Delphi 高级编程

刘啸 汪启伟 茹黎涛 等 编著

人民邮电出版社

图书在版编目（CIP）数据
Delphi 高级编程 / 刘啸，汪启伟，茹黎涛编著；——北京：人民邮电出版社，2002.2
（程序员加油站系列图书）
ISBN 7-115-09995-2
Ⅰ．D... Ⅱ．①刘... ②汪... ③茹... Ⅲ．DELPHI 语言－程序设计 Ⅳ．TP312
中国版本图书馆 CIP 数据核字（2002）第 004445 号

内 容 提 要

本书介绍了 Delphi 各个方面的高级编程技术，实例涉及的范围比较广，涵盖了 Windows 开发领域中包括系统、对象模型（COM）、网络、多媒体与数据库等各个方面。每个例子均有一定深度，对于解决读者在开发工作中遇到的具体问题有相当大的帮助。

本书适合于 Delphi 的中高级读者阅读，同时也可用于 Delphi 的初学者在掌握 Delphi 基本编程技巧后的进一步学习。

程序员加油站系列图书

Delphi 高级编程

◆ 编　　著　刘　啸　汪启伟　茹黎涛等

责任编辑　张立科

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ pptph.com.cn
网址　http://www.pptph.com.cn
读者热线：010-67129212　　010-67129211（传真）
北京汉魂图文设计有限公司制作
北京　　印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：720×980　1/16
印张：23.75
字数：579 千字　　2002 年 1 月第 1 版
印数：1 –　　册　　2002 年 1 月北京第 1 次印刷

ISBN 7-115-09995-2/TP・2706

定价：49.00 元

本书如有印装质量问题，请与本社联系　电话：（010）67129223

关 于 本 书

学习一门编程语言尤其是 Windows 下的编程语言，不仅仅需要掌握其语法，同时还应熟悉其思想。在学习过程中，如果企图光依靠阅读编程书籍来掌握编程的话，无异于纸上谈兵。只有亲身实践，多多参与实际开发，才是精通一门编程语言的“捷径”。

Delphi 可以说是一种 Windows 平台上的快速应用开发工具（RAD）。自从 Borland 公司推出该拳头产品以来，Delphi 便以其方便快捷而且功能强大的特性，在 Windows 的开发领域中占据了重要地位。目前，国内的 Delphi 用户也有迅速增多的趋势。

“真正的程序员用 C，聪明的程序员用 Delphi。”这句话体现了 C 与 Delphi 的最大区别。Delphi 的特色就在于它的 VCL 完美地封装了 Windows API，使用 VCL 可以非常方便地实现许多复杂的功能。选择 Delphi，就意味着可以获得相当数量的第三方组件的支持，可以使用大量的源代码，从而节省程序员的时间与精力。可是选择 C 或者 C++就不一样了，在编程的时候需要程序员深入了解并掌握 Windows 的复杂体系。

本书收集了 Delphi 的大量典型编程实例，包括界面设计、控件使用、系统编程、COM/DCOM、多媒体、网络、数据库等，基本上涵盖了 Windows 下程序开发的方方面面。各个例子都有一定深度，适合于中高级读者阅读。

本书第一、五、八、十一章由汪启伟编写，第二、三、四、九章由茹黎涛编写，第六、七、十一、十二章由刘啸编写，全书由刘啸定稿。由于编者水平有限，书中不足之处在所难免，恳请广大读者批评指正。

编者

此为试读，需要完整PDF请访问：www.ertongbook.com

目　　录

此为试读,需要完整PDF请访问: www.ertongbook.com

第 1 章　界面设计

许多程序员在进行软件开发的时候，往往着重于功能的实现而忽视界面的设计，或者认为把功能设计好并满足了客户的需求就万事大吉。实际上，这种想法常常会导致灾难。

许多客户可能根本不懂编程，他们也不会关心底层的功能实现是多么的复杂繁琐。客户对软件产品只有两个要求：一个是功能齐全，另一个便是界面友好并且容易操作。一个友好的界面，能让使用者易于上手，从而能扩大用户群，也能间接地拓宽产品市场、增加产品利润。相反，界面不友好的软件，即使功能做得非常完善，也可能会由于不易使用的缺陷而被用户拒之门外，这应该是每一位软件作者都不希望看到的。

那么，怎样设计比较好的、让用户满意的界面呢？其中有许多因人而异的原则，难以一一描述，下面列举了实现一些典型界面的几种方法。在实际的编程中，这些方法可以融合到具体的某个软件中，同时也可以加以扩展，从而设计出更美观友好的界面来。

1.1　可停泊窗体的设计

拥有可停泊窗体是目前许多软件的常见风格，如微软公司的 Word、Excel 等产品以及现在使用的 Delphi 都有这个特点。“ 停泊 ” 的意思就是某个元件如工具栏、菜单栏、面板等可以从它所属的容器中通过拖动的方式分离出来成为单独的浮动元件，也可以再次拖动回到容器中重新融合。下面这里实现了一个拥有简单的停泊功能的窗体。只要清楚了可停泊窗体的设计原理，就可以随心所欲地设计这种风格的元件了。

做一个可停泊窗体的步骤主要有两步。

- 选择一个容器作为需要停泊的地方。可停泊的容器可以是 Form 或 Panel 等，原则上只要拥有 DockSite 属性的可视化控件，也就是 TWinControl 的子类，都可以作为停泊区。
- 选择需要拥有可停泊风格的控件，可以是任何属于 TControl 子类的控件。

1.1.1　定义停泊区

要让一个属于 TWinControl 子类的控件成为一个停泊区，只要设置其 DockSite 属性为 True 即可。如果要让停泊区没有子对象时不显示，则需要设置其 AutoSize 属性为 True。如果有子对象停泊进去时，停泊区会自动调整大小。

1.1.2　如何定义停泊对象

让一个 TControl 构件成为一个可停泊的对象，需要按如下方法设置其属性。

- 设置 DragKind 为 dkDock 以使其可以执行停泊的动作。
- 设置 DragMode 为自动模式 dmAutomatic，这表示当用户拖动本对象时自动进入待

停泊状态。如果设置为手工方式 Manual，则必须调用其 BeginDrag 方法才能开始停泊。

♦ 设置 FloatingDockSiteClass 属性，当可停泊对象从可停泊区被拖出来时，可使用这个类建立作为可停泊区的窗体。如果可停泊对象是 TWinControl 的子类，这个属性可不需要指定。

1.1.3 几个重要的事件

Procedure TForm1.Panel1GetSiteInfo(Sender: TObject; DockClient: TControl; var InfluenceRect: TRect; MousePos: TPoint; var CanDock: Boolean);

当用户拖动可停泊对象经过可停泊的区域时，这个事件就被触发，这个事件通过设置参数 CanDock 的值来回答能否停泊对象以及停泊到何处的问题。

procedure TForm1.Panel1DockOver(Sender: TObject; Source: TDragDockObject; X, Y: Integer; State: TDragState; var Accept: Boolean);

如果 onGetSiteInfo 没有拒绝，那么 onDockOver 事件被继续触发，回答了当前能否接受这个子对象。

procedure TForm1.Panel1DockDrop(Sender: TObject; Source:TDragDockObject; X, Y: Integer);

这个事件发生在用户投放了子对象之后，停泊区开始停泊这个子对象，有关子对象的信息可以从 source 中读取。

procedure TForm1.Panel1UnDock(Sender: TObject; Client: TControl; NewTarget: TWinControl; var Allow: Boolean);

在自动模式下，只要用户往外拖这个对象，可停泊区自动释放子对象。往外拖的时候，会触发 onUnDock 事件。在这个事件中，可以通过属性 DockClientCount 和 DockClients 来访问这些子对象。在开始停泊和结束停泊的时候，会触发 onStartDock 和 onEndDock。类似于托访操作，在事件 onStartDock 中，可以建立 TDragDockObject 对象。

1.1.4 简单示例

下面我们制作一个简单的程序，以加深对可停泊窗体的理解：在 Delphi 中选择 File | New Application 生成一个默认的 Application，然后在主 Form 上放置一个 Panel、一个 CoolBar。在 CoolBar 上新建一个 Band，然后加入一个 ToolBar 工具栏，在 ToolBar 上新建几个按钮，各个控件的属性设置如表 1.1 所示。

表 1.1 可停泊窗体的控件属性设置

构件	类型	属性	属性值
Coolbar1	TcoolBar	Autosize\Docksite Ondockdrop\Ondockover	True True
Dockpanel1	Tpanel	Align\DockSite	AlBottom\True
ToolBar1	TtoolBar	AutoSize\DragKind DragMode	True\DkDock dmAutomatic
TtoolForm	Tform	Borderstyle\Caption Dragkind\Dragmode Name\Onclose Ondockdrop	BsSizeToolWin ToolForm DkAutomatic ToolForm

此为试读，需要完整PDF请访问：www.ertongbook.com

源代码如程序清单 1.1 所示。

（程序清单 1.1）

```
//////////// mainform 窗体
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, ToolWin, ExtCtrls;
type
  Tmainform = class(TForm)
    Panel1: TPanel;
    CoolBar1: TCoolBar;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    procedure FormCreate(Sender: TObject);
    procedure CoolBar1DockOver(Sender: TObject; Source: TDragDockObject; X,
      Y: Integer; State: TDragState; var Accept: Boolean);
    procedure CoolBar1DockDrop(Sender: TObject; Source: TDragDockObject; X,
      Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  mainform: Tmainform;
implementation
uses Unit2;
{$R *.DFM}

procedure Tmainform.FormCreate(Sender: TObject);
begin
  toolbar1.FloatingDockSiteClass:=TToolForm;
end;

procedure Tmainform.CoolBar1DockOver(Sender: TObject; Source: TDragDockObject;
  X, Y: Integer; State: TDragState; var Accept: Boolean);
var arect:TRect;
begin
  Accept:=(source.control is TToolForm);
  if accept then
  begin
    aRect.topleft:=coolbar1.clienttoscreen(coolbar1.clientrect.topleft);
    arect.bottomRight:= CoolBar1.ClientToScreen(CoolBar1.ClientRect.BottomRight);
    Source.DockRect:=ARect;
  end;
end;

procedure Tmainform.CoolBar1DockDrop(Sender: TObject;
```

```
    Source: TDragDockObject; X, Y: Integer);
begin
    if Source.control is TToolForm then
    begin
        TToolBar(TToolForm(source.control).controls[0]).dragmode:=dmAutomatic;
        TToolForm(source.control).controls[0].ManualDOck(CoolBar1,nil,alNone);
        TToolForm(source.control).close;
    end;
end;
end.
//////////////////////////////// Toolform 窗体
unit Unit2;
interface
uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    ComCtrls, ToolWin, ExtCtrls;

type
    TToolForm = class(TForm)
        procedure FormDockDrop(Sender: TObject; Source: TDragDockObject; X,
            Y: Integer);
        procedure FormClose(Sender: TObject; var Action: TCloseAction);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    ToolForm: TToolForm;

implementation

{$R *.DFM}

procedure TToolForm.FormDockDrop(Sender: TObject; Source: TDragDockObject;
    X, Y: Integer);
begin
    caption:=(source.control as TToolBar).caption;
    (source.control as TToolBar).dragmode:=dmManual;
    visible:=true;
end;

procedure TToolForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    action:=cafree;
end;
end.
```

程序运行结果如图 1.1 所示。

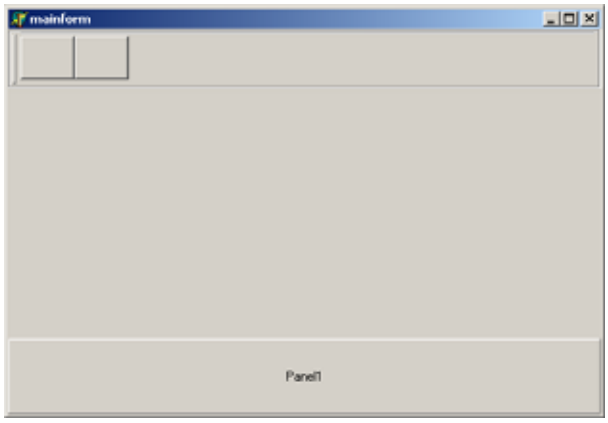

图 1.1　拖动前的窗口

而将 CoolBar 拖动出来后的界面如图 1.2 所示。

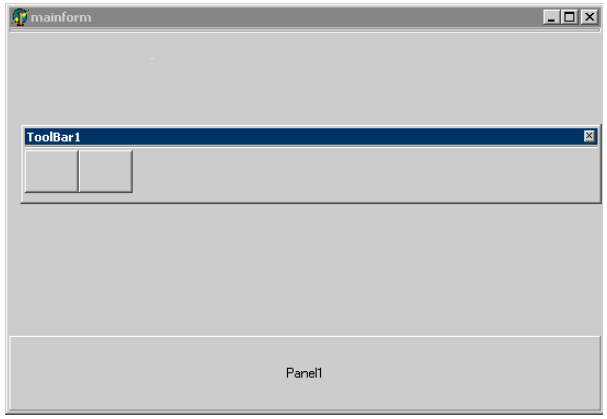

图 1.2　拥有浮动 ToolBar 的窗体

当 ToolBar 停泊在 Panel1 上时如图 1.3 所示。

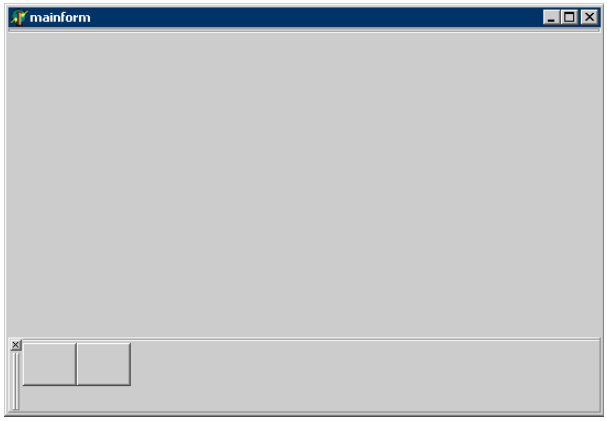

图 1.3　已停泊到 Panel1 上的工具栏

主窗体建立的过程中，首先设置了 ToolBar 的浮动窗体类 FloatingDockSiteClass，设置了这个属性之后，当 ToolBar1 从主界面拖出来后，自动建立 TtoolForm 浮动窗体。使其成为

ToolBar1 的新宿主。载有 ToolBar1 的 TtoolForm 窗体也是一个停泊对象，它代替 ToolBar1 询问停泊目标。适当的时候把 ToolBar1 再停泊回去。

需要考虑 3 种情况。

♦ 把 ToolBar1 从 CoolBar1 或者 dockPanel 中拖出来，使之成为浮动窗体的对象。这个过程是自动进行的，只是在 TtoolForm 接受时，才调用了 FormDockDrop 事件。在此事件中，ToolBar1 的拖动方式被设置为手工方式。

♦ 把 ToolBar1 从浮动窗体中拖出来，再回到 Coolbar1 中。这时 ToolBar1 是手工方式，也没有调用 BeginDrag 方法，所以不具有被拖动的特性。TtoolForm 本身是一个可停泊对象，当其经过 CoolBar1 时，CoolBar1 在 onDockOver 事件中回答是否接受、如何接受。

♦ 当确定要把 TtoolForm 停泊到 CoolBar1 时，触发事件 onDockDrop。在此事件中，使用手工方式将 ToolBar1 停泊到了 CoolBar 上，并将 ToolBar1 的停泊方式再设置为手动方式。其他停泊方式，如从 CoolBar1 停泊到 DockPanel 中，都是自动进行的。

1.2 透明表格

透明表格的设计效果比较奇特，Form 上有一图片，在其上放置 DBGrid 等表格元件时，表格显示时能以 Form 上的图片为背景，从而呈现出一种“透明”的效果。

以下是一个简单的示例来演示如何让一个表格透明并且显示表格下的图片，这个示例有完整的代码。首先窗体上控件的分布如下。

♦ ADOTable1：（也可以使用通常的 TTable）用来显示需要的数据，并将其激活，也就是设置 Active 为 True。

♦ DataSource1：连接上面的 ADOTable1。

♦ Image1：用于显示一个位图 Bmp 文件，设置其 align 为 alClient 以覆盖整个窗体。

♦ Panel1：将其大小设置为足够放下我们的表格控件。

♦ DBGrid1：它作为一个子控件放在 Panel1 上，以 alBottom 方式对齐，并在 Panel1 的上部留一部分可见空间。不要忘记将其链接到 DataSource1。

以下的代码包含两个过程，第 1 个过程用来处理 DBGrid1 的 OnDrawDataCell 事件，它包含了让表格透明的代码。第 2 个过程用来处理 Panel1 的 OnMouseDown 事件，这会使 Panel1 移动的同时保持“透明”。

源代码如程序清单 1.2 所示：

（程序清单 1.2）

```
unit fTransparentGrid;
  interface
  uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    Db, Grids, DBGrids, ADODB, ExtCtrls, DBTables;
```

```
type
  TForm1 = class(TForm)
  Image1: TImage;
  DataSource1: TDataSource;
  Panel1: TPanel;
  DBGrid1: TDBGrid;
  Table1: TTable;
   procedure DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
       Field: TField; State: TGridDrawState);
     procedure Panel1MouseDown(Sender: TObject; Button: TMouseButton;
       Shift: TShiftState; X, Y: Integer);
  private
  public
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
  Field: TField; State: TGridDrawState);
var
  Text: string;
  Rct: TRect;
begin
  Text := Field.AsString;
  Rct:= Rect;
  BitBlt(DBGrid1.Canvas.handle,
            Rct.left,
            Rct.top,
            Rct.right - Rct.left,
            Rct.bottom - Rct.top,
            Image1.Canvas.Handle,
            Rct.left + DBGrid1.Left + Panel1.Left,
            Rct.Top + DBGrid1.Top + Panel1.Top,
            SRCCOPY);
  SetBkModE(DBGrid1.Canvas.Handle, TRANSPARENT);
  DBGrid1.Canvas.Font.Style := [fsBold];
  DrawtextEx(DBGrid1.Canvas.Handle,
                  PChar(Text),
                  Length(Text),
                  Rct,
                  DT_WORDBREAK,
                  nil);
 end;

procedure TForm1.Panel1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
```

```
begin
   ReleaseCapture;
   Panel1.Perform(WM_SYSCOMMAND, $F012, SC_MOVE);
   Application.ProcessMessages ;
   BitBlt(GetDc(Panel1.Handle),
            0,
            0,
            Panel1.Width,
            Panel1.Height,
            Image1.Canvas.Handle    ,
            Panel1.Left, Panel1.Top,
            SRCAND);
     DBGrid1.refresh;
end;
end.
```

本例的运行效果如图 1.4 所示。

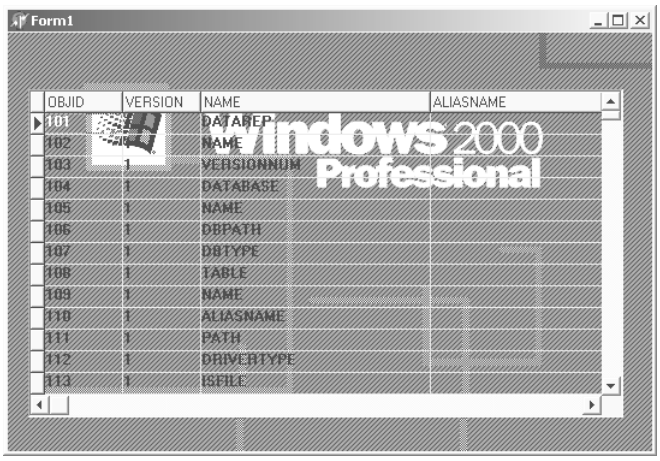

图 1.4　透明背景的表格

下面仔细分析一下前面的两段代码。

在 onDrawDataCell 事件中，因为此事件是当 DBGrid1 画某一个格子的时候响应的，如果在这个事件中用 BitBlt 函数把 Image1 相应地方的图片复制到 DBGrid1 画布的相应的格子中，并把背景设置为透明，然后再把数据重新画到格子中。这样看起来就是透明的效果。

让 Panel1 透明的方法是和让 DBGrid1 透明的方法差不多，都是用了 BitBlt 函数，Panel1 透明后，然后刷新 DBGrid。

1.3　自适应分辨率的窗体的实现

许多程序员在编写程序的时候，可能都没有考虑过要让程序中的窗体及控件适应显示分辨率的问题。各个用户的使用习惯不同，有的用户可能喜欢 800dpi×600dpi，而另外的用户可能偏好 1024dpi×768dpi。如果你没有在程序中进行自适应调整，那么窗体及其控件在不同

此为试读，需要完整PDF请访问：www.ertongbook.com