地理信息系统二次开发教程
--组件篇

清华大学出版社

第8章 地理编码

地理信息系统的数据库中存在大量的地址及地点名称等地理信息。数据库中可以没有诸如纬度值、经度值或图像坐标之类的空间信息，但却存有大量有关地理邮政方面的数据。计算机可根据地址确定的一些位置自动地建立一个绘图层，并能自动访问到大量地图中的属性信息。MapObjects 中包括一系列 OLE Automation 对象，这些对象是专为地址匹配及根据地点名称查找定位任务而设计的。地址匹配是指在地图上找到并标明每条地址所对应的位置，它也称为地理编码。

GeoCoder 对象要求指定一个 GeoDataset 对象，其中包括街道中心线、地址变动范围以及分别适用于单个和成批地址匹配的方法的设定。AddressLocation 对象中的代码具有表明一条地址是否被解析以及如何被解析的作用，如果地址已被解析，它还能显示匹配地址的地理位置。

8.1 匹配地址

GeoCoder、AddressLocation 和 Standardizer 对象用于交互式和批地址匹配。首先必须了解街道文件成立的具体要求，然后将讨论如何使用这些对象进行地址匹配。

8.1.1 用于地址匹配的专用文件

要进行地址匹配操作，就必须有一个街道文件。街道文件中包括具有地址信息的街道中心线段。街道文件可以是一个 Shape 文件，也可以是一个 SDE 层。一个适用于地址匹配的街道文件必须满足以下三个基本条件：

（1）街道文件中必须有街道中心线段，并在街道交叉处形成清晰的结点。除了街道中心线段外，街道文件中不能再有其他任何特征。

（2）这些中心线段中的每一段都必须至少具备以下五个要素（字段）：街道名称、街道左边的起始地址、街道右边的起始地址、街道左边的结束地址与街道右边的结束地址。

（3）右边的一对地址和左边的一对地址必须分别编上双号和单号。也就是说，任何一边的起点和终点地址必须同时为双号或同时为单号，而每一对左右两边的地址编号则形成互补，即一单一双。显然，一条道路的左和右取决于前进的方向，街道文件则遵循以下的协定：街道的左右视起点到终点的方向而定。

MapObjects 中的地址匹配目标是为美国使用而设计的，但只要能够使用符合上述要求的街道文件，即使在世界其他地区，也能够使用它们进行地址匹配。不过街道两边的地址编号必须是一边为双、一边为单。

另外，地址匹配对象也适用于中等长度的街道文件，但它们并不能设计成为大范围内

的国家街道文件。如果街道文件确实跨越了一个很大的地理区域，那么可以自由使用一些地址中的城市名、邮政编码及乡村名以帮助解决地址匹配问题。

下面是街道文件的图解一览表：

（1）街道文件是由具有规定的左、右两边及起点、终点特征的中央线路组成。其中左、右的区分取决于由起点到终点的方向。

（2）地址数据是按各个路段储存的。储存的 4 个数值有 Left-From（左起点）、Right-From（右起点）、Left-To（左终点）和 Right-To（右终点）。在美国，街道一边的地址号为双数，而另一边则为单数。

（3）地址匹配或地理编码，就是一个通过地址中某路段的起始、终了位置，并同时考虑到单双号因素以确定地理位置的过程。

（4）如果指定了一个分支，那么估算出的地址位置将落在该路段垂直线上的左边或右边。

（5）除了返回一个位置，地址匹配还返回标准化地址数据值，包括房号、街道名、首字符串、尾字符串和邮区号码。

8.1.2 街道绘制文件

当利用街道文件和地址匹配工作时，如果使描绘道路的详细设计图层可见，而使用于地址匹配的街道中心线图层不可见，那么就能绘制出一个精彩的界面。然而，那些绘有详细的街道边的设计图层通常并不很实用。这里可提供帮助，即在实际运用过程中使绘制出的街道文件实现的技巧。

表格左边的街道文件是用一根简单的 1 个单位宽度的细实线绘制的，而表格右边,同样的街道文件被绘制了两次。首先用 5 个单位宽的粗黑实线绘制一次，然后再用 3 个单位宽的白实线绘制一次，将两个图层重叠起来就形成了双线道路的效果。

只要用几分钟的时间，便可以在地图窗口自己绘制出这张图。绘制两张同一街道文件的图层，如上所述分别使用不同颜色及粗细的线条，然后叠加起来，将黑线的图层放在下面。

如果用这种方法绘图，建议最好仅在某一区域内使用双线，以免交叉引起混乱。在更小的范围内，用单线绘制即可。通过在图像控制器的 BeforeLayerDraw 操作中编写代码来估算当前图形范围、定义符号对象（这些符号对象与街道文件的矢量图层对象相关联）的可见性及大小尺寸，就可以有条理地绘制出一个街道文件。

如果使用 ValueMapRenderer 对象将道路分类绘制，如公路用较粗的线绘制等等，那么就能更有效地利用这些代码。这样绘制街道文件时，一定要注意规定好不同线条各自的比例尺寸。

8.1.3 GeoCoder 对象

GeoCoder 对象用在街道网络中，根据地址、路口或地址列表等信息进行定位。

StreetTable（街道表）属性设置街道网络的地理数据集。如果 StreetTable 是第一次使用，可以使用 AddIndex 和 BuildIndices 方法产生地理数据集的地理索引，用 IndexStatus 属性值可以验证是否建立了索引。SearchQueries 属性规定了如何搜索索引。Valid 属性表示

StreetTable 中的字段是否满足要求。如果无效，可以根据 LastError 属性检查错误原因。

在进行地址匹配之前，需要将地址进行标准化。在对一地址进行编码之前，必须创建一 Standardizer 对象，然后将该对象赋予 GeoCoder 对象的 Standardizer 属性。

MapObjects 支持匹配不同类型的地址。通过设置 MatchRules 和 IntersectionMatchRules 属性为某个文件，这样可以选择合适的匹配规则。在 MapObjects 的安装路径中的 Georules 目录中有一些匹配规则文件。

给定一街道或路口地址（两个街道名之间用“ & ”符号隔开）后，可以调用 GenerateCandidates 方法得到可能的匹配结果，接着调用 LocateCandidate 方法将返回的匹配结果与最佳候选地址进行匹配，该方法返回一 AddressLocation 对象。

如果在一个表中有许多地址需要匹配，则可以调用 BatchMatch 方法对每一个记录进行匹配，并创建一包含匹配结果的 Shape 文件。

在进行批地址匹配中，通过调整 MatchWhenAmbiguous 和 MinimumMatchScore 两个属性值，可以控制匹配精度。还可以设置其他属性，例如 SpellingSensitivity，Offset 和 SqueezeFactor 属性，在交互匹配和批匹配中控制地理编码方式。

GeoCoder 对象是可创建对象，在 Visual Basic 可使用如下语句来创建一新 GeoCoder 对象。

```
Dim geo as New MapObjects2.Geocoder
```

8.1.4　AddressLocation 对象

AddressLocation 对象描述地址匹配的结果。MatchCode 属性表示匹配的状态。如果匹配成功，Location 属性指向一个 Point 对象。

Location 属性表示匹配位置的点对象，它是一个只读对象。当成功调用 GeoCoder 对象的 LocateCandidate 方法后，返回一 AddressLocation 对象。该对象通过 Location 属性显示在地图中。如果设置了 GeoCoder 对象的 SqueezeFactor 和 Offset 属性，那么编码后的 Location 会受上述两个属性的影响。

MatchScore 属性表示地址匹配程度的好坏。如果 MatchScore 属性值为 100，表示完全匹配，如果为 0，表示没有任何匹配结果。通常情况下，MatchScore 属性值在 30~70 之间。

StreetSide 属性表示匹配的地址是在街道的哪一边，它可以是 moLeftSide（街道的左边）或 moRightSide（街道的右边）。

8.1.5　Standardizer 对象

Standardizer 对象用于将单个地址字符串或街道路口进行标准化。

在进行地址匹配之前，地址字符串必须经历两个标准化程序。首先是将地址字符串分割成不同的字段，然后这些不同字段转化成适当的标准值，例如 N、North 或 Nrth 等。标准化以后的地址的各个字段才能与街道表中的字段进行比较，找到相应的地理位置。

在创建 Standardizer 对象时，通过该对象 StandardizingRules 属性指定合适的标准化规则。MapObjects 提供了一系列标准化规则，这些规则保存在.stn 文件中。将该属性指定一合适的.stn 文件便可以进行标准化。

.stn 文件需要依据文件名寻找其他文件。一些文件甚至指向其他的表，例如 prefix.tbl。因此，必须清楚进行标准化时使用的一系列规则文件，并把这一系列文件放在

StandardizingRules 属性指定的.stn 文件所在路径中。通常，标准化地址时也需要扩展名为.dct，.mat，.pat 和.cls 的文件。

Standardizer 对象通常使用一个标准规则来标准化地址。而标准化规则文件可以通过 StandardizingRules 属性指定。但是如果需要对街道交叉口进行编码，还必须设置 IntersectionStandardizingRules 属性。如果不需要考虑道路交叉口，该属性可以为空。

下面是创建和设置 Standardizer 对象的 Visual Basic 代码：

```
Dim stan as New MapObjects2.Standardizer
stan.StandardizingRules = "c:\esri\mapobjects\Georules\us_addr.stn"
stan.IntersectionStandardizingRules = "c:\esri\mapobjects\Georules\us
_intsc.stn"
```

通过 Valid 属性可以检查 Standardizer 对象是否设置正确。如果设置不正确，可以根据 LastError 属性检查错误原因。

Standardizer 对象的惟一方法是 StandardizeAddress，该方法根据 StandardizingRules 属性或 IntersectionStandardizingRules 属性指定的规则，将地址进行标准化。通过 FieldValue 属性可以得到标准化的结果。

实例 8.1 演示了如何在 C++ Builder 中使用 Standardizer 对象和查看标准化地址的结果。

[实例 8.1] 新建一工程。在窗体 Form1 中加入一 TButton 控件、一 TEdit 控件和一 TListBox 控件。双击 Button1，创建该控件的 OnClick 事件响应函数，在其中加入如下所示的代码。

```
//---------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  IMoStandardizerPtr stan;
  stan = (IDispatch*)CreateOleObject("MapObjects2.Standardizer");
  stan->StandardizingRules = WideString("E:\Program Files\ESRI\MapObjects2
  \ GeoRules\us_addr.stn");
  stan->IntersectionStandardizingRules = WideString("E:\Program Files-
  \ESRI\MapObjects2\GeoRules\us_intsc.stn");
  if(!bool(stan->Valid))
  {
    if(stan->LastError == mgErrorNone)
        MessageBox(NULL,"The standardizer is not valid. Error ", "Error",
MB_OK);
    return;
  }

  //可以将地址设置为270 North Main Avenue，North Main Street&First Ave SW等
  AnsiString address1 = Edit1->Text;//""
  if(stan->StandardizeAddress(WideString(address1)))
  {
    stan->set_FieldValue(WideString("ZN"), WideString("53702"));
```

```
    int f = stan->FieldCount;
    for(int i=0; i<f; i++)
    {
      WideString name = stan->get_FieldName(i);
      ListBox1->Items->Add(stan->get_FieldValue(name));
    }
  }
}
//---------------------------------------------------------------------
void __fastcall TForm1::Edit1KeyDown(TObject *Sender, WORD &Key,
     TShiftState Shift)
{
  if(Key == VK_RETURN)
    Button1Click(Sender);
}
//---------------------------------------------------------------------
```

编译并运行程序。在文本框中输入一地址，然后按 Enter 键或单击“解析地址”命令按钮，便可在列表框中得到标准化后的地址，如图 8.1 所示。

图 8.1　程序运行界面

8.1.6　交互式地址匹配

下面是 MapObjects 中进行交互式地址匹配的步骤：

（1）创建一 Standardizer 对象，指定该对象的标准化规则，即指定 StandardizingRules 和 IntersectionStandardizingRules 为某个标准化规则文件名，并将该对象赋予 GeoCoder 对象的 Standardizer 属性。

（2）为 GeoCoder 对象指定一个街道网络作为 StreetTable 的属性，它是一个 GeoDataset 对象，包括街道中心线。这些街道都有一组标准化字段提供街道及地址信息。

（3）检查 IndexStatus 属性以确认 StreetTable 的地理编码索引是否已建立。如果没有，

则利用 AddIndex 方法或 BuildIndices 方法建立索引。

（4）检查 Valid 属性以确认 StreetTable 和指定地址字段（未显示在这张图形中）是否符合地址匹配的要求。

（5）利用 GeoCoder 对象的 SearchQueries 属性设置查询条件。

（6）调用 GeoCoder 对象的 GenerateCandidates 方法进行地址匹配。该方法将搜索到的地址按照它们的得分从高到低排列。可以通过 LocateCandidate 方法得到指定位置处的匹配结果，该结果是一 AddressLocation 对象，该对象的 MatchScore 属性反映了地址匹配精度。

下面这个实例演示了如何在 Visual Basic 中进行交互式地址匹配。

[实例 8.2] 新建一工程。在窗体 Form1 中加入一 CommandButton 控件、两个 Edit 控件和一 Map 控件。

切换到代码编辑窗口。在其中加入如下所示的代码。

```
'------------------------------------------------------------------------
Option Explicit

Dim geo As New MapObjects2.Geocoder
Dim stan As New MapObjects2.Standardizer
Dim extRect As MapObjects2.Rectangle
'设置StreetTable中字段的变量
Private Const m_FromLeft = "L_f_add"
Private Const m_FromRight = "R_f_add"
Private Const m_ToLeft = "L_t_add"
Private Const m_ToRight = "R_t_add"
Private Const m_PreDir = "Prefix"
Private Const m_PreType = "Pre_type"

Private Const m_StreetName = "Name"
Private Const m_StreetType = "Type"
Private Const m_SufDir = "Suffix"
Private Const m_LeftZone = "Zipl"
Private Const m_RightZone = "ZipR"
'------------------------------------------------------------------------
Private Sub Command1_Click()
  '寻找最佳匹配地址
  Dim foundLoc As MapObjects2.AddressLocation

  If stan.StandardizeAddress(Text1.Text) Then
    stan.FieldValue("ZN") = Text2.Text
    Dim result As Integer
    result = geo.GenerateCandidates()
    If geo.CandidateCount > 0 Then
      Set foundLoc = geo.LocateCandidate(0)
```

```
      Dim score As Integer
      score = foundLoc.MatchScore
      Map1.FlashShape foundLoc.location, 3  '闪烁三次
    End If
  End If
End Sub
'--------------------------------------------------------------------
Private Sub Form_Load()
  Dim dc As New MapObjects2.DataConnection
  Dim gd As Object
  Dim lyr As New MapObjects2.MapLayer
  Dim f, i As Integer
  Dim name As String

  '设置Standardizer对象
  stan.StandardizingRules = "E:\Program Files\ESRI\MapObjects2\GeoRules
   \us_addr.stn"
  stan.IntersectionStandardizingRules="E:\Program Files\ESRI\MapObjects2\
GeoRules\us_intsc.stn"
  '将GeoCoder对象的Standardizer属性设置为stan
  geo.Standardizer = stan

  dc.Database = "E:\Program Files\ESRI\MapObjects2\Samples\Data\Redlands"

  dc.Connect
  If Not dc.Connected Then
    MsgBox "dc.connected error"
    End
  End If

  '设置GeoCoder对象StreetTable属性
  Set gd = dc.FindGeoDataset("redlands")
  lyr.GeoDataset = gd
  lyr.Symbol.Color = moBlue
  Map1.Layers.Add lyr
  geo.StreetTable = gd

  '设置匹配规则和变量
  geo.MatchRules="E:\ProgramFiles\ESRI\MapObjects2\GeoRules\us_ addr1
.mat"
  geo.IntersectionMatchRules="E:\Program  Files\ESRI\MapObjects2\GeoRules
                          \us_intsc1.mat"
```

```
  ' 将匹配变量和StreetTable中的字段连接起来
  geo.MatchVariableField("FromLeft") = m_FromLeft
  geo.MatchVariableField("FromRight") = m_FromRight
  geo.MatchVariableField("ToLeft") = m_ToLeft
  geo.MatchVariableField("ToRight") = m_ToRight
  geo.MatchVariableField("PreDir") = m_PreDir
  geo.MatchVariableField("PreType") = m_PreType
  geo.MatchVariableField("StreetName") = m_StreetName
  geo.MatchVariableField("StreetType") = m_StreetType
  geo.MatchVariableField("SufDir") = m_SufDir
  geo.MatchVariableField("LeftZone") = m_LeftZone
  geo.MatchVariableField("RightZone") = m_RightZone

  ' 连接道路交叉口第一组变量
  geo.MatchVariableIntersectionLink("PreDir", mgLinkPrimary) = "PreDir1"
  geo.MatchVariableIntersectionLink("PreType", mgLinkPrimary) = "PreType1"
  geo.MatchVariableIntersectionLink("StreetName", mgLinkPrimary) =
"StreetName1"
  geo.MatchVariableIntersectionLink("StreetType", mgLinkPrimary) =
"StreetType1"
  geo.MatchVariableIntersectionLink("SufDir", mgLinkPrimary) = "SufDir1"
  geo.MatchVariableIntersectionLink("LeftZone", mgLinkPrimary) =
"LeftZone1"
  geo.MatchVariableIntersectionLink("RightZone", mgLinkPrimary) =
"RightZone1"

  ' 连接道路交叉口第二组变量
  geo.MatchVariableIntersectionLink("PreDir", mgLinkSecondary) = "PreDir2"
  geo.MatchVariableIntersectionLink("PreType", mgLinkSecondary) =
"PreType2"
  geo.MatchVariableIntersectionLink("StreetName", mgLinkSecondary) =
"StreetName2"
  geo.MatchVariableIntersectionLink("StreetType", mgLinkSecondary) =
"StreetType2"
  geo.MatchVariableIntersectionLink("SufDir", mgLinkSecondary) = "SufDir2"
  geo.MatchVariableIntersectionLink("LeftZone", mgLinkSecondary) =
"LeftZone2"
  geo.MatchVariableIntersectionLink("RightZone", mgLinkSecondary) =
"RightZone2"

  '如果没有索引，则先创建索引
  If Not geo.IndexStatus = MapObjects2.IndexStatusConstants.mgIndexExists
Then
```

```
    If Not geo.AddIndex(m_StreetName, "", mgIndexTypeSoundex) Then
      MsgBox "不能创建索引", vbCritical
      End
    End If

    If Not geo.AddIndex(m_LeftZone, m_RightZone, mgIndexTypeNormal) Then
      MsgBox "不能创建索引.", vbCritical
      End
    End If

    If Not geo.BuildIndices(True) Then
      MsgBox "不能创建索引.", vbCritical
      End
    Else
      MsgBox "索引创建成功."
    End If
  End If

  '设置搜索查询
  Dim queries As New MapObjects2.Strings
  queries.Add "SN? & ZN"
  queries.Add "SN?"
  Set geo.SearchQueries = queries

  Command1.Caption = "寻找地址"
  Text1.Text = "260 Cajon St"
  Text2.Text = "92373"

End Sub
'--------------------------------------------------------------------
Private Sub Form_Resize()
  Map1.Move Map1.Left, Map1.Top, ScaleWidth - Map1.Left, ScaleHeight -
Map1.Top
End Sub
'--------------------------------------------------------------------
Private Sub Map1_MouseDown(Button As Integer, Shift As Integer, X As Single,
Y As Single)
  If Button = vbLeftButton Then
    If Shift = 0 Then
      Set Map1.extent = Map1.TrackRectangle
    ElseIf Shift = 1 Then
      Set extRect = Map1.extent
      extRect.ScaleRectangle 1.5
```

```
    Map1.extent = extRect
  End If
 ElseIf Button = vbRightButton Then
  Map1.Pan
 End If
End Sub
'--------------------------------------------------------------------
```

编译并运行程序。应用程序执行界面如图 8.2 所示。

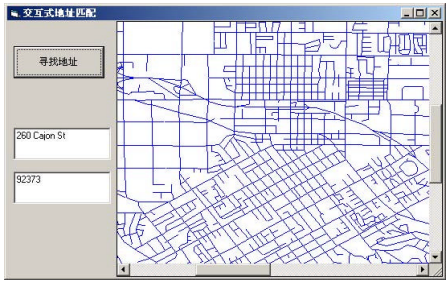

图 8.2　应用程序运行界面

8.1.7　批地址匹配

GenerateCandidates 方法适用于单个地址匹配，而对于一次进行大量地址匹配就不那么得心应手了。对于批地址匹配，一般使用 GeoCoder 对象 BatchMatch 方法。步骤如下：

（1）建立一个新的 DataConnection 对象，将其 Database 属性设置为包含 Shape 文件或 SDE 数据库的路径名称。

（2）建立一个 GeoCoder 对象，将其 StreetTable 属性设置在带有地址信息的 GeoDataset 对象。

（3）检查 IndexStatus 属性以确认 StreetTable 的地理编码索引是否已建立。如果没有，则利用 AddIndex 方法或 BuildIndices 方法建立索引。

（4）建立一个 Table 对象，并将 Database 和 Name 属性设置到带有需要匹配的地址的一个 ODBC 数据源中。

（5）利用 BatchMatch 方法创建一同名的表，该表包括地址以及一个包含需要匹配地址的字段。这种方式将利用所指定的 DataConnection 对象创建一张新表，它的名字由程序指定。该方法原型如下：

```
BatchMatch(addressTable, addressField, dataConnection, outputTable,
streetFields)
```

其中参数 StreetFields 的字符串集所指定的字段是需要从 addressTable（地址表）复制到 outputTable（输入表）中的字段。

当用 BatchMatch 进行地址匹配时，是否需要一个有矢量图层的地图控件是任意的。可以就用 OLE 自动化对象进行地址匹配操作，而无需什么可见的图层。

下面这个实例演示了如何在 Visual Basic 中利用 BatchMatch 和 BatchMatchVariableField 属性来匹配一文件中的所有地址。

[实例 8.3]　新建一工程。在窗体 Form1 中加入一 CommandButton 控件、一 CommandDlg 控件和一 Map 控件。切换到代码编辑窗口中，在其中加入如下所示的代码。

```
'-----------------------------------------------------------------------
Option Explicit

Dim geo As New MapObjects2.Geocoder
Dim stan As New MapObjects2.Standardizer
Dim dcx As New MapObjects2.DataConnection
Dim extRect As MapObjects2.Rectangle

'设置StreetTable表中字段变量
Private Const m_FromLeft = "L_f_add"
Private Const m_FromRight = "R_f_add"
Private Const m_ToLeft = "L_t_add"
Private Const m_ToRight = "R_t_add"
Private Const m_PreDir = "Prefix"

Private Const m_PreType = "Pre_type"
Private Const m_StreetName = "Name"
Private Const m_StreetType = "Type"
Private Const m_SufDir = "Suffix"
Private Const m_LeftZone = "Zipl"
Private Const m_RightZone = "ZipR"
'-----------------------------------------------------------------------
Private Sub Command1_Click()
  Dim AddressTable As New MapObjects2.Table
  Dim ml As New MapObjects2.MapLayer
  Dim ft As String, fn As String
  Dim count As Integer

  geo.MinimumMatchScore = 70
  geo.MatchWhenAmbiguous = True

  With CommonDialog1

    .FileName = ""
```

```
    .Filter = "dBASE (*.dbf)|*.dbf"
    .DefaultExt = "*.dbf"
    .CancelError = True
    On Error Resume Next
    .ShowOpen
    If Err.Number = cdlCancel Then
      Unload Me
    End If
    ft = .FileTitle
    fn = .FileName
  End With

  AddressTable.Database = "CustomerDB"      ' 设置ODBC数据源名称
  AddressTable.name = Left(ft, Len(ft) - 4) ' 不带扩展名的文件名

  dcx.Database = Left(fn, Len(fn) - Len(ft) - 1) ' 指定输出表所在路径

  If AddressTable.name <> "" And dcx.Connect Then
    geo.BatchMatchVariableField("LeftZone") = "ZIPl"
    geo.BatchMatchVariableField("RightZone") = "ZIPr"

   count = geo.BatchMatch(AddressTable, "ADDRESS", dcx, "results", Nothing)
    MsgBox "Successfully geocoded " & count & " addresses!"

    Debug.Print AddressTable.Records.count

    If count <> 0 Then
      Set ml.GeoDataset = dcx.FindGeoDataset("results")
      Map1.Layers.Add ml
      With Map1.Layers("results").Symbol
        .SymbolType = moPointSymbol
        .Color = moRed
        .Style = moCircleMarker
      End With
      Map1.Refresh
    End If

  Else
    MsgBox "Didn't match addresses in dbf file"
  End If

End Sub
'--------------------------------------------------------------------
```

```
Private Sub Form_Load()
  Dim dc As New MapObjects2.DataConnection
  Dim gd As Object
  Dim lyr As New MapLayer
  Dim f, i As Integer
  Dim name As String

  '设置Standardizer对象的属性
  stan.StandardizingRules="E:\Program Files\ESRI\MapObjects2\GeoRules\us_
                           addr.stn"
  stan.IntersectionStandardizingRules="E:\Program Files\ESRI\MapObjects2\
                                       GeoRules\us_intsc.stn"

  '指定GeoCoder对象的Standardizer属性
  Set geo.Standardizer = stan

  dc.Database = "E:\Program Files\ESRI\MapObjects2\Samples\Data\Redlands"
  dc.Connect
  If Not dc.Connected Then
    MsgBox "dc.connected error"
    End
  End If

  '设置StreetTable属性
  Set gd = dc.FindGeoDataset("redlands")
  lyr.GeoDataset = gd
  lyr.Symbol.Color = moBlue
  Map1.Layers.Add lyr
  geo.StreetTable = gd

  '设置匹配规则和变量
  geo.MatchRules="E:\ProgramFiles\ESRI\MapObjects2\GeoRules\us_addr1.mat"
  geo.IntersectionMatchRules="E:\Program  Files\ESRI\MapObjects2\GeoRules
                              \us_intsc1.mat"

  ' 将匹配变量与StreetTable表中的字段相连
  geo.MatchVariableField("FromLeft") = m_FromLeft
  geo.MatchVariableField("FromRight") = m_FromRight
  geo.MatchVariableField("ToLeft") = m_ToLeft
  geo.MatchVariableField("ToRight") = m_ToRight

  geo.MatchVariableField("PreDir") = m_PreDir
  geo.MatchVariableField("PreType") = m_PreType
```

```
  geo.MatchVariableField("StreetName") = m_StreetName
  geo.MatchVariableField("StreetType") = m_StreetType
  geo.MatchVariableField("SufDir") = m_SufDir
  geo.MatchVariableField("LeftZone") = m_LeftZone
  geo.MatchVariableField("RightZone") = m_RightZone


  geo.MatchVariableIntersectionLink("PreDir", mgLinkPrimary) = "PreDir1"
  geo.MatchVariableIntersectionLink("PreType", mgLinkPrimary) = "PreType1"
  geo.MatchVariableIntersectionLink("StreetName", mgLinkPrimary) =
"StreetName1"
  geo.MatchVariableIntersectionLink("StreetType", mgLinkPrimary) =
"StreetType1"
  geo.MatchVariableIntersectionLink("SufDir", mgLinkPrimary) = "SufDir1"
  geo.MatchVariableIntersectionLink("LeftZone", mgLinkPrimary) =
"LeftZone1"
  geo.MatchVariableIntersectionLink("RightZone", mgLinkPrimary) =
"RightZone1"

  geo.MatchVariableIntersectionLink("PreDir", mgLinkSecondary) = "PreDir2"
  geo.MatchVariableIntersectionLink("PreType", mgLinkSecondary) =
"PreType2"
  geo.MatchVariableIntersectionLink("StreetName", mgLinkSecondary) =
"StreetName2"
  geo.MatchVariableIntersectionLink("StreetType", mgLinkSecondary) =
"StreetType2"
  geo.MatchVariableIntersectionLink("SufDir", mgLinkSecondary) = "SufDir2"
  geo.MatchVariableIntersectionLink("LeftZone", mgLinkSecondary) =
"LeftZone2"
  geo.MatchVariableIntersectionLink("RightZone", mgLinkSecondary) =
"RightZone2"

  '判断是否建立了索引，如果没有则创建索引
  If Not geo.IndexStatus = MapObjects2.IndexStatusConstants.mgIndexExists
Then
     If Not geo.AddIndex(m_StreetName, "", mgIndexTypeSoundex) Then
       MsgBox "不能创建索引.", vbCritical
       End
     End If

     If Not geo.AddIndex(m_LeftZone, m_RightZone, mgIndexTypeNormal) Then
       MsgBox "不能创建索引.", vbCritical
       End
```

```
    End If

    If Not geo.BuildIndices(True) Then
      MsgBox "不能创建索引.", vbCritical
      End
    Else
      MsgBox "成功创建索引."
    End If
  End If

  '设置搜索查询
  Dim queries As New MapObjects2.Strings
  queries.Add "SN? & ZN"
  queries.Add "SN?"
  Set geo.SearchQueries = queries

  Command1.Caption = "Addresses"

End Sub
'--------------------------------------------------------------------------
Private Sub Form_Resize()
  Map1.Move Map1.Left, Map1.Top, ScaleWidth - Map1.Left, ScaleHeight -
 Map1.Top
End Sub
'--------------------------------------------------------------------------
Private Sub Map1_MouseDown(Button As Integer, Shift As Integer, X As Single,
Y As Single)
  If Button = vbLeftButton Then
    If Shift = 0 Then
      Set Map1.Extent = Map1.TrackRectangle
    ElseIf Shift = 1 Then
      Set extRect = Map1.Extent
      extRect.ScaleRectangle 1.5
      Map1.Extent = extRect
    End If
  ElseIf Button = vbRightButton Then
    Map1.Pan
  End If
End Sub
'--------------------------------------------------------------------------
```

8.2 定 位 查 找

不用其他地址匹配对象而独立使用 PlaceLocater 对象也可以返回地点名称所在的位置。可以单独为 PlaceLocater 对象建立一个街道文件，但是实际上只要是含有地点名称字段的 GeoDataset 对象都能接受。

可以利用 PlaceLocater 对象建立的这种工具有时也叫做地名一览表，用一个引导类型定位一个地名，光标就会自动移到指定的位置。

PlaceLocator 对象用于匹配在 PlaceNameTable（地点名称数据表）属性中规定的地理数据库的地点名。一旦建立了地点名称数据表，便可以利用 BuildIndex 方法对字段建立索引，以便匹配。Indexed 属性表示是否已经建立好索引。Locate 方法匹配地点名，返回一个包含匹配后的地理位置的点集对象。另外，FindApproximateMatches 利用近似的方法匹配给定的地址。FindAllPlaceNames 方法搜索所有以给定字符开始的地点名。这两个方法返回一个 Strings 集合对象。

以下是使用 PlaceLocater 对象的一般步骤：

（1）为 PlaceLocater 对象的 PlaceNameTable 属性指定一个 GeoDataset 对象，该 GeoDataset 只要包含有地名字段即可。

（2）利用 BuildIndex 方法为地名建立索引，并检查 Indexde 属性，以确定该索引是否已经建立。

（3）利用如下 PlaceLocater 对象中的一种方法：

- FindAllPlaceNames ——利用该方法可返回所有以指定子字符串开头的地名字符串集；
- FindApproximateMatches——利用该方法返回与指定地名相似的地名字符串集；
- Location——利用该方法可返回有指定地名特征的所有位置的点集。

使用 GeoDatasets 时，点、线和多边形都可作为 PlaceLocater 对象的 PlaceNameTable 的属性。如果 GeoDataset 包含的是线特征，则 Locate 方法返回的位置都是沿线的起点；如果 GeoDataset 包含的是多边形特征，则 Locate 方式返回的位置是多边形自己的中心位置。

下面这个实例演示了如何在 Visual Basic 中使用 PlaceLocater 对象进行地址查找。

[实例 8.4] 新建一工程。在窗体 Form1 中加入一 Map 控件、一 CommandButton 控件、一 TextBox 控件、一 CheckBox 控件和一 ListBox 控件。

切换到代码编辑窗口中，在其中加入如下所示的代码。

```
'-----------------------------------------------------------------------
Option Explicit
Dim pl As New MapObjects2.PlaceLocator
Dim pts As MapObjects2.Points
Dim gdname As String
Dim fldname As String
Public places As New Collection
Dim extRect As MapObjects2.Rectangle
```