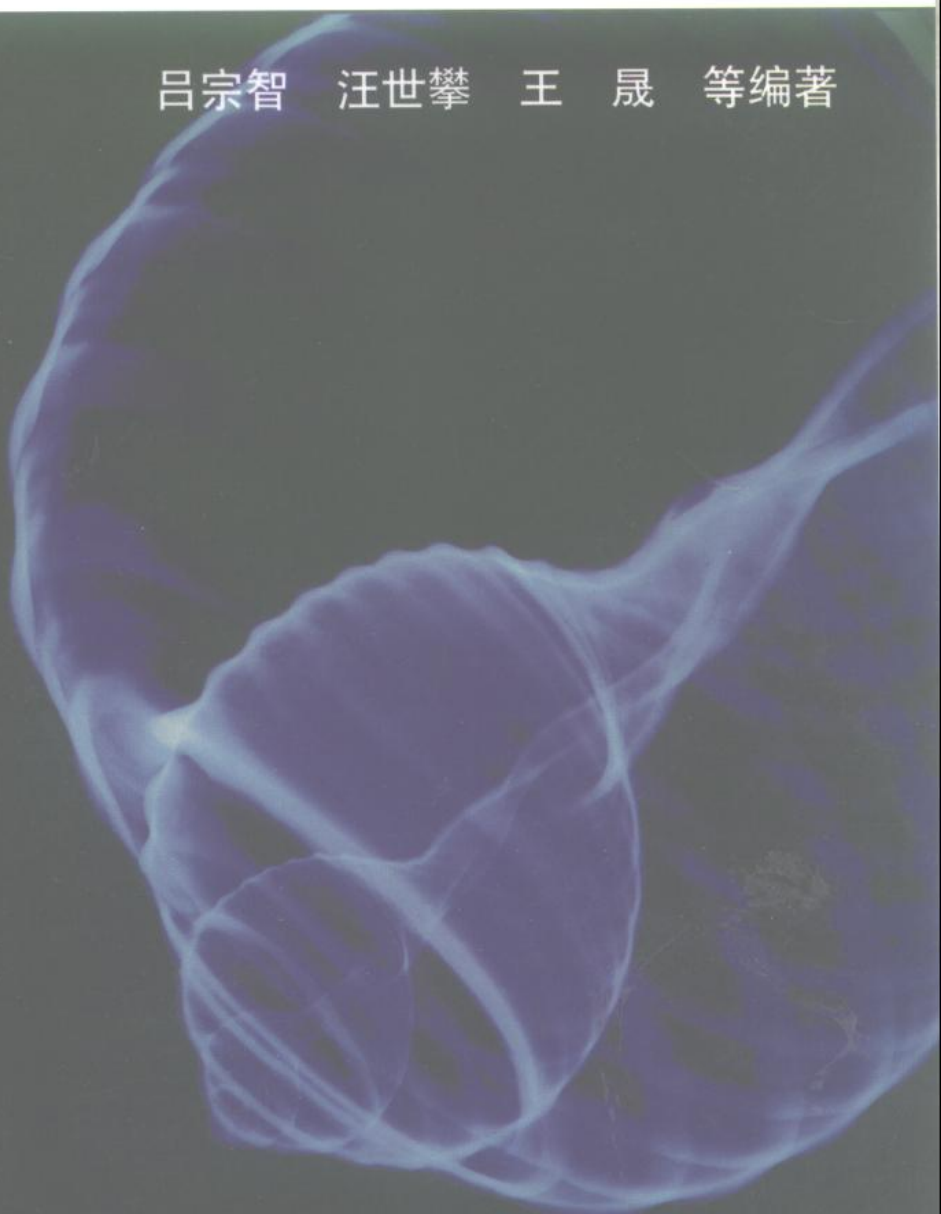
Delphi

实用技术精粹

吕宗智　汪世攀　王　晟　等编著



人民邮电出版社

Delphi 实用技术精粹

吕宗智　汪世攀　王 晟　等编著

人 民 邮 电 出 版 社

内 容 提 要

本书有别于通常的 Delphi 编程图书，从技术性和实用性角度出发，详细讲解用 Delphi 进行程序设计所必需的知识与原则，以及高级程序设计的经验与技巧。本书以“问题——解答——实例”的形式编写，针对 Delphi 的高级使用技巧进行全面的剖析，其中大部分是程序员梦寐以求的问题的解决方案。本书主要针对具有一定 Delphi 应用经验或具有一定 Windows 程序设计经验的读者编写，同时也考虑了不同层次读者的需求，既可帮助程序人员提高 Delphi 程序开发的能力，切实提高应用 Delphi 编程的技能；也可供 Windows 程序设计人员了解、学习 Delphi 之用。

Delphi 实用技术精粹

◆ 编　　著　吕宗智　汪世攀　王　晟　等
责任编辑　陈万寿
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
北京朝阳隆昌印刷厂印刷
新华书店总店北京发行所经销
◆ 开本：787 × 1092　1/16
印张：23.25
字数：581 千字　　2000 年 3 月第 1 版
印数：1 – 5 000 册　　2000 年 3 月北京第 1 次印刷
ISBN 7-115-08410-6/TP·1540

定价：35.00 元

前　言

Delphi 是 Inprise 公司（原 Borland 公司）推出的可视化、面向对象的高效率的快速应用程序开发工具(RAD)，提供了大量的较新较全的开发功能。作为一个优秀的前端开发工具，Delphi 广受赞誉。业界如此形容 Delphi：真正的程序员用 C，聪明的程序员用 Delphi。由此可见 Delphi 受重视的程度。

Delphi 充分利用了 Windows 平台的 32 位体系结构，提高了应用程序的性能。同时，Delphi 完全支持 Windows95/98 以及 Windows NT 的新特性，支持多线程，支持 Windows 的界面特征，支持 Oracle、Sybase 等大型数据库，并且提供了一百多个可重用和可扩展的组件，大大方便了软件开发。

目前市场上已经有很多介绍 Delphi 的图书，但大多数都是从组件出发来介绍系统。本书从另一个角度，按照程序设计的步骤，通过大量的实例来介绍 Delphi 的应用，增强了学习的针对性，使读者能在较短的时间内获得必要的知识。

本书共分为 10 章，分别为：窗体；控件；文件、目录、驱动器；报表、图形、打印；键盘鼠标；Windows API；对 Windows 系统的操作；数据库；网络通信和其他应用。本书从不同的角度，详细地介绍了运用 Delphi 进行应用程序设计的经验和方法，同时也为疑难问题的解决提供了可供借鉴的参考示例。

本书由吕宗智、汪世攀主编。参加本书编写及资料整理的人员还有：王晟、张中彪、唐文韬、李光辉、尤瑞亮、桂敏文、梅景志、

贾春丽、张曙光、冉华、刘志敏、华祖银、徐佩锋、黄宝坤、王勇、魏超、赵记华、闫俊锋、崔敏、陆保军、栾海波、黄琦等。

由于 Delphi 功能强大，内容包罗万象，而作者的水平有限，加之时间仓促，书中缺点和错误在所难免，欢迎广大读者批评指正。

作　者

目　　录

第一章 窗 体

窗体是大多数应用程序最基本的核心部分。在运行应用程序时，与用户交互的大多是含有菜单、按钮、列表框、图片框以及编辑框等控件的实际窗口。要想使用 Delphi 编写程序，首先要了解窗体的特性。本章涉及的技巧虽简单却非常实用，读者可以把它们直接用于自己的运用程序中，如让窗体关闭时再去执行某一程序、实现任意多边形的窗体、实现透明的窗体等等。

本章的主要内容有：

- 给 MDI 主窗口加背景
- 限制窗体的大小
- 查阅可视窗口标题
- 控制窗体
- 不规则窗口的实现
- 在程序运行前显示 Splash 窗体
- 窗体关闭时再去执行某一程序
- 窗体生成时的事件次序
- 在窗体上设置一个热键
- 如何通过窗体的客户区实现拖动
- 怎样使用 PD/PU 滚动窗体
- 制作可移动的分割窗体
- 透明窗体
- 相对于窗体上的点转换成相对于屏幕上的点
- 如何实现标准 setup 程序呈现的变色窗口背景
- 闪动标题栏
- 如何实现窗体标题栏上的按钮
- 如何给窗体增加系统菜单
- 用 Delphi 实现窗口融合技术
- 使窗体适应不同的显示分辨率

1.1 给 MDI 主窗口加背景

问题

在很多 MDI 程序中，MDI 的主窗口基本的功能是提供子窗口显示的位置和提供菜单、

工具条、状态条等，而窗口的客户区一般不会有其他的用途，有时这种情况显得很不协调。如果在这里画上一些软件的商标、公司的标志或者其他的背景图案的话，不仅可以使 MDI 的主窗口更加充实、美观，而且还可以更加突出公司的形象，取得更协调一致的效果。

实例

打开 Delphi，创建一个新的工程。将 Form1 的 FormStyle 设置为 fsMDIForm，即设置成 MDI 主窗口。再在 Form1 上增加一个 Image 元件，并选择要设置的背景的图形文件到 Image 的 Picture 中。

在 Form1 的 Private 中加入如下定义：

```
FClientInstance,
FPrevClientProc : TFarProc;
procedure ClientWndProc(VAR Message: TMessage);
```

在实现(implementation)部分中加入上述过程的具体内容：

```
procedure TForm1.ClientWndProc(VAR Message: TMessage);
VAR
  MyDC : hDC;
  Ro, Co : Word;
begin
  with Message do
    case Msg of
      WM_ERASEBKGND:
        begin
          MyDC := TWMEraseBkGnd(Message).DC;
          For Ro := 0 TO ClientHeight   DIV   Image1.Picture.Height Do
           For Co := 0 TO ClientWIDTH   DIV   Image1.Picture.Width Do
            BitBlt( MyDC, Co*Image1.Picture.Width, Ro*Image1.Picture.Height,
                  Image1.Picture.Width, Image1.Picture.Height,
                  Image1.Picture.Bitmap.Canvas.Handle, 0, 0, SRCCOPY);
          Result := 1;
        end;
      else
        Result := CallWindowProc(FPrevClientProc, ClientHandle, Msg, wParam, lParam);
      end;
end;
```

在 Form1 的创建事件中加入：

```
FClientInstance := MakeObjectInstance(ClientWndProc);
FPrevClientProc := Pointer(GetWindowLong(ClientHandle, GWL_WNDPROC));
SetWindowLong(ClientHandle, GWL_WNDPROC, LongInt(FClientInstance));
```

新增加一个 Form，并将 FormStyle 设置为 fsMDIChild。现在编译运行这个程序，就会发现 Image 控件并不会在 Form 上显示出来，但是整个 MDI Form 的客户区域被 Image 中的

图像所铺满，如图 1-1 所示。

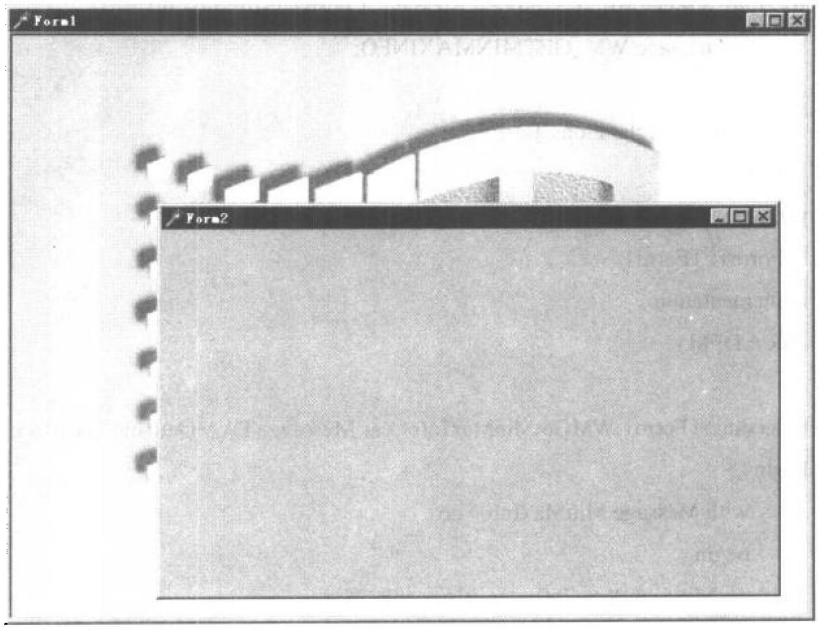


图 1-1 给 MDI 主窗口加上背景图片的运行结果

1.2 限制窗体的大小

问题

在使用 Delphi、Visual Basic 等软件时，用户都会注意到该软件本身最上面的窗口，当它极大时只占屏幕的一小部分，它是如何实现的呢？

技巧

实现窗口极大化时只占屏幕的一部分这一功能的实质是通过自定义消息来处理的，当窗口极大化时，在自定义消息的处理事件中限定窗口的大小和位置，从而实现预定目的。

实例

新建一个工程，程序示例如下：

```
unit Unit_size;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1 = class(TForm)
  private
```

```
{ Private declarations }
procedureWMGetMinMaxInfo(var Message:TWMGetMinMaxInfo );
        message WM_GETMINMAXINFO;
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}

procedure TForm1.WMGetMinMaxInfo( var Message :TWMGetMinMaxInfo );
begin
      with Message.MinMaxInfo^ do
      begin
       ptMaxSize.X := 750;      //最大化时宽度
       ptMaxSize.Y := 300;      //最大化时高度
       ptMaxPosition.X := 10; //最大化时左上角横坐标
       ptMaxPosition.Y := 10; //最大化时左上角纵坐标
      end;
      Message.Result := 0;    //告诉 Windows 你改变了 minmaxinfo
      inherited;
end;

end.
```

提示

也可以在自定义消息的处理事件中加入其他代码，来实现更加丰富的功能。

1.3 查阅可视窗口标题

问题

如何查阅在 Windows 中激活的各个可视窗口标题栏中的标题？

技巧

通过调用 Windows API 函数 GetWindow()，再配合 GetWindowText()，可逐一查出各个可视窗口标题栏中的标题。

API 函数声明如下：

GetWindow() 函数声明

```
HWND GetWindow(
    HWND hWnd,      //原窗口的句柄
    UINT uCmd       //关联标志
  );
```

GetWindowText() 函数声明

```
int GetWindowText(
                    HWND hWnd,        // 窗口的句柄
                    LPTSTR lpString,  // 字符串缓冲区的地址
                    int nMaxCount     // 复制的最大字符数
                     );
```

实例

新建一个工程，其程序示例如下：

```
unit Unit_Find;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    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
 hCurrentWindow: HWnd;
 szText: array[0..254] of char;
begin
 Memo1.Clear;
```

```
  hCurrentWindow := GetWindow(Handle, GW_HWNDFIRST);
  while hCurrentWindow <> 0 do
  begin
   if GetWindowText(hCurrentWindow, @szText, 255)>0 then
      Memo1.Lines.Add(StrPas(@szText));          //将找到的窗口标题添加到 Memo1 中
      hCurrentWindow:=GetWindow(hCurrentWindow, GW_HWNDNEXT);
  end;
end;


end.
```

运行本程序，程序的运行结果如图 1-2 所示。

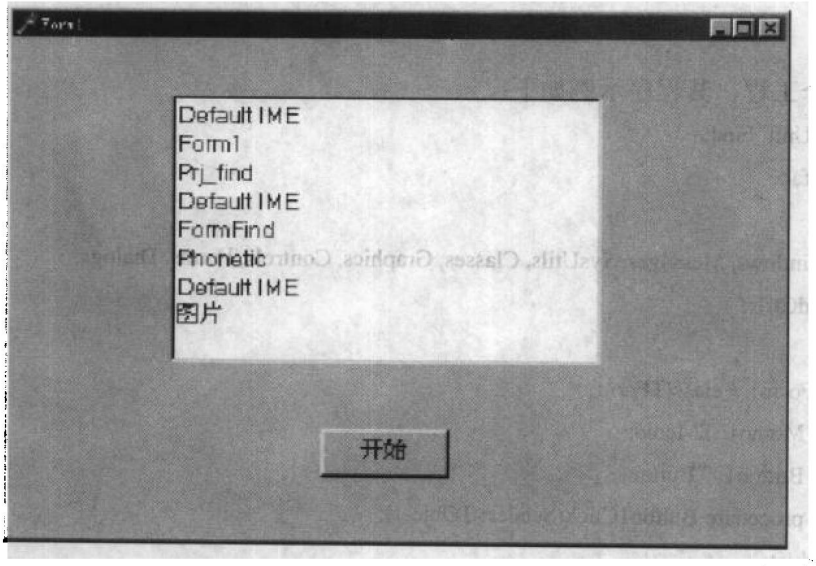


图 1-2　查阅可视窗口标题程序的运行结果

1.4　控制窗体

问题

控制窗体：如何在 Delphi 中把 Form 控制成不能放大/缩小/移动/关闭的状态？

技巧

1. 把 Form 的 BorderIcons 下的几个子属性值全设为 False;
2. 修改 Form 的 BorderStyle 的值为 bsSingle ;
3. 为了让窗口不能移动,可以自己拦截 WM_NCHITTEST 消息，对该消息的处理为：一概回应鼠标点在窗口的 Client 区域，相信这个视窗就呆在原地不会动了。

实例

新建一个工程，其代码示例如下：

```
unit Unit_FormControl;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls,Forms, Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
  Button1: TButton;
  procedure Button1Click(Sender: TObject);
private
{ Private declarations }
  procedure WMNCHitTest(var Msg: TMessage); message WM_NCHITTEST;
public
{ Public declarations }
end;
var
  Form1: TForm1;
implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Close; // 不可少，因为已经没有其他方法能关闭此窗口了
end;

procedure TForm1.WMNCHitTest(var Msg: TMessage);
begin
  inherited; // 这样,移动就不可能了...
  Msg.Result := HTCLIENT;
end;

end.
```

提示

通过对窗体进行控制，可以达到减少用户的误操作等特殊的应用效果。