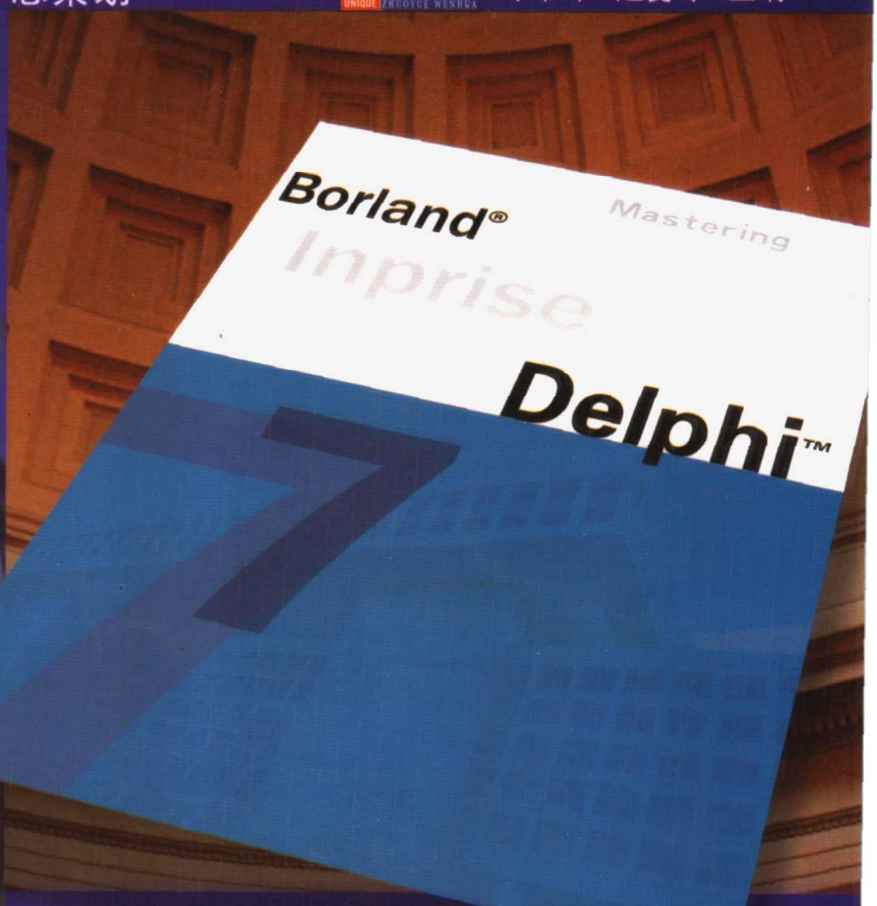
计算机教育图书研究室
Computer Education Books
总策划

卓越文化 毕维峰 赵爱玲 主编



Borland®

Delphi™ 7

编程实例与技巧

航空工业出版社

Delphi7编程实例与技巧

计算机教育图书研究室
Computer Education Books 总策划

主　编　毕维峰　赵爱玲
编　委　崔慧勇　林　峰
任立功　侯丽敏

航空工业出版社

内 容 提 要

Inprise 公司推出的企业级开发工具 Delphi 具有面向对象编程、支持团队开发、提供工程管理以及对数据库的良好支持等多种特性，它所提供的大量可重用组件和用户自建模板技术，极大地提高了应用系统的开发速度。

本书通过诸多典型的编程实例讲解了界面编程、图形图像编程、文件操作、多媒体技术、系统功能、数据库编程、网络编程以及游戏制作的各种方法与技巧，使读者在实际操作中领会使用 Delphi 7 编程的基本思路与方法，从而达到学以致用的目的。在最后一章，编者还讲解了几个非常具有典型性的综合实例，将对读者进行实际编程提供一些借鉴和帮助。

本书结构合理，内容翔实，知识新颖，难度适中，主要面向初、中级 Delphi 用户，也可作为 Delphi 高级用户的参考手册。

图书在版编目（CIP）数据

Delphi 7 编程实例与技巧 / 毕维峰等主编. —北京：航空工业出版社，2003.12
ISBN 7-80183-216-7

Ⅰ.D… Ⅱ.毕… Ⅲ.软件工具—程序设计 Ⅳ.TP311.56

中国版本图书馆 CIP 数据核字（2003）第 083244 号

航空工业出版社出版发行
（北京市安定门外小关东里 14 号 100029）
北京市燕山印刷厂印刷　　全国各地新华书店经售
2003 年 12 月第 1 版　　2003 年 12 月第 1 次印刷
开本：787×1092　1/16　　印张：28.5　　字数：478 千字
印数：1-6000　　定价：36.80 元

前　言

Inprise 公司推出的企业级开发工具 Delphi 具有面向对象编程、支持团队开发、提供工程管理以及对数据库的良好支持等多种特性，它所提供的大量可重用组件和用户自建模板技术，极大地提高了应用系统的开发速度。因此，业界如此形容 Delphi：真正的程序员用 C，聪明的程序员用 Delphi。自从 Delphi 面世以来，深受广大程序员的青睐。

Delphi 7 的基础语言为 Pascal，它继承了 Pascal 语言严谨的优点，如代码结构清晰、可读性强和代码执行效率高等。Delphi 从 1.0 版本发展到 Delphi 7，在性能上有了很大的提高，其中包括数据库的体系结构、ActiveX 控件的开发、Web 应用程序和安全线程等，从而使得 Delphi 的功能更加强大，使用起来也更加方便、灵活，在很大程度上提高了应用程序的开发效率。

Delphi 7 为开发人员提供了极为方便的技术和服务，不论是初学 Windows 程序设计的学生，还是开发关键性任务、大型应用系统的企业开发人员，Delphi 完整的产品线都能满足其不同的需求，使之可以很容易地开发出实用的产品。

Delphi 7 轻松跨越 Windows/Linux 双平台，高效开发面向对象企业级应用，Borland Delphi 6 是当前 Windows 平台上第一个全面支持最新 Web 服务的快速开发工具。无论是企业级用户，还是个人开发者，都能够利用 Delphi 7 轻松、快捷地构建新一代电子商务应用。

本书共分为九篇，通过诸多典型的编程实例讲解了界面编程、图形图像编程、文件操作、多媒体技术、系统功能、数据库编程、网络编程以及游戏制作的各种方法与技巧，使读者在实际操作中领会使用 Delphi 7 编程的基本思路与方法，从而达到学以致用的目的。在最后一篇，编者还讲解了几个非常典型性的综合实例，将对读者进行实际编程提供一些借鉴和帮助。

本书由计算机教育研究室总策划，毕维峰、赵爱玲主编，同时参加编写的还有崔慧勇、林锋、任立功、侯丽敏、吴闯、芦淑珍、秦志敏、杜同顺、董金波等多位老师，在此向他们表示衷心的感谢！本书主要针对初、中级 Delphi 用户，也可作为 Delphi 高级用户的参考手册。由于时间仓促，书中难免存在不足和疏漏之处，恳请广大读者批评指正。

编　者

2003 年 10 月

目 录

第一篇 界面编程

界面编程几乎是任何一个实用编程工具都必须涉及的内容，界面编制的好坏与否在相当大的程度上影响用户的心情和使用该软件的兴趣。Delphi 在进行界面编程方面有着十分丰富的技巧和方法，它为界面编程提供了十分方便、丰富而又有效的控件和函数的支持。

本篇讨论了使用 Delphi 7 在窗口标题栏上设置自绘按钮、定制提示窗口、实现不规则窗口和制作溅射屏幕等方面的应用，同时也讲解了 Delphi 7 编程中的一些基本问题，为以后的学习打下基础。

实例 1 第一个应用程序

为了适应不同层次的读者，在本实例中编写了一个简单的应用程序，如图 1-1 所示。程序运行后，单击【点击我】按钮，将会在窗口中出现“这是我的第一个 Delphi 程序”字样，单击【退出】按钮将会退出程序。

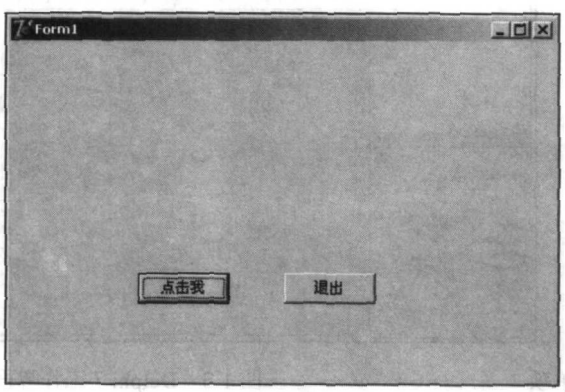


图 1-1 程序的运行结果

Delphi 实际上是 Object Pascal 语言的一种版本，不过，它与传统的 Pascal 语言有着天壤之别。一个 Delphi 程序首先是一个应用程序框架，而这一框架正是应用程序的“骨架”。在框架上即使没有附着任何东西，仍可以严格地按照设计运行。默认的应用程序是一个空白的窗体（Form），它可以运行，得到一个空白的窗口，这个窗口具有 Windows 窗口的通用属性，即可以被放大、缩小、移动、最大化和最小化等，但却没有编写一行程序。因此可以说，应用程序框架通过提供所有应用程序共有的东西，为应用程序的开发打下了良好的基础。Delphi 已经为用户做好了一切基础工作——程序框架就是一个已经完成的可运行的应用程序，只是不处理任何事情。用户需要做的，只是在程序中加入完成所需功能的代码而已。

在了解了 Delphi 的基本原理之后，用户就可以进行应用程序的开发了。Delphi 应用程序开发的基本步骤为：

（1）建立窗体；
（2）在窗体上添加控件；
（3）检查和设定对象属性；
（4）编写响应事件处理程序；
（5）保存文件；
（6）编译、运行程序。

编程步骤

启动程序

启动程序的具体操作步骤如下：

（1）选择【开始】|【程序】|Borland Delphi 7|Delphi 7 菜单项，如图 1-2 所示。即可启动 Delphi 7 应用程序，启动后的 Delphi 7 工作界面如图 1-3 所示。

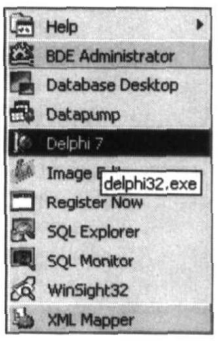


图 1-2 选择 Delphi 7 菜单项

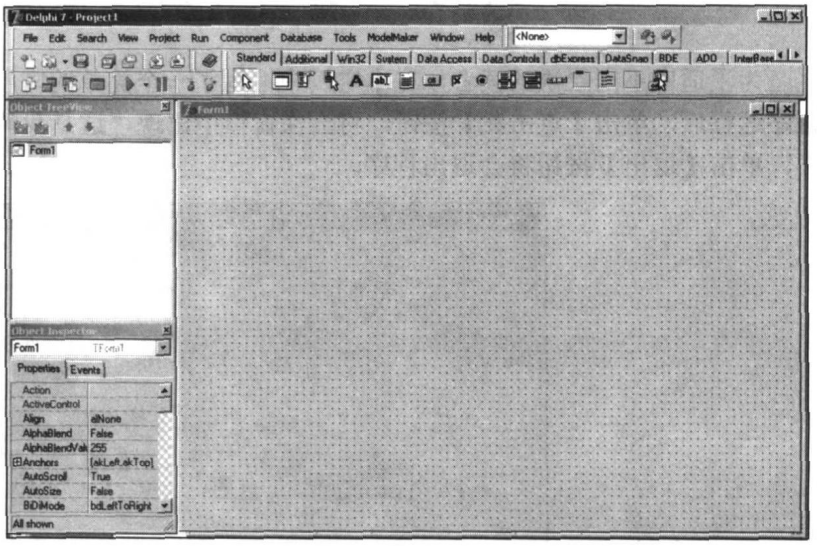


图 1-3 Delphi 7 工作界面

（2）此时程序自动生成一个窗体，名称为 Form1（参见图 1-3）。如果用户不小心关掉了该窗体，可以通过选择 File|New|Application 菜单项创建一个新的窗体文件。

此时 Delphi 自动生成的代码如下：

```
unit Unit1;          //单元文件名
interface            //这是接口关键字，用它来标识文件所调用的单元文件
uses                 //使用的公共单元
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs;
type                 //这里定义了程序所使用的控件以及控件所对应的事件
  TForm1 = class(TForm)
  Private            //定义私有变量和私有过程
    { Private declarations }
  public             //定义公共变量和公共过程
    { Public declarations }
```

```
  end;
var                    //定义程序使用的公共变量
  Form1: TForm1;
implementation //程序代码实现部分
{$R *.dfm}
end;
```

这些自动生成的代码主要是定义一些程序的基本信息。其中，第一行指明单元的名称为 Unit1.pas，该单元将以 Unit1.pas 为名称保存到存储器上；interface 下面主要是程序的接口部分；uses 下面定义了程序要使用的系统单元，如 Windows、Messages 等单元，这些单元将在后面介绍；type 部分主要用来声明程序所使用的公共变量以及系统控件等，这些将在第二篇进行详细介绍；implementation 部分是程序的具体实现内容，是需要程序员来编写代码的部分。

创建程序界面

创建程序界面的具体操作步骤如下：

（1）在控件面板的 Standard 选项卡中单击 Button 控件，如图 1-4 所示。

图 1-4　Standard 选项卡

（2）在窗体设计窗口中单击鼠标左键，Button 控件就会出现在窗体中。用同样的方法再向窗体中添加一个 Button 控件和一个 Label 控件。此时，窗体中的控件布置如图 1-5 所示。

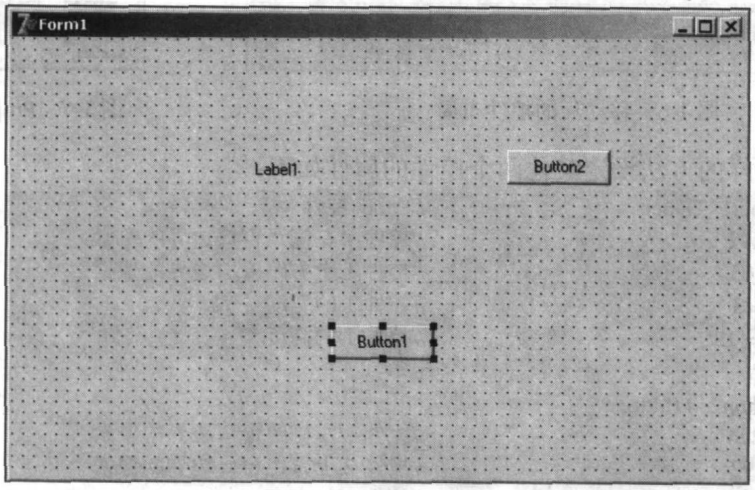


图 1-5　Form1 窗体中的控件

Delphi 提供了“提示”功能，在选择控件时，如果鼠标指针在某个控件上方停留几秒钟，在鼠标指针旁边就会出现一个小的提示框，显示该控件的作用。如果用户不知道哪个控件是 Shape 控件，就可以利用这个功能找到该控件。

设定控件属性

现在摆在用户面前的已经是一个应用程序界面的雏形了。用户也许并不满意这个界面，因为所有的按钮上面都是系统默认的名称，如 Button1、Button2 等，并且也没有提供键盘快捷方式，这对于一个应用程序的界面来说并不合适。如何改变这种情况呢？这里可以在对象监视器中对这个界面中控件的属性进行更改，具体操作方法如下：

（1）拖动控件到合适的位置。具体的方法是：用鼠标左键单击要移动的控件，使其变为选中状态（控件四周将显示 8 个黑色小矩形），然后单击并拖动该控件到合适位置即可。最后各控件的位置如图 1-6 所示。

（2）选中 Button1 控件，在对象监视器的 Properties 选项卡中将会出现和 Button1 控件相关的属性。单击 Caption 属性后面的编辑框，如图 1-7 所示。此时编辑框中的文字变为可编辑状态，向其中输入文字即可改变按钮的显示名称，这里改为“点击我”。

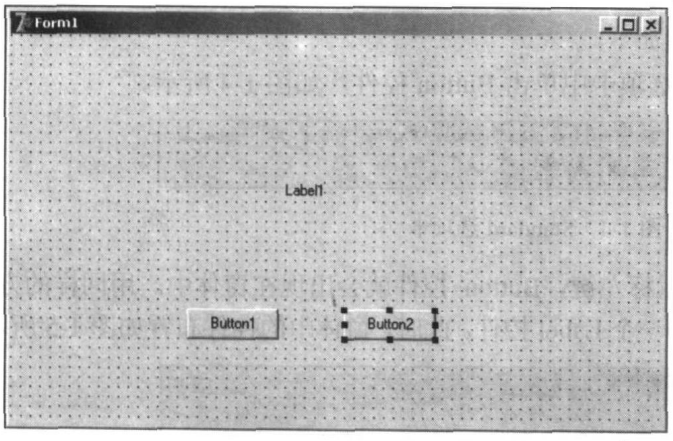


图 1-6　移动后的控件布局

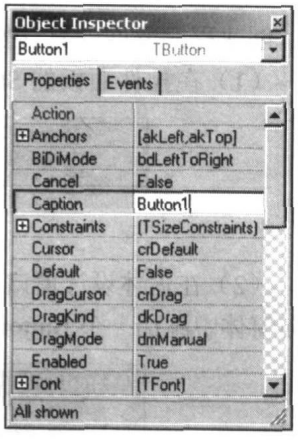


图 1-7　Properties 选项卡

（3）用同样的方法设定其他两个控件的属性如下：

```
object Label1: Tlabel                    //Label 控件
  Left = 112                             //控件 1 左侧距窗口边界的像素
  Top = 105                              //控件 1 上方距窗口边界的像素
  Width = 3                              //控件 1 的宽度
  Height = 13                            //控件 1 的高度
end
object Button2: TButton
  Left = 224                             //控件 2 左侧距窗口边界的像素
  Top = 192                              //控件 2 上方距窗口边界的像素
  Width = 75                             //控件 2 的宽度
  Height = 25                            //控件 2 的高度
  Caption = '退出'                       //按钮显示的文字
end
```

编写程序代码

具体操作步骤如下：

（1）选中 Button1 控件，单击对象监视器中的 Events 选项卡，双击 OnClick 事件右侧

的编辑框（如图 1-8 所示），进入代码编辑窗口，在窗口左侧显示了程序框架。在窗体上双击 Button1 控件，也能进入代码编辑窗口，如图 1-9 所示。

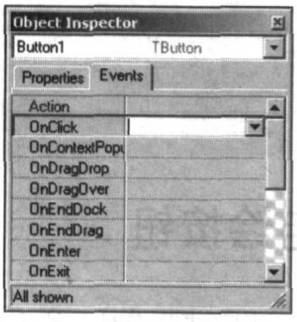


图 1-8　选择响应事件

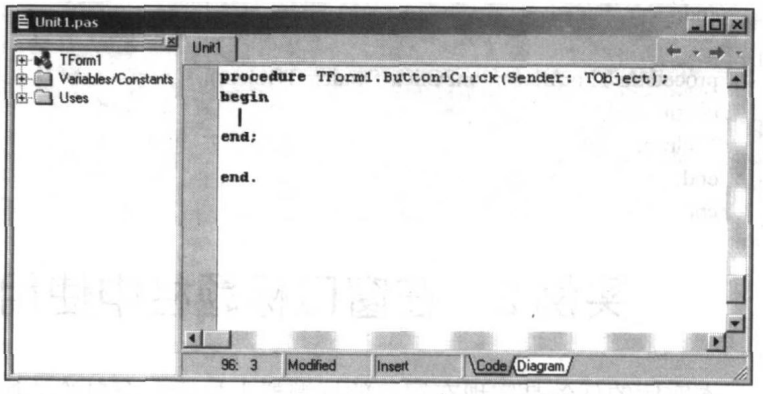


图 1-9　代码编辑窗口

（2）在代码编辑窗口中输入如下代码：

```
Label1.Caption:='这是我的第一个 Delphi 程序';
```

这段程序表示设定 Label1 控件的 Caption 属性为字符串“这是我的第一个 Delphi 程序”，当用户单击【点击我】按钮时，Label 控件中就会出现这段文字。

（3）用同样的方法设定【退出】按钮的 OnClick 事件。其代码如下：

```
procedure TForm1.Button2Click(Sender: TObject);
//退出程序
begin
  Close;
end;
```

此程序完整源代码如下：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Caption:='这是我的第一个 Delphi 程序';
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  close;
end;
end.
```

实例 2　在窗口标题栏中使用自绘按钮

本实例的任务是实现在窗口的标题栏上显示一个自绘的按钮，当它被单击时，能实现我们想要实现的功能。

本实例主要介绍一些 Delphi 7 编程的基本知识，例如，Delphi 7 的界面和操作、基本的程序编写过程、如何创建程序窗口、如何运行程序、消息处理、Windows SDK 函数和 inherited 关键字等。

在窗口标题栏中显示自绘按钮并不难，只需通过一些简单计算就可确定要把自绘按钮显示在标题栏的位置，本例把它放在三个常规按钮的左边。但是，赋予它一定的功能则稍微复杂一些，关键是响应两个消息——WM_NCPaint 和 WM_NCLButtonDown。

Delphi 是一门消息驱动的面向对象的语言。所谓消息驱动就是：用户动作、应用程序动作和一些系统消息等被传给 Windows，Windows 再把消息发送到相应应用程序中的相关处理函数。Delphi 把一些常用的消息处理函数作了包装，在程序中可以直接使用 Delphi 控件的 Event 事件属性处理函数来处理事件。还有一些消息则不存在于控件的 Event 事件属性列表中，它们必须由程序员明确地在程序中指出一个函数来专门处理，并且这个函数的写法有一定的规则。

本实例中的自绘按钮的制作方法是绘制一个矩形框，然后在矩形框中输入一个字符。由于矩形框大小的限制，字符不能完全显示出来，但是自绘按钮看起来已经不单调了。

至于消息的响应方面，WM_NCPaint 消息负责绘制按钮，当 WM_NCActivate 和 WM_SetText 消息到来时，窗口的非 Client 区域将被重绘，因此在这两个消息的相应区域中也需要绘制按钮。这样，把绘制按钮的函数单独列出供上述三个消息的处理函数调用。

另外，还需要判断鼠标指针是否在该按钮上单击，这个判断也是通过一个 WM_CHitTest 消息处理函数来得到。这个消息会把单击鼠标时鼠标指针的位置传给处理函数，从而可以判断这个位置是否在按钮的范围内。

最后，要赋给按钮的功能将被放在 WM_NCLButtonDown 事件的处理函数中，这个处理函数首先引用 WM_CHitTest 消息处理函数的结果来判断当单击鼠标时，鼠标指针是否在自绘按钮上，如果在，则执行赋予它的功能。

上面谈到了好几个消息，读者无需过多地了解它们，只需了解它们在什么时候触发即可。具体的使用方法可以参考程序，并且，以后可遵照程序中的写法去使用它们。所有的 Windows 消息都使用 WM_开头。下面即是对它们的详细介绍：

● WM_NCPaint：当应用程序需要重绘窗口时，它就向 Windows 发送 WM_NCPaint 消息。该消息的 wParams 成员是一个 HRGN 类型的值，变量名是 hrgn，这个变量代表需要 Windows 重绘的窗口中的矩形区域。如果应用程序需要自己处理这个消息，处理函数的返回值应该是 0。

● WM_NCActivate：当一个窗口被激活或是失去焦点时，它的非 Client 区域（一般是指标题栏、菜单栏等）需要改变，这时它就向 Windows 发送 WM_NCActivate 消息。该消息的 wParams 成员是一个 BOOL 类型的值，变量名是 fActive。当窗口区的非 Client 区域需要改变时，这个变量用来代表窗口的激活或是非激活的状态。当窗口被激活时，该变量的值为 True，否则该变量的值为 False。如果需要自己处理该消息，那么当 fActive 的值为 False 时，可用 Windows 的默认处理函数来处理它，这时处理函数的返回值必须是 True，即意味着此消息还没有被处理完，Windows 将会自动调用默认的处理函数来处理它；如果不希望 Windows 的默认处理函数处理它，那么用户可以使处理函数返回 False，这种情况下，如果程序中没有设定，那么窗口的非 Client 区域将不会失去焦点。当 fActive 的值为 True 时，返回值将被忽略，非 Client 区将被重绘。

● WM_SetText：应用程序需要设置窗口标题时，它将发送 WM_SetText 消息给 Windows。该消息的 wParams 成员必须是 0，而它的 lParams 成员则是一个 LPCTSTR 类型的值，变量名是 lpsz。它是一个指针，指向用来设置窗口标题的以空字符结束的字符串。当设置成功时，处理函数应返回 True。否则，如果该消息被发送至一个没有可编辑控件的组合框，则函数返回 CB_ERR；若该消息被发送至有可编辑控件的组合框，但是没有足够的空间显示则返回 CB_ERRSPACE；如果被发送至列表框，但没有足够的空间显示，则函数返回 LB_ERRSPACE；如果被发送至一个可编辑控件，但没有足够的空间显示，则函数返回 False。

● WM_NCHitTest：当鼠标被按下或是弹起时，应用程序将会给 Windows 发送 WM_NCHitTest 消息。该消息的 lParams 成员的低位字是一个变量 xPos，用来指代鼠标指针的横坐标位置，高位字是变量 yPos，用来指代鼠标指针的纵坐标位置（注意，这里的坐标都是相对于显示器屏幕左上角而言的）。Windows 默认该消息的处理函数是 DefWindowProc 函数（它是所有 Windows 标准消息的默认处理函数），处理该消息的返回值可以是表示鼠标指针位置的值，如表 2-1 所示。

表 2-1　WM_NCHitTest 消息处理函数的返回值

返回值	区　域	返回值	区　域
HTBORDER	不能改变大小的窗口的边界	HTLEFT	窗口的左边界
HTBOTTOM	窗口的下边界	HTMENU	窗口的菜单栏

HTBOTTOMLEFT	窗口的左下角	HTNOWHERE	在屏幕上或是窗口的分隔线上
HTBOTTOMRIGHT	窗口的右下角	HTREDUCE	最小化按钮
HTCAPTION	窗口的标题栏	HTRIGHT	窗口的右边界
HTCLIENT	窗口的 Client 区域	HTSIZE	可改变大小的框（同 HTGROWBOX）
HTERROR	在屏幕上或是窗口的分隔线上	HTSYSMENU	系统菜单或是子窗口的关闭按钮
HTGROWBOX	可改变大小的框（同 HTSIZE）	HTTOP	窗口的上边界
HTHSCROLL	水平滚动条	HTTOPLEFT	窗口的左上角
HTTOPRIGHT	窗口的右上角	HTVSCROLL	竖直滚动条
HTTRANSPARENT	被另一窗口覆盖的窗口	HTZOOM	最大化按钮

HTERROR 和 HTNOWHERE 的作用范围一致，不同点在于 DefWindowProc 函数会为 ERROR 产生一个蜂鸣声，警告产生了一个错误。

● WM_NCLButtonDown：当在窗口的非 Client 区域单击鼠标时，应用程序将给 Windows 发送 WM_NCLButtonDown 消息。该消息的 wParams 成员是一个 INT 类型的变量，变量名为 nHittest，该值用来表示 DefWindowProc 函数处理上面的 WM_NCHitTest 消息时的返回值（即上述表 2-1 中众多值之一）。消息的 lParams 成员被一个预定义的宏 MAKEPOINTS 转换成一个 POINTS 结构的变量 pts，它包含了鼠标指针的横坐标和纵坐标的信息。当然，这里的坐标也是基于屏幕坐标而言的。

编程步骤

创建新项目

启动 Borland Delphi 7，对自动新建的项目选择 File | Save All 菜单项，弹出 Save Unit1 As 窗口，将该单元文件保存为 titlebtn.pas，然后弹出 Save Project1 As 窗口，将该项目文件保存为 titlebutton.dpr。

创建程序窗口

在控件面板的 Additional 选项卡（如图 2-1 所示）中将 Image 控件放到窗口上，具体过程是：单击 Image 控件，然后在程序窗口的任意位置单击鼠标左键即可。

打开 Object Inspector 对话框，如图 2-2 所示，从中设置 Form1 和 Image1 控件的属性：

```
Form1:
  Left = 192
```

```
    Top = 107
    Width = 388
    Height = 319
    Caption = '自绘按钮'
  Image1:
    Left = -24
    Top = 107
    Width = 401
    Height = 313
    Stretch = True
```

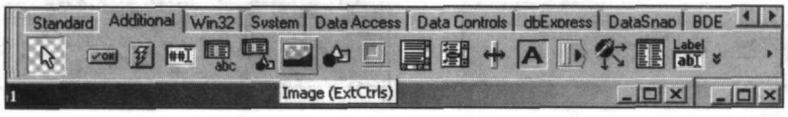


图 2-1　在 Additional 选项卡中选择一个 Image 控件

然后单击 Picture 属性后面的按钮，弹出如图 2-3 所示的对话框。

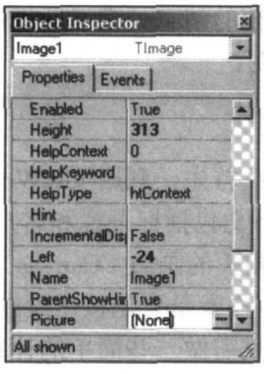


图 2-2　Object Inspector 对话框

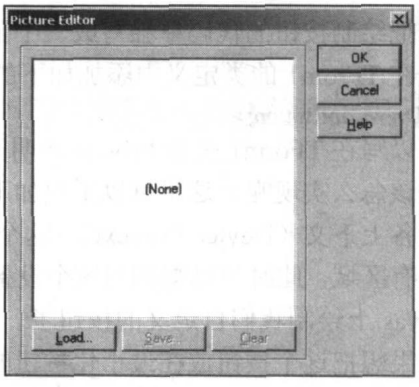


图 2-3　Picture Editor 对话框

单击 Load... 按钮，弹出如图 2-4 所示的 Load Picture 对话框。选择一幅图片，例如，选择 0003.jpg，则此时 Image1 控件的 Picture Editor（图片编辑器）对话框如图 2-5 所示。

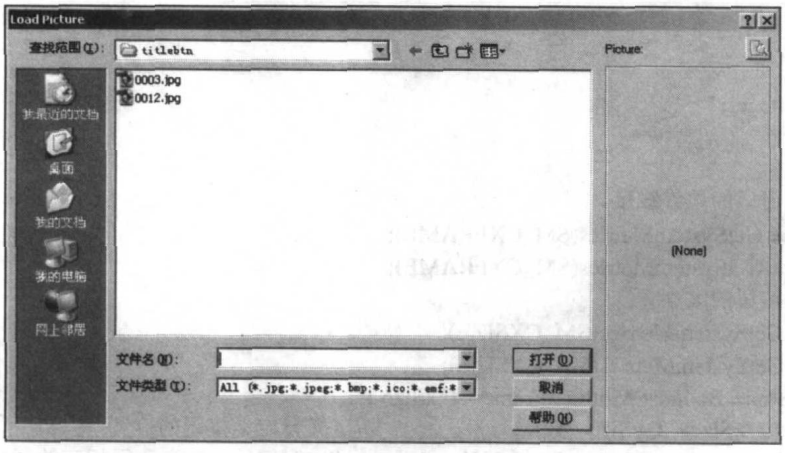


图 2-4　Load Picture 对话框

单击 OK 按钮，此时应用程序窗口如图 2-6 所示。

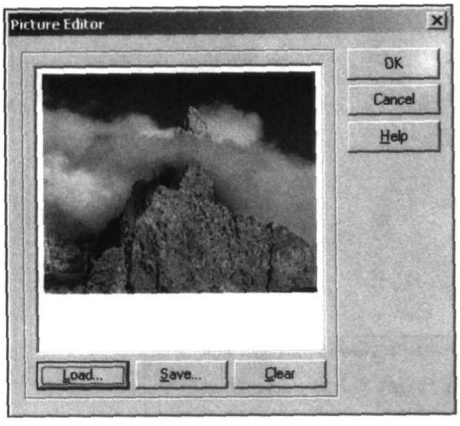


图 2-5 选择图片后的 Picture Editor 对话框

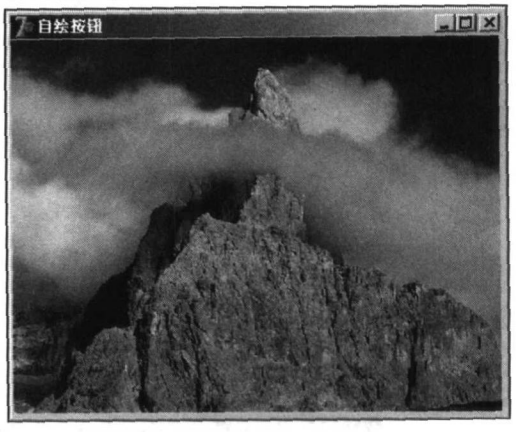


图 2-6 应用程序窗口

添加实现代码

前面提到把绘制按钮的代码单独写成一个函数供其他的消息处理函数调用，这里首先来实现它。先在 TFrom1 的类定义中添加如下的函数声明：

```
procedure DrawCaptButton;
```

此声明可以写在 TForm1 类的 Private 声明区域。

这个函数该怎么实现呢？这里可以采取如下的方法：首先，利用 GetWindowDC 函数获得窗口的设备上下文（Device Context），这个设备上下文包括标题栏和客户（Client）区域等窗口的所有区域，此时可以得到与这个设备上下文相关的画布（Canvas），这样，我们就可以在 Canvas 上绘制我们自定义的按钮了。剩下的工作就是要把按钮绘制在 Canvas 的什么地方，如果想把这个按钮放在最小化按钮的左边，只需做一些简单的计算就可以获得它的位置，然后再把它画上去，具体算法可以参看下面的程序代码。最后，一定要释放设备上下文。具体程序如下：

```
procedure TForm1.DrawCaptButton;
var
  xFrame,
  yFrame,
  xSize,
  ySize : Integer;
  R : TRect;
begin
  //可移动窗口的边界线度
  xFrame := GetSystemMetrics(SM_CXFRAME);
  yFrame := GetSystemMetrics(SM_CYFRAME);
  //标题栏按钮的大小
  xSize := GetSystemMetrics(SM_CXSIZE);
  ySize := GetSystemMetrics(SM_CYSIZE);
  CaptionBtn := Bounds(Width - xFrame - 4*xSize + 2,
  yFrame + 2, xSize - 2, ySize - 4);                    //定义自定义按钮的位置
  Canvas.Handle := GetWindowDC(Self.Handle);     //由窗口的 DC 来获得相应的画布
  Canvas.Font.Name := 'Symbol';
  Canvas.Font.Color := clBlue;
```

```
    Canvas.Font.Style := [fsBold];
    Canvas.Pen.Color := clYellow;
    Canvas.Brush.Color := clBtnFace;
    try
      DrawButtonFace(Canvas, CaptionBtn, 1, bsAutoDetect, False, False, False);
      //绘制按钮
      R := Bounds(Width - xFrame - 4 * xSize + 2,
          yFrame + 3, xSize - 6, ySize - 7);          //在按钮的内部定义一个能写字符的小矩形框
      with CaptionBtn do
        Canvas.TextRect(R, R.Left + 2, R.Top - 1, 'W')
    finally
      ReleaseDC(Self.Handle, Canvas.Handle);
      Canvas.Handle := 0
    end;
  end;
```

上面使用的 GetSystemMetrics 函数是一个重要函数，这里做一下说明。这是一个 Windows SDK（SDK 就是 Software Development Kit，也就是软件开发工具包的简称）函数，它的原型是：

int GetSystemMetrics(int nIndex)

它的作用是返回一个 Windows 显示元素的大小或系统设置，其单位是像素。其参数 nIndex 用来指代相应的元素或是系统设置。一般以 SM_CX 开头的值都是指宽度，以 SM_CY 开头的值都是指高度。nIndex 的可取值有很多，这里就不一一说明了，有兴趣的读者可以参考 Delphi 的 Windows SDK 帮助。程序中用到的 SM_CXSIZE 和 SM_CYSIZE 是用来指出窗口标题栏上以像素为单位的按钮的宽度和高度，SM_CXFRAME 和 SM_CYFRAME 用来指出可调整大小的窗口的边界厚度，其中 SM_CXFRAME 指代窗口的垂直边界的厚度（或者宽度），SM_CYFRAME 指代窗口的水平边界的厚度（或者高度）。

设置按钮的位置时使用了 Bounds 函数，这个函数原型如下：

function Bounds(Aleft,Atop,Awidth,Aheight:Integer):TRect;

它的作用是根据参数返回一个 TRect 类型的值，其中，Aleft 和 Atop 指出矩形相对于窗口左上角的横坐标和纵坐标，Awidth 和 Aheight 指出矩形的宽度和高度。

把自绘按钮的横坐标设置为 Width - xFrame - 4*xSize + 2，这相当于在整个窗口的宽度上减去四个按钮的宽度（三个系统按钮和自定义的按钮本身），再减去窗口的右边界的厚度，要加 2 是因为第三个参数也就是自定义的按钮本身的宽度是 xSize-2，这意味着减去 4*xSize 的时候多减了 2，所以现在补回来。至于其他的几个参数也不难理解，这里就不一一介绍了。

设置好按钮的位置后，就可使用 DrawButtonFace 函数在该位置上绘制按钮。此函数的原型是：

function DrawButtonFace(Canvas: TCanvas; const Client: TRect; BevelWidth: Integer; Style: TButtonStyle; IsRounded, IsDown, IsFocused: Boolean): TRect;

它的作用是用来绘制一个标准的按钮，而且会绘制出按钮的边界和背景。Canvas 参数是用来绘制按钮的场所，Client 参数用来指出 Canvas 上的一个区域，它被用来绘制按钮的 Client 区域，BevelWidth 参数用来指出按钮的外斜边界的宽度，Style 参数用来指出按钮风格，它的值一般是 bsAutoDetect，IsRounded 参数用来指出按钮是否为圆角矩形，IsDown 参数用来指出按钮是否为凹下去的，IsFocused 参数用来指出按钮是否可以接受键盘焦点。