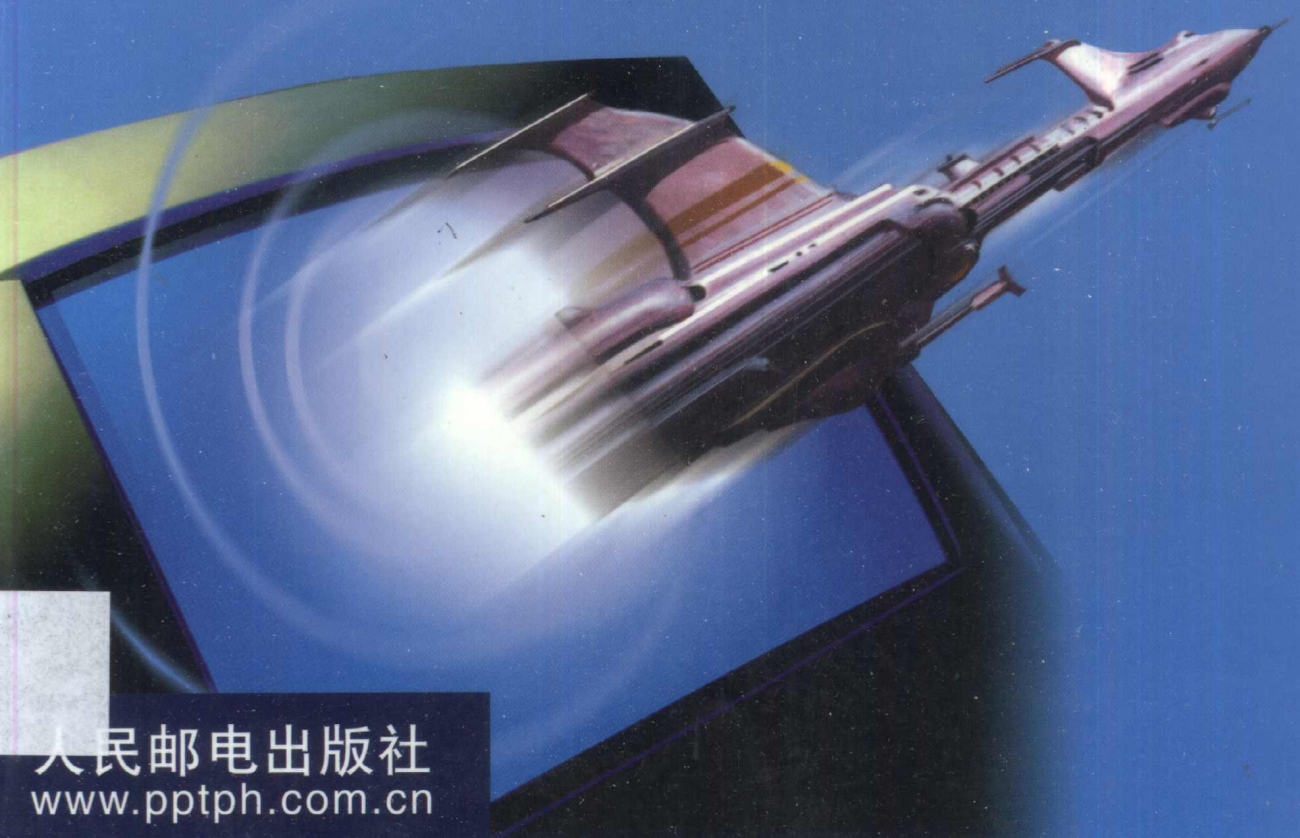
特效视窗——

Delphi

开发高级界面实例

李睿　方坤　等编著



人民邮电出版社
www.pptph.com.cn

特效视窗

——Delphi 开发高级界面实例

李 睿 方 坤 等编著

人民邮电出版社

图书在版编目(CIP)数据

特效视窗：Delphi 开发高级界面实例/李睿等编著.北京：人民邮电出版社，2000.9
ISBN 7-115-08696-6

I.特… II.李… III.Delphi 语言-程序设计 IV.TP312

中国版本图书馆 CIP 数据核字(2000)第 41086 号

特效视窗——Delphi 开发高级界面实例

◆ 编　著　李　睿　方　坤　等
责任编辑　王晓明
◆ 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
北京汉魂图文设计有限公司制作
北京密云春雷印刷厂印刷
新华书店总店北京发行所经销
◆ 开本：787×1092　1/16
印张：25.75
字数：646 千字　2000 年 10 月第 1 版
印数：1－5 000 册　2000 年 10 月北京第 1 次印刷
ISBN 7-115-08696-6/TP·1758

定价：38.00 元

前　言

在当今的信息化社会中，计算机已成为人们工作和生活中很重要的一个组成部分　对于大多数的人来说，使用计算机是为了处理各种具体的问题；但是对于从事编程工作的人员来说，却远不止这些，他们还必须编制出必要的程序来解决计算机的应用问题。评价一个程序人员的标准是多方面的，其中，编程能力是很重要的一方面。一般来说，一个编制出来的应用程序仅仅能够运行还是不够的，还需要有友好而漂亮的界面，让人们用起来感到方便和赏心悦目，这样才能算是一个好的应用程序。由于现在为广大的计算机用户所广泛接受的操作系统是图形界面的 Windows 操作系统，因此设计好一个应用程序的界面也就更加重要了。本书内容就是专门介绍怎样采用 Delphi 编程工具来编制一般的应用程序界面的方法和技巧。

Delphi 是由著名的 Borland 公司开发的基于 Windows 9x/Windows NT 的可视化软件开发工具，具有高效、优化的可视化应用程序开发环境和可扩展的数据库技术等特点。Delphi 的版本经过了从 1.0 到 5.0 的变迁，现在已经发展得相当成熟，本书所介绍的内容涉及到的对应版本是 Delphi 5.0。Delphi 所使用的编程语言是面向对象的语言，用它来开发的应用程序具有可重用性的特点以及很强的异常处理能力。Delphi 与 Visual Basic 相比，有更漂亮而又方便的图形窗口界面和更严谨的语法结构，而且也具有与 Visual Basic 相同的简单易用特性。对于一个刚刚涉足计算机编程技术的人来说，Delphi 是一种非常便于入门的优秀的编程工具。

本书内容在编写时基于以下假设：

- 读者熟悉 Pascal 语言，懂得基本的语法规则。
- 读者对面向对象的编程有所了解。
- 读者基本了解 Delphi 的编程环境。

本书的特点是以实例的方式来介绍相关内容，全书共分为 19 个实例，这 19 个实例基本上包括了界面设计方面的绝大部分内容。在每一个实例中除了按一条或者几条具体的主线来具体讲述各种不同界面的设计方法外，还对每个实例中所用到的特殊控件和使用方法都进行了比较清楚的介绍。

本书在编写过程中力求语言通俗，深入浅出。帮助程序设计初学者顺利入门，同时也向对 Delphi 有一定的了解的读者介绍一些有实用价值的应用程序界面开发技巧和方法，是作者编写本书的目的之一。

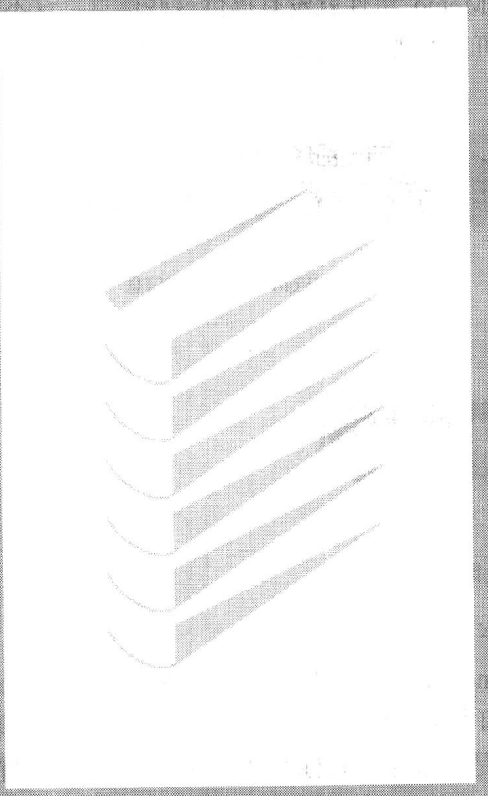
由于作者的水平有限，书中的错误在所难免，诚挚地希望广大读者批评指正。

作　者

2000 年 4 月

目　录

实例1 Memo控件的光标定位



实例简介

作为一个功能强大的可视化程序开发工具，Delphi 提供了大量的简单便捷的控制组件，通过这些组件，程序设计者可以设计出规范美观的界面，并且可以通过这些控件规范化使用者的输入内容。其中，Delphi 提供的编辑组件中的 Memo 控件的功能是非常丰富的，它可以完成对文字的编辑和修改，对文件的调入和存储文件等功能。但是与一般的字处理软件相比，还是有一些缺憾的，比如在用 Word 进行文字编辑的时候，如图 1-1 所示就会把光标的行列位置显示出来，但是 Memo 组件就没有提供这些信息，这不能说不是一个缺憾。其实，通过很简单的几行程序就可以实现这一要求，下面就通过编制一个简单的文本编辑程序来说明这种功能的实现。

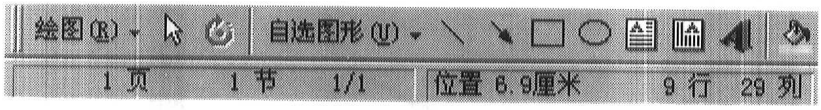


图1-1　Word 显示的行列信息

实现过程

☛ 所用控件属性的说明

1. 实例中所用到的控件

首先，从工具栏的 Standard 控件组中，选择编辑文本用的 Memo 控件，把它放在 Form 当中，并选择合适的位置和大小；然后，再从 Win32 控件组中选择 StatusBar 控件放到 Form 的最下端。StatusBar 控件是用来显示光标在 Memo 控件中的位置而设立的，读者也可以灵活地通过其它控件来显示这些信息。图 1-2 所示就是 Form 上控件的具体情况。

2. Memo 控件属性的介绍

Memo 有一些属性是值得注意的，比如 Lines 属性就是 Memo 里的 Tstrings 变量，可以通过读写 Lines 的值来实现对文本文件的存取，这在后面的编程内容中会有所涉及。应该注意 Memo.Lines 的缺省值在 Delphi 中是 Memo 的名字（Name 属性），如果不改变这个属性的值的话，运行程序以后会发现最一开始 Memo 里写的是 Memo 的名字。另外，Memo 的 ScrollBar 属性也是很有用的，它可以设四个值，分别是 ssNone、ssVerticle、ssHorizontal 以及 ssBoth,分别表示无滚动条、在垂直方向有滚条、在水平方向有滚动条以及在水平垂直方向都有滚动条。Memo 的其它属性是共有的。图 1-3 中所示的是 Memo 的属性表。

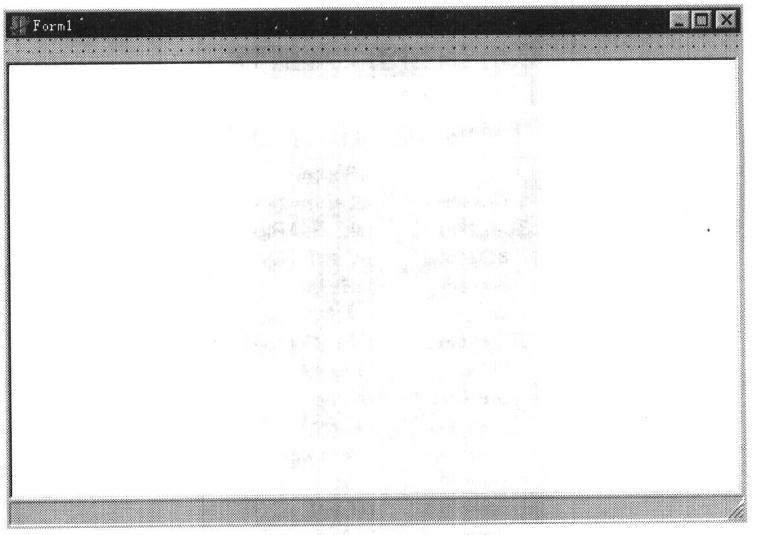


图1–2　实例控件放置图

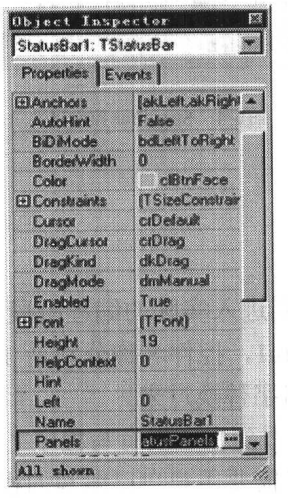


图1–3　Memo 控件属性

3. StatusBar 控件的属性

设置 StatusBar 控件的具体属性的方法是：如图 1–4 所示，选中 StatusBar 控件可以看到它的一些基本属性，有一些是从别的类里继承过来的。在一般情况下 StatusBar 的 Align 属性都设为 alBottom，这样 StatusBar 就到了表单的最下方，并与整个表单对齐了，所介绍的这个例子也是这样。StatusBar 的另外一个重要属性就是 SimplePanel 的属性，如果 SimplePanel 的属性为 True，则在 StatusBar 中只能看到一个面版。如果想把 StatusBar 设为多个面版就得把 SimplePanel 属性改为 False，然后在 Panel 属性里对各个面版进行设置，对这方面的知识在以后会有所涉及。由于这里所讲的内容只涉及一个面版，因此就只需要用 SimpleText 属性了。StatusBar 中显示的文字就保存在 SimpleText 属性中，这样就可以通过改动 SimpleText，在 StatusBar 里面显示出所要表示的光标位置了。

有关 StatusBar 的其它属性读者可以自己动手去试验一下，这样对一般控件的属性就会

比较熟悉了。

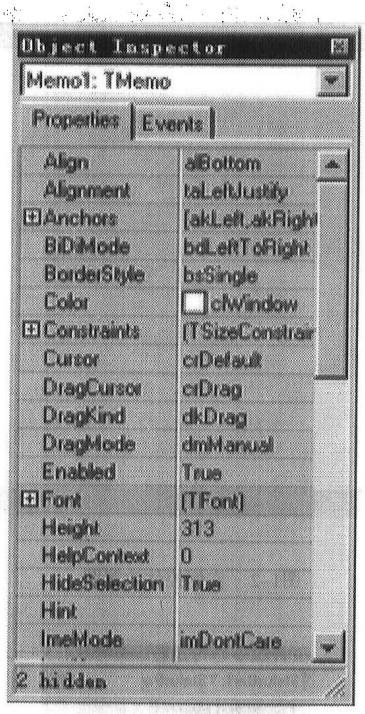


图1-4　StatusBar 控件属性

☛ 程序的源码及其注释

1. 有关控件的说明

完成对控件的设置之后，就要进入实际的编程阶段了。在进入具体编程阶段之前，还有一些约定：Form 的名称是 Form1，控件名称分别是 Memo1、StatusBar1，读者可以根据自己的实际情况把程序中相应的名字替换过来。

首先，把编辑窗口打开就可以看到在还没有写程序之前，Delphi 就已经生成了很多的类、变量及函数，这大大方便了编程。需要用户自己编写的程序都要放在 implementation 之后。

2. 程序代码部分

本程序的源代码如下：

(1) 先在 implementation 之后加入三个全局变量：

```
var
Lpos,Cpos,LineLength:Integer;
```

Lpos 用来记录当前光标的行位置，Cpos 用来记录当前光标的列位置，LineLength 用来记录当前行的文字的个数（确切地说是字节数）。

(2) Memo 控件的 MouseUp 和 KeyUp 事件

单击选中表单上的 Memo 控件，然后选中 Object Inspector 中有关 Memo 控件的事件（Event）的标签，如图 1-5 所示。

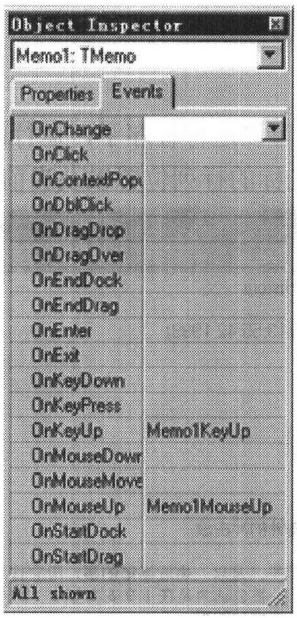


图1-5　Memo 控件事件列表

双击 OnKeyUp 事件，会发现在编辑窗口中多了一个过程：

procedure TForm1.Memo1KeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);

双击 OnMouseUp 事件，又增加了一个过程：

procedure TForm1.Memo1MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

在这两个过程的代码编辑部分分别插入如下的程序：

```
begin
  Lpos:=SendMessage(Memo1.Handle,EM_LINEFROMCHAR,Memo1.SelStart,0);
  //得到当前光标所在的行值
  Cpos:=SendMessage(Memo1.Handle,EM_LINEINDEX,Lpos,0);
  //得到当前光标所在行第一个字符之前的所有字符个数
  LineLength:=SendMessage(Memo1.Handle,EM_LINELENGTH,Cpos,0);
  //得到当前光标所在的行的总字数
  Cpos:=Memo1.SelStart-CPos;
  //用当前光标的之前的所有字符数减去当前光标所在行第一个字符之前的所有字
  //符数得到当前光标的列位置
  Lpos:=Lpos+1;
  Cpos:=Cpos+1;
  //为了符合平常的习惯，把行、列数加 1
  StatusBar1.SimpleText:='行：'+InttoStr(Lpos)+ '列：'+InttoStr(Cpos)+ '此行字数'+InttoStr(Linelength);
  //向 StatusBar.SimpeText 中写入要显示的行列以及当前行字数的信息
end;
```

这两个过程的主要作用是得到当前光标所在的行数和列数以及当前行的总字符数，然后把这些信息显示到 StatusBar 控件中。

☛ 程序的运行结果及说明

1. 程序的运行结果

添加完相应的程序代码后，便可以运行程序了，运行的结果如图 1-6 所示。从图 1-6 中可以看到 StatusBar 中显示了当前光标的行列位置以及当前行的所有字符数。

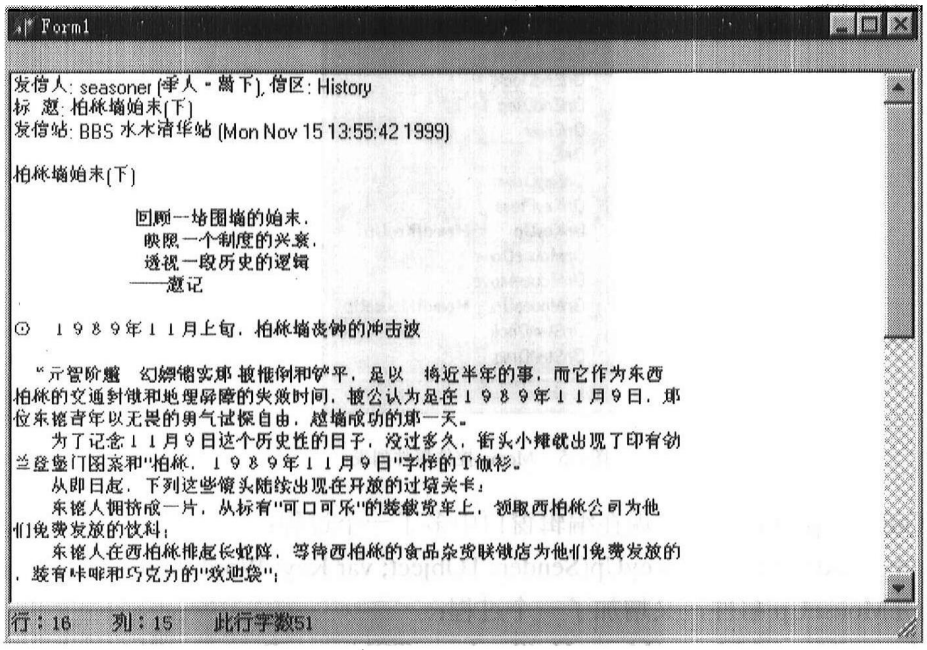


图1-6 程序运行图

2. 实例的说明

(1) 为什么要设立全局变量

在程序中根据实际要求，敲击键盘或者单击鼠标时，都可能引起光标位置的变化。因此，要在键盘按键或者鼠标按键抬起时，读行列数以及当前行的字数值。这就需要在两个过程中用到这些变量，而用了全局变量，就可以在程序中少加入一些变量。

(2) 为什么要用 OnKeyUp 以及 OnMouseUp 事件

读者可以试着把 OnKeyUp 以及 OnMouseUp 事件改为 OnKeyDown 和 OnMouseDown，看看运行程序时会有什么效果。这时可以看得出行列值以及当前行的字符数表示的都是上一次光标所在的位置。这是很容易理解的，因为当 KeyDown 以及 MouseDown 发生的时候光标是在上一个位置，他读出的当然是上一次光标的位置；而当 KeyUp 以及 MouseUp 事件发生时的光标就已经移动到了当前的位置。

(3) 对程序中的一些特殊符号的说明

Memo1.SelStart：SelStart 的属性是整形的（Integer），在文本控件当中它所表示的有两种情况：

第一种情况，当文本控件中有字符被选中的时候，它表示的是被选中的首字符在整个文本中的位置。

第二种情况，当文本控件中没有字符被选中时，它表示的是当前光标在整个文本中的位置。

在本程序中大部分应用的是第二种情况，但是当选中 Memo 中的某一部分字符时，就可以看得出它所指示的是选中的第一个字符的位置以及第一个字符所在行的字符数。

(4) SendMessage 函数

SendMessage 函数向 Windows 传输特定的消息，它的返回值是由它所发出的值所决定的，具体的参数如下：

```
LRESULT SendMessage(
  HWND hWnd,// 目标窗口的句柄
  UINT Msg,   // 发送的消息
  WPARAM wParam,// 第一个消息参数
  LPARAM lParam   // 第二个消息参数
);
```

EM_LINEFROMCHAR：一种消息，它的第一个消息参数是字符在整个文本中的位置，第二个参数为 0。它所得到的返回值是整形的，是第一个消息参数对应的字符所在的行数。因此，SendMessage(Memo1.Handle,EM_LINEFROMCHAR,Memo1.SelStart,0)所得到的就是当前光标所在的行数。

EM_LINEINDEX：也是一种消息，它的第一个消息参数是行数，第二个参数是 0。它所得到的是当前行之前的所有字符数。因此，SendMessage(Memo1.Handle, EM_LINEINDEX,Lpos,0)所得到的就是当前光标所在的行第一个字符的位置。

EM_LINELENGTH：同上面的两个消息，它的第一个消息参数是某一行的第一个字符的位置，第二个消息参数是 0。它所得到的是当前行的总字符数。因此，SendMessage(Memo1.Handle,EM_LINELENGTH,Cpos,0)所得到的就是当前光标所在行的总字符数。

(5) 为什么要在得出光标位置后要对行列值分别加 1

因为在上面得到的行列值都是以 0 为基准的，这与人们的习惯不同，所以它们都加了 1。

(6) 为什么程序对中文的统计不正确

首先说明这个程序对英文的统计是完全正确的。其实，应当知道所得到的行列值，以及当前行的字符数都是字节的数目，而中文一个字占用的是两个字节，所以程序在统计汉字时，把一个字当成了两个字节。所以就出现了上述的问题，可以看得出来它所统计的汉字的列数正好是真正列数的两倍。

(7) 程序的初始设置

细心的读者可能已看出，在这个程序运行的一开始 StatusBar 里是没有显示信息的，这是因为在运行程序的一开始并没有激发 KeyUp 或者 MouseUp 事件。怎样来避免这种情况的出现呢？这就需要使用 Form1 的 OnCreat 事件。选中 Form1 的 Event 选项卡，如图 1-7 可以看得出有 OnCreat 事件，双击 OnCreat 事件就会发现在你的源代码中加入：

```
procedure TForm1.FormCreate(Sender: TObject);
```

在这段程序中加入：

```
begin
  Lpos:=1;
  Cpos:=1;
```

```
  LineLenth:=0;
  //设行列以及当前行长度的初始值
  StatusBar1.SimpleText:='行：'+InttoStr(Lpos)+'    列：'+InttoStr(Cpos)+'    此行
  字数'+InttoStr(Linelength);
  //在 StatusBar 中显示信息
end;
```

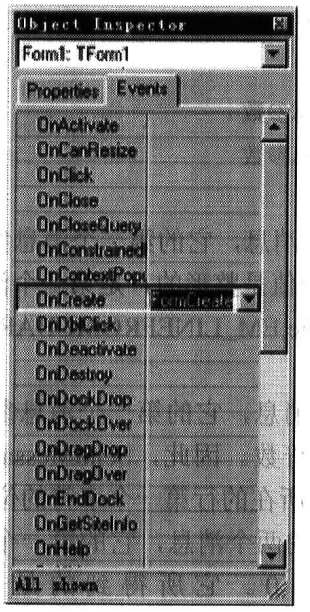


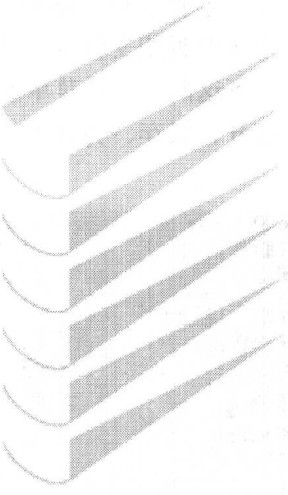
图1-7　表单的事件列表

这样，在程序运行的开始阶段产生 Form1 时，就赋予了 Lpos、Cpos、LineLenth 的初始值，就不会出现在启动程序之后没有行数和列数以及该列总字数的显示。

小　结

前面介绍了有关在 Memo 中光标定位的程序，这里所要说明的是，以上讲的只是一种方法，读者可以根据自己的实际要求对程序进行改动。比如，可以把 Memo 控件换成 RichEdit 控件，或者不在 StatusBar 里显示信息而是在其它的控件中显示等等。以上所讲的程序对所有的文本编辑控件都是适用的。有了当前光标的行列值，会让用户在使用应用程序时感到非常方便。另外，这里有关消息的说明所讲的只是一些对文本操作的消息，应用这些消息解决问题是非常方便的。

实例2 类似于 Office 菜单的实现



实例简介

在上一个实例中介绍了怎样在文本编辑的应用程序中得到当前光标的位置以及当前行的总字数。但是，当用户在 Memo 窗口里对文本进行编辑时，要想像其它的文本编辑软件那样在编辑完文本后进行存盘还是做不到的。在本例中就介绍一下怎样制作类似于 Office 菜单的菜单，以便用户可以顺利地完成对文本编程的各种操作。

图2-1 Word 的菜单栏

如图 2-1 所示是 Word 的菜单栏，包括文件、编辑、视图、插入、格式、工具、表格、窗口、帮助几个主菜单选项。在这里主要说明文件窗口的下拉菜单以及编辑窗口下拉菜单中内容的实现。现在首先让我们来看一下这两个下拉菜单的内容。如图 2-2 所示是 Word 的文件下拉菜单，在这里包括了新建、打开、保存、另存为、退出等几项内容。

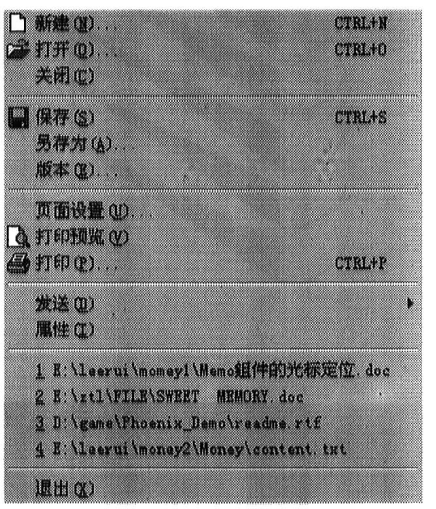


图2-2 Word 的文件菜单

图 2-3 所示为编辑菜单的菜单项，它包括撤消、剪切、复制、粘贴、查找和替换等几项内容。

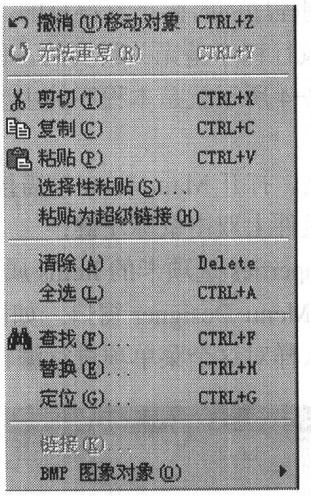


图2–3　Word 的编辑菜单

下面就通过介绍具体程序的编制，来说明这些菜单项的功能是如何实现的。

实 现 过 程

☛ 关于控件属性的说明

首先应设置好控件，然后从 Standard 选项卡中把 MainMenu 控件拖放在 Form1 上，这时，Forml 中就可以看到 MainMenu 的图标，这个 MainMenu 控件就是用来控制菜单的格式的。然后再应用两个对话框控件：一个是 Dialogs 选项卡中的 OpenDialog 控件，另外一个是 Dialogs 选项卡中的 SaveDialog 控件，它们分别控制的是打开文件的对话框和保存文件的对话框。把这两个控件都拖放到适当的位置放好。这里的三个控件——MainMenu、OpenDialog 和 SaveDialog

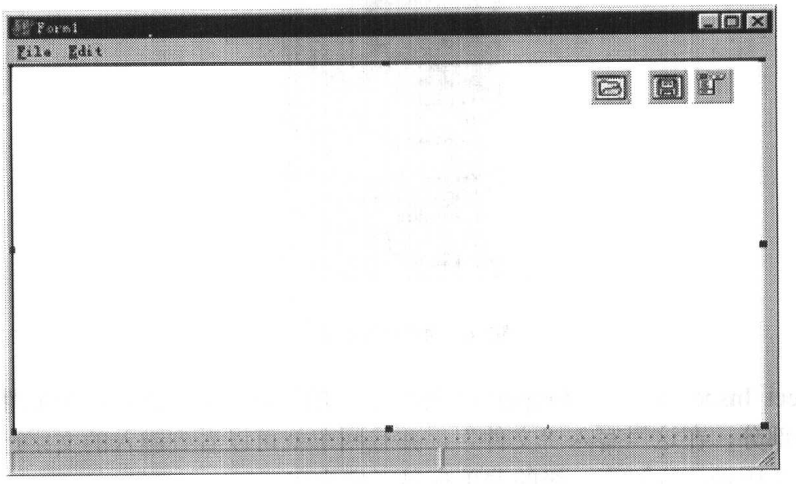


图2–4　实例控件图

都是隐藏的控件，也就是说在编制程序的时候是看得见的，而在运行时却是看不见的。因此，这几个控件放在什么地方对于可执行程序是无关紧要的。但是为了在编程序时好看一些，也要把它们放到适当的位置。如图 2-4 所示就是本程序的主要控件图。

1. 主菜单中菜单项的设置

现在来设置 MainMenu 的属性。打开 MainMenu 只需按下列两种方法之一进行操作：

- 在表单中的 MainMenu 组件上双击鼠标左键；
- 在 Object Inspector 中 Properties 选项卡的 Items 属性中双击（或单击省略号）。

于是 Delphi 就会显示出 MainMenu Designer 窗口，如图 2-5 所示。这时就可以看到已经有一个菜单项被激活了，但是怎么样对这个菜单项进行编辑呢？

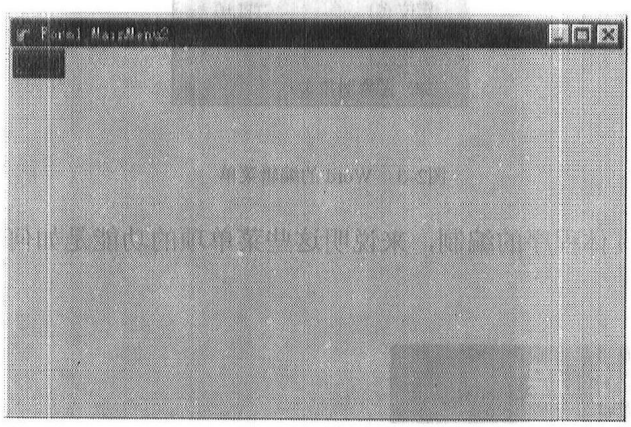


图2-5　菜单编辑器

如图 2-6 所示为菜单项的属性设置栏。

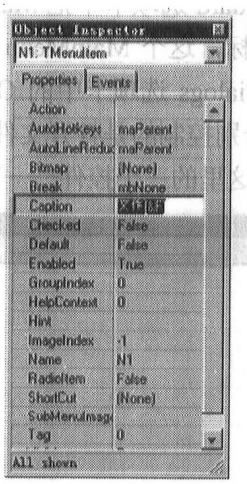


图2-6　菜单项属性列表

选中 Object Inspector 中的 Properties 选项卡中的 Caption 属性，这个属性表示的就是菜单项里的文字部分，在这里写入“文件”，并且回车就可以看到键入的文字已经出现在菜单中了。如图 2-7 所示，这时下一级的菜单也显示出来了。