c = PlayerC + 1
        Case vbKeyDown
            r = PlayerR + 1
        Case vbKeyUp
            r = PlayerR - 1
        Case Else
            Exit Sub
    End Select
    If LegalMove(r, c) Then PositionPlayer r, c
End Sub
'初始化地图和玩家的位置
Private Sub Form_Load()
    ScaleMode = vbPixels
    AutoRedraw = True
    picPlayer.Visible = False

    '初始化地图
    LoadMaze
End Sub
'绘制迷宫
Private Sub DrawMaze()
Dim r As Integer
Dim c As Integer
Dim clr As Long
  Cls
        For r = 1 To NumRows
        For c = 1 To NumCols
            If LegalMove(r, c) Then
                If r = RFinish And c = CFinish Then
                    clr = vbYellow
                Else
                    clr = vbWhite
                End If
            Else
                clr = RGB(128, 128, 128)
            End If
```

```
                Line (c * SQUARE_WID, r * SQUARE_HGT)-Step(-SQUARE_WID, -SQUARE_HGT), clr,
BF
            Next c
        Next r
    End Sub
    ' 初始化迷宫
    Private Sub LoadMaze()
    Dim fnum As Integer
    Dim r As Integer
    Dim c As Integer
    Dim ch As String
    Dim row_info As String
        ' 打开迷宫信息文件
        fnum = FreeFile
        Open App.Path & "\maze.dat" For Input As #fnum
        ' 读取其中的数据
        Input #fnum, NumRows, NumCols
        ReDim LegalMove(1 To NumRows, 1 To NumCols)
        For r = 1 To NumRows
            Line Input #fnum, row_info
            For c = 1 To NumCols
                ch = Mid$(row_info, c, 1)
                LegalMove(r, c) = (ch <> "#")
                If LCase$(ch) = "s" Then
                    ' 这是起点
                    PlayerR = r
                    PlayerC = c
                ElseIf LCase$(ch) = "f" Then
                    ' 这是终点
                    RFinish = r
                    CFinish = c
                End If
            Next c
        Next r
        ' 关闭文件
        Close #fnum
        ' 改变窗口的大小
        Width = ScaleX(SQUARE_WID * NumCols, ScaleMode, vbTwips) + _
            Width - ScaleX(ScaleWidth, ScaleMode, vbTwips)
        Height = ScaleY(SQUARE_HGT * NumRows, ScaleMode, vbTwips) + _
            Height - ScaleY(ScaleHeight, ScaleMode, vbTwips)
```

```
    '绘制迷宫
    DrawMaze
    '定义玩家的位置
    PositionPlayer PlayerR, PlayerC
    '保存开始时的时间
    StartTime = Timer
End Sub
'绘制玩家头像
Private Sub PositionPlayer(r As Integer, c As Integer)
Dim x As Single
Dim y As Single
    '擦除玩家以前的头像
    If PlayerR > 0 Then
        x = (PlayerC - 1) * SQUARE_WID + (SQUARE_WID - picPlayer.Width) / 2
        y = (PlayerR - 1) * SQUARE_HGT + (SQUARE_HGT - picPlayer.Height) / 2
        Line (x - 1, y - 1)-Step(picPlayer.Width, picPlayer.Height), vbWhite, BF
    End If
    '移动头像
    PlayerR = r
    PlayerC = c
    '绘制头像
    x = (c - 1) * SQUARE_WID + (SQUARE_WID - picPlayer.Width) / 2
    y = (r - 1) * SQUARE_HGT + (SQUARE_HGT - picPlayer.Height) / 2
    PaintPicture picPlayer.Picture, x, y
    '判断玩家是否已经到达目的地
    If r = RFinish And c = CFinish Then
        If MsgBox("You finished in " & _
            Int(Timer - StartTime) & " seconds." & _
            vbCrLf & "Play again?", vbYesNo, _
            "Congratulations") = vbYes _
        Then
            Form_Load
        Else
            Unload Me
        End If
    End If
End Sub
```

第三篇

经典游戏

本篇总览

本篇讲解了多个经典游戏的制作方法，包括纸牌游戏、捡金豆、华容道、拼图、扫雷、数字魔方和贪吃蛇等游戏。

这些游戏有的在 Windows 的附件中出现过，有的在手机或者文曲星中出现过。

在游戏的编写过程中，通常需要对窗体界面进行布置，达到简洁美观。美工可以制作一些华丽的界面，以使玩家使用起来更舒适。

实例 26　纸牌人机交互

实例说明

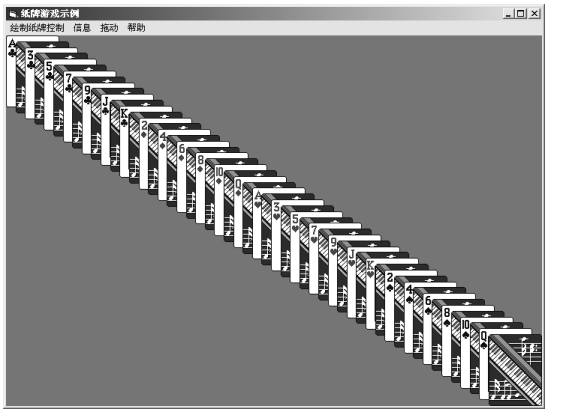


图 26-1　效果图

本例编写纸牌游戏，如图 26-1 所示。

在该实例中使用了 qcard32.dll 函数。程序演示了如何绘制纸牌、排列纸牌、查看纸牌背面，显示纸牌信息以及拖动纸牌等。在程序中附带了 qcard32.dll 文件的帮助，读者可以根据该帮助文件学习其他函数的使用方法，编写出更有意思的小游戏。

在本实例中主要讲述程序的窗体设计以及程序模块中所调用的方法。

编程思路

本实例使用 Visual Basic 6.0 来实现。利用 VB 中的窗体控件产生了一个 Windows 系统下标准的窗口。然后使用菜单编辑器制作了一个菜单，包括纸牌游戏的各个功能演示。利用 VB 的模块调用了 qcard32 中的各种方法。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程，向其中添加一个 Timer 控件。

2．设置窗体的属性如下：

```
Appearance = 0  'Flat
BackColor = &H00808000&
Caption = "纸牌游戏示例    "
ClientHeight = 5055
ClientLeft = 1335
ClientTop = 2085
ClientWidth = 7470
LinkMode = 1  'Source
LinkTopic = "Form1"
PaletteMode = 1  'UseZOrder
ScaleHeight = 337
```

```
    ScaleMode = 3   'Pixel
    ScaleWidth = 498
```

3．**设置** Timer **控件的** Interval **属性为** 200。

4．**单击菜单编辑器，向其中添加四个菜单，各菜单的属性设置如下：**

```
Begin VB.Menu View
    Caption         =   "纸牌控制"
    Begin VB.Menu MenuDrawCard
        Caption         =   "绘制纸牌"
    End
    Begin VB.Menu MenuDealCard
        Caption         =   "排列纸牌"
    End
    Begin VB.Menu MenuDrawBack
        Caption         =   "纸牌背面"
    End
    Begin VB.Menu MenuDrawSymbol
        Caption         =   "纸牌标记"
    End
    Begin VB.Menu MenuRemoveCard
        Caption         =   "删除纸牌"
    End
    Begin VB.Menu o
        Caption         =   "-"
    End
    Begin VB.Menu MenuExit
        Caption         =   "退出"
    End
End
Begin VB.Menu Information
    Caption         =   "信息"
    Begin VB.Menu MenuCardInformation
        Caption         =   "纸牌信息"
    End
End
Begin VB.Menu Dragging
    Caption         =   "拖动"
    Begin VB.Menu MenuDoDrag
        Caption         =   "开始拖动"
    End
```

```
End
Begin VB.Menu MenuHelp
   Caption         =    "帮助"
   Begin VB.Menu dllhelp
      Caption      =    "Dll 帮助"
   End
End
```

5．向程序中添加一个模块，命名为 GLOBAL，**用来调用** dll **中的函数，其代码如下：**

```
Declare Function WinHelp Lib "user32" Alias "WinHelpA" (ByVal hwnd As Long, ByVal lpHelpFile As String, ByVal wCommand As Long, ByVal dwData As Long) As Long
'定义常量
Global Const FACEDOWN = 0
Global Const FACEUP = 1
Global Const CARDWIDTH = 71
Global Const CARDHEIGHT = 96
Global Const OFFSET = 16
'导入 dll
Declare Function InitializeDeck Lib "qcard32.dll" (ByVal hwnd As Long) As Long
'重置所有的变量
Declare Sub SetDefaultValues Lib "qcard32.dll" ()
Declare Sub SetCurrentBack Lib "qcard32.dll" (ByVal nIndex As Long)
'绘制纸牌函数
Declare Sub DrawSymbol Lib "qcard32.dll" (ByVal hwnd As Long, ByVal nValue As Long, ByVal x As Long, ByVal y As Long)
Declare Sub DrawCard Lib "qcard32.dll" (ByVal hwnd As Long, ByVal nCard As Long, ByVal x As Long, ByVal y As Long)
Declare Sub DrawBack Lib "qcard32.dll" (ByVal hwnd As Long, ByVal nValue As Long, ByVal x As Long, ByVal y As Long)
Declare Sub DealCard Lib "qcard32.dll" (ByVal hwnd As Long, ByVal nCard As Long, ByVal x As Long, ByVal y As Long)
Declare Sub RemoveCard Lib "qcard32.dll" (ByVal hwnd As Long, ByVal nCard As Long)
'得到纸牌的信息函数
Declare Function GetCardColor Lib "qcard32.dll" (ByVal nCard As Long) As Long
Declare Function GetCardSuit Lib "qcard32.dll" (ByVal nCard As Long) As Long
Declare Function GetCardValue Lib "qcard32.dll" (ByVal nCard As Long) As Long
Declare Function GetCardStatus Lib "qcard32.dll" (ByVal nCard As Long) As Long
Declare Function GetCardBlocked Lib "qcard32.dll" (ByVal nCard As Long) As Long
```

……**（此处代码略，详见光盘）**

实例 27　纸 牌 组 件

实例说明

续前面的实例。如图 27-1 所示。

本例中主要讲解了程序所使用的变量；在程序加载阶段，导入 dll 文件，并进行一些初始化工作以及响应鼠标的按下、抬起操作，以实现纸牌的拖动。

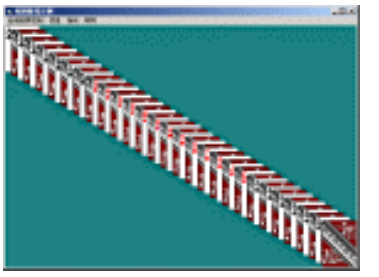


图 27-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。通过 InitializeDeck 函数的返回值判断是否已经加载 qcard.dll，如果没有加载，将返回错误信息。然后调用 qcard.dll 中的相应函数，以实现对纸牌的拖动。

创作步骤

程序设计步骤续前面的实例。

1．程序中定义了多个公共变量，用来记录纸牌的相关信息，它们的含义如下：

```
' 声明计数器变量
Dim i As Integer
Dim j As Integer
Dim n As Integer
定义开关变量
Dim bDragDemo As Integer
Dim bSingleDragging As Integer
Dim bBlockDragging As Integer
Dim bMouseMoved As Integer
Dim nDrawSelection As Integer
定义纸牌标识
Dim nSourceCard As Integer
Dim nSourceArrayID As Integer
Dim nSourceArrayPos As Integer
Dim nDestCard As Integer
Dim nDestArrayID As Integer
Dim OldX As Integer
```

```
Dim OldY As Integer
Dim Temp() As Long
Dim nItems As Integer
Dim nInformationCard
'保存鼠标双击时的信息
Dim DblClickX As Integer
Dim DblClickY As Integer
'定义两个数组
'保存每组牌的数目
Dim CardArray(1 To 4, 1 To 26) As Integer
'定义每一组的计数器
Dim Counter(1 To 4) As Integer
```

2．在程序加载阶段，导入 dll 文件，并进行一些初始化工作，其代码如下：

```
Private Sub Form_Load()
'导入 dll 文件，如果失败，显示提示信息
Dim nReturn As Integer
nReturn = InitializeDeck(Form1.hwnd)
If nReturn = False Then
    MsgBox "Problem loading QCards32.DLL"
    End
End If
'使程序和 dll 使用同一个坐标系
Form1.ScaleMode = 3
Form1.ScaleTop = 0
Form1.ScaleLeft = 0
'使程序全屏
Form1.Width = Screen.Width
Form1.Height = Screen.Height
Form1.Top = 0
Form1.Left = 0
'一些初始化工作
n = 13
bDragDemo = False
bSingleDragging = False
bBlockDragging = False
bMouseMoved = False
nDrawSelection = 0
nInformationCard = 0
End Sub
```

3．响应鼠标的按下、拖动以及双击操作，用来移动纸牌，它们的代码如下：

```
Private Sub Form_DblClick()
双击纸牌时的操作
Dim nNewX As Integer
Dim nNewY As Integer
Dim nThisSourceCard As Integer
Dim nThisDestCard As Integer
'只有在拖动模式下才响应该操作
If bDragDemo = False Then
    Exit Sub
End If
nThisSourceCard = PointInFreeCard(DblClickX, DblClickY)
If nThisSourceCard <> 0 Then
    nSourceArrayID = GetUser4(nThisSourceCard)
    将纸牌从起始位置放置到终了位置
    Select Case nSourceArrayID
……（此处代码略，详见光盘）

If bSingleDragging = True Then
    ' 结束拖动操作，查看纸牌的新位置
    nDestCard = EndDrag(Form1.hwnd, x, y)
    nSourceColor = GetCardColor(nSourceCard)
    nDestColor = GetCardColor(nDestCard)
    nSourceArrayPos = GetUser3(nSourceCard)
    '进行颜色判断，如果是相同颜色，则可以拖动，否则，不能拖动
    If nDestCard = 0 Or nSourceColor <> nDestColor Or nSourceArrayPos = 1 Then
        ReturnDrag Form1.hwnd, nSourceCard, OldX, OldY
        bSingleDragging = False
    Else
        nSourceArrayID = GetUser4(nSourceCard)
        ' 还原计数器数字
        Counter(nSourceArrayID) = Counter(nSourceArrayID) - 1
        nDestArrayID = GetUser4(nDestCard)
        Counter(nDestArrayID) = Counter(nDestArrayID) + 1
        AdjustCardBlocked nDestCard, True
        ' 安装新的数组 ID 和位置
        SetUser3 nSourceCard, Counter(nDestArrayID)
        SetUser4 nSourceCard, nDestArrayID
        ' 将纸牌左对齐
```

```
            nNewX = GetCardX(nDestCard)
            nNewY = GetCardY(nDestCard)
            RemoveCard Form1.hwnd, nSourceCard
            DealCard Form1.hwnd, nSourceCard, nNewX, nNewY + OFFSET
            AdjustCardBlocked CardArray(nSourceArrayID, Counter(nSourceArrayID)), False
            '将纸牌添加到新的数组中
            CardArray(nDestArrayID, Counter(nDestArrayID)) = nSourceCard
            bSingleDragging = False
        End If
    ElseIf bBlockDragging = True And bMouseMoved = True Then
        '可以重新使用 Temp 数组
        '结束拖动，找到纸牌的终点位置
        nDestCard = EndBlockDrag(Form1.hwnd, Temp(0), nItems, x, y)
        nSourceColor = GetCardColor(nSourceCard)
        nDestColor = GetCardColor(nDestCard)
        nSourceArrayPos = GetUser3(nSourceCard)
        '进行颜色检测，判断是否可以拖动
        If nDestCard = 0 Or nSourceColor <> nDestColor Or nSourceArrayPos = 1 Then
            '如果不能拖动，则返回
            ReturnBlockDrag Form1.hwnd, Temp(0), nItems, OldX, OldY
            bBlockDragging = False
            bMouseMoved = False
        Else
            nSourceArrayID = GetUser4(nSourceCard)
            '还原计数器
        ……（此处代码略，详见光盘）

    ElseIf bBlockDragging = True And bMouseMoved = False Then
        AbortDrag
        bBlockDragging = False
    End If
    End Sub
```

本游戏其他代码详见光盘。

实例 28　弹　　珠

实例说明

本例制作弹珠小游戏。效果如图 28-1 所示。

本例运行后，黄色小球到达窗口的边缘后会被弹回来，通过 Option 菜单还可以调节黄色小球的移动速度。

本例通过 VB 的图像控件及编写动画程序实现。

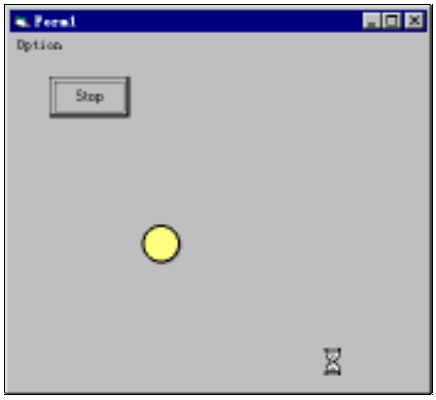


图 28-1　效果图

编程思路

在各种游戏中，动画是不可缺少的一部分。本例为了在 VB 中实现动画，利用定时器控件，每隔一定时间让图像移动一定距离，或者象播放电影胶片似的连续显示一系列图片，给人以视觉上的错觉，从而实现动画效果。

本例制作的是一个弹珠的动画，利用定时器控件，每隔一定时间让图像移动一定距离来实现效果。

创作步骤

1．启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程。

2．在 Form1 中创建一个命令按钮（Command）、一个图像控件（Image）、一个计时器控件（Timer），把 Command1 的 Caption 项设为“Start”。单击 Image1，然后在属性栏中单击 Picture 项中 None 后面的小方块，将出现如图 28-2 所示的打开图片文件对话框，选择要载入的图片或图标即可。如果你的 Visual Basic 的安装路径是默认值的话，本例需要的图标在 C 盘的 Program Files\Microsoft Visual Studio\Common\Graphics\Icon\Elements 目录下的 Moon05.ico。

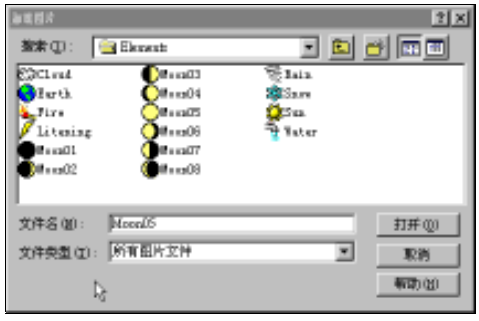


图　28-2

3．然后按<Ctrl>+<E>键打开菜单编辑窗口，如图 28-3 所示。

图　28-3

用菜单编辑器创建一个 Option **菜单。然后在** Option **菜单下，创建** Fastest、Faster、Fast、Nommal **和** slow **等五个子菜单。**Name **分别为** opti、fastest、faster、fast、nommal **和** slow**。同时把** Nommal **的** Checked **项选上。如图** 28-4 **所示。**

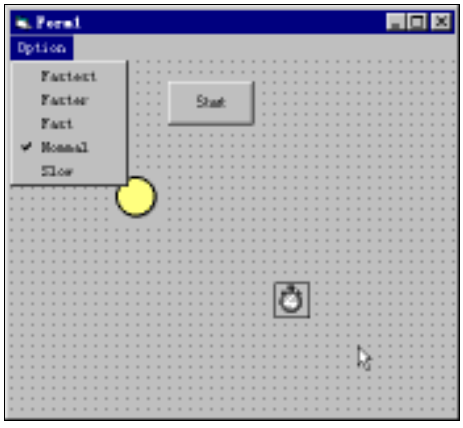


图　28-4

编写代码（详见光盘）。

实例 29　模拟正态分布

实例说明

本例制作模拟正态分布的小游戏，效果如图 29-1 所示。

本程序运行后，窗口中将不断落下小球，通过与上方的小格子相撞击、模拟正态分布的概率。小球落到合适的地方，运行一段时间后，会看到概率大的位置出现的小球多，概率小的位置出现的小球少。中间黑色线条是正态分布概率的曲线。运行过程中，可在画面中点击鼠标暂停或继续做游戏。

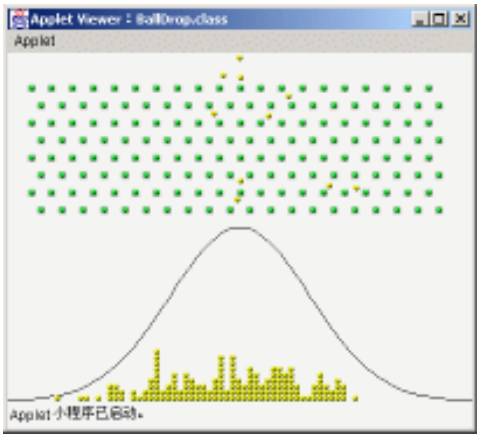


图 29-1　效果图

编程思路

本例应用了 Applet 中的类。Applet 中只有两个类：Ball 和 BallDrop 类。Ball 类从 awt 包中的 ImageObserver 类继承而来，用来表示下落的小球。BallDrop 类是主类，实现了 Runnable 接口。程序运行后，每当过一定的延迟时间就会重新画屏幕。根据计算将小球的位置和小球的积累情况画到屏幕上。调用的方法是：UpdateRack,UpdateBalls 和 paintBackground。根据需要，也可以加入背景声音。

创作步骤

1．Ball 类的主要创作内容解释如下：

```
class Ball implements ImageObserver {
  AudioClip itsBoink; // 声音对象
   void SetBoinkSound(AudioClip aClip) {                    // 设置声音对象
     itsBoink=aClip;
  }
  void Accelerate(double ax,double ay) {                    // 加速
     vx += ax;
     vy += ay;
     vlen=Math.sqrt(vx*vx+vy*vy);
  }
        boolean Boink(int pinx,int piny,double pinr) {
```

```
…
}                                                    // 画 pink
void SetImage(Image aImage) {
      itsImage=aImage;
      w=itsImage.getWidth(this);
      h=itsImage.getHeight(this);
      r=w/2.0;
   }                                                 // 设置图像
public boolean imageUpdate( …)
   {
      if((infoflags & (WIDTH | HEIGHT)) == (WIDTH | HEIGHT)) {
         if(img == itsImage) {
            w=width;
            h=height;
            r=w/2;
         }
         return false;
      }
      return true;
   }
}                                                    // 重新画小球
```

2．BallDrop 类的主要内容如下：

```
public class BallDrop extends Applet implements Runnable {
      Thread itsThread = null;                              // 线程对象
      MediaTracker itsTracker = null;
      AudioClip itsBoink = null;                            // 声音对象
      Vector itsBalls;
      Dimension offDimension,backDimension;
      Image offImage,backImage;
      static final int numrows=8,numcolumns=20,numballs=10,delay=10,
         topspace=30,sidespace=20;
      Image pin,ball;
      double pinr;
      int ballw,ballh,pinw,pinh,numracks,rackheight[],rackdel[];
…..
public void init() {                  // 初始化 Applet，调入小球和格子的图像备用
         Dimension d = size();
         Ball aBall;
         itsTracker=new MediaTracker(this);
```

```
        ball = getImage(getDocumentBase(), "smallball.gif");
        pin = getImage(getDocumentBase(), "smallpin.gif");
        itsTracker.addImage(ball,0);
        itsTracker.addImage(pin,0);
    public boolean mouseDown(Event e, int x, int y) {          // 处理鼠标事件，当点击鼠标时，
按照上下文进行相应的程序状态改变，即暂时停止或继续运行。
        if (itsThread == null) {
          start();
        }
        else {
          itsThread = null;
        }
        return false;
      }
      public void paint(Graphics g) {                  // 绘制前面的帧
        Graphics offGraphics;
        Ball aBall;
        Dimension d = size();
        // 如果需要，创建背景图像
        if ( (backImage == null)
            || (d.width != backDimension.width)
            || (d.height != backDimension.height) ) {
          Graphics backGraphics;
          backDimension = d;
          backImage = createImage(d.width, d.height);
          backGraphics = backImage.getGraphics();
            // 清除掉前面的帧
          backGraphics.setColor(getBackground());
          backGraphics.fillRect(0, 0, d.width, d.height);
          backGraphics.setColor(Color.black);
          // 绘制当前帧到图像中
          paintBackground(backGraphics);
        }
```

实例 30　捡金豆人机交互

实例说明

本例编写捡金豆游戏，如图 30-1 所示。

游戏规则如下 :捡豆子最多者可以赢得游戏，玩家的碗为绿色，每个碗上的数字是这个碗里豆子的个数，按“←”键和“→”键可以移动小手，按“回车”键可以从碗里捡出豆子，小手将按逆时针方向给每个碗里放入一个豆子。如果最后的一个豆子落入了你的大碗，你将获得一次新的机会。你捡出的豆子不会掉入对手的碗里，如果最后一个豆子落入你的某个空碗，你将从对手的小碗中得到豆子。

在本实例中主要讲述了程序的窗体设计，同时实现了如何计算碗中的豆子数目以及判断是否结束。

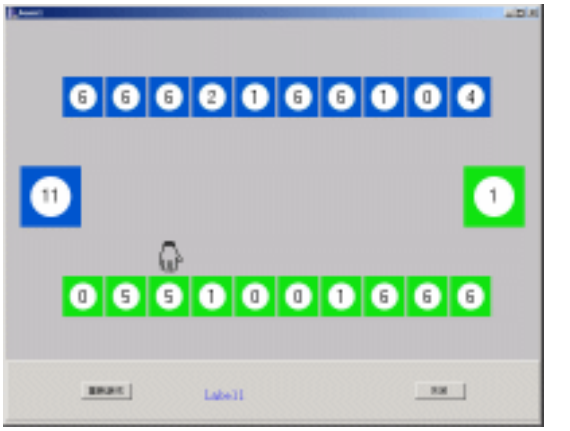


图 30-1　效果图

编程思路

本实例使用 C++Builer 6.0 来实现。通过调用 Compute 函数计算玩家和电脑的得分，使用 Computego 函数判断是否结束。

创作步骤

1．启动 C++ Builder 6.0，打开一个新的标准工程。如果 C++ Builder 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程，向其中添加两个 Panel 控件，向第一个 Panel 控件中添加一个 Image 控件，向第二个 Panel 控件中添加两个 Button 控件和一个 Label 控件。如图 30-2 所示。

图　30-2

2．各控件只是改变了它们的 Caption 属性，其他属性全部保持默认值，这里不再介绍。

3．在程序的头文件中定义了要使用的变量和过程，其代码如下（这里只列出了部分代码，其他代码详见配套光盘）：

```
// ---------------------------------------------------------------------------
#ifndef BeanH
#define BeanH
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
// ---------------------------------------------------------------------------
struct POT
{
    int Index;
    int Bean;
};
……
private:    //  User declarations
    int nJudge; // 判断是否结束
    int MyBean,YourBean,nRand[1000],nCur,nMyCur,nYourCur;          // 双方的豆子数目
    POT MyPot[10],YourPot[10];        // 结构数组，具体内容见前面的定义
    TPoint MyPoint,YourPoint;
    TRect *rYBean,*YourRect[10],*MyRect[10],*rMBean;
    TColor clTrans;
    Graphics::TBitmap *bmpPot1,*bmpPot2,*bmpBean1,*bmpBean2,*bmpHandO1,
            *bmpHandO2,*bmpHand1,*bmpHand2,*bmpGround,*bmpHandOver;
            // 显示碗的图片
    Graphics::TBitmap *Over;
public:             //  User declarations
    __fastcall TForm1(TComponent* Owner);
    void Init();
    int Computer();
    void TForm1::ComputerGo(int nTmp);
    void TForm1::ShowOver1(int NumHand, int NumText, int Hand, String Text,Graphics::TBitmap
*bmpHand);
    void TForm1::ShowOver2(int NumHand, int NumText, int Hand, String Text,Graphics::TBitmap
*bmpHand);
    void TForm1::ShowOver3(int NumHand, int NumText, int Hand, String Text,Graphics::TBitmap
*bmpHand,Graphics::TBitmap *Bean);
};
// ---------------------------------------------------------------------------
```

```
extern PACKAGE TForm1 *Form1;
// ---------------------------------------------------------------------------
#endif
```

4．Compute **函数的代码如下：**

```
int TForm1::Computer()
// 计算碗中的豆子数目
{
    nJudge=1;
    while(nJudge>0)
    {
        nJudge=1;
        imgShow->Canvas->Draw(0,308,Over);
        imgShow->Canvas->Draw(0,145,Over);
        for(int k=0;k<10;k++)
            if(YourPot[k].Bean==0)
                for(int i=k+1;i<10;i++)
                    if(YourPot[i].Bean==i-k&&YourPot[i].Bean!=0)
                     {
                        ComputerGo(i);
                        YourBean+=MyPot[9-k].Bean;
                        MyPot[9-k].Bean=0;
                        imgShow->Canvas->Draw(22,218,bmpBean1);
                        if(YourBean>9)
                            imgShow->Canvas->TextOutA(53,245,IntToStr(YourBean));
                        else
                            imgShow->Canvas->TextOutA(59,245,IntToStr(YourBean));
                        imgShow->Canvas->Draw(85+64*k,381,bmpPot2);
                        imgShow->Canvas->TextOutA(108+64*k,395,"0");
                        imgShow->Refresh();
                        nJudge=0;
                        return 1;
                     }
        for(int k=0;k<10;k++)
            if(YourPot[k].Bean==k+1)
             {
                ComputerGo(k);
                nJudge=2;
                break;
             }
        if(nJudge==1)
        {
            for(int k=nCur;;k++)
```

```
                    if(YourPot[nRand[k]].Bean!=0)
                    {
                        nCur=k;
                        break;
                    }
                ComputerGo(nRand[nCur++]);
                return 1;
            }
        }
}
```

5．Computego **函数的代码如下：**

```
void TForm1::ComputerGo(int nTmp)
// 判断是否结束
{
    nYourCur=nTmp;
    YourPoint=Point(100+64*nYourCur,150);
    ShowOver2(YourPoint.x,YourPoint.y,2,IntToStr(YourPot[nYourCur].Bean),bmpHandO2);
    ShowOver2(YourPoint.x,YourPoint.y,2,IntToStr(YourPot[nYourCur].Bean),bmpHand2);
    int n=YourPot[nYourCur].Bean;
    YourPot[nYourCur].Bean=-1;
    for(int k=1;k<=n;k++)
    {
    ……（此处代码略，详见光盘）
    if(MyBean>YourBean)
        Label1->Caption="Good! You WIN!!";
    else if(MyBean<=YourBean)
        Label1->Caption="Sorry! You LOSE!";}
```

实例 31　捡金豆组件

实例说明

续前面的实例。如图 31-1 所示。

在本实例中主要讲述如何响应玩家的按键。根据游戏规则，计算机将响应玩家的按键，判断是移动小手还是捡金豆，同时向后面的碗中添豆子。

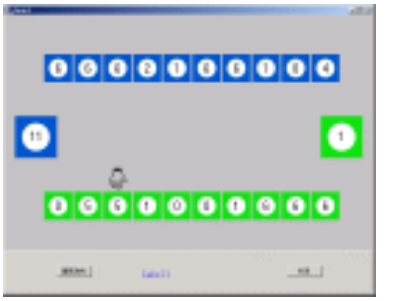


图 31-1　效果图

编程思路

本实例使用 C++ Builder 6.0 来实现。当玩家按下“←”键或者“→”键时，可以移动小手，当玩家按下“回车”键时，将从碗中捡出豆子。

创作步骤

程序的 KeyDown 事件用来响应玩家的按键，其代码如下：

```
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)
// 响应用户的按键，移动小手
{
    if(Key==VK_LEFT&&nMyCur!=9)
    {
        MyPoint=Point(MyPoint.x-64,MyPoint.y);
        imgShow->Canvas->Draw(MyPoint.x+64,MyPoint.y,bmpHandOver);
        imgShow->Canvas->Draw(MyPoint.x,MyPoint.y,bmpHandO1);
        nMyCur++;
    }
    else if(Key==VK_RIGHT&&nMyCur!=0)
    {
        MyPoint=Point(MyPoint.x+64,MyPoint.y);
        imgShow->Canvas->Draw(MyPoint.x-64,MyPoint.y,bmpHandOver);
        imgShow->Canvas->Draw(MyPoint.x,MyPoint.y,bmpHandO1);
        nMyCur--;
    }
    else if(Key==VK_RETURN)
```

```
{
    ShowOver1(MyPoint.x,MyPoint.y,2,IntToStr(MyPot[nMyCur].Bean),bmpHandO1);
    ShowOver1(MyPoint.x,MyPoint.y,2,IntToStr(MyPot[nMyCur].Bean),bmpHand1);
    int n=MyPot[nMyCur].Bean;
    MyPot[nMyCur].Bean=-1;
    for(int k=1;k<=n;k++)
    {
        if(k<=nMyCur)
        {

            ShowOver1(MyPoint.x+64*(k-1),MyPoint.y,0,IntToStr(++MyPot[nMyCur-k+1].Bean),bm
            pHand1);
            ShowOver1(MyPoint.x+64*k,MyPoint.y,1,IntToStr(MyPot[nMyCur-k].Bean),bmpHand1);
            ShowOver1(MyPoint.x+64*k,MyPoint.y,2,IntToStr(MyPot[nMyCur-k].Bean),bmpHandO
            1);
              if(k==n)
              {
            ShowOver1(MyPoint.x+64*k,MyPoint.y,0,IntToStr(++MyPot[nMyCur-k].Bean),bmpHan
      dO1);
                    if(MyPot[nMyCur-k].Bean==1)
                    {
                        MyBean+=YourPot[9-nMyCur+n].Bean;
                        YourPot[9-nMyCur+n].Bean=0;
                        imgShow->Canvas->Draw(681,218,bmpBean2);
                        if(MyBean>9)
                            imgShow->Canvas->TextOutA(712,245,IntToStr(MyBean));
                        else
                            imgShow->Canvas->TextOutA(718,245,IntToStr(MyBean));
                        imgShow->Canvas->Draw(85+64*(9-nMyCur+n),85,bmpPot1);
                        imgShow->Canvas->TextOutA(108+64*(9-nMyCur+n),99,"0");
                        imgShow->Refresh();
                    }
              }
        }
```

……**（此处代码略，详见光盘）**

```
    if(MyBean>YourBean)
        Label1->Caption="Good! You WIN!!";
    else if(MyBean<=YourBean)
        Label1->Caption="Sorry! You LOSE!";}}
```

实例 32　捡金豆初始化

实例说明

续前面的实例。如图 32-1 所示。

本例主要实现程序的初始化部分，在 Init（）函数中可以初始化各碗中的豆子数目，在 fastcall TForm1:: TForm1(Tcom-ponent*Owner)过程中调用 Init（）函数对窗体中的内容进行初始化。另外在程序退出时释放内存资源。

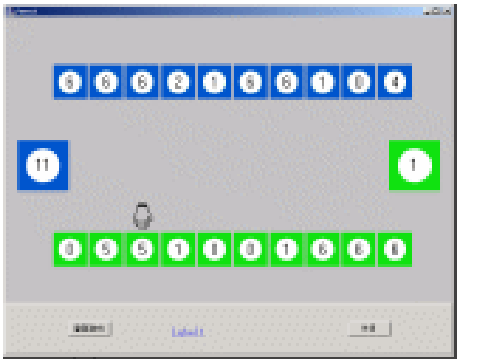


图 32-1　效果图

编程思路

本实例使用 C++ Builder 6.0 来实现。在程序的初始化阶段利用 bmpPotx=new Graphics::TBitmap 方法产生位图对象，然后使用位图对象的 LoadFromFile 方法导入外部图像文件，最后将窗体布置成如图 32-1 所示。

创作步骤

程序设计步骤续前面的实例。程序的其他代码如下：

```
// ---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Bean.h"
// ---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
// ---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    // 初始化窗体中的内容
      : TForm(Owner)
{
      clTrans=TColor(0x00C0C0C0);
```

```
    for(int k=0;k<10;k++)
    {
……（此处代码略，详见光盘）
// ---------------------------------------------------------------------------
void TForm1::Init()
// 初始化各碗中的豆子数目
{
    imgShow->Canvas->Draw(0,0,bmpGround);
    for(int k=0;k<10;k++)
    {
        MyPot[k].Index=YourPot[k].Index=k;
        MyPot[k].Bean=YourPot[k].Bean=4;
    }
    MyBean=YourBean=0;
    imgShow->Canvas->Draw(22,218,bmpBean1);
    imgShow->Canvas->Draw(681,218,bmpBean2);
    imgShow->Canvas->TextOutA(59,245,IntToStr(MyBean));
    imgShow->Canvas->TextOutA(718,245,IntToStr(YourBean));
    for(int k=0;k<10;k++)
    {
        imgShow->Canvas->Draw(85+64*k,85,bmpPot1);
        imgShow->Canvas->Draw(85+64*k,381,bmpPot2);
        imgShow->Canvas->TextOutA(108+64*k,99,IntToStr(MyPot[k].Bean));
        imgShow->Canvas->TextOutA(108+64*k,395,IntToStr(MyPot[k].Bean));
    }
    MyPoint=Point(100,330);
    imgShow->Canvas->BrushCopy(Rect(100,330,136,376),bmpHandO1,Rect(0,0,36,46),clTrans);
    imgShow->Refresh();
    srand(time(NULL)*100+StrToInt(FormatDateTime("zzz",Now()))/10);
    for(int k=0;k<1000;k++)
        nRand[k]=((rand())%10);
    nCur=0; nMyCur=9;
    Form1->ActiveControl=NULL;
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
// 关闭程序
{
    delete [] YourRect;
```

```
    delete [] MyRect;
    delete rMBean;
    delete rYBean;
    Close();
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
// 开始游戏
{
    Init();
}
// ---------------------------------------------------------------------------
void TForm1::ShowOver1(int NumX, int NumY, int Hand, String Text, Graphics::TBitmap *bmpHand)
// 显示结束信息
{
    imgShow->Canvas->Draw(NumX-64,NumY,bmpHandOver);
    imgShow->Canvas->Draw(0,145,Over);
    imgShow->Canvas->Draw(NumX,NumY,bmpHandOver);
    imgShow->Canvas->Draw(NumX-15,NumY+51,bmpPot2);
    imgShow->Canvas->BrushCopy(Rect(NumX,NumY+30*Hand,NumX+36,NumY+46+30*Hand),bmp
    Hand,Rect(0,0,36,46),clTrans);
    if(StrToInt(Text)>9)
        imgShow->Canvas->TextOutA(NumX-1,NumY+65,Text);
    else
        imgShow->Canvas->TextOutA(NumX+8,NumY+65,Text);
    imgShow->Refresh();
    Sleep(300);
}
// ---------------------------------------------------------------------------
void TForm1::ShowOver2(int NumX, int NumY, int Hand, String Text,Graphics::TBitmap *bmpHand)
// 显示结束信息
{
    ……（此处代码略，详见光盘）
    imgShow->Refresh();
    Sleep(300);
}
// ---------------------------------------------------------------------------
```

实例 33　华容道人机交互

实例说明

本例编写华荣道游戏，如图 33-1 所示。

本例游戏取材于三国时期曹操败走华容道的故事，玩家要做的是用你的智慧帮助曹操躲过马超等人的围追堵截，顺利逃出华容道。玩家可以单击任何一个人物，如果玩家单击人物图片的左侧，同时改人物左侧正好有一个空位，那么被单击的人物将会移动到空位中去。同理，单击人物的上方、下方以及右侧也会移动相应的图片。单击“重新开始”按钮，将会把画面复原到初始状态，开始新一轮游戏。单击“退出”按钮，将会退出游戏。

在本实例中主要讲述程序的窗体设计以及程序头文件中所定义的控件和变量。

图 33-1　效果图

编程思路

本实例利用 C++ Builder 来实现。在窗体中添加了多个 Panel 控件和 Image 控件，通过 Panel 控件来控制 Image 控件的移动位置，另外在程序的头文件中定义了本游戏所使用的公共变量和公共过程。

创作步骤

1．启动 C++ Builder，打开一个新的标准工程。如果 C++ Builder 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Panel 控件，在该控件中添加一个 Image 控件作为游戏的主画面。然后再在 Panel 中添加 10 个 Panel 控件，每个控件中放置一个 Image 控件，作为 10 个人物，各人物的名称如图 33-1 所示。

然后再向窗体的中下方添加一个 Panel 控件，再该 Panel 控件中添加两个 Button 控件，分别为“开始游戏”和“退出”按钮。

在窗体的最下方添加一个 Label 控件，用来显示玩家所走的步数。各控件的属性设置比较简单，这里不再介绍。

2．在程序的头文件中添加如下代码：

```
#ifndef HuaRongDaoH
```

```
#define HuaRongDaoH
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
enum Way {Up, Down, Left, Right};
class TForm1 : public TForm
{
__published:       //    IDE-managed Components
        TPanel *Panel1;
        TPanel *PanelCao;
```

……**（此处代码略，详见光盘）**

```
        void __fastcall ImageWeiMouseDown(TObject *Sender,
          TMouseButton Button, TShiftState Shift, int X, int Y);
        void __fastcall ImageWangMouseDown(TObject *Sender,
          TMouseButton Button, TShiftState Shift, int X, int Y);
private:      //    User declarations
        bool Hollow[4][5];
        int TypeCur,Step;
        TPoint Hollow1,Hollow2,CoorCur;
        TPoint
CoorCao,CoorGuan,CoorZhang,CoorZhao,CoorHuang,CoorMa,CoorWei,CoorJiang,CoorWang,CoorZhou;
        TPoint WayCur;
        TPoint Hollow0;
public:               //    User declarations
        __fastcall TForm1(TComponent* Owner);
        void MoveTo(TPanel *Panel,TPoint Coor);
        TPoint GetWay(TImage *Image, int X, int Y);
        bool JudgeWay();
        bool JudgePT(TPoint pt1, TPoint pt2);
        void GetHollow();
        void TForm1::Init();
        void Result();
        void Move(TPanel *Pan, TImage *Image, TPoint &Point, int Cur, int x, int y);
        void PutHollow(TPoint pt,int a1,int b1,int a2,int b2,int a3,int b3,int a4,int b4);
};
extern PACKAGE TForm1 *Form1;
// ---------------------------------------------------------------------------
#endif
```

实例 34　华容道组件

实例说明

续前面的实例。如图 34-1 所示。

本例主要实现了程序的几个关键函数。其中 Move 函数用来移动游戏中的各个角色；JudgeWay 函数用来判断玩家的移动是否合理；GetWay 函数用来判断角色的移动方向；JudgePT 用来判断两个点（或方向）是否相等；PutHollow 函数用来重新设定空位。

图 34-1　效果图

编程思路

本实例中通过已经设定好的各个人物的坐标来判断如何移动角色以及移动的方向，建立不同的函数来实现具体的功能，以便后面的事件响应进行调用。

创作步骤

程序设计步骤续前面的实例。

本实例主要定义了 Move、Moveto、Getway、Judge、JudgePT、GetHollow、PutHollow 和 Result 函数，它们的含义和代码如下：

```
// ---------------------------------------------------------------------------
void TForm1::Move(TPanel *Pan, TImage *Image, TPoint &Point, int Cur, int x, int y)
{    // 移动函数
     CoorCur=Point;        // 设置当前移动块
     TypeCur=Cur;          // 当前块的类型(1---大块，2---中块，3---小块)
     WayCur=GetWay(Image,x,y); // 要移动的方向
     if(JudgeWay())     // 判断是否能移动
     {
          Hollow0.x=CoorCur.x+WayCur.x;
          Hollow0.y=CoorCur.y+WayCur.y;   // 当前块要移动的位置
          MoveTo(Pan, Hollow0);           // 移动当前块
          GetHollow();                    // 设置移动后的空块
          Point=CoorCur;                  // 移动后的块坐标
```

```
    }
}
// ---------------------------------------------------------------------------
void TForm1::MoveTo(TPanel *Panel,TPoint Coor)          // 最简单直接的移动函数
{
    Panel->Left=8+82*Coor.x;
    Panel->Top=8+82*Coor.y;
    Label1->Caption="                    "+IntToStr(++Step);      // 显示移动总步数
}
// ---------------------------------------------------------------------------
TPoint TForm1::GetWay(TImage *Image, int X, int Y)      // 根据单击的位置判断应该移动的方向
{
    ……（此处代码略，详见光盘）
void __fastcall TForm1::Button1Click(TObject *Sender)
{                   // 重新游戏
    Init();
}
// ---------------------------------------------------------------------------
void TForm1::Result()               // 显示成绩
{
    if(Step<=46)
        Label1->Caption="Perfect！      " +IntToStr(Step)+"    Steps !!!";
    else if(Step<60)
        Label1->Caption="Good！      "+IntToStr(Step)+"    Steps !!";
    else if(Step<80)
        Label1->Caption="Yeah！      "+IntToStr(Step)+"    Steps !";
    else if(Step<120)        Label1->Caption="Yes！      "+IntToStr(Step)+"    Steps !";
    Else      Label1->Caption="My God！      "+IntToStr(Step)+"    Steps .";
}
```

本例其他代码详见光盘。

实例 35　象 棋 麻 将

实例说明

本例制作象棋麻将游戏。效果如图 35-1 所示。

本例游戏定义了一些玩法规则，当你学习或工作感觉到疲劳时，不妨坐下来感受一下本游戏带给你的轻松和愉快。

本例通过 VB 的窗体、类、ComboBox、CommandBox、API 函数等知识制作完成。

图 35-1　效果图

编程思路

象棋麻将游戏是用麻将的规则而用象棋棋子来玩的，由于象棋的棋子没有麻将那么多，所以象棋麻将的规则也就跟麻将有些差别，不过大致玩法是相同的，下面就来说明一下象棋麻将的规则。

手上的五张牌中，要配成三张+两张才能胡牌。

三张：将士象、车马炮、帅士相、车马炮、兵兵兵、卒卒卒。

二张：将帅或其他两张一样。

游戏功能说明如下：

开新棋局：中断目前的棋局，重开一个新的棋局。

游戏设定：设定玩家的名字和电脑的名字。

排行榜：列出每个人的分数排行。

本游戏的整体编程思路是摸牌或发牌后判断是否胡牌。用到的一些方法或属性如下所示：

Cls 方法：用于清除运行时 Form 或 PictureBox 所生成的图形和文本。

Line 方法：在对象上画直线和矩形。

PSet 方法：将对象上的点设置为指定颜色。

PaintPicture 方法：用以在 Form, PictureBox 或 Printer 上绘制图形文件（.bmp、.wmf、.emf、.cur、.ico 或 .dib）的内容。不支持命名参数。

AddItem 方法：用于将项目添加到 ListBox 或 ComboBox 控件，或者将行添加到 MS Flex Grid 控件。不支持命名参数。

ListIndex 属性：返回或设置控件中当前选择项目的索引，在设计时不可用。

DoEvents 函数：转让控制权，以便让操作系统处理其他的事件。

Int、Fix 函数：返回参数的整数部分。

Rnd 函数：返回一个包含随机数值的 Single。

创作步骤

一、创建应用程序界面

1．启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程。

2．首先，需要一些图片，如图 35-2、图 35-3 所示。

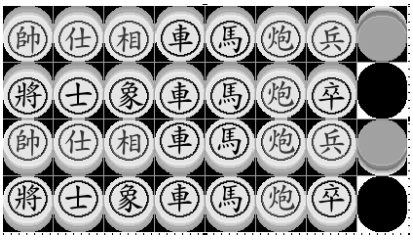


图　35-2

图　35-3

3．在 Form 中创建两个 PictureBox 控件，设置属性见表 35-1。

表　35-1

控　　件	属　　性	值
Form	Caption Name	象棋麻将 MainForm
PictureBox	BorderStyle Name Picture	0-None Pic2 c:\myvbbook\xq.bmp
PictureBox	BorderStyle Name Picture	0-None Pic3 c:\myvbbook\hp.bmp

设置完的界面如图 35-4 所示。

图　35-4

4．再在 Form 中创建一个 PictureBox 控件，在该 PictureBox 控件上添加两个 CommandButton 控件、一个 ListBox 控件。

5．按照表 35-2 来设置各个控件的属性。

表 35-2

控　　件	属　　性	值
CommandButton	Caption Index Name	摸牌 0 Command0
CommandButton	Caption Index Name	吃牌 1 Command1
PictureBox	Name Picture	Pic3 c:\myvbbook\xqbj.bmp
ListBox	Name BackColor	List1 &H00D3E9E8&

6．按<Ctrl>+<E>键打开菜单编辑窗口。用菜单编辑器创建一个“棋局”菜单。然后在“棋局”菜单下创建“新棋局”、“游戏设定”、“排行榜”、“退出”等四个子菜单。Name 分别为 ChessGame、NewGame、Set、GameTop 和 Quit；名称分别为棋局（&G）、新棋局（&N）、游戏设定（&S）、排行榜（&T）、退出（&X）。如图 35-5 所示。

图 35-5

7．按“确定”按钮，返回设计界面，设计好的界面如图 35-6 所示。

图 35-6

8．选中工程菜单的添加窗体子菜单项，该窗体用于游戏设置。

9．在 Form 中创建一个 Label 控件、两个 Frame 控件、一个 TextBox 控件、两个 CommandButton 控件、三个 ComboBox 控件。

10．按照表 35-3 来设置各个控件的属性。设置好的界面如图 35-7 所示。

表　35-3

控　　件	属　　性	值
Form	BackColor BorderStytle Caption Name	&H00FFFF00& 0-None 游戏设置 SetForm
Label	ForeColor BorderStytle Font Name	&H000000FF& 0-None 隶书，二号 Label1
Frame	BackColor Caption Name	&H00D3E9E8& 玩家名称 Frame1
Frame	BackColor Caption Name	&H00D3E9E8& 对手名称 Frame2
TextBox	Name Text	Text1 空
CommandButton	BackColor Caption Name	&H0000FF& 确定 Command1
CommandButton	BackColor Caption Name	&H0000FF& 预设 Command2
ComboBox	Index ItemData List Name	0 0，0，0 孙悟空，猪八戒，沙僧 Combo1
ComboBox	Index ItemData List Name	0 0，0，0 孙悟空，猪八戒，沙僧 Combo1
ComboBox	Index ItemData List Name	0 0，0，0 孙悟空，猪八戒，沙僧 Combo1

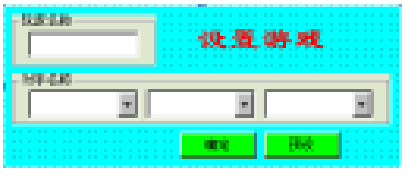


图　35-7

11．选中工程菜单的添加窗体子菜单项，该窗体用于显示排行榜。

12．在 Form 中创建一个 ListView 控件、一个 Frame 控件、四个 OptionButton 控件、一个 CommandButton 控件。

13．按照表 35-4 来设置各个控件的属性。

表 35-4

控　　件	属　　性	值
Form	Caption Name	排行榜 TopForm
Frame	Caption Name	无 Frame1
ListView	Name	ListView1
CommandButton	Caption Name	确定 Command1
OptionButton	Caption Index Name	以自摸次数排序 0 Option1
OptionButton	Caption Index Name	以胡牌次数排序 1 Combo1
OptionButton	Caption Index Name	以放枪次数排序 2 Option1
OptionButton	Caption Index Name	以分数排序 3 Option1

设置好的界面如图 35-8 所示。

14．选中工程菜单的添加窗体子菜单项，该窗体用于游戏结束时显示的画面。

15．在 Form 中创建一个 Label 控件，一个 CommandButton 控件。

16．按照表 35-5 来设置各个控件的属性。设置好的界面如图 35-9 所示。

表 35-5

控　　件	属　　性	值
Form	Caption Name	象棋麻将 AboutForm
Label	ForeColor BorderStytle Font Name	&H000000FF& 0-None 隶书，二号 Label1
CommandButton	Caption Name	退出 Command1

图　35-8

图　35-9

二、创建过程模块

1．选择工程菜单下的添加模块选项，弹出添加模块对话框，如图 35-10 所示。

2．单击“打开”按钮，出现一新的模块编辑器，在此可输入程序代码。输入完毕后，单击文件菜单下的保存模块子菜单项，弹出“文件另存为”对话框，将该模块保存为 mc.bas，如图 35-11 所示。该模块定义了一些数据结构，并用于作为游戏起始程序。

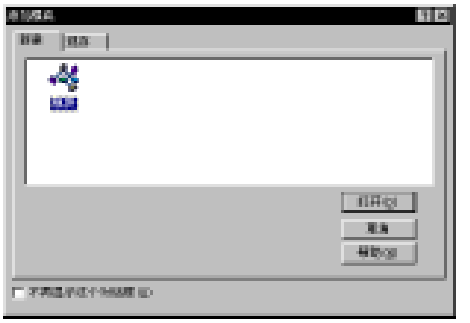


图　35-10

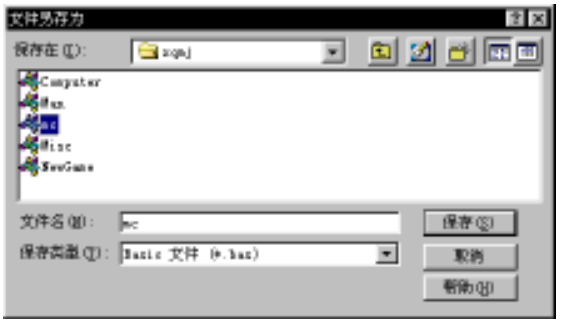


图　35-11

3．按同样的方法创建 NewGame.bas、Misc.bas、Man.bas、Computer.bas 等四个过程模块。其中 NewGame.bas 做了一些初始化程序的工作；Misc.bas 判断是否自摸或胡牌；Man.bas 使玩家丢牌；Computer.bas 使计算机丢牌。

4．添加 API 函数。玩家可以在模块中手工输入 API 函数说明，但这样做比较麻烦，VB 工具中提供了一个 API 文本浏览器，玩家可以快速、准确地从中复制 API 函数到模块中。首先从“开始”菜单中选择“Miscrosoft Visual Baic 6.0 中文版”，进入“Miscrosoft Visual Baic 6.0 中文版工具”，然后再进入 API 文本浏览器。

随后出现 API 文本浏览器，如图 35-12 所示。

图　35-12

选择文件菜单中的加载文本文件子菜单，弹出选择一个文本 API 文件对话框，选择 Win32api，如图 35-13 所示。

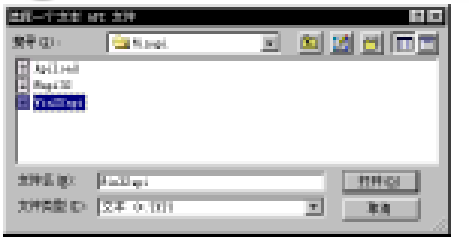


图　35-13

单击“打开”按钮，此时 Win32api 文件被加载到 API 阅览器中，如图 35-14 所示。

在“键入您要查找的内容的开头几个字母”中，输入 GetSystemmenu 并按下回车键，如图 35-15 所示。

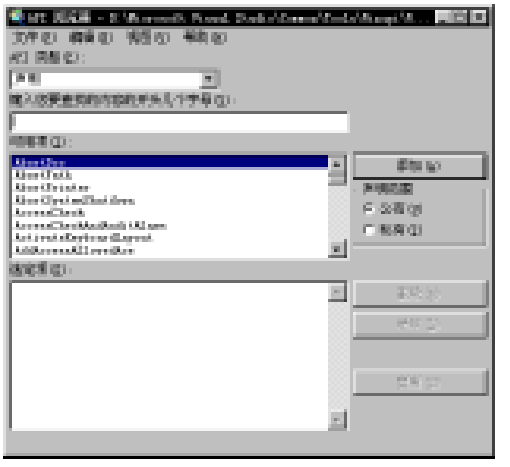


图　35-14

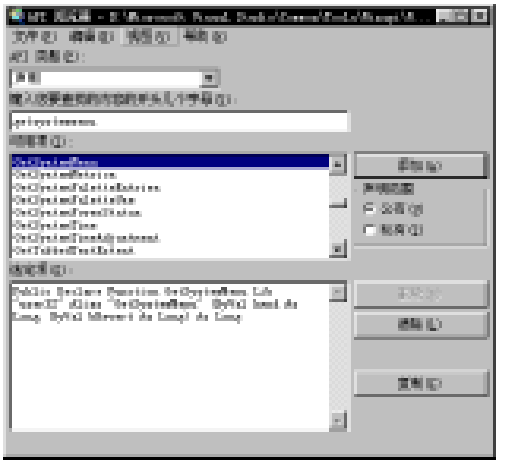


图　35-15

按下“复制”按钮，这样就可以返回到原来的程序模块中，执行“粘贴”命令就可以了。

三、创建类模块

1．选择工程菜单下的添加类模块选项，弹出添加类模块对话框，如图 35-16 所示。

2．单击“打开”按钮，出现一新的类模块编辑器，在此可输入程序代码。输入完毕后单击文件菜单下的保存类模块子菜单项，弹出“文件另存为”对话框，将该模块保存为 chess.cls，如图 35-17 所示。该类模块定义了一些变量和发棋、洗棋、及画出棋子正反面等函数。

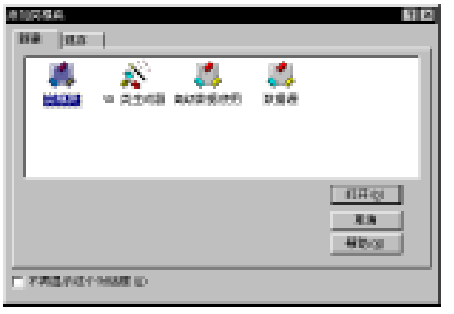


图　35-16

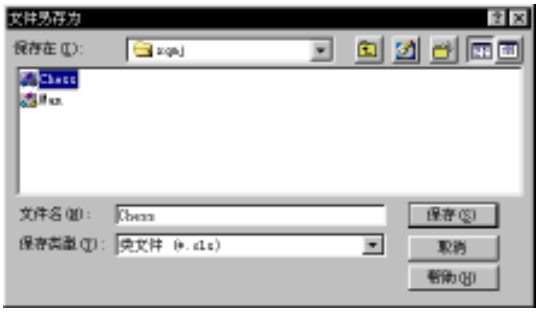


图　35-17

代码及注释（详见光盘）。

实例 36　趣味图画人机交互

实例说明

本例编写趣味图画，如图 36-1 所示。

单击“主菜单”中“打开位图”菜单项，导入一张图片，开始游戏。在窗口的上方记录游戏开始时间和当前时间。程序将会把图片分成 9 份，重新打乱。玩家可以使用“←”、“↑”、“→”和“↓”键来移动小图片，直到将 9 张小图片拼成一张大图片。

本实例主要讲述程序的窗体设计，以及如何将一张图片分割成大小相等的 9 部分。

图 36-1　效果图

编程思路

本实例利用 C++ Builder 中的画布对象导入原始图像，然后使用 Canvas. CopyRect 方法将图像分成 3×3 的 9 部分，并将它们存储到不同的 Image 控件中。

创作步骤

1．启动 C++ Builder，打开一个新的标准工程。如果 C++Builde 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向窗体中添加 10 个小的 Panel 控件，位置如图 36-1 中的用户控制区，并向每个 Panel 控件中添加一个 Image 控件用来显示被分割的图片。在窗体的右侧部分添加一个较大的 Panel 控件，在该控件中添加一个 Image 控件，用来显示整张图片。再向窗体中添加一个 OpenPictureDialog 控件、一个 PopupMenu 控件、一个 MainMenu 控件和一个 Timer 控件。最后向窗体中添加五个 Label 控件用来显示时间。

2．设置 MainMenu 控件和 PopupMenu 控件的菜单项，如图 36-2 所示。

图　36-2

3．设置窗体的关键属性如下：

BorderIcons = [biSystemMenu, biMinimize]

BorderStyle = bsSingle

Caption = 拼图游戏’

Menu = MainMenu1

OldCreateOrder = False

PopupMenu = PopupMenu1

其他控件的属性设置比较简单，这里不再介绍。

4．单击 Project→Options 菜单项，将会出现项目属性设置对话框，选择 Application 标签页，如图 36-3 所示。单击 Load Icon 按钮，将会弹出选择文件图标对话框，在其中选择一个有特色的图标，然后单击 OK 按钮。

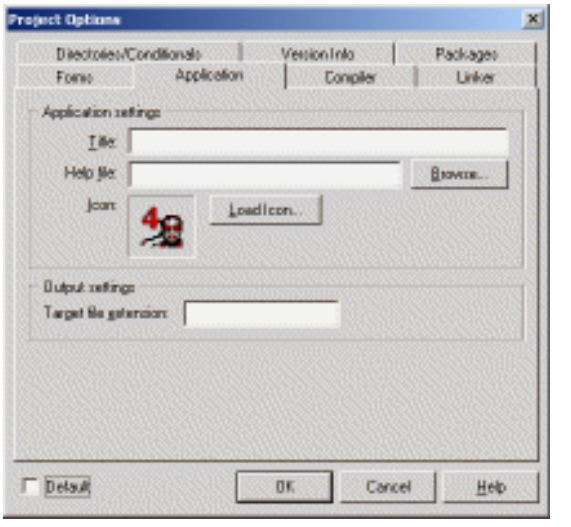


图 36-3

5．在程序的开始部分定义了多个变量，各变量的类型如下：

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;
  table=array[1..3]of integer;
  tab=array[1..3,1..3] of integer;
var
  Form1: TForm1;
  bm:tbitmap;
  x,y:table;
  number:tab;
```

```
  m:integer;
```

6．当程序导入一张位图时，将会把位图分割为 9 部分，其代码如下：

```
procedure TForm1.N2Click(Sender: TObject);
// 打开位图
var
    wi,he:integer;
    i,j,p,t,r,q:integer;
    myrect,otherrect:trect;
    timenow:tdatetime;
    rox,roy:integer;
begin
    ……（此处代码略，详见光盘）
     // 储存图片信息
       for i:=1 to 3 do begin
           for j:=1 to 3 do begin
            number[i,j]:=i+(j-1)*3;
            end;
            end;
         panel1.Caption:=inttostr(number[1,1]);
         panel2.Caption:=inttostr(number[2,1]);
         panel3.Caption:=inttostr(number[3,1]);
         panel4.Caption:=inttostr(number[1,2]);
         panel5.Caption:=inttostr(number[2,2]);
         panel6.Caption:=inttostr(number[3,2]);
         panel7.Caption:=inttostr(number[1,3]);
         panel8.Caption:=inttostr(number[2,3]);
         panel9.Caption:=inttostr(number[3,3]);
         panel10.Caption:='';
         timenow:=time;
         label4.Caption:=timetostr(timenow);
      end;
      bm.FreeImage;
end;
```

实例 37　趣味图画组件

实例说明

续前面的实例。如图 37-1 所示。

本例主要实现如何响应键盘操作，也就是如何响应 FormKey Down 事件，在该事件中判断用户的操作，然后做出相应的响应。在该事件中要判断图片是否可以移动，同时玩家移动之后判断是否已经把图片拼好，如果是，则显示成功信息。

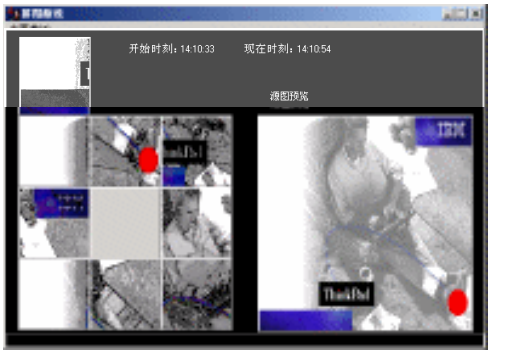


图 37-1　效果图

编程思路

本实例主要是响应玩家的按键，在 Delphi 中提供了一系列的虚拟键值，这些键值的含义如下：

VK_LEFT = 37：对应向左移动键

VK_UP = 38：对应向上移动键

VK_RIGHT = 39：对应向右移动键

VK_DOWN = 40：对应向下移动键

创作步骤

本实例关键是响应键盘操作，用以移动小图像，其代码如下：

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
   Shift: TShiftState);
// 响应按键操作
begin
      if panel1.Caption=”
        then begin
          if key=vk_up then begin
                image1.picture:=image4.picture;
                image4.Picture:=nil;
                panel1.Caption:=panel4.Caption;
                panel4.caption:=”;
                      end;
          if key=vk_down then begin
```

```
                image1.picture:=image10.picture;
                image10.Picture:=nil;
                panel1.Caption:=panel10.Caption;
                panel10.caption:='';
                        end;
            if key=vk_left then begin
                image1.picture:=image2.picture;
                image2.Picture:=nil;
                panel1.Caption:=panel2.Caption;
                panel2.caption:='';
                        end;
                end;
        if panel2.Caption=''
           then begin
            if key=vk_up then begin
                image2.picture:=image5.picture;
                image5.Picture:=nil;
                panel2.Caption:=panel5.Caption;
                panel5.caption:='';
                    end;
            if key=vk_right then begin
                image2.picture:=image1.picture;
                image1.Picture:=nil;
                panel2.Caption:=panel1.Caption;
                panel1.caption:='';
                        end;
            if key=vk_left then begin
                image2.picture:=image3.picture;
                image3.Picture:=nil;
                panel2.Caption:=panel3.Caption;
                panel3.caption:='';
                        end;
                end;
```

本例其他代码详见光盘。

实例 38　扫　　雷

实例说明

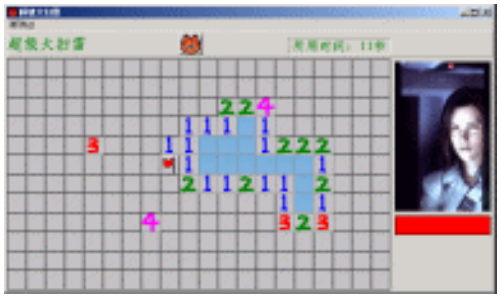


图 38-1　效果图

本例编写“扫雷”游戏，如图 38-1 手示。

本例的游戏规则与 Windows 自带的“扫雷”相同。

本实例中主要完成窗体设计，以及定义一些变量和常量。

编程思路

本实例使用 Delphi 6.0 来实现，扫雷游戏定义了一个 20×12 的雷区，其中有 48 个方格是有雷的地方。对整个雷区是否被探开或标记也定义一个二维数组：

flag:array[1..20,1..12] of byte;其中：

- 未被探开的方格：flag[i,j]:=0
- 已被探开的方格：flag[i,j]:=1
- 标记有雷的方格：flag[i,j]:=2

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，执行 File→New→Application 菜单项，新建一个工程。向窗体中添加三个 Image 控件、一个 Label 控件、一个 MainMenu 控件、一个 Timer 控件和一个 Panel 控件，并且向 Panel 控件中添加一个 Bevel 控件、一个 Label 控件和一个 SpeedButton 控件。各控件之间的关系如图 38-2 所示，向右侧的 Image2 中添加一张图片，在 Image 中不添加任何东西，它们的布置如图 38-1 所示。

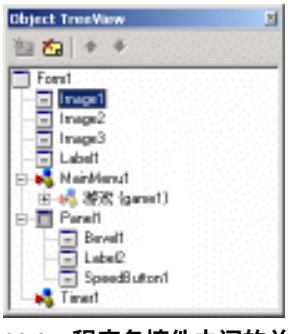


图 38-2　程序各控件之间的关系

2．设置 MainMenu 控件的关键属性如下：

```
object game1: TMenuItem
  Caption = '游戏'
  object N1: TMenuItem
    Caption = '开始'
    OnClick = SpeedButton1Click
  end
  object N2: TMenuItem
    Caption = '退出'
```

OnClick = N2Click

End

3．设置 Timer 控件的 Interval 属性为 1000。

4．定义图片，当雷区未被探开时，方格有三种状态：如图 38-3 所示：

k1　　ks　　kf

图 38-3　雷区未被探开时的三种状态

这三种状态定义为 Tbitmap 类型：

k1,kf,ks:tbitmap;

当初始化时，flag[i,j]=0，方格显示 k1。其中鼠标左键按下时，显示 ks。当鼠标右键单击未被探开的方格时，flag[i,j]=2，方格显示 kf。

当雷区被探开时，方格有 10 种状态：如图 38-4 所示：

fk0　fk[1]　fk[2]　fk[3]　fk[4]　fk[5]　fk[6]　fk[7]　fk[8]　fk[9]

图 38-4　雷区探开时的十种状态

这十种状态实现是通过构造一个 Tbitmap 类型的一维数组实现的。定义如下：

fk:array[1..9] of tbitmap;

fk0:tbitmap;

对整个雷区的状态用一个二维数组来定义：

type map=array[1..20,1..12] of byte;

private

map1:map;

当 flag[i,j]=1，map1[i,j]=0 时，方格显示 fk0。

当 flag[i,j]=1，map1[i,j]=1 时，方格显示 fk[1]。

当 flag[i,j]=1，map1[i,j]=2 时，方格显示 fk[2]。

当 flag[i,j]=1，map1[i,j]=3 时，方格显示 fk[3]。

当 flag[i,j]=1，map1[i,j]=4 时，方格显示 fk[4]。

当 flag[i,j]=1，map1[i,j]=5 时，方格显示 fk[5]。

当 flag[i,j]=1，map1[i,j]=6 时，方格显示 fk[6]。

当 flag[i,j]=1，map1[i,j]=7 时，方格显示 fk[7]。

当 flag[i,j]=1，map1[i,j]=8 时，方格显示 fk[8]。

当 flag[i,j]=1，map1[i,j]=9 时，方格显示 fk[9]。

5．窗体创建时，首先生成了一些位图对象，然后对变量进行初始化。窗体创建响应程序代码如下。这里 k0，k1，fk0，kf，ks，fk[1..9]，life[1..4]都是 Tbitmap 类型的对象，每一个对象装载一个位图文件。k0，k1，fk0，kf，ks，fk[1..9]装载的位图很简单，都是正方形。边长是 30 个像素点。每张位图显示雷区的不同状态。在窗体创建时，这些位图被载入，供重画调用，直到窗体撤消才被释放。全局变量 lifenum 用来记录生命值，每当踩中一个地雷，生命值减 1，当生命值减为 0 时，游戏失败结束。

创建窗体的代码详见光盘。

实例 39　扫雷初始化

实例说明

续前面的实例，如图 39-1 所示。

本例主要对地图进行初始化，确定雷区的位置；响应 Timer 控件的 OnTimer 事件计算玩家所用的时间；另外还设定了程序菜单的各种响应代码，包括“开始”菜单项和“退出”菜单项的内容。

图 39-1　效果图

编程思路

本实例的一个关键点就是如何对雷区进行初始化，这里编写了一个 Init 过程对游戏进行初始化，该过程首先判断用户选择的是什么类型的地图，然后利用一个循环给地图的位置定义坐标，在利用 drawmap 函数绘制地图，接下来使用随机函数初始化雷区的布置，最后等待玩家的的响应。

创作步骤

程序设计步骤续前面的实例。

1．单击 TspeedButton 时，初始化雷区的地雷分布，并对雷区的初始状态进行绘制。事件响应代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
        image1.Enabled:=true;
        init(map1);
        drawmap(1);
        image3.Canvas.Draw(0,0,life[4]);
        lifenum:=3;
        label1.Caption:='';
end;
```

2．上面一段程序调用两个过程。这里首先来看初始化数据结构的 init 过程。Init 过程有一个参数。这个参数是一个 20*12 的二维数组，用于保存雷区的地雷分布状况。Init 过程实现了地雷随机分布的初始化。本过程中还初始化了时间控件，并从本过程开始计时。

Init 过程的程序代码如下：

```
procedure tform1.init(var map1:map);
```

```
var i,j,c:integer;
begin
    for i:=1 to 20 do
      for j:=1 to 12 do
        begin
          map1[i,j]:=0;
          flag[i,j]:=0;
        end;
    randomize();
    c:=1;
    while c<=49 do
     ……（此处代码略，详见光盘）
      d:=0;
      time:=0;
      timer1.Enabled:=true;
end;
function tform1.judge(j:byte):byte;
begin
  if j=9 then
      judge:=1
  else
      judge:=0;
end;
```

3．对雷区初始化状态的绘制，调用了 drawmap 过程。这个过程先获得画布的绘制区域，根据雷区标记的二维数组 flag[I,j]和雷区分布的二维数组 map[I,j]判断每一个方格的绘制方法。初始化时雷区的标记全是 0，所以雷区全都显示 k1，即□，处于雷区未被探开的状态。

程序代码详见光盘。

4．本实例中 Timer 控件的 OnTimer 事件用来计算扫雷使用的时间，其代码如下：

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  time:=time+1;
  label2.Caption:='Ë ùÓÃÊ±?ä£o'+inttostr(time)+'秒';
end;
```

5．“开始”菜单用来开始游戏，其代码详见光盘。

6．“退出”菜单用来退出程序，其代码如下：

```
procedure TForm1.quitClick(Sender: TObject);
begin
     close;
end;
```

实例 40　扫雷组件

实例说明

续前面的实例，如图 40-1 所示。

本例主要用来编写用户的 MouseMove 事件、MouseUp 事件，这两个事件是玩家最常使用的，通过它们可以判断玩家扫雷的进程。

图 40-1　效果图

编程思路

当处于鼠标按下的状态，即 d=1 时，移动鼠标，调用重绘雷区的函数，并将光标所在的方格区域置为按下状态 ks，即 。当鼠标未按下时，不做任何具体操作。鼠标键弹起时，调用了用于探开方格的左键单击的事件处理过程 kgdeal、雷区显示区域重绘过程 drawmap、游戏结束条件的判断过 judgeover。踩中了地雷后，如果生命值减为了 0，游戏结束。如果生命值不为 0，显示地雷爆炸。如果区域全部探开，显示过关！

创作步骤

程序设计步骤续前面的实例。

1．鼠标移动时触发鼠标移动事件 Image1MouseMove，程序代码如下：

```
procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
   Y: Integer);
var i,j:byte;
begin
    i:=(x div 30)+1;
    j:=(y div 30)+1;
    if i>20 then
        i:=20;
    if j>12 then
        j:=12;
    if (d=1) and (flag[i,j]=0)    then
       begin
            drawmap(1);
            image1.Canvas.Draw(i*30-29,j*30-29,ks);
```

```
        end;
    end;
```

2．当处于鼠标按下的状态，即 d=1 **时，移动鼠标，调用重绘雷区的函数，并将光标所在的方格区域置为按下状态** ks**，即　。当鼠标未按下时，不做任何具体操作。鼠标键弹起时，触发鼠标弹起事件** Image1MouseUp**。程序代码如下：**

```
    procedure TForm1.Image1MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    var i,j:byte;
    begin
       i:=(x div 30)+1;
       j:=(y div 30)+1;
       if i>20 then
           i:=20;
       if j>12 then
           j:=12;
       ……（此处代码略，详见光盘）
            image3.Canvas.Draw(0,0,life[l+1]);
            lifenum:=l;
            if lifenum=0 then
              begin
                 label1.Caption:='游戏结束！';
                 image1.Enabled:=false;
              end
            else
               if lifenum<3 then
               label1.Caption:='爆炸了！';
       end;
       if lifenum<>0 then
          begin
              k:=0;
              for i:=1 to 20 do
                  for j:=1 to 12 do
                     if   (flag[i,j]=1) or (flag[i,j]=2) then
                        begin
                           k:=k+1;
                        end;
              if k=240 then
              label1.Caption:='过关！';
          end;
    end;
```

实例 41　UFO 攻击游戏

实例说明

本例编写用导弹攻击 UFO 的小游戏。效果如图 41-1 所示。

游戏开始有声音提示信息。通过点击鼠标开始新游戏。新的 UFO 不断地从游戏界面的顶部产生，并以不规则的动作向下方移动。一旦有一个 UFO 移动到最下方，游戏失败并结束。游戏界面的下部有导弹发射器，玩家通过点击鼠标发射导弹。

游戏的右上方会显示游戏者的战绩。每当击中一个 UFO，战绩就会加 1。

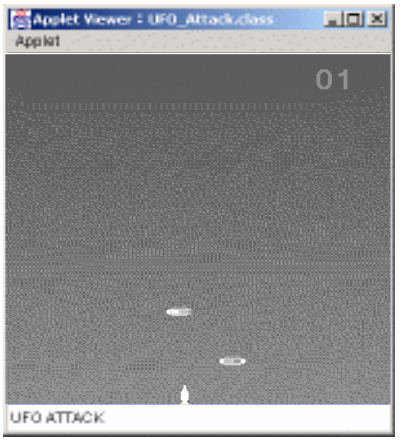


图 41-1　效果图

编程思路

本实例使用 JDK1.3 来实现。程序的主类是 UFO_Attack 类。另外还有一个基本类 Piece 和四个继承自 Piece 的类：Launcher, Missile, UFO 和 Explosion。

创作步骤

1。设计 UFO_Attack 类。它的主要属性如下：

```
public class UFO_Attack extends Applet implements Runnable {
  Image buffer      = null ;       // 临时图像缓存
  Image backdrop = null ;    // 背景图像
  Image bgimg       = null ; // 原来的背景图像
  Image ufostrip = null ; // UFO 序列图像
  Image missile    = null ; // Misile  序列图像
  Image missile_explosion = null ; // 导弹爆炸
  Graphics    buf_g = null ; // 缓存图像对象
  Graphics    bkd_g = null ; // 背景图像对象
 // 这是几个声音剪辑，分布代表爆炸，新 UFO 产生和导弹发射时的声音
  AudioClip explosion = null ;    AudioClip newufo = null ;    AudioClip missile_launch = null ;
  boolean game_over = true ;    // 游戏状态标志
```

```
        Launcher L    = null ;
        Missile    M    = null ;
        Vector      UV = new Vector() ;
        Vector      EV = new Vector() ;
      // 几种颜色
        Color gunColor;……..
```

2．UFO_Attack 类的主要方法：

```
 public void init() {                    // 初始化程序，调入几种图像并设置好颜色
 public void start() {                   // 改变光标形状，显示启动画面和提示信息，提醒游戏者点击鼠标
来开始新游戏，初始化导弹发射器
          mouse_x = window_size.width/2 ;
                    L = new Launcher(this) ;
                    L.set_color(gunColor) ;
// 初始化导弹
                    M = new Missile(this) ;
                    M.set_color(mColor) ;
// 载入声音文件
          if (explosion == null)
             explosion = getAudioClip(getCodeBase(),"explosion.au") ;
          if (newufo == null)
             newufo       = getAudioClip(getCodeBase(),"sonar.au") ;
          if (missile_launch == null)
             missile_launch = getAudioClip(getCodeBase(),"rocket.au") ;
          game_over = true ;
      // 演奏声音
          newufo.play() ;      missile_launch.play() ;       explosion.play() ;
 }
 public void stop() {// 将光标返回原来的形状
 public void display_score() { // 显示战绩
 public void display_game_over() {    // 显示游戏结束信息
 public boolean mouseMove(Event e, int x, int y) { // 处理鼠标移动事件
 public boolean mouseDown(Event e, int x, int y) { // 处理鼠标点击事件
public Frame getFrame(Component c) { // 获取 Applet 的当前帧
 public void run() { // 主要的运行体
// 等待图像的载入
// 绘制背景缓存
          buf_g = backdrop.getGraphics();
          buf_g.drawImage(bgimg,0,0,window_size.width,window_size.height,this) ;
// 绘制屏幕
```

```
buf_g = getGraphics();
buf_g.drawImage(backdrop,0,0,this) ;
        repaint() ; // 显示缓存
        display_score() ;
        L.draw() ;
        showStatus("UFO ATTACK") ;
            for (;;) {// 消息循环
          ti = System.currentTimeMillis() ;
  // 如果有足够的空间容纳更多的 UFO，试着随机性地增加一个
            if (score > 10 && Math.random() > 0.7) U.vy -= 1 ;
            UV.addElement(U) ;
          }
          // 在背景上面绘制爆炸的图像，并在结束的时候消除它
          for (int j=EV.size()-1; j>=0 ; --j) {
            E = (Explosion) EV.elementAt(j) ;
            if (E.active) {
              // System.out.println("Drawing Explosion " + j) ;
              E.draw() ;
                    }
                    else {
              E.erase() ;
           EV.removeElementAt(j) ;
            }
          }
        // 移动发射器和导弹
        // 移动每个 UFO
          for (int i=0; i < UV.size(); ++i) {
  // 控制难度，每当 10 个 UFO 被摧毁之后，就将最大 UFO 数增加 1
  // 但最大是 5 个
                        if ((NU < 5) && (score % 10) == 1)    ++NU ;
              // 消除 UFO 并禁止它
              // 消除 UFO 击中
              // 显示爆炸现象
                    E = new Explosion(this,U.px,U.py) ;
                    EV.addElement(E) ;
              // 绘制 UFO 或者从列表中删除它
              // 如果一个 UFO 到达最底部，游戏结束，玩家失败
```

实例 42　UFO 发射器

实例说明

续前面的实例。发射器随着游戏者的鼠标在左右方向移动。但上下方向不移动。

玩家要尽量做到百发百中，如果一枚导弹没有击中，UFO 就可能在这段时间中移动到底部，导致游戏失败。

游戏的难度是逐渐增加的。难度的增加是通过增加 UFO 的数量来体现，即同时在界面上出现的 UFO 数目。数目越大，游戏难度即越大。但最多不会超过 5 个。

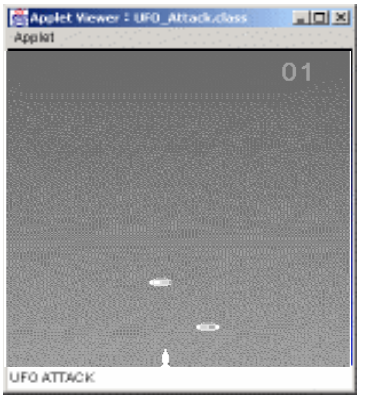


图 42-1　效果图

编程思路

本例介绍基本类 Piece 和四个继承自 Piece 的类：Launcher, Missile, UFO 和 Explosion。这四个继承类分别代表了发射器、导弹、UFO 和爆炸。其中每个继承类的 draw()方法绘制图像，都不相同。

创作步骤

程序设计步骤续前面的实例。

1。设计 Piece 类。Piece 类抽取了四种对象的共同特征，比如判断是否碰撞和相应的绘制图像及擦除图像的方法。它的主要方法如下：

```
public boolean collision(Piece p) {
    int dpx = Math.abs(px - p.px) ;
    int dpy = Math.abs(py - p.py) ;
// 判断是否处于相撞击状态
// 即两个物体的距离小于给定的距离，包括高度和宽度
// w 和 h 的值和影响到 dpx 和 dpy 的 px 及 py 值是由预先设定的。
// 对扩展的 Launcher,UFO,Missile 和 Explosion 类来说，预先设定的值并不相同
    if ((dpx < (Math.max(w/2,p.w/2))+1) && (dpy < (Math.max(h/2,p.h/2)+1)))
        return true ;

    return false ;
}
```

```
    public void draw() { // 绘制图像
    public void erase() {
        // 擦掉图像函数，先将背景获取，然后拷贝加入缓存中，
        // 再从缓存中调出绘制到屏幕上。
```

2．设计 Launcher 类，主要方法是：

```
public Launcher (UFO_Attack a) {   // 构造方法，初始化 w,h,px,py,opx 和 opy 等的值。
    this.a = a ;
    w   = 12 ;
    h   = 22 ;
    px = opx = a.window_size.width/2 ;
    py = opy = w/2+1 ;
    active = true ;
    img = a.missile ;
  }
// 处理移动事件，使得发射器在左右方向的移动尽量和玩家的鼠标移动相吻合
public void move() {
    opx = px ;      opy = py ;
    int dx        = a.mouse_x - px ;
    int abs_dx = Math.abs(dx) ;
    int step = 1 ;
    if (abs_dx > 10)          step = 5 ;
    else if (abs_dx > 1)          step = abs_dx/2 ;
    if (dx != 0) {
      px += step*(dx/abs_dx) ;
      if (px < w/2)
        px = w/2 ;
      else if (px > (a.window_size.width - w/2))
        px = a.window_size.width - w/2 ;
    }   }
  public boolean has_moved() {// 判断是否已经移动完成
  public void draw() { // 绘制图像
```

3．设计 Missile 类。

```
  public Missile (UFO_Attack a) { // 构造方法
  public void move() { // 处理导弹的移动，使得移动速度更加实际
```

4．设计 UFO 类。

```
public void draw() {
    set_draw_rectangles(a.paint_area, a.new_area) ;
    // 清除旧的图像
    Graphics bg = a.buffer.getGraphics() ;
```

```
        bg.clipRect(a.paint_area.x, a.paint_area.y, w, h);
        bg.drawImage(a.backdrop,0,0,a) ;
        bg.dispose() ;
        // 选择图像序列，每四次绘制改变一次帧。
        if ((++seq2 % 4) == 0) seq = ++seq % 4 ;
         // 添加旧的和新的区域来预备重新刷新
        a.paint_area.add(a.new_area) ;
        Graphics g = a.getGraphics() ;
        g.clipRect(a.paint_area.x ,a.paint_area.y, a.paint_area.width, a.paint_area.height);
        g.drawImage(a.buffer, 0, 0, a);
        g.dispose() ;
    }}
```

5．**设计** Explosion **类。**

```
 public void draw() {
        // 清除旧的图像
        Graphics bkd_g = a.backdrop.getGraphics();
        bkd_g.clipRect(a.paint_area.x, a.paint_area.y, w, h);
        bkd_g.drawImage(a.bgimg,0,0,a.window_size.width,a.window_size.height,a) ;
        if ((++seq2 % 4) == 0) seq = ++seq % 5 ;
        // 在已经绘制完成最后一次爆炸后，将 active 属性设置为 false
        if (seq == 4) active = false ;
        // 将新的区域绘制到缓存中
        // 将改变写入缓存
        // 将改变写入屏幕缓存
        Graphics g = a.getGraphics() ;
        g.clipRect(a.paint_area.x ,a.paint_area.y, a.paint_area.width, a.paint_area.height);
        g.drawImage(a.buffer, 0, 0, a);
        g.dispose() ;
    }
    public void erase() {
        set_draw_rectangles(a.paint_area, a.new_area) ;
        // 清除旧图像
        Graphics bkd_g = a.backdrop.getGraphics();
        bkd_g.clipRect(a.paint_area.x, a.paint_area.y, w, h);
        bkd_g.drawImage(a.bgimg,0,0,a.window_size.width,a.window_size.height,a) ;
        bkd_g.dispose() ;
        // 同样对缓存和屏幕进行操作
        super.erase() ;
    }}
```

实例 43　俄罗斯方块

实例说明

本例通过 Java 编写俄罗斯方块小游戏。效果如图 43-1 所示。

游戏启动后，玩家可通过键盘控制游戏，p 键暂停，S 键启动游戏，+键增加游戏执行速度，-键减少速度，上键可以旋转下落的方块，下键能快速地将方块落入最下面，r 键就绪。游戏界面可以看到已经成功销掉的行数和游戏的当前速度，右下脚可以预览到将要产生的下一个方块的形状。下一个形状、颜色和旋转样式都是随机产生的。

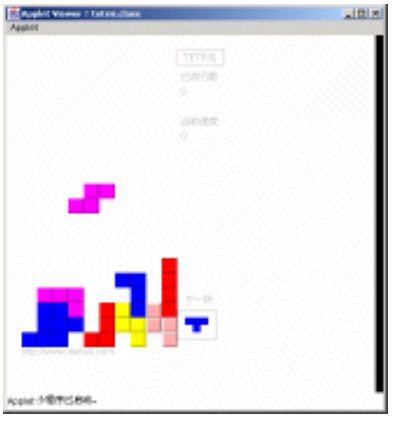


图 43-1　效果图

编程思路

本例通过 Applet 的 Tetris 类实现了 Runnable 接口。Tetris 类的几个属性定义了游戏的数值和状态：如 SHAPE 数组存储了各种方块的形状及其旋转样式。Ground [10][20]定义在 10×20 的场地游戏,预留顶部放置方块的空间一格。CurrentColorIndex 是当前造型色彩索引；NextShapeIndex 代表下一块出现的造型的索引；CurrentMode 记录游戏状态变量等。根据玩家不同的击键，游戏进行分析，并进行相应地变换，从而模拟俄罗斯方块的游戏。

创作步骤

1．Tetris 类有很多属性，记录游戏的状态等。如 SHAPE 数组是一个四维数组，里面存储了各种方块的形状及其旋转样式。如下面代码：

```
byte SHAPE[][][][] = // [造型索引][旋转索引][4][坐标]
    {           {   // 造型二
            {{-1, 0},{ 0, 0},{ 0, 1},{ 1, 1}}, // 旋转一                    []
            {{ 0,-1},{ 0, 0},{-1, 0},{-1, 1}}, // 旋转二   []()        []()
            {{-1, 0},{ 0, 0},{ 0, 1},{ 1, 1}}, // 旋转三     [][]      []
          {{ 0,-1},{ 0, 0},{-1, 0},{-1, 1}}   // 旋转四
            …..
```

2．定义游戏中出现的状态

```
int CurrentX;                    // 当前 X 左边
int CurrentY;                    // 当前 Y 坐标
int CurrentShapeIndex;           // 当前造型索引
int CurrentTurnIndex;            // 当前旋转索引
int CurrentColorIndex;           // 当前造型色彩索引
int NextShapeIndex;              // 下一块出现的造型索引
int GameSpeed = 0;               // 游戏速度
int SpeedVar = 0;                // 延时计数
int FlashSpeed = 60;             // 信息提示闪烁速度
int FlashVar = 0;                // 闪烁延时计数
int ClearTotalLine = 0;          // 消掉的总行数
```

3．Tetris 类的重要方法

```
public void MakeNextShape()    // 随机产生下一个形状
    {
        NextShapeIndex = (int)(Math.random()*70) /10;
        NextShape = SHAPE[NextShapeIndex][0];
    }
public void ClearGround()    // 清除场地
    {
        for(int i=0; i<10; i++)
            for(int j=0; j<20; j++) Ground[i][j]=0;
        CurrentMode = READY;
        ClearTotalLine = 0;
    }
    public void StartGame()    // 开始游戏
    {
        ClearGround();
        CurrentShapeIndex = (int)(Math.random()*70) /10;
        CurrentX = 4;
        CurrentY = 0;
        CurrentMode = GAMING;
        CurrentColorIndex = (int)(Math.random()*70) /10 +1;
        MakeNextShape();
    }
  public void drawABlock(Graphics g,int x,int y,int colorIndex)    // 用指定的颜色画一个方块
    {
        g.setColor(Colors[colorIndex]);
        g.fill3DRect(BaseX + x * BlockSize, BaseY + y * BlockSize , BlockSize, BlockSize, true);
        g.fill3DRect(BaseX + x * BlockSize + 1, BaseY + y * BlockSize +1, BlockSize -2, BlockSize-2,
```

```
true);
        }
        public void drawShape(Graphics g,int x,int y, int shapeindex,int trunindex) // 在指定的位置画指定的
造型
        {
            for(int i=0; i<4; i++)
            {
                if(y+SHAPE[shapeindex][trunindex][i][1]>=0)     // 不画最顶上不在区域的块
                    drawABlock(g,x+SHAPE[shapeindex][trunindex][i][0]
                                    ,y+SHAPE[shapeindex][trunindex][i][1],CurrentColorIndex);
            }       }
        public void drawGround(Graphics g)    // 画整个画面
    {……
    }
    public void drawNextBlock(Graphics g)    //    画下一块
        {
            g.setColor(Color.lightGray);
            g.drawString("下一块",BaseX+11*BlockSize - BlockSize / 2,BaseY+17*BlockSize);
            g.drawRect(BaseX + BlockSize * 10, BaseY + BlockSize *18 - BlockSize / 2
                                        ,BlockSize * 2 + BlockSize / 2,BlockSize * 2);
            g.setColor(Color.blue);
            for(int i=0; i<4; i++)
                g.fill3DRect(BaseX + 11 * BlockSize + NextShape[i][0]*BlockSize / 2
                            , BaseY + 18 * BlockSize + NextShape[i][1]*BlockSize / 2 , BlockSize / 2 ,
BlockSize /2, true);
        }
    public void drawInfo(Graphics g)    // 绘制提示的信息
        {
            g.setColor(Color.white);
            Font old = g.getFont();
            g.setFont(new Font("宋体",0,24));
            g.drawString(InfoStr,BaseX+5*BlockSize - InfoStr.length()* 7 ,BaseY+10*BlockSize);
            g.setFont(old);
        }
      public boolean CanMoveTo(int shape,int turn,int x,int y)    // 判断是否能移动到指定位置
        {……}
    public synchronized void DownIt()    // 处理下落过程
    public boolean keyDown(Event e, int key) // 处理键盘事件
```

实例 44　数字魔方人机交互

实例说明

本例编写数字魔方游戏，游戏画面如图 44-1 所示。

游戏规则如下：计算机首先打乱棋盘中的数字顺序，玩家使用鼠标单击没有数字位置的四周的按钮，则被单击的按钮将会移动到空位上，同时按钮原来的位置变为空位。以此来移动按钮，直到所有按钮回到初始位置为止。

本游戏可以选择棋盘的大小，有 3*3、4*4、5*5、6*6、7*7 五种选择。棋盘越大，难度越大。在游戏开始之后，在窗口的标题栏将会记录游戏所用的时间，当玩家成功后，将会出现鼓掌欢庆的音乐。

在本实例中主要实现了程序界面的设计。

图 44-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。在程序中使用了 Timer 控件和 ComboBox 控件。这些控件在游戏制作中有着广泛的应用，同时还使用菜单编辑器制作了一个标准菜单。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加一个 Timer 控件、两个 CommandButton 控件、一个 Label 控件和一个 ComboBox 控件。如图 44-2 所示。

2．设定 Timer 控件的关键属性如下：

Enabled =　　0 'False

Interval = 1000

3．设定 ComboBox 控件的关键属性如下：

Height = 315

ItemData = "PUZZLE.frx":030A

Left = 720

List = "PUZZLE.frx":030C

Style = 2　'Dropdown List

TabIndex = 1

Top　= 4440

Width = 975

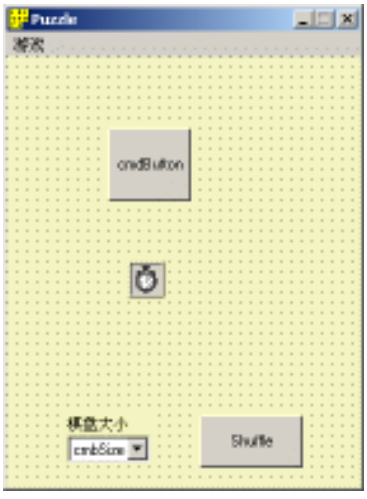


图　44-2

4．**单击工具栏中的菜单编辑器按钮，弹出菜单编辑器，设置程序菜单如图** 44-3 **所示。**

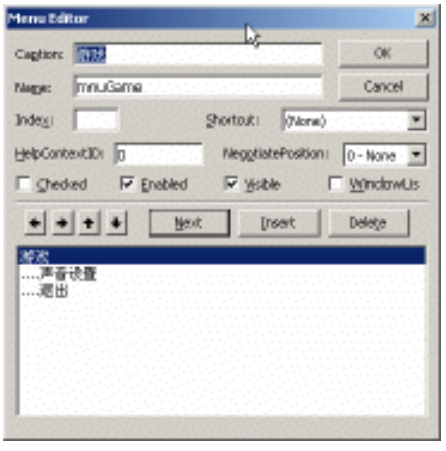


图　44-3

实例 45　数字魔方组件

实例说明

续前面的实例，如图 45-1 所示。

本例主要介绍了程序的代码实现，在程序的 Load 事件中初始化棋盘，默认情况下是 4×4 的棋盘。

在响应玩家的 Mouse Down 事件中，首先判断其单击的按钮四周是否有空位，如果有，则将其移动到空位上，同时把按钮原来的位置设置为空位。每次玩家单击完之后，判断玩家是否已经成功，如果成功则显示出胜利的提示信息。

图 45-1　效果图

编程思路

在程序的 Load 事件中初始化棋盘，产生一个 4×4 的棋盘，当用户单击“新的一局”按钮后，将开始游戏。Visual Basic 利用 Timer 控件计算时间，并且在改变标题栏的内容，显示玩家所用时间。当玩家单击数字按钮时，程序判断按钮的上、下、左、右是否有空位，如果有，将按钮移到空位上，并且将按钮以前的位置置为空位；如果没有则不移动按钮。

创作步骤

程序设计步骤续前面的实例。

1．向程序中添加一个模块（Module），并且向其中添加如下代码：

```
Option Explicit
Private Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" _
    (ByVal lpszSoundName As String, ByVal uFlags As Long) As Long
播放声音
Private Const SND_SYNC = &H0
Private Const SND_ASYNC = &H1
Private Const SND_NODEFAULT = &H2
Private Const SND_LOOP = &H8
Private Const SND_NOSTOP = &H10
Public Sub PlaySound(strSound As String)
```

```
    Dim wFlags%
    wFlags% = SND_ASYNC Or SND_NODEFAULT
    sndPlaySound strSound, wFlags%
End Sub
```

2．本实例源代码如下：

```
Option Explicit
Private mbPuzzleSolved As Boolean      布尔变量，
Private miEmptyIndex As Integer        空按钮的序号
Private miSize As Integer              矩阵大小
Private mlTime As Long                 玩家所用的时间
Private Const MIN_SIZE As Byte = 3
Private Const MAX_SIZE As Byte = 7
'Text constants
Private Const TEXT_SHUFFLE As String = "打乱顺序"
Private Const TEXT_NEW_GAME As String = "新的一局"
Private Const TEXT_PUZZLE As String = "数字谜题"
Private Const TEXT_TIME As String = "时间:"
Private Const TEXT_HIGH_SCORE As String = "High score"
Private Const TEXT_SIZE As String = "Size"
Private Const TEXT_TIME_S As String = "Time"
Private Const TEXT_PLAYER As String = "Player"
Private Const TEXT_INPUT_PLAYER As String = "Write your name!"
Private Const TEXT_ANDERS_GAMES As String = "Anders Franssons Made In Home Games"
Private Static Sub Form_Load()
    Dim i%
    '初始化随机数发生器
    Randomize
    '导入按钮
    ……（此处代码略，详见光盘）
                                    Shift As Integer, X As Single, Y As Single)
    Dim i%, xEmpty%, yEmpty%, xClicked%, yClicked%
    '计算按钮的是否可以移动
    xEmpty = (miEmptyIndex) Mod miSize
    yEmpty = (miEmptyIndex) \ miSize
    xClicked = (Index) Mod miSize
    yClicked = (Index) \ miSize
    '如果旁边有空按钮，则改变按钮与空按钮的位置
    If (xClicked = xEmpty + 1 And yClicked = yEmpty) Or _
        (xClicked = xEmpty - 1 And yClicked = yEmpty) Or _
        (yClicked = yEmpty + 1 And xClicked = xEmpty) Or _
```

```
        (yClicked = yEmpty - 1 And xClicked = xEmpty) Then
            ChangeButtons (Index)
            If mnuSound.Checked Then PlaySound App.Path & "\Move.wav"
    End If
    检查问题是不是已经解决了
    For i = 0 To miSize ^ 2 - 2
        If Val(cmdButton(i).Caption) = i + 1 Then
            mbPuzzleSolved = True 检查按钮的序号是不是有序
        Else
            mbPuzzleSolved = False
            Exit For     ’退出循环
        End If
    Next i
    If mbPuzzleSolved Then    如果问题解决，则播放声音
        If (Timer1.Enabled And mnuSound.Checked) Then PlaySound App.Path & "\Applause.wav"
        Timer1.Enabled = False
        mlTime = 0
        cmdShuffle.Caption = TEXT_SHUFFLE
        cmdShuffle.SetFocus
    Else
        cmdShuffle.Caption = TEXT_NEW_GAME
    End If
End Sub
    这个函数打乱数字顺序
Private Sub cmdShuffle_Click()
  If mbPuzzleSolved Then
        Shuffle    打乱数字顺序
    Else
        NewGame    新的一局
    End If
    播放相应的声音
    If mnuSound.Checked Then PlaySound App.Path & "\Shuffle.wav"
  End Sub
Private Sub cmbSize_Click()
    If Not (miSize = cmbSize.Text) Then
        miSize = cmbSize.Text    改变矩阵大小
        NewGame     新开一局
    End If
End Sub
Private Sub mnuExit_Click()
```

```
    Unload Me    退出程序
End Sub
Private Sub mnuSound_Click()
     mnuSound.Checked = Not mnuSound.Checked
End Sub

Private Sub Timer1_Timer()

    mlTime = mlTime + 1 时间秒数＋1
    Me.Caption = TEXT_TIME & " " & mlTime & " s"
    改变窗口标题，显示玩家所用的时间
End Sub
Private Static Sub NewGame()
    Dim i%, j%, iSide%
    Me.Caption = TEXT_PUZZLE 改变窗口标题
    mlTime = 0                          玩家所用的时间置零
    Timer1.Enabled = False              'disable 定时器控件
    mbPuzzleSolved = True
    iSide = Int((90 / miSize)) * 2 + 10
    隐藏按钮，并设定按钮的 Caption
    For i = 0 To MAX_SIZE ^ 2 - 1
        cmdButton(i).Visible = False
        cmdButton(i).Caption = i + 1
    Next i

    再按照当前 miSize 的大小，在屏幕上放置好按钮
    For i = 0 To miSize - 1
        For j = 0 To miSize - 1
            cmdButton(i * miSize + j).Height = iSide
            cmdButton(i * miSize + j).Width = iSide
            cmdButton(i * miSize + j).Left = iSide / 2 + iSide * j
            cmdButton(i * miSize + j).Top = iSide / 2 + iSide * i
            cmdButton(i * miSize + j).Visible = True
        Next j
    Next i
    miEmptyIndex = miSize ^ 2 - 1
    cmdButton(miEmptyIndex).Visible = False    隐藏标号最大的那个按钮
    cmdShuffle.Caption = TEXT_SHUFFLE     设置 cmdShuffle 按钮的标题
End Sub
```

```
' 随机地把按钮打乱顺序
Private Static Sub Shuffle()
    Dim bMove As Boolean
    Dim i%, xCoord%, yCoord%, iRand%

    在打乱顺序之前，隐藏窗口
    For i = 0 To miSize ^ 2 - 1
        cmdButton(i).Visible = False
    Next i
    空位的坐标
    xCoord = (miEmptyIndex) Mod miSize
    yCoord = (miEmptyIndex) \ miSize

    随机地移动 empty button
    i = 0
    While i < miSize ^ 4
        bMove = False
        iRand = Int(4 * Rnd)    '产生一个 0～3 的随机数
    ……（此处代码略，详见光盘）
    For i = 0 To miSize ^ 2 - 1
        cmdButton(i).Visible = True    '使打乱后的按钮可见
    Next i

    cmdShuffle.Caption = TEXT_NEW_GAME    '设置按钮的标题
    cmdButton(miEmptyIndex).Visible = False    '使空的按钮不可见
    mbPuzzleSolved = False        '置成功标记为 False
    Timer1.Enabled = True         '启动定时器控件
End Sub
'与空的按钮交换
Private Sub ChangeButtons(Index As Integer)

    '改变按钮的标题
    cmdButton(miEmptyIndex).Caption = cmdButton(Index).Caption
    cmdButton(miEmptyIndex).Visible = True
    cmdButton(miEmptyIndex).SetFocus
    miEmptyIndex = Index
    cmdButton(Index).Visible = False
    cmdButton(Index).Caption = ""
End Sub
```

实例 46　贪吃蛇人机交互

实例说明

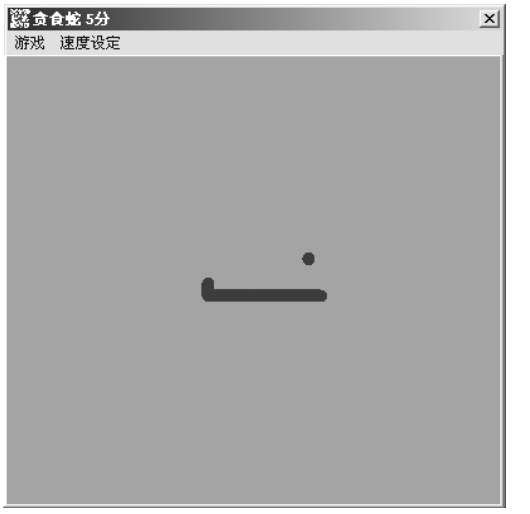


图 46-1　效果图

本例编写贪吃蛇的游戏，如图 46-1 所示。

窗口中随机产生一些食物，贪吃蛇就是吃这些食物来变长。玩家可以使用上、下、左、右键控制贪吃蛇行走的方向。在贪吃蛇找食物的过程中，它不能超过窗口边界，也不可以使贪吃蛇的头部接触身体的任何部分，否则贪吃蛇就会死亡。在游戏菜单中，可以开始游戏、暂停游戏以及退出游戏；在“游戏速度”菜单中，可以设定贪吃蛇的行走速度。

在本实例中主要讲述了程序界面的设计。

编程思路

本实例使用 Visual Basic 6.0 来实现。这里制作了一个对话框类型的窗体，具体方法是设置其 BorderStyle 属性为 Fixed Single，然后利用 Project 菜单中的 Add Form 菜单项，向程序中添加了一个窗体，使程序成为一个多窗口程序。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加一个 Timer 控件。

2．设定 Timer 控件的关键属性如下：

```
Enabled = 0      'False
Interval = 1
```

3．设置窗体的关键属性如下：

```
AutoRedraw      =    -1    'True
BackColor       =    &H80000001&
BorderStyle     =    1     'Fixed Single
Caption         =    "贪食蛇"
```

4．单击工具栏上的菜单编辑器按钮，向窗体中添加菜单，如图 46-2 所示。

5．**单击** Project→Add Form **菜单项，出现如图** 46-3 **所示的窗口，再单击其中的** OK **按钮，向项目中添加一个窗体。**

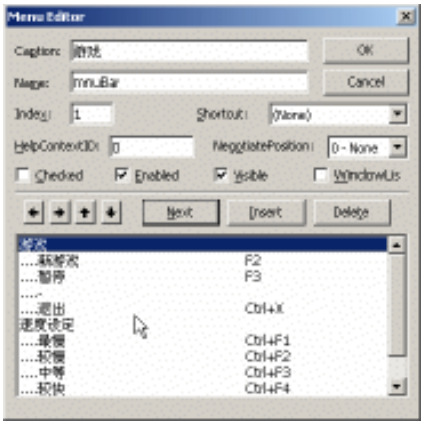


图　46-2

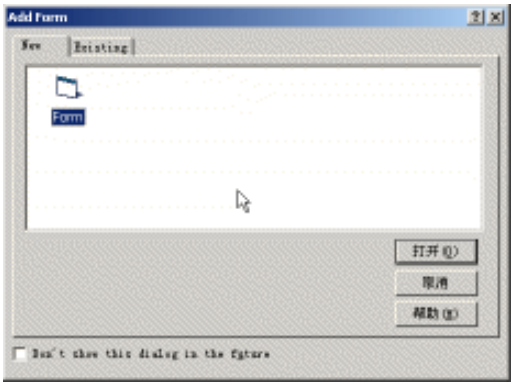


图　46-3

6．**向窗体中添加两个** Label **控件，并设置** Caption **输入，如图** 46-4 **所示。**

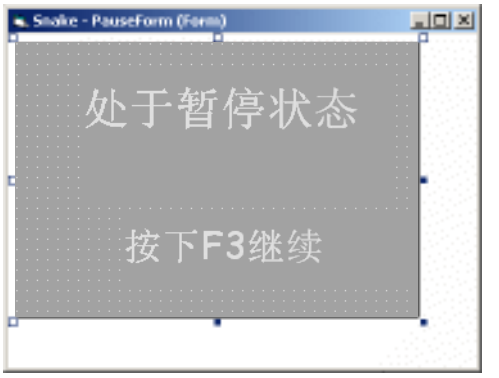


图　46-4

7．**设置该窗体的关键属性如下：**

```
BackColor = &H80000006&
BorderStyle = 1    'Fixed Single
ClientHeight = 3195
ClientLef = 15
ClientTop = 15
ClientWidth = 4680
ControlBox = 0     'False
```

8．**设置该窗体的** KeyDown **事件如下：**

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
'当按下 F3 键时，退出程序
If KeyCode = 114 Then Unload Me
End Sub
```

实例 47　贪吃蛇组件

实例说明

续前面的实例，如图 47-1 所示。

本例主要完成程序的代码设计。通过响应 Form_KeyDown 事件，不断地改变贪吃蛇的移动方向，然后调用相应的函数，对贪吃蛇的状态进行判断，具体的函数的用法详见“编程思路”部分。

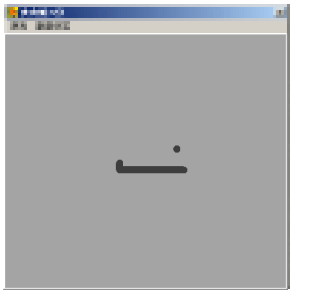


图 47-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。在程序中定义了一些函数来实现相应的功能，其中，使用 CheckDotPosition 函数来判断生成的食物是否在贪吃蛇身上；使用 CheckForDot 函数判断贪吃蛇有没有吃到食物；使用 CheckForLose 函数，判断贪吃蛇是否吃到自己和出边界；使用 CheckSpeed 函数判断蛇的速度；使用 DrawSnake 函数绘制蛇；使用 ExtendSnake 函数延长贪吃蛇的长度；利用窗体的 KeyDown 事件来响应玩家的按键，同时调用上面的函数就可以完成贪吃蛇游戏了。

创作步骤

程序设计步骤续前面的实例。主窗体的代码如下：

```
Option Explicit
' 蛇的坐标，长度
Private theXPoints(), theYPoints(), theSnakeLength As Integer

' 点的坐标
Public xPos As Integer, yPos As Integer, theScore As Integer

' 点的分值，难度
Dim DotIsWorth As Integer, difficulty As Integer

' 蛇的方向
Dim Direction As Byte
Private Const dLEFT As Byte = 1
Private Const dUP As Byte = 2
```

```
Private Const dRIGHT As Byte = 3
Private Const dDOWN As Byte = 4
Private Const TEXT_Caption As String = "贪吃蛇"

'这个函数保证随机生成的点不会放在蛇上
Public Function CheckDotPosition() As Boolean
Dim i As Integer
For i = 1 To theSnakeLength
   If xPos = theXPoints(i) And yPos = theYPoints(i) Then _
      CheckDotPosition = True: Exit Function      '比较蛇的坐标
Next i
CheckDotPosition = False
End Function


'这个函数看蛇有没有吃到点
Public Function CheckForDot() As Boolean
Dim i As Integer
For i = 1 To theSnakeLength - 1    检查蛇的坐标有没有与点重合
   If theXPoints(theSnakeLength) = xPos _
      And theYPoints(theSnakeLength) = yPos Then _
      CheckForDot = True: Exit Function '有则退出程序
Next i
CheckForDot = False
End Function

这个函数检查蛇有没有出边界和吃到自己
Public Function CheckForLose() As Boolean
Dim i As Integer
For i = 1 To theSnakeLength        '检查蛇有没有碰到边界，有则结束
   If theXPoints(i) > Me.ScaleWidth _
      Or theXPoints(i) < 0 Then CheckForLose = True: Exit Function
   If theYPoints(i) > Me.ScaleHeight _
      Or theYPoints(i) < 0 Then CheckForLose = True: Exit Function
Next i
For i = 1 To theSnakeLength - 1        检查蛇头有没有吃到自己
   If theXPoints(theSnakeLength) = theXPoints(i) _
      And theYPoints(theSnakeLength) = theYPoints(i) Then _
      CheckForLose = True: Exit Function
Next i
```

```
CheckForLose = False
End Function

'检查贪吃蛇的速度
Private Sub CheckSpeed()
Dim i As Integer
For i = 1 To 5
   If mnuSpeed(i).Checked = True Then Call mnuSpeed_Click(i): Exit Sub
Next i
End Sub

'这个函数随机生成一个新点
Public Sub DrawDot()
xPos = ((Int(Int(Me.ScaleWidth / 10) * Rnd)) * 10)   随机生成在窗口范围之内的点
yPos = ((Int(Int(Me.ScaleHeight / 10) * Rnd)) * 10) 随机生成在窗口范围之内的点
Do While CheckDotPosition = True   检查点是不是恰好在蛇上，否则循环
   xPos = ((Int(Int(Me.ScaleWidth / 10) * Rnd)) * 10)
   yPos = ((Int(Int(Me.ScaleHeight / 10) * Rnd)) * 10)
Loop
   Me.PSet (xPos, yPos), RGB(0, 0, 0)     生成点
End Sub

Public Sub DrawSnake(ByVal dir As Byte)
Dim i As Integer
'用背景色画第一到第二点的线
Me.Line (theXPoints(1), theYPoints(1))-(theXPoints(2), theYPoints(2)), Me.BackColor
For i = 1 To theSnakeLength - 1
   theXPoints(i) = theXPoints(i + 1) '改变点的坐标
   theYPoints(i) = theYPoints(i + 1) '改变点的坐标
Next i
Select Case dir     '根据方向计算最前面的点的坐标
……（此处代码略，详见光盘）
For i = 1 To theSnakeLength - 1
' 重画蛇
   Me.Line (theXPoints(i), theYPoints(i))-(theXPoints(i + 1), theYPoints(i + 1)), RGB(0, 0, 0)
Next i
'检查玩家有没有失败
If CheckForLose = True Then Timer1.Enabled = False
'检查玩家有没有吃到点，如果吃到生成新的点，蛇的长度增加
If CheckForDot = True Then theScore = theScore + DotIsWorth: DrawDot: Call ExtendSnake _
```

```
: MainForm.Caption = TEXT_Caption & " " & theScore & "分"
End Sub

'当蛇吃到点时，增加蛇的长度
Public Sub ExtendSnake()
Dim i As Integer
theSnakeLength = theSnakeLength + 1 蛇的长度加一
ReDim Preserve theXPoints(1 To theSnakeLength)   重分配数组
ReDim Preserve theYPoints(1 To theSnakeLength)
For i = theSnakeLength To 2 Step -1
   theXPoints(i) = theXPoints(i - 1)   设定新的数组
   theYPoints(i) = theYPoints(i - 1)
Next i
End Sub

'capture 玩家按下的方向键
Public Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = 37 And Direction <> dRIGHT Then 按下 Left 键
   Direction = dLEFT
ElseIf KeyCode = 38 And Direction <> dDOWN Then 按下 Up 键
   Direction = dUP
ElseIf KeyCode = 39 And Direction <> dLEFT Then 按下 Right 键
   Direction = dRIGHT
ElseIf KeyCode = 40 And Direction <> dUP Then 按下 Down 键
   Direction = dDOWN
End If
End Sub

'开始游戏
Public Sub Form_Load()
Randomize          初始化随机数生成器
Call NewGame
End Sub

'初始化蛇的信息
Public Sub InitializeSnake()
Dim i As Integer
ReDim theXPoints(1 To 10)   定义数组
ReDim theYPoints(1 To 10)
theSnakeLength = 10
```

```
For i = 1 To theSnakeLength
   theXPoints(i) = 10
   theYPoints(i) = (10 * i)
Next i
For i = 1 To 9
   Me.Line (theXPoints(i), theYPoints(i))-(theXPoints(i + 1), theYPoints(i + 1)), RGB(0, 0, 0)
   ’画一条线代表蛇
Next i
End Sub

’开始游戏
Public Sub NewGame()
Me.Cls                    ’清空画面
Direction = dDOWN     ’初始方向向下
Call InitializeSnake      ’初始化蛇
Call DrawDot              ’画蛇
theScore = 0
CheckSpeed
End Sub

Private Sub mnuGame_Click(Index As Integer)
Select Case Index
   Case 1 ’New
      NewGame                        开始新的一局
      Timer1.Enabled = True          启动 Timer
   Case 2 ’Pause
      Timer1.Enabled = False
      PauseForm.Show vbModal, Me    ’显示 Pause 对话框
      Timer1.Enabled = True
   Case 6 ’Exit
      Unload Me
End Select
End Sub
’设定蛇的速度
Private Sub mnuSpeed_Click(Index As Integer)
Dim i As Integer
For i = 1 To 5
   mnuSpeed(i).Checked = False     所有菜单项都处于不选中状态
Next i
mnuSpeed(Index).Checked = True    保留一个菜单项 checked
```

```
Select Case Index
   Case 1 ’Slowest
      difficulty = 10
      Timer1.Interval = 150    设定定时器间隔
      DotIsWorth = 5
   Case 2 ’Slow
      difficulty = 10
      Timer1.Interval = 120    设定定时器间隔
      DotIsWorth = 10
   Case 3 ’Medium
      difficulty = 10
      Timer1.Interval = 100     设定定时器间隔
      DotIsWorth = 15
   Case 4 ’Fast
      difficulty = 10
      Timer1.Interval = 80    设定定时器间隔
      DotIsWorth = 20
   Case 5 ’Fastest
      difficulty = 10
      Timer1.Interval = 50     设定定时器间隔
      DotIsWorth = 25
End Select
End Sub
Private Sub Timer1_Timer()
Call DrawSnake(Direction) 画蛇
End Sub
```

第四篇

对抗游戏

本篇总览

对抗类游戏的特点是游戏至少包括甲、乙双方参与(当然可以有多方参加)。

判别胜利、失败的规则有很多,比如五子棋就是哪一方先连成五子,而 21 点是看哪一方的点数多,并且没有超过 21 点就为胜。

本篇主要制作了 21 点、机智加分、黑白棋以及五子棋等游戏。这些游戏的规则比较简单,编写起来不是特别复杂,如果读者想编写复杂的游戏,可以参照麻将的规则,自己编写一个支持多人对战的麻将游戏。

实例 48　21　　点

实例说明

本例编写博弈游戏21点，如图48-1所示。

双方可轮流要点数，也可选择任意时刻停止。双方都停止后计算各自总点数，超过 21 点者为败，都没超过或都超过 21 点时，总点数更靠近 21 点者为胜。本游戏一共可要 7 次数字，每次都是在图 48-1 中随机产生数字，右边的数字显示为各自的累积总点数，其中下面的数字是玩家的点数，上面的数字是电脑的点数。若总点数一样时，则判玩家为胜。

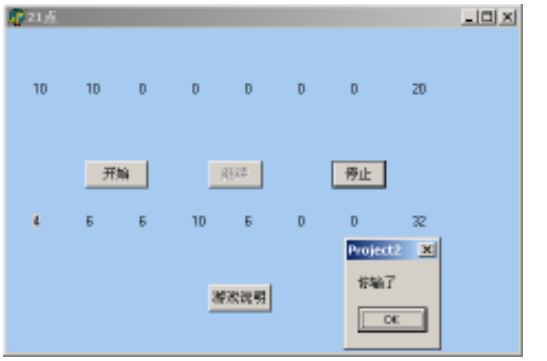


图 48-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。当玩家单击“开始”按钮之后，创建一个 14 维的数组，用来存储玩家和电脑的点数，并且给玩家和电脑每人一个 1~10 之间的随机数。之后如果玩家单击“继续”按钮，则产生第三、四个随机数，此时电脑判断自己是否已经超过 16 点了，如果大于 16 点，则停止要数，如果小于或者等于 16 点，则继续要数。同时判断是否超过了 7 次，如果超过了，则提醒玩家只能选 7 个数字。当玩家单击“停止”按钮时，则按照游戏规则判断哪一方获胜。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向窗体中添加四个 Button 控件和 14 个 Label 控件，并设置所有 Label 控件的 Caption 属性为 0，Button 控件的 Caption 属性为“开始”、“继续”、“停止”、“游戏说明”。

2．设置窗体的关键属性如下：

Left = 192
Top = 107
Width = 435
Height = 300
Caption = '21 点'

3．本程序源代码如下（只列出部分关键代码，其余部分详见光盘）：

```
unit Unit1;
interface
```

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
…;
var
  Form1: TForm1;
  total1:integer;
  total2:integer;
  i:integer;
  labeli:array[1..14] of tlabel;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
labeli[1]:=label1;
labeli[2]:=label2;
labeli[3]:=label3;
labeli[4]:=label4;
labeli[5]:=label5;
labeli[6]:=label6;
labeli[7]:=label7;
labeli[8]:=label8;
labeli[9]:=label9;
labeli[10]:=label10;
labeli[11]:=label11;
labeli[12]:=label12;
labeli[13]:=label13;
labeli[14]:=label14;
{数组中的各个元素}
button2.Enabled:=true;
{击活 button2 按钮}
for i:=1 to 14 do
labeli[i].caption:='0';
{将 label1 到 14 的 caption 值重新设为 0}
i:=1;
{变量 i 设为 1}
randomize;
{开始随机函数}
label1.Caption:=inttostr(random(10)+1);
{随机产生一个从 1 到 10 的数字,并赋给 label1 的 caption}
label8.Caption:=inttostr(random(10)+1);
```

```
{随机产生一个从 1 到 10 的数字,并赋给 label8 的 caption}
total1:=strtoint(label1.Caption);
{将 label1 的值赋给 total1,作为玩家的总点数}
total2:=strtoint(label8.Caption);
{将 label2 的值赋给 total2,作为电脑的总点数}
label15.Caption:=inttostr(total1);
{在 label15 中显示玩家的总点数}
label16.Caption:=inttostr(total2);
{在 label16 中显示电脑的总点数}
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
i:=i+1;
{每单击一次 button2 按钮,变量 i 自身加 1}
if i in [1..7] then
{判断 i 是否在 1~7 中,如果在,则执行以下程序}
begin
labeli[i].Caption:=inttostr(random(10)+1);
{随机产生一个 1~10 之间的数字,并赋给 labeli[i]的 caption}
total1:=total1+strtoint(labeli[i].Caption);
{玩家的总点数自身加上新的 labeli[i]的 caption 值并刷新}
if 21-total2>=5 then
{如果电脑的总点数小于或等于 16,执行以下程序}
labeli[i+7].Caption:=inttostr(random(10)+1)
{随机产生一个 1~10 之间的数字,并赋给 labeli[i+7]的 caption}
else labeli[i+7].caption:=inttostr(0);
{如果电脑的总点数已经大于 16,则 labeli[i+7]的 caption 设为 0}
total2:=total2+strtoint(labeli[i+7].caption);
{电脑的总点数自身加上新的 labeli[i+7]的 caption 值并刷新}
label15.Caption:=inttostr(total1);
{label15 的 caption 显示为玩家的当前总点数}
label16.Caption:=inttostr(total2);
{label16 的 caption 显示为电脑的当前总点数}
end
else showmessage('只能选 7 个数字');
{如果连续单击 button2 超过六次,则跳出该提示}
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
```

```
button2.Enabled:=false;
{将 button2 的 enabled 设为 false}
for i:=i to 6 do
{设定循环}
begin
if 21-total2>=5 then
{如果电脑的总点数小于或等于 16,则执行以下程序}
begin
labeli[i+8].caption:=inttostr(random(10)+1);
{随机产生一个 1~10 之间的数字,并赋给 labeli[i+8]的 caption}
total2:=total2+strtoint(labeli[i+8].caption);
{电脑的总点数自身加上新的 labeli[i+7]的 caption 值并刷新}
label16.caption:=inttostr(total2);
{label16 的 caption 显示为电脑当前的总点数}
end
else break;
{如果电脑的总点数已经大于 16,则跳出该循环}
end;
if ((total1<=21) and (total2>21))
then showmessage('你赢了')
{如果玩家总点数不大于 21 且电脑总点数大于 21,则显示玩家胜利}
else if ((total1<=21) and ((21-total1)<=(21-total2)))
then showmessage('你赢了')
{如果玩家和电脑的总点数都不大于 21,且玩家更接近 21,则显示玩家胜利}
else if ((total1<=21) and ((21-total1)>(21-total2)))
then showmessage('你输了')
{如果玩家和电脑的总点数都不大于 21,且电脑更接近 21,则显示玩家失败}
else if ((total1>21) and ((total1-21)>(total2-21)))
then showmessage('你输了')
{如果玩家总点数大于 21,且玩家总点数与 21 点的差距大于电脑与 21 点的差距,则显示玩家失败}
else showmessage('你赢了');
{其他情况显示为玩家胜利}
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
showmessage('这是古老的博弈游戏之一,双方可轮流要点数,也可选择任意时刻停止.都停止后计算各自总点数,超过 21 点者为败,都没超过或都超过 21 点时总点数更靠近 21 点者胜.本游戏一共可要 7 次数字,每次都是在 1~10 中随机产生数字,右边的数字显示为各自的累积总点数,总点数一样时,判玩家胜.');
{单击该按钮,弹出游戏说明,游戏说明内容如上}
end;
end.
```

实例 49　机智加分人机交互

实例说明

本例编写对抗智力机智加分小游戏。如图 49-1 所示。

单击“开始游戏”按钮之后，在程序的主窗口出现一个随机产生的 10×10 的棋盘。游戏双方分别为“甲方”和“乙方”，首先由甲方选择分数；甲方选择完毕后，由乙方选择甲方选的数字所在列中的一个数字；乙方选择完毕后，甲方再选择乙方选的数字所在行的一个数字……以此循环，直到选完所有数字为止。在窗口的右侧显示了双方的加分情况，最后分数多者为赢。

在本实例中主要实现程序的窗体设计、变量定义以及 RefreshBg 过程的实现。

图 49-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。首先向窗体中添加了所使用的控件，然后定义了一个 RefreshBg 函数。该函数实现了如下几个功能：首先按照开始游戏时产生的随机数，给棋盘赋值；然后由哪个玩家选择，当其选择完毕后，把相应行或列以“高亮”显示，并变为可以选择状态；最后累计双方的分数。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Panel 控件，在该控件中添加三个 Panel 控件和两个 Button 控件。最后窗体的布局如图 49-2 所示。

2．各控件之间的关系如图 49-3 所示。

图　49-2

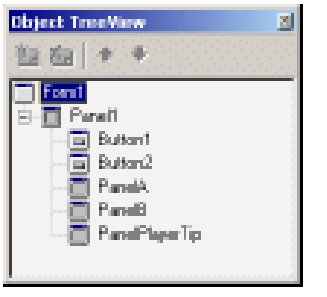


图　49-3

3．设置窗体关键属性如下：

Caption = 'Form1'
ClientHeight = 401
ClientWidth = 573
Color = clGreen

4．设置 Panel1 控件的关键属性如下：

Align = alRight
Color = clGreen
TabOrder = 0

其他控件的属性设置比较简单，这里不再介绍。

5．本程序定义了多个公共变量，各变量的含义如下：

```
Node: array [0..9,0..9] of integer;          // 0 表示没有,否则表示该节点的数值
bPlayer, bGameStart,bFlag: boolean;          // bPlayer 表示用户,bGameStart 表示游戏状态,bFlag 第一次
                                                单击标记
nLine: integer;                 // nLine 表示需要加亮的行或者列
nPlayerA,nPlayerB: integer;     // 分别表示玩家的分数,A 只能在某一列上选,B 只能在某一行上选
bmpBg: array [0..6] of TBitmap;              // 分别表示总背景(0),普通空白(1),普通加亮(2),普通红(3),
                                                加亮红(4),普通蓝(5),加亮蓝(6)
```

6．定义一个函数和一个过程，如下：

```
procedure RefreshBg;                 // 刷新显示的函数
function JudgeEnd:boolean;           // 判断游戏是否结束
```

7．本实例的核心部分就是 RefreshBg 函数，该函数实现了如下几个功能：首先按照开始游戏时产生的随机数，给棋盘赋值，然后由哪个玩家选择，当其选择完毕后，把相应行或列以“高亮”显示，并变为可以选择状态，最后累计双方的分数。该函数的代码如下：

```
procedure TForm1.RefreshBg;
var cx,cy: integer;
begin
  for cy := 0 to 9 do
    for cx := 0 to 9 do
      if Node[cx,cy] = 0 then                  // 如果该点数字是 0,则表示空
          bmpBg[0].Canvas.CopyRect(RECT(30*cx,30*cy,30*cx+30,30*cy+30),bmpBg[1].Canvas,REC
          T(0,0,30,30))
      else if Node[cx,cy] > 0 then             // 如果大于 0,则拷贝蓝色空白图案
          bmpBg[0].Canvas.CopyRect(RECT(30*cx,30*cy,30*cx+30,30*cy+30),bmpBg[5].Canvas,R
          ECT(0,0,30,30))
      else if Node[cx,cy] < 0 then             // 拷贝红色空白图案
          bmpBg[0].Canvas.CopyRect(RECT(30*cx,30*cy,30*cx+30,30*cy+30),bmpBg[3].Canvas,R
          ECT(0,0,30,30));
  if not bPlayer then                          // A 玩家
  begin
    if nLine <> -1 then                        // 游戏已经开始
      for cx := 0 to 9 do
```

```
            if Node[cx,nLine] = 0 then              // 把 A 玩家对应的行以“高亮”显示,具体过程和上面一
                                                    样,根据数字不同，拷贝图像也不同
                bmpBg[0].Canvas.CopyRect(RECT(30*cx,30*nLine,30*cx+30,30*nLine+30),bmpBg[2].Ca
                nvas,RECT(0,0,30,30))
            else if Node[cx,nLine] > 0 then
                bmpBg[0].Canvas.CopyRect(RECT(30*cx,30*nLine,30*cx+30,30*nLine+30),bmpBg[6].Ca
                nvas,RECT(0,0,30,30))
            else if Node[cx,nLine] < 0 then
                bmpBg[0].Canvas.CopyRect(RECT(30*cx,30*nLine,30*cx+30,30*nLine+30),bmpBg[4].Ca
                nvas,RECT(0,0,30,30));
        end else// B 玩家
          if nLine <> -1 then
            for cx := 0 to 9 do
              if Node[nLine,cx] = 0 then            // 同上,把相应的列以“高亮”显示
                  bmpBg[0].Canvas.CopyRect(RECT(30*nLine,30*cx,30*nLine+30,30*cx+30),bmpBg[2].
                  Canvas,RECT(0,0,30,30))
              else if Node[nLine,cx] > 0 then
                  bmpBg[0].Canvas.CopyRect(RECT(30*nLine,30*cx,30*nLine+30,30*cx+30),bmpBg[6].
                  Canvas,RECT(0,0,30,30))
              else if Node[nLine,cx] < 0 then
                  bmpBg[0].Canvas.CopyRect(RECT(30*nLine,30*cx,30*nLine+30,30*cx+30),bmpBg[4].
                  Canvas,RECT(0,0,30,30));
        for cy := 0 to 9 do
          for cx := 0 to 9 do
            if Node[cx,cy] < 0 then
            begin
               bmpBg[0].Canvas.Brush.Color := TColor($FF00FF);        // 根据正负在背景相应位置上写上
相应的数字
               bmpBg[0].Canvas.TextOut(30*cx+6,30*cy+5,IntToStr(Node[cx,cy]));
            end else if Node[cx,cy] > 0 then
            begin
              bmpBg[0].Canvas.Brush.Color := clBlue;
              bmpBg[0].Canvas.TextOut(30*cx+10 ,30*cy+5,IntToStr(Node[cx,cy]));
            end;
        Canvas.Draw(60,50,bmpBg[0]);          // 把图像直接画在 Form 的画布上
        PanelA.Caption := '甲方分数:'+IntToStr(nPlayerA);      // 提示 A 和 B 的分数
        PanelB.Caption := '乙方分数:'+IntToStr(nPlayerB);
        if bPlayer then
          PanelPlayerTip.Caption := '当前玩家: 乙'              // 提示当前玩家
        else
          PanelPlayerTip.Caption := '当前玩家: 甲';
      end;
```

实例 50　机智加分组件

实例说明

续前面的实例。如图 50-1 所示。

本例主要实现程序的代码部分，在程序的 Create 事件中初始化窗口中的内容，对用户的单击进行判断，通过 bPlayer 变量标识游戏双方，并对其得分进行计算。

图 50-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。程序启动后，产生灰色背景。单击“开始”游戏则会产生 100 个在–9 到 9 之间的随机数（0 除外），分别对应棋盘中的每一个位置，使用 RefreshBg 函数刷新背景。其中，以粉红色代表小于 0 的数，蓝色代表大于 0 的数，接下来等待用户的操作，累计用户的得分，当所有的数字都被选择完之后，统计双方的分数。

创作步骤

本实例代码如下：

```
unit Unit1;
interface
uses
    ……（此处代码略，详见光盘）
  private
    Node: array [0..9,0..9] of integer;          // 0 表示没有,否则表示该节点的数值
    bPlayer, bGameStart,bFlag: boolean;          // bPlayer 表示用户,bGameStart 表示游戏状态,bFlag 第一次单击标记
    nLine: integer;                              // nLine 表示需要加亮的行或者列
    nPlayerA,nPlayerB: integer;                  // 分别表示玩家的分数,A 只能在某一列上选,B 只能在某一行上选
    bmpBg: array [0..6] of TBitmap;              // 分别表示总背景(0),普通空白(1),普通加亮(2),普通红(3),加亮红(4),普通蓝(5),加亮蓝(6)
    procedure RefreshBg;                         // 刷新显示的函数
    function JudgeEnd:boolean;                   // 判断游戏是否结束
```

```
    ……（此处代码略，详见光盘）
  bGameStart := false;        // 初始化游戏
  for cx := 1 to 6 do
  begin
    bmpBg[cx] := TBitmap.Create;
    bmpBg[cx].LoadFromFile(strCurPath+'RES\bg'+IntToStr(cx)+'.bmp');
  end;              // 读入 6 张不同状态时的图片
  bmpBg[0] := TBitmap.Create;           // 总背景产生,填充成白色
  bmpBg[0].PixelFormat := pf24bit;
  bmpBg[0].Height := 301;
  bmpBg[0].Width := 301;
  bmpBg[0].Canvas.Brush.Color := clWhite;
  bmpBg[0].Canvas.FillRect(RECT(0,0,301,301));
  bmpBg[0].Canvas.Font.Size := 12;
  bmpBg[0].Canvas.Font.Style := [fsBold];
  bmpBg[0].Canvas.Font.Color := clLime;       // 定义好背景的画布属性,在上面写字的时候用

  for cy := 0 to 9 do
    for cx := 0 to 9 do                       // 初始化棋盘
    bmpBg[0].Canvas.CopyRect(RECT(30*cx,30*cy,30*cx+30,30*cy+30),bmpBg[1].Canvas,RECT(0,0,3
    0,30));
end;

procedure TForm1.FormDestroy(Sender: TObject);          // 释放系统变量
var cx: integer;
begin
  for cx := 0 to 6 do
    bmpBg[cx].Free;
end;

procedure TForm1.Button2Click(Sender: TObject);         // 关闭程序
begin
  Close;
end;

procedure TForm1.Button1Click(Sender: TObject);         // 开始游戏
var cx,cy:integer;
begin
  randomize;
```

```
  bPlayer := false;              // 初始化玩家为 A
  bGameStart := true;
  bFlag := false;
  nLine := -1;
  nPlayerA := 0;
  nPlayerB := 0;
  for cy := 0 to 9 do
    for cx := 0 to 9 do
      repeat
        Node[cx,cy] := (random(19000) mod 19) - 9;
      until   Node[cx,cy] <> 0;          // 产生棋盘上的数据,如果产生了 0,则重新产生
  RefreshBg;             // 刷新背景
end;

procedure TForm1.FormPaint(Sender: TObject);              // 重绘窗口
begin
  Canvas.Draw(60,50,bmpBg[0]);
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
                                        Integer);
var nX,nY: integer;
begin
  if not bGameStart then Exit;      // 如果游戏还没有开始,则退出
  nX := (X-60) div 30;
  nY := (Y-50) div 30;
  if Node[nX,nY] = 0 then Exit;    // 如果该节点是空点,则退出
  if not bPlayer then              // 如果是 A 玩家
  begin                            // 由于是 A 先行,所以要在 A 设置一个 bFlag 标记,否则第一次不知道
                                   nLine 的值,无法进行比较
  if bFlag and (nY <> nLine) then Exit;        // 这里是判断单击是否合法,也就是是否在高亮的行或者
                                                列上
    nPlayerA := nPlayerA + Node[nX,nY];
    nLine := nX;                // 记录下当前的行数
    bFlag := true;
  end
  else
  begin
    if nX <> nLine then Exit;              // 判断以及后续处理同上
```

```
      nPlayerB := nPlayerB + Node[nX,nY];
      nLine := nY;
    end;
    Node[nX,nY] := 0;          // 把当前位置清空
    bPlayer := not bPlayer;    // 玩家交换
    RefreshBg;
    if JudgeEnd then           // 判断是否结束,如果结束的话，则给予相应的提示
    begin
      Application.MessageBox(PChar('游戏结束'#13'甲乙比分为:'+IntToStr(nPlayerA)+':'+IntTo
      Str(nPlayerB)+'!'),'游戏结束提示',MB_OK);
      bGameStart := false;
    end;
end;

function TForm1.JudgeEnd: boolean;
var cx,cy: integer;
begin
    if not bPlayer then        // A 玩家,由于 A 只能在 nLine 规定的行里行动,所以判断该行还有没有未
                               清空的数字
    begin
      for cx := 0 to 9 do
        if Node[cx,nLine] <> 0 then
        begin
          Result := false;     // 如果找到一个数,则游戏还可以继续下去,否则判定游戏结束
          Exit;
        end;
    end else                   // B 玩家
      for cy := 0 to 9 do
        if Node[nLine,cy] <> 0 then          // 同上
        begin
          Result := false;
          Exit;
        end;
    Result := true;
end;
end.
```

实例 51　黑白棋人机交互

实例说明

本例编写双人对抗游戏黑白棋，如图 51-1 所示。

游戏双方分别执黑棋和白棋，单击“开始新游戏”按钮后，由白棋先下，具体规则是：只有在黑棋所在的行或列的前面有白棋时，才可在黑棋的后面放置白棋。放置以后，两个白棋之间的黑棋全部变为白棋，接下来由黑棋下，规则同白棋。最后子数多者为胜。

本实例主要实现程序的界面设计以及如何响应用户的 MouseDown 事件。

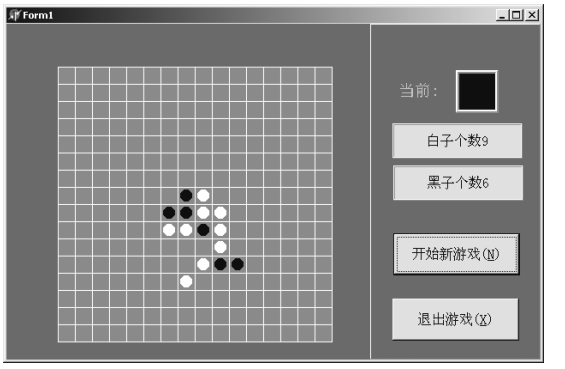


图 51-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。这个游戏的关键在于如何判断玩家单击的位置是否有效。比如当前该白棋出，如果所单击的位置的旁边，比如左边一格也是白子，则向左寻找位置会遭失败。只有单击的位置左边一格是黑子(空白也不行)，才可以继续向左寻找。如果是黑子，则继续向左，直到找到一个白子或者到边界为止；如果找到了一个白子，则结束寻找，并从该位置开始返回到单击的点，同时把所经过的所有位置均置成白子。同样对右、上、下、左上、右上、左下、右下这七个方向进行同样的寻找。如果所有的方向都没有找到白子，则该单击无效，黑子同理。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Panel 控件，在该控件中添加一个 Label 控件、三个 Panel 控件和两个 Button 控件。如图 51-2 所示。

图　51-2

2．设置第一个 Panel 控件的关键属性如下：

Align = alRight

Color = clGreen

其他控件的属性设置比较简单，这里不再介绍。

3．程序的 MouseMove 事件处理代码如下，它主要实现了对玩家单击窗口的响应。

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;Shift: TShiftState; X, Y: Integer);
var
  nX,nY,cx,cy: integer;
  bFind,bChange: boolean;
  ptFind: TPoint;
  strTip:string;          // ptFind 记录找到的同色棋子的棋盘坐标
begin       // nX,nY 为鼠标对应的棋盘坐标,cx,cy 分别是 x,y 方向的计数器,bFind 表示某方向上是否找到同色棋子,bChange 表示是否改变棋子颜色
  nX := (X-60) div 20;       // 把窗口坐标转化成为棋盘坐标
  nY := (Y-50) div 20;
  if bGameStart then
  begin
    bChange := false;        // 初始化为还没有找到
    bFind := false;          // 开始从鼠标落点向左找
    for cx := nX-1 downto 1 do
      if ((not bCurPlayer) and (Node[cx,nY] <> 1)) or( (bCurPlayer) and (Node[cx,nY] <> 0)) then
        break  // 如果当前格的左边一格中棋子的颜色,与当前下棋方的棋子颜色一样,则向左寻找同色点失败
      else if ((not bCurPlayer) and (Node[cx-1,nY] = 0)) or( (bCurPlayer) and (Node[cx-1,nY] = 1)) then
      begin      // 如果当前格的左边两格中棋子的颜色,与当前下棋方的棋子颜色一样,则向左寻找同色棋子得以成功
        bFind := true;
        bChange := true;
        ptFind := POINT(cx-1,nY);          // 记录下当前找到的同色棋子的位置,为刷新棋盘数据用
        break;
      end;
    if bFind then                          // 如果向左找到了同色棋子
      for cx := ptFind.X+1 to nX do        // 从同色的棋子开始刷新,一直刷新到鼠标单击的位置(和查找正好反向)
        if not bCurPlayer then Node[cx,nY] := 0       // 如果是白色,则置所经过的棋格为 0,否则置 1
        else Node[cx,nY] := 1;
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    bFind := false;          // 开始向右查找,算法同向左查找
```

……（此处代码略，详见光盘）

```
    nBlack := 0;
    nWhite := 0;
    for cx := 0 to 15 do                    // 刷新棋格的显示
      for cy := 0 to 15 do
      if Node[cx,cy] = 0 then
      begin
        nWhite := nWhite + 1;
          bmpBg.Canvas.CopyRect(RECT(20*cx,20*cy,20*cx+20,20*cy+20),bmpWhite.Canvas,RECT(
          0,0,20,20));
      end
      else if Node[cx,cy] = 1 then
      begin
        nBlack := nBlack + 1;
        bmpBg.Canvas.CopyRect(RECT(20*cx,20*cy,20*cx+20,20*cy+20),bmpBlack.Canvas,RECT(0,0
        ,20,20));
      end;
    if bChange then              // 如果所有的方向都没有查找到,则黑白方不变,否则轮换
      bCurPlayer := not bCurPlayer;
    if bCurPlayer then
      Panel2.Color := clBlack
    else
      Panel2.Color := clWhite;
    Canvas.Draw(60,50,bmpBg);            // 显示图像数据以及更新黑白子数量的数据
    Panel3.Caption := '白子个数'+IntToStr(nWhite);
    Panel4.Caption := '黑子个数'+IntToStr(nBlack);
    if (nBlack + nWhite = 256) then              // 总个数为 16×16,游戏结束
    begin
      bGameStart := false;
      if nBlack > nWhite then strTip := '黑子以'+IntToStr(nBlack)+':'+IntToStr(nWhite)+'胜白子!'
      else if nBlack < nWhite then strTip := '白子以'+IntToStr(nWhite)+':'+IntToStr(nBlack)+'胜黑子!'
      else strTip := '黑子和白子不分胜负,战成平局!';
      Application.MessageBox(PChar('游戏结束'+#13+strTip),'提示',MB_OK);
    end;
  end;
end;
```

实例 52　黑白棋组件

实例说明

续前面的实例，如图 52-1 所示。

本例主要实现程序的其他代码。包括在程序的初始化阶段对棋盘进行初始化；响应 Paint 事件对窗体进行重绘；设置“开始新游戏”和“退出游戏”按钮的 OnClick 事件代码。

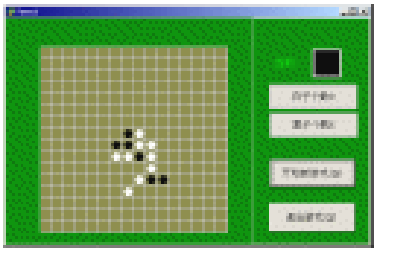


图 52-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。首先创建一个棋盘 16×16，然后在“开始新游戏”按钮的 OnClick 事件中生成中心的 4 个棋子（两个白色两个黑色）。接下来在窗体的 OnMouseDown 事件中，响应用户的按键操作，判断用户单击的地方是否可以放置棋子，如果可以，就按照游戏规则，改变棋盘中的内容。同时，在棋盘的右侧记录双方的棋子个数。

创作步骤

程序设计步骤续前面的实例。本实例的其他代码如下：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Math;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Panel3: TPanel;
    Panel4: TPanel;
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button2Click(Sender: TObject);
```

```
        procedure Button1Click(Sender: TObject);
        procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,Y: Integer);
        procedure FormMouseDown(Sender: TObject; Button: TMouseButton;Shift: TShiftState; X, Y:
Integer);
    private
        Node: array [0..15,0..15] of integer;        // -1 表示无子,0 表示白,1 表示黑
        bmpBg,bmpBlock,bmpBlack,bmpWhite: TBitmap;          // 背景、小块、黑子、白子图
        strCurPath: string;        // 当前文件路径
        bGameStart,bCurPlayer: boolean;          // bGameStart,0 结束,1 开始;bCurPlayer,0 表示白,1 表示黑
        nBlack,nWhite:integer;          // 分别记录白子和黑子个数
    public
    end;
var
    Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
var cx,cy: integer;
begin
    strCurPath := ExtractFilePath(Application.ExeName);
    bmpBg := TBitmap.Create;
    bmpBg.PixelFormat := pf24bit;
    bmpBg.Height := 321;
    bmpBg.Width := 321;
    bmpBg.Canvas.Brush.Color := clWhite;
    bmpBg.Canvas.FillRect(RECT(0,0,321,321));   // 初始化背景图并填充背景为白色(为了使得右边和
                                                  下边也有白边)
    bmpBlock := TBitmap.Create;
    bmpBlock.LoadFromFile(strCurPath+'RES\bg.bmp');
    bmpBlack := TBitmap.Create;
    bmpBlack.LoadFromFile(strCurPath+'RES\black.bmp');
    bmpWhite := TBitmap.Create;
    bmpWhite.LoadFromFile(strCurPath+'RES\white.bmp');        // 读入相应的位图文件
    for cy := 0 to 15 do
        for cx := 0 to 15 do
        bmpBg.Canvas.CopyRect(RECT(20*cx,20*cy,20*cx+20,20*cy+20),bmpBlock.Canvas,RECT(0,0,20,2
0));
end;
procedure TForm1.FormPaint(Sender: TObject);          // OnPaint,显示背景图,在 Form 的画布上直接画
begin
    Canvas.Draw(60,50,bmpBg);
end;
```

```
procedure TForm1.FormDestroy(Sender: TObject);          // 释放资源
begin
  bmpBg.Free;bmpBlock.Free;bmpBlack.Free;bmpWhite.Free;        // 释放资源
end;
procedure TForm1.Button2Click(Sender: TObject);          // 退出程序
begin
  Close;
end;
procedure TForm1.Button1Click(Sender: TObject);          // 开始游戏
var cx,cy: integer;
begin
  bCurPlayer := false;          // 白子先行
  bGameStart := true;
  for cy := 0 to 15 do
    for cx := 0 to 15 do
    begin
    bmpBg.Canvas.CopyRect(RECT(20*cx,20*cy,20*cx+20,20*cy+20),bmpBlock.Canvas,RECT(0,0,20,2
0));
    Node[cx,cy] := -1;          // 初始化所有节点为-1,表示无子
    end;
  bmpBg.Canvas.CopyRect(RECT(140,140,160,160),bmpBlack.Canvas,RECT(0,0,20,20));
  bmpBg.Canvas.CopyRect(RECT(160,160,180,180),bmpBlack.Canvas,RECT(0,0,20,20));
  bmpBg.Canvas.CopyRect(RECT(160,140,180,160),bmpWhite.Canvas,RECT(0,0,20,20));
  bmpBg.Canvas.CopyRect(RECT(140,160,160,180),bmpWhite.Canvas,RECT(0,0,20,20));
  Node[7,7] := 1;
  Node[8,8] := 1;
  Node[7,8] := 0;
  Node[8,7] := 0;        // 初始化最中间的 4 个位置
  Canvas.Draw(60,50,bmpBg);
  Panel3.Caption := '白子个数:2';
  Panel4.Caption := '黑子个数:2';
end;
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,Y: Integer);
begin
  if bGameStart and (X < 60) or (X > 411) or ( Y < 50) or (Y > 401) or (Node[(X-60) div 20,(Y-50) div 20]
  <>1 ) then
    Cursor := crDefault              // 如果鼠标在棋盘内,则 cursor 为 hand,否则为 arrow
  else
    Cursor := crHandPoint;
end;end;
```

实例 53　三子棋人机交互

实例说明

本例编写三子棋游戏，如图 53-1 所示。

游戏规则如下：由人和电脑轮流在棋盘上抢位置，轮到人抢时，人单击棋盘上的按钮，并在按钮上出现“ X ”记号；而电脑抢到的位置用“√”表示。哪一方首先在横排、竖排或对角线上连成一条线就算赢。这个程序没有使用高深的编程知识，但是提供了一个很好的程序设计的思路。

在本实例中主要介绍程序的界面设计以及本实例所定义的公共变量和过程。

图 53-1　效果图

编程思路

本实例使用 C++ Builder 6.0 来实现，在程序中利用九个按钮作为三子棋的棋盘布置，然后响应用户的单击，在按钮上绘制“√”或者“×”。

创作步骤

1．启动 C++ Builder 6.0，打开一个新的标准工程。如果 C++ Builder 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。在新建的界面上加入九个 BitBtn 控件和一个 MainMenu 控件。如图 53-2 所示。

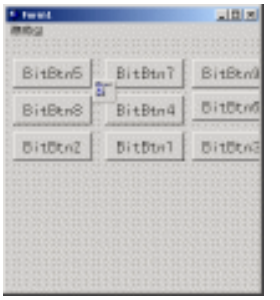


图　53-2

这里，BitBtn 控件的属性很简单，不再介绍。

2．双击 MainMenu 控件，向其中添加一个菜单项，如图 53-3 所示。

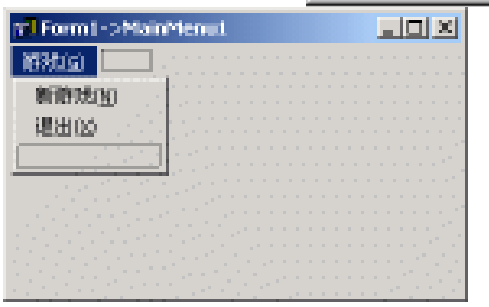


图　53-3

3．**这里使用的公共函数和变量如下：**

```
int Flag;                          // 设置标记
TBitBtn *Button[3][3];             // 存储各按钮的状态
int win[3][3];
int pin_who_win();                 // 判断是否已经连成一线
void pppp();                       // 显示胜负的对话框
int computer_put();                // 判断计算机应该把棋子放到哪里
void putxy(int x,int y,int who);   // 放置棋子
```

实例 54　三子棋组件

实例说明

续前面的实例，如图 54-1 所示。

本例主要实现程序的代码部分。在程序的 OnCreate 事件中，初始化各个按钮的大小和位置，当开始游戏之后，玩家单击一个按钮，接下来调用程序的 computer_put()函数，利用这个函数来计算电脑要下的棋子的位置。

当棋盘中的位置全部被下满之后，再调用 pin_who_win()函数判断哪一方获胜。

图 54-1　效果图

编程思路

本实例分两个过程，人抢位置和电脑抢位置。程序首先给窗体的九个按钮编号，用数组确定每一个按钮；然后编写一个 pin_who_win()函数，该函数用来判断哪方取得了胜利；再编写 computer_put()函数，该函数是程序的精华部分，用来决定轮到电脑走时，电脑应该抢占哪个位置；接着编写 pppp()函数，该函数用来显示交战的结果。

接下来在各按钮的 OnClick 事件中，调用以上几个函数就可以实现游戏目标了。

创作步骤

程序设计步骤续前面的实例。

单元文件代码如下（只列出部分关键代码，其余部分详见配套光盘）：

……

```
// ---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
        Button[0][0]=BitBtn1;
        Button[0][1]=BitBtn2;
        Button[0][2]=BitBtn3;
        Button[1][0]=BitBtn4;
        Button[1][1]=BitBtn5;
        Button[1][2]=BitBtn6;
        Button[2][0]=BitBtn7;
        Button[2][1]=BitBtn8;
```

```
        Button[2][2]=BitBtn9;
    ……（此处代码略，详见光盘）

        BitBtn9->Left=196;
        BitBtn9->Top=196;
        BitBtn9->Width=92;
        BitBtn9->Height=92;
        BitBtn9->Caption="";
        BitBtn9->Cursor=crHandPoint;
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::MenuExitClick(TObject *Sender)
{// 退出程序
        Close();
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
        if(win[0][0]==0)
        {
            BitBtn1->Glyph->LoadFromFile("tt2.bmp");
            win[0][0]=1;
            computer_put();
            pppp();
            Flag=2;
        }
}
    ……（此处代码略，详见光盘）
  else for(l=0;l<3;l++)
              for(o=0;o<3;o++)
                  if (win[l][o]==0)
                    {
                        putxy(l,o,COMPUTER);
                        win[l][o]=2;
                        return win[l][o];
                    }
          return 3;
}
// ---------------------------------------------------------------------------
int TForm1::pin_who_win()
{// 判断谁胜利
        int i;
```

```
for(i=0;i<3;i++)
        if (win[i][0]==win[i][1]&&win[i][1]==win[i][2]&&win[i][0]!=0) return win[i][0];
for(i=0;i<3;i++)
        if (win[0][i]==win[1][i]&&win[1][i]==win[2][i]&&win[0][i]!=0) return win[0][i];
if (win[0][0]==win[1][1]&&win[1][1]==win[2][2]&&win[0][0]!=0) return win[0][0];
else if (win[0][2]==win[1][1]&&win[1][1]==win[2][0]&&win[0][2]!=0) return win[0][2];
else return 0;

}
// ---------------------------------------------------------------------------
void TForm1::putxy(int x,int y,int who)
{// 向按钮中添加图片
      Button[x][y]->Glyph->LoadFromFile("tt1.bmp");
}
// ---------------------------------------------------------------------------
void TForm1::pppp()
{ // 通过对话框显示是人胜利还是电脑胜利
      Form2=new TForm2(Application);
      if(pin_who_win()==1)
                {
                     Form2->Label1->Caption="人胜利了";
                     Form2->ShowModal();
                }
                else if(pin_who_win()==2)
                {
                     Form2->Label1->Caption="电脑胜利了";
                     Form2->ShowModal();
                }
      delete Form2;
}
void __fastcall TForm1::MenuNewGameClick(TObject *Sender)
{// 开始新游戏
      Flag=0;
      for(int i=0;i<3;i++)
           for(int j=0;j<3;j++)
           {
                Button[i][j]->Glyph=NULL;
                win[i][j]=0;
           }
}
// ---------------------------------------------------------------------------
```

实例 55　五 子 连 线

实例说明

本例编写五子连线小游戏，如图 55-1 所示。

甲、乙两人轮流下棋，在窗口中单击鼠标，任何一方能在横向、纵向或对角线方向使五个棋子连成一线就算胜利，即出现“X 方胜利”对话框。单击 OK 按钮，对话框消失并清空窗口中的棋子，重新开始新局。

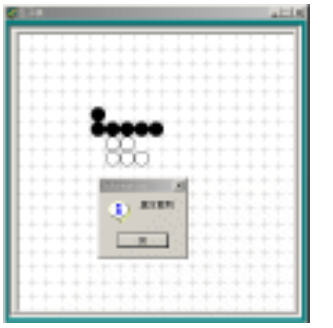


图 55-1　效果图

编程思路

本程序要处理好几个响应事件：第一步是 OnCreate 事件，要初始化窗口，即清空窗口，这需要一个循环语句将 DrawGrid 控件的各单元置为 0；第二步是编写一个公共函数，判断哪一方能在横向、纵向或对角线方向使五个棋子连成一线；第三步是响应用户的 OnMouseDown 事件，在该事件中要实现轮流改变绘制棋子的颜色，同时调用 Inwin 函数，判断是否已连成了五子。如果是，提示 X 方胜利，清空窗口。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。在新建的界面上加入一个 Panel 控件和一个 DrawGrid 控件。

2．设置 DrawGrid 控件的关键属性如下：

Left = 16
Top = 16
Width = 402
Height = 402
ColCount = 19
DefaultColWidth = 20
DefaultRowHeight = 20
DefaultDrawing = False
FixedCols = 0
RowCount = 19
FixedRows = 0
ScrollBars = ssNone
TabOrder = 0

本程序源代码详见配套光盘。

实例 56　智能黑白棋人机交互

实例说明

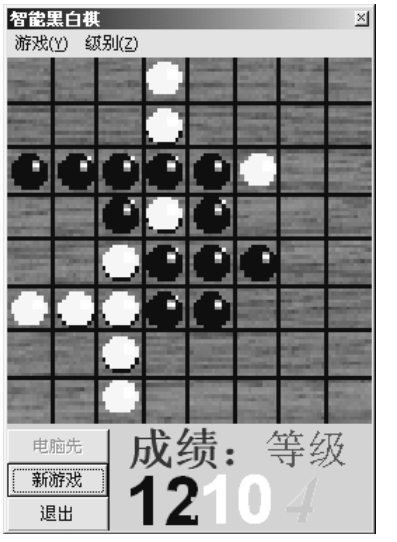


图 56-1　效果图

本例编写智能黑白棋游戏，游戏画面如图 56-1 所示。

本例游戏与前面游戏不同的是，这个游戏具有智能，也就是说，玩家可以和电脑进行对弈。电脑可以选择合理的位置下棋，以达到棋盘上子数最多。这种人工智能，在大多数情况下可以看作是某种形式的搜索、比较与判断。对于非计算机专业的程序员来说，制作一个游戏，人工智能部分可能是最困难的。在这里将以一个简单的例子说明如何实现人工智能。

在本实例中主要实现程序的窗体设计。

编程思路

在本实例中主要制作程序的窗体部分。本实例只使用了 Delphi 6.0 的标准控件，包括 MainMenu 控件、Button 控件以及 Image 控件，它们构成了程序界面的基本部分。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。在新建的界面上加入一个 MainMenu 控件、一个 Timer 控件、三个 Button 控件（分别为“电脑先”、“新游戏”和“退出”按钮）、三个 Image 控件（这三个控件分别用来保存棋盘、黑子和白子）和七个 Label 控件，这些 Label 控件用来显示成绩以及等级。

2．窗体的关键属性如下：

Left = 393
Top = 207
BorderStyle = bsToolWindow
Caption = 智能黑白棋'
ClientHeight = 336
ClientWidth = 260
Color = clSilver

单击窗体的 Icon **属性，将会弹出选择文件图标属性对话框，如图** 56-2 **所示。单击其中的“** Load **”按钮，将会弹出选择图标对话框，在其中选择一个图标文件。**

3．**设置程序菜单如图** 56-3 **所示。**

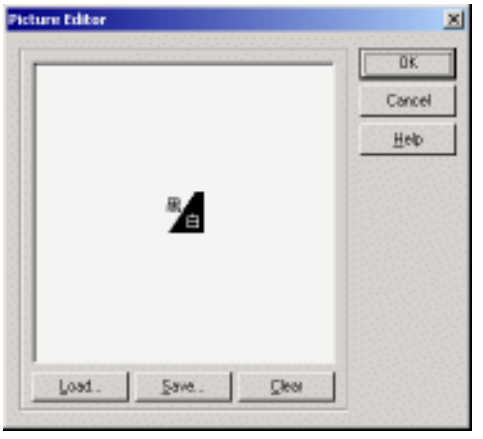


图　56-2

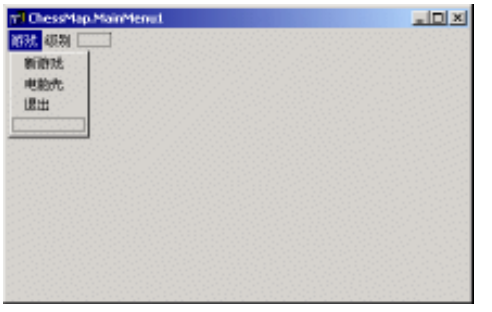


图　56-3

4．**设置** Timer **控件的** Interval **属性为** 2000。

5．**在程序的开始部分声明变量和常量如下：**

```
const
        count: byte=0;
        human=-1;
        computer=1;
        emp =0;
        d : array [1..8,1..2] of shortint
                = ((1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1),(1,-1));
var
    tem,n,m,xx,yy,ix,iy,max_x,max_y,e,f,p,q,k,skip: integer;
    map: array [1..8,1..8] of shortint;
    ChessMap: TChessMap;
    ss1:byte;
    ss2:byte;
    newround:boolean=true;
    hardness:byte =4;
```

6．**在棋盘上除了那些已经有子的地方不能走子外，那些不能吃子的点也不能走。这里定义一个** Cango **函数用来判断能不能走子，其代码如下：**

```
function cango(who:shortint):boolean;
begin
        cango:=false;
        for p:=1 to 8 do
              for q:=1 to 8 do
```

```
                if (point(p,q,who)>0)
                    then    begin
                                cango:=true;
                                exit;
                            end;
end;
```

7．定义 Drawmap **函数用来在棋盘上绘制棋子，其代码如下：**

```
procedure drawmap;
begin
    ss1:=0; ss2:=0;
    for m:=1 to 8 do
        for n:=1 to 8 do
            case map[n,m] of
             human: begin
             ChessMap.table.canvas.draw(round((n-1)*12.5+1),round((m-1)*12.5+1),chessmap.image5.
             Picture.Graphic);
             inc(ss1);
                    end;
            computer: begin
             ChessMap.table.canvas.draw(round((n-1)*12.5+1),round((m-1)*12.5+1),chessmap.image6.
             Picture.Graphic);
            inc(ss2);
                      end;
            end;
    ChessMap.label2.caption:=inttostr(ss1);
    ChessMap.label3.caption:=inttostr(ss2);
end;
```

实例 57　智能黑白棋组件

实例说明

续前面的实例。如图 57-1 所示。

本例主要实现电脑的智能部分。通过计算判断往那里下棋子最有利，同时还定义 Point 函数，用来判断相同颜色棋子之间有多少个另外一种颜色的棋子；通过 eat 函数来控制吃棋子。

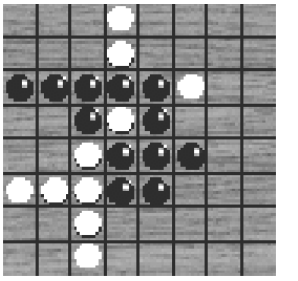


图 57-1　效果图

编程思路

这个程序关键是对算法的处理，也就是说如何使电脑“思考”。具体方法是计算每一个可以放置的点的“有利值”，选出“有利值”最大的点，然后走棋子。但是要考虑到，按照这种方式，则忽略了由此而引出的下一层问题。假设这样走了，会不会给对手（即玩家）摆出一步好棋呢？若让电脑走这一步棋的时候，应该让它考虑到下一步，这是关键。在下面的程序代码中有详细说明。

创作步骤

程序设计步骤续前面的实例。

1．定义 Point 函数，用来判断相同颜色棋子之间有多少个另外一种颜色的棋子，其代码如下：

```
function point(a,b:byte;who:shortint):integer;
// 判断相同颜色棋子之间有多少个另外一种颜色的棋子
var
    xx,yy,step: byte;
begin
      n:=0;
      if map[a,b]=emp then
      for m:=1 to 8 do
      begin
          xx:=a; yy:=b; step:=0;
          repeat
                inc(step);
                xx:=xx+d[m,1];
                yy:=yy+d[m,2];
```

```
            until (map[xx,yy]<>who*(-1)) or not((xx+d[m,1]) in [1..8])
                                          or not((yy+d[m,2]) in [1..8]);
            if (map[xx,yy]=who) and (step>1) then inc(n,(step-1));
        end;
        point:=n;
end;
```

2．下面这段代码用来实现电脑的“思考”，其基本思路在前面已经介绍过了，这里只列出实现代码。

```
function think(l:byte;who:shortint): integer;
var
    max,i,j,value,ss,maxx,maxy:integer;
    rec:array [0..50,1..2] of byte;
procedure trick;
begin
      if (i in [1,8]) and (j in [1,8]) then inc(value,5);
      if (i in [3,6]) and (j in [3,6]) then inc(value,2);
      if (i=1) and ((j=2) or (j=7)) then dec(value,2);
      if (i=8) and ((j=2) or (j=7)) then dec(value,2);
      if (j=1) and ((i=2) or (i=7)) then dec(value,2);
      if (j=8) and ((i=2) or (i=7)) then dec(value,2);
    ……（此处代码略，详见光盘）
        end else begin
      for i:=1 to 8 do
          for j:=1 to 8 do
              begin
                  value:=point(j,i,who);
                  if value>0 then begin
                {    nextmap;}
                trick;
                ss:=0;    map[j,i]:=who; rec[ss,1]:=j; rec[ss,2]:=i;
                for p:=1 to 8 do
                begin
                 xx:=j;yy:=i;q:=0;
                 repeat
                          inc(q);
                          inc(xx,d[p,1]);
                          inc(yy,d[p,2]);
                 until not(map[xx,yy]=who*(-1)) or not((yy+d[p,2]) in [1..8])
                                              or not((xx+d[p,1]) in [1..8]);
                 if (map[xx,yy]=who) and (q>1) then
                 begin
                     xx:=j; yy:=i;
```

```
                for k:=1 to q-1 do
                  begin
                        inc(xx,d[p,1]);
                        inc(yy,d[p,2]);
                        map[xx,yy]:=who;
                        inc(ss); rec[ss,1]:=xx; rec[ss,2]:=yy;
                  end;
              end;
            end; {for}
              who:=who*(-1);
              value:=value-think(l+1,who);
              who:=who*(-1);
             { oldmap;}
             map[rec[0,1],rec[0,2]]:=emp;
             for p:=1 to ss do map[rec[p,1],rec[p,2]]:=who*(-1);
              if value>max then
                 begin
                       max:=value;
                       maxx:=j; maxy:=i;
                 end;
            end;{if}
          end;{for}    end;{if}
    if max=-100 then max:=0;
    think:=max; max_x:=maxx; max_y:=maxy;
end;
```

要想知道“走过这一步之后对手可能获取的最大‘有利值’”，就必须在 think **内部再次调用** think **本身。即假设对手（即玩家）也按照和我们一样的思考模式进行思考，即认为他的** think **也等于“在该点所吃子数，走过这一步之后对手可能获取的最大“有利值””。在这样假设后，会发现为了求“走过这一步之后对手可能获取的最大‘有利值’”，**think **被第三次调用了。事实上，如果不加以控制，这样的自我调用会无止境地进行下去。因此必须设一个嵌套调用的最大层数（深度），也就是让电脑思考的“步数”。当达到这个“最大层数”时，就可以近似地认为最大“有利值”点就是“可以吃棋子数”最大的点。**

3．**下面这个** eat **函数用来控制吃棋子，其代码如下：**

```
procedure eat(a,b:byte;who:shortint);
begin
          map[a,b]:=who;
            for p:=1 to 8 do
            begin
              xx:=a;yy:=b;q:=0;
              repeat
                    inc(q);
                    inc(xx,d[p,1]);
```

```
                inc(yy,d[p,2]);
            until not(map[xx,yy]=who*(-1)) or not((yy+d[p,2]) in [1..8])
                                    or not((xx+d[p,1]) in [1..8]);
            if (map[xx,yy]=who) and (q>1) then
            begin
                xx:=a; yy:=b;
                for k:=1 to q-1 do
                 begin
                        inc(xx,d[p,1]);
                        inc(yy,d[p,2]);
                        map[xx,yy]:=who;
                 end;
            end;
        end;
    drawmap;
end;
```

实例 58　小 球 射 门

实例说明

图 58-1　效果图

本例制作小球射门小游戏，效果如图 58-1 所示。

游戏中，电脑为一方，游戏者为另一方，用挡板弹射小球射对方的球门。蓝色挡板为电脑一方，红色挡板为游戏者一方。

用鼠标上下移动来控制挡板，力图将小球射入对方的球门。可以看出双方各有一个挡板是守门用的。要注意，己方的两个挡板是同时上下移动的。

编程思路

本例通过标准的 Java AWT 来实现。整个程序有四类：Main 类是主类；Ball 类代表小球，Computer 类代表游戏的电脑一方，Player 类代表游戏者一方。本例首先介绍 Main 类。Main 类负责整个程序的总体调度和运行，包括分数获得一方的计算、何时游戏决出胜负等。

创作步骤

1．Main 类的主要属性如下：

```
public class Main extends Applet implements Runnable
{public int score1;                    // 游戏者的得分记录
 public int score2;                    // 电脑一方的得分记录
 Ball puck;                            // 小球
// 一共四条挡板，两条属于电脑，两条属于游戏者
 Player bat1;
 Player bat2;                          // 游戏者一方的两条挡板
 Computer bat3;
 Computer bat4;                        // 电脑一方的两条挡板

 public int whosgoal;                  // 表示得分一方
 private Image dbImage;
 private Graphics dbg;
```

```
        // 声音效果，分别在射门命中和被挡板弹射时候播放
        AudioClip kicknoise;          // l
        AudioClip goalnoise;
       // 记录游戏状态的布尔值
        boolean play;
        boolean end;
        boolean title;
```

2．Main 类的主要方法：

```
public void init ()
{
// 游戏的双方结果记录清零
        score1 = 0;
        score2 = 0;
// 背景设置
        setBackground (Color.black);
// 产生一个合适大小的小球
        puck = new Ball (8, 250, 175, 0, 0, Color.black, this);
// 产生需要的四条挡板
        bat1 = new Player (15, 175);
        bat2 = new Player (375, 175);
        bat3 = new Computer (475, 175);
        bat4 = new Computer (115, 175);
// 初始化游戏状态
        play = false;
        end = false;
        title = true;
        titleImg = getImage(getCodeBase(), "Title.gif");
// 调入两种声音
        kicknoise = getAudioClip (getCodeBase(), "hit.au");
        kicknoise.play();
        kicknoise.stop();
        goalnoise = getAudioClip (getCodeBase(), "chime.au");
        goalnoise.play();
        goalnoise.stop();
}
public void run ()
{
// 移动小球
                puck.move ();
```

```
            bat3.ComputerMove (puck);
// 计算是否有得分的，将判断结果记入 whosgoal 变量，以利于下面进行判断并采取相应的动作
            whosgoal = puck.wheresBall ();
            if (whosgoal != 0)
{
// 如果有一方射门命中，则进行判断到底是哪一方射门成功
                  if (whosgoal == 1)
                  {
// 如果是游戏者射门成功，则记分增加，并采取动作，如果满了 10 分，游戏结束
                        score1 += 1;
                        goalnoise.play();
                        puck.pos_x = 250;
                        puck.vx = 4;
                        puck.vy = -4;
                        if (score1 == 10)
                        {
                              play = false;
                              end = true;
                              puck.isStoped = true;
                              puck.vx = 0;
                              puck.vy = 0;
                        }
                  }
                  else if (whosgoal == 2)
// 否则，是电脑得分，同样采取类似动作
public boolean mouseMove (Event e, int x, int y)
{
// 处理鼠标移动事件，挡板随之采取适当的运动
      maus_y = y;
      if (maus_y > 25 & maus_y < 325)
      {
            bat1.PaddelMove (maus_y);
            bat2.PaddelMove (maus_y);
      }

      return true;
}
public void paint (Graphics g)
}
```

实例 59　射 门 战 绩

实例说明

续前面的实例，本例主要讲述射门战绩的实现。

游戏过程中有适当的声音代表挡板弹出小球和射门命中。游戏界面的下部显示了对战双方的战绩，即分别射入了几次小球。先达到 10 个进球的一方获胜。

游戏中蓝色的一方（电脑）可以说是一个强劲的对手，但经过练习，游戏者可以最终战胜电脑。

图 59-1　效果图

编程思路

本例通过 Ball、Player 和 Computer 类实现。Ball 类负责小球的运行轨迹，并判断哪方得分和处理挡板之间对小球的碰撞过程。Player 和 Computer 两个类很相似，所以重点讲述 Computer 类。它们之间的不同主要是移动的方式不同，Player 类的移动是根据游戏者的鼠标移动状况来进行的，而 Computer 类是自动地进行，这个不同体现在 Player 类的 PaddelMove（）和 Computer 类的 ComputerMove（）方法上。

在讲述 Computer 类和 Player 类时，会看到二者各有一个 size_y 属性，这代表双方挡板的长度。显而易见，要让己方的球门不易被射入，己方的防守挡板越长越好。因此，读者可以通过控制这个属性的值来控制游戏的难度。默认的挡板长度是 50 个像素，如果读者将己方（Player 类）的属性值加大，则游戏难度将变低，因为电脑一方不容易射入己方的球门；也可以通过减小电脑方（Computer 类）的这个属性值来减小游戏难度。注意，只要长度设置为 200 个像素，就可以刚好盖住球门的大小，从而可以保证一分不失，百战百胜。

创作步骤

程序设计步骤续前面的实例。

1．Ball 类的主要方法：

```
public Ball (int r, int x, int y, int vx, int vy, Color c, Component parent)
{// 构造方法
 // 初始化状态值
    pos_x = x;
    pos_y = y;
```

```
  // 调入图像
        ballImg = ((Applet) parent).getImage (((Applet) parent).getCodeBase(), "Ball.gif");
}
    public int wheresBall ()
{
// 判断是哪方得分，这是根据小球和双方球门的相对位置坐标来判断的，判断结果放入返回的 int 类型值中
.....}
    public void move ()
  {
// 小球的移动方法
    if (vx > max_vx) vx = max_vx;
            else if (vx < -max_vx) vx = -max_vx;
            else if (vy > max_vy) vy = max_vy;
            else if (vy < -max_vy) vy = -max_vy;
            else if (vx == 0 && !isStoped) vx = 1;

            pos_x += vx;
            pos_y += vy;
    }
    public void PCollision (Player bat1, Player bat2, AudioClip k)
{
// 处理和游戏者挡板的碰撞情况，与之相对应的是：
    public void CCollision (Computer bat3, Computer bat4, AudioClip k)
    {
// 处理和电脑一方的挡板碰撞情况
    public void DrawBall (Graphics g)
    {
// 描画小球的方法
        g.setColor (ballfarbe);
        g.fillOval (pos_x - radius, pos_y - radius, 2 * radius, 2 * radius);
        g.drawImage (ballImg, pos_x - 10, pos_y - 10, mom);
    }
```

2．Computer 类的主要属性和主要方法

```
    {
    public final int pos_x;         // 横坐标
    public static int pos_y;        // 纵坐标
    public final int size_x;        // 挡板的宽度
    public final int size_y;        // 挡板的长度
    private Color paddelcolor;      // 电脑一方挡板的颜色
```

```
public Computer (int x, int y)
{
// 构造方法，将挡板的宽度和长度设置为 10 个像素和 50 个像素，如果读者改变挡板的长度，就可以改变游戏的难度大小
    pos_x = x;
    pos_y = y;
    size_x = 10;
    size_y = 50;
    paddelcolor = Color.blue;
    vy = 3;
}
public void ComputerMove (Ball b)
{
// 电脑运动的方式，自动地进行
    real_y = pos_y + (size_y / 2);
    if (b.vx < 0)
    {
        // Paddel oberhalb der Mitte
        if (real_y < 175)
        {pos_y += vy;
        }
        // Paddel unterhalb der Mitte
        else if (real_y > 175)
        {    pos_y -= vy;
        }
    }
    else if (b.vx > 0)
    {.....              }
    }
}
public void DrawPaddel (Graphics g)
{
// 绘制挡板的新位置
    g.setColor (paddelcolor);
    g.fillRect (pos_x, pos_y, size_x, size_y);
}
}
```

3．Player 类与 Computer 类非常相似，只有移动的方式不同，即 PaddelMove 方法：

```
public void PaddelMove (int maus_y)
    {
        int new_y_pos;
        new_y_pos = maus_y - (size_y / 2);
        if (new_y_pos < Player.pos_y) vy = (Player.pos_y - new_y_pos) / 2;
        else vy = (new_y_pos - Player.pos_y) / 2;
        Player.pos_y = new_y_pos;
    }
```

本例其他源代码详见光盘。

第五篇

控制类游戏

本篇总览

控制类游戏主要是锻炼玩家的反应能力以及对鼠标、键盘的操作能力。

本篇制作了飞行器着陆、超级放大镜、神奇画笔、拯救地球、消灭害虫、打地鼠、打字练习以及空间大战等游戏。

在编写这类游戏时要注意对程序进行优化，使用优秀的算法，以防止不连续画面的出现。对于大型游戏通常要用到 DirectX 或者 OpenGl，如果制作的游戏只要求在 Windows 系统下运行，可以使用 DirectX 进行编程；如果要编写跨平台的游戏，可以使用 OpenGL 进行编程。

实例 60　飞行器着陆人机交互

实例说明

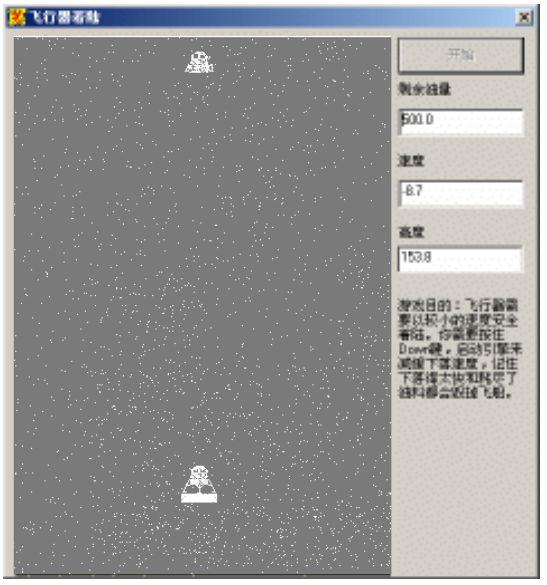


图 60-1　效果图

本例编写飞行器着陆游戏。游戏画面如图 60-1 所示。

当玩家单击“开始”按钮后，飞行器将从窗口上方落下（如果飞机自由下落，那么它将以重力加速度向下落下），飞机在飞行的过程中需要使用油料。当飞行器落得太快或者油料用尽时，将会烧毁。玩家可以使用向下的箭头“↓”，减小飞行器下落的速度，当飞行器的降落速度不是太大，并且油料没有用尽时，则着陆成功。

在本实例中主要实现了程序的界面设计以及 Timer 控件的 OnTimer 事件响应代码。

编程思路

本实例使用 Visual Basic 6.0 来实现。使用一个 Timer 控件来实现飞行器的飞行，每秒飞行速度增加 9.8（即重力加速度），同时当玩家给飞行器减速时，减少油料的数量，当油料为 0 时，飞行器坠毁；在“开始”按钮的 Click 事件中激活该 Timer 控件。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。首先其中添加三个 PictureBox 控件，其中两个用来保存飞行器图形，第三个用来显示飞行器落下的情况；然后添加一个 CommandButton 控件、三个 Edit 控件和四个 Label 控件，如图 60-2 所示。

2．设置窗体的关键属性如下：

BorderStyle =　1　Fixed Single
Caption = "飞行器着陆"
ClientHeight = 7365
ClientLeft = 4155
ClientTop = 2370

ClientWidth = 7125

LinkTopic = "Form1"

MaxButton = 0 'False

MinButton = 0 'False

ScaleHeight = 491

ScaleMode = 3 'Pixel

ScaleWidth = 475

3．选择窗体，选择其 Icon 属性，将会出现 Load Icon 对话框，如图 60-3 所示。选择其中的 xplo3.ico 文件，则将该文件作为程序运行后的图标。

图 60-2 程序的窗体设计

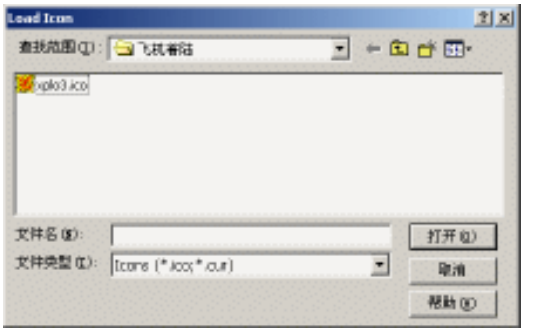


图 60-3 导入图标

4．设定三个 TextBox 控件的 Locked 属性为 True，表示文本框中的内容不可改变。

5．设定两个用来保存飞行器的图标的 PictureBox 控件的 Visible 属性为 False，使其在程序运行过程中不可见。

6．设置主画面中的 PictureBox 控件的关键属性如下：

BackColor = &H00000000&

BorderStyle = 0 'None

Height = 8535

Left = 120

ScaleHeight = 601.056

ScaleMode = 0 'User

ScaleWidth = 389.155

TabIndex = 1

Top = 120

Width = 5055

7．设置 Timer 控件的关键属性如下：

Enabled = 0 'False

Interval = 1

Left = 6201

Top = 6669

其他控件的属性设置比较简单，这里不再介绍。

8．**本实例中的核心代码是** Timer **控件的** Timer **事件，它的代码如下：**

```
Private Sub tmrgravity_Timer()
Static curtime As Long
Dim timenow As Long
Dim timediff As Long
'curtime=0 第一次进入
If curtime = 0 Then
   picEarth.Line (0, picEarth.ScaleHeight - 30)-(picEarth.ScaleWidth, picEarth.ScaleHeight), vbWhite, BF
   Randomize Timer
   画背景，随机产生一些黄颜色的点
   Dim starx As Long, stary As Long
   For starx = 0 To picEarth.ScaleWidth
      For stary = 0 To picEarth.ScaleHeight - 30
         If Rnd * 1000 < 5 Then
            SetPixelV picEarth.hdc, starx, stary, vbYellow
         End If
      Next
   Next
   timenow = GetTickCount
   curtime = timenow
Else
   timenow = GetTickCount
      擦除以前的画面
   BitBlt picEarth.hdc, 150, LandY, piclander.ScaleWidth, piclander.ScaleHeight, piclander.hdc, 0, 0,
vbSrcInvert
End If
timediff = timenow - curtime
' 计算新的垂直速度，使用重力加速度 g
vSpeed = vSpeed - ((timediff / 1000) * 10)
' 如果 Down Key 被按下，推进器将起作用，减小下落速度
' 同时检查燃料够不够用
If GetAsyncKeyState(VK_DOWN) <> 0 Then
   If Fuel > 0 Then
      ' 推进器产生的加速度是 15
      vSpeed = vSpeed + ((timediff / 1000) * 15)

      Fuel = Fuel - ((timediff / 1000) * 150)
      ' 检查油料是否为零
      If Fuel < 0 Then Fuel = 0
   Else
```

```
            Beep
        End If
    End If
    '飞行器的坐标改变
    LandY = LandY - vSpeed
    '更新 textbox 的值
    txtvspeed.Text = Format(vSpeed, "0.0")
    txtfuel.Text = Format(Fuel, "0.0")
    txtheight.Text = Format(picEarth.ScaleHeight - piclander.ScaleHeight - 30 - LandY, "0.0")
    '更新 cutime
    curtime = timenow
    '如果已经着陆
    If LandY >= picEarth.ScaleHeight - 30 - piclander.ScaleHeight Then
        LandY = picEarth.ScaleHeight - 30 - piclander.ScaleHeight
        txtheight.Text = Format(picEarth.ScaleHeight - piclander.ScaleHeight - 30 - LandY, "0.0")
        '定时器 disable 掉
        tmrgravity.Enabled = False
        cmdgo.Enabled = True
        Fuel = 500
        LandY = 0
        curtime = 0
        '通过速度检查是不是 safe landing, 根据情况画不同的飞行器
        If vSpeed > Safe_Speed Then
            '显示安全降落的图片
            BitBlt picEarth.hdc, 150, LandY, piclander.ScaleWidth, piclander.ScaleHeight, piclander.hdc, 0, 0,
            vbSrcInvert
            MsgBox "祝贺，你已经安全降落!"
            vSpeed = 0
    Else
            '非安全降落的情况
            BitBlt picEarth.hdc, 150, LandY, piclander.ScaleWidth, piclander.ScaleHeight, picsmash.hdc, 0, 0,
vbSrcInvert
            MsgBox "Oh，飞行器废了!"
            vSpeed = 0
        End If
    Else
    '重新绘制飞机
        BitBlt picEarth.hdc, 150, LandY, piclander.ScaleWidth, piclander.ScaleHeight, piclander.hdc, 0, 0,
vbSrcInvert
    End If
    End Sub
```

实例 61　飞行器着陆组件

实例说明

续前面的实例，如图 61-1 所示。

本例主要完成了程序的其他代码，包括剩余油量、飞行器速度、飞行器高度等数值的显示，以及在程序的 Load 事件中初始化游戏所使用的变量等。

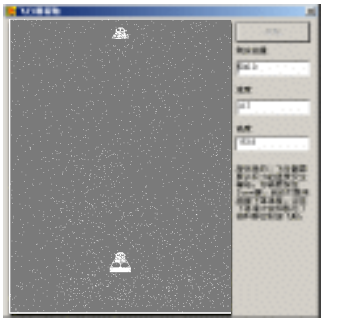


图 61-1　效果图

编程思路

在本实例中实现了程序的一些辅助功能，定义了程序所使用的变量，通过程序的 Load 事件初始化了这些变量的值。

创作步骤

续前面的实例。本实例的其他代码如下：

```
Option Explicit
'API Declares
Private Declare Function GetTickCount Lib "kernel32" () As Long
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 SetPixelV Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long,
ByVal crColor As Long) As Long

Private Declare Function GetAsyncKeyState Lib "user32" (ByVal vKey As Long) As Integer
Private Const VK_DOWN = &H28
Private Const Safe_Speed = -1.7    能够安全降落的速度
' 飞行器的垂直速度
Private vSpeed As Double
' 飞行器的 y 坐标
Private LandY As Double
```

```
'油料的数量
Private Fuel As Double
Private Sub cmdgo_Click()
tmrgravity.Enabled = True    '启动定时器
cmdgo.Enabled = False
End Sub

Private Sub Form_Load()
'初始化油料和垂直速度
Fuel = 500
vSpeed = 0
LandY = 0
txtvspeed.Text = Format(vSpeed, "0.0")
txtfuel.Text = Format(Fuel, "0.0")
txtheight.Text = Format(picEarth.ScaleHeight - piclander.ScaleHeight - 30 - LandY, "0.0")
Dim msg$
msg$ = msg$ + "游戏目的：飞行器需要以较小的速度安全着陆。"
msg$ = msg$ + "你需要按住 Down 键，启动引擎来减缓下落速度，"
msg$ = msg$ + "记住下落得太快和耗尽了油料都会毁掉飞船。"
GameInfo.Caption = msg$
End Sub
```

实例 62　超级放大镜人机交互

实例说明

本例制作超级放大镜，如图 62-1 所示。

在窗口中以指定的放大倍数显示了鼠标所在位置的内容。在“放大倍数”菜单中可以设置放大倍数。

本实例主要实现了程序的界面设计，定义了程序要使用的公共变量。

图 62-1　效果图

编程思路

本实例通过 MainMenu 控件和 Label 控件完成了程序的界面设计。然后定义了多个公共变量用来保存系统的信息，它们的具体含义下面有详细介绍。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 MainMenu 控件、一个 Label 控件和一个 Timer 控件。如图 62-2 所示。

图　62-2

2．设置窗体的关键属性如下：

```
BorderIcons = [biSystemMenu]
BorderStyle = bsSingle
Caption = 'Form1'
```

ClientHeight = 240

ClientWidth = 240

Color = clBtnFace

FormStyle = fsStayOnTop

Menu = MainMenu1

3．**双击** MainMenu **控件，打开** MainMenu **控件设置窗口，向其中添加两个菜单，如图** 62-3 **所示。**

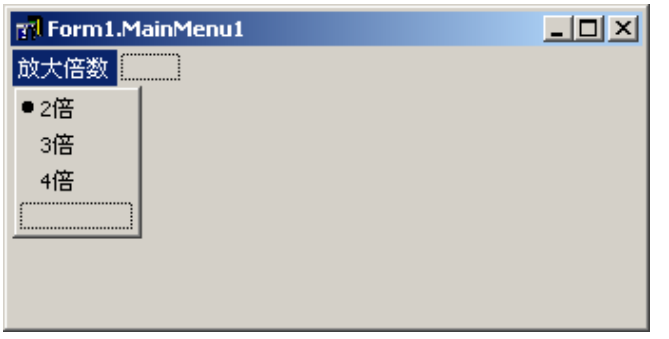


图 62-3

4．**设置** Timer **控件的关键属性如下：**

Interval = 1

5．**本实例使用了多个公共变量，其含义如下：**

```
FnZoomMultiple: integer;      // 放大倍数
ptNow,ptNow2: TPoint;         // 分别用来即时采集鼠标位置以及用来处理的位置,之所以分开来用,
                                 是防止共享冲突
rectSrc,rectDest: TRect;      // 用来保存将被放大的源矩形位置和目标矩形位置
CanvasWork: TCanvas;          // 工作画布对象
procedure IdleLoop(Sender: TObject;var DoneState:boolean);          // IDLE 事件的处理函数
```

实例 63　超级放大镜组件

实例说明

续前面的实例，如图 63-1 所示。

本例讲述了程序的代码实现，包括如何得到 Windows 窗口的信息，如何把这些信息保存到图片中，并且显示到放大镜窗口。

在这个例子中详细讲述了使用 Delphi 中的 API 函数控制系统信息的方法。

图 63-1　效果图

编程思路

本程序首先设置窗体始终显示在最前，然后利用 Windows API 函数得到鼠标在窗口中的位置，将桌面窗口中的内容复制下来，并以指定的倍数显示出来。这里用到了多个 API 函数，下面介绍它们的含义和用法。

CreateCompatibleDC 函数：该函数创建一个与指定设备兼容的 DC（句柄）。其原型如下：

```
HDC CreateCompatibleDC(
    HDC hdc
  );
```

CreateCompatibleBitmap 函数：该函数创建与指定的设备环境相关的设备兼容的位图。其原型如下：

```
HBITMAP CreateCompatibleBitmap(
    HDC hdc,    // 设备主题句柄
    int nWidth, // 位图的宽度，以像素计算
    int nHeight // 位图的高度，以像素计算
  );
```

SelectObject 函数：该函数选择一对象到指定的设备上下文环境中，该新对象替换先前的相同类型的对象。其原型如下：

```
HGDIOBJ SelectObject(
    HDC hdc,    // 设备相关主题
    HGDIOBJ hgdiobj      // 对象句柄
  );
```

SelectPalette 函数：该函数选择指定的逻辑调色板到一个设备环境中。其原型如下：

```
HPALETTE SelectPalette(
    HDC hdc,    // 设备相关主题
    HPALETTE hpal,       // 标识被选择的逻辑调色板
```

BOOL bForceBackground // 确定逻辑调色板是否被强行作为一个背景调色板，如果该值为 True，RealizePalette 函数就使逻辑调色板以最好的可能方式映射成物理调色板中已有的颜色，这种情况常发生；如果该值为 False，RealizePalette 函数使逻辑调色板拷贝到设备调色板中，这时应该用在前景色（如果 hdc 参数是一个内存设备环境，该参数被忽略）

);

RealizePalette 函数：该函数从当前逻辑调色板中映射调色板入口点到系统调色板中。其原型如下：

```
UINT RealizePalette(
    HDC hdc    // 设备相关主题
);
```

GetCursorPos 函数：该函数检取光标的位置，以屏幕坐标表示。其原型如下：

```
BOOL GetCursorPos(
    LPPOINT lpPoint        // POINT 结构指针，该结构接收光标的屏幕坐标
);
```

创作步骤

续前面的实例。本实例源代码如下：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    ……（此处代码略，详见光盘）
  private
    { Private declarations }
    FnZoomMultiple: integer;      // 放大倍数
    ptNow,ptNow2: TPoint;         // 分别用来即时采集鼠标位置以及用来处理的位置,之所以分开来用,是防止共享冲突
    rectSrc,rectDest: TRect;      // 用来保存将被放大的源矩形位置和目标矩形位置
    CanvasWork: TCanvas;          // 工作画布对象
    procedure IdleLoop(Sender: TObject;var DoneState:boolean);     // IDLE 事件的处理函数
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation

{$R *.dfm}
```

```
procedure TForm1.FormCreate(Sender: TObject);
var
   ScreenCanvas: TCanvas;     // 保存当前整个桌面的画布对象
   dc: HDC;                   // dc 是当前桌面的设备环境
   bmpWork: HBITMAP;          // 用来指定当前的工作画布使用的 BITMAP 对象句柄
   dcWork: HDC;               // 用来产生需要使用的画布的设备环境
begin
   FnZoomMultiple := 2;
   rectDest := RECT(0,0,ClientWidth,ClientHeight);
   rectSrc := RECT(0,0,ClientWidth div FnZoomMultiple,ClientHeight div FnZoomMultiple);
   ScreenCanvas := TCanvas.Create;
   CanvasWork := TCanvas.Create;
   dc := GetDC(0);
   ScreenCanvas.Handle := dc;             // 得到桌面设备环境的句柄
   dcWork := CreateCompatibleDC(Canvas.Handle);
   bmpWork := CreateCompatibleBitmap(Canvas.Handle,ClientWidth,ClientHeight);
   SelectObject(dcWork,bmpWork);
   SelectPalette(dcWork,0,false);
   CanvasWork.Handle := dcWork;       // 初始化工作环境画布
   RealizePalette(ScreenCanvas.Handle);
   CanvasWork.CopyRect(rectDest,ScreenCanvas,rectSrc);
   Canvas.CopyRect(rectDest,CanvasWork,rectDest);              // 初始化显示
   Application.OnIdle := IdleLoop;
   ReleaseDC(0,dc);
   ScreenCanvas.Free;
end;

procedure TForm1.IdleLoop(Sender: TObject;var DoneState:boolean);
var
   ScreenCanvas: TCanvas;          // 保存当前整个桌面的画布对象
   dc: HDC;       // dc 是当前桌面的设备环境
begin
   DoneState := false;
   ptNow2 := ptNow;           // 把即时采集的鼠标坐标转移给工作坐标
   ScreenCanvas := TCanvas.Create;
   rectSrc.Left := ptNow2.X - ClientWidth div (2*FnZoomMultiple);
   rectSrc.Right := ptNow2.X + ClientWidth div (2*FnZoomMultiple);
   rectSrc.Top := ptNow2.Y - ClientHeight div (2*FnZoomMultiple);
```

```
    rectSrc.Bottom := ptNow2.Y + ClientHeight div (2*FnZoomMultiple);
    // 根据采集到的鼠标坐标和放大倍数,调整矩形的位置,保证鼠标所在的位置为中央
    dc := GetDC(0);
    ScreenCanvas.Handle := dc;
    RealizePalette(ScreenCanvas.Handle);
    CanvasWork.Brush.Style := bsSolid;
    CanvasWork.Brush.Color := clBlack;
    CanvasWork.FillRect(rectDest);
  CanvasWork.CopyRect(rectDest,ScreenCanvas,rectSrc);          // 把屏幕上相应位置的内容拷贝到工作
                                                                  画布上
    Canvas.CopyRect(rectDest,CanvasWork,rectDest);      // 显示工作画布上的内容
    ReleaseDC(0,dc);
    ScreenCanvas.Free;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
    GetCursorPos(ptNow);              // 即时采集鼠标的位置
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
    RealizePalette(Canvas.Handle);
    Canvas.CopyRect(rectDest,CanvasWork,rectDest);      // 刷新窗口画布上的内容
end;

procedure TForm1.MenuZoom2Click(Sender: TObject);
begin
    FnZoomMultiple := 2;        // 更改放大倍数
end;

      ……（此处代码略，详见光盘）
procedure TForm1.FormDestroy(Sender: TObject);
begin
    CanvasWork.Free;
end;
end.
```

实例 64　神奇画笔人机交互

实例说明

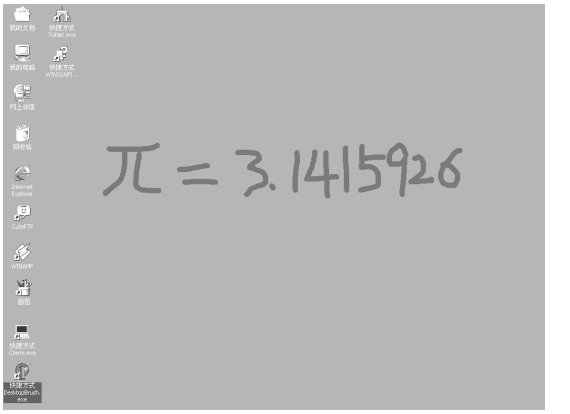


图 64-1　效果图

本例制作神奇画笔，如图 64-1 所示。

本例程序运行后，将把当前的桌面作为背景，鼠标成为画笔，可以在背景中绘制图形。右击鼠标将弹出快捷菜单，在其中可以选择画笔的颜色和宽度，还可以退出程序。

本实例主要实现了程序的界面设计，同时还实现了程序的初始化工作，在程序的 OnCreate 事件中完成了桌面背景的捕获，并把该背景作为程序的背景。

编程思路

本实例使用 Delphi 6.0 来实现。在程序的 OnCreate 事件中获得当前桌面的句柄，并将其保存到一个位图对象中，然后把该位图作为程序的背景全屏显示；同时屏蔽掉系统的功能键（比如 Ctrl+Alt+Del 键结束任务等）。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 PopupMenu 控件。

2．设置窗体的关键属性如下：

BorderStyle = bsNone
Caption = 'Form1'
ClientHeight = 453
ClientWidth = 688
Color = clBtnFace
PopupMenu = PopupMenu1
WindowState = wsMaximized

3．设置 PopupMenu 控件的菜单项如图 64-2 所示。

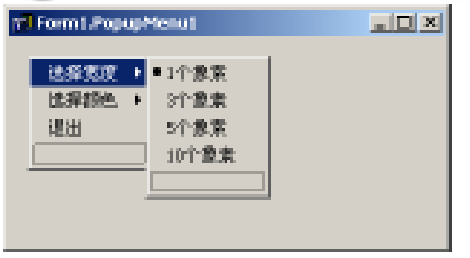


图 64-2

4．本实例使用了多个私有变量，它们的含义如下：

```
FclColor: TColor;          // 定义当前画笔的颜色
FnWidth: integer;          // 定义当前画笔的宽度
bStartDraw: boolean;       // 定义当前画笔的状态,1 表示开始画,0 表示停止画
ptStart,ptEnd:TPoint;      // 定义每一段画线的起始点和终点
rectWork: TRect;           // 用来保存当前整个桌面的矩形对象,其值为(0,0,Screen.Width,Screen.Height)
CanvasWork: TCanvas;       // 工作画布
```

5．在程序的 OnCreate 事件中获得了当前窗口的句柄，又把当前窗口作为背景显示了出来，同时屏蔽掉系统的功能键（比如 Ctrl+Alt+Del 键结束任务等）。该过程的具体实现如下：

```
procedure TForm1.FormCreate(Sender: TObject);
var
  ScreenCanvas: TCanvas;       // 保存当前整个桌面的画布对象
  ScreenBitmap: TBitmap;       // 保存当前整个桌面的位图对象
  dc: HDC;                     // dc 是当前桌面的设备环境
  bmpWork: HBITMAP;            // 用来指定当前的工作画布使用的 BITMAP 对象句柄
  dcWork: HDC;                 // dcWork 为产生的工作画布的设备环境
begin
  FnWidth := 1;                // 初始化宽度为 1
  FclColor := clRed;           // 初始化画笔颜色红色
  Sleep(300);                  // 暂停 300s,为了保证在开始拷贝桌面以前,自己的窗口不出现,避免
                               挡住桌面
  rectWork := Rect(0,0,Screen.Width,Screen.Height);
  ScreenCanvas := TCanvas.Create;
  CanvasWork := TCanvas.Create;
  ScreenBitmap := TBitmap.Create;
  ScreenBitmap.Width := Screen.Width;
  ScreenBitmap.Height := Screen.Height;     // 初始化桌面位图对象
  dc := GetDC(0);                           // 以下三行代码把桌面的内容拷贝到桌面位图对象
                                            ScreenBitmap 中
  ScreenCanvas.Handle := dc;
  ScreenBitmap.Canvas.CopyRect(Rect(0,0,Screen.Width,Screen.Height),ScreenCanvas,Rect(0,0,Screen.W
```

```
    idth,Screen.Height));
    dcWork := CreateCompatibleDC(Canvas.Handle);
    bmpWork := CreateCompatibleBitmap(Canvas.Handle,Screen.Width,Screen.Height);
    SelectObject(dcWork,bmpWork);
    SelectPalette(dcWork,ScreenBitmap.Palette,false);
    CanvasWork.Handle := dcWork;              // 初始化 CanvasWork 的设备环境以及调色板
    CanvasWork.Draw(0,0,ScreenBitmap);        // 绘制底图
    CanvasWork.Pen.Width := FnWidth;
    CanvasWork.Pen.Color := FclColor;         // 设置画笔的宽度和颜色
    RealizePalette(CanvasWork.Handle);
    Canvas.CopyRect(rectWork,CanvasWork,rectWork);      // 把工作画布上的相应内容拷贝到整个画布
                                                           上显示出来
    ReleaseDC(0,dc);
    ScreenBitmap.Free;
    ScreenCanvas.Free;                                  // 释放资源
    SystemParametersInfo(SPI_SCREENSAVERRUNNING, 1, nil, 0);        // 屏蔽掉系统键
end;
```

6．在程序的 OnDestory 事件中将要释放变量，并且启动程序的功能键，其代码如下：

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
    SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, nil, 0);        // 打开系统键
    CanvasWork.Free;
end;
```

实例 65　神奇画笔组件

实例说明

续前面的实例，如图 65-1 所示。

本例主要实现了和用户的交互响应。包括捕捉窗体的 OnMouseDown 事件和 OnMouseMove 事件，在这两个事件中调用 Delphi 的画布对象（Canvas）进行图形绘制。

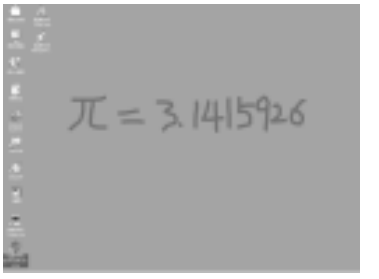


图 65-1　效果图

编程思路

本实例中捕捉窗体的 OnMouseDown 事件和 OnMouseMove 事件，在这两个事件中调用 Delphi 的绘图方法进行图形的绘制，然后使用 Canvas.CopyRect 方法将绘制的内容复制到当前窗体的画布中。

创作步骤

续前面的实例。本实例的其他代码如下：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    ……（此处代码略，详见光盘）
  private
    FclColor: TColor;          // 定义当前画笔的颜色
    FnWidth: integer;          // 定义当前画笔的宽度
    bStartDraw: boolean;       // 定义当前画笔的状态,1 表示开始画,0 表示停止画
    ptStart,ptEnd:TPoint;      // 定义每一段画线的起始点和终点
    rectWork: TRect;           // 用来保存当前整个桌面的矩形对象,其值为(0,0,Screen.Width,Screen.Height)
    CanvasWork: TCanvas;       // 工作画布

  public
  end;
var
  Form1: TForm1;
```

```
implementation
{$R *.dfm}
// 窗体的 OnCreate 事件和 OnDestroy 事件在前面一节
// 该函数根据鼠标右键所选定的颜色来设置画笔的颜色,利用了子菜单对象的 tag 属性来区分
procedure TForm1.ColorClick(Sender: TObject);
begin
   TMenuItem(Sender).Checked := true;
   case TMenuItem(Sender).Tag of
      1001: FclColor := clRed;
      1002: FclColor := clGreen;
      1003: FclColor := clBlue;
      1004: FclColor := clYellow;
      1005: FclColor := clBlack;
      1006: FclColor := clFuchsia;
   end;
   CanvasWork.Pen.Color := FclColor;
end;
// 该函数根据鼠标右键所选定的颜色来设置画笔的宽度,利用了子菜单对象的 tag 属性来区分
procedure TForm1.WidthClick(Sender: TObject);
begin
   TMenuItem(Sender).Checked := true;
   case TMenuItem(Sender).Tag of
      1101: FnWidth := 1;
      1102: FnWidth := 3;
      1103: FnWidth := 5;
      1104: FnWidth := 10;
   end;
   CanvasWork.Pen.Width := FnWidth;
end;

procedure TForm1.MenuExitClick(Sender: TObject);
begin
   Close;
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
   Shift: TShiftState; X, Y: Integer);
begin
   if Button in [mbLeft] then          // 如果鼠标左键被按下,则开始画图(bStartDraw := true)
```

```
  begin
    bStartDraw := true;
    ptStart.x := X;
    ptStart.y := Y;      // 记录起始点为当前按下的点
  end;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if Button in [mbLeft] then        // 如果鼠标左键被释放,则画图结束,bStartDraw := false;
    bStartDraw := false;
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  if bStartDraw = true then                    // 如果正在画图中,则开始进行画图工作
  begin
    ptEnd.x := X;
    ptEnd.y := Y;                              // 初始化当前的结束点为当前鼠标移动的位置
    CanvasWork.MoveTo(ptStart.x,ptStart.Y);
    CanvasWork.LineTo(ptEnd.x,ptEnd.y);        // 从当前的起始点画线到当前的结束点
    ptStart := ptEnd;                          // 起始点被用过,结束点变成起始点
    RealizePalette(CanvasWork.Handle);
    Canvas.CopyRect(rectWork,CanvasWork,rectWork);      // 把当前图像的改动反映到 Form 的
                                                        Canvas 上
  end;
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
  RealizePalette(CanvasWork.Handle);
  Canvas.CopyRect(rectWork,CanvasWork,rectWork);      // 在该函数中刷新,把工作画布上的内容
                                                      拷贝到当前 Form 的画布上
end;
end.
```

实例 66　弹性小球人机交互

实例说明

本例编写弹性小球游戏，画面如图 66-1 所示。

本例运行时小球会来回弹跳，当它撞着区域的边缘时，会沿相反的方向行进，并且在行进的过程中，小球的半径会不断变大，而且利用窗口右下角的 TrackBar 控件，便可以随意调整小球来回运行的速度。熟悉这个例子之后，还可以进一步扩展它，使小球沿着不同的角度在窗口上做弹性碰撞。

本例主要实现了程序的界面设计。

图 66-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。程序使用了 TrackBar 控件，通过对其 Position 属性的设计可以控制小球的移动速度。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。在面板控件中添加一个 Panel 控件，其中放置一个 Shape 控件和一个 Image 控件，另外添加一个 Timer 控件、一个 TrackBar 控件、一个 SpinEdit 控件和两个 Button 控件。

2．设置窗体的关键属性如下：

Left = 192
Top = 107
Width = 441
Height = 263
Caption = 'Form1'

3．设置 Panel 控件的关键属性如下：

BevelInner = bvLowered
BevelOuter = bvNone
BiDiMode = bdRightToLeft
BorderStyle = bsSingle

Enabled = False

4．**设置** Shape **控件的关键属性如下：**

Left = 248

Top = 80

Width = 25

Height = 25

Hint = '**这是个运动的球体**'

Brush.Color = clRed

ParentShowHint = False

Shape = stCircle

ShowHint = True

5．**向** Image **中添加一张图片，并设置其关键属性如下：**

Stretch = True

6．**设置** TrackBar **控件的关键属性如下：**

Max = 1000

Min = 1

Orientation = trHorizontal

Frequency = 1

Position = 1

SelEnd = 0

SelStart = 0

TabOrder = 3

TickMarks = tmBottomRight

TickStyle = tsManual

OnChange = TrackBar1Change

7．**设置** SpinEdit **控件的关键属性如下：**

MaxValue = 0

MinValue = 0

TabOrder = 4

Value = 0

OnChange = SpinEdit1Change

8．**设置** Timer **控件的关键属性如下：**

Enabled = False

Interval = 1

OnTimer = Timer1Timer

Left = 336

Top = 16

实例 67　弹性小球组件

实例说明

续前面的实例，如图 67-1 所示。

本例主要完成程序的代码部分。另外本程序还使用了一个 initialization 标识符，在该标识符下，可以初始化一些变量的值。

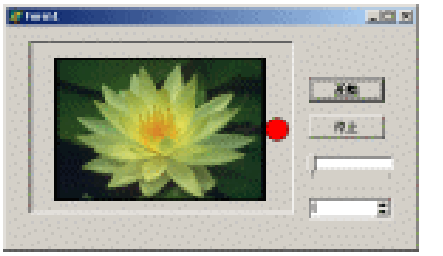


图 67-1　效果图

编程思路

在 Delphi 中，有一个 Timer 组件，在运行程序的时候，看不见它，也不能直接对它进行操作。Timer 组件每隔一段固定的时间就会自动触发 OnTimer()事件，本例正是利用这个功能，在每隔一段时间触发的 OnTimer()事件中，小球沿着给定的轨道前进一段距离，当小球到达指定区域的边缘时，再令它沿相反的方向前进。利用控件 TButton 可以激活 Timer 组件，触发 OnTimer()事件，利用 TrackBar 控件可以控制 OnTimer()事件触发的时间间隔，可设为 1~1000ms，可以随意控制小球行进的速度。

创作步骤

续前面的实例。

本实例的源代码如下：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Spin, ComCtrls, jpeg;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Shape1: TShape;
    Image1: TImage;
    Button1: TButton;
    Button2: TButton;
    TrackBar1: TTrackBar;
    SpinEdit1: TSpinEdit;
```

```
    Timer1: TTimer;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

……**（此处代码略，详见光盘）**

```
begin
  if shape1.left>0 then
  begin
    shape1.left:=shape1.left-10;
    shape1.width:=(shape1.width+1)mod 70;
    shape1.height:=(shape1.height+1)mod 70;
  end
  else
    i:=2;
  end;
  if i=2 then
  begin
    if shape1.left<(panel1.width-shape1.width-5) then
    begin
      shape1.left:=shape1.left+10 ;
      shape1.width:=(shape1.width+1)mod 70;
      shape1.height:=(shape1.height+1)mod 70;
    end
    else
      i:=1;
    end;
  end;
initialization
  i:=1;
end.
```

实例 68　拯救地球人机交互

实例说明

本例编写拯救地球游戏，画面如图 68-1 所示。

地球从上方跌落下来，玩家使用鼠标控制下面的小滑块将地球接住，然后地球就会再次弹起，按照反射规律弹到边墙上，最后还要落到下面，玩家需要再次将其接住，如此往复，当地球落到下面，玩家没有接住时，玩家输。另外，玩家还可以设置地球的下落速度。

本例实现了程序的窗体设计以及其核心部分 IdleLoop 函数的代码编写。

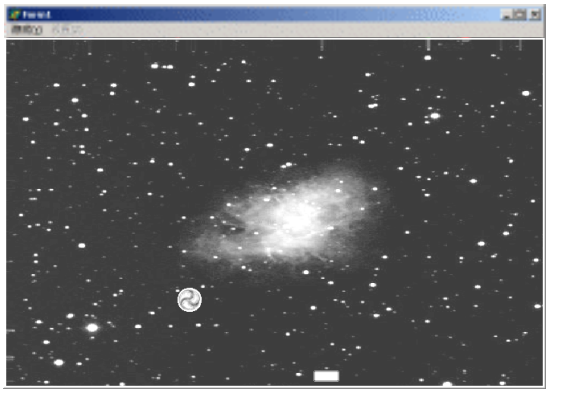


图 68-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。程序的核心部分是 IdleLoop 函数，在该函数中利用了小变的技巧，即每次复制的时候，只拷贝改变部分的矩形，比如矩形(0，0，10，10)向右下移动到了(5，5，15，15)，则先把原来所占的(0，0，10，10)用背景上对应的部分给覆盖上，然后再定义改变后的矩形大小为(0，0，15，15)，只把这个矩形内，工作画布上的内容拷贝到背景上，这样大大加快了游戏的运行速度。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 MainMenu 控件。

2．设置窗体的关键属性如下：

Left = 192
Top = 107
BorderStyle = bsNone
Caption = 'Form1'
ClientHeight = 434
ClientWidth = 688
Color = clBtnFace
Menu = MainMenu1

3．设置 MainMenu 控件的菜单如图 68-2 所示。其中“游戏”菜单中包含“开始新游戏”和“退出游戏”两个菜单项。

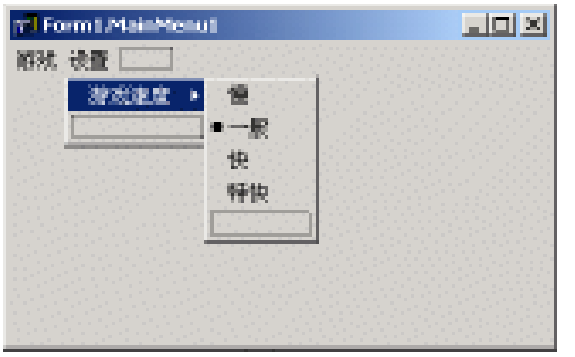


图 68-2

4．本实例的核心函数 IdleLoop 的代码如下：

```
procedure TForm1.IdleLoop(Sender: TObject;var DoneState:boolean);
begin
  DoneState := false;
  Sleep(FnTime);// 暂停一段时间,调整游戏的速度
  CanvasWork.CopyRect(rectBall,CanvasBg,rectBall);
  CanvasWork.CopyRect(rectPaddle,CanvasBg,rectPaddle);//
  if ptBall.Y <= 0 then// 如果向上出了边界,则改变方向为下
    nBallYDir := 1
      else if (ptBall.Y >= ClientHeight-56) and (ptBall.X+32 >= ptPaddle.X) and (ptBall.X <=
  ptPaddle.X+32) then // 如果向下有球拍接着,则改变方向为上
    nBallYDir := -1;
  if ptBall.X <= 0 then
    nBallXDir := 1
  else if ptBall.X >= ClientWidth-32 then
    nBallXDir := -1;
  ptBall.X := ptBall.X + 2*nBallXDir;// 修改小球的左上角的坐标
  ptBall.Y := ptBall.Y + 2*nBallYDir;
  if (ptBall.Y > ClientHeight - 32) then
  begin
    bGameStarted := false;// 游戏结束
    ShowCursor(true);
        if (Application.MessageBox(地球没有了,你也死掉了,要重新开始吗?','^\_^',MB_YESNO) =
  IDYES) then
    begin
      ptBall := POINT(0,0);// 继续开始,则将小球定位到最左上角
```

```
            bGameStarted := true;
            ShowCursor(false);
        end else
        begin
            Application.OnIdle := nil;
            MenuSetup.Enabled := TRUE;// 暂停游戏,允许设置游戏的速度
        end;
    end;
    rectBall2 := rectBall;
    rectBall := RECT(ptBall.X,ptBall.Y,ptBall.X+picBall.Width,ptBall.Y+picBall.Height);
    if (rectPaddle2.Left - rectPaddle.Left < 0) then
        rectPaddle2.Right := rectPaddle.Right// 以下的几个 if 的具体变化过程,见函数前的注释
    else
        rectPaddle2.Left := rectPaddle.Left;
    rectPaddle2 := rectPaddle;
    rectPaddle := RECT(ptPaddle.X,ptPaddle.Y,ptPaddle.X+picPaddle.Width,ptPaddle.Y+picPaddle.Height);
    if (rectBall2.Left - rectBall.Left < 0) then
        rectBall2.Right := rectBall.Right
    else
        rectBall2.Left := rectBall.Left;
    if (rectBall2.Top - rectBall.Top < 0) then
        rectBall2.Bottom := rectBall.Bottom
    else
        rectBall2.Top := rectBall.Top;
    CanvasWork.Draw(ptBall.X,ptBall.Y,picBall.Icon);
    CanvasWork.Draw(ptPaddle.X,ptPaddle.Y,picPaddle.Icon);
    RealizePalette(CanvasWork.Handle);
        Canvas.CopyRect(rectBall2,CanvasWork,rectBall2);
    // 把背景画布上的相应内容拷贝到整个画布上显示出来
    Canvas.CopyRect(rectPaddle2,CanvasWork,rectPaddle2);
end;
```

实例 69　拯救地球组件

实例说明

续前面的实例。如图 69-1 所示。

本例完善了“拯救地球”游戏。在程序的初始化阶段定义了小球的位置以及小球移动速度的大小；在程序的 Destory 事件中释放变量和内存空间；同时还响应用户的鼠标移动事件 OnMouse Move，不断移动下方的挡板，以防止地球落下了。

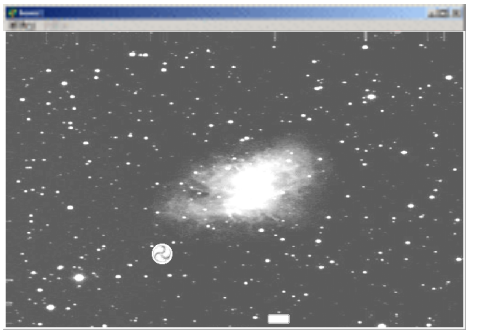


图 69-1　效果图

编程思路

在本实例中定义了多个公共变量，通过它们来判断小球的位置和挡板的位置，判断当小球即将落下时挡板是否能够接住小球，如果挡板接住小球，那么小球接下来的移动路线是什么样的。同时还判断小球是否坠毁，如果坠毁将显示提示信息。

创作步骤

续前面的实例。本实例的其他代码如下：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Menus;
type
  TForm1 = class(TForm)
    ……（此处代码略，详见光盘）
  private
    picBg,picPaddle,picBall: TPicture;          // 用来保存背景、球拍和球的图像
    CanvasBg,CanvasWork: TCanvas;               // 背景画布以及工作画布
    rectBall,rectBall2,rectPaddle,rectPaddle2,rectBg: TRect;   // 球、球拍变化前后的矩形、背景矩形
    ptBall,ptPaddle: TPoint;                    // 球和球拍的左上角的坐标
    nBallXDir,nBallYDir: integer;               // 球的 X,Y 方向判断,+1 表示向右(下),-1 表示向左(上)
    nPaddleCenter:integer;                      // 球拍中心的 x 坐标
    bGameStarted: boolean;                      // 判断游戏的状态,1 表示开始,0 表示结束
```

```
    strCurPath: string;                          // 当前可执行文件的路径
    FnTime:integer;                              // 小球每次移动的间隔时间,用来修改游戏的速度
    procedure IdleLoop(Sender: TObject;var DoneState:boolean);     // IDLE 事件的处理函数
  public
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
begin
  ptBall := POINT(0,0);       // 初始化小球在最左上角
  nBallXDir := 1;
  nBallYDir := 1;             // 初始化小球移动方向为向右、向下,
  FnTime := 2;                // 初始化间隔 2/1000s
  strCurPath := ExtractFilePath(Application.ExeName);
  bGameStarted := false;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  picBg.Free;
  picBall.Free;
  picPaddle.Free;
  CanvasWork.Free;
  CanvasBg.Free;
end;

procedure TForm1.MenuExitClick(Sender: TObject);
begin
  ShowCursor(true);           // 显示鼠标,因为在游戏中隐藏了鼠标的显示
  Close;
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
  RealizePalette(CanvasWork.Handle);
  Canvas.CopyRect(rectBg,CanvasWork,rectBg);        // 在该函数中刷新,把工作画布上的内容拷贝到
                                                    当前 Form 的画布上
end;
```

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  nPaddleCenter := X;
  if (nPaddleCenter < (picPaddle.Width div 2 )) then
    nPaddleCenter := picPaddle.Width div 2;
  if(nPaddleCenter > (ClientWidth - picPaddle.Width div 2)) then
    nPaddleCenter := ClientWidth - (picPaddle.Width div 2);
  ptPaddle.X := nPaddleCenter - (picPaddle.Width div 2);
end;

procedure TForm1.FormActivate(Sender: TObject);
var
  bmpWork,bmpBg: HBITMAP;          // 用来指定当前的工作及背景画布使用的 BITMAP 对象句柄
  dcWork,dcBg: HDC;                // 产生的工作画布和背景画布的设备环境
begin
  rectBg := RECT(0,0,ClientWidth,ClientHeight);
  ptPaddle := POINT(ClientWidth div 2,ClientHeight - 40);
  CanvasWork := TCanvas.Create;
  CanvasBg := TCanvas.Create;
  picBg := TPicture.Create;        // 初始化背景、球拍、小球的图像以及所占的矩形
  picBg.Bitmap.LoadFromFile(strCurPath+'RES\bg.bmp');
  picPaddle := TPicture.Create;
  picPaddle.Icon.LoadFromFile(strCurPath+'RES\Paddle.ico');
  nPaddleCenter := ptPaddle.X + (picPaddle.Width div 2);
  rectPaddle := RECT(ptPaddle.X,ptPaddle.Y,ptPaddle.X+picPaddle.Width,ptPaddle.Y+picPaddle.Height);
  picBall := TPicture.Create;
  picBall.Icon.LoadFromFile(strCurPath+'RES\BALL1.ico');
  rectBall := RECT(ptBall.X,ptBall.Y,ptBall.X+picBall.Width,ptBall.Y+picBall.Height);
  dcBg := CreateCompatibleDC(Canvas.Handle);
  bmpBg := CreateCompatibleBitmap(Canvas.Handle,ClientWidth,ClientHeight);
  SelectObject(dcBg,bmpBg);
  SelectPalette(dcBg,picBg.Bitmap.Palette,false);
  CanvasBg.Handle := dcBg;         // 初始化 CanvasBg 的设备环境以及调色板
  CanvasBg.StretchDraw(rectBg,picBg.Bitmap);
  dcWork := CreateCompatibleDC(Canvas.Handle);
  bmpWork := CreateCompatibleBitmap(Canvas.Handle,ClientWidth,ClientHeight);
  SelectObject(dcWork,bmpWork);
  SelectPalette(dcWork,picBg.Bitmap.Palette,false);
  CanvasWork.Handle := dcWork;     // 初始化 CanvasWork 的设备环境以及调色板
  RealizePalette(CanvasBg.Handle);
```

```
    CanvasWork.CopyRect(rectBg,CanvasBg,rectBg);
    CanvasWork.Draw(0,0,picBall.Icon);
    CanvasWork.Draw(ptPaddle.X,ptPaddle.Y,picPaddle.Icon);
    RealizePalette(CanvasWork.Handle);
    Canvas.CopyRect(rectBg,CanvasWork,rectBg);    // 把背景画布上的相应内容拷贝到整个画布上显示出来
  end;
  procedure TForm1.MenuNewGameClick(Sender: TObject);
  begin
    MenuSetup.Enabled := false;                   // 开始游戏,禁止设置游戏的速度
    ShowCursor(false);
    bGameStarted := true;
    Application.OnIdle := IdleLoop;
  end;
  procedure TForm1.N5Click(Sender: TObject);
  begin
    FnTime := 5;
  end;
  procedure TForm1.N7Click(Sender: TObject);
  begin
    FnTime := 1;
  end;
  procedure TForm1.N8Click(Sender: TObject);
  begin
    FnTime := 0;
  end;
  end.
```

实例 70　消灭害虫人机交互

实例说明

图 70-1　效果图

本例编写消灭害虫游戏，画面如图 70-1 所示。

程序运行后，单击“游戏”菜单下的“新游戏”菜单项，将会开始游戏，然后将会在窗口中的上、下、左、右以及中间随机地出现害虫，玩家需要使用小锤子将害虫消灭。在窗口的下方显示了游戏进行的时间、打中的害虫数、失误的单击数、逃跑的害虫数以及得分。

本实例主要实现了程序的界面设计。

编程思路

本实例使用 Delphi 6.0 实现，使用 Image 控件来显示游戏的开始和结束画面，使用多个 Label 控件来显示游戏当前的状态，同时还创建了一个副窗体，用来显示游戏设置窗口。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Timer 控件、一个 MainMenu 控件、一个 Image 控件、一个 GroupBox 控件（在该控件中包含 10 个 Label 控件）。

2．设置窗体的关键属性如下：

Left = 338
Top = 175
Width = 290
Height = 459
Cursor = crIBeam
Caption = ’消灭害虫’
Color = clSilver
Menu = MainMenu1

3．设定 Timer 控件的关键属性如下：

Enabled = False

Interval = 100

OnTimer = Timer1Timer

Left = 32

4．设定 MainMenu 控件的菜单项如图 70-2 所示。其中“帮助”菜单下包括“帮助”和“关于”两个菜单项。

5．向 Image 控件中导入一张图片，如图 70-3 所示。同时设定该控件的 AutoSize 属性和 Transparent 属性为 True。

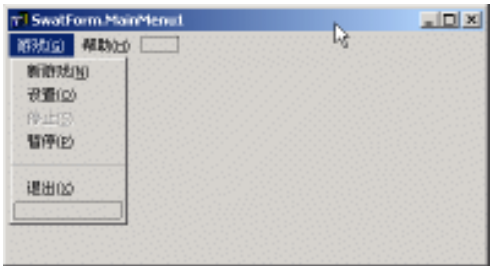


图　70-2

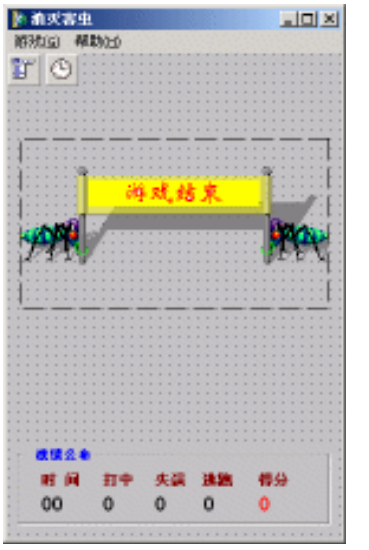


图　70-3

其他控件的属性设置比较简单，这里不再介绍。

6．单击 Tool 菜单下的 Image Editor 菜单项，将弹出图像编辑器，在其中绘制一个活的害虫和一个死的害虫的 bmp 图片，然后再绘制两个锤子的图片，如图 70-4 所示，最后将文件保存为 Extrares.res。

7．单击 File→New→Form 菜单，新建一个窗体，将其命名为 OptionsDlg，向其中添加控件，该窗体用来设置游戏的相关属性，如图 70-5 所示。

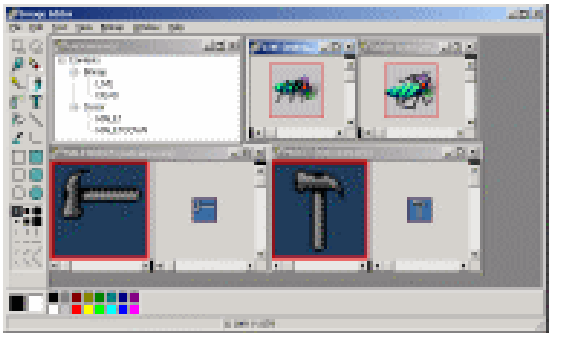


图　70-4

图　70-5

8．再新建一个窗体，将其命名为 Help，该窗体用来说明游戏规则。如图 70-6 所示。

9．创建第四个窗体，将其命名为 About，该窗体用来说明版本信息，如图 70-7 所示。

图　70-6

图　70-7

10．在 main 单元的 publi 中声明三个全局变量，用来控制游戏的时间，害虫出现频率和害虫出现的数量。

```
public
    { Public declarations }
    LiveTime,    Frequence, GameTime : integer;
  end;
```

11．在 private 中定义全局私有变量和 WriteScore 过程，用来记录得分情况，打中、失误、逃跑情况，游戏的结束暂停的状态以及两个数组分别记录虫子出现的位置和信息。WriteScore 过程把成绩信息记录在状态栏中，其定义代码如下：

```
    Score : integer;
    Hits, Miss, Escaped : integer;
    IsGameOver, IsPause : Boolean;
    Live : TBitmap;
    Dead : TBitmap;
    HoleInfo : array[0..4] of THole;
    Holes : array[0..4] of TPoint;
    procedure WriteScore;
```

12．游戏中还用到了 extrares.res 资源文件。它包含了害虫 live 和 dead 两种状态的图片以及锤子在 up 和 down 两种状态，在程序中直接调用，可以提高程序的运行时间。在 main 单元的编译开关中加入如下代码：

```
{$R *.dfm}
{$R extrares.res}
```

实例 71　消灭害虫组件

实例说明

续前面的实例。如图 71-1 所示。

本例通过 Timer 控件来控制害虫的显示，其具体实现方法在编程思路中有详细介绍。同时还实现了程序的初始化处理，在程序的 OnCreate 事件处理函数中，初始化了各个害虫洞的位置，设定了光标的形状为一个小锤子，然后种下了随机种子，以便害虫随机地在不同位置上出现。

本实例还完成了与玩家交互的部分，也就是响应窗体的 FormMouseDown 事件，判断玩家单击结果，是打中还是失误。

图 71-1　效果图

编程思路

游戏的进行通过 Timer 控件来控制，这段代码是整个程序的核心部分。首先在 Holes 数组定义了 5 个位置，随即出现 live 状态的虫子，并判断它们是否在存活时间内，如果在超出存活时没有打中，虫子将自动消失（HoleInfo[i].Time 置为 0），并扣除一定的分数。如果时钟的 tag 属性值超过规定的时间，调用 Stop1Click 事件结束游戏，并根据得分多少，进行鼓励。

创作步骤

续前面的实例。

1．SwatForm 窗体的初始化，包括定义虫子出现的位置，调入锤子 up 和 down 两种状态，调入 live 和 dead 两种状态的图片，定义游戏的难度的默认值以及在游戏最小化后恢复窗口的运行状态。代码如下：

```
procedure TSwatForm.FormCreate(Sender: TObject);
begin
  Holes[0] := Point( 10, 10 );
  Holes[1] := Point( 200, 10 );
  Holes[2] := Point( 100, 100 );
  Holes[3] := Point( 10, 200 );
  Holes[4] := Point( 200, 200 );
  Screen.Cursors[crMaletUp] := LoadCursor(HInstance, 'Malet');
```

```
    Screen.Cursors[crMaletDown] := LoadCursor(HInstance, 'MaletDown');
    Screen.Cursor := TCursor(crMaletUp);
    randomize;
    Live := TBitmap.Create;
    Live.LoadFromResourceName(HInstance, 'Live');
    Dead := TBitmap.Create;
    Dead.LoadFromResourceName(HInstance, 'Dead');
    IsGameOver := true;
    IsPause := false;
    LiveTime := 10;
    Frequence := 20;
    GameTime := 150;              // 15s
    Application.OnMinimize := Pause1Click;
    Application.OnRestore := Pause1Click;
end;
```

2．Timer **控件的** OnTimer **事件处理函数代码如下：**

```
procedure TSwatForm.Timer1Timer(Sender: TObject);
var
   i : integer;
begin
   Timer1.Tag := Timer1.Tag + 1;
   i := random(Frequence);
   if (i < 5) then
   begin
      if (HoleInfo[i].Time = 0) then
      begin
         HoleInfo[i].Time := Timer1.Tag + LiveTime;
         HoleInfo[i].Dead := false;
         Canvas.Draw(Holes[i].x, Holes[i].y, Live);
      end;
   end;
   for i := 0 to 4 do
   begin
      if ( (Timer1.Tag > HoleInfo[i].Time ) and ( HoleInfo[i].Time <> 0 ) ) then
      ……（此处代码略，详见光盘）

         application.MessageBox('继续努力,别放弃！','加油啊',MB_OK)
      else if score<70 then
           application.MessageBox('不错了，再加把劲！','及格了',MB_OK)
```

```
        else if score<80 then
            application.MessageBox('再接再厉'+#13+'向顶级冲锋！','恭喜',MB_OK)
        else
            application.MessageBox('祝贺你！'+#13+'**大牛人**','牛不牛',MB_OK);
      end;
```

3．设定窗体的 FormMouseDown 事件，这个过程比较简单。Main 窗体发生此事件时，就判断是否打中虫子，如果打中，则把 HoleInfo 的 dead 属性设置为 true，并增加打中次数和得分。反之则增加失误次数和扣分。程序代码如下：

```
procedure TSwatForm.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
var
  i : integer;
  hit : boolean;
begin
  Screen.Cursor := TCursor(crMaletDown);
  if (IsGameOver or IsPause) then
    exit;
  hit := false;
  for i := 0 to 4 do
    if ( (not HoleInfo[i].Dead) and (HoleInfo[i].Time <> 0) ) then
      if (X > Holes[i].x ) and ( X < (Holes[i].x + Live.Width) ) and
          ( Y > Holes[i].y ) and ( Y < (Holes[i].y + Live.Height)) then
      begin
        inc( Score, HitPoints );
        HoleInfo[i].Dead := true;
        HoleInfo[i].Time := Timer1.Tag + 2 * LiveTime;
        inc( Hits );
        hit := true;
        Canvas.Draw(Holes[i].x, Holes[i].y, Dead);
      end;
  if not(hit) then
  begin
    inc ( Score, MissedPoints );
    inc( Miss );
  end;
  WriteScore;
end;
```

4．New1Click 事件，打开时钟记时，并把得分记录都设置为 0 值。Boolean 变量 IsPause 设为 false。把 Pause1 的 caption 属性置为“暂停”。同时进行一些运行前的初始化工作，下面是程序代码：

```
procedure TSwatForm.New1Click(Sender: TObject);
begin
   Timer1.Enabled := true;
   Timer1.Tag := 0;
   Score := 0;
   Hits := 0;
   Miss := 0;
   Escaped := 0;
   if (IsPause)
   then begin
      IsPause := false;
      Pause1.Caption := '暂停(&P)';
   end;
   GameOverImage.Visible := false;
   IsGameOver := false;
   FillChar(HoleInfo, sizeof(HoleInfo), 0);
   New1.Enabled := false;
   Options1.Enabled := false;
   Stop1.Enabled := true;
 end;
```

本例其他源代码详见光盘。

实例 72　射　　击

实例说明

图 72-1　效果图

本例编写射击游戏。效果如图 72-1 所示。

本例程序运行后将依次产生 30 个小精灵，每击中一个得分将加 100，并在开枪时，同时有枪响效果。

本例通过 VB 的图像、声音和鼠标光标处理等知识制作完成。

编程思路

在游戏设计中，最重要的是声音、鼠标光标及动画的处理。通常，处理声音、鼠标光标及动画需要调用 Windows API 函数及动态链接库（DLL），对于鼠标光标的处理，VB 也提供一些方法，所以本例中不用 Windows API 函数，仅在播放声音时使用 Windows API 函数。当然，要想实现非常有个性化的鼠标光标，则必须调用 Windows API 函数。

创作步骤

1．**启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程。**

2．**在 Form1 中创建两个命令按钮（Command）、三个标签（Label）、一个定时装置（Timer）、一个图像控件（Image）。然后，把 Command1 的 Caption 项设为**”Continue”，**把 Command2 的 Caption 项设为**”End”，**如图 72-2 所示。**

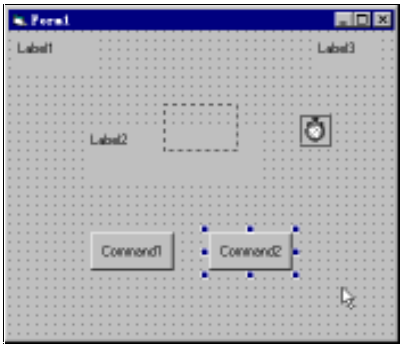


图　72-2

3．设置各控件属性，各控件主要属性如下：

Form1：

Name = Form1

Picture = “d:\ldl\g301.bmp” ‘读者可以自己找一个自己喜欢的图片。

Timer1：

Interval = 1

Image1：

Picture=”C:\program files\microsoft visual studio\common\graphics\icon\traffic\trffc16.icon”

整个窗体如图 72-3 所示。

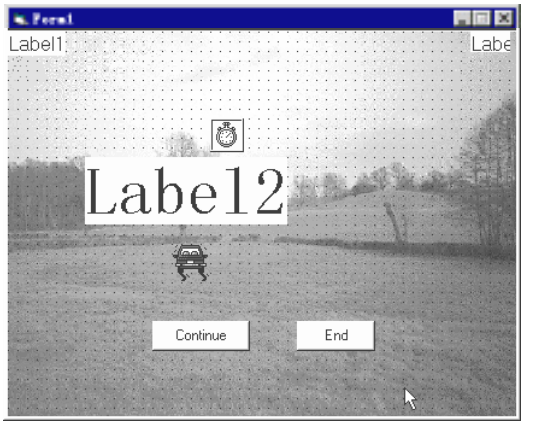


图 72-3

4．完成属性设置之后，接下来就可以编写代码了。一个“精灵”被击中或离开窗口区域，另一个“精灵”将出现，但要求出现的位置是随机的。另外，在程序中处理声音要用到 Windows API 函数。

完整源代码详见光盘。

实例 73　打　地　鼠

实例说明

图 73-1　效果图

本例编写打地鼠的游戏，如图 73-1 所示。

程序运行后，鼠标在窗口中变为小锤子形状，地鼠在九个地洞中随机出现，玩家可以用小锤子打地鼠。一个回合地鼠出现 100 次，一回合结束后，显示玩家的成绩。如果玩家把 100 次全部击中，则显示出祝贺信息。

编程思路

本实例使用 Visual Basic 6.0 来实现。程序利用一个 Timer 控件产生随机数，在不同的位置显示地鼠，同时响应鼠标的 MouseMove 事件，不断地移动锤子对象；响应 MouseDown 事件，判断玩家是否击中了地鼠。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加六个 Image 控件、两个 Label 控件和一个 Timer 控件。如图 73-2 所示，其中 Image1～Image6 中的图片在图 73-1 中依次排列。

图　73-2

2．设置 Timer1 控件的 Interval 属性为 700。

3．向窗体中添加一个菜单项，名称为“游戏规则”。

4．本实例的源代码如下：

```
Dim 鼠 X As Single
Dim 鼠 Y As Single
Dim 地鼠 As Byte
Dim 对数 As Integer
Dim 总数 As Integer
 Sub 判断(X, Y)
Select Case 地鼠
 Case 0, 1, 2
 X = 地鼠
 Y = 0
 Case 3, 4, 5
 X = 地鼠 - 3
 Y = 1
 Case 6, 7, 8
 X = 地鼠 - 6
 Y = 2
 End Select
End Sub
Private Sub Form_Load()
初始化随机种子，绘制九个地鼠洞
 Randomize
Show
Print
 For 地鼠 = 0 To 8
    判断 X, Y
Form1.PaintPicture Image5.Picture, 1000 + 1050 * X, 1000 + 1050 * Y
Next
总数 = 1
 End Sub
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
鼠标在窗口中移动时，激活 Timer 控件
Image1.Move X - Image1.Width / 2, Y - Image1.Height / 2
鼠 X = Image1.Left
鼠 Y = Image1.Top
Timer1.Enabled = True
Label2.Caption = "运行中..."
End Sub
```

```
Private Sub Image1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
按下鼠标时，判断是否击中了地鼠
Image1.Picture = Image3.Picture
    xy = 99
    If 鼠 X > 800 And 鼠 X < 1850 Then
        If 鼠 Y + 1500 > 1600 And 鼠 Y + 1500 < 2350 Then xy = 0
        If 鼠 Y + 1500 > 2650 And 鼠 Y + 1500 < 3400 Then xy = 3
        If 鼠 Y + 1500 > 3700 And 鼠 Y + 1500 < 4450 Then xy = 6
  End If
 If 鼠 X > 1850 And 鼠 X < 2900 Then
        If 鼠 Y + 1500 > 1600 And 鼠 Y + 1500 < 2350 Then xy = 1
        If 鼠 Y + 1500 > 2650 And 鼠 Y + 1500 < 3400 Then xy = 4
        If 鼠 Y + 1500 > 3700 And 鼠 Y + 1500 < 4450 Then xy = 7
 End If
 If 鼠 X > 2900 And 鼠 X < 3950 Then
        If 鼠 Y + 1500 > 1600 And 鼠 Y + 1500 < 2350 Then xy = 2
        If 鼠 Y + 1500 > 2650 And 鼠 Y + 1500 < 3400 Then xy = 5
        If 鼠 Y + 1500 > 3700 And 鼠 Y + 1500 < 4450 Then xy = 8
 End If
 If 地鼠 = xy Then
 对数 = 对数 + 1
 Label1.Caption = "打中次数 :" & 对数
 判断 X, Y
 Form1.PaintPicture Image6.Picture, 1000 + 1050 * X, 1000 + 1050 * Y
End If
End Sub
Private Sub Image1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
使锤子和鼠标同时移动
Image1.Move 鼠 X - (Image1.Width / 2 - X), 鼠 Y - (Image1.Height / 2 - Y)
鼠 X = Image1.Left
鼠 Y = Image1.Top
End Sub

Private Sub Image1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
使锤子处于下落状态
Image1.Picture = Image2.Picture
End Sub
Private Sub Timer1_Timer()
```

```
在不同的位置出现地鼠，并且记录玩家的得分
总数 = 总数 + 1
  判断 X, Y
Form1.PaintPicture Image5.Picture, 1000 + 1050 * X, 1000 + 1050 * Y
地鼠 = Int(Rnd * 9)
   判断 X, Y
Form1.PaintPicture Image4.Picture, 1000 + 1050 * X, 1000 + 1050 * Y
     If 总数 > 100 Then
        Timer1.Enabled = False
        判断 X, Y
        Form1.PaintPicture Image5.Picture, 1000 + 1050 * X, 1000 + 1050 * Y
    If 对数 > 100 Then
    msg = MsgBox("恭喜" + Chr(13) + "你已经爆机了，还要继续吗？", vbYesNo, "提示")
    Else
    msg = MsgBox("你已经过关了，您打中了" & 对数 & "次，还要继续吗?", vbYesNo, "提示")
    End If
    If msg = vbYes Then
     总数 = 1
     对数 = 0
     Timer1.Enabled = True
     Else
     Unload Me
    End If
End If
End Sub
Private Sub 游戏规则_Click()
MsgBox "游戏规则" + Chr(13) + "使用说明:" + Chr(13) + "鼠标左右键都可打击" _
+ "想要能更快的打到地鼠连续打击左右键，" + Chr(13) + "那能加快打地鼠的次数", , "地鼠娱乐部"
End Sub
```

实例 74　打 字 练 习

实例说明

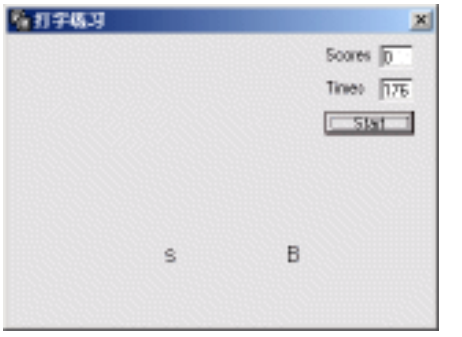


图 74-1　效果图

本例编写打字的游戏，如图 74-1 所示。

程序运行后，单击 Start 按钮，将开始游戏。此时窗口中将会随机地出现各种字母，包括大、小写字母、数字以及标点、特殊字母等，在一个字母下落的过程中，玩家可以按下键盘上的相应按键，如果正确，则在图中的 Scores 栏后面加分。一共要练习 200 个字符。

编程思路

本实例使用 Visual Basic 6.0 来实现。使用两个 Label 控件（Label1 和 Label6 控件）来显示随机生成的字母，并且利用 Timer1 来控制这两个 Label 控件的下落速度，使用 Timer2 控件来记录玩家的成绩，最后在 Label4 控件中显示玩家的成绩就可以了。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加两个 Timer 控件、六个 Label 控件、一个 Command 控件和一个 Frame 控件。

2．设置两个 Timer 控件的关键属性如下：

Timer1

　　Interval =　100

Timer2

　　Interval = 1000

3．设置 Label1 和 Label6 的 Caption 属性为空，放在 Frame 控件上。

4．Label4 用来显示玩家的得分，其属性如下：

```
BackColor = &H80000009&
BorderStyle = 1    'Fixed Single
Caption = "0"
Height = 255
Left = 4080
TabIndex = 2
Top = 120
Width = 375
```

5．Label5 控件用来显示剩余的字符数，其属性如下：

```
BackColor = &H80000009&
BorderStyle = 1    'Fixed Single
```

```
    Caption = "200"
    Height = 255
    Left = 4080
    TabIndex = 3
    Top = 480
    Width = 375
```

6．本实例的源代码如下：

```
Option Explicit
Dim score As Integer '记录分数
Dim speed As Integer '记录字母下落的速度

Sub init() '初始化程序（控制 Label1 控件）
Label1.Caption = Chr(Int(Rnd * 26) + 49) '使 Label1 控件上随机显示字母
speed = Int(Rnd * 100 + 100) '生成随机的速度
'设定 Label1 控件的位置
Label1.Left = Int(Rnd * Frame1.Width)
Label1.Top = Frame1.Top
End Sub

Sub init1() '初始化程序（控制 Label6 控件）
Label6.Caption = Chr(Int(Rnd * 26) + 97)
speed = Int(Rnd * 100 + 100)
Label6.Left = Int(Rnd * Frame1.Width)
Label6.Top = Frame1.Top
End Sub

Private Sub Command1_Click() '开始游戏，激活 Timer1 和 Timer2
init
Timer1.Enabled = True
Timer2.Enabled = True
Label5.Caption = 200
Label4.Caption = 0
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
'当用户按键时，判断输入的是否为落下的字母
If Chr(KeyAscii) = Label1.Caption Then
  init
  score = score + 1
  Label4.Caption = score
End If
If Chr(KeyAscii) = Label6.Caption Then
  init1
  score = score + 1
  Label4.Caption = score
```

```
End If
End Sub

Private Sub Form_Load()
程序启动时，使 Timer 组件处于不激活状态
Randomize
Timer1.Enabled = False
Timer2.Enabled = False
End Sub

Private Sub Timer1_Timer() 移动 Label1 和 Label6 控件
Label1.Top = Label1.Top + speed
If Label1.Top > Frame1.Height Then
  init
End If
Label6.Top = Label6.Top + speed
If Label6.Top > Frame1.Height Then
  init1
End If
End Sub

Private Sub Timer2_Timer()
记录成绩
Label5.Caption = Val(Label5.Caption) - 1
If Val(Label5.Caption) <= 0 Then
  Timer1.Enabled = False
  Label1.Caption = ""
  Label6.Caption = ""
  Select Case score
    Case Is <= 80
      MsgBox vbCrLf + "Don't give up,try again!"
    Case Is < 200
      MsgBox vbCrLf + "That's right. Come on!"
    Case Is < 350
      MsgBox vbCrLf + "Continue and you will be top gun!"
    Case Is > 350
      MsgBox vbCrLf + "Congraduation! You have been a top gun!"
  End Select
  Command1.Visible = True
  Label4.Caption = 0
  Label5.Caption = 200
  Timer1.Enabled = False
  Timer2.Enabled = False
End If
End Sub
```

实例 75　空间大战人机交互

实例说明

本例编写空间大战游戏，如图 75-1 所示。

程序运行后，窗口中有一个飞船和许多小行星，小行星随机地从窗口的右方出现，并且延随机的路径运动。玩家可以使用“↑”、“↓”、“←”、“→”键控制飞船，还可以按下空格键（Space）进行射击。在窗口的左上方显示玩家的成绩，右上方显示飞船所剩的生命力。

这是使用 DirectX 进行编程的实例，其中使用了很多常用的技术。

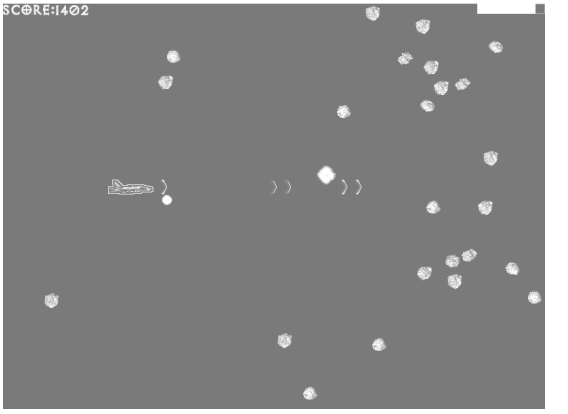


图 75-1　效果图

编程思路

本实例定义了多个 Timer 控件的 Interval 属性，通过 Timer 控件来产生小行星，以及记录玩家的成绩等。同时还定义了多个变量，每个变量的含义都有详细说明。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加六个 Timer 控件，如图 75-2 所示。

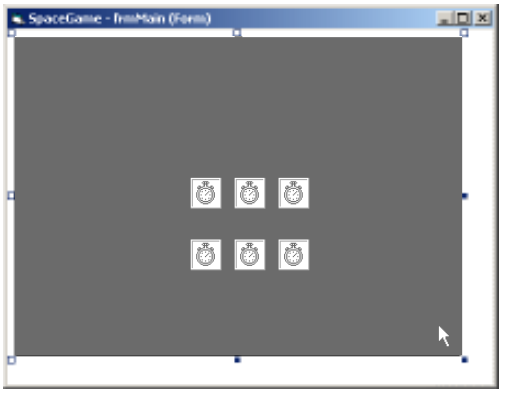


图　75-2

2．设置各 Timer 控件的关键属性如下：

```
Begin VB.Timer VolShow
   Enabled = 0      'False
   Interval = 1500
End
Begin VB.Timer VolChange
   Enabled = 0      'False
   Interval = 1
End
Begin VB.Timer HealthRegen
   Interval = 1000
End
Begin VB.Timer Timer3
   Enabled = 0      'False
   Interval = 200
End
Begin VB.Timer Timer2
   Enabled = 0      'False
   Interval = 200
End
Begin VB.Timer Timer1
   Interval = 100
End
```

3．本实例定义了多个变量和常量，它们的定义和含义如下：

```
'判断键盘是被按下还是没有被按下
Private Declare Function GetAsyncKeyState Lib "user32" (ByVal vKey As Long) As Integer
'判断用户键盘的输入
Dim DI As DirectInput
Dim diDevice As DirectInputDevice
'产生音乐缓存
Dim DS As DirectSound
Dim Perform As DirectMusicPerformance
Dim Loader As DirectMusicLoader
Dim Segment As DirectMusicSegment
Dim SegmentState As DirectMusicSegmentState
Dim VolumeCan As Boolean
'保存精灵的基本信息
Private Type Sprite
'保存当前的 x 坐标
    X As Long
```

保存当前的 y 坐标

    Y As Long

保存当前的 x 方向的移动坐标

    Xp As Long

保存当前的 x 方向的移动坐标

    Yp As Long

保存颜色

    Color As Long

保存当前的状态

    Phase As Long

判断精灵是否激活

    Active As Boolean

保存当前精灵有多少点生命

    Worth As Long

保存它有多少杀伤力

    Damage As Long

记录每个状态有多少帧

    Spin As Long

End Type

设置图片的大小

Dim Vol As DirectDrawSurface7

Const VOLUMEWIDTH = 5

Const VOLUMEHEIGHT = 21

Const VOLUMEPHASES = 0

Const VOLUMECHANGE = 1

Const MAXVOLUME = 1000

Const MINVOLUME = -3000

设置背景音乐

Dim Volume As Integer

每个状态最大的帧数 maximum number of frames between phases

Const MAXSPIN = 5

保存当前的帧号

Dim Frame As Long

保存当前的字体属性

Const FONTWIDTH = 22

Const FONTHEIGHT = 22

Const FONTPHASES = 37

Dim Fonts(FONTPHASES) As Sprite

保存玩家的分数

Dim PlayerScore As Double

```
'保存玩家的生命力
Dim PlayerHealth As Long
'判断武器是否有子弹 determines whether gun has reloaded
Dim CanShoot As Boolean
'判断游戏是否暂停
Dim Paused As Boolean
'等待时间是否超过了允许值
Dim CanUnPause As Boolean
'保存子弹的最大数目
Const NUMBULLETS = 10
'保存子弹帧的高度
Const BULLETHEIGHT = 22
'保存子弹帧的宽度
Const BULLETWIDTH = 10
'保存子弹帧的数目
Const BULLETPHASES = 8
'保存飞船帧的高度
Const SHIPWIDTH = 67
'保存飞船帧的宽度
Const SHIPHEIGHT = 22
'保存飞船的数目
Const SHIPPHASES = 3
'判断子弹从哪里射出来
Const GUNX = 50
Const GUNY = 0
'保存爆炸的信息
Const EXPLOSIONHEIGHT = 35
Const EXPLOSIONWIDTH = 35
Const EXPLOSIONPHASES = 19
'保存小行星的信息
Const ASTEROIDHEIGHT = 25 '75
Const ASTEROIDWIDTH = 25 '75
Const ASTEROIDPHASES = 29
'小行星的最大数量
Const NUMASTEROIDS = 25
'当飞船撞击小行星时，可以播放的最大声音数目
Const NUMSHIPHIT = 5
'保存武器的声音
Dim GunSound(NUMBULLETS - 1) As DirectSoundBuffer
'保存爆炸声音
```

```
Dim ExplosionSound(NUMASTEROIDS - 1) As DirectSoundBuffer
保存飞船的撞击声
Dim ShipHit(NUMSHIPHIT - 1) As DirectSoundBuffer
保存爆炸精灵的信息
Dim Explosions(NUMASTEROIDS - 1) As Sprite
Dim ExplosionsS As DirectDrawSurface7
保存小行星的信息
Dim Asteroids(NUMASTEROIDS - 1) As Sprite
Dim AsteroidsS As DirectDrawSurface7
判断程序何时运行
Dim RUNNING As Boolean
保存子弹信息
Dim Bullets(NUMBULLETS - 1) As Sprite
Dim BulletsS As DirectDrawSurface7
保存飞船位置
Dim ShipX As Long
Dim ShipY As Long
保存飞船当前所在帧
Dim ShipPhase As Long
主体 directx 对象
Dim DX As New DirectX7
主体 directdraw 对象
Dim DD As DirectDraw7
窗口设置
Dim Primary As DirectDrawSurface7
可以绘制的区域
Dim Backbuffer As DirectDrawSurface7
可以保存飞船位置的区域
Dim Ship As DirectDrawSurface7
保存字体的位置
Dim sFont As DirectDrawSurface7
保存表面信息
Dim DDSD1 As DDSURFACEDESC2
Dim DDSD2 As DDSURFACEDESC2
Dim DDSD3 As DDSURFACEDESC2
Dim DDCaps As DDSCAPS2
保存透明颜色
Dim cKey As DDCOLORKEY
保存应用程序的路径
Dim aPath As String
```

实例 76　空间大战组件

实例说明

续前面的实例。如图 76-1 所示。

本例首先初始化了各变量的值以及响应鼠标单击，之后退出游戏。

在程序的 Form_Load 事件处理函数中设置了飞船的移动速度、小行星的移动速度、小行星的输量及背景音乐等信息。

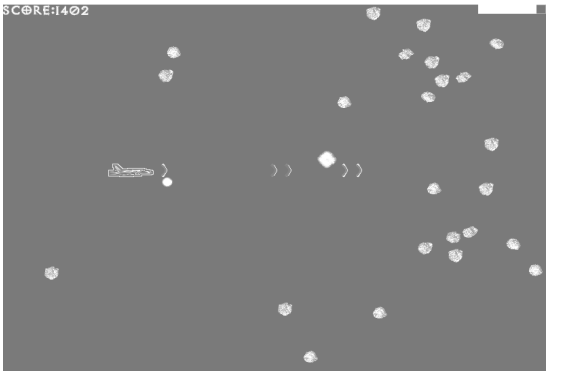


图 76-1　效果图

编程思路

在本实例中开辟了多块内存空间用来保存飞船的状态、小行星的状态、声音的状态以及文字的状态等多种信息。其中用到了多个 DirectX 函数以及 Windows API 函数，这些函数的具体用法响应 MSDN 帮助。

创作步骤

续前面的实例。

1．在程序产生时初始化各变量的值，其代码如下：

```
Private Sub Form_Load()
VolumeCan = True
产生 direct input 对象
Set DI = DX.DirectInputCreate()
产生 device 对象
Set diDevice = DI.CreateDevice("GUID_sysKeyboard")
响应键盘操作
diDevice.SetCommonDataFormat DIFORMAT_KEYBOARD
diDevice.SetCooperativeLevel Me.hWnd, DISCL_BACKGROUND + DISCL_NONEXCLUSIVE
允许读取键盘信息
diDevice.Acquire
声音开到最大
Volume = 100
放到第一帧
Frame = 0
```

设定玩家的生命力

PlayerHealth = 100

设定文字间的空隙

SetFontWidths

设定飞船当前的状态

ShipX = 0

ShipY = 300

设定应用程序的路径

aPath = App.Path

If Right(aPath, 1) <> "\" Then aPath = aPath & "\"

Set DD = DX.DirectDrawCreate("")

Set DS = DX.DirectSoundCreate("")

DS.SetCooperativeLevel Me.hWnd, DSSCL_PRIORITY

使图形绘制在窗体中

DD.SetCooperativeLevel Me.hWnd, DDSCL_ALLOWMODEX + DDSCL_ALLOWREBOOT + DDSCL_FULLSCREEN + DDSCL_EXCLUSIVE

DD.SetDisplayMode 800, 600, 16, 0, DDSDM_DEFAULT

设置基本背景信息

DDSD1.lFlags = DDSD_CAPS + DDSD_BACKBUFFERCOUNT

DDSD1.ddsCaps.lCaps = DDSCAPS_PRIMARYSURFACE + DDSCAPS_FLIP + DDSCAPS_COMPLEX + DDSCAPS_VIDEOMEMORY

设定缓存块数

DDSD1.lBackBufferCount = 1

Set Primary = DD.CreateSurface(DDSD1)

DDCaps.lCaps = DDSCAPS_BACKBUFFER

产生背景缓存

Set Backbuffer = Primary.GetAttachedSurface(DDCaps)

Backbuffer.GetSurfaceDesc DDSD1

设置文字颜色

Backbuffer.SetForeColor RGB(255, 255, 0)

Backbuffer.SetFontBackColor vbWhite

Backbuffer.SetFontTransparency True

DDSD3.lFlags = DDSD_CAPS + DDSD_HEIGHT + DDSD_WIDTH

DDSD3.ddsCaps.lCaps = DDSCAPS_OFFSCREENPLAIN

DDSD3.lWidth = SHIPWIDTH * (SHIPPHASES + 1)

DDSD3.lHeight = SHIPHEIGHT

Set Ship = DD.CreateSurfaceFromFile(aPath & "shipa.bmp", DDSD3)

cKey.high = RGB(255, 255, 0)

cKey.low = RGB(255, 255, 0)

Ship.SetColorKey DDCKEY_SRCBLT, cKey

设定背景颜色缓存

……（此处代码略，详见光盘）

```
ExplosionsS.SetColorKey DDCKEY_SRCBLT, cKey
'设定字体信息
DDSD3.lWidth = FONTWIDTH * (FONTPHASES + 1)
DDSD3.lHeight = FONTHEIGHT
'导入图片
Set sFont = DD.CreateSurfaceFromFile(aPath & "text.bmp", DDSD3)
'将不可视颜色设为白色
sFont.SetColorKey DDCKEY_SRCBLT, cKey
DDSD3.lWidth = VOLUMEWIDTH * (VOLUMEPHASES + 1)
DDSD3.lHeight = VOLUMEHEIGHT
Set Vol = DD.CreateSurfaceFromFile(aPath & "vlm.bmp", DDSD3)
Vol.SetColorKey DDCKEY_SRCBLT, cKey

Dim rSurf As RECT
rSurf.Top = 0
rSurf.Bottom = 600
rSurf.Left = 0
rSurf.Right = 800
RUNNING = True
Dim N As Long
If NUMBULLETS > NUMASTEROIDS Then
N = NUMBULLETS
Else
N = NUMASTEROIDS
End If
'设定背景音乐
Perform.SetMasterVolume 1000
Paused = False
CanUnPause = False
Do
Frame = Frame + 1
If Frame > MAXSPIN Then Frame = 0
'检查按键
ReadKeys
DoEvents
'将背景颜色设为黑色
Backbuffer.BltColorFill rSurf, RGB(0, 0, 0)
'绘制飞船
If PlayerHealth <= 100 Then ShipPhase = 0
```

……（此处代码略，详见光盘）

```
Call Primary.Flip(Nothing, DDFLIP_WAIT)
Loop Until RUNNING = False
Unload Me
End Sub
```

2．响应鼠标单击，退出程序，其代码如下：

```
Private Sub Form_Click()
'关闭程序
RUNNING = False
End Sub
```

3．在程序退出时，释放变量，其代码如下：

```
Private Sub Form_Unload(Cancel As Integer) 释放对象
    DD.RestoreAllSurfaces
    Set DD = Nothing
    Set DX = Nothing
End Sub
```

本例其他源代码详见光盘。

f

第六篇

趣味游戏

本篇总览

顾名思义，趣味游戏就是一些有意思的小游戏，以供人们消遣用。这些游戏有时可能用不到高深的编程知识，但是需要作者的奇思妙想。

本篇制作了桌面晃动、桌面下的奥秘、鼠标放大、隐藏时钟、桌面小精灵、掷骰子、吊小人等趣味游戏。

通过这些游戏可以学习一些编写趣味游戏的经验和技巧，读者也可以把这些小游戏改装一下，做成更有意思的小游戏，也可以把这些游戏和前面的游戏综合起来，制作一个大型游戏。

实例 77　阴 阳 缩 放

实例说明

本例制作阴阳缩放小游戏。效果如图 77-1 所示。

平时当人们吹气球时，气球会随着气体的进入而变得越来越大，这时松开嘴，吹进去的气体就会跑出来，气球也会瘪下去。

本例模拟了气球的缩放，通过 VB 实现不断变化的阴阳缩放效果。

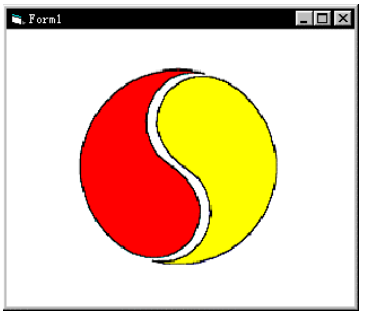


图 77-1　效果图

编程思路

阴阳图的膨胀或缩小是缩放动画的典型实例。在 Timer 事件过程中，修改 Image 对象的 Width 和 Height 属性，便可实现缩放动画。但如果要表现物体的同心缩放，则还应同时移动 Image 对象。下面是模拟阴阳图在缩放（同心）的例子。程序启动时，在窗体中显示一只阴阳图，用鼠标左键点击它，则阴阳图将开始膨胀，如碰到窗体的边界，阴阳图将缩小，缩小到原来大小时，又将膨胀；再用鼠标左键点击阴阳图，则阴阳图将停止缩放。在需显示动画的窗体（Form1）中，设置 Image 对象 Image1 和 Timer 对象 Timer1。

本例中，Move 方法用于移动 MDIForm、Form 或控件。不支持命名参数。其语法为 object.Move left, top, width, height。其中 object 可选，它是一个对象表达式，如果省略 object，带有焦点的窗体默认为 object。语法中的 left 是必需的，它指示 object 左边的水平坐标 (x-轴)。top 是可选的，它指示 object 顶边的垂直坐标 (y-轴)。width 是可选的，它指示 object 新的宽度。height 是可选的，它指示 object 新的高度。

只有语法中的 left 参数是必须的。但是要指定任何其他的参数，必须先指定出现在语法中该参数前面的全部参数。例如，如果不先指定 left 和 top 参数，则无法指定 width 参数。任何没有指定的尾部的参数则保持不变。

移动屏幕上的窗体或移动 Frame 中的控件时总是相对于左上角的原点 (0,0) 为坐标。移动 Form 对象或 PictureBox 中的控件（或 MDIForm 对象中的 MDI 子窗体）时，则使用该容器对象的坐标系统。坐标系统或度量单位是在设计时 用 ScaleMode 属性设置。在运行时 使用 Scale 方法可以更改该坐标系统。

创作步骤

1．启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程。

2．在窗体中添加一个 Image 控件和 Timer 控件。

3．按照表 77-1 所示，设置各个控件的属性。

表　77-1

控　　件	属　　性	设　置　值
Form	Name BackColor	Form1 &H00FFFFFF&
Image	Name Picture Stretch	Image1 C:\samples\yinyang.bmp True
Timer	Name Interval	Timer1 150

设置完后的界面如图 77-2 所示。

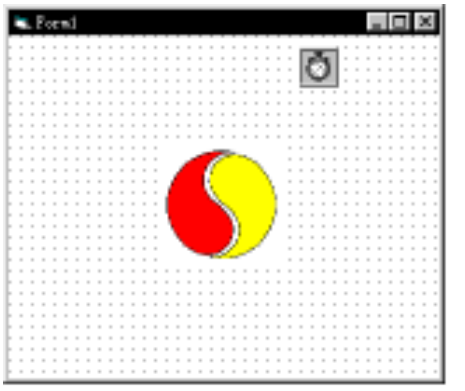


图　77-2

实例 78　叫 床 时 钟

实例说明

本例制作叫床时钟小游戏，效果如图 78-1 所示。

本例通过 VB 中的 Timer 时间控件判断时间的运行过程，一旦到了人为设定的时间时，可自动作出响应，并出现“时间到了，快起床啦！”的提示。

本例通过 VB 编写程序实现。

图 78-1　效果图

编程思路

本例利用 Timer 控件的 Timer 事件，测试系统时间，并显示时间，之后出现提示信息。

每个 Timer 控件都有 Interval 属性，指定定时器事件之间的毫秒数。除非禁止此属性，否则定时器在大致相等的时间间隔不断接受事件（称作定时器事件会更贴切）。

在为 Timer 控件编程时应考虑对 Interval 属性的几条限制：

如果应用程序或其他应用程序正在进行对系统要求很高的操作，例如长循环、高强度的计算或者正在访问驱动器、网络或端口，则应用程序定时器事件的间隔可能比 Interval 属性指定的间隔长。

间隔的取值可在 0~64 767 之间（包括这两个数值），这意味着即使是最长的间隔也不比 1min 长多少（大约 64.8s）。

间隔并不一定十分准确。要保证间隔准确，应在需要时才让定时器检查系统时钟，而不在内部追踪积聚的时间。系统每秒生成 18 个时钟信号，所以使用 ms 衡量 Interval 属性，间隔实际的精确度不会超过 1/18s。

每个 Timer 控件必须要与窗体关联。因此要创建定时器应用程序就必须至少创建一个窗体（如果不需要窗体完成其他操作就不必使窗体可见）。

创作步骤

1．启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程。

2．往窗体中放入两个 Text 控件、一个 CommandButton 控件、一个 Line 控件、一个 Timer 控件。

3．各个控件的属性，设置如表 78-1 所示。设置完后的界面如图 78-2 所示。

表　78-1

控　　件	属　　性	值
Text	Name Visible	TimeTxt False
Text	Name Visible	FirstTxt False
CommandButton	Name Caption	ComSet **定时**

4．**单击“工程”菜单下的“添加窗体”子菜单，弹出添加窗体对话框，选择窗体并按下“打开”按钮，如图** 78-3 **所示。**

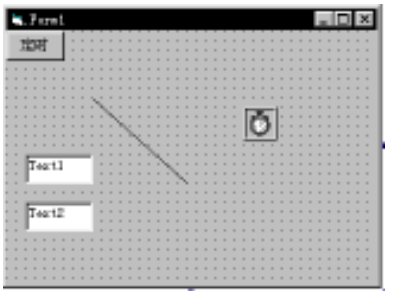


图　78-2

图　78-3

5．**向新的窗体中添加一个** Text **控件、两个** CommandButton **控件。**

6．**按照表** 78-2 **所示设置各个控件的属性。**

表　78-2

控　　件	属　　性	值
Form	Name Caption BackColor	Form1 **定时** &H000000FF&
Text	Name Text	Text1 **空**
CommandButton	Name Caption	CmdOk **确定**
CommandButton	Name Caption	CmdQuit **退出**

设置完后的界面如图 78-4 **所示。**

图　78-4

实例 79　五彩同心圆

实例说明

本例制作五彩同心圆绘图小游戏。效果如图 79-1 所示。

本例运行后，出现五彩缤纷的漂亮图案。不仅如此，还可以产生颜色艳丽的方形、凌形等图案。

本例通过 VB 的绘图程序来实现。

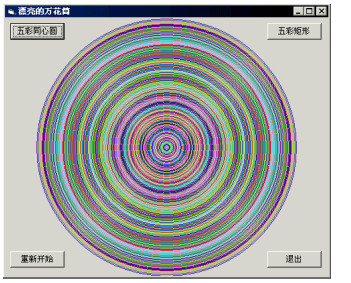


图 79-1　效果图

编程思路

本例中，将要告诉大家如何画出五彩的漂亮的同心圆五彩的矩形以及二者的叠加。窗体本身具有很多方便实用的方法供大家调用，比如 line()，用来画两点之间的直线，比如 circle()，可以画圆和椭圆。

本例主要介绍两个方法：line 、circle。

Line 方法的作用是在对象上画直线和矩形。

语法：object.Line [Step] (x1, 1) [Step] (x2, y2), [color], [B][F]

Circle 方法的作用是在对象上画圆、椭圆或弧。

语法：object.Circle [Step] (x, y), radius, [color, start, end, aspect]

创作步骤

1．启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程，新建一个窗体，如图 79-2 所示。

2．设置窗体的一些属性，把窗体的 caption 设为“漂亮的万花筒”，如图 79-3 所示。

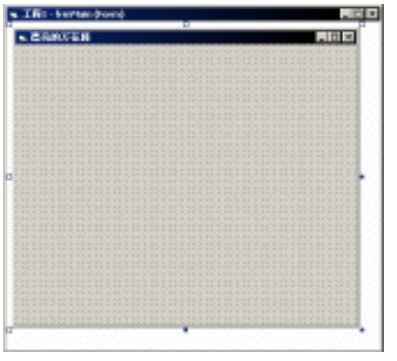


图　79-2

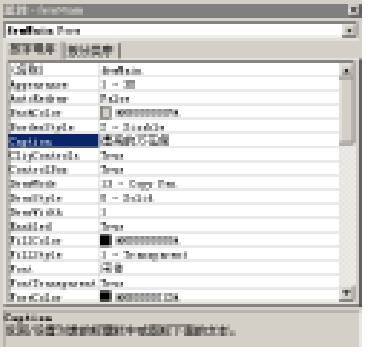


图　79-3

3．添加“命令”按钮，如图 79-4 所示。

4．设置按钮的属性，如图 79-5 所示。

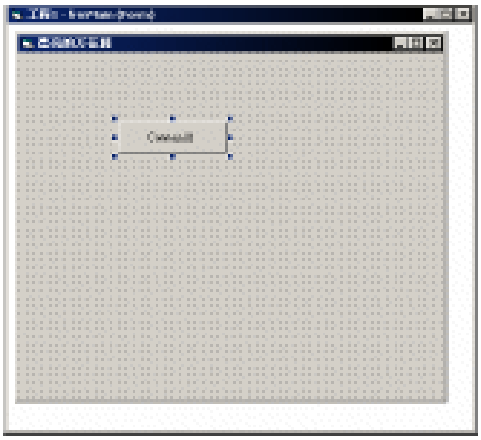


图　79-4

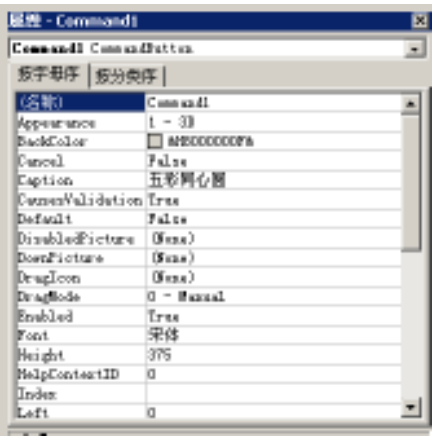


图　79-5

5．添加“命令”按钮，并设置属性，如图 79-6 所示。

图　79-6

6．在“五彩同心圆”按钮的 click 事件中写如下代码：

```
Private Sub Command1_Click()
    Dim CX, CY, Radius, Limit     'Declare variable.
    ScaleMode = 3     '以像素为单位。
    CX = ScaleWidth / 2     'X 位置。
    CY = ScaleHeight / 2     'Y 位置。
    If CX > CY Then Limit = CY Else Limit = CX
    For Radius = 0 To Limit     '半径。
       Circle (CX, CY), Radius, RGB(Rnd * 255, Rnd * 255, Rnd * 255)
    Next Radius
End Sub
```

7．**在“五彩矩形”按钮的** click **事件中写如下代码：**

```
Private Sub Command2_Click()
    Dim CX, CY, F, F1, F2, I     '声明变量。
    ScaleMode = 3     '设置 ScaleMode 为像素。
    CX = ScaleWidth / 2     '水平中点。
    CY = ScaleHeight / 2     '垂直中点。
    DrawWidth = 8     '设置 DrawWidth。
    For I = 50 To 0 Step -2
        F = I / 50     '执行中间步骤。
        F1 = 1 - F: F2 = 1 + F     '计算。
        ForeColor = QBColor(I Mod 15)     '设置前景颜色。
        Line (CX * F1, CY * F1)-(CX * F2, CY * F2), , BF
    Next I
    DoEvents     '做其他处理。
    If CY > CX Then     '设置 DrawWidth。
        DrawWidth = ScaleWidth / 25
    Else
        DrawWidth = ScaleHeight / 25
    End If
    For I = 0 To 50 Step 2     'Set up loop.
        F = I / 50     '执行中间。
        F1 = 1 - F: F2 = 1 + F     '计算。
        Line (CX * F1, CY)-(CX, CY * F1)     '画左上角。
        Line -(CX * F2, CY)     '画右上角。
        Line -(CX, CY * F2)     '画右下角。
        Line -(CX * F1, CY)     '画左下角。
        ForeColor = QBColor(I Mod 15)     '每次改变颜色。
    Next I
    DoEvents     '进行其他处理。
End Sub
```

8．**在“重新开始”按钮的** click **事件中写入如下代码：**

```
Private Sub Command3_Click()
    Unload Me 卸载当前窗体
    Load frmMain     重新载入窗体
    frmMain.Show
End Sub
```

9．**在“退出”命令按钮的** click **事件中写入如下代码：**

```
Private Sub Command4_Click()
  Unload Me    卸载窗体
End Sub
```

至此，程序编写完成，完整代码详见光盘。

实例 80　桌 面 晃 动

实例说明

本例编写桌面晃动的小游戏。画面如图 80-1 所示。

程序运行之后可以将当前的窗口分成 12 部分（4×3），然后以黑色作为背景，这 12 部分都在窗口中随机地移动。这个游戏可以放到 Windows98/2000 中的“开始”→“程序”→“启动”中，那么当 Windows 启动后就会出现桌面晃动的现象。

图 80-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。程序运行后，得到当前桌面的句柄，然后将桌面的内容拷贝到桌面位图对象，接下来将窗口图像分成 12 部分，随机地移动。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 PopupMenu 控件。

2．设置窗体的关键属性如下：

BorderStyle = bsNone
Caption = 'Form1'
ClientHeight = 453
ClientWidth = 688
Color = clBtnFace
PopupMenu = PopupMenu1
WindowState = wsMaximized

3．设置 PopupMenu 控件的菜单项，如图 80-2 所示。

图　80-2

4．本实例的源代码如下：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Menus;
type
  TForm1 = class(TForm)
    PopupMenu1: TPopupMenu;
    MenuExit: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure MenuExitClick(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  private
    rectScreen: TRect;        // 用来保存当前整个桌面的矩形对象,其值为(0,0,Screen.Width,
Screen.Height)
    rectBlobSrc,rectBlobChanged: array [0..11] of TRect;        // 保存桌面划分的矩形以及它们被移动
以后的矩形
    ptDir: array [0..11] of TPoint;        // 用来记录当前的移动变量,+表示右(下)移,-表示左(上)移
    CanvasBg,CanvasWork: TCanvas;        // 背景画布以及工作画布
    procedure IdleLoop(Sender: TObject;var DoneState:boolean);    // IDLE 事件的处理函数
  public
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
var
  ScreenCanvas: TCanvas;        // 保存当前整个桌面的画布对象
  ScreenBitmap: TBitmap;        // 保存当前整个桌面的位图对象
  dc: HDC;              // dc 是当前桌面的设备环境
  bmpWork,bmpBg: HBITMAP;        // 用来指定当前的工作及背景画布使用的 BITMAP 对象句柄
  dcWork,dcBg: HDC;        // 产生的工作画布和背景画布的设备环境
  nBlobWidth,nBlobHeight,cx,cy: integer;    // 记录每小块的宽高,cx,cy 用来表示在循环中使用的 x 方
                                            向和 y 方向的递增变量
begin
  Sleep(300);    // 暂停 300s,为了保证在开始拷贝桌面以前,自己的窗口不出现,避免挡住桌面
```

```
rectScreen := Rect(0,0,Screen.Width,Screen.Height);
nBlobWidth := Screen.Width div 4;
nBlobHeight := Screen.Height div 3;       // 把整个桌面 12 等分
for cx := 0 to 3 do
   for cy := 0 to 2 do
   begin
      rectBlobSrc[cy*3+cx]  :=  RECT(nBlobWidth*cx,nBlobHeight*cy,nBlob  Width*(cx+1),  nBlob
      Height*(cy+1));
      rectBlobChanged[cy*3+cx]  :=  RECT(nBlobWidth*cx,nBlobHeight*cy,nBlobWidth*(cx+1),nBlob
      Height*(cy+1));
   end;   // 初始化桌面划分的矩形以及移动后的矩形为初始位置
Randomize;
for cx := 0 to 11 do
begin
   ptDir[cx] := POINT(Random(9),Random(9));   // 先对每一个点产生一个随机的移动方向
   if rectBlobChanged[cx].Left <= 0 then// 该块向左出边界
      ptDir[cx].X := Random(9)
   else if rectBlobChanged[cx].Right >= Screen.Width then   // 该块向右出边界
      ptDir[cx].X := -Random(9);        // 产生向右的随机方向增量
   if rectBlobChanged[cx].Top <= 0 then
      ptDir[cx].Y := Random(9)
   else if rectBlobChanged[cx].Bottom >= Screen.Height then
      ptDir[cx].Y := -Random(9);
end;
ScreenCanvas := TCanvas.Create;
CanvasWork := TCanvas.Create;
CanvasBg := TCanvas.Create;
ScreenBitmap := TBitmap.Create;
ScreenBitmap.Width := Screen.Width;
ScreenBitmap.Height := Screen.Height;      // 初始化桌面位图对象
dc := GetDC(0);      // 以下 3 行代码把桌面的内容拷贝到桌面位图对象 ScreenBitmap 中
ScreenCanvas.Handle := dc;
ScreenBitmap.Canvas.CopyRect(Rect(0,0,Screen.Width,Screen.Height),ScreenCanvas,Rect(0,0,Screen.W
idth,Screen.Height));
dcWork := CreateCompatibleDC(Canvas.Handle);
bmpWork := CreateCompatibleBitmap(Canvas.Handle,Screen.Width,Screen.Height);
SelectObject(dcWork,bmpWork);
SelectPalette(dcWork,ScreenBitmap.Palette,false);
CanvasWork.Handle := dcWork;        // 初始化 CanvasWork 的设备环境以及调色板
```

```
    CanvasWork.Draw(0,0,ScreenBitmap);      // 绘制底图
    dcBg := CreateCompatibleDC(Canvas.Handle);
    bmpBg := CreateCompatibleBitmap(Canvas.Handle,Screen.Width,Screen.Height);
    SelectObject(dcBg,bmpBg);
    SelectPalette(dcBg,ScreenBitmap.Palette,false);
    CanvasBg.Handle := dcBg;        // 初始化 CanvasBg 的设备环境以及调色板
    RealizePalette(CanvasWork.Handle);
    CanvasBg.Brush.Style := bsSolid;
    CanvasBg.Brush.Color := clBlack;
    CanvasBg.FillRect(rectScreen);          // 把背景填成黑色
    CanvasBg.CopyRect(rectScreen,CanvasWork,rectScreen);    // 把工作画布上的相应内容拷贝到背景
画布上
    RealizePalette(CanvasBg.Handle);
    Canvas.CopyRect(rectScreen,CanvasBg,rectScreen);    // 把背景画布上的相应内容拷贝到整个画布
上显示出来
    ReleaseDC(0,dc);
    ScreenBitmap.Free;
    ScreenCanvas.Free;                  // 释放资源
    Application.OnIdle := IdleLoop;
    SystemParametersInfo(SPI_SCREENSAVERRUNNING, 1, nil, 0);        // 屏蔽掉系统键
  end;

  procedure TForm1.FormDestroy(Sender: TObject);
  begin
    SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, nil, 0);        // 打开系统键
    CanvasWork.Free;
    CanvasBg.Free;
  end;
  procedure TForm1.MenuExitClick(Sender: TObject);
  begin
    Close;
  end;
  procedure TForm1.FormPaint(Sender: TObject);
  begin
    RealizePalette(CanvasBg.Handle);
    Canvas.CopyRect(rectScreen,CanvasBg,rectScreen);  // 在该函数中刷新,把工作画布上的内容拷贝到
                                                      当前 Form 的画布上
  end;
  procedure TForm1.IdleLoop(Sender: TObject;var DoneState:boolean);
```

```
var
  cx:integer;
begin
  DoneState := false;
  for cx := 0 to 11 do          // 同 FormCreate 的程序段,功能也一样,当移动出边界的时候,产生一个反
                                向的随机增量
  begin
    if rectBlobChanged[cx].Left <= 0 then
      ptDir[cx].X := Random(9)
    else if rectBlobChanged[cx].Right >= Screen.Width then
      ptDir[cx].X := -Random(9);
    if rectBlobChanged[cx].Top <= 0 then
      ptDir[cx].Y := Random(9)
    else if rectBlobChanged[cx].Bottom >= Screen.Height then
      ptDir[cx].Y := -Random(9);
  end;
  for cx := 0 to 11 do
  begin     // 移动矩形块的坐标
    rectBlobChanged[cx].Left := rectBlobChanged[cx].Left + ptDir[cx].X;
    rectBlobChanged[cx].Right := rectBlobChanged[cx].Right + ptDir[cx].X;
    rectBlobChanged[cx].Top := rectBlobChanged[cx].Top + ptDir[cx].Y;
    rectBlobChanged[cx].Bottom := rectBlobChanged[cx].Bottom + ptDir[cx].Y;
  end;
  CanvasBg.Brush.Style := bsSolid;
  CanvasBg.Brush.Color := clBlack;
  CanvasBg.FillRect(rectScreen);
  RealizePalette(CanvasWork.Handle);
  for cx := 11 downto 0 do          // 从最后一块开始把所有的块都拷贝到背景画布上用来显示
  begin
    CanvasBg.CopyRect(rectBlobChanged[cx],CanvasWork,rectBlobSrc[cx]);
  end;
  RealizePalette(CanvasBg.Handle);
  Canvas.CopyRect(rectScreen,CanvasBg,rectScreen);
end;
end.
```

实例 81　桌面下面的奥秘界面

实例说明

本例编写桌面下面的奥秘小游戏，画面如图 81-1 所示。

程序运行后，将当前窗口作为一个背景，当用户按下鼠标时，光标变成一个橡皮擦的形状，可以擦除背景。如果右键单击鼠标，将弹出快捷菜单，可以选择使用工人或者皮球来帮忙擦除。桌面下面的奥秘到底是什么呢？读者自己去玩玩看吧。

在本实例中实现了程序的界面设计以及对公共函数、变量的说明。

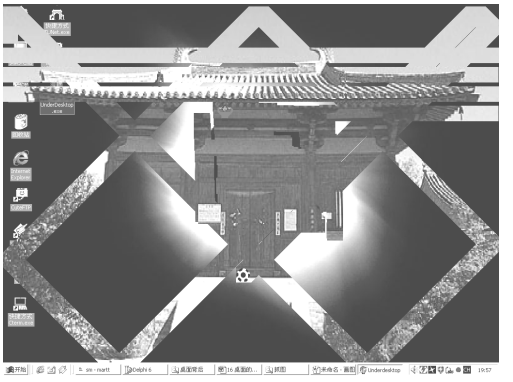


图 81-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。程序中只使用了一个 PopupMenu 控件，使其创建了一个快捷菜单。另外定义了 ScreenBitmap 变量，利用它来保存桌面背后的图片。同时在程序的 OnCreate 事件中获得了当前窗口的句柄，又把当前窗口作为背景显示了出来，同时屏蔽掉系统的功能键（比如 Ctrl+Alt+Del 键结束任务等）。其中调用了一个 Windows API 函数，该函数可以屏蔽系统的功能键，其原型如下：

B00L SystemParametersinfo（UINT uiAction，UINT uiParam，PVOID pvParam，UINT fWinlni）

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 PopupMenu 控件。

2．设置主窗体的关键属性如下：

BorderStyle = bsNone

Caption = 'Form1'

PopupMenu = PopupMenu1

WindowState = wsMaximized

3．设置 PopupMenu 控件快捷菜单如图 81-2 所示：

图 81-2　快捷菜单设计

4．本程序定义了多个私有变量，各变量的含义如下：

ScreenBitmap: TBitmap;　　// 保存当前整个桌面的位图对象

```
    bStartDraw: boolean;    // 定义当前画笔的状态,1 表示开始画,0 表示停止画
    bWorkActive,bBallActive: boolean;      // 用来指示当前是否已经有了工人以及球
    rectView: TRect;                       // 用来保存当前整个桌面的矩形对象,其值为
(0,0,Screen.Width,Screen.Height)
    ptMyBrush, ptWork, ptBall: TPoint;     // 我自己的画刷，工人以及球的最左上角的坐标
    CanvasBg: TCanvas;                     // 背景画布
    picMyBrush,picBg,picWork,picBall:TPicture;     // 分别用来保存我的画笔、背景、工人和球的图片
                                                   对象
    strCurPath:string;                     // 当前可执行文件的路径
    nWorkXDir,nBallXDir,nBallYDir: integer;  // 工人以及球的 X,Y 方向判断,+1 表示向右(下),-1 表示
                                                向左(上)
    rectWork,rectWork2,rectBall,rectBall2,rectBrush,rectBrush2: TRect;   // 用来记录各个元素在移动
                                                                           的前后变化的矩形
```

5．本实例定义的过程如下：

```
    procedure IdleLoop(Sender: TObject;var DoneState:boolean);      // IDLE 事件的处理函数
```

6．在程序的 OnCreate 事件中获得了当前窗口的句柄，又把当前窗口作为背景显示了出来，同时屏蔽掉系统的功能键（比如 Ctrl+Alt+Del 键结束任务等）。该过程的具体实现如下：

```
procedure TForm1.FormCreate(Sender: TObject);
var
  ScreenCanvas: TCanvas;                  // 保存当前整个桌面的画布对象
  dc: HDC;                                // dc 是当前桌面的设备环境
  bmpBg: HBITMAP;                         // 用来指定当前的工作画布使用的 BITMAP 对象句柄
  dcBg: HDC;                              // 用来产生需要使用的画布的设备环境
begin
  strCurPath := ExtractFilePath(Application.ExeName);  // 得到当前可执行文件目录
  bStartDraw := false;
  nBallXDir := 1;
  nBallYDir := 1;
  nWorkXDir := 1;
  bWorkActive := false;
  bBallActive := false;                   // 初始化相应变量,具体说明见声明部分
  Application.OnIdle := IdleLoop;
  Sleep(300);                             // 暂停 300s,为了保证在开始拷贝桌面以前,自己的窗口不出现,避免挡
                                             住桌面
  rectView := Rect(0,0,Screen.Width,Screen.Height);
  ScreenCanvas := TCanvas.Create;
  CanvasBg := TCanvas.Create;
  ScreenBitmap := TBitmap.Create;
  ScreenBitmap.Width := Screen.Width;
```

```
    ScreenBitmap.Height := Screen.Height;              // 初始化桌面位图对象
    picMyBrush := TPicture.Create;
    picMyBrush.Icon.LoadFromFile(strCurPath+'RES\cleaner1.ico');
    picBg := TPicture.Create;
    picBg.Bitmap.LoadFromFile(strCurPath+'RES\bg.bmp');
    picBall := TPicture.Create;
    picBall.Icon.LoadFromFile(strCurPath+'RES\soccer.ico');
    picWork := TPicture.Create;
    picWork.LoadFromFile(strCurPath+'RES\Work.ico');          // 将相应的资源文件读入内存中
    dc := GetDC(0);                    // 以下 3 行代码把桌面的内容拷贝到桌面位图对象 ScreenBitmap 中
    ScreenCanvas.Handle := dc;
    ScreenBitmap.Canvas.CopyRect(Rect(0,0,Screen.Width,Screen.Height),ScreenCanvas,Rect(0,0,Screen.W
                                                                    idth,Screen.Height));
    dcBg := CreateCompatibleDC(Canvas.Handle);
    bmpBg := CreateCompatibleBitmap(Canvas.Handle,Screen.Width,Screen.Height);
    SelectObject(dcBg,bmpBg);
    SelectPalette(dcBg,ScreenBitmap.Palette,false);
    CanvasBg.Handle := dcBg;
    CanvasBg.StretchDraw(rectView,picBg.Bitmap);       // 初始化背景画布对象
    ReleaseDC(0,dc);
    ScreenCanvas.Free;                                 // 释放资源
    SystemParametersInfo(SPI_SCREENSAVERRUNNING, 1, nil, 0);     // 禁止系统键
end;
```

7．在程序的 OnDestory 事件中将要释放变量，并且启动程序的功能键，其代码如下：

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
    SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, nil, 0);       // 打开系统键
    ScreenBitmap.Free;
    CanvasBg.Free;
    picBg.Free;picBall.Free;picMyBrush.Free;picWork.Free;
    ExitProcess(0); end;
```

实例 82　桌面下面的奥秘内核

实例说明

续前面的实例。如图 82-1 所示。

本例主要实现程序的代码部分。具体内容包括实现各快捷菜单项的响应，通过对快捷菜单中内容的选择，可以启动小球和机器工人。当用户拖动鼠标时，将响应 OnMouseMove 事件，鼠标所过之处将会被擦除。

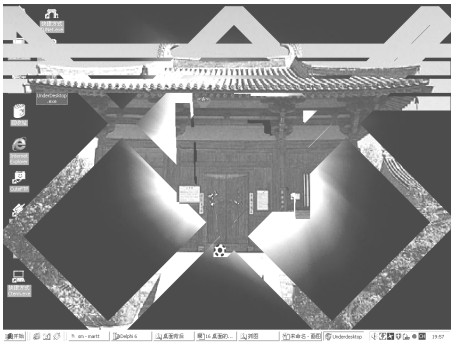


图 82-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。在程序的 OnCreate 事件中获得当前桌面的句柄，并将其保存到一个位图对象中，然后把该位图作为程序的背景全屏显示；同时屏蔽掉系统的功能键（比如<Ctrl+Alt+Del>键结束任务等）。接下来响应用户的按键，如果用户按下的是左键，则开始绘图；如图按下的是右键，将出现快捷菜单，在其中可以选择使用工人或者皮球来帮忙，同时还可以退出程序。

创作步骤

程序设计步骤续前面的实例。本实例的其他代码如下（这里只列出关键代码，其他代码详见配套光盘）：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Menus, MMSystem;
……
  private
    ScreenBitmap: TBitmap;                  // 保存当前整个桌面的位图对象
    bStartDraw: boolean;                    // 定义当前画笔的状态,1 表示开始画,0 表示停止画
    bWorkActive,bBallActive: boolean;   // 用来指示当前是否已经有了工人以及球
    rectView: TRect;                        // 用来保存当前整个桌面的矩形对象，其值为
(0,0,Screen.Width,Screen.Height)
    ptMyBrush, ptWork, ptBall: TPoint;      // 我自己的画刷，工人以及球的最左上角的坐标
    CanvasBg: TCanvas;              // 背景画布
```

```
    picMyBrush,picBg,picWork,picBall:TPicture;  // 分别用来保存我的画笔、背景、工人和球的图片对象
    strCurPath:string;          // 当前可执行文件的路径
    nWorkXDir,nBallXDir,nBallYDir: integer;     // 工人以及球的 X,Y 方向判断,+1 表示向右(下),-1 表示向左(上)
    rectWork,rectWork2,rectBall,rectBall2,rectBrush,rectBrush2: TRect;   // 用来记录各个元素在移动的前后变化的矩形
    procedure IdleLoop(Sender: TObject;var DoneState:boolean);           // IDLE 事件的处理函数
  public
  end;
var
  Form1: TForm1;
  hTimeID: integer;
  proTimeCallBack:TFNTimeCallBack;
  procedure TimeProc(uTimerID, uMessage: UINT; dwUser, dw1, dw2: DWORD) stdcall;
implementation
{$R *.dfm}
procedure TForm1.MenuExitClick(Sender: TObject);
begin
  Close;
  ExitProcess(0);
end;
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if Button in [mbLeft] then          // 如果左键被按下,则开始画图(bStartDraw := true)
  begin
    bStartDraw := true;
    ptMyBrush := POINT(X,Y);
    rectBrush :=
    Rect(ptMyBrush.X,ptMyBrush.Y,ptMyBrush.X+picMyBrush.Width+1,ptMyBrush.Y+picMyBrush.Height+1);
    SetCursor(0);       // 隐藏鼠标
    SetCapture(Handle);
    proTimeCallback:=TimeProc;
    hTimeID:=timeSetEvent(10,0,proTimeCallback,1,1);
  end;
end;
procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton;
```

```
    Shift: TShiftState; X, Y: Integer);
begin
    if Button in [mbLeft] then          // 如果左键被释放,则画图结束,bStartDraw := false;
    begin
        SetCursor(crDefault);           // 显示鼠标
        bStartDraw := false;
        ReleaseCapture;
        timeKillEvent(hTimeID);
        Canvas.CopyRect(rectBrush,CanvasBg,rectBrush);
    end;
end;
procedure TForm1.IdleLoop(Sender: TObject;var DoneState:boolean);
begin
    DoneState := false;
    Sleep(2);
    if bBallActive then             // 如果显示球
    begin
        Canvas.CopyRect(rectBall,CanvasBg,rectBall);
        if ptBall.Y <= 0 then       // 如果向上出了边界,则改变方向为下
            nBallYDir := 1
        else if ptBall.Y >= Screen.Height-32 then       // 如果向下出了边界,则改变方向为上
            nBallYDir := -1;
        if ptBall.X <= 0 then
            nBallXDir := 1
        else if ptBall.X >= Screen.Width-32 then
            nBallXDir := -1;
        if nBallXDir = 1 then
    end;
    if bWorkActive then             // 如果显示工人
        Canvas.CopyRect(rectWork,CanvasBg,rectWork);
        if ptWork.X < 0 then
        begin
            nWorkXDir := 1;
            ptWork.Y := ptWork.Y + 40;
        end else if ptWork.X >= Screen.Width-32 then
        begin
            nWorkXDir := -1;
            ptWork.Y := ptWork.Y + 40;
        end;
```

```
        ptBall.X := ptBall.X + nBallXDir;
        ptBall.Y := ptBall.Y + nBallYDir;
        ptWork.X := ptWork.X + nWorkXDir;        // 改变工人和球的刷新以及显示位置
          ……（此处代码略，详见光盘）
          rectWork2.Bottom := rectWork.Bottom
        else rectWork2.Top := rectWork.Top;
        RealizePalette(Canvas.Handle);
        if bBallActive then          // 把球所在位置的背景拷贝出来到显示中间画布上
          Canvas.Draw(ptBall.X,ptBall.Y,picBall.Icon);
        if bWorkActive then          // 把工人所在位置的背景拷贝出来到显示中间画布上
          Canvas.Draw(ptWork.X,ptWork.Y,picWork.Icon);
      end;
      procedure TForm1.MenuHireWorkClick(Sender: TObject);
      begin
        if bWorkActive then
          Application.MessageBox('你都有一个工人了，还想要啊？自己干吧！','嘿嘿',MB_OK)
        else begin
          ptWork := POINT(0,50);        // 初始化所有的点为 0
          rectWork := RECT(ptWork.X,ptWork.Y,ptWork.X+picWork.Width,picWork.Height);
          bWorkActive := true;
        end;
      end;
      procedure TForm1.MenuGetBallClick(Sender: TObject);
      begin
        if bBallActive then
          Application.MessageBox('球不是这么踢的！'#13'你什么时候见过一个足球场上有两个足球的？','
          嘿嘿',MB_OK)
          ……（此处代码略，详见光盘）
        Canvas.Draw(0,0,ScreenBitmap);
      end;
    end.
```

实例 83　鼠标放大人机交互

实例说明

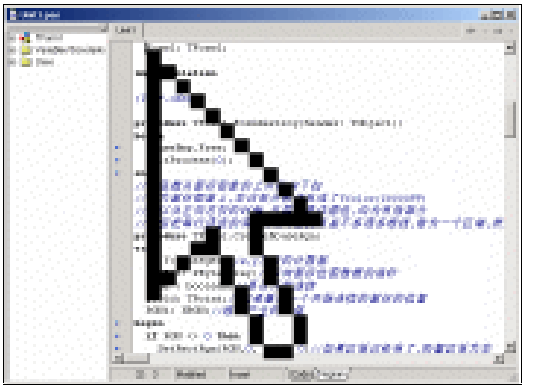


图 83-1　效果图

这里编写鼠标放大的一个特殊效果，如图 83-1 所示。

程序运行后将出现一个窗口，要求选择放大的倍数，选择之后（图 83-1 选择的是放大 20 倍），鼠标将会按照设置的效果显示。这个实例可以经过改装，变成吓人的小游戏。

在本实例中主要实现程序的窗体设计。

编程思路

本实例使用 Delphi 6.0 来实现。其中使用了多个控件，包括 Timer 控件、SpinEdit 控件、Button 控件以及 Label 控件，通过这些控件的设置完成程序的界面设计。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Timer 控件、一个 SpinEdit 控件、两个 Button 控件和一个 Label 控件。如图 83-2 所示。

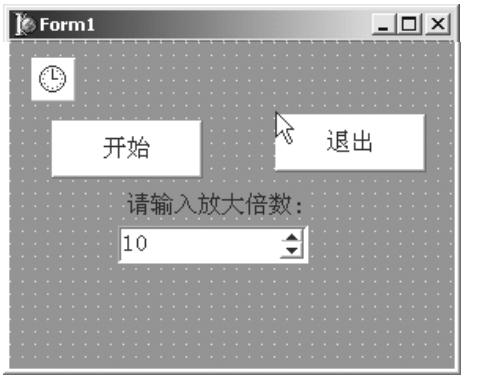


图　83-2

2．设置各控件的属性如下：

object Label1: TLabel

```
    Left = 75
    Top = 96
    Width = 113
    Height = 15
    Caption = '请输入放大倍数:'
  end
  object Button1: TButton
    Left = 27
    Top = 50
    Width = 96
    Height = 37
    Caption = '开始'
  end
  object SpinEdit1: TSpinEdit
    Left = 69
    Top = 117
    Width = 121
    Height = 24
    MaxValue = 30
    MinValue = 2
    TabOrder = 1
    Value = 10
  end
  object Button2: TButton
    Left = 169
    Top = 46
    Width = 96
    Height = 37
    Caption = '退出'
  end
  object Timer1: TTimer
    Enabled = False
    Interval = 1
  end
```

实例 84　鼠标放大组件

实例说明

续前面的实例，如图 84-1 所示。

本例主要完成游戏的代码部分。通过调用 Windows API 函数，完成对操作系统的修改，改变鼠标的大小。其中的核心部分是 CreateMouseRgn 函数，通过这个函数可以完成鼠标放大的基本设计。

图 84-1　效果图

编程思路

本程序的核心部分是 CreateMouseRgn 函数，该函数从鼠标图像的上方开始向下扫描。因为鼠标图像上，空白部分被填充成了 TColor($8000FF)，所以从左向右扫的时候，只要不是该颜色，均为有效部分，然后把每行连续的有效部分，也就是差不多很多根线，作为一个区域，最后把这些区域联合起来形成鼠标的区域。当然还要将程序处于未激活并且处于最小化状态。

创作步骤

程序设计步骤续前面的实例。本实例的源代码如下：

```
unit Unit1;
interface
    ……（此处代码略，详见光盘）
    MouseBmp: TBitmap;  // 鼠标的位图,根据鼠标的形状产生该位图,然后根据该位图产生鼠标状的窗口,跟随鼠标移动
    RGN: HRGN;      // 窗口的区域
    nZoomMul: integer;      // 放大倍数
    nHotX,nHotY: integer;  // 鼠标的热点,用来精确定位鼠标状窗口的位置
    bStart: boolean;    // 是否开始
    procedure CreateMouseRgn;  // 产生鼠标状区域的过程
  public
  end;
var
  Form1: TForm1;
implementation
```

```
{$R *.dfm}
procedure TForm1.FormDestroy(Sender: TObject);
begin
  MouseBmp.Free;
  ExitProcess(0);
end;
procedure TForm1.CreateMouseRgn;     // 产生鼠标区域
var
  X, Y: Integer;  // x,y 方向的计数器
  TmpBit: PByteArray;        // 指向鼠标位图数据的指针
  Line : boolean;        // 是否开始连续
  ptOld: TPoint;         // 记录最后一个开始连续的鼠标的位置
  RGN1: HRGN;        // 临时产生的区域
begin
  if RGN <> 0 then
    SetRectRgn(RGN,0, 0, 0, 0)  // 如果区域已经有了,则置区域为空
  else
    RGN := CreateRectRgn(0,0,0,0);    // 否则产生新区域
  Line := false;
  for Y := 0 to MouseBmp.Height - 1 do
  begin
    TmpBit := MouseBmp.ScanLine[Y];       // 得到第 y 行第一个字节的指针
    for X := 0 to MouseBmp.Width - 1 do
    begin
      if ((TmpBit[3*X] = 128) and (TmpBit[3*X+1] = 0) and (TmpBit[3*x+2] = 255)) or (X =
      MouseBmp.Width - 1) then
      begin      // 如果是填充色或者是到了行尾
        if Line then  // 如果此前已经有开始记录线
        begin
          RGN1 := CreateRectRgn(ptOld.X,ptOld.Y,X,Y+1);
        // 产生一个新区域(一条线段),并把它与总区域联合起来
          CombineRGN(RGN,RGN,RGN1,RGN_OR);
        end;
        Line := false;
      end;
      if ((TmpBit[3*X] <> 128) or (TmpBit[3*X+1] <> 0) or (TmpBit[3*x+2] <> 255)) and not Line then
      begin      // 如果该点不是填充色,而且还没有开始记录线,也就是碰到了新的起始不是填充色
的
        Line := true;  // 开始记录线
```

```
            ptOld := POINT(X,Y);// 记录下当前点的坐标,作为线段的起始点
          end;
        end;
      end;
    end;

    procedure TForm1.Button1Click(Sender: TObject);
    var
      infoCursor: ICONINFO;
      bmpCursor: TBitmap;
      ptNow:TPoint;
      crNow: HCURSOR;
    begin
      nZoomMul := SpinEdit1.Value;     // 当前放大倍数,最大为 30,最小为 2
      Button1.Visible := False;
      Label1.Visible := false;
      WindowState := wsMaximized;
      SpinEdit1.Visible := false;
      MouseBmp := TBitmap.Create; // 产生鼠标位图
      MouseBmp.PixelFormat := pf24bit;
      MouseBmp.Height := 32*nZoomMul;
      MouseBmp.Width := 32*nZoomMul;
      MouseBmp.Canvas.Brush.Color := TColor($8000FF);  // 填充上填充色
      MouseBmp.Canvas.FillRect(RECT(0,0,MouseBmp.Width,MouseBmp.Height));
      bmpCursor := TBitmap.Create;
      crNow := GetCursor;
      GetIconInfo(crNow,infoCursor);        // 得到当前鼠标的信息
      nHotX := infoCursor.xHotspot;
      nHotY := infoCursor.yHotspot;
      bmpCursor.Handle := infoCursor.hbmMask;
      if bmpCursor.Height = bmpCursor.Width then
      begin     // 如果当前的 mask 部分宽高相等,说明是彩色图像,color 部分表示 xor 部分,mask 表示 and
部分
        MouseBmp.Canvas.CopyMode := cmMergeCopy;
        MouseBmp.Canvas.CopyRect(RECT(0,0,MouseBmp.Width,MouseBmp.Height),bmpCursor.Canvas,R
        ECT(0,0,32,32));
        bmpCursor.Handle := infoCursor.hbmColor;
        MouseBmp.Canvas.CopyMode := cmSrcInvert;
    // 先后按照 and 和 xor 模式把鼠标的 mask 和 color 部分拷贝到鼠标图上
```

```
MouseBmp.Canvas.CopyRect(RECT(0,0,MouseBmp.Width,MouseBmp.Height),bmpCursor.Canvas,RECT(
    0,0,32,32));
      MouseBmp.Canvas.CopyMode := cmSrcCopy;
    end else
    begin   // 宽高不等,鼠标是黑白色,这样 mask 的上半部分是 and 部分,下半部分就是 xor 部分
      MouseBmp.Canvas.CopyMode := cmMergeCopy;
     MouseBmp.Canvas.CopyRect(RECT(0,0,MouseBmp.Width,MouseBmp.Height),bmpCursor.Canvas,R
     ECT(0,0,32,32));
      MouseBmp.Canvas.CopyMode := cmSrcInvert;

MouseBmp.Canvas.CopyRect(RECT(0,0,MouseBmp.Width,MouseBmp.Height),bmpCursor.Canvas,RECT(0,3
2,32,64));
      MouseBmp.Canvas.CopyMode := cmSrcCopy;
    end;
    CreateMouseRGN;// 根据产生的鼠标图像产生鼠标区域

    if RGN <> 0 then
    begin
      SetWindowRgn(Handle, RGN, true);    // 如果产生鼠标区域完成,则设置窗口为鼠标区域的形状
    end;
    GetCursorPos(ptNow);
    Canvas.Draw(0,0,MouseBmp);       // 把鼠标的图像画在窗口上

   MoveWindow(Handle,ptNow.X-nHotX*nZoomMul+1,ptNow.Y-nHotY*nZoomMul+1,Width,Height,true
   );   // 开始移动窗口
    bmpCursor.Free;
    Timer1.Enabled := true;
    bStart := true;
  end;

procedure TForm1.Timer1Timer(Sender: TObject);
var
  ptNow:TPoint;
begin
  GetCursorPos(ptNow);     // 得到鼠标当前位置,并把鼠标移动到指定位置;向右错一格是为了使鼠标
的单击依然有效,而不是单击在自己窗口上
    ……（此处代码略，详见光盘）
  Close;
end;
end.
```

实例 85　点你就是点我

实例说明

本例编写点你就是点我的游戏，如图 85-1 所示。

当用户单击“这是你”按钮时，将会响应“这是我”按钮中的事件，并且出现相应的对话框。

图 85-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。主要使用了 Mouse_Event 函数。该函数综合控制鼠标击键和鼠标动作。其原型如下：

```
VOID mouse_event(
    DWORD dwFlags,      //  定义相应的动作
    DWORD dx,           // 指定鼠标沿 x 轴的绝对位置或者从上次鼠标事件产生以来移动的数量，依赖于 MOOSEEVENTF_ABSOLOTE 的设置。给出的绝对数据作为鼠标的实际 x 坐标；给出的相对数据作为移动的 mickeys 数。一个 mickey 表示鼠标移动的数量，表明鼠标已经移动。
    DWORD dy,           // 指定鼠标沿 y 轴的绝对位置或者从上次鼠标事件产生以来移动的数量，依赖于 MOOSEEVENTF_ABSOLVTE 的设置。给出的绝对数据作为鼠标的实际 y 坐标，给出的相对数据作为移动的 mickeys 数。
    DWORD dwData,       // 如果 dwFlags 为 MOOSEEVENTF_WHEEL，则 dwData 指定鼠标轮移动的数量。正值表明鼠标轮向前转动，即远离用户的方向；负值表明鼠标轮向后转动，即朝向用户。一个轮击定义为 WHEEL_DELTA，即 120。
    DWORD dwExtraInfo   // 指定与鼠标事件相关的附加 32 位值。应用程序调用函数 GetMessgeExtrajnfo 来获得此附加信息
);
```

其中 dwFlags 指定单击按钮和鼠标动作的多种情况。此参数里的各位可以是下列值的任何合理组合：

MOOSEEVENTFMOVE：表明发生移动。

M00SEEVENTF_LEFTDOWN：表明按下鼠标左键。

M00SEEVENTF_LEFTUP：表明松开鼠标左键。

MOOSEEVENTF_RIGHTDOWN：表明按下鼠标右键。

MOOSEEVENTF_RIGHTUP：表明松开鼠标右键。

MOOSEEVENTF_MIDDLEDOWN：表明按下鼠标中键。

MOOSEEVENTF_MIDDLEUP：表明松开鼠标中键。

MOOSEEVENTF_WHEEL：在 Windows NT 中如果鼠标有一个轮，表明鼠标轮被移动。移动的数量由 dwData 给出。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Image 控件和两个 Button 控件。向 Image 控件中添加一张图片，并设置两个 Button 控件的 Caption 属性如图 85-1 所示。

2．本实例的源代码如下（这里只列出部分关键代码，其他代码详见配套光盘）：

```
……
procedure TForm1.Button1Click(Sender: TObject);
var
  x,y:integer;
begin
  judge:=not judge;
  x:= form1.Left+button2.Left+25;
  y:= top+button2.Top+27;
  SetCursorPos(x,y);
  Mouse_Event(MOUSEEVENTF_LEFTDOWN,X,Y,0,0);
  Mouse_Event(MOUSEEVENTF_LEFTUP,X,Y,0,0);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    if judge=true then
    begin
       showmessage(哈哈，点你就是点我);
       judge:=not judge;
    end
    else
    showmessage(哈哈，点我还是我)
end;
end.
```

实例 86　隐 藏 时 钟

实例说明

这个程序中只有一个按钮，单击该按钮之后，将会隐藏“开始”菜单和状态栏中的时钟，如图 86-1 所示。

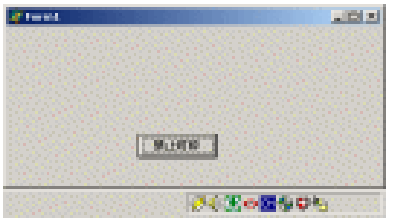


图 86-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。这里调用了多个 API 函数，包括 Findwindow、Showwindow、Getmem、GetClassName 和 PostMessage。下面介绍它们的含义：

Findwindow：该函数获得一个顶层窗口的句柄。该窗口的类名和窗口名与给定的字符串相匹配。这个函数不查找子窗口。在查找时不区分大小写。

Showwindow：该函数设置指定窗口的显示状态。

Getmem：该函数确定在调用线程的消息队列里，是否有鼠标键或键盘消息。

GetClassName：该函数获得指定窗口所属的类的类名。

PostMessage：该函数将一个消息放入（寄送）到与指定窗口创建的线程相联系消息队列里，不等待线程处理消息就返回。消息队列里的消息通过调用 GetMessage 和 PeekMessage 取得。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Button 控件。

2．本实例的源代码如下（这里只列出部分关键代码，其他代码详见配套光盘）：

```
unit Unit1;
interface
uses
   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
   Dialogs, StdCtrls;
……
procedure TForm1.Button1Click(Sender: TObject);
var
Handle,ChildHandle,SecLayerH,H:HWND;
clsName:pchar;
```

```
begin
  h:=findwindow('progman',nil);
  if h<>0 then
    showwindow(h,sw_hide);
  getmem(clsName,255);
  Handle:=FindWindow('Shell_TrayWnd',nil);
  if Handle<>0 then
    begin
      ChildHandle:=GetWindow(Handle,GW_Child);
      while ChildHandle<>0 do
        begin
          GetClassName(ChildHandle,clsName,255);
          if   string(clsName)='Button' then
            begin
              PostMessage(ChildHandle,WM_SYSCOMMAND,SC_CLOSE,0);
            end;
          if   string(clsName)='TrayNotifyWnd' then
            begin
              SecLayerH:=GetWindow(ChildHandle,GW_CHILD);
              while SecLayerH<>0 do
                begin
                  GetClassName(SecLayerH,clsName,255);
                  if string(clsName)='TrayClockWClass' then
                    begin
                      PostMessage(SecLayerH,WM_SYSCOMMAND,SC_CLOSE,0);
                    end;
                  SecLayerH:=GetWindow(SecLayerH,GW_HWNDNEXT);
                end;
            end;
          ChildHandle:=GetWindow(ChildHandle,GW_HWNDNEXT);
        end;
    end;
  freemem(clsName,255);
end;
end.
```

实例 87　做　朋　友

实例说明

本例编写做朋友的小游戏，游戏画面如图 87-1 所示。

在窗口中有两个按钮，供其选择，但是“不行”按钮总是随着鼠标的移动而不断改变位置，只能单击“好呀”按钮。单击“好呀”按钮之后，就会弹出一个“真是太高兴了”的对话框，之后出现“退出”按钮。如果用户使用 Tab 键将“不行”按钮激活，则会出现，非常沮丧的话语。

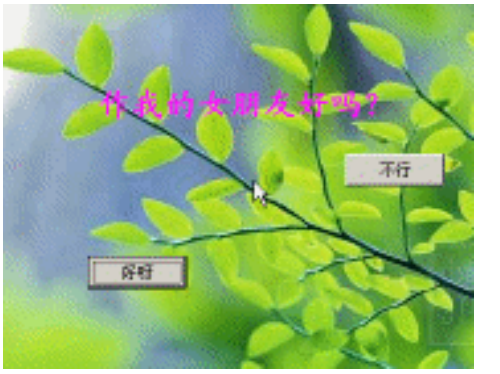


图 87-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。它的关键是响应鼠标的 OnMouseMove 事件，通过该事件得到鼠标的当前位置，然后利用 Button 控件的 Left 属性和 Top 属性不断改变按钮的位置，使鼠标不能单击按钮。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。在新建的界面上加入一个 Image 控件、一个 Label 控件和三个 BitBtn 控件，并将它们放到相应的位置。

2．设置“好呀”按钮的关键属性如下：

Left = 64　　Top = 192　　Width = 75　　Height = 25

Caption = '好呀'　　TabOrder = 0

3．设置“不行”按钮的关键属性如下：

Left = 200　　Top = 192　　Width = 75　　Height = 25

Caption = '不行'　　TabOrder = 1

4．本程序源代码如下（只列出部分关键代码，其余部分详见配套光盘）：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    ……（此处代码略，详见光盘）
```

```
procedure TForm1.image1MouseMove(Sender: TObject; Shift: TShiftState; X,
   Y: Integer);{监测鼠标的位置，同时设置按钮的位置}
begin
if nobt.Left<1 then
    nobt.Left:=form1.Width - nobt.Width–30;
if nobt.Left+nobt.Width>=form1.Width–10 then
    nobt.Left:=1;
if nobt.Top<1 then
    ……（此处代码略，详见光盘）
begin
beep();
showmessage('真是太高兴了！！！');
nobt.Visible:=false;
exitbt.visible:=true;
end;
procedure TForm1.exitbtClick(Sender: TObject);
begin    close; end;
procedure TForm1.FormCreate(Sender: TObject);{隐藏标题栏}
Var
   Save : LongInt;
Begin
   If BorderStyle=bsNone then Exit;
   Save:=GetWindowLong(Handle,gwl_Style);
   If (Save and ws_Caption)=ws_Caption then Begin
      Case BorderStyle of
         bsSingle,
         bsSizeable : SetWindowLong(Handle,gwl_Style,Save and
            (Not(ws_Caption)) or ws_border);
         bsDialog : SetWindowLong(Handle,gwl_Style,Save and
            (Not(ws_Caption)) or ds_modalframe or ws_dlgframe);
      End;
      Height:=Height - getSystemMetrics(sm_cyCaption);
      Refresh;
   End;
   brush.style:=bsClear;
   Inherited;
exitbt.Visible:=false;
end;
end.
```

实例 88　锁 定 鼠 标

实例说明

本例编写游戏可以锁定鼠标，如图 88-1 所示。

单击上面的按钮之后，鼠标被锁定到下面按钮的区域中，即使用户切换到其他窗口，鼠标可移动的区域还是那么大，只有等待 30s，然后单击下面的按钮，才能够取消锁定。

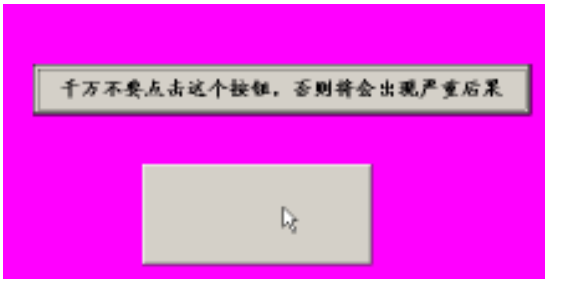


图 88-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现。通过调用 API 函数 ClipCursor 就可以限制鼠标活动的范围。该函数的原型如下：

```
BOOL ClipCursor(
    CONST RECT *lpRect
  );
```

其中：lpRect：指向 RECT 结构的指针，该结构包含限制矩形区域左上角和右下角的屏幕坐标。如果该指针为 NULL（空），则光标可以在屏幕的任何区域移动。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。在新建的界面上加入两个 Button 按钮和一个 Timer 控件。设置 Timer 控件的 Interval 属性为 30000。

2．本实例的源代码如下（这里只列出部分关键代码，其他代码详见配套光盘）：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls;
type
……
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
var
rtButton2: TRect;
begin
rtButton2 := Button2.BoundsRect;
MapWindowPoints(handle, 0, rtButton2, 2); //   坐标换算
ClipCursor(@rtButton2); //   限制鼠标移动区域
Button1.Caption:=完了吧，这下鼠标不能动了';
Timer1.Enabled:=True;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
     Button1.Caption:='单击一下，就好了';
     Timer1.Enabled:=false;
     Button2.Enabled:=True;
     Button2.Caption:=取消锁定';
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
Timer1.Enabled:=False;
Button2.Enabled:=false;
end;
procedure TForm1.Button2Click(Sender: TObject);
var   rtScreen: TRect;
begin
    rtScreen := Rect(0, 0, Screen.Width, Screen.Height);
    ClipCursor(@rtScreen);
    Button2.Caption:=退出';
end;
end.
```

实例 89　掷　骰　子

实例说明

本例编写掷骰子游戏，如图 89-1 所示。

游戏规则如下 :玩家可以定义自己的资金及下注的金额，但下注金额不能超过总资金。由玩家先掷骰子，第一次如果玩家掷 7 点或者 11 点（每次两个骰子，它们的和为投掷的点数），那么玩家胜，将会赢得和自己下注相同的金额；如果掷得 2、3 或者 12 点，那么是庞注，玩家将会输掉刚才的赌注；如果是其他点数（4、5、6、8、9、10），那么第一回合为平局，记录下总点数，游戏继续进行。在以下的回合中当玩家掷到第一次掷的点数时，则玩家赢；当玩家掷到 11 点时，玩家赢；当玩家掷到 7 点时，玩家输；当玩家知道其他点时，为平局。之后按前面的方法继续第三、四回合的游戏。

图 89-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。首先种下随机种子，随机地显示 2 个骰子，然后使用了多个 If 语句，判断玩家是输还是赢，最后给玩家加分或者减分。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加一个 PictureBox 控件、七个 Image 控件、五个 TextBox 控件和三个 Command 控件。

2．向 6 个 Image 控件中分别添加一张骰子的图片，如图 89-2 所示。其他 Command、TextBox 和 Label 控件的属性设置比较简单，这里不再介绍。

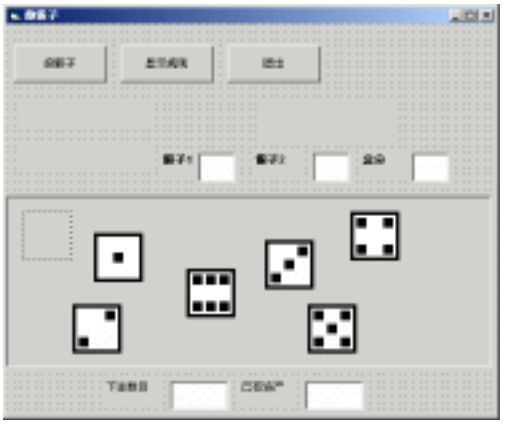


图　89-2

3．**本实例源代码如下：**

```
Option Explicit
Dim turn As Integer
Dim number As Integer
Dim i As Integer
Dim win, bet As Integer
Private Sub CmdExit_Click()
退出程序
Unload Me
End Sub

Private Sub cmdRules_Click()
显示提示信息
Dim rules As String
Dim x As String
x = "这是一个掷骰子的游戏，玩家可以定义自己拥有的资金，" + Chr(13)
x = x + "以及每次下注的金额，但是，下注金额不能太大，不能超过总资金。" + Chr(13)
x = x + "接下来由玩家掷骰子，第一次如果玩家掷 7 点或者 11 点" + Chr(13)
x = x + "（每次两个骰子，它们的和为投掷的点数），那么玩家胜，将会赢得和自己下注相同的金额；
        " + Chr(13)
x = x + "如果掷得 2、3 或者 12 点，那么这次是废注，玩家将会输掉刚才的赌注；" + Chr(13)
x = x + "如果是其他点数（4、5、6、8、9、10），那么第一回合为平局，记录下总点数，游戏继续进
      行。" + Chr(13)
x = x + "在以下的回合中当玩家掷到第一次掷的点数时，则玩家赢；当玩家掷到 11 点时，玩家赢；" +
Chr(13)
x = x + "当玩家掷到 7 点时，玩家输；当玩家知道其他点时，为平局。" + Chr(13)
x = x + "接下来按照前面的方法继续第三、四……回合的游戏。" + Chr(13)
x = MsgBox(x, vbInformation, "Craps Rules")
End Sub
Private Sub cmdThrow_Click()
开始掷骰子
Dim die1, die2, throw, i As Integer, total As Integer
Dim rollnumber As Integer
Dim pointnumber As Integer
使 6 个骰子全部隐藏起来
For i = 0 To 6
    imgDice(i).Visible = False
Next i
lblStatus = ""
```

```
If txtbet <> "" Then
    bet = CInt(txtbet.Text)
Else
    bet = bet
End If
If txtwin <> "" Then
    win = CInt(txtwin)
Else
    win = win
End If
```

判断玩家的下注是否合法

```
If bet > win Then
    MsgBox "你没有这么多资金了，下注少一点吧", vbCritical, "规则提示"
    txtbet.SetFocus
    Exit Sub
End If
```

判断是否结束

```
If turn >= 1 Then
    lblThrows.Caption = "下一次"
Else
    lblThrows.Caption = "第一次"
End If
```

初始化随机数

```
Randomize
die1 = Int(1 + 6 * Rnd)
die2 = Int(1 + 6 * Rnd)
txtDie1.Text = Str(die1)
txtDie2.Text = Str(die2)
total = die1 + die2
txtTotals.Text = Str(total)
```

随机显示 6 个骰子中的 2 个

```
imgDice(die1).Visible = True
    If die2 <> die1 Then
        imgDice(die2).Visible = True
    Else
        imgDice(0).Picture = imgDice(die2).Picture
        imgDice(0).Visible = True
    End If
```

判断结果

```
If turn = 0 Then
    lblPoint.Caption = ""
    Select Case total
        Case 7, 11
            lblStatus.ForeColor = &HFF&
            lblStatus.Caption = "你赢了!!!"
            win = win + bet
            txtwin = Str(win)
                If total = 11 Then
                    lblStatus.Caption = "祝贺祝贺!!!"
                End If
        Case 2, 3, 12
            lblStatus.ForeColor = &HFF&
            lblStatus.Caption = "你输了!!!"
            number = 0
            lblPoint.Caption = ""
            win = win - bet
            txtwin = Str(win)
        Case Else
            turn = turn + 1
            number = total
            lblPoint.Caption = "确定点数为" + Str(number)
            win = win - bet
            txtwin = Str(win)
        End Select
Else
    Select Case total
        Case number
            lblStatus.ForeColor = &HFF&
            lblStatus.Caption = "你赢了!!!"
            turn = 0
            number = 0
            win = win + bet
            txtwin = Str(win)
        Case 7
            lblStatus.ForeColor = &HFF&
            lblStatus.Caption = "你输了!!!"
            turn = 0
            lblPoint.Caption = ""
```

```
                win = win - bet
                txtwin = Str(win)
            Case 11
                lblStatus.ForeColor = &HFF&
                lblStatus.Caption = "你赢了!!!"
                win = win + bet
                txtwin = Str(win)
            Case Else
                turn = turn + 1
            End Select
        End If
End Sub
Private Sub Form_Load() 初始化控件的属性
            For i = 0 To 6
            imgDice(i).Visible = False
        Next i
lblStatus = ""
End Sub
```

实例 90　桌面小精灵人机交互

实例说明

本例编写桌面小精灵游戏，画面如图 90-1 所示。

程序运行后，桌面上出现一个可爱的头像。这个头像可以跟随用户的鼠标不断改变自己的位置。当用户的鼠标在头像上时，它将停止运动，右击鼠标可以弹出快捷菜单，在其中可以选择提示信息或者退出程序。

在本实例中主要完成了游戏的窗体控件设计以及一些简单代码的编写。

图 90-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现。使用 Timer 控件来定时地移动小精灵的位置，同时制作一个快捷菜单，用来实现退出功能。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程，向其中添加 三个 Image 控件和一个 Timer 控件，如图 90-2 所示。

2．设置窗体的关键属性如下：

```
AutoRedraw = -1  'True
BorderStyle = 0  'None
ClientHeight = 780
ClientLeft = 3060
ClientTop = 2565
ClientWidth = 3390
ControlBox = 0   'False
Icon =   "Frm.frx":0000
LinkTopic = "Form1"
ScaleHeight = 780
ScaleWidth = 3390
ShowInTaskbar = 0   'False
```

3．设置 Timer 控件的 Interval 属性值为 50。

4．向 3 个 Image 控件中添加 3 个不同方向的头像，如图 90-2 所示。

5．再新建一个窗体，向其中添加一个 Label 控件和一个 Command 控件，如图 90-3 所示。

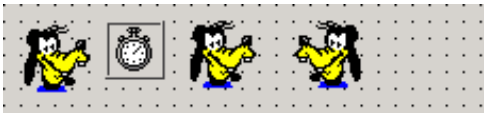


图　90-2

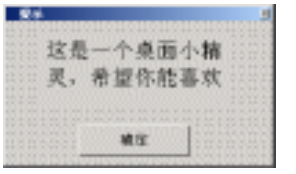


图　90-3

6．设置第二个窗体的属性如下：

```
Name ＝ Mnu
BorderStyle = 4 'Fixed ToolWindow
Caption = "　提示"
ClientHeight = 2310
ClientLeft = 2970
ClientTop = 2895
ClientWidth = 4335
LinkTopic = "Form1"
MaxButton = 0　'False
MinButton = 0　'False
ScaleHeight = 2310
ScaleWidth = 4335
ShowInTaskbar = 0　'False
```

其他两个控件的属性设置比较简单，这里不再介绍。

7．选中第二个窗体，单击工具栏上的菜单编辑器按钮，设计快捷菜单如图 90-4 所示。

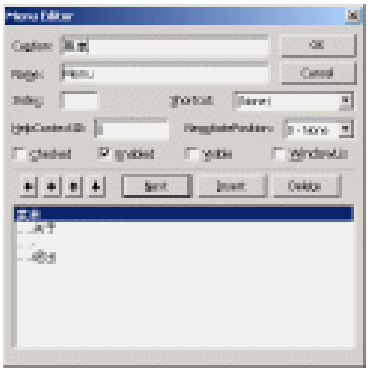


图　90-4

8．编写第二个窗体的事件响应代码如下：

```
Private Sub About_Click()
    Mnu.Show
End Sub
Private Sub Command1_Click()
    Unload Me
```

```
End Sub
Private Sub Exit_Click()
    Unload Frm
    Unload Mnu
End Sub
```

9．向程序中添加一个模块，向模块中添加如下代码：

```
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
Declare Function CreateCompatibleBitmap Lib "gdi32" (ByVal hdc As Long, ByVal nWidth As Long,
                                                     ByVal nHeight As Long) As Long
Declare Function CreateCompatibleDC Lib "gdi32" (ByVal hdc As Long) As Long
Declare Function DeleteDC Lib "gdi32" (ByVal hdc As Long) As Long
Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long, ByVal hdc As Long) As Long
Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject As Long) As Long
Declare Function ShowWindow Lib "user32" (ByVal hwnd As Long, ByVal nCmdShow As Long) As Long
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Declare Function GetCursorPos Lib "user32" (lpPoint As PointApi) As Long
Public ScrDC As Long
Public MemDC1 As Long
Public MemBmp1 As Long
Public Type PointApi
    PosX As Long
    PosY As Long
End Type
```

10．设置主窗体的 Timer 控件的事件响应代码如下：

```
Private Sub Timer1_Timer()
        Dim MyCurPos As PointApi
        GetCursorPos MyCurPos
        If MyCurPos.PosX < (Frm.Left + (Frm.Width / 2)) / Screen.TwipsPerPixelX - 4 Then Image1 =
        Image3: MoveLeft: Exit Sub
        If MyCurPos.PosX > (Frm.Left + (Frm.Width / 2)) / Screen.TwipsPerPixelX + 4 Then Image1 =
        Image2: MoveRight: Exit Sub
        If MyCurPos.PosY < (Frm.Top + (Frm.Height / 2)) / Screen.TwipsPerPixelY - 4 Then MoveUp
        If MyCurPos.PosY > (Frm.Top + (Frm.Height / 2)) / Screen.TwipsPerPixelY + 4 Then
MoveDown
End Sub
```

实例 91　桌面小精灵组件

实例说明

续前面的实例，如图 91-2 所示。

本例主要实现了小精灵的移动，包括 MoveRight()、MoveLeft、MoveDown() 和 MoveUp 四个函数，通过 Timer 控件的 OnTimer 事件或者响应鼠标的 MouseDown 和 MouseMove 事件，调用这四个函数可以实现小精灵的随机移动。

图 91-1　效果图

编程思路

程序运行后，利用 Windows API 函数来实现不规则窗体，接下来响应鼠标的 MouseDown 和 MouseMove 事件，实现头像的移动。在本程序中使用了多个 API 函数，它们的功能如下：

BitBlt：该函数对指定的源设备环境区域中的像素进行位块（bit_block）转换，以传送到目标设备环境。

CreateCompatibleBitmap：该函数创建与指定的设备环境相关的设备兼容的位图。

CreateCompatibleDC：该函数创建一个与指定设备兼容的内存设备上下文环境（DC）。

DeleteDC：该函数删除指定的设备上下文环境（Dc）。

DeleteObject：该函数删除一个逻辑笔、画笔、字体、位图、区域或者调色板，释放所有与该对象有关的系统资源，在对象被删除之后，指定的句柄也就失效了。

GetDC：该函数检索一指定窗口的客户区域或整个屏幕的显示设备上下文环境的句柄，以后可以在 GDI 函数中使用该句柄来在设备上下文环境中绘图。

ReleaseDC：函数释放设备上下文环境（DC）供其他应用程序使用。函数的效果与设备上下文环境类型有关。它只释放公用的和设备上下文环境，对于类或私有的则无效。

SelectObject：该函数选择一对象到指定的设备上下文环境中，该新对象替换先前的相同类型的对象。

ShowWindow：该函数设置指定窗口的显示状态。

GetCursorPos：该函数检取光标的位置，以屏幕坐标表示。

创作步骤

续前面的实例。

本实例的主程序代码如下（其中 Timer 控件的 OnTimer 事件在上一例中）：

```
Private Sub MoveLeft()
    Sleep 150
```

```
        Image1.Visible = False
        Frm.Cls
        Frm.AutoRedraw = True
        Frm.Cls
        Frm.Left = Frm.Left - 5 * Screen.TwipsPerPixelX
        BitBlt Frm.hdc, 0, 0, Frm.Width / Screen.TwipsPerPixelX, Frm.Height / Screen.TwipsPerPixelY,
                ScrDC, Frm.Left / Screen.TwipsPerPixelX, Frm.Top / Screen.TwipsPerPixelY,
                vbSrcCopy
        BitBlt Frm.hdc, 5, 0, Frm.Width / Screen.TwipsPerPixelX, Frm.Height / Screen.TwipsPerPixelY,
                                                MemDC1, 0, 0, vbSrcCopy
        BitBlt MemDC1, 0, 0, Frm.Width / Screen.TwipsPerPixelX, Frm.Height / Screen.TwipsPerPixelY,
                ScrDC, Frm.Left / Screen.TwipsPerPixelX, Frm.Top / Screen.TwipsPerPixelY,
                vbSrcCopy
        Frm.AutoRedraw = False
        Image1.Visible = True
End Sub
Private Sub MoveRight()
        Sleep 150
        Image1.Visible = False
        Frm.Cls
        Frm.AutoRedraw = True
        Frm.Cls
        Frm.Left = Frm.Left + 5 * Screen.TwipsPerPixelX
        BitBlt Frm.hdc, 0, 0, Frm.Width / Screen.TwipsPerPixelX, Frm.Height / Screen.TwipsPerPixelY,
                ScrDC, Frm.Left / Screen.TwipsPerPixelX, Frm.Top / Screen.TwipsPerPixelY,
                vbSrcCopy
        BitBlt Frm.hdc, 0, 0, (Frm.Width / Screen.TwipsPerPixelX) - 5, Frm.Height /
                Screen.TwipsPerPixelY, MemDC1, 5, 0, vbSrcCopy
        BitBlt MemDC1, 0, 0, Frm.Width / Screen.TwipsPerPixelX, Frm.Height / Screen.TwipsPerPixelY,
                ScrDC, Frm.Left / Screen.TwipsPerPixelX, Frm.Top / Screen.TwipsPerPixelY,
                vbSrcCopy
        Frm.AutoRedraw = False
        Image1.Visible = True
End Sub
Private Sub MoveDown()
        Sleep 150
        Image1.Visible = False
        Frm.Cls
        Frm.AutoRedraw = True
```

```
    Frm.Cls
    Frm.Top = Frm.Top + 5 * Screen.TwipsPerPixelY
    BitBlt Frm.hdc, 0, 0, Frm.Width / Screen.TwipsPerPixelX, Frm.Height / Screen.TwipsPerPixelY,
           ScrDC, Frm.Left / Screen.TwipsPerPixelX, Frm.Top / Screen.TwipsPerPixelY,
           vbSrcCopy
    BitBlt Frm.hdc, 0, 0, Frm.Width / Screen.TwipsPerPixelX, (Frm.Height / Screen.TwipsPerPixelY)
           - 5, MemDC1, 0, 5, vbSrcCopy
    BitBlt MemDC1, 0, 0, Frm.Width / Screen.TwipsPerPixelX, Frm.Height / Screen.TwipsPerPixelY,
           ScrDC, Frm.Left / Screen.TwipsPerPixelX, Frm.Top / Screen.TwipsPerPixelY,
           vbSrcCopy
```

……（此处代码略，详见光盘）

```
    BitBlt Frm.hdc, 0, 0, Frm.Width / Screen.TwipsPerPixelX, Frm.Height / Screen.TwipsPerPixelY,
           ScrDC, Frm.Left / Screen.TwipsPerPixelX, Frm.Top / Screen.TwipsPerPixelY,
           vbSrcCopy
    Frm.AutoRedraw = False
    A = True
End Sub

Private Sub Form_Initialize()
    ScrDC = GetDC(0)
    MemDC1 = CreateCompatibleDC(Frm.hdc)
    MemBmp1 = CreateCompatibleBitmap(Frm.hdc, Screen.Width / Screen.TwipsPerPixelX,
                                     Screen.Height / Screen.TwipsPerPixelY)
    SelectObject MemDC1, MemBmp1
End Sub
Private Sub Form_Resize()
    Frm.Width = 800
    Frm.Height = 800
End Sub
Private Sub Form_Unload(Cancel As Integer)
    DeleteObject MemBmp1
    DeleteDC MemDC1
End Sub
Private Sub Image1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
    If Button = 2 Then PopupMenu Mnu.Menu
End Sub
```

实例 92　小　画　笔

实例说明

这里制作小画笔游戏，效果如图 92-1 **所示。**

程序运行后，在窗口中单击鼠标，就可以生成随机的点，如果用户在一处单击鼠标，并且按下几秒钟，点的密度将越来越大，直到成为一个全黑的正方形。

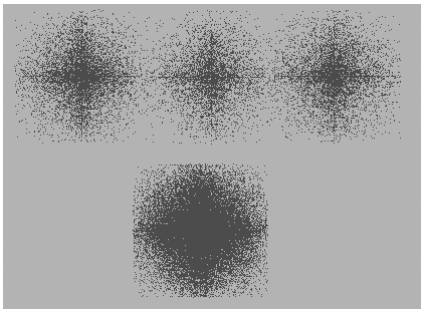


图 92-1　**效果图**

编程思路

本实例使用 Visual Basic 6.0 **来实现，关键是利用一个** Timer **控件随机的生成点，然后响应窗口的** MouseDown、MouseUp、MouseMove **事件，就可以绘制图形了。**

创作步骤

1．**启动** Visual Basic 6.0，**打开一个新的标准工程。如果** Visual Basic **已经运行，则执行** File→New→Project **菜单项，打开一个新的标准工程，向其中添加一个** Timer **控件。**

2．**设定** Timer **控件的关键属性如下：**

```
Interval = 1
```

3．**设置窗体的关键属性如下：**

```
BackColor = &H80000001&
Caption = "小画笔"
LinkTopic = "Form1"
StartUpPosition = 3    'Windows Default
```

4．**本实例源代码如下：**

```
Dim nX, nY As Long
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    绘制图形
    If Button = 1 Then
        Timer1.Enabled = True
        nX = X
        nY = Y
    End If
```

```
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    绘制图形
    If Button = 1 Then
        Timer1.Enabled = True
        nX = X
        nY = Y
    End If
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    停止绘制
    Timer1.Enabled = False
End Sub

Private Sub Timer1_Timer()
    随机的生成点
    Me.DrawWidth = 1
    For i = -10 To 10
        For j = -10 To 10
            PSet (nX + Rnd * i * 100, nY + Rnd * j * 100)
        Next j
    Next i
End Sub
```

实例 93　考考你的观察力界面

实例说明

这是一个考察观察力的小游戏，游戏画面如图 93-1 所示。

单击“开始游戏”按钮后，在四个黑色方框中随机变换颜色。玩家需要观察哪个方框颜色改变了，它改变一次就用鼠标单击一次或者按下 1~4 键（从左到右依次为 1、2、3、4），要求玩家的响应必须和方框颜色变换一致。玩家必须不漏、不错地响应方框的变换，否则即为失败。如果玩家响应正确，则在下面记录成绩，点对一次，记一分，同时显示玩家的等级，“教授”为最高级。

本实例主要向窗体中添加了 5 个 Timer 控件，用来完成不同的功能。

图 93-1　效果图

编程思路

本实例使用 Visual Basic 6.0 来实现，关键利用了随机数和 Timer 控件。本实例使用了 5 个 Timer 控件完成不同的任务。Timer1 用来随机地改变方框的颜色；Timer2 控件用来分析玩家的响应（看玩家是否点错或者漏点），同时给玩家加分；Timer3 用来将方框设置为初始颜色黑色；Timer4 用来及时更新玩家的等级；Timer5 用来判断是否等待用户响应。通过这几个 Timer 控件的有效组合，就可以编写出该游戏了。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加八个 Label 控件、两个 CommandButton 控件、四个 OptionButton 控件和五个 Timer 控件。

2．设置 Label1 控件的 Caption 属性为“成绩”，如图 93-1 所示。

3．设置 Label2 控件的 Caption 属性为“你可以使用键盘上的 1~4 键或者鼠标单击闪亮的方框”，如图 93-1 所示。

4．设定其中的四个 Label 控件的 Name 属性分别为：Lbl1、Lbl2、Lbl3 和 Lbl4，它们的 Caption 属性都设置为空，设置背景色为黑色。

5．设置两个 CommandButton 控件的 Caption 属性分别为“开始游戏”和“退出“。

6．设置 5 个 Timer 控件的关键属性如下：

Timer1 控件：

Enabled = 0 'False

Interval = 200

Left = 600

Top = 3720

Timer2 控件

Enabled = 0 'False

Interval = 3000

Left = 1080

Top = 3720

Timer3 控件：

Enabled = 0 'False

Interval = 200

Left = 1560

Top = 3720

Timer4 控件

Interval = 1

Left = 2040

Top = 3720

Timer5 控件：

Enabled = 0 'False

Interval = 200

Left = 1560

Top = 3720

7．本实例使用了多个公共变量，它们的含义如下：

Public x '公共变量

Public Amount As Integer 给玩家加分

Public Lit As Integer 判断是否有改变颜色的方框

Public Score As Integer 记录成绩

Public Answer As String 记录玩家应做的响应

Public Entry As String 记录玩家响应

Public OnOff As Boolean 记录改变颜色的方框是否已经被单击过

Public LightsDone As Boolean 记录改变颜色的方框是否已经被单击

Public NumOfCalls As Integer 等待玩家单击的方框的数目

Public GameOver As Boolean 判断游戏是否结束

Public BeepYesNo As Boolean '是否发出声音

Public AddScore As Boolean 加分

实例 94　考考你的观察力内核

实例说明

续前面的实例，如图 94-1 所示。

本例中主要定义了多个变量，同时设定了各个按钮的 Click 事件响应。另外还响应 KeyPress 事件，当玩家按下 1、2、3、4 键时，分别对应四个方框的单击响应。

图 94-1　效果图

编程思路

本实例通过 KeyPress 事件响应用户的按键，在该事件处理函数中包括一个参数（KeyAscii As Integer），通过它可以判断玩家按下的是哪一个键，具体的键值对应如下：

49：对应“1”键；

50：对应“2”键；

51：对应“3”键；

52：对应“4”键。

创作步骤

续前面的实例制作步骤。

1．本实例定义的变量如下：

Public x 公共变量

Public Amount As Integer 给玩家加分

Public Lit As Integer 判断是否有改变颜色的方框

Public Score As Integer 记录成绩

Public Answer As String 记录玩家应做的响应

Public Entry As String 记录玩家响应

Public OnOff As Boolean 判断改变颜色的方框是否已经被单击过

Public LightsDone As Boolean 判断方框是否已经被改变颜色

Public NumOfCalls As Integer 等待玩家单击的方框的数目

Public GameOver As Boolean 判断游戏是否结束

Public BeepYesNo As Boolean 是否发出声音

```
Public AddScore As Boolean 加分
```

2．菜单响应以及按键响应事件代码如下：

```
Private Sub CmdExit_Click()
退出程序
End
End Sub

Private Sub CmdExit_KeyPress(KeyAscii As Integer)
响应用户的按键
Select Case KeyAscii
    Case 49
        Call Lbl1_Click
    Case 50
        Call Lbl2_Click
    Case 51
        Call Lbl3_Click
    Case 52
        Call Lbl4_Click
End Select
End Sub

Private Sub CmdStart_Click()
开始游戏
Score = 0
LblScore = Score
Answer = ""
Entry = ""
Amount = 1
NumOfCalls = 0
CmdStart.Enabled = False
GameOver = False
OnOff = True
Timer2.Enabled = False
LightsDone = False
Call Main
End Sub

Private Sub CmdStart_KeyPress(KeyAscii As Integer)
响应按键
```

```
Select Case KeyAscii
    Case 49
        Call Lbl1_Click
    Case 50
        Call Lbl2_Click
    Case 51
        Call Lbl3_Click
    Case 52
        Call Lbl4_Click
End Select
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
响应按键
Select Case KeyAscii
    Case 49
        Call Lbl1_Click
    Case 50
        Call Lbl2_Click
    Case 51
        Call Lbl3_Click
    Case 52
        Call Lbl4_Click
End Select
End Sub
```

3．在窗体的 Load 事件中初始化程序，代码如下：

```
Private Sub Form_Load()
初始化程序
Amount = 1
Lit = 0
LightsDone = False
NumOfCalls = 0
GameOver = True
End Sub
```

本例其他源代码详见光盘。

实例 95　停 车 检 查

实例说明

本例制作一个模拟停车检查的游戏。效果如图 95-1 所示。

假设一个司机因超速被警察扣留，现在需要司机向警察陈述理由，如果司机的话能说服警察，就可以免交罚金，如果司机在规定的字数里没有说出什么有说服力的理由，那就要交罚金。

本例通过 VisualBasic 6.0 中文版制作完成。

图 95-1　效果图

编程思路

本例由于需要输入中文，而中文是双字节字符，也就是说一个汉字占两个 Ascii 码的位置，而 VB 是英语国家用 Len 函数设计的，无法返回汉字字节（也就是说一个汉字和一个 Ascii 字符在 Len 函数中得到的值是一样的），这就需要特殊处理。本例用 Asc（）函数加以实现，在括号内放入一个 ASC 字符或字符串，它会把第一个 ASC 码转化为 0~255 的数值，而把一个汉字放入，它的值必然不在 0~255，这也就可以作为判定汉字的基础，这样可以把英文算作半个汉字。

创作步骤

1．启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程。

2．在新窗体（Form1）中创建一个文本框（Text Box），两个标签（Label），三个图像框（Image），一个命令按钮（Command）。整个窗体界面如图 95-2 所示。

3．设置窗体及各个控件的属性。其中，导入 Image 的图标的方法是：单击要导入图标的 Image 控件，然后在属性栏中单击 Picture 项中 None 后面的小方块，将出现如图 95-3 所示的打开文件对话框，

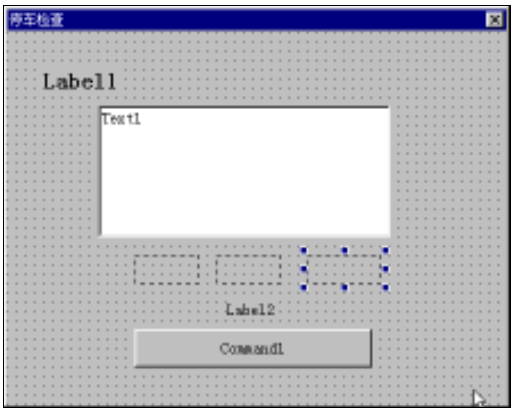


图　95-2

选择要载入的图片或图标即可。如果你的 Visual Basic 的安装路径是默认值的话，本例需要的图标在 C 盘的 Program Files\Microsoft Visual Studio\Common\Graphics\Icon\Traffic 目录下。

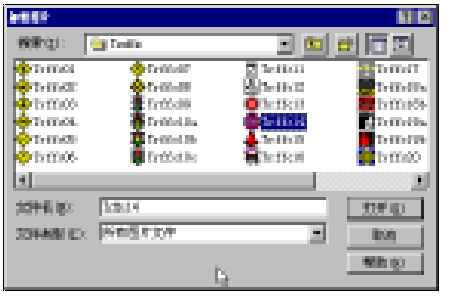


图 95-3

其余各控件的主要属性如下：

Form1 Caption = "停车检查"
BorderStyle = 3
Label1 Caption = "警官，我超速是有原因的，请听我说。"
Label2 Caption = "你必须在 30 字之内完成你理由的申诉，否则将受到罚款！"
Text1 text = ""
MultiLink =True
Command1 Name = Command1
Visible = False
Image1 Visible = True
Strtech = False
Picture 图标是绿灯
Image2 Visible = True
Strtech = False
Pcture 图标是黄灯
Image3 Visible = True
Strtech = False
Pocture 图标是红灯。

其效果图如图 95-4 所示。

图 95-4

4．编写代码。

在程序中将用到 Text 的 Change 事件，每当字符串改变一次就对它进行一次检查，算出已经说的字数，当字数超过 10 时，下面的红绿灯会变成黄色，并且在这时随机抽取一个数，它在一个范围内出现的可能性是 10%，如果在这个范围内就可以免交罚金，如果不在此范围则继续，当文字超过 20 的时候，红绿灯会变成红色，再一次抽取随机数字，步骤同上，如果 30 个字已满，再进行一次检查，如果真的那么不幸就该罚款了。

编写代码详见光盘。

实例 96　生 日 属 性

实例说明

图 96-1　效果图

本例制作通过输入生日然后推算属性（如星座、性格）的小游戏。如图 96-1 所示。

输入姓名、生日后，单击“ 确定 ”按钮，即可得知生日属性。

本例通过 VB 的添加窗体、模块的方法以及多窗体的运用技巧等知识制作完成。

编程思路

本例关键在于界面的制作。用户界面是一个应用程序最重要的部分，它是最直接的现实世界。对用户而言，界面就是应用程序，大家感觉不到正在执行的代码，所以本例通过优化代码生成较好的界面。本例讲解如何加载窗体，如何显示和隐藏窗体，从而实现多个窗体之间的切换，完成界面的制作。

创作步骤

1．**启动** Visual Basic，**打开一个新的标准工程。如果** Visual Basic **已经运行，那么可在“ 文件 ”菜单中单击“ 新建工程 ”菜单项，打开一个新的标准工程。**

2．**在** Form1 **中加入两个标签（**Label**）、七个文本框（**Text Box**）、三个垂直滚动条（**VScrollBar**）、两个命令按钮（**Command**），然后调整各控件的位置。整个窗体界面如图** 96-2 **所示。**

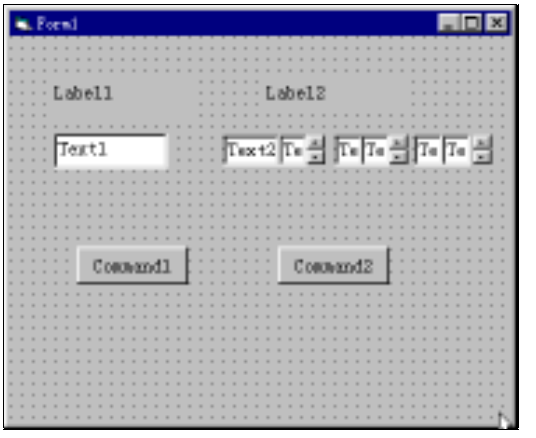


图　96-2

3．本例所讲解的是多窗体的建立和应用，下面要添加一个新的窗体。具体方法是：先单击“工程”菜单，然后选择“添加窗体”子菜单，系统将弹出如图 96-3 所示的添加窗体对话框，在添加窗体对话框中选择“窗体”后，单击“打开”按钮。这时可看到一个新的窗体 Form2 被添加了进来，然后在 Form2 中创建四个标签和一个命令按钮。接下来，调整各控件的位置。Form2 如图 96-4 所示。

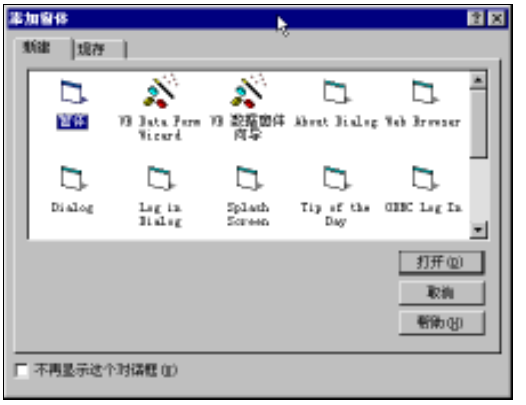


图 96-3

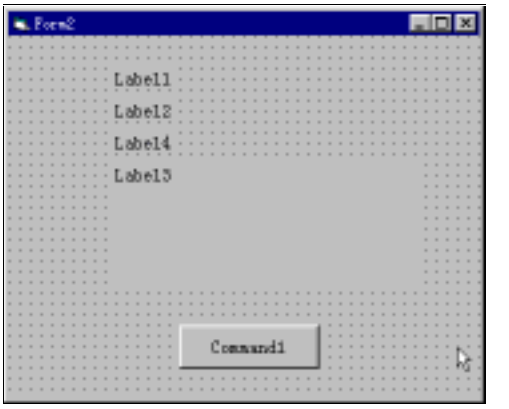


图 96-4

4．设置各控件的属性。在 Form1 中，注意年月日及两个垂直滚动条的位置。在 Form2 中，由于 Form1 要显示的文本较大，所以应把 Label4 适当拖大，各控件的 Font 属性均设为“宋体”（字号为 5 号字）。

各控件的主要属性设置如下：

Label1	Name	=	Label1
	Caption	=	“输入你的姓名”
Label2	Name	=	Label2
	Caption	=	“输入你的生日”
Text1	Name	=	Text1
	Text	=	“”
Text2	Name	=	Text2
	Text	=	“1977”
Text3	Name	=	Text3
	Text	=	“06”
Text4	Name	=	Text4
	Text	=	“15”
Text5	Name	=	Text5
	Text	=	“年”
Text6	Name	=	Text6
	Text	=	“月”
Text7	Name	=	Text7
	Text	=	“日”
VScroll1	Name	=	VScroll1
	LargeChange	=	5
	Max	=	2001

Min = 1800
SmallChange = 1
Value = 1823
Vscroll2 Name = Vscroll2
LargeChange = 2
Max = 11
Min = 0
SmallChange = 1
Value = 1
Vscroll3 Name = Vscroll3
LargeChange = 5
Max = 30
Min = 0
SmallChange = 1
Value = 1
Command1 Caption = “**确定**”
Command2 Caption = “**取消**”

5．**同添加** Form2 **的方法一样，先单击“工程”菜单，然后选择“添加模块”子菜单，添加一个** Modules。

在 Modules **中添入如下代码：**

```
Global sta(1 To 12) As String, stb(1 To 12) As String
Global stc(1 To 12) As String
Global a As String, b As String, c As String
```

编写代码详见光盘。

实例 97　万　花　规

实例说明

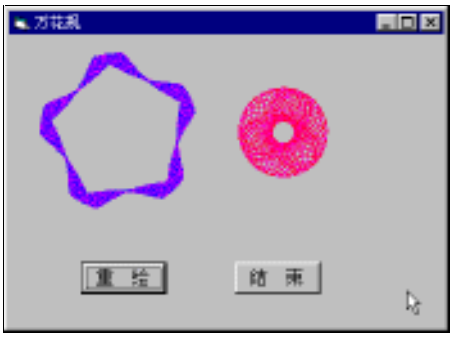


图 97-1　效果图

本例制作绚丽多姿的万花规小游戏。效果如图 97-1 所示。

由于本程序的一些参数是随机的，所以每一次所出现的图基本不一样，非常具有趣味性。

本例通过 VB 的绘点、颜色及随机数处理等知识制作完成。

编程思路

VB 中绘图方法简单、方便、易用，但由于计算机绘图后并不自动清除，如果按一定轨迹绘点，那么它们重叠后的效果就可以组成一个美丽的图案。

本例限定一个大圆，然后让一个小圆内切于大圆转动，在小圆上设一定点，通过这一定点就可以绘出美丽的图案，通过改变小圆直径和定点位置就可以绘出各种各样的图案。本例中小圆直径的定点位置由随机数随机给定。

创作步骤

1．启动 Visual Basic，打开一个新的标准工程。如果 Visual Basic 已经运行，那么可在“文件”菜单中单击“新建工程”菜单项，打开一个新的标准工程。

2．在新窗体（Form1）中创建两个命令按钮（Command）。整个窗体界面如图 97-2 所示。

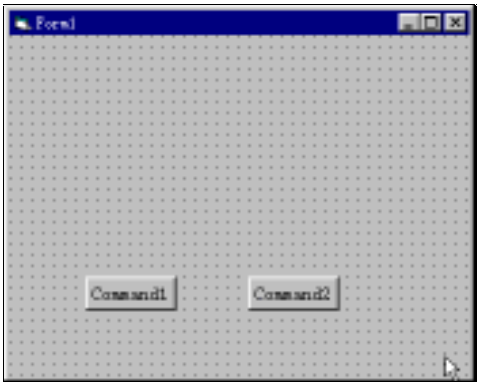


图　97-2

3．设置窗体及各个控件的属性。把 Command1 和 Command2 的 Caption 属性分别设为“绘图”和“结束”，字体为五号宋体，其效果图如图 97-3 所示。

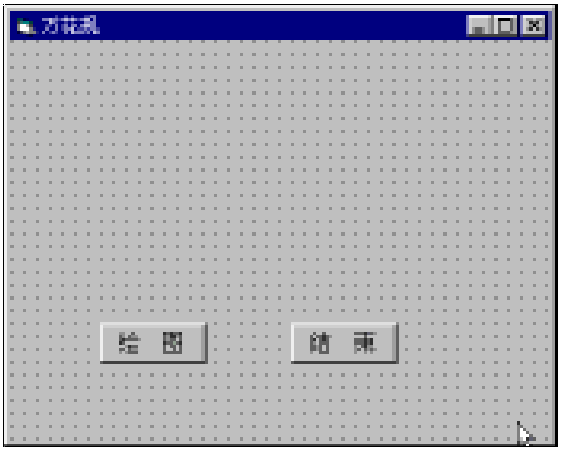


图　97-3

编写源代码之前，先看看原理：

如图 97-4 和图 97-5 所示，设大圆半径为 *R*1，圆心为 *O*，小圆半径为 *r*，圆心为 C，B 为定点，*s* 为 BC 间的距离，当 C 转过 α 时，OC 和 BC 的夹角为 θ，B 的坐标为：

$X=(R1-r)*\cos(\alpha)+s*\cos(\theta-\alpha)$

$Y=(R1-r)*\sin(\alpha)+s*\sin(\theta-\alpha)$

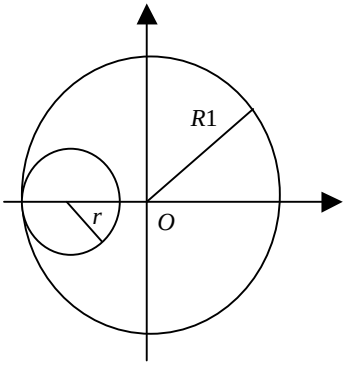


图　97-4

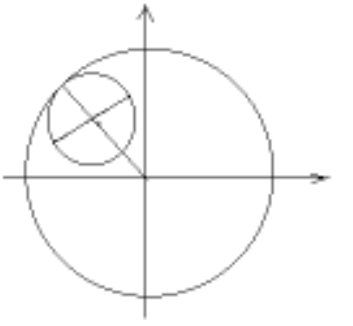


图　97-5

编写代码详见光盘。

实例 98　吊　小　人

实例说明

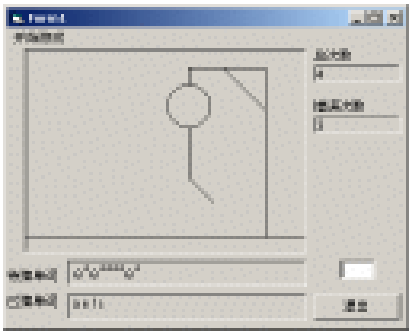


图 98-1　效果图

本例编写吊小人游戏，如图 98-1 所示。

单击“开始游戏”菜单，在窗口右下方输入玩家所猜的字母，如果猜对了，将在“待猜单词”后面显示出已经猜对的字母；如果猜不对，将会出现小人的一部分（比如头或者身子等），当玩家猜错 9 次时，小人将被画全，玩家即告失败。

编程思路

本实例使用 Visual Basic 6.0 来实现。首先使用 Shape 控件和 Line 控件绘制出一个小人的图形，然后在“开始游戏”的 OnClick 事件中将小人隐藏，并且从一个名称为 Word.txt 的文本文件中读入一个单词（该文件和程序在相同的目录下），同时将窗口右下方的 TextBox 控件变为可用状态，最后响应用户的输入，判断用户的输入是否正确，如果正确，在“待猜单词”中显示猜对的字母；如果有错误，将使小人的一部分显示出来。

创作步骤

1．启动 Visual Basic 6.0，打开一个新的标准工程。如果 Visual Basic 已经运行，则执行 File→New→Project 菜单项，打开一个新的标准工程。向其中添加八个 Label 控件、一个 Command 控件、一个 TextBox 控件；在窗体的正中添加一个 PictureBox 控件，并在其中使用 Shape 控件和 Line 控件绘制一个小人图形。如图 98-2 所示。

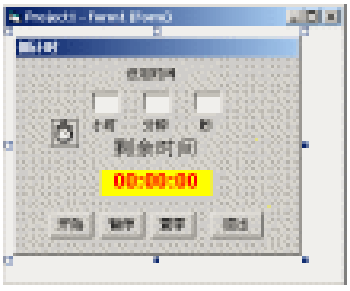


图　98-2

2．设置窗体的关键属性如下：

AutoRedraw = True

BorderStyle = Fixed Single

Caption = "倒计时"

PaletteMode = UseZOrder

3．设置 Timer 控件的关键属性如下：

Enabled= False

4．其他控件的属性设置比较简单这里不再介绍，本实例的源代码如下：

```
Dim Hours As Integer
Dim Minutes As Integer
Dim Seconds As Integer
Dim Time As Date
Private Sub Mydisplay()记录用户的输入
    Hours = Val(Text1.Text)
    Minutes = Val(Text2.Text)
    Seconds = Val(Text3.Text)
将数字转变为时间
    Time = TimeSerial(Hours, Minutes, Seconds)
在 Label1 中显示时间
    Label1.Caption = Format$(Time, "hh") & ":" & Format$(Time, "nn") & ":" & Format$(Time, "ss")
End Sub
Private Sub Command1_Click()
激活 Timer 控件
    Timer1.Enabled = True
    Command3.Enabled = False
End Sub
Private Sub Command2_Click()
暂停时间记录
    Timer1.Enabled = False
    Command3.Enabled = True
End Sub
Private Sub Command3_Click()
重置时间
    Hours = 0
    Minutes = 0
    Seconds = 0
    Time = 0
    Text1.Text = " "
    Text2.Text = " "
    Text3.Text = " "
    Text1.SetFocus 'put curser in the hour text box
End Sub
Private Sub Command4_Click()
```

退出程序

```
    End
End Sub
Private Sub Form_Load()
```

使窗口居中

```
Form1.Top = (Screen.Height - Form1.Height) / 2
Form1.Left = (Screen.Width - Form1.Width) / 2
    Timer1.Interval = 1000
    Hours = 0
    Minutes = 0
    Seconds = 0
    Time = 0
End Sub
Private Sub Text1_Change()
    Mydisplay
End Sub
Private Sub Text2_Change()
    Mydisplay
End Sub
Private Sub Text3_Change()
    Mydisplay
End Sub
Private Sub Timer1_Timer()
```

动态改变时间

```
    Timer1.Enabled = False
    If (Format$(Time, "hh") & ":" & Format$(Time, "nn") & ":" & Format$(Time, "ss")) <> "00:00:00"
    Then 'Counter to continue loop until 0
        Time = DateAdd("s", -1, Time)
        Label1.Visible = False
        Label1.Caption = Format$(Time, "hh") & ":" & Format$(Time, "nn") & ":" & Format$(Time, "ss")
        Label1.Visible = True
        Timer1.Enabled = True
    Else
        Timer1.Enabled = False
        Beep
        Beep
        Command3.Enabled = True
    End If End Sub
```

实例 99　点　　我

实例说明

本例运行后，出现一个按钮在所有窗口的上方，上面写着“快点我”三个字，画面如图 99-1 所示。

运行时当玩家将鼠标移动到按钮旁边时，按钮就会跳到一边，使鼠标点不到它。如果要关闭它有两种方法：一是激活该程序窗口，使按钮处于被选中状态，然后按下 Enter 键；二是在状态栏中单击程序的小图标，出现一个“退出”快捷菜单，单击该菜单项，即可关闭程序。

图 99-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现，在程序中着重讲述了使用 Delphi 调用 Windows 窗体消息的方法。其具体的实现步骤是：首先将主窗口变为透明，然后隐藏主窗口在状态栏中的显示，并且将其缩小为任务栏右侧的一个小图标，最后当鼠标靠近窗口时，随机移动窗口，使玩家单击不到按钮。

创作步骤

1．启动 Delphi 6.0，打开一个新的标准工程。如果 Delphi 已经运行，则执行 File→New→Application 菜单项，打开一个新的标准工程。向其中添加一个 Button 控件和一个 PopupMenu 控件。

2．设置 PopupMenu 控件如图 99-2 所示。

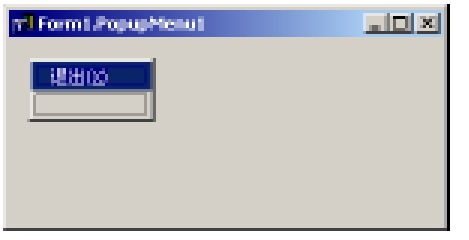


图　99-2

3．本实例源代码如下：

```
unit Unit1;
interface
```

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, ShellAPI, StdCtrls;
const
   ID_MYICON = 1900;     // MyIcon 的全局标识
   WM_MYEVENT = WM_USER + 1;// 自定义消息
type
  TForm1 = class(TForm)
    PopupMenu1: TPopupMenu;
    MenuExit: TMenuItem;
    Button1: TButton;
    procedure MenuExitClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
    procedure Button1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
  private
    MyIcon:TIcon;
    procedure IconOnClick(var msg:TMessage);message WM_MYEVENT;// 自定义消息的处理函数
  public
  end;

var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.MenuExitClick(Sender: TObject);
begin
  Close;
end;
procedure TForm1.FormCreate(Sender: TObject);
var
  IconData:TNotifyIconData;
  wsMyStyle: Longint;
begin
  MyIcon:=TIcon.Create;
  MyIcon.LoadFromFile('MyIcon.Ico');// 读入 icon
```

```
  with IconData do
  begin
    cbSize:=SizeOf(IconData);
    Wnd:=Handle;
    uID:=ID_MYICON;
    uFlags:=NIF_ICON+NIF_MESSAGE;
    uCallBackMessage:=WM_MYEVENT;// 设定 IconData 的属性,尤其是事件回调函数
    hIcon:= MyIcon.Handle;
    szTip:=嘿嘿,看你怎么点我?';
    if not Shell_NotifyIcon(NIM_ADD,@IconData) then// 产生任务栏图标
    begin
      ShowMessage('产生任务栏图标失败!');
      Application.Terminate;
    end;
  end;
  Width := Button1.Width;
  Height := Button1.Height;
  Randomize;
  wsMyStyle := GetWindowLong(Handle,GWL_EXSTYLE);
  SetWindowLong(Application.Handle,GWL_EXSTYLE,wsMYStyle+WS_EX_TOOLWINDOW);
  // 隐藏主窗口在标题栏中的显示
end;
procedure TForm1.IconOnClick(var msg:TMessage);
var
  ptCur:TPoint;
begin
  if (msg.LParam = WM_LBUTTONUP) or (msg.LParam = WM_RBUTTONUP) then
  begin// 自定义的消息回调函数,如果是左键或者右键击下,则跟踪鼠标的位置,显示提示
    GetCursorPos(ptCur);
    try
      if (Screen.ActiveForm.Handle <> 0) then
        SetForegroundWindow(Screen.ActiveForm.Handle);
    except
    end;
    PopupMenu1.Popup(ptCur.x,ptCur.y);
    PostMessage(Screen.ActiveForm.handle,WM_NULL, 0, 0 );
  end;
end;
```

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  MyIcon.Free;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var
  IconData: TNotifyIconData;
begin
  IconData.cbSize := sizeof(IconData); //   IconData 变量的字节数
  IconData.uID := ID_MYICON; // 内部标识，与加入小图标时的数一致
  IconData.Wnd := Handle; // 主窗口句柄
  Shell_NotifyIcon(NIM_DELETE, @IconData); // 程序关闭,去掉小图标
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Application.MessageBox('哇!!你太厉害了!!','我要退出了',MB_OK);
  Close;
end;

procedure TForm1.Button1MouseMove(Sender: TObject; Shift: TShiftState; X,Y: Integer);
var
  nX,nY: integer;
begin// 当鼠标靠近窗口的时候,随机远离鼠标的位置
  repeat
    nx := Random(10000) mod (Screen.Width-Width);
  until (nX > X) or ((nX+Width) < X);
  repeat
    nY := Random(10000) mod (Screen.Height-Height);
  until (nY > Y) or ((nY+Width) < Y);
  MoveWindow(Handle,nX,nY,Width,Height,true);
end;
end.
```

实例 100　定 制 形 状

实例说明

这个实例演示如果创建任意形状的窗体，如图 100-1 所示。

程序运行后，窗体是一个图片的形状，用户只能在该图片区域内，才能对程序进行操作。

图 100-1　效果图

编程思路

本实例使用 Delphi 6.0 来实现，需要调用多个 API 函数，下面介绍一下这些函数的使用方法。

制作不规则形状窗体需要调用如下 API 函数：GetWindowDC、CreateCompatibleDC、SelectObject、BeginPath、EndPath、PathToRegion 和 ReleaseDC。

下面分别介绍它们的用法。

GetWindowDC 函数用来得到窗体的标题栏、菜单和滚动条等信息，其格式如下：

```
HDC GetWindowDC(
    HWND hWnd      //    handle of window
   );
```

CreateCompatibleDC 函数用来存储和设备相一致的信息。其使用方法如下：

```
HDC CreateCompatibleDC(
    HDC hdc     //    handle to memory device context
   );
```

SelectObject 函数用来将一个对象选到设备中，新的设备将取代以前的设备。其使用方法如下：

```
HGDIOBJ SelectObject(
    HDC hdc,    //    handle of device context
    HGDIOBJ hgdiobj        //    handle of object
   );
```

BeginPath 函数为设备开辟一个新的空间，其使用方法如下：

```
BOOL BeginPath(
    HDC hdc     //    handle to device context
   );
```

CloseFigure 函数用来关闭一个打开的数据路径。其使用方法如下：

```
BOOL CloseFigure(
    HDC hdc     //    handle to device context
   );
```

EndPath 函数用来关闭指定的设备路径，其使用方法如下：

```
BOOL EndPath(
    HDC hdc    //  handle to device context
   );
```

PathToRegion **函数为指定的设备开辟空间，其使用方法如下：**

```
HRGN PathToRegion(
    HDC hdc    //  handle to device context
   );
```

ReleaseDC **函数用来释放设备所占的内存，其使用方法如下：**

```
int ReleaseDC(
    HWND hWnd,    //  handle of window
    HDC hDC    //  handle of device context
   );
```

创作步骤

1．**启动** Delphi 6.0，**打开一个新的标准工程。如果** Delphi **已经运行，则执行** File→New→Application **菜单项，打开一个新的标准工程，向其中添加一个** Button **控件和一个** Image **控件。如图** 100-2 **所示。**

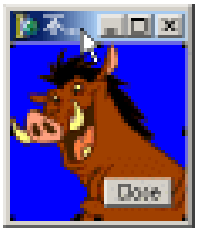


图 100-2 **程序的窗体设计**

2．**添加代码如下：**

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    Image1: TImage;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    function CreateRegion(wMask: TBitmap; wColor: TColor;
      hControl: THandle): HRGN;
```

```
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}

function Tform1.CreateRegion(wMask:TBitmap;wColor:TColor;hControl:THandle): HRGN;// 产生窗体区域
……此处代码略，详见光盘。

procedure TForm1.FormCreate(Sender: TObject);       // 初始化变量
var
  w1:TBitmap;
  w2:TColor;
  rgn: HRGN;
begin
  w1:=TBitmap.Create;
  w1.Assign(image1.Picture.Bitmap);
  w2:=w1.Canvas.Pixels[0,0];
  rgn := CreateRegion(w1,w2,Handle);
  if rgn<>0 then
  begin
     SetWindowRgn(Handle, rgn, true);
  end;
  w1.Free;
end;

procedure TForm1.Button1Click(Sender: TObject);     // 退出程序
begin
  Close;
end;

procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  SendMessage(Handle, WM_SYSCOMMAND, $F012, 0);
end;
end.
```