实验六 几何对象操纵

6. 1 背景知识

6. 1. 1 几何对象模型

Geometry 是 ArcObjects 中使用最广泛的对象集之一，主要用于新建、删除、编辑地理要素，以及进行空间选择、要素渲染、制作专题图、标注编辑和地理分析等。在 ArcObjects 中几何对象模型如图 6-1 所示。

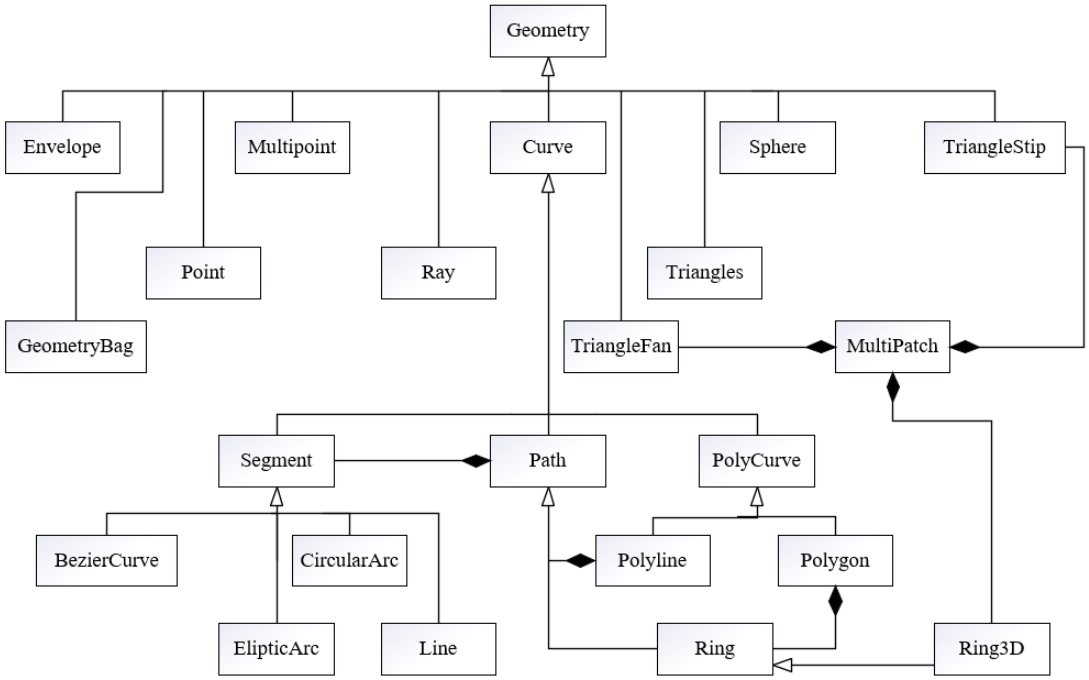

图 6-1 几何对象模型

在 ArcObjects 中几何对象模型对应的几何形状如图 6-2 所示，具体包括：

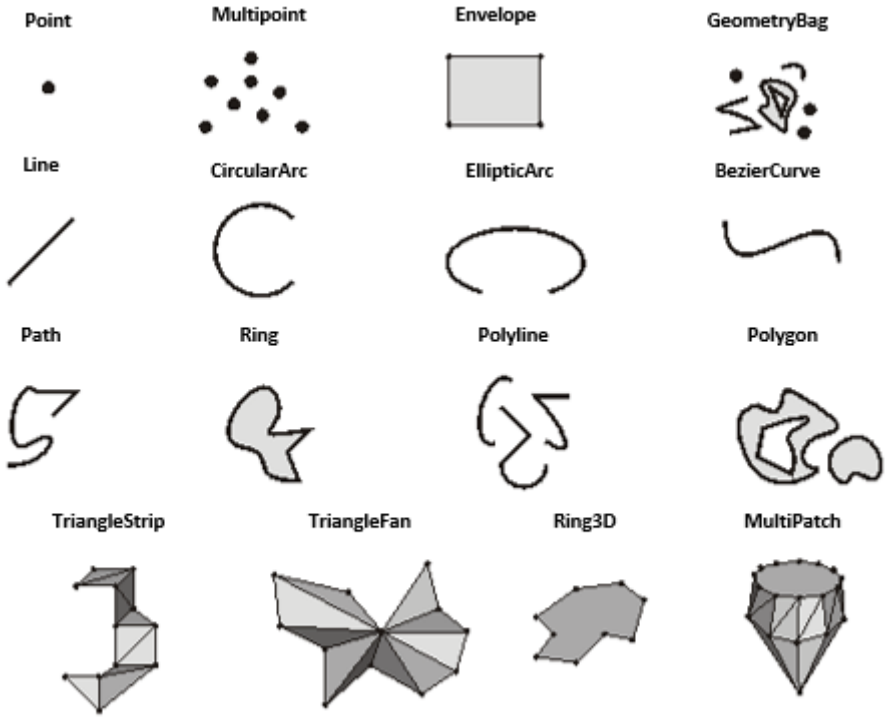

图 6-2 各类几何对象

（1）点 Point 是一个 0 维的具有 X、Y 坐标的几何对象，具有三种可以选择的属性，即 Z 值、M 值和 ID 值。Point 对象的作用有：（a）用于描述点类型的要素；（b）在寻址和符号化中使用；（c）用于组成一个网络 Network；（d）任何几何对象都可以使用点来产生。

（2）多点 MultiPoint 对象持有一个无序点对象的群集，这些点是具有相同属性设置的同一组点。MultiPoint 对象的作用有：（a）在构成高级几何对象、几何对象动态模拟等方面都起了重要的作用；（b）常常作为其他运算的结构而出现，如选择一条曲线上的平分点等。

（3）包络线 Envelope 是一个矩形区域，它是作为任何一个几何形体的最小边框区域而存在的。所有几何 Geometry 对象都拥有一个 Envelope 对象，表示几何对象的空间范围，即使是 Envelope 本身。它也常常作为地图的视图或地理数据库的范围和用户交互操作的结果而返回。

（4）曲线 Curve 对象，除去点、点集和包络线对象外，几乎其它所有的几何形体对象都可以看作是 Curve，它具有一维视图或者二维边界形状的几何对象。

（5）段 Segment 是 Curve 的一个子类，Segment 是一个起始点、一个终止点以及定义两点之间的曲线的函数组成的一维几何形体对象。更复杂的几何形体对象，如 Ring、Path、Polyline、Polygon 都可以有 Segment 对象集合来创建，其中 Ring、Path 类支持 ISegmentCollection 接口，而 Polyline、Polygon 类则支持 IGeometryCollection 接口。Segment 是一个抽象类，其四个子类包括线段 Line、圆弧 CircularArc、椭圆弧 EllipticArc 和贝塞尔曲线 BezierCurve（如图 6-3 所示）。Line 是最简单的 Segment，它是由起始点和终止点决定的一条直线段，它是一维几何对象；Line 也是最常使用的 Segment 对象，通常用于构造 Polyline、Polygon、Ring 和 Path 对象。

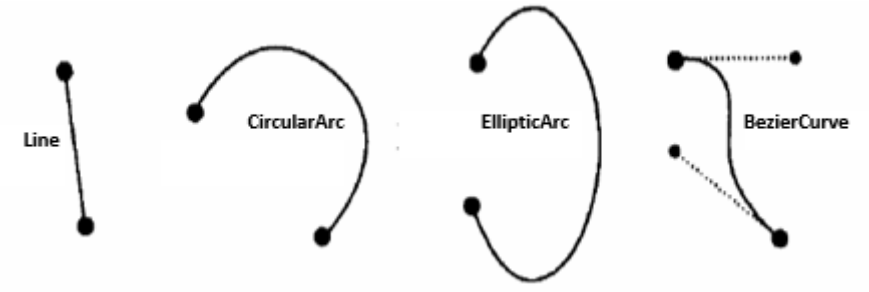

图 6-3 四类 Segment 对象

（6）路径 Path 是连续 Segment 对象的集合，除了路径的第一个和最后一个组成 Segment 外，每一个片断的起始点都是前一个片断的终止点。路径可以是任意数目的 Line、CircularArc、EllipticArc、BezierCurve 的组合；一个或多个路径对象组成一个 Polyline 对象。

（7）环 Ring 是起始点与终止点相重合的 Path 对象。Ring 对象具有以下几个关键特征：（a）它包含一系列首尾相连的同方向的 Segment 对象；（b）它是封闭的，即起始点与终止点是同一点；（c）它不能自相交。

（8）多曲线 PolyCurve 是一个抽象类，代表了一个 Polyline 或 Polygon 对象的边框线。它是由多个曲线构成的对象，Polyline 的组成部分是 Path，Polygon 的组成部分是 Ring。

（9）多义线 Polyline 对象是由一个或多个相连或者不相连的路径 Path 对象的有序集合（如图 6-4 所示），通常用来代表线状地物，如道路、河流、管线等。Polyline 对象需要满足的准则：（a）组成 Polyline 的 Path 对象都是有效的；（b）Path 不会重合、相交或自相交；（c）多个 Path 对象可以连接与某一节点，也可以是分离的；（d）长度为 0 的 Path 对象是不被允许的。

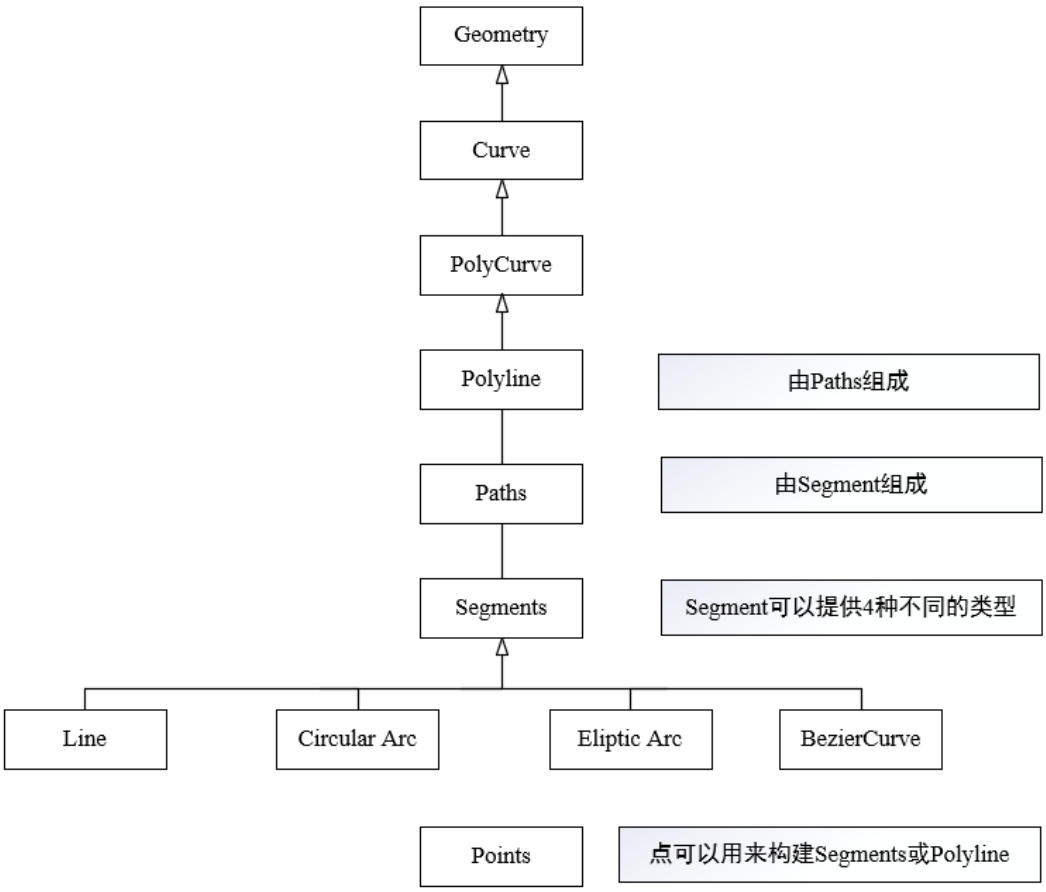

图 6-4 Polyline 及相关对象模型图

（10）多边形 Polygon 对象一个或多个环 Ring 对象组成的有序集合，它可以由单个 Ring 对象构成，也可以由多个 Ring 对象组成，甚至允许嵌套，形成岛（如图 6-5 所示）。Polygon 通常用来代表有面积的多边形矢量对象，如行政区、建筑物等。Polygon 对象满足下列条件：（a）每一个构成的 Ring 都是有效的；（b）Ring 之间的边界不能重合；（c）外部环是有方向的，它是顺时针方向；（d）内部环在一个多边形中定义了一个洞，它是逆时针方向；（e）面积为 0 的 Ring 是不允许的；（f）多边形上存在一个 Segment 对象或路径对象是无效的。

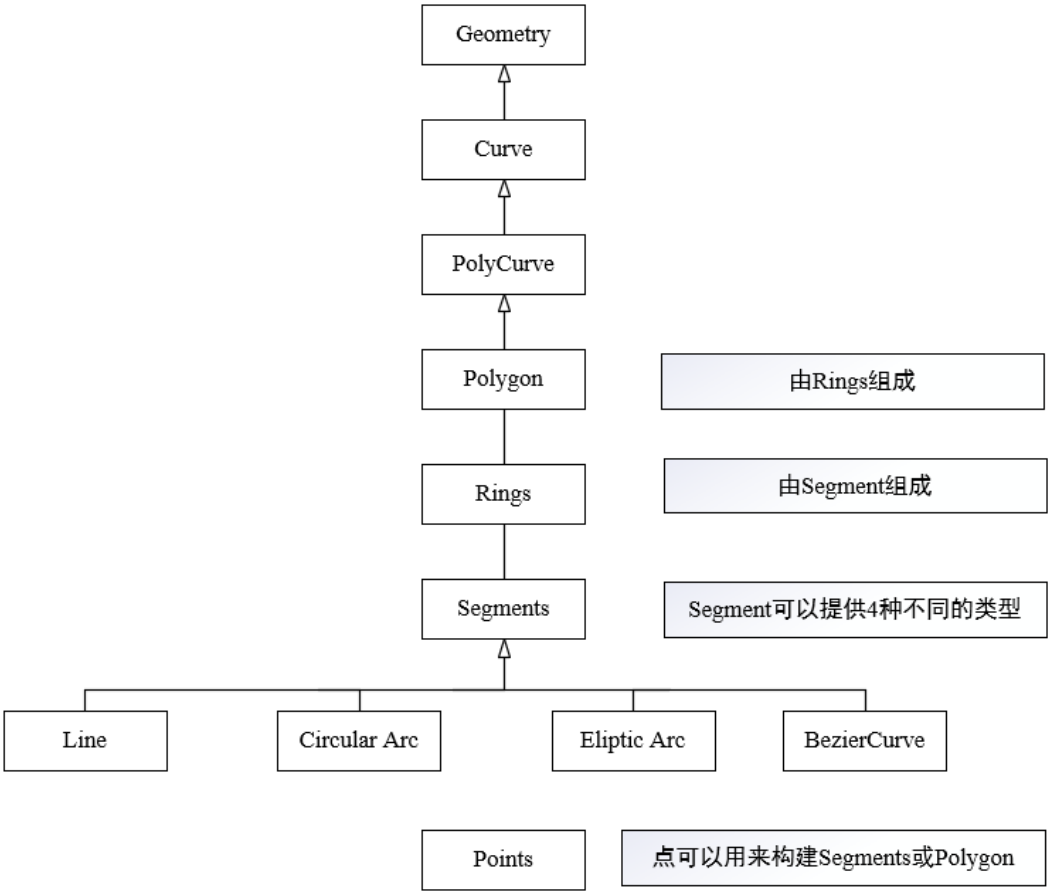

图 6-5 Polygon 及相关对象模型图

6. 1. 2 几何集合接口

几何 Geometry 模型中，几何形体对象被分为高级几何对象和构件几何对象两个层次。高级几何对象是构成元素、要素形状的几何图形，主要有点 Point、多点 MultiPoint、包络线 Envelope、多义线 Polyline 和多边形 Polygon。构件几何对象是用于组合构成高级几何对象（如表 6-1 所示），如 Segment 对象构成了 Path，Path 对象构成了 Polyline。但这并不意味着必须有“层次”地建立一个高级几何对象，实际上 Point 对象可以构成所有的几何形体。

表 6-1 高级几何对象与构件几何对象间的对应关系

几何对象	构件对象	用于创建和编辑此形状的接口
Path	Segment	ISegmentCollection
Ring	Segment	ISegmentCollection
MultiPatch	TriangleFan、TriangleStrip、Ring	IGeometryCollection
Multipoint	Point	IPointCollection、IGeometryCollection
Polyline	Path	IGeometryCollection
Polygon	Ring	IGeometryCollection
TriangleFan	Point	IPointCollection、IGeometryCollection
TriangleStrip	Point	IPointCollection、IGeometryCollection

除了点 Point 对象外，其他几何形体对象都可以通过集合的方式构成，如多点 MultiPoint 对象是点 Point 的集合，路径 Path 是段 Segment 的集合，多义线 Polyline 是路径 Path 的集合。AO 中几何对象所实现的几何集合接口有三个：IGeometryCollection、ISegmentCollection、IPointCollection。

实现 IGeometryCollection 接口的几何对象及相应的构件几何对象包括：Polygon—Ring，Polyline—Path，MultiPoint—Point，GeometryBag—任何类型几何对象，MultiPatch—Triangle、TriangleStrip 和 TriangleFan 等三维几何对象。用 IGeometryCollection 接口构建多义线 Polyline 对象，需要注意的情况有：（1）每一个路径对象都必须是有效的；（2）添加路径时必须注意顺序和方向；（3）为了保证 Polyline 的有效性，可以在产生这个形状后使用 Simplify 方法。用 IGeometryCollection 接口构建多边形 Polygon 对象，需要注意以下几点：（1）每一个组成多边形的环都是有效的；（2）产生一个多边形后，可以使用 Simplify 和 SimplifyPreserveToFrom 来保证内环和外环的方向的正确，环没有相交或自交，而且环是封闭的；（3）还可以用 IPointCollection 的方法来检查。

实现 ISegmentCollection 接口的几何形体对象有路径 Path、环 Ring、多义线 Polyline 和多边形 Polygon。

实现 IPointCollection 接口的几何形体对象包括多点 Multipoint、路径 Path、环 Ring、多义线 Polyline、多边形 Polygon、三角形扇 TriangleFan、三角形带 TriangleStrip、多面片 MultiPatch。

6. 2 演示实例

本例实现以鼠标点为圆心，按照一定角度和距离创建一系列点元素。具体步骤如下：

（1）在 Visual Studio 中新建 MapControl Application 项目，选择【Visual C#】→【ArcGIS】→【Extending ArcObjects】→【MapControl Application】模板，名称为“MapControlApplication ”，解决方案为“Test6”，如图 6-6 所示。

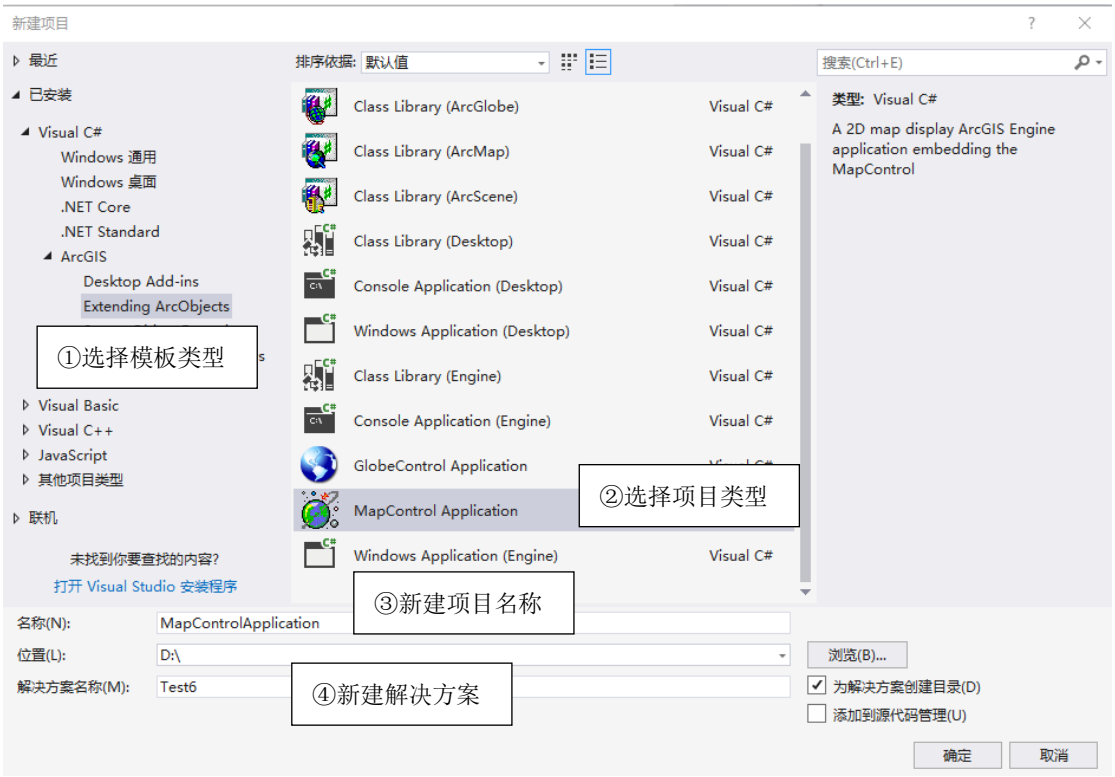

图 6-6 新建 MapControl Application 项目

（2）解决方案资源管理器选中“MapControlApplication”项目，右键选择【添加】→【新建项】，新建一个工具类“ToolAddPointEle”，如图 6-7 所示。

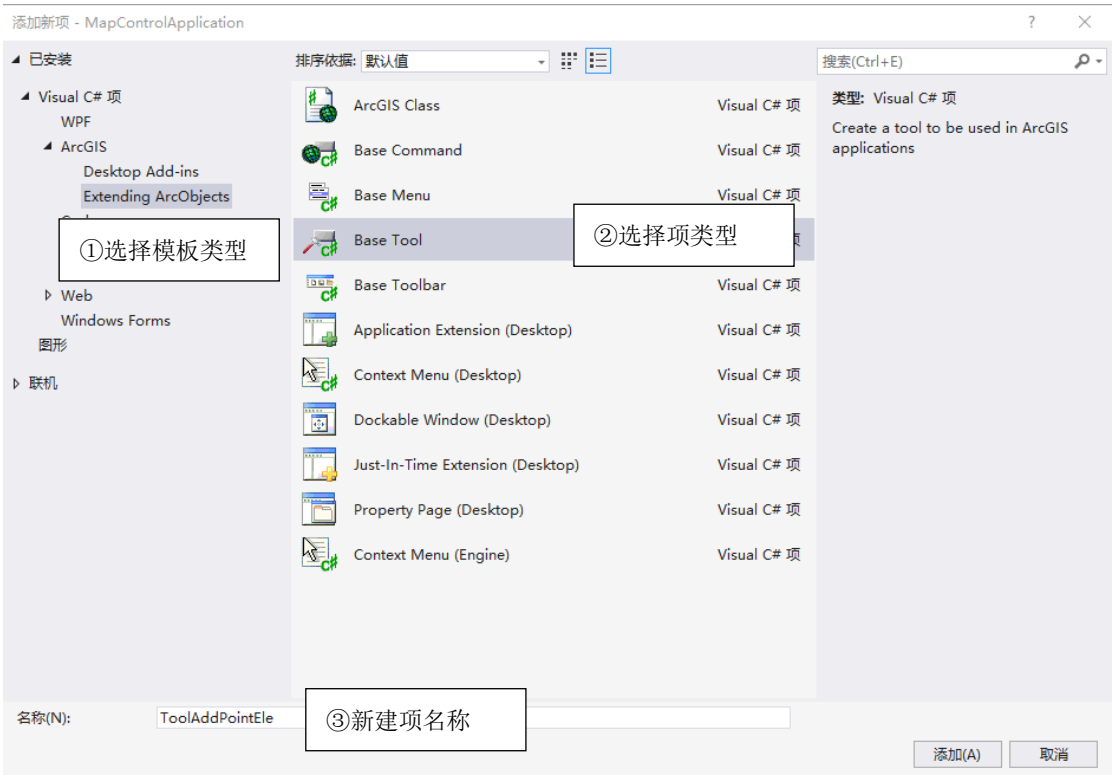

图 6-7 新建工具类

（3）在 ToolAddPointEle.cs 中添加引用：

```
using ESRI. ArcGIS. Geometry;
```

```
using ESRI.ArcGIS.Carto;
```

然后将 Geometry 的嵌入互操作类型改为 False。

并修改鼠标按键弹起事件响应函数 OnMouseUp 的代码如下：

ToolAddPointEle.cs（节选）	功能：按照一定角度和距离创建一系列点元素

```
public override void OnMouseUp(int Button, int Shift, int X, int Y)
{
    // TODO: Add ToolAddPointEle.OnMouseUp implementation
    IPoint pt = m_hookHelper.ActiveView.ScreenDisplay.
                                DisplayTransformation.ToMapPoint(X, Y);
    IElement marker = new MarkerElementClass();
    marker.Geometry = pt;
    //按照鼠标位置创建点元素
    m_hookHelper.ActiveView.GraphicsContainer.AddElement(marker, 0);
    //鼠标按键弹起创建6个点元素
    for (int j = 0; j < 6; j++)
    {
        IConstructPoint construct = new PointClass();
        //设置创建点与起始点的距离和相对角度
        construct.ConstructAngleDistance(pt, j * 2 * 3.14 / 6, 50);
        IElement mark = new MarkerElementClass();
        mark.Geometry = (IPoint)construct;
        m_hookHelper.ActiveView.GraphicsContainer.AddElement(mark, 0);
    }
    m_hookHelper.ActiveView.PartialRefresh(
                        esriViewDrawPhase.esriViewGraphics, null, null);
}
```

点 Point 对象实现的 IConstructPoint 接口可以使用多达 10 种方法来创建所需要的点：ConstructAlong 沿线创建法、ConstructAngleBisector 角平分线创建法、ConstructAngleIntersection 构造角度交点、ConstructAngleDistance 构造角度距离点（如图 6-8 所示）、ConstructDeflection 构造偏转角度点、ConstructDeflectionIntersection 构造偏移角交点、ConstructOffset 构造偏移点、ConstructParallel 构造平行线上点、ConstructPerpendicular 构造垂直线上点、ConstructThreePointResection 后方交会定点。

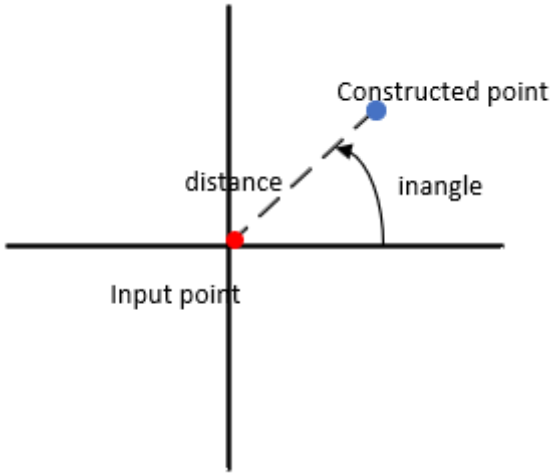

图 6-8 IConstructPoint 接口的 ConstructAngleDistance 定点法

多点 Multipoint 对象实现的 IConstructMultipoint 接口也有众多构造方法，包括：ConstructArcPoints 构造圆弧点、ConstructDivideLength 构造等长度点、ConstructDivideEqueal 构造等分点、ConstructIntersection 构造交点、ConstructTangent 构造切线点。

线段 Line 对象实现的 IConstructLine 接口提供的构建 Line 对象的方法包括：（a）ConstructAngleBiseetor 通过三个点对象构造一个夹角，然后通过这个夹角的顶点产生一个角平分线，再传入线段的长度；（b）ConstructExtended 扩展一个存在线段对象产生一个新的线段。

圆弧 CircularArc 对象实现的 IConstructCircularArc 接口提供了数目高达 35 种的构造器方法来产生一个 CircularArc 对象，包括：（a）ConstructCircle，传入圆心 CenterPoint 和半径 Radius 两个属性，产生一个圆对象；（b）ConstructArcDistance，传入圆心点、起始点和圆弧长度来产生一个圆弧对象；（c）ConstructChordDistance，基于一个起始点、圆弧的弧长、圆弧的方向和中心点产生一个圆弧对象；（d）ConstructEndPointsChordHeight，传入起始点、终止点和弦的中心高度，按顺/逆时针产生一个圆弧对象；（e）ConstructFilletPoint，产生两条线段或圆弧的内切弧；（f）ConstructTangentAndPoint，相切于一个 Segment 对象某点的圆弧；（g）ConstructThreePoints，通过三个给定点产生一个圆弧，等等。

其它各类几何对象也都实现了各自的 IConstructX 方法，用来产生一个对应类别的 X 几何对象。

（4）在 MainForm 窗体中添加菜单项“AddPointEle”，如图 6-9 所示。

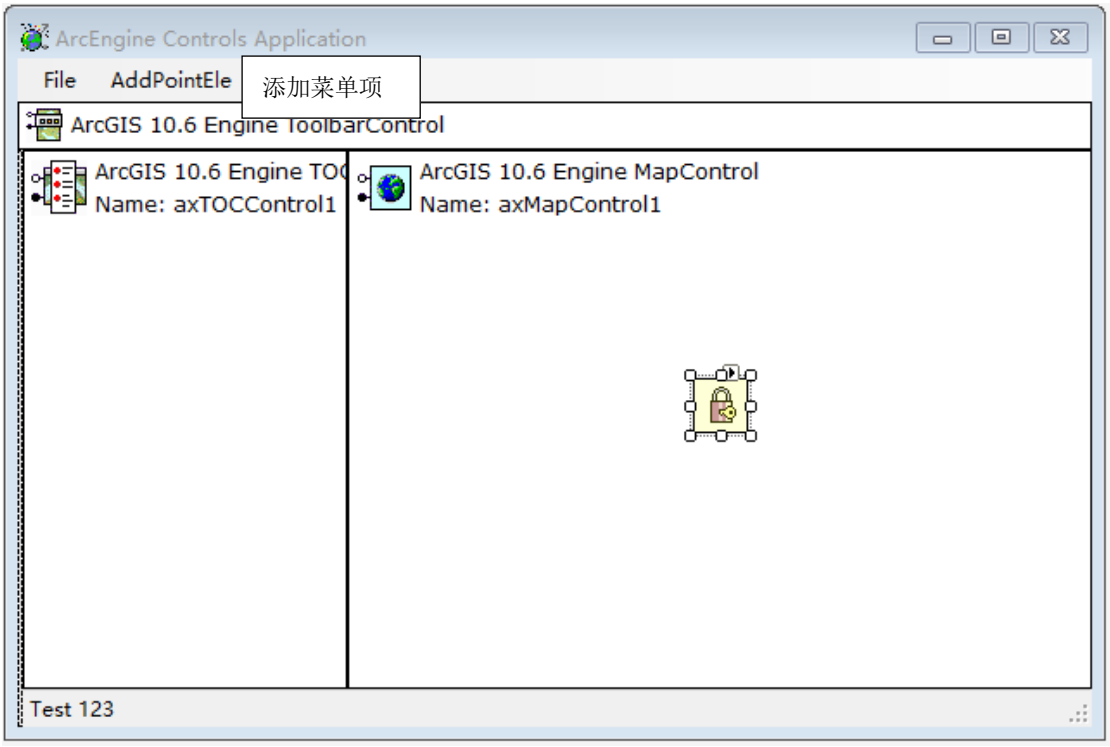

图 6-9 添加菜单项

双击该菜单项添加 Click 事件响应函数代码如下所示。

MainForm. cs（节选）	功能：添加菜单单击事件响应函数

```
//调用Tool工具创建一系列点
private void addPointEleToolStripMenuItem_Click(object sender, EventArgs e)
{
    ICommand command = new ToolAddPointEle();
    command. OnCreate(m_mapControl. Object);
    axMapControl1. CurrentTool = (ITool)command;
}
```

（5）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】，点击“AddPointEle”，利用鼠标点击创建一系列点元素，效果如图 6-10 所示。

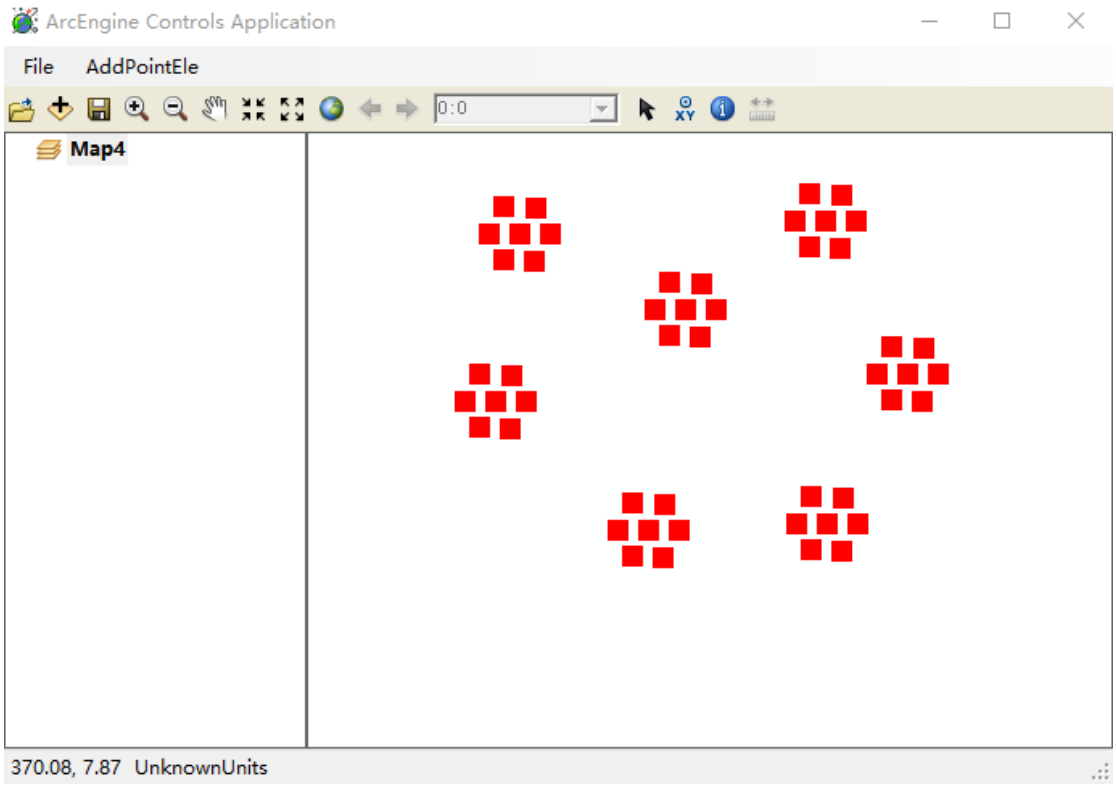

图 6-10 创建一系列点元素运行效果

6.3 实验目的

（1）熟悉几何对象的结构及其操纵方法；
（2）熟悉文本文件读取与解析的相关方法。

6.4 实验内容

（1）实现一个创建多边形要素类的按钮命令 Command，并将多边形要素类添加到当前地图中；
（2）实现一个工具 Tool 交互新建多边形要素，再实现一个工具 Tool 来交互移除多边形要素的顶点；
（3）读取文本文件并将其转换为点和多边形要素类。

6.5 实验数据

将以下文字内容保存为文本文件格式，重命名为“data.txt”。

点号	X	Y
1	510000	201000
2	520000	201000

3	520000	202000
4	510000	202000

6. 6 实验步骤

6. 6. 1 新建多边形要素类

本例实现创建一个多边形要素类 Shapefile 文件到本地目录，并添加到当前地图显示，具体步骤如下：

（1）新建按钮命令类，实现新建多边形要素类的功能。解决方案资源管理器下选择“MapControlApplication ”项目，右键选择【添加】→【新建项】，新建一个按钮类“CmdCreateFeatureClass”，如图 6-11 所示。

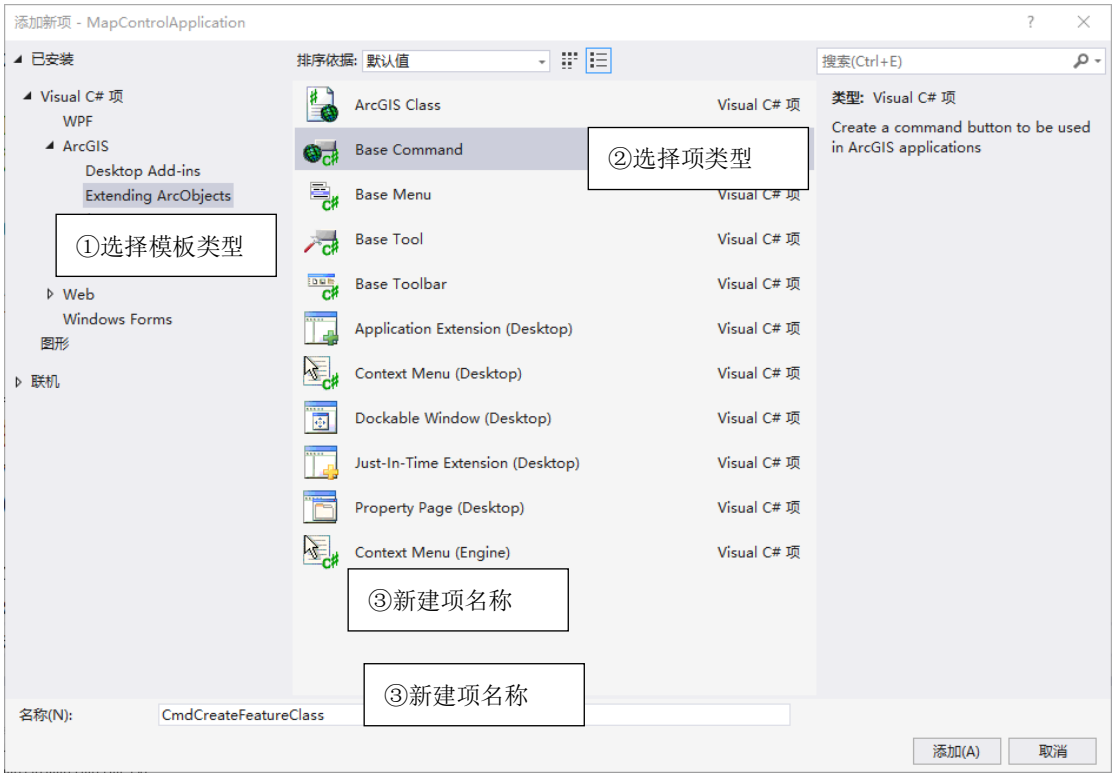

图 6-11 新建按钮命令类

（2）在 CmdCreateFeatureClass.cs 中添加引用：

```
using System. Windows. Forms;
using ESRI. ArcGIS. Carto;
using ESRI. ArcGIS. DataSourcesFile;
using ESRI. ArcGIS. Geodatabase;
using ESRI. ArcGIS. Geometry;
```

在 CmdCreateFeatureClass.cs 中添加按钮命令的 OnClick 事件响应函数代码，实现新建多边形要素类。

CmdCreateFeatureClass.cs（节选）	功能：创建多边形要素类

```
public override void OnClick()
{
    // TODO: Add CmdCreateFeatureClass.OnClick implementation
    //连接本地目录
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "（*.shp）|*.shp";
    saveFileDialog.Title = "新建多边形shp文件";
    saveFileDialog.CheckFileExists = false;
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        IWorkspaceFactory pWorkspaceFactory = new
                                              ShapefileWorkspaceFactory();
        string localFilePath = saveFileDialog.FileName.ToString();
        //获取文件名，不带路径
        string fileNameExt = localFilePath.Substring(
                                    localFilePath.LastIndexOf("\\") + 1);
        string fileName = fileNameExt.Substring(0,
                                    fileNameExt.LastIndexOf("."));
        //获得不带文件名的文件路径
        string FilePath = localFilePath.Substring(0,
                                    localFilePath.LastIndexOf("\\"));
        //打开Shapefile本地目录类型的地理数据库连接
        IWorkspace ipWorkspace = pWorkspaceFactory.OpenFromFile(
                                                    FilePath, 0);
        //创建Shapefile本地文件的要素类
        IFeatureClass fc = CreateShapeFeatureClass(
                        ipWorkspace as IFeatureWorkspace, filename,
                        esriGeometryType.esriGeometryPolygon);
        //添加新创建的多边形要素类到当前地图
        IFeatureLayer layer = new FeatureLayerClass();
        layer.Name = fc.AliasName;
        layer.FeatureClass = fc;
        m_hookHelper.FocusMap.AddLayer(layer as ILayer);
    }
}
```

（3）在 CmdCreateFeatureClass.cs 中添加私有成员函数 CreateShapeFeatureClass ()创建 Shapefile 要素类、添加私有成员函数 CreateSpatialReference()创建空间参考。

CmdCreateFeatureClass.cs（节选）	功能：创建多边形要素类及其空间参考

```
//创建点、线、面要素类
private IFeatureClass CreateShapeFeatureClass(IFeatureWorkspace ws,
                                    string FCName, esriGeometryType type)
{
    //设置字段组
```

```
        IFieldsEdit ipFields = (IFieldsEdit)new Fields();
        ipFields.FieldCount_2 = 2; //设置字段数
        IFieldEdit ipField = (IFieldEdit)new Field();
        ipField.Name_2 = "ObjectID";
        ipField.AliasName_2 = "FID";
        ipField.Type_2 = esriFieldType.esriFieldTypeOID;
        ipFields.set_Field(0, ipField);
        // 设置几何形状字段
        IGeometryDefEdit ipGeoDef = (IGeometryDefEdit)new GeometryDef();
        ISpatialReference ipSR = CreateSpatialReference();
        ipGeoDef.SpatialReference_2 = ipSR; //设置空间索引
        // 设置要素几何类型
        ipGeoDef.GeometryType_2 = type;
        IFieldEdit ipField3 = (IFieldEdit)new Field();
        ipField3.Name_2 = "Shape";
        ipField3.AliasName_2 = "shape";
        ipField3.Type_2 = esriFieldType.esriFieldTypeGeometry;
        ipField3.GeometryDef_2 = ipGeoDef;
        ipFields.set_Field(1, ipField3);
        // 在工作空间下创建要素类
        IFeatureClass ipFeatCls = ws.CreateFeatureClass(FCName, ipFields, null,
                                    null, esriFeatureType.esriFTSimple, "Shape", "");
        return ipFeatCls;
}
//创建空间参考
private ISpatialReference CreateSpatialReference()
{
        ISpatialReferenceFactory pSRF = new SpatialReferenceEnvironmentClass();
        ISpatialReference pSpatialReference = pSRF.CreateProjectedCoordinateSystem(
                        (int)esriSRProjCSType.esriSRProjCS_WGS1984UTM_21N);
        return pSpatialReference;
}
```

地理数据库中新建要素集或要素类，都要设置它们的空间参考，包括两方面：（a）坐标系统，定义了空间数据在地球上的位置；（b）精度，定义了存储在地理数据库中的数据细节程度。ArcObjects 中使用的两种坐标系统为：（a）地理坐标系统，以经纬度为地图存储单位；（b）投影坐标系统，是将三维地理坐标系统上的经纬网投影到二维平面地图上使用的坐标系统（等角投影、等积投影、正形投影等），地图单位通常为米。空间参考 SpatialReference 对象模型中的三个组件类 GeographicCoordinateSystem 、ProjectedCoordinateSystem 和 UnknownCoordinateSystem 都实现了 ISpatialReference 接口。

（4）添加菜单项“createFeatureClass”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：菜单项响应事件创建要素类

```
//调用Command命令创建多边形要素
```

```
private void createFeatureClassToolStripMenuItem_Click(object sender,
                                                        EventArgs e)
{
    ICommand command = new CmdCreateFeatureClass();
    command.OnCreate(m_mapControl.Object);
    command.OnClick();
}
```

（5）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】，点击菜单“CreateFeatureClass”，自定义需要创建的多边形要素类名称，点击“保存”，生成一个多边形的shp图层并添加到当前地图，如图6-12所示。

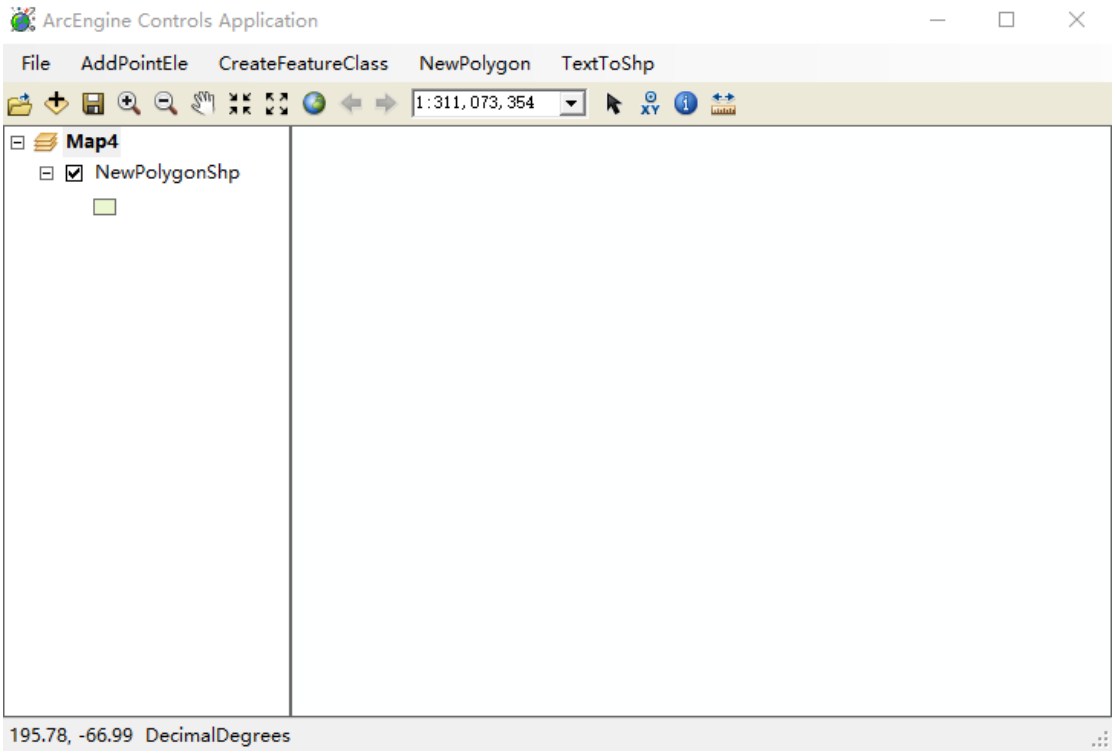

图6-12 创建的多边形要素shp图层

6. 6. 2 新建多边形要素

（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“ToolNewPolygon”，如图 6-13 所示。

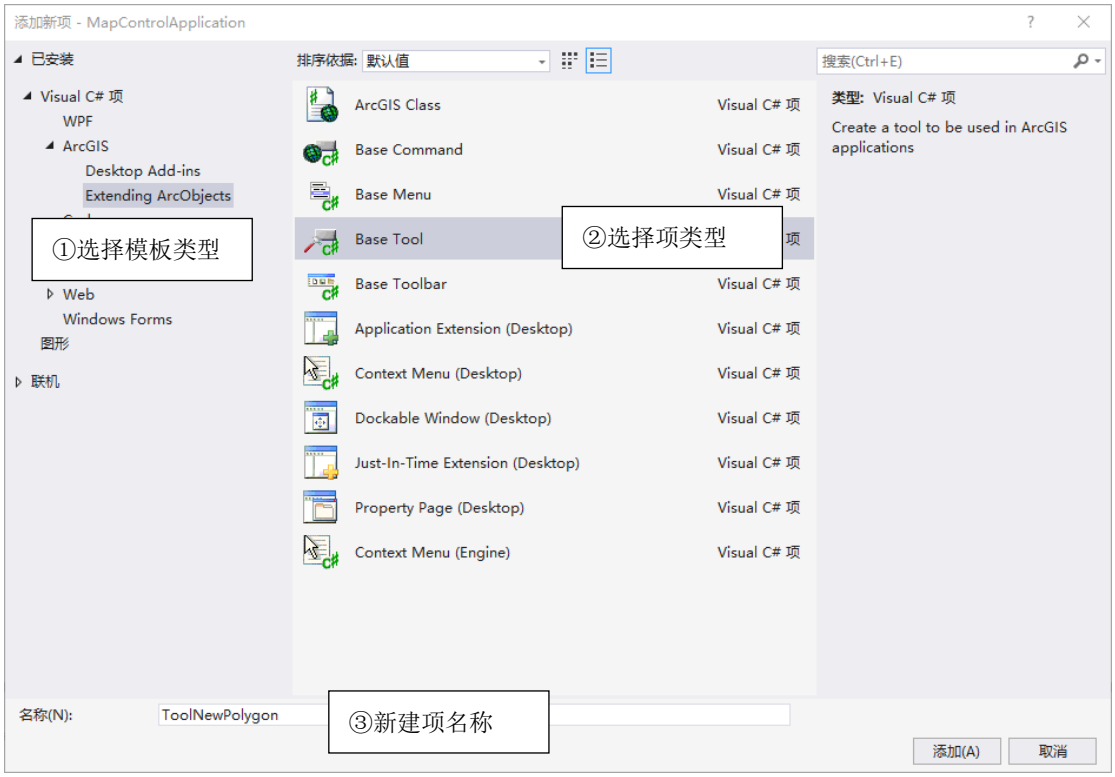

图 6-13 添加工具类

（2）首先在 ToolNewPolygon.cs 中添加引用：

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

然后定义一个局部变量 m_selectedLyr 保存当前选择的图层，

```
private IFeatureLayer m_selectedLyr = null;
```

再修改鼠标按下事件响应函数 OnMouseDown()的代码如下：

ToolSelectLayer.cs（节选）	功能：实现鼠标按下创建要素

```
//利用鼠标按下创建要素
public override void OnMouseDown(int Button, int Shift, int X, int Y)
{
    // TODO: Add ToolSelectLayer.OnMouseDown implementation
    IRubberBand ipRubber = new RubberPolygon();
    IGeometry ipGeo = null;
    ipGeo = ipRubber.TrackNew(
                    m_hookHelper.ActiveView.ScreenDisplay, null);
    //如果选中图层不为空，则将创建的要素保存至该图层
    if (m_selectedLyr != null)
    {
        IFeature feature = m_selectedLyr.FeatureClass.CreateFeature();
        feature.Shape = ipGeo;
        feature.Store();
        m_hookHelper.ActiveView.Refresh();
```

```
        }
    }
```

（3）在 ToolNewPolygon.cs 中添加鼠标按下事件响应函数 OnClick 添加代码。

ToolNewPolygon.cs（节选）	功能：选择图层用于保存创建的多边形要素

```
public override void OnClick()
{
    // TODO: Add ToolSelectLayer.OnClick implementation
    //调用FrmSelectLayer选择图层
    FrmSelectLayer dlg = new FrmSelectLayer(m_hookHelper);
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        m_selectedLyr = dlg.lyr as IFeatureLayer;
    }
}
```

（4）新建一个 Windows 窗体用来选择图层以供交互添加要素，窗体类名称为“FrmSelectLayer”，如图 6-14 所示。

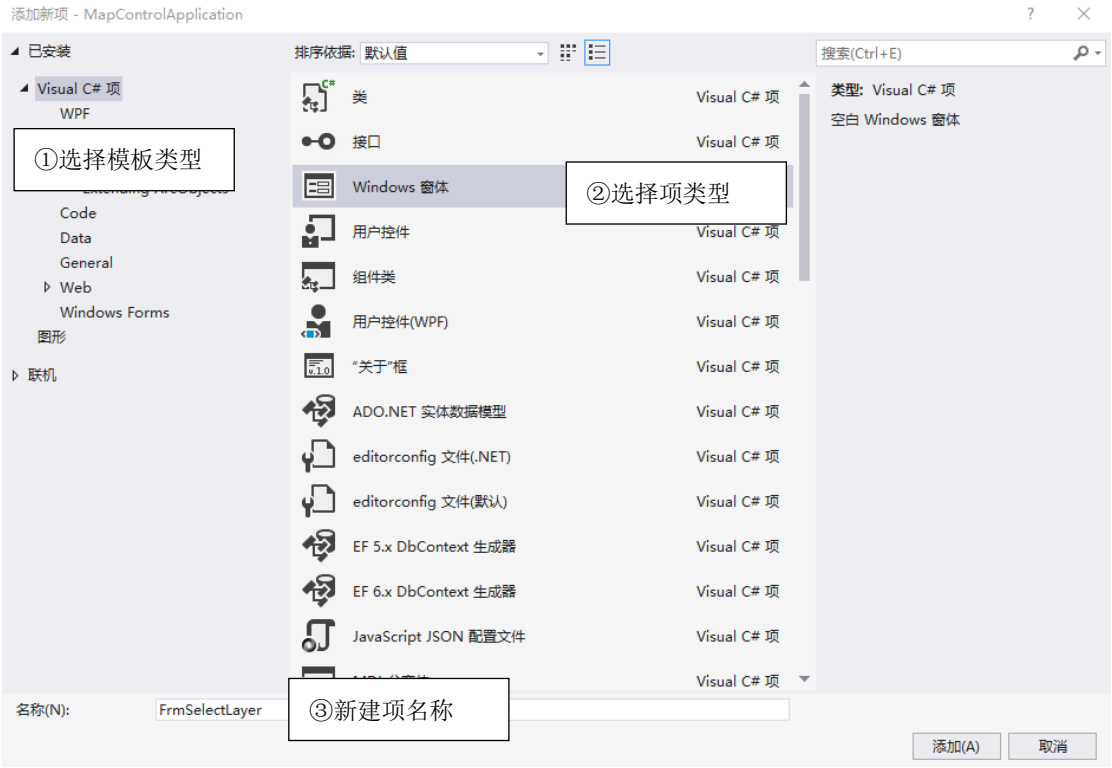

图 6-14 新建图层选择窗体

为窗体类“FrmSelectLayer”添加窗体控件并修改各控件的 Name 属性值，如图 6-15 所示。

图 6-15 修改各控件的 Name 属性值

（5）在 FrmSelectLayer.cs 中添加引用：

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

为窗体类“FrmSelectLayer”添加私有成员变量“ILayer m_layer”和只读属性“ILayer lyr”，用来保存和获取选择到的要素图层；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，代码如下：

FrmSelectLayer.cs（节选）	功能：类的属性及构造函数

```
//保存选择到的要素图层
private ILayer m_layer = null;
//定义IHookHelper接口获取主应用程序资源
private IHookHelper m_hookhelper = null;
public ILayer lyr
{
    get
    {
        return m_layer;
    }
}
public FrmSelectLayer(IHookHelper hook)
{
    m_hookhelper = hook;
    InitializeComponent();
}
```

（6）点击窗体 FrmSelectLayer 空白处添加 FrmSelectLayer_Load()窗体加载事件响应函数，然后添加 CbxLayersAddItems()私有成员函数，添加当前所有图层。

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

```
private void FrmSelectLayer_Load(object sender, EventArgs e)
{
    CbxLayersAddItems();
}
//获取当前地图所有的多边形要素图层，并添加到组合框
private void CbxLayersAddItems()
{
```

```
        IEnumLayer layers = m_hookhelper.FocusMap.Layers;
        layers.Reset();
        IFeatureLayer layer = layers.Next() as IFeatureLayer;
        while (layer != null)
        {
            //如果是多边形要素图层，添加到组合框控件
            if (layer is IFeatureLayer && layer.FeatureClass.ShapeType ==
                                        esriGeometryType.esriGeometryPolygon)
            {
                cbxLayersChoose.Items.Add(layer.Name);
            }
            layer = layers.Next() as IFeatureLayer;
        }
    }
```

（7）双击图层选择的 cbxLayersChoose 控件，添加图层选择事件响应函数代码；双击“确定”和“取消”按钮，添加按钮单击事件响应函数。

FrmSelectLayer.cs（节选）	功能：获得选中的要素图层并进行创建要素

```
//获得当前选择的多边形要素图层
private void cbxLayersChoose_SelectedIndexChanged(object sender, EventArgs e)
{
    IEnumLayer layers = m_hookhelper.FocusMap.Layers;
    layers.Reset();
    ILayer layer = layers.Next();
    while (layer != null)
    {
        if (layer is IFeatureLayer)
        {
            if (layer.Name == cbxLayersChoose.Text)
            {
                m_layer = layer;
                break;
            }
        }
        layer = layers.Next();
    }
}
private void buttonOk_Click(object sender, EventArgs e)
{
    DialogResult = DialogResult.OK;
    this.Close();
}
//关闭窗体
private void buttonCancel_Click(object sender, EventArgs e)
{
```

```
        m_layer = null;
        this.Close();
    }
```

（8）添加菜单项“NewPolygon”，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：调用创建多边形要素工具

```
//调用Tool工具创建要素类
private void newPolygonToolStripMenuItem_Click(object sender, EventArgs e)
{
    ICommand command = new ToolNewPolygon();
    command.OnCreate(m_mapControl.Object);
    axMapControl1.CurrentTool = (ITool)command;
}
```

（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】，点击“NewPolygon”弹出 FrmSelectLayer 窗体，选择需要创建要素的图层，用鼠标依次创建多个多边形要素，如图 6-16 所示。

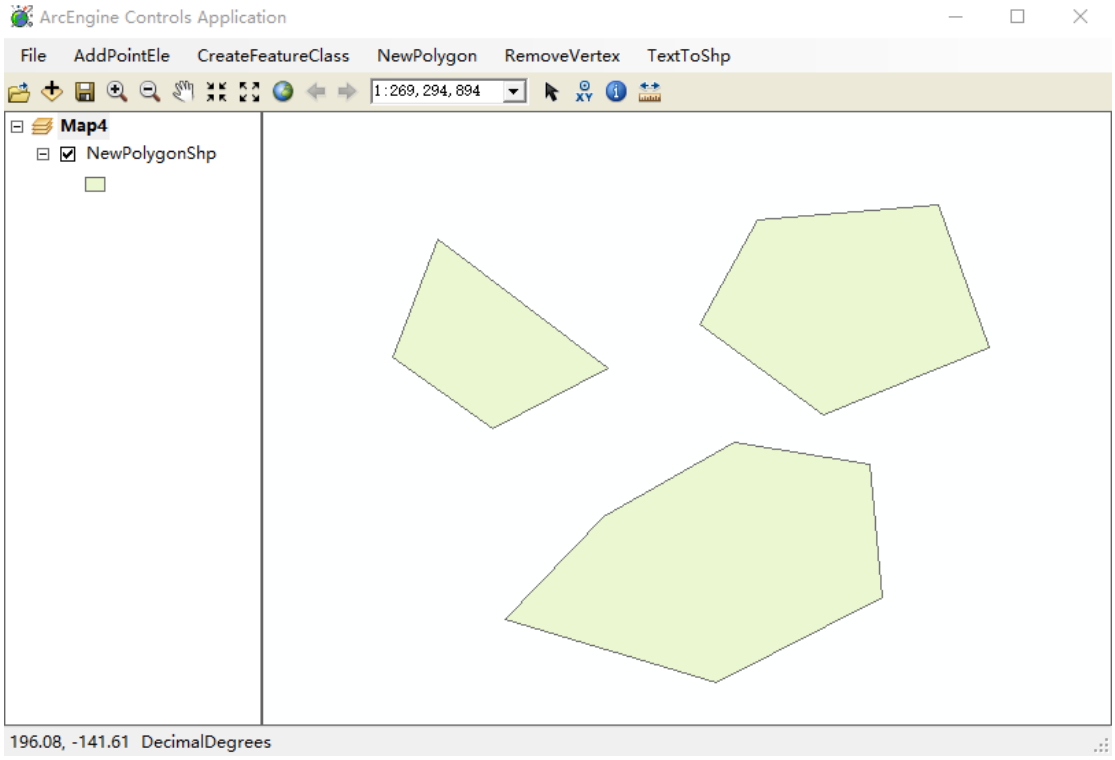

图 6-16 创建多边形要素功能实现效果

6. 6. 3 移除多边形的顶点

本例是为一个多边形 Polygon 要素删除一个顶点，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“ToolRomoveVertex”，如图 6-17 所示。

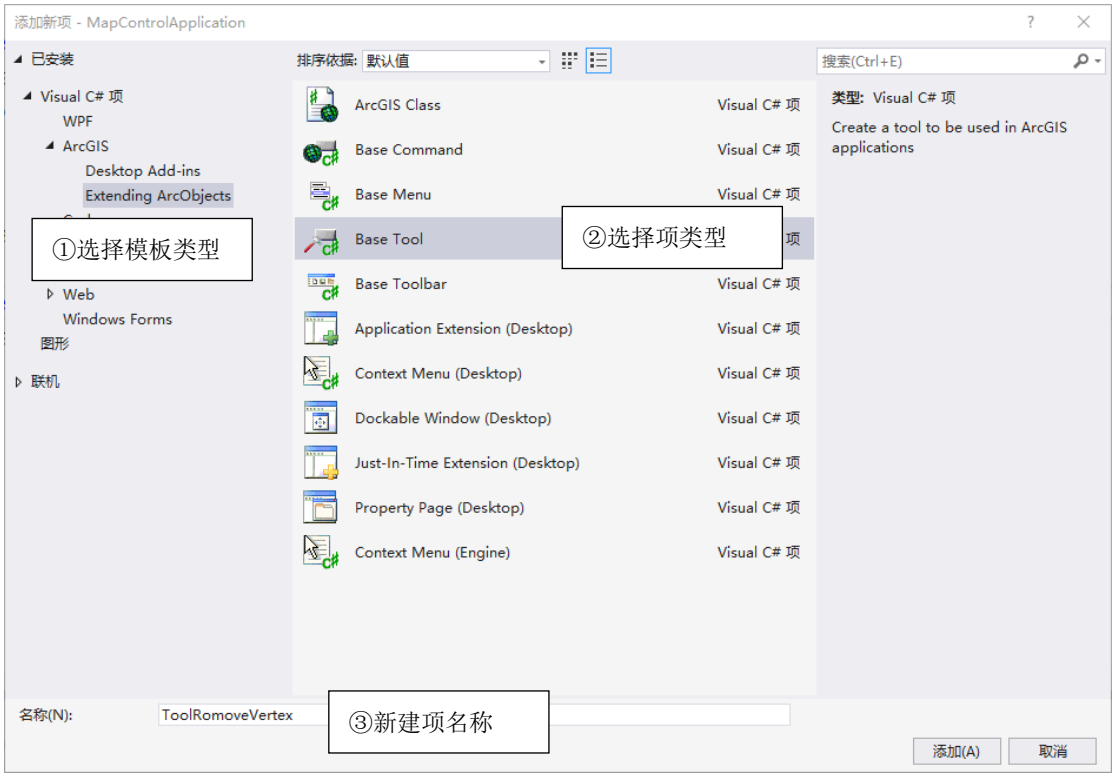

图 6-17 添加工具 Tool 类

（2）在 ToolRomoveVertex.cs 文件添加引用：

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

为工具类“ToolRomoveVertex”定义私有成员变量 m_layer 和 m_pgFeature，

```
private ILayer m_layer = null;
private IFeature m_pgFeature = null;
```

然后添加按钮单击事件 OnClick()响应函数代码。

ToolRomoveVertex.cs（节选）	功能：选择多边形要素图层

```
//调用 FrmSelectLayer 窗体进行图层选择
public override void OnClick()
{
    // TODO: Add ToolRomoveVertex.OnClick implementation
    FrmSelectLayer dlg = new FrmSelectLayer(m_hookHelper);
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        m_layer = dlg.lyr as IFeatureLayer;
    }
}
```

（3）在 ToolRomoveVertex.cs 添加私有成员函数 RemoveVertex ()，删除多边形的顶点；添加鼠标按下事件 OnMouseDown()的响应函数代码。

ToolRomoveVertex.cs（节选）	功能：删除多边形顶点

```
public override void OnMouseDown(int Button, int Shift, int X, int Y)
```

```
{
    // TODO:  Add ToolRomoveVertex.OnMouseDown implementation
    if (m_layer == null)
        return;
    if (m_pgFeature == null)//先拉框选择多边形
    {
        ISpatialFilter ipSF = new SpatialFilterClass();
        IRubberBand ipRB = new RubberEnvelopeClass();
        IEnvelope env = ipRB.TrackNew(
                m_hookHelper.ActiveView.ScreenDisplay, null) as IEnvelope;
        ipSF.Geometry = env;
        ipSF.SpatialRel = esriSpatialRelEnum.esriSpatialRelIntersects;
        ((IFeatureSelection)m_layer).SelectFeatures(ipSF,
                esriSelectionResultEnum.esriSelectionResultNew, true);
        m_hookHelper.ActiveView.PartialRefresh(
                esriViewDrawPhase.esriViewGeoSelection, null, null);
        ISelection ipSel = m_hookHelper.FocusMap.FeatureSelection;
        if (ipSel == null)
            return;
        ((IEnumFeature)ipSel).Reset();
        //获得当前选择多边形
        m_pgFeature = (IFeature)((IEnumFeature)ipSel).Next();
    }
    else//选择并删除多边形顶点
    {
        IPoint ipPoint = m_hookHelper.ActiveView.ScreenDisplay.
                    DisplayTransformation.ToMapPoint(X, Y);
        IGeometry ipGeo = m_pgFeature.Shape;
        IElement ele = new PolygonElementClass();
        ele.Geometry = ipGeo;
        IPolygon ipPolygon = (IPolygon)ele.Geometry;
        IHitTest ipHitTest = (IHitTest)ipPolygon;
        double hitDist = 1.0E12;
        double searchRadius = ipGeo.Envelope.Width/4;
        int partIndex = -1, segIndex = -1;
        bool bHit = false, bRSide = false;
        IPoint hitPoint = new PointClass() ;
        bHit = ipHitTest.HitTest(ipPoint, searchRadius,
            esriGeometryHitPartType.esriGeometryPartVertex,
            hitPoint, ref hitDist, ref partIndex, ref segIndex, ref bRSide);
        if (bHit)
        {
            RemoveVertex(ipPolygon, hitPoint, partIndex, segIndex);
            m_pgFeature.Shape = ipPolygon;
```

```
                m_pgFeature.Store();
            }
            ((IFeatureSelection)m_layer).Clear();
            m_hookHelper.ActiveView.Refresh();
            m_pgFeature = null;
        }
}
//删除多边形的节点
private void RemoveVertex(IPolygon ipPolygon, IPoint hitPoint,
                                        int hitPartIndex, int hitSegmentIndex)
{
        if (ipPolygon == null) return;
        IPointCollection ipPointCollection = null;
        ipPointCollection = ((IGeometryCollection)ipPolygon).get_Geometry(
                                                hitPartIndex) as IPointCollection;
        if (ipPointCollection.PointCount <= 3)
            return;
        IRelationalOperator iRO = hitPoint as IRelationalOperator;
        //与段起点重合
        if (iRO.Equals(ipPointCollection.get_Point(hitSegmentIndex)))
        {
            ipPointCollection.RemovePoints(hitSegmentIndex, 1);
        }
        //与段终点重合
        else if(iRO.Equals(ipPointCollection.get_Point(hitSegmentIndex+1)))
        {
            ipPointCollection.RemovePoints(hitSegmentIndex + 1, 1);
        }
}
```

代码中接口 IHitTest 的方法 HitTest(QueryPoint, earchRadius, eometryPart, itPoint, itDistance, itPartIndex, hitSegmentIndex, bRightSide)，用来判断要删除的点是否在图形设定的偏差范围内，如果是就将在该位置的顶点删除。其中 QueryPoint 是被用来查询的点，hitPoint 返回被点击图形中离查询点最近的一个点。

删除多边形的顶点用到接口 IPointCollection 中的方法 RemovePoints (Index, Count)，从指定的位置起移除指定个数的点。

（4）为主窗体“MainForm”添加菜单项“RemoveVertex”，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：添加菜单单击事件响应函数

```
//调用Tool工具删除多边形要素节点
private void romoveVertexToolStripMenuItem_Click(object sender, EventArgs e)
{
    ICommand command = new ToolRomoveVertex();
    command. OnCreate(m_mapControl. Object);
```

```
        axMapControl1. CurrentTool = (ITool)command;
}
```

（5）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】，点击“RemoveVertex”，拉框选中多边形要素，如图 6-18 所示。

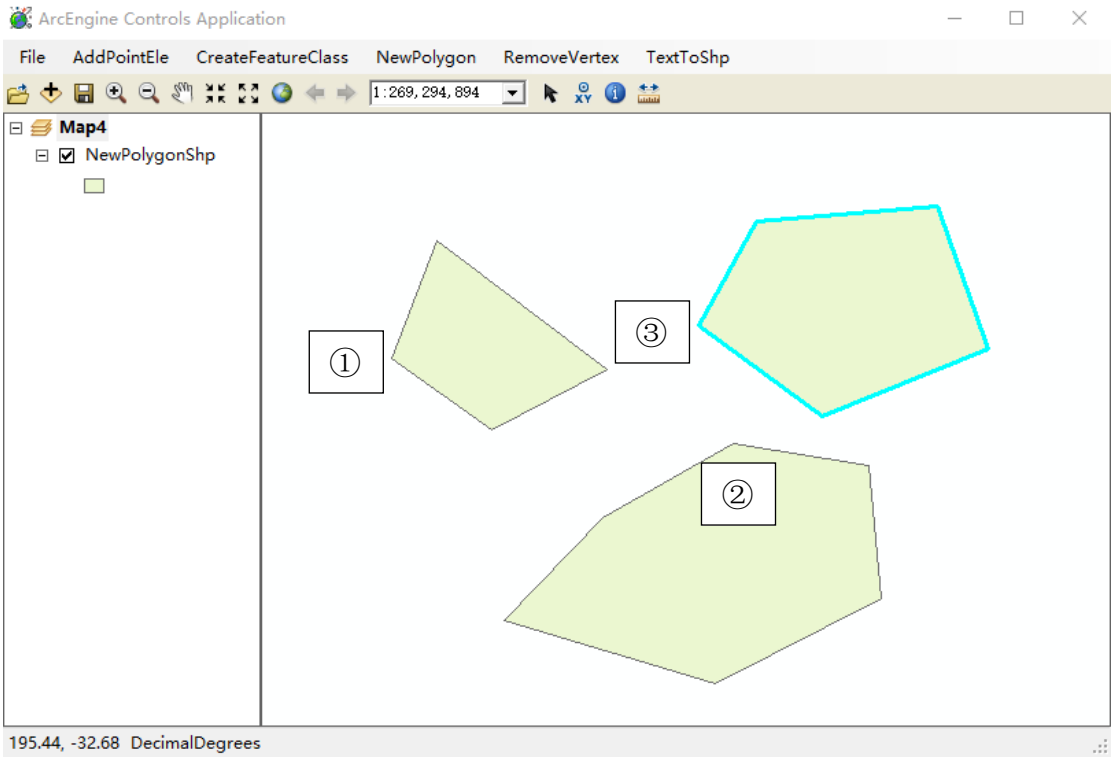

图 6-18 拉框选中多边形要素

分别选择多边形要素，并鼠标单击需要删除的多边形顶点①、②、③，多边形顶点被删除，如图 6-19 所示。

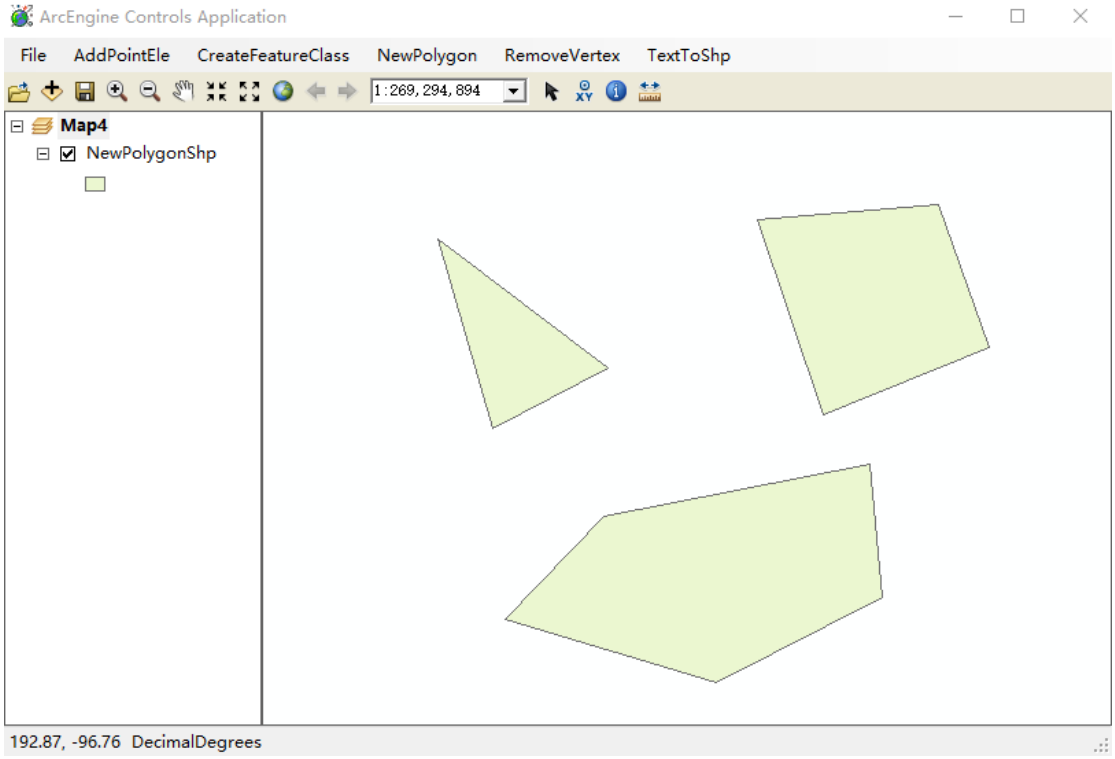

图 6-19 删除多边形节点运行效果

6. 6. 4 读取文本文件创建要素

本例实现的是读取含有记录测量点坐标的文本文件，生成点要素，存入点要素类并显示在地图上；再利用点坐标的构建一个多边形要素，存入多边形要素类并显示在地图上。具体步骤如下：

（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 窗体用来读取记录测量点坐标的文本文件并将其转化为点要素和多边形要素的 shapefile 文件，名称为“FrmTxtToShp”，如图 6-20 所示。

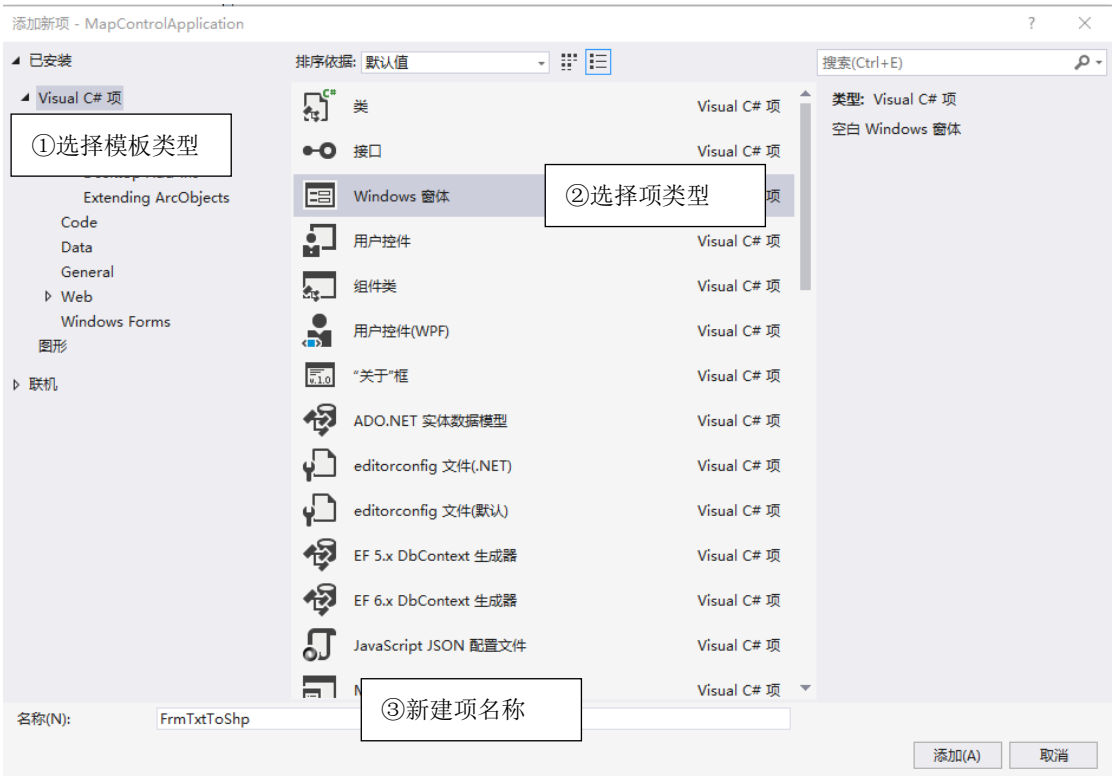

图 6-20 新建“文本转要素”窗体

添加控件并修改各控件的 Name 属性值，如图 6-21 所示。

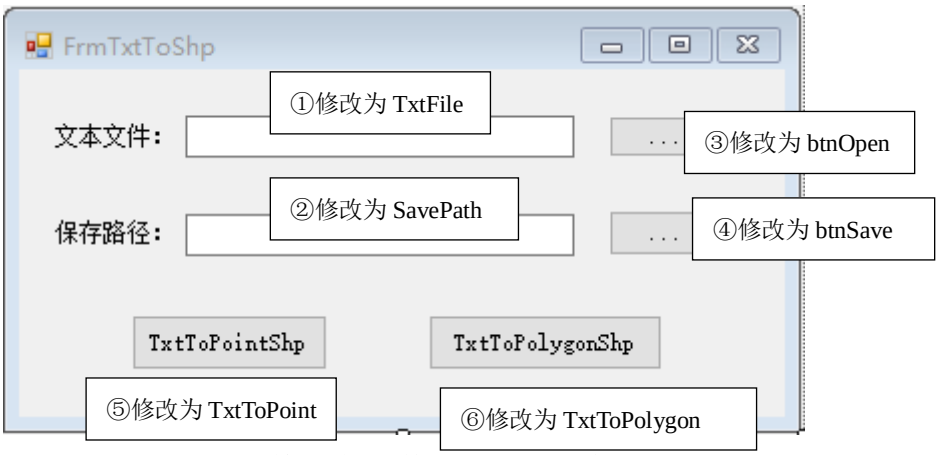

图 6-21 修改各控件的 Name 属性值

（2）在 FrmTxtToShp.cs 中添加引用：

```
using System.IO;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.DataSourcesFile;
using ESRI.ArcGIS.esriSystem;
```

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

为“FrmTxtToShp”添加私有成员变量，修改窗体类构造函数，添加私有成员函数 GetPoints()读取文本文件转换为点集，代码如下：

FrmTxtToShp.cs（节选）	功能：读取文本文件

```
private IPointCollection m_pts = null;
private IHookHelper m_hookHelper = null;
public FrmTxtToShp(object hook)
{
    InitializeComponent();
    if (m_hookHelper == null)
        m_hookHelper = new HookHelperClass();
    m_hookHelper.Hook = hook;
}
//读取文本文件转换为点集
private IPointCollection GetPoints(string txtFileName)
{
    try
    {
        IMultipoint pts = new MultipointClass();
        IPointCollection ptsCol = pts as IPointCollection;
        //常用的分隔符为逗号、空格、制位符
        char[] datachar = new char[] { ',', ' ', '\t' };
        //定义文件流
        System.IO.FileStream FileStream =
                    new System.IO.FileStream(txtFileName, FileMode.Open);
        StreamReader StreamReader = new StreamReader(
                    FileStream, Encoding.Default);//读取文件流
        //以行为单位进行读取，跳过一行文件头
        string stringLine = StreamReader.ReadLine();
        while ((stringLine = StreamReader.ReadLine()) != null) //读取点信息
        {
            string[] strArray = stringLine.Split(datachar);//以上述分隔符分割
            IPoint point = new PointClass();
            //将strArray赋给Point
            point.M = Convert.ToDouble(strArray[0].Trim());
            point.X = Convert.ToDouble(strArray[1]);
            point.Y = Convert.ToDouble(strArray[2]);
            ptsCol.AddPoint(point);
        }
        StreamReader.Close();
        return ptsCol;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return null;
    }
}
```

（3）双击“btnOpen”按钮，添加代码用于打开“.txt”文件，并将路径显示在左边的控件“TxtFile”中；双击“btnSave”按钮，添加代码用于保存生成shp文件，并将保存路径和自定义的shp要素类名称显示在左边的控件“SavePath”中。

FrmTxtToShp.cs（节选）	功能：打开txt文件和选择shp文件保存路径

```
//打开txt文件
private void btnOpen_Click(object sender, EventArgs e)
{
    OpenFileDialog txtDialog = new OpenFileDialog();
    txtDialog.Multiselect = false;
    txtDialog.Title = "打开txt文件";
    txtDialog.Filter = "txt坐标文件（*.txt）|*.txt";
    if (txtDialog.ShowDialog() == DialogResult.OK)
    {
        TxtFile.Text = txtDialog.FileName;
    }
}
//选择shapefile文件保存路径
private void btnSave_Click(object sender, EventArgs e)
{
    SaveFileDialog txtDialog = new SaveFileDialog();
    txtDialog.Title = "保存为shp文件";
    txtDialog.Filter = "shapefile文件（*.shp）|*.shp";
    if (txtDialog.ShowDialog() == DialogResult.OK)
    {
        SavePath.Text = txtDialog.FileName;
    }
}
```

（4）双击“TxtToPointShp”按钮添加代码将txt文件转换为点要素的shp图层：

FrmTxtToShp.cs（节选）	功能：将txt文件转换为点要素的shp图层

```
//将txt文件转换为点要素的shp图层
private void TxtToPoint_Click(object sender, EventArgs e)
{
    IMap map = m_hookHelper.FocusMap;
    m_pts = GetPoints(TxtFile.Text);
    if (m_pts == null)
    {
        MessageBox.Show("选择文件是空");
    }
    else
    {
        //调用CreatePointShpFromPoints函数
        IFeatureLayer pFeatureLayer = CreatePointShpFromPoints(
```

```
                                                    m_pts, SavePath.Text);
            map.AddLayer(pFeatureLayer);
        }
}
```

添加 CreatePointShpFromPoints()函数的定义，函数 CreateShapeFeatureClass()创建 Shapefile 要素类的参考代码略。

FrmTxtToShp.cs（节选） | 功能：将 txt 文件转换为点要素的 shp 图层

```
private IFeatureLayer CreatePointShpFromPoints(IPointCollection pts,
                                                    string FilePath)
{
    int index = FilePath.LastIndexOf('\\');
    string Folder = FilePath.Substring(0, index);
    string ShapeName = "pt" + FilePath.Substring(index + 1);
    IWorkspaceFactory ws = new ShapefileWorkspaceFactoryClass();
    IFeatureWorkspace fws = (IFeatureWorkspace)ws.OpenFromFile(Folder, 0);
    IFeatureClass ptFeatureCls = CreateShapeFeatureClass(fws, ShapeName,
                                        esriGeometryType.esriGeometryPoint);
    for (int j = 0; j < pts.PointCount; j++) //将点集的值逐个赋给新建的Point中
    {
        IPoint pt = new PointClass(); //新建PointClass
        pt.X = pts.get_Point(j).X;
        pt.Y = pts.get_Point(j).Y;
        IFeature Feature = ptFeatureCls.CreateFeature();//创建要素
        Feature.Shape = pt;//将Point赋给Feature.Shape
        Feature.Store();//存储要素
    }
    IFeatureLayer PoFeatureLayer = new FeatureLayerClass();
    PoFeatureLayer.Name = ptFeatureCls.AliasName;
    PoFeatureLayer.FeatureClass = ptFeatureCls;
    return PoFeatureLayer;
}
```

（5）双击“TxtToPolygonShp”按钮添加代码将 txt 文件转换为多边形要素的 shp 图层，并添加 CreatePolygonShpFromPoints()函数的定义。

FrmTxtToShp.cs（节选） | 功能：将 txt 文件转换为多边形要素的 shp 图层

```
//将txt文件转换为多边形要素的shp图层
private void TxtToPolygon_Click(object sender, EventArgs e)
{
    IMap map = m_hookHelper.FocusMap;
    m_pts = GetPoints(TxtFile.Text);
    if (m_pts == null)
    {
        MessageBox.Show("选择文件是空");
    }
    else
```

```
        {
            IFeatureLayer pFeatureLayer = CreatePolygonShpFromPoints(
                                                        m_pts, SavePath.Text);
            map.AddLayer(pFeatureLayer);
        }
}
private IFeatureLayer CreatePolygonShpFromPoints(IPointCollection pts,
                                                        string FilePath)
{
        int index = FilePath.LastIndexOf('\\');
        string Folder = FilePath.Substring(0, index);
        string ShapeName = "poly" + FilePath.Substring(index + 1);
        IWorkspaceFactory ws = new ShapefileWorkspaceFactoryClass();
        IFeatureWorkspace fws = (IFeatureWorkspace)ws.OpenFromFile(Folder, 0);
        IFeatureClass polyFeatureCls = CreateShapeFeatureClass(fws, ShapeName,
                                        esriGeometryType.esriGeometryPolygon);
        //创建多边形类
        IPolygon polygon = new PolygonClass();
        IPointCollection polygonCol = polygon as IPointCollection;
        polygonCol.AddPointCollection(pts);
        IFeature pFeature = polyFeatureCls.CreateFeature();//创建要素
        pFeature.Shape = polygon;
        pFeature.Store();
        IFeatureLayer pFeaturelayer = new FeatureLayerClass();
        pFeaturelayer.Name = polyFeatureCls.AliasName;
        pFeaturelayer.FeatureClass = polyFeatureCls;
        return pFeaturelayer;
}
```

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

FrmTxtToShp.cs（节选）	功能：调用窗体实现 txt 转 shp

```
private void textToShpToolStripMenuItem_Click(object sender, EventArgs e)
{
        FrmTxtToShp TxtToPoint = new FrmTxtToShp(m_mapControl.Object);
        TxtToPoint.ShowDialog();
}
```

（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】，点击“TextToShp”，打开文本文件的点数据，自定义需要保存的 shp 文件路径，如图 6-22 所示。

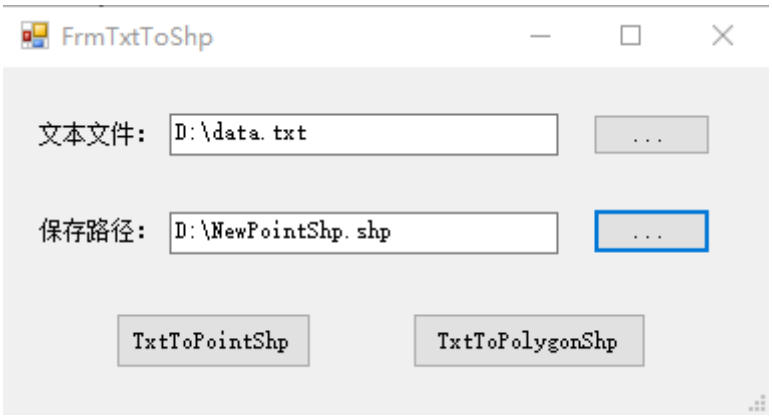

图 6-22 打开文本数据和设置 shp 文件保存名称和路径

点击“TxtToPointShp”按钮，运行效果如图 6-23 所示。

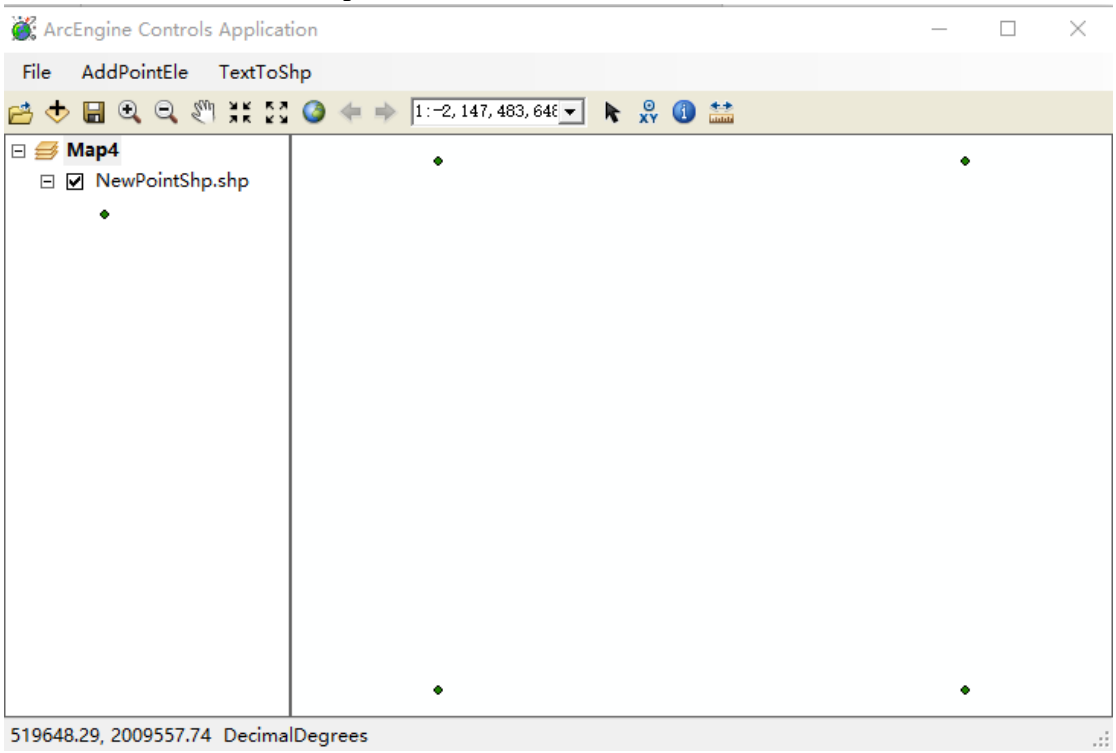

图 6-23 生成点要素的 shp 图层

点击“TxtToPolygonShp”按钮，运行效果如图 6-24 所示。

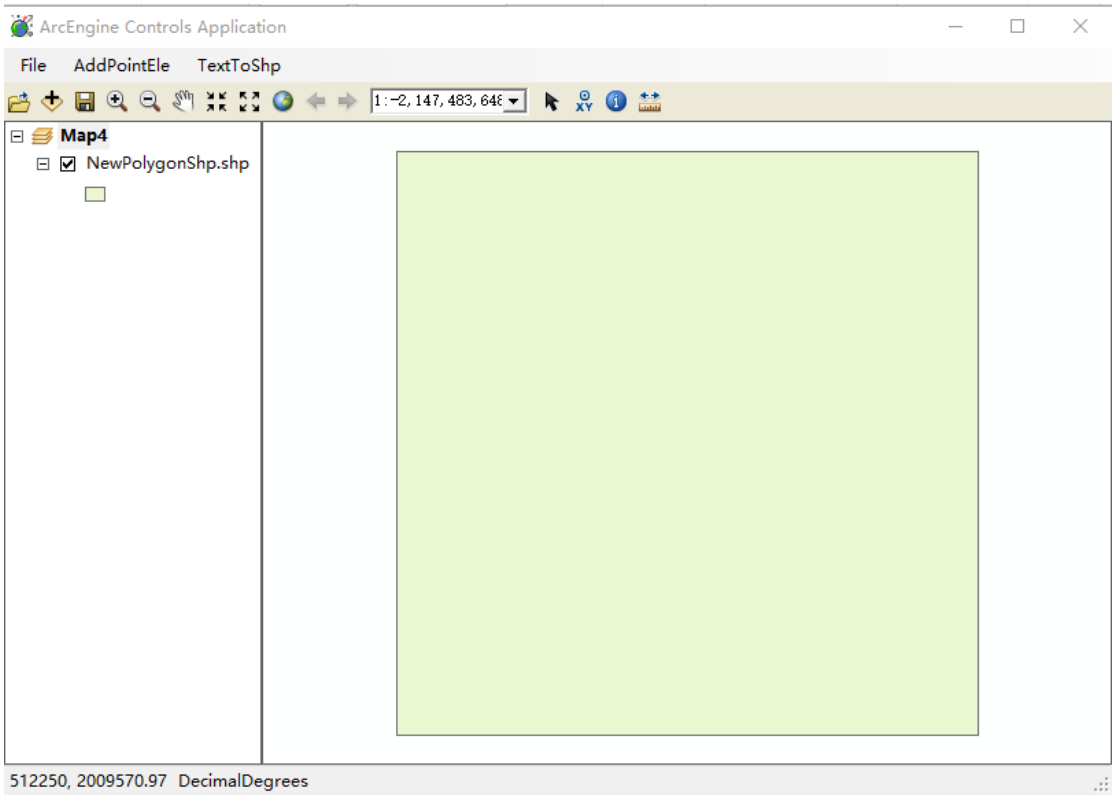

图 6-24 生成多边形要素的 shp 图层