万水 Delphi技术丛书

马健兵 朱 亮 张 雷 等编著

Delphi 7.0 应用编程实例精解

中国水利水电出版社
www.waterpub.com.cn

万水 Delphi 技术丛书

Delphi 7.0 应用编程实例精解

马健兵　朱 亮　张 雷　等编著

中国水利水电出版社

内 容 提 要

Delphi 7.0 是 Borland 公司最新推出的可视化面向对象开发工具，它使得各种应用程序的开发简单易行。本书用详尽的实例全面介绍了 Delphi 在各方面的编程应用。全书共分 11 篇，主要包括 Delphi 的界面编程、窗口技巧、汉字处理、游戏制作、图像处理、影音播放、数据库、网络编程、文件操作、系统功能以及 Delphi 的综合技巧等方面，涉及了 Delphi 应用方面的大部分内容。针对每一个实例，给出了实例说明、编程思路、步骤、实例效果和小结 5 个部分，具有很高的实用价值。

本书内容丰富、实例翔实、讲解精辟、深入浅出，适用于使用 Delphi 开发应用程序的用户，尤其适合已经有一定 Delphi 编程基础的读者。

图书在版编目（CIP）数据

Delphi 7.0 应用编程实例精解/马健兵等编著. —北京：中国水利水电出版社，2003

（万水 Delphi 技术丛书）

ISBN 7-5084-1513-2

Ⅰ. D… Ⅱ. 马… 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 038579 号

书　　名	Delphi 7.0 应用编程实例精解
作　　者	马健兵　朱亮　张雷　等编著
出版、发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@public3.bta.net.cn（万水） sale@waterpub.com.cn 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	787×1000 毫米　16 开本　24.125 印张　548 千字
版　　次	2003 年 7 月第一版　2003 年 7 月北京第一次印刷
印　　数	0001—5000 册
定　　价	38.00 元（含 1CD）

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

前　言

Delphi 是 Borland 公司推出的一种可视化面向对象开发工具。它的基础是著名的 Object Pascal 语言。Delphi 的使用十分方便，一个有 Pascal 语言基础的程序员可以很快地对 Delphi 上手。Delphi 也是各种语言中编译速度最快的语言。

Delphi 的功能十分强大，从功能方面说，它和 VC 可以做到不相上下，就使用来说，Delphi 要比 VC 简单。自 Borland 公司推出 Delphi 以来，它就在全世界范围内获得了程序员的热烈欢迎。目前 Delphi 的最高版本是 Delphi 7.0，其功能是十分完善的。

本书通过具体的实例，深入浅出地介绍了使用 Delphi 语言进行编程的基本方法和技巧，内容丰富。全书内容共分 11 篇，包括 50 个实例程序，从不同的方面介绍了 Delphi 的编程方法和应用：界面编程、窗口技巧、汉字处理、游戏制作、图像处理、影音播放、数据库、网络编程、文件操作、系统功能、综合应用。

为了使读者能从零开始学习 Delphi，本书的前面几例讲述得非常详尽，可以保证读者在不懂 Delphi 的情况下，也能按照本书的指导一步一步地把实例效果编写出来。

本书的每一个实例又包括实例说明、编程思路、步骤、实例效果和小结 5 个部分。实例说明部分说明该实例想要实现的效果和需要处理的一些问题，编程思路部分说明实现该实例的思路和算法，步骤部分则是逐步具体说明怎样实现该实例，实例效果则是展示该实例应呈现的效果，读者可以自己编出程序后与实例效果对照，小结则是讲述该实例编写过程中的一些体会和值得注意的问题。本书结构合理，讲解透彻，易于读者掌握。

本书所附光盘的内容包括了开发实例的所有程序源代码和用到的相应资源，所有程序代码都在 Delphi 7.0/Windows XP Professional 环境下编译通过。书中所有实例程序的代码都经过了严格的调试和测试，读者只要按照书中的步骤，最终一定能够圆满地完成程序。

本书主要由马健兵、朱亮、张雷编写，另外罗心晶、吉庆祥、苏瑞、王龙、索双有、袁博、陈海亮、关宁、姜仁武、贾全芳、王嘉宁、刘松涛、盛建武、王为之、杜建峰、徐伟、胡鹏、雷文波、韩中领、贾君琳、朱易昕、娄俊杰、郑东晖、宋怡、胡晓冰、杨现青、陈江龙、王亮等同志也参加了本书的整理和编写工作。本书编写过程中，得到许多领导和同仁的大力支持和帮助，在此一并表示感谢。

由于时间仓促及作者能力有限，书中难免有欠妥之处，恳请广大读者批评指正。

编者

2003 年 4 月

目　　录

前言
第一篇　界面编程.....1
实例一　在窗口标题栏中使用自绘按钮.....1
实例二　在 Delphi 中定制提示窗口.....11
实例三　编程实现不规则窗口.....18
实例四　用 Delphi 制作溅射屏幕.....28
第二篇　窗口技巧.....36
实例五　多技巧窗口.....36
实例六　文字倾斜的窗口.....44
实例七　简单工具栏的订制.....49
实例八　实现 Dockin 效果.....52
第三篇　汉字处理.....60
实例九　在 Delphi 中实现输入法切换.....60
实例十　获取汉字的拼音索引字母.....65
实例十一　实现界面无闪烁多语言切换.....70
实例十二　金额大写转换程序.....77
实例十三　汉字到拼音转换小程序.....82
第四篇　游戏制作.....90
实例十四　贪吃蛇游戏.....90
实例十五　俄罗斯方块.....99
第五篇　图像处理.....123
实例十六　简单的图像滤镜运算器.....123
实例十七　图像文件的压缩.....140
实例十八　快捷方便的图像浏览器.....145
实例十九　双缓冲技术实现动画.....165
第六篇　影音播放.....171
实例二十　打造自己的 RM 播放器.....171
实例二十一　综合多媒体播放器.....178
实例二十二　简单的录音机.....186
第七篇　数据库.....192

实例二十三　显示彩色数据表格 192
实例二十四　Access 数据库中的搜索 197
实例二十五　在 Access 数据库中添加/删除项 201
实例二十六　SQL Server 的连接和初步使用 207
实例二十七　用 TreeView 来显示数据库信息 215
第八篇　网络编程 **223**
实例二十八　获得局域网的计算机列表 223
实例二十九　实现 Ping 功能 232
实例三十　网络资源树形浏览 238
实例三十一　动态改变 DNS 地址 243
第九篇　文件操作 **249**
实例三十二　简单的文件编辑器 249
实例三十三　一个加密解密器 254
实例三十四　不同程序间的文件拖放 259
实例三十五　建立 Internet 快捷方式 262
实例三十六　做一个文件切割器 267
实例三十七　用 Delphi 将 IE 收藏夹导出为 HTML 文件 278
第十篇　系统功能 **285**
实例三十八　获得系统信息 285
实例三十九　创建控制面板的新项目 298
实例四十　程序显示在任务栏上 303
实例四十一　建立高精度计时器 312
实例四十二　剪贴板监控程序 318
第十一篇　综合应用 **325**
实例四十三　用 Delphi 编码实现程序自启动 325
实例四十四　制作有动画效果的按钮 330
实例四十五　公历日期转换为阴历 335
实例四十六　读写其他进程的内存 353
实例四十七　键盘鼠标动作记录与回放 358
实例四十八　屏幕保护程序的预览 363
实例四十九　动态链接库编程 369
实例五十　用 Delphi 制作简单桌面 373

第一篇　界面编程

界面编程是几乎任何一个实用软件都必须涉及到的内容，界面编制得好坏在相当大的程度上影响着用户的心情和使用该软件的兴趣。既然界面编程如此重要，Delphi 在进行界面编程方面当然也就有着十分丰富的技巧和方法，并为界面编程提供了十分方便、丰富而又有效的控件和函数支持。

本篇讨论在窗口标题栏上使用自绘按钮、定制提示窗口、实现不规则窗口以及制作溅射屏幕等方面的应用，同时也讲解了 Delphi 编程中的一些基本问题，为以后的学习打下了基础。

实例一　在窗口标题栏中使用自绘按钮

〖实例说明〗

本例的任务是实现在窗口标题栏上显示一个自绘的按钮，并且当它被点击时，赋予它我们想要实现的功能。

本例将主要介绍一些 Delphi 编程的基本知识，如 Delphi 的界面及其操作、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、inherited 关键字等。由于本例是本书的第一讲，因此会讲述得详尽一些。

〖编程思路〗

在窗口标题栏中显示自绘按钮并不难，只需通过一些简单的计算就可以确定我们想要把自绘按钮显示在标题栏的什么地方。在这里我们将把它放在 3 个常规按钮的左边。但是如果要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。

说明：我们都知道 Delphi 是一种消息驱动的面向对象语言。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，我们可以直接使用 Delphi 控件的 Event 事件属性处理函数来处理事件。但是还有一些消息则不存在于控件的 Event 事件属性列表中，它们必须由程序员在程序中明确指定由一个函数来专门处理，并且这个函数的写法有一定的规则。

这里的自绘按钮的做法是先画一个矩形框，再在矩形框中写入一个字符，由于矩形框大小的限制，字符不能完全显示出来，但是自绘按钮已经看起来不太单调了。

至于消息的相应方面，WM_NCPaint 消息负责绘制按钮。同样，当 WM_NCActivate 和 WM_SetText 消息到来时，窗口的非 Client 区域将要被重绘，因此在这两个消息的响应代码中也需要绘制按钮。这样，我们把绘制按钮的函数单独列出供上述 3 个消息的处理函数调用。

另外，需要判断鼠标是否点击在该按钮上，该判断也是通过一个 WM_CHitTest 消息处理函数得到的。这个消息会把鼠标的点击位置传给处理函数，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，该处理函数首先引用 WM_CHitTest 消息处理函数的结果来判断鼠标是否点击在自绘按钮上，如果是，则执行我们赋予它的功能。

说明：上面谈到了好几个消息，这里我们对它们进行介绍，读者无需过多地了解它们，只需要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，以后可以遵照程序中的写法去使用它们。所有的 Windows 消息都以 WM_开头。下面详细介绍各个消息。

- 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，那么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非激活状态。当窗口被激活时，该变量的值为 TRUE，否则该变量的值为 FALSE。如果你需要自己处理这个消息，则当 fActive 的值为 FALSE 时；如果你需要 Windows 的默认处理函数来处理它，那么你的处理函数的返回值必须是 TRUE，这意味着这个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，那么你可以使你的处理函数返回 FALSE，在这种情况下，如果你的程序中没有设定，则窗口的非 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，则返回 LB_ERRSPACE；如果被发送至一个可编辑控件，但没有足够的空间，则返回 FALSE。

- WM_NCHitTest：当鼠标被按下或是弹起时，应用程序将会向 Windows 发送 WM_NCHitTest 消息。该消息的 lParams 成员的低位字是一个变量 xPos，用来指代鼠标的横坐标位置，高位字是变量 yPos，用来指代鼠标的纵坐标位置。注意，这里的坐标都是相对于显示器屏幕的左上角而言的。处理该消息的 Windows 默认处理函数 DefWindowProc（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如表 1-1 所示。

表 1-1　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	菜单上
HTBOTTOMLEFT	窗口的左下角	HTNOWHERE	屏幕上或是窗口的分隔线上
HTBOTTOMRIGHT	窗口的右下角	HTREDUCE	最小化按钮上
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	屏幕上或是窗口的分隔线上	HTSYSMENU	系统菜单上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	最大化按钮上

注意：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，用于警告产生了一个错误。

- WM_NCLButtonDown：当鼠标点击在窗口的非 Client 区域时，应用程序将向 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，该值用来表示用 DefWindowProc 函数处理上面的 WM_NCHitTest 消息时的返回值（即表 1-1 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。

〖步骤〗

1. 创建新项目

启动 Borland Delphi 7.0，对自动新建的项目单击【File】|【Save All】菜单项，弹出【Save Unit1 As】窗口，将其保存为 titlebtn.pas，然后弹出【Save Project1 As】窗口，将其保存为 titlebutton.dpr。

2. 创建程序窗口

在如图 1-1 所示的 Additional 选项卡上选择一个 Image 控件放到窗口上，具体过程如下：

（1）单击图 1-1 所示的 Image 控件，然后在程序窗口上选一个位置单击即可在该窗口位置上放下 Image 控件。

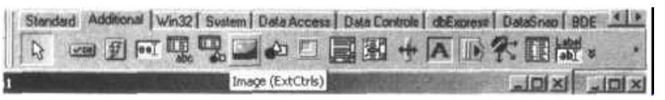

图 1-1 在 Additional 选项卡上选择一个 Image 控件

在如图 1-2 所示的 Object Inspector 中设置 Form1 和 Image1 的属性如下：

```
Form1:
  Left = 192
  Top = 107
  Width = 388
  Height = 319
  Caption = '自绘按钮'
Image1:
  Left = -24
  Top = -24
  Width = 401
  Height = 313
  Stretch = TRUE
```

（2）单击如图 1-2 所示的 Object Inspector 中 Image1 的 Picture 属性右边的…按钮，弹出如图 1-3 所示的对话框。

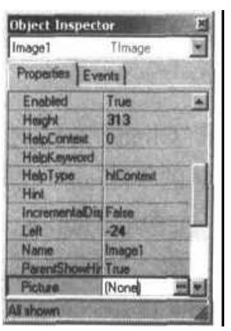

图 1-2 Image1 的设置图

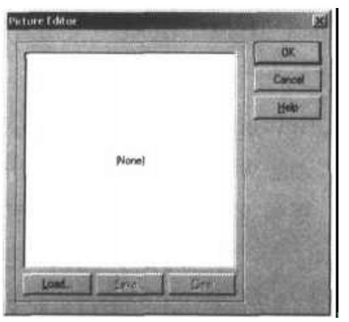

图 1-3 Image1 的 Picture 属性设置图

（3）单击图 1-3 中的 Load 按钮，弹出如图 1-4 所示的装载图片对话框。选择一幅图片，例如选取 0003.jpg，则 Image1 的 Picture Editor（图片编辑器）对话框变成如图 1-5 所示。

图 1-4　装载图片对话框

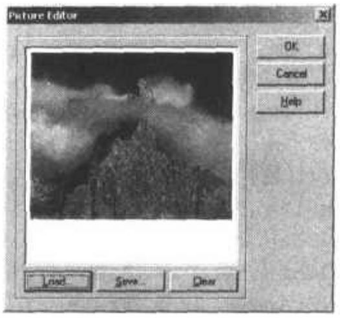

图 1-5　选择好图片后的图片编辑器

（4）单击图 1-5 的 OK 按钮，此时应用程序窗口应如图 1-6 所示。

图 1-6　程序窗口图

3. 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 的类定义中写入如下函数声明：

```
procedure DrawCaptButton;
```

此声明可以写在 TForm1 类的 Private 声明区域。

该怎么实现这个函数呢？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）（注：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），然后就可以得到与这个设备上下文相关的画布（Canvas），这样，就可以在 Canvas 上画图了，或者说，可以绘制我们自定义的按钮了。余下的工作就是，要把按钮绘制在 Canvas 的什么地方。想把这个按钮放在最小化按钮的左边，只需要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，一定要记得释放设备上下文。具体的程序如下：

```
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);
      //画按钮
      R := Bounds(Width - xFrame - 4 * xSize + 2,
         yFrame + 3, xSize - 6, ySize - 7);    //在按钮的内部定义一个小矩形文字区域
      with CaptionBtn do
        Canvas.TextRect(R, R.Left + 2, R.Top - 1, 'W')
    finally
      ReleaseDC(Self.Handle, Canvas.Handle);
      Canvas.Handle := 0
    end
  end;
```

上面使用了 GetSystemMetrics 函数，它是一个 Windows SDK（所谓 SDK 就是 Software Development Kit，也就是软件开发工具包的简称）函数，是一个比较重要的函数，这里做一个说明，它的原型是：

```
int GetSystemMetrics(int nIndex)
```

它的作用是返回一个 Windows 显示元素的大小或是系统设置，其单位是像素，其参数 nIndex 用来指代哪个元素或是系统设置。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 的可能值有很多，这里就不一一讲述了，有兴趣的读者可以参看 Delphi 的 Windows SDK 帮助。程序中用到的 SM_CXSIZE 和 SM_CYSIZE 用来指出窗口标题栏上的按钮以像素为单位的宽度和高度；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说是宽度，SM_CYFRAME 指代窗口的水平边界的厚度或者说是高度。

设置按钮的位置使用了 Bounds 函数，该函数的原型如下所示：

```
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 指出矩形的长度和高度。

不难看到，把自绘按钮的横坐标设置为 Width - xFrame - 4*xSize + 2，相当于在整个窗口的宽度上减去 4 个按钮的宽度（3 个系统按钮和自定义按钮本身），再减去窗口的右边界的厚度，至于为什么要加 2，原因很简单，因为第 3 个参数也就是自定义按钮本身的宽度是 xSize-2，这意味着在减去 4*xSize 时多减了 2，所以现在补回来。至于其他几个参数也不难理解。

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

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

它的作用是用来画一个标准的按钮。它将会画出按钮的边界和背景。Canvas 参数用来画按钮的场所。Client 参数用来指出 Canvas 上的一个区域，它被用来画按钮的 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是圆角矩形，IsDown 参数用来指出按钮是否是凹下去的，IsFocused 参数用来指出按钮是否可以接受键盘焦点。

Delphi 程序就是使用这个函数在 Canvas 上真正画出自定义按钮的。此时这个按钮仍然稍显单调，这时使用 Bounds 函数在自定义按钮内部再画出一个小矩形，然后使用 TextRect 函数在该小矩形内写一个 W 字符。当然，由于空间和位置所限，W 字符没有完全显示出来，但是使按钮不太单调的目的已经达到了。

最后使用 ReleaseDC 函数释放设备上下文。

现在绘制按钮的函数已经完成了，在消息处理函数中调用这个函数，如下所示。

```
procedure TForm1.WMNCPaint(var Msg : TWMNCPaint);
begin
  inherited;
  DrawCaptButton
end;
procedure TForm1.WMNCActivate(var Msg : TWMNCActivate);
begin
  inherited;
  DrawCaptButton
end;
procedure TForm1.WMSetText(var Msg : TWMSetText);
begin
  inherited;
  DrawCaptButton
end;
```

注意：不要忘了在上述的 3 个消息处理函数中写上 inherited 关键字。

在 Delphi 中，inherited 关键字的作用十分“关键”，可以说，如果没有它，Delphi 的面

向对象特性就要减少一半。

Inherited 关键字的作用涉及到面向对象的相关知识，由于篇幅所限，在这里不作详细说明，有兴趣的读者可以参考 Delphi 的帮助。Interited 关键字在上述 3 个消息处理函数中的作用是调用默认的消息处理函数，并且把 Msg 参数传给默认处理函数。

现在，可以考虑判断鼠标是否点击在自绘按钮上的代码了，也就是 WM_NCHitTest 消息处理函数，代码如下：

```
procedure TForm1.WMNCHitTest(var Msg : TWMNCHitTest);
begin
   inherited;
   with Msg do
     if PtInRect(CaptionBtn, Point(XPos - Left, YPos - Top)) then
       Result := htCaptionBtn
end;
```

这里首先利用 inherited 关键字调用默认处理函数，然后使用 Msg 的方法 PtInRect 来判断鼠标点击的位置是否在按钮的范围内。注意，这里使用了 CaptionBtn 变量，该变量必须是全局变量，因此，我们在 TForm1 类中的 Private 区域中写入变量声明 CaptionBtn: TRect。另外，这里的 htCaptionBtn 是一个常量，用来指出鼠标确实点击在自定义按钮上了。因此，需要在 implementation 下面添上一行常量声明，如下所示：

```
const
  htCaptionBtn = htSizeLast + 1;
```

在 implementation 中定义的常量可以在 implementation 实现的函数中使用，这里的 htSizeLast 是一个预定义的常量，它的值是 17。

下面就是实现自绘按钮功能的代码段。我们准备实现的功能是，当单击该按钮时，Image1 中的图片在两张图片之间切换。代码首先判断鼠标是否点击在按钮上，如果是，则根据目前显示的图片来决定现在显示哪一幅图片。这里我们使用了一个变量 id: Integer 来判断目前使用了哪一幅图片。如果 id=0，则表示正在显示图片文件 0003.jpg，id=1 则正在显示图片文件 0012.jpg。由于初始时，Image1 中的图片是 0003.jpg，因此，id 必须初始化为 0。我们将在后面的函数中实现这个初始化。这样的话，这个变量也是全局变量，因此我们把这个变量放在 implementation 的变量声明中，也就是在上面的常量声明后添上如下代码：

```
var
  id: Integer;
```

现在可以参看下面的代码了：

```
procedure TForm1.WMNCLButtonDown(var Msg : TWMNCLButtonDown);
begin
  inherited;
  if(Msg.HitTest = htCaptionBtn) then
    if(id=0) then begin
      Image1.Picture.LoadFromFile('0012.jpg');
      id := 1;
    end
```

```
    else if(id=1) then begin
      Image1.Picture.LoadFromFile('0003.jpg');
      id := 0;
    end;
end;
```

其中 Picture 的 LoadFromFile 方法的作用是装载参数路径所给定的图片。这两个图片都放在程序的同一目录下，因此不用写绝对路径。

现在考虑 id 变量的初始化，我们把它写在 Form1 的 OnCreate 事件的处理函数中。这个事件在当 Form1 被创建时触发，是常用的初始化函数。具体的做法如图 1-7 所示。

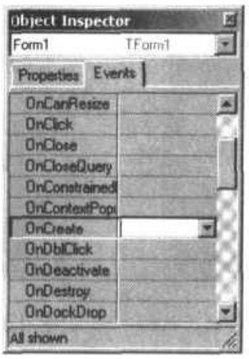

图 1-7　进入 OnCreate 事件处理函数图

用鼠标双击 Object Inspector 中 Form1 的 OnCreate 事件右边的空白处即可自动进入 OnCreate 事件的处理函数。在该函数中添上 id 变量初始化的语句，代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  id := 0;
end;
```

最后，需要考虑的一点是，当窗体大小被拖动改变时，应该保证我们的自绘按钮的位置仍然在最小化按钮旁边，也就是说，当窗体的大小变动时，需要重新在最小化按钮的旁边绘制我们的自定义按钮。这一点，可以通过处理窗体的 OnResize 事件来实现。窗体的 OnResize 事件在窗体需要改变大小时触发。我们需要在这个处理函数中写入我们的重绘代码，显然，只需要调用绘制按钮的函数即可。不过我们也可以通过向窗体发送 WM_NCACTIVATE 消息来实现，因为一旦 Form1 接收到这个消息，它就会调用绘制按钮的函数。这里我们使用了一个 Perform 方法，这个方法将会绕过先把消息发送给 Windows 再接收消息的过程，直接使调用这个方法的控件接收到相应的消息。这里由 Form1 调用这个方法，意味着 Form1 立刻接收到了 WM_NCACTIVATE 消息，后面两个参数是该消息的 wParams 和 lParams 成员。Active 是一个 TApplication 的成员变量，用来判断该应用程序是否被激活或是保有焦点。由于它是 BOOL 类型的，因此首先需要把它强制转换成 Word 类型。

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

```
    Perform(WM_NCACTIVATE, Word(Active), 0)          //强制重新绘制按钮
  end;
```

〖实例效果〗

现在开始运行这个程序看看效果如何。打开【Run】|【Run】菜单项，或者直接按 F9 快捷键，或者直接单击工具栏上的▶按钮，即可运行程序。程序运行结果如图 1-8 所示。

现在可以试验一下我们的自绘按钮的功能了。单击这个按钮，即可看到运行结果变成如图 1-9 所示，若再次单击它，则图片又可变成前一张，如图 1-8 所示。

图 1-8 程序运行结果

图 1-9 按钮单击效果图

〖小结〗

本例主要讲了 Windows 消息的处理机制，以此为基础，我们编写出了这个可以在窗口的标题栏上自绘按钮并且实现我们所需要的功能的应用程序，还对编程步骤作了详细的介绍。

Delphi 7.0 会在每次打开时自动生成一个 Project，主要包括一个程序窗口和一个 Unit 单元文件窗口（该窗口在程序窗口的后面，但它比程序窗口大，读者可以轻松地找到它）。Delphi 是一门所见即所得的可视化编程语言，也就是说，当在程序窗口上放置一些控件的时候，也就意味着在运行的时候，这些控件也会在同样的位置出现。而 Unit 单元文件（一般是一个.pas 文件）则是负责在其中编写所需要的功能代码。

由于这是第一个 Delphi 程序，我们将详细地介绍一下 Unit 的各个组成部分。初始的 Unit 单元文件如下：

```
unit Unit1;                  //表示该 Unit 的名称是 Unit1，如果文件存为 titlebtn.pas，则
                             //Unit 名会自动变为 titlebtn
Interface                    //表示以下为该单元的接口部分
Uses                         //此行表示以下为该单元中可能会引用的其他单元，一般的程序中通过
                             //引用这些单元中定义的一些变量、常量、方法和对象等即可完成，如
                             //果需要完成的功能需要引用另外的单元，可以自行手动添上该单元名
                             //称，当然，必须首先知道那个单元名称
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs;
Type                         //以下为本单元的类型定义，这里首先会预定义一个 TForm1 类，作为
                             //本程序窗口的类，向本程序窗口中添加控件时 Delphi 会自动给这
```

```
                              //个类定义中添加新的成员，这些控件和成员是一一对应的，我们可
                              //以在程序中使用这些成员来调用相应控件的属性和方法，从而控制
                              //程序窗口上的那些控件
    TForm1 = class(TForm)     //程序窗口类继承自 TForm 类，这里显示的 private 和 public 是
                              //它的两个访问控制符，处于 private 声明区的成员和方法只能被
                              //本单元的方法访问，处于 public 声明区的成员和方法可以被所有
                              //的单元访问。如果不在这两个声明区（也就是说在 private 声明
                              //区上面）声明的成员和方法则是默认为处于 published 访问控制
                              //符的作用下，published 访问控制符的功能和 public 的功能大
                              //致相似，不同点在于程序会为 published 访问控制符下的变量生
                              //成运行时类型信息（RTTI-RunTime Type Information），从
                              //而可以动态地访问一个对象的域、成员和方法。
    private
      { Private declarations }
    public
      { Public declarations }
    end;
  var                         //这里是本单元的全局变量定义区，Delphi 初始定义了一个 TForm1
                              //类的变量 Form1，可以使用这个变量来引用程序窗口
    Form1: TForm1;
  Implementation              //以下是本单元的实现部分，用于实现接口中提到的方法
  {$R *.dfm}                  //此行语句是一个编译开关，表示引用.dfm 文件表示的窗口资源
  end.
```

程序单元就先讲述这些，其他部分会在以后的讲解中逐步提到，下面讨论一下消息处理函数的写法。

消息处理函数的写法有一定之规，像程序中所用到的几个消息，它们在定义时的写法是：

```
    procedure WMNCPaint(var Msg : TWMNCPaint); message WM_NCPaint;
    procedure WMNCActivate(var Msg : TWMNCActivate); message WM_NCACTIVATE;
    procedure WMSetText(var Msg : TWMSetText); message WM_SETTEXT;
    procedure WMNCHitTest(var Msg : TWMNCHitTest); message WM_NCHITTEST;
    procedure WMNCLButtonDown(var Msg : TWMNCLButtonDown); message
WM_NCLBUTTONDOWN;
```

我们把这些函数定义放在 TForm1 类的 Private 区域中。

除了消息处理机制外，为了画出按钮来，还使用了诸如设备上下文、画布等 Windows 编程的非常重要的基本知识，这些知识不可能在一讲中完全讲述明白，但是随着五十讲的进展，读者会逐渐熟练地使用它们。有兴趣的读者可以参看 Delphi 的帮助和 MSDN 等参考资料。

实例二　在 Delphi 中定制提示窗口

〖实例说明〗

在程序设计中，有时可能需要一种特定的提示窗口，但是系统提供的简单提示窗口却不能满足需要，所以只能自己编写。本例将介绍如何在 Delphi 中定制自己的提示窗口。本例将