基于MapObjects的图层文件编辑器

Shape File Editor Based on MapObjects

吕丹阳（中国科学院计算技术研究所 100080）

摘 要：基于MapObjects的应用中，经常会遇到对图层元素修改的问题。有很多软件可以完成这个功能，但由于这些软件不能再次自由分发，也不能方便的嵌入到应用系统中，因此我们创建了基于MapObjects的图层元素编辑器。本文介绍了基于MapObjects的图层文件编辑器的系统设计方案，并对实现过程中的几个关键性技术问题及其解决方法作了详细地介绍。

关键词：GIS MapObjects Shape文件

1 引言

MapObjects是ESRI公司出品的一组供应用开发人员使用的制图与GIS功能组件。它由一个OLE控件和一系列可编程OLE对象组成。利用MapObjects，开发人员可以在应用程序中添加制图和GIS功能。在基于MapObjects的应用中，经常会遇到对某些图层元素进行修改的问题。ESRI本身没有提供实现图形元素编辑等高级功能。尽管我们可以使用许多软件编辑图层元素、更新图层文件，如ESRI公司的ARC/INFO、ArcView等软件，但由于这些软件不能再次自由分发，也不能方便的嵌入到应用系统中，因此我们创建了基于MapObjects的图层文件编辑器。本文介绍了基于MapObjects的图层文件编辑器的系统设计方案，并对实现过程中的几个关键性技术问题及其解决方法作了详细地论述。

2 系统设计方案

在MapObjects中，一个综合性的地图由多个图层构成，图层数据来源广泛，既可以是GIS矢量图层，也可以是CAD图层，甚至影像数据。对于GIS和CAD的矢量图层，MapObjects在内部统一用记录集（Recordset）来表达。Shape文件格式是ESRI提供的存储地理数据的矢量格式，它已经成为GIS业界事实上的基于桌面应用的标准，因此我们以Shape文件为例说明如何创建基于MapObjects的图层文件编辑器。基于MapObjects的图层文件编辑器实现的功能包括新建图层文件、打开图层文件、编辑图层文件、更新图层文件、定位图层元素、缩放图层文件等，系统功能模块如图1所示：

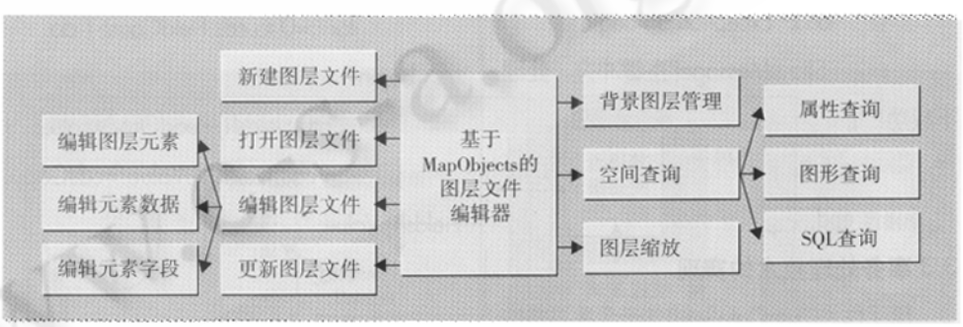

图1 图层编辑器的系统功能模块

Shape文件中有三种类型的图层元素：点(Point)、线(Line)和面(Polygon)，但在同一个Shape文件中仅能有某一种类型的图层元素。由于MapObjects没有直接提供对图形元素的编辑，因此在实现这个图形元素编辑器时，我们采用了一种新的方法，即在打开图层文件时将图形元素分别读入到图形元素数组中，而不是将整个图形文件直接加载到MapObjects中，通过分别绘制每个图形元素来实现图形文件的显示。在对图形元素编辑过程中仅修改内存中的图层元素数组，直到更新图层文件时才将内存中的图层元素更新到磁盘中的图层文件。基于MapObjects的图层文件编辑器采用Visual C++[2]作为开发环境，在系统的主窗口中使用四个窗体分别显示当前正在编辑的图层文件及其元素数据、图例和定位图，可以拖动分割条设置各个窗体的大小，其系统界面如图2所示：

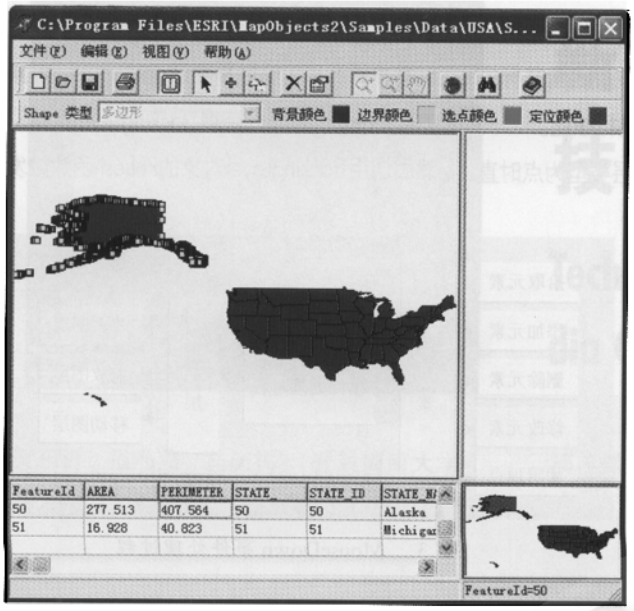

图 2　Shape 文件编辑器的系统界面

3 几个关键性技术问题及其解决方法

3.1 打开图层文件

这主要涉及到如何打开图层文件和浏览记录集。

第一步：首先创建图层对象(MapLayer)，对将要打开的图层文件的全路径名进行分析以获得文件名和路径名，创建数据连接对象(DataConnection)。然后设置数据连接对象的数据库(Database)，查找地理数据集(GeoDataset)，最后为图层对象设置地理数据集，检查是否成功打开图层文件，如果成功则保存图层类型。

第二步：获得图层对象的记录集(Recordset)，通过记录集获得表描述(TableDesc)。然后通过记录集对象和表描述对象将地理数据集中的记录(Record)信息和字段(Field)信息顺次读出，并保存在内存中的数组。

相应的程序代码如下所示：

```
CMoMapLayer moMapLayer;
CString strDBName,strGDName;

AnalyseName(strFileName, strDBName,strGDName); //分析全路经名
moMapLayer.CreateDispatch ("MapObjects2.MapLayer"); //创建MapLayer对象

CMoDataConnection moDataConnection;
moDataConnection.CreateDispatch ("MapObjects2.DataConnection"); //创建DataConnection对象

moDataConnnection.SetDatabase(strDBName); //设置DataConnection对象的Database
CMoGeoDataset moGeoDataset;
moGeoDataset=moDataConnection.FindGeoDataset(strGDName); //查找空间数据集
if(moGeoDataset.m_lpDispatch)
{
moMapLayer.SetGeoDataset (moGeoDataset); //设置MapLayer对象的空间数据集
CMoRecordset moRecordset=moMapLayer.GetRecords(); //获得MapLayer对象的记录集
CTableDesc moTableDesc=moRecordset.GetTableDesc(); //获得Recordset对象的表描述
// 调用Recordset对象的GetCount、MoveFirst、MoveNext等函数浏览记录信息
// 调用TableDesc对象的GetFieldName、GetFieldType、GetFieldLength、
// GetFieldPrecision、GetFieldScale等函数浏览字段信息
// 将记录信息和字段信息保存到内存中，并保存图层文件类型
}
```

3.2 保存图层文件

这主要涉及到如何更新图层文件和浏览记录集。

第一步：首先创建图层对象、数据连接对象、表描述对象。然后设置表描述对象中的字段数量和各字段的详细信息(如名字、类型、长度、精度等)。然后对将要保存的图层文件的全路径名进行分析以获得文件名和路径名，为数据连接对象设置数据库和添加地理数据集，检查添加是否成功。如果成功则将返回的地理数据集关联到图层对象上。

第二步：获得图层对象的记录集，顺次将内存中的数据更新到记录集中，最后停止编辑。

相应的程序代码如下所示：

```
CMoMapLayer moMapLayer;
CString strDBName,strGDName;
CMoDataConnection moDataConnection;
CMoTableDesc moTableDesc;
CMoGeoDataset moGeoDataset;
CMoRecordset moRecordset;

AnalyseName(strFileName, strDBName,strGDName); //分析全路径名
moMapLayer.CreateDispatch ("MapObjects2.MapLayer"); //创建MapLayer对象
moDataConnection.CreateDispatch ("MapObjects2.DataConnection"); //创建DataConnection对象
moTableDesc.CreateDispatch ("MapObjects2.TableDesc"); //创建TableDesc对象
moDataConnection.SetDatabase (strDBName); //设置数据库
//调用FieldCount、SetFieldName、SetFieldType、SetFieldPrecision、SetFieldScale等函数设置字段信息
```

```
moGeoDataset=moDataConnection.AddGeoDataset(strGDName,m_nShapeType,,moTableDesc,COleVariant((short)TRUE,VT_I2),COleVariant((short)TRUE,VT_I2));
//向DataConnection对象添加空间记录集
if(moGeoDataset.m_lpDispatch)
{
moMapLayer.SetGeoDataset(moGeoDataset); //设置MapLayer对象的空间数据集
moRecordset=moMapLayer.GetRecords(); //获得MapLayer对象的记录集
//调用Recordset对象的AddNew、GetFields、Update等函数
//设置每一条记录并且更新到图层文件中
//调用StopEditing函数停止更新记录集
}
```

3.3 编辑元素形状

这主要涉及到元素拾取、视觉反馈等、编辑顶点。

（1）由于在图层文件中存在三种类型元素：点、线、面，它们的拾取方法各不相同，但拾取元素时都使用某个预先设定的阈值来判断元素被是否拾取。各种拾取方法都会涉及到由于图层缩放带来的阈值改变的问题，因此我们在打开图层文件时同时保留原始大小，在图层经过缩放时将阈值也按照缩放比例同时改变。

① 对于拾取点，我们仅需要判断某点到当前点的距离是否小于规定的阈值。

② 对于拾取线，由于线可能有多个点集组成，因此对于每个点集，我们顺次选择两点，然后计算着两点间的线段与当前点的距离是否小于规定的阈值。

③ 对于拾取面，我们仅需计算当前点与面的距离是否小于规定的阈值。

（2）视觉反馈包含两个方面的内容，即拾取反馈和修改反馈。拾取反馈是指当某个图层元素被选择后，将它的每个顶点用特定的颜色标注出来。更新反馈是指当按下鼠标按钮选择某个顶点后，被选择的顶点用特定的颜色标注出来并且顶点的位置随着鼠标的移动而不断改变，直到释放鼠标按钮。

① 在拾取反馈中，当图层类型为点时直接标注被拾取点，当图层类型为线时标注被拾取线的每个点集中的全部点，当被拾取的图层元素类型为面时标注被拾取面的每个点集中的全部点。

② 在修改反馈中，当图层类型为点时直接修改被拾取点，当图层类型为线时修改被拾取线的相应点集，当图层类型为面时修改被拾取面的相应点集。

（3）编辑顶点主要指在编辑线、面时可能需要向其某个点集中添加或删除顶点。在编辑顶点前需要首先拾取线、面，然后在相应的点集中添加或删除一个点，我们规定按下Inert键添加点、按下Delete键删除点。

3.4 加、删除图层元素

添加图层元素主要通过鼠标点击和拖动来实现。当图层类型为点时直接在点击处创建点对象，当图层类型为线时通过控件的TrackLine函数来生成线，当图层类型为面时通过控件的TrackPolygon函数来生成面。

删除图层元素主要通过在内存中的树组中删除相应的元素来实现。

3.5 鼠标事件处理过程

在MouseDown事件处理过程中主要完成拾取元素、添加元素、删除元素、编辑顶点、缩放图层等多项功能，其调用关系如图3所示：

在MouseMove事件处理过程中主要完成调整图层元素顶点位置的功能。在MouseUP事件处理过程中主要完成结束调整图层元素顶点位置的功能。

3.6 AfterTrackingLayerDraw事件处理过程

MapObjects为了性能上的优化，将所有图层元素的绘制都放在AfterTrackingLayerDraw事件中完成[1][2]。因此在上述所有操作完成后需要更新显示时就需要通过MapLayer对象的TrackingLayer属性获得TrackingLayer对象，然后调用TrackingLayer对象的Refresh函数触发AfterTrackingLayerDraw事件[1][2]。在AfterTrackingLayerDraw事件处理过程中主要完成绘制当前所有的图层元素、被拾取元素的边界顶点、被拾取的图层元素。注意，治理的绘制顺序不能改变，否则图层元素被拾取后视觉反馈就不正确了。

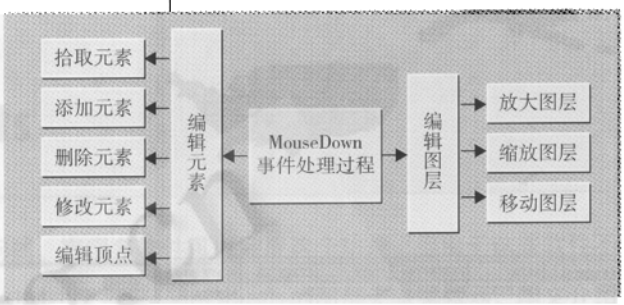

图 3　MouseDown 事件处理过程

4 小结

本文介绍了一种基于MapObjects的图层元素编辑器，系统已应用于水利部水资源所开发的防汛管理信息系统中。本文所述的这种MapObjects中图层元素编辑和更新的方法可以很容易的集成到任何基于MapObjects的GIS应用中。

参考文献

1 ESRI，MapObjects 2.0 Online Refrence，www.esri.com，2003。

2 Robert Hartman，Focus on GIS Component Software Featuring ESRI's MapObjects，ONWORD PRESS，1997。

3 Scot Wingo，Visual C++6.0技术内幕，北京希望电子出版社，1999。