清华大学
Tsinghua University

《土木与建筑工程CAE》结题报告

# 面向云渲染的BIM模型网络端加载与展示

小组成员： 闵妍涛、安芃

指导教师： 胡振中 副教授

# 目录

# 1. 选题背景与研究方案

选题背景

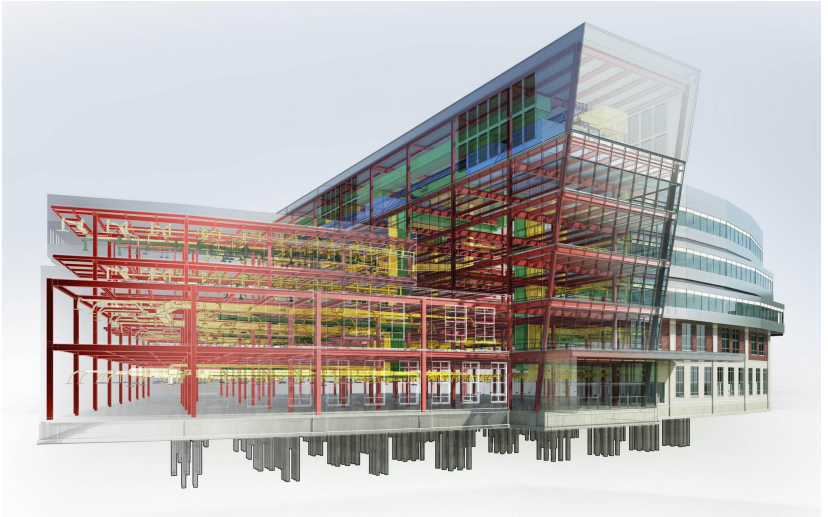
研究目标

研究内容

## ➢ 建筑信息模型

(Building Information Modeling, BIM)

- **核心功能：**工程模型的三维可视化
- **性质：**完整、真实且支持动态更新的建筑信息库
- **作用：**高效的信息交换与共享的平台
- **应用阶段：**全生命周期

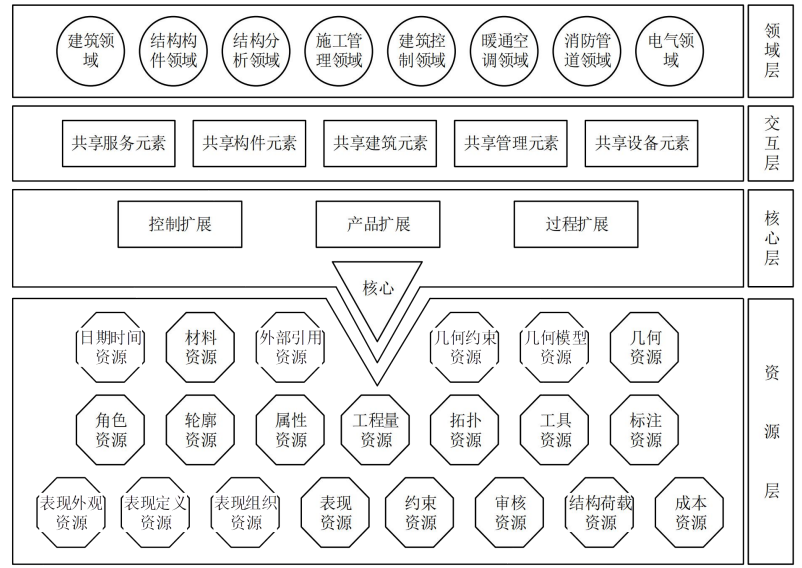


BIM模型

## ➢ BIM数据标准框架：建筑业国际数据标准IFC

(Industry Foundation Classes, IFC)

- **基本架构：**资源层、核心层、交互层、领域层
- **基本组成：**基于继承型层次结构的实体和关系对象
- **信息类别：**构件信息(IfcElement)、几何信息(IfcRepresentaion)、材质信息(IfcMaterial)、空间关联信息(IfcRelConnectsElement)等




IFC基本架构

选题背景

研究目标

研究内容

# 1. 选题背景与研究方案

- **数据转换问题：**数据文件规模大、冗余度高
- **场景加载问题：**网络资源有限、模型数据体量较大
- **资源架构问题：**Web端能力有限、加载任务繁重

- **端**：获取使用场景信息
- **云**：数据转换与集中运算、渲染模型效果

**IFC文件的解析与转换**

- 以构件为单位对IFC文件进行拆分；添加LOD（Level of Details）信息
- 为面向场景的管理提供**细粒度**数据支撑

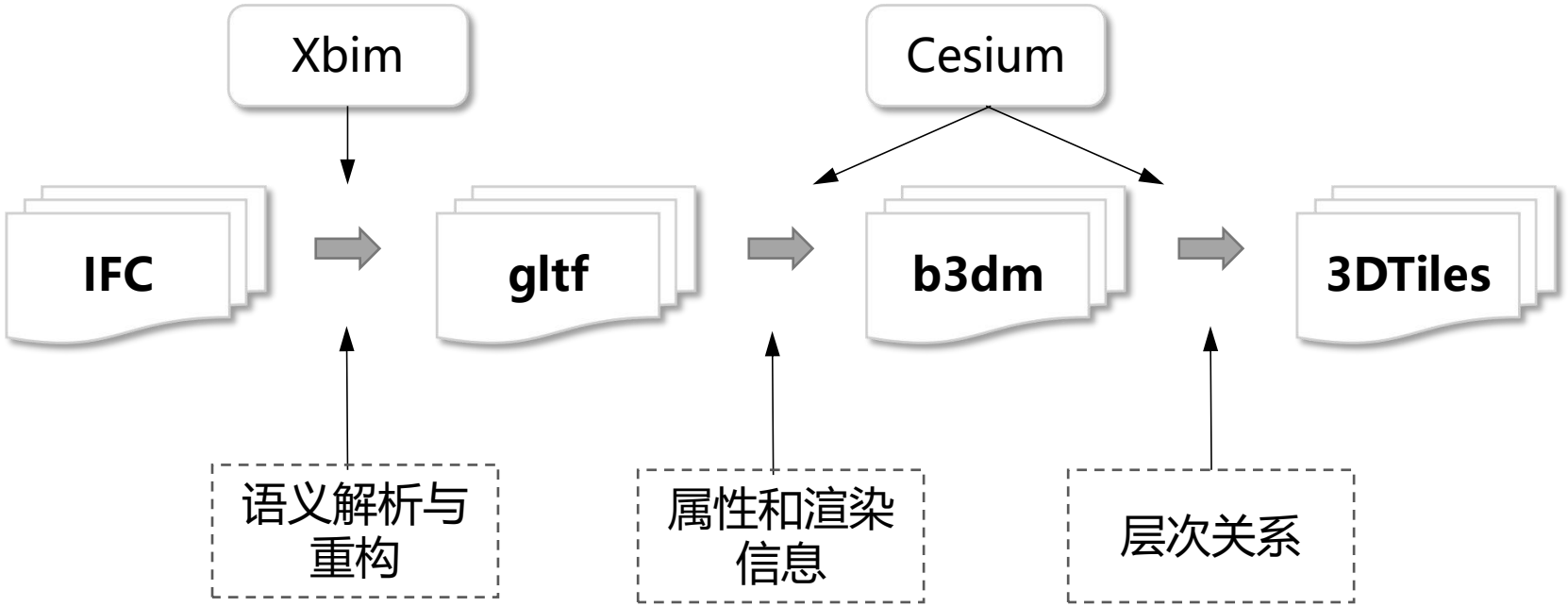
**面向云端机制的网络服务架构**

- 数据云处理，网页端显示交互
- 实时传输**稳定性**及交互效率



# 2. 基于Cesium的模型可视化平台

## ➢ 格式转换




### ➢ gltf:

- 贴合图形渲染 API 的逻辑，存储顶点、法线、顶点颜色等基础信息
- 具有可解释的组织结构

### ➢ 3DTiles:

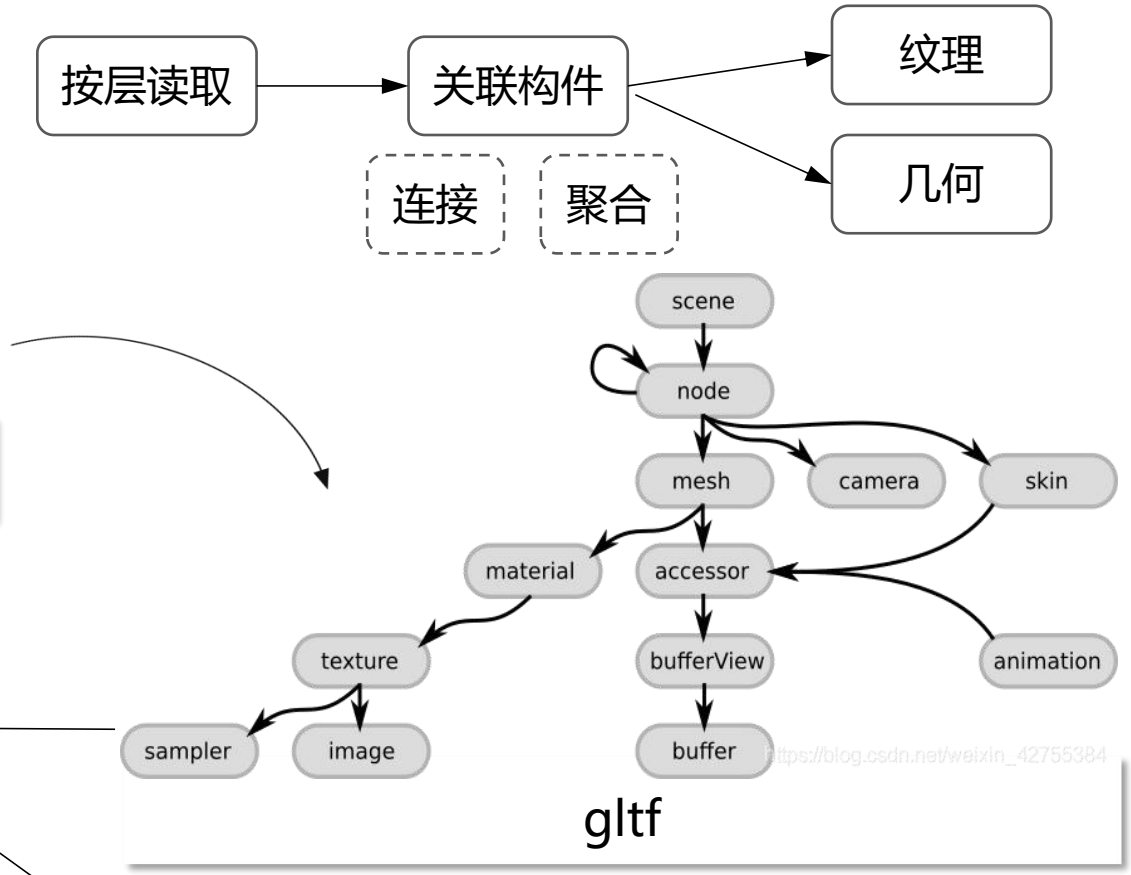
- 继承了 gltf 的优点
- 对大规模的三维数据进行组织（层次细节模型、模型的属性数据、模型的层级数据等）

# 2. 基于Cesium的模型可视化平台

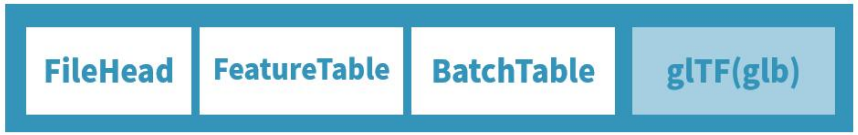
## ➢ 格式转换

```
DATA;
#1= IFCORGANIZATION($,'Autodesk Revit 2020 (ENU)',$,$,$);
#5= IFCAPPLICATION(#1,'2020','Autodesk Revit 2020 (ENU)','Revit');
#6= IFCCARTESIANPOINT((0.,0.,0.));
#9= IFCCARTESIANPOINT((0.,0.));
#11= IFCDIRECTION((1.,0.,0.));
#13= IFCDIRECTION((-1.,0.,0.));
#15= IFCDIRECTION((0.,1.,0.));
#17= IFCDIRECTION((0.,-1.,0.));
#19= IFCDIRECTION((0.,0.,1.));
#21= IFCDIRECTION((0.,0.,-1.));
```

IFC




gltf




B3dm（Tile）

要素表：记录模型的个数（逻辑上不可再分）

记录中心坐标

批量表：几何数据 (gltf) 和属性数据一一对应（BATCHID）

+ tileset.json

3DTiles

# 2. 基于Cesium的模型可视化平台

格式转换

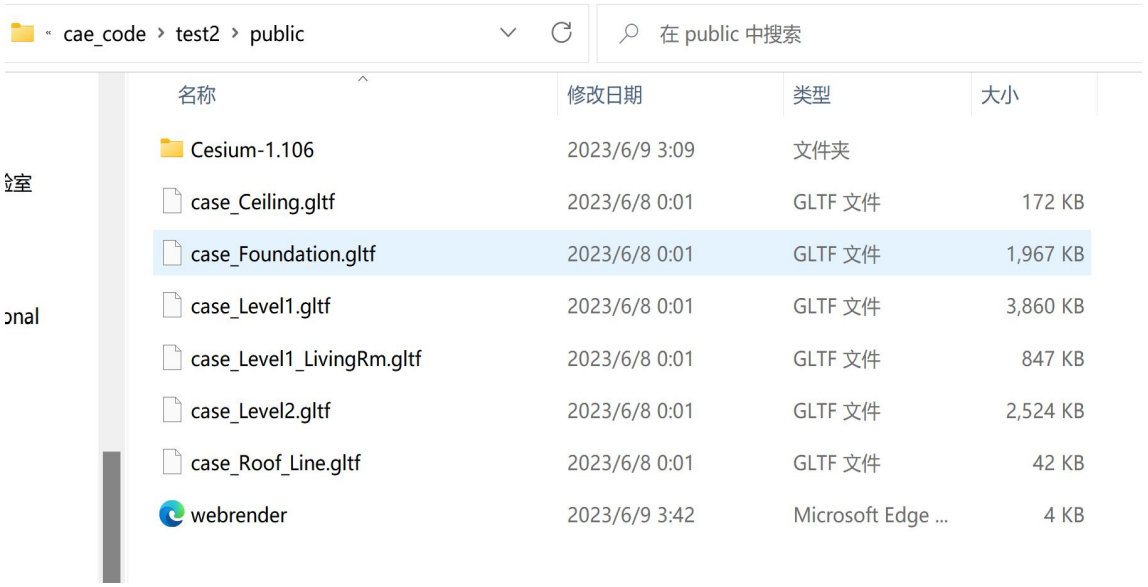
环境搭建

可视化平台

## Cesium

- ➢ 开源
- 三维地球和地图可视化开源JS库
- ➢ 高效
- 使用WebGL进行硬件加速图形开源
- ➢ 多元
- 使用3d tiles格式流式加载不同3d数据(倾斜摄影、CAD和BIM，点云等)
- 全球高精度地形数据可视化

**Node.js + http协议**




# 2. 基于Cesium的模型可视化平台

格式转换

环境搭建

可视化平台

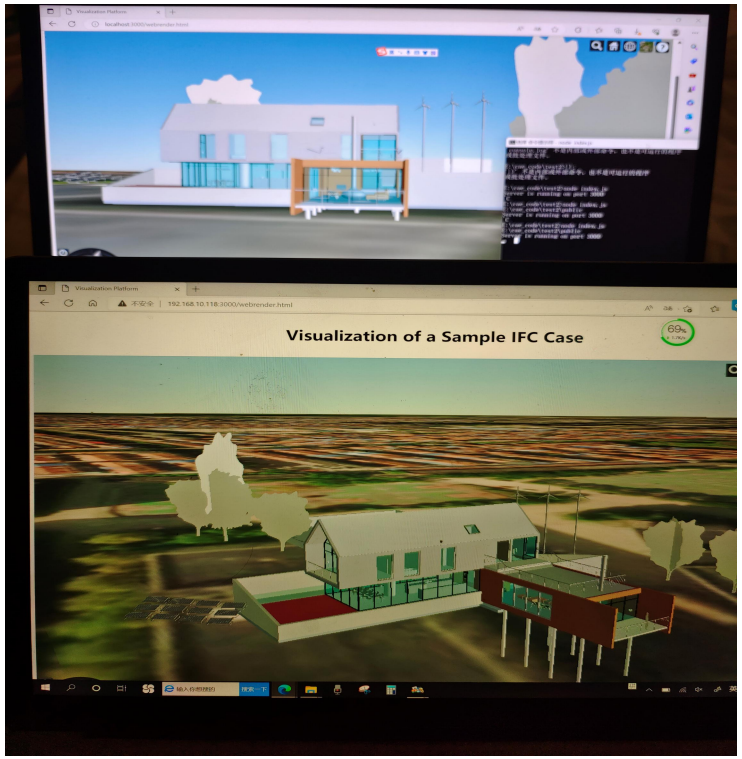
- 上传文件
- 监听鼠标事件
- 实现场景渲染
- 支持远程访问（验证云端部署可行性）

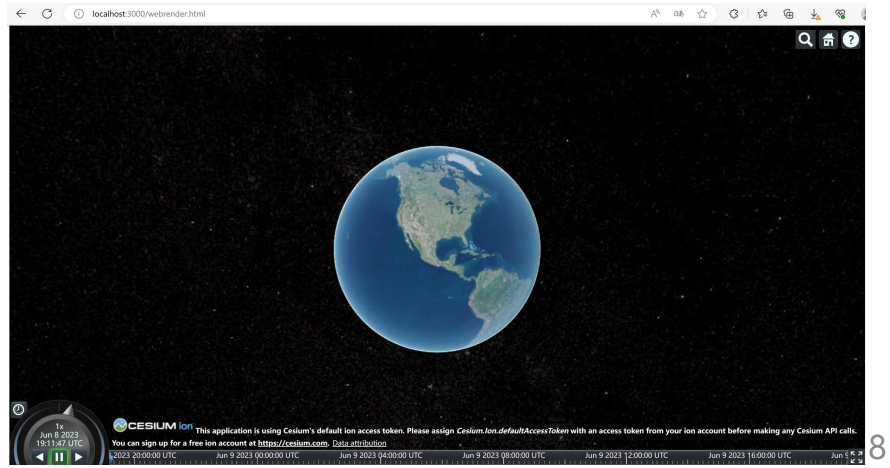
```
//监听文件选择器的变化事件
    fileInput.addEventListener('change', (event) => {
      const file = event.target.files[0];
      if (file) {
        const reader = new FileReader();
        reader.onload = (e) => {
          const img = document.createElement('img');
          img.src = e.target.result;
          //previewDiv.innerHTML = '';
          //previewDiv.appendChild(img);
          //调用发送IFC文件的函数
          console.log('oooooo')
          responseDiv.textContent = "Successfully Upload.";
        };
        reader.readAsDataURL(file);
      }
    });

    // 在这里添加Cesium渲染代码

    var viewer = new Cesium.Viewer("cesiumContainer");

    var entity1 = viewer.entities.add({
        position : Cesium.Cartesian3.fromDegrees(114.33,30.35)
```

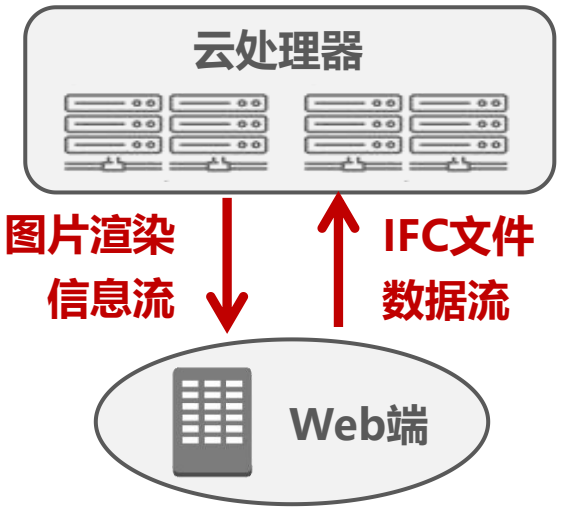


# 3. 基于图片传输和代码集成的云加载方案

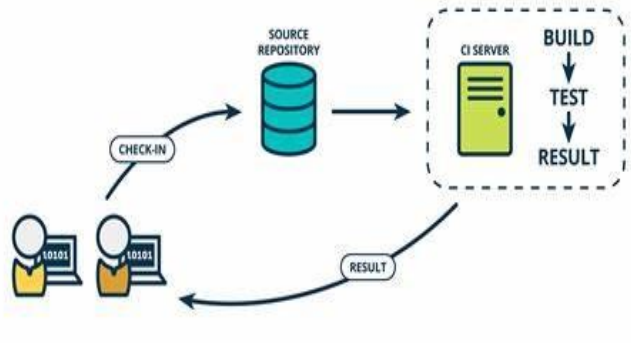
## ➢ 云渲染结果的图片形式传输

- **目标**：Web端上传的IFC文件至云服务器渲染，结果以**图片形式**传递至Web端，进行模型展示。后续通过云-端交互实现Web端鼠标操作的可视化响应。
- **优势**：有效利用云处理器强大的**计算资源**，减少传输过程的**时间损耗**和通道**资源占用**。




## ➢ 处理器端的代码集成

- **目标**：云处理器的后端代码实现数据解析、格式转化、窗口渲染、图片生成与响应回传等**业务流程的整合**。
- **优势**：实现计算任务在云服务器端的**集中部署**，满足客户端上传文件的**实时加载**需求。



加载思路

代码流程

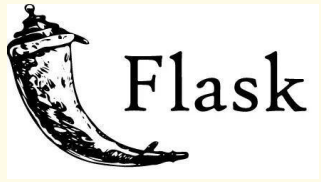
平台展示

# 3. 基于图片传输和代码集成的云加载方案

## ➢ 代码框架

### 后端：Python - Flask框架

- ➢ 交互触发：文件上传、鼠标操作
- ➢ 响应回传：图片、提示信息


```
backend × bc2 × test ×
D:\python\python37\python.exe E:/cae_code/backend.py
 * Serving Flask app 'backend'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
 * Running on http://192.168.10.118:5000
Press CTRL+C to quit
<ifcopenshell.file.file object at 0x000002431BE6B888>
[#115=IfcBuilding('3ZGD7y6S5209$mGLi_sPli',#41,'Samuel Macalister sample house design'
Finished
screenshot
RAC_basic_sample_project.ifc
<FileStorage: 'RAC_basic_sample_project.ifc' ('application/octet-stream')>
192.168.10.118 - - [09/Jun/2023 05:33:02] "POST /upload HTTP/1.1" 200 -
192.168.10.118 - - [09/Jun/2023 05:34:09] "OPTIONS /scroll HTTP/1.1" 200 -
```

### 前端：javascript

- ➢ 页面显示、交互、响应



```
<body>
  <h1>Upload the IFC File</h1>
  <input type="file" id="ifcFileInput">
  <div id="response"></div>
  <div id="preview"></div>

  <script>
    const fileInput = document.getElementById('ifcFileInput');
    const responseDiv = document.getElementById('response');
    const previewDiv = document.getElementById('preview');
```

# 3. 基于图片传输和代码集成的云加载方案

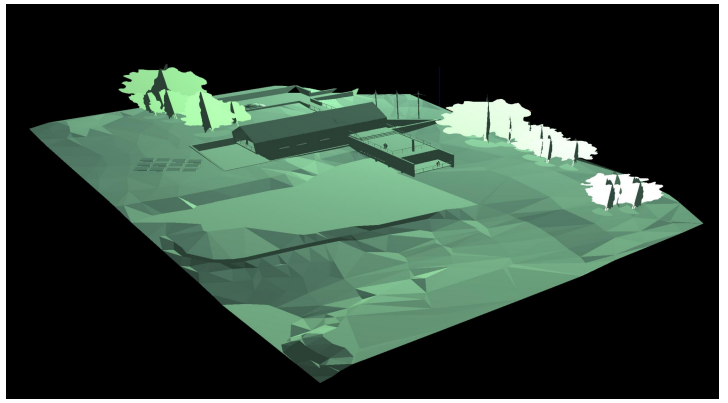
## ➢ 后端代码流程

输入：IFC文件

**➢ IFC预处理**
- 文件解析和几何信息提取
- ifcOpenshell

**➢ IFC格式转化**
- IFC转为OBJ文件
- IfcConverter

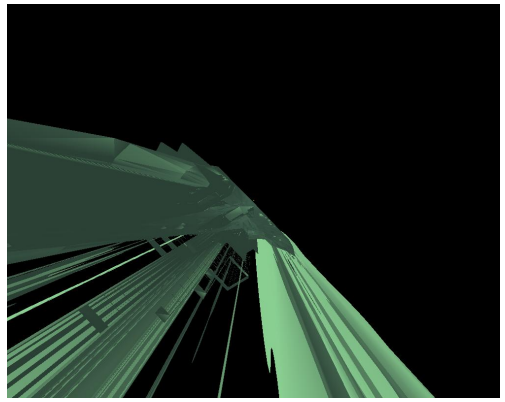
OBJ文件



鼠标滚动/拖拽数据

**➢ 模型渲染**
- OBJ文件的窗体渲染
- 设置相机视角和投影矩阵
- OpenGL

png文件



OpenGL窗体渲染结果

**➢ 响应回传**
- png文件的回传
- flask_cors

输出：图片/信息

# 3. 基于图片传输和代码集成的云加载方案

加载思路

代码流程

平台展示

# 4. 分析与讨论

方案分析

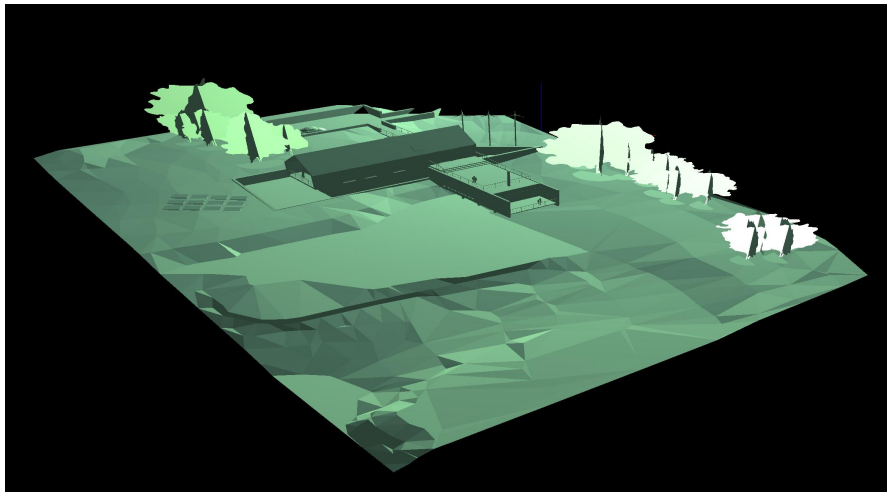
未来展望

## ➢ 方案一

- **优点**：基于Cesium成熟的**可视化渲染引擎**，和较为成熟的IFC**轻量化**工具，服务器端运行和渲染速度快，渲染结果稳定
- **不足**：尚未实现客户端IFC文件的实时渲染，业务流程有待整合


## ➢ 方案二

- **优点**：**图片传输**形式降低了传输过程中的资源消耗；后端**代码集成**实现全业务流程的整合，有助于实现Web端的实时交互和渲染
- **不足**：后端格式转换代码运行速度较慢，拖累渲染速度；渲染效果有待改进、鼠标交互的稳定性有待提升



方案分析

未来展望

# 4. 分析与讨论

## 1 考虑完善图片形式的渲染数据传输

- 将渲染和加载过程部署于云服务器 》》 **有效解决本地系统负载过大的问题**

## 2 深入挖掘云处理器的计算资源

- 在云服务器部署并行运算框架
- 数据传输、Web端处理考虑加载优先级

》》 **提升格式转换和渲染速度**

## 3 面向云-边-端架构的部署

- 实时交互的渲染任务下沉至边 》》 **提升渲染速度，降低资源消耗**

清華大學
Tsinghua University

# 敬请批评指正

汇报人：闵妍涛、安芃
单　位：清华大学深圳国际研究生院
邮　箱：ap22@mails.tsinghua.edu.cn
minyt22@mails.tsinghua.edu.cn