王晶

武汉理工大学

武汉理工大学

（申请工学硕士学位论文）

WebGL 三维桥梁动态仿真关键技术研究

培 养 单 位：计算机科学与技术学院

学 科 专 业：计算机应用技术

研　究　生：王晶

指 导 老 师：夏红霞　教授

2014 年 4 月

分类号_____ 学校代码 10497

UDC _____ 学号 104972101432

武汉理工大学

学 位 论 文

题 目 WebGL 三维桥梁动态仿真关键技术研究

英 文
题 目 Research On Key Technology Of WebGL 3D Bridge Dynamic Simulation

研究生姓名 王晶

指导教师 姓 名 夏红霞 职称 教授 学位 _____
单位名称 计算机科学与技术学院 邮编 430070

副指导教师 姓 名 邹承明 职称 教授
单位名称 计算机科学与技术学院 邮编 430070

申请学位级别 硕 士 学科专业名称 计算机应用技术

论文提交日期 2014 年 4 月 论文答辩日期 2014 年 5 月

学位授予单位 武汉理工大学 学位授予日期 _____

答辩委员会主席 邹承明 评阅人 袁景凌
岑丽

2014 年 4 月

独 创 性 声 明

本人声明，所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得武汉理工大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名：_____日 期：_____

学位论文使用授权书

本人完全了解武汉理工大学有关保留、使用学位论文的规定，即学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人承诺所提交的学位论文（含电子学位论文）为答辩后经修改的最终定稿学位论文，并授权武汉理工大学可以将本学位论文的全部内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存或汇编本学位论文。同时授权经武汉理工大学认可的国家有关机构或论文数据库使用或收录本学位论文，并向社会公众提供信息服务。

（保密的论文在解密后应遵守此规定）

研究生（签名）：　　导师（签名）：　　日期

摘　要

随着我国交通建设的迅猛发展，新桥梁建设与旧桥梁运营管理都面临着更艰巨的挑战，桥梁健康监测管理系统作为保障桥梁结构运营安全的重要手段被广泛的运用。但是，桥梁检测如何在新兴技术的发展和推动下，更准确的识别桥梁损伤，并以科学、直观的方式分析和预警，实现桥梁运营管理的标准化、自动化、网络化和可视化已成为国内外学术界的热点与难点。

本文深入研究了国内外桥梁健康监测与三维可视化技术的发展趋势，提出以 WebGL、HTML5 等前沿技术结合实际桥梁结构特点与运营管理需求，重点设计与开发了基于 Web 的桥梁健康监测可视化系统，实现了监测数据在 Web 上的高效、直观的呈现及预警分析，具有用户体验好、开发成本低、平台通用性强等优势。本文的主要研究工作与创新点概述如下：

第一，详细阐述了桥梁检测管理系统的发展趋势，深入探索 WebGL 技术的三维 GPU 绘制原理、跨平台开放性及其技术框架生态，并评测出最适宜于 Web 桥梁 3D 可视化实现的开发框架。

第二，结合相关技术框架，针对于实际桥梁检测管理的实际需求与目标，设计出桥梁健康监测视景仿真系统构架、模块功能、用户接口的最优方案。

第三，研究了基于 WebGL 的桥梁健康监测可视化系统的关键开发技术与实现。首先，结合桥梁实际结构与环境特点，利用 3D MAX 设计具有真实感的桥梁场景，并对场景模型优化，以便更利于 Web 环境的桥梁场景渲染和实现。然后，利用 WebGL Three.js 框架将桥梁场景在 Web 环境中重构渲染，着色优化。并根据 Three.js 框架动画及交互原理，实现了桥梁视景的漫游，及桥梁模型交互控制，为进一步实现监测预警，病害查询打下基础。最后，提出了利用 LOD 技术对大型桥梁场景的高效优化方案，进一步提升了系统的效率和可用性。

第四，以襄阳江汉三桥为例，概况的介绍了桥梁的结构状况，并结合本文的设计方案与技术原理将系统成功实践验证，总体达到了直观标记、分析数据的效果。

最后，分析和总结了本文的主要工作及未来工作的方向。在系统的进一步优化后，将会有广阔的应用前景，并对于桥梁检测可视化的研究具有一定的参考价值。

关键词：桥梁健康监测, HTML5, WebGL

Abstract

With the rapid development of China's transportation construction, construction of new bridges and operation management of the old bridges are faced with more challenges. Bridge health monitoring and management system is widely used as an important tool to ensure safe operation of the bridge structure. However, with the development and promotion of new technologies, how can bridge inspection accurately identify bridge damage, and make analysis and early warning of it to realize the standardization, automation, networking and visualization of bridge operations. The above has become a hot and difficult subject in academia at home and abroad.

This article studied the development trend of bridge health monitoring and 3D visualization technology at home and abroad, focusing on the design and development of Web-based bridge health monitoring visualization system by using WebGL, HTML5 and other frontier technologies combined with the actual characteristics of bridge structure and operational management needs. It achieves efficient and intuitive monitoring data presented on the web and warning analysis, and has good user experience, low development cost, platform versatility and other advantages. The main research work and innovation of this paper is summarized as follows:

First, this study elaborated on the development trend of bridge inspection management system, deeply explored three-dimensional rendering GPU principle of WebGL technology, cross-platform framework for openness and eco-technology, and evaluated of a web development framework best suited to achieve 3D visualization bridge.

Second, combined with relevant technical framework, the study designed bridge health monitoring visual simulation system, module function, and the optimal solution of the user interface according to the management of the actual bridge inspection actual needs and goals.

Third, it studied the development and implementation based on key technologies visualization of bridge health monitoring system of WebGL. At first, combined with

the actual structure of the bridge and environmental characteristics, the use of 3D MAX bridge scene with realistic design, and optimization scenario model to the Web environment can be more conducive to the bridge scene rendering and implementation. Then using WebGL Three.js framework will help to bridge scene reconstruction rendered in a Web environment, coloring optimization. In accordance with the Three.js framework animation and interactivity principle, a visual bridge of roaming, interactive control and bridge models can be achieved, which lays the foundation for achieving the monitoring, early warning and diseases inquiry. In addition, it offers the use of technology for large bridge scene LOD efficiently optimization program to further enhance the usability and efficiency of the system.

Fourth, it took Xiangyang No.3 Jianghan Bridge for example to describe the overview of the structural condition of the bridge, combined with design and technical principles of this article to verify the successful practice of the system, and has overall achieved effect visual markers and analysis of data

Finally, the study analyzed and summarized the direction of the main work and the direction of future work. After further optimization of the system, it will have a broad application prospect, and has a certain reference value for bridge inspection visualization research.

Keywords: Bridge Health Monitoring, HTML5,WebGL

目　录

第 1 章　绪论

1.1 课题研究的背景、目的与意义

当今，我国正处于城市化建设飞速发展的跃进时期，道路桥梁数量日趋庞大。据统计，截止到 2013 年，我国公路桥梁以达到七十余万座，新桥梁建设与旧桥梁运营管理都面临着更新更大的挑战，相关的道路桥梁管养部门也迫切的需要一种直观、高效的桥梁管理手段[1]。同时，计算机相关软硬件的高速发展也不断迎合变革的契机，监测方式已经从早期单机模式的上下位机通信，逐渐发展为以传感器信号采集节点为主的传感器网络模式，数据方面从早期的信息独享已经发展为基于互联网的网络数据共享，数据信息模式也在不断的向多元化、复杂化发展。尤其，在视景仿真技术的演化和推动之下，基于真实桥梁的设计数据，利用计算机重构为具有相似环境与场景的仿真桥梁模型，即可以对所建立的模型进行实时的视景漫游，亦可以对模型场景实现信息的交互。用户根据需求通过外设控制视角的位置与方向，可以实时查看相关传感器监测数据对桥梁的影响。桥梁的可视化提供了最真实、动态的途径，使用户更加深入的理解数据信息，更有效的指引桥梁信息的有效共享与检索[2] [3]。

在国内，桥梁视景仿真系统的研究一直在发展中，许多研究机构与人员都在不断的积极尝试。例如，北京公路一局开发的拱桥可视化软件系统，可根据用户需要进行各类桥梁相关的交互设计与三维仿真[4]。武汉道路桥梁管理中心委任武汉理工光科公司开发的长江大桥、二桥、白沙洲大桥等桥梁的数字仿真系统，对相应桥梁实现 Web 上的桥梁模型仿真、监测数据管理，并进一步通过其桥梁上传感器的拓扑布局采集实时环境数据，实现数据的共享，通过算法模型的数据分析，进而达到桥梁环境监测与预警等目的。

仿真技术在 PC 上的应用已非常成熟，伴随着 Web2.0 时代的到来，网络三维（Web3D）日渐成为新的发展热点，其良好的实时互动性、网络数据共享、跨软硬件平台壁垒等特点令其也成为道路桥梁仿真的必然趋势。通过 Web3D 技术，不但可以在网络上建立可互动的虚拟世界，突破空间与地域的限制，为用户带来新的体验，并且满足了传感器拓扑传输的实时信息精准的再现与预警分析的新要求，避免了以往的安装成本，跨越了软硬件壁垒[5]。

实际道路桥梁的运营期间，其承载能力、运营状态、耐久力等方面都需要这样的仿真系统，为桥梁的养护管理提供可靠的依据与帮助。本课题对于 Web3D 的桥梁动态仿真关键技术进行了相关研究，提出结合 HTML5、WebGL 等新兴技术，建立一套适合长远需要的桥梁安全监测可视化系统，开发出稳定、可靠的，具有实时监测、数据采集、分析、预警、健康评估等功能，并且操作简单支持跨平台网络远程访问的 B/S 桥梁动态仿真系统。具有相当广阔的应用前景与实际的经济效益。

1.2 桥梁管理系统的发展现状与趋势

现代桥梁管理系统（Bridge Management System）的发展始于 20 世纪 80 年代后期的美国 Pontis 系统，根据桥梁的构件退化分析和承载负荷退化分析等数学模型，对桥梁运营承载力和缺陷做出评估，根据结果依据相应标准制定对策，并同期利用预算资金分析模型优化项目维护工程。随后，日本的 MACHI 桥管系统，丹麦的 DANBRO 桥管系统也不断应运而生，由桥梁病害监测及状态评估模块、桥梁构建退化预测模块、工程预算模块、维护管养模块等部分组成，极大的提高了桥梁管理及养护的效率。

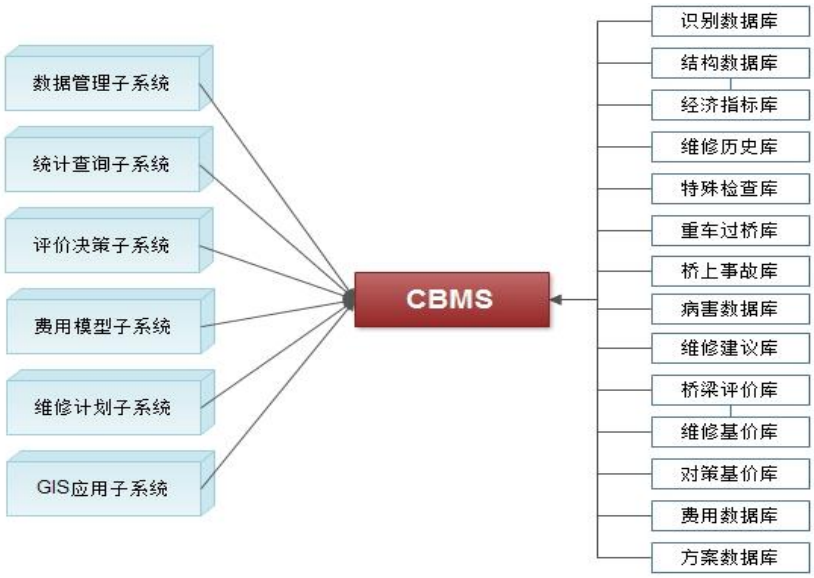


图 1-1 我国路桥管理系统（CBMS）

我国也于 20 世纪 90 年代初由交通部属的公路科学研究院率先开始了中国路桥管理系统（CBMS）的研究，系统基于桥梁结构工程为基础，广泛应用了病害极力监测、数据采集统计、工程经济学数据库系统，对现有桥梁基础档案进行

管理。城市桥梁管理系统也在 CBMS 的标准与基础之下得以发展，以北京市公路管理局、广东省桥梁研究所等为代表设计了多个各具特色的桥梁管理系统。早期的桥梁管理系统虽然都提供了数据管理、信息查询、决策评估等子模块，但缺乏完整的桥梁三维可视化子系统，无法为决策部门和管理机构以直观形象的桥梁仿真模型以满足决策需要[6]。

以武汉市为例，武汉市路桥管理中心承担了武汉长江大桥、长江二桥、白沙洲大桥、晴川桥、月湖桥、长丰桥等六座桥梁的管理工作。为构建“采集数据、分析数据、信息流转、预警提示”的智能化管养工作模式，不断完善其桥梁数字化管养系统，以人工检测结合数字化检测相结合，基于桥梁互动三维内容技术及物联网技术设计智慧桥梁云平台[7]。在现有的桥梁管理系统的基础之上，将系统直观呈现在三维仿真的桥梁之上，操作简易，面向不仅是技术人员还有桥梁维护者及决策者。同时，基于 B/S 构架的系统全面实现综合信息服务的集成，满足需求，促进优化整合，避免和消除信息孤岛，建立一个应用整合统一、标准化的技术平台。

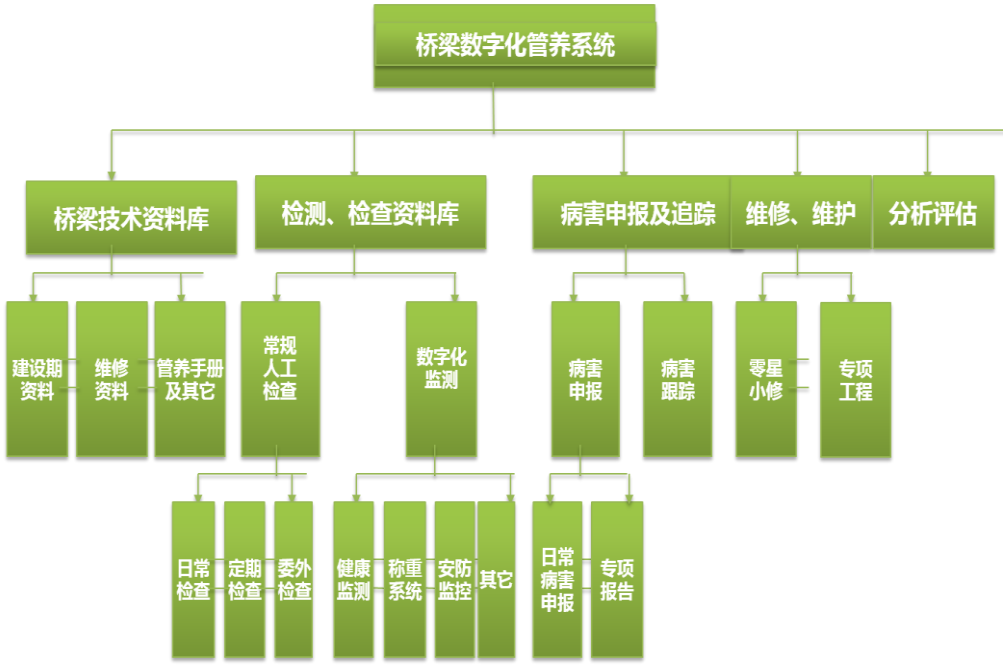


图 1-2 武汉桥梁数字化管养系统

相比于以往桥梁管理系统，基于 Web 的桥梁互动三维技术内容作为一个显示前端，具有显著的优势 ：

(1) 以 Web 浏览器为系统媒介，桥梁仿真作为系统的重要部分呈现 720 °场景全方位展示。

(2) 以设计图纸与视景照片为桥梁模型的设计依据，材质更具有真实感，以实物仿真替代传统的三维应用。

(3) 桥梁管理者可以根据需要自由控制桥梁漫游视角，按需实时查看桥梁的各个部件其桥面、拉索、各传感监测点的细节信息以及桥梁的整体结构。

(4) 将原有的桥梁传感自动检测方式与人工巡检方式更有机的结合，增强了检测数据的可读性与高效用，极大地方便了桥梁监管部门通过 Web 对桥梁健康状况的实时监控。

未来，桥梁管理系统还将会在传感器、Web3D、物联网等技术的不断创新下，脱离现有壁垒，多平台无缝信息采集、分析，图像化交互操作，大量减少使用者技术门槛，提升系统友好性，极大减少系统人力物力成本。

1.3Web3D 国内外研究现状与趋势

图 1-3 Web3D 发展趋势分析图

传统仿真技术的研究已取得卓著的成绩，有大量技术成熟的视景仿真软件如 VTree、MultiGen、OpenGVS、Vega 等，提供了高效的仿真解决方案。然而网络时代对于数据共享的要求和 PC 端软件安装的时间成本和运营成本已经使这些成熟的技术与平台无法满足现代高速变革的挑战和要求。

而基于 Web 的仿真还处于一种很不成熟的状态，因为很难在典型的 PC 和浏览器上广泛使用这种复杂的技术[8]。由于浏览器自身限制以及 Web 传输等问题，基于 Web 端的仿真技术目前在承载复杂 3D 内容时还很难在显示效果与帧率方

面与传统仿真相比。另外，对于需要实时通信的应用程序中包含三维内容也会让开发过程变的更加复杂。

尽管如此，依然有很多组织和厂商在为网络 3D 技术而努力，试将浏览器集成更多原生功能，以便在 Web 平台上得到如 PC 端应用的体验度。而众多的技术方案也应运而生。包括：VRML、X3D、Viewpoint、Cult3D、Shock—wave3D、Java3D、Unity3D、Virtools 等[9]。

以 Java3D 结合 VRML 的组合方式为例，从 1997 年发展至今已有广泛的应用基础，这种基础解决了 Web 传输问题、并且解决了三维可视化的平台无关性问题，具有非常强大的交互能力。但是，这种方式对 JVM 的依赖使内存占用率更高，速度较慢，其画面也非常粗糙[10]。

传统的 Web3D 解决方案有两大瓶颈，即依赖 Adobe Flash、SliverLight 等插件，以及很难支持 Web 端的 GPU 加速，尤其无法胜任大规模复杂 3D 场景的渲染[11][12]。3D 插件以 Flash 的使用最为广泛，目前市面上超过 75%的网页游戏都基于该技术，插件扩展虽然扩展了浏览器的功能，使网页游戏的性能得以提高，但同时也增加了插件安装的麻烦以及潜在的安全隐患[13]。另外，在 PC 与智能手机等移动设备中 GPU 的应用已经非常普及，其计算能力也可与 CPU 相媲美，但是实际应用却多数局限于 3D 游戏。这些因素都阻碍了 Web3D 的发展，使 3D 模型很难在 Web 网上自由发布，访问和检索[14]。

2009 年 8 月，Khronos 提出了 WebGL 绘图技术，它是基于 OpenGLES2．0 标准的一个跨平台、开源、用于在 Web 浏览器创建三维图形的 API。在 HTML5 的新特性 Canvas API 中，WebGL 可以直接的绘制 3D 动画并且提供 GPU 的渲染加速。Canvas 规范定义了 2D 的绘制环境界面 CanvasRenderingContext2D。另一界面 WebGLRenderingContext，提供 WebGL 的 API, 这些 API 可以在 Web 浏览器中通过 JavaScript 控制[15]。利用 WebGL 实现 Web3D 不需要安装浏览器插件，只需编写 JavaScript 代码即可轻松实现 3D 图像的展示甚至是动画交互[16]。

国内外研究人员也依然在不断的开拓 WebGL 的生态帝国，例如：MIT 提出的 WebGUI，旨在分离渲染问题，利用如优化的着色器开发接口等方法，简化代码复杂度和实施过程[17]。北京航空航天大学提出的 NetGL 框架，优化了资源加载、空间渲染，克服了网络环境中的 3D 应用限制，如不良的运算能力、文件访问限制、多线程等问题[18]。同时，C3DL、CopperLicht 、EnergizeGL、GammaJS、GLGE、GTW、O3D、OSGJS、SceneJS、SpiderGL、TDL、Three.js 和 X3DOM 等 WebGL 构架和支持库也日趋成熟。

WebGL 在应用方面，很多研究机构都已开始了积极的尝试。例如，我国山东大学利用 X3DOM 结合 HTML5、WebGL 等技术，开发出了 Web 电力智能模拟系统[19] [20]。基于 WebGL 的 GLGE 框架已经尝试应用于 3D 虚拟实验室[21]，以及台湾国立大学的结合云计算实现的建筑信息三维模型[22]。Three.js 作为目前应用较为广泛的框架，国内外已经有越来越多的开源教程和资料可供学习，并成功实现了诸多应用实例。在国外，尤其是医学领域，应用 Three.js 实现了大量的 Web 医学仿真应用[22]。同时，国内也有很多网页游戏正在借助 Three.js 第三方库，将原来的 flash 插件抛弃，将应用程序移植到新的 WebGL 平台上来。但是，基于 WebGL 的桥梁仿真相关研究依然属于萌芽阶段，依然有待于进一步的研究和应用。

1.4 本文研究内容

针对桥梁管理系统在互动三维技术方面的特点和发展趋势，提出一套 B/S 构架的桥梁动态仿真管理平台。利用 HTML5、WebGL 等前沿技术，结合流行技术框架，对桥梁健康监测可视化管理系统进行设计开发。将 3D MAX 模型移植到 Web 平台上，并实现桥梁仿真模型在 Web 上、跨平台的信息实时查询、录入及其他互动操作。结合桥梁灾害管理基础，进一步实现桥梁健康监测在此 Web 仿真系统上的数据呈现、数据分析及灾害预警功能。降低系统开发成本、节约开发时间，同时保证系统的可扩展性及可维护性。主要研究工作如下:

(1) 研究现有桥梁管理互动可视化及健康监测的问题，深入了解 Web3D 的前沿技术和发展趋势。

(2) 研究 WebGL 关键技术，深入探究其渲染性能、GPU 渲染原理及框架生态等方面。

(3) 提炼出一套桥梁健康监测可视化仿真系统的构架及功能模块等最优设计方案。

(4) 利用 3D MAX 设计桥梁真实模型，并优化移植到 WebGL 框架中，实现桥梁模型的 Web 重构，并进一步研究桥梁仿真系统的信息动态交互。

(5) 针对于大型桥梁高片面数在 Web 上的高效渲染问题，结合 WebGL 技术，给出相关优化方案。

(6) 以襄阳江汉三桥为例，并结合本文设计方案与 WebGL 关键技术原理实践验证系统效果。

1.5 论文组织结构

本文章节安排如下：

第一章 绪论：论述现代桥梁可视化检测管理系统的研究的背景，以及本题国内外的相关研究现状及趋势。并进一步介绍了 Web3D 技术在桥梁仿真方面的应用前景及优势。最后，对本文的研究的内容及论文的总体结构框架进行概述。

第二章 关键技术分析与系统方案设计：针对本文研究相关的技术 HTML5、WebGL 进行简介，测评及研究 WebGL 在 Web3D 中的优势、GPU 渲染原理及其中间件生态。多维度综合比较，选取最合适本系统的技术框架。从桥梁健康监测可视化系统实际需求入手，对该系统的功能模块、系统构架、人机接口进行概要设计。

第三章 桥梁健康监测可视化系统关键技术实现：研究和配置 WebGL 软硬件环境。将桥梁模型在 3D MAX 建模软件中设计出来，并优化移植到到 WebGL 环境中，实现桥梁场景模型的 Web 端重构。优化桥梁多片面的管线加速渲染，环境渲染，材质渲染，并进一步研究桥梁视景的漫游与交互，以实现桥梁检测数据的查询、录入及预警。

第四章 襄阳江汉三桥桥梁健康监测可视化系统：利用该实例验证系统设计方案及技术方案，展示系统的运行效果。.

第五章 总结与展望：对本课题的工作进行总结概括，并针对系统的不足提出未来的工作重点。

1.6 本章小结

本章介绍了课题相关的研究背景、意义。针对于 Web3D 及现代桥梁管理系统国内外研究现状与趋势，进一步论述 WebGL 技术应用于桥梁管理监测可视化系统的优越性及必要性。最后，简要概述了课题研究的主要内容及论文整体结构框架。

第 2 章　关键技术分析与系统方案设计

2.1 关键技术分析

2.1.1 HTML5 技术

HTML5 是由 WHATWG 等组织创立的下一代 HTML 标准，专门针对于 Web 应用开发新功能，W3C 组织于 2012 年发布了 HTML5 规范[24]。HTML5 并非颠覆性革命，其很大程度上保持了一切新特性的平滑过渡。为避免不必要的复杂性，HTML5 做出许多改进，如利用 Web 浏览器的原生能力代替外部插件及复杂的脚本，不但简化了 DOCTYPE 和字符申明集，其最重要的创新是其简单而强大的 HTML5 API[25]。

HTML5 众多创新 API 包括：Canvas（2D 和 3D）、Channel 消息传送、Geolocation、SVG(Scalable Vector Graphics)、Web Socket API 及协议等，极大的丰富了对音频、视频、绘图的支持，增加了 Web socket API 通信接口、Geolocation API 地理信息接口、及 Web Storage API 离线存储接口等。过去 Web 上的许多功能只能通过另外安装第三方插件或者复杂的脚本来实现，但这种方式存在很多问题，插件安装可能失败或者被禁用，同时也极易成为被攻击的对象[26]。HTML5 添加了许多脚本与布局的原生交互能力，可以实现以前不能实现的功能。

本文将会重点介绍以及在本系统中实际应用 WebGLCanvas、WebSockets API 及本地存储 Web Storage 技术。

- Canvas API

Canvas 的概念最早由苹果公司提出，用于 Mac OS X WebKit 中创建控制板部件，早期浏览器中使用绘图 API 只能用 Adobe Flash、SVG 等插件。HTML5 Canvas 提供了 Web 网页编程绘画环境，它定义了二维的绘图环境界面 CanvasRenderingContext2D，通过调用简单函数完成二维图像的绘制、变换、渲染等操作。同时在 WebGLRenderingContext 界面中，提供 WebGL 的 API, 这些 API 都在 Web 浏览器中通过 JavaScript 控制。

- HTML5 WebSockets API

HTML5 WebSockets 是 HTML5 中最强大的通信功能。它定义了一个全双工通信信道，只需要通过 Web 上的 Socket 即可进行通信，对于实时事件驱动的应

用而言，它代表着一次巨大的进步。传统的 Web 实时应用的实现大部分基于轮询和服务器的推送技术，其涉及到 HTTP 请求与相应报头中有大量不必要的数据，会造成传输延迟，或大量的资源耗费。HTML5 WebSockets API 在半双工的 HTTP 基础上模拟全双工通信，构建一个采用发布/阅读模式来利用后端数据源显示实时数据的 Web 程序。客户端和服务器建立 WebSocket 通信，第一次握手时 HTTP 协议会被升级为 WebSocket 协议，建立连接后，即可在全双工模式下进行实时消息通信[27]。

- HTML5 Web Storage

HTML5 本地存储 Web Storage（DOMStorage）优化了跨请求重复访问数据的旧模式，可以实现 Web 请求数据持久化存储在客户端浏览器的功能。Web Storage 提供两种方法 SessionStorage 和 LocalStorage。SessionStorage 可以在用户浏览器窗口关闭前存储数据到本地，且只在构建窗口或标签页使用。LocalStorage 数据生命周期在关闭窗口后依然可以延续，并且可以被同源的每个窗口和标签页共享。更重要的是，相较于以前 cookie 几 K 的存储量，Web Storage 可以保存高达数兆的大数据[28]。

2.1.2 WebGL 技术分析

2009 年 8 月 Khronos 提出针对 Web 上 3D 图像 API——WebGL。它是基于 OpenGLES2．0 标准的一个跨平台、免费的、用于在 Web 浏览器创建三维图形的 API。WebGL 可以通过 JavaScript、浏览器以及 Web 协议栈直接在 HTML5 的 Canvas 元素中绘制三维动画，利用计算机显卡进行 3D 加速渲染。不同于以往用户只能依赖于插件或本地应用下载来达到 3D 体验，利用 WebGL 只需要编写简单代码即可实现 3D 展示[29]。虽然 WebGL 并不在 HTML5 官方规范中，但是大多数支持 HTML5 的浏览器已经开始支持这一技术，不仅仅在 PC 的 Web 端，甚至是手机端也开始提供相应的支持。

WebGL 的优势主要有两点：良好的开放性，脱离插件支持多系统多浏览器。渲染管线机制调用 GPU。

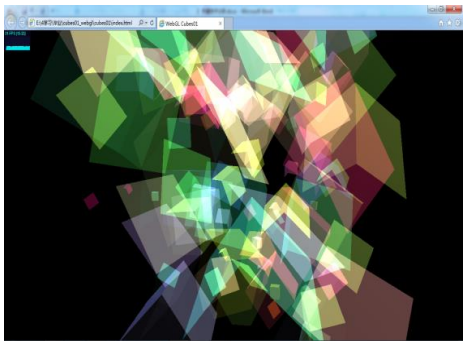
2.1.2.1 WebGL 开放性实验

虽然，目前主流的 3D 插件 Adobe FlashPlayer 与 Silver light 都可以支持 GPU 加速，但它们都是第三方插件，开放性较差，使用环节复杂，而 HTML5 的开放标准为 PC 和移动终端摆脱 IOS、安卓、WIN OS 等系统原生态应用的垄断与限制，提供了更多适用于智能手机等移动设备的特性，支持轻量级 3D 应用的 Web

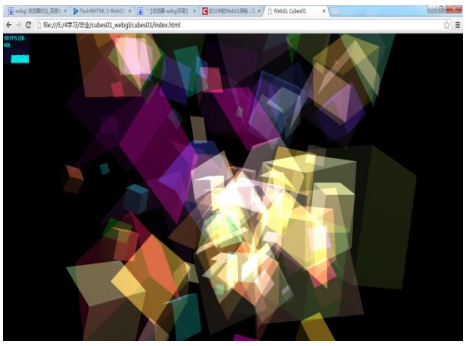
实现 [23]。主流的浏览器中 Chrome10 以上版本、Firefox4.0 以上版本、Opera12、IE11、Safari5.1 以上版本都可以很好的支持 WebGL。

以下实验将在动画场景中创建出 FPS(Frames per Second 每秒传输帧数)计数器，以检验动画的性能和平稳性。较高的 FPS 意味着较好的性能，通常 LCD 的更新时间是 60 帧每秒，即可视为最优值。

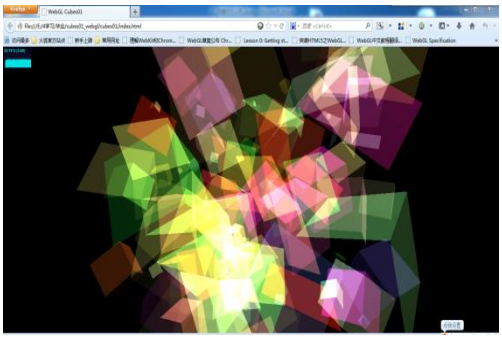
测试 Demo1 是利用 Three.js Geomotry 搭建的 100 个透明材质立方体，对场景布设两点光源，设置循环路径动画。监测 Demo 分别运行在 firfox、chrome 及 ie11 中的渲染性能。



A. IE11



B.Chrome

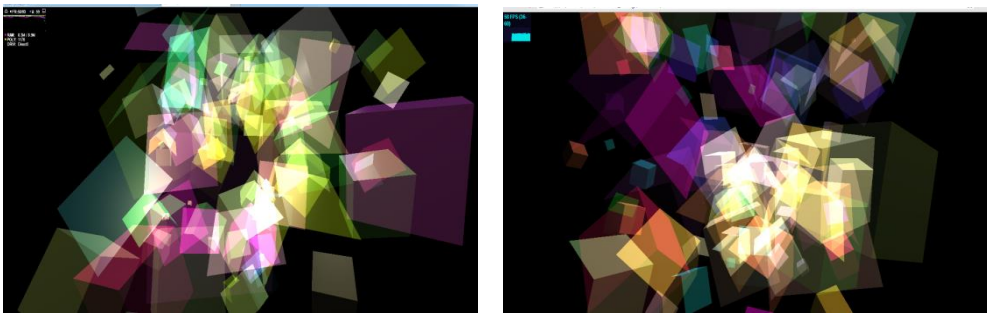


C.Firefox

图 2-1 IE11 与 Chrome 与 Firefox 对 WebGL 支持测试

Demo1 测试中 FPS 对比结果为，新版的 Chrome 与 Firefox 均达到 60fps 的高帧率，而 IE11 为 30fps 相对较低。实验结果表明，主流浏览器 IE11 和新版的 Firefox 和 Chrome 都可以很好的支持 WebGL。其中，最新 E11 采用以 DirectX 11 为基础的渲染器，改进了原有 JavaScript 引擎，成为首个支持 WebGLrander 的 IE 浏览器。Chrome 和 Firefox 对 WebGL 的支持非常好，其渲染效果和流畅度达到很高标准，完全满足将 3D 应用系统 Web 化的要求。

Demo2，将 Stage3D 构架的 Flash3D 与 WebGL 做对比测试。



A. Flash 3D　　B.WebGL

图 2-2 Stage3D 与 WebGL 测试对比

Demo2 测试中，相较于 WebGL，Flash 3D 的初始化更慢，在显示效果和流畅度方面 WebGL 与 Stage3D 效果相当，均可以达到 60fps 高帧率，但是 Flash 3D 在 Mac 上依赖于 OpenGL，而在 Windows 下依赖 DirectX，需要安装三方浏览器插件，并对显卡要求更严格。就此来说，WebGL 解放了硬件壁垒，并且有良好的性能，优势明显，更适宜于 Web3D 的发展趋势。

2.1.2.2 WebGL 的 GPU 渲染管线原理

WebGL 是一个低级的 API，可利用底层的 GPU 加速功能实现图元的绘制和场景渲染，OpenGL 上下文调用可直接驱动程序的图形硬件，通过渲染管线（Rendering Pipeline）的处理流程绘制[30] [31]。

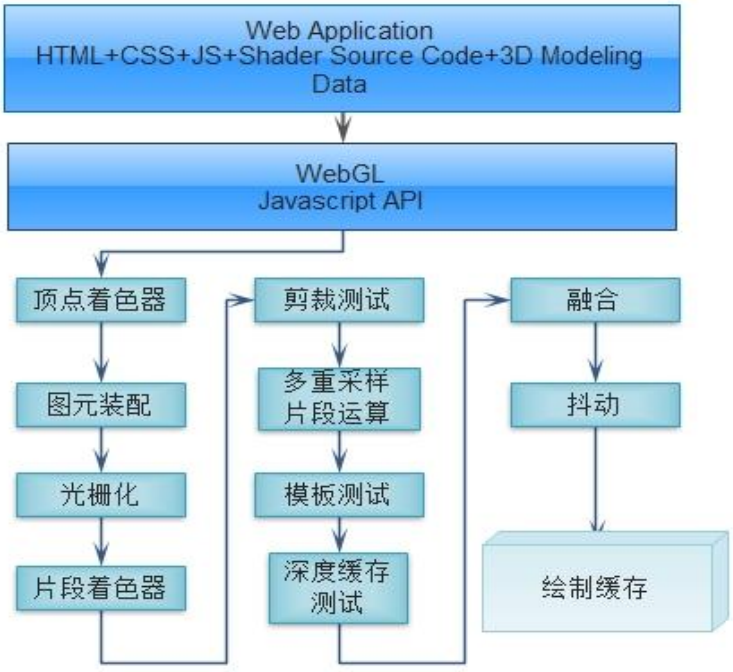


图 2-3 WebGL 渲染管线结构

WebGL 的图形 API 支持 CPU 与 GPU 并行构架。一般 Web 应用程序运行在 CPU 上，并使用 RAM 主存。而现实中 3D 图形的绘制过程需要不断调用底层的驱动程序，应用程序将图形数据发送到 GPU。通常 GPU 采用流水线结构将数据从一个阶段传递到下一阶段，通过流水线传送的绘制信息最终写入到帧缓存（framebuffer）中。帧缓存是保存屏幕所有信息的存贮器，它可以是主存的一部分，也可以是 GPU 的单独存储芯片，并由颜色缓存（color buffer）、Z-缓存（Z-buffer）及模板缓存（stencil buffer）组成。

基于 WebGL 的 Web 应用程序上层数据通常使用除了 HTML、JavaScript 及 CSS 等代码还包括着色器（shader）代码和表示 3D 对象的模型数据等信息，在浏览器中运行。通过 JavaScript 调用 WebGL API，并把模型的绘制信息传送给 WebGL Rendering Pipeline，图形数据通过 GPU 管线后写入到绘制缓存（drawing buffer）中，再传输到物理帧缓存，与页面 HTML DOM 组合显示到屏幕上。

着色器包括顶点着色器与片源着色器，都以一种专用的 GPU 友好语言 GL SL 编写而成。顶点数组（Vertex Array）包含 Attribute 属性变量和 Uniform 恒值变量，如顶点的空间位置、纹理信息、颜色信息和法线信息等。顶点数组信息被传输到 GPU 端的顶点缓冲（Vertex Buffer）中。GPU 利用顶点着色器读取顶点信息和属性，并将顶点信息乘以一个变换矩阵并输出一个新的属性集合。图元装配判断着色的顶点装配成的三角形片面、线段及点等几何图元是否在视锥体中。然后光栅器（Rasterizer）会切出这些三角形并忽略形状之外的其他部分，将当前剩余的可见部分打碎，填充到像素大小的片元 （Fragment）中。对于其他顶点属性（比如颜色和纹理），顶点着色器会在光栅化之后的三角形表面上，为一个顶点和另一个顶点之间的部分做线性插值，为每一个片元（也就是像素）产生一个平滑的渐变值。产生的片元再被传入片段着色器（Fragment Shader）中。逐片段操作将片段进行剪裁测试、多重采样片段操作、深度缓存测试等，以绘制缓存中的各个像素。为支持 GPU 同时运行多个着色器，着色器作为一次处理一个顶点或一个像素的回调（Draw Call）存在，GPU 就可以随意在任何一个 GPU 执行单元以任何顺序来运行着色器。

渲染管线机制促使 Web 开发人员利用 GPU 在浏览器中展示复杂的 3D 模型场景，并应用于数据可视化、网页 3D 游戏等方面。

2.1.2.3 WebGL 框架生态研究及对比

原生的 WebGL API 非常底层，这保证了 WebGL 充分的灵活性和适用于所有应用的可能性，但其对开发者的图形图像的基础及 GPU 底层渲染知识背景要

求非常高。目前，有许多的第三方开发框架与图形库，包括 C3DL、CopperLicht 、EnergizeGL、GammaJS、GLGE、GTW、O3D、OSG.JS、SceneJS、SpiderGL、TDL、Three.js 和 X3DOM 等[32]。许多基于 WebGL 可快速绘制 3D 元素，包括场景、相机、光源、环境光、内置集合体的形状材质、纹理及粒子效果等。另外因为 WebGL 很少支持动画与场景交互，所以许多框架也同时封装了交互事件接口，这比原生的 WebGL API 更具便捷性。国内外针对于 WebGL 框架的研究还在不断的探索中。目前具有代表性的 WebGL 框架有以下几种：

- GLGE：

GLGE 是一个基于 WebGL 的 JavaScript 图像引擎，极大增强了 WebGL 使用的友好性。它的 Browser JavaScript API，可被直接访问，允许在没有下载任何插件的情况下使用 2D/3D 硬件加速应用，支持快速创建 WebGL 应用。对于外部 3D 模型文件格式支持的很好，比如可以完美支持 COLLADA[33]。

- PhiloGL：

PhiloGL 是由 Sencha 实验室开发的一个新的 WebGL 开源框架，提供了强大的 API，可帮助开发者轻松将 WebGL 整合到 Web 应用中，很方便的实现数据可视化游戏开发。其内置的静态方法 PhiloGL.unpack()可将 3D 模型放入全局空间。绘图时使用的是 Canvas 标签，可以简单地制作出基本图元构建，并内含立体回转和渐层效果等渲染效果函数。PhilloGL 提供详细的 API 文档，详细描述了其图元、渲染方式等方法和接口，减少了开发者的学习成本。

- CubicVR:

CubicVR 是基于面对对象方法的 OpenGL2.0 及 OpenGL ES 3D 高性能引擎，从 C++导出成 Javascript。CubicVR 包含基本图元、数学库等，结合数组映射对象作为参数，代码结构清晰高效很少使用 set 与 get 方法。框架也支持各类动画，其自定义渲染器 shader 可有效提高 render buffer 的性能，内置多种光源和光照效果，并支持场景实体的可选天空盒。同时，CubicVR 也有非常好的文档和使用教程，以方便开发者学习。应用方面，Mazila Lab 的 Gladius 游戏引擎中便使用了这一框架[34]。

- Three.js:

Three.js 是由西班牙开发者 MR.Doob 利用 JavaScript 编写的 WebGL 框架，该项目的目标是创建一个开源、复杂度底、轻量级的 3D 库，用最简单、直观的方式封装 WebGL 中的常用方法，其代码托管在 Github 上面，并且有很多例子和小工具方便使用和学习。Three.js 将 WebGL 原生 API 的细节抽象，场景拆分为网格、材质、光源、摄影机、动作等部分,还有许多使用的内置对象方便开发

游戏和动画。Three.js 包含了如 WebGL、Canvas、SVG 等多种渲染模式，适用于不同的开发需求。Three.js 弥补了 WebGL 本身不提供的拾取（picking）功能，更方便 3D 场景的交互操作。另外，这个框架同样提供了多种方法支持包括 Blender，Maya，3D MAX 等设计工具的模型格式。目前，国内外有越来越多的开源教程和资料可供学习，也开始广泛应用于国内外的 Web3D 开发中。国外，尤其在医学领域，利用 Three.js 框架实现了大量的 WEB 医学仿真应用。国内，也有很多网页游戏正在借助这第三方库，将原来的 flash 插件抛弃，移植到新的 WebGL 平台上来[36]。

上述框架都可以很好的实现 Web3D 的绘制、渲染、动画甚至是互动。然而在选择框架时需要考量的不仅是他的特性和功能，因为任何额外的特性与功能都需要额外的开销。本文从体积、功能、源代码，支持四个维度去考量和选择应用的框架。

首先，体积对于 Web 上的应用来说影响到其运行速度和效率，是非常重要的考虑因素。虽然快餐式开发解决方案可以快速实现系统目标，但同时需要巨大的开销为代价，较小体积的开发框架也意味着更大的自由度、灵活性和拓展性。同时考虑到开发调试过程，最好使用 Google Closur、Firebug 此类工具中查看和维护。就此来说 Three.js 的 JavaScript 框架代码相对于 CubicVR C++框架代码更具优势。并且，可以根据用户需要选择 Three.js 需要的渲染、控制库文件，通常只需要几百 K。

第二，在本项目中，需要在 3D MAX 等专业建模工具中设计桥梁模型并移植到 Web 中，同时实现多种复杂模型交互工作。GLGE 和 Three.js 两种框架都支持了外部模型导入的功能，GLGE 可完美支持 COLLADA 模型的重构，Three.js 不但可以支持 3D MAX、MAYA、Blender 等主流建模工具的设计格式，甚至有很多外部研究机构也开始设计出更优化、专属于 Web 读取的模型格式都可应用于 Three.js 中。开发过程，GLGE 更加方便和快速，Three.js 具有更大的自由度和扩展性。

第三，速度对于渲染一个大片面的场景来说至关重要，需要将模型转化为最佳结构并且以优化算法高效利用 GPU 渲染模型场景。

第四，框架是否为开源，是否提供源代码，这些框架也都是开源的，很方便的下载到框架，在开源协议的前提下可以免费应用。

最后，框架是否有一个强大的开发社区或团队提供支持，有相关的研究人员和学术专家对框架进行更新和维护，使其不断适应 WebGL 和浏览器的发展。Three.js 框架在众多框架中几乎是最受欢迎的，其开源代码有许多人共同维护并

不断丰富其功能.在国内也逐渐开始有社区、论坛开始研究 Three.js 框架，相较于其他框架来说 Three.js 框架对图形图像的知识背景要求也较为底。

综合考虑，本系统可使用 Three.js 框架实现桥梁模型的 Web 功能开发。

2.2 桥梁健康监测可视化系统设计目标

为配合桥梁运营期健康与安全监测的需要，加强桥梁维护管理，保障桥梁结构的承载能力、耐久性能以延长桥梁的使用寿命。促进桥面行车安全，加强桥梁状况的日常性、周期性检查及检查信息的数字化工作，建立、健全桥梁状况实时技术档案，使大桥及其附属设施的技术状态便于可视化显示和分析，必须开发一套科学有效的可视化数字化桥梁系统[37] [38]。

本文以桥梁健康监测数字可视化系统软件需求分析，总结系统任务目标为：在 Web 环境中，无需浏览器插件支持情况下，通过本系统可实现复杂结构桥梁在网络环境中的视景三维重现。实时对桥梁结构、附属设施及环境数据等信息进行可视化展示与视景漫游。对于桥梁的整体信息、部件信息、桥梁病害信息、维护信息进行交互，实现相关信息综合分析、查询、录入、预警等功能。简化桥梁的管养工作，提高桥梁参数、监测维修记录查询效率，使桥梁管养部门全面掌握桥梁健康状况并合理安排维护策略的总体目标。

2.3 桥梁健康监测可视化系统设计

2.3.1 系统设计构架

桥梁健康监测数字可视化系统基于 B/S 模式，以网络作为通信平台，利用 HTML5、WebGL 技术、通信 Socket、数据采集等技术实现目标。无需三方插件，应用服务可以在 PC、手机、平板各类终端以及 Windows、MAC 各类系统中高效运行。

系统总体结构分为三层：表现层、业务逻辑层、数据层。表现层直接面向用户访问，界面设计要点为：充分展示三维动画的直观效果，以桥梁 3D 场景为主要呈现接口，简化菜单布局，启用弹出式菜单和响应式弹出窗口，设计风格图像化、3D 化。业务逻辑层监听访问信息和用户请求，并向数据层请求相应数据。它是系统框架的关键部分，主要负责具体操作，三维场景的绘制、控制，数据交互与更新等。数据层接受请求并向上反馈数据，数据层包括基本的数据库、

桥梁模型数据、桥梁传感器反馈的数据文件以及综合报表样式集，数据库在建立单独服务器上，数据库采用轻量级开源数据库 SQLite。

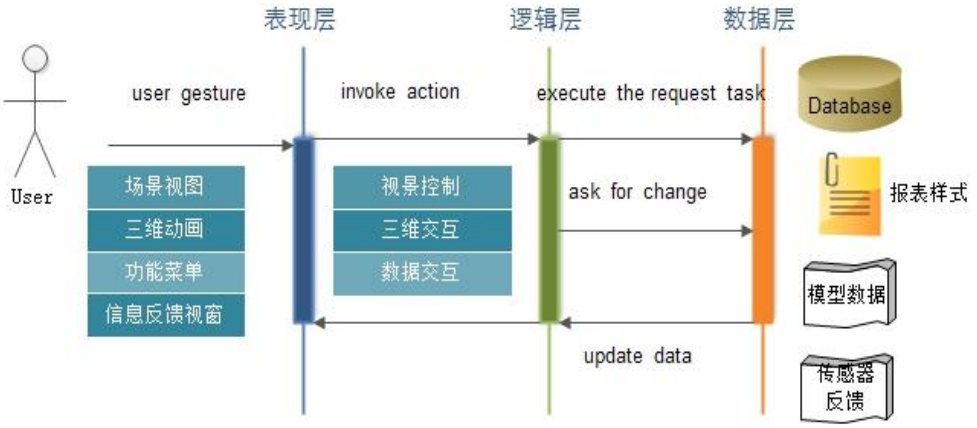


图 2-4　系统三层结构图

系统构架中，应用服务器介于数据库与客户端之间，各个客户端实例直接与应用服务器通信，数据库服务器与各传感器服务器相独立。这种多服务器构建出高性能集群，每个客户端可以有多个实例，便于系统实现更好的扩展性，例如外设的扩容等。应用服务器与各客户端通 TCP/IP 协议通信，数据库与应用服务器之间通过 SQLite 访问接口连接。

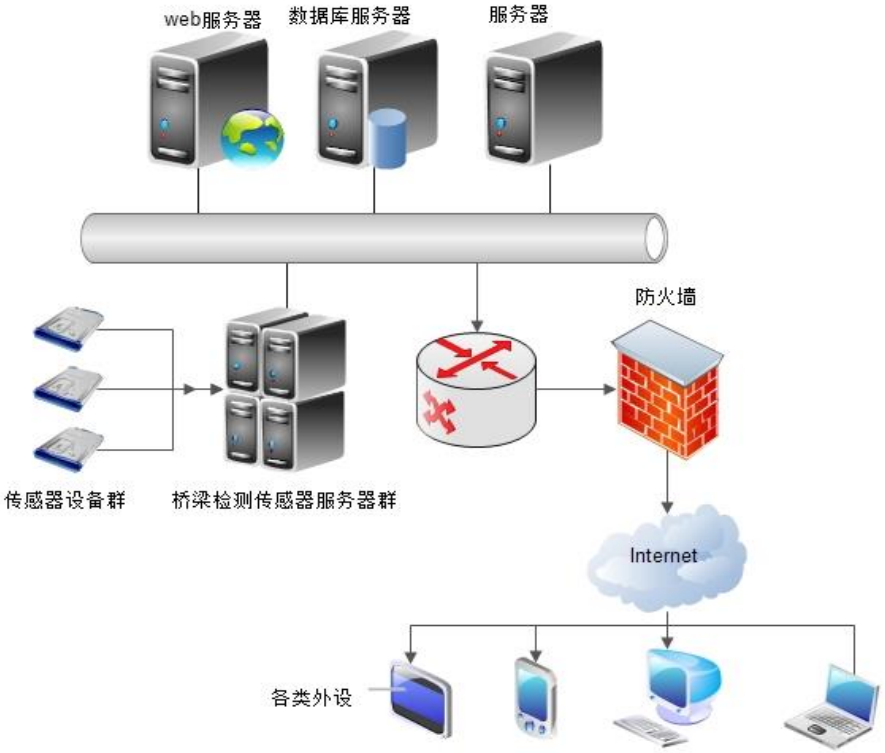


图 2-5 系统构架图

2.3.2 系统功能设计

桥梁健康监测可视化系统功能模块可划分为三维桥梁重建与漫游子模块、桥梁信息档案管理子模块、桥梁病害巡查子模块、巡检信息实时录入子模块、桥梁病害预警子模块、综合报表管理子模块。

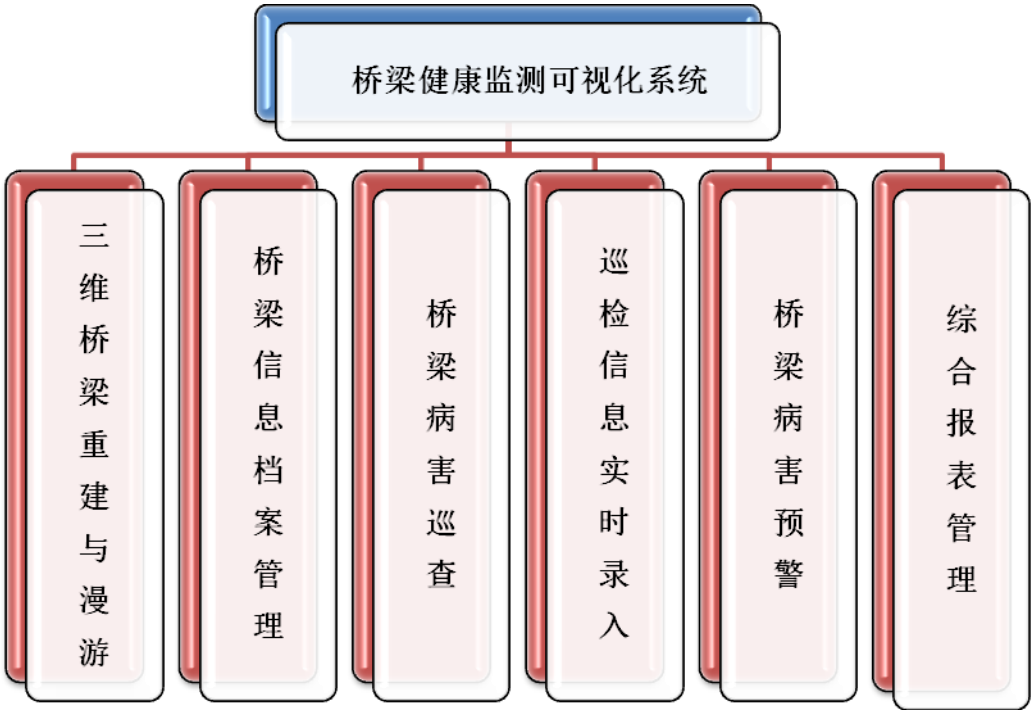


图 2-6 桥梁健康监测数字可视化系统功能结构图

- 三维桥梁重建与漫游

桥梁三维仿真系统中，鉴于桥梁结构的复杂性，通常需要利用专业的三维建模软件来制作。本文利用专业的三维建模软件 3D MAX，参考实际桥梁结构图纸和实景照片建立模型，使模型与桥梁对应比例关系。Web 上的桥梁三维重建将利用基于 WebGL 的 Three.js 框架实现，通过将 3D MAX 的模型文件.max 借助工具脚本转化为 Three.js 框架可读取格式，重构 Mash 网格数据及材质信息，载入和渲染网格模型，优化渲染速度和渲染效果。对场景进行着色，并调试灯光、视口位置及渲染器。借助 GPU 渲染实现桥梁的模型在 Web 浏览器内的三维重建。

通过键盘或鼠标的操作控制三维视角，支持场景的漫游、旋转、拾取等操作。全桥视景漫游，改变三维视口和移动步伐，实现缩放漫游和桥梁整体仰视、侧视和俯视多角度旋转漫游。实现局部构件的漫游查看，在视景中对构件的特征

精细化显示，将数据库相应病害实时局部渲染显示。同时，可以在漫游同时通过鼠标拾取局部 Mash 信息并与数据库交互，实现查看构件的相关参数如构件的尺寸、材质以及病害记录和维修记录等信息，其所包含的所有控件能够接收到信息并正确渲染，从而丰富交互界面。

桥梁的可视化重建与漫游模块作为系统各个子模块的基础与接口存在。首先，利用三维模型交互充分了解桥梁健康监测传感器布局与其对应点的实时数据状况。在系统运行时，各监测点预装的桥梁温度、索力、变形检测传感器传输的信息通过系统后台分析生成监测的数据图表，再与模型预设的相应构件观测点相关联。第二，在桥梁实地巡检对桥梁灾害的记录与查询过程中，可充分利用桥梁模型实时、直观标记或巡视问题部件。第三，可视化模块也作为系统的人机接口界面而存在，用户进入系统后通过对桥梁视角的操控来提升系统整体的友好性。

桥梁 3D 模型数据在整个系统的传输信息量巨大、而模型交互中对模型数据的解析和请求将严重影响系统流畅度，也是决定系统效率的关键因素之一。为此，设计过程中将重点考虑到存储优化、模型优化、渲染优化等方面。存储优化方面，系统考虑将模型文件利用 LocalStorage 存储到本地,在系统 3D 在 Web 内重建和交互时利用本地资源。模型优化方面，将模型数据以最优化结构组织与读取。渲染优化方面，考虑使用 LOD 优化方法，根据场景景深分层高效渲染。多方提高可视化应用的实际效用。

- 桥梁信息管理子模块

对桥梁档案、巡检信息、桥梁健康及维修信息进行查询和管理。桥梁档案包括桥梁基本数据、桥梁各部分构件信息、重要图纸等信息的电子文档的查询和预览等。实现桥梁巡检信息管理，可查询和录入所有检测项目，根据检测需求，设置巡检周期和巡检计划的查询和审核巡检记录。实现桥梁检修信息管理，可查询和录入桥梁检修相关信息，检修计划，检修目标，检修结果，检修周期等。本模块主要承担桥梁信息管理 OA 的功能模块，整合在可视化系统中方便用户实时的查询、添加和管理。

- 桥梁健康预警子模块

桥梁预警模块是通过全桥 8 个墩截面上的进场传感器、光纤光栅应变传感器、光纤光栅温度传感器得到的数据通过结构计算和分析计算的结构状态根据识别和判别算法判断是否超出安全阀值。超出阀值后弹出报警状态，并在桥梁三维场景中实时显示。局部渲染快速定位桥梁灾害位置，通过渲染颜色效果判断警报级别，令相关桥梁管理部门可以快速定位桥梁灾害。

预警报警模块设为顶级界面，并具有最高显示优先级，一旦预警或报警，立即弹出报警事件窗口，显示报警的时间、测点、传感器、实时值、报警级别限设定、报警次数，应答与否等信息。

- 桥梁信息录入子模块

实现人工巡检移动端的桥梁病害信息实时录入，根据实际桥梁对应的模型部件上标记问题，记录桥梁巡检问题描述，并且使用移动端照相功能和录音功能记录桥梁问题图片、声音等多媒体信息。

本文中 WebGL 相较于其他的 Web3D 方法来说，最大的优势在于其开放性优势，无插件，脱离系统和终端局限。系统不但可以应用到 PC 机中作为桥梁可视化健康监测系统实现信息的管理工作，另外它也可以在巡检过程中应用在移动端，作为人工巡检的可视化媒介存在。比之于以往的移动端巡检应用，实时、实地录入桥梁灾害的信息及图片。本系统有跨平台节省开发成本、可视化界面监测直观等优势。

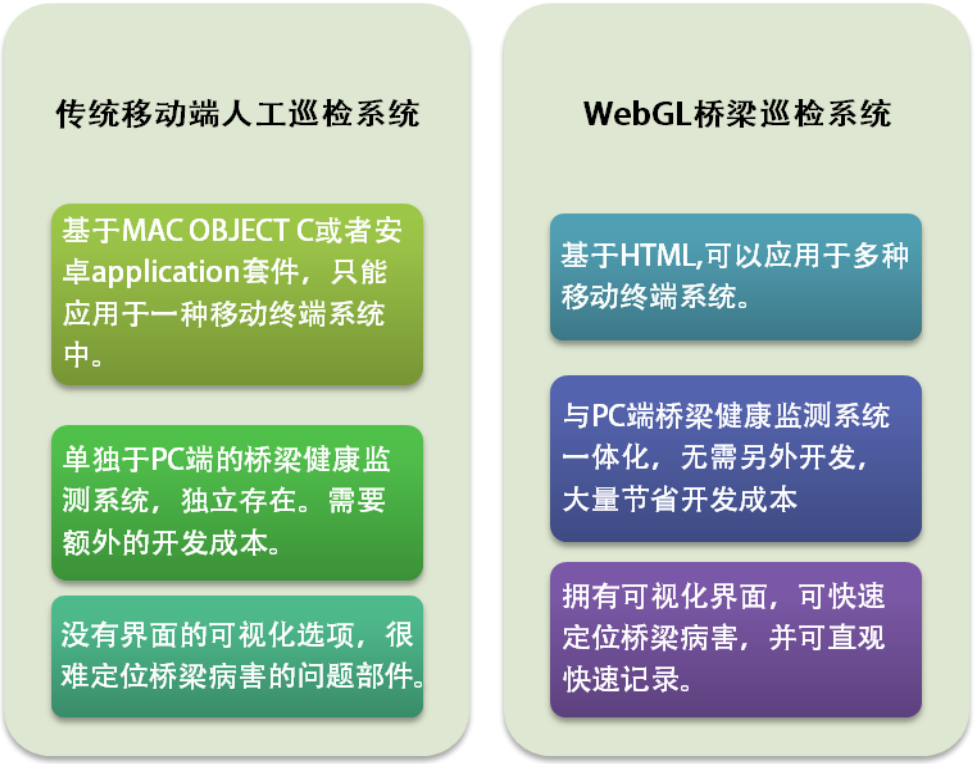


图 2-7 本系统与传统移动人工巡检系统对比

- 综合报表服务子模块

使用桥梁通用管理标准报表和实际汉江三桥管理报表，根据监测和计算数据

自动生成报表，可导出 PDF 或 WORD 文档格式报表，并能够存储打印。报表除数据显示外，最好能辅助一定的曲线显示。能够对提供的查询条件进行报表的查询。以供桥梁管理人员存档、分析和评估，其调用方式采用主菜单和在界面设置调用按钮两种结合的方式。

2.3.3 用户接口设计

本系统的用户接口界面设计要点是，主要凸显桥梁三维动画效果，在设计上整体保持风格的一致性，可以利用 HTML5 和 CSS3 技术实现 Web 界面的 3D 风格。同时，在界面漫游过程中为了不让菜单界面影响到用户体验，将界面菜单优化到侧边栏，需要时可以随时唤出。界面接口设计原则为直观、易用、极简主义、风格一致、操作性强。

界面主要部分为桥梁的 3D 视景，载入界面预设桥梁巡视动画，赋以用户身临其境的感受。桥梁视景漫游可由鼠标操控，作为其他子模块的基础存在。菜单栏是系统的主要功能的接口。设计图标易用、有透明感、视觉系认同感、不占用主要界面的特点。菜单栏目包括：病害巡查、预警报警、病害录入、桥梁档案、监测报表。菜单的动态效果彻底摆脱 JavaScript 束缚，完全使用 CSS3 原生动画特性 CSS3 transition 与@keyframes，系统载入速度更快，且不会出现 JS 冲突等问题。信息窗口设计同样融合了统一的 3D 感和透明感设计，使用 CSS3 DIV 的新原生特性圆角窗体 CSS3 border-radius、窗体投影 CSS3 box-shadow，调整窗口的透明度。信息窗口主要承载了桥梁病害巡查信息输出等功能。

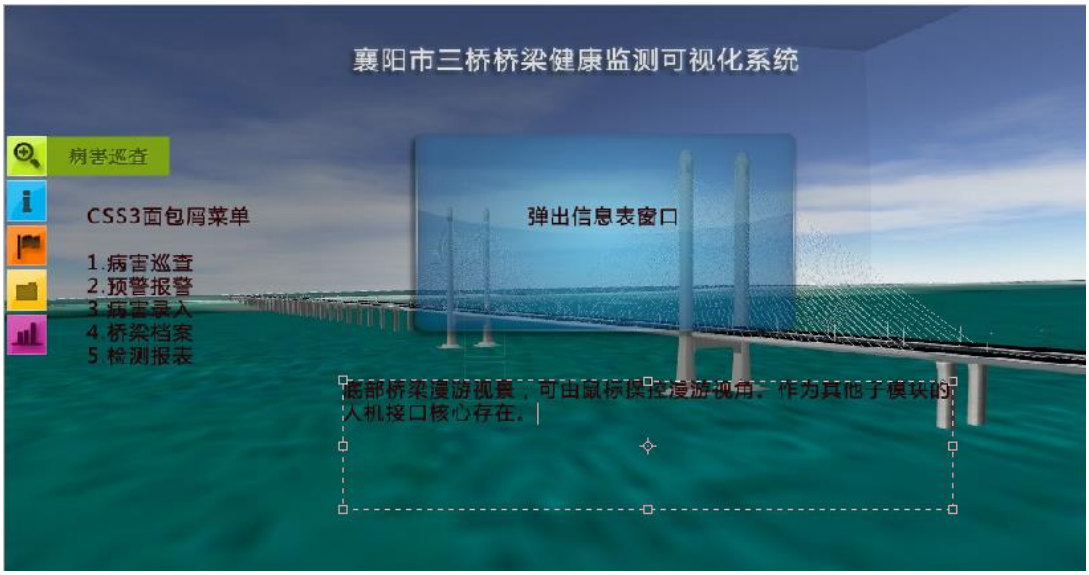


图 2-8 系统界面设计图

2.4 本章小结

本章首先介绍和研究了课题的关键技术 HTML5、WebGL。测评 WebGL 在 Web3D 的优势特性，脱离插件和系统壁垒的开放性，分析 WebGL 的内部绘制过程及 GPU 渲染管线原理，并研究了 WebGL 框架生态，横向测评选取最优方案，为下一步系统设计与实现打下基础。然后，针对桥梁健康监测可视化系统的实际需求分析，研究出合理的系统实现目标，并设计出合理的系统构架和系统功能。

第 3 章　桥梁健康监测可视化系统关键技术实现

3.1 WebGL 环境配置

3.1.1 WebGL 关键软硬件组成与配置

为实现本系统中 WebGL 桥梁可视化同时在智能手机、平板电脑、PC 机上的跨平台运行，需要从 WebGL 软硬件结构视角解读[40]。

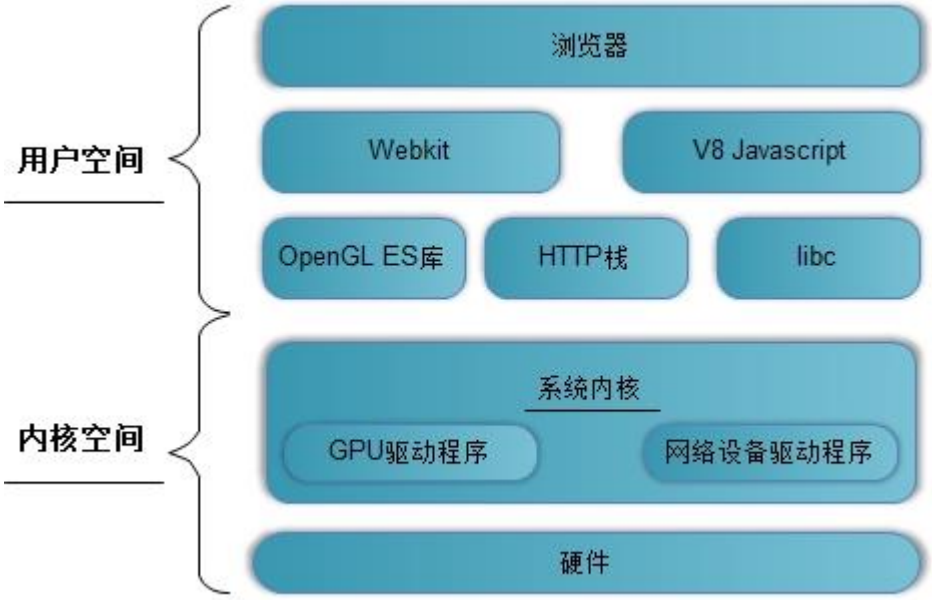


图 3-1 支持 WebGL 关键软硬件

从底层硬件角度解读，虽然 CPU 及网络连接带宽方面都会影响到用户体验，但是对于大数据量片面的 3D 模型，尤其是本系统的桥梁模型一般可以达到十万级数量片面的大模型而言，GPU 显然对于 WebGL 渲染性能是更重要的。GPU 所需要的软件驱动分为两类，用户控件库（User Space Library）和内核空间驱动（Kernel Space Driver）。WebGL 是以 OpenGL 2.0 为基础，AMD(ATI)或者 NVIDA 等主流显卡，其用户控件库需基于 OpenGL 2.0 以上版本。一般来说，用户安装最新的驱动后，WebGL 即可正常运行。但是，对于 Intel 的集成显卡，则需要配置相关软件，例如最新版本的 Mesa 开源、基于纯软件的 OpenGL 应用程序接口。

移动设备方面，以 NovaThor U8500 芯片为例，它采用 ARM Cortex-A9 处

理器，支持 Webkit 及 V8 JavaScript 引擎的 Web 浏览器，其 Interconnect 模块作为高性能 on-chip 总线，连接 GPU 为 Mali-400。Mali-400 可以利用其片源处理器的小块延迟技术，实现环境映射在片源着色器中的计算。而这款芯片和 GPU 大量应用于各种移动终端，为移动端实现 WebGL 的硬件环境提供了十分成熟的条件。

从上层用户空间的视角解读，Web 浏览器在浏览器引擎的作用下实现分析 HTML、构建 DOM 树和生成页面布局等功能，HTML5 的 WebGL 技术在 Chrome 和 Safari 的 WebKit 引擎，Firefox 的 Gecko 引擎，以及 Google 的 V8 JavaScript 等引擎下实现。但是，由于显卡及显卡驱动在设计时没有考虑安全问题，而相关安全问题是由操作系统完成，浏览器沙盒通过 WebGL 可以被安全的执行，如此会使脚本取得跨域名执行权限，甚至是访问本地文件的权限，带来安全隐患。所以，很多浏览器需要通过调试才能支持 WebGL。

PC 端调试 Chrome 和 Firefox 支持 WebGL 方法：

调试 Chrome.exe 属性，在路径后添加 –allow-file-access-from-files –ignore-gpu-blacklist，完成后重启 Chrome。

Firefox 的地址栏输入 about:config，设置 webgl.disabled 为 true。

移动终端方面，新版 Android chrome beta 基于 Chrome 25，可以支持 WebGL。在 Android 设备中利用 Platform Tools 工具设置 WebGL 支持。

IOS 移动设备可通过 Demoseen.com 开发的 WebGL Enabler 开启 WebGL 支持。安装 WebGL Enabler 后所有 UI WebView 控件都将支持 WebGL，不仅原装 Safari 浏览器，也包括例如腾讯浏览器此类使用 UI WebView 控件的的第三方浏览器。

3.1.2 调试工具 WebGL Inspector

WebGL Inspector 是由 Ben Vanik 开发的一款为 WebGL 专门打造的调试工具，作为 Chrome 开源的扩展插件可被安装在浏览器中，载入 WebGL 类网页时将自动唤醒 omnibox 栏的 GL 图标，用户点击的 UI 图标即可调出控制面板。相对于 Chrome Console 和 Firebug 这类通用前端调试工具来说，WebGL Inspector 则更专业化，例如它可以记录应用程序一帧的内容，并分析出各类相关信息。

其工具面板包含 Trace（跟踪）、Timeline（时间线）、State（状态）、Textures（纹理）、Buffer（缓冲）、Programs（程序）。Trace 面板显示当前捕获帧中所有

的 WebGL 调用，可以逐个执行记录中的命令，也可以在面板右侧看到场景创建过程。同时，它还可以高亮冗余调用，帮助用户优化 WebGL 应用程序的性能。Timeline 面板主要记录实时统计数据。State 面板显示状态快照。Textures 面板提供 WebGL 有关的纹理信息，可以读出纹理图像与纹理贴图， Textures 也提供了有关纹理的封装方式、纹理过滤及载入 URL 纹理信息。Buffer 面板可以显示程序缓冲内容、缓冲大小以及不同缓冲绑定对象等内容，对于复杂的几何对象非常有用。Programs 面板向用户提供所有着色器程序信息，甚至是着色器的源代码。

WebGL Inspector 可以极大的帮助 WebGL 开发人员快速定位程序错误，分析和优化系统代码，提高系统效率。

3.1.3 系统软硬件环境配置

（1）硬件环境：为开发过程更加流畅，实现大数量级片面的 Web3D 重建，建议硬件配置需满足：2.2G Hz 及以上主频 CPU，2G 及以上容量 RAM，1G 主流 GPU，1280*1024 显示器，7200 RPM 或更高转速的硬盘。

（2）软件环境：安装操作系统 Windows 7 中文版，GPU 驱动用户控件库支持 Open GL 2.0 及以上版本。系统开发平台采用开发工具为 Sublime Text 2，系统利用 HTML5、CSS3、JavaScript 语言编写，主要调试工具为 WebGL Inspector 与 Firebug。桥梁三维可视化模型采用 3D MAX 工具制作。

3.1.4Three.js 框架解析与配置

本文分析横向分析了 WebGL 框架生态，对比得出最适宜本系统开发功能及性能需求的框架为 Three.js 框架，它有开源、轻量、功能强大及支持外部模型导入等众多优势。Three.js 框架全部代码可直接从 GitHub 下载到本地，解压后其源码与内核文件目录为树状结构。

表 3-1 Three.js 框架目录结构

/build	three.js 和 three.min.js 两个可被引用的主要框架文件。
/doc	主要的帮助文档，有 three.js 框架的所有函数说明。
/editor	包含 Web 端三维模型的编辑工具。
/example	很多实用的 Demo，例如有 shader 脚本实用例子、多种格式 loader 例子和很多功能性的 js 脚本可供学习。

/src	包含了所有框架的核心代码，包括形状、摄像机、灯光、材质、动画、应用数学库等。
/test	测试代码。
/utils	存放很多工具脚本，例如各种模型导出工具，有 blender 及 max 导出插件，可以把.fbx 格式转成 Three.js 场景 python 脚本等。

与其他 JavaScript 的开源框架一样，Three.js 引擎嵌入网页只需要将 Three.js 或 three.min.js 文件引用到 HTML 文件即可。

验证 Three.js 启动，打开调试窗口，并在 Chrome Console 中输入命令 THREE.REVISON，如果得到版本信息即表示框架已成功植入。

3.2 桥梁模型建模 Web 视景重建

3.2.1 3D MAX 桥梁模型构建

3D MAX 是 Autodesk 推出的 PC 端 3D 动画专业制作软件，是目前最流行的动画建模软件之一。3D MAX 拥有易于掌握，提供直观、强大的建模功能、多种高速图像渲染生成功能和动画设计功能等优势。其扩展性也很好，拥有众多外挂插件，可以帮助设计师简化设计过程和优化三维模型。此外，3D MAX 支持也.obj 格式、 .dae 格式、 .wrl 格式 等多种格式模型导入和导出。被广泛应用于工程设计、电影特效、动态仿真、教学辅助等多个领域[41] [42]。

桥梁健康监测可视化系统目标是真实再现桥梁周边环境，实时直观分析监测数据。桥梁动画场景设计需要桥梁三维造型、桥梁周边环境地形、监测设备构建布置等构成。桥梁三维模型是根据实际桥梁结构设计，通过实地考场桥梁特征，记录桥梁悬索、塔柱、截面实际情况，结合桥梁设计图纸，实际纹理图片，同比例真实重建而成。

3D MAX 提供了多边形建模、片面建模、贴图建模以及 NURBS 建模法，分别使用于不同模型。本文系统桥梁三维基础模型基于标准几何体组成变形，运算处理实现多边形几何体，再使用材质贴图实现。例如，桥梁塔柱基于 ChamferCyl(倒角柱体)建模，悬索基于样条线放样或 Cylinder 建模，组建桥梁各部件多边形几何体，通过真实部件照片绘制材质贴图，实现部件材质渲染。三维地形根据实际桥梁周边环境设计，桥梁所处的河流由 Plane 加贴图绘制，山景、

树木等利用 BOX 转变多边形并贴图建模生成，天空创建 Sphere 贴图生成。监测设备构建需要在对应的桥墩、底板标注，用户可根据场景模型了解桥梁健康监测设备的基本布置情况。

3.2.2 3D MAX 桥梁模型优化

桥梁可视化模型结构非常复杂，尤其构造某一部件的三角片面数量决定了模型精细度。为力求模型的真实效果，在制作过程中常会应用到多边形细分动作的指令，但是大量的多边形将严重影响模型渲染时间、系统 GPU 占用和场景的实时性。尤其是对于 Web 应用来说，大数量级模型下载到本地渲染对系统实时性和体验度有极大的破坏性。针对于本文中复杂桥梁场景，其片面数量级通常可以达到十万级甚至百万级，模型文件巨大。为使模型片面数和显示效果达到平衡，可以通过使用 3D MAX 软件的 Optimize 工具或借助三方模型简化工具 Polygon Cruncher 来优化模型片面数量。

Polygon Cruncher 是一款支持多模型、多语言的 3D 模型优化插件，可以处理百万级片面模型，并且在不影响模型外观、高优化比的情况下减少片面数量，保留原模型 80%的细节、材质纹理、顶点色彩等。可直接加载到 3D MAX 使用工具面板中使用。Polygon Crunche 也提供三种最佳优化选项 Crunch borders，Protect Borders 以及 Exclude Borders。Crunch borders 不会顾虑物体造型，所以在减面数量很大时会导致造型的改变。Protect Borders 会在保持造型一致的基础上影响到原型的轮廓。Exclude Borders 会在一定的数量范围内保持原型的最佳优化。除此之外。Polygon Crunche 也提供局部减面优化的功能，可以在需要某些细节精细化的条件下，优化其他次要部件，达到最佳的模型优化效果。

以天兴洲大桥模型为例，做桥梁模型片面数优化。优化前桥梁模型对象 3244，顶点数 673277，片面数 573696，需要物理内存为 3370Mb。Polygon Cruncher 右侧面板有优化预览画面，左侧减面选项面板有 Optimization，Selection 以及 Advanced features 三个选项，Materials and textures 面板中勾选 Keep materials discontinuities 后右侧贴图出现红色标记，Optimization 面板输入减面比例，在右侧可以观察优化效果，达到预期效果后按下 Apply，并回到 Modeler。此时，优化后桥梁模型有对象数 3244，顶点数 119794，片面数 173576，需要物理内存 2785Mb。减面优化后模型 UV 结构并未破坏，而多边形面有被翻转的情况，手动可以将之修复。

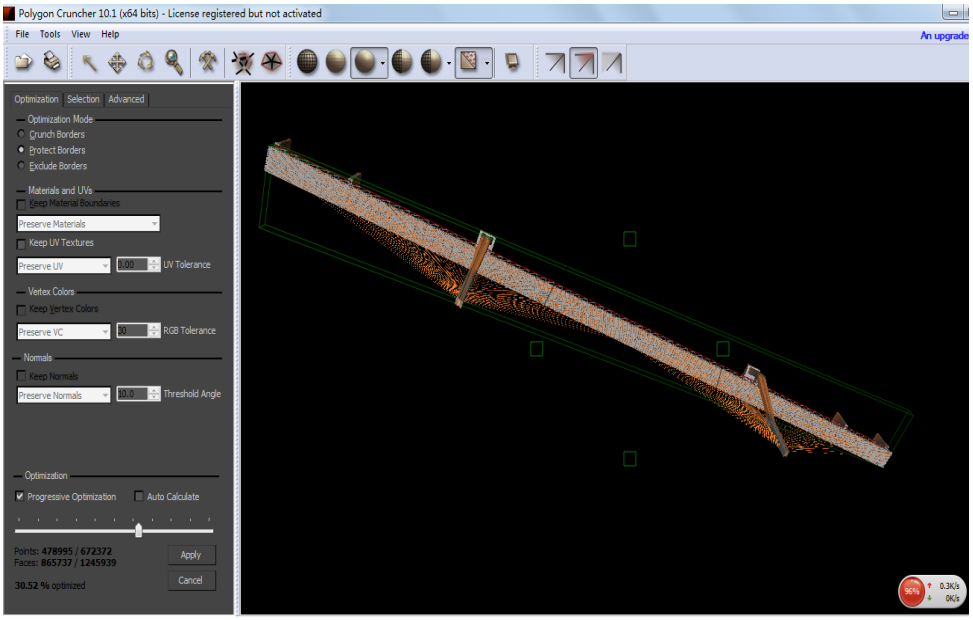


图 3-2 Polygon Cruncher 桥梁模型优化

3.2.3 桥梁模型重构

Three.js 框架虽然无法直接使用 3D MAX 的.max 格式模型重建桥梁视景，但是提供其他通用的 3D 格式模型的载入接口，支持例如.OBJ 格式、COLLADA 模型文件的. Dae 格式、VRML 的.wrl 格式以及专门为 Three.js 设计的 JSON 格式。与 COLLADA 等其他模型文件相比，JSON 格式模型文件更加简洁实用、网络传输和载入速度也更快，COLLADA 基于 XML、而 JSON 其本身就是 JavaScript，所以浏览器解析可以直接快速读取其数据结构。从浏览器渲染效率角度考虑，本文将使用优化的 JSON 模型格式。为方便使用，Three.js 本身提供 Blender、MAYA、3D MAX 三种 JSON 导出插件和脚本。

在 3D MAX 中植入转格式插件方式为：

首先，将\utils\exporters\max 目录下的 ThreeJSAnimationExporter.ms 以及 ThreeJSExporter.ms 插件放入 3D MAX 安装目录下的\Autodesk\3ds Max 2013\scripts\Startup 目录下。然后打开 3D MAX 工具栏的 MAXScript 选项，点击运行脚本，即可选择 Startup 目录下的两个转格式工具。

其中，ThreeJSExporter.ms 的功能是将 3D 场景导出到 three.js 可用的 JSON 格式中。ThreeJSAnimationExporter.ms 的功能是可将 3D 场景动画导出到 three.js 可用的 JSON 格式中。

然后，将桥梁模型进行全选，点击 ThreeJSExporter.ms 会弹出选项卡，勾选 Export vertex colors（输出顶点颜色）、Export UVS(输出 UV 信息)、Export normals(导出法线)、Flip YZ(YZ 翻转)、Flip UV(UV 翻转)。点击 Export selected objects，选择输出目录，即可输出.js 桥梁模型。

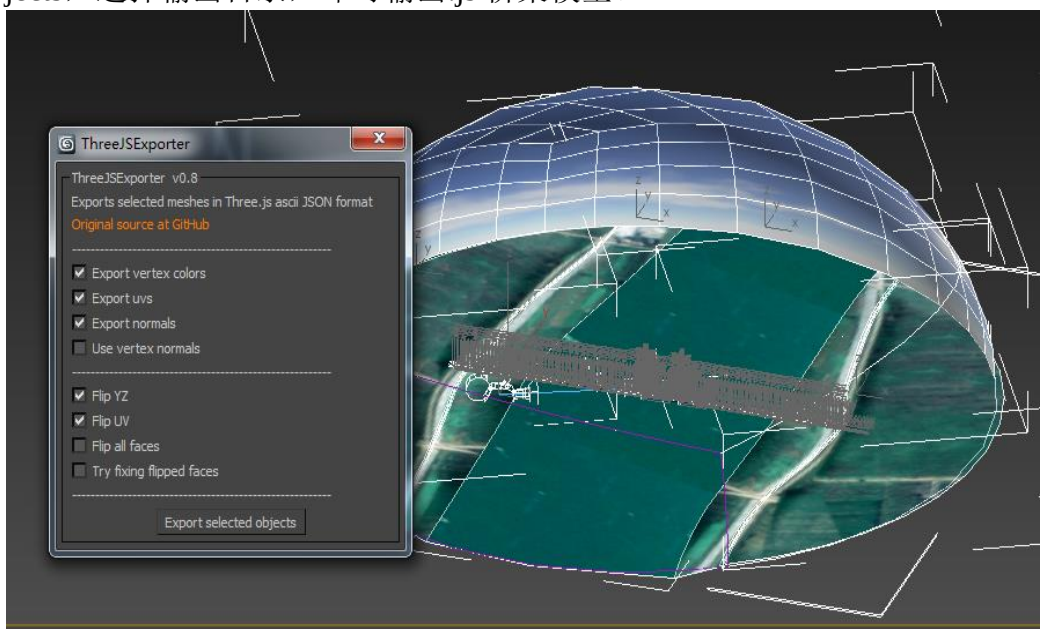


图 3-3　3D MAX 导出 Three.js JSON 工作界面

但是，导出的 JSON 模型常常会出现丢失平滑着色组，甚至会出现丢失材质着色和材质贴图等问题。相较于 3D MAX 的 Three.js JSON 导出来说，Blender 的 Three.js 转格式脚本可以更好的导出模型场景，并在 Web 浏览器中实现场景的最佳还原。Blender 是一个开源的轻量级三维动画制作软件，它以 python 为内建脚本，提供从建模，材质，渲染，动画，到音频、视频处理剪辑的制作解决方案。方法如下：

首先配置 blender 导出工具，将\utils\exporters\blender\2.65\scripts\addons 目录下的 io_mesh_threejs 文件夹及其全部文档考入到 Blander 安装目录的\scripts\addons 目录下。启用插件，在 Blender 中的 File 下 Preferences 下单击 Addons，在列表中选择 Import-Expore： three.js format 选项。

然后，利用 3D MAX 导出一个.OBJ 格式文档和一个.MTL 格式文档。其中，.OBJ 文件主要存储顶点信息，而.MTL 文件主要存贮材质贴图的信息。

Three.js 提供了由 Python 编写，用于转换 Wavefront Techonlogies 的 OBJ 文件的转换工具。可在 Blender 的脚本工具栏中直接输入如下命令行进行.obj 和 json 格式的转换:

Python <path-to-path> /utils/exporters/convert_obj_three.py\ -i mybridge.obj –o mybridge.js

但是，目前的 JSON 格式有需要传输的数据依然太大，所以可进一步将 Geometry 里的顶点数据用二进制存储，读取的时候直接读取到 ArrayBuffer，不论是加载的速度还是解析的速度都更快。在 Blender 的脚本工具栏中输入如下命令行：

Python <path-to-path> /utils/exporters/convert_obj_three.py\ -i mybridge.obj –o mybridge.js –t binary

除此之外，依然有许多研究组织针对于 WebGL 不断研究出更利于 OpenGL 解析和网络传输的优化模型格式，例如 WebGL-Loader 工程的 UTF-8 以及 COLLADA 的研究组织正在研究的 glTF。如图 3-4 是 glTF 的模型格式工作流。

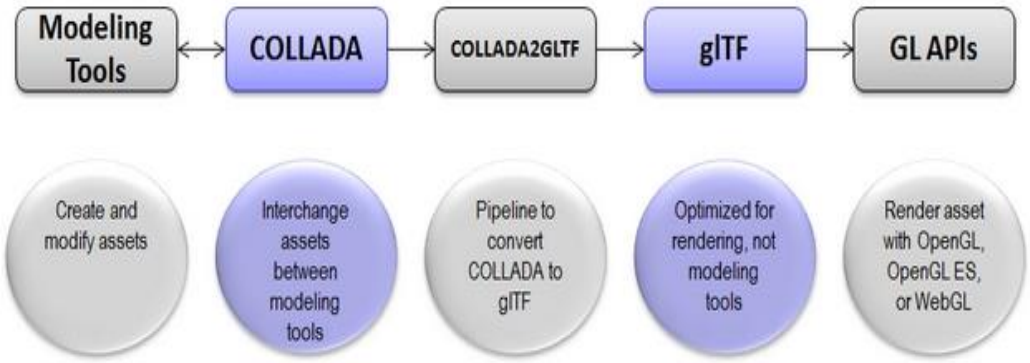


图 3-4　glTF 的模型格式工作流

从传统的 DCCTools 中导出 COLLADA 文件，并用其内置 Converter 转成 glTF 格式。转换后会有下面三种文件：

.JSON: 描述整个场景的 JSON 文件，可以直接在编辑器中编辑。

.BIN: 以二进制存储顶点信息的文件，可以直接利用 BufferView 创建，传输和解析的数度快，而且因为是用二进制存储，所以 glTF 会比纯文本的 COLLADA 小很多。

.JPG: 所有浏览器能够解析的纹理文件。

相较于原来的模型，其大小可以优化到原来的 1/3，并且其 JSON 结构更加贴合 WebGL 和 OpenGL ES2.0 的绘制命令。但是，由于这种标准还在研发阶段，材质定义在不断变化，动画和纹理和几何压缩工作仍处于早期阶段，所以为系统的稳定性考虑，本文只做研究讨论。

3.2.4 桥梁模型 JSON 解析及优化

导出后的 JSON 文件如图 3-5 所示，包括模型本身的信息、导出工具的版本、以及主要的 3D 数据描述，利用不同的标签描述顶点、纹理坐标、法线、几何场景结构、材质、贴图信息等。JSON 格式允许单一网格拥有多个材质，模型的不同部分可以用不同的颜色、着色效果、纹理及贴图，利用回调函数可以将这种单网格多材质合成一个真正的材质。

```
{

"metadata":
{
"sourceFile": "xfbr.max",
"generatedBy": "3ds max ThreeJSExporter",
"formatVersion": 3.1,
"vertices": 82209,
"normals": 144426,
"colors": 0,
"uvs": 147465,
"triangles": 144426,
"materials": 14
},

"materials": [
{
"DbgIndex" : 0,
"DbgName"  : "xfsq001",
"colorDiffuse"  : [0.5882, 0.5882, 0.5882],
"colorAmbient"  : [0.5882, 0.5882, 0.5882],
"colorSpecular"  : [0.9000, 0.9000, 0.9000],
"transparency"  : 1.0,
"specularCoef"  : 10.0,
"mapDiffuse"    : "xfsq001.jpg",
"vertexColors" : false
},

{
"DbgIndex" : 1,
"DbgName"  : "xfsq003",
"colorDiffuse"  : [0.5882, 0.5882, 0.5882],
"colorAmbient"  : [0.5882, 0.5882, 0.5882],
"colorSpecular"  : [0.9000, 0.9000, 0.9000],
"transparency"  : 1.0,
"specularCoef"  : 10.0,
"mapDiffuse"    : "xfsq003.jpg",
"vertexColors" : false
},
```

图 3-5 WebGL 标准 JSON 文件

Three.js 框架提供了 THREE.JSONLoader()类和 load()函数来载入 JSON 模型。载入时需要获得 3D 模型的顶点信息 Geometry 及材质信息 Materials。本项目中，模型材质利用了大量的 JPG 贴图，而 Three.js 提供的 THREE.MeshFaceMaterial()类可以将单网格多材质合成一个材质，并读取模型的材质和贴图。读取的贴图材质过程为，Three.js 内部调用 gl.texImage2D()将材质信息上传并保存在 GPU 纹理内存中。GPU 纹理内存是专供于 GPU 使用的一个特殊的视频内存，GPU 访问速度会比普通系统内存更快。纹理重组（swizzling）上传纹理，并在 WebGLBuffer 中依据材质的 UVs 坐标系在 Web 中重绘场景。

由于模型是由其他建模软件制作而成，而导入到 JSON 格式的过程中，模型的整个坐标系统、单位、法线等信息都会出现一定程度的混乱，使显示出现偏

移等问题。Three.js 中提供了计算模型信息的函数类，可以查看具体的模型相应的还原问题来做出相应调整。

Geometry 类提供了顶点表面构成的几何体有关的各种函数方法。利用 Geometry 获知顶点数组信息 vertics 和片面数组信息 faces。并根据顶点和片面数组计算出对应模型的各种信息。例如，Geometry 类可以利用 ComputeCentroid() 计算出几何体每个表面的重心，函数 computeFaceNormals()计算出 Face 数组的法线向量，computeVertexNormals()计算出 vertexNormal 属性。在本系统中虽然模型有多个面，但是其法线向量只有一个，并同时受多个片面的影响。如果希望得到模型的整体信息如宽度、高度和深度，Geometry 提供了可以计算包围盒的方法 computeBoundingBox()。以下代码可以计算桥梁模型包围盒，获得模型的完整信息：

```
function mymodelinfo(model,info){
    ver gemetry=model.mush.gemetry;
    gemetry.computeBoudingBox();//计算模型包围盒
    var mybox=geometry.boudingBox;
    var width=mybox.x[1]-mybox.x[0];
    var height=mybox.y[1]-mybox.y[0];
    var depth=mybox.z[1]-mybox.z[0];
    var nummaterials=geometry.materials.length;//计算模型材质数
    var numface=geometry.faces.length;//计算模型片面数
    var nplaces=3;
    var statsHTML =
"MODLEINFO:"+width.tofixed(nplaces)+"x"+height.tofixed(nplaces)+"x"+depth.tofixed(nplaces)+"</br>#materials:"+nummaterials+"<br/#faces:>"+numface;
//输出 mush 的所有信息：宽度、高度、深度、材质数及片面数
    info.model=model;
    info.stats=statsHTML;
    displayModleStats(statsHTML);
}
```

根据模型的信息相应调整模型的显示大小，可以利用 Obj.scale.set()设定模型在浏览器中的显示比例，利用 Obj.position.set()调整模型的相对位置。函数 THREE.UV 调整模型 UVs 轴，UVs 决定了材质的坐标，对应于 Geometry 的顶

点，光栅化后每个像素都有其材质坐标，材质上取色，实现纹理模型渲染。

loader.load()函数在模型文件导入的时候就会自动地调用 call back 函数把模型添加到场景中，再调整模型的轴心、显示比例及显示位置即可，并不需要另外设定监听。以下代码可以将模型载入场景中。

```
var loader = new THREE.JSONLoader();
loader.load( "xfbr2.js", function ( geometry, materials ) {
 //载入模型的 url 和顶点、材质信息
geometry.computeVertexNormals(); //修正模型法线
var mat=new THREE.MeshFaceMaterial(materials);//读取材质贴图
var mesh=new THREE.Mesh(geometry, mat);
mesh.position.set( -10000, 0, 0 );//模型相对位置
mesh.rotation.set( 0, 0, 0 );//模型旋转轴心
mesh.scale.set(0.7,0.7,0.7);//模型显示比例
scene.add(mesh);//将模型加载到场景
animate();
        } );
```

Mesh 从 JSON 中解析显示到屏幕前需要三步：首先，顶点数据和纹理数据从本地磁盘读取到内存。第二，程序在内存中做适当的处理，将需要绘制到屏幕的顶点和纹理数据传输到显存中。最后，在每一帧的渲染时将数据送入 GPU 装配、绘制。

根据金字塔数据传输模型理论，顶点和纹理数据被解析到内存是最慢的，而 WebGL 这类需要网络传输就会更慢。对于小场景解析到内存可以在程序初始化时加载到内存中，而对于大场景，就需要使用异步加载的方式。

顶点纹理数据传输到显存获取着色器程序的每个 uniform 和 attribute 数据都需要开销，所以需要将这些 WebGL Context 资源数据缓存到本地，为了 Canvas 在渲染过程中不会受到转场景的影响，可以改变缓存策略针对不同的 WebGL Context 分配不同的缓存空间，切换选择。利用缓存层把上层的数据和下层的环境资源隔离。

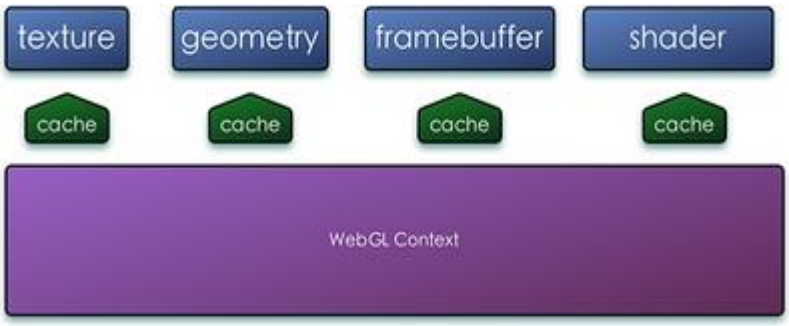


图 3-6　JSON 解析缓存优化

Three.js 对象有一个 needsUpdate 的属性，THREE.JSONLoader 会将绘制物体 Geometry 的顶点数据传输到显存，将这个 buffer 缓存到 geometry.__webglVertexBuffer，每次绘制前判断 verticesNeedUpdate 属性，如果不需要更新可以直接使用缓存信息，如果 verticesNeedUpate 为 true，则表示将 Geometry 中的顶点数据传输到 geometry.__webglVertexBuffer。对于静态场景是不需要 verticesNeedUpate 但是对于粒子系统或者本文中桥梁模型的进入动画及漫游交互这样每一帧都会改变自己的顶点的情况而言非常适用。对于纹理图片的异步加载来说，如果创建的 IMG 其 texture.needsUpdate=true，Three.js 的 Renderer 就会在这一帧中使用 _gl.texImage2D 存储纹理数据，当 texture.needsUpdate=false，图片加载完成后就无需更新显存数据，所以要在 onload 事件中图片数据加载完成后将 texture.needsUpdate 修订为 true。

此外，如果需要进一步优化 Geometry 的顶点数据，可以将其转为二进制存储，模型文件体积可以减小一半。载入方式是利用一个 THREE.BinaeryLoader 类对象，直接读取到 ArrayBuffer，加载的速度和解析的速度都可以得到很大提升。

3.2.5 场景着色与渲染

THREE.JSONLoader 方法载入了带有纹理贴图的桥梁模型。然而，对于一个有真实感的 3D 场景来说，还需要引入场景、光源、摄像机、渲染器等其他元素。

首先，设定场景，一个三维空间来盛放模型。代码如下：

```
var     WIDTH = document.documentElement.offsetWidth || 800,
        HEIGHT = document.documentElement.clientHeight || 600;
var     scene = new THREE.Scene();
```

在 3D 场景中，通常需要根据不同情况确定不同光源，及他们是如何影响场景中的材质。通常有局部光照模型、全局光照模型、Phong 反射模型模拟光照，全局光照模型会利用光源及光照反射的复杂行为。局部光照模型仅只参照选定光照模型。Phone 反射模型模拟真实场景中的光照，每条光线都需要预知漫反射和镜面反射属性，每个物体表面上的点都需要反射的环境光、反射的漫反射光、反射的镜面反射光、物体的反光度等四个属性，光照取决于环境光、漫射光及镜面光的总和[43]。

WebGL 基于不同模型实现相应的光照的着色器，在处理光照时需要向着色器传送光源的位置、灯光的颜色、顶点法线、法线矩阵信息。光源的位置、颜色及方向只需预设即可，但是顶点法线与法线矩阵则需要读取载入模型的法线信息，在着色器中计算出其法线矩阵，作为 uniform 变量传输到着色器。

Three.js 框架给定类型光源的接口，分别是环境光、点光源、聚光灯、平行光、区域光、方向光。光源的基类是 THREE.Light (hex) ，hex 参数代表 16 进制颜色值。其他类型的光源都是 Light 基类的继承。环境光从构造函数来看，只需要设定光源色，在场景中无处不在，对物体的影响也是均匀的，无论从任何角度，其物体颜色都一样，且其光线不会随距离衰减，强度是恒定的。

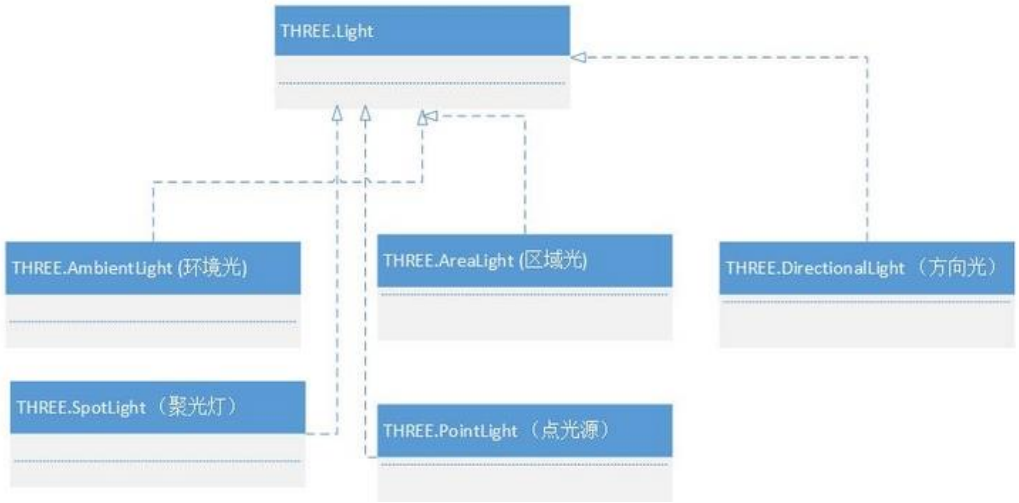


图 3-7 THREE.Light (hex)灯光函数

本文中的漫游系统利用环境光与平行光可以很好的模拟真实场景，代码如下：

```
var ambient = new THREE.AmbientLight(0xFFFFFF );
scene.add( ambient );//载入一个白色的全局光源
var directionalLight = new THREE.DirectionalLight( 0xFFFFFF );
```

```
directionalLight.position.set( 0, 100, 100 ).normalize();
scene.add( directionalLight );//载入一个平行光源
```

3D 模型从三维空间映射到二维平面的过程即为三维渲染，而这个渲染操作的软件即为渲染器。Three.js 框架中给定了 WebGLRenderer、CanvasRanderer、SVGRanderer 等渲染方式，从渲染效率和效果的角度来说，WebGLRenderer 使用 WebGL 渲染管线效率最高，无论是浏览器渲染速度，和场景模型的还原度精细度都最好，同时它还提供很多实用的函数优化渲染。但是 CanvasRanderer 和 SVGRanderer 两种渲染方法普遍适用于更多的浏览器。

WebGLRenderer 渲染过程一般为：获取画布宽和高，实例化渲染器对象，设定渲染器宽度和高度，然后将渲染元素追加到页面元素中，设置背景色和背景色透明度。除此之外 WebGLRenderer 还提供一些实用参数接口。如反锯齿（antialias）、着色精度（precision）、保存绘图缓冲（preserveDrawingBuffer）等。

本文中的桥梁模型考虑其渲染效率使用 WebGLRenderer 渲染方法，为优化模型显示效果设定其反锯齿参数为 true，代码如下：

```
webglRenderer = new THREE.WebGLRenderer({antialias:true});
webglRenderer.setSize( SCREEN_WIDTH, SCREEN_HEIGHT );
webglRenderer.domElement.style.position = "relative";
```

设置相机，即设置三维空间中的物体投影到二维空间的投影方式。Three.js 框架中 THREE.Camera 是相机的抽象基类，其子类有两种相机，分别是正投影相机 THREE.OrthographicCamera 和透视投影相机 THREE.PerspectiveCamera。

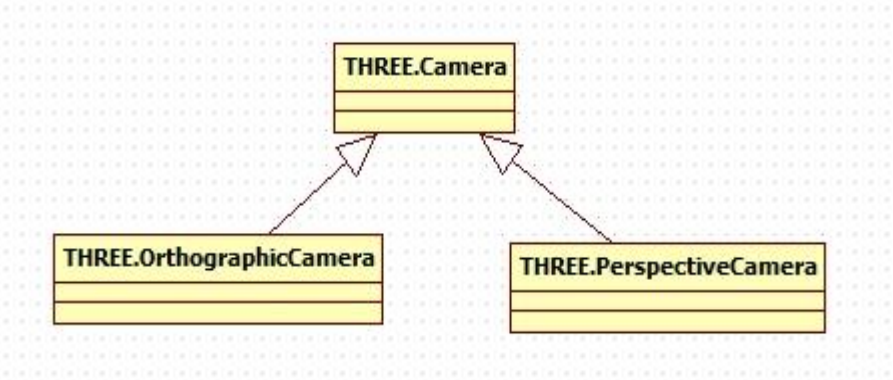


图 3-8 THREE.Camera 相机函数

两种相机的差别在于：透视投影有一个基本点，就是远处的物体比近处的物体小。相较之下透视相机更接近人类的观察习惯。设置透视相机，分别需要设四个参数：视锥角度、视口长宽比、近视距和远视距。设定相机位置后载入

场景即可。代码如下：

```
camera = new THREE.PerspectiveCamera( 45, SCREEN_WIDTH / SCREEN_HEIGHT, 0.1, 100000 );
    camera.position.z = 20000;
    camera.position.x = thex;
    camera.position.y = they;
```

THREE.PerspectiveCamera 还提供了应用于多显示器相机参数 setViewOffset，可应用于大型的拼接 LED 应用显示中。

3.3 桥梁模型视景漫游与视景交互

3.3.1 桥梁视景动画

Web 中一般有声明式动画和基于脚本的动画两种。声明式动画，如 CSS 样式层叠表动画或 SVG<animate>动画，用户只需要声明元素运动方法，并不需要用脚本来处理。基于脚本的动画，如 DOM 动画及 WebGL 绘制动画，需要用 JavaScript 定义每一帧的内容，作为回调函数的执行结果而存在。

模型在 WebGL 中的移动实际上是场景动画更新了位置后在窗口上的循环重绘制过程。原有静态场景只需渲染一次，而加入运动物体或者用户交互响应的场景需要多次渲染，这就需要通过循环函数来完成。通常在 JavaScript 中利用 setTimeout（）、setInterval（）以及 requestAnimationFrame（）三个方法来完成循环绘制动画的过程。setTimeout（）和 setInterval（）两个全局方法是在场景渲染完成后重制超时时差，在规定时间内调用第一个参数的函数。相较于前两者 requestAnimationFrame（）是专门为创建脚本式动画而创造，极大的优化了动画性能，有专门的动画创建方法，不需要人为预设最佳频率。在多标签页中，当动画处于后台，浏览器可以更新或暂停动画来优化动画性能，甚至可以优化电能使用，这对于移动设备运行系统来说至关重要。

本文中，Web 载入桥梁动画需要实现桥梁场景的循环漫游效果的动画，在设定频率下巡视桥梁整体场景。requestAnimationFrame（）将当前时间和绘制回调函数作为参数传入，调整 Mush 的 rotation 轴（旋转中心的 y 轴）调整场景相对位置，实现场景的重绘。以下代码实现桥梁场景的视景动画：

```
var start = new Date().getTime(),delta;
```

```
(function anime(){
    delta = new Date().getTime() - start;
    mush.rotation.y =      delta / 10000;
    renderer.render(scene, camera);
    return requestAnimationFrame(anime);
})();
```

这种方法主要基于更新运动模型的 rotation 旋转中心位置，在大多数情况下是可行的，但也会有缺点。在系统运行的过程中，由于 CPU、GPU 负载过大或内存总线的原因会引起 FPS 波动，动画的移动距离和画面也会非常奇怪。这时，需要考虑每个新创建帧的绘图频率，以优化动画的平稳度和流畅度。

虽然，在 WebGL 中并未预置太多的动画方法，但 Three.js 框架弥补了这些不足，提供了很多非常好的动画接口和工具。同时，利用诸如 Tween.js 这样的 JavaScript 开源动画框架来实现更多，例如关节动画、蒙皮动画等复杂的动画。

3.3.2 WebGL 交互事件处理

WebGL 交互事件处理的基本思想是当浏览器中发生某个重要事件，浏览器就创建并发送一个事件，用户需要事件处理程序来侦听这些事件。旧式事件处理机制是 DOM Level 0 基本事件处理，使用例如 onload、onmousedown、onmouseup 等，通过 HTML 元素属性定义事件，或将 JavaScript 定义并赋值给相应属性来处理程序。DOM Level 2 高级事件处理模型不同于 DOM Level 0 只是将 JavaScript 字符串或函数赋给 HTML，它是利用 addEventListener（）方法注册了元素的事件侦听处理程序。

DOM Level 2 的事件传播机制相较于 DOM Level 0 的执行元素注册事件机制要复杂的多。

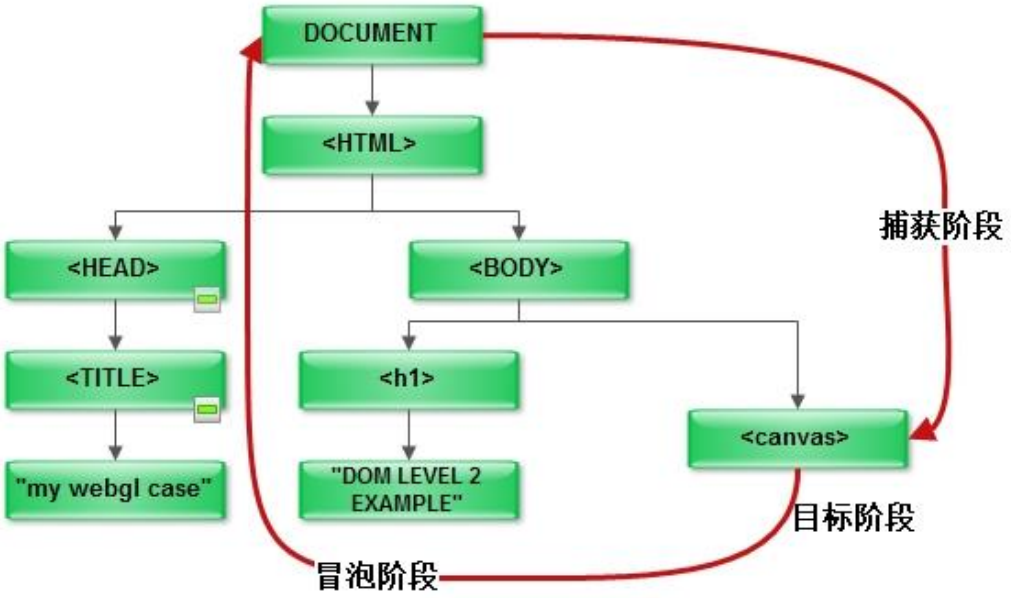


图 3-9 DOM Level 2 的事件传播机制

事件捕获阶段：即从 DOM 树顶端节点向下往目标节点传播，若目标节点的任何祖先节点注册了事件处理程序且激活了其捕获功能，就会在时间捕获阶段执行处理程序，调用 addEvenListerner（）函数，并将最后的参数设为 ture，注册事件处理程序并激活其捕获功能。

目标阶段：类似于 DOM Level 0，在目标节点上事件处理程序。

事件冒泡阶段：这一阶段与捕获阶段相反，事件自下而上传递到 DOM 树的顶端。

DOM Level 2 可以各种鼠标、键盘的交互事件，在本文中，将会需要实现大量此类交互事件。下面的函数实例是 addEventListener（）监听一个鼠标点击事件来执行 onDocumentMouseDown（）的方法。

3.3.3 Three.js 点击检测

在 3D 图形学中拾取（picking）是解析两个 3D 物体是否相交的处理过程，拾取算法原理是通过鼠标点击屏幕得到的二维坐标，投影到三维视景中成为一条射入的光线，光线与桥梁模型相交的 mesh 片面即为拾取片面。在 WebGL 中的 3D 拾取应用于点击检测中，实现与 3D 视景的复杂交互。在 WebGL 渲染管线一章中，本文曾分析过着色器的 3D 渲染过程实际上是将 3D 空间的顶点数组

映射到 2D 的投影矩阵中，将 3D 图像投影在 2D 的视口中显示。而 3D 点击检测与之相反。Three.js 框架提供了 THREE.Projector 实现矩阵运算转换为 2D 视口坐标的方法，提供了 THREE.Ray 找出视景中与射线相交的物体。在 Three.js 框架内实现点击监测的过程如下。

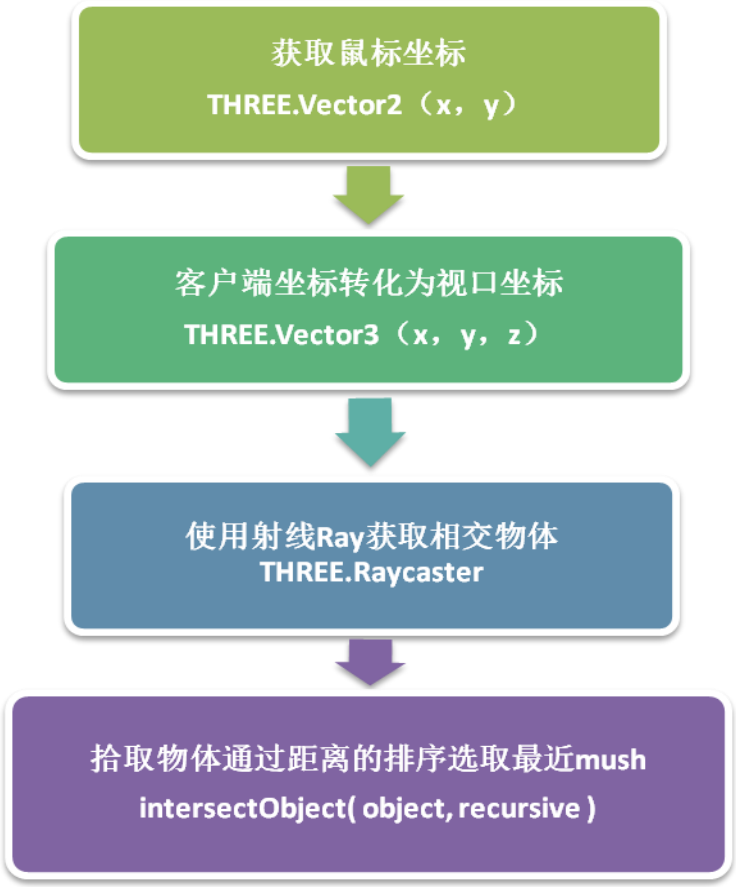


图 3-10　THREE.JS 点击监测过程

首先，将鼠标的坐标转换成视口坐标，实际上是一个坐标区间，原点的 x 轴与 y 轴的坐标范围-0.5 到+0.5 范围内。

```
return new THREE.Vector2(
//将客户端坐标转化为视口的 x，y 坐标
( clientX - _this.screen.offsetLeft ) / _this.radius * 0.5,
( clientY - _this.screen.offsetTop ) / _this.radius * 0.5);
```

然后，基于此发射一条射线 Ray，起于近视口截面相交处，止于远视口截面相交处。任何与射线相交的 Mush 都被认为是在鼠标的下方，选取相交物体，并实现与此物体的交互方法。Ray 的基本构造函数为 THREE.Ray，它只有两个参

数 origin 和 direction，顾名思义，也就是端点和方向。

THREE.Ray 是一种数学的方法，在 Three.js 中还有一种基于 Ray 数学模型封装的 Raycaster 方法，更易于实现 3D 网格的点击监测。THREE.Raycaster = function (origin, direction, near, far)。除了定义端点与方向，同时还有 near 和 far 两个参数，决定在某一段距离拾取的物体将被丢弃。

使用 Raycaster. intersectObject(object, recursive) 函数，拾取的物体通过其与监测点的距离排序，选取最近的片面，并获得其坐标。IntersectObject 方法的参数 recursive 如果设定，可以检测除了射线相交外所有的网格与检测点之间的距离，并选取最近。

3.3.4 桥梁视景漫游控制

本文中桥梁仿真视景并不是静态存在于 Web 浏览器中，也不仅仅在规定的路径内实现视景的动画，而是要实现更复杂的交互过程，利用鼠标和键盘来控制视景移动，实现第一人称视角的桥梁复杂环境的视景漫游。

在桥梁视景动画中本文使用的方法是操控 Mush 本身的 rotation 旋转中心坐标轴，但如果按照这种思路来实现桥梁视景的漫游就需要编写非常复杂的逻辑代码来实现场景内容，显然对于本文中的 3D MAX 导入模型来说，这样的操控方法需要遍历和解析 JSON 中的各个 DOM 节点数据，交互过程非常复杂。所以，本文提出另一种方法，操控相机对象，来实现桥梁环境的视景漫游。

在 Three.js 框架中，本文应用的透视相机（THREE.PerspectiveCamera）是一个顶级对象，可以通过平移和旋转来实现视景变换，且相机类提供了多个属性接口。结合前文中 WebGL 交互事件处理方法和 Three.js 点击监测方法，可以实现相机视景的轨迹球控制方法，实现通过鼠标滑轮来实现放大和缩小视景，通过鼠标点击拖拽的方法实现视景角度的改变。

Trackball 轨迹球主要是通过控制透视相机（PerspectiveCamera）的旋转、变焦和平移来实现视景控制。

相机的旋转函数 rotateCamera（）是基于 Three.js 提供的 Quaternion 的方法来计算透视相机的旋转轴。THREE. Quaternion 提供一个四元数，用以计算出三维场景的旋转角度。Quaternion 也可表示为 Q[w ，（x，y，z）]， w 与旋转角度相关，（x，y，z）与旋转轴相关，Quaternion 比 Matrix 更高效，存储空间占用小，且便于差值，可以避免万向节死锁（gimbal lock）的问题。在本文中利用

Quaternion.setFromAxisAngle 以绕任意轴旋转设定四维数旋转，实现相机视角的旋转控制。

相机的变焦函数 zoomCamera（）是通过调用 multiplyScalar 改变视景的 scale 显示比例达到变焦的效果。

相机的平移函数 panCamera（）是以鼠标平滑移动的距离，对应视景中相应的比例，移动相机的 x，y 轴，达到平移的目的。

轨迹球在实现操控相机后，还要通过 Web 的交互侦听来添加鼠标的滑轮事件和点击拖拽事件实现桥梁视景移动。

```
this.domElement.addEventListener( 'mousewheel', mousewheel, false );
//鼠标滑轮事件—缩放视景大小
this.domElement.addEventListener( 'mousedown', mousedown, false );
//鼠标点击拖拽中还包含了 mousemove 与 mouseup 的两个侦听事件。
document.addEventListener( 'mousemove', mousemove, false );
document.addEventListener( 'mouseup', mouseup, false );
```

Mousewheel 来实现视景缩放实际上就是应用了相机的变焦思想，在此不赘述。而鼠标的点击拖拽来实现视角的改变相对比较复杂。

首先，点击鼠标时获取鼠标点击的（x，y）坐标作为新的旋转轴、变焦轴和平移轴。然后，再拖动鼠标的同时获取到新的鼠标（x，y）坐标更新旋转轴、变焦轴和平移轴。最后放开鼠标，并停止监听（removeEventListener）。

完成轨迹球后，还需要将 Trackball 轨迹球引入到 HTML 的页面中，并设定其旋转速度、平移速度、边角速度等参数。代码如下：

```
controls = new THREE.TrackballControls(camera);   //初始化 trakballcontrls
        controls.rotateSpeed = 5.0;   //旋转速度
        controls.zoomSpeed = 5;     //变焦速度
        controls.panSpeed = 2;   //平移速度
        controls.noZoom = false;     //是否不变焦
        controls.noPan = false;   //是否不平移
        controls.staticMoving = true; //惯性  true 没有惯性
        controls.dynamicDampingFactor = 0.3; //动态阻尼系数  灵敏度
```

3.3.5 桥梁视景交互渲染

WebGL 桥梁健康监测可视化系统的重要优势在于三维视景作为交互入口对桥梁的每个构件、病害进行实时直观的标记和查询，对于桥梁温度索力传感器状态渲染报警等。

3D 桥梁的模型是由多个片面构成，每个片面又分别由顶点、着色、材质等信息，与桥梁视景的交互过程实际上由计算机仿真中的拾取算法获取。本文中曾叙述的 Three.js 点击监测方法 THREE.Raycaster 可以帮助获取到相应桥梁构件的空间三维坐标，继而在此坐标位置标记和渲染。

对于桥梁模型网格上的交互操作有两种实现思路。第一，在场景中拾取的空间坐标上绘制一个新的 geometry 物体来标记。第二，利用模型的原有的 JSON，拾取其片面 DbgIndex 索引信息，修改其片面的材质属性进行渲染。

在场景中绘制新的物体实现显然更简单。本文中，通过鼠标点击相应桥梁部件，监测出视景三维坐标，绘制圆球标记病害信息，以下为具体实现的代码。

```
function MarkmeMouseDown( event ) {
event.preventDefault();
var vector = new THREE.Vector3( ( event.clientX / window.innerWidth ) * 2 - 1,
- ( event.clientY / window.innerHeight ) * 2 + 1, 0.5 );
//将鼠标的二维坐标投影到三维空间得到 Vector3
    projector.unprojectVector( vector, camera );
    var raycaster = new THREE.Raycaster( camera.position,
vector.sub( camera.position ).normalize() );//实例化一个 raycaster 射线
    var intersects = raycaster.intersectObjects( objects ); //获取最近位置的物体
    if ( intersects.length > 0 ) {    //在该坐标绘制一个 sprite 圆球进行标记
    intersects[ 0 ].object.material.color.setHex( Math.random() * 0xffffff );
                var particle = new THREE.Sprite( particleMaterial );
                particle.position = intersects[ 0 ].point;
                particle.scale.x = particle.scale.y = 16;
                scene.add( particle );
            }
```

相较于第一种思路来说，第二种思路的实现过程显然更加复杂，需要对原始模型的细节层次树进行遍历。但是，这种方法的优势在于可以根据模型的片面索引信息数据库快速定位到某一构件，对该构件渲染出更真实的效果。尤其是在桥梁某部传感器预警报警子模块中，可以利用这种思路对传感器相应桥梁部

位通过修改材质颜色分级报警。

第二种思路拾取相应位置与第一种思路相同，都是通过点击监测拾取，在拾取相交 mush 的子节点 intersects，获取片面 DbgIndex 和 DbgName 并在页面上显示相应信息，在原有材质上渲染颜色。

```
var intersects = raycaster.intersectObjects(scene.children, true);
if (intersects.length) {
        var target=intersects[0].object;   //相交点片面复制给 target
        statsNode.innerHTML= 'ID: ' + target.DbgIndex+ '<br>'
         'Name: ' + target. DbgName;   //输出该片面的 ID 和名字
         target.face.color=highlightedcolor; //片面颜色高光
}
```

3.4 桥梁模型视景 LOD 渲染优化

桥梁的虚拟场景既要追求逼真的显示效果，又要在交互中快速的实现场景渲染，保证系统的流畅性和稳定性。但是，精致的场景细节必需要高分辨率的材质贴图，及更多更细致的片面数据量为基础。尤其是在 Web 环境中，巨大的数据量会严重影响模型的渲染时间，影响场景的实时性。GPU 内存被大量材质占用，甚至会导致模型上下文丢失，最终导致系统崩溃。本文中曾提出过多种对桥梁模型优化的方案，如在 3D MAX 建模阶段优化场景片面数量，将桥梁以优化的 JSON 二进制格式或者 glTF 格式优化模型结构和大小，以及在模型在 Web 中解析重建时利用 needsUpdate 属性，实现纹理图片的异步加载优化。为进一步优化系统的模型，本节中提出一种利用 LOD（Levels of Detail）层次细节优化算法，通过对模型层次逐级简化片面细节优化场景的几何复杂性，以提高绘制效率及 GPU 使用率。桥梁场景的 LOD 基本思想是：对桥梁原始的网格模型在三维空间中相对于视口的距离进行分层，近处渲染采用精细模型及较高分辨率的材质贴图，以显示桥梁构建的精细细节，远处的渲染采用较为粗糙的模型及低分辨率贴图显示。最后，使各层次间视觉平滑的过度，避免视点跳跃问题[44] [45]。

Three.js 框架在 Objects 类中有一个 THREE.Lod 作为对 LOD 方法的内置支持。其函数内部细节为：getObjectForDistance 以视口为基准计算出场景中的各个 Mush 的距离，addLevel 根据给定的距离标准划分显示层次，并返回该层次的网格，其他细节层次的网格则被丢弃，通过相应层次的材质渲染该层网格。通

过对距离层次的递归逐级渲染。以构造一个 IcosahedronGeometry 球体为例，分别利用 THREE.IcosahedronGeometry 构造大小相同，Mush 数不同的 5 个球体赋给 Geometry，lod.addLevel(mesh, geometry[i][1])根据景深对层次的划分决定不同视距物体将采用不同精度的球体在 GPU 中存储和渲染，

为显示当视距变近时高精度效果，本文设置一个循环动画，在每帧更新的时候， 通过检测照相机的相对位置的距离差， 达到阀值则更新整个场景，来展示变化。

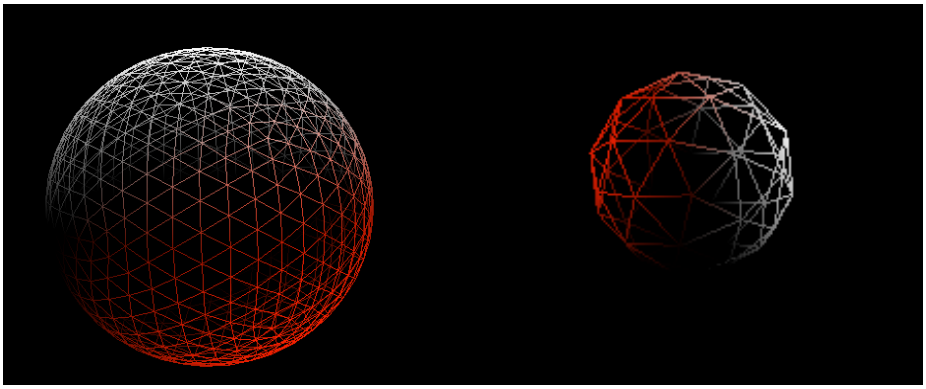


图 3-11 相机视距变化前后的球体细节展示

同理，用 LOD 的思想根据桥梁视景在交互过程中与相机视口的相对位置来划分渲染层次，距离相机近处以高分辨率绘制，距离相机远处以底分辨率绘制，进一步可以分割场景为环境模型和桥梁模型，环境模型以底分辨率存储和绘制，而桥梁模型以高分辨率绘制。这样，相应的要将原有的模型材质贴图对应的修改，这样只需在不同的层次中改变其相应的材质贴图属性路径，即可实现桥梁模型的 LOD 渲染优化。

3.5 本章小结

本章针对基于 WebGL 的桥梁检测可视化系统的关键技术进行了深入研究和实现。将桥梁模型在 3D Max 中绘制并优化，转为 Three.js 的 JSON 模型格式，重构到 Web 中，优化桥梁 GPU 加速和视景环境细节的渲染。研究 Web 上的桥梁仿真模型的动画与交互操作，实现桥梁的视景漫游，交互标记，细节层次渲染等。并进一步研究利用 LOD 技术优化桥梁模型场景的渲染效率。

第 4 章 襄阳江汉三桥健康监测可视化系统

4.1 襄阳江汉三桥健康监测概况

襄阳市江汉三桥（又名襄阳汉江卧龙大桥）是襄阳首座斜拉桥，其主线全长近 4.6 公里，主跨 300 余米，主结构为双塔双索面半漂浮体系斜拉桥，桥面宽近 32 米，全桥共有 208 根斜拉索，最长斜拉索长 172 米。桥梁斜拉索设有减震装置，用以固定拉索与桥身减少其共振力影响。斜拉索与塔冠共同形成扇形景观。襄阳市江汉三桥分南北两座主塔，塔高达到 120 余米，集装达到 75 米，其索塔成 H 型直立形塔柱，其断面可达 27 平方米，塔柱根部为实心段，可很好地承载桥梁的负荷。襄阳市汉江三桥现已布设传感器的 14 个塔截面，进场传感器 16 串，累积安装光纤光栅应变传感器 52 支，光纤光栅温度传感器 52 支，安装保护引纤后测试信号存活率达 100%。

4.2 襄阳江汉三桥三维桥梁模型

襄阳江汉三桥采用 3D MAX 建模，根据桥梁设计图纸同比例绘制桥梁桥面、塔柱、截面、斜拉索等结构细节，模拟真实环境设计水面、山景、天空等。基于标准几何体组成绘图，实现多边形几何体，采用材质贴图，最大限度模拟真实场景。

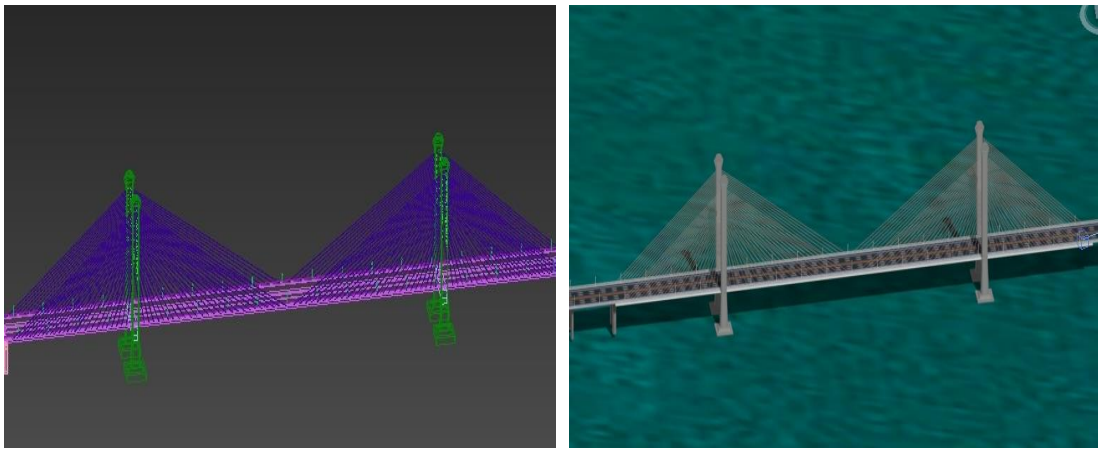


图 4-2　3D MAX 襄阳江汉三桥模型设计

4.3 系统运行概况

襄阳江汉三桥健康监测系统初始化运行启动规定路径动画，令用户对桥梁大概环境、结构有所了解。用户在使用系统前，能够对桥梁结构情况有所了解，有利于分析监测等操作的有序进行。

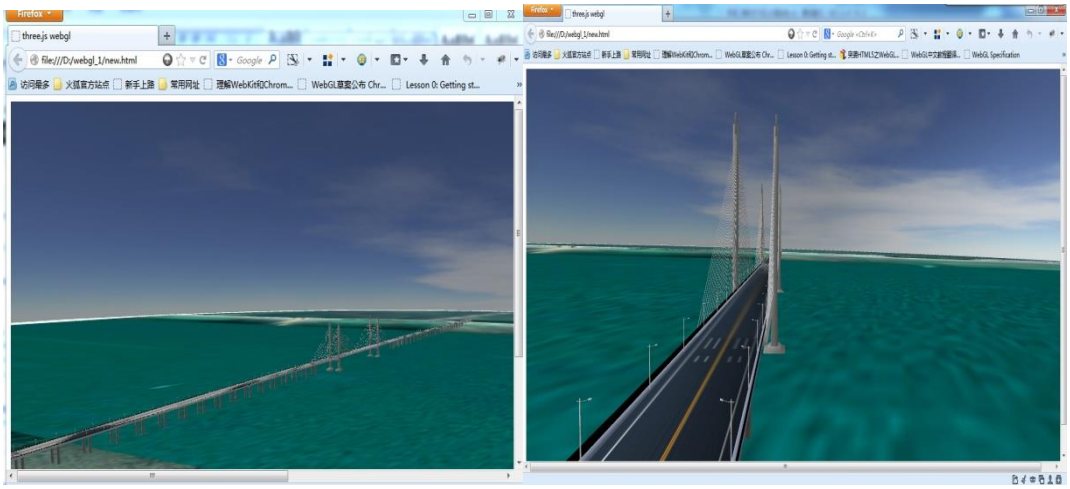
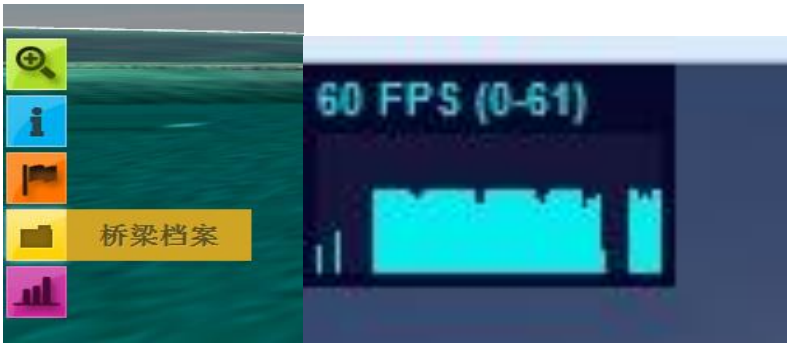


图 4-3 系统进入的规定路径漫游动画

进入系统主界面，侧边工具栏以象形图标，菜单栏目包括：病害巡查、预警报警、病害录入、桥梁档案、监测报表。鼠标滑过弹出标记说明，点击可以弹出相应的子窗口。主界面以桥梁视景为主，可以通过鼠标来控制，可以实现桥梁场景的俯视，仰视及部件细节的查看。运行监控，可以查看桥梁场景渲染的 FPS 速率达到 60，桥梁可视化系统运行流畅。

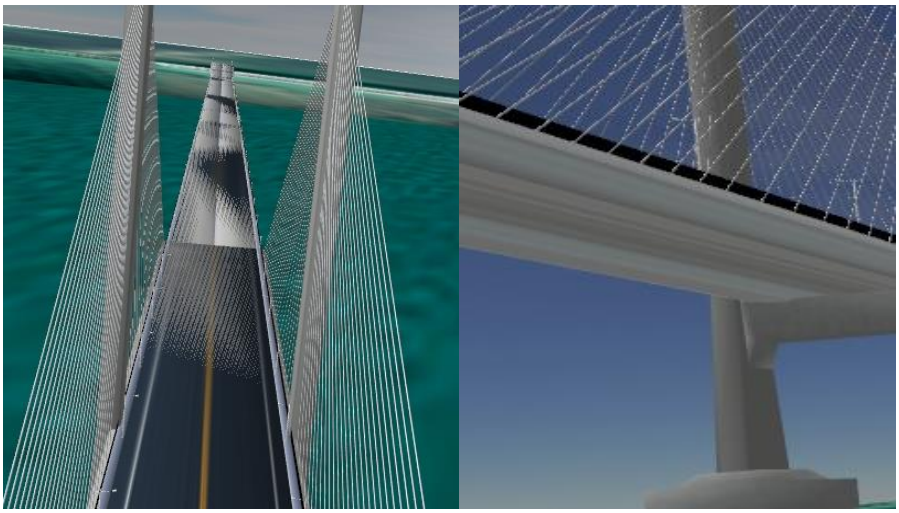
A 主界面



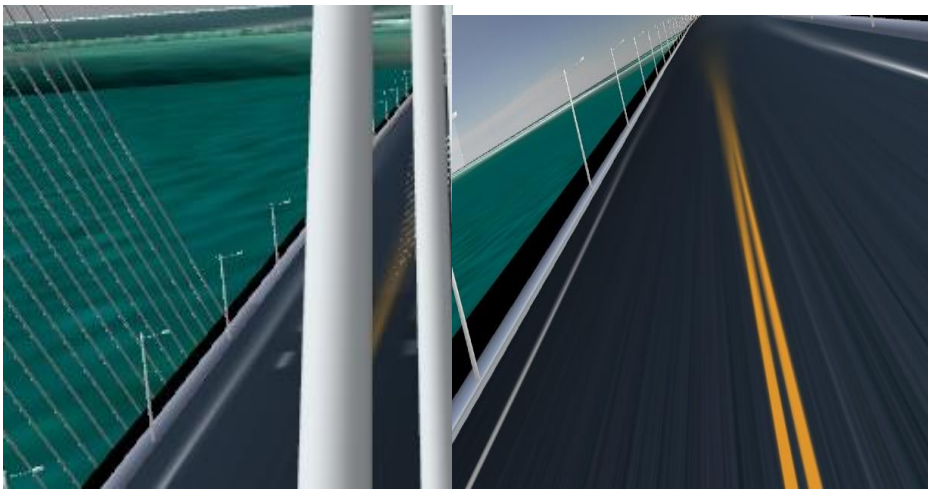
B 功能菜单　　C 系统运行监控

图 4-4 系统主界面

移动视角查看桥梁各部件，通过对模型的平滑渲染处理，可以看到桥梁每个构件的细节非常清晰，层次分明。



A 俯视　　B 仰视



C 桥梁拉索　　D 桥面

图 4-5 桥梁视景漫游查看桥梁细节

系统可实时对桥梁检测出的问题标记和录入，鼠标选择相应问题部件，录

入相应的桥段、桥区、结构和部件名称，并记录桥梁相应部件的问题描述。录入信息存贮到桥梁病害档案数据库，以方便后续的查询和维护。

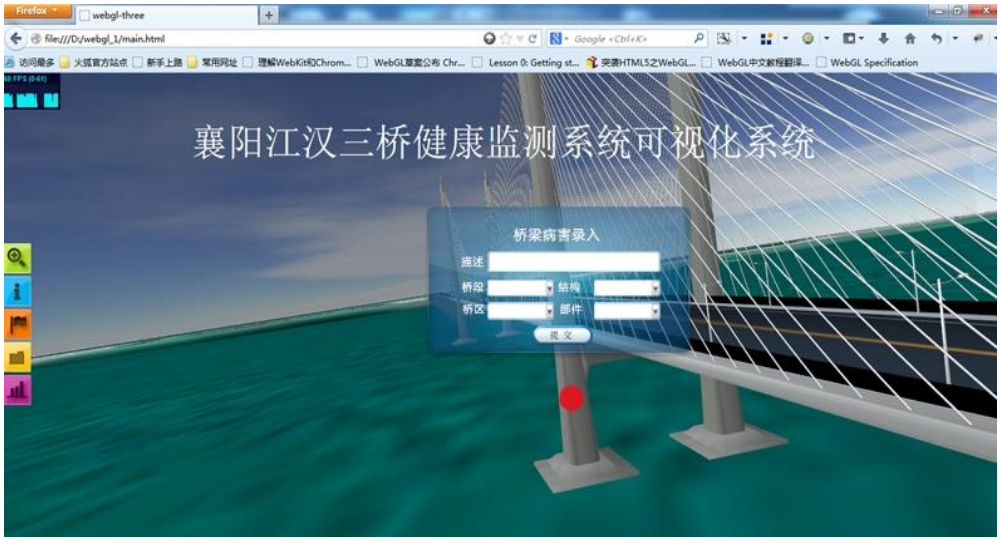


图 4-6 病害实时标记和录入

4.3 本章小结

以襄阳市江汉三桥为例，介绍该桥应用本文研究的 WebGL 桥梁健康监测可视化系统运行状况。通过对襄阳江汉三桥基本结构、周边环境、监测设备的布置情况，利用 3D MAX 设计其 3D 场景模型，并成功移植到 WebGL 环境中，在浏览器中展示系统的启动动画效果，桥梁视景的交互控制，部件细节的查看，对桥梁病害的实时标记与信息录入等直观效果。

第 5 章　总结与展望

5.1 论文工作总结

当今，我国交通环境发展势头迅猛，据统计，我国公路桥梁已经达到 71 万座，特大桥梁 2688 座，大桥 6.2 万座，旧桥老龄化日益严重，而新桥数量也成井喷式增长。桥梁运营的安全和持久，关乎到百姓的切身安全，近五年来已发生的桥梁坍塌事故竟达到 37 起，危桥达到 6000 座以上，桥梁的管养和维护工作正面临着更艰巨的挑战。逐步实现桥梁健康监测的标准化、自动化、网络化、可视化以提高桥梁检测的效率和准确性，是桥梁管养系统的重要方向。

但是，现有的桥梁健康监测系统还存在着非常多的问题：首先，桥梁监测不仅仅是数据的采集，更核心的技术是数据的分析及桥梁健康的预警，通过对桥梁的响应监测获得桥梁各个位置的结构特性和力学状况，如何更准确的对桥梁损伤进行识别，合理评估其安全性，并将其监测数据的分析结果以更科学、直观的手段展现出来。第二，随着科技的发展，原有的桥梁检测手段已不仅限于原有的红外、光纤传感，更不断的演进出如激光智能检测及智能桥梁支座等技术。新技术的应用大大提高了原有检测的科学性和准确性，但同时也相应的对信息的传输、存储以及管理提出了更高的要求。第三，桥梁管理人员原有的管理手段已无法再适应当今规模更巨大、问题更多样桥梁检测，必然需要在原有的基础上智能化，由本地信息升级到全网络，由 PC 端升级到移动、PC 的无缝一体化。

综上所述，桥梁健康监测通过 3D 可视化更直观的反应监测数据和实时预警，将传统的桥梁健康管理系统移植的 B/S 模式并实现 PC 端、移动端的统一是非常有意义的工作。

本文论述了基于 WebGL 技术的桥梁健康监测 3D 可视化系统的设计方案以及 HTML5、WebGL 等相关关键技术的研究与实现，结合襄阳市江汉三桥的应用实例，对本文中的系统设计方案给予验证。总的来说，论文的研究工作和成果如下：

（1）学习和研究了新兴的 HTML5、WebGL 等技术标准及原理。深入探索 WebGL 的三维 GPU 绘制原理、跨平台开放性及其技术框架生态。对比评测出

最适宜于 Web 桥梁 3D 可视化实现的开发框架。

（2）结合相关技术框架，针对于实际桥梁检测管理的实际需求与目标，设计出桥梁健康监测视景仿真系统构架、模块功能、用户接口的最优方案。系统构架方面采用多服务其分布式架构，有利于传感服务器的扩展支持。模块功能方面结合 HTML5、WebGL 技术 GPU 高效渲染、摆脱插件平台限制等特点，将桥梁 3D 场景更大限度的应用于桥梁病害的跨终端标记，及传感器部件的预警渲染。用户接口设计以简洁、友好、3D 效果的动态统一为设计准则，提高了系统的可用性。

（3）研究了基于 WebGL 的桥梁健康监测可视化系统的关键开发技术与实现。首先，结合桥梁实际结构与环境特点，利用 3D MAX 设计具有真实感的桥梁场景，并对场景模型优化，以便更利于 Web 环境的桥梁场景渲染和实现。然后，利用 WebGL Three.js 框架将桥梁场景在 Web 环境中重构渲染，着色优化。第三，研究 Three.js 框架动画及交互原理，实现了桥梁视景的漫游，及桥梁模型交互控制，为进一步实现监测预警，病害查询打下基础。最后，提出了利用 LOD 技术对大型桥梁场景的高效优化方案，进一步提升了系统的可用性和效率。

（4）以襄阳江汉三桥为例，概况的介绍了桥梁的结构状况，并结合本文的设计方案与技术原理将系统成功实践验证，总体达到了直观标记、分析数据的效果。

5.2 未来工作展望

鉴于研究时间与自身知识的局限，本文对于 WebGL 桥梁健康监测可视化系统只进行了有限的研究，其内容的深度还有待进一步的拓展，对于 WebGL 的特性研究及桥梁健康监测系统知识的探索还有许多工作要做。在未来，将进一步的进行以下方面的研究：

（1）基于 Web 的 3D 视景渲染需要进一步的优化其模型的结构，以提高系统的响应效率。将桥梁场景重新分割成多层模型分别载入，针对于传感器和关键部件实现多形式分部渲染，增加系统控制信息，进一步提升系统交互性。

（2）系统中桥梁病害的录入与查询方式上，有待进一步的完善。力求将查询的数据形式覆盖到图像、视频、声音等多种媒介，提升病害数据的可用性和多样性。标记手段将进一步从点标记提升为部件的网面渲染标记，加强仿真的真实感和直观性。

（3）目前，对于桥梁检测数据的分析和评估模型依然有待进一步的开发，将在未来的工作中结合实际桥梁健康监测数据模型原理，进一步研究监测数据在可视化界面的呈现方式，力求实现将原有的二维数据监测图表以三维形式的可视化，增强系统应用实感。

参考文献

[1] 金栋, 李兴田, 张丽萍. 基于 Web 的桥梁可视化管理研究[J]. 兰州交通大学学报, 2012, 31(4): 45-48.

[2] 张云.视频监控系统的发展趋势[J]. 中国科技博览，2011 年 11 期：32-35.

[3] 张启伟. 大型桥梁健康监测概念与监测系统设计[J]. 同济大学学报, 2001, 29(1): 65-69.

[4] 王长波, 杨克俭. 基于 OpenGL 的桥梁视景仿真系统设计[J]. 交通与计算机, 2009 (z1): 79-82.

[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] 张新占. 桥梁管理系统研究[J]. 博士学位论文, 长安大学, 2004.

[7] 祖巧红, 张海峰, 徐兴玉, 等. 基于物联网的桥梁健康监控系统设计与实现[J]. 图学学报, 2013, 34(5): 7-11.

[8] Ortiz Jr S. Is 3 D Finally Ready for the Web?[J]. Computer, 2010, 43(1): 14-16.

[9] Yi H. The New Development Trend in Web3D and Web Visualization——Taking WebGL and O3D as Example [J]. Science Mosaic, 2010, 5: 026.

[10] Nadeau D R. Building virtual worlds with VRML[J]. Computer Graphics and Applications, IEEE, 1999, 19(2): 18-29.

[11] Wei H, Yongfan L, Bing L. Research and optimization on key technology of virtual experiment based on Web3D[J]. Experimental Technology and Management, 2013, 6: 019.

[12] Hong-xiang R, Yi-cheng J, Yong Y. Real-time shadow rendering in scene system of marine simulator[C]//Image and Graphics, 2007. ICIG 2007. Fourth International Conference on. IEEE, 2007: 1010-1014.

[13] 冯科融, 王和兴, 连加美, 等. 基于 HTML5 的 3D 多人网页游戏实现方案[J]. 微型机与应用, 2013, 32(1): 4-6.

[14] Marrin C. Webgl specification[J]. Khronos WebGL Working Group, 2011.

[15] World Wide Web Consortium. Canvas 2D API 规范 1.0 W3C Working Draft,(March 10, 2012). Available online at: http://www. w3. org/TR/html5, 2011.

[16] Danchilla B. Beginning WebGL for HTML5[M]. Apress, 2012.

[17] DeLillo B P. WebGLU development library for WebGL[C]//ACM SIGGRAPH 2010 Posters. ACM, 2010: 135.

[18] Zhanpeng H, Guanghong G, Liang H. NetGL: A 3D Graphics Framework for Next

Generation Web[C]//Multimedia Information Networking and Security (MINES), 2011 Third International Conference on. IEEE, 2011: 105-108.

[19] Behr J, Jung Y, Keil J, et al. A scalable architecture for the HTML5/X3D integration model X3DOM[C]//Proceedings of the 15th International Conference on Web 3D Technology. ACM, 2010: 185-194.

[20] 胡文玲. 基于 X3DOM 的电力系统可视化技术的研究与应用[D]. 山东大学, 2012.

[21] Hering N, Rünz M, Sarnecki L, et al. 3DCIS: A Real-time Browser-rendered 3D Campus Information System Based On WebGL[J].

[22] LIN T H S I. Cloud BIM: A Web-Based BIM System with Application of Cloud Computing and WebGL[J]. 2012.

[23] Jacinto H, Kéchichian R, Desvignes M, et al. A web interface for 3D visualization and interactive segmentation of medical images[C]//Proceedings of the 17th International Conference on 3D Web Technology. ACM, 2012: 51-58.

[24] World Wide Web Consortium. HTML5 specification[J]. W3C Working Draft,(March 10, 2012). Available online at: http://www. w3. org/TR/html5, 2011.

[25] Lubbers P, Albers B, Salim F. HTML5 高级程序设计[J]. 2011.

[26] 张剑, 陈剑锋, 王强. HTML5 新特性及其安全性研究[J]. 信息安全与通信保密, 2013 (5): 87-89.

[27] Wang V, Salim F, Moskovits P. The Definitive Guide to HTML5 WebSocket[M]. Apress, 2013.

[28] West W, Pulimood S M. Analysis of privacy and security in HTML5 web storage[J]. Journal of Computing Sciences in Colleges, 2012, 27(3): 80-87.

[29] Parisi T. WebGL: Up and Running[M]. O'Reilly Media, Inc., 2012.

[30] Anyuru A. Professional WebGL Programming: Developing 3D Graphics for the Web[M]. John Wiley & Sons, 2012.

[31] 刘爱华. 基于 3G WebMapper 和移动通信技术的三维校园社区的构建[D]. 中国海洋大学, 2012.

[32] James L. Williams- Learning HTML5 Game Programming, 2011

[33] Brunt P. GLGE: WebGL for the lazy, 2010[J].

[34] Kim K S, Lee K W. Overlay rendering of multiple geo-based images using WebGL blending technique[J]. Journal of the Korean Association of Geographic Information Studies, 2012, 15(4): 104-113.

[35] Anttonen M, Salminen A. Building 3d webgl applications[J]. Tampereen teknillinen yliopisto. Ohjelmistotekniikan laitos. Raportti-Tampere University of Technology.

Department of Software Systems. Report; 16, 2011.

[36] 刘海娜. 基于 HTML5 的全景漫游技术研究[D]. 郑州大学, 2013.

[37] 程朋根，史文中. 基于 GIS 的桥梁结构健康监测与管理信息系统[J]. 华东公路，2002 (6): 67-71.

[38] 唐练. 插件式桥梁健康监测三维可视化系统研究[D]. 重庆交通大学, 2012.

[39] 谢全宁，雷跃明. 三维可视化桥梁健康监测系统的架构设计[J]. 计算机系统应用，2008, 1: 018.

[40] OpenGL Insights[M]. CRC Press, 2012.

[41] 扈春霞,王子茹. 基于 OpenGL 的参数化斜拉桥三维可视化的研究[J]. 江汉大学学报(自然科学版), 2008, 36(2): 47-49.

[42] 彭国安.3DMAX 建模与动画. 华中科技大学出版社.2012

[43] 施润尼. OpenGL 编程指南 (原书第六版)[J]. 2008.

[44] Sukin I. Game Development with Three. js[M]. Packt Publishing Ltd, 2013.

[45] Coic J M, Loscos C, Meyer A. Three LOD for the realistic and real-time rendering of crowds with dynamic lighting[J]. LIRIS UMR, 2005, 5205.

致 谢

在此论文完成之际，我谨向所有给予我指导、关心和帮助的老师，同学和家人致以衷心的感谢！

首先，感谢我的导师夏红霞教授，夏老师学识渊博，为人谦和，在学习和生活各方面都给予我非常大的帮助。在研究生学习期间，夏老师不仅教授我非常多的专业知识，提供给我很多学以致用的实践机会，并且也让我学习到了很多为人处世的道理，对我今后的工作和生活产生了深远的影响。

同时，我也要感谢邹承明老师，邹老师技术高超，平易近人，在研究项目中给予我很多指导和帮助，提供了很多宝贵的建议。

毕业之际，我向两位恩师致以最崇高的敬意。

感谢实验组的各位同窗好友，在研究生阶段让我收获了知识也收获了友谊。

感谢我的父母对我多年的抚育之恩，多年来无论我做什么样的选择，他们都支持我和理解我，并且倾尽所能的帮助我。

最后，感谢所有参加论文评审和答辩的老师，在百忙这中抽出宝贵的时间来审阅本论文，给予我悉心的指导。谢谢！