实验四　图层控制及渲染

4.1 背景知识

4.1.1 图层对象

ArcObjects 中的图层 Layer 本身并没有装载数据，而仅仅是获得了数据源的连接用于管理数据和地图显示而已。在 ArcObjects 中地理数据始终是保存在地理数据库 Geodatabase 或者地理文件中，图层 Layer 的种类如图 4-1 所示，其中：（1）要素图层 FeatureLayer 在地图显示地理数据库 Geodatabase 中的要素类，是承载要素 Feature（表示离散的矢量对象）的图层，FeatureLayer 包含的超链接 Hyperlink 用于管理如文本文档或网页等链接数据；（2）栅格图层 RasterLayer 用于从图像服务中检索数据并显示图像，栅格目录图层 RasterCatalogLayer 根据输入栅格目录创建临时图层；（3）不规则三角网图层 TinLayer 显示 TIN 三维表面数据；（4）CAD 图层 CadLayer 显示一幅 CAD 图，CAD 要素图层 CadFeatureLayer 显示一幅图中的 CAD 要素类，CAD 注记图层 CadAnnotationLayer 用来控制 CAD 图层中的注记；（5）图形图层 GraphicsLayer 管理一幅地图上的图形，它派生了两个子类，复合图形图层 CompositeGraphicsLayer 管理图形图层的集合，注记图形图层 FDOGraphicsLayer 管理地理数据库 Geodatabase 中的注记要素（annotation features）；（6）coverage 注记图层 CoverageAnnotationLayer 显示一幅 coverage 图中的注记；（7）维度图层 DimensionLayer 管理 Geodatabase 中的尺寸标注要素图层；（8）组合图层 GroupLayer 是图层的组合，它能够显示和操作图层组合的内容列表；（9）网络地图服务图层 IMSMapLayer 显示因特网上的一个地图服务数据。

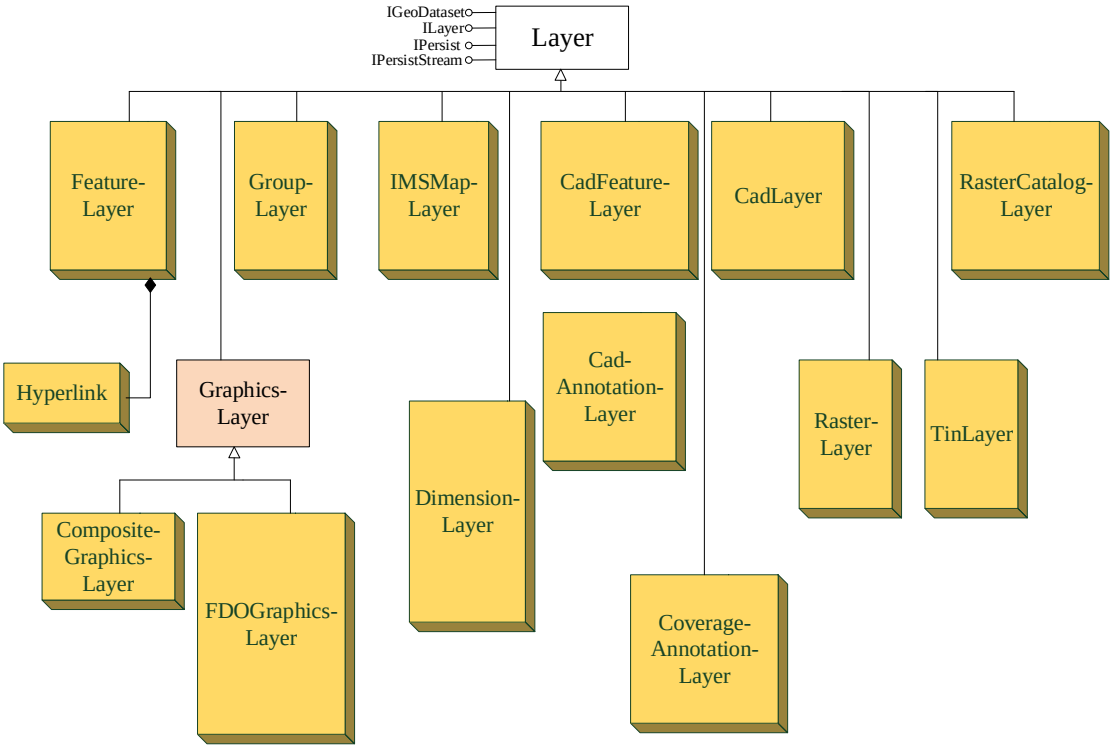

图 4-1 图层 Layer 的对象种类

4.1.2 要素图层渲染

ArcObjects 提供了让制图者根据需要对地理数据的显示方式进行设置的功能。要素渲染器 FeatureRenderer 用于绘制要素，包含有图例组合 LegendGroup 用来管理图例类 LegendClass 的集合，图例类 LegendClass 包含一个符号和一些用来描述符号代表什么类型的标注和文字。要素渲染器 FeatureRenderer 的常用绘制方法如图 4-2 所示，包括：（1）简单渲染器 SimpleRenderer 用同一个符号绘制所有要素；（2）唯一值渲染器 UniqueValueRenderer 根据要素某一属性值来确定绘制该要素的符号；（3）分级渲染器 ClassBreaksRenderer 可以用分级的颜色和符号来绘制要素；（4）双向唯一值渲染器 BiUniqueValueRenderer 结合了唯一值渲染器和分级渲染器；（5）比率符号渲染器 ProportionalSymbolRenderer 用不同大小的符号绘制要素，其大小对应要素某一字段值的比率；（6）点密度渲染器 DotDensityRenderer 在多边形要素内绘制不同密度的点；（7）图表渲染器 ChartRenderer 基于每个要素的属性绘制统计图（饼图、直方图和累计直方图）；（8）依比例渲染器 ScaleDependentRenderer 由多个渲染器组成，每个渲染工作限定在一定的比例尺范围内。

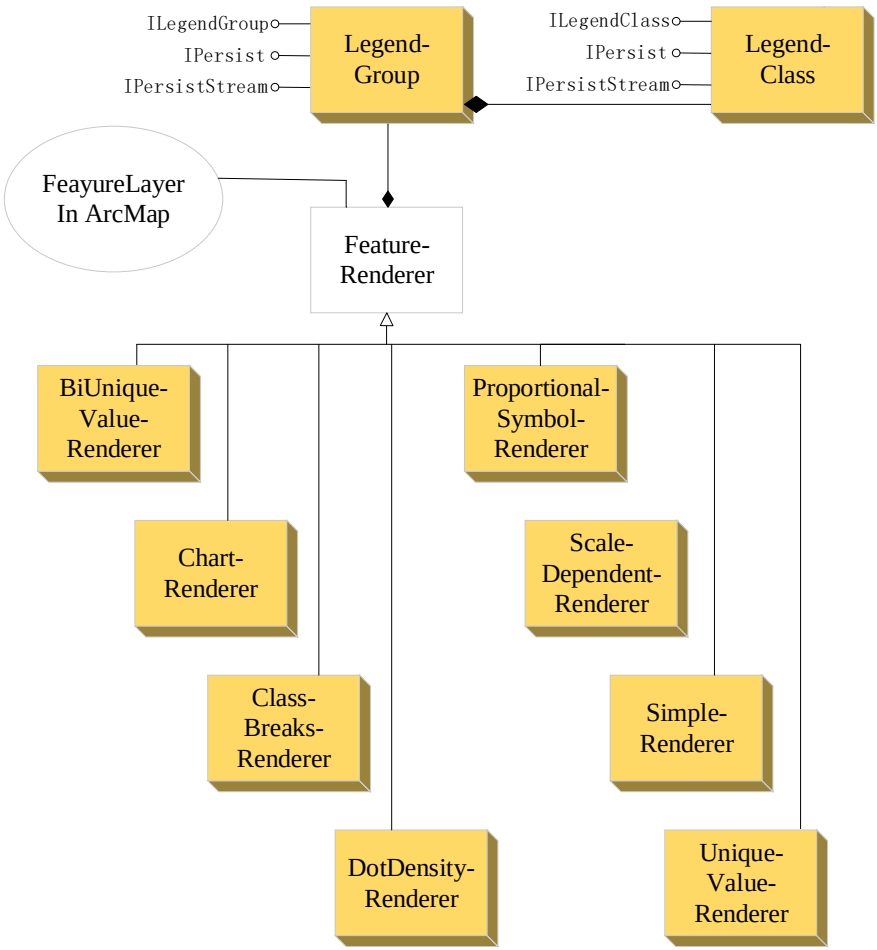

图 4-2　要素渲染器对象模型图

4.2 实验目的

（1）熟练组合使用 MapControl/TOCControl/ToolbarControl 控件；

（2）了解 MapControl 中图层的控制，掌握使用程序代码控制图层可视性、移动和移除等操作；

（3）熟悉矢量要素图层的符号化表达和渲染方式，掌握专题制图的常用方法。

4.3 实验内容

（1）图层控制；

（2）矢量要素图层的渲染。

4.4 实验数据

见安装目录：

…\DeveloperKit10.6\Samples\ArcObjectsSampleData\data\World

4.5 实验步骤

4.5.1 图层的控制

（1）新建“MapControl Application”项目，添加一个右键菜单 ContextMenuStrip 控件，在该控件上分别添加“向上一层”、“向下一层”、“删除图层”菜单项，如图 4-3 所示。

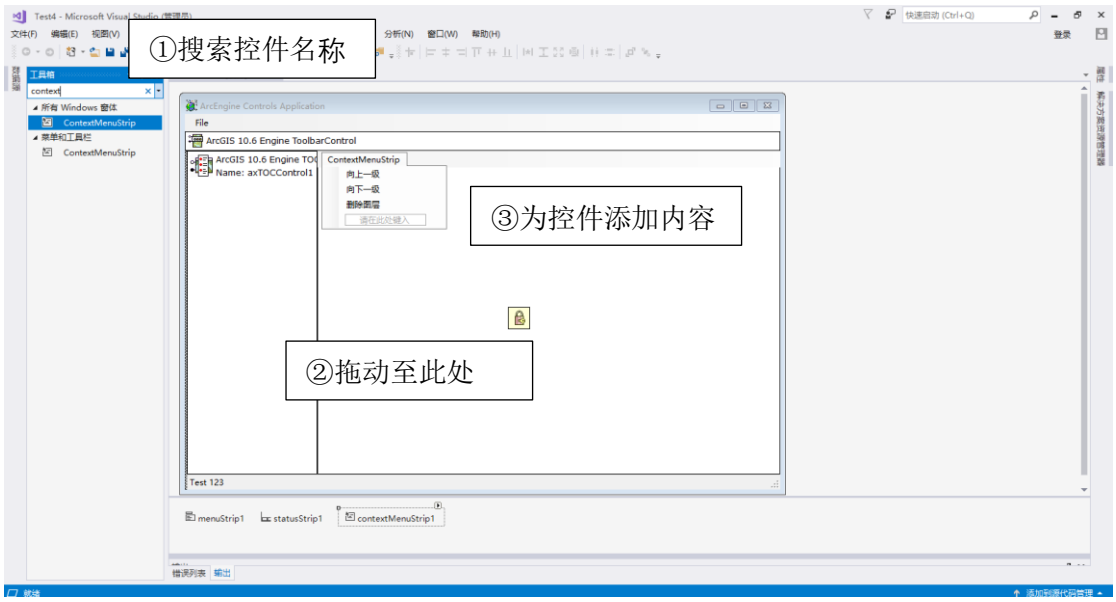

图 4-3　添加右键菜单 ContextMenuStrip 控件

该“ContextMenuStrip1”控件要作为右键菜单在 TOCControl 控件中弹出。TOCControl 要与一个“伙伴控件”协同工作，“伙伴控件”可以是实现了 ITOCBuddy 接口的 MapControl、PageLayoutControl、ReaderControl、SceneControl、GlobeControl 控件，本例中“axTOCControl1”的默认“Buddy”控件为“axMapControl1”。TOCControl 控件的主要方法有：（a）获得选择项 GetSelectedItem()方法，方法只能在 MouseUp 事件中来使用；(b)碰撞检测 HitTest()方法，可以在 MouseDown 和 MouseUp 事件中使用。

（2）为“MainForm”主窗体类的添加私有成员变量 m_layerSelected，用于存储当前交互选中的图层；添加私有成员变量和 m_tocControl，用于获取“axTOCControl1”控件对象，代码如下

```
private ILayer m_layerSelected = null;
private IMapControl3 m_mapControl = null;
private ITOCControl m_tocControl = null;
```

修改主窗体的 Form_Load 事件，代码如下：

MainForm.cs（节选）	功能：获取 IMapControl3 和 ITOCControl 接口

```
private void MainForm_Load(object sender, EventArgs e)
{
    //get the MapControl
    m_mapControl = (IMapControl3)axMapControl1.Object;
    m_tocControl = (ITOCControl)axTOCControl1.Object;
    //disable the Save menu (since there is no document yet)
    menuSaveDoc.Enabled = false;
}
```

（3）通过属性添加 axTOCControl1 的 OnMouseUp 事件响应函数，如图 4-4 所示。

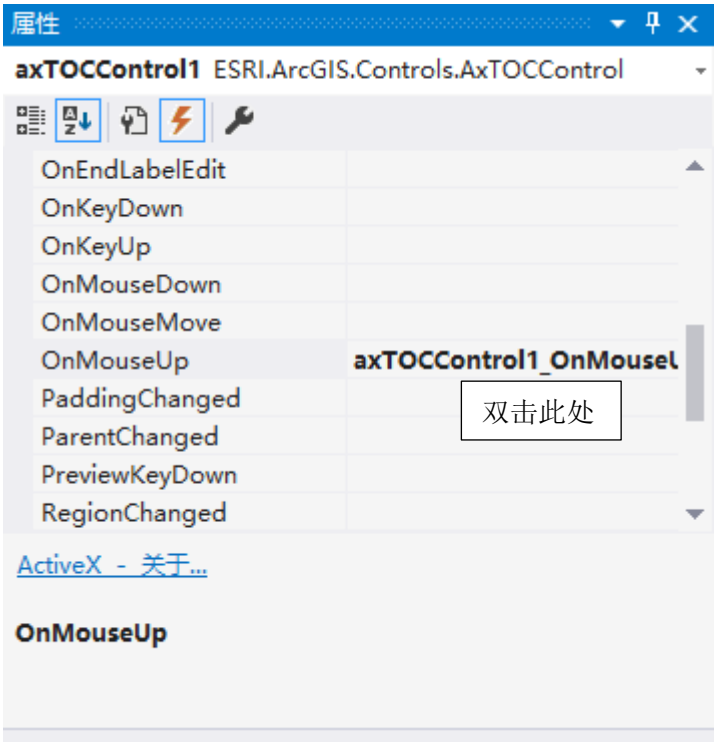

图 4-4　添加 axTOCControl1 控件的 OnMouseUp 事件响应函数

参考代码如下：

MainForm.cs（节选）	功能：获得选择项和弹出右键菜单

```
private void axTOCControl1_OnMouseUp(object sender,
                                     ITOCControlEvents_OnMouseUpEvent e)
{
    //获得axTOCControl1控件的当前选择项
```

```
        esriTOCControlItem type = esriTOCControlItem.esriTOCControlItemNone;
        IBasicMap basicMap = null;
        ILayer layer = null;
        object unk = null, data = null;
        axTOCControl1.GetSelectedItem(ref type, ref basicMap, ref layer, ref unk,
                                                                    ref data);
        //如当前选择项类型为图层对象，鼠标右键
        if (type == esriTOCControlItem.esriTOCControlItemLayer
            && layer != null
            && e.button == 2)
        {
            //存储当前选择图层
            m_layerSelected = layer;
            //弹出右键菜单
            contextMenuStrip1.Show(axTOCControl1, e.x, e.y);
        }
}
```

（4）双击“ContextMenuStrip1”控件右键菜单中各个菜单项，如图 4-5 所示，添加“ContextMenuStrip1”控件右键菜单中各个菜单项的 OnClick 事件响应函数。

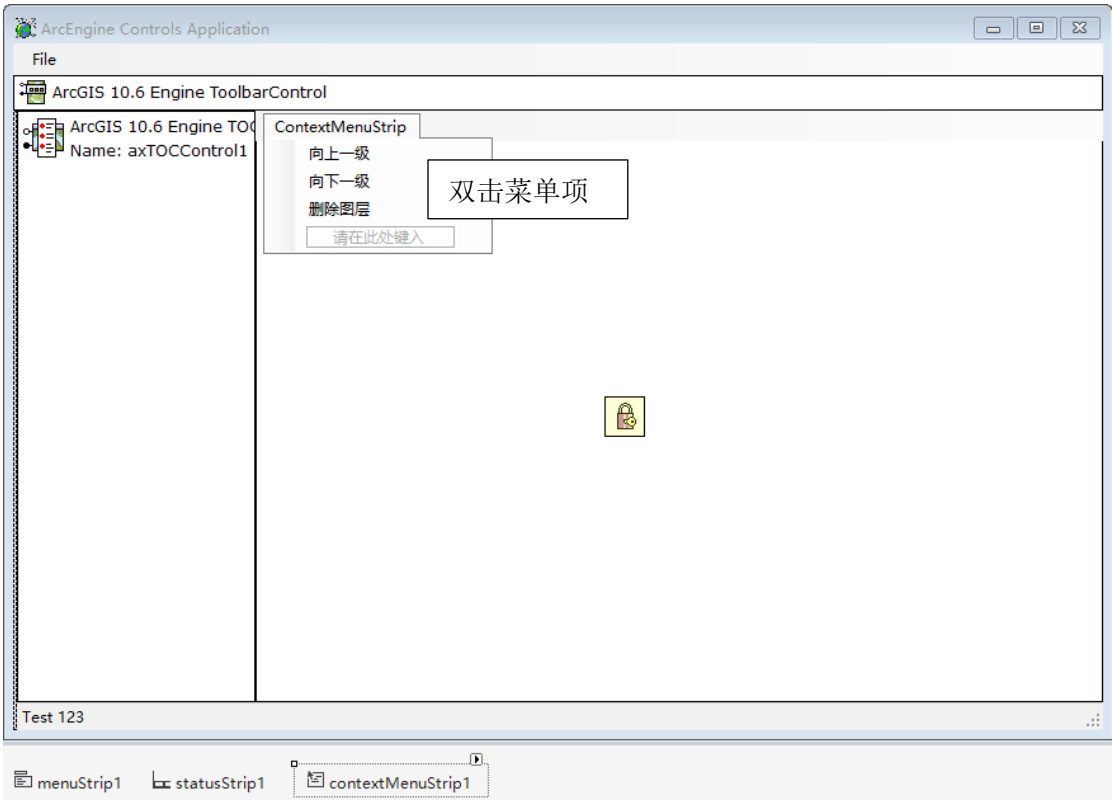

图 4-5　双击菜单项添加 OnClick 事件响应函数

参考代码如下：

MainForm.cs（节选）	功能：图层控制各菜单项 OnClick 事件响应函数

```
private void  向上一级ToolStripMenuItem_Click(object sender, EventArgs e)
{
    //获得axMapControl1中的图层个数
    int lCnt = axMapControl1.LayerCount;
    //循环axMapControl1中所有图层
    for (int i = 0; i < lCnt; i++)
    {
        //得到当前选择图层，且该层不是最上层
        if (axMapControl1.get_Layer(i).Name == m_layerSelected.Name
                                                    && i - 1 >= 0)
        {
            //将当前选择图层向上移动一层
            axMapControl1.MoveLayerTo(i, i - 1);
            break;
        }
    }
}
private void  向下一级ToolStripMenuItem_Click(object sender, EventArgs e)
{
    int lCnt = axMapControl1.LayerCount;
    for (int i = 0; i < lCnt; i++)
    {
        //得到当前选择图层，且该层不是最底层
        if (axMapControl1.get_Layer(i).Name == m_layerSelected.Name
                                                    && i + 1 < lCnt)
        {
            //将当前选择图层向下移动一层
            axMapControl1.MoveLayerTo(i, i + 1);
            break;
        }
    }
}
private void  删除图层ToolStripMenuItem_Click(object sender, EventArgs e)
{
    int lCnt = axMapControl1.LayerCount;
    for (int i = 0; i < lCnt; i++)
    {
        //得到当前选择图层
        if (axMapControl1.get_Layer(i).Name == m_layerSelected.Name)
        {
            //将当前选择图层从axMapControl1中删除
            axMapControl1.DeleteLayer(i);
            break;
```

```
            }
        }
}
```

（5）编译并运行程序，功能实现效果如图 4-6 所示。

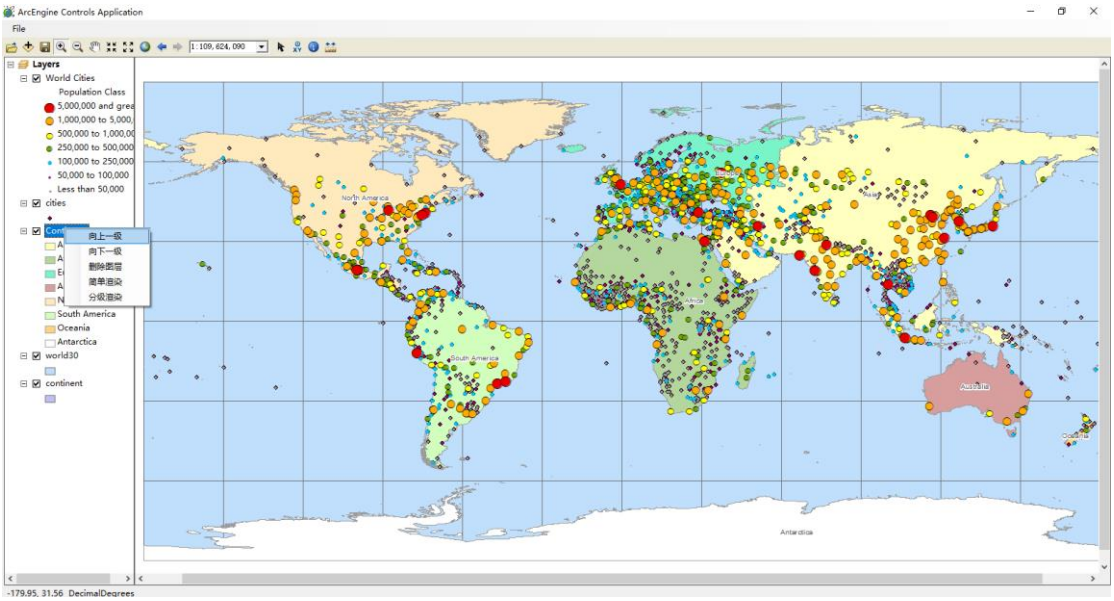

图 4-6　图层控制程序运行效果

4.5.2 图层的渲染

在窗体代码 MainForm.cs 导入引用：

```
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Controls;
using  ESRI.ArcGIS.SystemUI;
using  ESRI.ArcGIS.Display;
```

在该“contextMenuStrip1”控件上分别添加“简单渲染”和“分级渲染”，如图 4-7 所示。

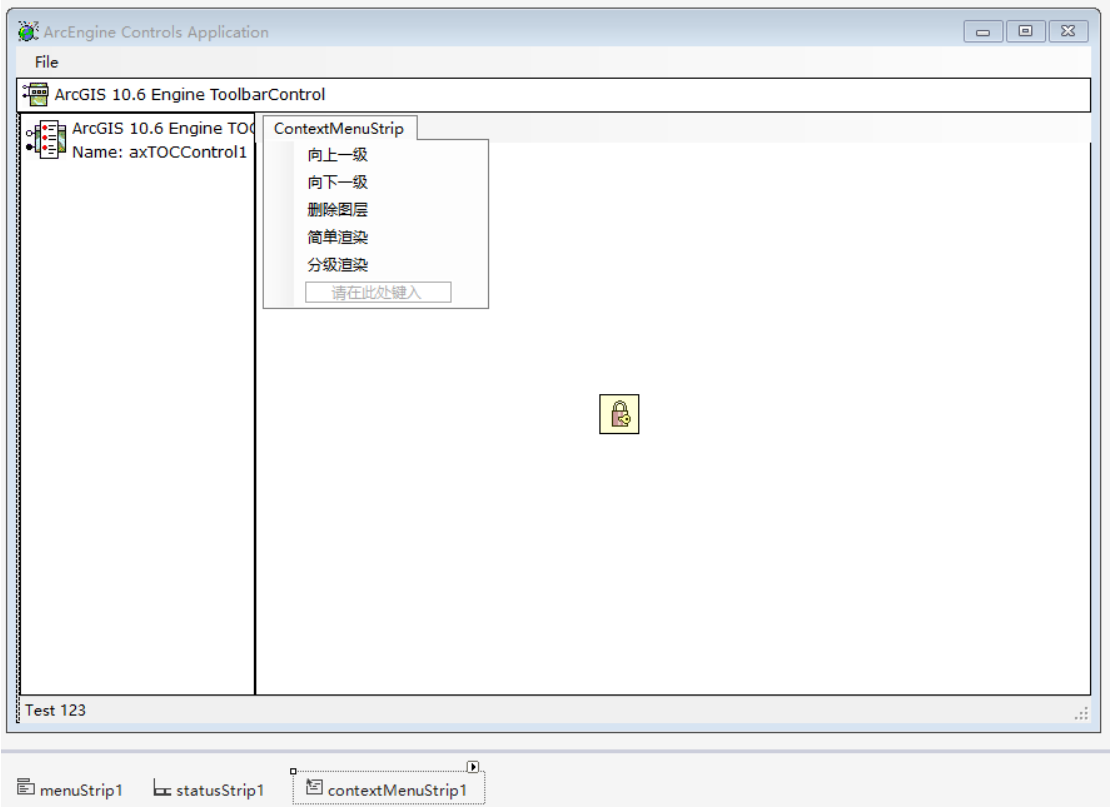

图 4-7　添加右键菜单“图层渲染”菜单项

1）简单渲染

（1）双击“contextMenuStrip1”控件菜单的“简单渲染”选项，为“简单渲染”选项的 Click 事件添加代码，代码如下：

MainForm.cs（节选）	功能：简单渲染 Click 事件响应函数

```
private void 简单渲染ToolStripMenuItem_Click(object sender, EventArgs e)
{
    IGeoFeatureLayer pGeoFeatureLayer = m_layerSelected as
                                                    IGeoFeatureLayer;
    //设置颜色属性
    IRgbColor pRgbColor = new RgbColorClass();
    pRgbColor.Red = 0;
    pRgbColor.Green = 200;
    pRgbColor.Blue = 100;
    //定义填充符号
    ISymbol sym = null;
    switch (pGeoFeatureLayer.FeatureClass.ShapeType)
    {
        case esriGeometryType.esriGeometryPoint:
            sym = new SimpleMarkerSymbolClass();
            //定义点状样式和颜色
            ISimpleMarkerSymbol pMarkerSymbol = sym as
```

```
                                                    ISimpleMarkerSymbol;
                pMarkerSymbol.Style = esriSimpleMarkerStyle.esriSMSSquare;
                pMarkerSymbol.Color = pRgbColor;
                break;
            case esriGeometryType.esriGeometryPolyline:
                sym = new SimpleLineSymbolClass();
                //定义线状样式和颜色
                ISimpleLineSymbol pLineSymbol = sym as ISimpleLineSymbol;
                pLineSymbol.Style = esriSimpleLineStyle.esriSLSDot;
                pLineSymbol.Color = pRgbColor;
                break;
            case esriGeometryType.esriGeometryPolygon:
                sym = new SimpleFillSymbolClass();
                //定义面状颜色
                ISimpleFillSymbol pFillSymbol = sym as ISimpleFillSymbol;
                pFillSymbol.Color = pRgbColor;
                break;
            default:
                return;
        }
        //初始化ISimpleRenderer对象
        ISimpleRenderer pSimpleRenderer;
        pSimpleRenderer = new SimpleRendererClass();
        pSimpleRenderer.Symbol = sym;
        //使图例中显示的字符串为所选字段
        pSimpleRenderer.Label = "SimpleSymbol";
        pSimpleRenderer.Description = "Description";
        //将渲染器赋给地理图层
        pGeoFeatureLayer.Renderer = pSimpleRenderer as IFeatureRenderer;
        axMapControl1.ActiveView.Refresh();
        axTOCControl1.Update();
}
```

（2）编译并运行程序，功能实现效果如图 4-8 所示。

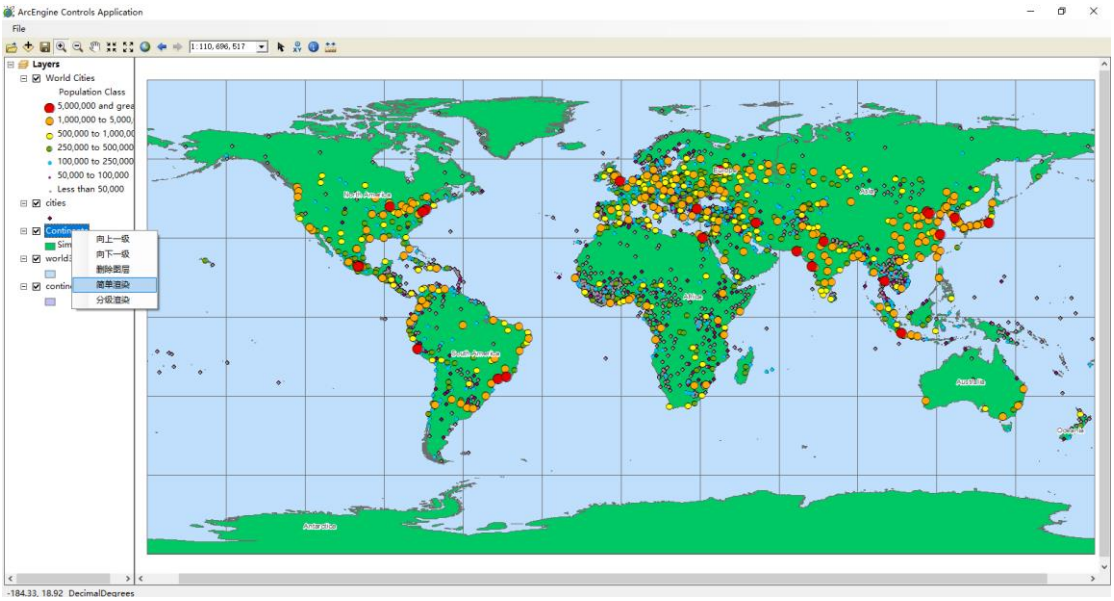

图 4-8 图层简单渲染运行效果

2）分级渲染

（1）双击“contextMenuStrip1”控件菜单的“分级渲染”选项，为“分级渲染”选项的 Click 事件添加代码，代码如下：

MainForm.cs（节选）	功能：分级渲染 Click 事件，弹出“分级符号化”窗体

```
private void 分级渲染ToolStripMenuItem_Click(object sender, EventArgs e)
{
    FrmClassSymbol frmClassSymbol = new FrmClassSymbol(
                    m_mapControl.Object, m_tocControl, m_layerSelected);
    frmClassSymbol.Show();
}
```

（2）在解决方案资源管理器中选择项目“Test4”，右键单击【添加】→【新建项】，如图 4-9 所示。

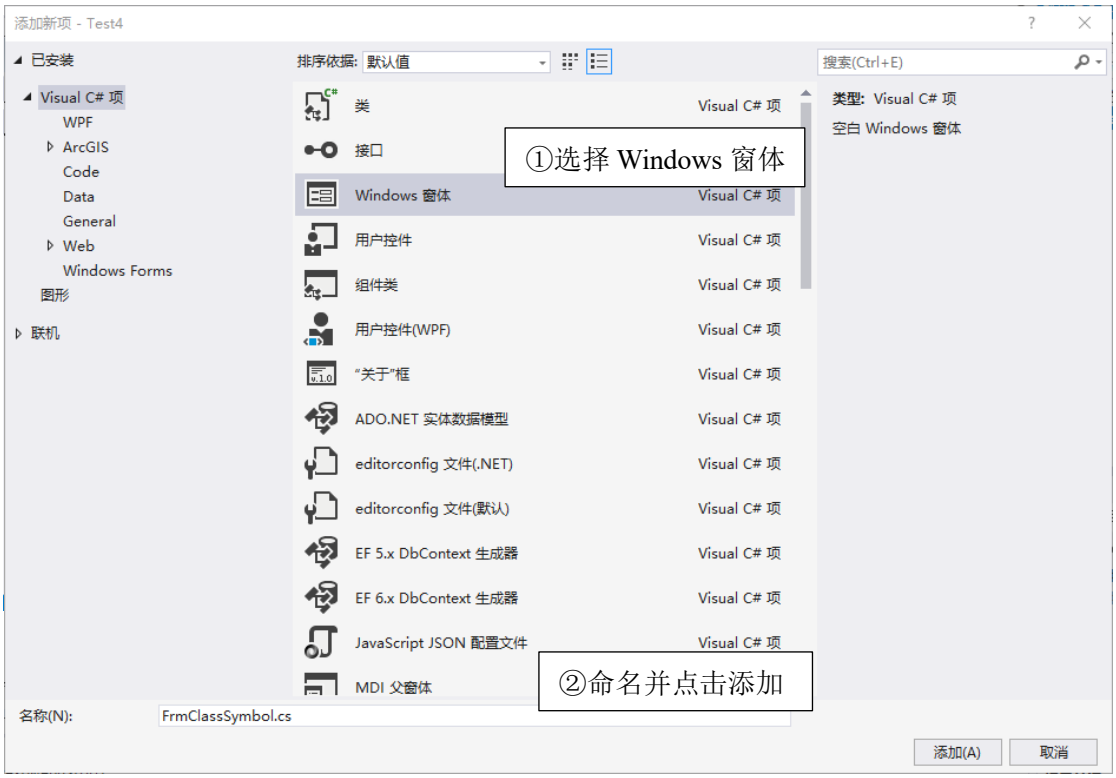

图 4-9 添加新建项

添加 Window 窗体，类命名为“FrmClassSymbol”，如图 4-10 所示。

图 4-10 添加新项“FrmClassSymbol”窗体类

通过属性窗口将窗体“FrmClassSymbol”的 Text 属性改为“分级符号化”，如图 4-11 所示。

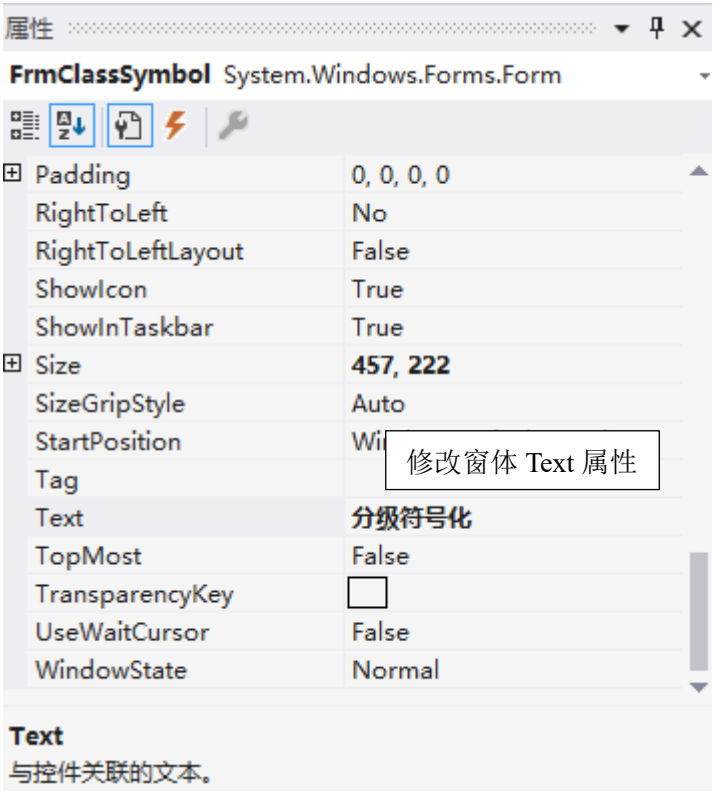

图 4-11　修改窗体 Text 属性

为窗体“FrmClassSymbol”添加控件（界面设计如图 4-12 所示）：（1）ComboBox 控件 cbxFields，用来选择所选中图层的属性字段；（2）NumericUpDown 控件 nudClassCount，用来调整分级数；（3）TextBox 控件 txtMinValue 和 txtMaxValue，用来显示所选中字段的最小值和最大值；（4）Button 控件 btnSymbolize 和 btnClose，用来执行符号化和关闭窗体。

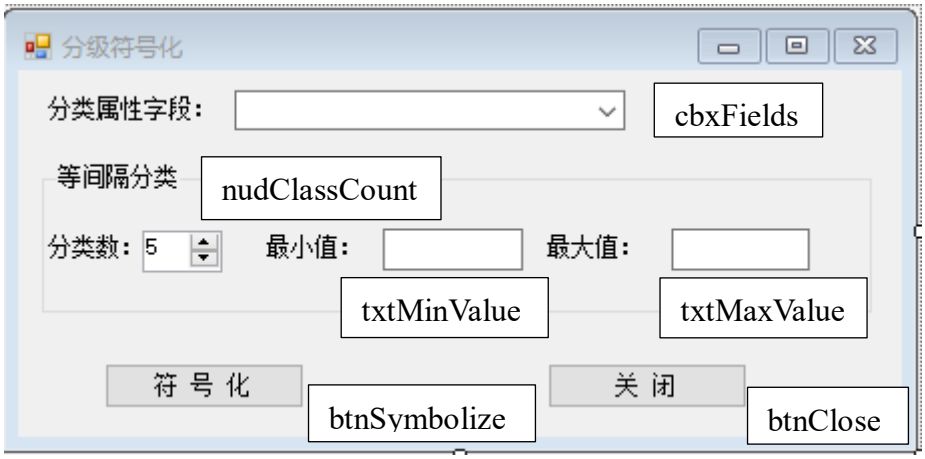

图 4-12 分级符号化窗体“FrmClassSymbol”界面设计图

（3）打开“分级符号化”窗体代码（FrmClassSymbol.cs），导入引用：

```
using ESRI.ArcGIS.Controls;
```

using ESRI.ArcGIS.Carto;

using ESRI.ArcGIS.Display;

using ESRI.ArcGIS.esriSystem;

using ESRI.ArcGIS.Geodatabase;

声明窗体类“FrmClassSymbol”的私有成员变量：

private IHookHelper m_hookHelper = null;

private ITOCControl m_tocControl = null;

private IFeatureLayer m_layer = null;

private string m_strRendererField = string.Empty;

private int m_classCount = 5; //默认初始分级数为 5

修改窗体类“FrmClassSymbol”构造函数，代码如下：

FrmClassSymbol.cs（节选）	功能：窗体类构造函数

```
public FrmClassSymbol(object hook, ITOCControl toc,    IFeatureLayer layer)
{
    //定义带有地图对象参数、ITOCControl参数和图层参数的构造函数
    InitializeComponent();
    m_layer = layer;
    if (m_hookHelper == null)
        m_hookHelper = new HookHelperClass();
    m_hookHelper.Hook = hook;
    m_tocControl = toc;
}
```

（4）为了使“分级符号化”窗体打开时将所选择图层中符合分级渲染要求的属性字段添加进 ComboBox 控件 cbxFields 中，通过属性窗口添加“FrmClassSymbol”窗体加载事件响应函数，如图 4-13 所示。

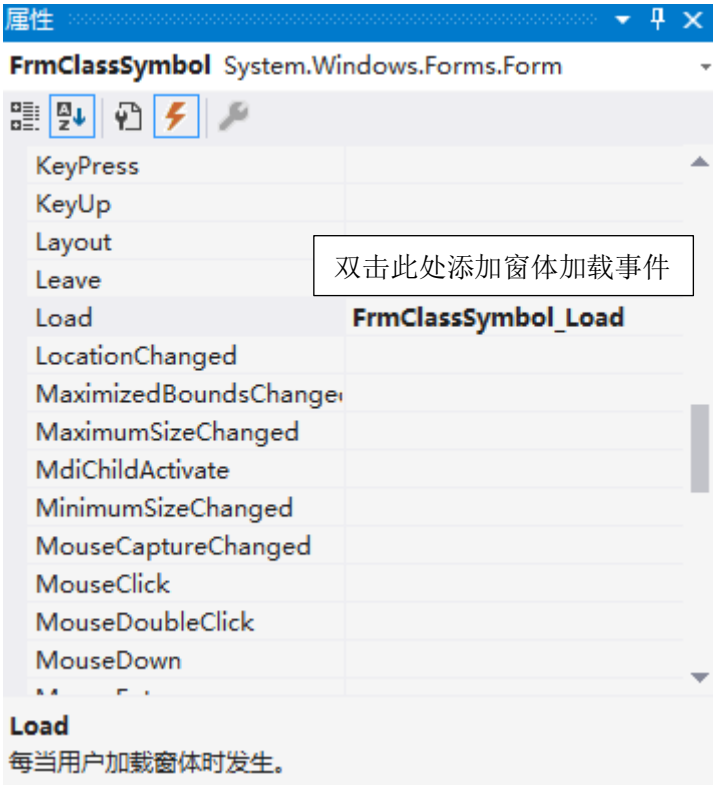

图 4-13　通过属性窗口添加窗体加载事件响应函数

参考代码如下：

FrmClassSymbol.cs（节选）	功能：窗体 Load 事件及所调用的加载字段函数

```
private void FrmClassSymbol_Load(object sender, EventArgs e)
{
    //该窗体加载完成时就已经将选中图层的属性字段加载进ComboBox
    CbxFieldsAdditems(m_layer as IFeatureLayer);
}
private void CbxFieldsAdditems(IFeatureLayer featureLayer)
{
    IFields fields = featureLayer.FeatureClass.Fields;
    //将属性为以下几个类型的字段添加进ComboBox控件
    for (int i = 0; i < fields.FieldCount; i++)
    {
        if ((fields.get_Field(i).Type == esriFieldType.esriFieldTypeDouble) ||
           (fields.get_Field(i).Type == esriFieldType.esriFieldTypeInteger) ||
           (fields.get_Field(i).Type == esriFieldType.esriFieldTypeSingle) ||
           (fields.get_Field(i).Type == esriFieldType.esriFieldTypeSmallInteger))
        {
            cbxFields.Items.Add(fields.get_Field(i).Name);
        }
    }
    cbxFields.SelectedIndex = 0;
```

```
}
```

（5）通过属性窗口为 cbxFields 控件添加 SelectedIndexChanged 事件，代码如下：

FrmClassSymbol.cs（节选） | 功能：选中属性字段及属性字段改变事件

```
private void cbxFields_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cbxFields.SelectedItem != null)
    {
        m_strRendererField = cbxFields.SelectedItem.ToString();
        //在要素图层中找到选中的字段
        IFeatureClass featureClass = m_layer.FeatureClass;
        IField field = featureClass.Fields.get_Field(featureClass.FindField
                                                    (m_strRendererField));
        //创建一个游标
        ICursor cursor = (ICursor)m_layer.Search(null, false);
        //创建一个数据统计对象并初始化其属性
        IDataStatistics dataStatistics = new DataStatisticsClass();
        dataStatistics.Field = field.Name;
        dataStatistics.Cursor = cursor;
        //得到统计结果
        IStatisticsResults statisticsResults = dataStatistics.Statistics;
        //计算窗体上显示的该属性字段的最大值和最小值
        txtMinValue.Text = statisticsResults.Minimum.ToString();
        txtMaxValue.Text = statisticsResults.Maximum.ToString();
    }
}
```

（6）添加 btnSymbolize 和 btnClose 两个 Button 控件的 Click 事件响应函数及调用的函数，代码如下：

FrmClassSymbol.cs（节选） | 功能：执行符号化和关闭窗体事件

```
private void btnSymbolize_Click(object sender, EventArgs e)
{
    //获得分级数
    m_classCount = Convert.ToInt32(nudClassCount.Value);
    if (m_strRendererField == null)
        return;
    else
        Render(m_strRendererField,m_classCount);
}
//分级渲染
private void Render(string RenderField, int classCount)
{
```

```
double[] classes;
IGeoFeatureLayer pGeoFeatureLayer = m_layer as IGeoFeatureLayer;
ITable pTable = pGeoFeatureLayer.FeatureClass as ITable;
//Itable接口位于Geodatabase类库中
ITableHistogram pTableHistogram = new BasicTableHistogramClass();
IBasicHistogram pBasicHistogram = pTableHistogram as IBasicHistogram;
pTableHistogram.Field = RenderField;
pTableHistogram.Table = pTable;
object dataValues;
object dataFrequent;
pBasicHistogram.GetHistogram(out dataValues, out dataFrequent);
//获取FeatureClass中的dataValues和datafrequent
IClassifyGEN pClassifyGEN = new EqualIntervalClass();
//根据上面的dataValues和datafrequent进行分级，分级数为classcout
pClassifyGEN.Classify(dataValues, dataFrequent, ref classCount);
//获取分段点
classes = (double[])pClassifyGEN.ClassBreaks;
IClassBreaksRenderer pClassBreakRenderer = new ClassBreaksRenderer();
//断点数
pClassBreakRenderer.BreakCount = classCount;
pClassBreakRenderer.Field = RenderField;
//按顺序排列
pClassBreakRenderer.SortClassesAscending = true;
IAlgorithmicColorRamp pColorRamp = new AlgorithmicColorRampClass();
//设置颜色
IRgbColor pRgbColor1 = new RgbColorClass();
IRgbColor pRgbColor2 = new RgbColorClass();
pRgbColor1.Red = 178;
pRgbColor1.Green = 34;
pRgbColor1.Blue = 34;
pRgbColor2.Red = 255;
pRgbColor2.Green = 193;
pRgbColor2.Blue = 193;
pColorRamp.FromColor = pRgbColor2;//起始
pColorRamp.ToColor = pRgbColor1;//终止
pColorRamp.Size = classCount;//颜色带数目
bool ok = true;
pColorRamp.CreateRamp(out ok);
IEnumColors pEnumColors = pColorRamp.Colors;
for (int i = 0; i < classCount; i++)
{
    IColor pColor = pEnumColors.Next();
    ISymbol pSymbol = GetSymbol(pColor);
    pClassBreakRenderer.set_Symbol(i, pSymbol);
```

```
            pClassBreakRenderer.set_Label(i, classes[i].ToString() + "-" +
                                                        classes[i + 1].ToString());
            pClassBreakRenderer.set_Break(i, classes[i + 1]);
        }
        pGeoFeatureLayer.Renderer = pClassBreakRenderer as IFeatureRenderer;
        //更新地图
        m_hookHelper.ActiveView.PartialRefresh(
                                esriViewDrawPhase.esriViewGeography, null, null);
        m_tocControl.Update();
}
//获得点、线、面符号
private ISymbol GetSymbol(IColor pColor)
{
        ISymbol sym=null;
        switch (m_layer.FeatureClass.ShapeType)
        {
            case esriGeometryType.esriGeometryPoint:
                sym = new SimpleMarkerSymbolClass();
                //定义点状样式和颜色
                ISimpleMarkerSymbol pMarkerSymbol = sym as
                                                        ISimpleMarkerSymbol;
                pMarkerSymbol.Style = esriSimpleMarkerStyle.esriSMSSquare;
                pMarkerSymbol.Color = pColor;
                break;
            case esriGeometryType.esriGeometryPolyline:
                sym = new SimpleLineSymbolClass();
                //定义线状样式和颜色
                ISimpleLineSymbol pLineSymbol = sym as ISimpleLineSymbol;
                pLineSymbol.Style = esriSimpleLineStyle.esriSLSDot;
                pLineSymbol.Color = pColor;
                break;
            case esriGeometryType.esriGeometryPolygon:
                sym = new SimpleFillSymbolClass();
                //定义面状颜色
                ISimpleFillSymbol pFillSymbol = sym as ISimpleFillSymbol;
                pFillSymbol.Color = pColor;
                break;
            default:
                return null;
        }
        return sym;
}
private void btnClose_Click(object sender, EventArgs e)
{
```

```
        this.Close();
}
```

（7）编译并运行程序，分级符号化参数设置窗体运行效果如图 4-14 所示。

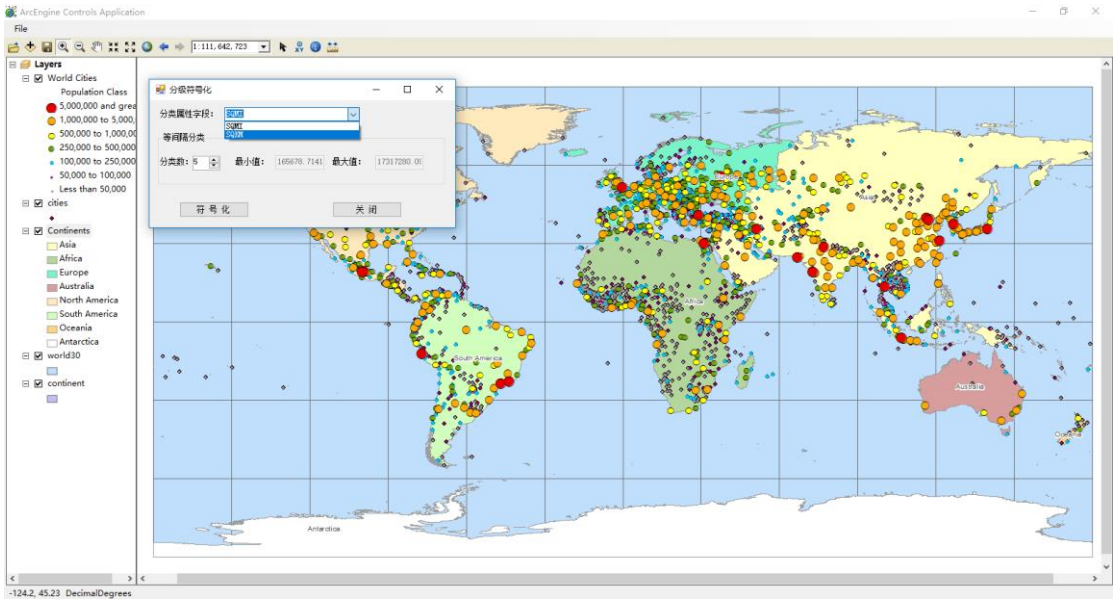

图 4-14　设置分级符号化参数

图层分级渲染功能实现效果如图 4-15 所示。

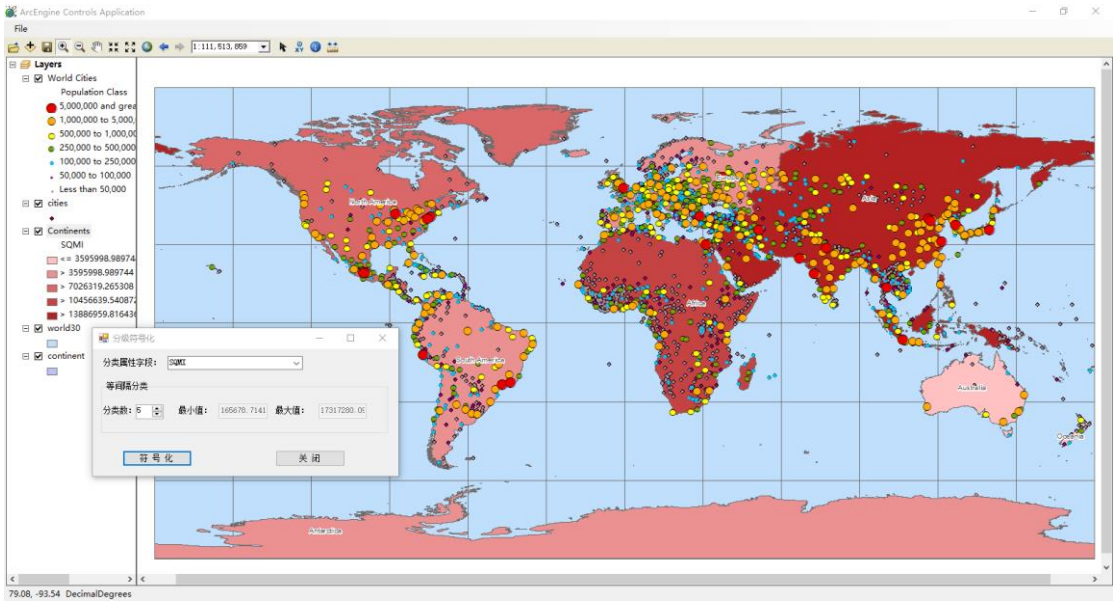

图 4-15　图层分级渲染程序运行效果