产品介绍.....2
概述.....2
产品架构.....2
模块结构.....3
主要功能.....5
系统配置要求.....5
硬件要求.....5
最低硬件配置.....5
推荐硬件配置.....5
软件要求.....6
操作系统要求.....6
基础软件要求.....6
开发环境要求.....6
产品入门.....6
第一步 创建一个工程.....7
第二步 加载程序集.....7
第三步 创建一个地球场景.....10
支持转换的数据格式.....12
开发组件类结构图.....13
基础知识.....13
GIS 简述.....13
地理坐标系.....15
投影坐标系.....16
坐标系转换.....17
栅格数据.....18
影像数据.....22
地形数据.....23
矢量数据.....25
专题图.....25
三维操作.....26
UGX.....32
UGT.....33
应用专题.....35
BIM.....35
倾斜摄影.....35
点云.....37
三维管线.....38
范例程序说明.....39
场景属性控制.....39
UGX 和 UGT 数据加载.....45
飞行演示.....49
标注.....52
三维测量.....57
三维空间分析.....64

三维专题图 69
常见问题解答 75
开发接口帮助 75

产品介绍

概述

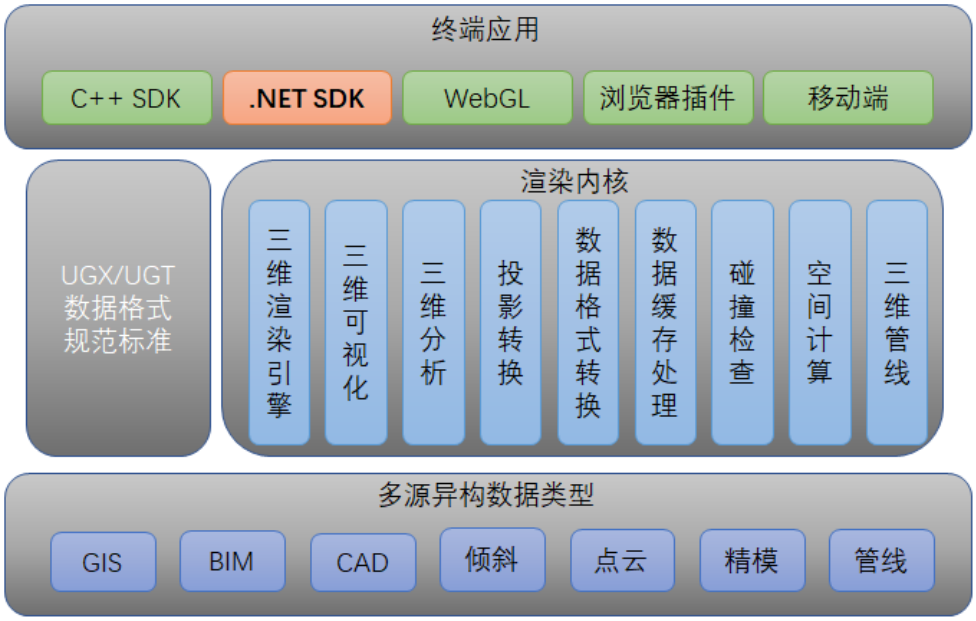
BcEngineX 是以三维 GIS 技术为基础，围绕 BIM、CAD、激光点云、倾斜摄影等多源异构三维数据的处理和渲染，构建的一套跨平台三维设计、施工、运维的通用技术体系，解决了工程数据与 GIS 数据的无缝融合，大规模设计数据高性能及高质量渲染等技术问题，并建立了一套跨终端的三维数据规范与格式标准。

BcEngineX 三维引擎的技术体系和底层组件包括:

- 统一的数据格式包括：UGX、UGTile(适用于 Windows、Linux 本地端开发和 WebGL 开发)；支持多数据源数据转换如：BIM、CAD、激光点云、倾斜摄影、精模等。
- 数据处理组件：实现三维各环节间的数据流转所需的格式转换、计算等功能。
- 渲染展示组件：采用延时渲染框架+物理渲染模型；多种渲染性能提升技术，包括渲染引擎技术、LOD、分页加载调度、批量渲染技术、实例化、并行更新、压缩、缓存等，呈现更好效果。

产品架构

BcEngineX SDK 是专门面向二次开发设计的组件式开发包，支持 C#、VB .NET、C++/CLI 等所有 .NET 开发语言，因此，适应了多种语言的开发用户。基于该 SDK 开发的应用软件易于分发和再部署，用户只需要将 BcEngineX SDK 运行库文件与 VC++ 2015 重分发包连同所开发的应用系统一起打包分发即可，只要使用该应用系统的目标机器上安装了 .NET Framework4.0 及以上版本，即可保证系统的正常运行。这种分发与再部署，有效地降低了成本，为用户最大程度地创造价值。



模块结构

BcEngineX 具有合理的功能组件划分，在 BcEngineX 功能组件中，主要分为数据处理功能、三维功能、地图功能、管线功能四个组件；其中数据处理功能组件不仅提供了对 BcEngineX_SDK 的支持，同时还对 C++ SDK、WebGL、浏览器插件、移动端等版本提供了统一的数据支持，体现了地理信息系统的数据概念，数据是 GIS 的血液。数据处理组件主要专注于对空间数据的处理，其他模块依赖于数据模块的同时又相对独立。

功能模块	对应的程序集	涵盖功能概要
基础数据模块	EngineX.Data.dll	提供对基础空间数据包围盒、空间点等数据及其属性的操作和处理，还包括多种 gis 数据地理坐标系及投影坐标系之间的转换功。
数据处理模块	EngineX.DataProcessing.dll	提供了多种数据处理，包括三维影像、地形和模型数据的缓存生成功能，包含了栅格数据、矢量数据、倾斜摄影、点云、UGX 及 BIM 数据。
地图模块	EngineX.MapControl.dll	提供了综合的地图显示，渲染，编辑。
三维模块	EngineX.SceneControl.dll	提供数据、显示、分析二三维一体化的三维场景展示，同时，全球尺度的地形数据以及全球尺度的高分辨率影像数据显示；支持矢量建模、管线和自定义数据格式 UGX、UGTile 高清渲染功能。另外，可以在三维窗口中进行各种方式的漫游、浏览，并且可以进行选择、查询和定位等操作。

管线模块	EngineX.PipeAnalysis.dll	提供基于二三维一体化技术的自适应管点符号，可由二维的点、线数据生成三维管线数据，根据管网走向、管线截面自动实时管点建模。

主要功能

- 支持各类 GIS 地理信息数据的三维展示
- 支持大规模点云、倾斜摄影的三维展示
- 支持各类 BIM 模型的数据格式转换，包括 AutoCAD、Revit、MicroStation、Catia、PDMS 等
- 支持三维网格模型精简处理
- 支持三维网格模型高性能碰撞检测及距离计算
- 提供配套完整的地理信息数据处理组件
- 提供配套完整的 BIM 模型、点云、倾斜摄影模型数据处理组件
- 建立统一的数据规范与格式标准

系统配置要求

硬件要求

最低硬件配置

- CPU：酷睿 i3 或同级别处理器，主频 2.00GHz 以上
- 内存：4GB（64 位系统建议 8GB）
- 硬盘空间：40GB 或以上
- 显卡：512MB 或以上显存（支持 OPENGL4.0 或以上），需要安装显卡驱动

推荐硬件配置

- CPU：酷睿 i7 或以上级别处理器

- 内存：4GB 或以上（64 位系统建议 16GB 或以上）
- 硬盘空间：100GB 或以上
- 显卡：GTX1050 或以上级别处理芯片，2GB 或以上显存（支持 OPENGL4.0 或以上），需要安装显卡驱动

注意：如需体验最佳三维效果，请选择 NVIDIA 系列显卡；建议不要在 16 位的颜色环境下运行三维内容。

软件要求

操作系统要求

- Microsoft Windows7系列
- Microsoft Windows8系列
- Microsoft Windows10系列

基础软件要求

- Microsoft .NET Framework 4.0

开发环境要求

- Microsoft Visual Studio 2015及以上版本

产品入门

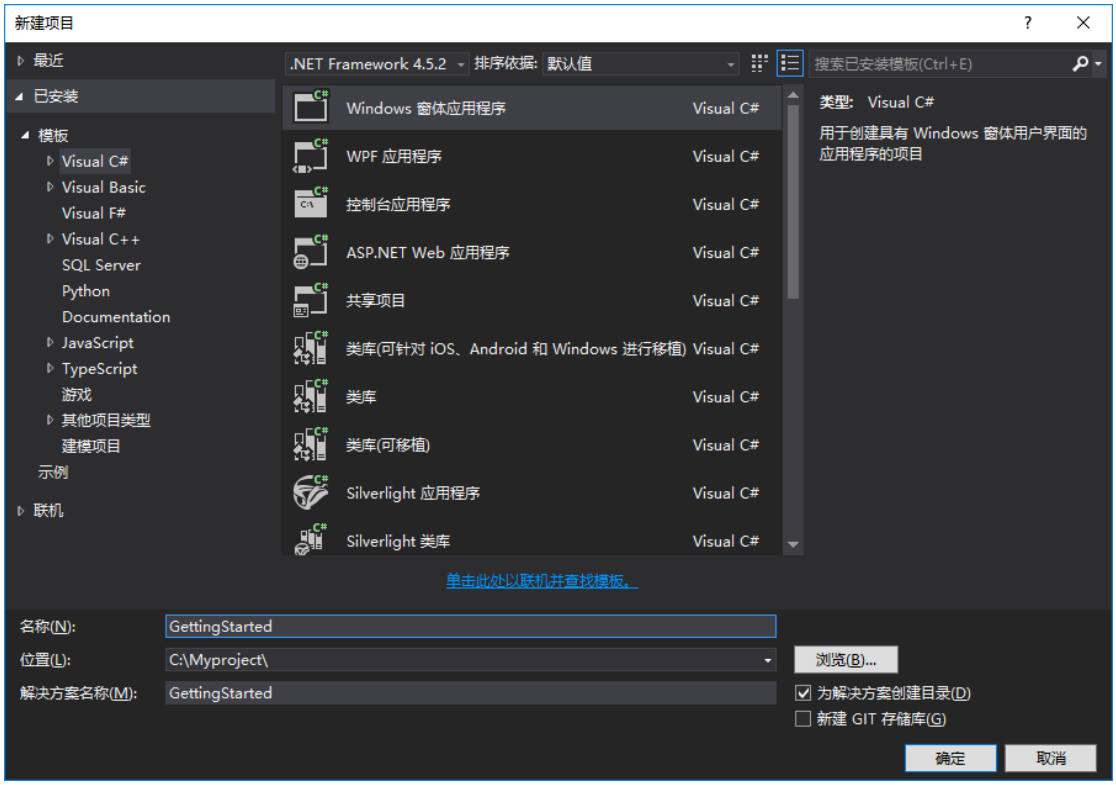
欢迎使用 BcEngineX .NET SDK 产品入门教程。本帮助文档将通过在 Microsoft Visual Studio 2015 开发平台上，基于 C＃语言的入门范例程序为初次接触 BcEngineX .NET SDK 进行开发的人员提供方便快捷的向导和简单而详尽的代码。

第一步 创建一个工程

创建一个工作目录 C:\Myproject

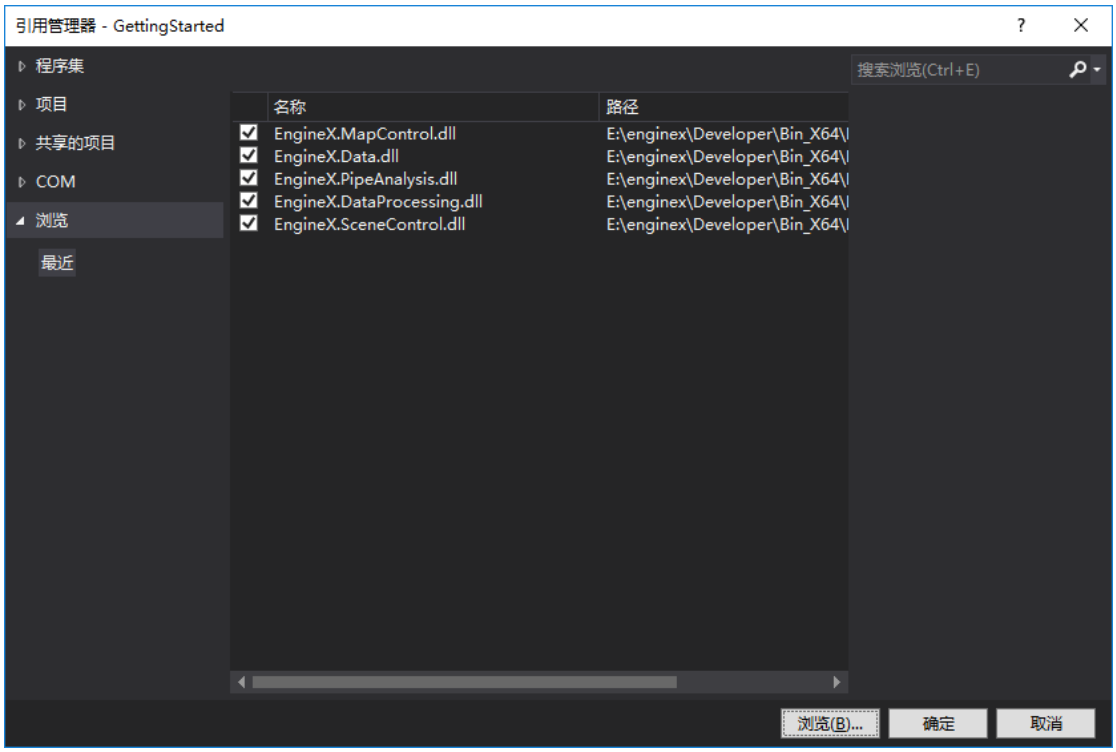
启动 Visual Studio 2015，然后创建新工程（文件 > 新建… > 项目）

在已创建好的工作目录 C:\Myproject 下创建一个新工程，在项目类型中选择 Visual C#，在模板中选择 Windows 窗体应用程序，设置该工程名称为 GettingStarted 。如下图所示：

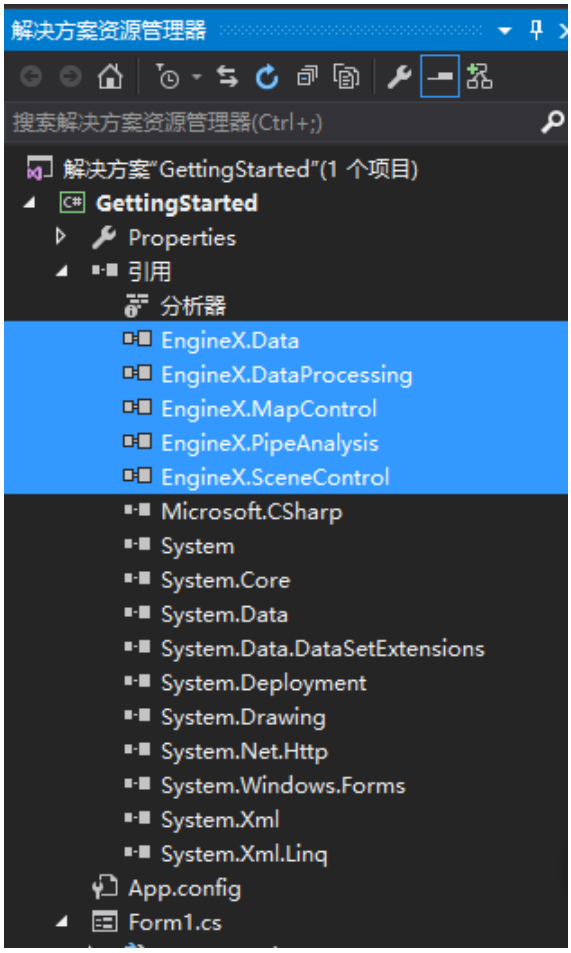


第二步 加载程序集

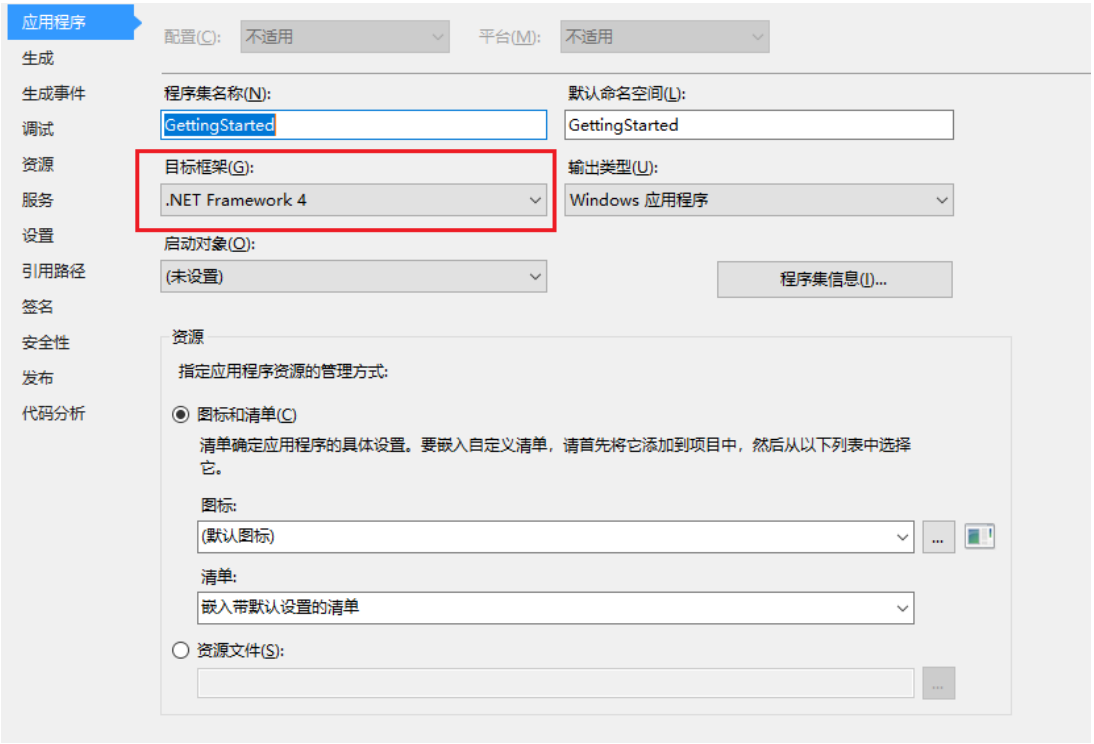
在工程 GettingStarted 引用管理器中点击“添加引用”，并加载 BcEngineX SDK 的 Developer\Bin_X64 目录中所提供的如下功能模块的程序集 EngineX.Data.dll 、EngineX.DataProcessing.dll 、 EngineX.MapControl.dll 、 EngineX.SceneControl.dll 、EngineX.PipeAnalysis.dll、EngineX.MapControl.dll，如下：



添加完成在引用管理器中可以看到所添加的程序集模块，如下：

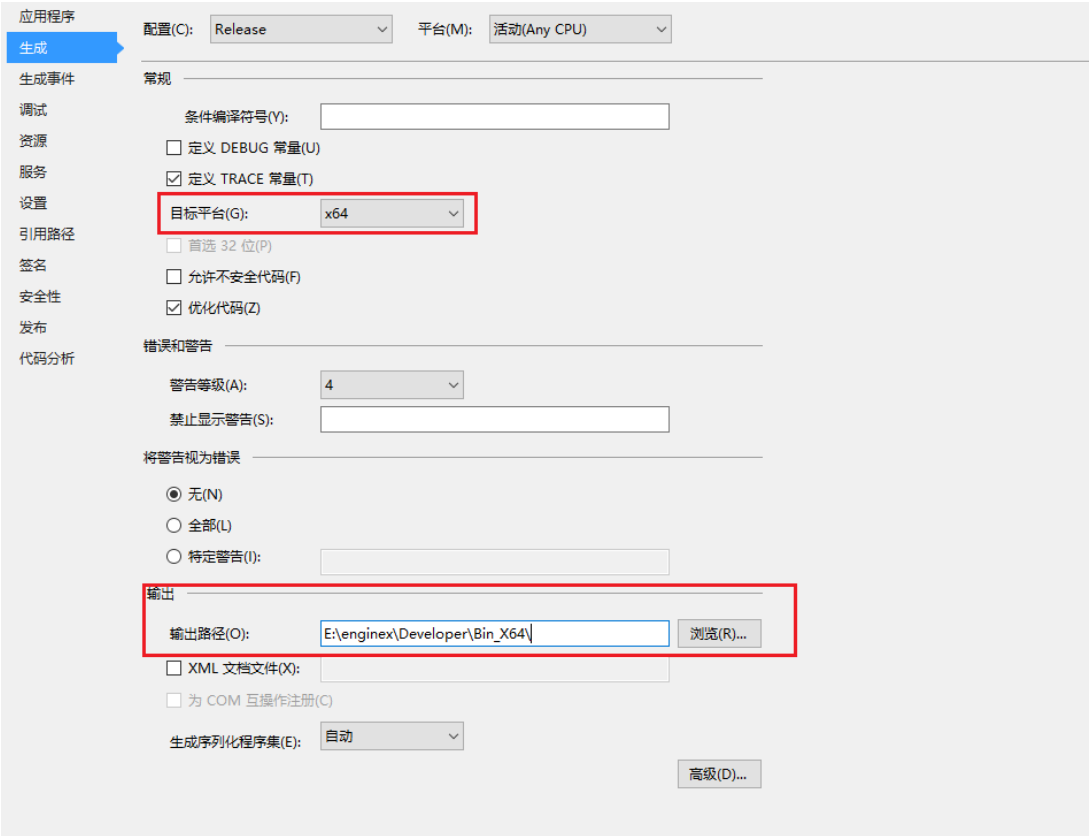


在工程 GettingStarted 右键-属性，打开“应用程序”将目标框架设置为“.NET Framework 4”



在“生成”配置中设置目标平台-x64（因为这里引用是 Bin_X64 的程序集 dll 库，如果引用 Bin 下的 dll 库则设置目标平台为 x86）并将程序输出路径也修改为 Developer\Bin_X64 下，如下：

s

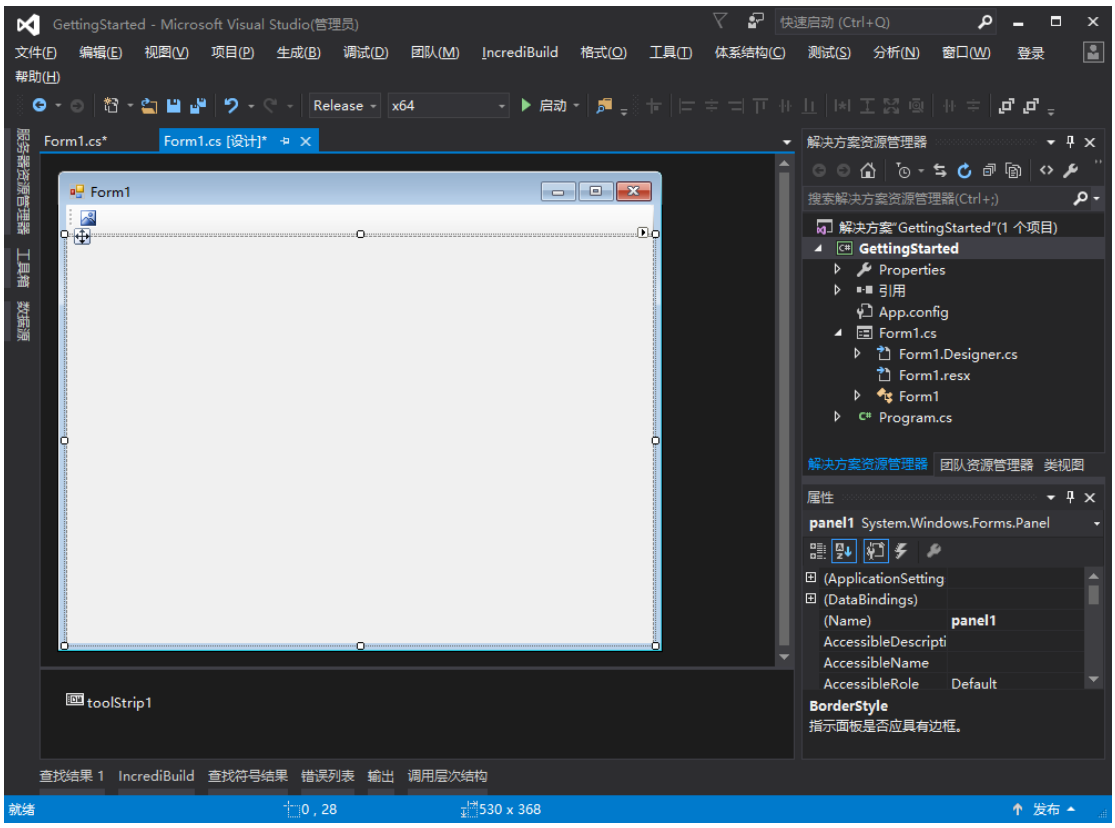


第三步 创建一个地球场景

在窗体中加入 ToolStript 工具条，在工具条上添加 一个 Button 按钮，设置如下的属性值（其余属性取默认值即可）：

Name	Text
toolStripCreateEarth	创建地球场景

再在窗体中添加 panel 窗体，将 panel1 窗体的属性“Dock”设置为”Fill”后如下图：



编写代码

在这里，我们以创建地球场景为例，关于如何创建其他类型的场景和地图，请参考相应接口在帮助文档中的说明。在正式编写示范程序之前，需要添加对 BcEngineX SDK 的 SceneControl 程序集的引用，进入代码编辑器，在最顶端添加如下代码：

```
using EngineX.Render;
```

双击 Form1 窗体中的 toolStripCreateEarth 按钮，进入代码编辑器。或者直接进入代码编辑器，手动添加 Form1 窗体的“ toolStripCreateEarth_Click ”事件。在“ toolStripCreateEarth_Click ”事件中插入如下代码：

```
SceneControl bc_sceneControl = null; //创建 SceneControl 对象
//******创建一个地球场景******
private void toolStripCreateEarth_Click(object sender, EventArgs e)
```

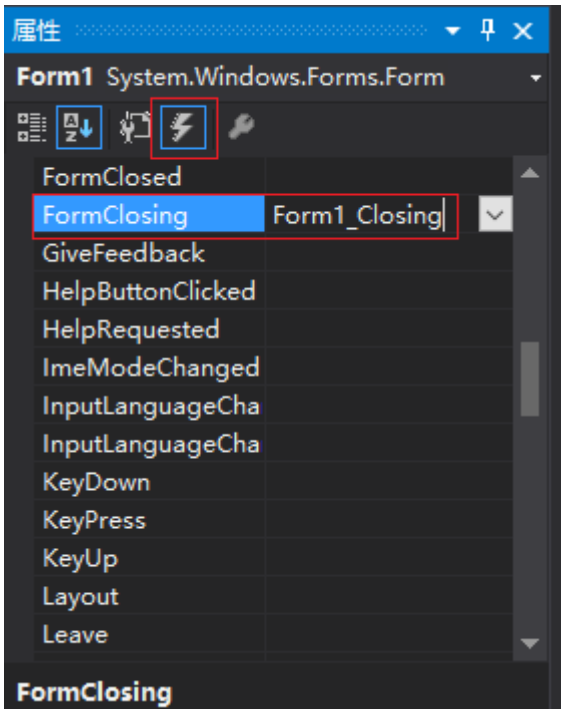
```
{
    //创建一个 WGS84 坐标系的地球场景
    bc_sceneControl = new SceneControl(SceneType.ST_WGS84);
    bc_sceneControl.Dock = DockStyle.Fill;
    this.panel1.Controls.Add(bc_sceneControl);
}
```

添加 Form1_FormClosing 事件

程序运行结束后，应当释放该程序所占用的内存资源，本例结束运行时需释放 SceneControl 对象所占用的资源。因此，按照以下步骤在 Form1 窗体的 FormClosing 事件中添加代码。

1、选择“视图”>“设计器”来显示设计视图。在设计视图中，请选定 Form1 窗体。

2、在“属性”窗口中，单击“事件”按钮，如图所示：



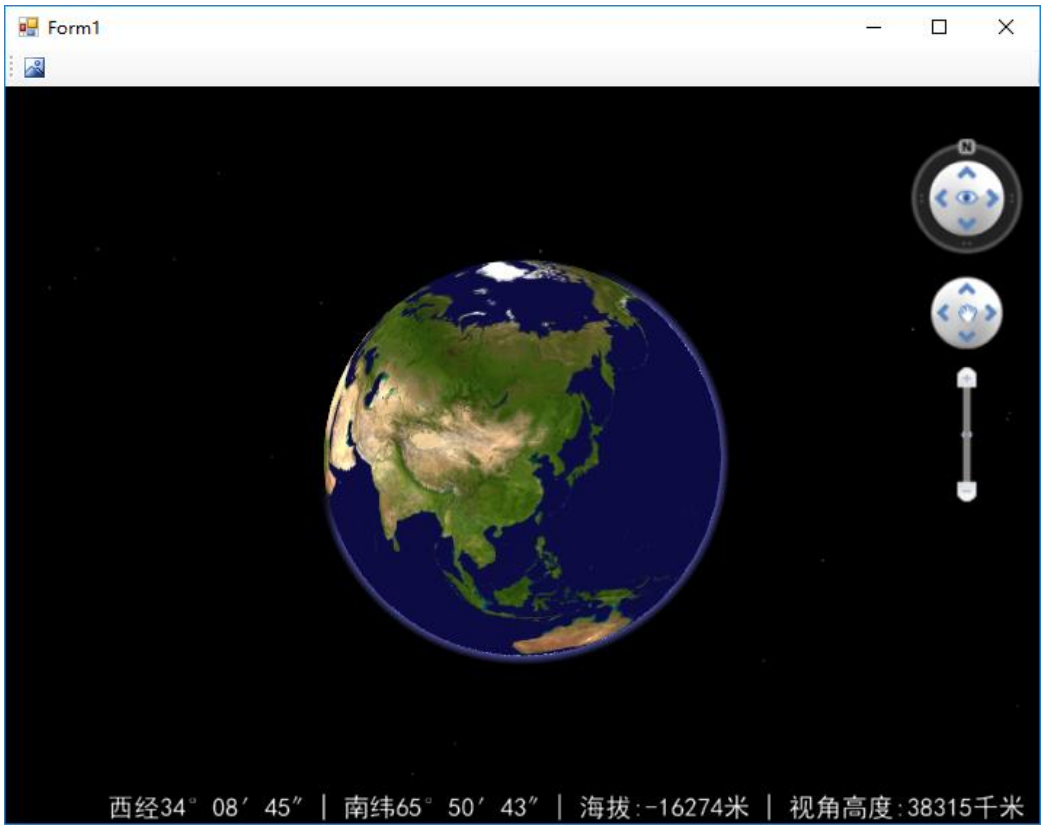
3、选择 FormClosing 事件。

4、在文本框中输入 Form1_Closing，然后按回车键。随后会创建一个名为 Form1_sClosing 的事件方法，并在“代码和文本编辑器”窗口中显示，添加如下代码：

```
// 释放：释放 SceneControl 对象
private void Form1_Closing(object sender, FormClosingEventArgs e)
{
    bc_sceneControl.Dispose();
    bc_sceneControl = null;
}
```

运行调试

运行代码后弹出如下界面，单击工具条上的“创建地球场景”按钮，打开，一个坐标系为 WGS84 类型的地球将显示在窗口中。



支持转换的数据格式

我们在进行 GIS 处理时，除了使用 BcEngineX 自己的数据格式外，还需要对多种外部的数据格式进行转换操作。数据转换功能提供了不同数据格式之间的互转，很好地支持了多种数据格式。

下文给出了数据导入和数据导出支持的数据格式列表，方便用户的使用。

支持导入的矢量文件格式

支持导入的矢量文件格式	文件说明
ArcView Shape 文件（*.shp）	是 ArcView GIS 软件特有的数据格式，用于存储地理要素的空间和属性信息，是常用的一种矢量数据格式。

支持导入的栅格文件格式

支持导入的栅格文件格式	文件说明
TIFF 影像数据（*.tif; *.tiff）	TIFF（Tagged Image File Format）标签图像文件格式，是常见的高位彩色图像格式，GeoTIFF 是一种包含地理信息的 TIFF 文件格式，其信息编码在 TIFF 文件预留的 Tag（标

	签）之中。
Erdas Image 文件（*.img）	是 ERDAS 平台下用于遥感分析的常见文件格式，可存储影像、栅格、DEM 等多种数据
USGDEM 和 GBDEM 栅格文件（*.dem）	是空间数据的交换格式，导入为栅格数据。
ASCII Grid 文件（*.asc）	是 ArcGIS for Desktop Advanced 格网交换文件。
PNG 文件（*.png）	网络图像文件存储格式，对连续像元值的像元有简单压缩。
JPG 文件（*.jpg; *.jpeg）	常见的图片格式，采用 24 位颜色存储单个图像。

支持导入的模型文件格式

支持导入的模型文件格式	文件说明
RVT 三维模型文件（*.rvt）	RVT 文件是 Revit 中，由 Autodesk 开发的建筑信息模型（BIM）软件创建的 CAD 文件。RVT 文件包含 3D 建筑设计。可导入为模型数据集。
倾斜摄影 OSGB 文件（*.osgb，*.osg）	倾斜摄影模型数据，可导入为模型数据集。
激光点云数据（*.las）	由三维扫描仪产生的，它可以测量物体外部表面上的大量点，可处理后导入为模型数据集。
IFC 模型文件（*.ifc）	IFC 即 Industry Foundation Classes，建筑工程数据交换标准。
Imodel（*.imodel）	Bentley 公司开发的对基础设施信息进行开放式交换的载体，可导入为模型数据集。
GIM(*.gim)	电网规范建筑模型格式

开发组件类结构图

基础知识

GIS 简述

地理信息系统（GIS）的基本概念

地理信息系统即 GIS(Geographic Information System)，是由计算机软件、硬件和不同方法组成的系统，该系统设计用来支持空间数据采集、管理、处理、分析、建模和显示，以便解决复杂的规划和管理问题。地理信息系统与其他信息的主要区别在于其存储和处理的信息是经过地理编码的，地理位置及与该位置有关的地物属性信息成为信息检索的重要部分。在地理信息系统中，现实世界被表达成一系列的地理要素和地理现象，这些地理特征至少由空间位置参考信息和非空间位置信息两个组成部分。
一个完整的 GIS 主要由计算机硬件系统、计算机软件系统、地理数据、系统管理操作人员四部分构成，其核心部分是计算机系统，空间数据反映 GIS 的地理内容，而管理人员和用户则决定系统的工作方式和信息表达方式。

GIS 的基本功能

数据采集与输入：在数据处理系统中将系统外部的原始数据传输给系统内部，并将这些数据从外部格式转换为系统便于处理的内部格式。

数据编辑与更新：数据编辑主要包括图形编辑和属性编辑。图形编辑主要包括拓扑关系建立、图形编辑、图形整饰、图幅拼接、图形变换、投影变换、误差校正等功能。属性编辑主要与数据库管理结合在一起完成。数据更新即以新的数据项或记录来替换数据文件或数据库中对应的数据项或记录，它是通过删除、修改、插入等一系列操作来实现的。

数据存储与管理：空间数据存储是 GIS 中最低层和最基本的技术，它直接影响到其他高层功能的实现效率，从而影响整个 GIS 的性能。属性数据管理一般既可利用 GIS 软件进行管理，也可直接利用商用数据库软件进行管理。空间数据管理是 GIS 数据管理的核心，各种图形或图像信息都以严密的逻辑结构存放在空间数据库中。

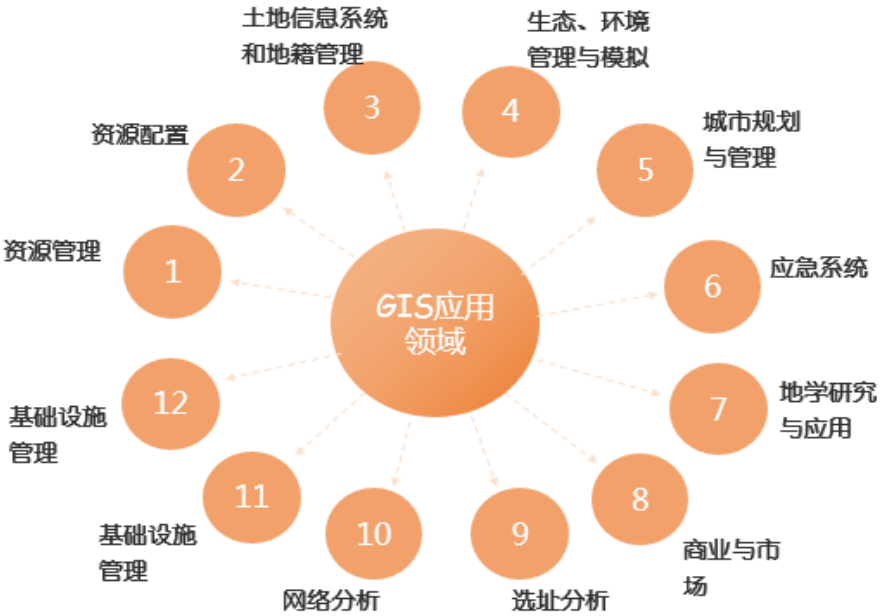
空间查询与分析：空间查询与分析是 GIS 的核心，主要包括数据操作运算、数据查询检索和数据综合分析三方面。通过 GIS 提供的空间分析功能，用户可以从已知的空间数据中得出隐含的重要结论，这对于许多应用领域是至关重要的。

数据显示与输出：将用户查询的结果或是数据分析的结果以合适的形式输出是 GIS 问题求解过程的最后一道工序。输出形式通常有两种：在计算机屏幕上显示或通过绘图仪输出。这方面的技术主要包括：编辑、图形整饰、符号制作、图景图例生成、出版印刷等。

图 GIS 基本功能

GIS 的应用领域

地理信息系统在近年内取得了很大的发展，主要应用于资源调查、环境评估、灾害预测、国土管理、城市规划、有点通讯、交通运输、军事公安、水里水电、公共设施管理、农林牧业、统计、商业金融等几乎所有领域。



地理坐标系

地理坐标系：为球面坐标。参考平面地椭球面，坐标单位：经纬度；

使用经纬度坐标来表示椭球上任意一点的坐标，又称为经纬度坐标系。地理坐标系中，通常包含对水平基准、中央子午线和角度单位的定义。常用的地理坐标系如：WGS84、Beijing1954、Clarke 1866 等。例如，Google Earth 上的 KML 数据，全球定位系统采集的数据，都是以 WGS84 为坐标系的。大地测量获取的控制点坐标以西安 80 或 Beijing1954 为坐标系。下图为 WGS84 坐标系的世界地图。

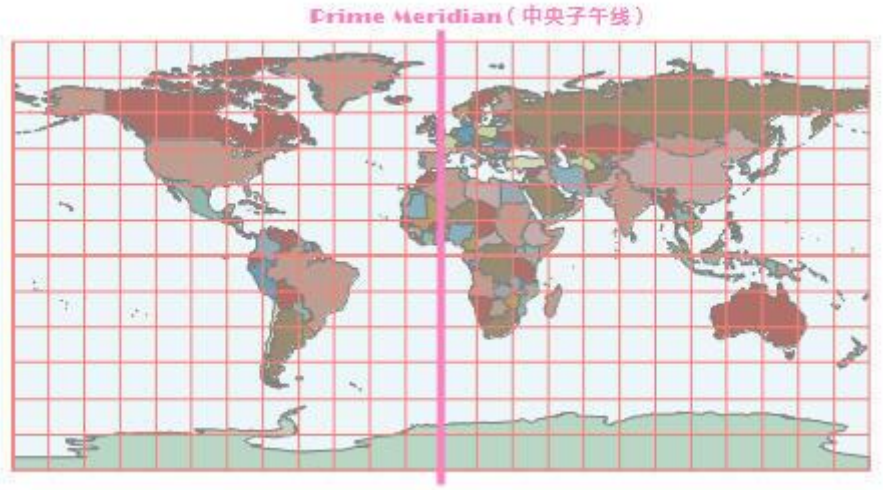


图 经纬度坐标系

投影坐标系

投影坐标系：为平面坐标。参考平面地是水平面，坐标单位：米、千米等;

通过某种投影方式和投影类型，将椭球上的任意一点投影到平面上。使用二维平面坐标 XY 来表示点线面地物的位置。投影坐标系中，通常包含对地理坐标系、地图投影、投影参数及距离单位的定义。常用的投影坐标系，例如：UTM、Gauss-Kruger、Albers、Mercator 等。一般，经过投影的地理数据，可进行地图量算、各种空间分析、制图表达等。例如，我国基本比例尺地形图中，1：100 万地形图采用 Albers 投影，其余大部分都采用了高斯-克吕格 6°带或者 3°带投影。而城市规划中用到的大比例尺地图，如 1：500，1：1000 等的道路施工图、建筑设计图等多采用平面坐标系。下图为原 WGS84 坐标系的世界地图进行了 Robinson 投影。

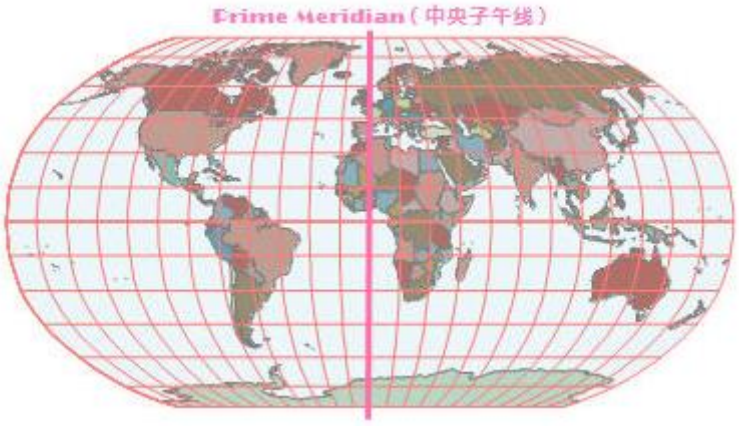


图 投影坐标系

一般情况下，一个投影坐标系包含以下几部分：投影坐标系的名称，坐标单位，投影方式，地理坐标系，投影参数。常用的投影参数有以下几种：

False Easting & False Northing
东伪偏移、北伪偏移值。该参数主要是为了保证横纵坐标不出现负值。比如 GK6 度分带投影中，一般都设置横坐标东偏 500 公里，保证位于中央经线以西的所有点横坐标值都是正值。由于我国位于北半球，不存在纵坐标是负值的情况，所以 False Northing 是 0。

Azimuth
用来定义投影面轴线的方位角，从北顺时针向东量算。如果是正轴，值为 0；横轴，值为 90 度，斜轴，x 度。

Central meridian /Longitude of origin
定义了投影区域内中央纬线，即投影区域内纵坐标的起算位置。有些投影设置了原点纵坐标后，就不用再设置北伪偏了。

Standard parallel 1& standard parallel 2
圆锥投影用的标准纬线；Lambert 等积投影，只有一条纬线，定义了原点的 Y 坐标。
在圆锥投影中，圆锥面通过地球并与地球纬线发生相切或相割，这些切线或割线就是标准纬线，切圆锥投影用户只需指定一条标准纬线，割圆锥投影则需指定两条标准纬线，即第一标准纬线与第二标准纬线。如果是单标准纬线，则第二标准纬线应与第一标准纬线的值相同。另外还要设原点纬线，即最南端纬线。

Longitude of first point Latitude of first point Longitude of second point & Latitude of second point
在两点等距方位投影时用，这四个参数分别定义了两个点的经纬度坐标。例如：Two-Point Equidistant and Hotine Oblique Mercator projections（斜轴墨卡托投影）。

Scale factor
中央经线或者投影中心的比例因子，比如 UTM 是 0.9996。

坐标系转换

在进行地理数据处理的工作中，坐标系转换问题会经常遇到。当前最便捷的定位方法就是 GPS 卫星定位，GPS 定位的数据是基于 WGS84 坐标系上的数据，而各国家采用的往往是早先建立的国家坐标系，为了避免出现矛盾，就需要将 WGS84 定位数据转换到国家坐标系上，也就是进行坐标系（参照系）的转换。

在进行坐标系变换方法的讲解前，首先需要清楚以下两种坐标系的概念：

- 大地经纬度坐标系——大地测量中以参考椭球面为基准面的坐标。地面点 P 的位置用大地经度 ϕ、大地纬度λ和大地高 h 表示。当点在参考椭球面上时，仅用大地经度和大地纬

度表示。大地经度是通过该点的大地子午面与起始大地子午面之间的夹角，大地纬度是通过该点的法线与赤道面的夹角，大地高是地面点沿法线到参考椭球面的距离。

- 空间直角坐标系——基于地心的直角坐标系，用 X、Y、Z 表示，其中直角坐标系的原点位于地心；Z 轴与地球旋转轴重合（极轴），向北为正方向；X 轴穿过本初子午线与赤道的交点，向东为正方向；Y 轴穿过赤道与东经 90°的交点。

在实际应用中，我们通常需要将某参照系下的经纬度数据转换到新参照系下的经纬度数据，这种坐标变换的基本步骤为：将原经纬度坐标变换为空间直角坐标，再将变换后的原空间直角坐标通过指定的转换方法(三参数转换法或七参数转换法)转换为目标空间直角坐标，最后将目标空间直角坐标变换为经纬度坐标，从而完成不同参照系下经纬度坐标的相互转换。

如果在投影转换中，如果原投影和目标投影的地理坐标系不同，则需要进行参照系（坐标系）的转换。

坐标系的转换往往是投影转换中的一个转换环节，在将数据从原投影坐标系转换到目标投影坐标系时，需要经过以下转换步骤：

将原投影坐标系转换为大地经纬度坐标系->大地经纬度坐标系转换为空间直角坐标系->空间直角坐标系转换为目标空间直角坐标系->目标空间直角坐标系转换为大地经纬度坐标系->大地经纬度坐标系转换为投影坐标系。

栅格数据

栅格数据：栅格数据结构是将一个平面空间进行行和列的规则划分，形成有规律的网格，实际上就是像元的阵列，像元是栅格数据最基本的信息存储单元，每个像元都有给定的属性值来表示地理实体或现实世界的某种现象。

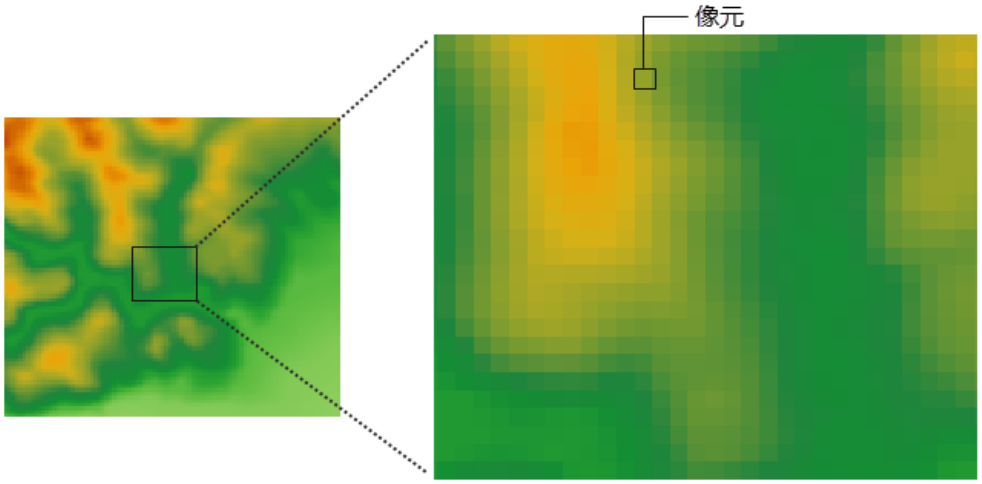


图 像元的显示值

跟栅格数据相关的一些基本概念有以下几点：

像元/像素：像元和像素都是指栅格数据中的最小组成单位。像素通常会作为像元的同义词使用，像素是图像元素的简称，通常用于描述影像数据，而像元则通常用于描述栅格数据。
像元值：在栅格数据集中，每个像元（像素）都有一个值。一个像元具有一个属性值，而像元都具有一定的空间分辨率，即对应着地表的一定范围的区域，因而像元值代表的是像元所覆盖的区域的占主导的要素或现象。比如卫星影像和航空相片中的光谱值反映了光在某个波段的反射率；DEM 栅格的高程值表示了平均海平面之上的地表高程，由 DEM 栅格生成的坡度图，坡向图和流域图的像元值分别代表了其坡度，坡向和流域属性；土地利用分类图中的类别值如耕地、林地、草地等；还可以表示降水量、污染物浓度、距离等数量值。另外，像元值可以是整数，也可以是浮点数。
行和列：栅格在 X 轴方向上的一组像元构成了一行；同样的，Y 轴方向上的一组像元构成了一列。栅格数据中每个像元都有唯一的行列坐标。
分辨率：栅格数据中涉及多种分辨率，如遥感影像的空间分辨率、光谱分辨率、时间分辨率、辐射分辨率。
空间分辨率：空间分辨率也称为像元大小，是单个像元所表示的地面上覆盖的区域的尺寸，单位为米或千米。例如，美国 QuickBird 商业卫星影像一个像元相当于地面面积 0.61m×0.61m，其空间分辨率为 0.61m；Landsat/TM 多波段影像一个像元约覆盖地面面积 28.5m×28.5m，其空间分辨率 28.5m。
表示地表同样大小的面积时，高空间分辨率的影像要比低空间分辨率的影像所需的像元数要多，即像元大小比较小的栅格需要更多的行和列来表示，从而可以显示出地表的更多信息和细节。因而，空间分辨率越高，所存储的地表的细节越多，所需的存储空间也就越大，同时数据处理的时间更长；相反，空间分辨率越低，反映的地表信息越粗糙，但存储空间较小，而且处理速度很快。所以在选择像元大小，即空间分辨率时，要兼顾实际应用对信息详细程度的要求以及对存储和数据处理时的处理时间和速度的需求。

光谱分辨率：是指成像的波段范围，分得愈细，波段愈多，光谱分辨率就愈高。一般来说，传感器的波段数越多，波段宽度越窄，地面物体的信息越容易区分和识别。
时间分辨率：是指同一区域进行相邻的两次遥感观测的最小时间间隔。时间间隔大，时间分辨率就低；反之，时间间隔小，时间分辨率就高。
辐射分辨：是指传感器能分辨的目标反射或辐射的电磁辐射强度的最小变化量。描述传感器在电磁光谱的同一部分中对所查看对象的分辨能力。
波段：是指具有确定波长的电磁波，在遥感技术中常用的波段有可见光波段、近红外波段、中红外波段、热红外波段、远红外波段。影像数据可分为单波段影像和多波段影像两种，单波段影像一般用黑白色的灰度图来描述，多波段多用 RGB 合成的彩色图来描述。可以使用多波段栅格数据集中的任意三个可用波段的组合来创建 RGB 合成图，如下图所示。数字高程模型 (DEM) 即是一个单波段栅格数据集的示例，还有一种有时被称为全色图像或灰度图像的单波段正射影像，多数卫星影像都具有多个波段，通常包含电磁光谱某个范围或波段内的值。

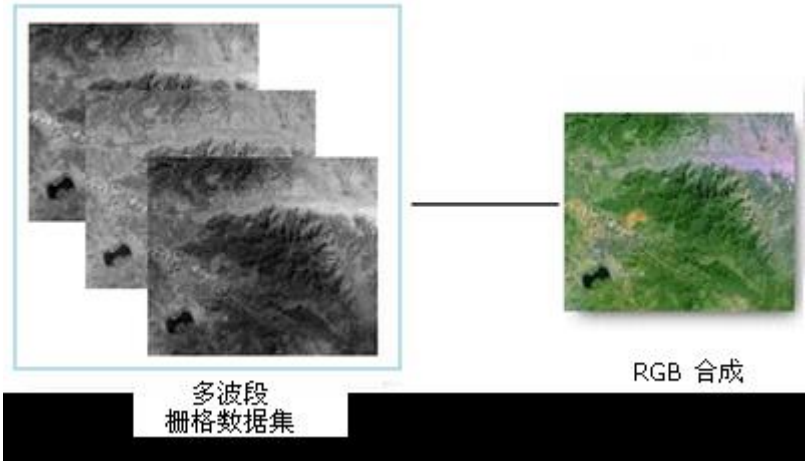


图 单波段合成

地理范围：栅格的地理范围是指其所代表的现实世界的空间范围，是区别于普通图像的一个重要特点。栅格所代表的地理范围可以是普通平面的坐标系，也可以是经纬度的地理坐标系下的一定范围；它可以通过行列数和单元格大小来推算，因此一般只要记录左上角的地理坐标就足够了。

影像金字塔：为减小影像的传输数据量和优化显示性能，有时需要为影像建立影像金字塔。影像金字塔是按照一定规则生成的一系列分辨率由细到粗的图像的集合。影像金字塔技术通过影像重采样方法，建立一系列不同分辨率的影像图层，每个图层分割存储，并建立相应的空间索引机制，从而提高缩放浏览影像时的显示速度。如图 2-2 所示的影像金字塔，底部是影像的原始最高分辨率的表示，为 512×512 图像分辨率，越往上的影像的分辨率越小，分别为 256×256，128×128，顶部是影像金字塔的最低分辨率的图像 64×64，因此这个影像金字塔共有 4 层，即 4 个等级的分辨率。显然影像的图像分辨率越高，影像金字塔的等级越多。

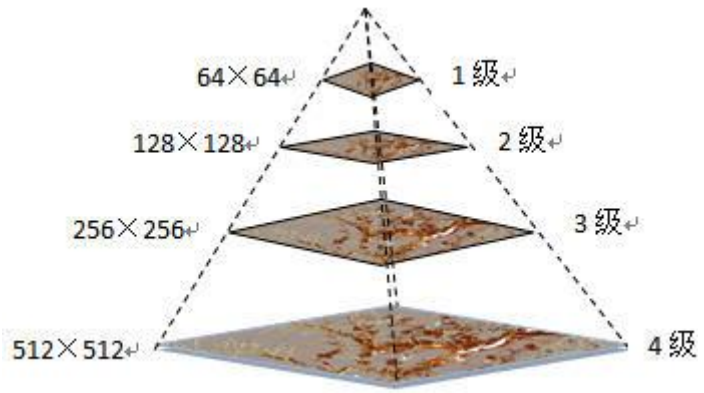


图 影像金字塔

为影像建立了影像金字塔之后，以后每次浏览该影像时，系统都会获取其影像金字塔来显示数据，当您将影像放大或缩小时，系统会自动基于用户的显示比例尺选择最合适的金字塔等级来显示该影像。

栅格数据的来源

栅格数据的来源通常有：

1. 来自于遥感数据：通过遥感手段获得的数字图像，是遥感传感器在特定时间、特定区域的地面景象的辐射和反射能量的扫描抽样，并按照不同的光谱段分光并量化后，以数字形式记录下来的像元值序列。

2. 通过对图片扫描：通过扫描仪对地图进行扫描，转化为栅格形式的数据。扫描得到的图像，将每个像元灰度值与属性表对应，得到像元值，再通过栅格编码，存储为栅格数据。

3. 由矢量数据转换而来：通过矢量数据栅格化技术，将矢量数据转换为栅格数据。

4. 手工方法获取：是指在专题图上均匀地划分网格，逐个确定网格的属性值，最终形成栅格数据文件。

栅格数据类型

对于多种来源的栅格数据，我们可以根据其对现实世界现象的描述的特点将其分成三类：

离散数据，如土地利用分类图，土壤类型图等典型的专题图，以及栅格形式的点、线、面要素等；

连续数据，如高程数据，即数字高程模型（DEM）数据，是最典型的和最常用的连续数据，其他还有降水量数据，污染物浓度数据等；

影像和数字图片，主要是指扫描的地图，绘图以及地面物体的照片等，这一类的栅格数据通常不作为栅格分析的数据源，可作为配准或数字化的底图或要素的属性等。

BcEngineX 支持导入的栅格文件格式

支持导入的栅格文件格式	**文件说明**
TIFF 影像数据（*.tif; *.tiff）	TIFF（Tagged Image File Format）标签图像文件格式，是常见的高位彩色图像格式，GeoTIFF 是一种包含地理信息的 TIFF 文件格式，其信息编码在 TIFF 文件预留的 Tag（标签）之中。
USGDEM 和 GBDEM 栅格文件（*.dem）	是空间数据的交换格式，导入为栅格数据。
ASCII Grid 文件（*.asc）	是 ArcGIS for Desktop Advanced 格网交换文件。
GDAL Virtual 数据格式（*.vrt）	(GDAL) 创建的一种文件格式。这种格式允许从 GDAL 可读的其他数据集中获取虚拟数据集。支持多种栅格数据格式，包含：TIFF、IMG、DEM 等多种格式。
PNG 文件（*.png）	网络图像文件存储格式，对连续像元值的像元有简单压缩。
JPG 文件（*.jpg; *.jpeg）	常见的图片格式，采用 24 位颜色存储单个图像。

影像数据

BcEngineX 支持对*.img、*.tif、*.tiff 格式的影像文件进行预处理，生成能够加载到三维场景的影像数据，并依据其坐标参考信息，添加到三维场景中的球体表面上。此处影像数据包括影像数据集、遥感\航测等直观反映地物地貌的一类栅格数据。

BcEngineX 采用对影像数据缓存进行数据切片生成瓦片数据，按照一定的命名规则后，存放到相应的缓存文件夹中，同时系统会在缓存根目录下生成一个*.xml（*.bci）格式的索引文件。瓦片数据一般为 PNG、JPG、TIF 格式，而*.xml 记录影像数据缓存信息的文件，即影像缓存的配置索引文件。用户在三维环境中加载影像，通过加载影像缓存的索引文件即可加载对应的数据，使用影像缓存可以大大提高海量影像数据的加载和浏览效率。

影像数据基本概念

影像数据是由卫星或飞机上的成像系统获得的影像，多为遥感影像数据。影像数据的每个像元都有一个值，表示传感器探测到像素对应地面面积上目标物的电磁辐射强度，也叫亮度值、灰度值。BcEngineX 支持的影像数据格式有：*.img、*.tif、*.tiff 等。

计算机以二进制记录数据，所以其量化的等级以二进制来划分，即 2n。若用 n 个比特（bit）来记录每个像元，则其灰度值范围可在 0 到 2n-1 之间，如 8-bit 的数据取 28=256 灰度级（其值 0～255）；若规定用 1 比特来记录每个像元，其灰度值仅为 0 和 1，即所谓的二值图像。若用彩色系统来记录图像，根据色度学原理，任何一种彩色均可由红（R）、绿（G）、蓝（B）三原色按适当比例合成，若用 8 比特的 RGB 彩色坐标系来记录像元，可记录 224=1677216 种不同的 RGB 组合。其中，若 RGB 的亮度值分别为 0，0，0，则产生黑色像元；若 RGB 为 255，255，255，则产生白色像元，若 RGB 的亮度值相等，则产生灰度效果。

遥感影像成像方式

遥感影像数据的成像方式主要有航空摄影、航空扫描、微波雷达三种:

- **航空摄影**：摄影成像是通过成像设备获取物体影像的技术。传统摄影成像是依靠光学镜头及放置在焦平面的感光胶片来记录物体影像。数字摄影则通过放置的焦平面的光敏元件，经光/电转换，以数字信号记录物体的影像。探测波段包括：近紫外波段、可见光波段、红外波段。
- **航空扫描**：扫描成像是依靠探测元件和扫描镜对目标物体以瞬时视场为单位进行的逐点、逐行取样，已得到目标物的电磁辐射特征信息，形成一定波段的图像。探测波段包括紫外、红外、可见光和微波波段。
- **微波雷达**：微博成像雷达的工作波长为 1mm-1m 的微波波段，由于微波雷达是一种自备能源的主动传感器，且微博具有穿透云雾的能力，所以微波雷达成像有全天时、全天候的特点。在城市遥感中，这种成像方式对于那些对微波敏感的目标物的识别，具有重要意义。同时，微波对冰、雪、森林、土壤具有一定的穿透能力，对海洋遥感也有特殊意义。探测波段包括微波波段、红外波段。

遥感航天平台

航天遥感影像卫星分为陆地卫星、海洋卫星、气象卫星三种系列，个类型的详细介绍如下:

- **陆地卫星系列**：以探测陆地资源为目的，这类卫星的特点是多波段扫描，地面分辨率为 5-30m。目前，主要的陆地资源卫星有：美国陆地卫星（Landsat）、法国陆地观测卫星（SPOT）、印度遥感卫星（IRS）、中巴资源卫星（CBERS）、日本地球资源卫星（JERS）、美国（IKONOS）、美国（QuickBird）等。
- **海洋卫星系列**：世界海洋卫星包括海洋水色卫星、海洋地形卫星和海洋环境卫星，目前，主要的海洋卫星有：加拿大的 Radarsat 卫星、欧洲 ERS 卫星、美国 Seasat 卫星等。
- **气象卫星系列**：气象卫星广泛应用于国民经济和军事领域，能连续、快速、大面积地探测全球大气变化情况。气象卫星的轨道分为低轨和高轨两种，短周期重复观测，实时性强。目前主要的气象卫星有：美国了 NOAA 卫星、日本的 GMS 气象卫星、中国的 FY 气象卫星等。

注意：关于栅格和影像数据，栅格数据相对于矢量数据集而言的，按照行列存储单一序列的简单数据组织方式，通过同一行列（网格）的不同波段组合，可以携带更多信息；影像数据专门指遥感\航测等直观反映地物地貌的一类栅格数据。

地形数据

BcEngineX 支持对*.asc、*.dem、*.tif 格式的地形文件进行预处理。地形数据是我们进行地形

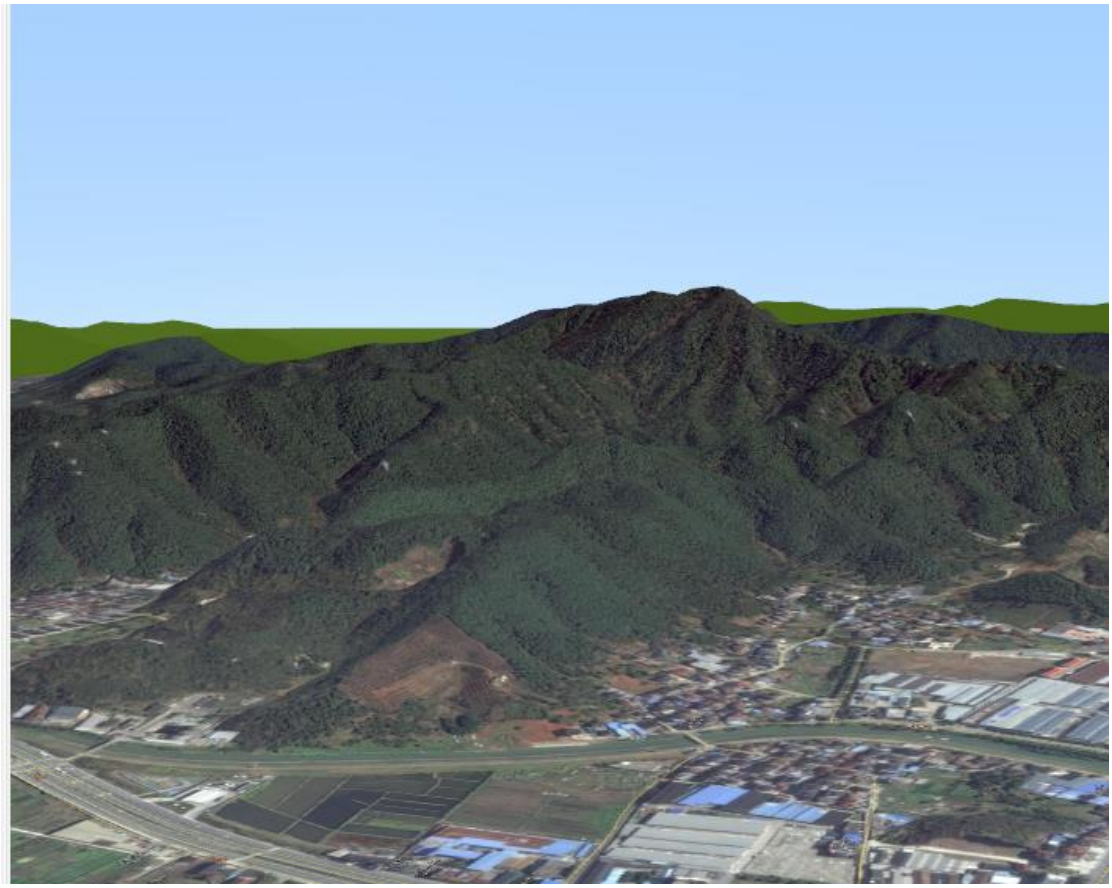
分析的基础。如我们可以利用地形数据提取坡度坡向的基础地形因子，以及进行三维分析等较复杂的地形分析功能。只有构建高质量的地形数据，才能保证我们后续分析结果的可靠。

地形数据基本概念

地形数据，是能够表示地球表面高低起伏状态的数据，即具有高程信息的数据，利用地形数据可以进行地形分析，包括视域分析、淹没分析、通视分析等，BcEngineX 提供了加载地形数据，并对地形数据进行浏览的操作，填挖方操作等。

能够加载到 BcEngineX 中的地形数据，必须通过 BcEngineX 数据预处理，才能获得支持的地形数据，BcEngineX 支持将*.asc(需要转换为 tif)、*.dem(需要转换为 tif) *.tif 地形数据进行预处理，同时，对地形数据进行分层分块，生成缓存目录，同时生成一个 *.xml 文件，该文件对生成的缓存进行了详细描述，记录地形数据缓存的信息文件，即地形缓存的配置索引文件，如缓存层数、缓存地理范围、缓存文件类型等，BcEngineX 通过加载 *.xml 文件，将地形数据作为地形图层显示在三维场景中，并自动基于用户的显示比例尺选择最合适的分层和分块来显示地形数据，使用地形缓存可以大大提高海量地形数据的加载和浏览效率。

地形数据会依据其坐标参考信息，添加到三维场景中的三维球体上，使球体表面真实地模拟地球表面的高低起伏形态。



矢量数据

BcEngineX 支持对*.shp 格式的矢量文件进行加载。矢量数据结构是通过记录空间对象的坐标及空间关系，尽可能精确地表现点、线、多边形等地理实体的空间位置。在矢量数据结构中，点数据可直接用坐标值描述；线数据可用均匀或不均匀间隔的顺序坐标链来描述；面数据可由多个弧段组成的封闭多边形表达。

图1：点　　图2：线　　图3：多边形

矢量数据结构是利用欧几里得集合学中的点、线、面极其组合体来表示地理试题空间分布的一种数据组织方式。这种数据组织方式能最好的逼近地理实体的空间分布特征，数据精度高，数据存储的冗余度低，便于进行地理实体的网络分析，但对于多层空间数据的叠加分析比较困难。

矢量数据的获取方式主要来源于以下几种方式：

- **外业测量**：可利用测量仪器（全站仪、经纬仪、GPS 等）自动记录测量成果，然后转到地理数据库中获得。
- **地图数字化**：是指将传统纸质或其他材料上的地图转换成计算机可识别图形数据，主要有跟踪数字化和扫描数字化两种。
- **栅格数据转换**：利用栅格数据矢量化的技术，将栅格数据转换为矢量数据。
- **数据分析**：可通过空间分析中的叠加、缓冲区分析等操作产生新的矢量数据。矢量数据常应用于城市分区或详细规划、土地管理、公用事业管理等方面。

专题图

BcEngineX 提供针对矢量数据制作专题图的功能，支持的专题图类型如下：

- **单值专题图**：是将专题值相同的要素归为一类，为每一类设定一种渲染风格，如颜色或符号等，专题值相同的要素采用相同的渲染风格，从而区分不同的类别。
- **分段专题图**：专题值按照某种分段方式被分成多个范围段，要素根据各自的专题值被分配到其中一个范围段中，在同一个范围段中的要素使用相同的颜色或符号风格进行显示。

- **标签专题图**：主要用于在矢量图层上做标注，即用专题值对点、线、面等对象做标注，如标注景区名称等信息。

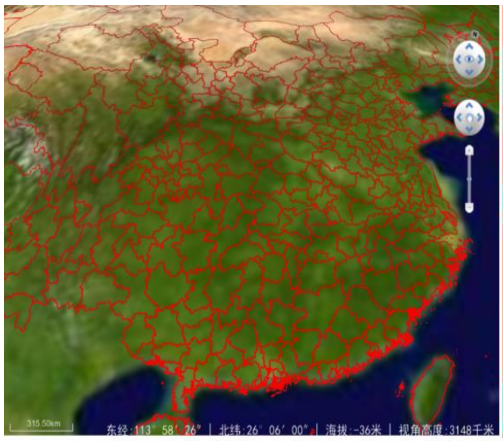


图 普通图层

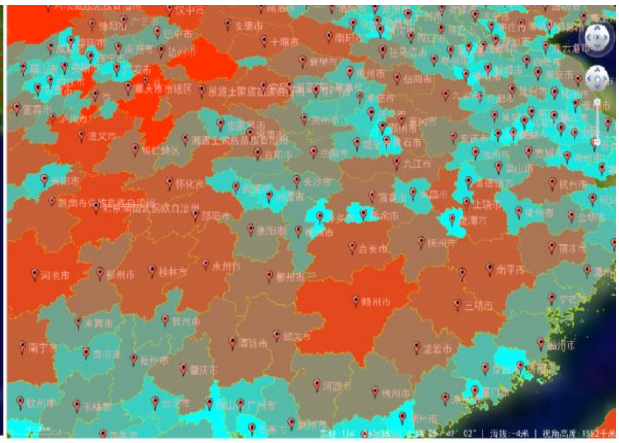


图 专题图层

三维操作

讲一下三维场景应该如何操作，包含导航栏的操作，普通场景和地球场景

BcEngineX 三维场景的三维操作提供了三维浏览的功能，包括放大、缩小、倾斜、拉平竖起、旋转等，可以通过三维窗口提供的三维导航工具、也可以通过提供鼠标键盘组合操作来实现浏览操作，还可以通过接口来实现这些操作。

地球场景下：

拉平竖起（Pitch）操作

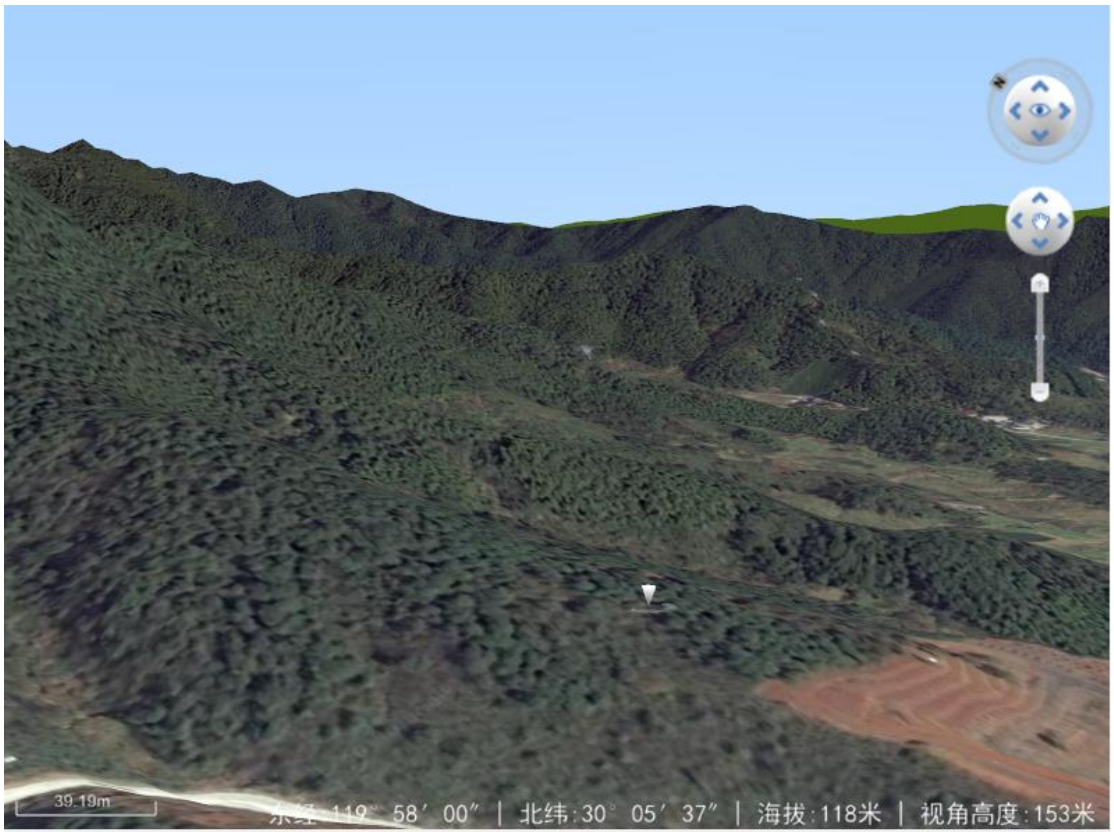


图 1 进行 Pitch 竖起操作后（地球场景）

图 2 进行 Pitch 拉平 操作后（地球场景）

旋转（Roll）操作

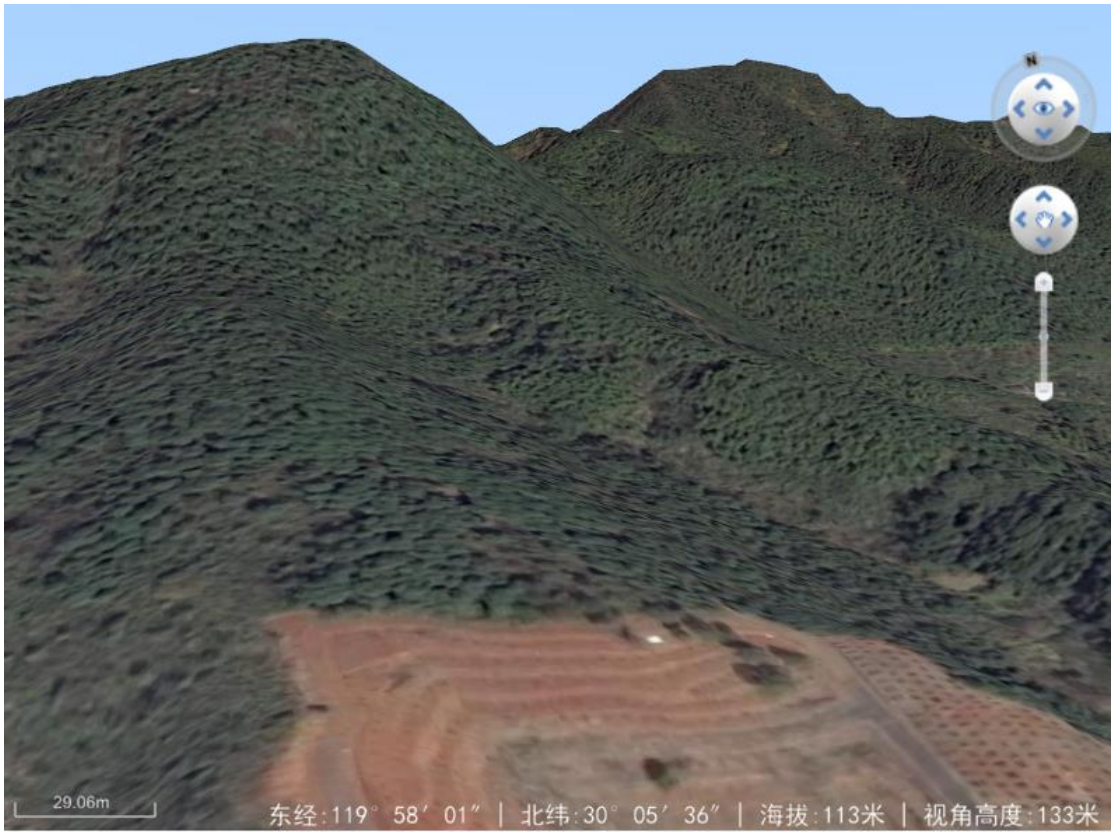


图 5 进行 roll 旋转操作后（地球场景）

普通场景下：

拉平竖起（Pitch）操作

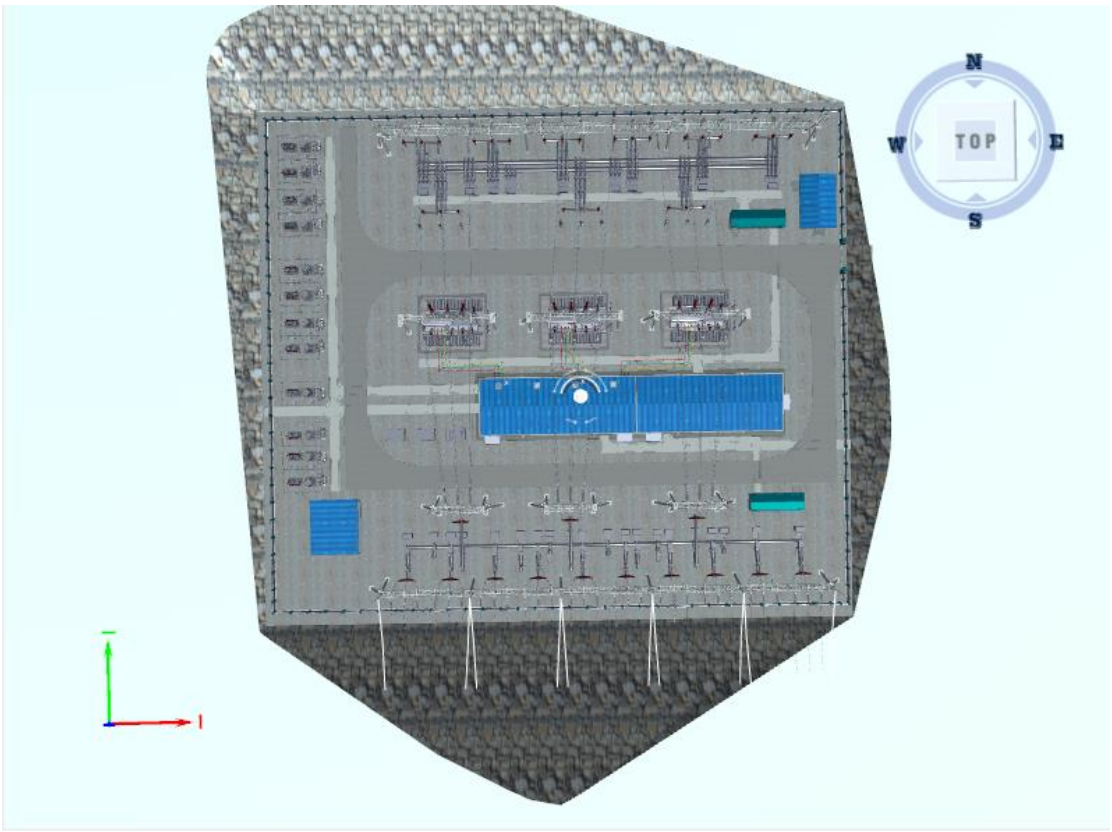


图 3 进行 Pitch 竖起操作后（普通场景）

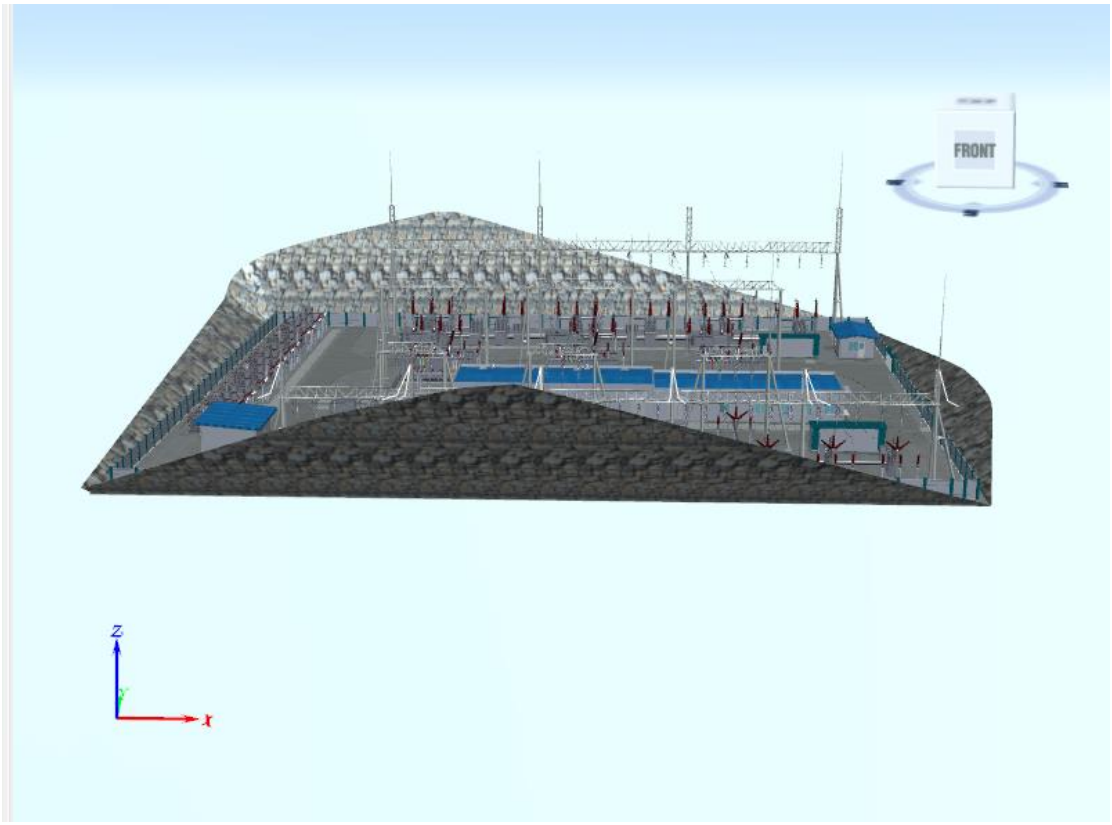


图 4 进行 Pitch 拉平 操作后（普通场景）

旋转（Roll）操作

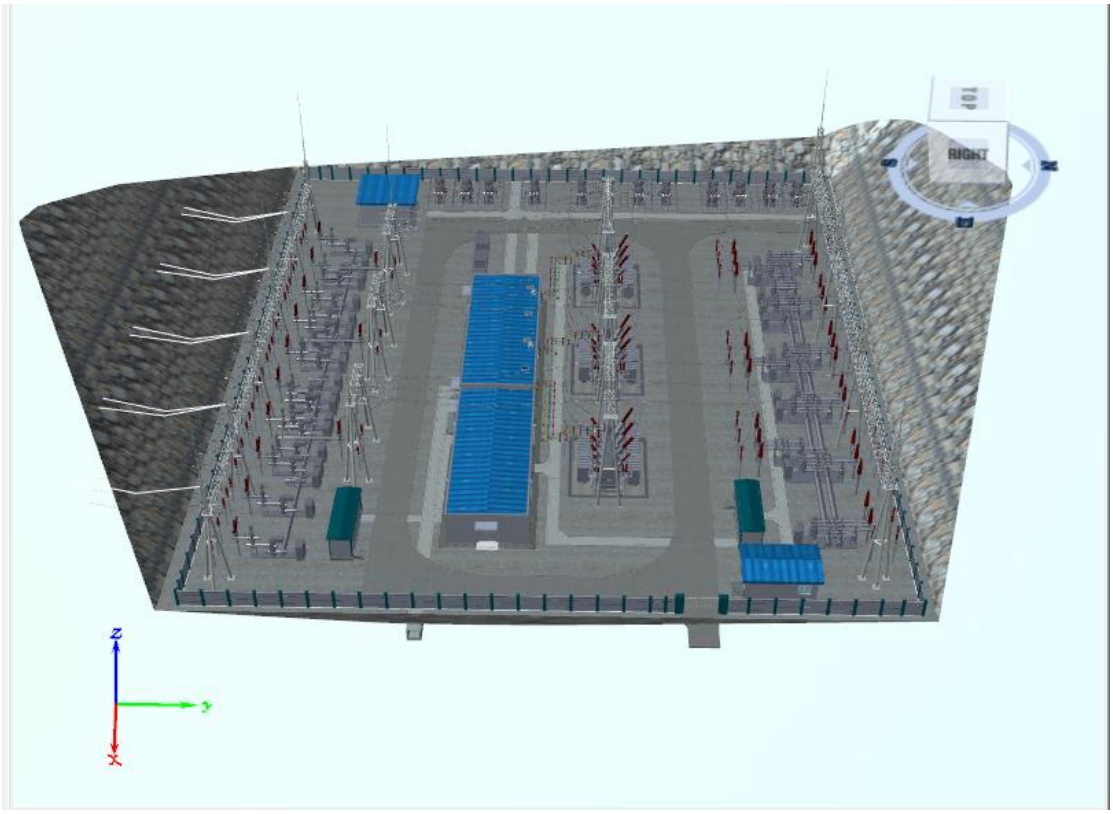


图 6 进行 roll 旋转操作后（普通场景）

表 地球场景浏览功能的鼠标键盘操作及对应的导航工具

浏览功能	鼠标操作	键盘操作	导航工具条
漫游	鼠标左键按下拖动	上下左右光标键	

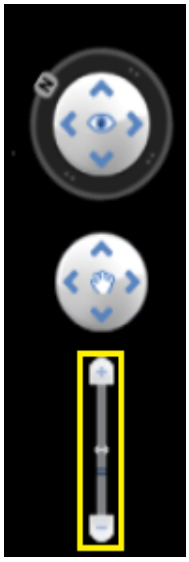
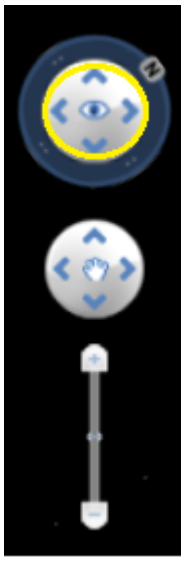
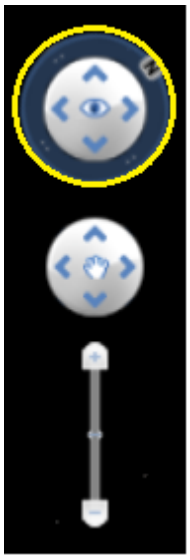
放大缩小	鼠标中键滚轮或鼠标右键按下上下拖动	PageUp 和 PageDown 键	
三维地图场景进行倾斜	按住鼠标中键上下拖动	Shift 键 + 上下光标键	
绕场景中心旋转	按住鼠标中键左右拖动	Shift 键 + 左右光标键	

表 普通场景浏览功能的鼠标键盘操作及对应的导航工具

浏览功能	鼠标操作	键盘操作	导航工具条

绕场景中心倾斜和旋转	按住鼠标中键上下左右拖动		

UGX

UGX 是基于 GLTF 格式标准扩展重新设计的一种新的模型数据格式--UGX。UGX 使用 JSON 格式进行描述，也可以编译成二进制的内容 UGB。UGX 可以包括场景、摄像机、动画等，也可以包括网格、材质、纹理，甚至包括了渲染技术、着色器以及着色器程序。

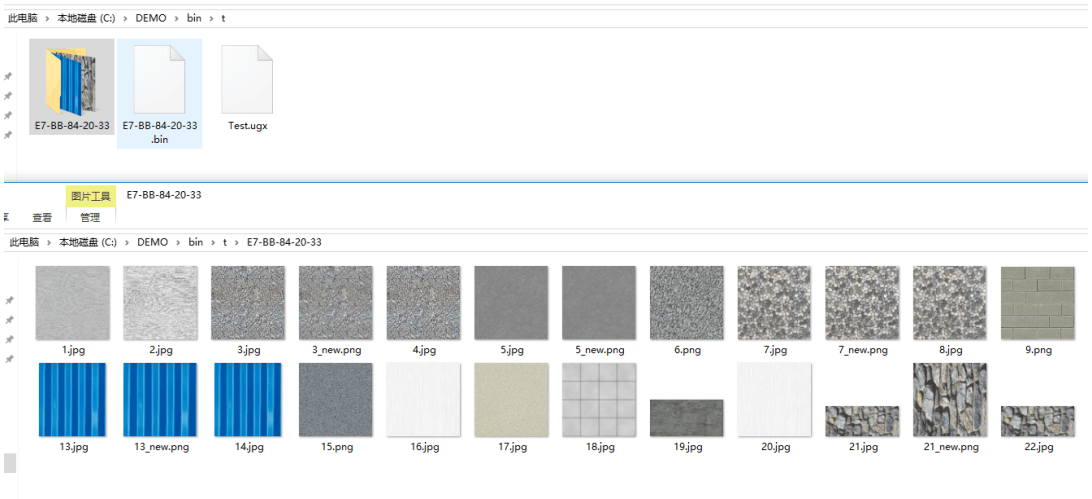
文件组织结构

UGX 文件主要由以下几个部分组成：

(1) UGX 文件(*.ugx)：JSON 文件，描述整个数组组织结构、场景节点、图元集、二进制文件及图片文件的相对路径；

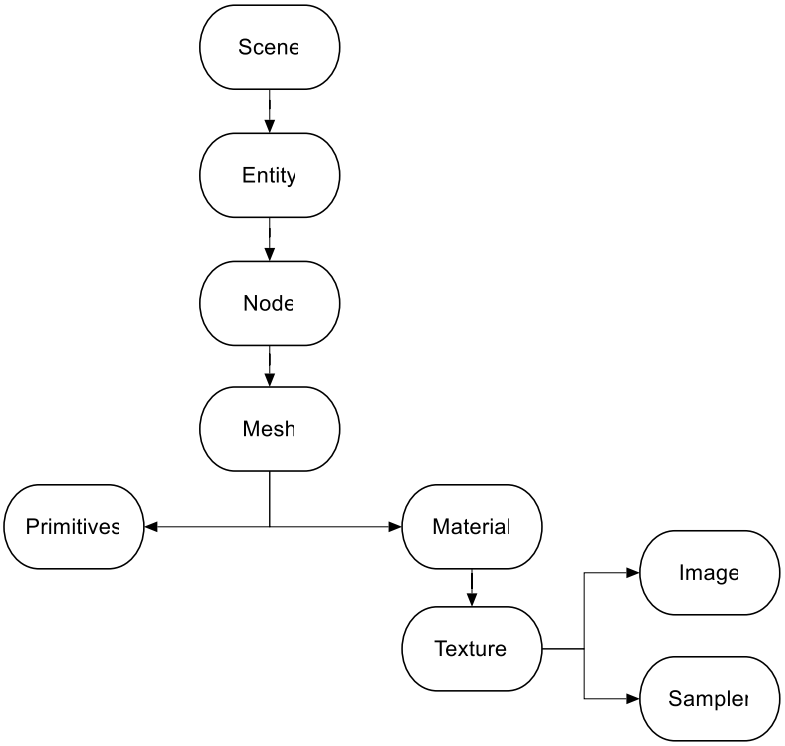
(2) 二进制文件(*.bin)：几何、动画、纹理的真正数据文件；

(3) 图片文件夹：纹理图片文件。



数据逻辑结构

UGX 的数据逻辑结构主要包括以下几个部分：
(1) Scene：数据的入口点，由 Entity 集合组成 Scene，一个 UGX 文件默认只有一个 Scene；
(2) Entity：一个实体对象对应的 Node 索引；
(3) Node：定义一个节点，包含 L0D 信息、平移、旋转、缩放及矩阵信息，也可以包含子 Node 或者 Mesh 索引；
(4) Mesh：定义实体对象对应的 Node 的几何信息，包含多个可绘制图元集；
(5) Primitive：可绘制图元集，包含顶点、法线、纹理坐标、绘制索引、颜色、绘制模式等信息；
(6) Material：定义的外观的参数，包含 PBR 材质模型参数、KHR 材质模型参数、法线纹理、双面渲染等；
(7) Texture：定义纹理图片对象、纹理采样器对象的索引；
(8) Image：定义纹理图片路径、格式信息；
(9) Sampler：定义纹理图片的环绕方式、过滤方式。

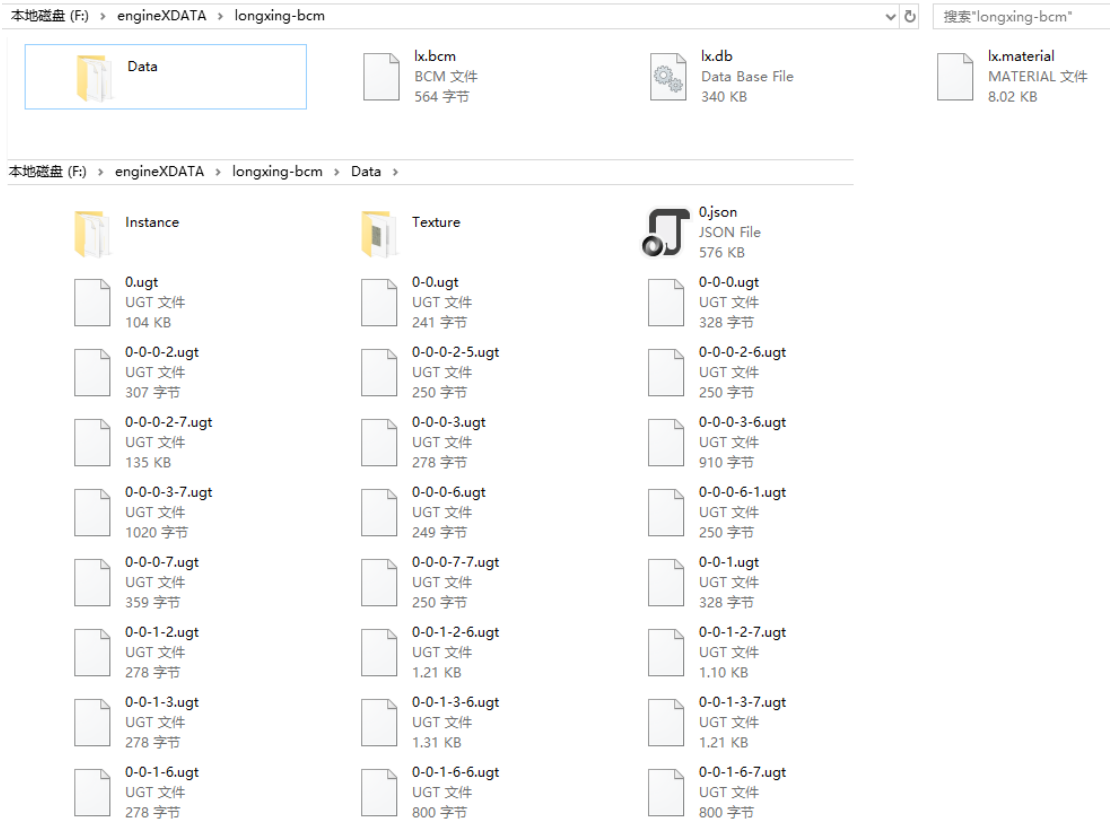


UGT

UGT 即 UGTile 是在 UGX 数据的基础上重新对三维模型按空间索引的组织，将 UGX 模型数据处理成 UGT 瓦片缓存格式，提高数据的加载浏览效率的同时大大优化占用内存。

文件组织结构
UGT 文件主要由以下几个部分组成：
(1) bcm 文件(*.bcm)：JSON 文件，描述整个数组组织结构、场景节点、图元集、二进制文件及图片文件的相对路径；
(2) db 文件(*.db)：存储的是实体的 GUID、包围盒、有向包围盒信息；

(3) material 文件(*.material)：图层材质文件，存储的是图层用到的材质信息；
(4) 二进制文件(*.ugt)：几何、动画、实例化对象的真正数据文件；
(5) 图片文件夹：纹理图片文件。



数据特点

(1) 八叉树深度设置：设置八叉树的深度，即数据总共分多少级；同时设置八叉树块数据的视距范围集合

(2) Lod 级别设置：设置 LOD 数据的级别（目前数据的 LOD 的层级默认是 3 级）

(3) Lod 丢弃设置：设置每级八叉树块（树深度 2 到树深度 4 的块），在分块时，丢弃实体数据的参考体积值，在对应树深度时丢弃设置的体积值数据，通过设置 LOD 切换距离来控制在一定距离范围显示 LOD 层级的某一层。假设 LOD 切换距离为 200 米，离相机 200 米以内的模型将显示 LOD 第 0 级（最精细层）；200-400 米的模型显示第 1 级（次精细层），以此类推。

(4) 空间分块设置：数据过大无法在物理内存中进行处理，则可以将其进行细分成四个等大的分块处理。然后再对子分块处理。如果分块中的数据任然过大，则可以再进一步细分成 4*n 块处理

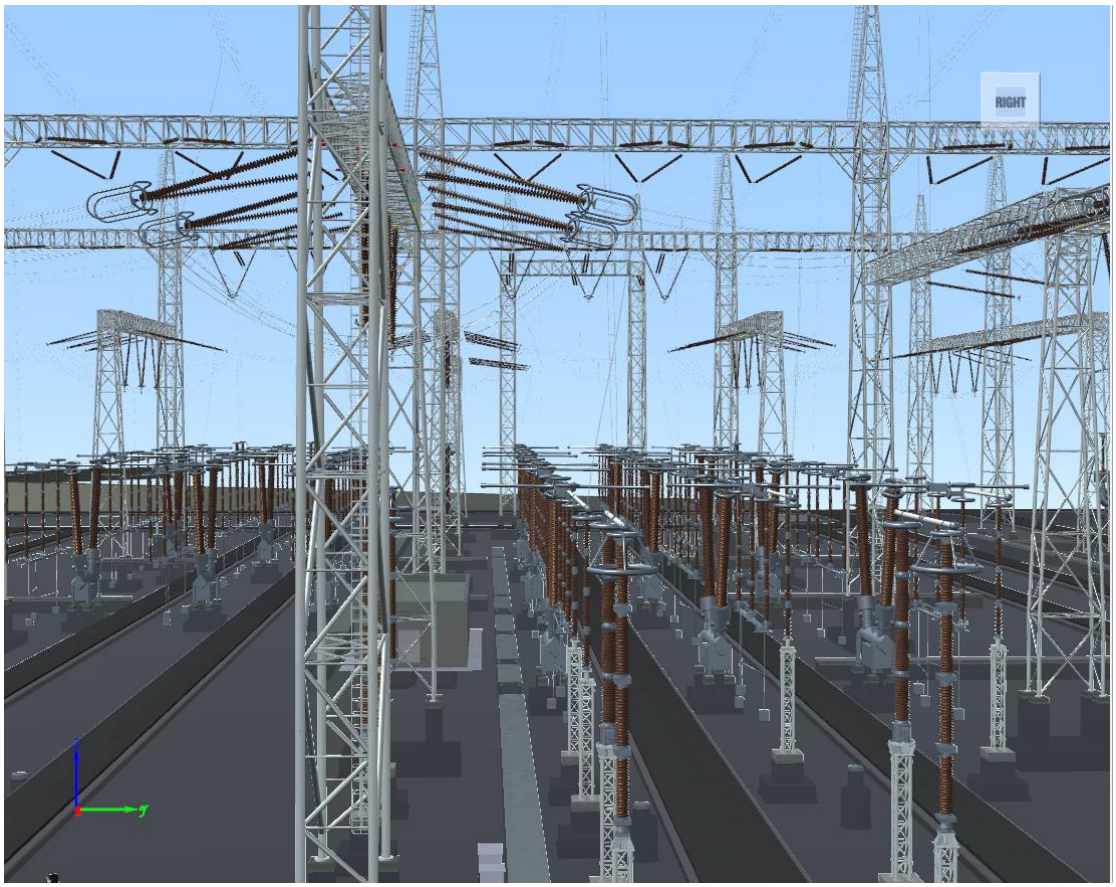
(5) 数据压缩设置：将 ugt 瓦块以 7z 压缩方式压缩成 ugtz 格式减小占用内存，提高在服务端访问下载数据时的速率。

应用专题

BIM

BIM 是一种高密度模型，它精准、详尽地展示了建筑物内、外部的模型，其数据量惊人，从而导致 BIM 模型的三维场景性能有待提高。某些 BIM 模型存在大量冗余的三角面，如桥梁墩柱、电器等。使用 BIM 三角网简化功能，实现对图层中所有模型对象或选中模型对象的三角网进行简化，降低内存的占用，满足大体量数据的性能需要。

BcEngineX SDK 提供支持 BIM 生成缓存，支持对批量 BIM 模型高清渲染。



倾斜摄影

倾斜摄影（oblique image）是指由一定倾斜角的航摄相机所获取的影像。倾斜摄影技术是国际测绘遥感领域近年发展起来的一项高新技术，它颠覆了以往正射影像只能从垂直角度拍摄的局限。通过在同一飞行平台上搭载多台传感器，同时从一个垂直、四个倾斜等五个不同的角度采集影像，获取地面物体更为完整的信息。

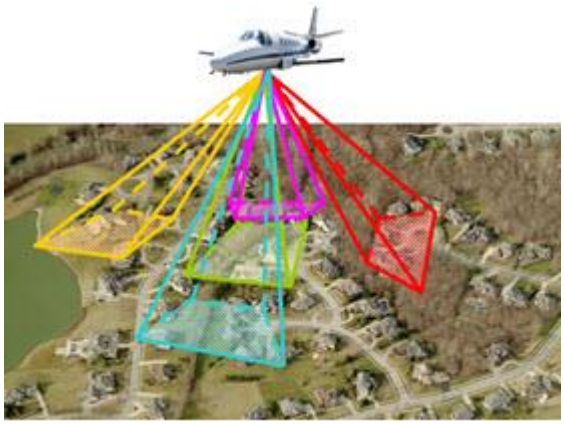


图 倾斜摄影采集

倾斜摄影技术有以下 4 个特点：

- 反映地物周边真实情况：可以获取多个视点和视角的影像，从而得到更为详尽的侧面信息，具有较高的分辨率和较大的视场角，地物遮挡现象较为突出。
- 倾斜摄影可实现单张影像测量：通过配套软件的应用，可基于成果影像进行包括高度、长度、面积、角度、坡度等测量，扩展了倾斜摄影技术在行业中的应用。
- 提升城市三维建模效率：针对各种三维数字城市的应用，利用航空摄影大规模成图的特点，加上从倾斜摄影影像提取及贴纹理的方式，能够有效的降低三维建模成本。
- 数据量小易于网络发布：相较于三维 GIS 技术应用庞大的三维数据，应用倾斜摄影技术获取的影像的数据量要小得多，其影像的数据格式可采用成熟的技术快速进行网络发布，实现共享应用。

倾斜摄影的应用：

航空倾斜影像不仅能够真实地反应地物情况，而且还通过采用先进的定位技术，嵌入精确的地理信息、更丰富的影像信息、更高级的用户体验，极大地扩展了遥感影像的应用领域。该技术可广泛应用于应急指挥、国土安全、城市管理、房产税收等领域。

BcEngineX SDK 提供支持倾斜摄影数据生成缓存，支持对倾斜摄影模型加载浏览，同时支持对倾斜摄影模型压平、挖洞等操作。

图 倾斜摄影浏览

点云

点云数据（Point Cloud）是某个坐标系下的点的数据集。点包涵了丰富的信息，包括三维坐标 X、Y、Z、颜色、分类值、强度值、时间等等，不一一列举。点云可以将现实世界原子化，通过高精度的点云数据可以还原现实世界。

点云数据来源
三维激光扫描进行数据采集：通过 LiDAR（Light Detection And Ranging 中文翻译激光探测与测量）获取的数据就是点云数据，同时也对点云数据进行处理加共及应用。LiDAR 获取数据主要分为三大类：星载、机载和地面。

二维影像进行三维重建：在重建过程中获取点云数据。

三维模型计算获取：通过三维模型计算获取点云数据。

BcEngineX SDK 提供支持点云数据生成缓存，支持对点云数据加载浏览，同时支持对点云压平、挖洞等操作。

图 普通点云

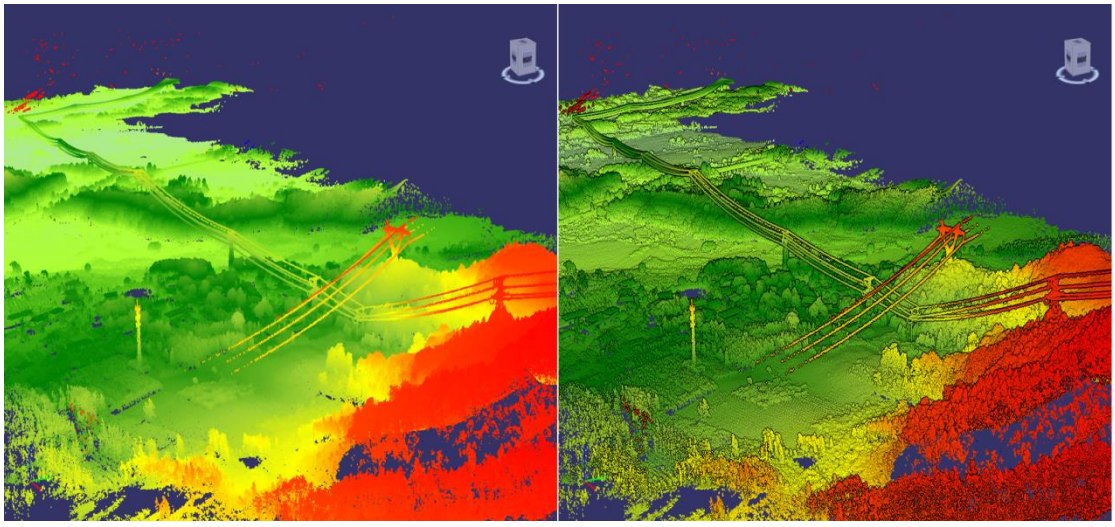


图 点云（左）点云 EDL 渲染（右）

三维管线

地上地下各类管网、管线是一个城市重要的基础设施，它不仅具有规模大、范围广、管线种类繁多、空间分布复杂、变化大、增长速度快、形成时间长等特点，更重要的它还承担着信息传输、能源输送、污水排放等与人民生活息息相关的重要功能，也是城市赖以生存和发展的物质基础。

随着我国城镇化进程的不断深入，传统的二维管理模式已根本无法满足对管网、管线大数据信息分析、表达、应用的实际需要，三维管线管理逐渐替代二维管理模式。二维系统只有点、线表示管网，系统建设成本并不高，但是在三维场景中，需要使用三维管点模型、管

线模型来展示管网系统，模型建设成本较高，管点模型的管口与管线管道的匹配效果差，并且业务属性信息需要从点、线数据中再次录入到三维模型中，同时维护两套数据而无法满足实时更新的需求。

BcEngineX SDK 提供基于二三维一体化技术的自适应管点符号，可由二维的点、线数据生成三维管线数据，根据管网走向、管线截面自动实时管点建模，快速构建三维场景的同时大幅降低三维管网场景的建设成本，并且提高了三维管线、三维管点的显示性能，系统资源占用减少使得数据承载力大幅提升，进而满足更加庞大复杂的三维管网系统展示、管理及应用。

三维管线场景通常由三维管线和三维管点两类组成，具体包括以下元素：

- 管线：包括圆管、方沟、管块、竖管等；
- 管点：包括特征点、井和附属设施三大类；
  - 特征点：包括弯头、直通、三通、四通、五通、多通、变径、盖堵、管帽等；
  - 井：包括方井、圆井、井室、偏心井、雨篦等；
  - 附属设施：包括阀门、水表、消防栓、控制柜、变压器、分线箱等；

不同元素采用不同方式实现快速构建三维管线场景，通常采用线型符号构建三维管线、自适应管点符号构建三维管点，而部分特殊特征点、井和附属设施采用模型符号展示，如下图所示：

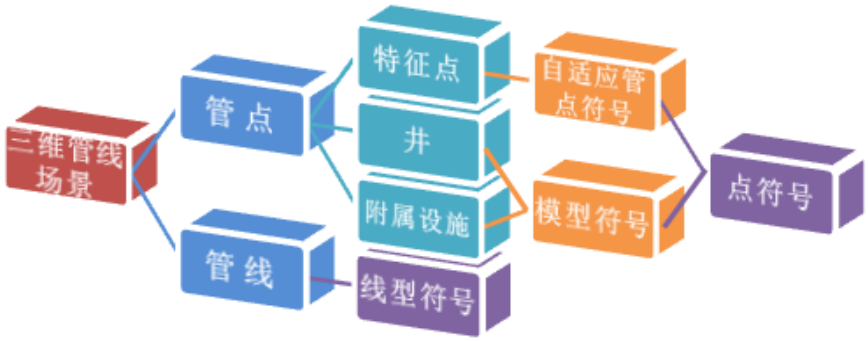


图 三维管线场景组成及展示方式

范例程序说明

场景属性控制

第一步 创建一个工程

创建一个工程，命名为 SceneControlSample，参考产品入门

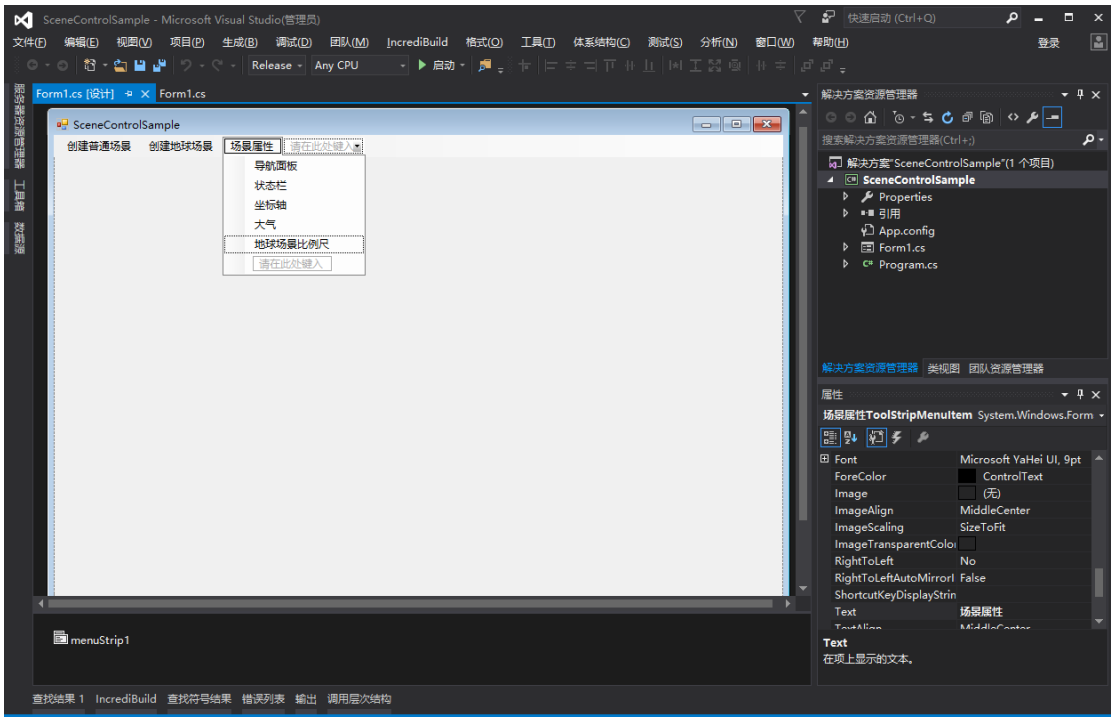
在窗体中加入 menuStrip 工具条上添加 三个 ToolStripMenuItem 的属性值（其余属性取默认值即可）：

Name	Text
NewCommon_MenuItem	创建普通场景
NewEarth_MenuItem	创建地球场景
SceneManager_MenuItem	场景属性

在 SceneManager_MenuItem 下添加 5 个子 ToolStripMenuItem 的属性值（其余属性取默认值即可）：

Name	Text
Navigation_MenuItem	导航面板
StatusBar_MenuItem	状态栏
Axis_MenuItem	坐标轴
Atmosphere_MenuItem	大气
EarthScale_MenuItem	地球场景比例尺

再在窗体中添加 panel 窗体，将 panel1 窗体的属性“Dock”设置为”Fill”后如下图：



第二步 编写代码

在正式编写示范程序之前，需要添加对 BcEngineX SDK 的 EngineX.SceneControl 程序集的引用，进入代码编辑器，在最顶端添加如下代码：

```
using EngineX.Render;
```

1.创建普通场景代码

双击 SceneControlSample 窗体中的 NewCommon_MenuItem(创建普通场景)按钮，进入代码编辑器。或者直接在属性界面，手动添加 NewCommon_MenuItem 的 Click 事件输入事件名“NewCommon_MenuItem_Click ”。在“NewCommon_MenuItem_Click "事件中插入如下代码：

```
//******创建一个普通场景******
private void NewCommon_MenuItem_Click(object sender, EventArgs e)
{
    if(bc_sceneControl != null)
    {
        bc_sceneControl.Dispose();
        bc_sceneControl = null;
    }
    //创建一个普通场景
    bc_sceneControl = new SceneControl(SceneType.ST_COMMON);
    bc_sceneControl.Dock = DockStyle.Fill;
    this.panel1.Controls.Add(bc_sceneControl);

    //导航盒默认勾选
    this.Navigation_MenuItem.Checked=bc_sceneControl.Scene3D.NavigationPanelVisible;
    //状态栏，普通场景没有状态栏默认不勾选
    this.StatusBar_MenuItem.Checked = false;
    //坐标轴默认勾选
    this.Axis_MenuItem.Checked = bc_sceneControl.Scene3D.AxisVisible;
    //大气默认勾选
    this.Atmosphere_MenuItem.Checked=bc_sceneControl.Scene3D.Atmosphere.IsVisible;
}
```

2.创建地球场景

双击 SceneControlSample 窗体中的 NewEarth_MenuItem (创建地球场景)按钮，进入代码编辑器。或者直接在属性界面，手动添加 NewEarth_MenuItem 的 Click 事件输入事件名“NewEarth_MenuItem_Click ”。在“NewEarth_MenuItem_Click "事件中插入如下代码：

```
//******创建一个地球场景******
private void NewEarth_MenuItem_Click(object sender, EventArgs e)
{
    if (bc_sceneControl != null)
    {
        bc_sceneControl.Dispose();
        bc_sceneControl = null;
    }
    //创建一个 WGS84 坐标系的地球场景
    bc_sceneControl = new SceneControl(SceneType.ST_WGS84);
```

```
        bc_sceneControl.Dock = DockStyle.Fill;
        this.panel1.Controls.Add(bc_sceneControl);
        //导航盒默认勾选
        this.Navigation_MenuItem.Checked= bc_sceneControl.Scene3D.NavigationPanelVisible;
        //状态栏默认勾选
        this.StatusBar_MenuItem.Checked= bc_sceneControl.Scene3D.StatusBarVisible;
        //坐标轴，地球场景没有坐标轴不勾选
        this.Axis_MenuItem.Checked = false;
        //大气默认勾选
        this.Atmosphere_MenuItem.Checked= bc_sceneControl.Scene3D.Atmosphere.IsVisible;
        //坐标轴默认勾选
        this.EarthScale_MenuItem.Checked= bc_sceneControl.Scene3D.EarthScaleVisible;
}
```

3.添加导航面板、状态栏、坐标轴、大气、比例尺控制代码

以导航面板控制为例，双击 SceneControlSample 窗体中的 Navigation_MenuItem (导航面板)按钮，进入代码编辑器。或者直接在属性界面，手动添加 Navigation_MenuItem 的 Click 事件输入事件名“Navigation_MenuItem_Click ”。在“Navigation_MenuItem_Click ”事件中插入如下代码：

```
//******开关场景导航盒******
private void Navigation_MenuItem_Click(object sender, EventArgs e)
{
    if (bc_sceneControl != null)
    {
        this.Navigation_MenuItem.Checked = !this.Navigation_MenuItem.Checked;
        bc_sceneControl.Scene3D.NavigationPanelVisible= !bc_sceneControl.Scene3D.Navigat
        ionPanelVisible;
    }
}
```

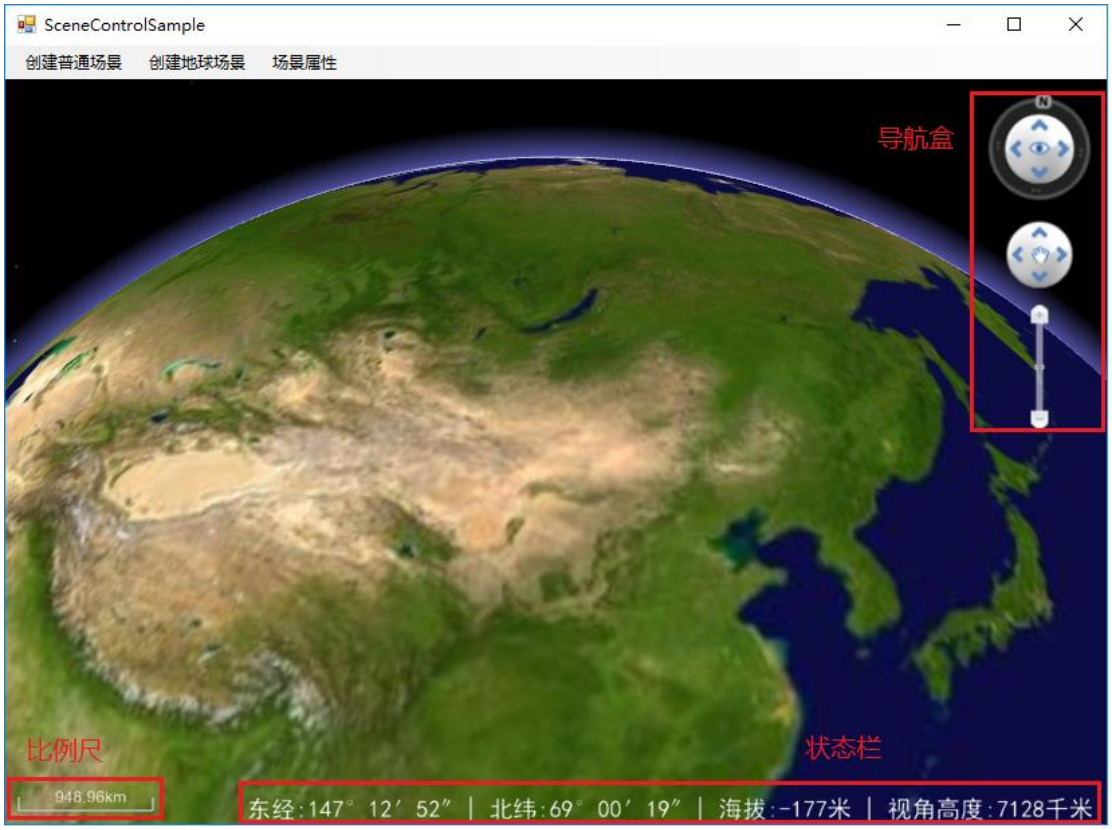
同添加导航面板控制，添加 StatusBar_MenuItem（状态栏）、Axis_MenuItem（普通场景坐标轴）、Atmosphere_MenuItem（大气）、EarthScale_MenuItem（地球场景比例尺）代码：

```
//******开关场景状态栏******
private void StatusBar_MenuItem_Click(object sender, EventArgs e)
{
    if (bc_sceneControl != null)
    {
    this.StatusBar_MenuItem.Checked = !this.StatusBar_MenuItem.Checked;
    bc_sceneControl.Scene3D.StatusBarVisible = !bc_sceneControl.Scene3D.StatusBarVisible;
    }
}
//******开关场景坐标轴******
```

```
private void Axis_MenuItem_Click(object sender, EventArgs e)
{
    if (bc_sceneControl != null)
    {
        this.Axis_MenuItem.Checked = !this.Axis_MenuItem.Checked;
        bc_sceneControl.Scene3D.AxisVisible = !bc_sceneControl.Scene3D.AxisVisible;
    }
}
//******开关场景大气******
private void Atmosphere_MenuItem_Click(object sender, EventArgs e)
{
    if (bc_sceneControl != null)
    {
        this.Atmosphere_MenuItem.Checked = !this.Atmosphere_MenuItem.Checked;
        bc_sceneControl.Scene3D.Atmosphere.IsVisible= !bc_sceneControl.Scene3D.Atmospher
        e.IsVisible;
    }
}
//******开关球场景比例尺******
private void EarthScale_MenuItem_Click(object sender, EventArgs e)
{
    if(bc_sceneControl != null)
    {
      this.EarthScale_MenuItem.Checked = !this.EarthScale_MenuItem.Checked;
      bc_sceneControl.Scene3D.EarthScaleVisible= !bc_sceneControl.Scene3D.EarthScaleVisible;
    }
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“创建地球场景”按钮，打开，一个坐标系为 WGS84 类型的地球场景，现在即可通过点击导航盒、状态栏、比例尺按钮控制场景属性显隐。



另外可以通过点击大气按钮设置场景天空盒显隐，如下有天空盒与无天空盒效果对比：

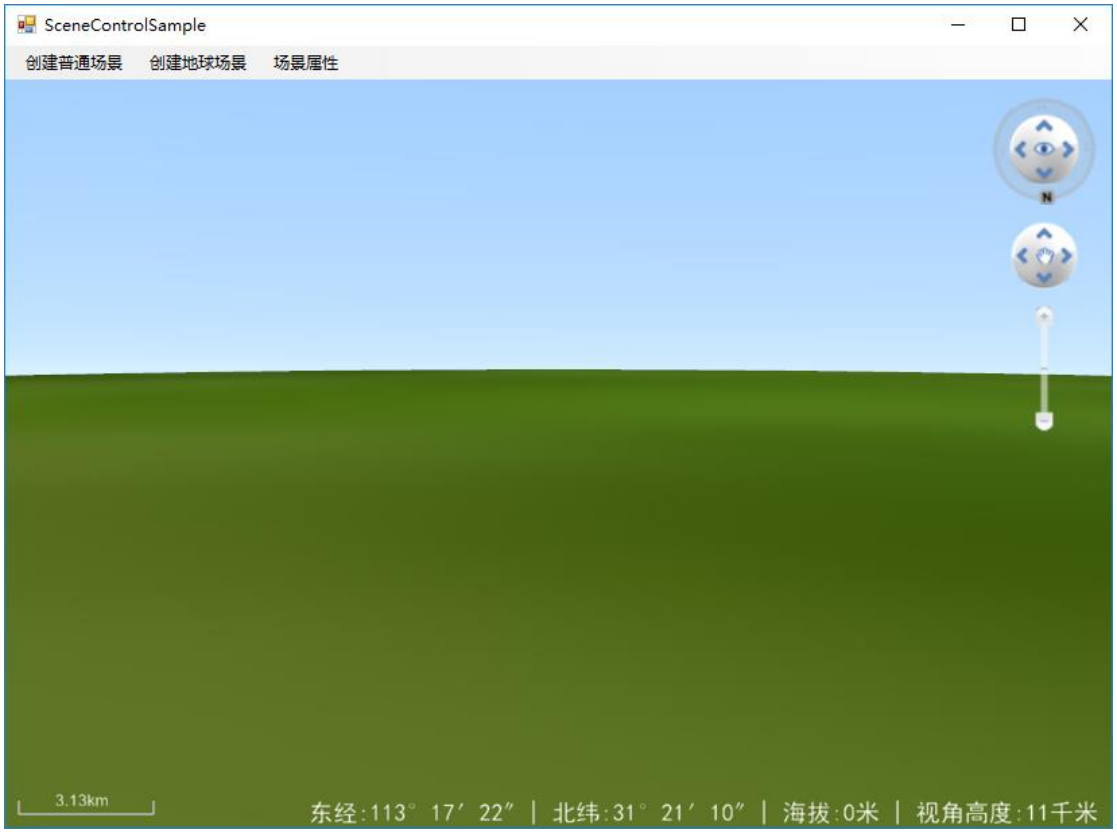


图 有天空盒效果

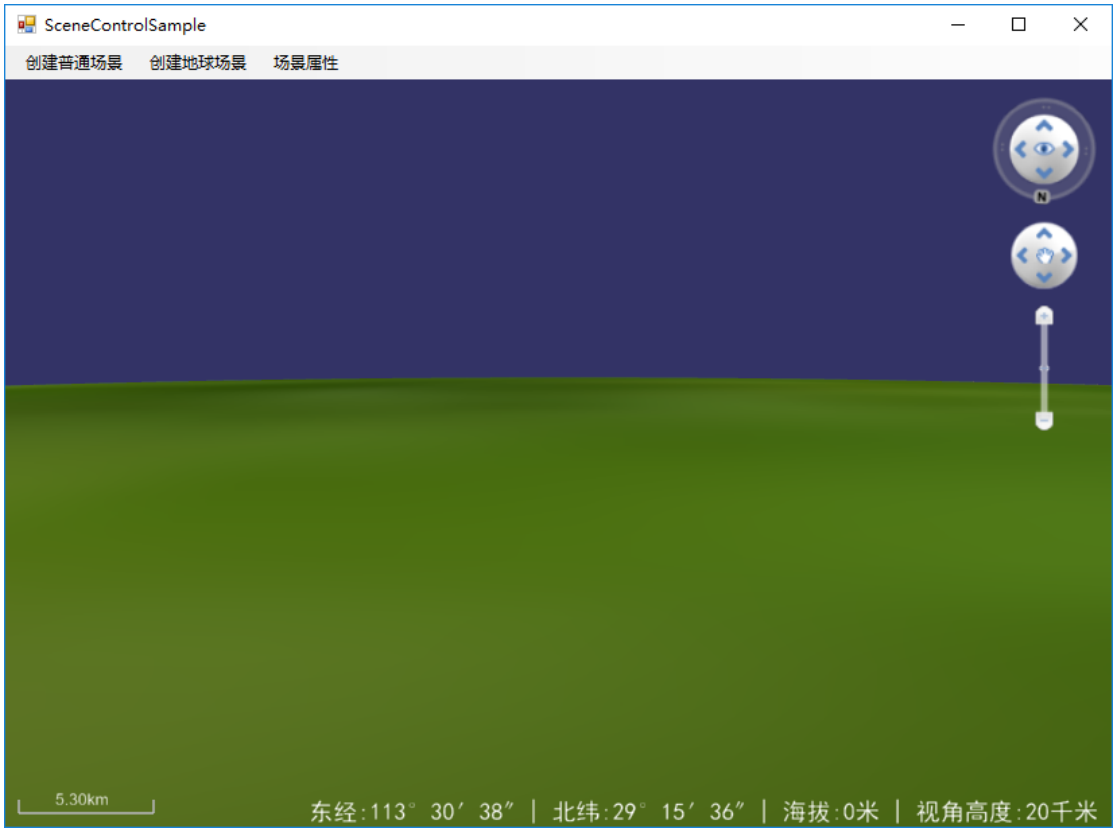


图 无天空盒效果

UGX 和 UGT 数据加载

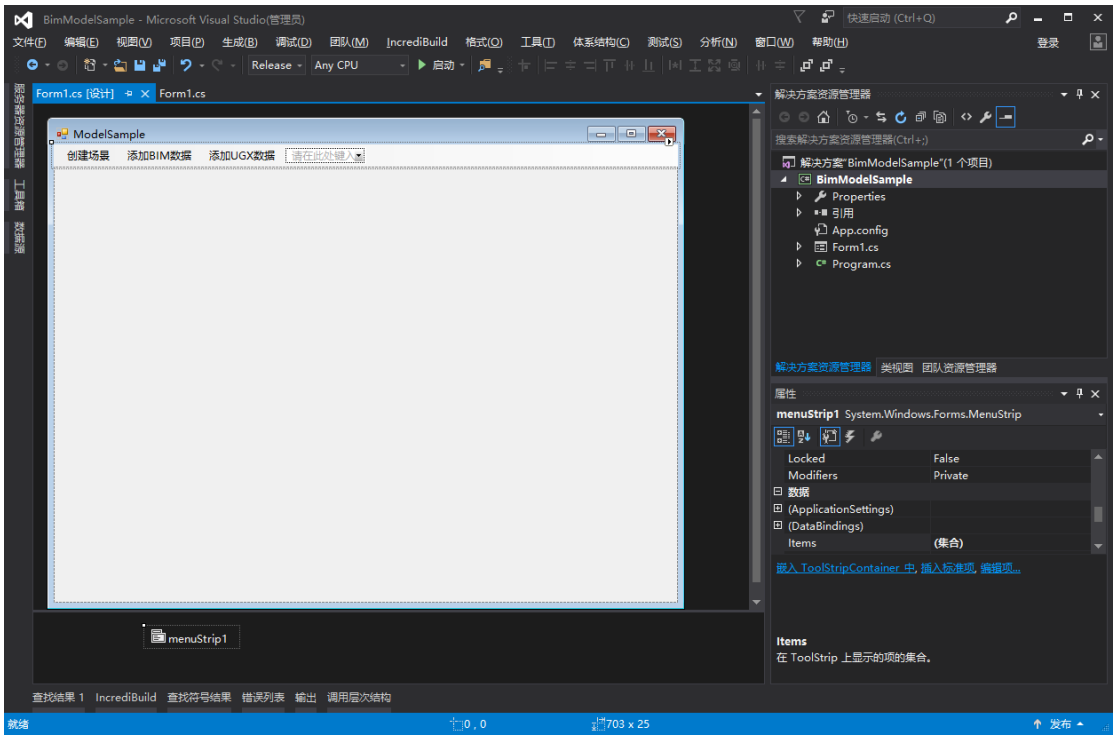
第一步 创建一个工程

创建一个工程，命名为 ModelSample，参考产品入门

在窗体中加入 menuStrip 工具条上添加 三个 ToolStripMenuItem 的属性值（其余属性取默认值即可）：

Name	Text
CreateEarth_MenuItem	创建普通场景
AddBIM_MenuItem	添加 bim 数据
SceneManager_MenuItem	添加 ugx 数据

再在窗体中添加 panel 窗体，将 panel1 窗体的属性“Dock”设置为”Fill”后如下图：



第二步 编写代码

在正式编写示范程序之前，需要添加对 BcEngineX SDK 的 EngineX.SceneControl 程序集的引用，进入代码编辑器，在最顶端添加如下代码:

```
using EngineX.Render;
```

1.创建场景代码

参考 场景属性控制-代码编写-2.创建地球场景

2.添加 bim 数据代码

双击 ModelSample 窗体中的 AddBIM_MenuItem (添加 BIM 数据)按钮，进入代码编辑器。或者直接在属性界面，手动添加 AddBIM_MenuItem 的 Click 事件输入事件名“AddBIM_MenuItem_Click ”。在“AddBIM_MenuItem_Click ”事件中插入如下代码:

```
//******添加一份 BIM 数据******
private void AddBIM_MenuItem_Click(object sender, EventArgs e)
{
    //判断
    if (bc_sceneControl != null)
    {
        using (OpenFileDialog ofd = new OpenFileDialog())
        {
          ofd.Filter = "(*.bcm)|*.bcm|All files(*.*)|*.*";
          if (ofd.ShowDialog() == DialogResult.OK)
          {
              //读取文件路径
```

```
                string strFilePath = ofd.FileName;
                //添加 bim 模型数据
                bc_modelLayer = bc_sceneControl.Scene3D.ModelLayerManager.Add(strFilePath,
                LayerType.LT_BIM);
                //设置地球场景加载位置：经度，纬度，高程
                bc_modelLayer.Center = new Point3D(119.9757004, 30.09651184, 120);
                //******设置相机信息并飞行到指定相机位置******
                //经度
                bc_camera.X = 119.9757004;
                //纬度
                bc_camera.Y = 30.09651184;
                //高层值
                bc_camera.Z = 40;
                //相机的方位角(以度为单位)
                bc_camera.Heading = 60;
                //相机的俯仰角(以度为单位)
                bc_camera.Tilt = -90;
                //相机与观察点的距离值
                bc_camera.Distance = 500;
                //飞行到指定相机位置
                bc_sceneControl.Scene3D.FlyTo(bc_camera, 200);
            }
        }
    }
}
```

添加 ugx 数据代码

双击 ModelSample 窗体中的 AddUGX_MenuItem (添加 UGX 数据)按钮，进入代码编辑器。或者直接在属性界面，手动添加 AddUGX_MenuItem 的 Click 事件输入事件名“AddUGX_MenuItem_Click ”。在“AddUGX_MenuItem_Click ”事件中插入如下代码：

```
//******添加一份 UGX 数据******
private void AddBIM_MenuItem_Click(object sender, EventArgs e)
{
    //判断
    if (bc_sceneControl != null)
    {
        using (OpenFileDialog ofd = new OpenFileDialog())
        {
            ofd.Filter = "(*.bcx)|*.bcx|All files(*.*)|*.*";
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                //读取文件路径
```

```
                string strFilePath = ofd.FileName;
                //添加 UGX 模型数据
                bc_modelLayer = bc_sceneControl.Scene3D.ModelLayerManager.Add(strFilePath,
                LayerType.LT_UGX);
                //设置地球场景加载位置：经度，纬度，高程
                bc_modelLayer.Center = new Point3D(119.9757004, 30.09651184, 120);
                //******设置相机信息并飞行到指定相机位置******
                //经度
                bc_camera.X = 119.9757004;
                //纬度
                bc_camera.Y = 30.09651184;
                //高层值
                bc_camera.Z = 40;
                //相机的方位角(以度为单位)
                bc_camera.Heading = 60;
                //相机的俯仰角(以度为单位)
                bc_camera.Tilt = -90;
                //相机与观察点的距离值
                bc_camera.Distance = 500;
                //飞行到指定相机位置
                bc_sceneControl.Scene3D.FlyTo(bc_camera, 200);
            }
        }
    }
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“创建场景”按钮，打开，一个坐标系为 WGS84 类型的地球场景，点“添加 BIM 数据”或者添加“添加 UGX 数据”即可添加一份数据并同时定位到数据加载位置。

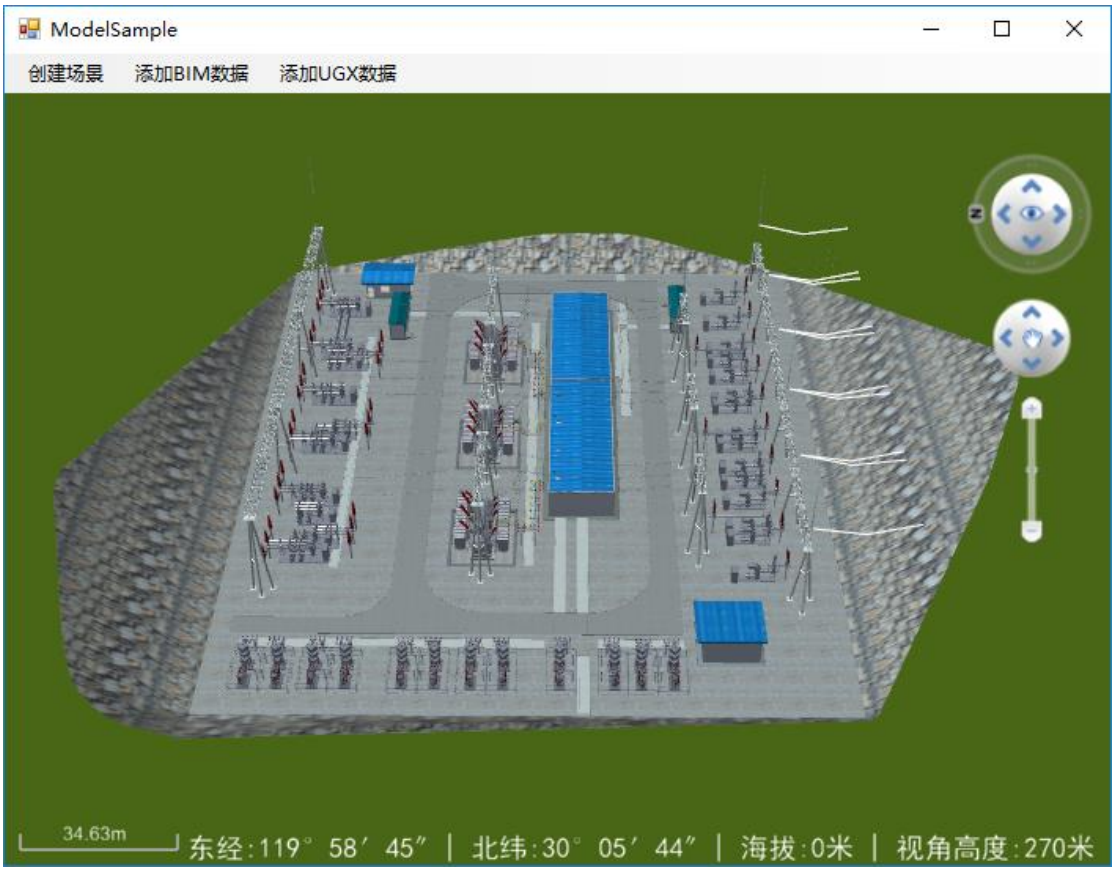


图 加载龙星 BIM 数据

飞行演示

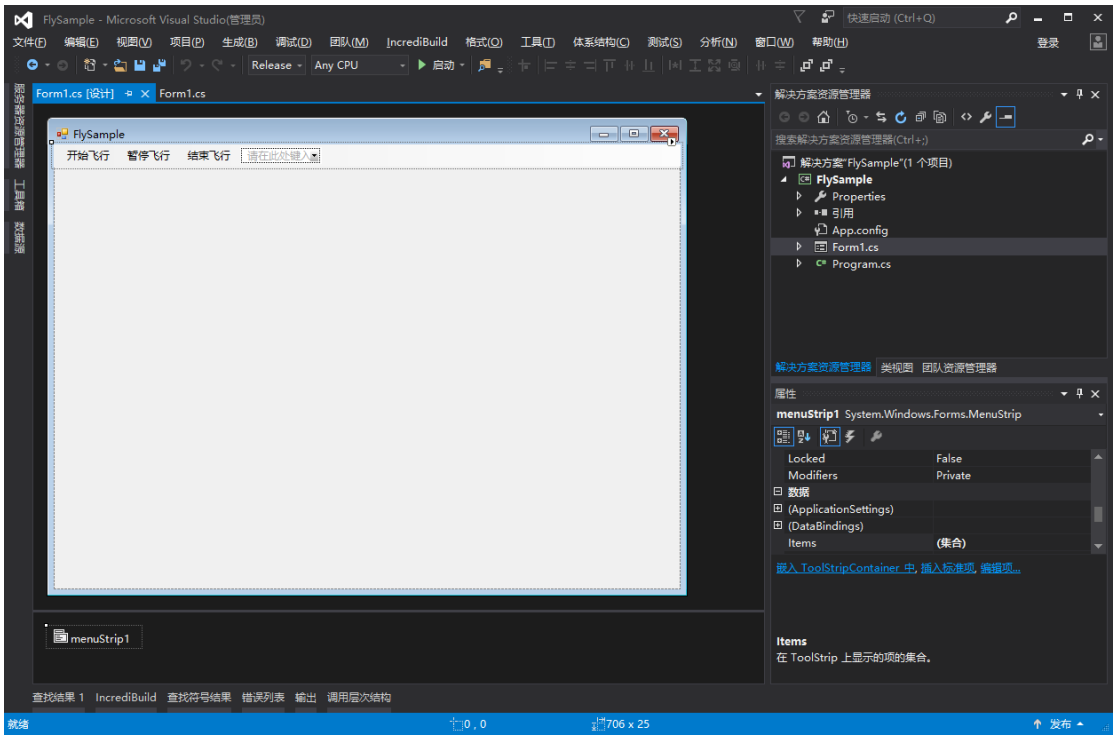
第一步 创建一个工程

创建一个工程，命名为 FlySample，参考产品入门

在窗体中加入 menuStrip 工具条上添加 三个 ToolStripMenuItem 的属性值（其余属性取默认值即可）：

Name	**Text**
Play_MenuItem	开始飞行
Pasue_MenuItem	暂停飞行
Stop_MenuItem	结束飞行

再在窗体中添加 panel 窗体，将 panel1 窗体的属性“Dock”设置为”Fill”后如下图：



第二步 编写代码

在正式编写示范程序之前，需要添加对 BcEngineX SDK 的 EngineX.SceneControl 和 EngineX.Data 程序集的引用，进入代码编辑器，在最顶端添加如下代码：

```
using EngineX.Render;
using EngineX.Data;
```

1.创建场景代码

参考 场景属性控制-代码编写-2.创建地球场景

2.添加开始飞行代码

双击 FlySample 窗体中的 Play_MenuItem (开始飞行)按钮，进入代码编辑器。或者直接在属性界面，手动添加 Play_MenuItem 的 Click 事件输入事件名“Play_MenuItem_Click ”。在“Play_MenuItem_Click ”事件中插入如下代码：

```
//******开始飞行******
private void Play_MenuItem_Click(object sender, EventArgs e)
{
    //建立一个飞行路线，这里添加了 4 个路径点
    List<Point3D> bc_pointList = new List<Point3D>();
    Point3D bc_point1 = new Point3D(-40.7904318392784, -20.0385904483137, 6);
    Point3D bc_point2 = new Point3D(37.5362240246837, -20.3676940444184, 6);
    Point3D bc_point3 = new Point3D(36.5489132366055, -55.2526752277457, 6);
    Point3D bc_point4 = new Point3D(-41.2840872331995, -56.0754342175356, 6);
    bc_pointList.Add(bc_point1);
    bc_pointList.Add(bc_point2);
```

```
            bc_pointList.Add(bc_point3);
            bc_pointList.Add(bc_point4);

            //绘制出飞行路线
        bc_sceneControl.Scene3D.LabelLayerManager.GetLayer().AppendPlineLabel(bc_pointList,
                3, Color.Red, true);
            //设置沿线飞行路径点、飞行总时间
            bc_sceneControl.Scene3D.FlyManager.SetPointPathAndTotalTime(bc_pointList, 15);
            //飞行视角模式(0:自由视角,1:锁定视角)。
            bc_sceneControl.Scene3D.FlyManager.ViewMode = FlyViewMode.FVM_LOCKED;
            //飞行结束后动作(0:飞行前的位置,1:飞行起点,2:飞行结束当前位置)。
            bc_sceneControl.Scene3D.FlyManager.StopAction = FlyStopAction.FSA_START;
            //是否锁定高度
            bc_sceneControl.Scene3D.FlyManager.HeightLocked = true;
            //是否显示模型
            bc_sceneControl.Scene3D.FlyManager.ModelShow = true;
            //飞行的速率(速度倍数,1.0 为正常速度)
            bc_sceneControl.Scene3D.FlyManager.SpeedMultiple = 1.0;
            //开始飞行
            bc_sceneControl.Scene3D.FlyManager.Play();
    }
```

3.添加暂停飞行代码

双击 FlySample 窗体中的 Pasue_MenuItem (暂停飞行)按钮，进入代码编辑器。或者直接在属性界面，手动添加 Pasue_MenuItem 的 Click 事件输入事件名“Pasue_MenuItem_Click ”。在“Pasue_MenuItem_Click ”事件中插入如下代码：

```
    //******暂停飞行******
    private void Pasue_MenuItem_Click(object sender, EventArgs e)
    {
        bc_sceneControl.Scene3D.FlyManager.Pause();
    }
```

4.添加停止飞行代码

双击 FlySample 窗体中的 Stop_MenuItem (结束飞行)按钮，进入代码编辑器。或者直接在属性界面，手动添加 Stop_MenuItem 的 Click 事件输入事件名“Stop_MenuItem _Click ”。在“Stop_MenuItem _Click ”事件中插入如下代码：

```
    //******结束飞行******
    private void Stop_MenuItem_Click(object sender, EventArgs e)
    {
        bc_sceneControl.Scene3D.FlyManager.Stop();
    }
```

运行调试

运行代码后弹出如下界面，单击工具条上的“开始飞行”按钮，开始进行飞行循环，飞行过程中单击“暂停飞行”即可暂停飞行再次点击“开始飞行”可继续飞行循环，单击“结束飞行”结束当前飞行。

标注

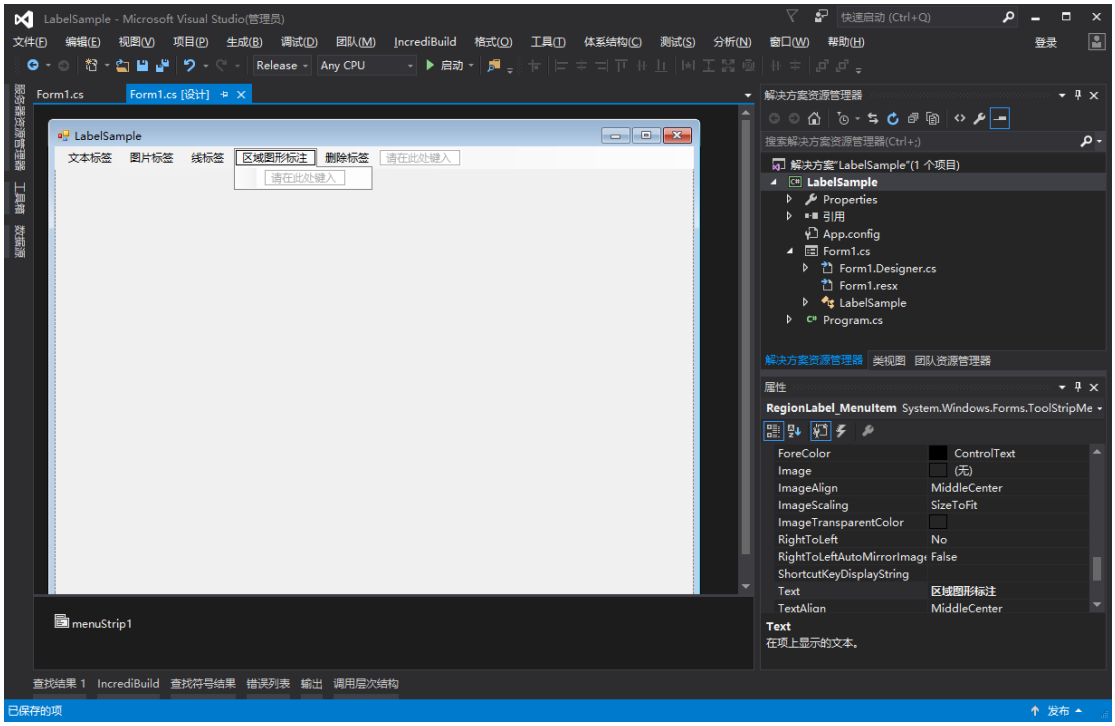
第一步 创建一个工程

创建一个工程，命名为 LabelSample，参考产品入门

在窗体中加入 menuStrip 工具条上添加 五个 ToolStripMenuItem 的属性值（其余属性取默认值即可）：

Name	Text
TextLabel_MenuItem	文本标签
PictureLabel_MenuItem	图片标签
PlineLabel_MenuItem	线标签
RegionLabel_MenuItem	区域图形标签
EraseLabel_MenuItem	删除标签

再在窗体中添加 panel 窗体，将 panel1 窗体的属性“Dock”设置为”Fill”后如下图：



第二步 编写代码

在正式编写示范程序之前，需要添加对 BcEngineX SDK 的 EngineX.SceneControl 和 EngineX.Data 程序集的引用，进入代码编辑器，在最顶端添加如下代码：

```
using EngineX.Render;
using EngineX.Data;
```

1.创建场景代码

参考 场景属性控制-代码编写-2.创建地球场景

2.添加标注代码

双击 LabelSample 窗体中的 TextLabel_MenuItem (文本标签)按钮，进入代码编辑器。或者直接在属性界面，手动添加 TextLabel_MenuItem 的 Click 事件输入事件名“TextLabel_MenuItem_Click ”。在“TextLabel_MenuItem_Click ”事件中插入如下代码：

```
//标签 guid 集合
String bc_labelGuidGather = null;
//******文本批注******
private void TextLabel_MenuItem_Click(object sender, EventArgs e)
{
  Point3D bc_point1 = new Point3D(119.9742898982051, 30.09684321836875, 80.790303409
  09958);
  Point3D bc_point2 = new Point3D(119.9762898982051, 30.09684321836875, 80.790303409
  09958);
```

```
    Point3D bc_point3 = new Point3D(119.9782898982051, 30.09884321836875, 80.790303409
    09958);
    //添加三个文本
    string guid, guid2, guid3;
    string bc_context = "博超文本标签 1";
    string bc_context2 = "博超文本标签 2";
    string bc_context3 = "博超文本标签 3";
    double bc_textSize = 15;
    //设置文本颜色
    Color bc_color = Color.Green;
    //是否遮挡
    Boolean bc_bAlwaysShow = true;
    //是否固定大小不随滚轮缩放
    Boolean bc_bFixedSize = true;
    //添加文本并降返回的 guid 存储到 bc_labelGuidGather
    guid = bc_sceneControl.Scene3D.LabelLayerManager.GetLayer().AppendTextLabel(bc_context,
    bc_point1, bc_textSize, bc_color, bc_bAlwaysShow, bc_bFixedSize);
    bc_labelGuidGather = bc_labelGuidGather + ";" + guid;

    guid2=bc_sceneControl.Scene3D.LabelLayerManager.GetLayer().AppendTextLabel(bc_context
    2, bc_point2, bc_textSize, bc_color, bc_bAlwaysShow, bc_bFixedSize);
     bc_labelGuidGather = bc_labelGuidGather + ";" + guid2;

    guid3=bc_sceneControl.Scene3D.LabelLayerManager.GetLayer().AppendTextLabel(bc_context
    3, bc_point3, bc_textSize, bc_color, bc_bAlwaysShow, bc_bFixedSize);
    bc_labelGuidGather = bc_labelGuidGather + ";" + guid3;
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“文本标签”按钮，会添加 3 个文本标签到代码所指定的经纬高位置，如下图所示。

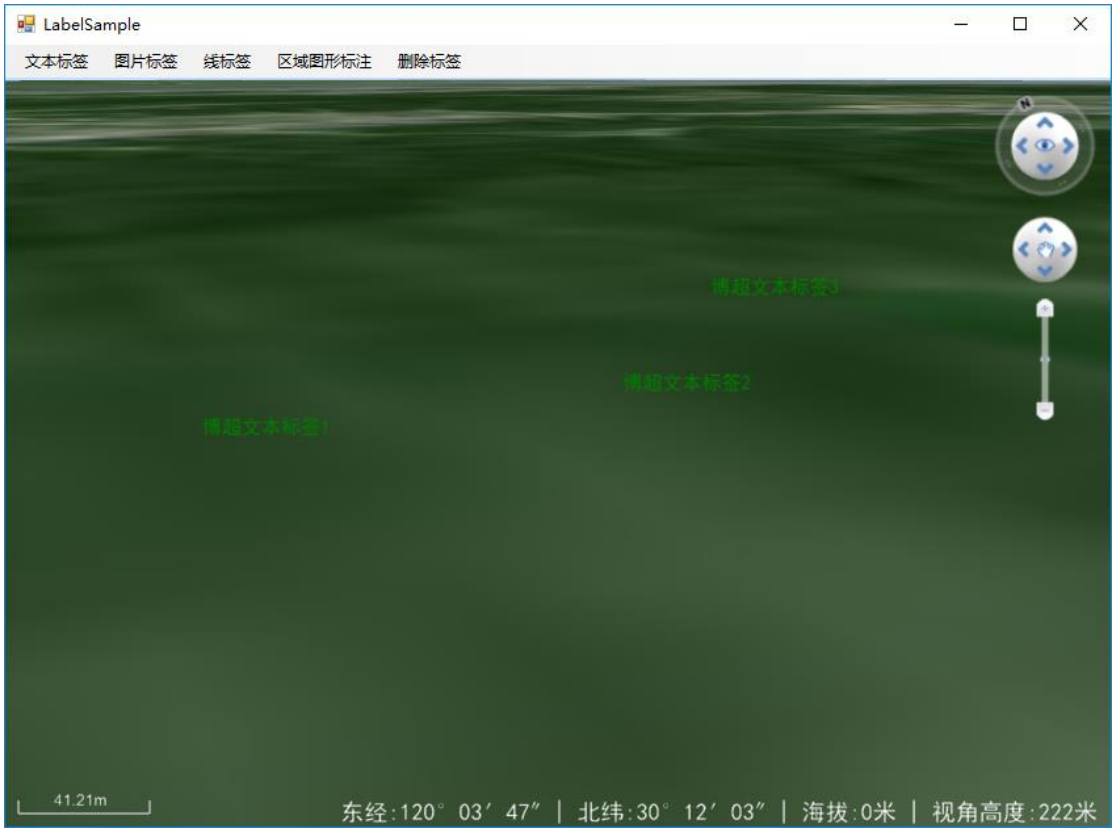


图 添加文本标签

3.添图片标注代码

双击 LabelSample 窗体中的 PictureLabel_MenuItem (图片标签)按钮，进入代码编辑器。或者直接在属性界面，手动添加 TextLabel_MenuItem 的 Click 事件输入事件名“PictureLabel_MenuItem_Click ”。在“PictureLabel_MenuItem_Click ”事件中插入如下代码:

```
//******图片批注******
private void PictureLabel_MenuItem_Click(object sender, EventArgs e)
{
  Point3D bc_point1 = new Point3D(119.9742898982051, 30.09684321836875, 80.790303409
  09958);
  Point3D bc_point2 = new Point3D(119.9762898982051, 30.09684321836875, 80.790303409
  09958);
  Point3D bc_point3 = new Point3D(119.9782898982051, 30.09884321836875, 80.790303409
  09958);

  string guid, guid2, guid3;
  string bc_picturePath = "F:/engineXDATA/placemark32.png";
  //图片缩放系数
  Double bc_scale = 1.0;
  //是否遮挡
  Boolean bc_bAlwaysShow = true;
  //是否固定大小不随滚轮缩放
```

```
    Boolean bc_bFixedSize = true;
    //添加三个图片标签,并将返回的 guid 存储到 bc_labelGuidGather
    guid=bc_sceneControl.Scene3D.LabelLayerManager.GetLayer().AppendPictureLabel(bc_picture
    Path, bc_point1, bc_scale, bc_bFixedSize, bc_bFixedSize);
     bc_labelGuidGather = bc_labelGuidGather + ";" + guid;

    guid2=bc_sceneControl.Scene3D.LabelLayerManager.GetLayer().AppendPictureLabel(bc_pictur
    ePath, bc_point2, bc_scale, bc_bFixedSize, bc_bFixedSize);
     bc_labelGuidGather = bc_labelGuidGather + ";" + guid2;

    guid3=bc_sceneControl.Scene3D.LabelLayerManager.GetLayer().AppendPictureLabel(bc_pictur
    ePath, bc_point3, bc_scale, bc_bFixedSize, bc_bFixedSize);
     bc_labelGuidGather = bc_labelGuidGather + ";" + guid3;
}
```

运行调试

运行代码后弹出如下界面，单击工具条上的“图片标签”按钮，会添加 3 个图片标签到代码所指定的经纬高位置，如下图所示。

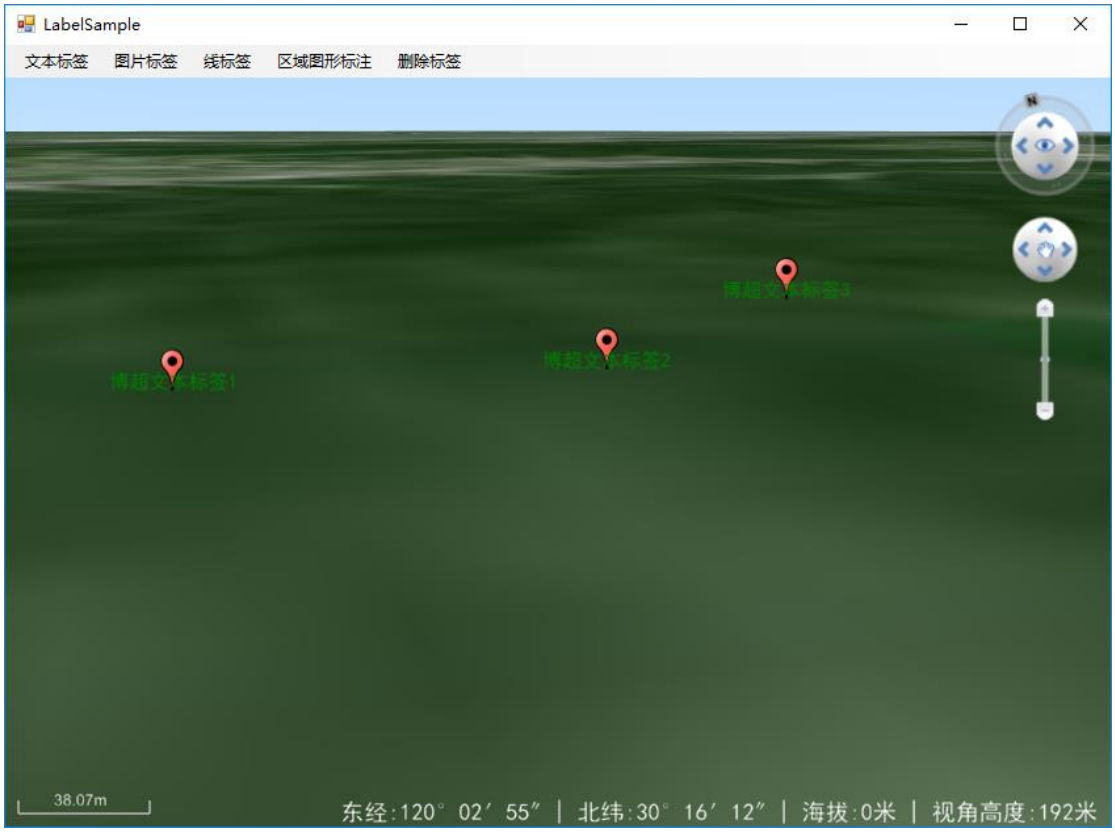


图 添加图片标签

添加线标签、区域图形标注

图 添加线标签和区域图形标签

三维测量

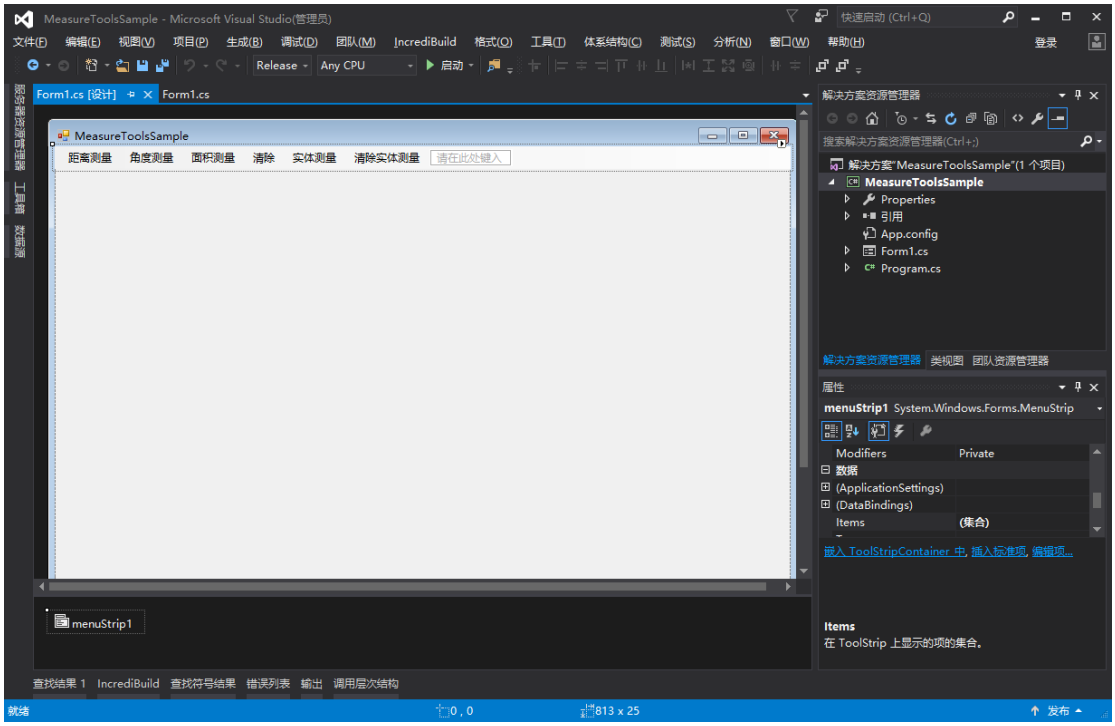
第一步 创建一个工程

创建一个工程，命名为 LabelSample，参考产品入门

在窗体中加入 menuStrip 工具条上添加 五个 ToolStripMenuItem 的属性值（其余属性取默认值即可）：

Name	Text
MeasureDistance_MenuItem	距离测量
MeasureAngle_MenuItem	角度测量
MeasureArea_MenuItem	面积测量
eraseMeasure_MenuItem	清除
MeasureEntity_MenuItem	实体测量
clearMeasureEntity_MenuItem	清除实体测量

再在窗体中添加 panel 窗体，将 panel1 窗体的属性“Dock”设置为”Fill”后如下图：



第二步 编写代码

在正式编写示范程序之前，需要添加对 BcEngineX SDK 的 EngineX.SceneControl 和 EngineX.Data 程序集的引用，进入代码编辑器，在最顶端添加如下代码：

```
using EngineX.Render;
using EngineX.Data;
```

1.创建场景代码

参考 场景属性控制-代码编写-2.创建地球场景

2.添加距离测量代码

双击 LabelSample 窗体中的 MeasureDistance_MenuItem (距离测量)按钮，进入代码编辑器。或者直接在属性界面，手动添加 MeasureDistance_MenuItem 的 Click 事件输入事件名“MeasureDistance_MenuItem_Click ”。在“MeasureDistance_MenuItem_Click ”事件中插入如下代码：

```
//创建实体测量工具对象
MeasureEntityTool bc_measureEntity = new MeasureEntityTool();
//存储实体测量
List<string> bc_list = new List<string>();
//******距离测量******
private void MeasureDistance_MenuItem_Click(object sender, EventArgs e)
{
  //判断是否已经新建过距离测量工具
  if (bc_sceneControl.Scene3D.HasSceneTool("BcMeasureDistance"))
  {
```

```
        MeasureDistanceTool bc_measureDistance= bc_sceneControl.Scene3D.GetSceneTool("Bc
        MeasureDistance") as MeasureDistanceTool;
        bc_measureDistance.Start();
    }
    else
    {
        MeasureDistanceTool bc_measureDistance = new MeasureDistanceTool();
        bc_sceneControl.Scene3D.AddSceneTool(bc_measureDistance);
        bc_measureDistance.Start();
       }
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“距离测量”按钮，开始进行取点测量，左键取点，右键结束，如下图所示。

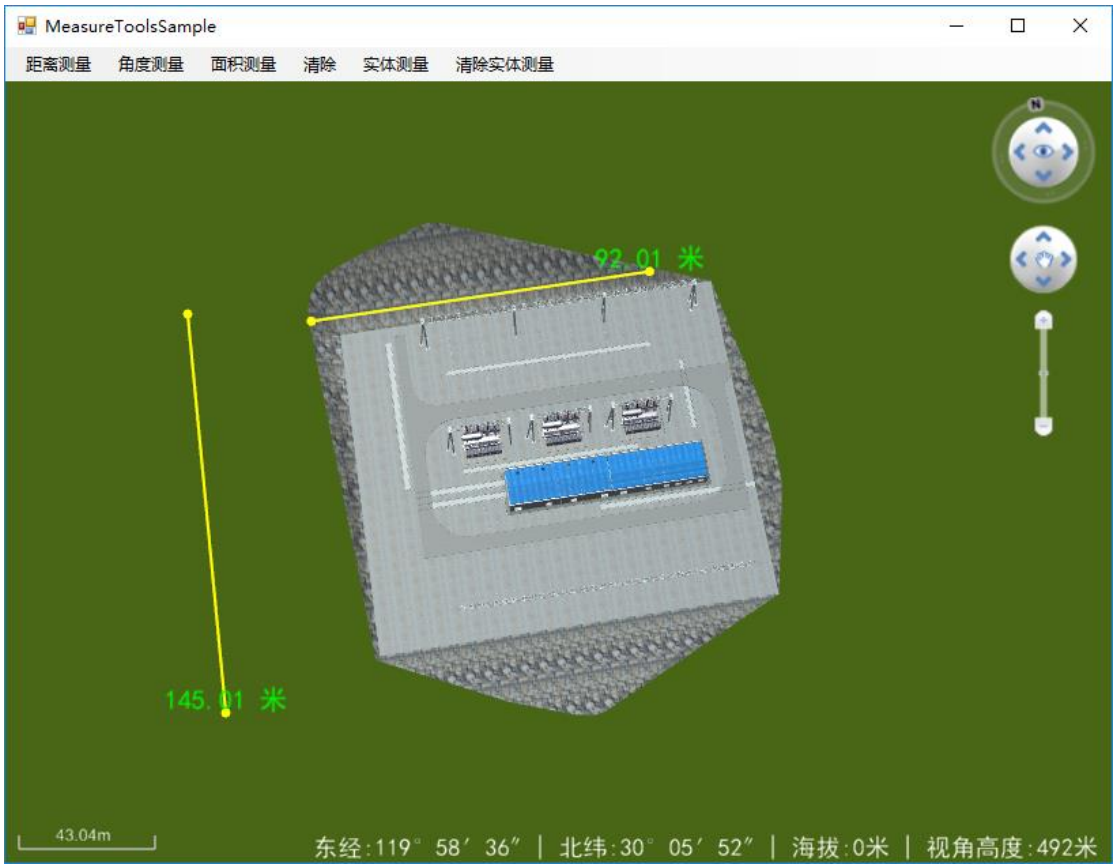


图 距离测量

3.添加角度测量和面积测量代码

双击 LabelSample 窗体中的 MeasureAngle_MenuItem（角度测量）按钮和 MeasureArea_MenuItem（面积测量)按钮，进入代码编辑器。或者直接在属性界面，手动添加 Click 事件。在“MeasureAngle_MenuItem _Click ”和“MeasureArea_MenuItem_Clicks”事件中插入如下代码：

```
//******角度测量******
private void MeasureAngle_MenuItem_Click(object sender, EventArgs e)
{
  //判断是否已经新建过角度测量工具
  if (bc_sceneControl.Scene3D.HasSceneTool("BcMeasureAngle"))
     {
          MeasureAngleTool bc_measureAngle = bc_sceneControl.Scene3D.GetSceneTool("Bc
          MeasureAngle") as MeasureAngleTool;
          bc_measureAngle.Start();
     }
     else
     {
          MeasureAngleTool bc_measureAngle = new MeasureAngleTool();
          bc_sceneControl.Scene3D.AddSceneTool(bc_measureAngle);
          bc_measureAngle.Start();
     }
}
//******面积测量******
private void MeasureArea_MenuItem_Click(object sender, EventArgs e)
{
     //判断是否已经新建过面积测量工具
     if (bc_sceneControl.Scene3D.HasSceneTool("BcMeasureArea"))
     {
          MeasureAreaTool bc_measureArea = bc_sceneControl.Scene3D.GetSceneTool("BcMe
          asureArea") as MeasureAreaTool;
          bc_measureArea.Start();
     }
     else
     {
          MeasureAreaTool bc_measureArea = new MeasureAreaTool();
          bc_sceneControl.Scene3D.AddSceneTool(bc_measureArea);
          bc_measureArea.Start();
     }
}
```

4.添加清除测量代码

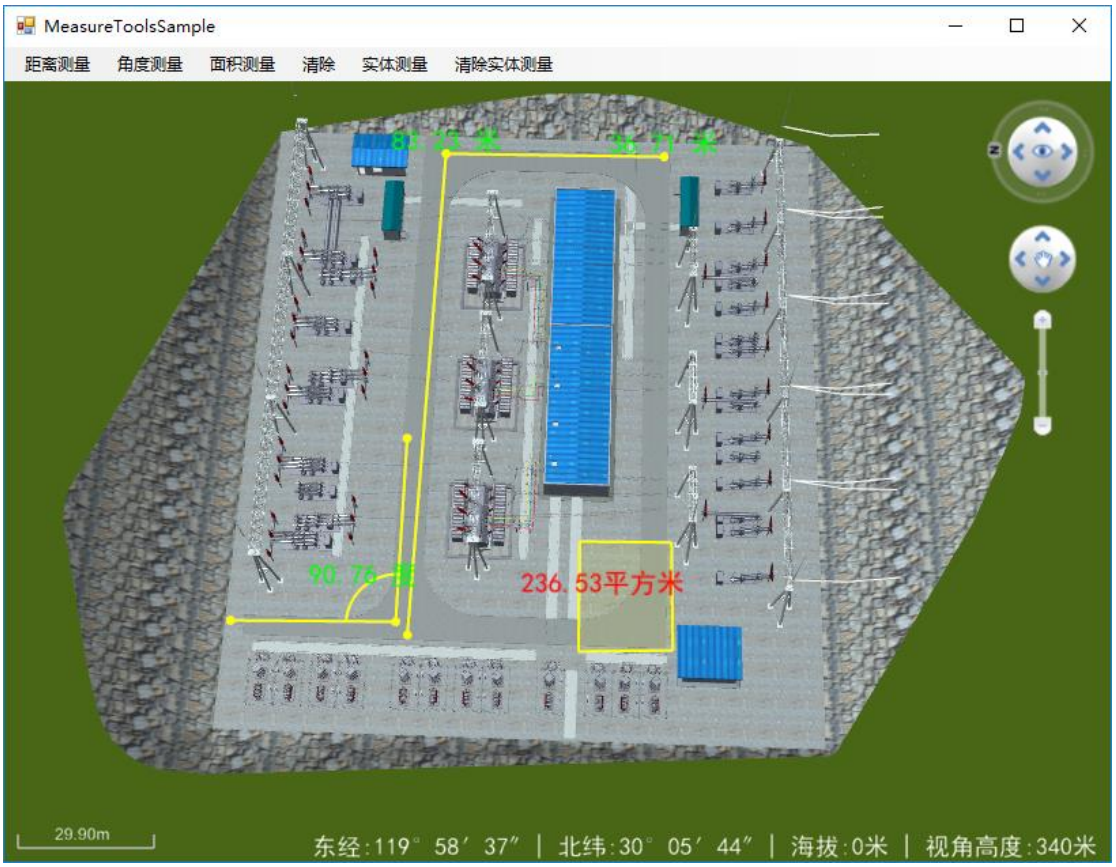
双击 MeasureToolsSample 窗体中的 eraseMeasure_MenuItem (清除)按钮，进入代码编辑器。或者直接在属性界面，手动添加 eraseMeasure_MenuItem 的 Click 事件输入事件名“eraseMeasure_MenuItem_Click ”。在“eraseMeasure_MenuItem_Click ”事件中插入如下代码：

```
//******清除测量******
private void eraseMeasure_MenuItem_Click(object sender, EventArgs e)
{
     //如果有距离测量标记，清除
```

```
        if (bc_sceneControl.Scene3D.HasSceneTool("BcMeasureDistance"))
        {
            MeasureDistanceTool bc_measureDistance = bc_sceneControl.Scene3D.GetSceneTool("BcMeasureDistance") as MeasureDistanceTool;
            bc_measureDistance.Clear();
        }
        //如果有面积测量标记，清除
        if (bc_sceneControl.Scene3D.HasSceneTool("BcMeasureArea"))
        {
            MeasureAreaTool bc_measureArea = bc_sceneControl.Scene3D.GetSceneTool("BcMeasureArea") as MeasureAreaTool;
            bc_measureArea.Clear();
        }
        //如果有角度测量标记，清除
        if (bc_sceneControl.Scene3D.HasSceneTool("BcMeasureAngle"))
        {
            MeasureAngleTool bc_measureAngle = bc_sceneControl.Scene3D.GetSceneTool("BcMeasureAngle") as MeasureAngleTool;
            bc_measureAngle.Clear();
        }
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“角度测量”和“面积测量”按钮，开始进行取点测量，左键取点，右键结束，如下图所示。



5.添加实体测量代码

双击 MeasureToolsSample 窗体中的 MeasureEntity_MenuItem（实体测量)和 clearMeasureEntity_MenuItem（清除实体测量）按钮，进入代码编辑器。或者直接在属性界面，手动添加实体测量和清除实体测量的 Click 事件，分别输入 Click 事件名“MeasureEntity_MenuItem_Click ”和“clearMeasureEntity_MenuItem_Click ”，在事件中插入如下代码：

```
//创建实体测量工具对象
MeasureEntityTool bc_measureEntity = new MeasureEntityTool();
//存储实体测量
List<string> bc_list = new List<string>();
//******实体测量******
private void MeasureEntity_MenuItem_Click(object sender, EventArgs e)
{
    bc_sceneControl.Scene3D.AddSceneTool(bc_measureEntity);
    bc_sceneControl.Scene3D.EntitySelectedHandler += MeasureEntitySelected;

    //开启点选拾取模式
    bc_sceneControl.Scene3D.ScenePickType = PickType.PT_POINT;
    bc_sceneControl.Scene3D.ScenePickMode = PickMode.PM_SINGLE;
}
//******实体选中事件******
```

```
private void MeasureEntitySelected(object sender, EntitySelectedEventArgs bc_entitySelectedEventArgs)
{
    List<EntitySelectedData> m_entityData = bc_entitySelectedEventArgs.Get();
    bc_entitySelectedEventArgs.Add(m_entityData[0]);
    bc_measureEntity.MeasureEntity(m_entityData[0].LayerGuid, m_entityData[0].GUID);
}
//******清除实体测量结果******
private void clearMeasureEntity_MenuItem_Click(object sender, EventArgs e)
{
    MeasureEntityTool bc_measureEntityTool = bc_sceneControl.Scene3D.GetSceneTool("BcMeasureEntity") as MeasureEntityTool;
    bc_measureEntityTool.Clear();
    bc_list.Clear();
    return;
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“实体测量”按钮，拾取模型实体，点击“清除实体测量”可以清除所有测量结果如下图所示。

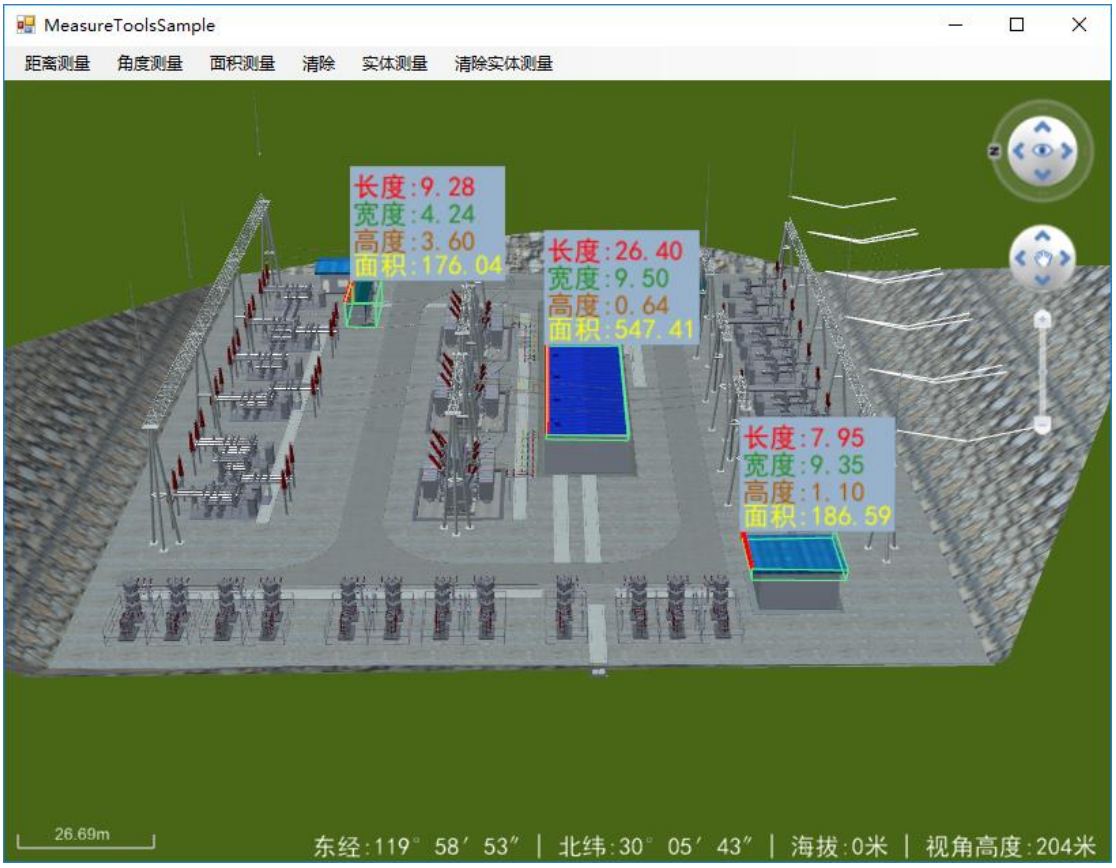


图 实体测量

三维空间分析

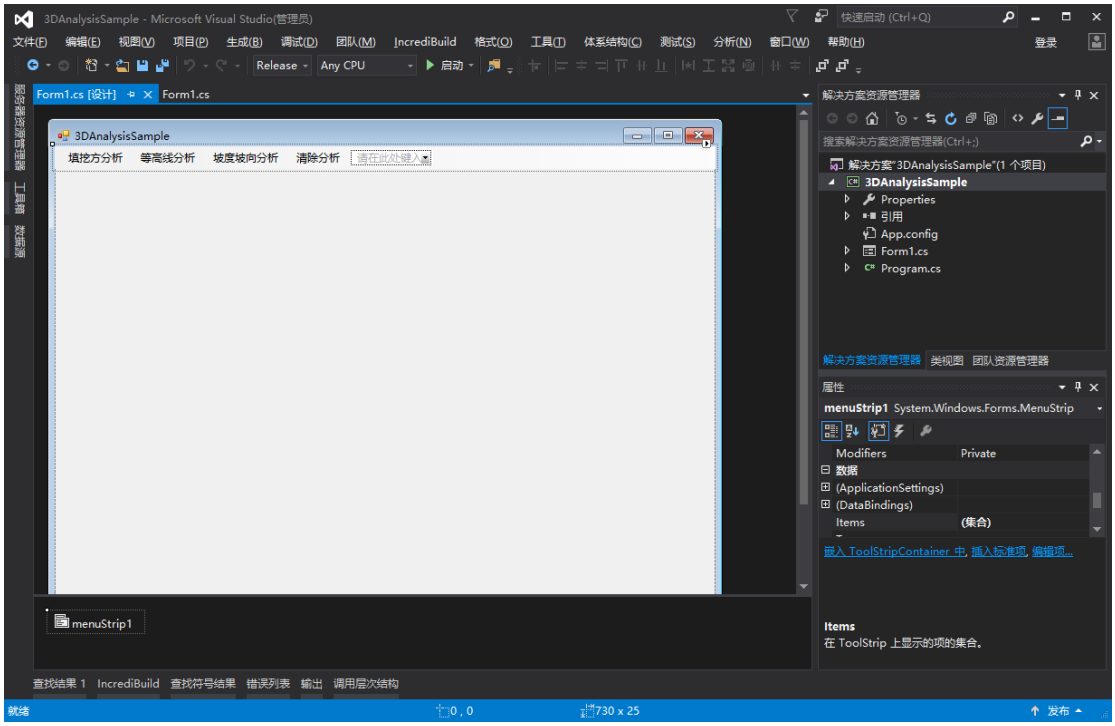
第一步 创建一个工程

创建一个工程，命名为 3DAnalysisSample，参考产品入门

目前三维空间分析支持淹没分析、填挖方分析、等高线分析、天际线分析、坡度坡向分析、通透分析、视域分析、模型截面分析、地形剖面分析，这里仅举例三个分析进行说明。
在窗体中加入 menuStrip 工具条上添加 五个 ToolStripMenuItem 的属性值（其余属性取默认值即可）：

Name	Text
Analysis3DCutFill_MenuItem	填挖方分析
ContourAnalysis_MenuItem	等高线分析
SlopeAspectAnalysis_MenuItem	坡度坡向分析
DisableAllSceneTool_MenuItem	清除分析

再在窗体中添加 panel 窗体，将 panel1 窗体的属性“Dock”设置为”Fill”后如下图：



第二步 编写代码

在正式编写示范程序之前，需要添加对 BcEngineX SDK 的 EngineX.SceneControl 和 EngineX.Data 程序集的引用，进入代码编辑器，在最顶端添加如下代码：

```
using EngineX.Render;
using EngineX.Data;
```

1.创建场景代码

参考 场景属性控制-代码编写-2.创建地球场景

创建地球场景的同时，还需要加载一份地形影像数据,代码如下：

```
//添加地形
bc_sceneControl.Scene3D.TerrainLayerManager.AddTMSTerrainLayer(@"192.168.2.90:27017", "ZJFY_32_0913_9_DEM", 0, 23, true);
//添加影像
bc_sceneControl.Scene3D.ImageLayerManager.AddTMSImageLayer(@"192.168.2.90:27017", "
    ZJFY_TMS_18", 0, 23, true);
```

2.添加填挖方分析代码

双击 MeasureToolsSample 窗体中的 Analysis3DCutFill_MenuItem (填挖方分析)按钮，进入代码编辑器。或者直接在属性界面，手动添加填挖方分析的 Click 事件，输入 Click 事件名“Analysis3DCutFill_MenuItem_Click，在事件中插入如下代码：

```
//******填挖方分析******
private void Analysis3DCutFill_MenuItem_Click(object sender, EventArgs e)
{
    //判断是否创建过填挖方分析工具
    if (bc_sceneControl.Scene3D.HasSceneTool("Analysis3DCutFillTool"))
    {
        //使用已创建的填挖方分析工具
        Analysis3DCutFillTool bc_analysis3DCutFill = bc_sceneControl.Scene3D.GetSceneTool
        ("Analysis3DCutFillTool") as Analysis3DCutFillTool;
        //激活工具
        bc_analysis3DCutFill.SetActive(true);
        //填挖方高度
        bc_analysis3DCutFill.SetCutfillHeight(120);
    }
    else
    {
        Analysis3DCutFillTool bc_analysis3DCutFill = new Analysis3DCutFillTool();
        bc_sceneControl.Scene3D.AddSceneTool(bc_analysis3DCutFill);
        bc_analysis3DCutFill.SetActive(true);
        bc_analysis3DCutFill.SetCutfillHeight(120);
    }
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“填挖方分析”按钮，鼠标开始拾取分析范围，右键结束取点，分析结果如下图所示。

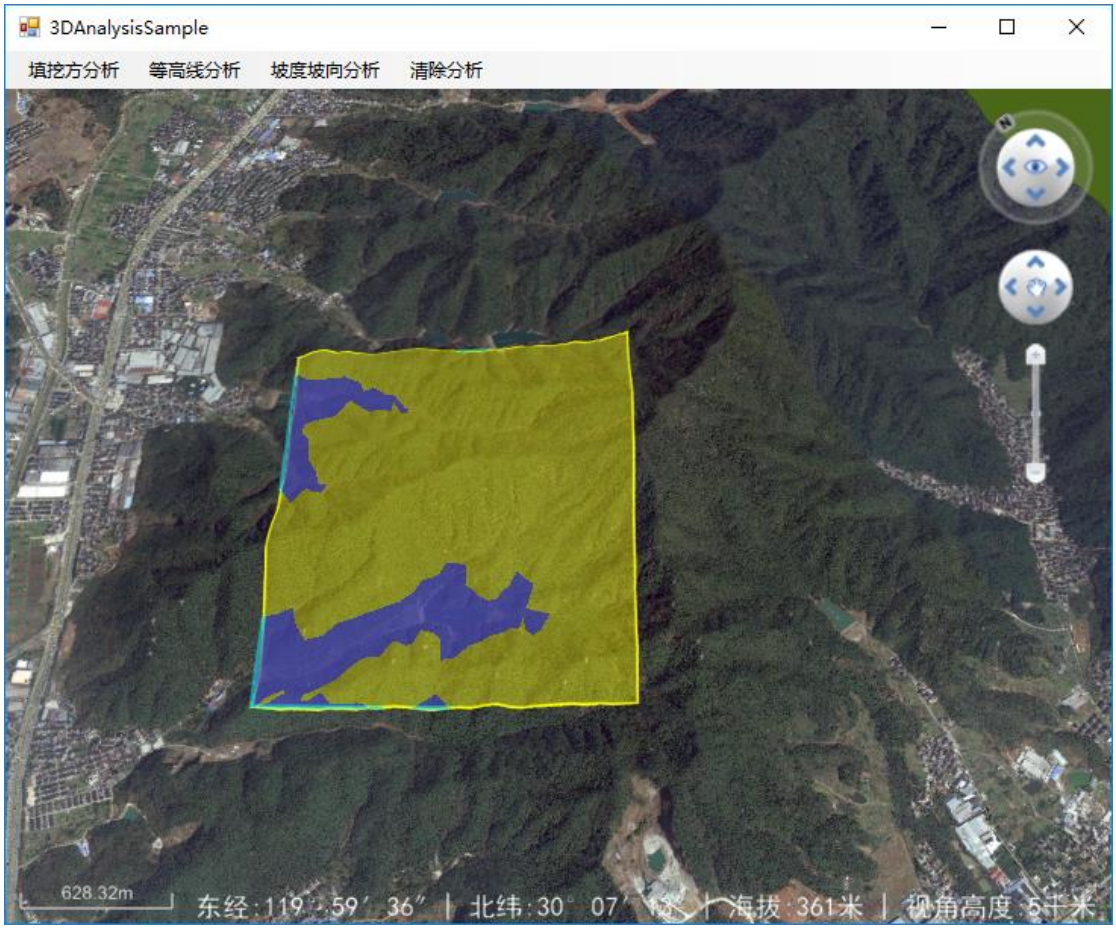


图 填挖方分析

3.添加坡度坡向分析代码

双击 MeasureToolsSample 窗体中的 SlopeAspectAnalysis_MenuItem（坡度坡向分析)按钮，进入代码编辑器。或者直接在属性界面，手动添加坡度坡向分析的 Click 事件，输入 Click 事件名“SlopeAspectAnalysis_MenuItem_Click，在事件中插入如下代码：

```
//******坡度坡向分析******
private void SlopeAspectAnalysis_MenuItem_Click(object sender, EventArgs e)
{
    if (bc_sceneControl.Scene3D.HasSceneTool("BcSlopeAspectAnalysisTool"))
    {
        //使用已创建的坡度坡向分析工具
        SlopeAspectAnalysisTool bc_slopeAspectAnalysisTool = bc_sceneControl.Scene3D.Ge
        tSceneTool("BcSlopeAspectAnalysisTool") as SlopeAspectAnalysisTool;
        //SlopeAspectType坡度坡向分析类型：坡度分析，坡向分析，坡度坡向分析
        bc_slopeAspectAnalysisTool.SetAnalysisType(SlopeAspectType.SAT_SLOPEANDASPEC
        T);
        //坡度坡向箭头是否移动
        bc_slopeAspectAnalysisTool.SetArrowMove(true);
        //激活工具
```

```
            bc_slopeAspectAnalysisTool.SetActive(true);
        }
        else
        {
            SlopeAspectAnalysisTool bc_slopeAspectAnalysisTool = new SlopeAspectAnalysisTo
            ol();
            //将分析工具加载进Scene3D
            bc_sceneControl.Scene3D.AddSceneTool(bc_slopeAspectAnalysisTool);
            //SlopeAspectType坡度坡向分析类型：坡度分析，坡向分析，坡度坡向分析
            bc_slopeAspectAnalysisTool.SetAnalysisType(SlopeAspectType.SAT_SLOPEANDASPEC
            T);
            //坡度坡向箭头是否移动
            bc_slopeAspectAnalysisTool.SetArrowMove(true);
            //激活工具
            bc_slopeAspectAnalysisTool.SetActive(true);
        }
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“坡度坡向分析”按钮，鼠标开始拾取分析范围，右键结束取点，分析结果如下图所示。

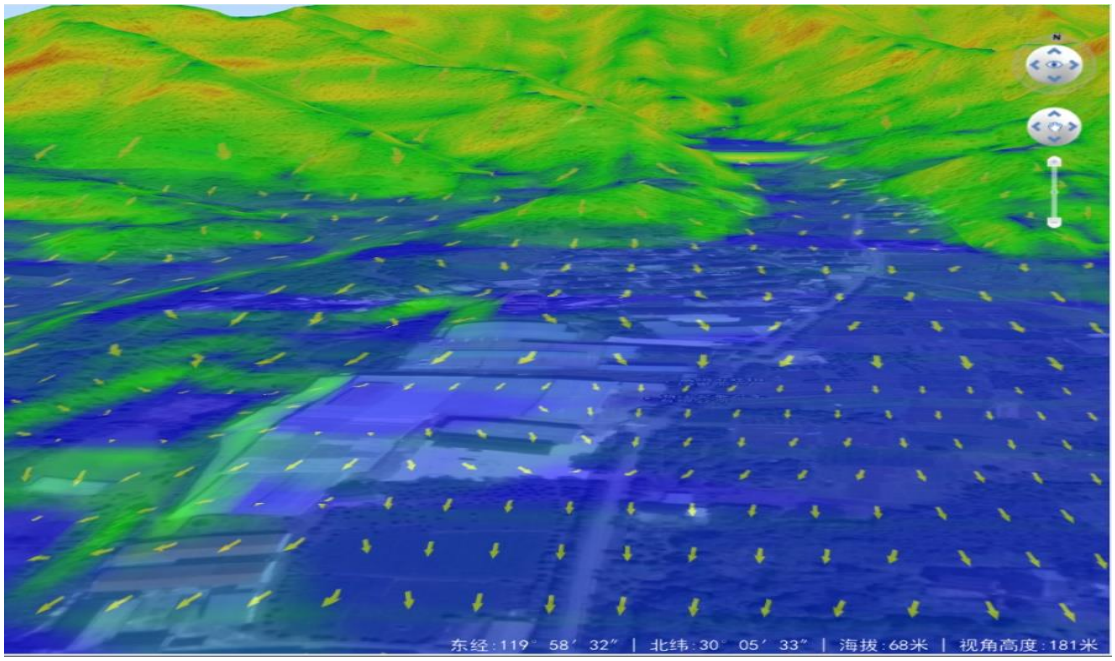


图 坡度坡向分析

4.添加等高线分析代码

双击 MeasureToolsSample 窗体中的 ContourAnalysis_MenuItem (等高线分析)按钮，进入代码编辑器。或者直接在属性界面，手动添加等高线方分析的 Click 事件，输入 Click 事件名“ContourAnalysis_MenuItem_Click，在事件中插入如下代码：

```
//******等高线分析******
private void ContourAnalysis_MenuItem_Click(object sender, EventArgs e)
{
    //判断是否创建过等高线分析工具
    if (bc_sceneControl.Scene3D.HasSceneTool("BcContourAnalysisTool"))
    {
        //使用已创建的等高线分析工具
        ContourAnalysisTool bc_contourAnalysisTool = bc_sceneControl.Scene3D.GetSceneT
        ool("BcContourAnalysisTool") as ContourAnalysisTool;
        //ContourType：等高线、面分析类型
        bc_contourAnalysisTool.SetAnalysisType(ContourType.CT_SURFACEWITHLINE);
        //设置分析高度间距
        bc_contourAnalysisTool.SetContourInterval(10);
        //设置线面分析画线颜色
        bc_contourAnalysisTool.SetContourLineColor(Color.Black);
        //激活工具
        bc_contourAnalysisTool.SetActive(true);
    }
    else
    {
        //创建一个等高线分析工具
        ContourAnalysisTool bc_contourAnalysisTool = new ContourAnalysisTool();
        bc_sceneControl.Scene3D.AddSceneTool(bc_contourAnalysisTool);
        bc_contourAnalysisTool.SetAnalysisType(ContourType.CT_SURFACEWITHLINE);
        bc_contourAnalysisTool.SetContourInterval(10);
        bc_contourAnalysisTool.SetContourLineColor(Color.Black);
        bc_contourAnalysisTool.SetActive(true);
    }
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“等高线分析”按钮，鼠标开始拾取分析范围，右键结束取点，分析结果如下图所示。

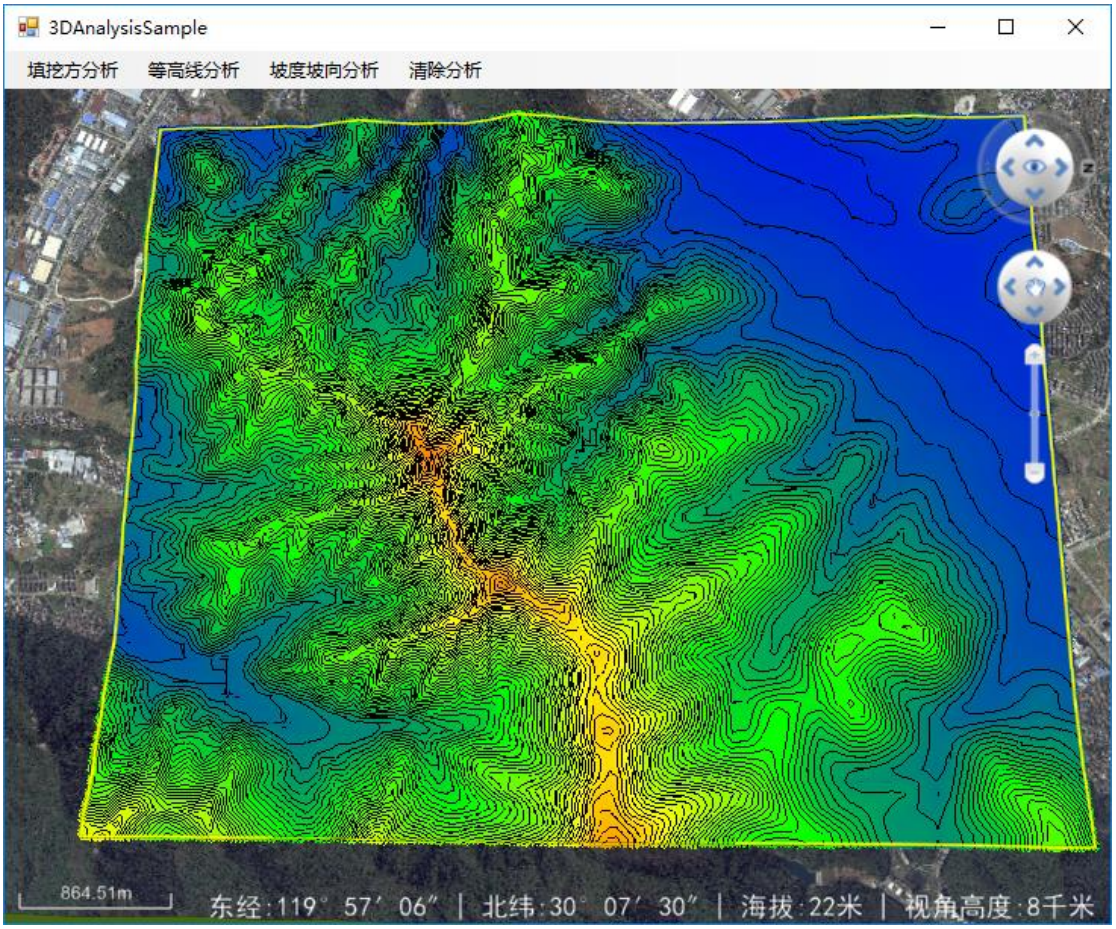


图 等高线分析

5.添加清除分析代码

双击 MeasureToolsSample 窗体中的 DisableAllSceneTool_MenuItem（清除分析)按钮，进入代码编辑器。或者直接在属性界面，手动添加清除分析的 Click 事件，输入 Click 事件名“DisableAllSceneTool_MenuItem_Click，在事件中插入如下代码：

```
//******清除所有分析******
private void DisableAllSceneTool_MenuItem_Click(object sender, EventArgs e)
{
    bc_sceneControl.Scene3D.DisableAllSceneTool();
}
```

三维专题图

第一步 创建一个工程

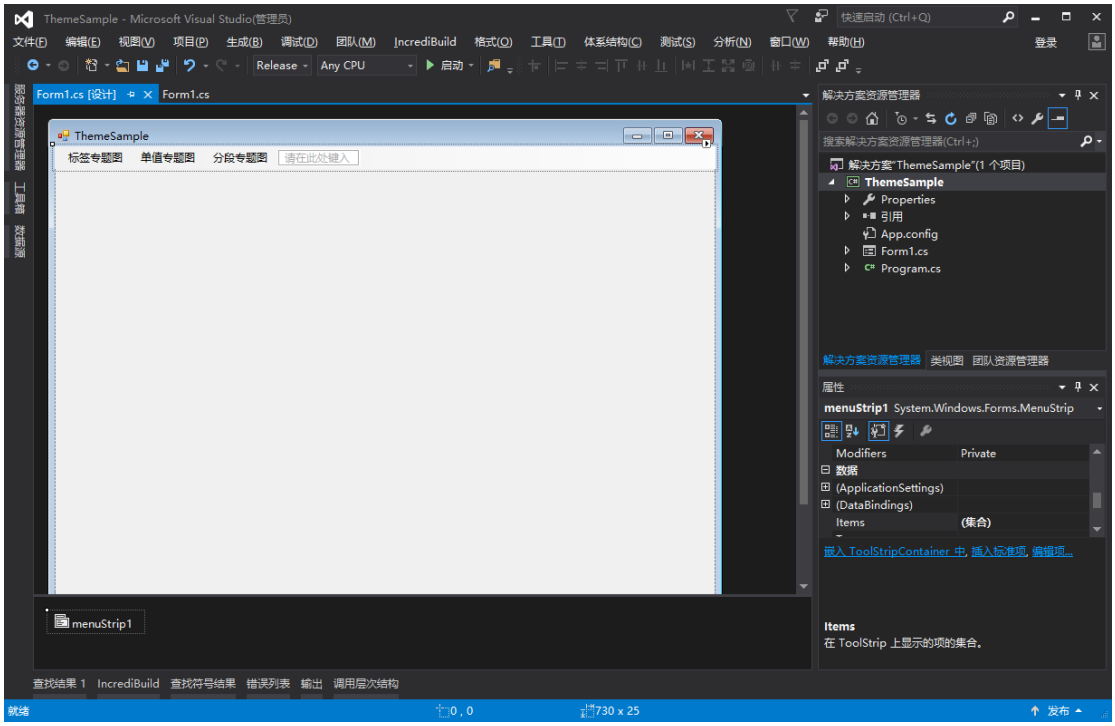
创建一个工程，命名为 ThemeSample，参考产品入门

在窗体中加入 menuStrip 工具条上添加 五个 ToolStripMenuItem 的属性值（其余属性取默

认值即可)：

Name	Text
labelTheme_MenuItem	标签专题图
uniqueTheme_MenuItem	单值专题图
rangeTheme_MenuItem	分段专题图

再在窗体中添加 panel 窗体，将 panel1 窗体的属性“Dock”设置为”Fill”后如下图：



第二步 编写代码

在正式编写示范程序之前，需要添加对 BcEngineX SDK 的 EngineX.SceneControl 和 EngineX.Data 程序集的引用，进入代码编辑器，在最顶端添加如下代码：

```
using EngineX.Render;
```

1.创建地球场景代码

参考 场景属性控制-代码编写-2.创建地球场景

2. 添加标签专题图代码

双击 ThemeSample 窗体中的 PlabelTheme_MenuItem(标签专题图)按钮，进入代码编辑器。或者直接在属性界面，手动添加标签专题图的 Click 事件，输入 Click 事件名“PlabelTheme_MenuItem_Click，在事件中插入如下代码：

```
//******标签专题图******
private void labelTheme_MenuItem_Click(object sender, EventArgs e)
{
    Theme3DLabel bc_theme3DLabel = new Theme3DLabel();
```

```
        //设置标注字段
        bc_theme3DLabel.LabelExpression = "NAME2004";
        bc_theme3DLabel.LabelStyle = new Theme3DStyle()
        {
        //图片
        IconFile = "",
        //图标显示大小
        IconScale = 0.6,
        //文本大小
        TextSize = 16.0,
        //文本颜色
        TextColor = Color.Green,
        };
        //添加一份SHP数据，并设置标签专题图样式
        bc_sceneControl.Scene3D.Theme3DManager.Add(@"C:\Myproject\Data\shp\shp_tjw_nc_ar
        ea_cgcs2000.shp", "标签1", bc_theme3DLabel);
        //更新数据
       bc_theme3DLabel.UpdateData();
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“标签专题图”按钮，添加标签专题图结果如下图所示。

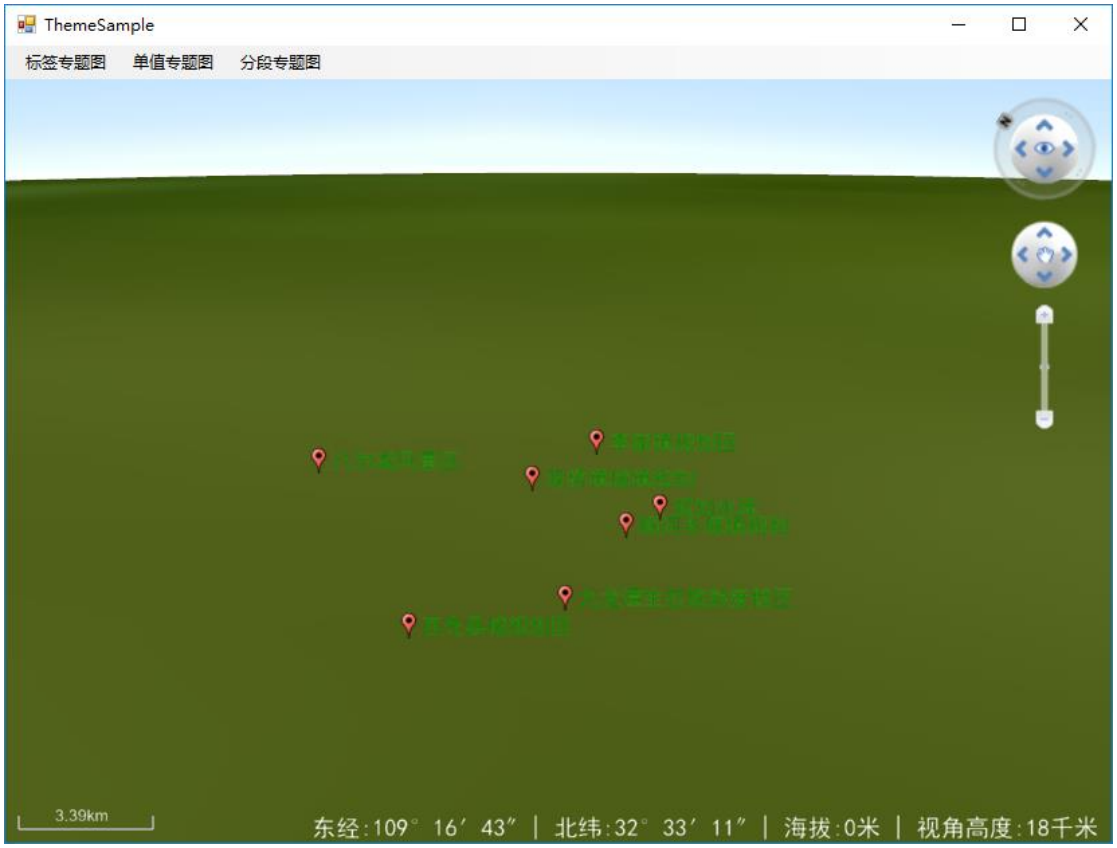


图 标签专题图

3. 添加单值专题图代码

双击 ThemeSample 窗体中的 uniqueTheme_MenuItem（单值专题图)按钮，进入代码编辑器。或者直接在属性界面，手动添加标签专题图的 Click 事件，输入 Click 事件名“uniqueTheme_MenuItem_Click，在事件中插入如下代码：

```
//******单值专题图******
private void uniqueTheme_MenuItem_Click(object sender, EventArgs e)
{
    //单值专题图子项
    Theme3DUnique bc_theme3DUnique = new Theme3DUnique();
    //设置单值字段
    bc_theme3DUnique.UniqueExpression = "NAME2004";

    Theme3DUniqueItem bc_uniqueItem1 = new Theme3DUniqueItem();
    //设置单值专题图字段值
    bc_uniqueItem1.Unique = "荆州市";
    bc_uniqueItem1.Style = new Theme3DStyle() { PolygonColor = Color.Red };
    //添加单值专题图子项
    bc_theme3DUnique.Add(bc_uniqueItem1);

    //添加第二个字段
    Theme3DUniqueItem bc_uniqueItem2 = new Theme3DUniqueItem();
    bc_uniqueItem2.Unique = "宜昌市";
    bc_uniqueItem2.Style = new Theme3DStyle() { PolygonColor = Color.Green };
    bc_theme3DUnique.Add(bc_uniqueItem2);

    //添加第三个字段
    Theme3DUniqueItem bc_uniqueItem3 = new Theme3DUniqueItem();
    bc_uniqueItem3.Unique = "武汉市";
    bc_uniqueItem3.Style = new Theme3DStyle() { PolygonColor = Color.Blue };
    bc_theme3DUnique.Add(bc_uniqueItem3);
    //添加一份SHP数据，并设置单值专题图样式
    bc_sceneControl.Scene3D.Theme3DManager.Add(@"F:\engineXDATA\shape\newshp\China
    CityArea.shp", "单值1", bc_theme3DUnique);
    //更新数据
    bc_theme3DUnique.UpdateData();
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“单值专题图”按钮，添加标签专题图结果如下图所示。

图 单值专题图

4. 添加分段专题图代码

双击 ThemeSample 窗体中的 rangeTheme_MenuItem (分段专题图)按钮, 进入代码编辑器。或者直接在属性界面，手动添加标签专题图的 Click 事件，输入 Click 事件名“rangeTheme_MenuItem_Click，在事件中插入如下代码:

```
//******分段专题图******
private void rangeTheme_MenuItem_Click(object sender, EventArgs e)
{
    Theme3DRange bc_theme3DRange = new Theme3DRange();
    //设置分段字段
    bc_theme3DRange.RangeExpression = "POP_CNTRY";

    //分段专题图分段，分段起始值0，结束值11527260
    Theme3DRangeItem bc_rangeItem1 = new Theme3DRangeItem();
    bc_rangeItem1.Start = 0;
    bc_rangeItem1.End = 11527260;
    //分段样式，设置颜色rgb 120 255 0 0
    bc_rangeItem1.Style = new Theme3DStyle() { PolygonColor = Color.FromArgb(120, 255,
    0, 0) };
    bc_theme3DRange.Add(bc_rangeItem1);
```

```
        //分段专题图分段，分段起始值11527260，结束值894608700
        Theme3DRangeItem bc_rangeItem2 = new Theme3DRangeItem();
        bc_rangeItem2.Start = 11527260;
        bc_rangeItem2.End = 894608700;
        //分段样式，设置颜色rgb 120 0 255 0
        bc_rangeItem2.Style = new Theme3DStyle() { PolygonColor = Color.FromArgb(120, 0, 2
        55, 0) };
        bc_theme3DRange.Add(bc_rangeItem2);
        //添加一份SHP数据，并设置分段专题图样式
       bc_sceneControl.Scene3D.Theme3DManager.Add(@"F:\engineXDATA\shape\newshp\world.
       shp", "分段1", bc_theme3DRange);
}
```

运行调试

编译运行代码后弹出如下界面，单击工具条上的“分段专题图”按钮，添加标签专题图结果如下图所示。

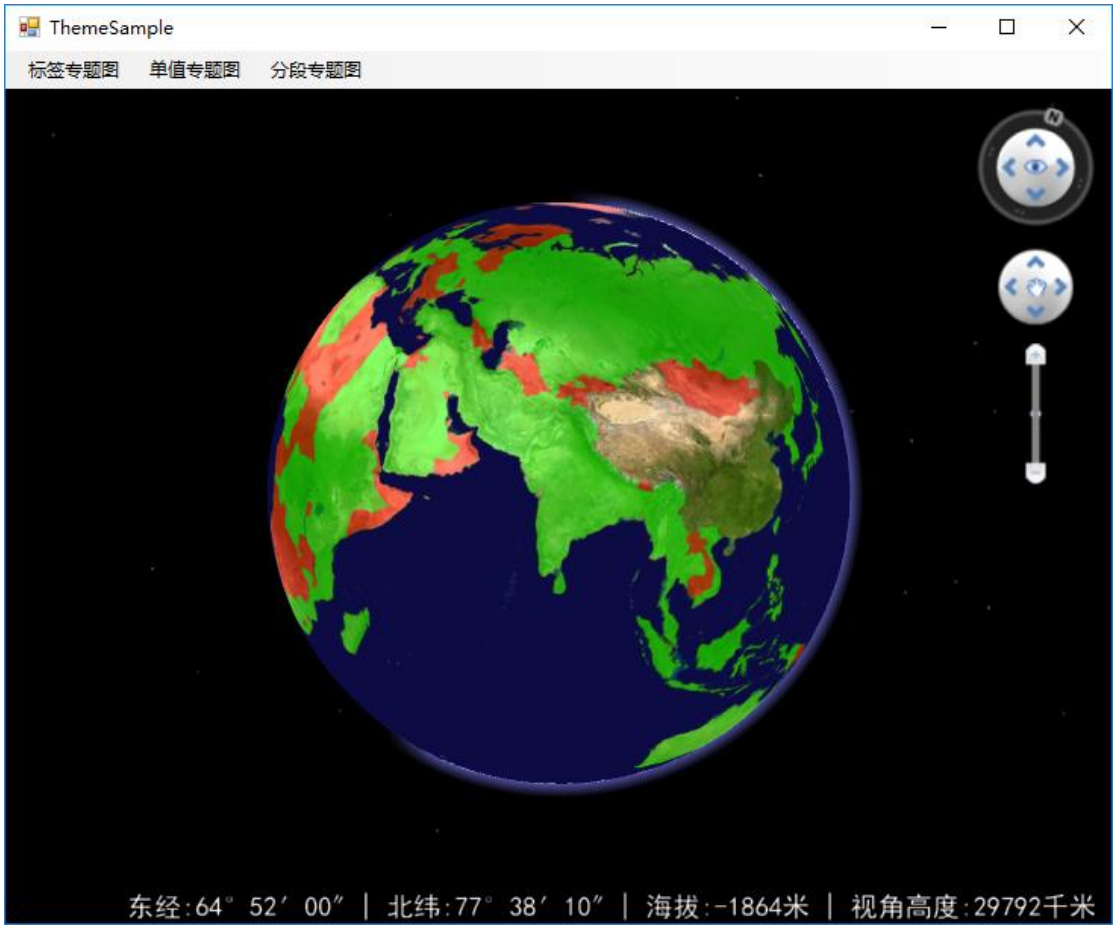


图 分段专题图

1. 倾斜摄影单体化（从 cgwj 找一两 tile 出来，画一些匹配的矢量面，实现单体化）

常见问题解答

产品安装和配置

问：BcEngineX 所需要的动态库及依赖库有哪些？

答：　　.NET 组件依赖 .NET Framework 4.0 及其以上版本。

　　.VC++2015 的一些运行库；

　　.NET 组件支持 2.0 及以上版本的 OpenGL；

问：为什么 BcEngineX Pro 绿色包启动出现卡在初始化界面并弹出错误信息？

答：　　检查需要的依赖.NET Framework 4.0 或以上版本、VC2015 和 VC2012 运行环境。

开发接口帮助

BcEngineX SDK 中所有的对象的编程接口列表，以及对每个对象的属性、方法、事件的详细信息，提供索引和全文搜索查询方式进行参考信息查找