第44卷第6期
2016年12月

浙江工业大学学报
JOURNAL OF ZHEJIANG UNIVERSITY OF TECHNOLOGY

Vol. 44 No. 6
Dec. 2016

基于WebGL的Revit三维建筑模型重建

陈志杨,罗　飞
(浙江工业大学 计算机科学与技术学院,浙江 杭州 310023)

摘要:结合WebGL和BIM技术,在浏览器端重建Revit三维建筑模型可以使用户方便地在浏览器上查看三维模型.针对Revit三维建筑模型全信息重建,通过Revit API将Revit Architecture三维建筑模型文件转换成包含几何信息和属性信息的JSON格式文件,并使用WebGL解析JSON文件达到在PC浏览器端和移动浏览器端实现Revit Architecture三维模型重建的目的.同时提出一种在JSON文件中模型对象和模型属性信息关联的方法,并提出了一种针对大场景,基于Revit"族"对象的LOD优化算法.

关键词:Revit;JSON;WebGL;Three.js;三维重建;多分辨率
中图分类号:TP391　　**文献标志码**:A　　**文章编号**:1006-4303(2016)06-0608-06

Revit three-dimensional building model reconstruction based on WebGL

CHEN Zhiyang, LUO Fei
(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: The Revit 3D building model reconstructed in the browser combining WebGL and BIM technology can help user browsing 3D model in the browser conveniently. This article aims to reconstruct Revit 3D building model with full information. Through Revit API, the file of Revit 3D architecture model can be converted to JSON format file with geometric information and property Information. Then the WebGL is used to parse JSON file in order to achieve Revit Architecture 3D model reconstruction in the PC browser and mobile browser. In the meantime, a associated method between model object in JSON file and attribute information associated with the model is proposed. The LOD optimization algorithm based on Revit family in big scene is put forward.

Keywords: Revit; JSON; WebGL; Three. js; three-dimensional reconstruction; LOD

建筑信息模型(Building information modeling, BIM)在建筑设计领域内被广泛讨论[1]. Revit Architecture软件是Autodesk公司针对建筑行业开发的BIM工具,在建筑领域应用广泛.虽然Revit Architecture提供了良好的建筑设计支持,但是Revit Architecture软件程序庞大,操作复杂、对计算机和使用者有较高要求,非设计人员直接通过Revit Architecture浏览三维建筑模型有一定的操作难度,还有可能不小心误操作破坏三维建筑模型的原始数据.云计算是当下计算机专业的热门话题,结合云计算的思想建立Revit Architecture私有云符合CAD发展潮流[2].计算获得三维建筑模型的几何和信息数据、存储在云端数据库,并通过网络远程访问在本地浏览器渲染三维建筑模型.相比较Autodesk

收稿日期:2016-03-05
基金项目:国家自然科学基金资助项目(61303138)
作者简介:陈志杨(1971—),男,河北秦皇岛人,教授,研究方向为计算机辅助设计(CAD)、几何造型和图形处理,E-mail:czy@zjut.edu.cn.

360 公有云、用户敏感数据存在安全隐患[3]. 通过建立私有云的方式，企业能够完全掌控自己的敏感数据，普通用户通过网络连接到私有云即可在 PC 端浏览器和移动端浏览器上浏览三维建筑模型.

将 Revit Architecture 三维建筑模型转变成私有数据存储和私有数据在浏览器端显示是构建 Revit Architecture 三维建筑模型私有云的两个重要问题. 对于第一个问题，Autodesk 公司提供了 Revit SDK (Revit Software Development Kit，Revit 软件开发工具包). SDK 中包含开发必需的 API、文档和样例. Revit API 提供了一系列命名空间、方法和类库. 方便用户在 Revit 平台上二次开发. 对于第二个问题，随着 Javascript 语言逐步完善和计算机硬件发展，浏览器脚本语言的图形图像处理能力得到加强. 基于 OpenGL 2.0 的 WebGL 技术支持在浏览器渲染二维、三维图形图像. 同时 WebGL 能够直接调用底层 GPU 显卡指令集渲染图形图像，兼容 PC 端浏览器和移动端浏览器，甚至能支持基于 Native 架构的移动端 APP 应用[4-5]. 具有跨平台性和可移植性. 笔者阐述基于 WebGL，通过 Revit API 实现 Revit Architecture 三维建筑模型重建过程. 提出一种三维建筑模型几何数据和信息数据的关联方法，同时针对大场景的渲染性能问题，设计了基于 Revit“族”的 LOD 优化算法.

1　WebGL 和 Revit Architecture 二次开发研究现状

WebGL 是基于 OpenGL ES 2.0 版本并且针对浏览器和移动端优化的底层 3D 图形 API，WebGL 版本如图 1 所示. 通过 HTML5 的 Canvas 元素作为文档对象模型接口[6].

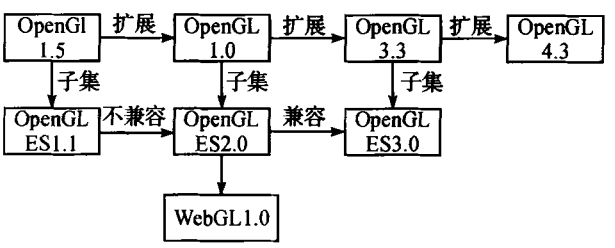


图 1　WebGL 与 OpenGL 关系

Fig. 1　Relationship between WebGL and OpenGL

虽然 WebGL 提供基于 Javascript 语言的 API，但是本质上 WebGL 是使用 GLSL (OpenGL Shading Language，OpenGL 着色语言) 的 API，开发过程和 C 语言相似. 这使得使用原生 API 实现三维图像平移、旋转和缩放等功能变得复杂枯燥. 利用 WebGL 框架能够快速开发实例程序. 流行的 WebGL 框架有 Three.js，PhiloGL，Babylon.js，SceneJS，x3dom，Copper-Licht 等，以 Three.js 图形库在开源社区最为流行.

Three.js 图形库实现了基本的图像算法；提供常见的光照模型，贴图纹理、层次模型和坐标系以及提供了像 OBJ，MTL 等格式文件的加载方法；统一的动画接口.

2002 年 Autodesk 收购 Revit Technology 公司，发布 Revit SDK，目前 Revit SDK 属于快速发展时期，不同版本存在接口变更和不兼容问题. Revit Architecture 二次开发目有商业软件和少量开源社区软件可供参考. 最著名的是 Autodesk360，提供了 Revit 云盘服务，用户将 Revit 文件上传 Autodesk 360 云服务器，服务器自动完成模型的渲染和浏览器端模型重建. 国外商业软件如 CADMAI 软件公司，开发了基于 Revit 和 WebGL 的二次开发软件，提供了本地 Revit 文件到 WebGL 文件的转换服务. 国内有北京橄榄山软件有限公司的 Revit 模块软件. 开源爱好者 Jeremy Tammik 编写的一系列 Revit BIM 开源软件和博客.

2　基于 WebGL 的 Revit 三维建筑模型重建

三维模型重建是建立私有云的核心步骤[7]. 流程包括使用 Revit API 将 Revit Architecture 三维建筑模型文件转换成 JSON 格式文件存储；使用 WebGL 解析 JSON 文件并在 PC 浏览器端和移动浏览器端实现 Revit Architecture 三维模型重建. 同时提出一种模型对象和模型属性信息关联方法以及讨论大场景下的 LOD 显示优化问题.

2.1　JSON 数据接口定义

JSON (JavaScript object notation，JavaScript 对象标记语言) 是轻量级的数据交换语言. JSON 和常用 XML 及 FSV 数据传输格式相比，在开销、传输时间以及客户端数据反序列化效率上明显优于其他数据传输格式[8]. 因而更适合作为服务器端与 Web 浏览器数据传输的接口.

JSON 格式数据接口主要使用的数据结构是对象、名/值对、数组、数字、字符串与布尔值[9]. JSON 的格式描述可以参考 RFC 4627. JSON 格式，如图 2 所示.

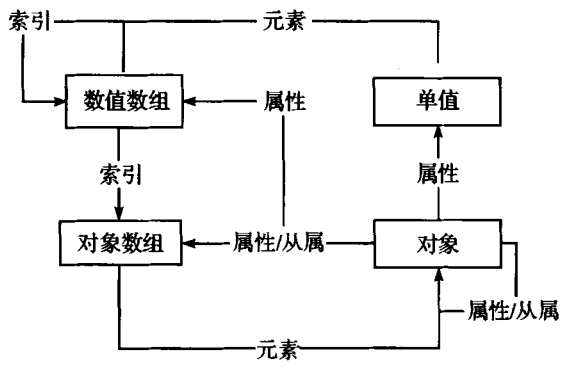


图 2 JSON 数据格式中主要数据结构以及相互关系

Fig. 2 Main data structures and the relationship of JSON data format

针对 Reivt 建筑模型设计的 JSON 文件数据结构需要符合 OBJ 模型格式. 数据结构包含四个部分:Geometries 记录几何模型数据,materials 记录几何模型的纹理数据,metadata 记录自定义信息,object 记录 Revit 对象的属性数据. 通过标识符 ID 关联对象的几何模型、纹理和参数信息. 实现了 Revit 建筑模型和属性信息关联.

2.2 Revit 二次开发

Revit SDK 支持 visual studio C# 二次开发. 所需的环境包括引入 Revit SDK,配置 Microsoft. NET Framework 等环境依赖. 核心内容包括创建新的引用、添加 RevitAPI. dll 和 RevitAPIUI. dll 包、定义类和编写代码[10],Revit 二次开发流程如图 3 所示.

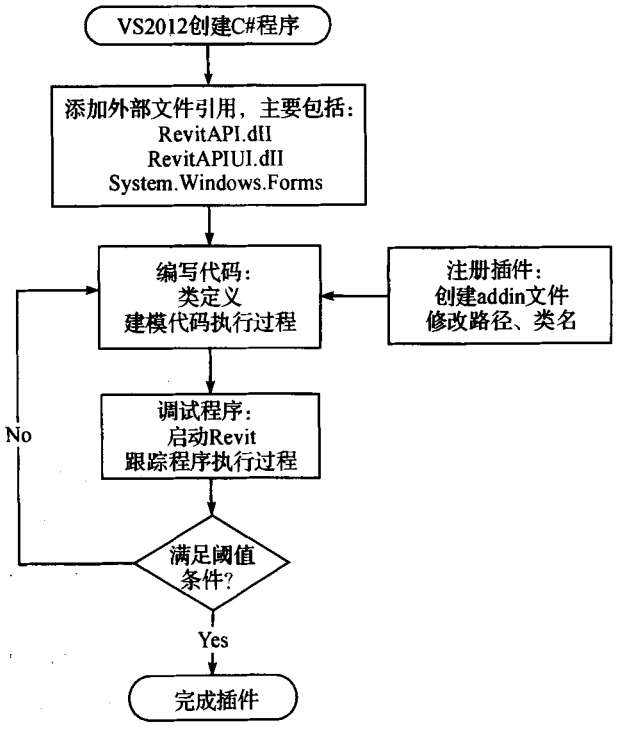


图 3 Revit 二次开发流程

Fig. 3 Revit secondary development process

Revit 标准长度单位使用英制单位,Three. js 使用国际标准单位,因此需要将英制单位转换成标准国际单位. 单位转换存在精度误差,按照惯例设置合适的阈值. 当对象长度小于 10^{-9} m 时判定长度等于 0. 由于 Revit 坐标系不是标准的 3D 计算机图形坐标系,Revit 对象在转换 JSON 存储文件过程中需要转换成标准坐标系,如图 4 所示.

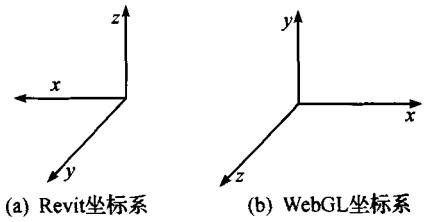


图 4 Revit 坐标系和标准计算机图形坐标系对比

Fig. 4 The comparison between Revit and standard computer coordinate system

Three. js 对象的数据结构和 Revit API 封装的 Rvt 格式不同,需要数据类型转换. 这些转换包括浮点数,点的三维坐标转换成字符串,Revit 颜色对象转换成 24 位字符串类型,Revit 对象基本属性的描述和对象 ID 与值的字典序构建. 由于 JSON 对象都是以键/值对的形式存储,Revit API 实例需要转换成字符串类型存储在 JSON 文件中.

利用 Revit API 转换三维建筑模型的核心是实现 IexportContext 接口方法. 2015 版 API 提供了 19 个方法处理三维模型对象. 这些方法用于处理 Revit 的簇对象、材料、多边形、曲面和光线. 利用这些方法将 Revit 对象转换成 OBJ 格式并存储在 JSON 文件中,Revit 转换 JSON 文件二次开发流程如图 5 所示.

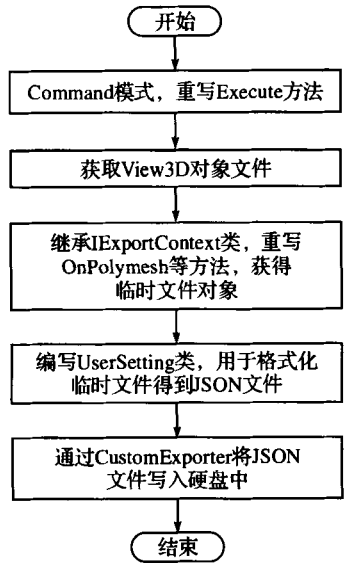


图 5 Revit 对象转换 JSON 流程

Fig. 5 Process Revit object to JSON

2.3　三维模型显示和基于 Revit“族”的 LOD 优化

Javascript 作为浏览器脚本语言，计算能力因为 google 的 V8 引擎得到了大大提高，这使得浏览器端运行 3D 程序成为可能. Three. js 是当下最流的开源 WebGL 库. Three. js 渲染器结构如图 6 所示，包括渲染器（Renderer）、相机（Camera）和场景（Scene）.

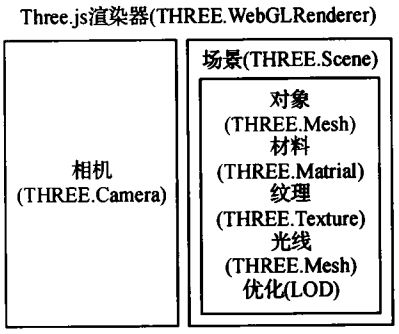


图 6　Three. js 渲染器结构

Fig. 6　The structure of Three. js Render

Revit 三维建筑模型几何信息和数据信息存储在 JSON 格式的中间文件中. Web 程序加载并解析 JSON 中间文件并解析成 OBJ 格式将解析后的几何信息和数据信息加载到 THREE. Scene 对象中、渲染器在浏览器上渲染并重建 Revit 三维建筑模型.

THREE. Scene 对象包含了建筑模型的几何信息、纹理信息以及属性信息. 利用 ObjectLoader 类解析 JSON 中间文件，分别使用 JSONLoader，MaterialLoader，Object3D 等类处理几何信息、纹理信息和属性信息. 对象返回给 THREE. Scene，显示在屏幕上. 通过统一标识符 ID 关联三种信息从而实现属性数据查询.

WebGL 动画渲染需要消耗大量的 GPU 资源，帧率（Frames per second，FPS）能够直观表达模型重建对 GPU 资源的消耗. 一般而言 30FPS 是人肉眼分辨卡顿的分界线，低于 30FPS 人观察模型旋转、缩放时能够感知卡顿. FPS 越高表明渲染能力越强，通常 WebGL 渲染上限为 60FPS. FPS 能够比较不同场景、不同 GPU 下的渲染性能区别，为性能优化提供直观动画.

Revit“族”（family）是一类具有相似性的抽象集合. Autodesk Revit 中的所有图元都是基于“族”. 每个族图元能够在其内定义多种类型，根据族创建者的设计，每种类型可以具有不同的尺寸、形状、材质设置或其他参数变量. 比如，铁门、木门和防盗门等都属于同一个“族”. 典型的现代建筑模型通常由墙面、门、窗、玻璃以及水电设施等对象组成. 同时 Revit API 也是以“族”的实例为单位处理对象.

实际测试显示，主流 GPU 显卡无法流畅地显示包含有 6 009 个“族”对象的建筑场景. 针对 Revit “族”的特点，设计以 Revit“族”为单位的 LOD 显示优化能够减少对 GPU 的资源需求.

LOD 技术的核心思想是计算对象到视点的距，如果这个距离超过了阈值则不渲染对象[11]. 通过设置合理的阈值，使得人眼观察到的图形图像和全部渲染得到的图像没有感官上的显著差别，从而取得显示效果和性能的均衡.

Revit 文件二次开发后，以“族”实例为子集顺序存储在 JSON 文件中. 对不同大小的 Revit 建筑场景自适应设置合理的阈值，遍历子集并计算“族”实例对象的包围球到相机的距离，设置对象 visible 属性，当距离小于阈值 visible 值为 true，显示并渲染，大于阈值 visible 值为 false，隐藏且不渲染. 这种基于 Reivt-three. js 对象层的 LOD 场景优化能减少对 GPU 的消耗. 算法伪代码描述如下：

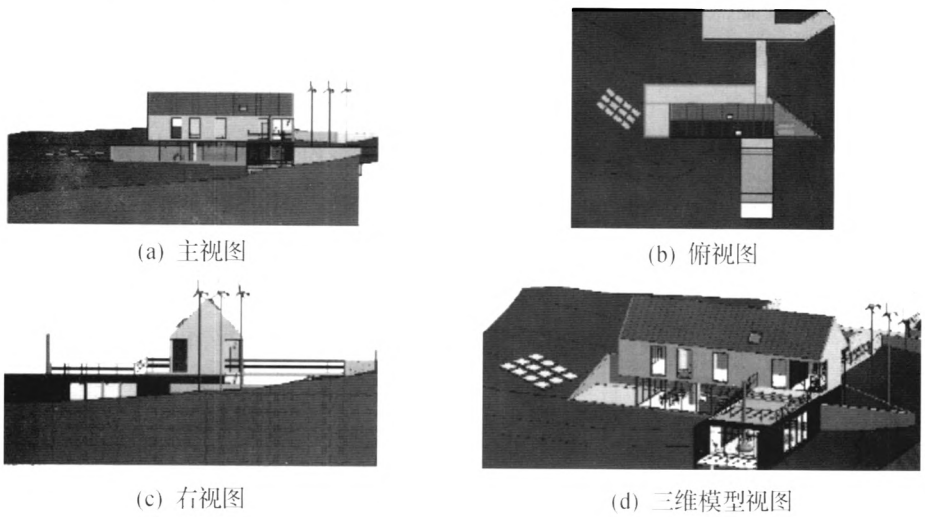
```
//camera:视点
//object: three.js 几何对象
//boundingSphere:几何对象所在的最小包围球
//radius:半径
//center,positon:点坐标信息
//thresHold:阈值
var thresHold=setThresHold();
for each object in Three.js render Array{
var distance=
calculateDistance (camera. position, object. boundingSphere. center);
var radius = getRadius (object. boundingSphere. radius);
radius / distance <= thresHold ? visible = false : visible=true;
}
```

3　系统应用

3.1　构建实例程序

按照上述程序构建实例程序. 程序以 Revit 插

件形式(Command 模式)将三维建筑模型导出成 OBJ 格式的 JSON 中间文件. JSON 文件可以在本地浏览器直接浏览,也能以 C/S 形式,浏览器从远程获得资源并显示,实验结果如图 7 所示. 图 7 为 Revit 软件绘制的一栋别墅三维建筑模型. 图 8 为在 Chrome 浏览器端重建的带有对象属性信息三维建筑模型(主视图选中对象属性信息显示在左侧信息表格里).

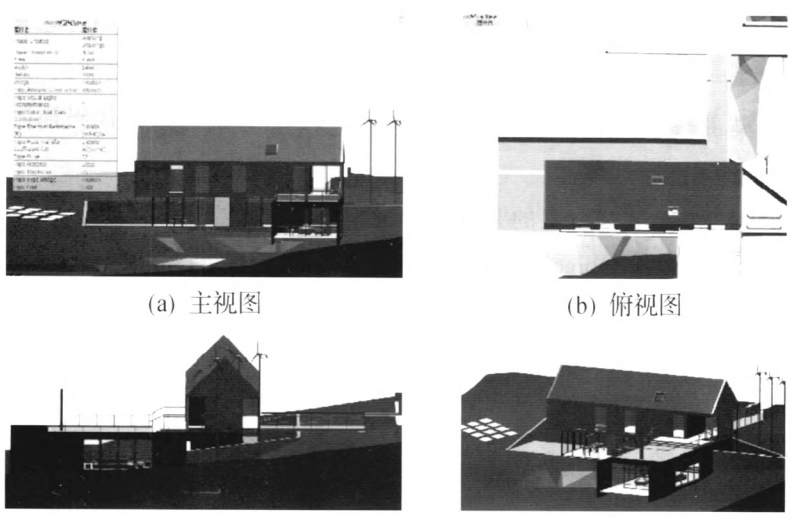


(a) 主视图 (b) 俯视图

(c) 右视图 (d) 三维模型视图

图 7 Revit 模型主视图、俯视图、右视图和三维模型视图

Fig. 7 Revit model's front view, plan view, right view and 3D model view



(a) 主视图 (b) 俯视图

(c) 右视图 (d) 三维模型视图

图 8 浏览器模型主视图、俯视图、右视图和三维模型视图

Fig. 8 Browser model's front view, plan view, right view and 3D model view

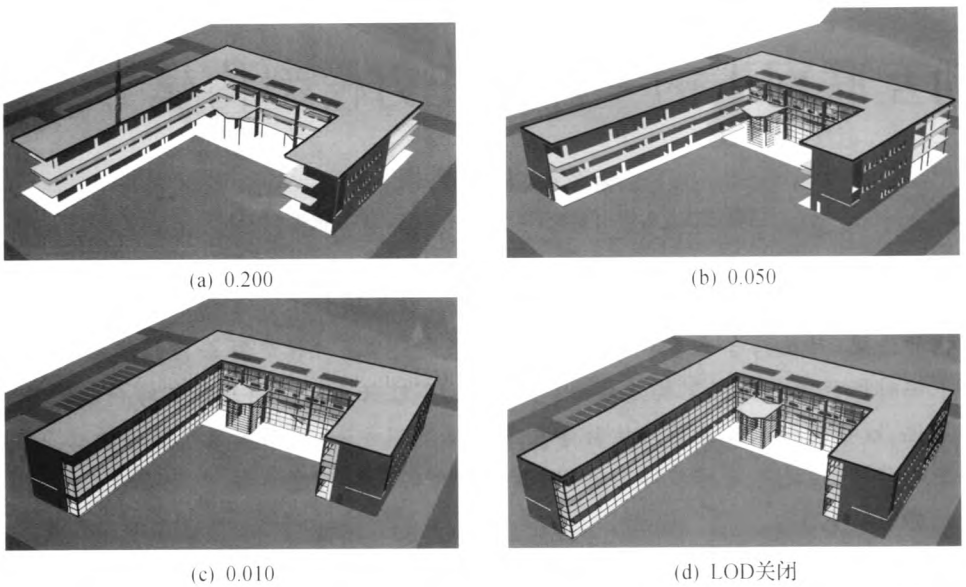
从图 8 中可以看到:在浏览器中显示的三维模型与在 Revit 中显示的模型几乎没有差异,除了光照模型不同外,仅在部分纹理贴图(地面和屋顶黑色条纹)上有些许不同,这是因为 Revit API 没有完全支持纹理输出的原因. 同时,高亮部分集合模型的属性数据在浏览器中也能够查看. 至此,我们实现了脱离 Revit 软件的三维建筑模型显示.

由于显卡资源不能流畅渲染大场景三维建筑模型. 因此需要在场景引入 2.3 节中提出的基于 Revit "族"的 LOD 技术. 以包含 6 009 个"族"实例对象的建筑场景为例,表 1 为场景在 HD6550 M 和 HD7770 M 显卡下,不同阈值的 LOD 优化数据. 图 9 为不同阈值下的 LOD 显示效果. 实验数据表明:通过设置阈值,能够提高渲染的速度,减少对 GPU 资源的消耗.

表 1 显卡和阈值对渲染数据的影响

Table 1 Effect of rending with GPU and thresholds

阈值	LOD 关闭	0.050	0.010	0.007
渲染数量/个	6 009	305	2 768	5 251
FPS_{6550}	4	14	10	6
FPS_{7770}	10	60	19	12



(a) 0.200　(b) 0.050　(c) 0.010　(d) LOD关闭

图 9　不同阈值下 LOD 显示效果

Fig. 9　LOD display under different thresholds

3.2　系统尚存缺陷

3.2.1　真实性差异

和原生 Revit 三维建筑模型场景相比，实例程序侧重于三维场景重建. 缺乏对路面贴图纹理、光线追踪和光照模型进行处理. 同时 Revit API 2015 自身对色彩输出的限制以及自定义精度造成了三维重建场景存在细节上的色差和精度损失.

3.2.2　性能优化

C/S 架构对于网络带宽有很强烈的依赖. 在保证数据完整前提下 JSON 文件越小所依赖的网络带宽越小. 主流的方法是用 GZIP 对 JSON 文件压缩传输. 2.3 节中指出 WebGL 渲染需要消耗大量的 GPU 资源，引入 LOD 场景优化虽然减少了对 GPU 资源的消耗，但是 LOD 距离的计算过程又增加了 CPU 资源的消耗. 寻找 GPU/CPU 性能均衡的显示方式是未来继续探究的新方向.

4　结　论

基于 WebGL 的 Revit 三维建筑模型重建主要解决了四个问题：提出了建立 Revit 三维建筑模型私有云的核心流程；实现了基于 OBJ 数据结构的 JSON 存储格式；提出了一种三维建筑模型几何对象和属性数据关联的方法，实现了 Revit 三维建筑模型在浏览器端呈现；提出并实现了基于 Revit“族”的 LOD 算法. 实例程序表明：在精度范围内，浏览器端渲染的三维建筑模型能够替代原生 Revit 三维建筑模型实现实时浏览功能，同时也可以快速的查询建筑对象的属性信息，权衡显示效果和性能需求，具有实际应用价值和理论价值.

参考文献：

[1] 刘照球，李云贵. 建筑信息模型的发展及其在设计中的应用[J]. 建筑科学，2009，25(1)：96-99.

[2] 林闯，苏文博，孟坤，等. 云计算安全：架构、机制与模型评价[J]. 计算机学报，2013(9)：1765-1784.

[3] 邢雪霞，黄虎，胡传皓，等. 云计算安全研究[J]. 信息通信，2014(3)：153.

[4] 方强. 基于 WebGL 的 3D 图形引擎研究与实现[D]. 合肥：安徽大学，2013.

[5] 刘爱华，韩勇，张小垒，等. 基于 WebGL 技术的网络三维可视化研究与实现[J]. 地理空间信息，2012，10(5)：79-81.

[6] CANTOR D，JONES B. WebGL beginner's guide[M]. Birmingham：Packt Publishing Ltd.，2012.

[7] 管秋，金俊杰，张剑华，等. 基于最优 RANSAC 算法的非增加式多视图三维重建[J]. 浙江工业大学学报，2015，43(4)：473-478.

[8] 高静，段会川. JSON 数据传输效率研究[J]. 计算机工程与设计，2011，32(7)：2267-2270.

[9] 陈玮，贾宗璞. 利用 JSON 降低 XML 数据冗余的研究[J]. 计算机应用与软件，2012，29(9)：188-190.

[10] 薛忠华，谢步瀛. Revit API 在空间网格结构参数化建模中的应用[J]. 计算机辅助工程，2013，22(1)：58-63.

[11] LUEBKE D P. Level of detail for 3D graphics[M]. San Francisco：Morgan Kaufmann，2003.

（责任编辑：陈石平）