第9章

三维可视化

9.1 简介

三维可视化即三维显示，是地理信息系统软件具备的一项基本功能。三维可视化能够为我们提供更多的知识，而这些知识是很难从二维平面地图上得到的。例如，通过三维可视化可以实际地看到山谷的存在，而不需要从等高线数据推断山谷的存在。

ArcScene 用于创建多图层的场景，并且可以控制每个图层场景以及在三维空间的定位和渲染等功能。可以控制整个场景的某些属性，也可以在场景中通过特征的属性或该特征相对于其他特征的位置来选取它们，同时可以从任意一个视角浏览整个三维场景。

9.2 数据的三维显示

在 ArcEngine 中提供了与地图控件类似的三维显示控件 SceneViewer Control，通过该控件，可以方便地显示三维数据。可以在三维场景中进行缩放、拖动、旋转等操作。在该控件中，鼠标光标默认是漫游状态，该控件直接提供了基本的三维操作。

9.2.1 DEM 数据的加载

DEM 数据加载需要用到两个组件类：Scene 和 SceneGraph。Scene 是一个矢量、栅格和图形数据显示与处理的容器，该类实现了 IScene 接口，提供了控制 Scene 的方法和属性，如 AddLayer 方法用于向场景中增加一个图层，SelectionCount 属性用于获取选择的实体数目。SceneGraph 是一个记录在 Scene 中出现的数据和事件的容器，该类实现了 ISceneGraph 接口，提供了控制和处理 Scene 中图形的方法和属性，如 Locate 方法用于通过单击场景中的任意点定位一个对象，RefreshViewers 用于重绘所有的视图，Remove 用于删除角色的一个对象。

9.2.2 叠加纹理数据

加载完 DEM 数据后，需要在三维物体上描绘细节，即纹理；建立 DEM 表面的点与纹理空间的点之间的关系，即纹理映射。

9.2.2.1 叠加影像数据

叠加影像数据就是把影像看做纹理，将其贴在地形表面，让其具有地形起伏的三维效果。叠加影像数据需要用到 4 个组件类：TinLayer、RasterLayer、Tin3DProperties 和 Raster3DProperties。TinLayer 组件类用于创建 Tin 图层对象，RasterLayer 组件类用于创建 Raster 图层对象，Tin3DProperties 组件类用于创建 Tin 数据的三维场景属性对象，Raster3DProperties 组件类用于创建 Raster 数据的三维场景属性对象。

TinLayer 和 RasterLayer 类都实现了 ILayerExtensions 接口，提供了两个方法和两个属性，如 ExtensionCount 用于获取当前图层对象扩展项的个数，RemoveExtension 用于移动指定的扩展项。

Tin3DProperties 和 Raster3DProperties 组件类实现了 I3DProperties 接口，提供了一个方法和一些属性，如 Apply3DProperties 用于应用三维属性到某个对象，BaseOption 用于设置赋予对象基本属性的方式。

9.2.2.2 叠加矢量数据

叠加矢量数据就是将具有高程属性矢量图层的数据加载到三维场景中，这样加载后的矢量数据会作为纹理自动贴在地形表面，具有起伏的三维显示效果。加载矢量数据使用的接口在前面已做过介绍，这里不再重复。

9.2.3 分层设色

根据地面高度划分的高程层，逐层设置不同的颜色，称为地貌分层设色法。通过分层设色，可使地貌高程分布及其相互对照鲜明。实现分层设色需要使用以下几个组件类：AlgorithmicColorRamp、RgbColor、RasterClassifyColorRampRenderer、SimpleFillSymbol 和 TinElevationRenderer。

AlgorithmicColorRamp 类实现了 IAlgorithmicColorRamp 接口，提供了一个方法和一些属性，如 CreateRamp 方法用于创建一定长度的颜色坡面，Size 属性用于设置产生的颜色数目。

RasterClassifyColorRampRenderer 类实现了 IRasterClassifyColorRampRenderer 接口，提供了按类别渲染 Raster 数据的相关属性，如 ClassCount 用于设置类别数目，Symbol 用于设置类别使用的符号。该类还实现了 IRasterRenderer 接口，提供了渲染的方法和属性，如 Update 用于更新修改后的渲染效果，Raster 用于设置需要渲染的 Raster 对象。

TinElevationRenderer 类实现了 ITinColorRampRenderer 接口，提供了基于连续属性值，如高程、坡度等。该接口提供了多个属性，如 BreakCount 用于设置分类数目，Break 用于设置分类的类别值。该类还实现了 IClassBreaksUIProperties 和 ITinRenderer 接口。

创建一个新 Windows 应用程序项目，命名为“lesson1”，在左边工具箱中拖放“axTOCControl”和“axSceneControl”控件到窗体上，在窗体上放置 4 个按钮控件，一个“openFileDialog”控件，一个“folderBrowserDialog”控件，然后修改控件的文本，如图 9-1 所示。

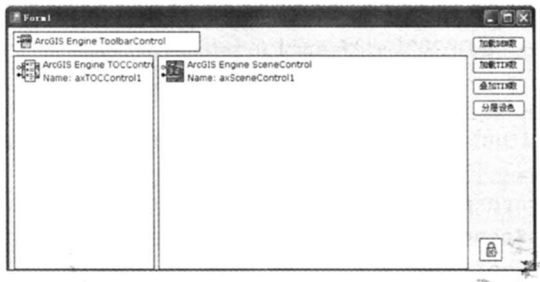
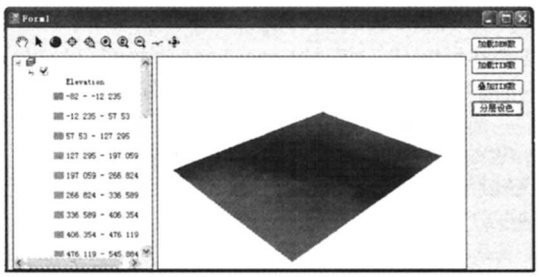
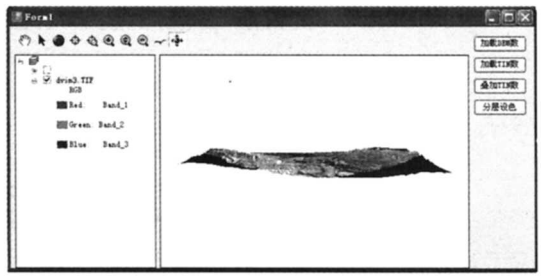


图 9-1　修改控件的文体

“加载 DEM 数”功能用于将 RASTER 数据加载到 SceneControl 控件；“加载 Tin 数”功能用于将 Tin 数据加载到 SceneControl 控件；“叠加 Tin 数”功能用于将 Tin 数据应用到 Raster 数据，使之具有三维起伏效果；“分层设色”功能可以用一个颜色带来显示 TIN 数据。程序运行效果如图 9-2 所示。



分层设色



叠加 TIN 数

图 9-2　程序运行效果

程序代码如下。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Analyst3D;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.DataSourcesRaster ;
using ESRI.ArcGIS.DataSourcesFile;
using ESRI.ArcGIS.esriSystem ;

namespace lesson1
{
    public partial class Form1 : Form
    {
        public Form1()
```

```
        {
            InitializeComponent();
        }
        //加载DEM数
        private void button1_Click(object sender, EventArgs e)
        {
            ISceneGraph pSceneGraph = this.axSceneControl1.SceneGraph;
            IScene pScene = pSceneGraph.Scene;
            IRasterLayer  pRasterLayer =new RasterLayerClass ();
            ILayer pLayer ;

            this.openFileDialog1.Title ="Raster layer";
            this.openFileDialog1.DefaultExt =".TIF";
            this.openFileDialog1.Filter ="(*.tif)|*.tif";
            if (this.openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                string pPathName = this.openFileDialog1.FileName;
                string pPath = pPathName.Substring(0, pPathName.LastIndexOf('\\'));
                string fileName = pPathName.Substring(pPath.Length+1, pPathName.
Length - pPath.Length-1);
                IWorkspaceFactory pwsf = new RasterWorkspaceFactoryClass();
                IRasterWorkspace pRasterWorkspace;
                if (pwsf.IsWorkspace(pPath))
                {
                    pRasterWorkspace = pwsf.OpenFromFile(pPath, 0) as IRasterWorkspace;
                    IRasterDataset pRasterDataset = pRasterWorkspace.OpenRaster
Dataset (fileName);
                    //pRasterDataset.OpenFromFile(pPath);
                    pRasterLayer.CreateFromDataset(pRasterDataset);
                    pLayer = pRasterLayer as ILayer;
                    pScene.AddLayer(pLayer, true);
                    //pScene.ExaggerationFactor = 6;
                    pSceneGraph.RefreshViewers();

                }
            }
        }
        //加载TIN数
        private void button2_Click(object sender, EventArgs e)
        {
            ISceneGraph pSceneGraph = this.axSceneControl1.SceneGraph;
            IScene pScene = pSceneGraph.Scene;
            ITinLayer tinLayer = new TinLayerClass();
            ILayer pLayer;
            FileInfo fileInfo;
            string tinPath;
            IWorkspaceFactory tinWorkspaceFactory = new TinWorkspaceFactoryClass();
            ITinWorkspace tinWorkspace;
            ITin tin;

            if (this.folderBrowserDialog1.ShowDialog() == DialogResult.OK)
```

```
        {
            tinPath = this.folderBrowserDialog1.SelectedPath;
            fileInfo =  new FileInfo(tinPath);
            if (tinWorkspaceFactory.IsWorkspace(fileInfo.DirectoryName))
            {
                tinWorkspace = tinWorkspaceFactory.OpenFromFile(fileInfo.
Directory Name, 0) as ITinWorkspace ;
                //tinWorkspace.OpenTin(fileInfo.DirectoryName);
                tin = tinWorkspace.OpenTin(fileInfo.Name);
                tinLayer.Dataset = tin;
                tinLayer.Visible = false;
                pScene.AddLayer(tinLayer as ILayer, true);
                pSceneGraph.RefreshViewers();
            }
        }
    }
    //叠加 TIN 数
    private void button3_Click(object sender, EventArgs e)
    {
        ISceneGraph pSceneGraph = this.axSceneControl1.SceneGraph;
        IScene pScene = pSceneGraph.Scene;
        ILayer layer = pScene.get_Layer(0);
        ITinLayer tinLayer = layer as ITinLayer;
        layer = pScene.get_Layer(1);
        IRasterLayer rasterLayer = layer as IRasterLayer;
        ITinAdvanced tinAdvanced;
        ISurface surface;
        tinAdvanced =tinLayer.Dataset as ITinAdvanced ;
        surface = tinAdvanced.Surface;

        ILayerExtensions layerExtensions = (ILayerExtensions)rasterLayer ;
        I3DProperties i3dProperties = null;

        for (int i = 0; i < layerExtensions.ExtensionCount; i++)
        {
            if (layerExtensions.get_Extension(i) is I3DProperties)
            {
                i3dProperties = (I3DProperties)layerExtensions.get_Extension(i);
            }
        }//get 3d properties from extension

        i3dProperties.BaseOption = esriBaseOption.esriBaseSurface;
        i3dProperties.BaseSurface = surface;
        i3dProperties.Apply3DProperties(rasterLayer);
    }
    //分层设色
    private void button4_Click(object sender, EventArgs e)
    { ILayer layer;
        ISceneGraph pSceneGraph = this.axSceneControl1.SceneGraph;
        IScene pScene = pSceneGraph.Scene;
```

```
                for (int i = 0; i < pScene.LayerCount; i++)
                {
                    layer = pScene.get_Layer(i);
                    if (layer is ITinLayer)
                    {
                        ITinLayer tinLayer = layer as ITinLayer;
                        //设置高程缩放因子
                        I3DProperties i3DProperties=null;
                        ILayerExtensions layerExtensions = layer as ILayerExtensions;
                        for (int j = 0; j < layerExtensions.ExtensionCount; j++)
                        {
                            if (layerExtensions.get_Extension(j) is I3DProperties)
                            {
                                i3DProperties = layerExtensions.get_Extension(j) as
I3DProperties;
                            }
                        }
                        if (i3DProperties != null)
                        {
                            i3DProperties.ZFactor = 0.0005;
                            i3DProperties.RenderMode = esriRenderMode.esriRenderCache;
                            i3DProperties.Apply3DProperties(layer);
                        }
                        ITinRenderer tinRenderer = new TinElevationRendererClass() as
Itin Renderer;
                        if (tinRenderer is ITinColorRampRenderer)
                        {
                            if (tinRenderer.Name == "Elevation")
                            {
                                ITinAdvanced tinAdvanced = tinLayer.Dataset as ITin
Advanced;
                                double dZMin = tinAdvanced.Extent.ZMin;
                                double dZMax = tinAdvanced.Extent.ZMax;
                                double dInterval = (dZMax - dZMin) /30;
                                ITinColorRampRenderer tinColorRampRenderer=tinRenderer
as ITinColorRampRenderer ;
                                IClassBreaksUIProperties classBreakUIProperties
=tinRenderer as IClass BreaksUIProperties ;
                                INumberFormat numberFormat=classBreakUIProperties.
NumberFormat ;
                                int lClasses=30;
                                tinColorRampRenderer.BreakCount =lClasses ;
                                double dLowBreak =dZMin ;
                                double dHighBreak=dLowBreak +dInterval ;
                                classBreakUIProperties.ColorRamp ="Custom";
                                IAlgorithmicColorRamp algorithmicColorRamp =Create
AlgorithmicColorRamp ();
                                IEnumColors enumColor =algorithmicColorRamp.Colors ;
                                enumColor.Reset ();

                                for (int k =0;k<lClasses ;k++)
```

```
                                {
                                    classBreakUIProperties.set_LowBreak (k,dLowBreak );
                                    tinColorRampRenderer.set_Break (k,dHighBreak );
                                    tinColorRampRenderer.set_Label(k,
numberFormat.Value ToString(dLowBreak) + " - " + numberFormat.ValueToString(dHigh
Break));
                                    dLowBreak = dHighBreak;
                                    dHighBreak = dHighBreak + dInterval;
                                    ISimpleFillSymbol  symbol =new SimpleFillSymbol
Class ();
                                    symbol.Color =enumColor.Next ();
                                    tinColorRampRenderer.set_Symbol (k,symbol as
ISymbol );
                                }
                                tinLayer.ClearRenderers ();
                                (tinColorRampRenderer as ITinRenderer ).Visible =true ;
                                tinLayer.InsertRenderer (tinColorRampRenderer  as
ITinRenderer ,0);
                            }
                            pScene.SceneGraph.Invalidate (tinLayer ,true,false );
                            this.axSceneControl1.SceneViewer.Redraw (true );
                            pScene.SceneGraph.RefreshViewers ();
                            this.axTOCControl1.ActiveView.PartialRefresh (esriView
DrawPhase.esriViewForeground ,tinLayer ,pScene.Extent );
                            this.axTOCControl1.Update ();

                        }

                    }
                }
            }
            private IAlgorithmicColorRamp CreateAlgorithmicColorRamp()
            {
                //创建一个新 AlgorithmicColorRampClass 对象
                IAlgorithmicColorRamp algColorRamp = new AlgorithmicColorRampClass();
                IRgbColor fromColor = new RgbColorClass();
                IRgbColor toColor = new RgbColorClass();
                //创建起始颜色对象
                fromColor.Red = 255;
                fromColor.Green = 0;
                fromColor.Blue = 0;
                //创建终止颜色对象
                toColor.Red = 0;
                toColor.Green = 255;
                toColor.Blue = 0;
                //设置 AlgorithmicColorRampClass 的起止颜色属性
                algColorRamp.ToColor = fromColor;
                algColorRamp.FromColor = toColor;
                //设置梯度类型
                algColorRamp.Algorithm = esriColorRampAlgorithm.esriCIELabAlgorithm;
                //设置颜色带颜色数量
                algColorRamp.Size = 30;
```

```
            //创建颜色带
            bool bture = true;
            algColorRamp.CreateRamp(out bture);
            return algColorRamp;
        }
    }
}
```

9.3 三维分析

三维分析是指在数字高程模型的基础上，利用空间分析算法获取研究区域中与空间特征相关的一些信息的过程。三维分析一般实现如下功能：任意点的定位查询、获取三维坐标值、距离查询、面积查询、体积查询、填挖方分析、剖面图绘制、地质特征三维分析等。

ArcEngine 提供了支持 Raster 数据和 TIN 数据进行各种三维分析工作，本节介绍三维场景属性查询、坡度分析、通视分析及剖面图绘制等功能。

9.3.1 三维场景属性查询

三维场景属性查询，即在单击时返回该位置的坐标、高程等信息。通过 ISceneGraph 接口的方法 Locate，可以获取 IPoint 接口对象，利用 IPoint 接口获得单击处的地理坐标和高程值。

9.3.2 坡度分析

坡度即水平面与局部地表之间的正切值，它包含两个部分：斜度和坡向。斜度即高度变化的最大值比率，坡向即变化比率最大值的方向。坡度分析需要用到如下几个类：Point、RasterWorkspace、RasterBand、PixelBlock 和 DblPnt。

Point 组件类用于创建一个二纬点对象，通常有长度、高度和 ID 属性。

RasterWorkspace 类，Raseter 工作区对象，该对象不可创建，对该对象的引用必须通过其他对象接口函数获得。

RasterBand 类，在磁盘上的 Raster 数据集的某一个波段，该类对象不可创建，对该对象的引用必须通过其他对象接口函数获得。

PixelBlock 类，该类的对象是个不可创建对象，对该对象的引用必须通过其他对象接口函数获得，可以从 Raster 数据集或 Raster 数据的某个波段直接读写像素值。

9.3.3 通视分析

通视分析有着很强的应用背景，如观察哨所位置的选择，哨所的位置应该设置在能监视某一感兴趣的区域，视线不能被地形挡住；低空侦察飞机在飞行时选择雷达盲区等。根据实际问题输出维数的不同，可分为点通视、线通视、面通视等。由于通视分析使用的方法 Visibility 只适用于 Raster 表面，因此 ArcEngine 没有提供对于 Tin 表面的分析方法。

9.3.4 剖面图绘制

剖面图可以反映地形表面沿着一条线的高程变化情况，可以帮助我们设计路线或评估一条给

定路线是否适合进行铁轨的铺设等。剖面图的绘制通常使用以下几个类：SceneGraph、Polyline、Tin、RasterSurface、LineElement、RgbColor、SimpleLineSymbol、GraphicsLayer3D 和 DataGraph。

SceneGraph 类实现了 IDisplay3D 接口，提供了 6 个方法和两个属性，如 FlashLocation 用于在指定的位置产生闪光效果，AddFlashFeature 用于添加实体对象到动画列表。

Tin 和 RasterSurface 类实现了 ISurface 接口，提供了一些属性和方法，如 InterpolateShape 用于为一个实体内插高程值。RasterSurface 类还实现了 IRasterSurface 接口，提供了操作和分析一个 Raster 表面的方法和属性。

GraphicsLayer3D 类实现了 IGraphicsContainer3D 接口，提供了操作图形容器的方法和属性。该类还实现了 IGraphicsLayer 和 IGraphicsSelection 接口，IGraphicsLayer 接口提供了控制图形图层的方法和属性，IGraphicsSelection 接口提供了选择三维图形的方法和属性。

DataGraph 类实现了 IDataGraph 和 IDataGraphProperties 接口，IDataGraph 接口提供了控制数据图表的方法和属性，IDataGraphProperties 接口提供了控制 DataGraph 对象属性的方法和属性。

创建一个新 Windows 应用程序项目，命名为“lesson2”，在左边工具箱中拖放“axTOCControl”和“axSceneControl”控件到窗体上，在窗体上放置 3 个按钮控件，一个“openFileDialog”控件，一个“folderBrowserDialog”控件，然后修改控件的文本，如图 9-3 所示。

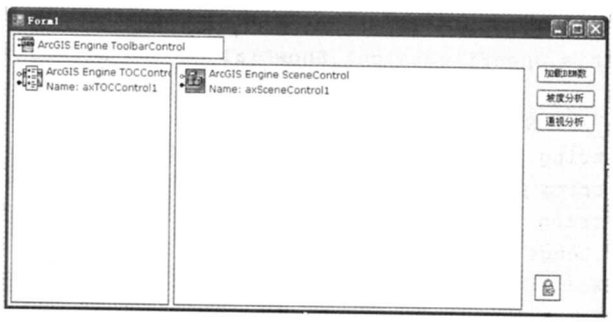


图 9-3　修改控件的文本

添加如下代码。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Analyst3D;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.DataSourcesRaster;
using ESRI.ArcGIS.DataSourcesFile;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Geometry ;
using ESRI.ArcGIS.GeoAnalyst;
using ESRI.ArcGIS.DataSourcesGDB;
```

```
namespace lesson2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        string m_FileName;
        //添加TIN数
        private void button1_Click(object sender, EventArgs e)
        {
            ISceneGraph pSceneGraph = this.axSceneControl1.SceneGraph;
            IScene pScene = pSceneGraph.Scene;
            IRasterLayer pRasterLayer = new RasterLayerClass();
            ILayer pLayer;

            this.openFileDialog1.Title = "Raster layer";
            this.openFileDialog1.DefaultExt = ".TIF";
            this.openFileDialog1.Filter = "(*.tif)|*.tif";
            if (this.openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                m_FileName = this.openFileDialog1.FileName;
                string pPathName = this.openFileDialog1.FileName;
                string pPath = pPathName.Substring(0, pPathName.LastIndexOf('\\'));
                string fileName = pPathName.Substring(pPath.Length + 1, pPath
Name.Length - pPath.Length - 1);
                IWorkspaceFactory pwsf = new RasterWorkspaceFactoryClass();
                IRasterWorkspace pRasterWorkspace;
                if (pwsf.IsWorkspace(pPath))
                {
                    pRasterWorkspace = pwsf.OpenFromFile(pPath, 0) as Iraster
Workspace;
                    IRasterDataset pRasterDataset = pRasterWorkspace.OpenRaster
Dataset(fileName);
                    //pRasterDataset.OpenFromFile(pPath);
                    pRasterLayer.CreateFromDataset(pRasterDataset);
                    pLayer = pRasterLayer as ILayer;
                    pScene.AddLayer(pLayer, true);
                    //pScene.ExaggerationFactor = 6;
                    pSceneGraph.RefreshViewers();

                }
            }
        }
        //坡度分析
        private void button2_Click(object sender, EventArgs e)
        {
            IWorkspaceFactory workspaceFactory = new RasterWorkspaceFactoryClass();
            FileInfo fileInfo = new FileInfo(m_FileName);
            string filePath = fileInfo.DirectoryName;
```

```
            string fileName = fileInfo.Name;
            //filePath =filePath +@"\workspace1";
            IRasterWorkspace workspace = workspaceFactory.OpenFromFile(filePath,
0) as IRasterWorkspace;

            ILayer layer = this.axSceneControl1.Scene.get_Layer(0);
            IRasterLayer rasterLayer = layer as IRasterLayer;
            IRasterDataset rasterDataset = workspace.OpenRasterDataset(fileName);

            ISurfaceOp surfaceOp = new RasterSurfaceOpClass();
            IRasterAnalysisEnvironment rasterAnalysisEnveronment;
            rasterAnalysisEnveronment = surfaceOp as IRasterAnalysisEnvironment;
            rasterAnalysisEnveronment.OutWorkspace = workspace as IWorkspace;
            object zFactor = new object();
            IGeoDataset geoDataset, rasterGeoDataset;
            rasterGeoDataset = rasterDataset as IGeoDataset;
            geoDataset = surfaceOp.Slope(rasterGeoDataset, esriGeoAnalysisSlope
Enum.esriGeoAnalysisSlopePercentrise, ref  zFactor);

            IRasterBandCollection rasterBandCollection = geoDataset as IrasterBand
Collection;

            rasterBandCollection.SaveAs("podu.tif", workspace as IWorkspace,
"TIFF");

        }

        private void axSceneControl1_OnMouseDown(object sender, ESRI.ArcGIS.
Controls.ISceneControlEvents_OnMouseDownEvent e)
        {
            ISceneGraph pSceneGraph = this.axSceneControl1.SceneGraph;
            IScene pScene = pSceneGraph.Scene;
            int px = e.x;
            int py = e.y;
            IPoint point = null;
            object pOwner;
            object pObject;
            pSceneGraph.Locate(this.axSceneControl1.SceneViewer, px, py,
esriScene PickMode.esriScenePickAll, true, out point, out pOwner, out pObject);
            if (point != null)
            {
                MessageBox.Show(point.X + "_" + point.Y + "_" + point.Z);
            }
        }
        //通视分析
        private void button3_Click(object sender, EventArgs e)
        {
            IWorkspaceFactory workspaceFactory = new RasterWorkspaceFactoryClass();
            FileInfo fileInfo = new FileInfo(m_FileName);
            string filePath = fileInfo.DirectoryName;
            string fileName = fileInfo.Name;
```

```
                IRasterWorkspace rasterWorkspace = workspaceFactory.Open FromFile
(filePath, 0) as IrasterWorkspace;

                IWorkspaceFactory pointWorkspaceFactory = new AccessWorkspaceFactory
Class();
                IWorkspace featureWorkspace = pointWorkspaceFactory.OpenFromFile
(filePath + @"\aaa.mdb", 0);
                //vIFeatureClass featureClass = featureWorkspace.OpenFeatureClass
("point");
                IEnumDataset enumDataset = featureWorkspace.get_Datasets(esriDataset
Type.esriDTAny);
                enumDataset.Reset();
                IGeoDataset featureDataset = enumDataset.Next() as IGeoDataset;


                ILayer layer = this.axSceneControl1.Scene.get_Layer(0);
                IRasterLayer rasterLayer = layer as IRasterLayer;
                IRasterDataset  rasterDataset  =  rasterWorkspace.OpenRasterDataset
(fileName);

                ISurfaceOp surfaceOp = new RasterSurfaceOpClass();
                IRasterAnalysisEnvironment rasterAnalysisEnveronment;
                rasterAnalysisEnveronment = surfaceOp as IRasterAnalysisEnvironment;
                rasterAnalysisEnveronment.OutWorkspace = rasterWorkspace as IWorkspace;
                object zFactor = new object();
                IGeoDataset geoDataset, rasterGeoDataset;
                rasterGeoDataset = rasterDataset as IGeoDataset;
                geoDataset = surfaceOp.Visibility(rasterGeoDataset, featureDataset,
esriGeo AnalysisVisibilityEnum.esriGeoAnalysisVisibilityFrequency);

                IRasterBandCollection rasterBandCollection = geoDataset as IrasterBand
Collection;

                rasterBandCollection.SaveAs("tongshi.tif", rasterWorkspace as IWorkspace,
"TIFF");
            }
        }
    }
```

9.4 本章小结

三维可视化是GIS应用的一个重要方面，本章讲述了三维数据的显示，通过三维数据加载、图层叠加、分层设色3个方面做介绍。三维分析是GIS应用系统中的高级分析功能，是衡量GIS功能的一个重要尺度，通过三维分析，用户可以得到更接近实际的分析结果，本章介绍了三维分析中常用的3个功能：坡度分析、通视分析、剖面图绘制。希望了解更多三维分析功能的读者可以参考专门的三维分析书籍或者查看ArcEngine在线帮助。