AE+C# TIN 相关三维功能模块介绍（一）

作者：刘志远

Email：592418843@qq.com

引言

最近，群里有不少朋友都在问关于 TIN 二次开发方面的问题，前不久正好做过这方面的内容，也一直想把这一块整理一下。现在的打算是将其按写成一个小的系列，每个系列涉及一些紧密联系的功能，最后将所涉及的内容整合到一个程序中，以便更好的理解，希望能对刚刚接触这块的朋友一点帮助。

本篇将介绍 TIN 数据的加载、TIN 图层属性设置和 TIN 图层渲染。这里说明一下，在使用本文的代码时请事先根据各个接口引用所需的命名空间。

1.TIN 数据的加载

这里直接贴上代码，解释见代码注释：

```
//下面代码是打开 E:\data\ 文件夹下名称为 tin 的 TIN数据。
//定位TIN数据
IWorkspaceFactory pWSFact = new TinWorkspaceFactoryClass();
IWorkspace pWS = pWSFact.OpenFromFile(@"E:\data\", 0);
ITinWorkspace pTinWS = pWS as ITinWorkspace;
ITin pTin = pTinWS.OpenTin("tin");
//将TIN变为TIN图层
ITinLayer pTinLayer = new TinLayerClass();
pTinLayer.Dataset = pTin;
pTinLayer.Name = "TIN";
//用三维控件AxSceneControl加载（也可以用AxMapControl，但看到了只能是平面效果）
this.axSceneControl1.Scene.AddLayer(pTinLayer, true);
```

2.TIN 图层属性设置

这里介绍 TIN 模型 Z 轴缩放、沿 Z 轴上下偏移以及 TIN 的平面显示与三维显示之间的切换。其效果与在 ArcScene 中右击 TIN 图层，点击属性下设置 Z unit conversion factor 和 Base Heights 的效果一样。主要用到了访问三维数据属性的接口 I3DProperties，这段代码也比较简单，相信大家稍微看一下就能明白，对应的功能可以复制后直接使用。

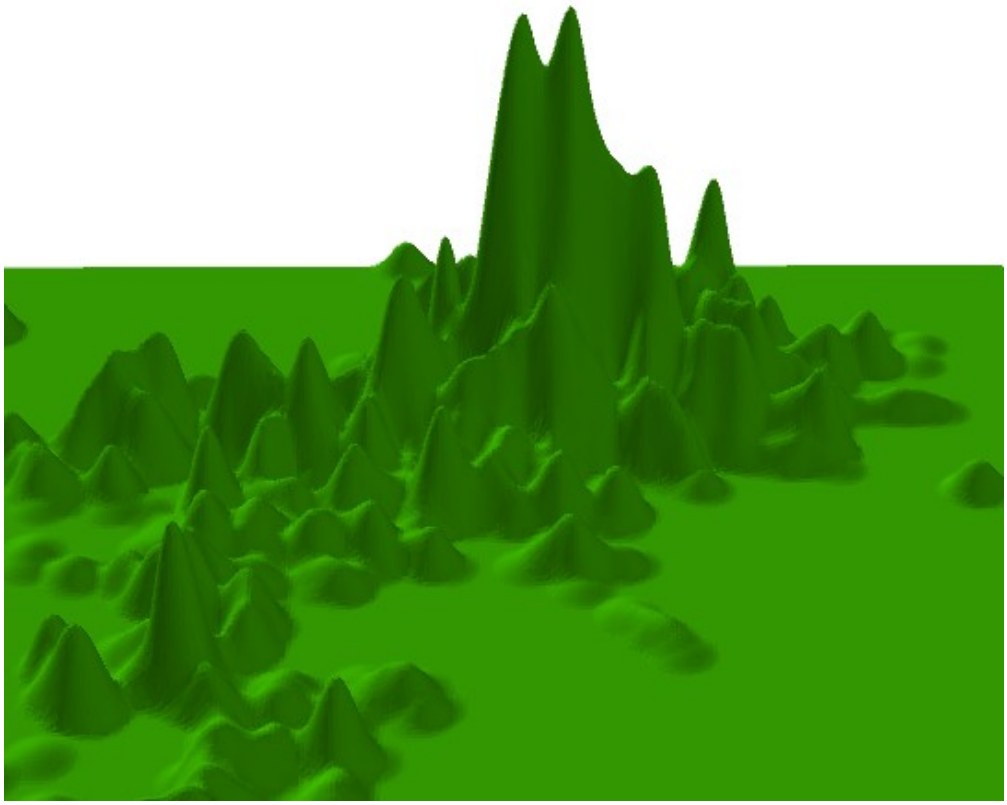
```
/// <summary>
/// 获取TIN图层三维属性
/// </summary>
/// <param name="pTinLayer">TIN图层</param>
/// <returns></returns>
public I3DProperties get3DProps(ITinLayer pTinLayer)
{
    I3DProperties p3DProps = null;
    ILayerExtensions lyrExt = pTinLayer as ILayerExtensions;
```

```
        for (int i = 0; i < lyrExt.ExtensionCount; i++)
        {
            if (lyrExt.get_Extension(i) is I3DProperties)
            {
                p3DProps = lyrExt.get_Extension(i) as I3DProperties;
            }
        }
        return p3DProps;
    }

    /// <summary>
    /// TIN模型Z轴缩放，设置缩放因子,此函数要位于控件加载TIN图层事件后
    /// </summary>
    /// <param name="pTinLayer">TIN图层</param>
    /// <param name="ZFactor">缩放因子：0-1缩小；1不变；>1拉伸</param>
    public void SetTinLayerZFactor(ITinLayer pTinLayer, double ZFactor)
    {
        I3DProperties p3DProps = get3DProps(pTinLayer);
        if (p3DProps != null)
        {
            p3DProps.ZFactor = ZFactor;//设置高程缩放因子
            p3DProps.Apply3DProperties(pTinLayer);
        }
    }

    /// <summary>
    /// TIN模型整体上下偏移
    /// </summary>
    /// <param name="pTinLayer">TIN图层</param>
    /// <param name="offset">偏移量：正数向上偏移；负数向下偏移</param>
    public void SetTinLayerOffset(ITinLayer pTinLayer, double offset)
    {
        I3DProperties p3DProps = get3DProps(pTinLayer);
        if (p3DProps != null)
        {
            p3DProps.OffsetExpressionString = offset.ToString();
            p3DProps.Apply3DProperties(pTinLayer);
        }
    }

    /// <summary>
    ///将TIN变为一个平面，并设置该平面的高程
    /// </summary>
```

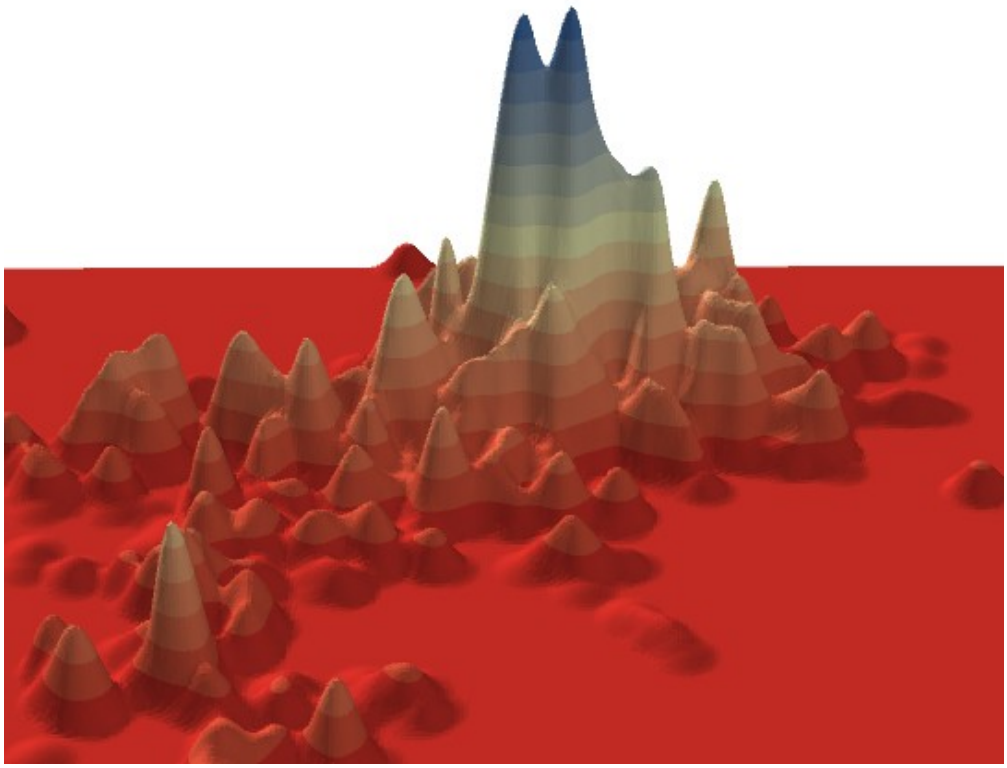
```
/// <param name="pTinLayer">TIN图层</param>
/// <param name="tinHeight">平面的高程</param>
public void SetTinLayerHeight(ITinLayer pTinLayer, double tinHeight)
{
    I3DProperties p3DProps = get3DProps(pTinLayer);
    if (p3DProps != null)
    {
        p3DProps.BaseOption = esriBaseOption.esriBaseExpression;
        p3DProps.BaseExpressionString = tinHeight.ToString();
        p3DProps.Apply3DProperties(pTinLayer);
    }
}

/// <summary>
/// 将平面TIN还原为三维立体形式
/// </summary>
/// <param name="pTinLayer">TIN图层</param>
public void SetTinAsBase(ITinLayer pTinLayer)
{
    I3DProperties p3DProps = get3DProps(pTinLayer);
    if (p3DProps != null)
    {
        p3DProps.BaseOption = esriBaseOption.esriBaseSurface;
        p3DProps.BaseSurface = pTinLayer as IFunctionalSurface;
        p3DProps.Apply3DProperties(pTinLayer);
    }
 }
```

3.TIN 模型渲染

TIN 模型有多种渲染方案，如：单色面渲染（Faces）、高程分级渲染（Elevation）、网格线渲染（Edges）、结点渲染（Nodes）等，这些渲染方案可以在 ArcScene 中通过 TIN 图层属性中的 Symbology 轻松地设置。下面介绍的是如何通过代码，以二次开发的方式进行设置。这里主要介绍单色面渲染和高程分级渲染，其他渲染方案感兴趣的朋友可以自己尝试一下。



单色面渲染（Faces）



高程分级渲染（Elevation）

3.1 单色面渲染（Faces）

可以设置渲染的颜色，该颜色要使用 IColor 类型的值。

```
/// <summary>
/// 单色渲染TIN三维模型
/// </summary>
/// <param name="TinLayer">要渲染的TIN图层</param>
/// <param name="pColor">渲染使用的颜色</param>
public void TinFaceRenderer(ITinLayer TinLayer, IColor pColor)
{
    ITinLayer pTinLayer = TinLayer;
    ITinRenderer pRenderNew = new TinFaceRenderer() as ITinRenderer;
    ITinSingleSymbolRenderer pUVRenderer = pRenderNew as ITinSingleSymbolRenderer;
    ISimpleFillSymbol pSymbol = new SimpleFillSymbolClass();
    pSymbol.Color = pColor;
    pUVRenderer.Symbol = pSymbol as ISymbol;

    pTinLayer.ClearRenderers();
    pTinLayer.InsertRenderer(pRenderNew, 0);//插入一个渲染模型
 }
```

3.2 高程分级渲染（Elevation）

高程分级渲染是按 TIN 模型的高程值将其分为指定数目的等级，每个等级按照给定的渐变色依次设色的方案。

```
/// <summary>
/// 渐变（分级）渲染TIN三维模型
/// </summary>
/// <param name="TinLayer">要渲染的TIN图层</param>
/// <param name="classCount">分级数目</param>
/// <param name="FColor">起始颜色</param>
/// <param name="TColor">终止颜色</param>
public void TinFaceGradientRenderer(ITinLayer TinLayer, int classCount, IColor FColor,
IColor TColor)
{
    ITinLayer pTinLayer = TinLayer;
    pTinLayer.ClearRenderers();
    ITinRenderer pRenderNew = new TinElevationRenderer() as ITinRenderer;

    //设置样式
    if (pRenderNew is ITinColorRampRenderer)
    {
        int ClassCount = classCount;
```

```
                double dZMin = pTinAdv.Extent.ZMin; //tin的Z轴最高值
                double dZMax = pTinAdv.Extent.ZMax; //tin的Z轴最低值
                double dInterval = (dZMax - dZMin) / ClassCount;

                ITinAdvanced pTinAdv = pTinLayer.Dataset as ITinAdvanced;
                ITinColorRampRenderer pElevRenderer = pRenderNew as ITinColorRampRenderer;
                IClassBreaksUIProperties pCBUI = pRenderNew as IClassBreaksUIProperties;
                pCBUI.ColorRamp = "Custom";
                pElevRenderer.BreakCount = ClassCount;//设置断面个数

                double dLowBreak = dZMin;
                double dHighBreak = dLowBreak + dInterval;
                //创建颜色集合
                IAlgorithmicColorRamp pColorRamp = new AlgorithmicColorRampClass();
                pColorRamp.Algorithm = esriColorRampAlgorithm.esriCIELabAlgorithm;
                pColorRamp.FromColor = FColor;
                pColorRamp.ToColor = TColor;
                pColorRamp.Size = ClassCount;
                bool bOK = false;
                pColorRamp.CreateRamp(out bOK);

                ISimpleFillSymbol pSymbol = null;
                for (int j = 0; j < ClassCount; j++)
                {
                    pCBUI.set_LowBreak(j, dLowBreak);
                    pElevRenderer.set_Break(j, dHighBreak);
                    //用于图层控制中分级标示显示
                    pElevRenderer.set_Label(j, dLowBreak.ToString("#.#") + " - " +
dHighBreak.ToString("#.#"));
                    dLowBreak = dHighBreak;
                    dHighBreak = dHighBreak + dInterval;

                    pSymbol = new SimpleFillSymbolClass();
                    pSymbol.Color = pColorRamp.get_Color(j);
                    pElevRenderer.set_Symbol(j, pSymbol as ISymbol);
                }

                //创建TinEdgeRendererClass 类型的Renderer
                pTinLayer.ClearRenderers();
                pTinLayer.InsertRenderer(pRenderNew, 0);//插入一个渲染模型
            }
        }
```

注意,TIN 模型的渲染方案是可以叠加显示的,所以每次都是使用的 InsertRenderer()方法。

4.下一篇简介

本篇先介绍到这。在使用渲染方案的时候，上面给的方法中的颜色都是由代码给出的，有人不禁想：要是让用户自己选择颜色（类似 ArcGIS 中的颜色选择下拉框）多好。可惜的是 AE 中并没有封装这样的颜色选择模块。

下一篇打算介绍如何制作单一颜色选择下拉框控件和渐变颜色选择下拉框控件，而后将其整合到 TIN 渲染模式选择中就更完善了。当然，掌握了后使用在其他需要颜色选择的地方也是可以的。这一个系列介绍完后，我会将这些介绍过的功能都集成到一个独立程序中，然后将其发上来，以供感兴趣的朋友参考。