第 7 章

数据编辑

7.1 简介

空间数据编辑是 GIS 的基本功能之一，ArcEngine 提供了丰富的编辑功能。其中 IWorkspaceEdit 接口是空间数据编辑功能最重要的接口，通过它可以启动或停止一个编辑流程。GIS 的编辑操作往往是长事务，因此需要确保用户在这段时间内所做的编辑能够恢复。该接口提供了 13 种方法，如 StartEditing 方法用于启动一个编辑流程，WithUndoRedo 参数用来确定工作空间是否支持“恢复/取消恢复”操作；AbortEditoperation 方法用于在编辑过程中如果出现异常，则取消所有的编辑操作；StopEditoperation 方法用来确保编辑操作的完成；UndoEditoperation 方法用于编辑操作的回滚；StopEditing 方法用来完成编辑。本章通过两个例子程序介绍要素的编辑和高级编辑。

7.2 捕捉功能设计与实现

捕捉是新建地理空间对象与已有地理空间对象的一种关联关系。比如，要在自来水管线上增加一个阀门，来控制管线中水的流动，因此阀门应该位于管线之上，并在此处打断管线，如何将阀门准确地定位到管线上，有没有偏离？这需要用捕捉来实现。当用户在创建阀门的时候，应当尽可能将输入点置于管线附近，如果阀门与管线的距离小于一定的范围（即容差），就认为用户捕捉到管线，同时返回阀门向管线引垂线与管线交点的坐标。

新建一个 VS2005 工程，工程命名为“snap”，在 Form1 窗体上添加一个 MapControl 控件，一个 LicenseControl 控件，一个 Label 控件，一个 Combox 控件，然后修改控件的 Text 属性，效果如图 7-1 所示。

在代码窗口中添加如下代码。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

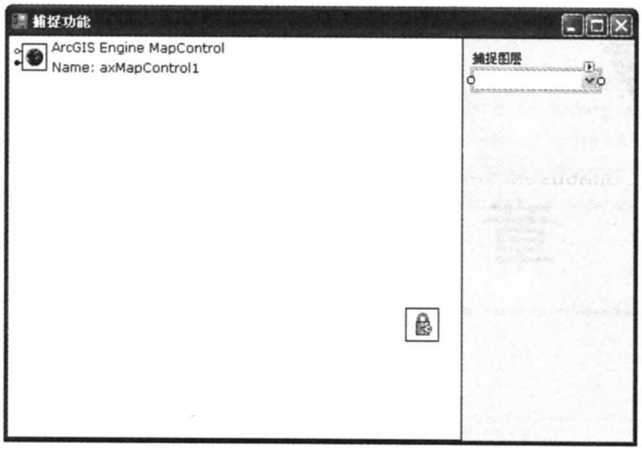


图 7-1　新建工程 Snap

```
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.Geodatabase;

namespace Snap
{
    public partial class Form1 : Form
    {
        bool bCreateElement = true;
        int internalTime = 5;                          //时间间隔
        int snapTime = 10;                             //初始值
        IElement m_element = null;                     //界面绘制点元素
        IPoint currentPoint = new PointClass();        //当前鼠标点
        IPoint snapPoint = null;                       //捕捉到的点
        IMovePointFeedback movePointFeedback = new MovePointFeedbackClass();
        string snapLayer ="";
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            //加载地图文档
            loadMapDocument();
            for (int i = 0; i < this.axMapControl1.LayerCount; i++)
            {
                cbLayerName.Items.Add(this.axMapControl1.get_Layer(i).Name);
            }
            cbLayerName.Text = cbLayerName.Items[0].ToString();
```

```
            snapLayer =cbLayerName.Text ;
        }

        //鼠标光标移动事件
        private void axMapControl1_OnMouseMove(object sender, ESRI.ArcGIS.Controls.
IMapControlEvents2_OnMouseMoveEvent e)
        {
            currentPoint.PutCoords(e.mapX, e.mapY);

            snapTime++;
            snapTime = snapTime % internalTime;
            ILayer layer = GetLayerByName(snapLayer, axMapControl1);
            IFeatureLayer featureLayer = layer as IFeatureLayer;
            if (bCreateElement)
            {
                CreateMarkerElement(currentPoint);
                bCreateElement = false;
            }
            if (snapPoint == null)
                ElementMoveTo(currentPoint);
            //鼠标光标自动捕获顶点
            if (snapTime == 0)
                snapPoint = Snapping(e.mapX, e.mapY, featureLayer );
            if (snapPoint != null && snapTime == 0)
                ElementMoveTo(snapPoint);

        }
        //捕捉
        public IPoint Snapping(double x, double y, IFeatureLayer featureLayer)
        {
            IMap map = this.axMapControl1.Map;
            IActiveView activeView = this.axMapControl1.ActiveView;
            IFeatureClass featureClass = featureLayer.FeatureClass;
            IPoint point = new PointClass();
            point.PutCoords(x, y);

            IFeature  feature = featureClass.GetFeature(0);
            IPoint hitPoint1 = new PointClass();
            IPoint hitPoint2 = new PointClass();
            IHitTest hitTest = feature.Shape as IHitTest;
            double hitDist = 0;
            int partIndex = 0;
            int vertexIndex = 0;
            bool bVertexHit = false;

            double tol = ConvertPixelsToMapUnits(activeView , 8);
            if (hitTest.HitTest(point, tol, esriGeometryHitPartType.esriGeometryPartBoundary,
                hitPoint2, ref hitDist, ref partIndex, ref vertexIndex, ref bVertexHit))
            {
                hitPoint1 = hitPoint2;
            }
```

```
        axMapControl1.ActiveView.Refresh();
        return hitPoint1;
    }
    public void CreateMarkerElement(IPoint point)
    {
        IActiveView activeView = this.axMapControl1.ActiveView ;
        IGraphicsContainer graphicsContainer = axMapControl1.Map as Igraphics Container;
        //建立一个 marker 元素
        IMarkerElement markerElement = new MarkerElement() as IMarkerElement  ;
        ISimpleMarkerSymbol simpleMarkerSymbol = new SimpleMarkerSymbol();
        //符号化元素
        IRgbColor rgbColor1 = new RgbColor();
        rgbColor1.Red = 255;
        rgbColor1.Blue = 0;
        rgbColor1.Green = 0;
        simpleMarkerSymbol.Color = rgbColor1;
        IRgbColor rgbColor2 = new RgbColor();
        rgbColor2.Red = 0;
        rgbColor2.Blue = 255;
        rgbColor2.Green = 0;
        simpleMarkerSymbol.Outline = true;
        simpleMarkerSymbol.OutlineColor = rgbColor2 as IColor;
        simpleMarkerSymbol.OutlineSize = 1;
        simpleMarkerSymbol.Size = 5;
        simpleMarkerSymbol.Style = esriSimpleMarkerStyle.esriSMSCircle;
        ISymbol symbol = simpleMarkerSymbol as ISymbol;
        symbol.ROP2 = esriRasterOpCode.esriROPNotXOrPen;
        markerElement.Symbol = simpleMarkerSymbol;
        m_element = markerElement as IElement;
        m_element.Geometry = point as IGeometry;
        graphicsContainer.AddElement(m_element, 0);
        activeView.PartialRefresh(esriViewDrawPhase.esriViewGraphics, m_element, null);
        IGeometry geometry = m_element.Geometry;
        movePointFeedback.Display = activeView.ScreenDisplay;
        movePointFeedback.Symbol = simpleMarkerSymbol as ISymbol;
        movePointFeedback.Start(geometry as IPoint, point);
    }
    //移动元素到新的位置
    public void ElementMoveTo(IPoint point )
    {
        //移动元素
        movePointFeedback.MoveTo(point);
        IGeometry geometry1 = null;
        IGeometry geometry2 = null;
        if (m_element != null)
        {
            geometry1 = m_element.Geometry;
            geometry2 = movePointFeedback.Stop();
            m_element.Geometry = geometry2;
            //更新该元素的位置
            this.axMapControl1.ActiveView.GraphicsContainer.UpdateElement
```

```
(m_element);
                //重新移动元素
                movePointFeedback.Stop();//(geometry1 as IPoint, point);
                this.axMapControl1.ActiveView.PartialRefresh(esriViewDrawPhase.
esriViewGeography, null, null);
            }
        }
        //通过名称获取图层
        public ILayer GetLayerByName(string layerName, AxMapControl axMapControl1)
        {
            for (int i = 0; i < axMapControl1.LayerCount; i++)
            {
                if (axMapControl1.get_Layer(i).Name.Equals(layerName))
                    return axMapControl1.get_Layer(i);
            }
            return null;
        }
        //转换像素到地图单位
        public double ConvertPixelsToMapUnits(IActiveView activeView, double pixelUnits)
        {
            double realDisplayExtent;
            int pixelExtent;
            double sizeOfOnePixel;
            pixelExtent = activeView.ScreenDisplay.DisplayTransformation.get_Device
Frame().right - activeView.ScreenDisplay.DisplayTransformation.get_DeviceFrame().left;
            realDisplayExtent = activeView.ScreenDisplay.DisplayTransformation.
Visible Bounds.Width;
            sizeOfOnePixel = realDisplayExtent / pixelExtent;
            return pixelUnits * sizeOfOnePixel;
        }

        //加载地图文档
        private void loadMapDocument()
        {
            System.Windows.Forms.OpenFileDialog openFileDialog;
            openFileDialog = new OpenFileDialog();
            openFileDialog.Title = "打开地图文档";
            openFileDialog.Filter = "map documents(*.mxd)|*.mxd";
            openFileDialog.ShowDialog();
            string filePath = openFileDialog.FileName;
            if (axMapControl1.CheckMxFile(filePath))
            {
                axMapControl1.MousePointer = esriControlsMousePointer.esriPointerHourglass;
                axMapControl1.LoadMxFile(filePath, 0, Type.Missing);
                axMapControl1.MousePointer = esriControlsMousePointer.esriPointerDefault;
            }
            else
            {
                MessageBox.Show(filePath + "不是有效的地图文档");
            }
        }
```

```
        //更改当前捕捉图层名
        private void cbLayerName_TextChanged(object sender, EventArgs e)
        {
            snapLayer = cbLayerName.Text;
        }
    }
}
```

运行代码，移动鼠标光标到线附近时，效果如图 7-2（图片没有抓住鼠标光标的形状）所示。

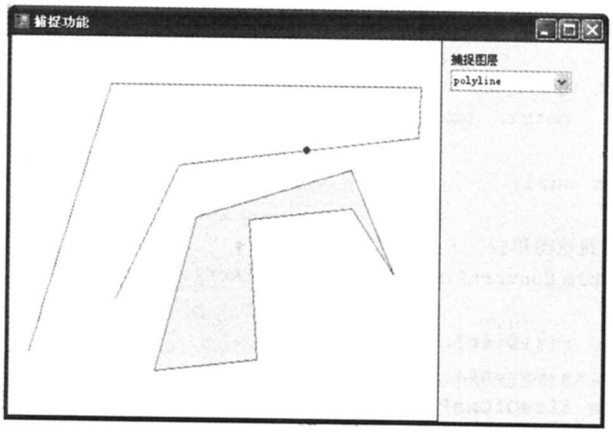


图 7-2　程序运行效果

7.3　要素编辑

空间数据编辑是 GIS 的重要功能之一，ArcEngine 提供了丰富的编辑功能，本节分类介绍编辑功能。

7.3.1　开始编辑

IWorkspaceEdit 接口是空间数据编辑功能的重要接口，可以通过它来启动或停止一个编辑流程，在这个编辑流程内，可以对地理数据进行更新操作。由于 GIS 的编辑操作往往都是长事务过程，所以依次在流程内的编辑操作可以恢复。

开始编辑使用该接口的 StartEditing 方法，该方法的 withUndoRedo 参数用来确定工作空间是否支持“恢复/取消恢复”操作。如果需要用到此功能，则应将所有对象相关的更改放在一个流程内。启动编辑后，可以使用 StartEditOperation 方法开始编辑操作，这个操作不是一个流程，可以理解为只是数据更改的一个动作。但在编辑的过程中出现异常时，可以使用 AbortEditOperation 方法取消所有的编辑操作。

7.3.2　结束编辑

用户完成数据的编辑后，可以使用 StopEditOperation 方法来确保编辑操作的完成。UndoEditOperation 方法可以用于处于编辑状态的回滚操作，如果发现编辑过程有误，通过执行

这个方法可以恢复到最近变化前的状态。 StopEditing 方法用来完成编辑，执行该方法后，先前所做的编辑操作都不能再回滚，类似数据库中的 Commit 命令。

下面的程序演示如何使用：

```
private void WorkspaceEdit()
        {
            IWorkspaceFactory pWorkspaceFactory;
            pWorkspaceFactory = new AccessWorkspaceFactoryClass();
            IFeatureWorkspace pFeatureWorkspace;
            pFeatureWorkspace = pWorkspaceFactory.OpenFromFile(@"D:\Usa.mdb", 0)
            as IFeatureWorkspace;
            IFeatureClass pFeatureClass;
            pFeatureClass = pFeatureWorkspace.OpenFeatureClass("States");
            IWorkspaceEdit pWorkspaceEdit;
            pWorkspaceEdit = pFeatureWorkspace as IWorkspaceEdit;
            IFeature pFeature ;
            bool bHasEdits = true;
            pWorkspaceEdit.StartEditing(true);
            pWorkspaceEdit.RedoEditOperation();
            pFeature = pFeatureClass.GetFeature(1);
            pFeature.Delete();
            pWorkspaceEdit.StopEditOperation();
            DialogResult iResponse;
            iResponse = MessageBox.Show("Edit Operation", "Undo operation?",
            Message BoxButtons.YesNo);
            if (iResponse == DialogResult.Yes)
            {
                pWorkspaceEdit.UndoEditOperation();
            }
            pWorkspaceEdit.HasEdits(ref bHasEdits);
            if (bHasEdits)
            {
                iResponse = MessageBox.Show("Edit Operation","Save edits?", Message
                BoxButtons.YesNo);
                if (iResponse == DialogResult.Yes)
                {
                    pWorkspaceEdit.StopEditing(true);
                }
                else
                {
                    pWorkspaceEdit.StopEditing(false);
                }
            }
        }
```

7.3.3 图形编辑

图形编辑是数据编辑的主要实现，即对空间要素几何形状所做的修改，如移动位置、旋转、删除节点、增加节点。主要用到以下几个接口：IFeature 、IFeatureEdit、IWorkspaceEdit、

IFeatureClass、IFeatureCursor。

IFeatureEdit 中的 MoveSet、RotateSet、DeleteSet 分别是移动、旋转、删除由一个或者多个要素组成的要素集。Split 主要用来分割几何形体，通过点分割线，通过多义线分割多边形；分割后旧的被删除，新的要素自动产生。因为每个要素都有对应的属性，因此要素改变了，其对应的属性也会相应地改变。Splitattributes 用来分割要素属性字段中的值，该方法可以在 Split 方法执行后自动执行。

Feature 又派生出很多类，如 networkfeature、rastercatalogfeature、coverageannotationfeature、annotationfeature、dimensionfeature。feature 的主要接口是 Ifeature，其中的 Extent 用来返回要素对象的包络线，显示要素的空间范围；Featuretype 用来返回要素的类型；Shape 用来返回要素的几何形体对象；ShapeCopy 用来得到几何形体对象的一个复制。

新建一个 VS2005 工程，工程命名为“FeatureEdit”，在 Form1 窗体上添加一个 MapControl 控件，一个 LicenseControl 控件，一个 Label 控件，一个 Combox 控件，3 个 Button 控件；然后修改控件的 Text 属性，效果如图 7-3 所示。

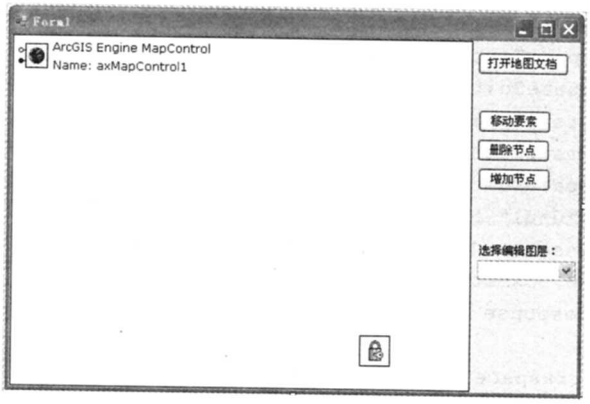


图 7-3　新建工程 FeatureEdit

在代码窗口中添加如下代码。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

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

namespace FeatureEdit
{
    public partial class Form1 : Form
    {
```

```
//操作类型
string strOperator="";
//当前地图视图
IActiveView m_activeView = null;
//当前操作图层
IFeatureLayer m_FeatureLayer = null;
//当前操作实体
IFeature m_Feature = null;
 //当前点移动反馈对象
IMovePointFeedback m_MovePointFeedback = new MovePointFeedbackClass();
//当前线移动反馈对象
IMoveLineFeedback m_MoveLineFeedback = new MoveLineFeedbackClass();
//当前面移动反馈对象
IMovePolygonFeedback m_MovePolygonFeedback = new MovePolygonFeedbackClass();

public Form1()
{
    InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
    //加载地图文档
    loadMapDocument();
    //将图层名添加到下拉列表中
    for (int i = 0; i < this.axMapControl1.LayerCount; i++)
    {
        ILayer layer = this.axMapControl1.get_Layer(i);
        this.comboBox1.Items.Add(layer.Name);
    }
}
//选择操作图层，并让 Map 选中图层的第 1 个实体
private void comboBox1_TextChanged(object sender, EventArgs e)
{
    if (this.comboBox1.Text != "")
    {
        for (int i = 0; i < this.axMapControl1.LayerCount; i++)
        {
            ILayer layer = this.axMapControl1.get_Layer(i);
            if (layer.Name == this.comboBox1.Text.ToString())
            {
                m_FeatureLayer = layer as IFeatureLayer;
                m_Feature = m_FeatureLayer.FeatureClass.GetFeature(0);

                if (m_Feature != null)
                {
                   this.axMapControl1.Map.SelectFeature(m_FeatureLayer, m_Feature);
                }
                m_activeView = this.axMapControl1.ActiveView;
                return;
            }
        }
```

```
        }
    }
    //加载地图文档
    private void loadMapDocument()
    {
        System.Windows.Forms.OpenFileDialog openFileDialog;
        openFileDialog = new OpenFileDialog();
        openFileDialog.Title = "打开地图文档";
        openFileDialog.Filter = "map documents(*.mxd)|*.mxd";
        openFileDialog.ShowDialog();
        string filePath = openFileDialog.FileName;
        if (axMapControl1.CheckMxFile(filePath))
        {
            axMapControl1.MousePointer = esriControlsMousePointer.esriPointerHourglass;
            axMapControl1.LoadMxFile(filePath, 0, Type.Missing);
            axMapControl1.MousePointer = esriControlsMousePointer.esriPointerDefault;
        }
        else
        {
            MessageBox.Show(filePath + "不是有效的地图文档");
        }
    }
    //初始化反馈对象参数
    private void button3_Click(object sender, EventArgs e)
    {
        strOperator ="move";
        m_MovePointFeedback = new MovePointFeedbackClass();
        m_MoveLineFeedback = new MoveLineFeedbackClass();
        m_MovePolygonFeedback = new MovePolygonFeedbackClass();
    }
    //删除节点
    private void button4_Click(object sender, EventArgs e)
    {
        IPointCollection pointCollection;
        if (m_Feature.Shape.GeometryType == esriGeometryType.esriGeometryPolyline)
        {
            pointCollection = new PolylineClass();
            IPolyline polyline = m_Feature.Shape as IPolyline;

            pointCollection = polyline as IPointCollection;
            //如果点个数少于两个就无法构成线
            if (pointCollection.PointCount > 2)
            {
                //移除指定的节点
                pointCollection.RemovePoints(pointCollection.PointCount - 1, 1);
            }
        }
        else if (m_Feature.Shape.GeometryType == esriGeometryType.esriGeometry Polygon)
        {
            pointCollection = new  PolygonClass ();
            IPolygon  polygon = m_Feature.Shape as  IPolygon ;
```

```
        pointCollection = polygon  as IPointCollection;
        //如果点个数少于 3 个就无法构成面
        if (pointCollection.PointCount > 3)
        {
            //移除指定的节点
            pointCollection.RemovePoints(pointCollection.PointCount - 1, 1);
        }
    }
    IWorkspaceEdit workspaceEdit;
    IWorkspace workspace;
    IDataset dataset = m_FeatureLayer.FeatureClass as IDataset;
    workspace = dataset.Workspace;
    workspaceEdit = workspace as IWorkspaceEdit;
    //开始编辑
    workspaceEdit.StartEditing(true);
    workspaceEdit.StartEditOperation();
    //保存数据
    m_Feature.Store();
    //结束编辑
    workspaceEdit.StopEditOperation();
    workspaceEdit.StopEditing(true);
    m_activeView.Refresh();
}

private void button5_Click(object sender, EventArgs e)
{
    IPointCollection pointCollection;
    IPoint point = new PointClass();
    IPoint fromPoint = new PointClass();
    IPoint toPoint = new PointClass();
    if (m_Feature.Shape.GeometryType == esriGeometryType.esriGeometryPolyline)
    {
        pointCollection = new PolylineClass();
        IPolyline polyline = m_Feature.Shape as IPolyline;

        object missing1 = Type.Missing;
        object missing2 = Type.Missing;
        pointCollection = polyline as IPointCollection;
        //获取线对象的最后两个点
        fromPoint = pointCollection.get_Point(pointCollection.PointCount - 2);
        toPoint = pointCollection.get_Point(pointCollection.PointCount - 1);
        //根据线最后两个点创建一个新点
        point.PutCoords((fromPoint.X + toPoint.X) / 2, (fromPoint.Y +
        toPoint.Y) / 2 + 50);
        //将新点添加到线对象的点集合中
        pointCollection.AddPoint(point, ref missing1, ref missing2);

    }
    else if (m_Feature.Shape.GeometryType == esriGeometryType.esriGeometry Polygon)
    {
        pointCollection = new PolygonClass();
```

```
            IPolygon polygon = m_Feature.Shape as IPolygon;

            object missing1 = Type.Missing;
            object missing2 = Type.Missing;
            pointCollection = polygon as IPointCollection;
            //获取面对象点集最后两个点
            fromPoint = pointCollection.get_Point(pointCollection.PointCount - 2);
            toPoint = pointCollection.get_Point(pointCollection.PointCount - 1);
            //根据线最后两个点创建一个新点
            point.PutCoords((fromPoint.X + toPoint.X) / 2, (fromPoint.Y + toPoint.Y)
            / 2 + 50);
            //将新点添加到线对象的点集合中
            pointCollection.AddPoint(point, ref missing1, ref missing2);
        }
        IWorkspaceEdit workspaceEdit;
        IWorkspace workspace;
        IDataset dataset = m_FeatureLayer.FeatureClass as IDataset;
        workspace = dataset.Workspace;
        workspaceEdit = workspace as IWorkspaceEdit;
        //开始编辑
        workspaceEdit.StartEditing(true);
        workspaceEdit.StartEditOperation();
        //保存数据
        m_Feature.Store();
        //结束编辑
        workspaceEdit.StopEditOperation();
        workspaceEdit.StopEditing(true);
        m_activeView.Refresh();
    }

    private void axMapControl1_OnMouseDown(object sender, IMapControlEvents2_
    On MouseDownEvent e)
    {
        IPoint point = new PointClass();
        IFeatureClass featureClass = null;
        if (m_Feature == null) return;
        switch (strOperator)
        {
            case "move":
                //将当前鼠标光标位置的点转换为地图上的坐标
                point = m_activeView.ScreenDisplay.DisplayTransformation.ToMap
                Point(e.x, e.y);
                if (m_Feature.Shape.GeometryType == esriGeometryType.esriGeometry
                Point)
                {
                    //设置显示对象，并启动移动
                    m_MovePointFeedback.Display = m_activeView.ScreenDisplay;
                    m_MovePointFeedback.Start(m_Feature.Shape as IPoint, point);
                }
                else if (m_Feature.Shape.GeometryType == esriGeometryType.esri
                GeometryPolyline)
```

```
            {
                //设置显示对象，并启动移动
                m_MoveLineFeedback.Display = m_activeView.ScreenDisplay;
                m_MoveLineFeedback.Start(m_Feature.Shape as IPolyline, point);
            }
            else if (m_Feature.Shape.GeometryType == esriGeometryType.esri
            GeometryPolygon)
            {
                //设置显示对象，并启动移动
                m_MovePolygonFeedback.Display = m_activeView.ScreenDisplay;
                m_MovePolygonFeedback.Start(m_Feature.Shape as IPolygon, point);
            }
            break;
        }
    }

    private void axMapControl1_OnMouseMove(object sender, IMapControlEvents2_
    On MouseMoveEvent e)
    {
        IPoint point = new PointClass();
        switch (strOperator)
        {
            case "move":
                if (m_Feature.Shape.GeometryType == esriGeometryType.esriGeometry
                Point)
                {
                    if (m_MovePointFeedback != null)
                    {
                        //将当前鼠标光标位置的点转换为地图上的坐标
                        point = m_activeView.ScreenDisplay.DisplayTransformation.
                        ToMapPoint(e.x, e.y);
                        //移动对象到当前鼠标光标位置
                        m_MovePointFeedback.MoveTo(point);
                    }
                }
                else if (m_Feature.Shape.GeometryType == esriGeometryType.esri
                GeometryPolyline)
                {
                    if (m_MoveLineFeedback != null)
                    {
                        //将当前鼠标光标位置的点转换为地图上的坐标
                        point = m_activeView.ScreenDisplay.DisplayTransformation.
                        ToMapPoint(e.x, e.y);
                        //移动对象到当前鼠标光标位置
                        m_MoveLineFeedback.MoveTo(point);
                    }
                }
                break;
        }
    }
```

```
        private void axMapControl1_OnMouseUp(object sender, IMapControlEvents2_On
        MouseUpEvent e)
        {
            if (m_Feature == null) return;
            IGeometry resultGeometry = null;
            switch (strOperator)
            {
                case "move":
                    if (m_Feature.Shape.GeometryType == esriGeometryType.esriGeometryPoint)
                    {
                        //停止移动
                        resultGeometry = m_MovePointFeedback.Stop() as IGeometry;
                        m_Feature.Shape = resultGeometry;
                    }
                    else if (m_Feature.Shape.GeometryType == esriGeometryType.esri
                    GeometryPolyline)
                    {
                        //停止移动
                        resultGeometry = m_MoveLineFeedback.Stop() as IGeometry;
                        m_Feature.Shape = resultGeometry;
                    }
                    else if (m_Feature.Shape.GeometryType == esriGeometryType.esri
                    GeometryPolygon)
                    {
                        //停止移动
                        resultGeometry = m_MovePolygonFeedback.Stop() as IGeometry;
                        m_Feature.Shape = resultGeometry;
                    }
                    IWorkspaceEdit workspaceEdit;
                    IWorkspace workspace;
                    IDataset dataset = m_FeatureLayer.FeatureClass as IDataset;
                    workspace = dataset.Workspace;
                    workspaceEdit = workspace as IWorkspaceEdit;
                    //开始编辑
                    workspaceEdit.StartEditing(true);
                    workspaceEdit.StartEditOperation();
                    //保存实体
                    m_Feature.Store();
                    //结束编辑
                    workspaceEdit.StopEditOperation();
                    workspaceEdit.StopEditing(true);
                    m_MovePointFeedback = null;
                    m_MoveLineFeedback = null;
                    m_MovePolygonFeedback = null;
                    break;
            }
            m_activeView.Refresh();
            this.axMapControl1.Map.ClearSelection();
        }
    }
}
```

7.4　高级编辑

ArcEngine 高级编辑操作主要会用到 ITopologicalOperator 接口，通过该接口可以实现要素分解、要素合并、生成平行线、打断线、延长线、裁剪等操作，ItopologicalOperator 将在第 10 章介绍。

7.5　本章小结

GIS 的显示内容主要是由各种要素构成，在实际应用中，经常需要捕获地图上的要素，这些捕获这些要素可以通过空间查询或捕捉来实现，本章分析了地图捕捉的使用场景，并给出了线、面捕捉的实例。GIS 的二次开发中，地图编辑是一个重要的功能，地图是在分布式环境下应用，因此编辑的长事务处理是地图编辑应该着重关心和考虑的问题，通过 ArcGISEngine 提供的工作空间编辑接口可以实现地图要素编辑的长事务管理，确保地图编辑的完整性和一致性。

本章最后讨论了地图交互对象 DisplayFeedback 对象，这些对象可以实现移动、旋转、节点编辑等重要高级功能，ArcEngine 平台提供了丰富的 DisplayFeedback 对象，本文将这些对象划分为两大类：产生新对象 DisplayFeedback 和基于显示目的的 DisplayFeedback ，并将无法划分到这两大类的 DisplayFeedback 对象进行了简单介绍。