# 实验八 栅格数据处理

## 8.1 演示实例

栅格数据由一系列的规则格网单元组成，用于表达专题、光谱及图像等数据，可以在一定空间范围内模拟连续变化的地理现象或图片数据。栅格数据模型用连续空间的像元代表要素，每个栅格单元都有一个值，用于表示某个位置的某种属性，如高程、反射率、颜色等。栅格数据分为两种类型：（1）专题数据，用于地理分析；（2）影像数据，用于地图的背景显示等。

打开栅格数据时需要使用栅格工作空间工厂 RasterWorkspaceFactory，然后再使用 IRasterWorkspace 接口提供的打开栅格数据集方法即可打开一个栅格数据集，访问栅格数据的流程如图 8-1 所示。

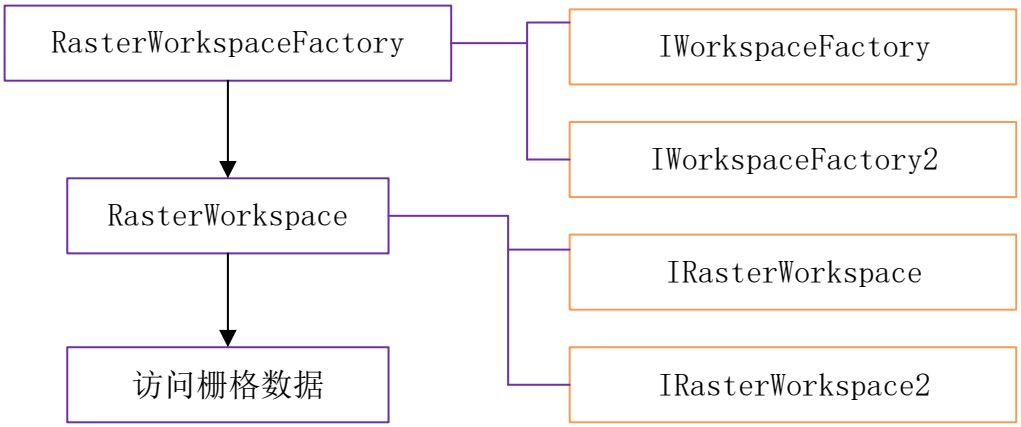


图 8-1　访问栅格数据流程

本例要实现的是创建一个具有三个波段的栅格数据集，并通过像素块操作的方式修改部分区域的像素值。具体步骤如下：

（1）新建“MapControlApplication”项目，为项目命名为“Test8”并设置保存位置。创建完成后，在默认生成的窗体菜单 menuStrip1 控件上添加一个一级菜单为“栅格操作”，在一级菜单“栅格操作”下添加二级菜单为“创建栅格数据集”，如图 8-2 所示。

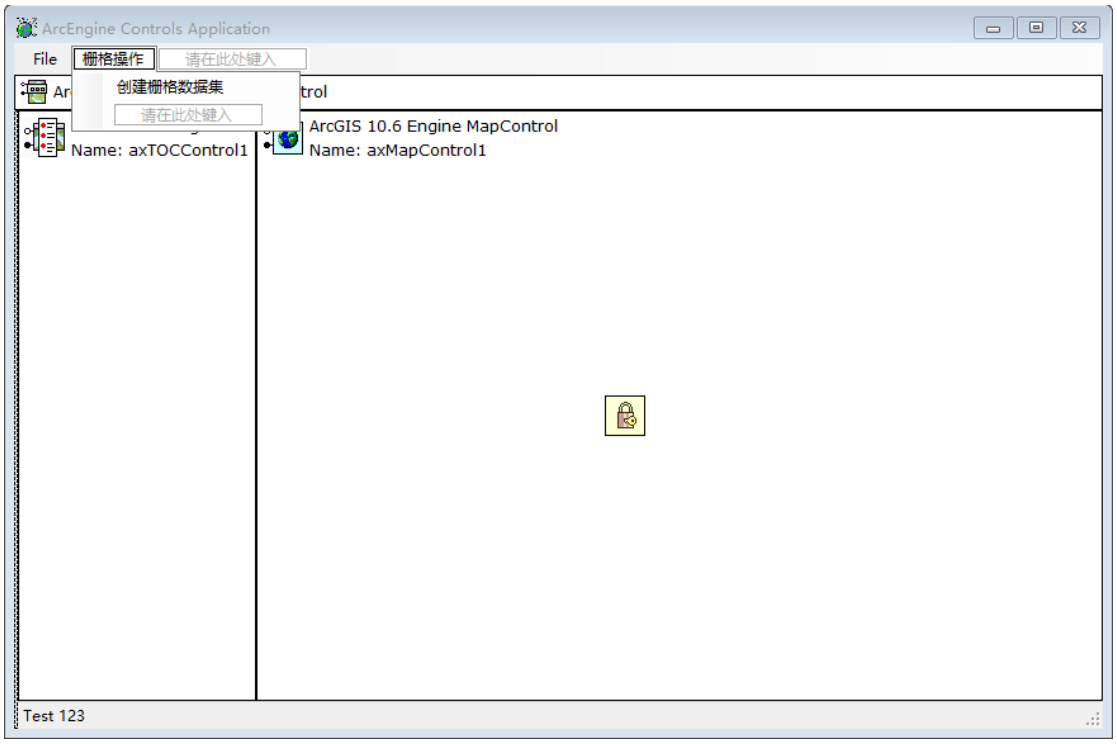


图 8-2　添加“创建栅格数据集”菜单

（2）为 Test8 项目添加新建项“Base Command”按钮命令，按钮命令类命名为“CmdCreatRaster”。选中项目“Test8”，右键点击【添加】→【新建项】，如图 8-3 所示。

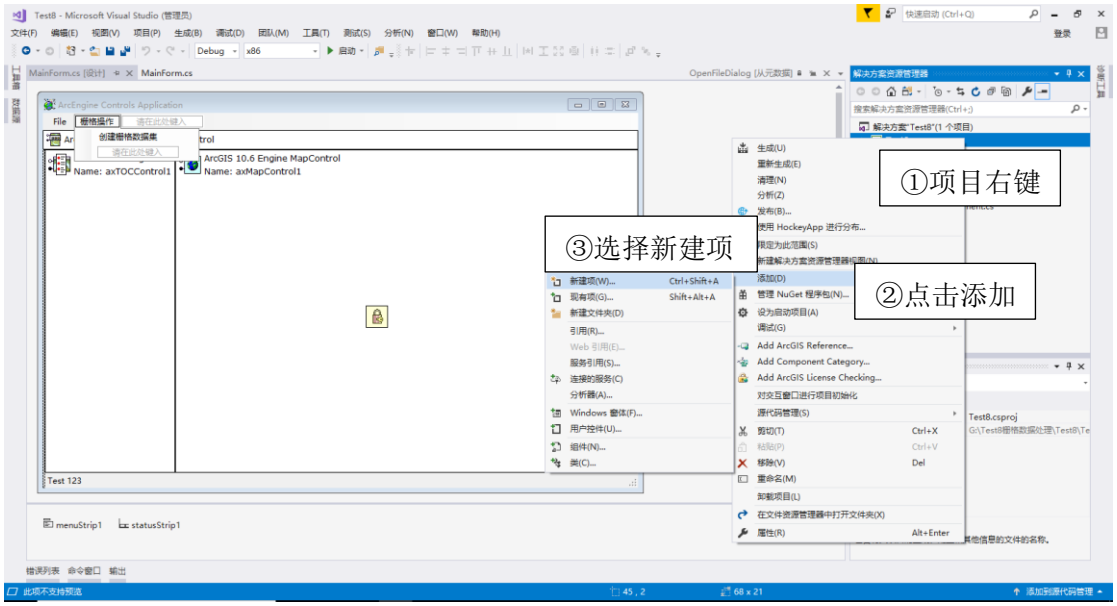


图 8-3　添加新建项

选择【ArcGIS】→【Extending ArcObjects】→【Base Command】，如图 8-4 所示。

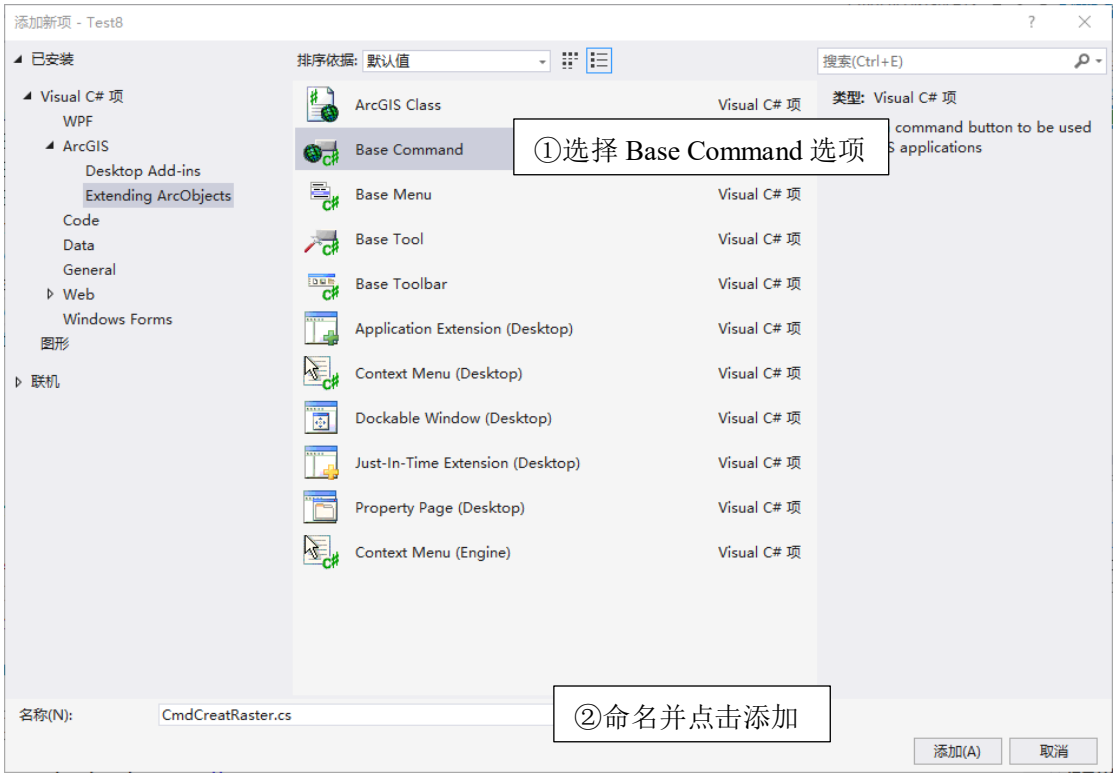


图 8-4　添加“Base Command”按钮命令类

（3）为按钮命令类“CmdCreatRaster”导入引用：

```
using System.Windows.Forms;
using ESRI.ArcGIS.DataSourcesRaster;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Carto;
```

为类“CmdCreatRaster”添加私有成员函数 CreatRasterDS，并修改 OnClick 事件响应函数代码：

| CmdCreatRaster.cs（节选） | 功能：创建栅格数据集并加载到当前地图 |
|---|---|

```
//创建并修改栅格数据集
private IRasterDataset CreatRasterDS(string filePath, string rasterName)
{
    //创建栅格工作工厂
    IRasterWorkspace2 rasterWorkspaceEx;
    IWorkspaceFactory pWorkspaceFactory = new
                                        RasterWorkspaceFactoryClass();
    rasterWorkspaceEx = pWorkspaceFactory.OpenFromFile(filePath, 0)
                                                as IRasterWorkspace2;
    IRasterStorageDef storageDef = new RasterStorageDef();
    storageDef.CompressionType =
                    esriRasterCompressionType.esriRasterCompressionJPEG;
```

```
IRasterDef rasterDef = new RasterDef();
IPoint Origin = new PointClass();
Origin.X = 0;
Origin.Y = 0;
//生成100*100的栅格数据
int ColumnCnt = 100, RowCnt = 100;
double sizex = 10, sizey = 10;
int numBands = 3;
IRasterDataset rasterDataset = null;
rasterDataset = rasterWorkspaceEx.CreateRasterDataset(rasterName,
                                    "TIFF", Origin, ColumnCnt, RowCnt, sizex,
                                    sizey, numBands, rstPixelType.PT_FLOAT);
IRaster pRaster = rasterDataset.CreateDefaultRaster() as IRaster;
//读取50*50的栅格数据（左上角部分）作为像素块
int Width = 50, Height = 50;
IPnt blocksize = new PntClass();
blocksize.SetCoords(Width, Height);
IPixelBlock3 pPixelBlock3 = pRaster.CreatePixelBlock(blocksize) as
                                                        IPixelBlock3;
IPnt pnt = new PntClass();
pnt.SetCoords(0, 0);
pRaster.Read(pnt, pPixelBlock3 as IPixelBlock);
System.Array pixels0 = (System.Array)pPixelBlock3.get_PixelData(0);
System.Array pixels1 = (System.Array)pPixelBlock3.get_PixelData(1);
System.Array pixels2 = (System.Array)pPixelBlock3.get_PixelData(2);
//修改像素块的像素值
for (int row = 0; row < Height; row++)
{
    for (int col = 0; col < Width; col++)
    {
        float value0 = 0, value1 = 0, value2 = 0;
        value0 = (float)Math.Abs(Math.Sin(row)) * row;
        value1 = (float)Math.Abs(Math.Sin(col)) * col;
        value2 = (float)Math.Abs(Math.Sin(row)) * col;
        pixels0.SetValue(Convert.ToByte(value0), col, row);
        pixels1.SetValue(Convert.ToByte(value1), col, row);
        pixels2.SetValue(Convert.ToByte(value2), col, row);
    }
}
pPixelBlock3.set_PixelData(0, pixels0);
pPixelBlock3.set_PixelData(1, pixels1);
pPixelBlock3.set_PixelData(2, pixels2);
//将像素块写回栅格数据集
IRasterEdit pRasterEdit = pRaster as IRasterEdit;
```

```
        pRasterEdit.Write(pnt, (IPixelBlock)pPixelBlock3);
        pRasterEdit.Refresh();
        return rasterDataset;
}
public override void OnClick()
{
        // TODO: Add RasterCmd.OnClick implementation
        string pRasterFileName = null;
        string pPath = null;
        string pFileName = null;
        //调用系统“保存文件”窗体
        SaveFileDialog saveFileDialog = new SaveFileDialog();
        saveFileDialog.Filter = " (*.tif)|*.tif ";
        saveFileDialog.Title = "选择栅格数据存放位置";
        saveFileDialog.FilterIndex = 1;
        if(saveFileDialog.ShowDialog()==DialogResult.OK)
        {
                //获取栅格数据的保存路径和栅格数据名称
                pRasterFileName = saveFileDialog.FileName;
                if (pRasterFileName == "")
                        return;
                //获取栅格数据的保存路径
                pPath = System.IO.Path.GetDirectoryName(pRasterFileName);
                //获取栅格数据的名称
                pFileName = System.IO.Path.GetFileName(pRasterFileName);
                IRasterDataset pRasterDataset = CreatRasterDS(pPath,pFileName);
                //将栅格数据集添加到当前地图中
                IRasterLayer pRasterLayer = new RasterLayerClass();
                pRasterLayer.CreateFromDataset(pRasterDataset);
                m_hookHelper.FocusMap.AddLayer((ILayer)pRasterLayer);
                m_hookHelper.ActiveView.Extent =
                                                m_hookHelper.ActiveView.FullExtent;
        }
}
```

（3）双击主窗体菜单 menuStrip1 控件二级菜单的“创建栅格数据集”，为其添加点击事件响应函数，代码如下：

| MainForm.cs（节选） | 功能：“创建栅格数据集”单击事件响应函数 |
|---|---|

```
private void toolStripMenuItem1_Click(object sender, EventArgs e)
{
        ICommand command = new CmdCreatRaster();
        command.OnCreate(m_mapControl.Object);
        command.OnClick();
}
```

（4）编译并运行程序，点击“创建栅格数据集”命令，运行效果如图 8-5 所示。

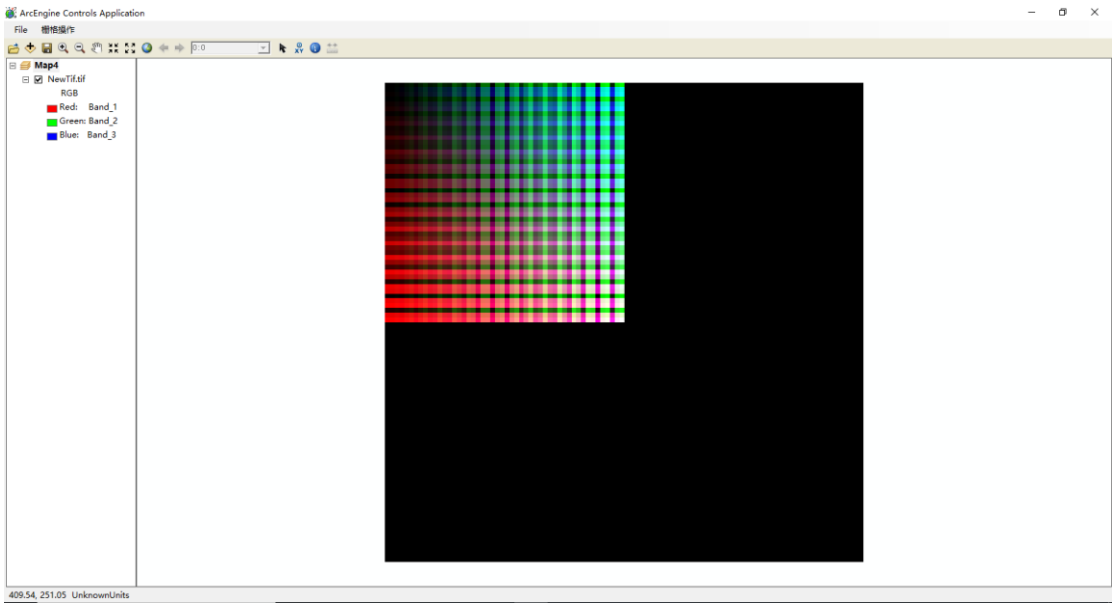


图 8-5　创建栅格数据集功能运行效果

## 8.2 实验目的

（1）了解栅格数据的结构，掌握栅格数据集的创建和渲染方法；

（2）掌握栅格数据的查询、统计和分析方法；

## 8.3 实验内容

（1）实现距离或方向栅格数据集的创建及渲染；

（2）实现栅格数据查询与统计；

（3）实现栅格数据等值线提取。

## 8.4 实验数据

见安装目录：

…\DeveloperKit10.2\Samples\data\AirportsAndGolf\

## 8.5 实验步骤

### 8.5.1 创建距离或方向栅格

（1）为项目添加“Windows 窗体”新建项，窗体类命名为“FrmEucDistance”，修改窗体“Text”属性为“欧氏距离栅格”，如图 8-6 所示。

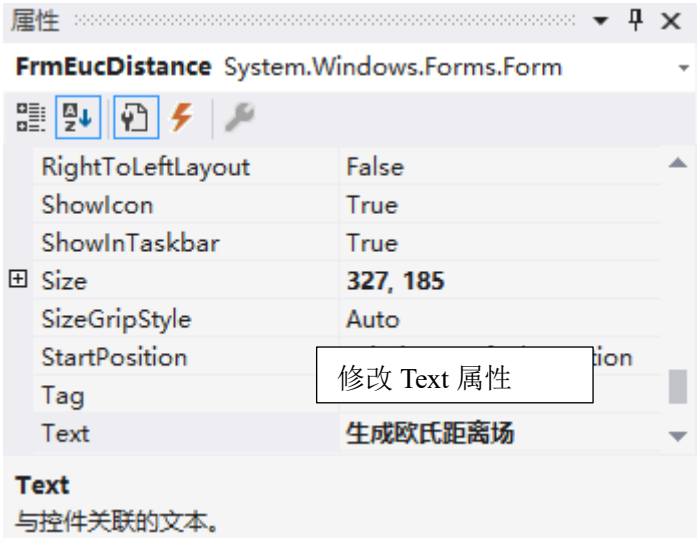


图 8-6　修该窗体 Text 属性

为窗体添加控件并调整页面布局，布局设计如图 8-7 所示。

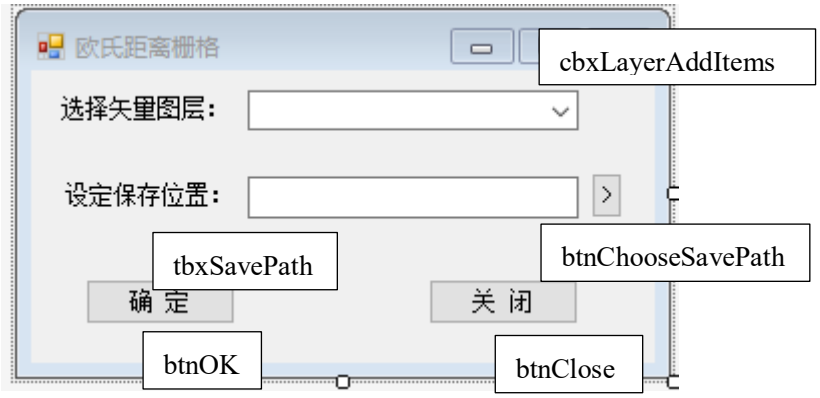


图 8-7　修改“欧氏距离栅格”窗体控件 Name 属性

（2）首先为窗体类“FrmEucDistance”导入引用：

```
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Geoprocessor;
using ESRI.ArcGIS.SpatialAnalystTools;
using ESRI.ArcGIS.Geodatabase;
```

```
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.DataSourcesRaster;
```

然后定义类私有成员变量：

```
private IHookHelper m_hookHelper = null;
private IMap m_map = null;
private IFeatureLayer m_FeatureLayer = null;
private string m_RasterFileName = null;
private string m_Path = null;
private string m_FileName = null;
```

再修改窗体的构造函数，代码如下：

| FrmEucDistance.cs（节选） | 功能：修改窗体构造函数 |
|---|---|

```
public FrmEucDistance(object hook)
{
    InitializeComponent();
    if (m_hookHelper == null)
        m_hookHelper = new HookHelperClass();
    m_hookHelper.Hook = hook;
    m_map = m_hookHelper.FocusMap;
}
```

（3）在窗体打开时，为窗体 Load 事件添加响应代码，将地图中的所有矢量图层添加进 cbxLayerAddItems 组合框控件，代码如下：

| FrmEucDistance.cs（节选） | 功能：窗体 Load 事件响应函数 |
|---|---|

```
private void FrmEucDistance_Load(object sender, EventArgs e)
{
    CbxFeatureLayersAddItems();
}
//将当前地图中所有矢量图层添加到组合框控件
private void CbxFeatureLayersAddItems()
{
    if (GetLayers() == null)
        return;
    IEnumLayer layers = GetLayers();
    layers.Reset();
    ILayer pLayer = layers.Next();
    while (pLayer != null)
    {
        if (pLayer is IFeatureLayer)
```

```
            {
                cbxLayerAddItems.Items.Add(pLayer.Name);
            }
            pLayer = layers.Next();
        }
}
//获取当前地图中所有矢量图层
private IEnumLayer GetLayers()
{
        UID uid = new UIDClass();
        //筛选矢量图层
        uid.Value = "{40A9E885-5533-11d0-98BE-00805F7CED21}";
        // IFeatureLayer
        if (m_map.LayerCount != 0)
        {
            IEnumLayer layers = m_map.get_Layers(uid, true);
            return layers;
        }
        return null;
}
```

（4）当改变 cbxLayerAddItems 控件中的图层时，触发 SelectedIndexChanged 事件，添加响应函数代码如下：

| FrmEucDistance.cs（节选） | 功能：选择图层变化时所触发事件响应函数 |
|---|---|

```
private void cbxLayerAddItems_SelectedIndexChanged(object sender,
                                                   EventArgs e)
{
        //如果CbxLayerAddItems控件中的图层不为空
        if (cbxLayerAddItems.SelectedItem != null)
        {
            string strRasterSelected = cbxLayerAddItems.SelectedItem.ToString();
            //获得选择图层
            m_FeatureLayer = GetFeatureLayer(strRasterSelected);
        }
}
//根据图层名获取图层
private IFeatureLayer GetFeatureLayer(string layerName)
{
        //通过所选择图层的名字来得到该图层
        if (GetLayers() == null)
            return null;
        IEnumLayer layers = GetLayers();
        layers.Reset();
        ILayer pLayer = null;
```

```
        while ((pLayer = layers.Next()) != null)
        {
            if (pLayer.Name == layerName)
                return pLayer as IFeatureLayer;
        }
        return null;
}
```

（5）为设置栅格数据集保存路径按钮 btnChooseSavePath 的点击事件 OnClick 添加响应代码如下：

| FrmEucDistance.cs（节选） | 功能：设定栅格数据集保存路径和文件名 |
|---|---|

```
private void btnChooseSavePath_Click(object sender, EventArgs e)
{
    //调用系统“保存文件”窗体
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "栅格文件（*.*）|*.tif|(*.tif)|*.tif";
    saveFileDialog.Title = "选择欧式距离栅格存放位置";
    saveFileDialog.FilterIndex = 1;
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        //获取栅格数据的保存路径和栅格数据名称
        m_RasterFileName = saveFileDialog.FileName;
        if (m_RasterFileName == "")
            return;
        //获取栅格数据的保存路径
        m_Path = System.IO.Path.GetDirectoryName(m_RasterFileName);
        //获取栅格数据的名称
        m_FileName = System.IO.Path.GetFileName(m_RasterFileName);
        //在tbxSavePath中显示路径和名称
        tbxSavePath.Text = m_RasterFileName;
    }
}
```

（6）点击确定 btnOK 按钮，生成欧氏距离栅格图层并渲染；点击关闭（btnClose）按钮关闭窗体，代码如下：

| FrmEucDistance.cs（节选） | 功能：生成并渲染欧氏距离栅格 |
|---|---|

```
private void btnOK_Click(object sender, EventArgs e)
{
    Geoprocessor GP = new Geoprocessor();
    GP.OverwriteOutput = true;
    //使用GeoProcessor工具生成欧氏距离栅格
    EucDistance eucDist = new EucDistance(m_FeatureLayer,
                                          m_RasterFileName);
```

```
        GP.Execute(eucDist,null);
        //pRasterLayer是生成的欧氏距离栅格图层
        IRasterLayer pRasterLayer = new RasterLayerClass();
        IWorkspaceFactory pWorkspaceFactory = new
                                                RasterWorkspaceFactoryClass();
        IRasterWorkspace pRasterWorkspace = (IRasterWorkspace)
                                    pWorkspaceFactory.OpenFromFile(m_Path, 0);
        IRasterDataset pRasterDataset = pRaster
                                        Workspace.OpenRasterDataset(m_FileName);
        pRasterLayer.CreateFromDataset(pRasterDataset);
        //渲染生成的欧氏距离场栅格图层
        RasterStretchColorMapRender(pRasterLayer);
        m_hookHelper.FocusMap.AddLayer(pRasterLayer);
        m_hookHelper.ActiveView.Refresh();
}
//栅格图层渲染函数
public void RasterStretchColorMapRender(IRasterLayer pRasterlayer)
{
        try
        {
                IRaster pRaster = pRasterlayer.Raster;
                int intTransPValue = 30;
                IColor pFromColor = new RgbColorClass();
                //Red + (0x100 * Green) + (0x10000 * Blue);
                pFromColor.RGB = 255 + 0x100 * 255;
                IColor pToColor = new RgbColorClass();
                pToColor.RGB = 0x10000 * 255;
                //新建栅格颜色拉伸渲染器
                IRasterStretchColorRampRenderer pStretchRender =
                        (IRasterStretchColorRampRenderer)pRasterlayer.Renderer;
                IRasterRenderer pRasterRender = default(IRasterRenderer);
                pRasterRender = (IRasterRenderer)pStretchRender;
                pRasterRender.Raster = pRaster;
                pRasterRender.Update();
                IAlgorithmicColorRamp pColorRamp = new AlgorithmicColorRamp();
                pColorRamp.Size = 255;
                pColorRamp.FromColor = pFromColor;
                pColorRamp.ToColor = pToColor;
                bool outvalue = true;
                pColorRamp.CreateRamp(out outvalue);
                pStretchRender.BandIndex = 0;
                pStretchRender.ColorRamp = pColorRamp;
                if (intTransPValue > 0)
                {
```

```
                IRasterDisplayProps pRRenProp =
                                        (IRasterDisplayProps)pStretchRender;
                pRRenProp.TransparencyValue = intTransPValue;
            }
            pRasterRender.Update();
        }
        catch (Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
}
private void btnClose_Click(object sender, EventArgs e)
{
    this.Close();
}
```

对栅格数据集进行渲染用到的栅格数据集渲染对象（如图 8-8 所示）主要包括：（a）RGB 渲染器 RasterRGBRenderer 允许以“红、绿、蓝”合成方式组合多个波段，例如彩色航空影像；（b）唯一值渲染器 RasterUniqueValueRenderer 用于使用随机颜色显示栅格图层的每个值，例如专题栅格图层可显示土壤类型或者土地利用的离散类别；（c）分类颜色表渲染器 RasterClassifyColorRampRenderer 要求栅格数据集具有颜色映射表，否则要使用“添加颜色映射表”工具从其他栅格数据集中添加颜色映射表，或者直接导入一个色彩映射表（.clr）或者.act 文件，或者使用“唯一值”渲染器的颜色映射表；（d）拉伸颜色渲染器 RasterStretchColorRampRenderer 用于以平滑渐变的颜色显示连续的栅格像元值，使用此渲染器显示单波段或者连续数据，例如影像、航空摄影或者高程模型中的数据；（e）离散颜色渲染器 RasterDiscreteColorRenderer 可以使用一种随机颜色来显示栅格数据集中的值，该渲染器只能用于整型栅格数据集，而且不会生成图例。

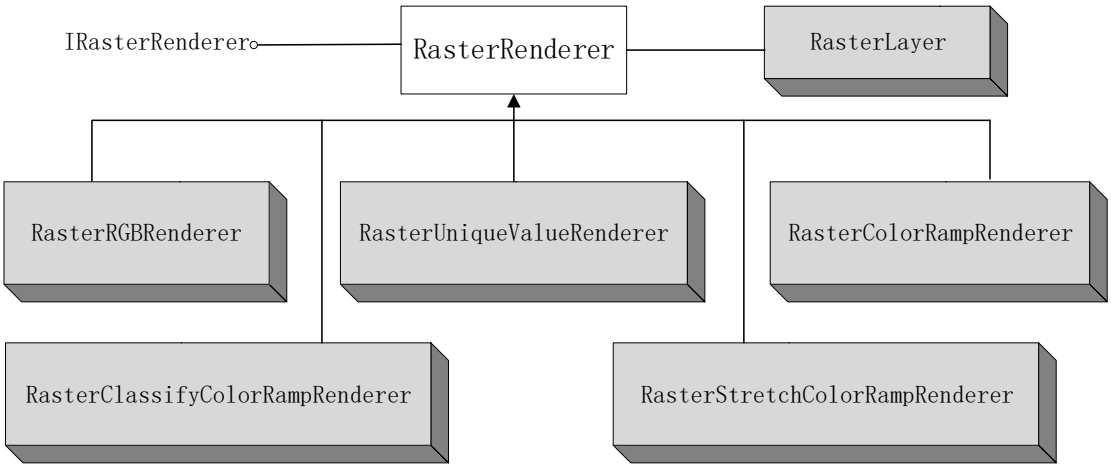


图 8-8　栅格数据集渲染对象

（7）为主窗体菜单 menuStripe1 添加“欧氏距离栅格”二级菜单，如图 8-9 所示。

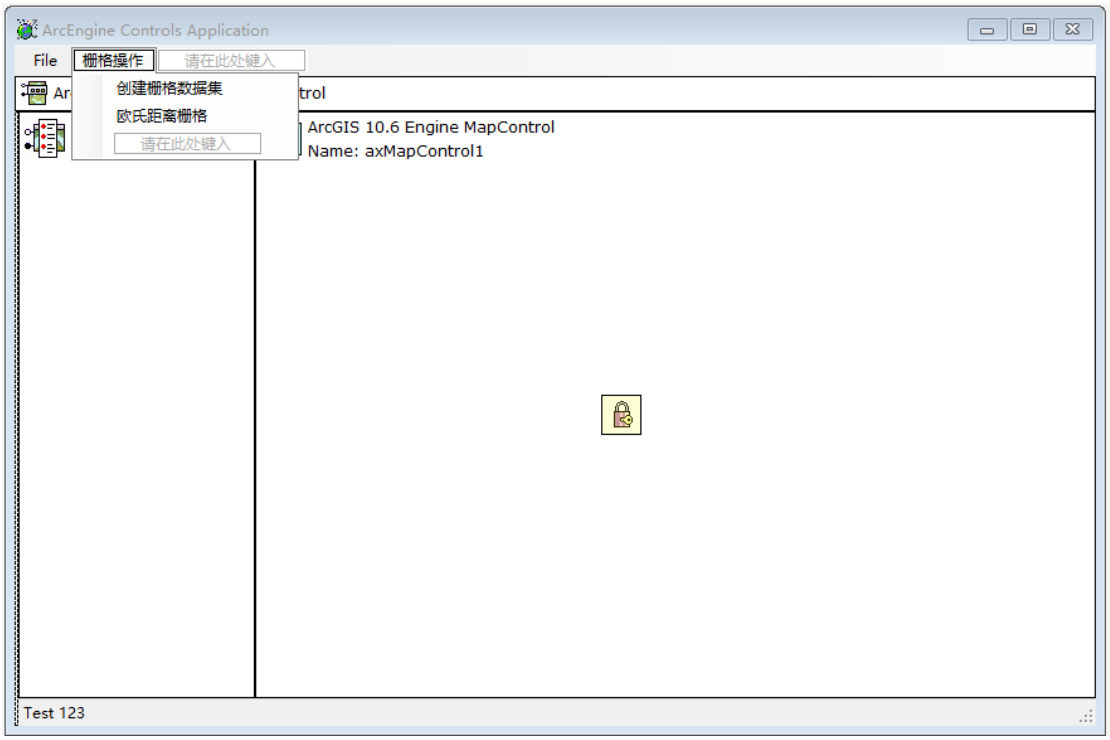


图 8-9　添加“欧氏距离栅格”二级菜单

双击此“欧氏距离栅格”二级菜单，调用窗体“FrmEucDistance”代码如下：

| MainForm.cs（节选） | 功能：调用“欧氏距离栅格”窗体 |
|---|---|

```
private void 欧氏距离栅格ToolStripMenuItem_Click(object sender, EventArgs e)
{
    FrmEucDistance frmEucDistance = new
                                FrmEucDistance(m_mapControl.Object);
    frmEucDistance.ShowDialog();
```

```
}
```

（8）编译并运行程序，添加实验数据后，打开“欧氏距离栅格”窗体选择矢量图层、设置保存位置，如图 8-10 所示。

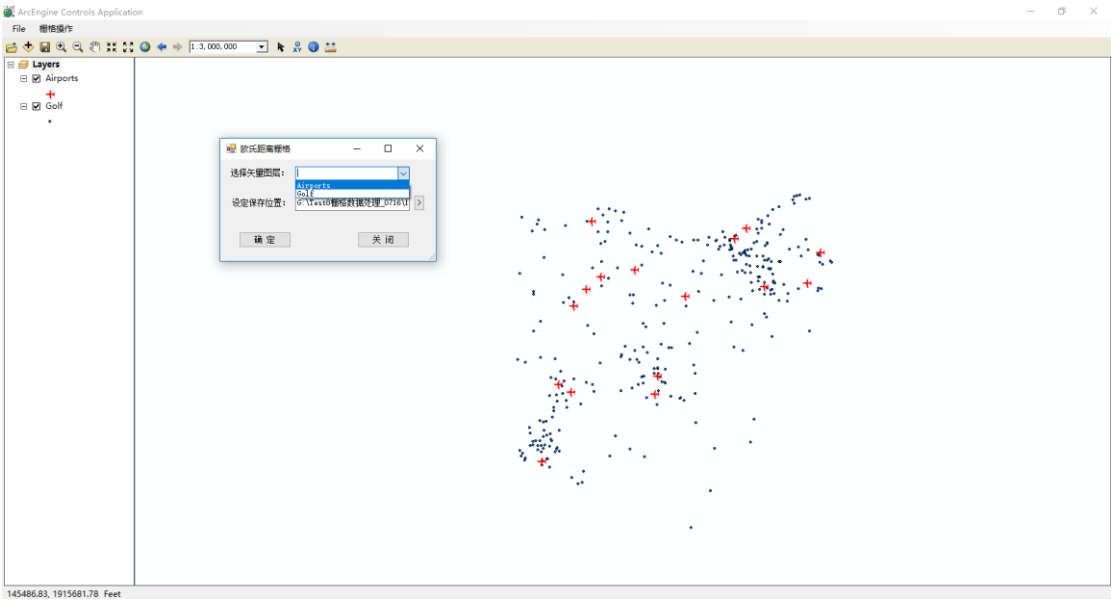


图 8-10　选择操作图层并设置保存位置

点击确定，程序运行结果如图 8-11 所示。

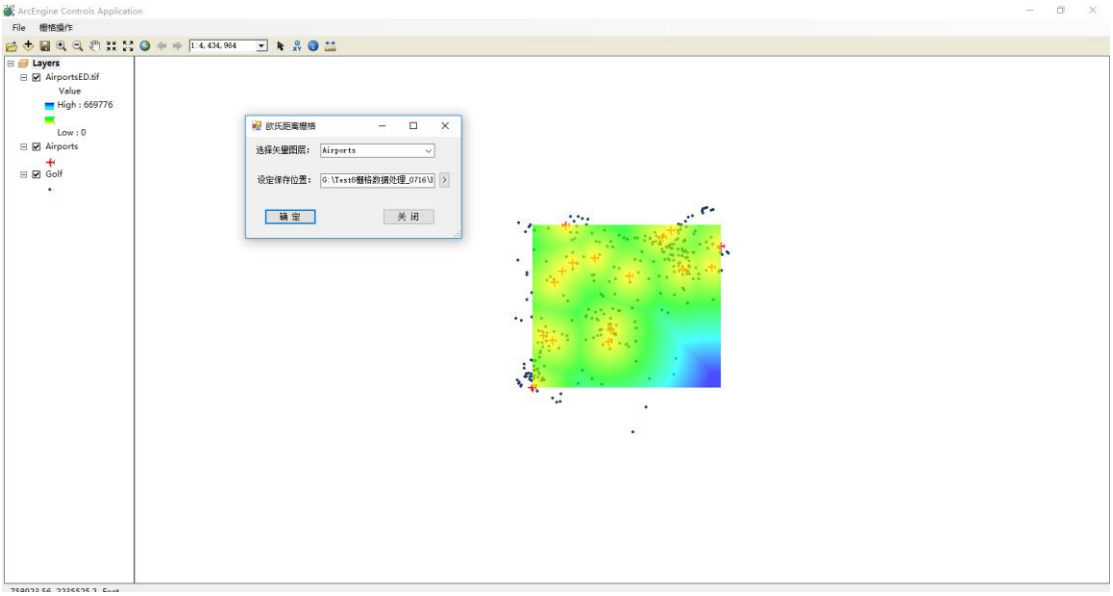


图 8-11　欧式距离栅格功能运行效果

## 8.5.2 查询栅格属性值工具

（1）为 Test8 项目添加“Window 窗体”新建项用于选择图层以供栅格像元属性值查询，名称为“FrmSelectRasterLayer”，并修改该窗体“Text”属性为“选

择栅格图层”，为窗体类“FrmSelectRasterLayer”添加窗体控件并修改各控件的 Name 属性，具体界面设计如图 8-12 所示。


图 8-12 “选择栅格图层”界面设计图

在该窗体代码页导入引用：

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

添加类私有成员变量：

```
private IHookHelper m_hookHelper = null;
private IMap m_map = null;
private IRasterLayer m_RasterLayer = null;
```

添加只读属性“IRasterLayer rasterLayer”，用于获取选择到的栅格图层；修改该窗体的构造函数，代码如下：

| FrmSelectRasterLayer.cs（节选） | 功能：窗体类属性和构造函数 |
|---|---|

```
//只读属性获取选择图层
public IRasterLayer rasterLayer
{
    get { return m_RasterLayer; }
}
public FrmSelectRasterLayer(IHookHelper hookHelper)
{
    InitializeComponent();
    m_hookHelper = hookHelper;
    m_map = hookHelper.FocusMap;
}
```

（2）将当前地图中的所有栅格图层添加进 cbxAddRasterLayers 控件，这个工作要在窗体 Load 事件中完成，代码如下：

| FrmSelectRasterLayer.cs（节选） | 功能：栅格图层选择窗体 Load 事件 |
|---|---|

```
private void FrmSelectRasterLayer_Load(object sender, EventArgs e)
```

```
{
    CbxRasterLayersAddItems();
}
//将当前地图中的所有栅格图层添加到组合框控件
private void CbxRasterLayersAddItems()
{
    if (GetLayers() == null)
        return;
    IEnumLayer layers = GetLayers();
    layers.Reset();
    ILayer pLayer = layers.Next();
    while (pLayer != null)
    {
        if (pLayer is IRasterLayer)
        {
            cbxAddRasterLayers.Items.Add(pLayer.Name);
        }
        pLayer = layers.Next();
    }
}
//获得当前地图的所有栅格图层
private IEnumLayer GetLayers()
{
    UID uid = new UIDClass();
    //筛选栅格图层
    uid.Value = "{D02371C7-35F7-11D2-B1F2-00C04F8EDEFF}";
    // IRasterLayer
    if (m_map.LayerCount != 0)
    {
        IEnumLayer layers = m_map.get_Layers(uid, true);
        return layers;
    }
    return null;
}
```

（3）双击“确定”和“取消”按钮，添加按钮单击事件响应函数，当单击“确定”按钮时，获取选中的栅格图层，代码如下：

| FrmSelectRasterLayer.cs（节选） | 功能：获取选中的栅格图层 |
|---|---|

```
private void btnOK_Click(object sender, EventArgs e)
{
    //如果cbxAddRasterLayers控件中的图层不为空
    if (cbxAddRasterLayers.SelectedItem != null)
    {
        string strRasterSelected = cbxAddRasterLayers.SelectedItem.ToString();
```

```
            m_RasterLayer = GetRasterLayer(strRasterSelected);
        }
        this.Close();
}
private IRasterLayer GetRasterLayer(string layerName)
{
        //通过所选择图层的名字来得到该图层
        if (GetLayers() == null)
            return null;
        IEnumLayer layers = GetLayers();
        layers.Reset();
        ILayer pLayer = null;
        while ((pLayer = layers.Next()) != null)
        {
            if (pLayer.Name == layerName)
                return pLayer as IRasterLayer;
        }
        return null;
}
private void btnClose_Click(object sender, EventArgs e)
{
        m_RasterLayer = null;
        this.Close();
}
```

（4）为 Test8 项目添加新建项“Base Tool”工具，工具类命名为“ToolPixValue”。选择项目 Test8，右键点击【添加】→【新建项】，具体操作如图 8-13 所示。

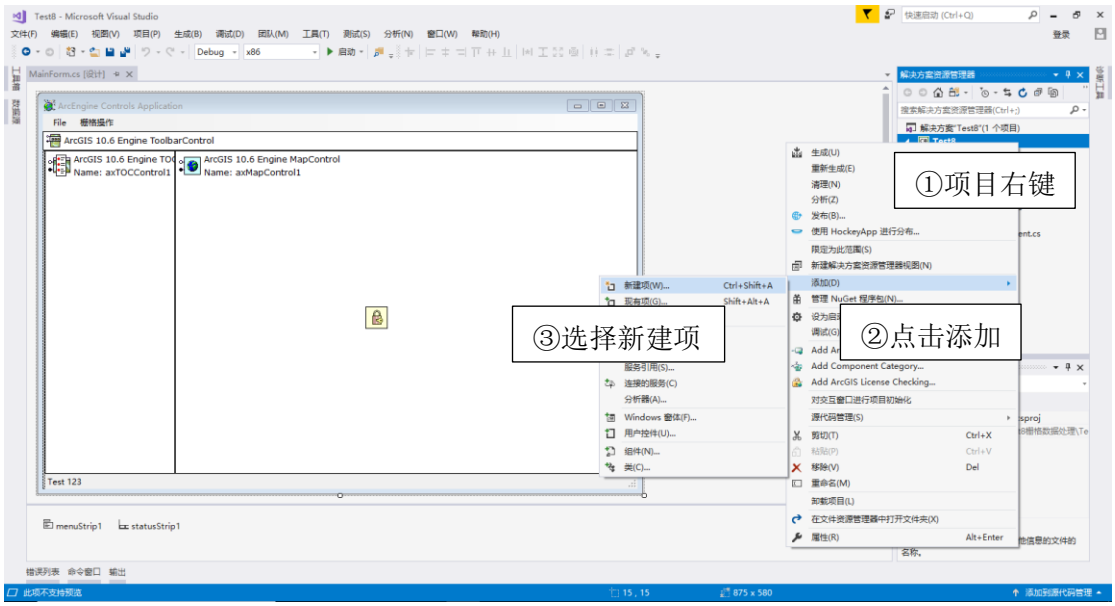


图 8-13 添加工具新建项

选择【ArcGIS】→【Extending ArcObjects】→【Base Tool】，具体操作如图 8-14 所示。

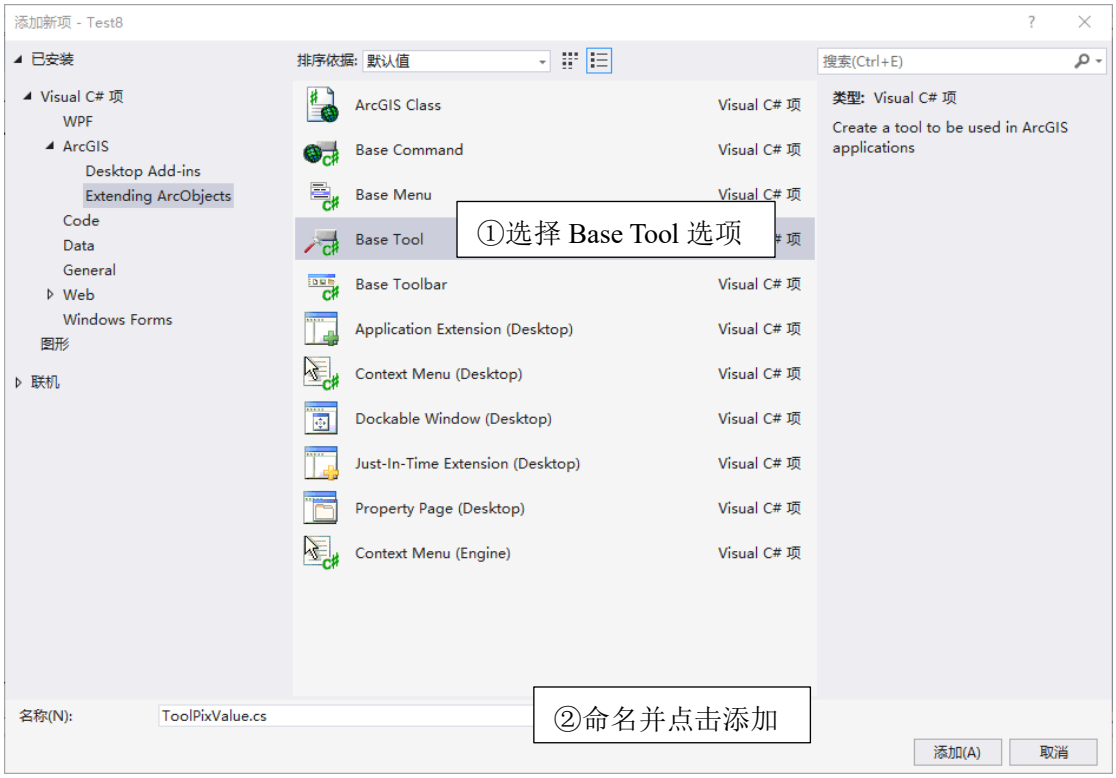


图 8-14　添加“Base Tool”工具类

（5）首先为“ToolPixValue.cs”导入引用：

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

在 OnClick 重载事件中添加代码调用“选择栅格图层”窗体并获取需要待查询的栅格图层，代码如下：

| ToolPixValue.cs（节选） | 功能：调用窗体并获取待查询的栅格图层 |
|---|---|

```
public override void OnClick()
{
    // TODO: Add ToolPixValue.OnClick implementation
    FrmSelectRasterLayer FrmSelectRasterLayer = new
                                    FrmSelectRasterLayer(m_hookHelper);
    FrmSelectRasterLayer.ShowDialog();
    m_rasterLayer = FrmSelectRasterLayer.rasterLayer;
```

```
}
```

（6）再添加栅格像元查询的类私有成员函数 GetPixValue；在 OnMouseDown 重载事件中调用 GetPixValue 函数，代码如下：

| ToolPixValue.cs（节选） | 功能：查询栅格像元值 |
|---|---|

```
//根据点坐标查询栅格像元值
public string    GetPixValue(IRasterLayer pRasterlayer, IPoint pt)
{
      IRaster pRaster = pRasterlayer.Raster;
      IRasterProps rasterProps = (IRasterProps)pRaster;
      IEnvelope extent = rasterProps.Extent;
      IRelationalOperator pRO = pt as IRelationalOperator;
      if (!pRO.Within(extent)) //点坐标不在栅格范围内
            return null;
      IRaster2 pRaster2 = pRaster as IRaster2;
      //根据点坐标查询栅格行列号
      int row = pRaster2.ToPixelRow(pt.Y);
      int col = pRaster2.ToPixelColumn(pt.X);
      string strInquirePVResult = "Inquire Pixel Value Result:\n";
      IRaster2 r2 = pRasterlayer.Raster as IRaster2;
      IRasterDataset rasterDataset = r2.RasterDataset;
      IRasterBandCollection rasterBands = (IRasterBandCollection)rasterDataset;
      int i = 0, cntBands = rasterBands.Count;
      //根据点坐标查询栅格各波段像元值
      for (i=0;i<cntBands;i++)
      {
            strInquirePVResult += rasterBands.Item(i).Bandname + ": ";
            strInquirePVResult += pRaster2.GetPixelValue(i,col,row).ToString()
                                                                  + "\n";
      }
      return strInquirePVResult;
}
//鼠标按下查询栅格像元值
public override void OnMouseDown(int Button, int Shift, int X, int Y)
{
      //// TODO:    Add ToolPixValue.OnMouseDown implementation
      IPoint pPoint = m_hookHelper.ActiveView.ScreenDisplay.
                                          DisplayTransformation.ToMapPoint(X, Y);
      string msg = GetPixValue(m_hookHelper.FocusMap.get_Layer(0) as
                                                      IRasterLayer, pPoint);
      MessageBox.Show(msg);
}
```

（7）在主窗体的菜单 menuStripe1 上添加“查询栅格属性值”二级菜单，如

图 8-15 所示。

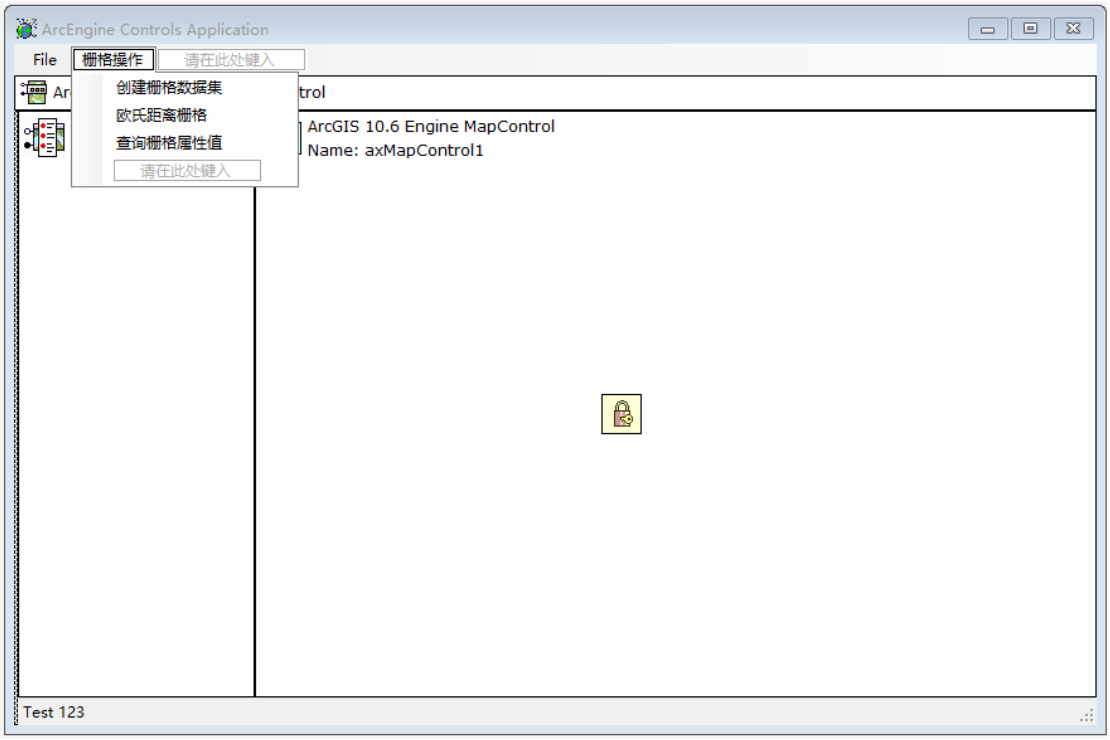


图 8-15　添加“查询栅格属性值”二级菜单

双击“查询栅格属性值”菜单，为其 Click 事件添加代码，代码如下：

| MainForm.cs（节选） | 功能：调用 ToolPixValue 按钮 |
|---|---|

```
private void 查询栅格属性值ToolStripMenuItem_Click(object sender,
                                                    EventArgs e)
{
    ICommand command = new ToolPixValue();
    command.OnCreate(m_mapControl.Object);
    m_mapControl.CurrentTool = command as ITool;
}
```

（8）编译并运行程序，对前面生成的栅格图层，进行栅格查询操作，结果如图 8-16 所示。

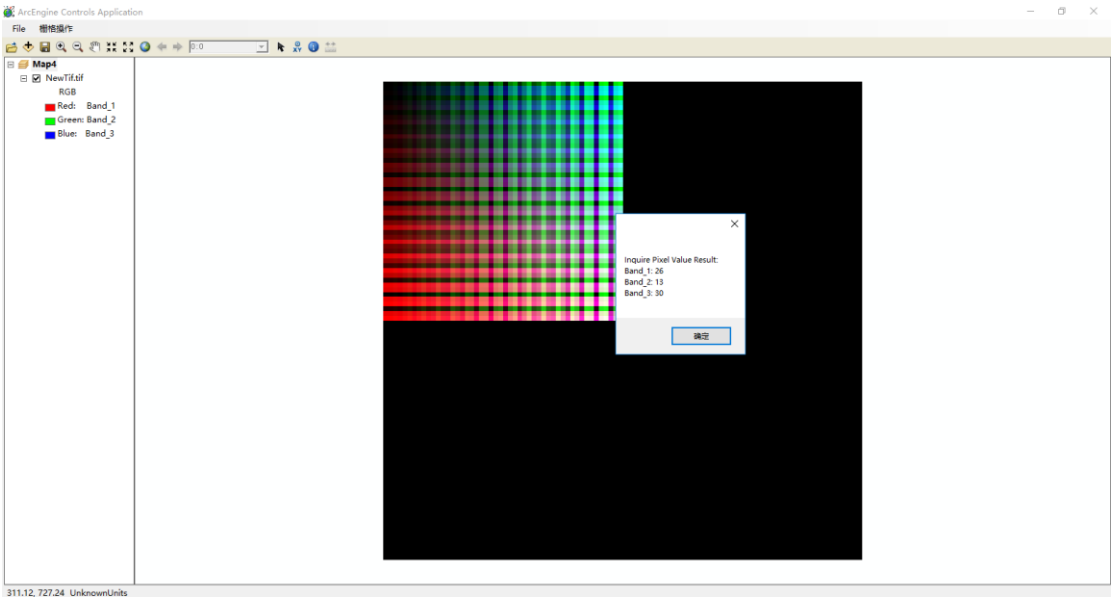


图 8-16　栅格查询功能运行效果

## 8.5.3 栅格数据集统计

（1）为项目添加“Windows 窗体”新建项，窗体类命名为“FrmStatistics”，修改窗体“Text”属性为“栅格数据集统计”。为该窗体添加控件并调整页面布局，窗体界面设计如图 8-17 所示。


图 8-17　修改“栅格数据集统计”窗体控件 Name 属性

将控件 rtbxResult 的“ReadOnly”属性改为“True”，如图 8-18 所示。

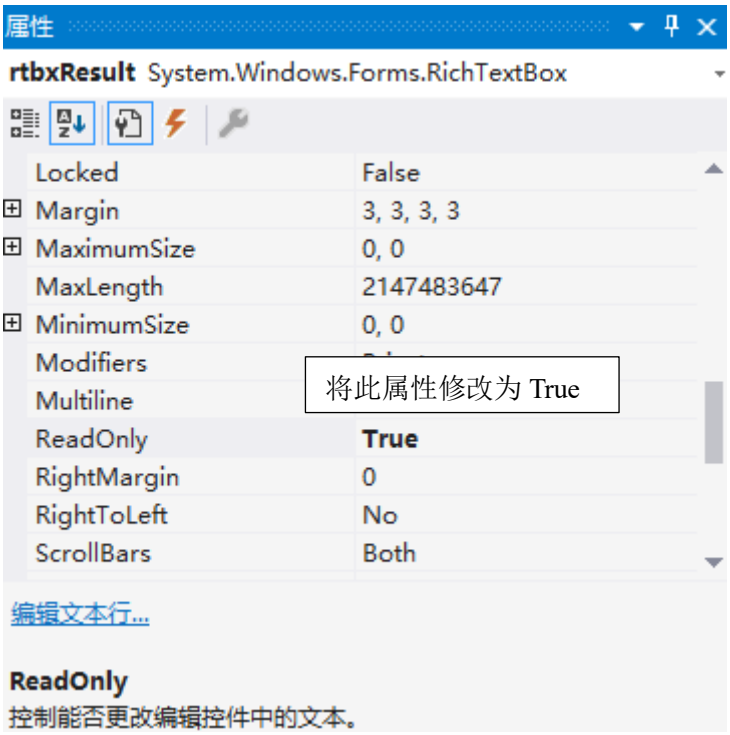


图 8-18 修改 rtbxResult 控件的 ReadOnly 属性

（2）首先在“栅格数据集统计”窗体代码页导入引用：

```
using System.Windows.Forms;
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.DataSourcesRaster;
```

然后定义类私有成员变量：

```
private IHookHelper m_hookHelper = null;
private IMap m_map = null;
private IRasterLayer m_RasterLayer = null;
```

再修改窗体类“FrmStatistics”的构造函数，代码如下：

| FrmStatistics.cs（节选） | 功能：窗体类构造函数 |
|---|---|

```
public FrmStatistics(object hook)
{
    InitializeComponent();
    if (m_hookHelper == null)
        m_hookHelper = new HookHelperClass();
```

```
        m_hookHelper.Hook = hook;
        m_map = m_hookHelper.FocusMap;
}
```

（3）将当前地图中的所有栅格图层添加进 cbxAddRasterLayers 控件，这个工作要在窗体 Load 事件中完成，代码如下：

| FrmStatistics.cs（节选） | 功能：窗体 Load 事件 |
|---|---|

```
private void FrmStatistics_Load(object sender, EventArgs e)
{
        CbxRasterLayersAddItems();
}
//将当前地图中的所有栅格图层添加到组合框控件
private void CbxRasterLayersAddItems()
{
        if (GetLayers() == null)
                return;
        IEnumLayer layers = GetLayers();
        layers.Reset();
        ILayer pLayer = layers.Next();
        while (pLayer != null)
        {
                if (pLayer is IRasterLayer)
                {
                        cbxAddRasterLayers.Items.Add(pLayer.Name);
                }
                pLayer = layers.Next();
        }
}
//获得当前地图的所有栅格图层
private IEnumLayer GetLayers()
{
        UID uid = new UIDClass();
        //筛选栅格图层
        uid.Value = "{D02371C7-35F7-11D2-B1F2-00C04F8EDEFF}";
        // IRasterLayer
        if (m_map.LayerCount != 0)
        {
                IEnumLayer layers = m_map.get_Layers(uid, true);
                return layers;
        }
        return null;
}
```

（4）当选择或者改变 cbxAddRasterLayers 控件中的图层时触发

SelectedIndexChanged 事件，添加响应函数代码如下：

| FrmStatistics.cs（节选） | 功能：选择图层变化时所触发事件响应函数 |
|---|---|

```
private void cbxAddRasterLayers_SelectedIndexChanged(object sender,
                                                     EventArgs e)
{
    //如果cbxAddRasterLayers控件中的图层不为空
    if (cbxAddRasterLayers.SelectedItem != null)
    {
        string strRasterSelected = cbxAddRasterLayers.SelectedItem.ToString();
        m_RasterLayer = GetRasterLayer(strRasterSelected);
    }
}
//通过名字来得到图层
private IRasterLayer GetRasterLayer(string layerName)
{
    if (GetLayers() == null)
        return null;
    IEnumLayer layers = GetLayers();
    layers.Reset();
    ILayer pLayer = null;
    while ((pLayer = layers.Next()) != null)
    {
        if (pLayer.Name == layerName)
            return pLayer as IRasterLayer;
    }
    return null;
}
```

（5）当点击“确定”按钮时，对选中的栅格图层进行统计，统计结果显示在 rtbxResult 控件中；添加栅格图层统计函数，并在“确定”按钮的 Click 事件中调用；点击“关闭”按钮时窗体关闭。代码如下：

| FrmStatistics.cs（节选） | 功能：栅格统计函数和按钮 Click 事件 |
|---|---|

```
private void btnOK_Click(object sender, EventArgs e)
{
    RasterStistics(m_RasterLayer);
}
//栅格数据集数据统计
private void RasterStistics(IRasterLayer rLayer)
{
    IRaster2 r2 = rLayer.Raster as IRaster2;
    IRasterDataset rasterDataset = r2.RasterDataset;
    IRasterBandCollection rasterBands = (IRasterBandCollection)rasterDataset;
```

```
        IEnumRasterBand enumRasterBand = rasterBands.Bands;
        string sRasterStisticsResult = "Raster Statistics Result:\n";
        //统计栅格图层各波段数据
        IRasterBand rasterBand = enumRasterBand.Next();
        while (rasterBand != null)
        {
            bool tmpBool;
            rasterBand.HasStatistics(out tmpBool);
            if (!tmpBool)
                rasterBand.ComputeStatsAndHist();
            sRasterStisticsResult += GetRasterStistics(rasterBand) + "\n";
            rasterBand = enumRasterBand.Next();
        }
        //统计结果在rtbxResult中显示
        rtbxResult.Text = sRasterStisticsResult;
    }
    //栅格某波段数据统计
    private string GetRasterStistics(IRasterBand rasterBand)
    {
        IRasterStatistics rasterStatistics = rasterBand.Statistics;
        string statisticsResult;
        statisticsResult = "" + rasterBand.Bandname + " Mean is：" +
                                rasterStatistics.Mean.ToString() + " SD is：" +
                                rasterStatistics.StandardDeviation.ToString();
        return statisticsResult;
    }
    private void btnClose_Click(object sender, EventArgs e)
    {
        this.Close();
    }
```

（6）在主窗体的菜单 menuStrip1 控件中添加“栅格数据集统计”二级菜单项，如图 8-19 所示。

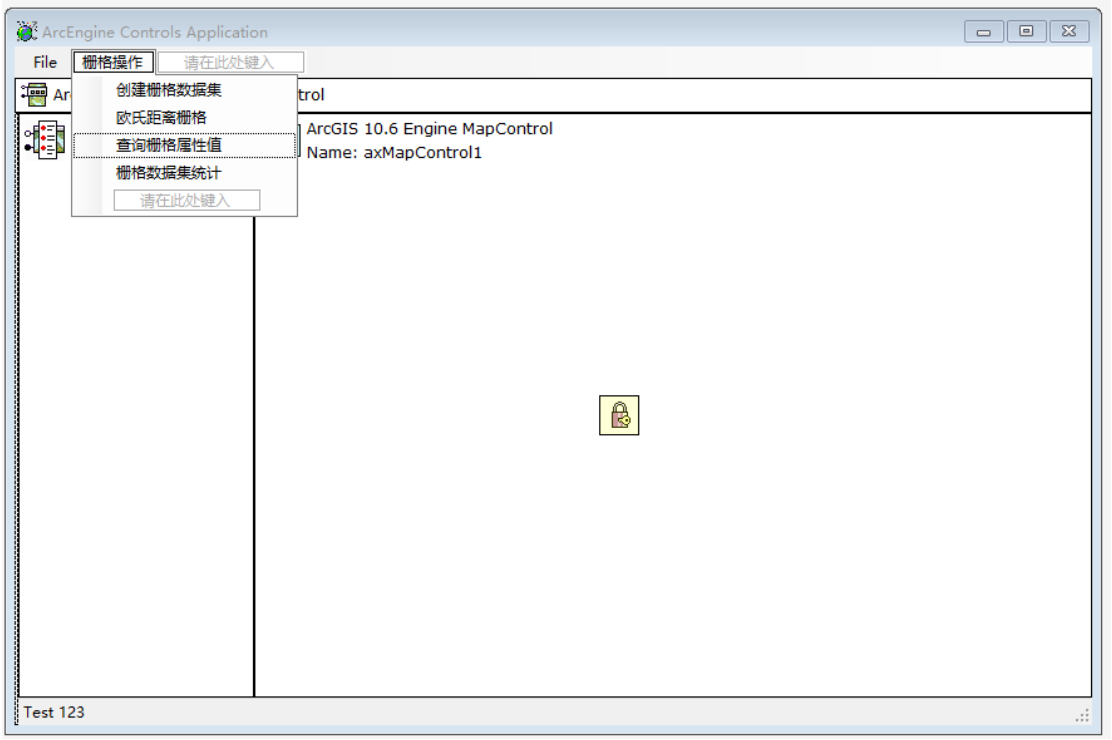


图 8-19　添加“栅格数据集统计”二级菜单项

双击此命令为 Click 事件添加代码调用“栅格数据集统计”窗体，代码如下：

| MainForm.cs（节选） | 功能：调用“栅格数据集统计”窗体 |
|---|---|

```
private void 栅格数据集统计ToolStripMenuItem_Click(object sender, EventArgs e)
{
    FrmStatistics frmStatistics = new FrmStatistics(m_mapControl.Object);
    frmStatistics.ShowDialog();
}
```

（7）编译并运行程序，对前面生成的栅格数据集图层进行统计，程序运行及统计结果如图 8-20 所示。

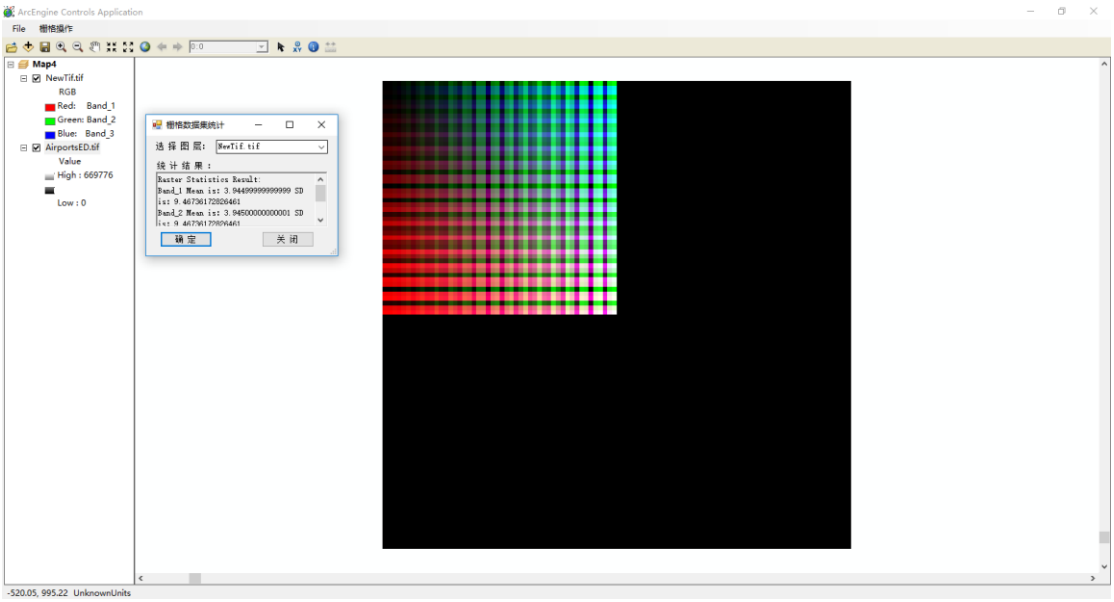

图 8-20　栅格数据集统计功能运行效果

## 8.5.4 提取栅格等值线

（1）为项目添加新建项 Window 窗体，命名为“FrmCreatCoutour.cs”，修改窗体“Text”属性为“提取等值线”，为窗体添加控件并调整界面布局，界面设计如图 8-21 所示。

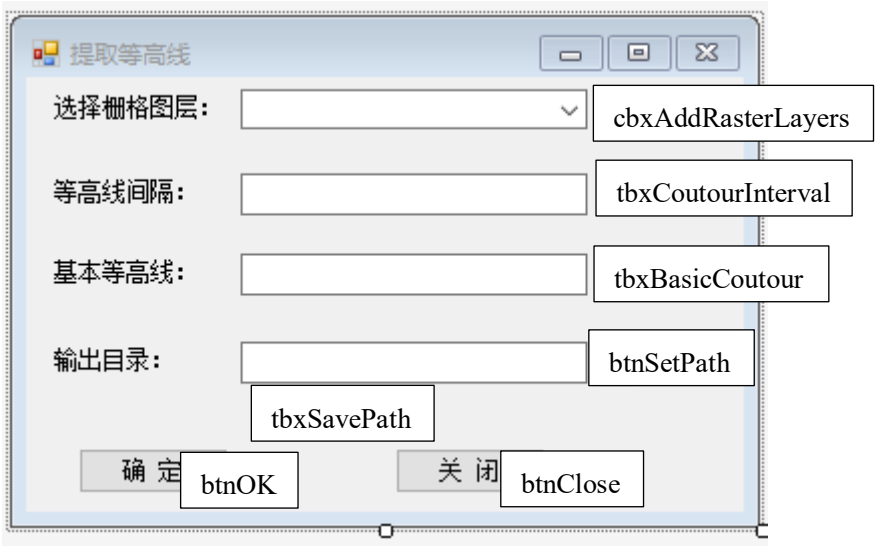


图 8-21 修改“提取等值线”窗体控件 Name 属性

（2）首先在“FrmCreatCoutour.cs”窗体代码页开头导入引用：

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

```
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.GeoAnalyst;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.DataSourcesFile;
```

然后定义类“FrmCreatCoutour”私有成员变量：

```
private IHookHelper m_hookHelper = null;
private IMap m_map = null;
private IRasterLayer m_RasterLayer = null;
private string m_ShapeFileName = null;
private string m_Path = null;
private string m_FileName = null;
```

再修改窗体类的构造函数，代码如下：

| FrmCreatCoutour.cs（节选） | 功能：窗体类构造函数 |
|---|---|

```
public FrmCreatCoutour(object hook)
{
    InitializeComponent();
    if (m_hookHelper == null)
        m_hookHelper = new HookHelperClass();
    m_hookHelper.Hook = hook;
    m_map = m_hookHelper.FocusMap;
}
```

（3）在窗体打开时，已经将当前地图中的所有栅格图层添加进组合框 cbxAddRasterLayers 控件，添加窗体 Load 事件响应函数代码如下：

| FrmCreatCoutour.cs（节选） | 功能：窗体 Load 事件 |
|---|---|

```
private void FrmCreatCoutour _Load(object sender, EventArgs e)
{
    CbxRasterLayersAddItems();
}
//将当前地图中所有栅格图层添加到组合框
private void CbxRasterLayersAddItems()
{
    if (GetLayers() == null)
        return;
    IEnumLayer layers = GetLayers();
    layers.Reset();
    ILayer pLayer = layers.Next();
```

```
        while (pLayer != null)
        {
            if (pLayer is IRasterLayer)
            {
                cbxAddRasterLayers.Items.Add(pLayer.Name);
            }
            pLayer = layers.Next();
        }
}
//获取当前地图中的所有栅格图层
private IEnumLayer GetLayers()
{
        UID uid = new UIDClass();
        //筛选栅格图层
        uid.Value = "{D02371C7-35F7-11D2-B1F2-00C04F8EDEFF}";
        // IRasterLayer
        if (m_map.LayerCount != 0)
        {
            IEnumLayer layers = m_map.get_Layers(uid, true);
            return layers;
        }
        return null;
}
```

（4）当选择或者改变 cbxAddRasterLayers 控件中的图层时，触发 SelectedIndexChanged 事件，其响应函数代码如下：

| FrmCreatCoutour.cs（节选） | 功能：选择图层变化时所触发事件响应函数 |
|---|---|

```
private void cbxAddRasterLayers_SelectedIndexChanged(object sender,
                                                     EventArgs e)
{
        //如果cbxAddRasterLayers控件中的图层不为空
        if (cbxAddRasterLayers.SelectedItem != null)
        {
            string strRasterSelected = cbxAddRasterLayers.SelectedItem.ToString();
            m_RasterLayer = GetRasterLayer(strRasterSelected);
        }
}
//通过名字来得到图层
private IRasterLayer GetRasterLayer(string layerName)
{
        if (GetLayers() == null)
            return null;
        IEnumLayer layers = GetLayers();
        layers.Reset();
```

```
        ILayer pLayer = null;
        while ((pLayer = layers.Next()) != null)
        {
            if (pLayer.Name == layerName)
                return pLayer as IRasterLayer;
        }
        return null;
}
```

（5）先设定等值线间隔、基本等值线和保存路径，当点击“确定”按钮时，对选中的栅格图层提取等值线；添加提取等值线函数，并在“确定”按钮的 Click 事件中调用；点击“关闭”按钮时关闭窗体。参考代码如下：

| FrmCreatCoutour.cs（节选） | 功能：提取等值线函数和按钮 Click 事件 |
|---|---|

```
private void btnSetPath_Click(object sender, EventArgs e)
{
    //调用系统“保存文件”窗体
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "(*.shp)|*.shp";
    saveFileDialog.Title = "选择等值线图层存放位置";
    saveFileDialog.FilterIndex = 1;
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        //获取等值线图层的保存路径和等值线图层名称
        m_ShapeFileName = saveFileDialog.FileName;
        if (m_ShapeFileName == "")
            return;
        //获取等值线图层的保存路径
        m_Path = System.IO.Path.GetDirectoryName(m_ShapeFileName);
        //获取等值线图层的名称
        m_FileName = System.IO.Path.GetFileName(m_ShapeFileName);
        //在tbxSavePath中显示路径和名称
        tbxSavePath.Text = m_ShapeFileName;
    }
}
private void btnOK_Click(object sender, EventArgs e)
{
    CreatCoutour(m_RasterLayer);
}
//提取等值线
private void CreatCoutour(IRasterLayer rasterLayer)
{
    IRasterLayer rLyr = rasterLayer;
    ISurfaceOp2 pSurfaceOp = default(ISurfaceOp2);
```

```
        pSurfaceOp = new RasterSurfaceOp() as ISurfaceOp2;
        IGeoDataset pRasterDataset = rLyr as IGeoDataset;
        IWorkspace pShpWS = default(IWorkspace);
        //打开Shapefile工作空间
        IWorkspaceFactory pShpWorkspaceFactory = new
                                                ShapefileWorkspaceFactory();
        pShpWS = pShpWorkspaceFactory.OpenFromFile(m_Path, 0);
        //提取等值线
        pSurfaceOp = new RasterSurfaceOp() as ISurfaceOp2;
        IRasterAnalysisEnvironment pRasterAEnv =
                                        (IRasterAnalysisEnvironment)pSurfaceOp;
        pRasterAEnv.OutWorkspace = pShpWS;
        IGeoDataset pOutput = default(IGeoDataset);
        IFeatureClass pFeatureClass = default(IFeatureClass);
        IFeatureLayer pFLayer = default(IFeatureLayer);
        string strInterval = tbxCoutourInterval.Text;
        string strBase = tbxBasicCoutour.Text;
        double douInterval = Convert.ToDouble(strInterval);
        double douBase = Convert.ToDouble(strBase);
        object tmpbase;
        tmpbase = (object)douBase;
        object tmpmy = 1;
        pOutput = pSurfaceOp.Contour(pRasterDataset, douInterval,
                                                ref tmpbase, ref tmpmy);
        pFeatureClass = (IFeatureClass)pOutput;
        //添加等值线到当前地图
        pFLayer = new FeatureLayer();
        pFLayer.FeatureClass = pFeatureClass;
        IGeoFeatureLayer pGeoFL = default(IGeoFeatureLayer);
        pGeoFL = (IGeoFeatureLayer)pFLayer;
        pGeoFL.DisplayAnnotation = false;
        pGeoFL.DisplayField = "CONTOUR";
        pGeoFL.Name = m_FileName;
        m_hookHelper.FocusMap.AddLayer(pGeoFL);
}
private void btnClose_Click(object sender, EventArgs e)
{
        this.Close();
}
```

（6）在主窗体的菜单 menuStrip1 控件中添加“提取等值线”二级菜单项，如图 8-22 所示。

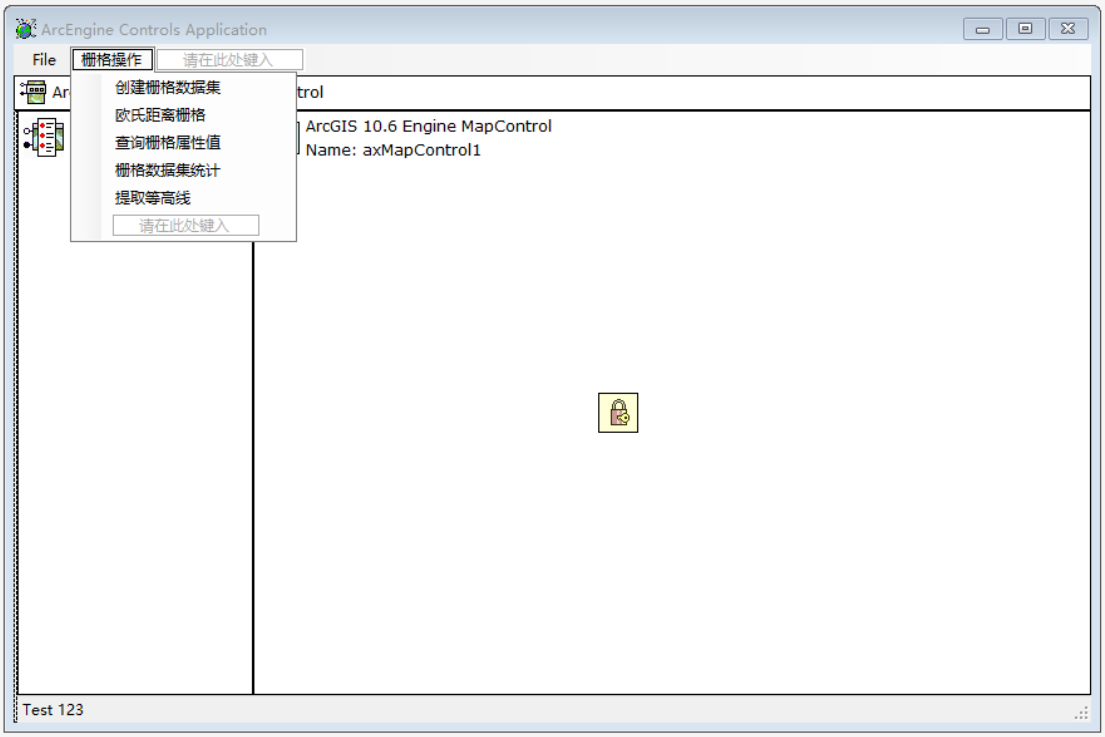


图 8-22　添加“提取等值线”二级菜单

双击“提取等值线”菜单命令，为 Click 事件添加代码调用“提取等值线”窗体，代码如下：

| MainForm.cs（节选） | 功能：调用“提取等值线”窗体 |
|---|---|

```
private void 提取等值线ToolStripMenuItem_Click(object sender, EventArgs e)
{
    FrmCreatCoutour frmCreatCoutour = new
                                    FrmCreatCoutour(m_mapControl.Object);
    frmCreatCoutour.ShowDialog();
}
```

（7）编译并运行程序后设置参数，程序运行结果如图 8-23 所示。

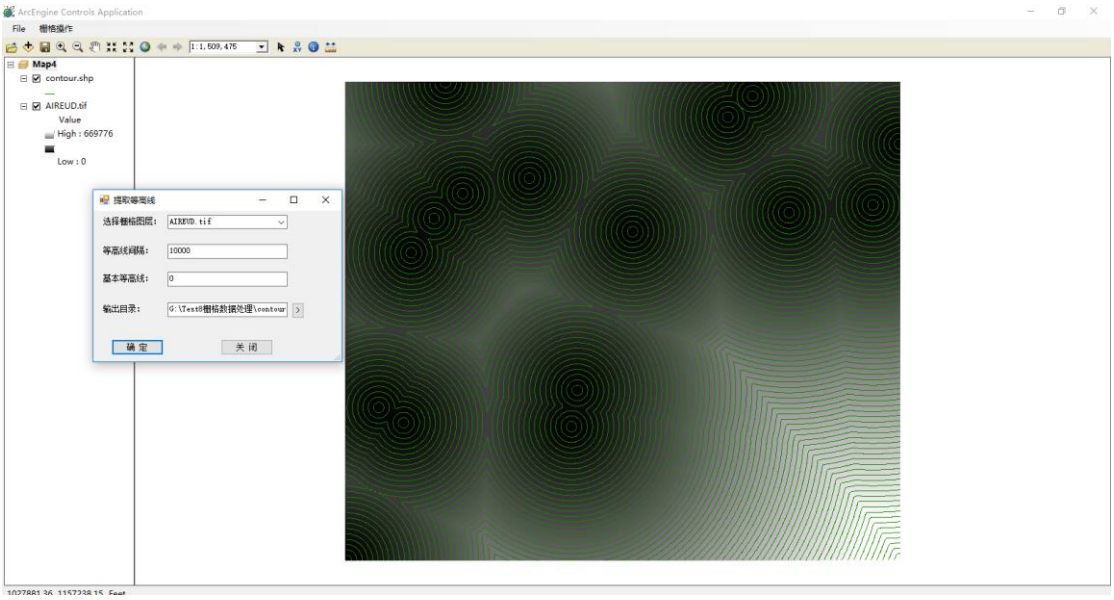


图 8-23　等值线提取功能运行效果