BcEngineX-WebGL 用户手册

北京博超时代软件有限公司
二〇二一年四月

目录

欢迎使用

新闻、信息、支持、例子、指南等详细内容可联系博超时代软件公司三维引擎项目组进行咨询。

感谢您使用博超产品。
欢迎访问：www.bochao.com.cn

产品介绍

概述

BcEngineX WebGL 基于 Cesium，构建的一套跨平台高性能的 Web 三维引擎产品。技术框架主要分内核层和管理层两个层次对 Cesium 进行封装和优化。内核层包含优化后的 Cesium 源码，以及基于 Cesium 的扩展模块：渲染模块、调度模块、操作模块和分析模块。管理层对内核层的功能进行设计封装，主要对场景、图层、数据和事件功能进行分类管理 BcEngineX WebGL 三维引擎的技术体系和底层组件包括：

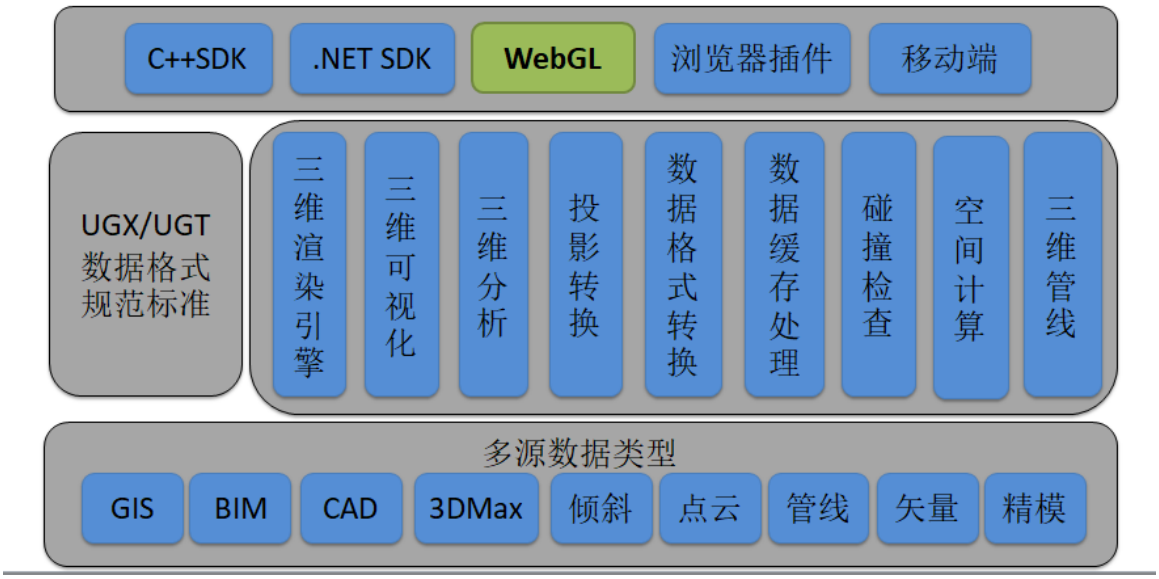
- 统一的数据格式包括：UGX、UGTile；支持多数据源数据如：BIM、CAD、激光点云、倾斜摄影、精模等。
- 内核模块：提供对三维场景可视化、对场景数据调度及解析、三维场景交互和分析。
- 管理模块：提供对数据类型的管理，数据管理结构化加工和数据简化，场景相关事件的捕获和派发。
- JavaScript SDK 模块：提供调用三维引擎接口，web 三维产品应用示例和开发文档。

BcEngineX 产品体系

BcEngineX 产品体系包含 BcEngineX C++ SDK、BcEngineX .NET SDK、BcEngineX WebGL、BcEngineX OCX 和基于 WebGL 的移动端。

产品架构

BcEngineX WebGL 是 BcEngineX 系列产品终端应用中的一员。BcEngineX WebGL 是一款具有跨平台高性能的 Web 三维引擎产品，支持 WINDOWS、MAC、IOS、ANDROID 等主流平台的各类浏览器平台。对外提供调用三维引擎的 JavaScript 接口开发效率更高，同时由于基于 Cesium 底层可以支持更加丰富的扩展。



模块结构

BcEngineX WebGL 产品结构包含接口层、功能封装层和核心层。接口层基于功能封装层和核心层，对外提供简单易用的接口，以及接口文档和 BcEngineX 展示例子。
功能封装层主要对通用功能进行封装，如场景设置、数据加载/卸载、用户事件处理等。
核心层包括 Cesium 和 Cesium 扩展模块，主要负责三维场景的渲染、数据调度、场景交互以及对核心功能进行封装。BcEngineX 主要对 Cesium 的数据调度和渲染流程进行了大量的扩展和优化。博超公司自定义 BIM 数据、BIM 缓存数据、点云数据、倾斜摄影数据、管线数据、矢量建模数据、基础几何体数据，均采用自定义着色器，提升了数据的扩展性和数据

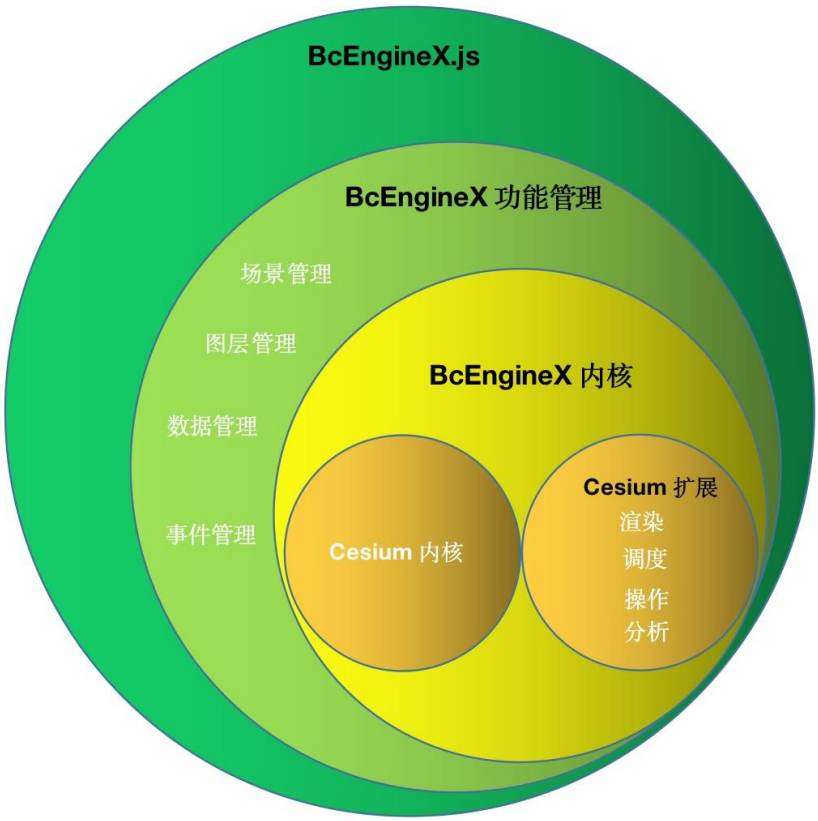
使用的灵活性。在数据调度上，自主设计实现的调度机制极大提升了数据调度的效率。

用户不仅能够通过 BcEngineX 的接口使用三维场景，熟悉 Cesium 的用户还可以通过 BcEngineX 暴露的 Cesium 接口对三维场景进行更灵活丰富的操作。

具体模块包含功能如下表：

模块	一级子模块	二级子模块	功能
内核	渲染模块		对三维场景进行可视化
	调度模块		对场景数据的调度和解析
	操作模块		对三维场景进行交互
	分析模块		对三维场景进行分析
管理	场景管理模块		管理场景各功能模块运转
	图层管理模块		对不同数据类型进行增/删/改
	数据管理模块		对所有数据进行结构化加工，数据转换、数据简化
	事件管理模块		捕获和派发三维场景中事件
接口	JavaScript SDK 模块		对外提供调用三维引擎接口，混淆的 JS 文件
	DEMO 模块		Web 三维产品应用示例
	文档模块		开发文档/部署文档/更新文档

各模块包含关系如下图：



主要功能

- 数据支持
  - 支持专业设计软件模型：PDMS、Smart Plant 3D、PDS、REVIT、CAD 等常用图形设计平台；
  - 支持常规三维模型格式：fbx、obj、3ds 等模型格式；
  - 支持专业机械软件模型： 包括 CATIA、SOLIDWORKS、UG、INVENTOR 图形平台；
  - 支持各类 GIS 地理信息数据的三维展示，如：TIN 地形、在线影像数据、GeoJson、MVT、Gltf、3D Tiles 等
  - 支持博超公司自定义 UGX 模型、BIM 缓存数据、点云数据、倾斜摄影数据、管线数据、矢量建模数据、基础几何体数据
  - 模型效果支持：对三维模型的材质、纹理、光照、天空盒等有良好的渲染能力；
- 数据优化能力
  - 模型格式：基于国际通用的 IFC 标准进行扩展，形成专有的三维格式，支持所有符合 IFC 标准的模型；
  - 模型轻量化：不仅能精简模型顶点、三角面的数量，还能从源数据中分析模型层次与引用关系，实现模型面的复用；
  - 场景优化：支持复杂场景优化，包括 LOD、八叉树分块、八叉树视距剔除、遮挡剔除、像素剔除、纹理压缩、流式加载等优化技术；
- 三维浏览功能
  - 相机操作：提供 WASD 按键操作和鼠标操作，支持上帝视觉（第三人称）和第一人称视觉。同时提供重力、穿透等物理特性；
  - 模型操作：提供高亮、隐藏、剖切、旋转、平移、缩放（非等比缩放）、定位、颜色设置等常用模型操作；
  - 2D 元素支持：可以在场景中增加云线、文字批注，支持现场图像采集；
  - 基本建模功能：可以在场景中绘制球、圆柱、长方体、导线、钢管等三维模型；
  - 测量功能：当前提供实体测量、面积测量、距离测量（模型到模型距离、模型到地形距离、模型到指定点的距离）等测量功能，支持碰撞检测；
  - 工程巡游：可以在场景中播放已经录制好的巡游路径或者巡游数据，支持回退、暂停及取消；
  - 地形管理：支持地形的显隐及整平；
  - 图层管理：当前可对地形影像及高程进行添加、隐藏及删除操作；
  - 数据展示：支持倾斜摄影、点云数据的展示、透明及挖洞操作；
  - 动画展示：提供飞机巡游、淹没分析等动画效果；
- 移动端与 VR 设备
  - 移动端：webgl 支持移动端应用，能流畅的展示工厂、变电站、轨道交通等模型。Android 端及 Surface 端的应用也已上线；
  - VR 设备：基于 HTC 的 VR 设备，提供沉浸式体验，在 VR 设备上展示三维虚拟场景，并提供在 VR 场景中的人机交互；
- 场景编辑与 API
  - 场景编辑：支持模型的导入与增减、状态修改（颜色设置），支持 3D 场景进行导出与复用；
  - 二次开发 API：提供标准的二次开发 API，完善的开发文档、丰富的开发案例，为客户的业务拓展提供良好的开放性；
  - 其他操作：引擎提供模型在三维场景中快速定位，设置巡航路径并进行自动巡航等功能。

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。

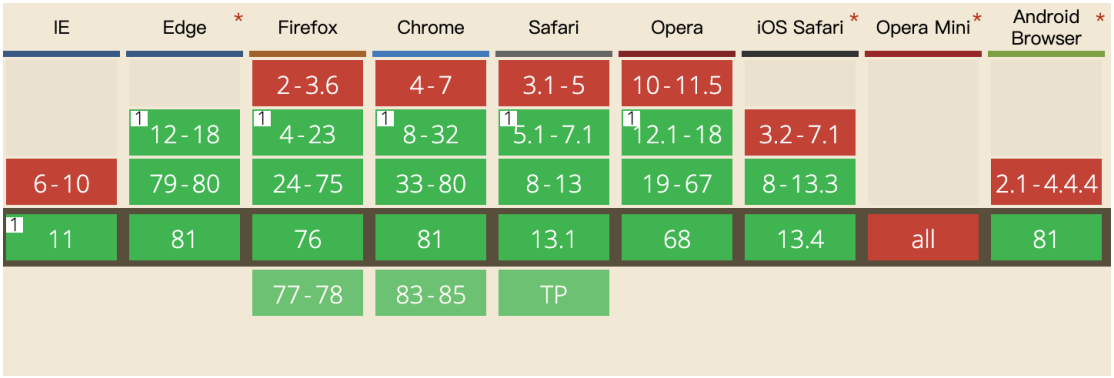
软件要求

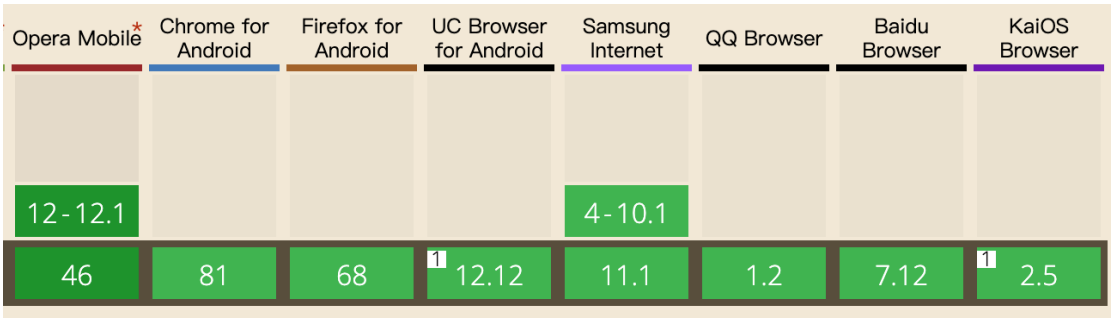
操作系统要求

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

基础软件要求

- 跨平台，与操作系统无关，仅需支持WebGL1.0浏览器即可。当前支持的主流浏览器如下图：





开发环境要求

- Node.js（编译调试需要）

产品入门

欢迎使用 BcEngineX WebGL 产品入门教程。本帮助文档将以发布的 BcEngineX WebGL 包为基础，基于 JavaScript 语言的入门范例程序为初次接触 BcEngineX WebGL 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 配置 WebGL 环境

打开一份 BcEngineX WebGL 包，会看到提供的 WebGL 引擎库和资源文件、开发文档和示例程序及展示例子内容，如下图所示：

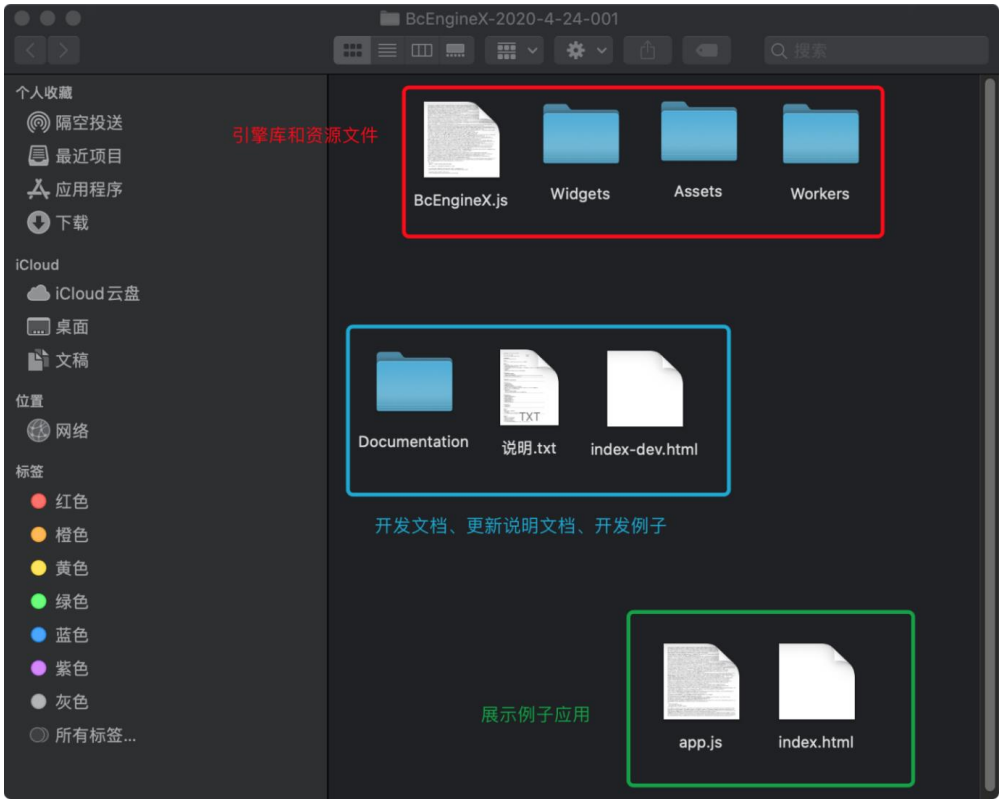
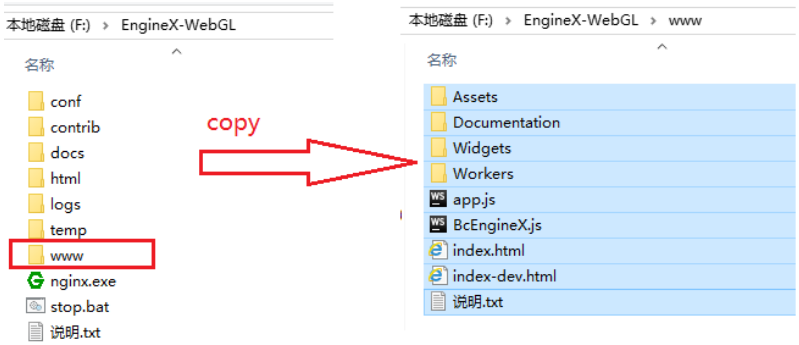


图 BcEngineX Webgl 包结构

解压 BcEngineX WebGL 的包到 nginx 发布目录 www 下：



打开 nginx 目录下 conf/nginx.conf 配置文件，找到 http 下 server 节点对发布主机进行配置，配置如下内容：

listen:端口号这里配置为 8009 端口

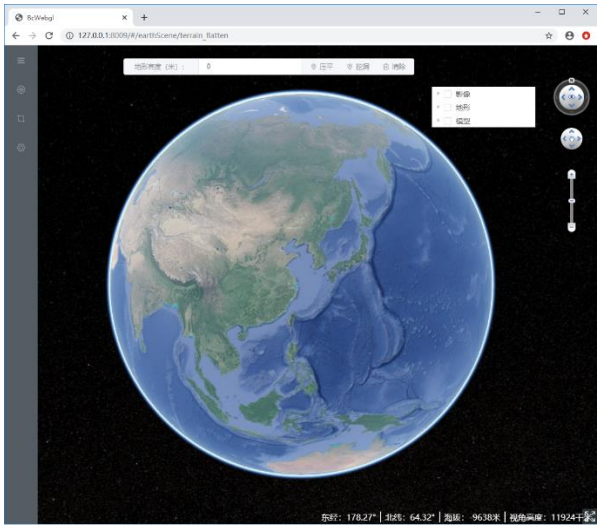
location:URL 匹配特定指定位置后的设置,每部分包涵若干个命令

root：配置网页位置，目录为./www/

add_header:为 HTTP 响应添加头 add_header Access-Control-Allow-Origin * always;

```
34
35      server {
36          listen       8009;
37          server_name  localhost;
38
39          #charset koi8-r;
40
41          #access_log  logs/host.access.log  main;
42
43          location / {
44
45              root   ./www/;
46              add_header Access-Control-Allow-Origin * always;
47
48          }
49
50          #error_page  404              /404.html;
51
```

启动 nginx 服务，打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/，运行：



第二步 创建一个 WebGL 工程

在 webgl 工程发布 www 目录下创建一个 SceneManager.html 文件，使用编辑器打开在 head 头中设置 html 页面显示样式并导入 WebGL 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl-SceneManger</title>
    <style>
        html,
        body {
            height: 100%;
            width: 100%;
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，在<script>标签中创建两个方法 createEarthScene 和 createCommonScene 分别创建地球场景和普通场景，并在创建场景前判断场景是否为空，不为空则销毁前场景再创建，添加代码如下：

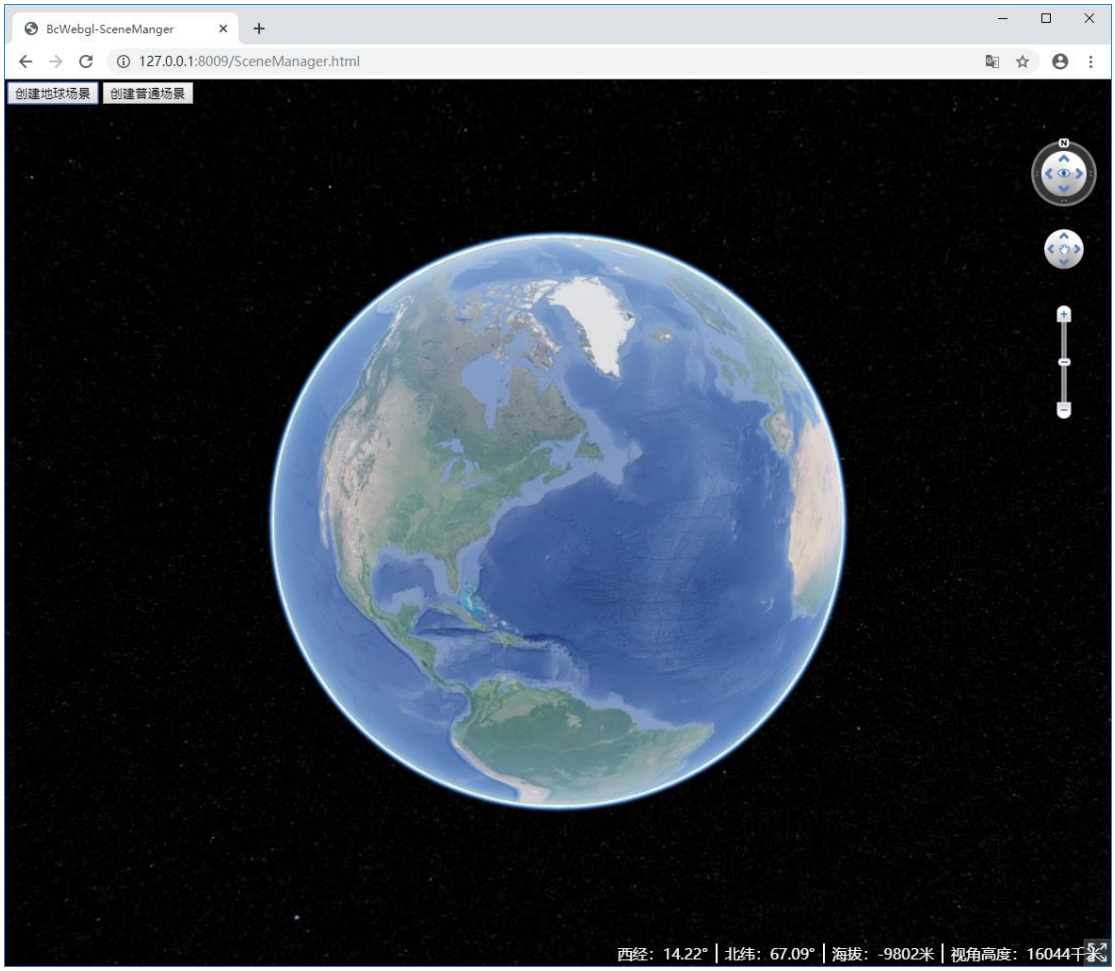
```
<div id = "engineContainer"></div>
```

```
<script type="text/javascript" src="./BcEngineX.js"></script>
<script>
    var scene;//场景
    window.scene = scene;
    //创建地球场景,0 地球场景，1 普通场景（目前不支持）
    function createEarthScene() {
        if(scene != null)
         {
              scene.destroy();
         }
         scene = BcEngineX.createBcScene("engineContainer", {type:0});
    }
     //创建普通场景
     function createCommonScene(){
          if(scene != null)
          {
              scene.destroy();
          }
          scene = BcEngineX.createBcScene("engineContainer",{type:1});
     }
</script>
```

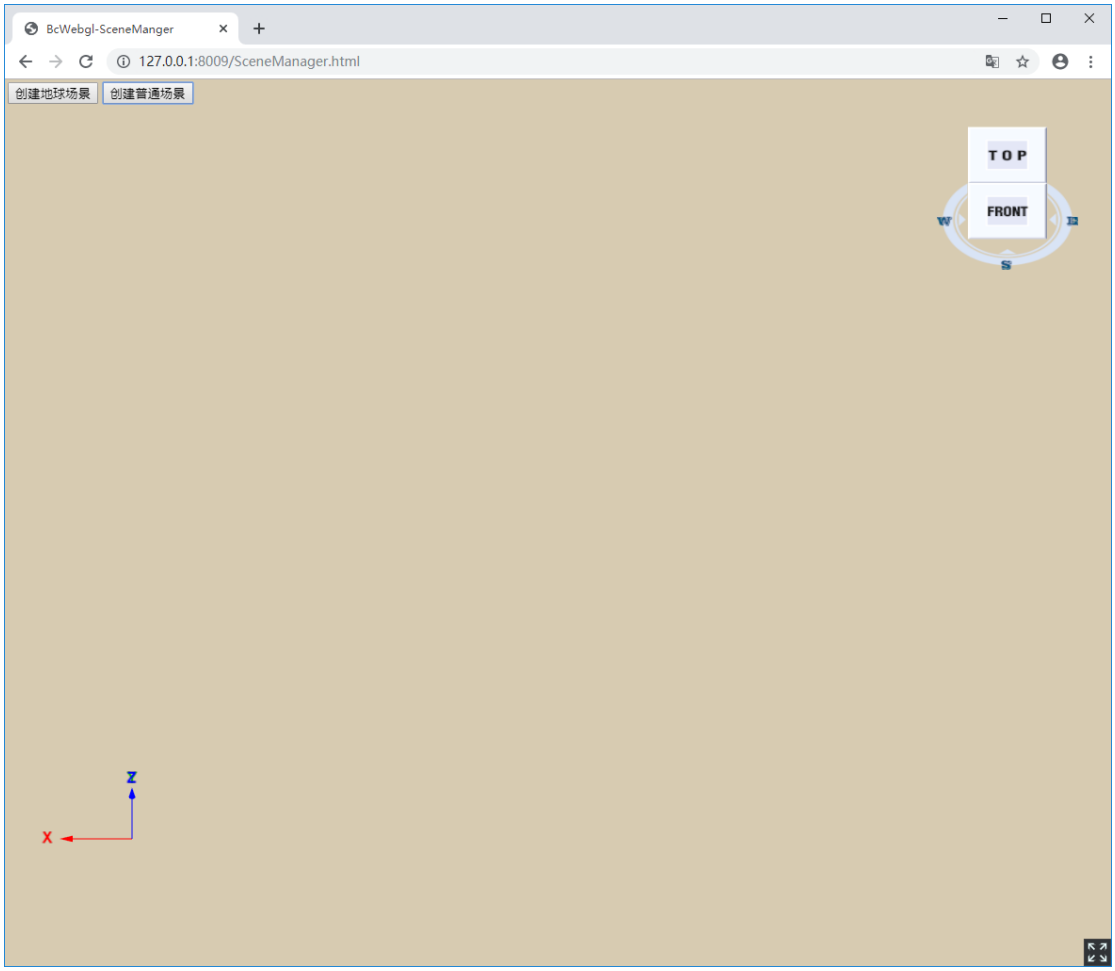
添加完创建场景的方法，继续再<body>中添加一个 table，并添加两个<button>来调用创建场景的方法，添加代码如下：

```
<table>
    <tbody>
    <tr>
        <td>
            <button id="createEarthScene" onclick="createEarthScene()">创建地球场景</bu
         tton>
            <button id = "createCommonScene" onclick = "createCommonScene()">创建
普通场景</button>
        </td>
    </tr>
    </tbody>
</table>
```

打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/SceneManager.html，运行，第一个 webgl 工程就创建成功，点击“创建地球场景”button:



再点击“创建普通场景”button:


BcWebgl-SceneManger
127.0.0.1:8009/SceneManager.html
创建地球场景
创建普通场景
TOP
FRONT
W
E
S
Z
X

基础知识

影像数据

BcEngineX_WebGL 支持将预处理*.img、*.tif、*.tiff 格式的影像文件，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

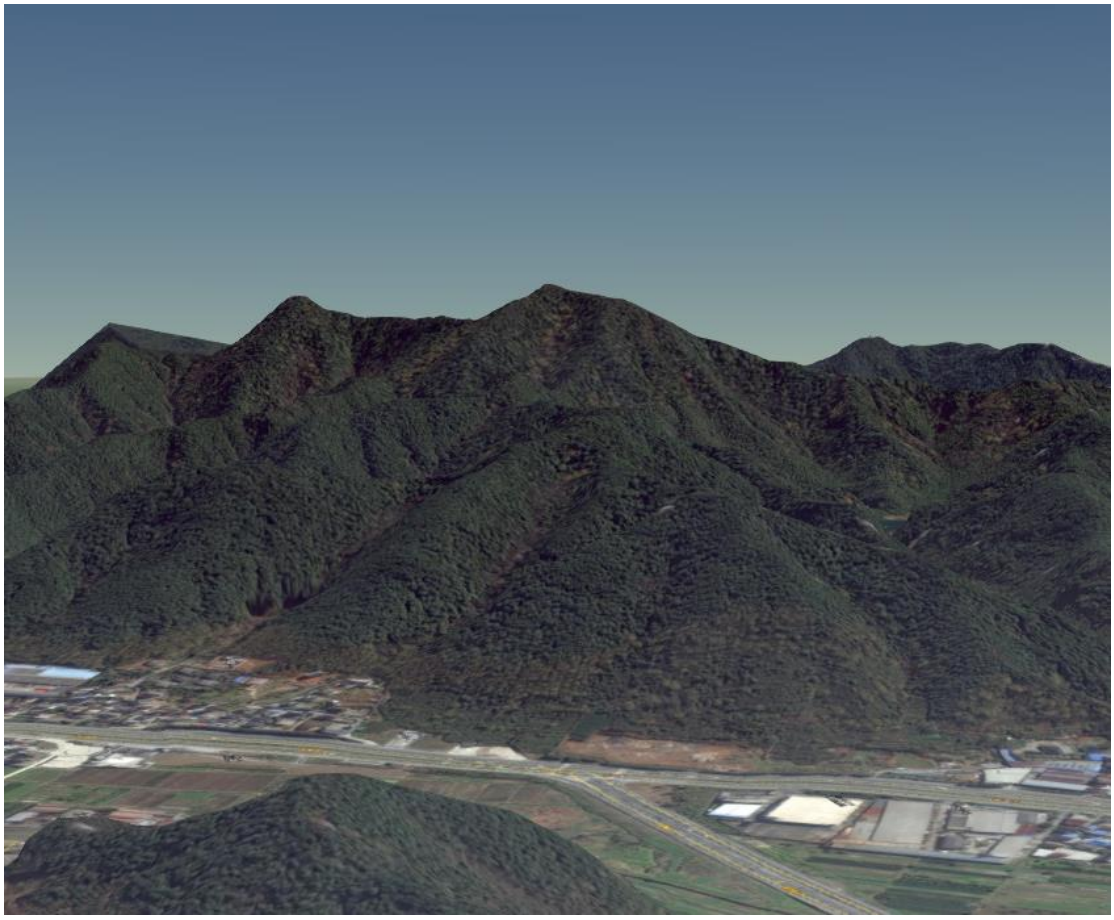
BcEngineX_Webgl 支持预处理后*.asc、*.dem、*.tif 格式的地形文件加载。地形数据是我们进行地形分析的基础。如我们可以利用地形数据提取坡度坡向的基础地形因子，以及进行三维分析等较复杂的地形分析功能。只有构建高质量的地形数据，才能保证我们后续分析结果的可靠。

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_Webgl 中的地形数据，必须通过 BcEngineX Pro 平台数据预处理，才能获得支持的地形数据，BcEngineX Pro 平台支持将*.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_Webgl 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。

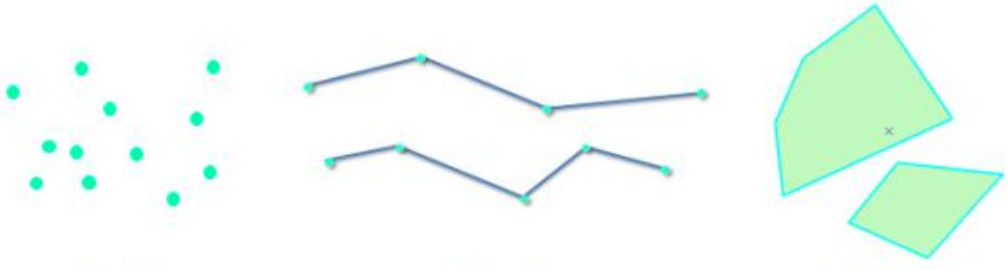


图1：点　　图2：线　　图3：多边形

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_Webgl 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。

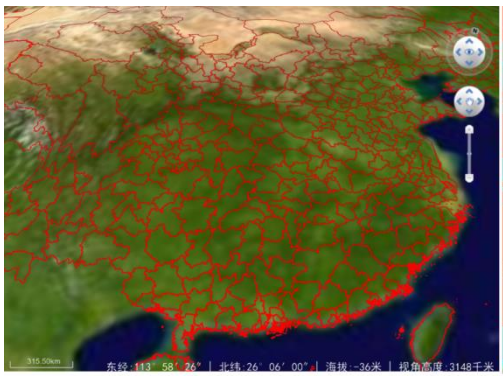


图 普通图层

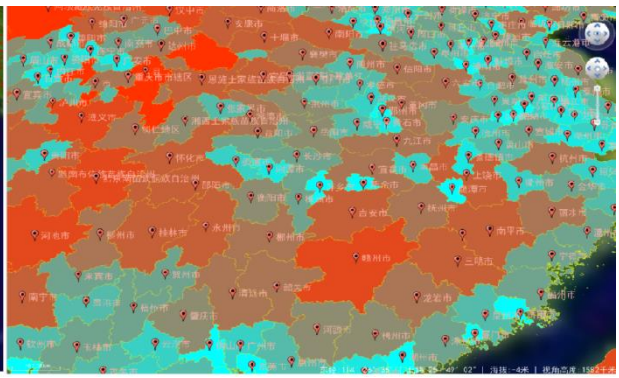


图 专题图层

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_Webgl 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）

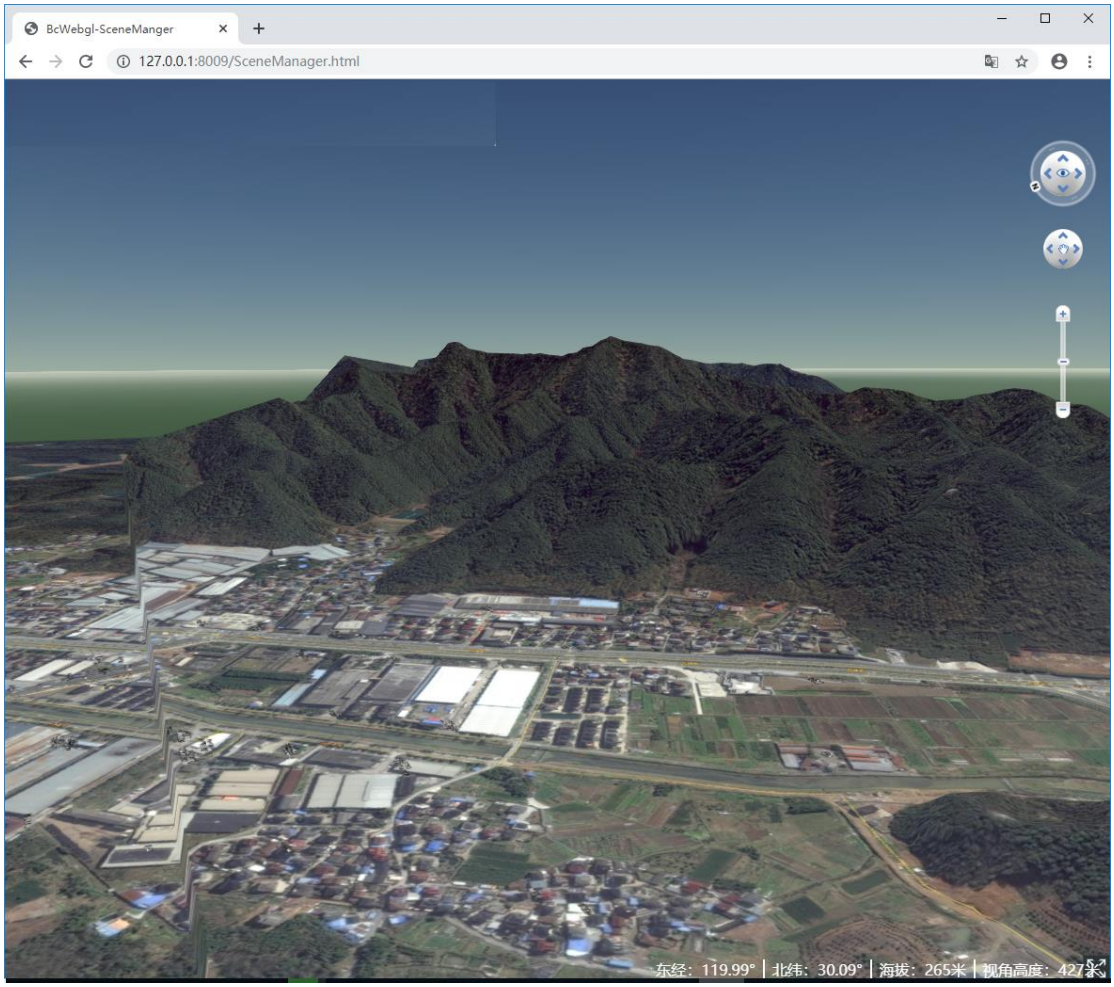


图 2 进行 Pitch 拉平 操作后（地球场景）

旋转（Roll）操作

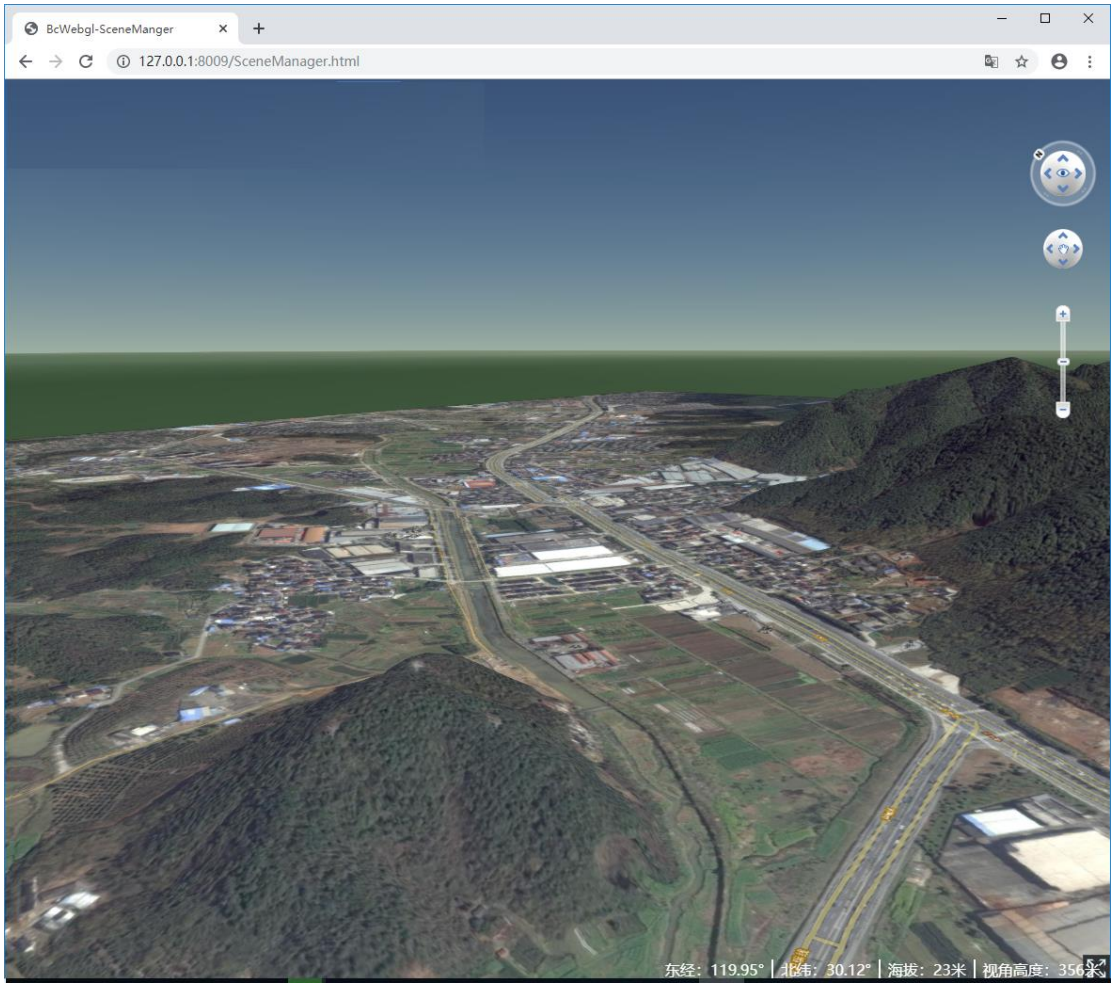


图 5 进行 roll 旋转操作后（地球场景）

普通场景下：

拉平竖起（Pitch）操作

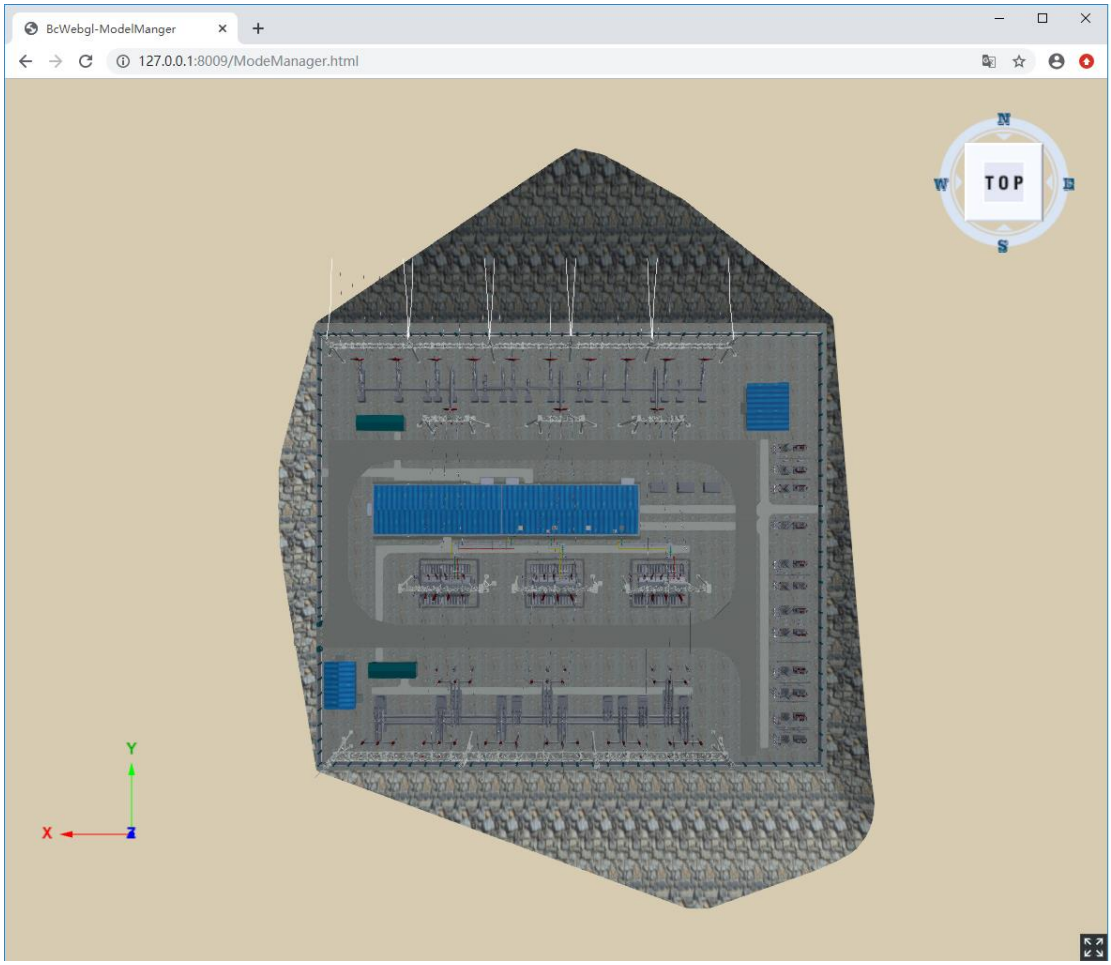


图 3 进行 Pitch 竖起操作后（普通场景）

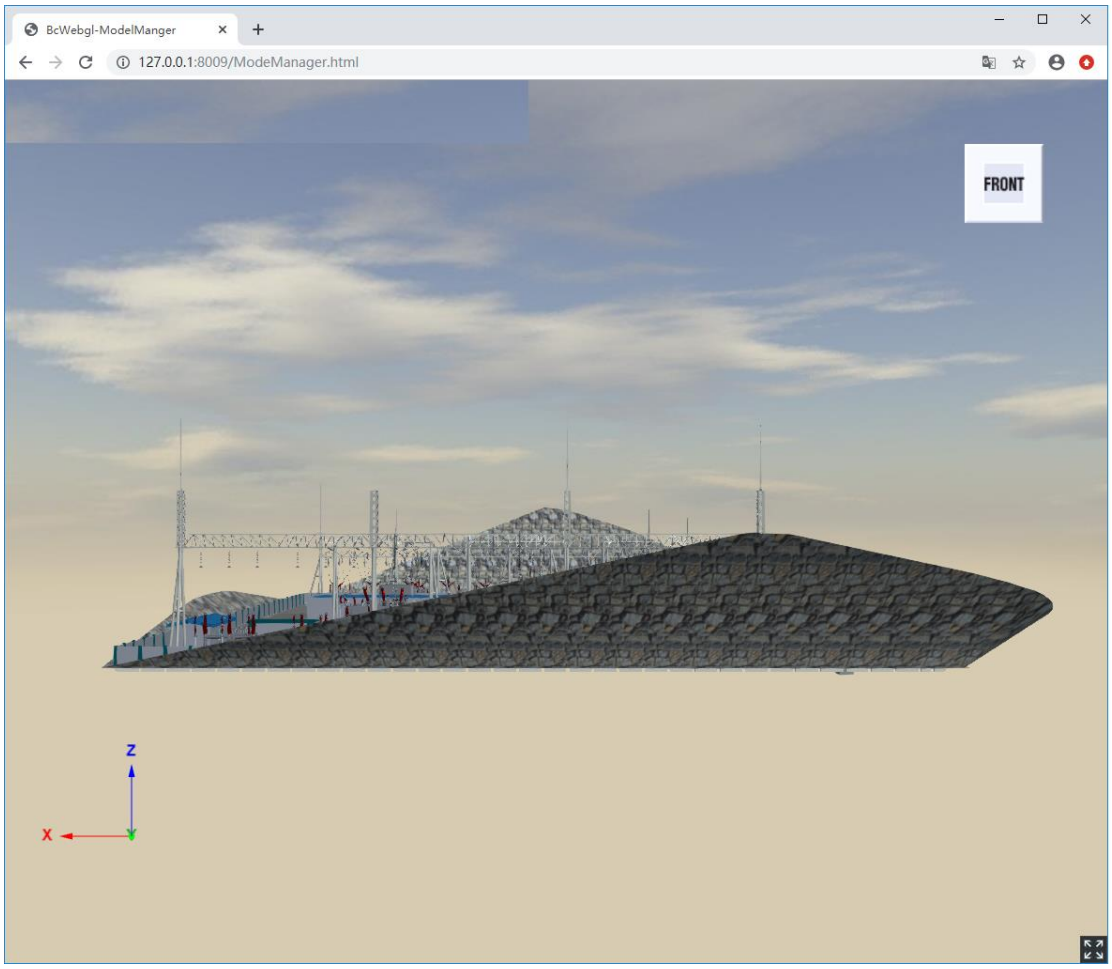


图 4 进行 Pitch 拉平 操作后（普通场景）

旋转（Roll）操作

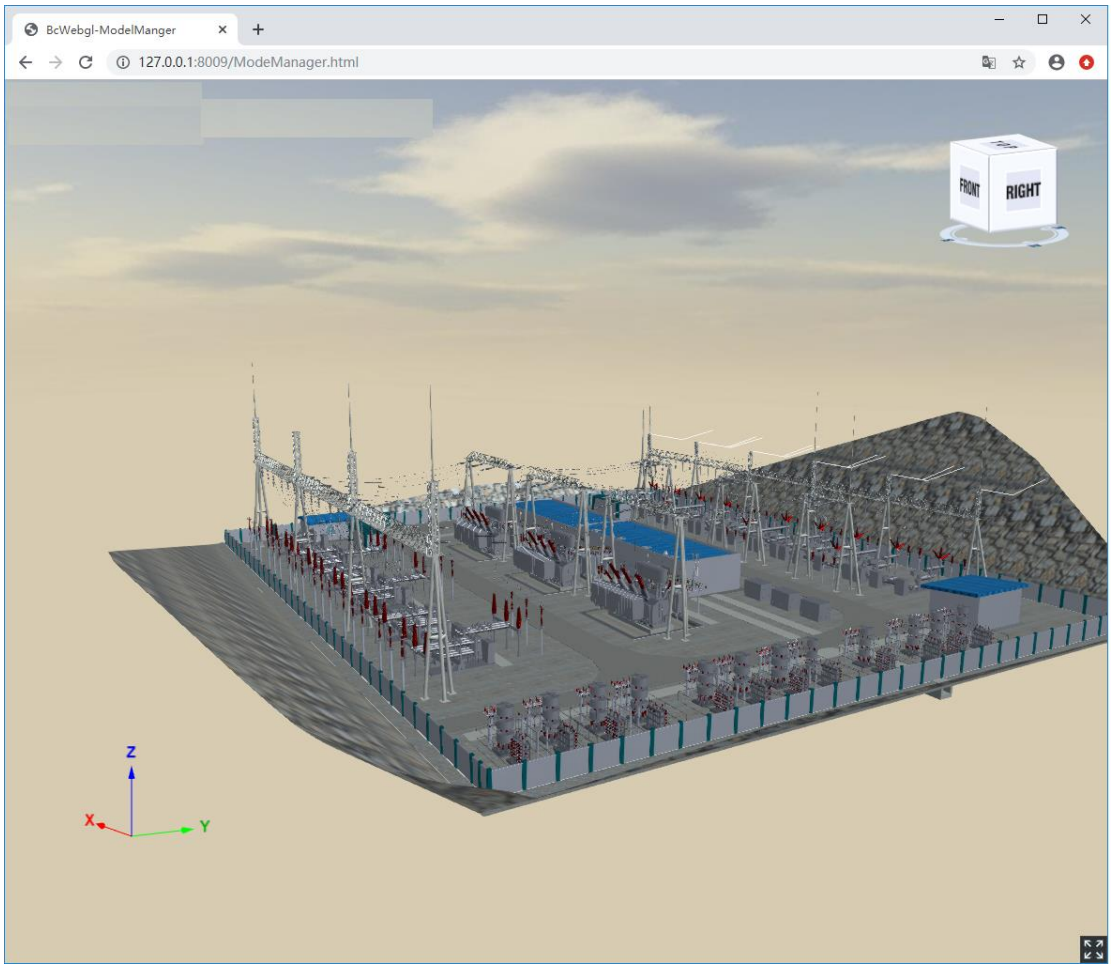


图 6 进行 roll 旋转操作后（普通场景）

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	

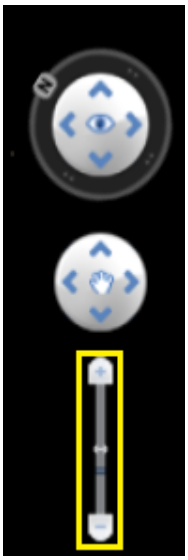
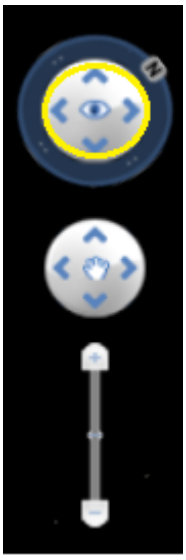
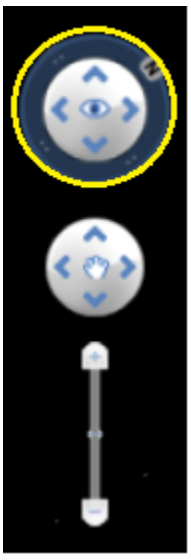
放大缩小	鼠标中键滚轮或鼠标右键按下上下拖动		
三维地图场景进行倾斜	按住鼠标中键上下拖动		
绕场景中心旋转	按住鼠标中键左右拖动		

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。

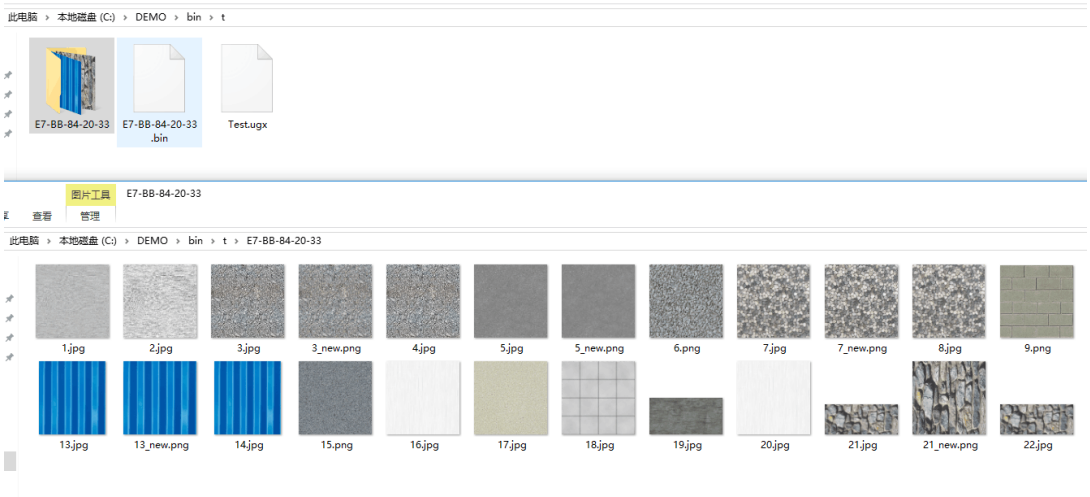
文件组织结构

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；

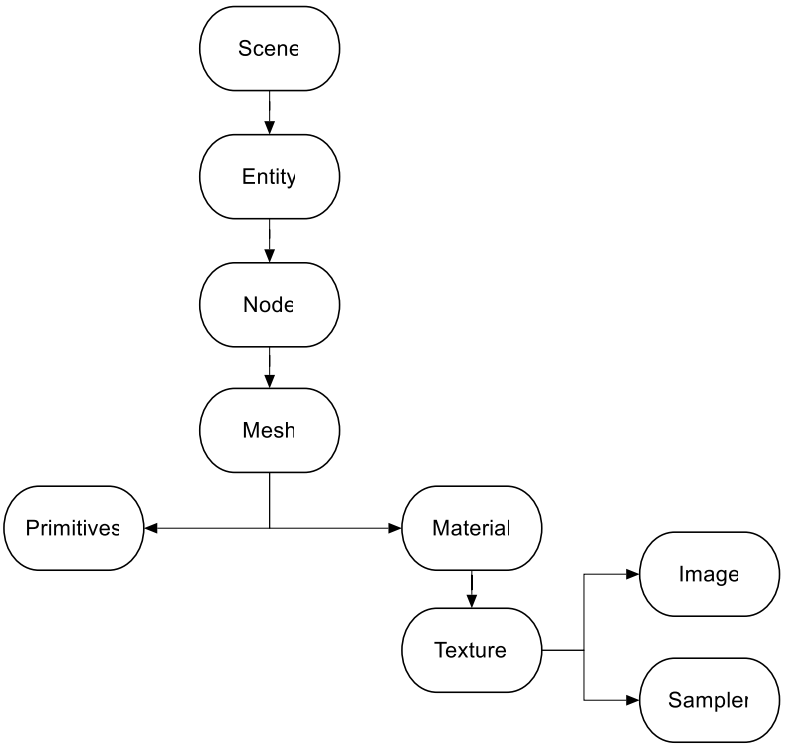
(3) 图片文件夹：纹理图片文件。



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。



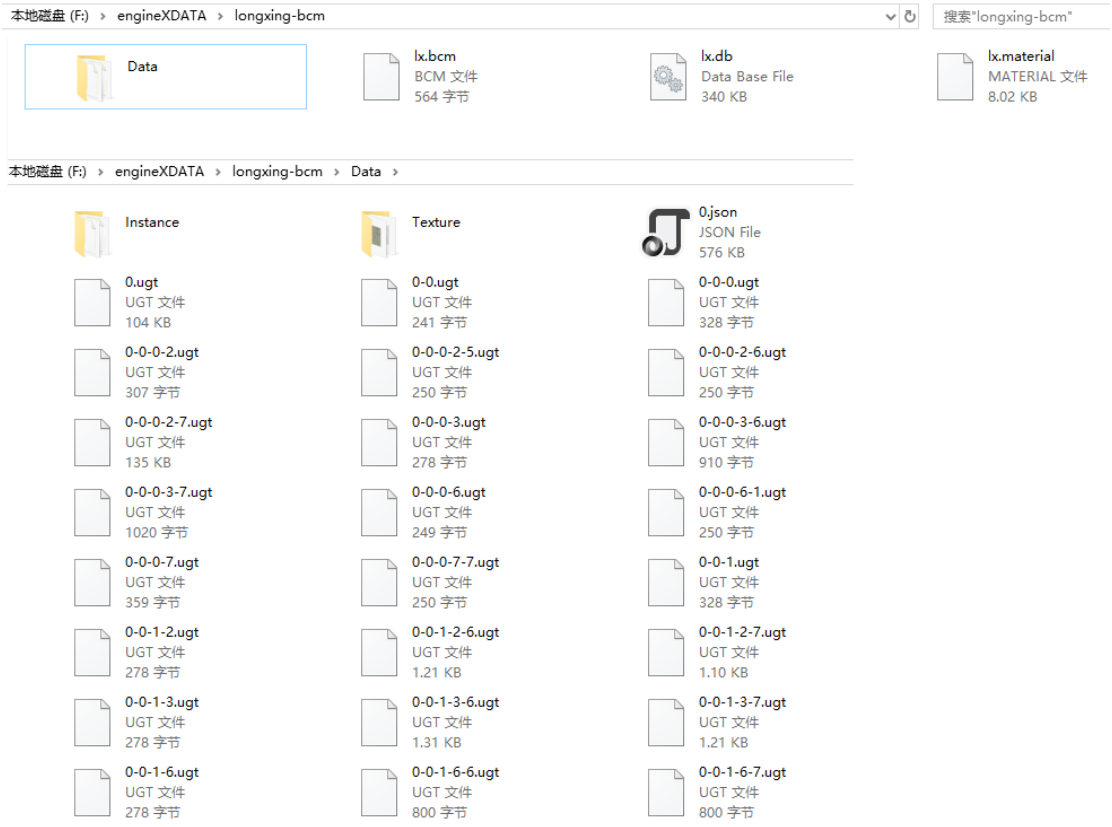
UGT

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

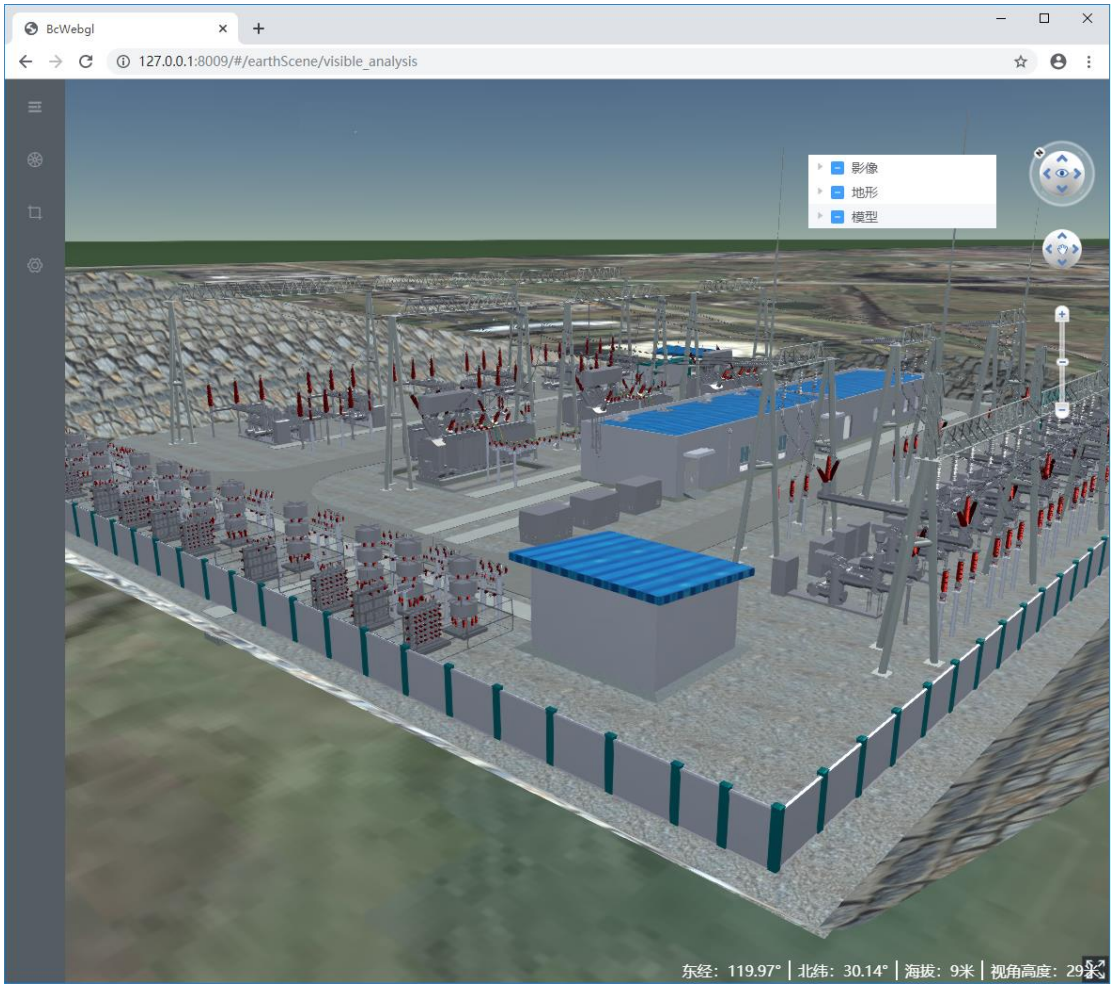
(5) 数据压缩设置：将 ugt 瓦块以 7z 压缩方式压缩成 ugtz 格式减小占用内存，提高在服务端访问下载数据时的速率。

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 Webgl 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。

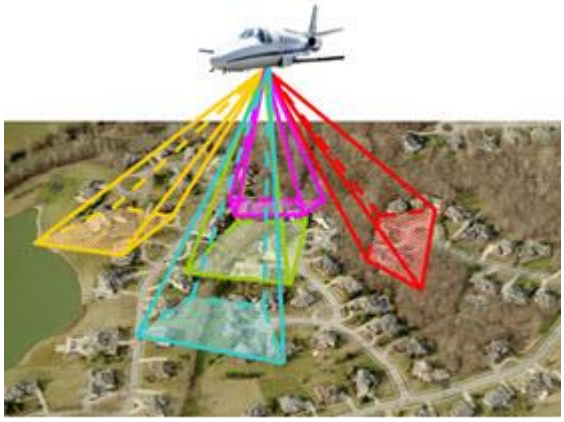


图 倾斜摄影采集

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 WebgI 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。

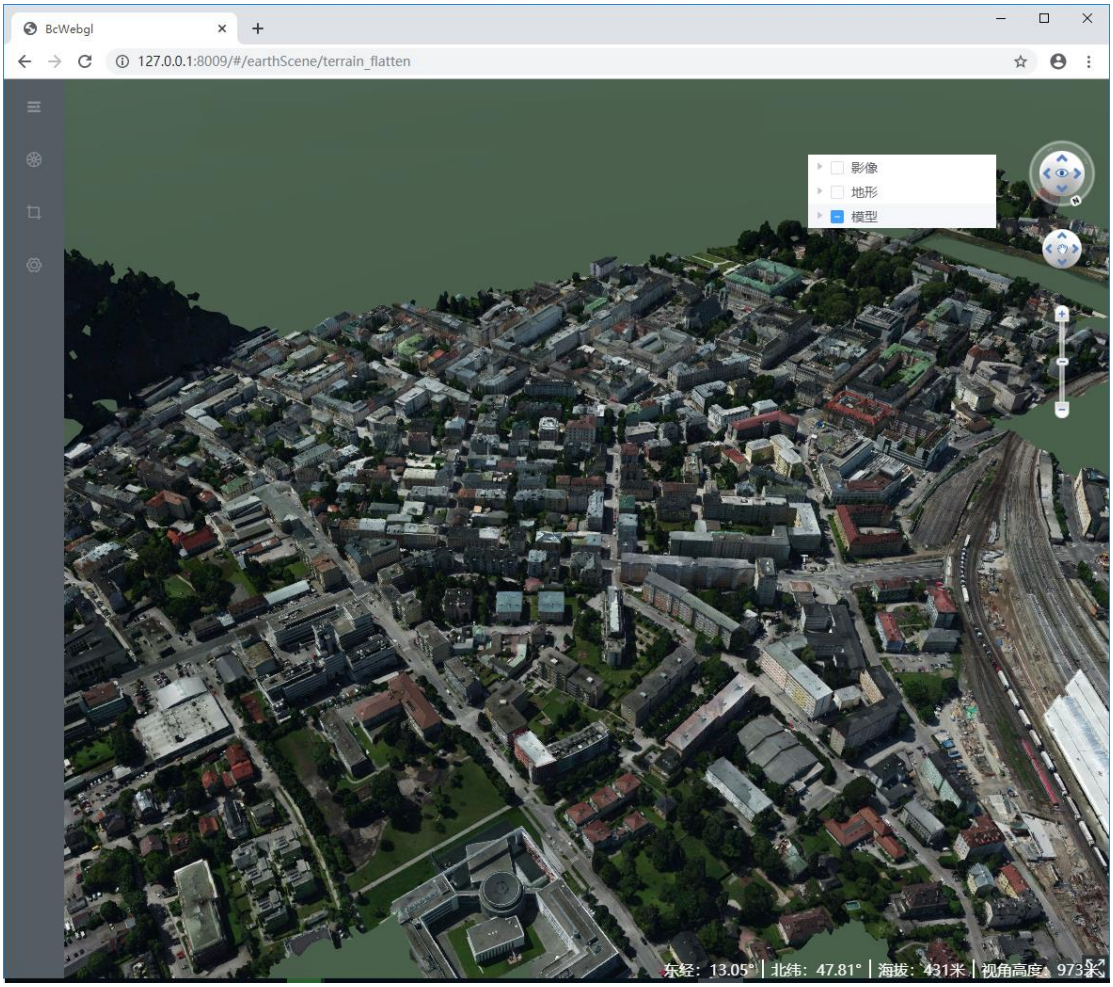


图 倾斜摄影浏览

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 Webgl 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 点云

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 Webgl 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：

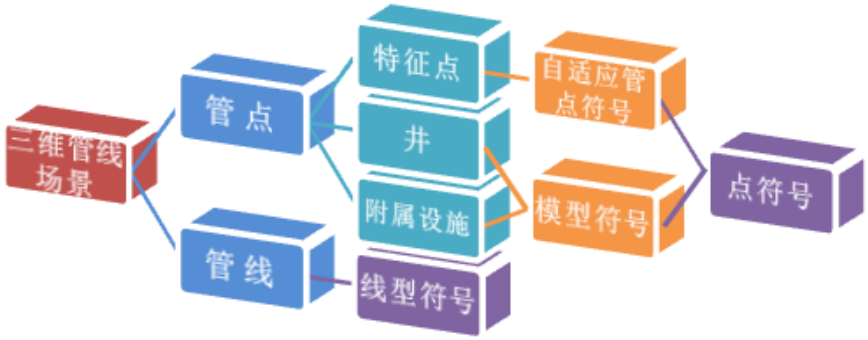


图 三维管线场景组成及展示方式

BcEngineX WebgI 支持对管线模型加载浏览，同时支持对管线数据查询、分析、批注等操作。

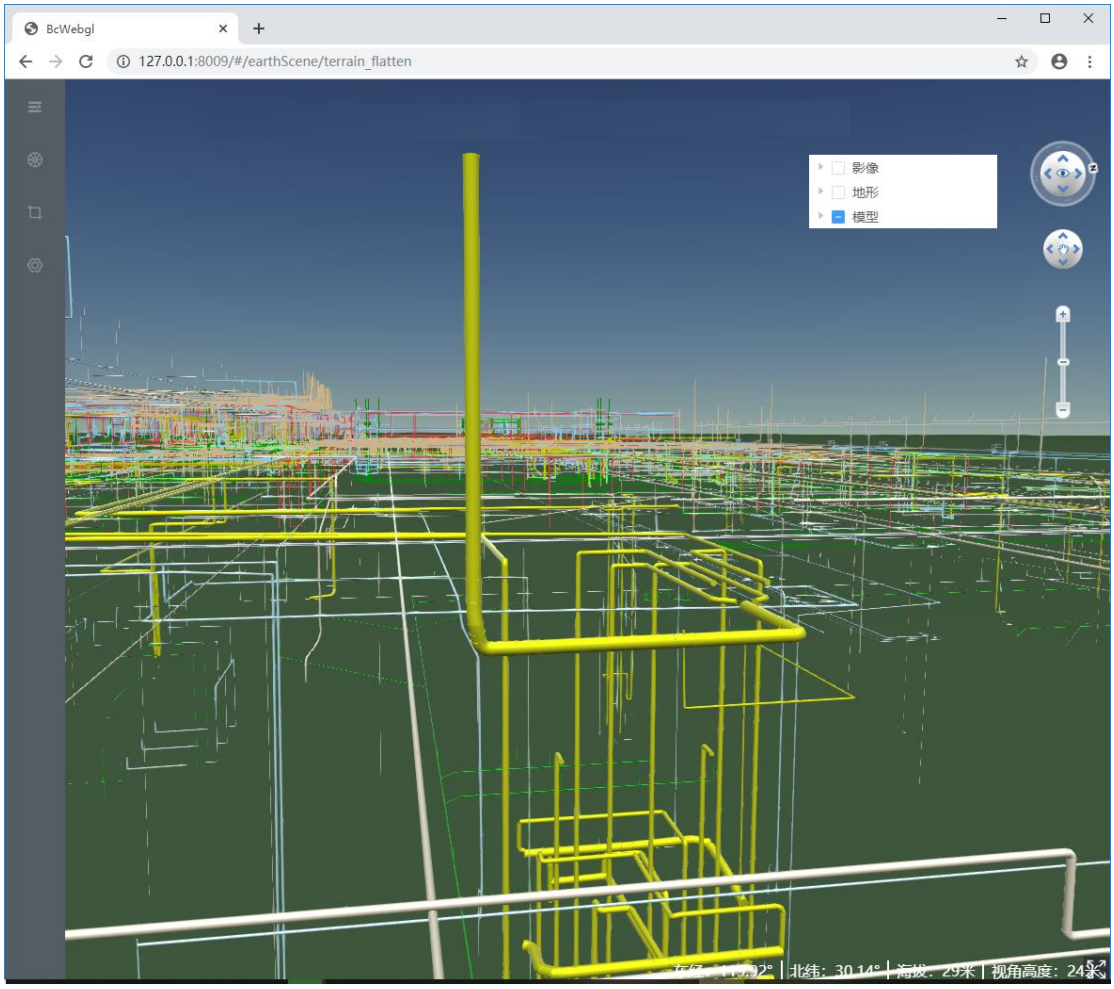


图 管线

范例程序说明

场景属性控制

第一步 创建一个工程

1.在 webgl 工程发布 www 目录下创建一个 SceneControlsManger.html 文件，参考创建一个 webgl 工程，
在 head 头中设置 html 页面显示样式并导入 Webgl 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl-SceneManger</title>
    <style>
        html,
        body {
            height: 100%;
            width: 100%;
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

2.在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，如下：

```
<div id = "engineContainer"></div>
<script type="text/javascript" src="./BcEngineX.js"></script>
```

3.在<boday>中再添加一个<script>标签，并添加创建场景的方法 createEarthScene()和 CreateCommonScene（），开关导航盒 setNavigationVisibility（)、开关状态栏 setInfoBarVisibility（)、设置相机位置 setCameraPosition()、开关检查 setInspector()，添加代码如下：

Function	Content
CreateCommonScene ()	添加普通模型
createEarthScene ()	创建地球场景
setNavigationVisibility ()	开关导航栏
setInfoBarVisibility ()	开关状态栏
setCameraPosition()	设置相机位置
setInspector()	开关检查（帧率、网格等）

```
<script>
    var scene;//场景
    window.scene = scene;
    //创建地球场景,type:0 地球场景，type:1 普通场景
    function createEarthScene() {
        if(scene != null)
        {
            scene.destroy();
        }
        scene = BcEngineX.createBcScene("engineContainer", {type:0});
    }
     //创建普通场景
     function CreateCommonScene() {
        if(scene != null)
        {
            scene.destroy();
        }
            scene = BcEngineX.createBcScene("engineContainer", {type:1});
            scene.setPickEnable(true);
              //模型选中事件
            if (scene.pickObjectEvent){
                scene.pickObjectEvent.addEventListener(function (scene) {
                    console.log(scene);
                })
            }
        }
         //设置导航栏是否可见
    var NavigationVisibility = false;
    function setNavigationVisibility() {
```

```
        if (scene != null)
        {
            scene.setNavigationVisibility(NavigationVisibility);
            NavigationVisibility = !NavigationVisibility;
        }
    }
    //设置状态栏是否可见
    var InfoBarVisibility = false;
    function setInfoBarVisibility() {
        if (scene != null)
        {
            scene.setInfoBarVisibility(InfoBarVisibility);
            InfoBarVisibility = !InfoBarVisibility;
        }
    }
    //设置相机位置(经度、纬度、高，单位：米)
     function setCameraPosition(){
        if (scene.sceneType == 0)
        {
            scene.setCameraPosition(119.0, 30.0, 50.0);
        }
     }
     //设置Inspector显示
     function setInspector(){
        scene.showViewerInspector = !scene.showViewerInspector;
    }
</script>
```

4.在< boday >标签中添加一个<table>标签并添加<button>按钮用于调用 function 方法，如下：

Buttons	Text
CreateCommonScene	添加普通模型
createEarthScene	创建地球场景
setNavigationVisibility	开关导航栏
setInfoBarVisibility	开关状态栏
setCameraPosition	设置相机位置
setInspector	setInspector 开关

```
<table>
    <tbody>
        <tr>
            <td>
            <button id = "createEarthScene" onclick="createEarthScene()">创建地球场景
```

```
            </button>
            <button id ="CreateCommonScene" onclick ="CreateCommonScene()">创建普通场
            景</button>
              </td>
          </tr>
          <tr>
              <td>
            <button id ="setNavigationVisibility" onclick ="setNavigationVisibility()">开关导航
            栏</button>
            <button id ="setInfoBarVisibility" onclick ="setInfoBarVisibility()">开关状态栏
            </button>
            <button id ="setCameraPosition" onclick ="setCameraPosition()">设置相机位置
            </button>
            <button id ="setInspector" onclick ="setInspector()">Inspector开关</button>
              </td>
          </tr>
      </tbody>
</table>
```

第二步 运行

在浏览器端打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/SceneManager.html，运行，示例 demo 就启动成功，点击“创建地球场景”创建一个 wgs84 地球场景，再点击“开关导航盒”、“开关状态栏”则控制场景属性显隐，如下：

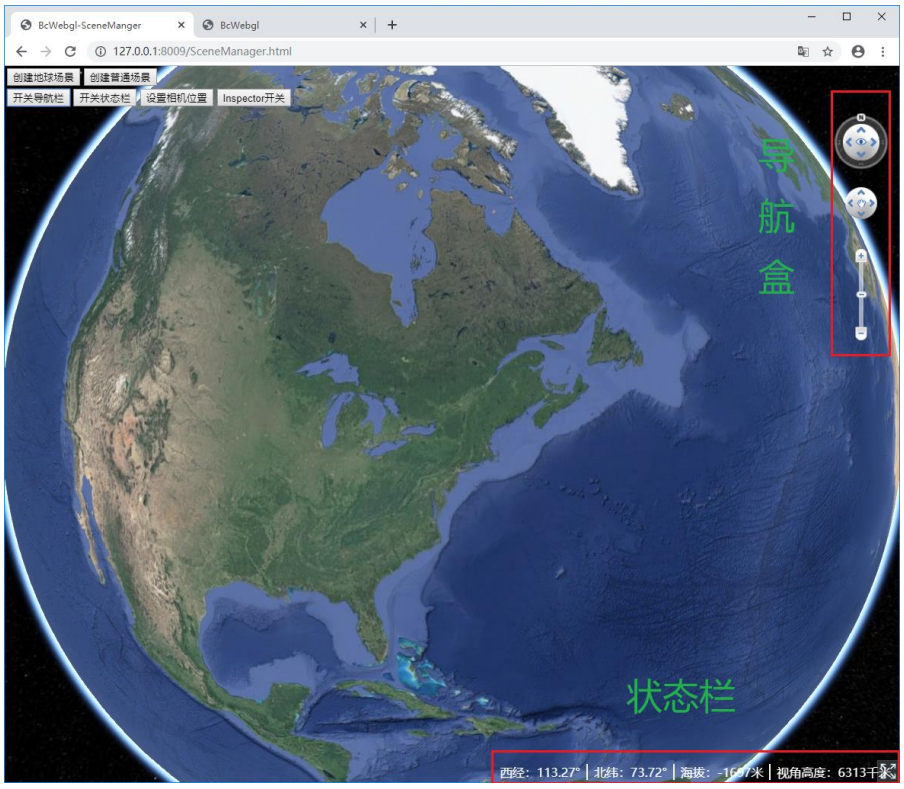


图 地球场景

点击“创建普通场景”，则可以控制普通场景属性信息显隐控制，如下：

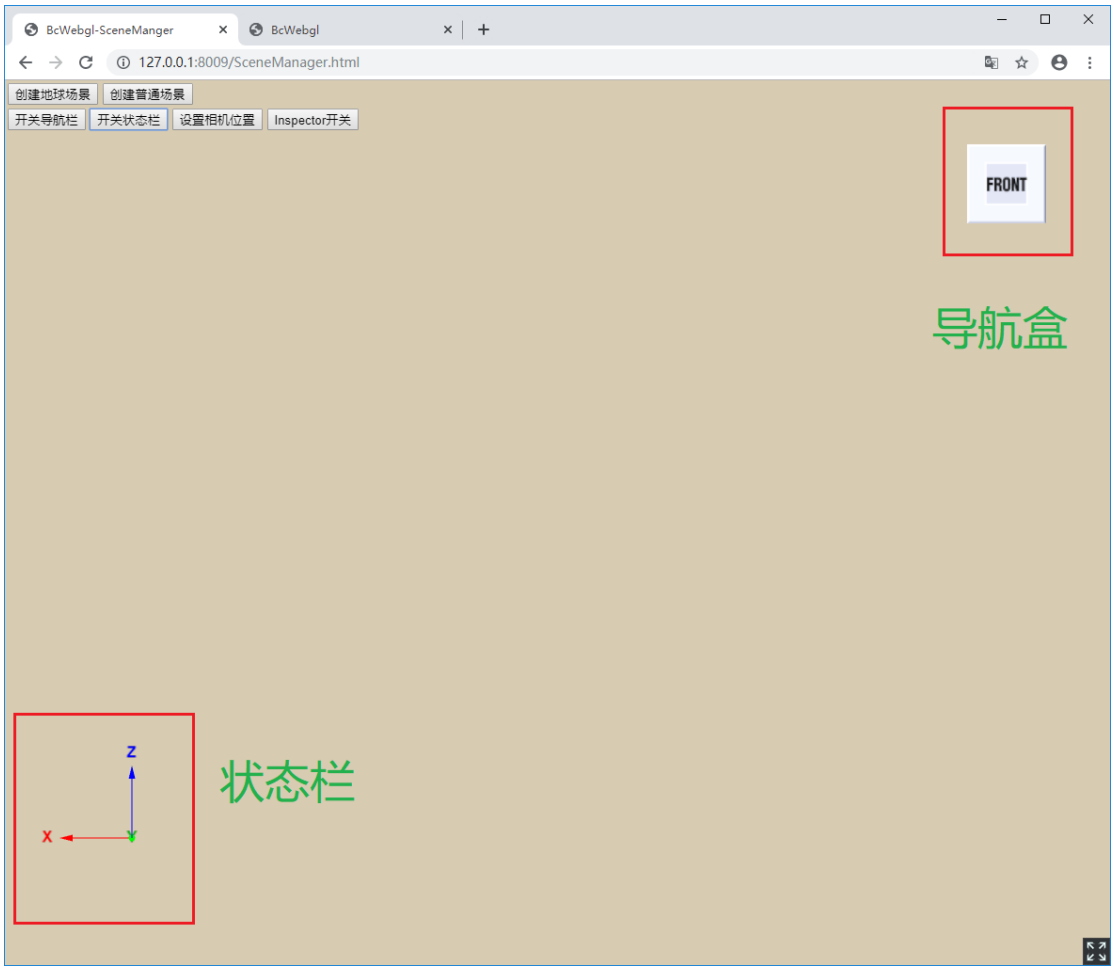
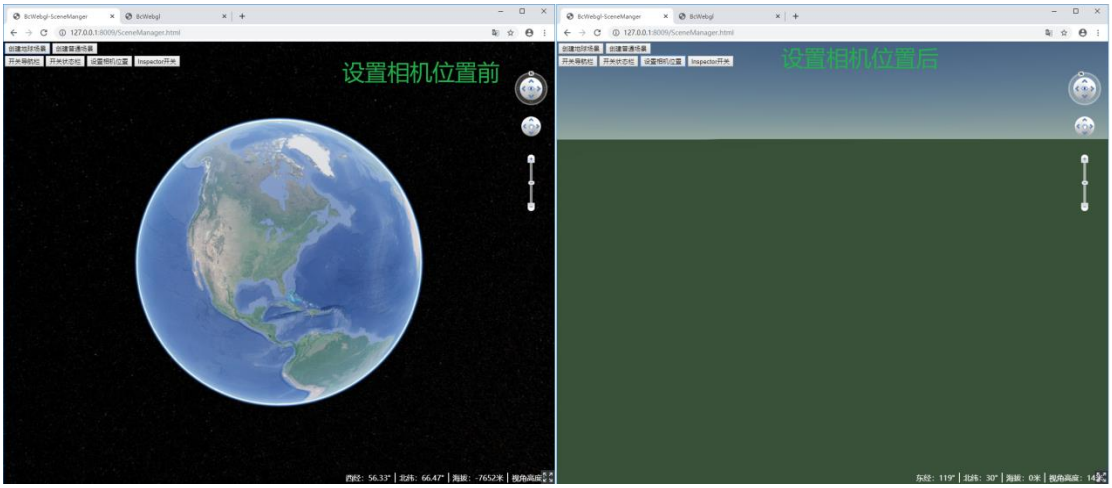


图 普通场景

点击“设置相机位置”，相机会飞行到代码指定的相机位置，如下：



UGX 和 UGT 数据加载

第一步 创建一个工程

1.在 webgl 工程发布 www 目录下创建一个 ModelManger.html 文件，参考创建一个 webgl 工程，
在 head 头中设置 html 页面显示样式并导入 Webgl 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl- ModelManger </title>
    <style>
        html,
        body {
            height: 100%;
            width: 100%;
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

2.在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，如下：

```
<div id = "engineContainer"></div>
```

```
<script type="text/javascript" src="./BcEngineX.js"></script>
```

3.在<boday>中再添加一个<script>标签，并添加创建地球场景的方法 createEarthScene()和加载 UGX/UGT 数据方法 loadData()，移除模型图层 removeModel()，图层显隐 setModelVisibility()，添加代码如下：

Function	Content
createEarthScene ()	创建地球场景
loadData ()	添加一份数据
removeModel ()	移除模型
setModelVisibility ()	设置模型显隐

```
<script>
    var scene;//场景
    window.scene = scene;
    //创建地球场景,type:0 地球场景，type:1 普通场景
    function createEarthScene() {
        if(scene != null)
        {
            scene.destroy();
        }
        scene = BcEngineX.createBcScene("engineContainer", {type:0});
    }
     //添加一份ugx数据
     function loadData (){
        modelLayer = scene.modelLayerManager.addLayer( {
                    "name": "龙星ugx",
                      "description": "龙星ugx",
                    "version": "2.0",
                    "url": " http://127.0.0.1:8009/longxing/longxing.bcx",
                    "position": {
                       "x": 119.93759,
                       "y": 30.139317,
                       "z": 10.0
                    }
                },);
        //模型索引文件加载完成
        if(modelLayer.modelReadyEvent)
        {
            modelLayer.loadedEvent.addEventListener(function (layer) {
              console.log(layer);
          });
        }
        // 模型加载后事件
        if (modelLayer.modelReadyEvent) {
```

```
                modelLayer.modelReadyEvent.addEventListener(function (layer) {
                    //定位模型
                    scene.fly2front(modelLayer);
                });
            }
        }
        //移除模型
        function removeModel(){
            scene.modelLayerManager.remove(modelLayer);
        }
        //图层显隐
        function setModelVisibility() {
            modelLayer.isShow = !modelLayer.isShow;
        }
</script>
```

4.在< boday >标签中添加一个<table>标签并添加<button>按钮用于调用 function 方法，如下：

Button	Text
createScene	创建地球场景
loadData	添加模型
removeModel	移除模型
setModelVisibility	图层显隐

```
<table>
    <tbody>
    <tr>
        <td>
            <button id = "createScene" onclick="createScene()">创建地球场景</button>
        </td>
    </tr>
    <tr>
        <td>
            <button id="loadData" onclick="loadData()">添加模型</button>
            <button id="setModelVisibility" onclick="setModelVisibility()">图层显隐</button>
            <button id="removeModel" onclick="removeModel()">移除模型</button>
        </td>
    </tr>
    </tbody>
</table>
```

第二步 运行

在浏览器端打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/ModelManager.html，运行，webgl-ModelManager 示例 demo

就创建成功，点击“创建地球场景”创建一个 wgs84 地球场景，按钮再点击“添加模型”，则会自动定位到模型加载的坐标位置（x:119.93759, y: 30.139317,z: 10.0），点击“图层显隐”可以隐藏模型图层，“移除模型”会从场景移除加载的模型，如下：

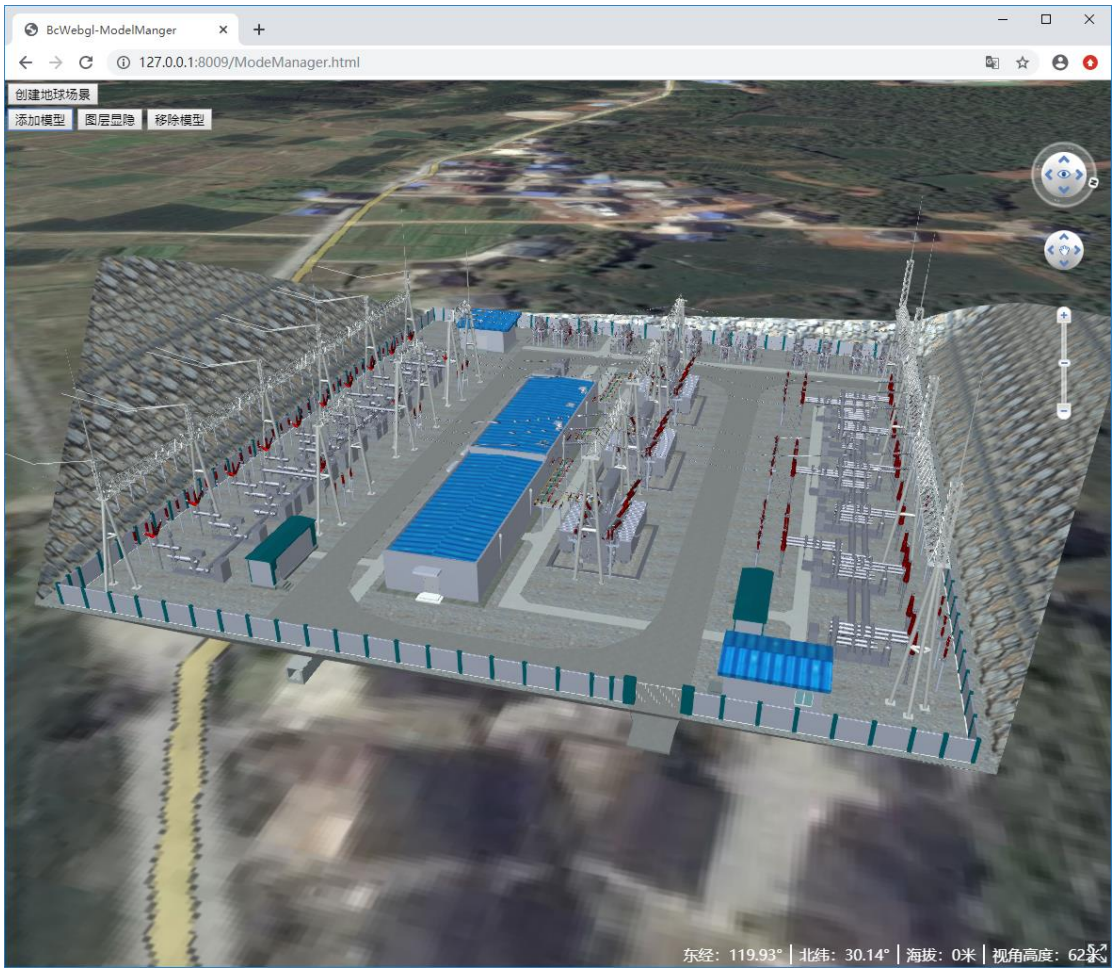


图 加载模型数据

飞行演示

第一步 创建一个工程

1.在 webgl 工程发布 www 目录下创建一个 flyManger.html 文件，参考创建一个 webgl 工程，在 head 头中设置 html 页面显示样式并导入 Webgl 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl- flyManger</title>
    <style>
        html,
        body {
            height: 100%;
```

```
            width: 100%;
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

2.在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，如下：

```
<div id = "engineContainer"></div>
<script type="text/javascript" src="./BcEngineX.js"></script>
```

3.在<boday>中再添加一个<script>标签，并添加创建飞行站点方法 setFlyLocationPath ()，添加代码如下：

Function	Content
setFlyLocationPath ()	飞行

```
<script>
    //影像图
    var imageLayer;
    window.scene = scene;
    //创建一个地球场景
    var scene = BcEngineX.createBcScene("engineContainer", {type:0});
    //添加一份影像
    imageLayer = scene.imageLayerManager.addImageLayer("http://192.168.2.85:8099/Data/ImageLayer/ZheJiang_FuYang_TMS/ZheJiang_FuYang_TMS.bci");
    //飞行倒指定位置
```

```
    scene.fly2degrees([119.95,30.12,800])
    //设置飞行站点
    function setFlyLocationPath() {
        const routes = [
            {
              //站点经纬度坐标
              position:[119.92,30.15,280],
              //以当前速度方向和水平向右的右手坐标系计算的俯仰角、偏转角、视点偏移距离
              hprang:[240,30,100],
            },
            {
              //站点经纬度坐标
              position:[120.0,30.08,280],
              //以当前速度方向和水平向右的右手坐标系计算的俯仰角、偏转角、视点偏移距离
              hprang:[240,30,100],
            }
          ];
        //设置飞行路线及视角
         scene.flyManager.setFlyLocationPath(routes,15,true);
         //显示飞行轨迹
         scene.flyManager.showPath = true;
         //显示路线站点
         scene.flyManager.showPoint  = true;
        //是否显示模型
         scene.flyManager.setFlyModelShow  = true;
         //显示模型路径
         scene.flyManager.setModel = "http://127.0.0.1:8009/Assets/models/CesiumAir/Cesium_Air.gltf";
         //锁定视角
        scene.flyManager.setFlyViewMode(0);
         //设置飞行结束后动作
        scene.flyManager.setFlyStopAction(1);
         scene.flyManager.play();
        }
</script>
```

4.在< boday >标签中添加一个<table>标签并添加<button>按钮用于调用 function 方法，如下：

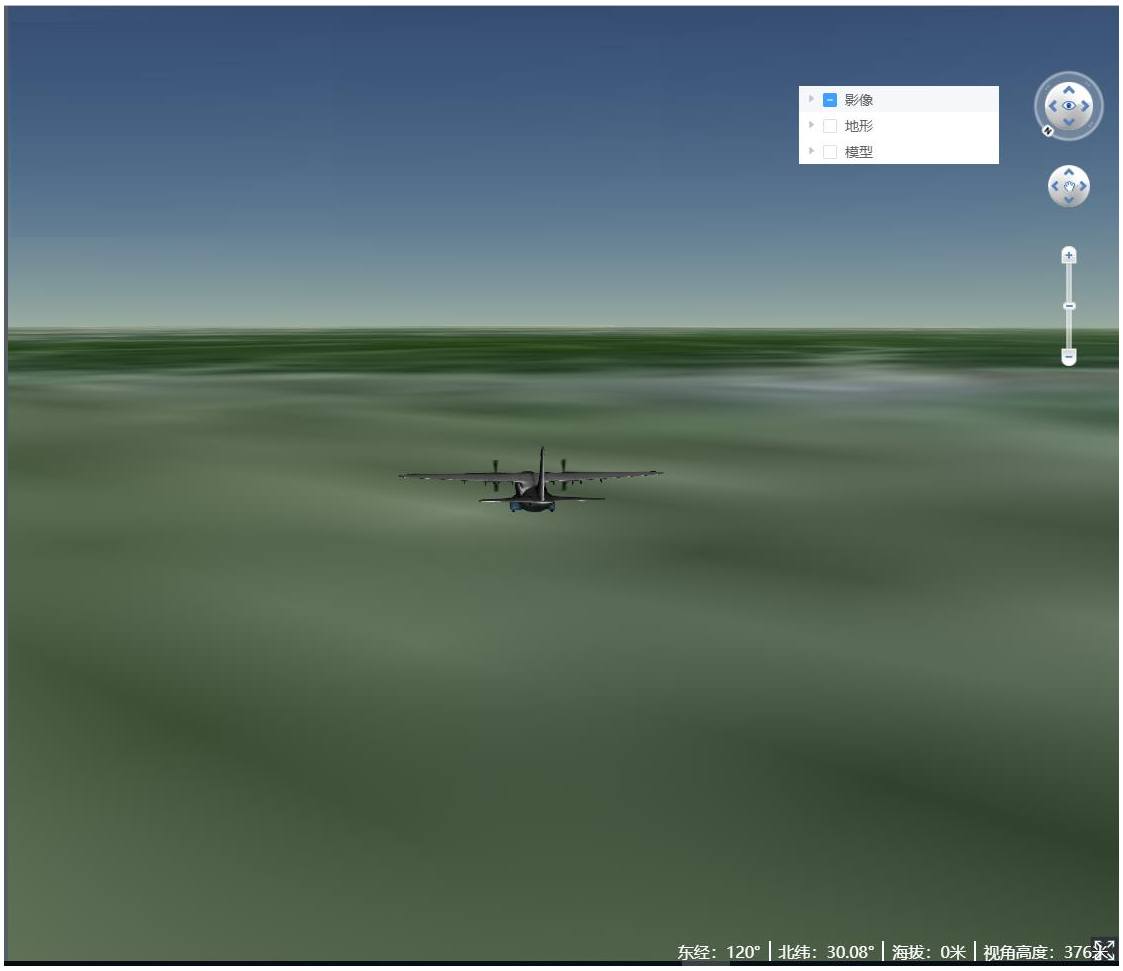
Button	**Text**
setFlyLocationPath	飞行

```
<table>
    <tbody>
    <tr>
```

```
        <td>
            <button id = " setFlyLocationPath " onclick=" setFlyLocationPath ()">飞行</button>
        </td>
    </tr>
    </tbody>
</table>
```

第二步 运行

在浏览器端打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/flyManager.html，运行，webgl-flyManager 示例 demo 就创建成功，如下：



标签

第一步 创建一个工程

1.在 webgl 工程发布 www 目录下创建一个 labelManager.html 文件，参考创建一个 webgl

工程，
在 head 头中设置 html 页面显示样式并导入 Webgl 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl- labelManager </title>
    <style>
        html,
        body {
            height: 100%;
            width: 100%;
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

2.在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，如下：

```
<div id = "engineContainer"></div>
<script type="text/javascript" src="./BcEngineX.js"></script>
```

3.在<boday>中再添加一个<script>标签，并添加添加文字标签 appendDynamicLabelPlot ()、添加图片标签 appendDynamicBillboardPlot ()、添加屏幕图片 appendScreenImage ()、添加屏幕文字 appendScreenText ()、添加球体 appendDynamicSphere ()，删除标签 removeDynamicPlot（）等方法，添加代码如下：

Function	Content

appendDynamicLabelPlot ()	添加文字标签
appendDynamicBillboardPlot ()	添加图片标签
appendScreenImage ()	添加屏幕图片
appendScreenText ()	添加屏幕文字
appendDynamicSphere ()	添加球体
removeDynamicPlot ()	删除标签

```
<script>
    //影像图
    var imageLayer;
    //标签
    var label;
    window.scene = scene;
    //创建一个地球场景
    var scene = BcEngineX.createBcScene("engineContainer", {type:0});
    //添加一份影像
    imageLayer = scene.imageLayerManager.addImageLayer("http://192.168.2.85:8099/Data/ImageLayer/ZheJiang_FuYang_TMS/ZheJiang_FuYang_TMS.bci");
    //添加一份地形
    scene.terrainProviderManager.setTerrainProvider("http://192.168.2.85:8099/Data/TerrainLayer/zjfy");
    //飞行倒指定位置
    scene.fly2degrees([119.95,30.12,800])
    //距离测量
    function DistanceMeasure() {
        result =scene.measureManager.BeginDistanceMeasure();
    }
    //添加文字标签
    function appendDynamicLabelPlot() {
        label = scene.plotManager.getLayer().appendDynamicLabelPlot({text:"博超 webgl 文字标签"});
    }
    //添加图片标签
    function appendDynamicBillboardPlot() {
        label = scene.plotManager.getLayer().appendDynamicBillboardPlot({image:"http://127.0.0.1:8009/placemark32.png"});
    }
    //添加屏幕图片
    function appendScreenImage() {
        label = scene.plotManager.getLayer().appendScreenImage("http://127.0.0.1:8009/compass.png", 100, 100);
    }
    //添加屏幕文字
    function appendScreenText() {
```

```
        label = scene.plotManager.getLayer().appendScreenText("博超webgl屏幕文字标签", undefined, undefined, {font:"18px sans-serif", fillStyle:"red"});
    }
    //添加球体
    function appendDynamicSphere() {
        label = scene.plotManager.getLayer().appendDynamicSphere({pixelSize:"30",position:undefined ,color:"red"});
    }
    //删除标签
    function removeDynamicPlot () {
        scene.plotManager.getLayer().removeDynamicPlot();
    }
</script>
```

4.在< boday >标签中添加一个<table>标签并添加<button>按钮用于调用 function 方法，如下：

Buttons	Text
appendDynamicLabelPlot	添加文字标签
appendDynamicBillboardPlot	添加图片标签
appendScreenImage	添加屏幕图片
appendScreenText	添加屏幕文字
appendDynamicSphere	添加球体
removeDynamicPlot	删除标签

```
<table>
    <tbody>
    <tr>
        <td>
            <button onclick ="appendDynamicLabelPlot()">添加文字标签</button>
            <button onclick ="appendDynamicBillboardPlot()">添加图片标签</button>
            <button onclick ="appendScreenImage()">添加屏幕图片标签</button>
            <button onclick ="appendScreenText()">添加屏幕文字</button>
            <button onclick ="appendDynamicSphere()">添加球体</button>
            <button onclick ="setShowPlot()">标签显隐</button>
            <button onclick ="removeDynamicPlot()">删除标签</button>
        </td>
    </tr>
    </tbody>
</table>
```

第二步 运行

在浏览器端打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/labelManager.html，运行，示例 demo 就启动成功，即可开始进行测

量，如下：

点击“删除标签”可以删除指定 guid 的标签。

批注

第一步 创建一个工程

1.在 webgl 工程发布 www 目录下创建一个 plotManager.html 文件，参考创建一个 webgl 工程，

在 head 头中设置 html 页面显示样式并导入 Webgl 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl- plotManager</title>
    <style>
        html,
        body {
```

```
            height: 100%;
            width: 100%;
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

2.在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，如下：

```
<div id = "engineContainer"></div>
<script type="text/javascript" src="./BcEngineX.js"></script>
```

3.在<boday>中再添加一个<script>标签，并添加开始批注 beginDynamicPlot ()、直线批注 appendDynamicLinePlot ()、云线批注 appendDynamicCloudPlot ()、等方法，添加代码如下：

Function	Content
beginDynamicPlot ()	开始批注
appendDynamicLinePlot ()	直线批注
appendDynamicCloudPlot ()	云线批注
eraseCurrentDynamicPlot ()	擦除当前批注
finishDynamicPlot()	结束批注
destroy()	删除批注

```
<script>
    //影像图
    var imageLayer;
    //批注
```

```
    var plot;
    window.scene = scene;
    window.plot = scene.plotManager.getLayer();;
    //存储批注 guid
    var plotArr=[];
   var deletePlot=[];
    //创建一个地球场景
    var scene = BcEngineX.createBcScene("engineContainer", {type:0});
    //添加一份影像
   imageLayer = scene.imageLayerManager.addImageLayer("http://192.168.2.85:8099/Data/ImageLayer/ZheJiang_FuYang_TMS/ZheJiang_FuYang_TMS.bci");
    //飞行倒指定位置
   scene.fly2degrees([119.95,30.12,800])
    //开始批注
   function beginDynamicPlot() {
        plot.beginDynamicPlot()
   }
    //直线批注
   function appendDynamicLinePlot() {
        plot.appendDynamicLinePlot()
   }
     //云线批注
    function appendDynamicCloudPlot() {
        plot.appendDynamicCloudPlot()
     }
       //擦除当前批注
    function eraseCurrentDynamicPlot() {
            plot.eraseCurrentDynamicPlot();
        }
       //结束批注
    function finishDynamicPlot() {
          plot.finishDynamicPlot()
      }
          //销毁所有批注
     function destroy() {
            plot.destroy();
        }
</script>
```

4.在< boday >标签中添加一个<table>标签并添加<button>按钮用于调用 function 方法，如下：

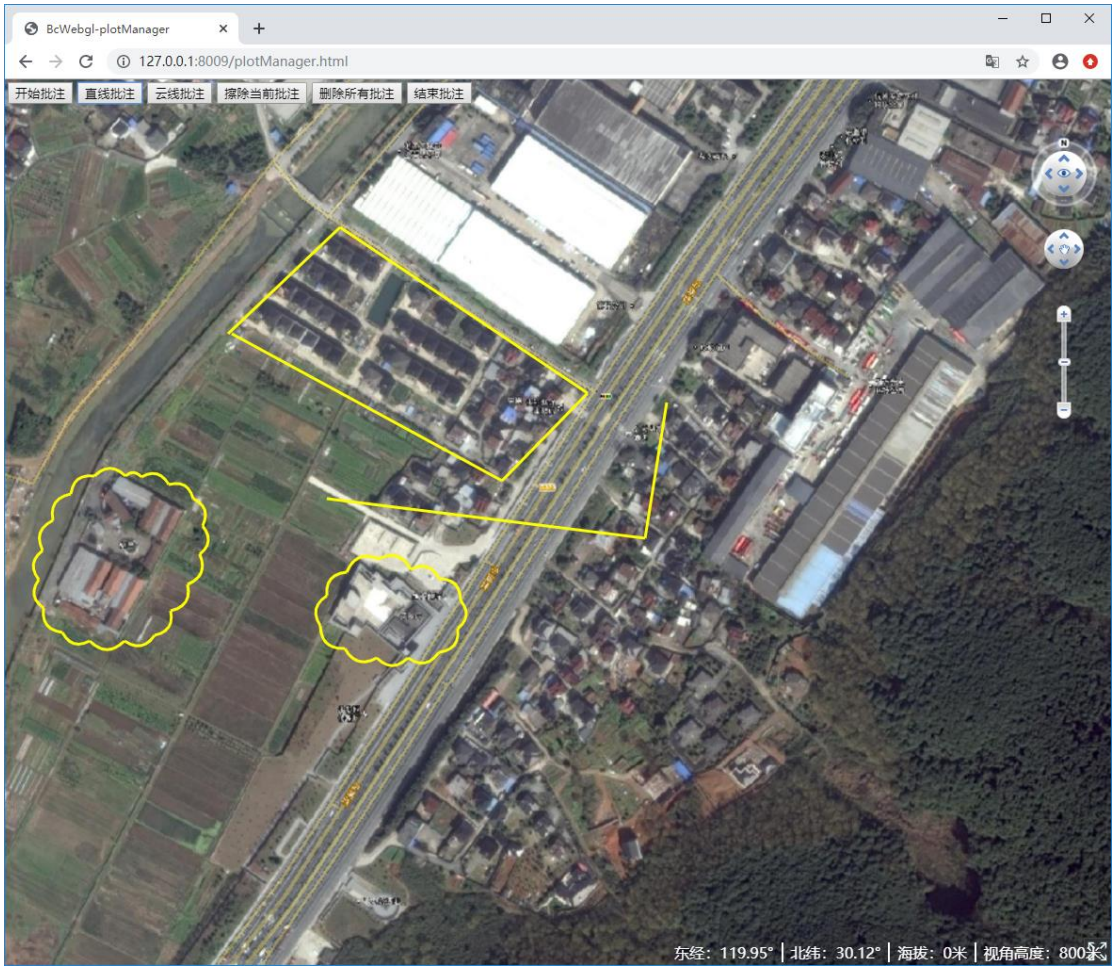
Buttons	**Text**
beginDynamicPlot ()	开始批注

appendDynamicLinePlot ()	直线批注
appendDynamicCloudPlot ()	云线批注
eraseCurrentDynamicPlot ()	擦除当前批注
finishDynamicPlot()	结束批注
destroy()	删除所有批注

```
<table>
    <tbody>
    <tr>
        <td>
            <button onclick ="beginDynamicPlot()">开始批注</button>
            <button onclick ="appendDynamicLinePlot()">直线批注</button>
            <button onclick ="appendDynamicCloudPlot()">云线批注</button>
           <button onclick ="eraseCurrentDynamicPlot()">擦除当前批注</button>
            <button onclick =" destroy()">删除所有批注</button>
            <button onclick ="finishDynamicPlot()">结束批注</button>
        </td>
    </tr>
    </tbody>
</table>
```

第二步 运行

在浏览器端打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/plotManager.html，运行，示例 demo 就启动成功，即可开始进行测量，如下：



在结束批注前可以点击“擦除当前批注”来擦除目前绘制的批注，结束批注后可以点击“删除所有批注”将批注全部删除。

三维测量

第一步 创建一个工程

1.在 webgl 工程发布 www 目录下创建一个 measureManager.html 文件，参考创建一个 webgl 工程，

在 head 头中设置 html 页面显示样式并导入 Webgl 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl-MeasureManger</title>
    <style>
        html,
        body {
```

```
            height: 100%;
            width: 100%;
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

2.在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，如下：

```
<div id = "engineContainer"></div>
<script type="text/javascript" src="./BcEngineX.js"></script>
```

3.在<boday>中再添加一个<script>标签，并添加距离测量 DistanceMeasure ()、面积测量 AreaMeasure ()、角度测量 AngleMeasure ()、实体测量 EntityMeasure ()、清除测量结果 clearResult ()，结束测量 end ()等方法，添加代码如下：

Function	Content
DistanceMeasure ()	距离测量
AreaMeasure ()	面积测量
AngleMeasure ()	角度测量
EntityMeasure ()	实体测量
clearResult ()	清除测量结果
end ()	结束测量

```
<script>
    //影像图
    var imageLayer;
```

```
//测量结果
var result;
window.scene = scene;
//创建一个地球场景
var scene = BcEngineX.createBcScene("engineContainer", {type:0});
//添加一份影像
imageLayer = scene.imageLayerManager.addImageLayer("http://192.168.2.85:8099/Data/ImageLayer/ZheJiang_FuYang_TMS/ZheJiang_FuYang_TMS.bci");
//添加一份地形
scene.terrainProviderManager.setTerrainProvider("http://192.168.2.85:8099/Data/TerrainLayer/zjfy");
//开启拾取模式
scene.setPickEnable(true);
//添加一份模型
modelLayer = scene.modelLayerManager.addLayer( {
        "name": "龙星 ugx",
        "description": "龙星 ugx",
        "version": "2.0",
        "url": "/longxing/longxing.bcx",
        "position": {
          "x": 119.95,
          "y": 30.12,
          "z": 50.0
        }
},);
//飞行倒指定位置
scene.fly2degrees([119.95,30.12,800])
//距离测量
function DistanceMeasure() {
    result =scene.measureManager.BeginDistanceMeasure();
}
//面积测量
function AreaMeasure() {
    result = scene.measureManager.BeginAreaMeasure();
}
//角度测量
function AngleMeasure()  {
    scene.measureManager.BeginAngleMeasure();
}
//实体测量
function EntityMeasure() {
     scene.measureManager.BeginEntityMeasure();
}
//清除测量结果
```

```
    function clearResult() {
        scene.measureManager.clear();
    }
    //结束测量
    function end() {
        scene.measureManager.end();
    }
</script>
```

4.在< boday >标签中添加一个<table>标签并添加<button>按钮用于调用 function 方法，如下：

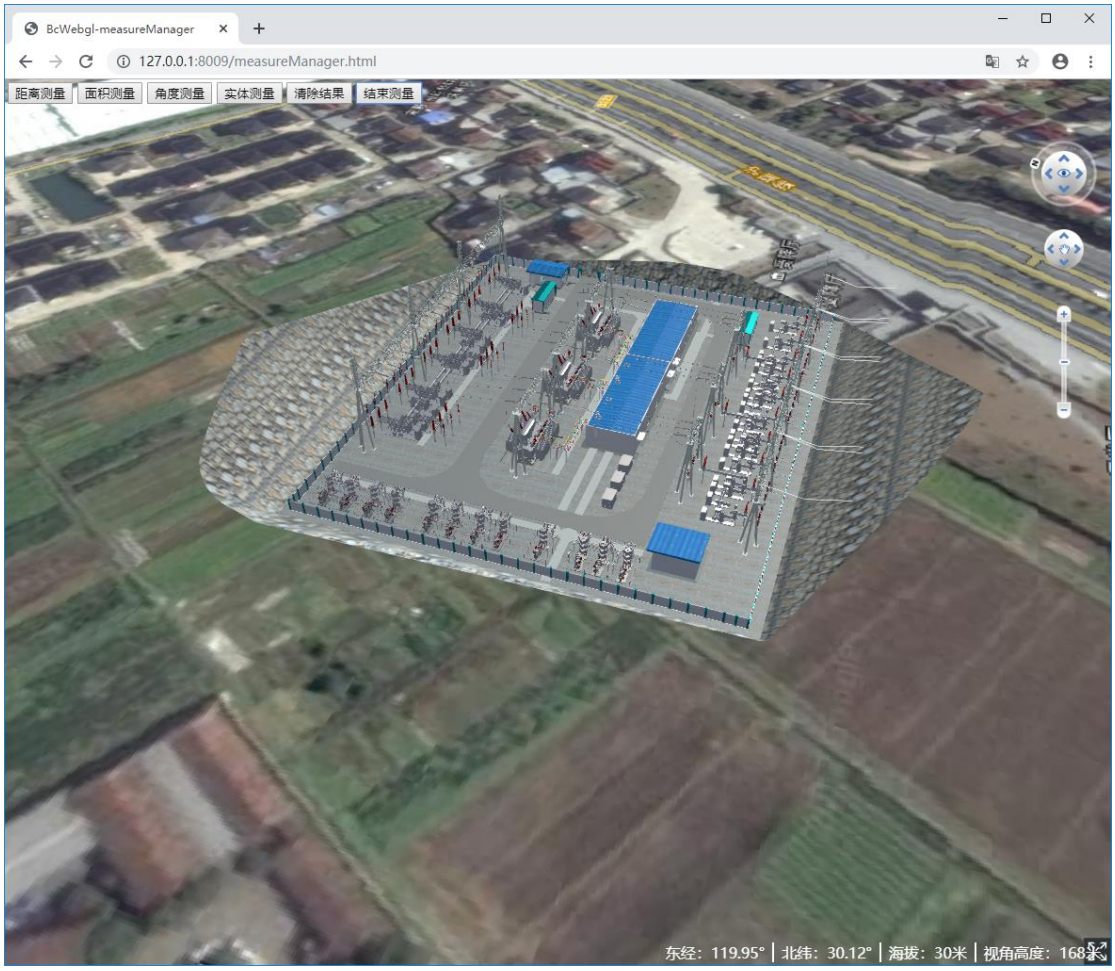
Buttons	Text
DistanceMeasure	距离测量
AreaMeasure	面积测量
AngleMeasure	角度测量
EntityMeasure	实体测量
clearResult	清除测量结果
end	结束测量

```
<table>
    <tbody>
    <tr>
        <td>
            <button id="DistanceMeasure" onclick="DistanceMeasure()">距离测量</button>
            <button id ="AreaMeasure" onclick ="AreaMeasure()">面积测量</button>
            <button id ="AngleMeasure" onclick ="AngleMeasure()">角度测量</button>
            <button id ="EntityMeasure" onclick ="EntityMeasure()">实体测量</button>
              <button id ="clearResult" onclick ="clearResult()">清除结果</button>
              <button id ="end" onclick ="end()">结束测量</button>
        </td>
    </tr>
    </tbody>
</table>
```

第二步 运行

在浏览器端打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/MeasureManager.html，运行，示例 demo 就启动成功，即可开始进行测量，如下：

点击“清除测量结果”可以清除所有测量结果显示，点击“结束测量”则结束三维测量，如下：



三维空间分析

第一步 创建一个工程

1.在 webgl 工程发布 www 目录下创建一个 3DAnalysisManager.html 文件，参考创建一个 webgl 工程，
在 head 头中设置 html 页面显示样式并导入 Webgl 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl-3DAnalysisManager</title>
    <style>
        html,
        body {
            height: 100%;
            width: 100%;
```

```
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

2.在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，如下：

```
<div id = "engineContainer"></div>
<script type="text/javascript" src="./BcEngineX.js"></script>
```

3.在<boday>中再添加一个<script>标签，并添加通视分析 VisibleAnalysis ()、可视域分析 ViewshedAnalysis ()、淹没分析 SubmergenceAnalysis ()等方法，添加代码如下：

Function	Content
VisibleAnalysis ()	通视分析
ViewshedAnalysis ()	可视域分析
SubmergenceAnalysis ()	淹没分析

```
<script>
    //影像图
    var imageLayer;
    window.scene = scene;
    //创建一个地球场景
    var scene = BcEngineX.createBcScene("engineContainer", {type:0});
    //添加一份影像
    imageLayer = scene.imageLayerManager.addImageLayer("http://192.168.2.85:8099/Data/ImageLayer/ZheJiang_FuYang_TMS/ZheJiang_FuYang_TMS.bci");
```

```
    //添加一份地形
    scene.terrainProviderManager.setTerrainProvider("http://192.168.2.85:8099/Data/TerrainLay
er/zjfy");
    //飞行定位倒指定位置
   scene.fly2degrees([119.95,30.12,800])

    //通视分析
    function VisibleAnalysis() {
        var m_visibleAnalysis = new BcEngineX.BcVisibleAnalysis(scene);
        m_visibleAnalysis.start(); //开始通视分析
    }
    //可视域分析（当前只支持地形）
    function ViewshedAnalysis() {
        var m_viewshedAnalysis = new BcEngineX.BcViewshedAnalysis(scene);
        m_viewshedAnalysis.start(); //开始可视域分析
    }
    //淹没分析
    function SubmergenceAnalysis() {
        var m_submergenceAnalysis = new BcEngineX.BcSubmergenceAnalysis(scene);
        //start(height：淹没高度, minHeight：淹没最低高度)
        m_submergenceAnalysis.start(200,20);
    }
</script>
```

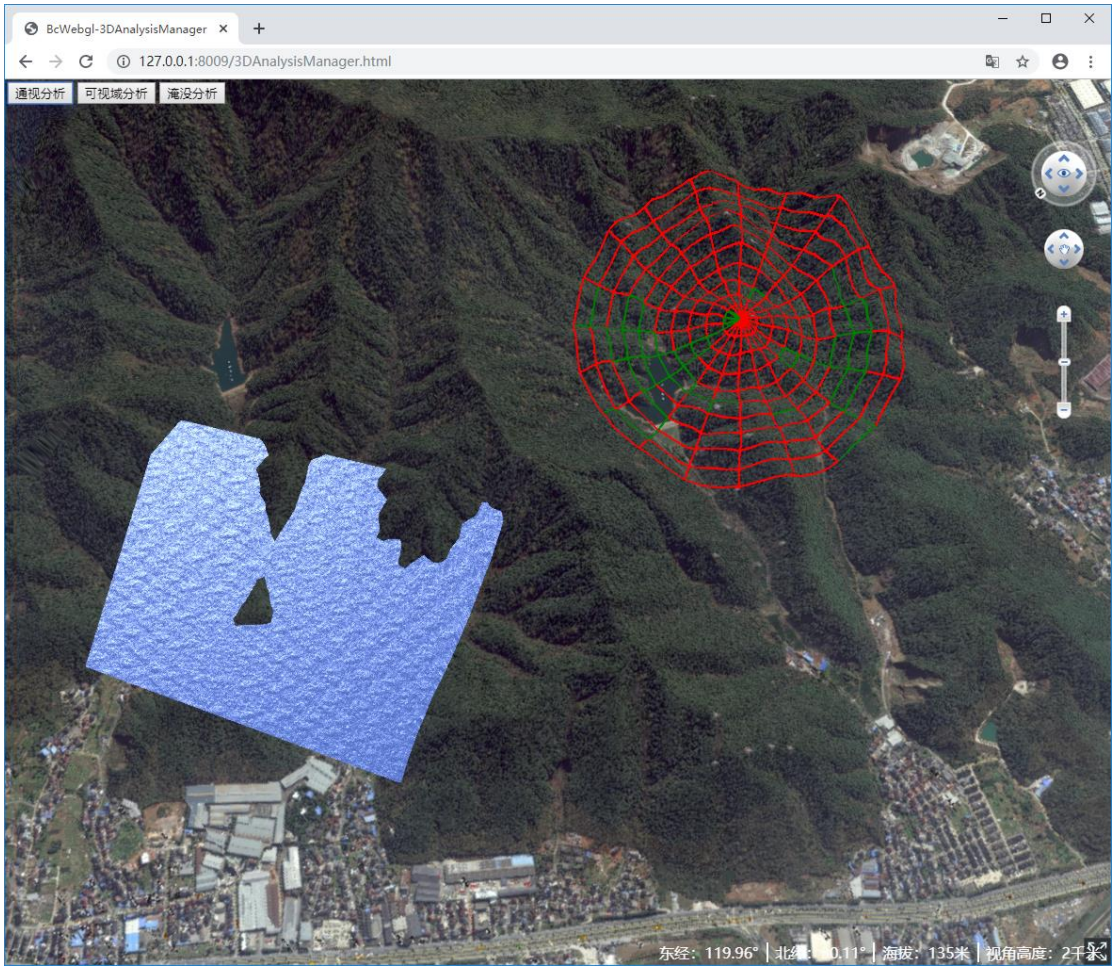
4.在< boday >标签中添加一个<table>标签并添加<button>按钮用于调用 function 方法，如下：

Buttons	Text
VisibleAnalysis	通视分析
ViewshedAnalysis	可视域分析
SubmergenceAnalysis	淹没分析

```
<table>
    <tbody>
    <tr>
        <td>
          <button id="VisibleAnalysis" onclick ="VisibleAnalysis()">通视分析</button>
          <button id="ViewshedAnalysis" onclick="ViewshedAnalysis()">可视域分析</button>
          <button id="SubmergenceAnalysis" onclick="SubmergenceAnalysis()">淹没分析</bu
    tton>
        </td>
    </tr>
    </tbody>
</table>
```

第二步 运行

在浏览器端打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/3DAnalysisManager.html，运行，示例 demo 就启动成功，即可开始进行三维分析，如下：



矢量加载

第一步 创建一个工程

1.在 webgl 工程发布 www 目录下创建一个 VecterLayerManager.html 文件，参考创建一个 webgl 工程，

在 head 头中设置 html 页面显示样式并导入 Webgl 包中 css 资源文件 widgets.css，插入代码如下：

```
<head>
    <meta charset="UTF-8">
    <title>BcWebgl- VecterLayerManager</title>
```

```
    <style>
        html,
        body {
            height: 100%;
            width: 100%;
            margin: 0px;
        }
    </style>
    <style>
        @import url(./Widgets/widgets.css);
        html,
        body,
        #engineContainer {
            position: absolute;
            left: 0px;
            top: 0px;
            z-index: -1;
            background: black;
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }
    </style>
</head>
```

2.在<boday>中使用 head 中创建的 engineContainer 容器，导入的导入 webgl 引擎库 BcEngineX.js，如下：

```
<div id = "engineContainer"></div>
<script type="text/javascript" src="./BcEngineX.js"></script>
```

3.在<boday>中再添加一个<script>标签，并添加添加 shp 矢量文件的方法 loadShpFile ()和加载 shp 回调 cb ()，移除矢量图层 removeDataSource ()；添加代码如下：

Function	Content
loadShpFile ()	添加 shp 矢量文件
cb ()	加载 shp 回调
removeDataSource ()	移除矢量图层

```
<script>
    //场景
    var scene;
    window.scene = scene;
```

```
    //矢量图层对象
    var dataSource;
     //创建一个地球场景
     var scene = BcEngineX.createBcScene("engineContainer", {type:0});
     //加载shp矢量文件
     function loadShpFile(){
          scene._vecterLayerManager.loadShapefile({
               shp:"http://192.168.2.85:8099/data/vecterLayer/prov_region.shp",
               extrudedHeight: 10000,//拉高
               extrudedHeightField: "Population",//用于拉高的字段
               extrudedHeightScale: 0.002,//拉高系数
               outline: true,//是否显示轮廓
               outlineColor: [1.0, 0.0, 0.0, 1],//轮廓线颜色
               color: [1.0, 0.0, 0.0, 1],//几何体颜色
               styleField: "name",//分类字段
               colorTable: {
                    "Massachusetts":[1.0,0.0,0.0,1.0],
                    "Minnesota":[1.0,1.0,0.0,1.0],
                    "Montana":[1.0,0.0,1.0,1.0],
               }//颜色表
          },
          cb);
     }
     //加载shp回调
     function cb(isSucceed,data){
          if(isSucceed){
               dataSource = data;
           }else{
                console.error(data);
          }
          scene._vecterLayerManager.loadShapefile("http://192.168.2.85:8099/data/vecterLayer/
          prov_region.shp",cb)
     }
     //移除矢量图层
     function removeDataSource(){
          scene._vecterLayerManager.removeDataSource(dataSource);
     }
</script>
```

4.在< boday >标签中添加一个<table>标签并添加<button>按钮用于调用 function 方法，如下：

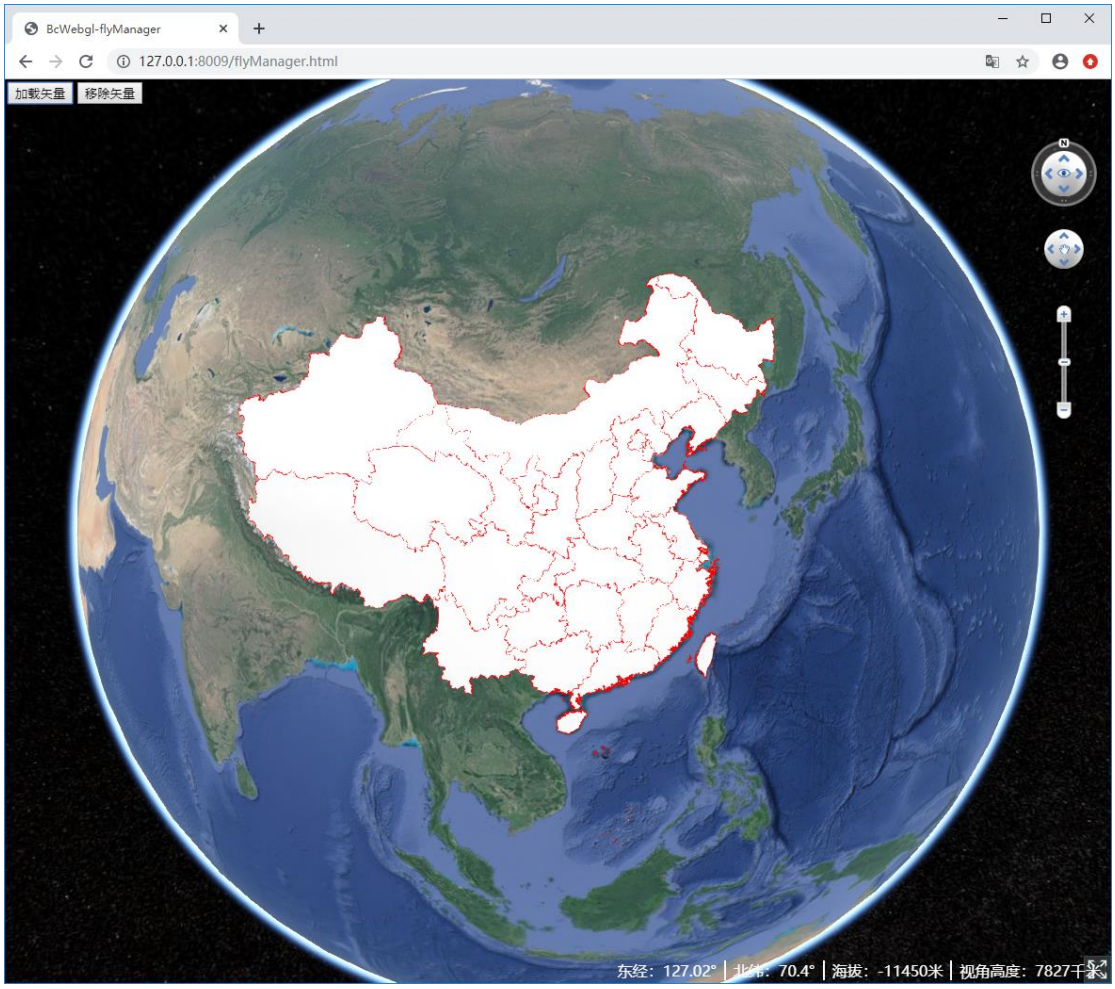
Buttons	**Text**
loadShpFile ()	加载矢量

removeDataSource ()	移除矢量

```
<table>
    <tbody>
        <tr>
            <td>
            <button id = " loadShpFile" onclick=" loadShpFile ()">加载矢量</button>
            <button id =" removeDataSource" onclick =" removeDataSource ()">移除矢量
            </button>
              </td>
         </tr>
     </tbody>
</table>
```

第二步 运行

在浏览器端打开 Chrome 浏览器输入展示例子 URL 地址：http://127.0.0.1:8009/VecterLayerManager.html，运行，示例 demo 就启动成功，点击“加载矢量”加载指定发布的矢量 shp，点击“移除矢量”移除加载的矢量数据如下：



数据接入

WebGL 支持的数据

BcEngineX_WebGL 支持接入数据除地形数据外，所有接入数据与 BcEngineX 其他平台统一，即在 BcEngineX_Pro PC 端上进行生成的数可以直接发布加载到 BcEngineX_WebGL 上，做到了 GIS 数据和工程数据跨平台无缝融合使用。BcEngineX_WebGL 由于是基于 Cesium 内核目前仅支持.terrain（简称 TIN 地形）格式地形文件。

webgl 支持加载的数据

支持加载影像地形数据	**文件说明**
TMS 影像瓦片(*.bci)	tms（使用 OSGeo 规范存储 tms 格式的瓦片数据）
Terrain 地形	Cesium 支持的 STK World Terrain 地形图层
XYZ 影像(*.bci)	xyz 读取 Web xyz 瓦片格式的数据
支持加载的模型数据	
UGX 数据(*.bcx)	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。
UGTile 数据(*.bcm)	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。
点云(*.bcm)	将 las 点云数据生成以空间索引组织的 UGTile 格式缓存。
倾斜摄影(*.bcm)	将 osgb 倾斜摄影数据生成以空间索引组织的 UGTile 格式缓存。
管线(*.bcm)	根据二维点、线数据，采用自适应三维管点、管线参数化建模方法，构建三维管线。
电缆(*.bcm)	将存储有电缆数据信息的 json 文件生成以空间索引组织的 UGTile 格式缓存。
3DMax(*.bcm)	将 3Dmax 导出的模型生成以空间索引组织的 UGTile 格式缓存。
矢量建模(*.bcm)	矢量数据根据拉伸字段拉伸生并贴上纹理生成的成模型。
支持加载的矢量数据	
shp 矢量（*.shp）	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。
MVT(*.json)	以创建金字塔方式对矢量数据进行分层分块，然后使用.MVT/.PDF 格式存储数据描述信息,生成瓦片风格文件。

除上述支持数据外，BcEngineX_Webgl 还支持 Cesium 自身支持的数据格式如：GeoJson、MVT、Gltf、3D Tiles 等。

二维数据处理

BcEngineX_Webgl 支持加载的数据除 Terrain 地形和 Cesium 本身支持的数据类型外全部与 BcEngineX 其他平台共用，下面会介绍如何使用 BcEngineX_Pro 平台数据处理工具，以满足 BcEngineX_Webgl 各类型二维数据的处理工作。

地形格式转换

使用说明

将*.asc 或者*.dem 格式的地形数据转换为*.tif 格式地形数据。

操作步骤

1. 在“gis”选项卡的“gis 转换”中，打开“地形格式转换”工具；在“文件类型”中提供了 asc 和 dem 两种类型选择，这里选择需要转换的目标文件类型；然后单击鼠标选择“打开”，弹出打开数据对话框选择目标文件。

2. 点击“坐标系”按钮设置转换后目标坐标系，点击“确定”开始转换，将在输出目录下生成转换完成的 tif 文件如下图：

影像格式转换

使用说明

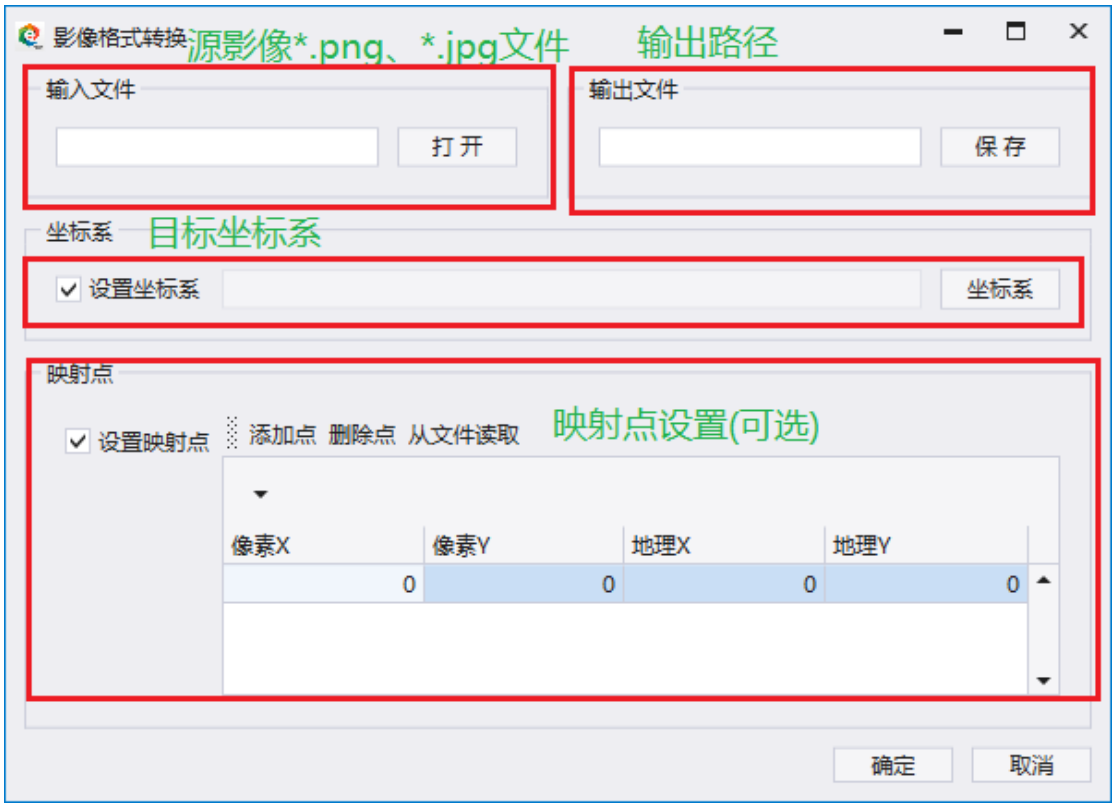
将*.jpg 或者*.png 格式的影像数据转换为 tif 格式影像数据。

操作步骤

1. 在“gis”选项卡的“gis 转换”中，打开“影像格式转换”工具；在”输入文件“这里点击”打开“按钮选择需要转换的 jpg 或者 png 目标文件；在”输出文件“中点击”保存“按钮选择保存路径和保存文件名。

2. 如果勾选设置”设置坐标系“，需要点击“坐标系”按钮设置转换后目标坐标系。

3. 如果勾选设置”映射点“，需要添加最少一个映射点，设置点的 x、y 像素值和地理 x、y 坐标值，设置完成点击“确定”开始转换，将在输出文件目录下生成转换完成的 tif 文件如下图：



地形切片

使用说明

将*.tif/格式地形数据转换为 BcEngineX_Webgl 支持的 terrain 地形瓦片数据。

BcEngineX_Webgl 由于是基于 Cesium 内核目前仅支持.terrain (简称 TIN 地形) 格式地形文件，所以地形切片与 BcEngineX 其他平台不共用，需要单独生成 terrain 地形的切片工具。此工具单独提供。

操作步骤

1. 打开“地形 GIS 地形处理”工具目录 Config 文件夹下 config.xml 文件配置好导入数据库 mongodb 数据库的相关信息，如下图。

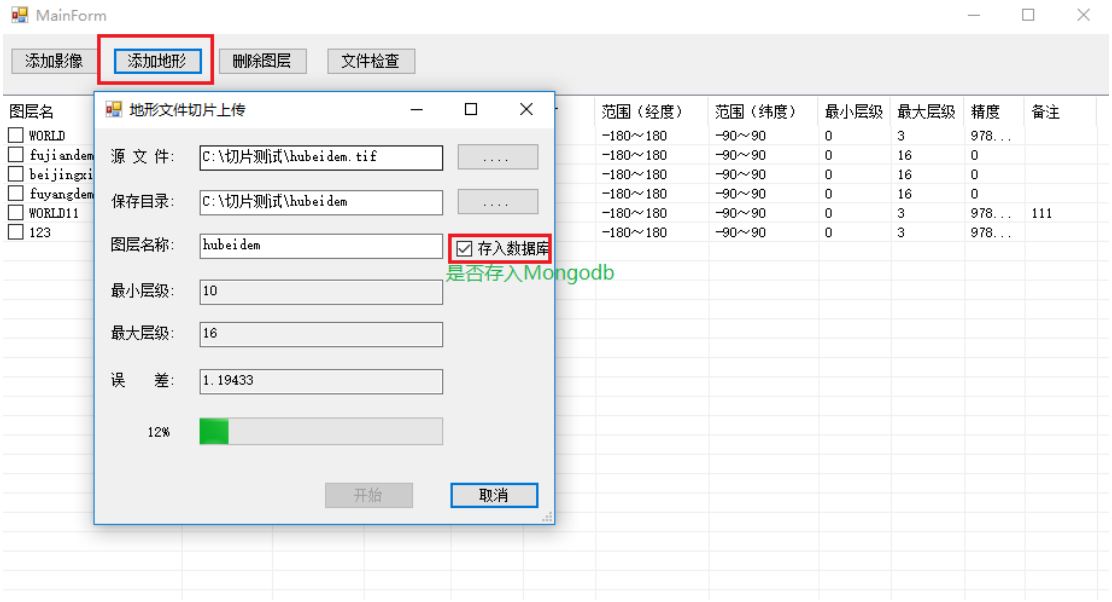
2. 打开“地形 GIS 地形处理”工具-“添加地形”，选择需要处理的 tif 地形数据（仅支持 WGS84 坐标系）

存入数据库：自定义选择是否切片完成是否存入 mongodb 数据库；

最大最小级别:设置即切片的地形显示最大最小级别，注意最小级别不能大于最大级别，这里默认已经读取了原始 tif 数据的最大最小层级数。

图层名：数据入库时在数据库下创建的表名，也是入库后 BcEngineX 加载该数据的图层名，所以图层名建议不要使用中文、空格、非法字符等

3. 点击“开始”即可开始生成 terraind 地形数据并在生成完成后导入 mongodb 数据库，如图：



影像切片

使用说明

将*.tif 格式影像数据转换为 BcEngineX 支持的 tms 缓存。

操作步骤

1．在“gis”选项卡的“二维数据处理”中，打开“影像切片”工具；点击”打开“按钮打开需要进行切片处理的 tif 影像数据，打开后在”文件属性“中将会显示数据的文件类型、坐标系等信息，如下图。

2. ”切片参数“设置中，各设置选项：

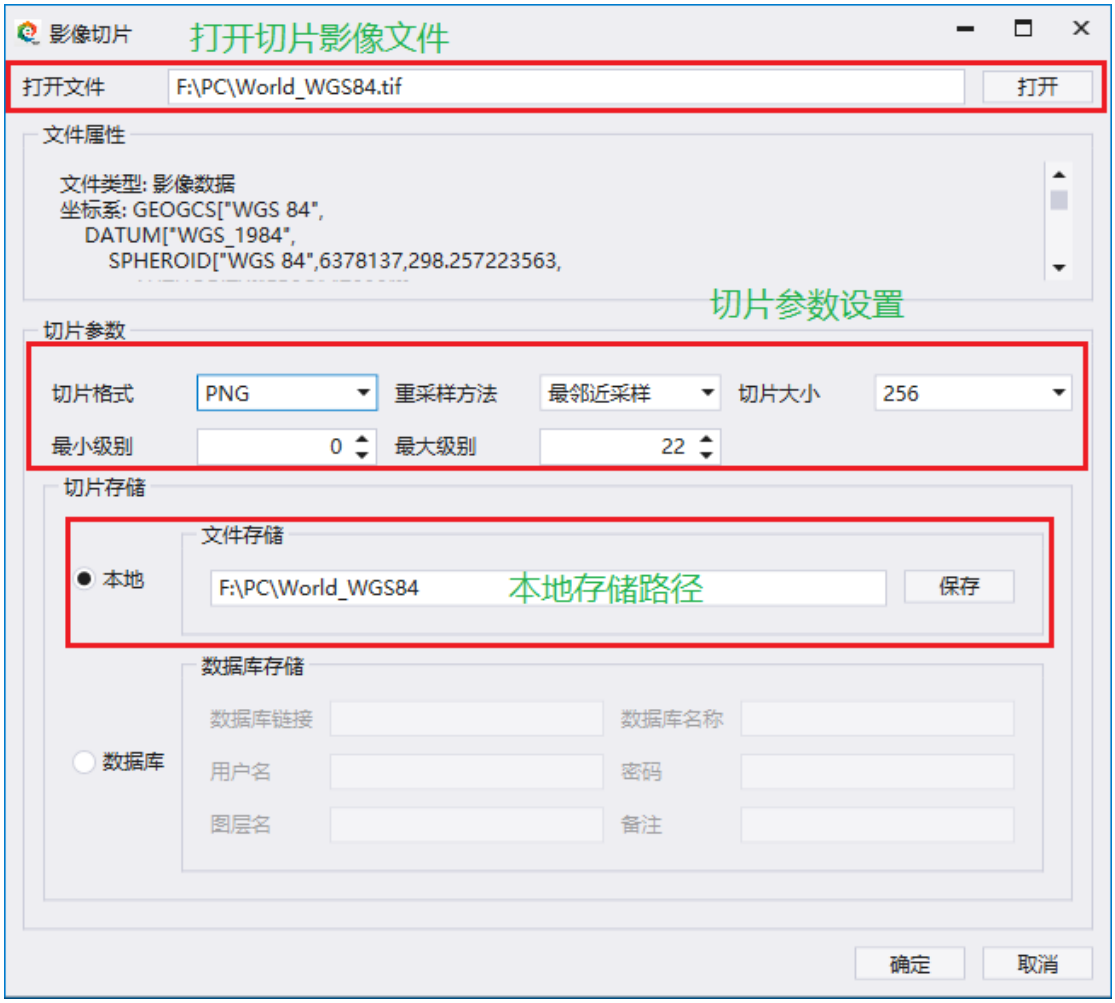
切片格式：影像切片支持生成”PNG”、”TIF“、”JPG”格式；

重采样：是是从高分辨率遥感影像中提取出低分辨率影像的过程，目前支持 5 中重采样方式；

切片大小：默认 256（即切片后的 tif 尺寸为 256*256），还支持 128*128 和 512*512 大小；

最大最小级别： 设置即切片的地形显示最大最小级别，注意最小级别不能大于最大级别。

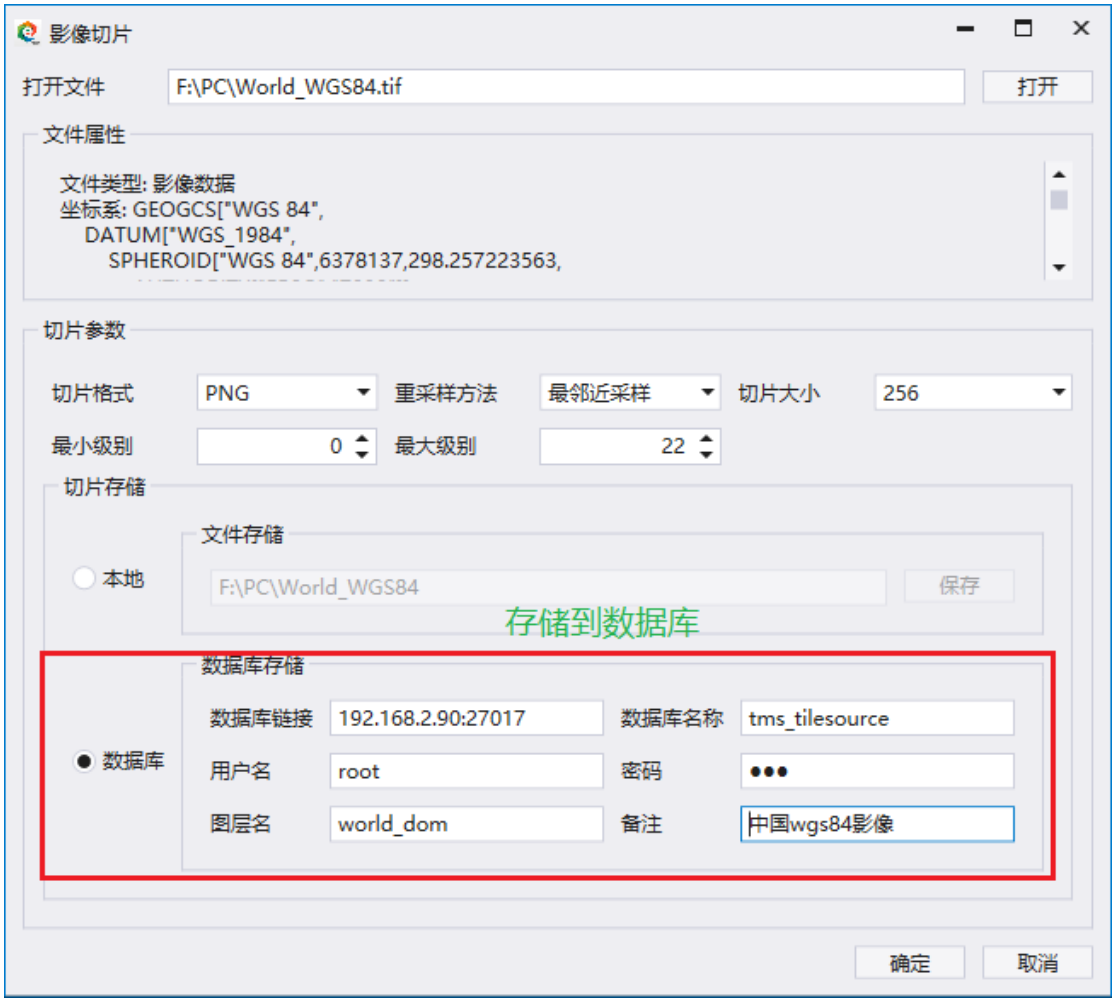
3. 在”切片存储“选择本地存储，点击”保存“选择保存路径后点击”确定“，将在存储路径下生成影像的切片，如图：



4. 如果要生成后直接导入数据库（目前支持的入库数据库类型是 mongodb），在”切片存储“中选择数据库存储，在数据库存中设置

数据库链接：数据库服务地址：端口号；
数据库名称：数据入库时导入的目标数据库名；
用户名：数据库访问用户名
密码：数据库访问密码
图层名：数据入库时在数据库下创建的表名，也是入库后 BcEngineX 加载该数据的图层名，

所以图层名建议不要使用中文、空格、非法字符等;
备注：对数据入库的补充描述信息，入库后会写入在数据库 message 字段中
设置完成如下图:



切片文件转换

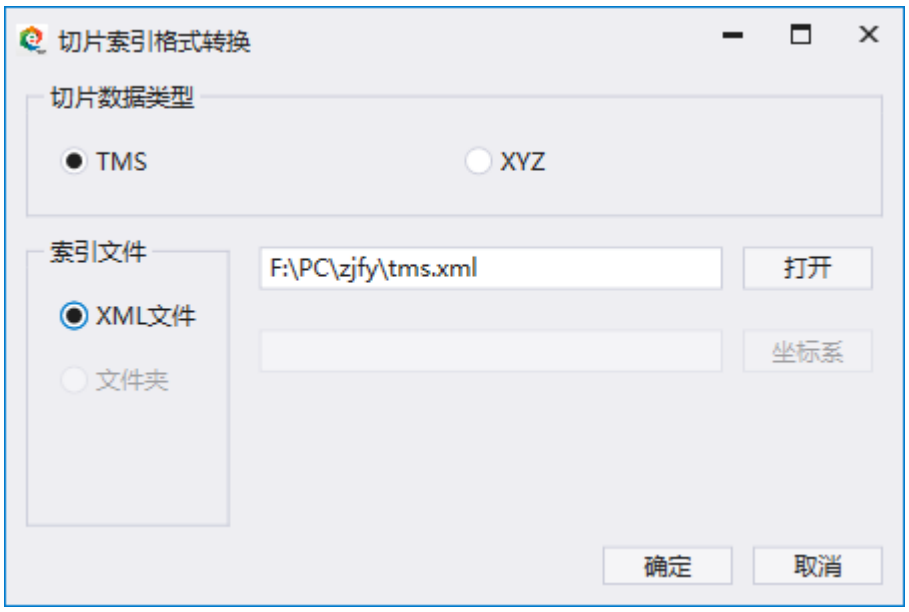
使用说明

为 tms 的影像地形缓存数据转生成 bci（影像缓存生成的描述文件）和 bct（地形缓存生成的描述文件）的 json 描述文件。生成 bci（影像缓存生成的描述文件）的目的是为了兼容 webgl 版本加载。

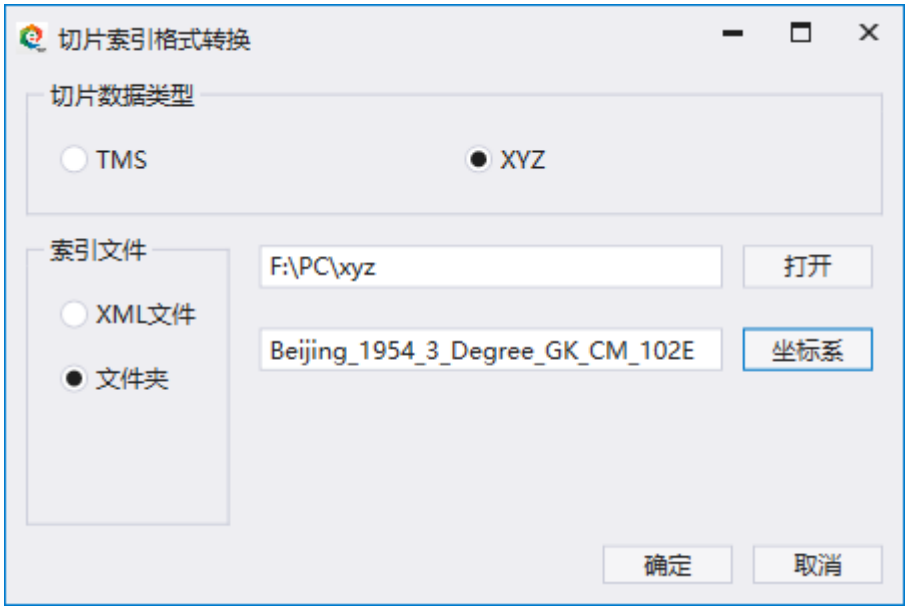
操作步骤

1. 在“gis”选项卡的“二维数据处理”中，打开“切片文件转换”工具；选择要处理的切片数据类型 TMS(影像缓存、地形缓存)，XYZ(XYZ 切片规则影像缓存)。

2. 在“索引文件”中选择”xml 文件“则点击右侧“打开”按钮直接读取 tms 缓存的文件夹下的 xml 文件，点击“确定“则处理完成，如下图：



3. 当选择 xyz 切片规则数据类型时，在“索引文件”中可以选择”XML 文件“也可以选择“文件夹”点击右即包涵 xml 文件的缓存文件目录，如果选择是“文件夹”此时因为没有直接读取 xml 文件所以需要点击”坐标系“手动指定缓存文件的坐标系，点击“确定“则处理完成，如下图：



矢量切片

使用说明

将*.shp 格式矢量数据转换为 BcEngineX 支持的 MVT 缓存（目前仅支持地理 WGS84 坐标系）。

操作步骤

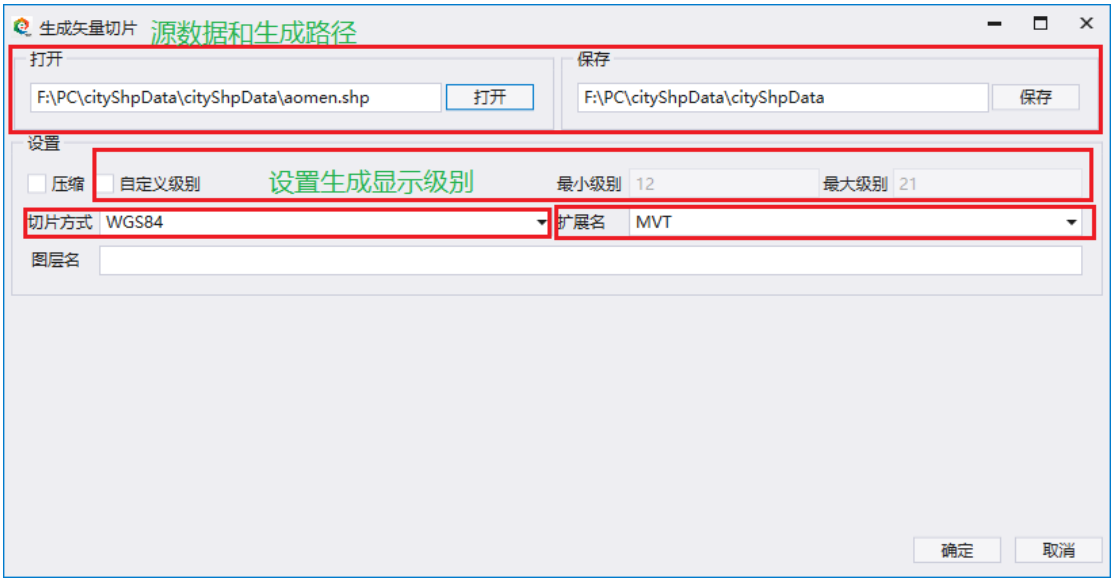
1. 在“gis”选项卡的“二维数据处理”中，打开“矢量切片”工具。

2. “压缩”和“自定义级别”是可选设置，设置压缩可以节省数据占用空间，自定义级别即数据显示的最大最小级别。

3. “切片方式”分为 WGS84 和 WebMercator 两种，WGS84 是 BcEngineX Pro Windows 桌面版本加载使用，WebMercator 是基于 Cesium 的 webgl 版本加载使用（目前 WGS84 切片方式在 PC 端和 wegbl 端均可加载，可以只切 WGS84 一份数据即可）。

4. 扩展名支持 mvt 和 pbf 两种生成的数据格式。

5. 图层名不可为空，设置完成，如下图：



七参数计算

使用说明

已知两个不同空间直角坐标系中的六对 XYZ 坐标值，计算源坐标系坐标点转换为目标坐标系的坐标点所要的七个未知参数（即七参数）。

七参数介绍：

两个不同的三维空间直角坐标系之间转换时，通常使用七参数模型（数学方程组）。在该模型中有七个未知参数，即：

（1）三个坐标平移量（$\triangle X$，$\triangle Y$，$\triangle Z$），即两个空间坐标系的坐标原点之间坐标差值；

（2）三个坐标轴的旋转角度（$\triangle\alpha$，$\triangle\beta$，$\triangle\gamma$），通过按顺序旋转三个坐标轴指定角度，可以使两个空间直角坐标系的 XYZ 轴重合在一起。

（3）尺度因子 K，即两个空间坐标系内的同一段直线的长度比值，实现尺度的比例转换。通常 K 值几乎等于 1。 以上七个参数通常称为七参数。运用七参数进行的坐标转换称为七参数坐标转换。

关于七参数详细介绍可以参考：参照系转换

操作步骤

1. 在“gis”选项卡的“二维数据处理”中，打开“七参数计算”工具；在“源坐标系“和”目标坐标系“中设置源坐标系和目标坐标系。

2. 在”控制点“设置中设置源坐标和转换后的目标坐标，至少添加 3 个控制点；点击”确定“即可计算出源坐标系坐标点转换为目标坐标系所需要的七参数值，如图：

文件投影转换

使用说明

将*.tif 格式数据和*.Shp 格式数据进行坐标系转换。

操作步骤

1. 在“gis”选项卡的“二维数据处理”中，打开“文件投影转换”工具；点击”打开“按钮打开需要进行投影转换处理的 tif 数据或者 shp 数据，打开后在”文件属性“中将会显示数据的文件类型、坐标系等信息；在输出文件中点击”保存“选择处理后保存路径。

2. 在”属性“配置中选择目标坐标系，点击”确定“开始进行转换，如下图：

在”属性“配置中选择目标坐标系，勾选”七参数“输入参数值，可在做坐标系转换同时对三维空间直角坐标系进行偏移（关于七参数详细介绍可以参考：参照系转换），如下图：

坐标投影转换

使用说明

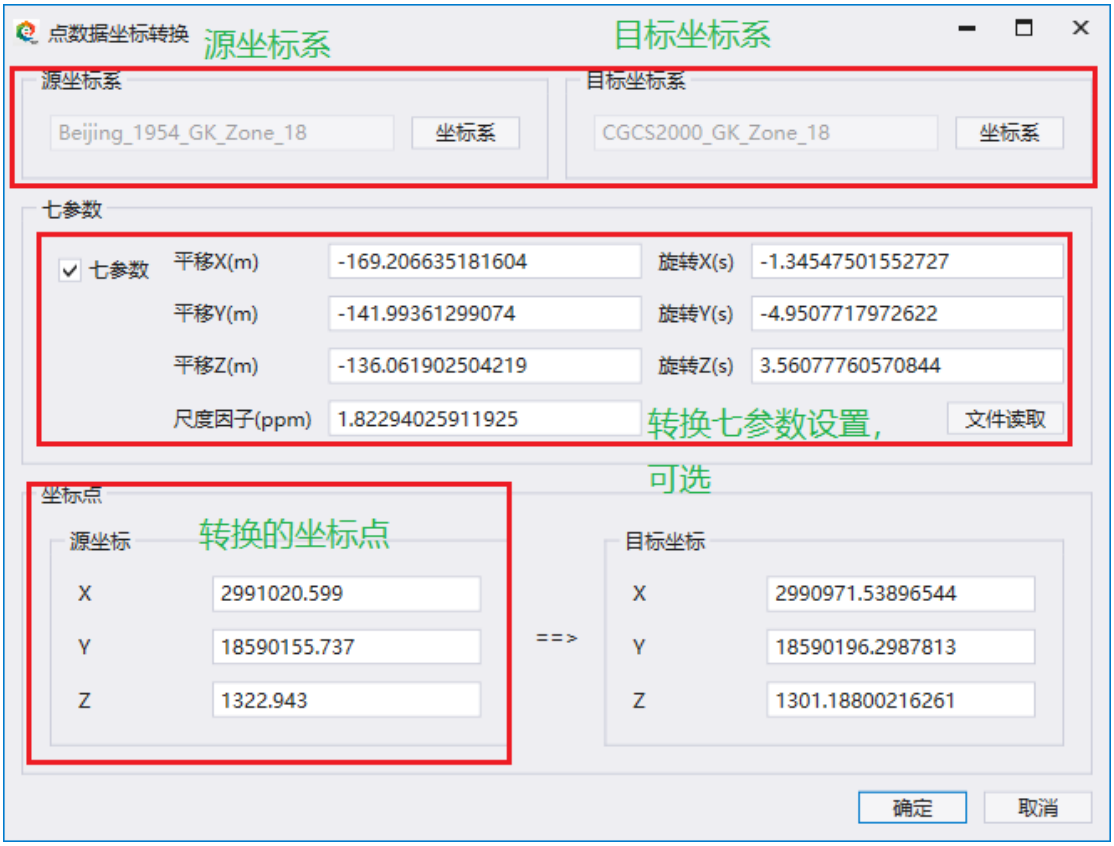
将目标坐标系坐标点转换为指定坐标系坐标点。

操作步骤

1. 在“gis”选项卡的“二维数据处理”中，打开“坐标投影转换”工具；分别在源坐标系和目标坐标系中选择对应的坐标系。

2. 勾选”七参数“，输入正确的七参数值，也可以直接从保存好七参数的 txt 文档中直接读取。

3. 在”坐标点“中输入源坐标点后，点击”确定“即可计算出目标坐标点的值，如图：



几何校正

使用说明

将一幅 tif 数据指定坐标系，并通过 tif 数据上的像素点与其对应的真实的地理坐标来消除 tif 数据的显示的几何误差。

操作步骤

1. 在“gis”选项卡的“二维数据处理”中，打开“几何校正”工具；在”输入文件“中点击”打开“输入需要校正的 tif 数据文件，在”输出文件“中点击”保存“设置保存路径。

2. 在”坐标系“中勾选”设置坐标系“，点击”坐标系“选择对应坐标系；在”映射点“添加多个点，输入像素 X、像素 Y、地形 X、地理 Y 的值点击”确定“开始校正，如下图：

三维数据处理

BcEngineX_Webgl 支持加载的数据除 Terrain 地形和 Cesium 本身支持的数据类型外全部与 BcEngineX 其他平台共用，下面会介绍如何使用 BcEngineX_Pro 平台数据处理工具，以满足 BcEngineX_Webgl 各类型三维数据的处理工作。

UGX 索引

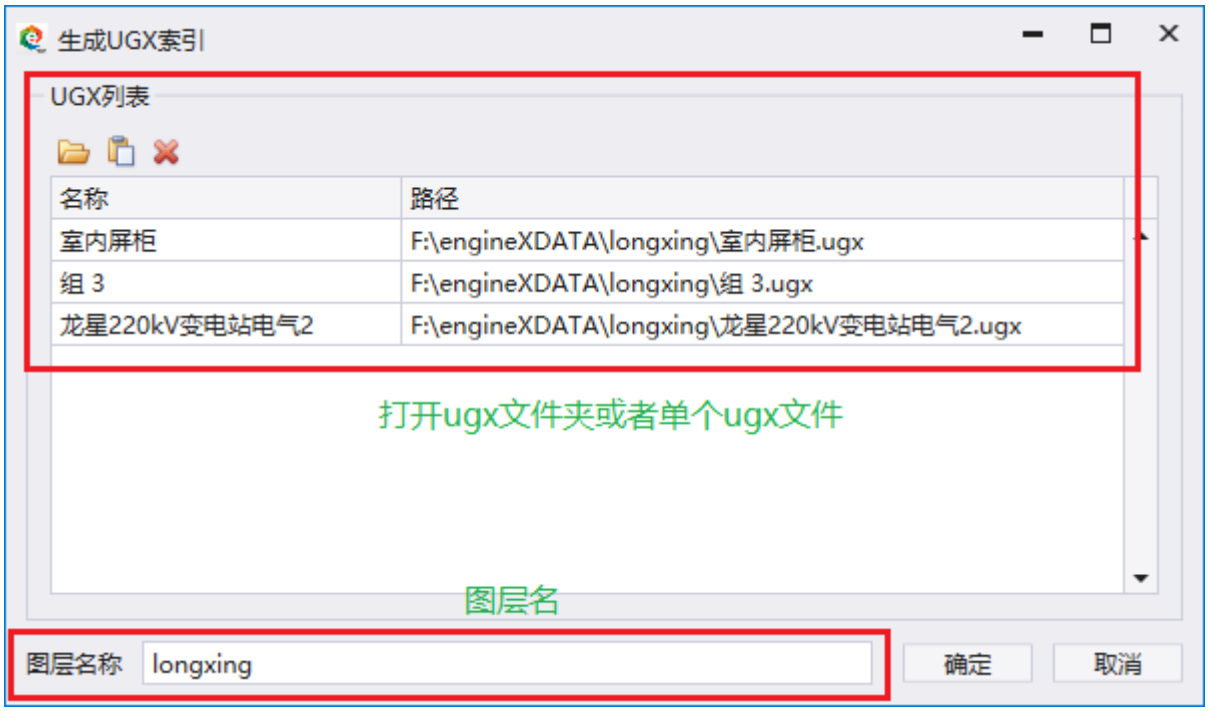
使用说明

生成指定 ugx 文件的索引文件或者指定目录下所有 ugx 文件的统一索引文件。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“UGX 索引”工具；在“ugx 列表”中选择“打开文件夹中的 ugx”（打开整个目录下的 ugx 文件）或者“打开 ugx”（打开指定的单个 ugx 文件）。

2. 在“图层名称”中输入一个图层名（生成的索引文件*.bcx 名称），点击“确定”开始生成，如下图：



BIM 缓存

使用说明

将 bim 模型生成以空间索引组织的 UGTile 格式缓存，用于海量数据索引及快速展示。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“BIM 缓存”工具；在“ugx 文件”中选择“打开文件夹中的 ugx”（打开整个目录下的 ugx 文件）或者“打开 ugx”（打开指定的单个 ugx 文件）。

2. “缓存方案”中提供了生成 bim 缓存默认参数设置，但为了生成的缓存展示效率最大化需要针对数据对缓存方案的参数进行调整，

参数说明：

八叉树最大深度：设置八叉树的深度，即数据总共分多少级（默认深度 5，生成 ugt 时 0.ugt 深度就是 0，0-0.ugt 的深度就是 1，0-0-1.ugt 的深度就是 2 以此类推）。

LOD 距离列表：设置八叉树块数据的视距范围。例如八叉树深度为 0（即第一级）的视距范围为 rangeVec[maxTreeDepth - 1]~rangeVec[maxTreeDepth]，八叉树深度为 1（第二级）的视距范围为 rangeVec[maxTreeDepth - 2]~rangeVec[maxTreeDepth-1]。

最大 LOD 级别：设置 LOD 数据的层级数，目前 UGX 数据 LOD 的层级都是 3 级

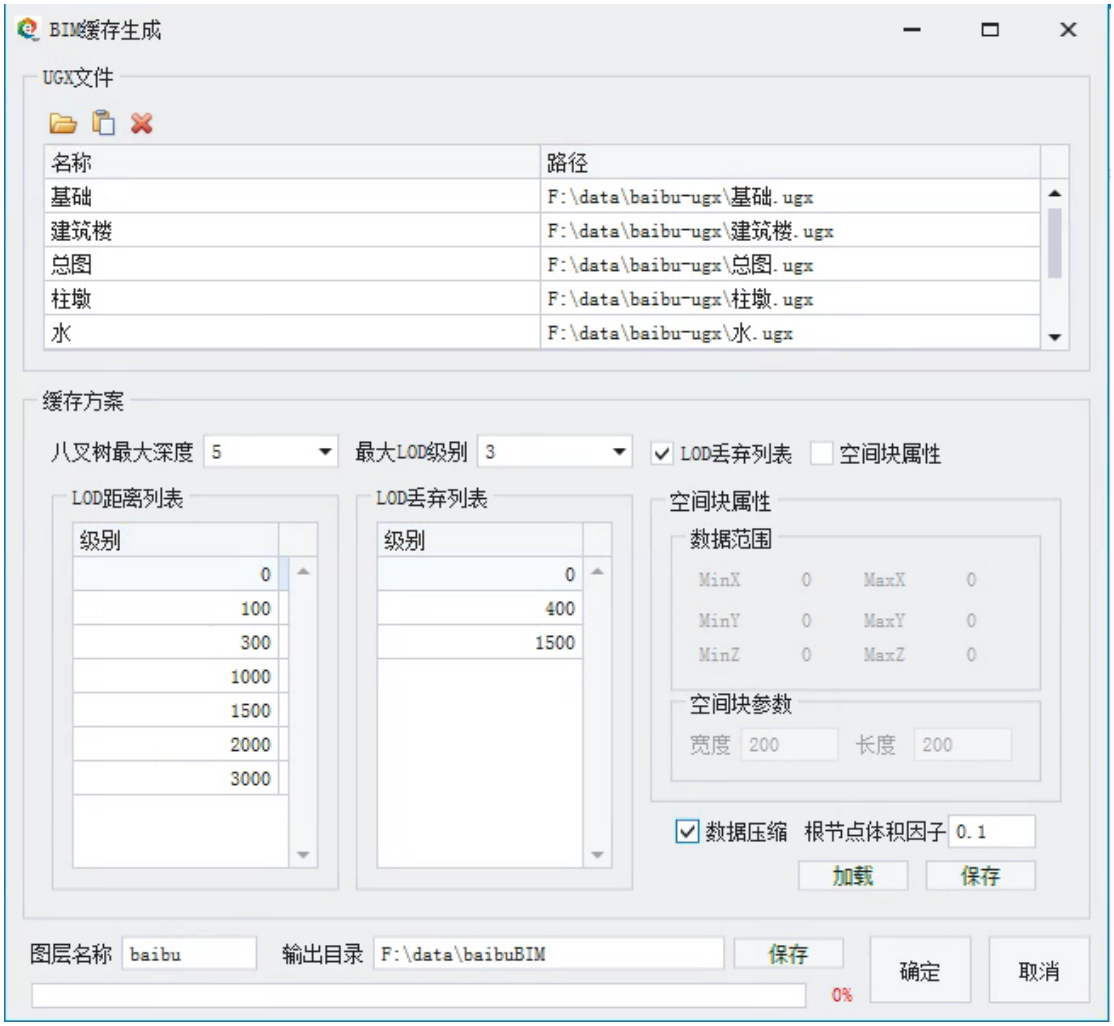
LOD 丢弃列表：设置每级八叉树块（树深度 2 到树深度 4 的块）在分块时丢弃实体数据的参考体积值。例如最大 LOD 级别为 3 时设置丢弃列表第一级为 1500，表示在分块时将分到八叉树深度为 2 的块中的实体对象中有体积小于 1500 的实体给丢弃掉。

空间块参数：将八叉数块较大的数据，以 BoundingBox 的范围大小将数据分为多块（如可以分 2*2、4*4 块）。

数据压缩：是否压缩生成的 ugt 数据;未压缩生成 ugt 数据格式为*.ugt，压缩生成格式为*.ugtz

体积因子：设置八叉树根瓦块（树深度为 0）中保留的实体体积因子。例如体积因子为 0.1，表示根瓦块中将保留显示体积大于 0.1*总数据包围求体积的实体。

3. 配置完缓存方案后在“图层名称”中输入保存的图层名称，“输出目录”后点击“保存”按钮选择存储路径，点击“确定”开始生成 BIM 缓存，如下图：



点云缓存

使用说明

将 las 点云数据生成以空间索引组织的 UGTile 格式缓存，用于海量数据索引及快速展示。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“点云缓存”工具；在“ugx 文件”中选择“打开文件夹中的点云”（打开整个目录下的*.las 文件）或者“打开 las”（打开指定的单个*.las 文件）。

2. 点击”保存“按钮选择”输出目录“。

3. 在”参数配置“中”源参考坐标系“、”输出场景类型”、”参考点“、”压缩“均为选择性配置，可以不做设置；”图层名称“为必须设置参数，

参数说明：

源参考坐标系：生成数据的目标坐标系

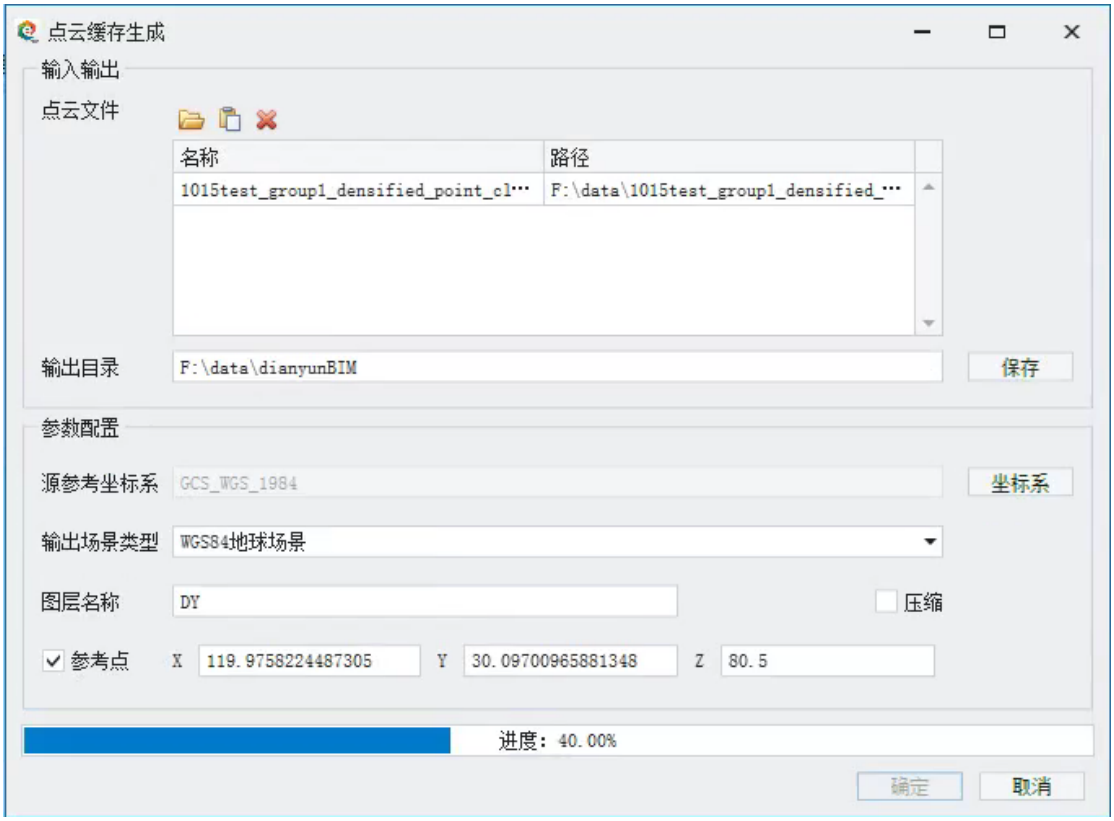
输出场景类型：生成数据加载的场景类型（普通场景、wgs84 地球场景、cgcs 地球场景）

图层名称：生成 bcm 文件名称

压缩：是否压缩生成的 ugt 数据;未压缩生成 ugt 数据格式为*.ugt，压缩生成格式为*.ugtz

参考点：生成的点云缓存数据的初始加载坐标。

4. 设置完”参数配置“点击”确定“按钮开始生成点云缓存，如下图：



电缆缓存

使用说明

将存储有电缆数据信息的 json 文件生成以空间索引组织的 UGTile 格式缓存，用于海量数据索引及快速展示。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“电缆缓存”工具；在“电缆文件”后点击“打开电缆文件”图标打开要处理的 json 文件，在“输出目录”点击选择输出目录图标设置输出路径。

2. 生成电缆缓存还有“压缩数据”、“空间块设置”、“LOD 设置”等选择性参数设置，

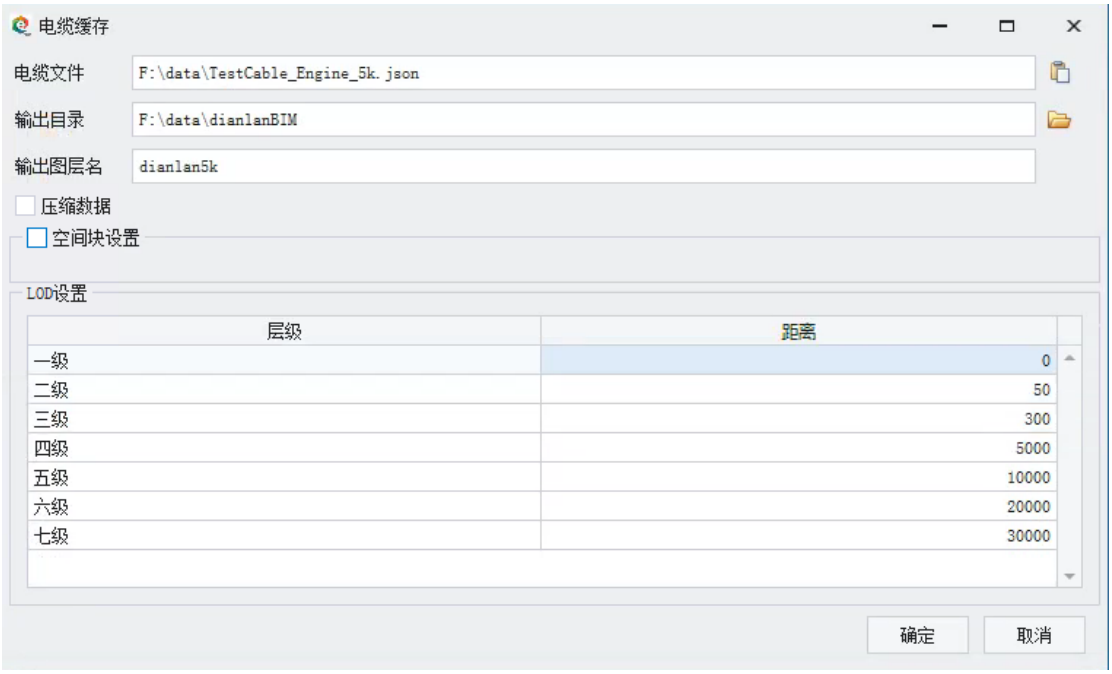
参数说明：

压缩数据：是否压缩生成的 ugt 数据;未压缩生成 ugt 数据格式为*.ugt，压缩生成格式为*.ugtz

空间块设置：将八叉数块较大的数据，以 BoundingBox 的范围大小将数据分为多块（如可以分 2*2、4*4 块）。

LOD 设置：设置八叉树块数据的视距范围。例如八叉树深度为 0（即第一级）的视距范围为 rangeVec[maxTreeDepth - 1]~rangeVec[maxTreeDepth]，八叉树深度为 1（第二级）的视距范围为 rangeVec[maxTreeDepth - 2]~rangeVec[maxTreeDepth-1]。

3. 设置完如上选择性配置参数点击”确定“按钮开始即可生成点云缓存，如下图：



倾斜缓存

使用说明

将 osgb 倾斜摄影数据生成以空间索引组织的 UGTile 格式缓存，用于海量数据索引及快速展示。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“倾斜缓存”工具；在“输入输出”中点击“打开”打开要处理的倾斜摄影数据选择到 Data 目录，点击“保存”选择输出保存目录。

2. “属性配置”中在“图层名称”中输入图层名称为必须输入参数，其他参数为选择性配置参数

参数说明：

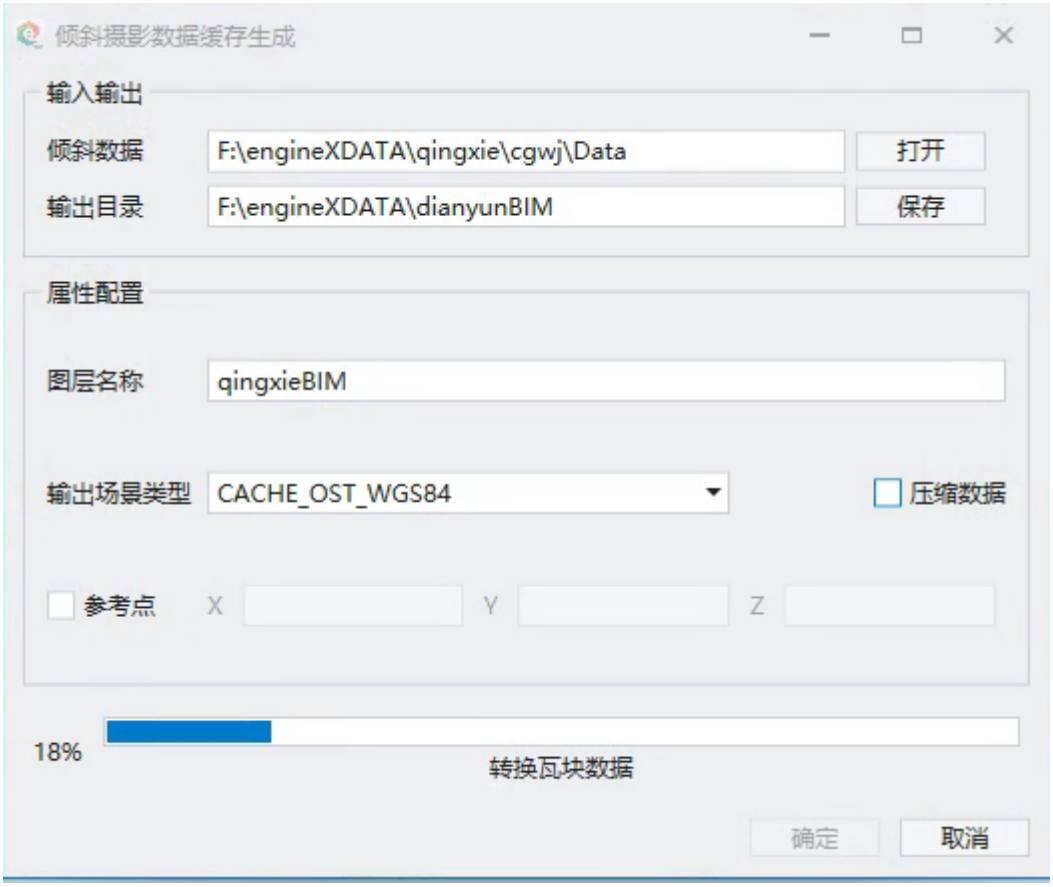
输出场景类型：生成数据加载的场景类型（普通场景、wgs84 地球场景、cgcs 地球场景）

图层名称：生成 bcm 文件名称

压缩数据：是否压缩生成的 ugt 数据;未压缩生成 ugt 数据格式为*.ugt，压缩生成格式为*.ugtz

参考点：生成的点云缓存数据的初始加载坐标。

3. 设置完”属性配置“点击”确定“按钮开始生成倾斜缓存，如下图：



倾斜文件索引

使用说明

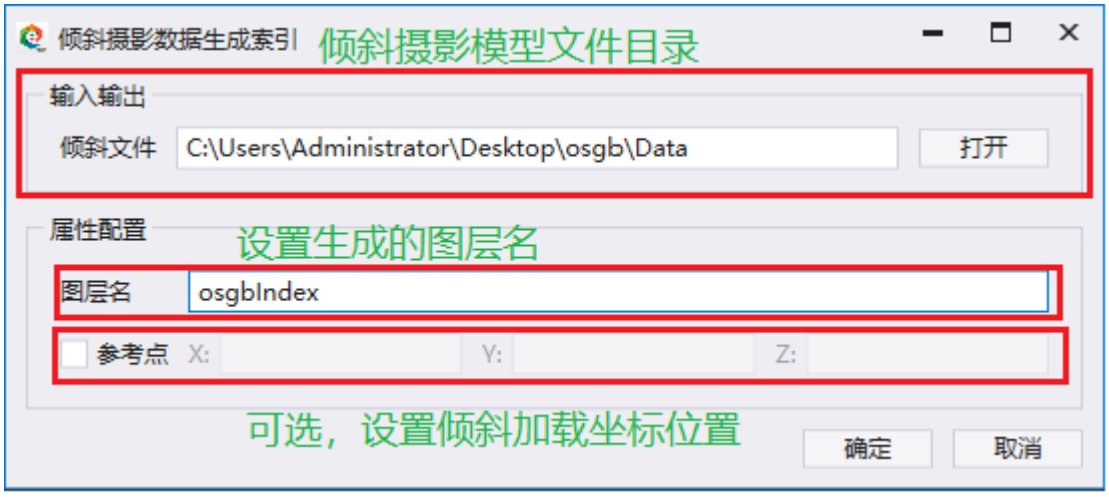
为倾斜缓存数据生成文件索引文件，可通过文件索引加载倾斜缓存。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“倾斜文件索引”工具；

2. 打开倾斜文件目录，选至 Data 目录下；

3. 在“属性配置”中输入图层名，参考点是给倾斜数据一个加载的参考点位置信息为可选项。



倾斜空间索引

使用说明

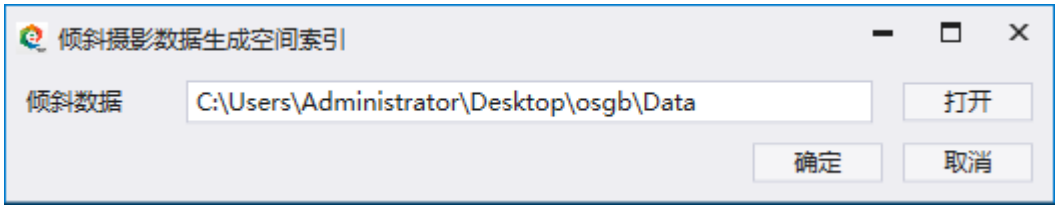
为倾斜缓存数据生成空间索引文件，达到快速高效查询、检索和显示。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“倾斜空间索引”工具；

2. 打开倾斜文件目录，选至 Data 目录下；

3. 点击“确定”开始生成，生成完成后会在 Data 目录下生成一个 Tile_Xml 文件夹，文件夹下存储的倾斜数据的 xml 格式空间索引文件。



矢量建模

使用说明

矢量数据根据拉伸字段拉伸生成模型。

操作步骤

1.在“处理”选项卡的“三维数据处理”中，打开“矢量建模”工具；

设置“基础设置”参数说明：

shp 文件：需要处理的 shp 文件，

输出目录：处理完成存储路径，

图层名：场景加载的图层名，

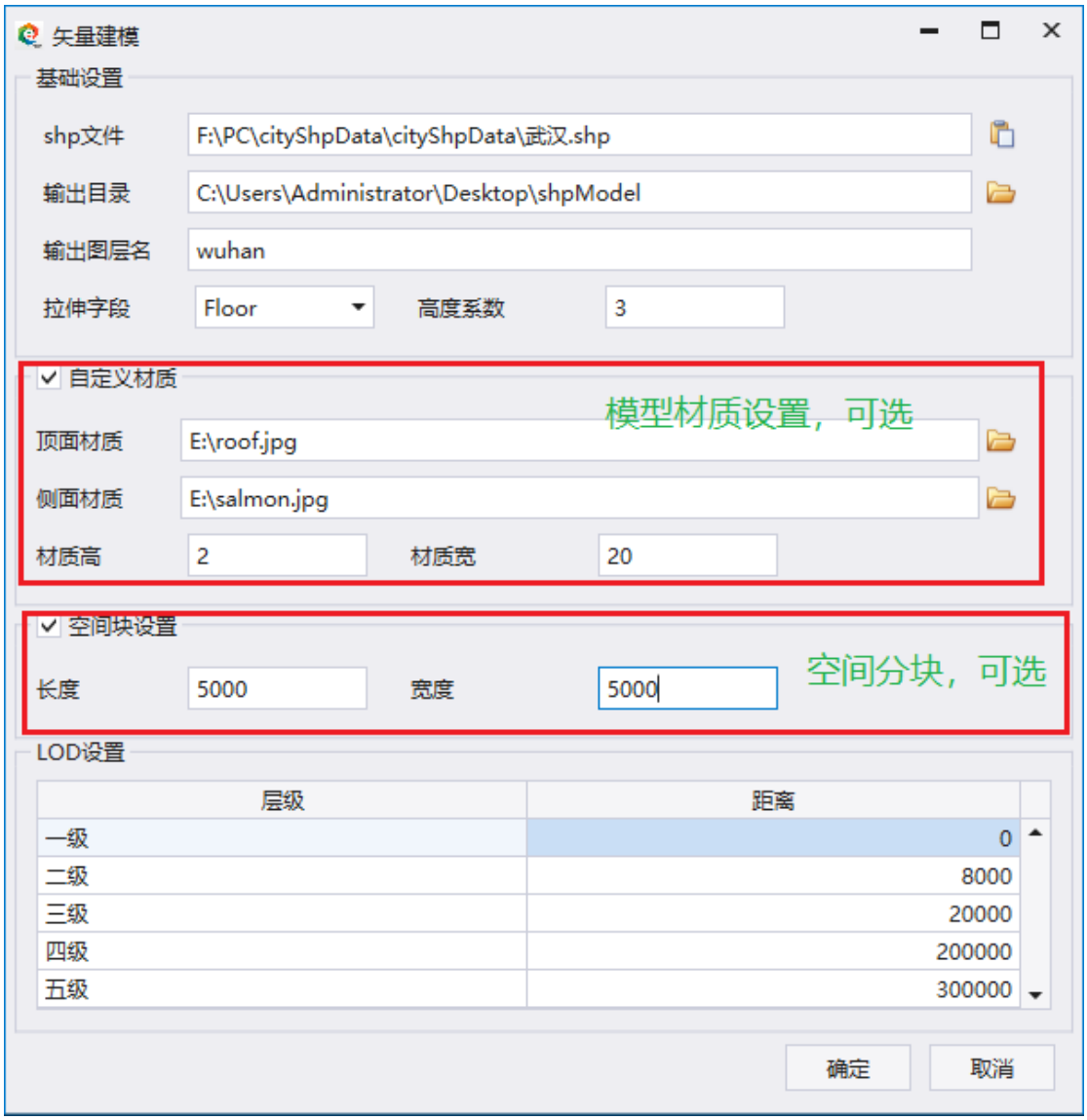
拉伸字段：选择根据 shp 数据的哪个字段值进行拉伸，这里选择“Floor”楼层字段，

高度系数：默认 3m 即每一层楼的默认高度（例如 Floor 楼层为 5 楼，高度系数为 3m，则拉伸时模型高度为 5*3m）。

2.“自定义材质”设置是可选项，若不选择会按照默认材质设置，材质高（墙面纹理层数）、材质宽（墙面纹理宽度，单位 m）；

3.“空间块设置”设置是可选项，当数据量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多个等大的分块处理，然后再对子分块处理。这里设置每一块的长度和宽度；

4.“LOD 设置”设置八叉树块数据的视距范围。例如八叉树深度为 0（即第一级）的视距范围为 rangeVec[maxTreeDepth - 1]~rangeVec[maxTreeDepth]，八叉树深度为 1（第二级）的视距范围为 rangeVec[maxTreeDepth - 2]~rangeVec[maxTreeDepth-1]。



三维管线

使用说明

根据二维点、线数据，采用自适应三维管点、管线参数化建模方法，构建三维管线。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“三维管线”工具；

2. 在“CSV 目录“中选择生成管线的二维点和线数据的 csv 目录；

3. “管网映射文件“中选择管网映射文件的 json 文件；

4. 在“模板库文件“中打开模板库文件的 json 文件；

5. 设置“缓存方案设置“，缓存方案设置的参数说明如下：

八叉树最大深度：设置八叉树的深度，即数据总共分多少级（默认深度 5，生成 ugt 时 0.ugt 深度就是 0，0-0.ugt 的深度就是 1，0-0-1.ugt 的深度就是 2 以此类推）。

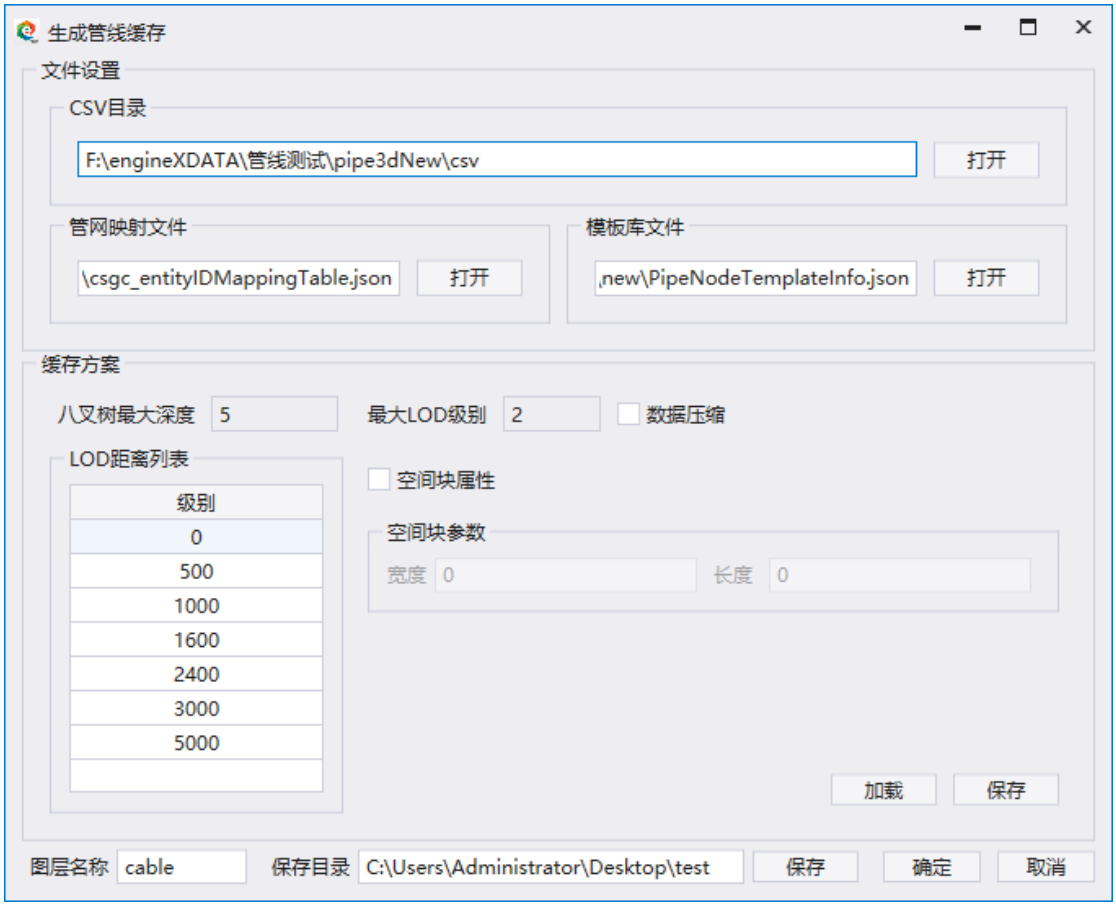
LOD 距离列表：设置八叉树块数据的视距范围。例如八叉树深度为 0（即第一级）的视距范围为 rangeVec[maxTreeDepth - 1]~rangeVec[maxTreeDepth]，八叉树深度为 1（第二级）的视距范围为 rangeVec[maxTreeDepth - 2]~rangeVec[maxTreeDepth-1]。

最大 LOD 级别：设置 LOD 数据的层级数，目前 UGX 数据 LOD 的层级都是 3 级。

空间块参数：当数据量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多个等大的分块处理，然后再对子分块处理。这里设置每一块的长度和宽度。

数据压缩：是否压缩生成的 ugt 数据;未压缩生成 ugt 数据格式为*.ugt，压缩生成格式为*.ugtz。

6. 设置图层名和输出目录点击确定开始生成。



*.scv 文件：CSV（即 Comma Separate Values）是以文本形式记录数据的文件（通常以逗号为分隔符），这种格式经常用来作为不同程序之间的数据交互的格式。

UGX 检查

使用说明

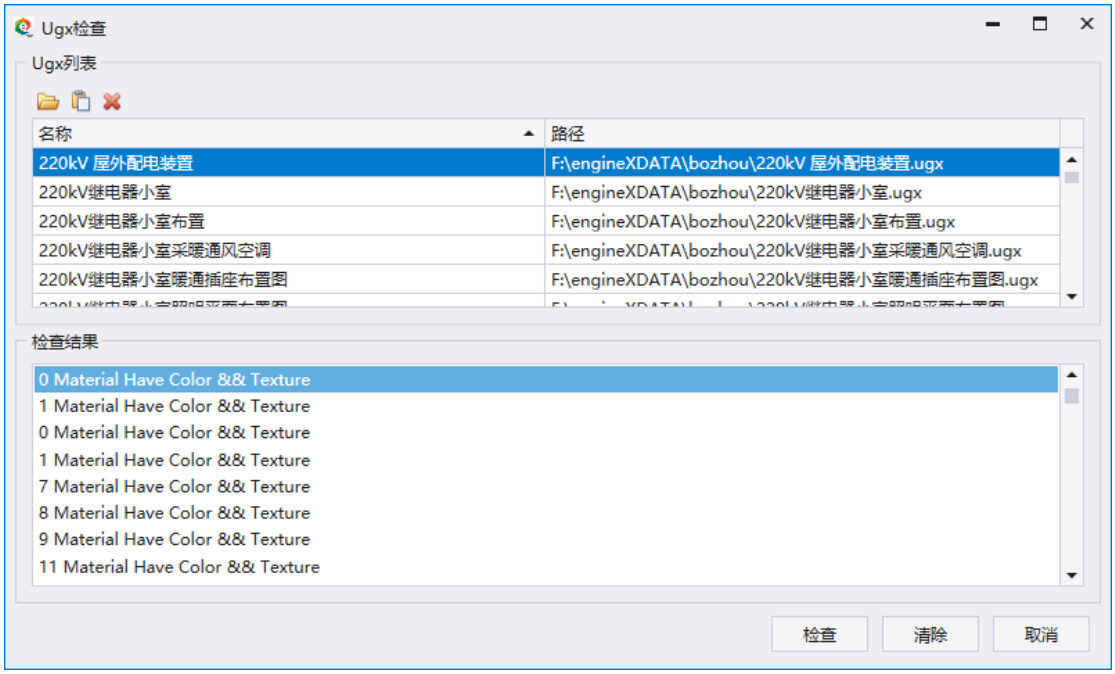
检测导入的 ugx 数据的完整性和规范性。

操作步骤

1. 在“处理”选项卡的“三维数据处理”中，打开“UGX 检查”工具；

2. 打开 ugx 文件夹或者单个 ugx 文件，开始检查；

3. 若 ugx 数据生产的有异常，可以在检查结果中查看检查结果。


Ugx检查
Ugx列表
名称
路径
220kV 屋外配电装置
F:\engineXDATA\bozhou\220kV 屋外配电装置.ugx
220kV继电器小室
F:\engineXDATA\bozhou\220kV继电器小室.ugx
220kV继电器小室布置
F:\engineXDATA\bozhou\220kV继电器小室布置.ugx
220kV继电器小室采暖通风空调
F:\engineXDATA\bozhou\220kV继电器小室采暖通风空调.ugx
220kV继电器小室暖通插座布置图
F:\engineXDATA\bozhou\220kV继电器小室暖通插座布置图.ugx
检查结果
0 Material Have Color && Texture
1 Material Have Color && Texture
0 Material Have Color && Texture
1 Material Have Color && Texture
7 Material Have Color && Texture
8 Material Have Color && Texture
9 Material Have Color && Texture
11 Material Have Color && Texture
检查
清除
取消

常见问题解答

产品安装和运行

问：BcEngineX_Webgl 支持 32 位操作系统环境吗？

答：　　支持；但是 32 位操作系统环境上 32 位浏览器会有内存限制。

问：BcEngineX_Webgl 地形数据是否和 BcEngineX PC 平台共用？

答：　　不共用；BcEngineX_Webgl 支持的地形数据是 TIN 地形三角网瓦片。
所以需要单独生成；但是，影像数据是可以共用的。

问：BcEngineX_Webgl 是否支持设置默认加载的影像底图？

答：　　支持；在 webgl 发布包 Assets\ImageLayer 目录下存放的默认全球底图，
如果需要修改默认底图也可以在创建场景时预先加载上自定义的影像底图。

问：BcEngineX_Webgl 运行时出现浏览器崩溃（浏览器白屏显示崩溃了）是怎么回事？

答：　　出现浏览器白屏崩溃了，可能是浏览加载过程中发送 http 请求响应超时导致，刷新界面即可恢复，若出现加载某一份数据就浏览器崩溃的情况，请检查网络环境和数据正确性。

问：BcEngineX_Webgl 运行时出现黑屏并弹出提示"An error occurred while rendering. Rendering has stopped"？

答：　　这种情况需要先进行排查是否数据问题，首先清除浏览器 db 缓存以 chorm 浏览器为例 F12 进入控制台-Application-IndexDB 然后删除下面的 db 缓存，重新刷新页面运行；查看 Console 控制台输出可以尽量定位到问题。

问：BcEngineX_Webgl 支持从数据库直接加载影像地形或者模型数据吗？

答：　　webgl 为 B/S 模式，只接受 http 请求；如果需要从数据库加载影像地形或者模型服务，需要做 http 服务转发来达到需求；BcEngineX_Webgl 本身不提供直接访问数据库方式。