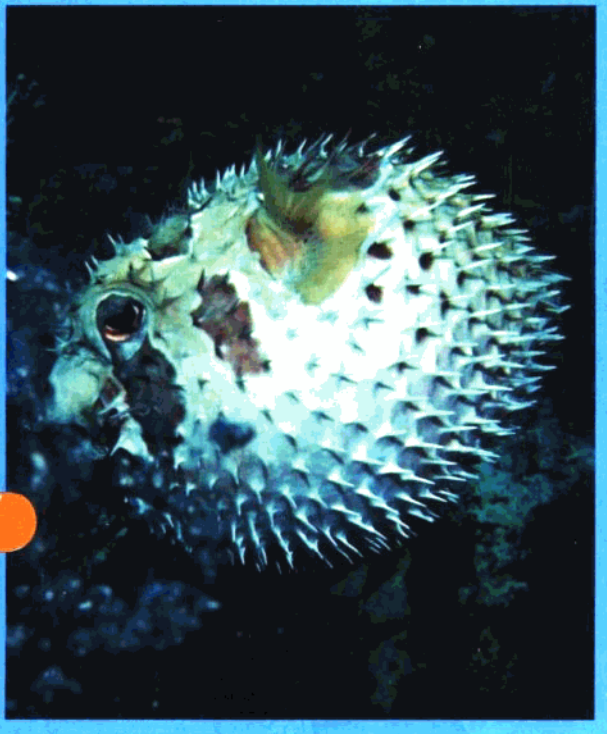
北京科海培训中心

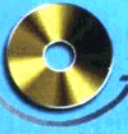
以完成作品为教学导向，真正整合理论与实务

Delphi 案例教程



- 全书案例立足于经典实用、且突出设计思想上的新颖之处，以启发读者的思路并提高编程水平为目的。也可作为设计程序时的参考
- 100多个案例，从表层的界面设计到底层的API函数，以及消息机制、网络编程等等，应有尽有
- 光盘包括每个案例的完整源代码和执行程序，另外还有200多个实用控件

陈志华 编著



- 案例源文件导读
- 实用控件大派送

中科多媒体电子出版社

前　言

本书是一本通过讲解 Delphi 案例来说明其编程思路以及设计方法的计算机书籍。通过对每个案例的编程思路和代码实现进行详细分析和讲解，使读者能够快速地提高自己的程序设计水平，并达到举一反三的目的。

本书主要面向中、高级用户，所以读者最好具有一定的 Delphi 程序设计基础（例如，熟悉 Object Pascal 语言），有一定的程序设计经验，最好有一定的项目管理经验。但初级用户肯定也能从中学到很多新颖的东西，包括设计思想和技巧等。不过这需要读者在遇到不懂的地方多多参考 Delphi 提供的帮助以及 Microsoft 的 MSDN。通过本书的学习，相信读者的 Delphi 水平会上一个新的台阶。

本书的案例内容涉及面比较广泛，从表层的界面设计到 Windows 底层的 API 函数以及消息机制，从图像处理到多媒体制作，从网络编程到数据库技术都进行了相应地讲解。另外，还针对 Delphi 某些自带控件的缺点作了分析验证，并提出了相应的解决办法，而且对某些控件的使用方法提出了提高效率的措施并进行了验证。这些内容基本上囊括了通常程序设计中的各个方面，凝聚了作者多年的 Delphi 编程经验，相信对启发读者的思路并提高读者的编程水平会有很大帮助。本书立足于经典但更突出新颖，而不是简单的控件的使用。

在众多内容中，本书也有所侧重，力求将笔者理解最深入的部分介绍给读者。例如 Windows API 函数的使用、Delphi 中 Windows 消息的处理机制、图像的处理、网络编程以及数据库技术等。选择这些作为侧重点的原因有 4 个方面：第一，目前市面上类似书籍对这些方面的内容涉及不多，而且不够深入；第二，这些内容在通常的程序设计中使用非常频繁；第三，这些内容能使读者更加了解 Windows 系统的工作机制，更加深入地了解 Delphi，更加明白 Inprise 公司辛勤的工程师们为我们做了什么，从而更加喜爱 Delphi 这个美妙的开发工具；第四，消除很多读者的认识误区，例如，认为 Delphi 只是在界面设计和数据库方面很强大，并因此将 Delphi 这个概念仅仅定位在“界面设计”或者“数据库”上。事实上，通过阅读本书，读者将会发现 Delphi 的功能是如此之强大。在 Windows 编程中，Visual C++ 能实现的功能，Delphi 基本上都能实现。而且，利用 Delphi 能使开发工作事半功倍，极大地提高开发效率。

本书精选了 100 多个经典的、有独特的设计思想和编写技巧的案例来进行分析讲解。随书光盘中包含所有案例的源程序和执行文件，读者可随时查看程序的运行结果，以加深对代码的理解。

本书共分 8 章。第 1 章“界面设计”讲述了一些新颖的界面设计技巧；第 2 章“系统编程”深入讲解了 Windows 系统编程方法，读者从而可以体会 API 函数和消息处理在程序设计中的强大威力；第 3 章“Shell API 编程”讲解了 Windows Shell 的 API 函数使用技巧；第 4 章“组件编程”讲解了 Delphi 自带的一些组件的通常使用方法及其提高效率的优化方法，并针对某些组件的缺点作了改进；第 5 章“图像编程”讲述了 Delphi 中若干图像编程

技巧；第 6 章“多媒体编程”分析了几个多媒体编程案例；第 7 章“网络编程”涉及了大量的非常有用的网络编程技巧；第 8 章“数据库编程”分析了数据库编程中常见的问题及其产生的原因，并给出了相应的解决办法。

书中案例全部在 Delphi 6.0+Windows 98/2000 Advance Server 环境下编译通过，除非有特殊说明，在 Delphi 5+Windows 98/2000 中也能调试通过。由于本书注重各版本间的通用性且比较偏向于底层，故尽量避免使用 Delphi 6 的新控件。

本书在很多案例的重要部分都有提示、注意等特殊说明。目的是为了加深读者的印象，避免不必要的错误，以使读者能够更高效地利用 Delphi 这个强大的程序开发工具。

由于作者水平有限，书中难免会出现一些错误和不当之处，敬请批评指正。

编者

2001 年 8 月

光盘使用说明

运行环境：

- Pentium 166以上的处理器
- VGA显卡
- 光盘驱动器
- Windows 95/98/NT/2000操作系统
- Internet Explorer 4.0及以上版本浏览器
- 800×600分辨率
- 16位真彩色以上显示模式

操作方法：

- 一般情况下，将本光盘放到光驱中后，光盘就会自动运行。
- 如果本光盘没有自动运行，请双击光盘根目录下的Start.exe或Index.htm文件，即可开始运行。
- 如果计算机上没有安装Winzip解压软件，请先运行光盘根目录下的Winzip80.exe程序进行安装。

光盘内容：

- 案例源文件
- 实用控件

目 录

第1章 界面设计

本章主要介绍 Delphi 中常用的界面设计技巧。这里的案例都比较简单，通常涉及的只是一个或几个消息的处理或 API 函数的使用，但这些技巧在程序设计中会经常用到。用户通过使用这些设计技巧，可以使自己的程序界面更加丰富多彩。

案例 1.1 制作不可移动的窗体

〖案例说明〗

本例制作的是一个不可移动的窗体。窗体外观类似于对话框，其主要特点是不可移动，也不可以更改尺寸。

〖设计思想〗

窗体移动时会触发系统消息 WM_WINDOWPOSCHANGING，该消息中包含有新窗体的位置，用来通知内核该窗体要发生移动，内核接收到该消息后，开始销毁该窗体并在新位置重绘。如果更改消息的内容，使窗体保持在其原来的位置，那么窗体就不会移动了。

〖设计步骤〗

1．新建一个应用程序。将窗体的 BorderStyle 属性设置为 bsDialog，这样窗体就不可改变大小了。

2．在窗体类的 protected 部分加入消息处理过程 OnPosChange(var Msg: TWMWindow PosChanging);，用来更改系统消息 WM_WINDOWPOSCHANGING，使其所包含的窗体新位置与原来的相同（见〖代码分析〗部分）。

〖代码分析〗

本例主要代码如下：

```
protected
  //过程声明
  procedure OnPosChange(var Msg:TWMWindowPosChanging);
    message WM_WINDOWPOSCHANGING;
    ...

//过程实现
procedure TForm1.OnPosChange(var Msg:TWMWindowPosChanging);
begin
```

```
    //消息的 WindowPos.x 部分包含新窗体的左边位置，将其改成原来窗体的左边坐标
    Msg.WindowPos.x:=Left;
    //Left 的完整名称为 Form1.Left
    //消息的 WindowPos.y 部分包含新窗体的顶部位置，将其改成原来窗体的顶部坐标
    //Top 的完整名称为 Form1.Top
    Msg.WindowPos.y:=Top;
    Msg.Result:=0;
  end;
```

〖总结〗

通过修改系统消息 WM_WINDOWPOSCHANGING 的内容，就可以随意控制窗体对用户拖动的反应，本例只是使窗体不随用户鼠标的拖动而移动。不难理解，同样可以实现用户向左拖动而窗体向右移、用户向上拖动而窗体向下移动等特殊效果。

案例 1.2　制作圆形窗体

〖案例说明〗

本例制作的是一个圆形窗体，如图 1.1 所示。只要将代码稍作改变，即可制作出椭圆、三角形和梯形等形状的窗体。

图1.1　圆形窗体

〖设计思想〗

API 函数 SetWindowRgn 可以设置窗口的显示区域，而该区域外的窗体系统不予显示。在 MSDN 中此函数的声明如下：

```
int SetWindowRgn(
  HWND hWnd,          //待处理窗口的句柄
  HRGN hRgn,          //窗口区域句柄
  BOOL bRedraw        //窗口重绘标识
);
```

如果参数 hRgn 是指向圆形区域的句柄，则窗体就为圆形。圆形区域的创建可用 API 函数 CreateEllipticRgn 来实现，该函数用来创建椭圆区域。MSDN 中此函数的声明如下：

```
HRGN CreateEllipticRgn(
  int nLeftRect,     //椭圆边界矩形左上角的x坐标
  int nTopRect,      //椭圆边界矩形左上角的y坐标
  int nRightRect,    //椭圆边界矩形右下角的x坐标
  int nBottomRect    //椭圆边界矩形右下角的y坐标
);
```

注意：“圆”是特殊的“椭圆”。

〖设计步骤〗

新建一个应用程序，双击窗体，添加 OnCreate 事件的处理过程（见〖代码分析〗部分）。

〖代码分析〗

本例主要代码如下：

```
...
TForm1=class(TForm)
  //过程声明
  procedure FormCreate(Sender:TObject);
  ...

  //过程实现
procedure TForm1.FormCreate(Sender:TObject);
var
  Region:HRgn;
begin
  //创建圆，边界矩形为正方形的椭圆就是圆
  Region:=CreateEllipticRgn(1,1,200,200);
  //将窗体区域设为新创建的圆，参数 bRedraw 为 True 表示需要系统重绘窗口，此处也可
  //为 False
  SetWindowRgn(Handle,Region,True);
end;
```

〖总结〗

窗体的区域通过参数 hRgn 来表示，除了椭圆外，还可以是梯形、三角形、星形、七边形等。总之，想要什么形状就可以是什么形状，这可以通过调用 API 函数 CreatePolygonRgn 来实现。该函数的详细使用方法请参见 MSDN 中的相关说明。

案例 1.3 制作不可见窗体

〖案例说明〗

本例制作的是一个不可见窗体，这样可以隐藏程序。如果不希望程序轻易被别人发现，

这是必须实现的一步（当然，只是使窗体不可见还不够）。鉴于窗体的不可见特性，此处没有给出效果图，请读者自行运行配套光盘中的相应程序，以验证其效果。

〖设计思想〗

Delphi 的窗体有一个 Visible 属性，但仅将其设为 False 还不能实现窗体的隐藏，还要将 Application 的 ShowMainForm 属性设为 False 才行。

〖设计步骤〗

新建一个应用程序，双击窗体，添加 OnCreate 事件的处理过程，并将窗体的 Visible 属性设置为 False（见〖代码分析〗部分）。

〖代码分析〗

本例主要代码如下：

```
//处理窗体的 OnCreate 事件
procedure TForm1.FormCreate(Sender:TObject);
begin
  Application.ShowMainForm:=False;
end;
```

〖总结〗

本例很简单，如果要将自己的程序隐藏得更深，还需要作进一步地处理，这些技巧将在后面的案例中介绍。

案例 1.4　制作始终位于最上层的窗体

〖案例说明〗

应用程序在失去焦点后，其窗体一般会被系统调度到新激活程序窗体的后面。而在某些情况下，为了用户的使用方便或者别的目的，希望窗体始终位于最上层。本例将介绍如何实现这种功能，程序运行效果如图 1.2 所示（背景为当前激活的 IE 窗口的一部分，本程序窗口在其上层，而且是非激活的）。

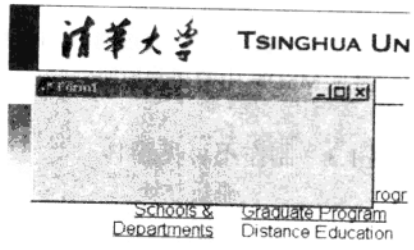


图 1.2　制作始终位于最上层的窗体

〖设计思想〗

本例可以用两种方法来实现：

- 简单方法是将窗体的 FormStyle 属性设为 fsStayOnTop。
- 复杂方法是调用 API 函数 SetWindowPos。

本质上，Delphi 在处理窗体的 fsStayOnTop 属性时就是调用 API 函数 SetWindowPos，所以后一种方法更灵活。因为通过设置其参数还可以实现窗体的隐藏与显示，以及防止一些消息（如 SWP_DEFERERASE）的产生，本例即采用此方法。

MSDN 中 API 函数 SetWindowPos 的声明如下：

```
BOOL SetWindowPos(
  HWND hWnd,                  //窗体句柄
  HWND hWndInsertAfter,       //放置窗体位置的句柄
  int X,                      //窗体水平位置坐标
  int Y,                      //窗体竖直位置坐标
  int cx,                     //窗体宽度
  int cy,                     //窗体高度
  UINT uFlags                 //窗体位置标识
);
```

通过将参数 hWndInsertAfter 设置为 HWND_TOPMOST，就可使窗体始终位于最上层。

〖设计步骤〗

新建一个应用程序，双击窗体，添加 OnCreate 事件的处理过程（见〖代码分析〗部分）。

〖代码分析〗

本例主要代码如下：

```
//处理窗体的 OnCreate 事件
procedure TForm1.FormCreate(Sender:TObject);
begin
  with Form1 do
    SetWindowPos(Handle,
      //将参数 hWndInsertAfter 设置为 HWND_TOPMOST
      HWND_TOPMOST,
      //使窗体的位置和尺寸不变
      Left,
      Top,
      Width,
      Height,
      //SWP_NOACTIVATE：不激活窗体
      //SWP_NOMOVE：保持窗体当前位置，忽略 X 和 Y 参数
      //SWP_NOSIZE：保持窗体当前尺寸，忽略 cx 和 cy 参数
```

```
        SWP_NOACTIVATE or SWP_NOMOVE or SWP_NOSIZE);
    end;
```

〖总结〗

通过参数hWndInsertAfter的不同设置，可以实现不同的效果，此参数的取值及其效果说明见表1.1。

表1.1 参数hWndInsertAfter的主要取值及其效果说明

参数 HWndInsertAfter 的取值	效果说明
HWND_TOP	窗体位于Z order顶部，但在topmost窗体之下
HWND_TOPMOST	窗体位于所有非topmost窗体之上，即使没有激活
HWND_NOTOPMOST	窗体位于所有非topmost窗体之上，但在所有topmost窗体之下
HWND_BOTTOMMOST	窗体位于Z order底部

另外，参数 uFlags 也有很多值可选，具体请参见 MSDN。

案例 1.5 为窗体创建动画光标

〖案例说明〗

本例程序的功能是为窗体创建动画光标，其运行效果如图 1.3 所示（图中抓取的只是其中的一个画面）。

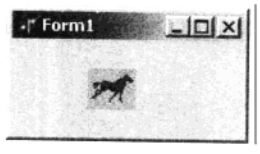


图 1.3 为窗体创建动画光标

〖设计思想〗

Delphi 窗体有一个属性为 Screen，该属性有子属性 Cursors，它属于光标数组类型，用来存放屏幕光标；Screen 还有一个子属性 Cursor，用来指定屏幕当前使用的光标。要添加自己的光标，只需将该光标资源加入数组 Cursors 中，然后指定 Cursor 为新加入的值即可。加入光标资源，可用 API 函数 LoadCursorFromFile（直接读光标文件）或 LoadCursor（读 Delphi 资源文件）来实现。

〖设计步骤〗

1．新建一个应用程序，添加全局常量 crMyCursor，其值可以取-22~0 范围之外、-32768~32767 范围之内的任何整数。

2．双击窗体，添加 OnCreate 事件的处理过程（见〖代码分析〗部分）。

〖代码分析〗

本例主要代码如下：

```
...
const
  //常量 CrMyCursor 此处取值为 1，也可以取上述指定范围内的任意值
  crMycursor=1;
  ...

//处理窗体的 OnCreate 事件
procedure TForm1.FormCreate(Sender:TObject);
begin
  //往光标数组中添加资源
  Screen.Cursors[crMyCursor]:=LoadCursorFromFile('C:\WinNT\Cursors\
    Horse.ani');
  //更改当前屏幕光标
  Screen.Cursor:=crMyCursor;
end;
```

〖总结〗

本例比较简单，但一般说来，出于兼容性的考虑，应当将光标资源添加到程序的资源文件中，然后用 API 函数 LoadCursor 来装入，这是因为不同的系统，其光标资源的目录也不同。当然，如果将该光标文件拷贝到该应用程序的目录（或程序自建的任何目录）之下，也可以直接用函数 LoadCursorFromFile 来装入，而不必顾忌其兼容性。

案例 1.6 使窗体始终最小化

〖案例说明〗

本例制作的是一个始终保持最小化的窗体。由于窗体始终最小化，故此处没有给出效果图，读者可以自行运行配套光盘中的相应程序，以验证其效果。

〖设计思想〗

Delphi 的窗体有一个属性为 WindowState，如果将其设置为 wsMinimized，则窗体就可以最小化，但此时用户可以恢复或最大化该窗体。要使窗体始终保持最小化，程序必须干预系统对用户恢复或最大化该窗体动作的响应。由于窗体恢复或最大化时要向系统发送消息 WM_QUERYOPEN，因此程序只需拦截此消息并告诉系统：此消息已由程序处理完毕，不用再处理，则窗体就永远保持最小化了。

〖设计步骤〗

1．新建一个应用程序，将窗体的属性 WindowState 设置为 wsMinimized。

2．在窗体的 protected 部分加入消息处理过程 WMQueryOpen 的声明：

```
procedure WMQueryOpen(var Msg:TWMQueryOpen);message WM_QUERYOPEN;
```

3．编写消息处理过程 WMQueryOpen（见〖代码分析〗部分）。

〖代码分析〗

本例主要代码只有一句：

```
procedure TForm1.WMQueryOpen(var Msg:TWMQueryOpen);
begin
  //告诉系统该消息已被处理（采用欺骗手段）
  Msg.Result:=0;
end;
```

〖总结〗

同样，可以制作始终最大化的窗体。只是此时处理的消息不是想当然的 WM_QUERYCLOSE，因为此消息是用户关闭程序时触发的，最大化窗体时并不能触发此消息。所以要另想办法，这将在下一案例中详细介绍。

案例 1.7　使窗体始终最大化

〖案例说明〗

本例制作的是一个始终保持最大化的窗体。当用户单击程序的“最小化”或者“恢复”这两个系统菜单项时，就会弹出“不能最小化”对话框，如图 1.4 所示。

图 1.4　“不能最小化”对话框

〖设计思想〗

Delphi 的窗体有一个属性为 WindowState，如果将其设置为 wsMaximized，则窗体就可以最大化，但此时用户可以恢复或最小化该窗体。要使窗体始终最大化，程序必须干预系统对用户恢复或最小化该窗体的响应。由于窗体恢复或最小化时要向系统发送消息 WM_SYSCOMMAND，因此程序只需拦截此消息并告诉系统：此消息已由程序处理完毕，不用再处理，则窗体就永远保持最大化了。

〖设计步骤〗

1．新建一个应用程序，将窗体的属性 WindowState 设置为 wsMaximized。