基于 WebGL 的建筑大模型实时显示系统设计与实现

文/陈敏　杨阳

摘要

本文将结合其他相关研究资料，尝试设计一种基于 WebGL 的建筑大模型实时显示系统，并运用仿真实验的方式验证系统的应用性。

【关键词】WebGL 建筑模型 实时显示 系统设计

通过运用 WebGL 技术具有的优势特性，并以此为基础构建起建筑大模型实时显示系统，可以在有效实现建筑大模型实时在线显示的同时，以可视化的方式帮助建筑工程设计人员全面了解建筑设计施工效果，进而及时发现其中存在的问题并进行有效改善。因此该系统的设计不仅有助于保障建筑工程质量，同时也可以在一定程度上推动建筑行业进一步落实信息化建设。

1 基于WebGL的建筑大模型实时显示系统设计分析

1.1 系统分层设计

在本文所设计的基于 WebGL 建筑大模型实时显示系统中，采用分层设计理念对系统结构进行设计。即在采用 B/S 架构作为系统基础框架下，按照模型数据的获取、预处理、存储、显示的基本流程，将整个系统依次划分成模型数据的获取层与预处理层、模型数据的存储层与显示预处理层以及显示层。首先在模型数据的获取层中，将通过利用专业的 BIM 建模软件直接将 IFC 格式文件导出，随后在预处理层中对数据进行解析、去重以及转化为 glTF 格式等处理，并将其存储至相应的数据库中后。根据用户实际需要，将数据请求命令传输至模型数据显示预处理层中，最终由模型数据的显示层利用可视化技术对其进行直观、清晰显示。如图 1 所示。

1.2 系统主要功能

1.2.1 数据的预处理

由于来自于各 BIM 建模软件中的建筑模型数据文件格式不尽相同，因此为了使得各项建筑模型数据均可以在基于 WebGL 的建筑大模型实时显示系统中进行有机整合与统一处理，需要对数据进行相应的预处理，即对模型数据文件格式进行有效统一。在本文所设计的系统中，统一将具有良好通用性的 IFC 文件作为模型数据来源，系统在对调用的各 BIM 建模软件中的 IFC 格式数据文件进行预处理时，首先需要对 IFC 进行解析并将其中的几何数据与属性数据精准提取出来。随后将重复的几何数据筛除，并对其进行八叉树划分。此时系统只需严格依照 glTF 格式标准对模型数据进行相应转化即可。

1.2.2 在线读取数据

在线读取数据这一系统功能模块主要负责帮助前端完成模型数据的获取和解析，随后将其添加至相应的场景当中，由于建筑大模型中的模型数据量较大，因此本文在系统设计中选择使用 IndexedDB 浏览器缓存模型数据，用以确保系统在离线状态下也可以随时调用所需数据。在文件解析过程中，该功能模块将自动对 glTF 的 JSON 文件进行解析，在获得其中的结构信息后利用 JSON 文件当中所含有的二进制文件索引信息，在对应的文件当中精准提取出模型数据并将其添加至相应的场景图当中。

1.2.3 场景更新管理

为有效协调系统当中的其他模型，本文在设计基于 WebGL 的建筑大模型实时显示系统时，还专门设有场景管理功能，用于对其他系统功能模块进行灵活调度，从而完成对整个模型显示流程。在构建场景图时，系统主要运用渲染队列计算的方式获取所需装载的场景图节点列表，随后系统将在线读取数据建立与场景图相对应的节点。在更新场景时，系统首先将对 JSON 文件进行八叉树划分，在确定模型数据位于椎体内的情况下，对所需渲染的模型数据集运行 LOD 算法，并在获取需要渲染的模型数据的 LOD 映射后，精准判断 Cache 命中与否。如果命中则直接交由 Cache 模块对命中的模型数据集进行处理，并通过渲染队列的方式实时更新场景图。如果 Cache 没有命中，则需要通过系统对所需加载的模型数据集进行数据在线读取后再进行渲染队列，由此有效完成对场景图的动态更新。

1.2.4 模型数据渲染

系统在渲染模型数据时，主要通过对 WebGL 底层绘图 API 接口进行调用，绘制出渲染队列当中可渲染的基础几何图元，随后运用可视化技术将其直接显示在屏幕中即可。具体来说，需要在渲染队列初始化后，由系统负责自动获取渲染对象材质信息，同时运用专门的着色器接口并借助 GPU 进行部分计算以获取对应着色器信息，最终有效确定渲染模型对象，并调用相适宜的着色器完成模型数据渲染

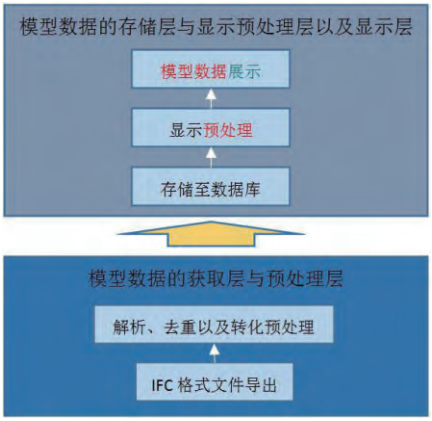

图 1：系统结构设计图

即可。系统在设计渲染模型数据功能时，选择将 Three.js 当中的 renderer 渲染器引入其中，这主要是由于该引擎拥有众多可供 Web 端使用的渲染器，能够有效满足 WebGL 渲染要求。

1.2.5 模型数据缓存

由于建立在 WebGL 的建筑大模型实时显示系统中涉及的模型数据量众多，因此系统需要专门设计出可以有效缓存海量的模型数据的功能，以快速完成模型数据读取与传输工作，同时避免在流量器段占用过多内存。为此，本文通过结合相关研究资料，选择使用专业的压缩算法对建筑大模型进行压缩处理，在有效防止模型过大的基础上，引入多级缓存机制以达到有效缓存庞大模型数据量的效果。数据在线读取模块在正确写入 IndexedDB 之后，位于前端内存当中的系统缓存模块将负责对未来可能加入至渲染队列当中的模型数据进行有效预测，并直接将其加载至相应的内存中，进而可以为系统处理提供有效便利。在运用已有的多级缓存算法时，鉴于系统中的模型数据经常因摄像机缩放或旋转而发生变动，因此为实现动态模型数据的实时缓存与有效显示，本文还选择将模型权重值加入到多级缓存算法中。即使用如下所示的公式：

$$Q=W(L_1+L_2)$$

在该公式当中，Q 代表模型权重值，W 为模型在当前可视范围内与否，L1 与 L2 分别代表模型数据与摄像机的间距，以及模型数据不在可视范围内时，其与可视范围最小欧式距离之间的间距。一般情况下，工作人员可以根据 LOD 算法分层范围确定 L1 的值，如果模型数据在可视范围内，则可以将 L2 值设置为 0。在这一系统模型数据缓存功能中，模型数据权重值越大，其将会被优先置换出系统缓存空间。

2 基于WebGL的建筑大模型实时显示系统

实验分析

2.1 实验准备

为有效验证本文所设计的基于 WebGL 的建筑大模型实时显示系统的应用成效，本文选用 Chrome 与 Firefox 浏览器负责完成系统试验运行。在 BIM 模型数据选择上，采用专业 BIM 技术公司提供的大规模建筑 BIM 技术模型。其中 A 建筑模型大小在 930MB 左右，拥有超过 2330 万个模型顶点数，模型的三角面片数与几何单元数分别约为 778 万个与 79800 个，模型转换后大小为 296MB。B 建筑模型大小则接近 2.3GB，拥有大约 1.5 亿个模型顶点数，模型的三角面片数与几何单元数分别约为 4900 万个与 16.4 万个，模型转换后大小为 448MB。所选模型如图 2、图 3 所示。

2.2 实验结果

根据相关实验结果显示，在实验平台中分别渲染两个数据集，得到 A 建筑模型数据集的渲染帧率在 28 到 35fps 之间，B 建筑模型数据集的渲染帧率在 16 到 21fps 之间。而渲染帧率的大小可以在一定程度上体现出系统流畅性，当渲染帧率相对较高的情况下，系统中画面之间可以实现无缝跳转，用户能够获得较好的使用体验。在本实验平台中，无论面对小规模还是大规模的建筑模型，系统均可以有效实现流畅漫游浏览。而对于 A 建筑模型数据集而言，系统只需 12s 便可以完成渲染操作，由于 B 数据集规模相对较大，因此系统渲染时间相对较长，但通过从若干面中选取所需渲染部分并直接将其送至渲染队列中，使得渲染系统压力得到相应减小，因此在 110s 左右系统便可以完成大规模数据集的渲染。而如果在条件允许的情况下，运用高性能的计算机与浏览器，则可以将所占内存降至最小，即便在渲染时需要进行大量数据传输，也不会消耗过多时间而影响系统渲染成效。不同模型渲染性能对比如图 4 所示。

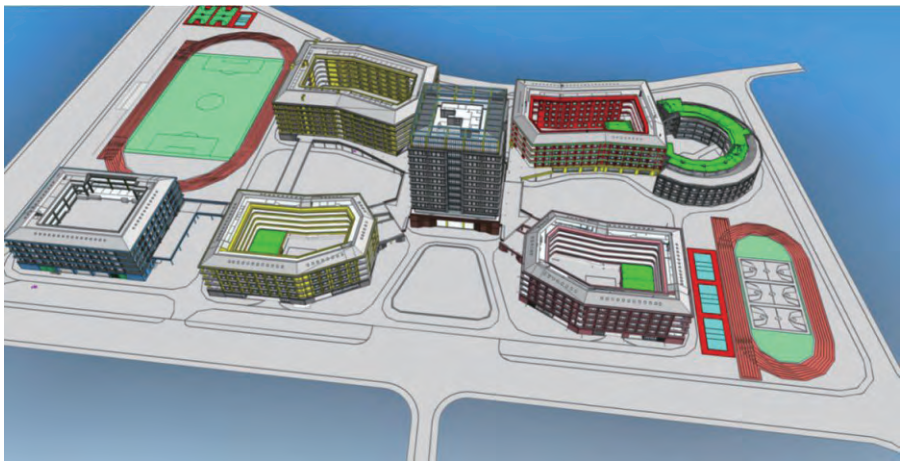

图 2：A 模型显示图

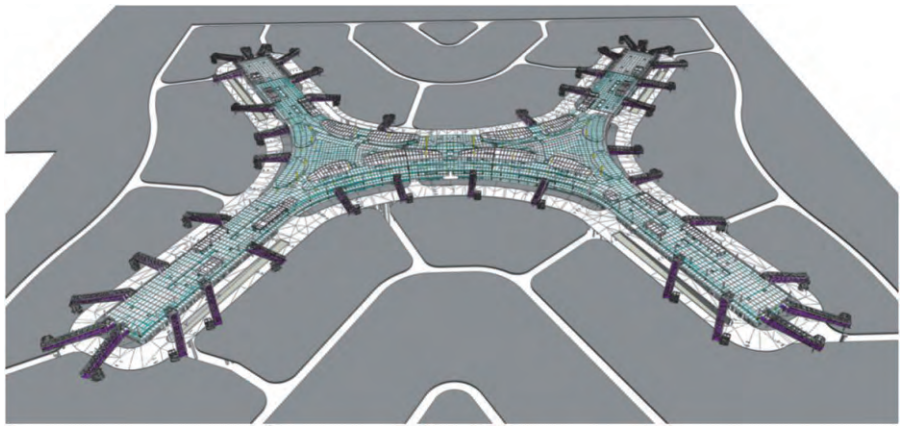

图 3：B 模型显示图

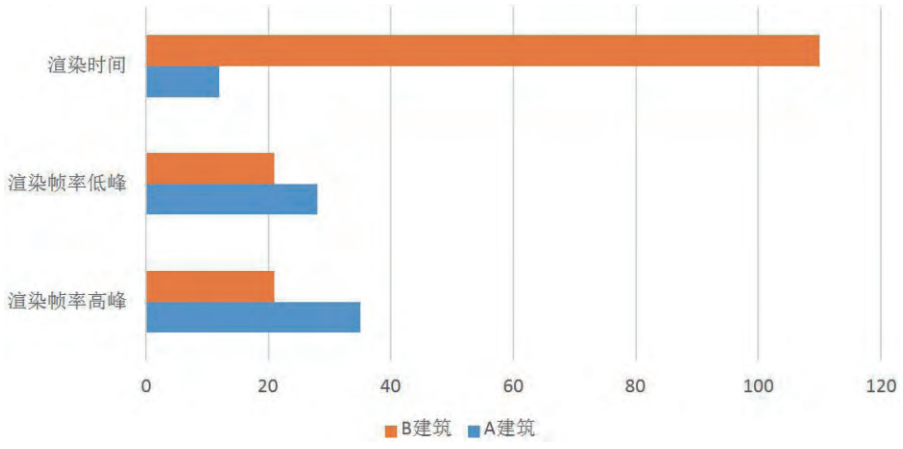

图 4：模型渲染性能对比

3 结束语

总而言之，本文通过将八叉树与 Three.js 等运用在基于 WebGL 的建筑大模型实时显示系统中，围绕 BIM 建筑模型数据量较大的特点，在设计使用 IndexedDB 浏览器缓存模型数据的同时，对系统显示场景进行八叉树管理，同时使用 WebGL 技术当中的着色器技术以及 Three.js 当中的渲染组件渲染数据集，可以实现建筑大模型的实时在线显示，便于相关工作人员随时浏览、查看建筑模型。

参考文献

[1]KhronosGroup.glTF–Runtime 3D Asset Delivery[EB/OL]].https://github.com/KhronosGroup/glTF/blob/master/README.md,2019.

[2] 唐路明．基于 Web GL 的大规模 3D 重建场景渲染系统的设计与实现 [D]．华南理工大学，2016.

[3] 利广杰．基于 OSG 的大规模建筑模型渲染系统设计与实现 [D]．华南理工大学，2017.

作者简介

陈敏（1981-），男，浙江省杭州市人。大学本科学历。高级工程师。研究方向为企业信息化管理，企业流程引擎集成及开发、工程数字化等。

作者单位

中国电建集团华东勘测设计研究院有限公司
浙江省杭州市　311122