实验七 矢量空间分析

7.1 背景知识

矢量数据是大多数 GIS 系统的主要数据源，对矢量数据的各种空间分析是 GIS 系统的主要功能之一。矢量数据位置明显、属性隐含的特点便于高精度地刻画边界明确的地理特征，常用于高精度的地图制图和智能化地理特征的行为控制，传统测绘项目的成果数据也几乎都是以矢量数据形式表达的。矢量空间分析根据使用的数据性质不同，可以分为：基于空间图形数据的分析运算、基于非空间属性的数据运算、空间和非空间数据的联合运算。

7. 1. 1 属性查询与空间查询

矢量数据属性查询与控件查询主要涉及到查询、光标和选择集等对象，如图 7-1 所示。

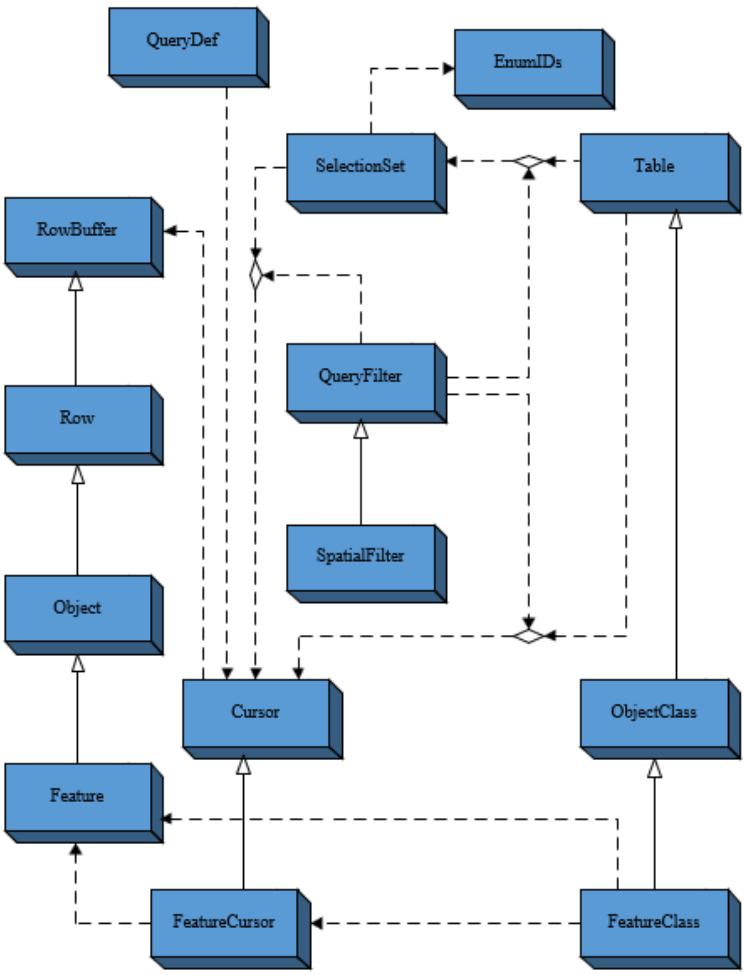

图 7-1 查询、光标与选择集对象

（1）查询定义 QueryDef 对象代表了对一个或多个表或要素类的数据库查询。查询定义可以被求值，引发在数据库服务器上的查询的执行。查询的结果以光标 Cursor 的形式返回给应用程序，由应用程序来通过光标获取查询结果集中的行对象。

（2）选择集 SelectionSet 对象允许应用程序指向一个选中对象的集合，这些对象应当是同属一个表或特征类的行。选择集在 ArcObjects 中通常用在某些需要产生临时的行或要素的子集的操作。注意选中对象必须是同属一张表的，不可以把来自不同表的选择集合并起来。

（3）光标 Cursor 是一个数据访问对象，不仅可以用来依次完整地复述一个表或查询的集合，也可以用来向表中插入新行。ArcObjects 中有三种光标，分别叫做搜索 search 光标、插入 insert 光标和更新 update 光标，各自有表或要素类的对应方法返回，其中搜索和更新方法需要查询定义 QueryDef 作为输入，以限定返回的集合。

（4）过滤查询对象 QueryFilter 通过设置 WhereClause 语句来扩展查询。

（5）空间查询过滤对象 SpatialFilter 同时包含空间约束和属性约束的查询过滤，必须设置 Geometry（几何）、GeometryField（几何字段）和 SpatialRel（空间关系），其中空间关系 SpatialRel 类型如表 7-1 所示。空间查询过滤对象 SpatialFilter 的应用包括：（a）选出与搜索范围相交的地理特征；（b）找到靠近某一特征的地理特征；（c）为特征显示定义一个有限的区域。

表 7-1 空间关系类型

空间关系	描述（A 为查询几何对象，B 为目标几何对象）
esriSpatialRelUndefined	关系为定义
esriSpatialRelIntersects	A 与 B 相交
esriSpatialRelEnvelopeIntersects	A 的包络线和 B 的包络线相交
esriSpatialRelTouches	A 与 B 相接，即其边界处相接
esriSpatialRelOverlaps	A 与 B 相叠加，它们必须是同维对象，如都是多边形等
esriSpatialRelCrosses	A 与 B 相交，如两条线交于一点，面和线交于一条线，面与面无此关系
esriSpatialRelWithin	A 在 B 的内部
esriSpatialRelContains	A 被 B 包含

7. 1. 2 空间拓扑与空间关系运算

拓扑（topology）是几何形状的空间关联。几何 Geometry 对象实现的拓扑算子 ITopologicalOperator 接口，包含有若干空间拓扑运算方法：（1）缓冲（buffer）设置缓冲距离形成多边形；（2）裁剪（clip）通过包络面裁剪几何对象；（3）凸包（convex hull）包含几何图形的最小多边形；（3）切割（cut）沿切割曲线的方向把几何图形分为左右两半；（3）差集（difference）基本几何图形前去对照几何图形；（4）交集（intersect）既在基本几何图形中，又在对照几何图形中；（5）异集（symmetric difference）可以在基本几何图形中，也可以在对照几何图形中，并除去两者都包含的；（6）并集（union）既包含基本几何图形，又包含对照几

何图形。

几何 Geometry 对象实现的关系算子 IRelationalOperator 接口，可以实现基本几何对象与对照几何对象之间空间关系的判别，其方法均为布尔型 Boolean 返回值。其可判别的空间关系主要有：（1）相离（disjoint）没有公共点；（2）相接（touch）两个几何图形在边界相交；（3）重叠（overlap）相交部分与两者有着相同维度的几何图形；（4）相交（cross）在比最高维度更低的维度中相交；（5）内含（within）一个几何图形不能内含于另一个更低维的几何图形。

几何 Geometry 对象实现的邻近算子 IProximityOperator 接口，可以查询返回基本几何对象到参照几何对象的最近距离和最近点，如图 7-2 所示。

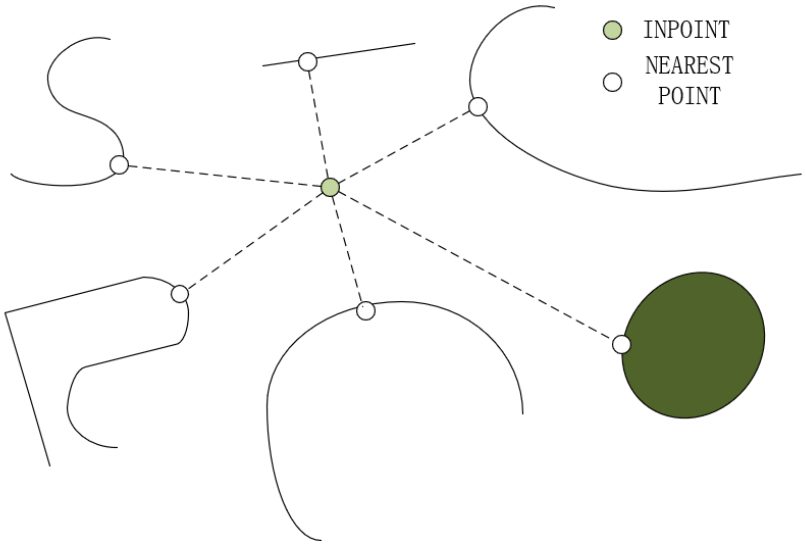

图 7-2 IProximityOperator 接口的最近点查询

7. 1. 3 图层叠加与空间统计分析

矢量图层叠加分析根据叠加对象分类，分为：点与多边形叠加分析、线与多边形叠加分析、多边形与多边形叠加分析。根据叠加操作分类，分为：叠加求交 Intersect、叠加求并 Union。矢量图层叠加分析调用的是可创建类基础地理处理器 BasicGeoprocessor 的相关接口的方法。

栅格图层叠加包含以下几种类型：（1）基于常数进行的代数运算；（2）基于数学变换进行指数、对数、三角变换等；（3）基于多个栅格要素层的运算，包括代数运算（加、减、乘、除、乘方等）和逻辑运算（与、或、非、异或等）。栅格图层叠加分析调用的是可创建类栅格代数算子集 RasterMathOps 的相关接口的方法。

矢量数据的空间统计分析主要调用的是可创建类基础统计组件 BaseStaticstics 和数据统计组件 DataStaticstics 的相关接口的方法。基础统计组件 BaseStaticstics 可生成和报告统计结果，实现的接口有：（1）IFrequencyStatistics 提供对用来报告频率统计的成员的访问；（2）IGenerateStaticstics 提供对用来生成统计结果的成员的访问；（3）IStaticsticsResults 提供对用来报告统计结果的成员的访问。数据统计组件 DataStaticstics 可返回统计结果及单个字段的唯一值（unique value）等。

7.2 演示实例

在 ArcGIS 桌面系统中通过 ArcToolbox 提供了大量的地理处理工具(Geoprocessing Tools)，在 ArcEngine 中每个工具都有一个对应的类；每个工具箱的名称对应类库的命名空间，如“ESRI.ArcGIS.*Tools”。地理处理工具的使用方法有两种：

其一，Geoprocessing 组件使用方法，其调用方式如下：

功能：Geoprocessing 组件使用方法

IGeoProcessor2 gp= new GeoProcessorClass();
//创建 IVariantArray 接口的对象，设置参数
gp.Execute(“tool_name”, parameters, null);

其二，Geoprocessor 组件方法，其调用方式如下：

功能：Geoprocessor 组件使用方法

Geoprocessor GP= new Geoprocessor ();
//实例化工具类，设置参数
GP.Execute(toolObj,null);

下面分别演示这两种调用方法：

（1）在 Visual Studio 中新建项目，选择【Visual C#】→【ArcGIS】→【Extending ArcObjects】→【MapControl Application】模板，名称为“MapControlApplication”，解决方案为“Test7”，如图 7-3 所示。

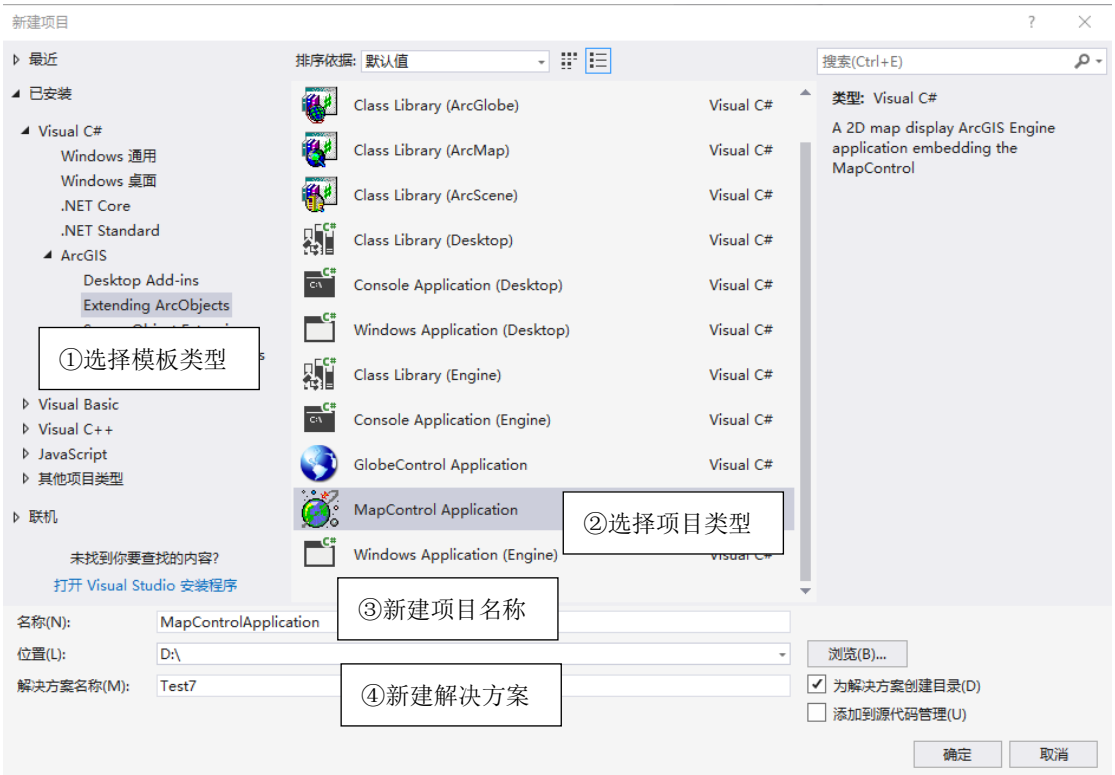

图 7-3 新建 MapControl Application 项目

设置 Linsence 权限如图 7-4 所示。

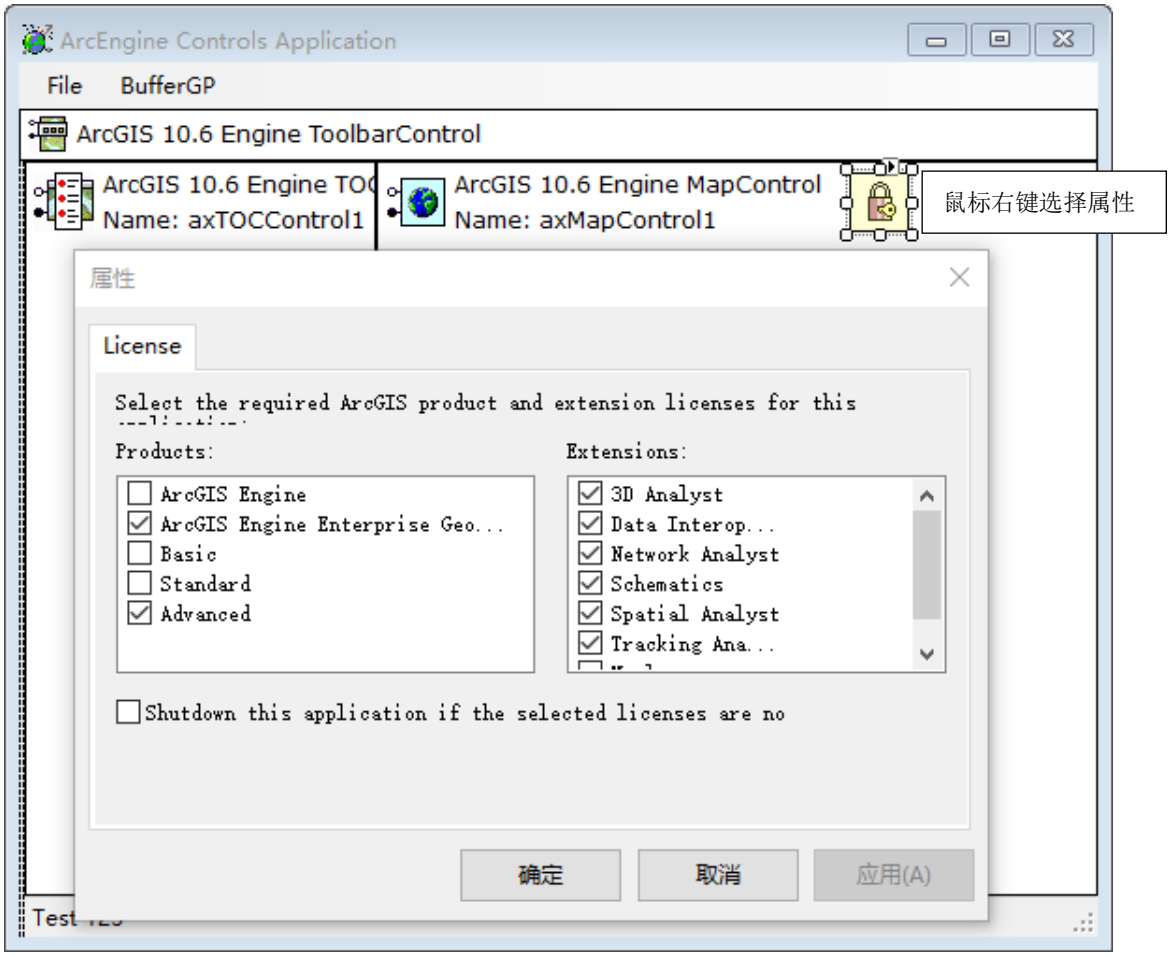

图 7-4 设置 License 控件的授权

（2）解决方案资源管理器选中“MapControlApplication”项目，右键选择【添加】→【新建项】，新建一个 Windows 窗体用来调用 GP 工具生成缓冲区，名称为“FrmBufferGP”，如图 7-5 所示。

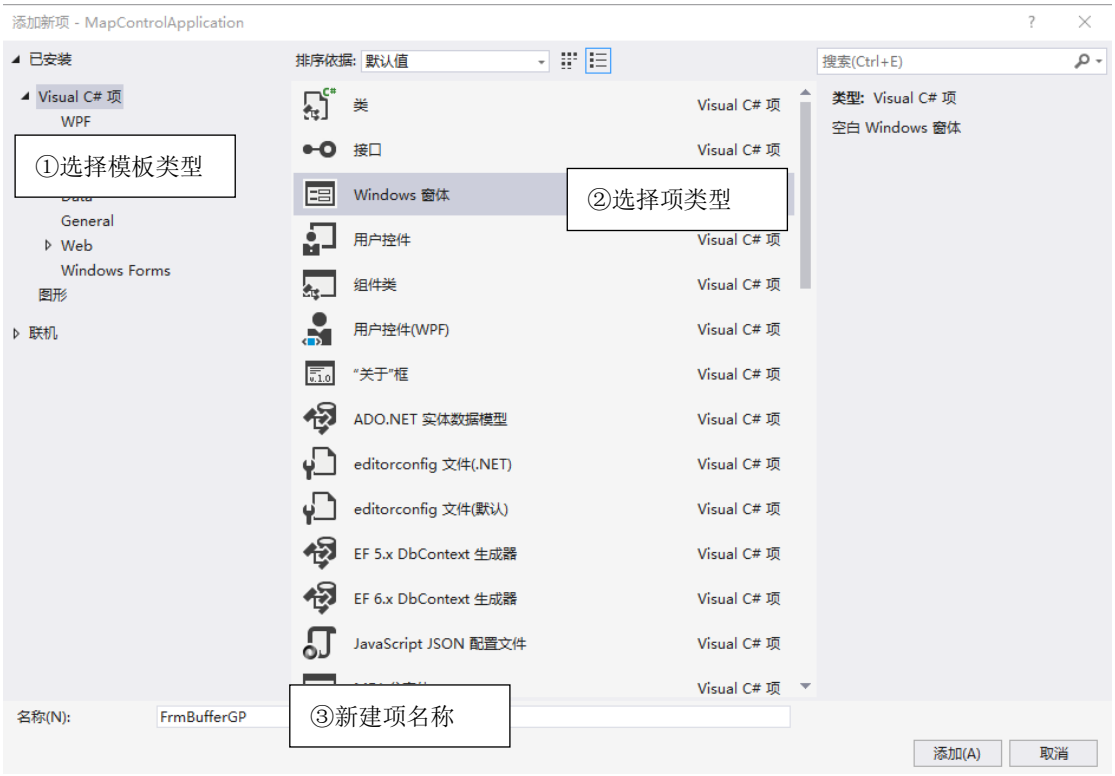

图 7-5 新建创建缓冲区窗体

为窗体类“FrmBufferGP”添加窗体控件并修改各控件的 Name 属性值，如图 7-6 所示。

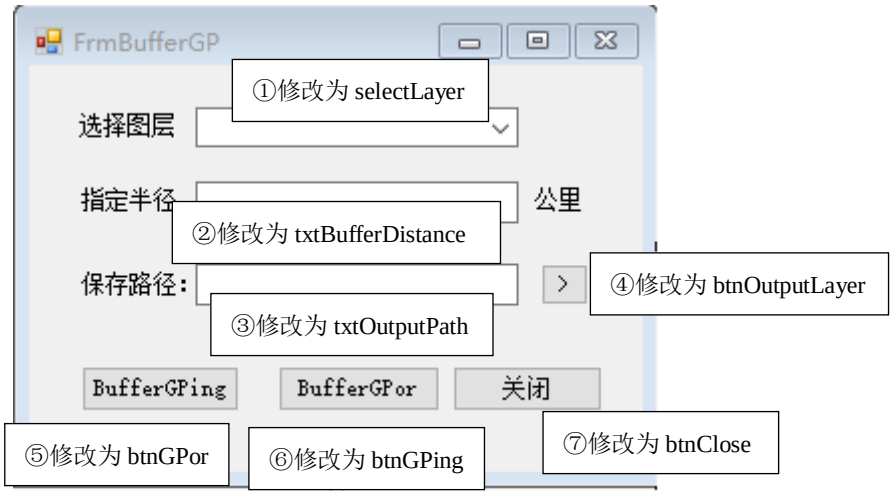

图 7-6 修改各控件的 Name 属性值

（3）在 FrmBufferGP.cs 中添加引用：

```
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.DataSourcesFile;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geoprocessing;
using ESRI.ArcGIS.AnalysisTools;
```

```
using ESRI. ArcGIS. Geoprocessor;
```

并将 Geoprocessing、Geodatabase、DataSourcesFile 等引用的嵌入互操作类型改为 False。

为类“FrmBufferGP”添加“IHookHelper”接口类型的私有成员变量 m_hookHelper，在类的构造函数中将该引用变量指向具体的对象。

FrmBufferGP.cs（节选） | 功能：定义 IHookHelper 接口获取主应用程序资源

```
private IHookHelper m_hookHelper = null;
public FrmBufferGP(object hook)
{
    if (m_hookHelper == null)
        m_hookHelper = new HookHelperClass();
    m_hookHelper. Hook = hook;
    InitializeComponent();
}
```

（4）点击窗体 FrmBufferGP 空白处添加 FrmBufferGP_Load()窗体加载事件响应函数，然后添加 CbxLayersAddItems()私有成员函数，获取当前所有矢量图层。

FrmBufferGP.cs（节选） | 功能：将当前地图所有图层添加到组合框

```
private void FrmBufferGP_Load(object sender, EventArgs e)
{
    CbxLayersAddItems();
}
//获取当前地图所有矢量图层，并添加到组合框
private void CbxLayersAddItems()
{
    IEnumLayer layers = m_hookHelper. FocusMap. Layers;
    layers. Reset();
    ILayer layer = layers. Next();
    while (layer != null)
    {
        if (layer is IFeatureLayer)
        {
            selectLayer. Items. Add(layer. Name);
        }
        layer = layers. Next();
    }
}
```

（5）双击“btnOutputLayer”控件，添加按钮单击事件响应函数，设置输出图层保存路径；双击“关闭”按钮，添加按钮单击事件响应函数，关闭窗体。

FrmBufferGP.cs（节选） | 功能：设置输出图层保存路径

```
private void btnOutputLayer_Click(object sender, EventArgs e)
{
    //设置输出图层的路径
    SaveFileDialog saveDlg = new SaveFileDialog();
```

```
        saveDlg. Title = ″保存为shp文件″;
        saveDlg. Filter = ″shapefile文件（*. shp）|*. shp″;
        if（saveDlg. ShowDialog() == DialogResult. OK）
        {
            txtOutputPath. Text = saveDlg. FileName;
        }
}
//关闭窗体
private void btnClose_Click(object sender, EventArgs e)
{
        this. Close();
}
```

（6）双击“BufferGPing”控件，添加按钮单击事件响应函数，调用Geoprocessing 组件生成缓冲区。

FrmBufferGP.cs（节选）	功能：调用 Geoprocessing 组件生成缓冲区。

```
private void btnGPing_Click(object sender, EventArgs e)
{
        double bufferDistance;                  //缓冲距离
        double.TryParse(txtBufferDistance.Text, out bufferDistance);
        //获取需要生成缓冲区的图层
        IFeatureLayer pFeatureLayer = (IFeatureLayer)
                                GetLayerByName(selectLayer.SelectedItem.ToString());
        Geoprocessor gp = new Geoprocessor();
        //设置是否覆盖原有文件
        gp.OverwriteOutput = true;
        gp.AddOutputsToMap = true;
        string unit = "Kilometers";
        //实例化Buffer对象
        ESRI.ArcGIS.AnalysisTools.Buffer buffer =
                                        new ESRI.ArcGIS.AnalysisTools.Buffer(
                                            pFeatureLayer, txtOutputPath.Text,
                                    Convert.ToString(bufferDistance) + " " + unit);
        //执行地理处理工具
        IGeoProcessorResult results = (IGeoProcessorResult)gp.Execute(buffer, null);
        //添加缓冲区图层到当前地图中
        Add2Map();
}
```

在 FrmBufferGP.cs 中添加私有成员函数 GetLayerByName ()获取图层和Add2Map()添加缓冲区图层到当前地图。

FrmBufferGP.cs（节选）	功能：根据名字获取图层

```
//根据名字获取图层
private ILayer GetLayerByName(string strLayerName)
{
        ILayer pLayer = null;
```

```
        for (int i = 0; i <= m_hookHelper.FocusMap.LayerCount - 1; i++)
        {
            if (strLayerName == m_hookHelper.FocusMap.get_Layer(i).Name)
            { pLayer = m_hookHelper.FocusMap.get_Layer(i); break; }
        }
        return pLayer;
}
//添加缓冲区图层到当前地图中
private void Add2Map()
{
        string fileDirectory = txtOutputPath.Text.ToString().Substring(0,
                                                txtOutputPath.Text.LastIndexOf("\\"));
        int j;
        j = txtOutputPath.Text.LastIndexOf("\\");
        string tmpstr = txtOutputPath.Text.ToString().Substring(j + 1);
        IWorkspaceFactory pWorkspaceFactory = new ShapefileWorkspaceFactory()
                                                            as IWorkspaceFactory;
        IWorkspace pWS = pWorkspaceFactory.OpenFromFile(fileDirectory, 0);
        IFeatureWorkspace pFS = pWS as IFeatureWorkspace;
        IFeatureClass pfc = pFS.OpenFeatureClass(tmpstr);
        IFeatureLayer pfl = new FeatureLayer() as IFeatureLayer;
        pfl.FeatureClass = pfc;
        pfl.Name = pfc.AliasName;
        IRgbColor pColor = new RgbColor() as IRgbColor;
        pColor.Red = 255;
        pColor.Green = 0;
        pColor.Blue = 0;
        pColor.Transparency = 255;
        //产生一个线符号对象
        ILineSymbol pOutline = new SimpleLineSymbol();
        pOutline.Width = 2;
        pOutline.Color = pColor;
        //设置颜色属性
        pColor = new RgbColor();
        pColor.Red = 255;
        pColor.Green = 0;
        pColor.Blue = 0;
        pColor.Transparency = 100;
        //设置填充符号的属性
        ISimpleFillSymbol pFillSymbol = new SimpleFillSymbol();
        pFillSymbol.Color = pColor;
        pFillSymbol.Outline = pOutline;
        pFillSymbol.Style = esriSimpleFillStyle.esriSFSSolid;
        ISimpleRenderer pRen;
```

```
        IGeoFeatureLayer pGeoFeatLyr = pfl as IGeoFeatureLayer;
        pRen = pGeoFeatLyr.Renderer as ISimpleRenderer;
        pRen.Symbol = pFillSymbol as ISymbol;
        pGeoFeatLyr.Renderer = pRen as IFeatureRenderer;
        ILayerEffects pLayerEffects = pfl as ILayerEffects;
        pLayerEffects.Transparency = 150;
        m_hookHelper.FocusMap.AddLayer(pfl);
}
```

（7）双击“BufferGPor”控件，添加按钮单击事件响应函数，调用 Geoprocessor 组件生成缓冲区。

FrmBufferGP.cs（节选）	功能：调用 Geoprocessor 组件生成缓冲区

```
private void btnGPor_Click(object sender, EventArgs e)
{
        double bufferDistance;                  //缓冲距离
        double.TryParse(txtBufferDistance.Text, out bufferDistance);
        //获取需要生成缓冲区的图层
        IFeatureLayer pFeatureLayer = (IFeatureLayer)GetLayerByName(
                                                selectLayer.SelectedItem.ToString());
        IGeoProcessor2 gp = new GeoProcessorClass();
        IGeoProcessorResult results = new GeoProcessorResultClass();
        gp.OverwriteOutput = true;
        string unit = "Kilometers";
        IVariantArray parameters = new VarArrayClass();
        parameters.Add(pFeatureLayer);
        parameters.Add(txtOutputPath.Text);
        parameters.Add(Convert.ToString(bufferDistance) + " " + unit);
        //执行地理处理工具
        results = gp.Execute("Buffer_analysis", parameters, null);
        //添加缓冲区图层到当前地图中
        Add2Map();
}
```

（8）添加菜单项 BufferGP 并双击该菜单项，添加 Click 事件响应函数如下所示。

MainForm. cs（节选）	功能：添加菜单项响应事件

```
//调用FrmBufferGP窗体生成缓冲区
private void bufferGPToolStripMenuItem_Click(object sender, EventArgs e)
{
        FrmBufferGP frmBufferGP = new FrmBufferGP(m_mapControl. Object);
        frmBufferGP. Show();
}
```

（9）点击菜单栏【生成】→【重新生成解决方案】编译通过后，然后点击菜单栏【调试】→【开始执行】，加载实验数据，点击“BufferGP”，选择需要创建缓冲区的图层，设置生成缓冲区半径，自定义生成缓冲区的名称和保存路径，如图7-7所示。

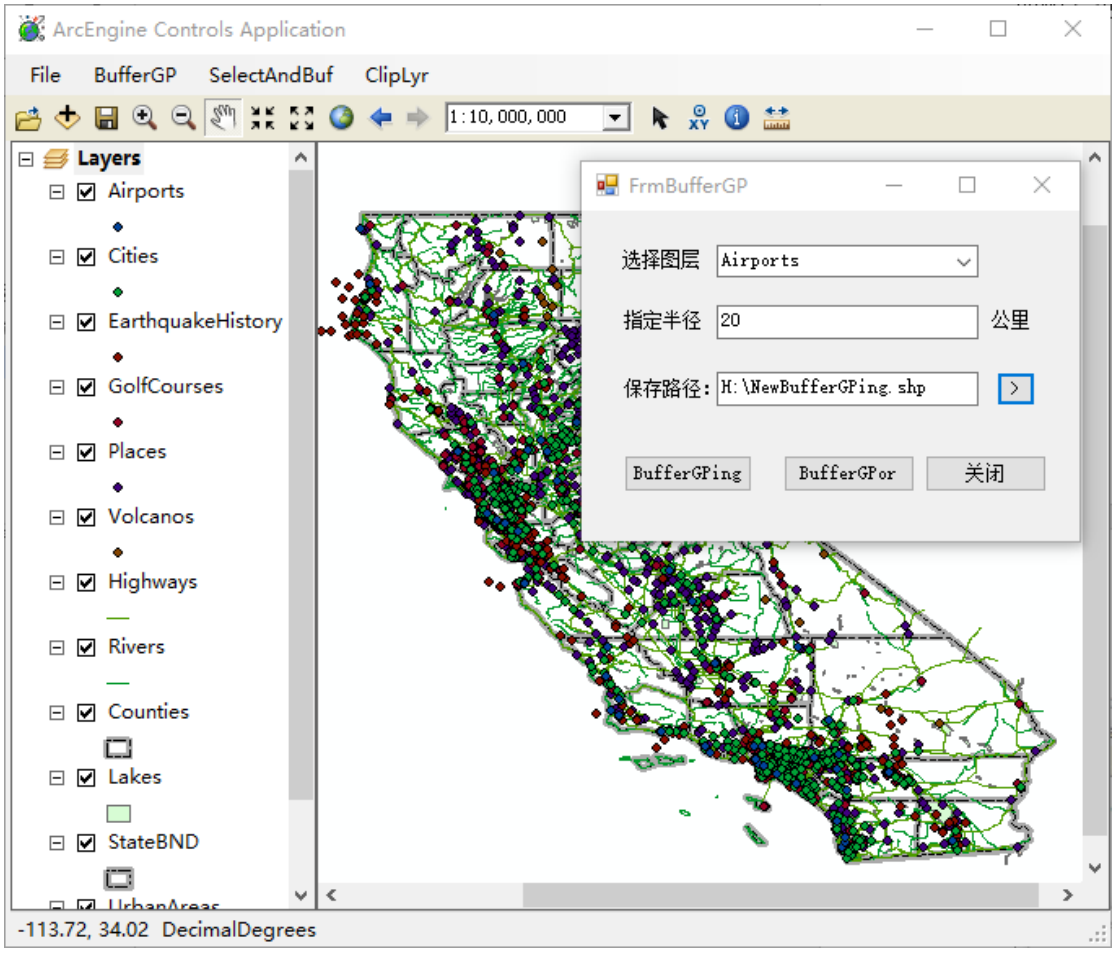

图 7-7 设置生成缓冲区的参数

设置缓冲区半径为 20，点击 BufferGPing，设置输出图层为“NewBufferGPing”，如图 7-8 所示。

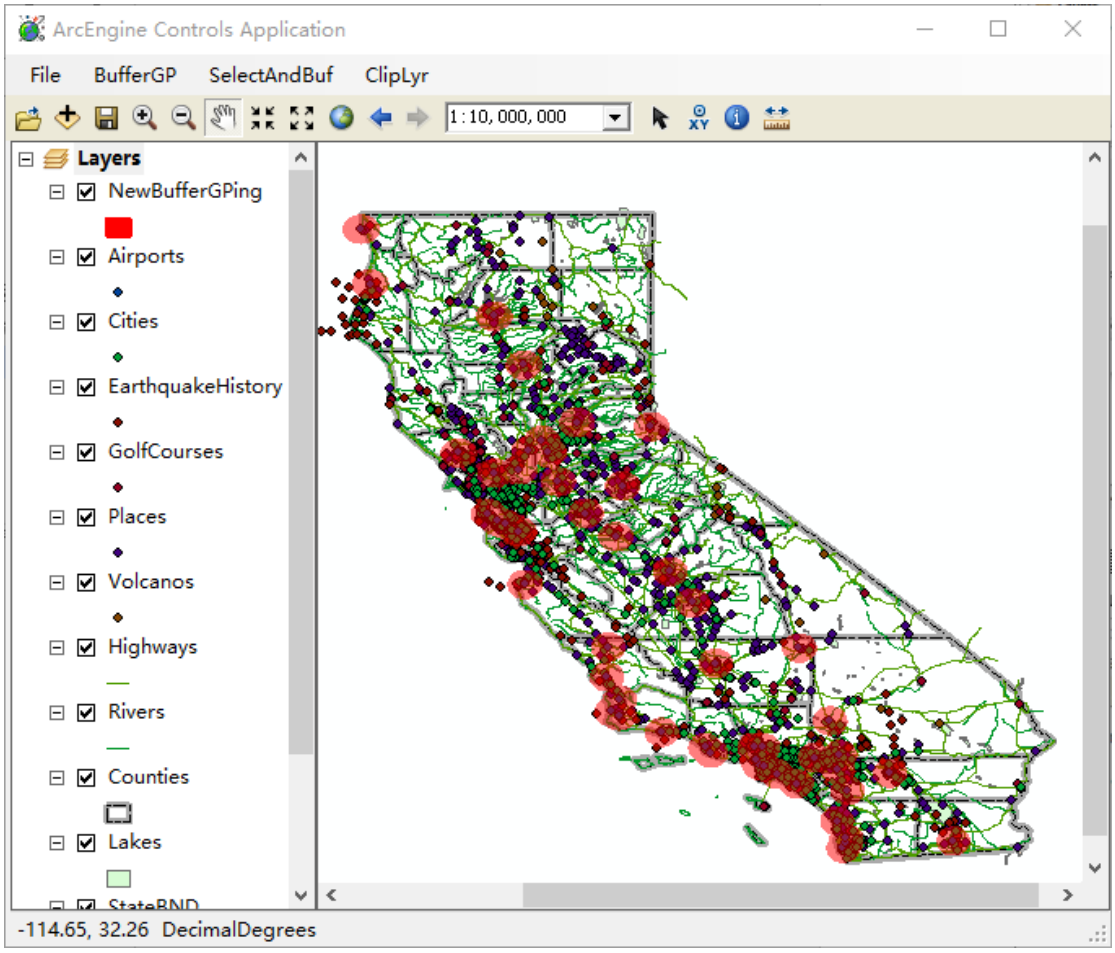

图 7-8 调用 Geoprocessing 生成缓冲区

设置缓冲区半径为 30，点击 BufferGPor，设置输出图层为“NewBufferGPor”，如图 7-9 所示。

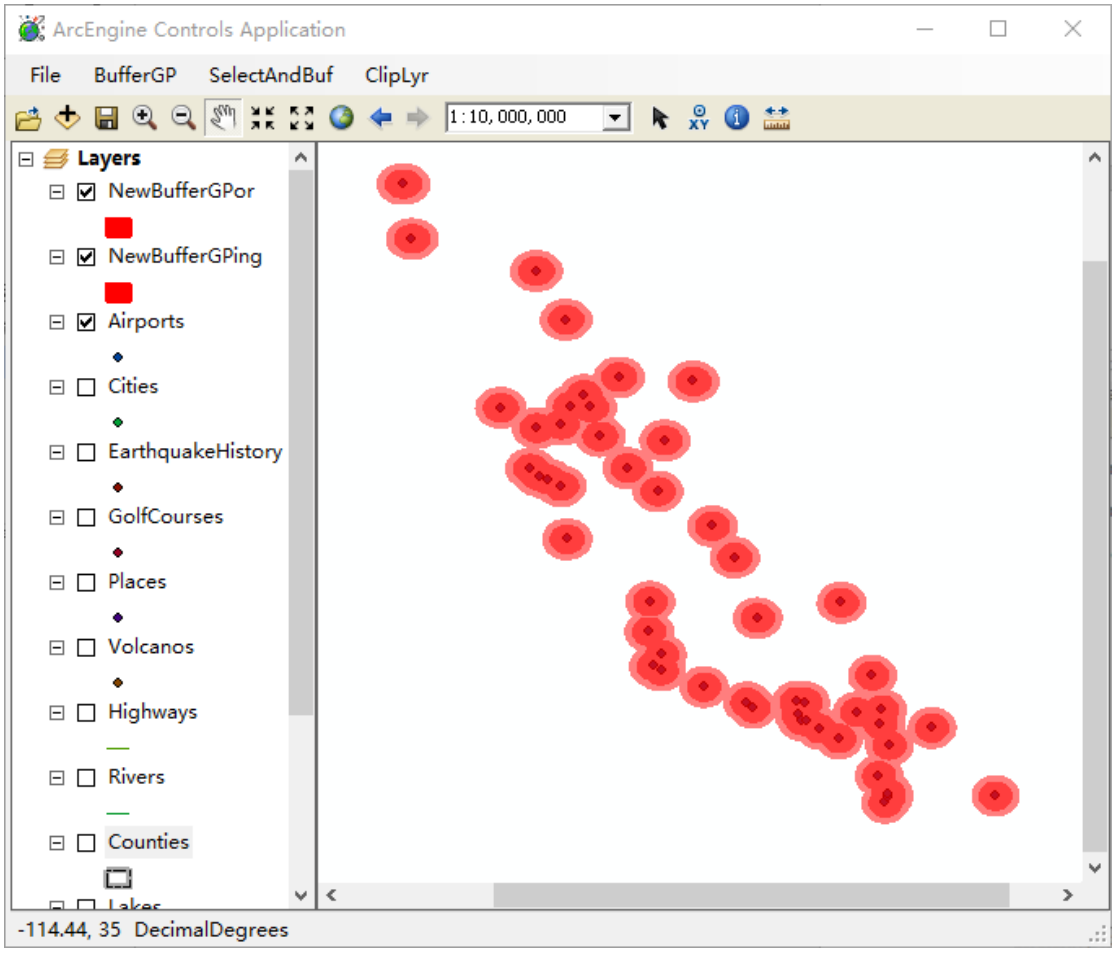

图 7-9 调用 Geoprocessor 生成缓冲区

7.3 实验目的

熟悉矢量数据的相关空间分析方法以及地理处理（GP）工具的调用方法。

7.4 实验内容

（1）添加工具 Tool 进行交互选择要素，选择方式（点、矩形、圆形、多边形）不限，对选中的要素建立缓冲区，并将缓冲区添加到多边形要素类图层中；

（2）使用地理处理（GP）工具对要素图层进行裁剪等。

7.5 实验数据

见安装目录：

...\DeveloperKit10.6\Samples\data\ California

7.6 实验步骤

7. 6. 1 单个要素的缓冲区分析

本例实现的是通过工具 Tool 交互选择单个要素，对要素做缓冲区并添加到多边形要素类中。具体步骤如下：

（1）解决方案资源管理器选中“MapControlApplication”项目，右键选择【添加】→【新建项】，新建一个工具类“ToolSelectAndBuf”，如图 7-10 所示。

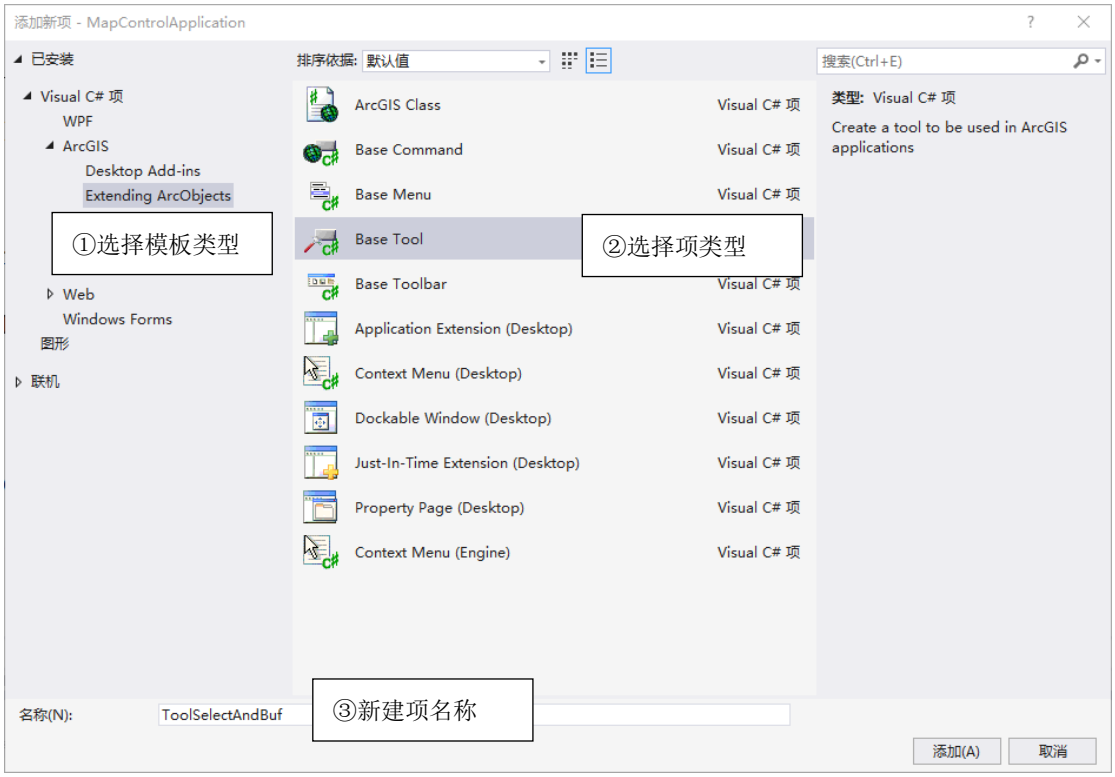

图 7-10 添加工具类

（2）在 ToolSelectAndBuf.cs 中添加引用：

```
using ESRI. ArcGIS. Carto;
using ESRI. ArcGIS. Display;
using ESRI. ArcGIS. Geometry;
using ESRI. ArcGIS. Geodatabase;
```

将 Geometry 的嵌入互操作类型改为 False。为类“ToolSelectAndBuf”定义私有成员变量 m_selectedLyr 用于选择要素的图层、私有成员变量 m_polygonLyr 用于保存生成缓冲区、私有成员变量 m_dBufDist 用于自定义生成缓冲区的半径。

```
private IFeatureLayer m_selectLyr = null;
private IFeatureLayer m_polygonLyr = null;
private double m_dBufDist = 0. 0;
```

在 ToolSelectAndBuf.cs 中添加点击事件 OnClick 响应函数的代码。

ToolSelectAndBuf.cs（节选）	功能：选择图层，设置创建缓冲区参数
//调用FrmSelectAndBuf窗体设置生成缓冲区的参数	

```
public override void OnClick()
{
    // TODO: Add ToolSelectAndBuf.OnClick implementation
    FrmSelectAndBuf frmBuffer = new FrmSelectAndBuf(m_hookHelper);
    if (frmBuffer.ShowDialog() == DialogResult.OK)
    {
        m_selectLyr = frmBuffer.selectLyr;
        m_polygonLyr = frmBuffer.polygonLyr;
        m_dBufDist = frmBuffer.bufDist;
    }
}
```

（3）然后添加鼠标按下事件响应函数 OnMouseDown()，代码如下：

ToolSelectAndBuf.cs（节选）	功能：实现鼠标按下交互选择创建缓冲区

```
public override void OnMouseDown(int Button, int Shift, int X, int Y)
{
    // TODO:  Add ToolSelectAndBuf.OnMouseDown implementation
    //判断图层是否为空
    if (m_selectLyr == null || m_polygonLyr == null)
        return;
    IFeatureLayer ipSelLyr = m_selectLyr;
    IFeatureLayer ipPolygonLyr = m_polygonLyr;
    //进行多边形选择
    IRubberBand ipRubber = new RubberEnvelopeClass();
    IGeometry polygon = ipRubber.TrackNew(m_hookHelper.
                                    ActiveView.ScreenDisplay, null);
    ISpatialFilter ipSpatialFilter = new SpatialFilterClass();
    ipSpatialFilter.Geometry = polygon;
    ipSpatialFilter.SpatialRel = esriSpatialRelEnum.esriSpatialRelIntersects;
    IFeatureSelection ipFeatSelect = ipSelLyr as IFeatureSelection;
    ipFeatSelect.Clear();
    //选择单个要素
    ipFeatSelect.SelectFeatures(ipSpatialFilter,
                    esriSelectionResultEnum.esriSelectionResultNew, true);
    ipFeatSelect.SelectionSet.Refresh();
    m_hookHelper.ActiveView.PartialRefresh(esriViewDrawPhase.
                                    esriViewGeoSelection, null, null);
    //生成选择要素的缓冲区
    BufferSeclected(ipFeatSelect, ipPolygonLyr);
    m_hookHelper.ActiveView.PartialRefresh(esriViewDrawPhase.
                                    esriViewGeoSelection, null, null);
}
//调用 ITopologicalOperator 接口生成缓冲区
private void BufferSeclected(IFeatureSelection ipFeatSelect,
                                    IFeatureLayer ipPolygonLyr)
```

```
{
    ICursor cur = null;
    if (ipFeatSelect.SelectionSet.Count <= 0) return;
    ipFeatSelect.SelectionSet.Search(null, true, out cur);
    IFeature feature = cur.NextRow() as IFeature;
    ITopologicalOperator to = feature.Shape as ITopologicalOperator;
    //自定义缓冲区半径
    IPolygon poly = to.Buffer(m_dBufDist) as IPolygon;
    IFeature polyFeature = ipPolygonLyr.FeatureClass.CreateFeature();
    polyFeature.Shape = poly;
    polyFeature.Store();
    m_hookHelper.ActiveView.Refresh();
}
```

（4）新建一个 Windows 窗体用来选择图层以供交互选择要素，名称为“FrmSelectAndBuf”，为窗体类“FrmSelectAndBuf”添加窗体控件并修改各控件的 Name 属性值，如图 7-11 所示。

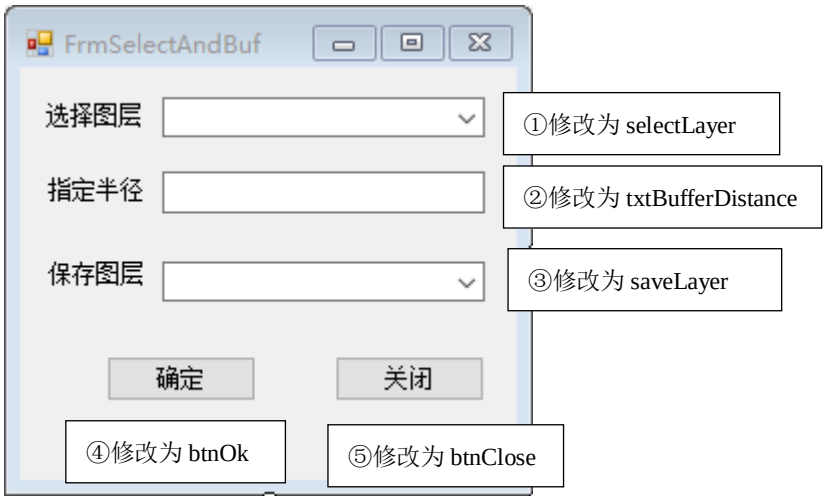

图 7-11 修改各控件的 Name 属性值

（5）为窗体类“FrmSelectAndBuf”添加引用：

```
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.Geometry;
```

为窗体类“FrmSelectAndBuf”添加私有成员变量“IFeatureLayer m_ipSelectedLyr”和“IFeatureLayer m_ipPolygonLyr”，用来选择图层生成缓冲区和保存缓冲区至多边形要素图层；为类添加“IHookHelper”接口类型的私有成员变量 m_hookHelper，在类的构造函数中将该引用变量指向具体的对象。

FrmSelectAndBuf.cs（节选）	功能：定义 IHookHelper 接口获取主应用程序资源

```
private IFeatureLayer m_ipSelectedLyr = null;
private IFeatureLayer m_ipPolygonLyr = null;
private IHookHelper m_hookHelper = null;
public FrmSelectAndBuf(IHookHelper hook)
{
    InitializeComponent();
```

```
    m_hookHelper = hook;
}
```

（6）点击窗体“FrmSelectAndBuf”空白处添加 FrmSelectAndBuf_Load()窗体加载事件响应函数获取当前地图中的图层列表；双击名称为“selectLayer”的控件，添加图层选择事件响应函数；双击名称为“saveLayer”的控件，添加图层选择事件响应函数。

FrmSelectAndBuf.cs（节选） | 功能：获取图层列表和当前选择图层

```
//获取当前图层
private void FrmSelectAndBuf_Load(object sender, EventArgs e)
{
    for (int i = 0; i < m_hookHelper.FocusMap.LayerCount; i++)
    {
        ILayer lyr = m_hookHelper.FocusMap.get_Layer(i);
        IFeatureLayer fLyr = lyr as IFeatureLayer;
        if (fLyr != null)
            selectLayer.Items.Add(lyr.Name);
        if (fLyr != null && fLyr.FeatureClass.ShapeType ==
                                esriGeometryType.esriGeometryPolygon)
            saveLayer.Items.Add(lyr.Name);
    }
}
//获取进行要素选择和缓冲区分析的图层
private void selectLayer_SelectedIndexChanged(object sender, EventArgs e)
{
    for (int i = 0; i < m_hookHelper.FocusMap.LayerCount; i++)
    {
        ILayer lyr = m_hookHelper.FocusMap.get_Layer(i);
        if (lyr.Name == selectLayer.Text)
        {
            m_ipSelectedLyr = lyr as IFeatureLayer;
            break;
        }
    }
}
//获取保存缓冲区多边形要素的图层
private void saveLayer_SelectedIndexChanged(object sender, EventArgs e)
{
    for (int i = 0; i < m_hookHelper.FocusMap.LayerCount; i++)
    {
        ILayer lyr = m_hookHelper.FocusMap.get_Layer(i);
        if (lyr.Name == saveLayer.Text)
        {
            m_ipPolygonLyr = lyr as IFeatureLayer;
            break;
```

```
            }
        }
}
```

（7）双击“确定”和“取消”按钮，添加按钮单击事件响应函数。

FrmSelectAndBuf.cs（节选）	功能：获得选中的要素图层

```
//获得选择图层
private void btnOk_Click(object sender, EventArgs e)
{
    DialogResult = DialogResult.OK;
    this.Close();
}
//关闭窗体
private void btnClose_Click(object sender, EventArgs e)
{
    this.Close();
}
```

（8）添加菜单项 SelectAndBuf，双击该菜单项，添加 Click 事件响应函数如下所示。

MainForm.cs（节选）	功能：菜单事件响应函数

```
//调用Tool工具创建缓冲区
private void selectAndBufToolStripMenuItem_Click(object sender, EventArgs e)
{
    ICommand command = new ToolSelectAndBuf();
    command.OnCreate(m_mapControl.Object);
    m_mapControl.CurrentTool = command as ITool;
}
```

（9）点击菜单栏【生成】→【重新生成解决方案】编译通过后，然后点击菜单栏【调试】→【开始执行】，加载实验数据，点击“SelectAndBuf”弹出 FrmSelectAndBuf 窗体，选择需要选择要素的图层和缓冲区保存图层，自定义缓冲区半径，如图 7-12 所示。

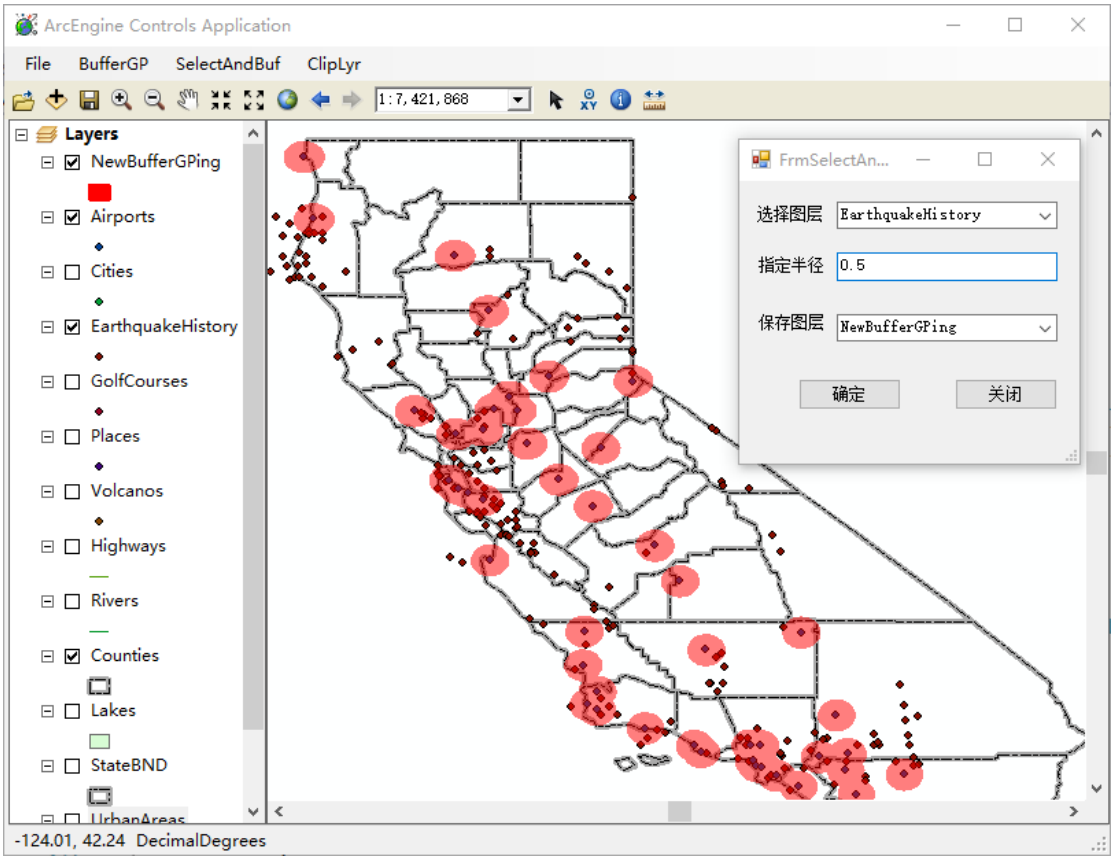

图 7-12 设置缓冲区分析参数

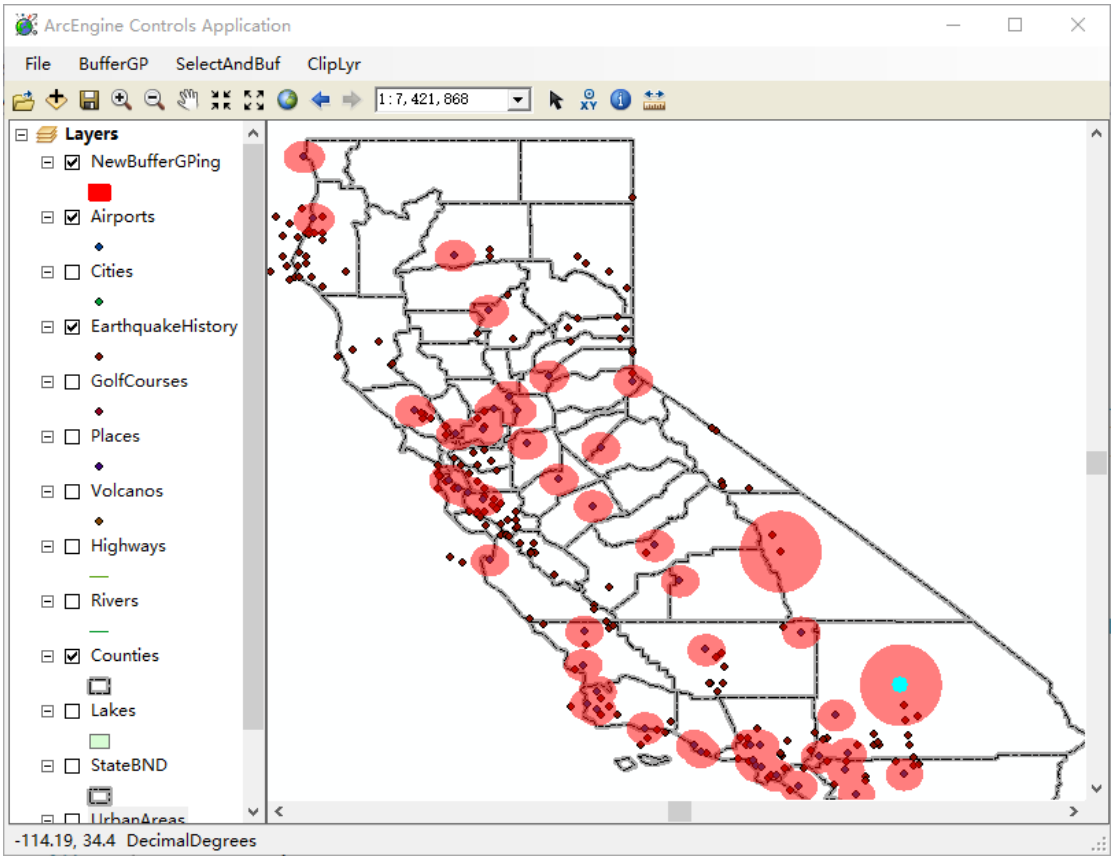

图 7-13 生成缓冲区功能运行效果

7. 6. 2 要素图层的裁剪分析

（1）在解决方案资源管理器下选择“MapControlApplication ”项目，右键选择【添加】→【新建项】，新建一个 Windows 窗体调用 GP 工具对特定图层进行裁剪，名称为“FrmClipLyr”，如图 7-14 所示。

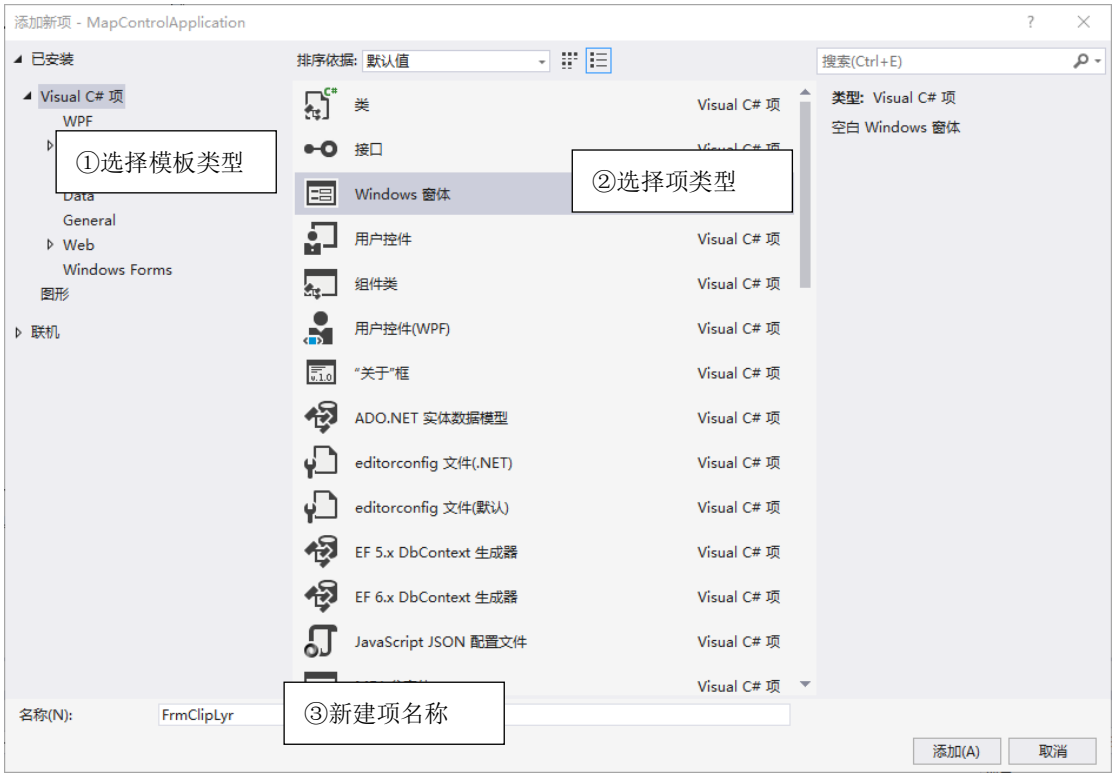

图 7-14 新建图层裁剪窗体

添加控件并修改各控件的 Name 属性值，如图 7-15 所示。

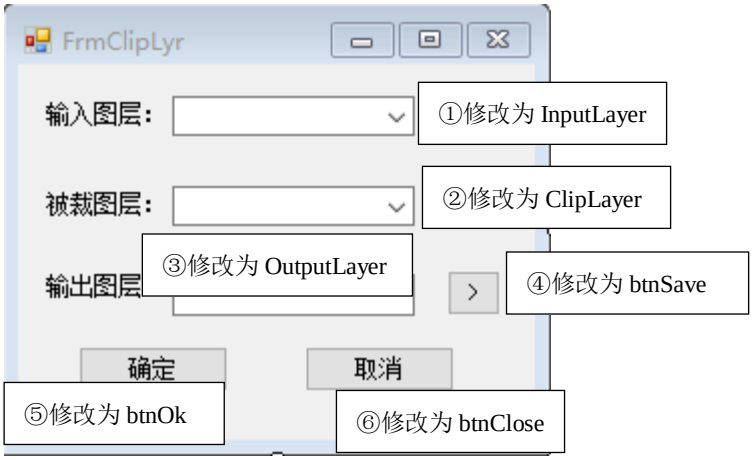

图 7-15 修改各控件的 Name 属性值

（2）在 FrmClipLyr.cs 中添加引用：

```
using ESRI. ArcGIS. Controls;
using ESRI. ArcGIS. AnalysisTools;
using ESRI. ArcGIS. Carto;
using ESRI. ArcGIS. DataSourcesFile;
using ESRI. ArcGIS. Geodatabase;
```

```
using ESRI.ArcGIS.Geoprocessor;
```

为类“FrmClipLyr”添加“IHookHelper”类型接口的私有成员变量 m_hookHelper，在类的构造函数中将该引用变量指向具体的对象；添加一个字符串型的私有成员变量 txtOutputPath 保存输出图层保存路径。

FrmClipLyr.cs（节选）	功能：定义 IHookHelper 接口获取主应用程序资源

```
private string txtOutputPath = null;
private IHookHelper m_hookHelper = null;
public FrmClipLyr(object hook)
{
    InitializeComponent();
    if (m_hookHelper == null)
      m_hookHelper = new HookHelperClass();
    m_hookHelper.Hook = hook;
 }
```

（3）点击窗体 FrmClipLyr 空白处添加 FrmClipLyr_Load()窗体加载事件响应函数，利用 m_hookHelper 获取当前的输入图层（多边形要素图层）和被裁图层。

FrmClipLyr.cs（节选）	功能：获取当前的输入图层和被裁图层

```
private void FrmClipLyr_Load(object sender, EventArgs e)
{
    for (int i = 0; i < m_hookHelper.FocusMap.LayerCount; i++)
    {
        //获取输入图层
        ILayer lyr = m_hookHelper.FocusMap.get_Layer(i);
        IFeatureLayer fLyr = lyr as IFeatureLayer;
        if (fLyr != null && fLyr.FeatureClass.ShapeType ==
                                esriGeometryType.esriGeometryPolygon)
            InputLayer.Items.Add(lyr.Name);
        //获取被裁的要素图层
        if (fLyr != null)
            ClipLayer.Items.Add(lyr.Name);
    }
}
```

（4）双击“btnSave”控件，添加按钮单击事件响应函数，设置输出图层保存路径；双击“取消”按钮，添加按钮单击事件响应函数，关闭窗体。

FrmClipLyr.cs（节选）	功能：设置输出图层保存路径

```
//设置输出图层保存路径
private void btnSave_Click(object sender, EventArgs e)
{
    SaveFileDialog dbfiledlg = new SaveFileDialog();
    dbfiledlg.Filter = "ShapeFile（*.shp）|*.shp";
    dbfiledlg.RestoreDirectory = true;
    if (dbfiledlg.ShowDialog() == DialogResult.OK)
    {
        txtOutputPath = dbfiledlg.FileName.ToString();
```

```
            if (System.IO.File.Exists(txtOutputPath))
                System.IO.File.Delete(txtOutputPath);
            OutputLayer.Text = txtOutputPath;
        }
    }
    //关闭窗体
    private void btnClose_Click(object sender, EventArgs e)
    {
        this.Close();
    }
```

（5）双击“确定”按钮，添加按钮单击事件响应函数，调用 GP 工具进行图层裁剪。

FrmClipLyr.cs（节选）	功能：执行图层裁剪

```
private void btnOk_Click(object sender, EventArgs e)
{
    Geoprocessor gp = new Geoprocessor();
    gp.OverwriteOutput = true;
    gp.AddOutputsToMap = true;
    IFeatureLayer pfl = null, pf2 = null;
    for (int i = 0; i < m_hookHelper.FocusMap.LayerCount; i++)
    {
        ILayer lyr = m_hookHelper.FocusMap.get_Layer(i);
        if (InputLayer.Text == lyr.Name)
            pfl = lyr as IFeatureLayer;
        if (ClipLayer.Text == lyr.Name)
            pf2 = lyr as IFeatureLayer;
    }
    //调用GP工具
    Clip myclip = new Clip();
    myclip.clip_features = pfl.FeatureClass;
    myclip.in_features = pf2.FeatureClass;
    myclip.out_feature_class = OutputLayer.Text;
    //执行裁剪工具
    gp.Execute(myclip, null);
    MessageBox.Show("裁剪完成");
    string fileDirectory = OutputLayer.Text.ToString().Substring(0,
      OutputLayer.Text.LastIndexOf("\\"));
    int j;
    j = OutputLayer.Text.LastIndexOf("\\");
    string tmpstr = OutputLayer.Text.ToString().Substring(j + 1);
    //添加缓冲区到当前图层
    IWorkspaceFactory pWorkspaceFactory = new
                ShapefileWorkspaceFactory() as IWorkspaceFactory;
    IWorkspace pWS = pWorkspaceFactory.OpenFromFile(
```

```
                                                    fileDirectory, 0);
    IFeatureWorkspace pFS = pWS as IFeatureWorkspace;
    IFeatureClass pfc = pFS.OpenFeatureClass(tmpstr);
    IFeatureLayer pf3 = new FeatureLayer() as IFeatureLayer;
    pf3.FeatureClass = pfc;
    pf3.Name = pfc.AliasName;
    m_hookHelper.FocusMap.AddLayer(pf3);
    this.Close();
}
```

（6）添加菜单项 ClipLyr 并双击该菜单项，添加 Click 事件响应函数如下所示。

MainForm.cs（节选）	功能：添加菜单项响应事件

```
//调用FrmClipLyr窗体进行图层裁剪
private void clipLyrToolStripMenuItem_Click(object sender, EventArgs e)
{
    FrmClipLyr frmClipLyr = new FrmClipLyr(m_mapControl.Object);
    frmClipLyr.Show();
}
```

（7）点击菜单栏【生成】→【重新生成解决方案】编译通过后，然后点击菜单栏【调试】→【开始执行】，加载实验数据，点击“ClipLyr”，选择裁剪图层和被裁图层，自定义输出图层的名称和保存路径，如图7-16所示。

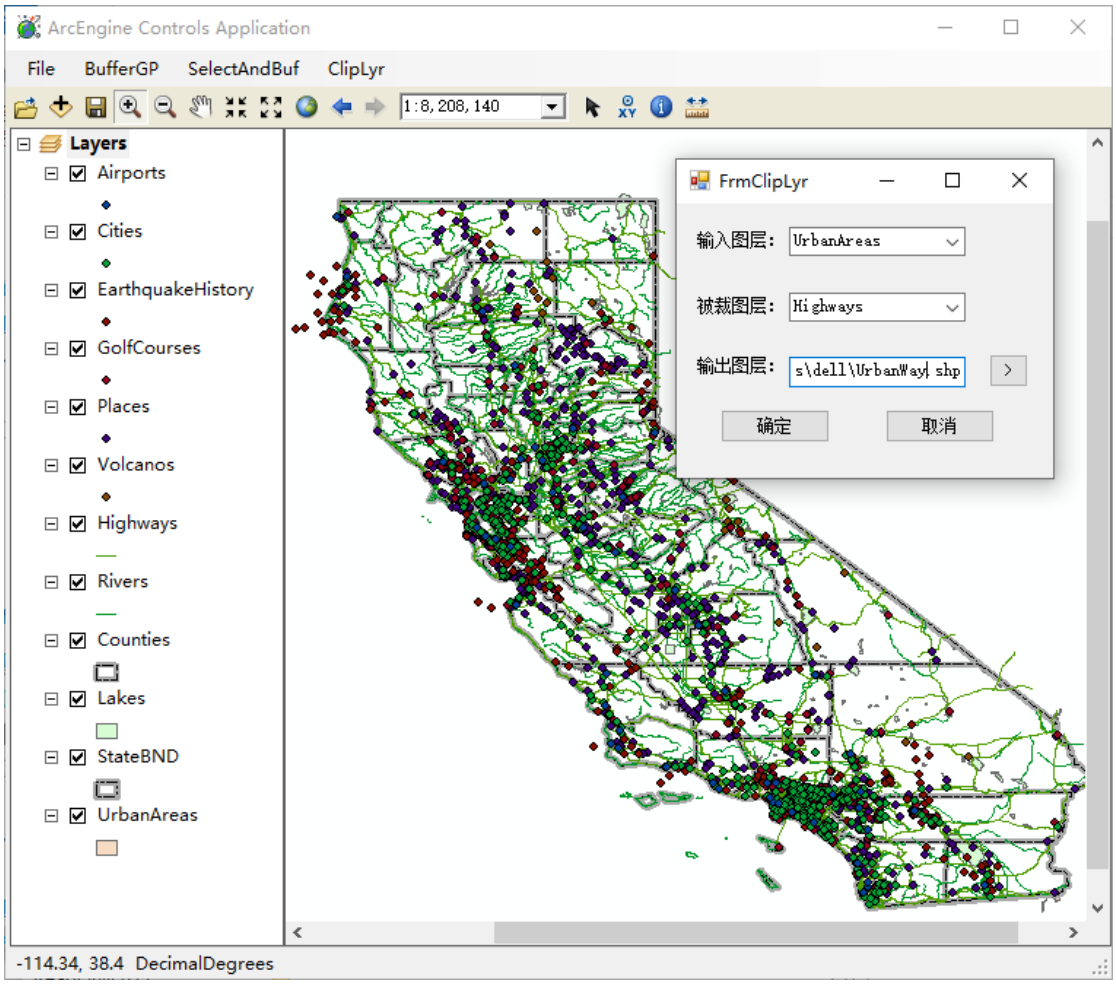

图 7-16 设置图层裁剪的参数

设置输出图层的名称为“UrbanWay.shp”，点击确定，如图 7-17 所示。

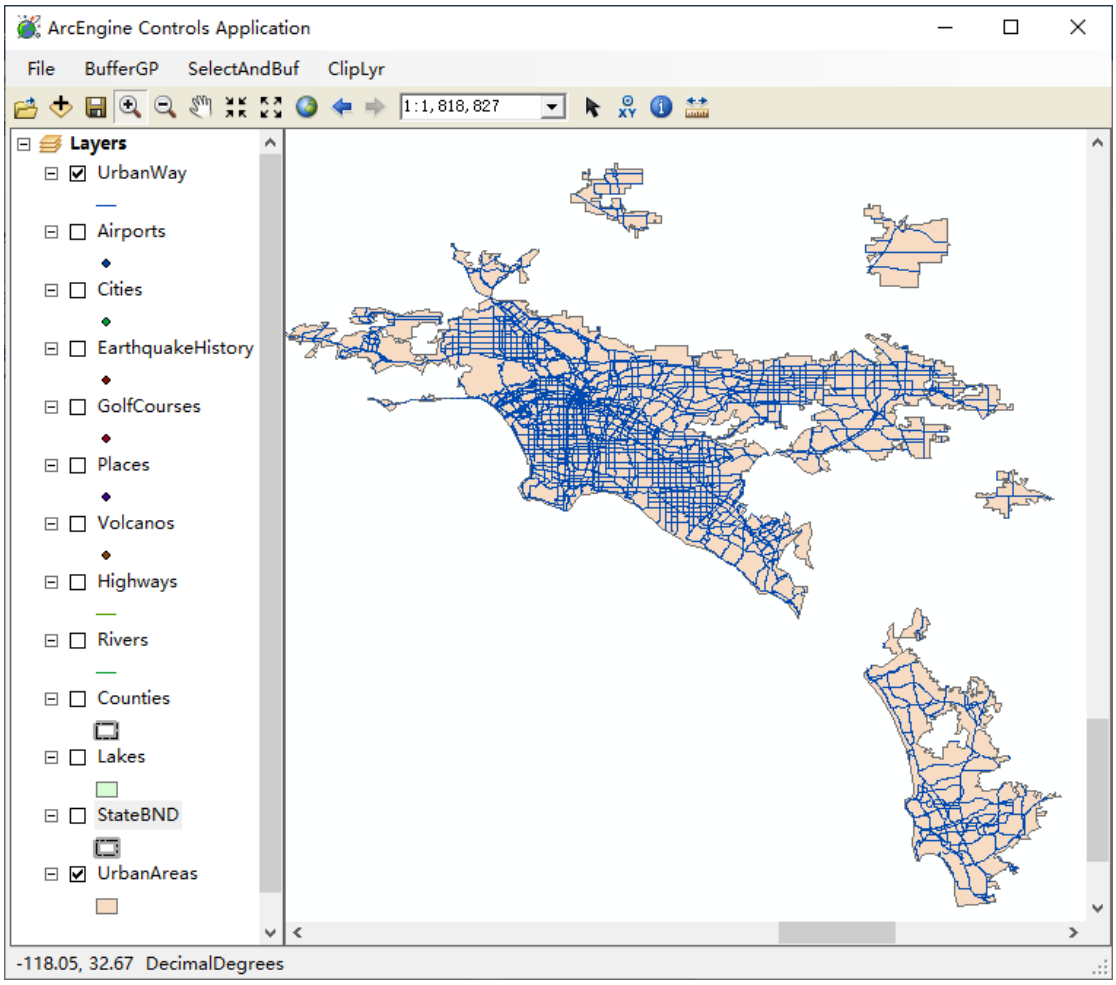

图 7-17 显示裁剪所得图层