基于 WebGL 的在线房产展示系统研究

专业：计算机科学与技术　　年级:2015 级　　学生：邹驼玲　指导教师：黄风华

摘　要

基于 WebGL 的在线房产展示系统研究的提出源于现代网站大多数还是以二维视图来展示，二维视图在表现能力和交互性能方面存在一定的局限。三维技术的应用将为广大房地产销售人员和用户提供一个具有室内外场景展示、场景交互和室内查询功能的虚拟现实的三维展示系统。销售人员可以利用该系统真实感强、采光无死角的优势来辅助销售，从而有效提高客户的视觉感触，加强客户对房型的了解。购房者也可以提前在网上预览，找到符合自己要求的房子。本文主要阐述了运用数据库和 WebGL 技术进行房产查询及三维建筑场景的动态加载展示。

基于 WebGL 的在线房产展示系统的实现技术包括模型的建立、模型的加载和场景交互三个方面，即用 3D 软件实现房屋模型的建立、使用 Three.js 中的 OBJMTLLoader()函数对 OBJ 和 MTL 文本格式模型进行加载，以及引用 dat.gui.js 库实现灯光控制、场景切换、对象拉伸及移动控制。首页面运用 Bootstrap 框架将静态页面变成响应式页面，结合遮罩功能实现查询功能。其中，所建模型包括室外房屋、室内卧室、客厅、卫生间、厨房、餐厅及家具等，上述三维模型以整体场景或个体对象形式被加载显示到页面中。采用 WebGL 的 Three.js 库进行系统开发，用户无需下载使用专用渲染插件，无需下载运行环境，只需借助系统显卡就能在浏览器上展示 3D 场景和模型。

该系统的实现一方面能为普通客户提供了良好的体验，另一方面能降低房产销售人员工作的难度，提高房产交易的效率。系统测试结果表明，基于 WebGL 的在线房产展示系统具能虚拟再现房屋室内外真实场景的效果也有人性化的图形交互性能。

关键词：虚拟现实　　WebGL　　Three.js　　房产展示系统

目　　录

1 绪论

1.1 研究背景

随着社会的发展，时代的进步，互联网技术无处不在，迅速发展的计算机图形学技术应用在生活的方方面面。现代展示趋势不再是静止、被动、实地考察方式，而是用动态、主动、虚拟在线方式推动展示。虽然，现在在市场上，二维视图的应用非常广泛，几乎覆盖了各行各业，但是，在二维视图的设计实现过程中，仍然存在许多不足处，它太过于抽象，不够直观、不能清晰地表达和解决一些更为复杂的空间现象，不能给人 360 度的实景感触，非专业人士很难看懂二维图纸，甚至完全看不懂，无法理解图纸内容。尤其在房产展示时，因为房屋内部和外部的空间结构及形态的多样化和高复杂度，人们对房产展示系统的要求不再满足于传统二维视图平面、静止的表现效果和交互性，对房产实景展示更倾向于追求立体、动态的效果，在交互性能方面也提出了更高的要求。

1.2 研究目的与意义

本次毕业论文的选题确定为“基于 WebGL 的在线房产展示系统研究”的目的是，针对二维房产展示在表现能力、场景交互性能方面的局限性，利用无需任何渲染插件的 WebGL 技术进行三维展示研究。在不用另外下载运行环境，仅使用浏览器进行房屋模型的动态展示。

近年来，三维可视化技术的应用日益增多，国内外各行各业都陆续推出此类系统，市场上大多数 3D 展示系统只实现模型的移动、旋转、缩放功能，并未涉及到模型的查询功能。然而查询显示在展示时又尤为重要。[1]比如，在大商场中，我们经常会遇到找厕所难的问题，若是有了查询功能，我们可以直接找到厕所的模型、位置等相关信息。本论文将引入数据库查询功能。开展基于 WebGL 的在线房产展示系统及相关课题的研究符合信息化社会的发展趋势，具有重要的理论和现实意义。

1.3 国内外研究现状

国内房产销售现状还处于二维展示阶段。在大街上，能够发现很多宣传单，都是简单的附上一两张室内的照片。在各大网络销售平台，也依旧是二维图片的轮播展示。三维房产的展示系统在国外已经开始应用，而国内市场相对较少。

在过去，三维图形的渲染需要用专门的高配置计算机才能实现。但近年来，VR（虚拟现实）技术一直扮演着“外衣”角色，不少房地产服务商，尝试用 VR 技术，融合实景拍摄技术和空间测量技术来呈现虚拟看房服务，为传统三维房屋展示添加了科技感，带来更

为直观、真实的看房体验，但是由于价格相对昂贵并不适用于小型的房产展示[2]。在个人计算机和浏览器性能不断增强的条件下,三维图形的创建和渲染也能通过 Web 技术来实现。虚拟现实三维房产技术是一种价格相对低廉，展示效果出众的技术。市场中，硅谷创业公司 Matterport 是这一类服务商的经典代表。三维建模派和三维实景派这两大技术流派广泛应用在期房市场和现房市场。3D 技术广泛地用于设计阶段，在建设阶段和展示方面的应用还很少见，不过现在这种状况正在改变。因此，本文在房产展示时运用三维软件对现实场景进行虚拟再现，建立三维景观模型，使用 Three.js 库可简化直接用 WebGL 编程来创建三维动画场景的复杂度，可以进一步提高在线房产实景展示的交互性，从而解决空间形态难以表现问题。

1.4 主要研究内容

基于 WebGL 的在线房产展示系统研究的主要内容包括建立 3D 房屋模型、进行系统的场景搭建、在页面中加载 3D 模型、实现场景及对象的交互和数据库查询显示等功能。

1.5 结构安排

第 1 章：绪论。首先简单介绍项目的研究背景，阐述了视图展示由二维转三维的趋势，国内外房产展示的现状和研究该项目的必要性。

第 2 章：阐述 Web 页面三维展示核心技术。本章节主要介绍了基于 WebGL 的在线房产展示系统采用技术、工作原理和相关框架等。

第 3 章：系统设计。本章节主要介绍了系统的主要功能及设计过程，完成系统整体功能模块设计。

第 4 章：系统实现。分三点阐述，分别是模型的创建与实现、事件监听处理实现场景交互、可视化处理实现光照交互和模型显示切换。

第 5 章：系统测试。本章测试主要时针对模型加载进行展示测试、交互测试和数据库查询功能测试。

最后，是对本系统的完成情况进行工作总结，描述系统实现了哪些功能，还存在的哪些不足之处，如何进行改进。

1.6 本章小结

本章主要介绍了课题的背景和研究的目的和意义，分析了三维展示在国内外的现状，描述了本课题研究的内容，最后列出了论文的组织结构。

2 Web 页面三维展示核心技术

系统运用 Bootstrap 框架来制作响应式页面，运用 MySQL 数据库来存储信息，使用 HTML、JavaScript 和着色器语言（GLSL ES）开发 WebGL 程序，WebGL 使用 Three.js 开发框架能降低开发难度。

2.1 HTML5 标准概述

HTML5 是网页的制作不可或缺的一部分。HTML 的发展突飞猛进，2008 年 HTML5 的第一份正式草案公布。HTML、CSS 和 JavaScript 是 HTML5 的三项主要技术[3]。HTML 用来描述超文本中内容的显示方式；CSS 用来控制 HTML 文档内容的显示效果，提供了丰富的字体、颜色、位置、边框等样式外观，使格式化处理内容完美呈现在用户面前;JavaScript 是通过使用事件处理程序对页面进行操作的，从而响应用户的各种操作,为 HTML 增加动态效果。HTML5 引入了很多新特性,其中最大的特点是 HTML5 引入的 canvas 画板标签[4]。canvas 定义了网页上的绘图区域，在该区域内可以使用 JavaScript 动态地绘制任何图形，解决了 img 标签只能显示静态图片,不能进行实时绘制和渲染的问题。在一个页面定义多个 Canvas 应用时，必须将对应代码分开，将变量和函数进行封装能够避免 canvas 在屏幕上独占特定区域的问题。HTML5 的使用让互联网标准有了重大飞跃，大量优秀的网页作品不断涌现。HTML5 能快速迅猛的发展不仅是增加了这些新特征的原因，还有一个重要的原因是浏览器的支持。网站通讯流量监测机构 StatCounter 公布了 2018 年全球浏览器市场份额比例如图 2-1 所示。

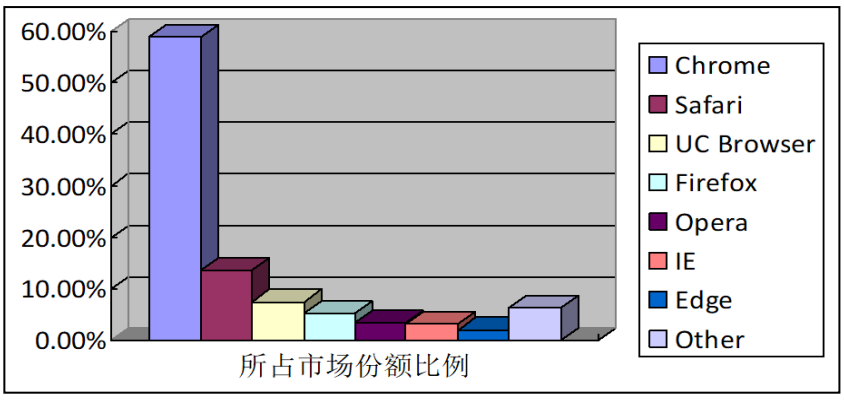

图 0-1 2018 年全球浏览器市场份额

Chrome、Firefox、Safari 、Opera 以及 Mozilla 都支持 HTML5 特性。再结合上图可知，支持 HTML5 的浏览器用户在节节攀升，不支持的则在逐步萎缩。

2.2 Bootstrap 框架

Bootstrap 框架常用于制作响应式页面。它里面含有基本 css 样式、交互组件以及 js 文件。通过这些内容，可以制作出响应式页面。在 Bootstrap 这个框架里，网格系统有着非常重要的作用。网格系统是通过行和列来确定的，列最多不能超过十二个。网格系统的工作原理是行必须放在容器 container 内，以便获得适当的对齐，通过内边距来创建对应列。

在页面轮播图部分可以通过采用轮播插件来实现。这与静态页面里的写 js 的方式不同，它不需要写很多的 js 文件，只需要用到 div 里的 class 进行开发。通过基本结构、CSS、组件组合、JavaScript 插件的组合，一个响应式页面就完成了。为了让设计的页面更加灵活，通过调用 singlePageNav 插件里的方法来让页面跳转更具有条理性，使用 wow 插件和 animate 样式让页面的进出方式更具有生动性，使用 clipboard 插件可以进行文本复制，让文本使用起来更加方便。

2.3 WebGL 工作原理

2.3.1 WebGL 简介

WebGL 结合了 HTML5 和 JavaScript 的优点，是一项能够实现在网页上绘制和渲染复杂 3D 图形的技术，有 3D 绘制标准之称，在网页游戏开发时该技术经常被用来创建和设计一些 3D 网页[5]。HTML5 提供的 Canvas 标签的 3D 图形就能够有效烦的提高渲染速度；网页交互可由 HTML 脚本实现三维动画效果[6]。WebGL 技术的使用将带来一种直观便捷的用户界面，方便用户与之进行交互。有了 WebGL 技术的支持，在三维网页设计时三维图形的创建和渲染条件将不再那么高，可以没有专业编辑器、可以没有专用渲染插件，同时也无需高配置计算机，只要增加一个 JavaScript 绑定，通过本机电脑系统显卡来驱动即可在普通浏览器中展示 3D 视图，再与导航技术和数据面板结合便可实现全景漫游展示[7]。WebGL 应用程序通常在系统开发完毕后都要将应用部署到 Web 服务器上，然后只需提供一个支持 WebGL 的浏览器用来运行项目程序即可。WebGL 开发常用的 JavaScript 库有 SceneJS、PhiloGL、Babylon.js 和 Three.js。本系统采用的是 Three.js 开发框架。

2.3.2 着色器与渲染管线

渲染管线的工作原理：首先向程序中输入需要被渲染的三维模型数据，其次，进行渲染管线处理，最后输入的数据将会以帧为单位输出图像。渲染的实质是执行绘制过程[8]。

顶点着色器顾名思义是用来处理顶点几何信息的程序。顶点在经过着色器进行顶点变化、光照、材质处理后的会产生纹理坐标、颜色、点坐标等各中顶点属性。

片元着色器是以片元为单位来执行纹理采样、颜色汇总和雾计算的程序[9]。例如，光

照就是经过片元着色器操作处理的。下面通过图 2-2 来描述从执行 JavaScript 程序开始到在浏览器中显示结果的过程。

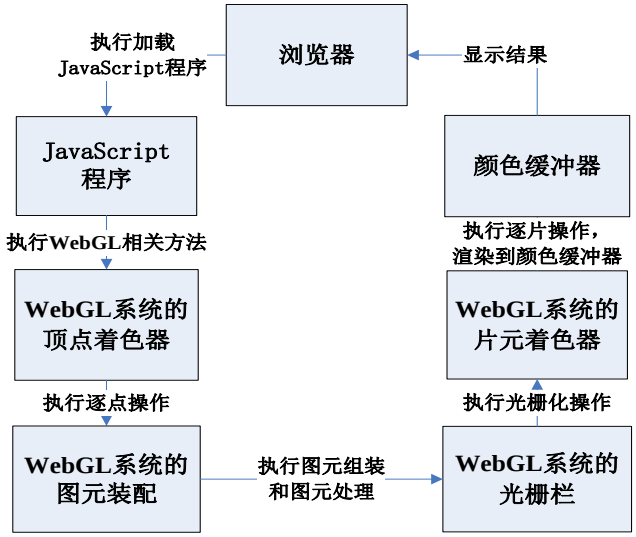

图 0-2 程序执行流程

程序的执行流程:首先，在浏览器上执行 JavaScript 程序；其次，在 JavaScript 程序中调用 WebGL 相关方法；再次，WebGL 系统的顶点着色器执行逐点操作、图元装配阶段执行图元组装（顶点数据结合绘制方式组合成图元）和图元处理（完成剪裁工作，将剪裁平面所定义的半空间外的几何图元剪切掉）、执行光栅化处理（将虚拟三维世界的物体投影到视平面，将连续的数学量离散化得到片元）、接着在片元着色器执行逐片操作后，在缓冲区内执行图形的绘制；最后在浏览器中显示绘制的结果。

2.3.3 正交投影与透视投影

要在二维显示屏上显示虚拟 3D 世界的几何信息，需要将虚拟 3D 世界的连续数学量几何信息物体进行分解片元并投影到视平面上。首先，要有相机，Three.js 库提供正交投影摄像机和透视投影摄像机。[10]在虚拟三维场景内，只有借助于相机工具用户才能看的到场景内的环境，摄像机相当于人的眼睛，没了摄像机就好比一个瞎子，无法看到万千实物景观，因此无法对世界进行观察。其次，确保场景中物体能投影到近平面上，需将要显示的对象放在视景体（渲染管线所确定的可视空间区域）内，随后将近平面上的投影内容渲染映射到屏幕的窗口上。

正交投影摄像机所拍摄出来的所有物体渲染出来的尺寸大小都是一样的。因为正交投影是一种平行投影，它的投影线始终是平行的，尽管改变了摄像机的拍摄位置，在近平面上得到的投影对象都无法实现“近大远小”效果，它的视景体区域始终都是一个长方体[11]。

一般采用透视投影透来实现现实世界中人眼所看见的物体的“近大远小”效果，因为

透视投影的投影线不平行的，会相交于视点，得到锥台形的视景体区域的透视视图。当待投影对象距离摄像机越远时，被渲染的就越小，反正，距离越近所渲染的物体呈现便会越大。因此，相机的位置相当重要，它决定了用户观察的视点。系统中场景会发生移动的实质并非场景在移动，而是摄像机从不同点所拍摄场景不同，所带来的效果就是场景在移动。其摄像机透视投影示意图如图 2-3 所示。

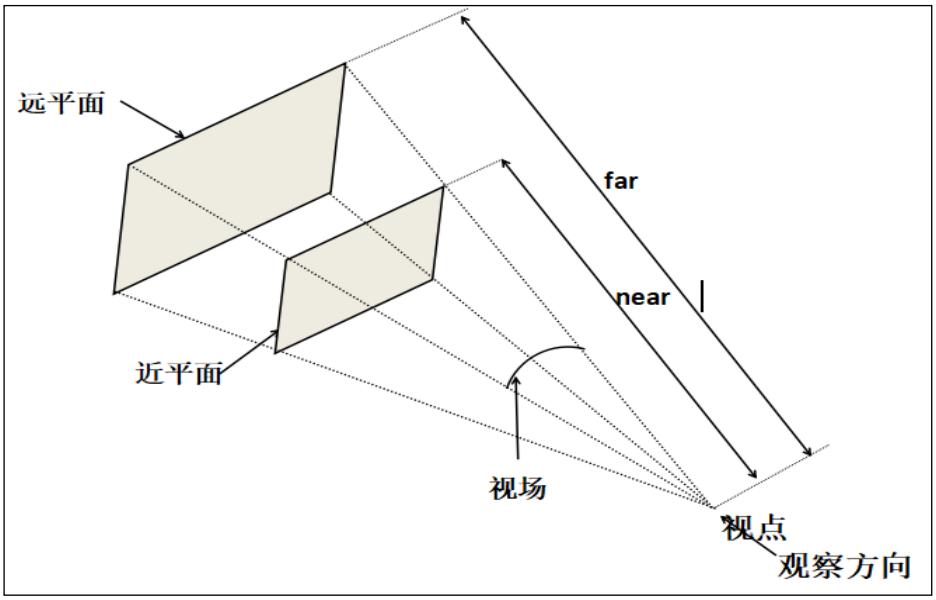

图 0-3 摄像机透视投影示意图

只要横向视场由摄像机的视场（fav）属性决定后，基于纵向视场的长宽比（aspect）属性也会相应确定。变焦（zoom）属性的使用可缩小放大场景。near 属性决定近面距离，far 属性决定远面距离，而远近平面间的区域将会被渲染。

2.4 Three.js 技术与应用

Three.js 是一种采用 JavaScript 语言的编写的能够直接在浏览器上运行的 3D 引擎框架[12]。在三维展示中它如地基般存在，对 3D 领域的开发起到推动作用，为开发者提供了很多丰富的 3D 显示的功能。Three.js 的使用，简化了直接使用 WebGL 编程来创建三维动画场景复杂度，且不容易出错，最重要的一点在于不需要展望太多大的图形学知识，只需要使用少量的 JavaScript 代码就能创建 3D 场景。场景就像一个容器，Three.js 只要使用 scene.add()语句就能快速简便地将各种对象（cameras 摄影机、lights 光源、objects 物体、materials 材质等）添加到 scene 场景中，使用 scene.remove()函数就将其物体从场景中移除。因此，通过 Three.js 可以创建很多炫丽的 3D 场景。

Three.js 定义的场景、摄像机和渲染器对象是页面渲染和模型展示的关键[12]。如果没

有 THREE.Scene 对象，将无法进行 Three.js 渲染。场景就好比一个桶，只有放在桶内的水才是有效的，不会丢失的。如果场景是一个球体，那么摄像机会在球体内部充当一个观察者，相机的位置和方向决定了能够看到在场景内的什么对象，所看到的这个画面都是在相机当前可见范围内的，往往是不完整的。就像现实中的人，当我们站在不同的位置看不同方向所看见的环境都是不同的。渲染器会基于摄像机的角度来计算场景对象浏览器中渲染成什么样子，是实时进行的过程。屏幕上显示的画面是将 3D 空间内的物体映射到摄像机内经过渲染得到的。场景用于存放物体，摄像机则用于拍摄场景，而渲染器把相机拍下的场景传到浏览器上显示。

Three.js 除了场景、摄像机和渲染器这三个重要组件外，光源、几何对象和材质也是不可或缺的组件。运用 WebGL 创建几何对象需要给出对应顶点位置或从外部进行加载，过程繁琐负责。而 Three.js 提供很多自带的几何对象构造函数，开发人员只需要调用对应的 API 即可实现几何体的创建。Three.js 不仅提供了很多几何体对象，还提供了六种灯光源，其中包括了点光源、环境光、聚光灯、平行光、半球光和面光源。通过 Three.js 库来创建光源简单、快捷、效果好，比 WebGL 需要在着色器中加入一系列计算过程会简便很多[12]。Three.js 也有很多自带的材质。材质是实现场景显示的关键。几何对象只表示大小和形状，它只有与描述颜色和透明度的材质相结合才能构成对象。

2.5 MySQL 数据库

信息时代很多的数据都是依靠数据库进行存储的，数据库服务器提供了数据查询、更新的访问操作。其中，Oracle 和 MySQL 是世界上较为流行的数据库服务器。两者相比较可知，Oracle 虽然功能强大，但是前期投入相对较大，访问速度比 MySQL 慢；而 MySQL 提供了多种 API 接口，能够存储多种数据类型，支持多平台，多线程服务，因而非常受广大用户喜爱。其中，还有一个最主要的原因在于他不仅安全还是免费的。

2.6 本章小结

本章除了简单介绍 HTML5 标准的新特征，阐述了 HTML5 快速发展的原因，对 MySQL 数据库相关知识进行阐述，对三维展示的核心技术 WebGL 工作原理和 Three.js 技术进行了介绍。

3 系统分析与设计

在进行系统开发前需要进行系统需求分析，在了解系统需要实现哪些功能后进行系统功能结构和数据库设计，掌握系统设计的技术流程。

3.1 系统需求分析

随着计算机图形学技术的发展，房产销售是生活中不可或缺的一部分，房屋模型的展示是至关重要的，然而，现在大多数的系统都是以二维方式进行展示的，无法实现三维模型的展示。三维展示系统能够提供直观、真实的实现对房屋模型的展示。下面对系统功能进行需求分析：

首先，需要建立 3D 模型为系统开发实现模型加载显示功能做好材料准备。下面将需要建立的模型进行罗列：

（1）室外模型：包含路，花草树木、椅子、凉亭、车子、房屋等。

（2）室内模型：是所有房间模型的组合。

（3）房间模型：包括卧室、卫生间、厨房、书房、餐厅和客厅等几个房间。

（4）家具模型：在每个房间内都有不同的家具，室内常用的家具有各种桌椅（饭桌、书桌、沙发、茶桌）、衣柜、床、书柜、冰箱、电视这些模型。

其次，系统需要实现光照调节、模型切面、场景交互和对象交互功能。下面将描述每种交互实现了哪些控制。

（1）光照调节：光照的强度及颜色控制、打开和关闭控制。

（2）模型切换：实现将模型添加到场景中，从场景中移除模型。

（3）场景交互：主要实现场景的移动、缩放和平移功能和调整光照强度的功能。

（4）对象交互：主要实现模型切换显示，对对象模型的上下拉伸控制和上下、前后、左右移动。

最后，系统要有数据库查询功能。点击不同的房间会显示出对应房间模型的属性值。

3.2 系统功能结构设计

本系统开发的重点在于三维模型的加载展示与交互操作。在线房产展示系统包含系统展示和系统交互这两大功能模块。系统的功能结构如下图 3-1 所示。

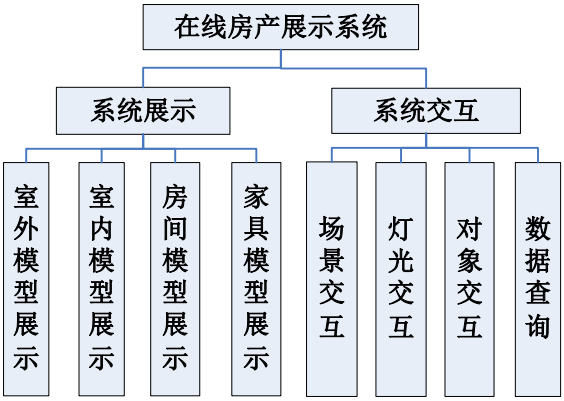

图 0-1 系统功能结构

由以上系统功能结构图可知系统展示模块包括室外、室内、房间和家具模型的展示；系统交互模块包括场景交互、灯光交互、对象交互和数据库查询功能。

3.3 系统技术流程设计

基于 WebGL 的在线房产展示系统可实现对房屋室内外 3D 模型及周边环境的加载显示、并可通过鼠标及定时器来控制场景的移动、旋转、缩放、模型添加与删除、光照交互和数据库查询功能。下面将通过技术流程图来描述系统设计过程中涉及到的关键问题。系统设计技术流程图如图 3-2 所示。

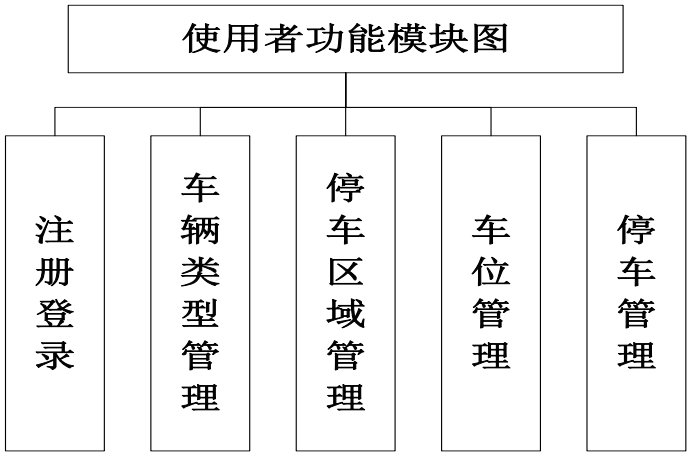

图 0-2 系统设计技术流程图

通过以上流程图可知，系统的设计流程最主要的四个内容分别是：应用 3D 软件建立房屋模型、应用 Three.js 技术实现页面加载显示 3D 模型、事件监听控制场景交互的实现和通过 JSP 与 MySQL 数据库实现连接实现属性查询。

3.4 数据库设计

虚拟在线房产展示系统涉及到三维模型展示和场景交互问题。在系统展示时，希望能够获取所展示模型的基本信息。因此，这里使用了 MySQL 数据库进行房间信息数据存储。其中，房间信息表的属性包括房间编号、房间名称、房间的长宽高和房间的简介。数据库以房间编号为主键作为查询条件，从而查找显示符合条件的整条记录。下面对房间信息表进行物理结构设计，如表 3-1 所示。

表 0-1 房间信息表

字段名	数据类型	功能描述
sno	Varchar	主键：房间编号
sname	Varchar	房间名称
sage	Varchar	房间长度：单位为米
sdepartment	Varchar	房间宽度：单位为米
sintro	Varchar	房间高度：单位为米
simage	Varchar	房间简介：对房间的占地面积和家具进行描述

3.5 本章小结

本章先是对系统进行了需求分析，随后对系统功能结构和技术流程、数据库内容进行了设计。

4 系统实现

前面的章节已经对系统进行了分析与设计，对 WebGL 开发中需要掌握的知识和技巧也进行了详细的介绍，本章将详细介绍系统将如何实现每个功能模块，其中包括了模型的建立、场景搭建、天空盒与模型加载、事件监听、设置可视化界面、模型切换和数据库查询等七个功能模块。

4.1 模型的建立

WebGL 能够对导入的模型进行解析[13]。因此在建 3D 房屋模型时，有两种方案。其中，第一种采用 Three.js 建模，采用程序运行数学公式的方式来创建简单模型，第二种运用建模软件可实现复杂模型的建立，房屋模型相对复杂，适合采用第二种方式，可用的建模软件有 Revit、3dsMAX、SketchUp、Blender。

使用 3D 制图软件建立房屋 3D 模型的流程如下：首先需设计好建筑物周边的抽象场景，提取数据后，先用多边形创建墙体、添加门窗、创建屋顶等初始模型，并添加光照、材质等元素，最后创建门窗、屋顶、家具、墙体等模型结构，添加光照、材质等基本元素，然后进行纹理贴图，提高渲染效果，最后是对所建的模型创建群组，方便后期对该模块进行移动和引用。室内、室外整体场景建模示意图如图 4-1 和图 4-2 所示。

图 0-1 室外场景整体模型

室外场景建设：先建好房屋、树、椅子、车等模型，分别创建群组。再创建矩形，进行草坪贴图、再将之前创建好的模型添加到草坪面上。

图 0-2 室内场景整体模型

室内场景建设：家具模型建好后，创建群组。然后画出房屋区域，进行拉伸创建墙体，再将之前创建好的家具摆放到对应的空间环境中。

4.2 场景搭建

在阐述 Three.js 框架时，已经知道一个三维场景空间展示系统是少不了场景、摄像机和渲染器这三个基本元素，而用户要在场景中显示的物体是主要展示对象。

本次的设计是基于 WebGL 的三维建筑类场景的展示，因此房屋的周边环境也是很重要的，首先，天空盒地面是必要的。如若不采用天空和地面，只是将模型加载到场景中，那么背景不是一片黑就是设置其它颜色都显得很单调，不真实；其次，光照也是必不可少的，如若没有光照，展现出来的将会是一片黑，无法看到我们往场景中添加的对象。最后就是加载主体对象房屋模型了。将模型摆放在合适的位置合适的角度也是很重要的，这样能避免模型间相交发生碰撞，还能让整体画面看起来丰富而有序。天空、地面模型的加载后续会慢慢介绍，这里先对场景、相机和渲染器进行设置，代码如下：

```
//创建场景
function initScene(){
    scene = new THREE.Scene();
//开启雾化效果：颜色军蓝绿、最近距离、最远距离
    scene.fog = new THREE.Fog( 0x5f9ea0, 3500, 15000 );}
//创建相机，并添加到场景中
function initCamera(){
    camera = new THREE.PerspectiveCamera(45, window.innerWidth /
window.innerHeight,1,10000);//相机参数：相机夹角，宽高比，最近距离，最远距离
```

```
    camera.position.set(650,80,-100);//相机的坐标（x,y,z）
    camera.lookAt(scene.position); //将相机添加到场景中
}
// 创建 Three.js 渲染器,并添加到<div>标签中
function initRenderer(){
    //WebGL 渲染器,开启反锯齿，设置背景色为透明
    renderer = new THREE.WebGLRenderer({ antialias: true, alpha: true  } );
    //渲染范围（渲染屏幕大小）
    renderer.setSize( window.innerWidth, window.innerHeight );
    renderer.setClearColor(0x000000); //背景色
    renderer.shadowMapEnabled = true;//接受阴影
    container.appendChild( renderer.domElement );//渲染器添加到 canvas 块
}
//添加轨道控件
function controls() {
    //将轨道控件绑定到相机、canvas 块
    controls = new THREE.OrbitControls( camera,renderer.domElement);
    controls.maxPolarAngle = Math.PI * 0.5; //设置垂直限制
    controls.maxDistance = 4000;//设置最远距离
}
```

4.3 天空盒与模型加载

4.3.1 天空盒

天空盒与天空穹技术可以为场景添加真实天空效果，很大程度的提升了场景真实性，都是实现天空效果的技术。

天空盒技术的实现：首先先创建一个正方体，接着为正方体的六个面添加纹理贴图，实现面与面之间的无缝连接，完成天空盒创建。天空展示效果如图 4-3 所示，实现天空功能的关键代码如下：

```
//载入天空
function tiankong() {
    var imageurl = "../skyboxs/"; //天空图片位置
    var textures = ["RT", "LF", "UP", "DN", "BK", "FR"]; //天空纹理图片
    var imageSuffix = ".png"; //图片的后缀
```

```
//创建一个天空盒的立方体
    var skyGeometry = new THREE.CubeGeometry(8000, 8000, 8000);
    var materialArray = [];//定义一个数组
    for (var i = 0; i < 6; i++)
        materialArray.push(new THREE.MeshBasicMaterial({
            map: THREE.ImageUtils.loadTexture(imageurl + textures[i] +
imageSuffix), side: THREE.BackSide}));//纹理图片
//创建一个 Face 材质来处理着色，并传递给纹理映射
    var skyMaterial = new THREE.MeshFaceMaterial(materialArray);
    //将天空盒的立方体和材质放到一个网络中
    var skyBox = new THREE.Mesh(skyGeometry, skyMaterial);
    scene.add(skyBox);//把天空加入到场景中
}
```

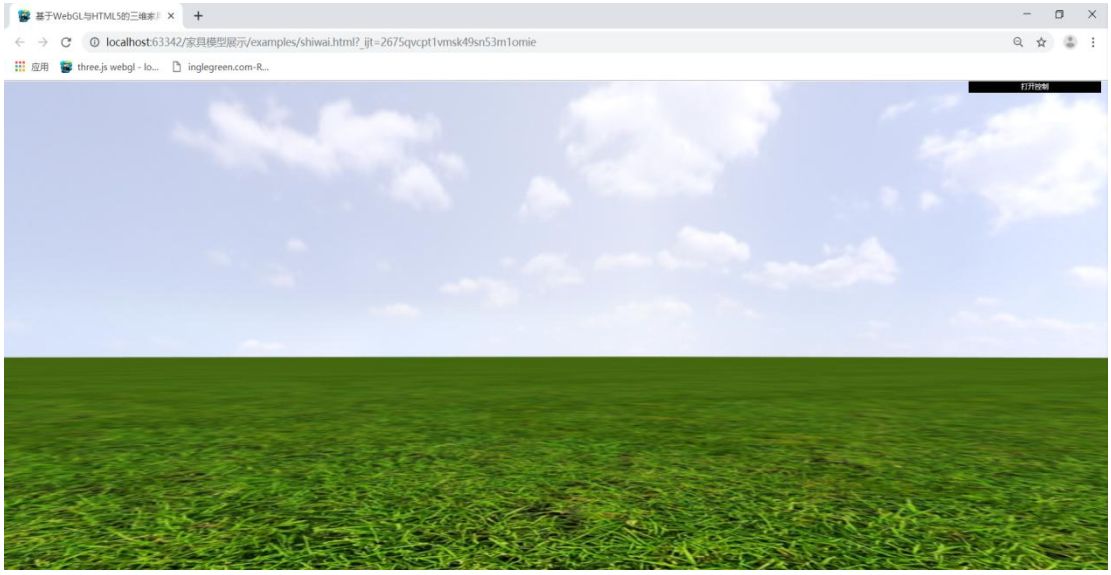

图 0-3 天空

4.3.2 模型加载

在模型加载时，Three.js 能够支持 JSON、OBJ 和 MTL、Collada、STL、VTK,PDB,PLY 等格式的三维文件，无论加载哪种格式模型，WebGL 的 Three.js 库支持对导入的模型进行解析，得到若干 Object3D 的组合对象。[8]

如果使用 Three.js 中的 OBILoader()函数，则只加载模型而没有材质信息，展示效果不尽人意。因此，本文使用 Three.js 中的 OBJMTLLoader()函数加载 OBJ 和 MTL 文本格式模型。首先要创建一个模型加载对象 loader 并调用 OBJMTLLoader()函数，将 OBJ 几何模

型文件和 MTL 材质文件加载到已创建的 loader 对象中，其次对模型的大小、位置和摆放方向进行设置，最后往场景中添加模型。用 WebGL 渲染的电脑桌 3D 模型展示效果如图 4-4 所示，其模型加载的核心代码如下：

```
var loader = new THREE.OBJMTLLoader();// 创建加载器 loader
//加载模型和材质
loader.load('assets/models/diannaozuo.obj',
    'assets/models/diannaozhuo.mtl', function (object) {
        object.scale.set(1, 1, 1);// 设置模型大小
        object.position.set(-2800, -6, 2800); //设置模型坐标
        object.rotation.set(1, 0, 0)//设置模型方向
        mesh = object;//获取模型
        scene.add(mesh);//将模型添加到场景
    });
```

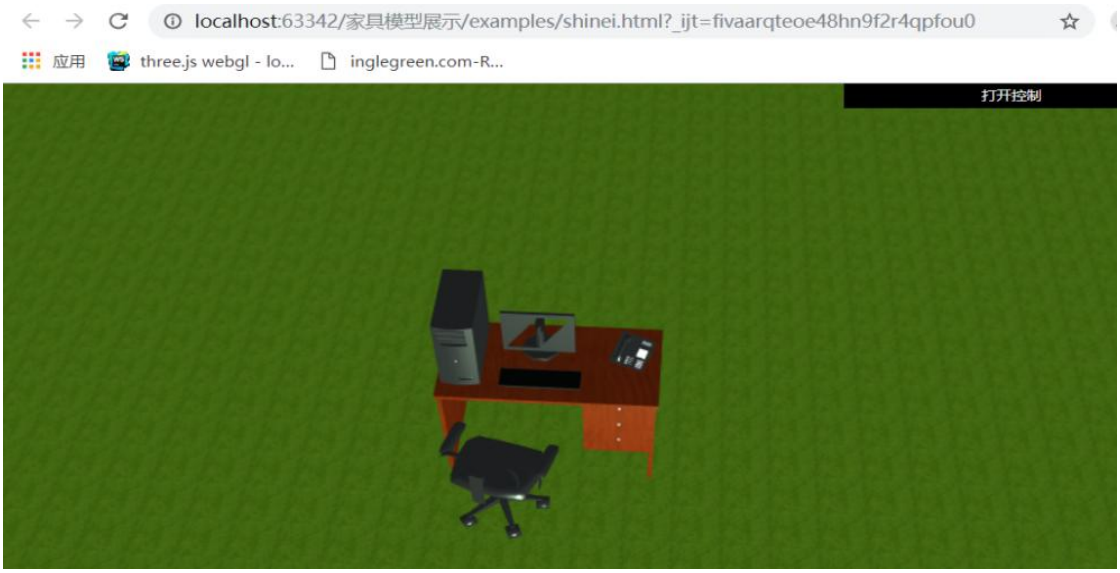

图 0-4 WebGL 渲染的电脑桌模型

4.4 事件监听控制

模型载入场景后，为了让用户从不同的角度来查看模型，系统引用了 OrbitControls.js 封装的鼠标控制类，通过选择鼠标进入场景的相关操作及定时器来觉得对场景进行什么操作（移动、旋转、缩放），其中鼠标的滚轮滚动对应放大缩小的实现；鼠标左键对应的是旋转操作；鼠标右键实现镜头的移动；使用键盘上的“Z、S、C、X”或“→、↓、←、↑”键，可以实现左移、上移、右移、下移的场景漫游。具体操作是按住鼠标左键，进行鼠标拖动，按住鼠标右键，进行鼠标拖动，整个操作过程对鼠标不进行释

放。

旋转控制有两种方式，第一种，使用模型视图投影矩阵来变换顶点的坐标。第二种，使用鼠标来控制物体旋转，根据鼠标移动情况创建旋转矩阵，更新模型视图投影矩阵，并对物体的顶点坐标进行变换。

鼠标控制的实现：在鼠标左键按下是记录鼠标的初始坐标，然后在鼠标移动的时候用当前坐标减去初始坐标，获得鼠标移动的位移，然后再根据这个位移来计算旋转矩阵。其中，需要监听鼠标的移动事件，并在事件响应函数中计算鼠标的位移、旋转矩阵，从而实现旋转。旋转前后运行效果如图 4-5 和图 4-6 所示。

图 0-5 旋转前运行效果

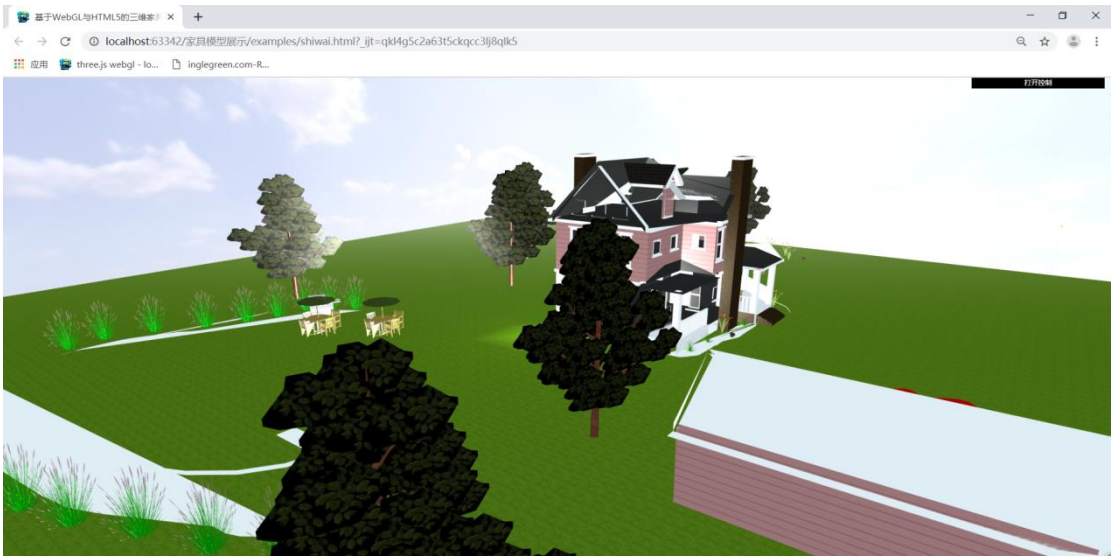

图 0-6 旋转后运行效果

4.5 设置可视化操作界面

为了让用户得到不同的渲染效果，添加了灯光可视化操作界面，系统将引用 dat.gui.js 库，在 addLightAndGUI()方法中定义了光照的颜色、坐标，将光照添加到场景中，创建 controls 和 gui 内容。

4.5.1 基本光照效果的实现

物体立体感的产生是因为光照照射后每个面接受的光照强度不同，从而产生明暗差异。光照照射到现实生活中物体上时会产生一部分反射光。人们能够辨别物体颜色的原因不仅与物体材质有关，还与放射光线是否进入到眼睛有关。光照的影响因素有很多，计算方式也十分复杂。本系统中采用的光照效果进行了很大程度的简化，着色器在执行着色前要先确定光照的类型和物体如何产生反射光，光的三种基本类型：环境光、平行光和点光源光，如图 4-7 所示。

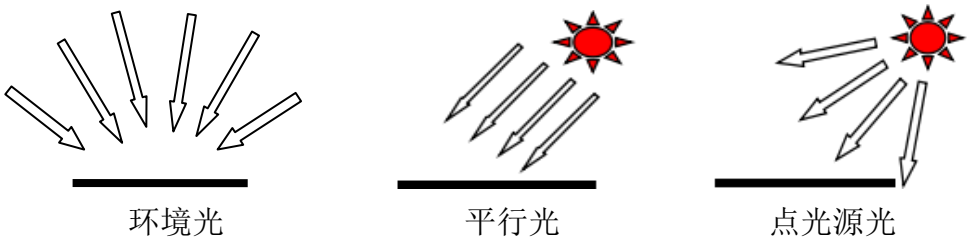

图 0-7 光的基本类型

环境光光照原理：环境光从光源发出光照后，会经过多次均匀反射最终照到物体表面。[13]其特点不仅与光源的位置无关，而且还没有方向性，真正影响环境光的因素是光照的强度和材质的反射系数。

平行光光照原理：它的光线是彼此平行不相交的。平行光相对于环境光会简单很多，只需要一个颜色和一个方向就能表示。[13]

点光源照射原理：从一点向四周发射的光。当光源位置和颜色发生变换时，光照效果变化明显。[13]根据光源位置和被照射位置不同所计算出来的光线方向也是不同的。运行效果如图 4-8 和 4-9 所示，下面将进行点光源光和环境光的创建及光照控制的实施代码如下：

```
var radius = 50;//点光源的圆形轨道半径
var pointLightAngle = 0;//点光源旋转的角度
function addLightAndGUI(){
    var ambiColor = "#42545e";
    var ambientLight = new THREE.AmbientLight(ambiColor);
    scene.add(ambientLight);
```

```
var pointColor = "#828982";//点光源的颜色值
pointLight = new THREE.PointLight(pointColor);//创建点光源
pointLight.position.x =0;//设置点光源的 x 坐标
pointLight.position.y =30;//设置点光源的 y 坐标
pointLight.position.z =0;//设置点光源的 z 坐标
scene.add(pointLight);//将点光源添加到场景中
var controls = new function () {
    this.ambiColor = ambiColor;
    this.pointColor = pointColor;
    this.intensity = 1;
    this.distance = 100;
    this.closePointLight = false;
    this.closeAmbientLight= false;
};
var gui = new dat.GUI();
//点光源颜色
gui.addColor(controls, 'pointColor').onChange(function (e) {
    pointLight.color = new THREE.Color(e);
});
//点光源光线强度
gui.add(controls, 'intensity', 0, 3).onChange(function (e) {
    pointLight.intensity = e;
});
//点光源照射的最大距离
gui.add(controls, 'distance', 0, 200).onChange(function (e) {
    pointLight.distance = e;
});
//是否关闭点光源
gui.add(controls, 'closePointLight').onChange(function (e) {
    pointLight.visible = !e;
});
//环境光颜色
gui.addColor(controls, 'ambiColor').onChange(function (e) {
    ambientLight.color = new THREE.Color(e);
});
```

```
    //是否关闭环境光
    gui.add(controls, 'closeAmbientLight').onChange(function (e) {
        ambientLight.visible=!e;
    });
}
```

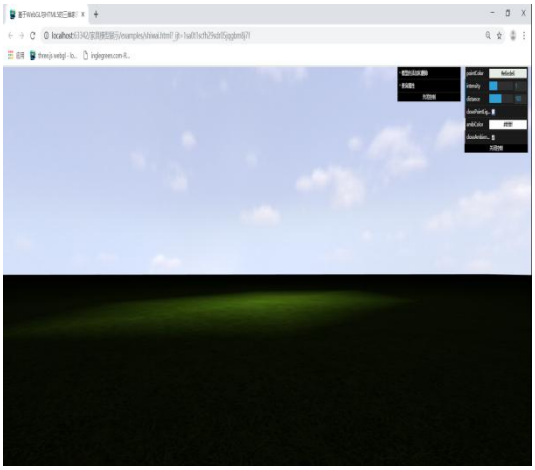

图 0-8 打开点光源运行图

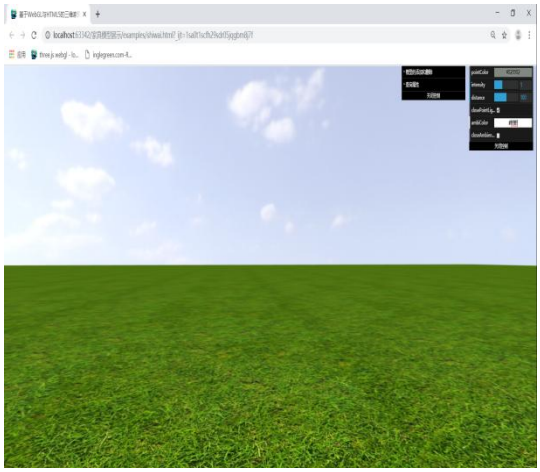

图 0-9 打开环境光运行图

4.5.2 模型切换实现

同理，模型的交互也可以通过可视化界面实现。通过点击跳转功能实现餐桌展示及属性介绍，运行效果如图 4-10 所示，可视化操作界面代码如下：

```
function guiconture() {
    var controls = new function() {
        this.添加餐桌及属性 = function() {
            scene.add(chaxu2);//将属性添加到场景
            scene.add(canzhuo2);//将餐桌添加到场景
        }
        this.移除餐桌及属性 = function() {
            scene.remove(chaxu2);//将属性从场景中移除
            scene.remove(canzhuo2);//将餐桌从场景中移除
        }
    }
    var gui = new dat.GUI();
    //设置 gui 显示内容
    guiSX = gui.addFolder('查询属性');
```

```
    guiSX.add(controls, '添加餐桌及属性');
    guiSX.add(controls, '移除餐桌及属性');
}
```

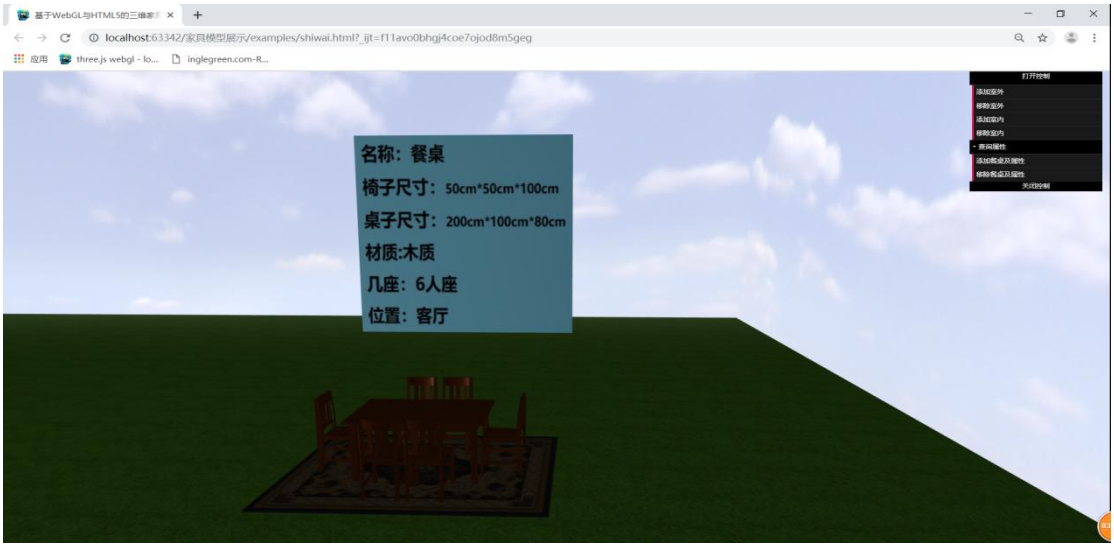

图 0-10 点击跳转展示餐桌模型效果图

4.6 房间基本信息展示

在系统展示时，往往需要知道房间的基本信息，那么需要用到数据库的查询功能来实现对某个房间进行查询。

在 MySQL 数据库中建好 stutable 房间信息表后，在 MyEclipse 环境下通过 JSP 与 MySQL 进行交互。数据库连接过程：先定义 MySQL 数据库连接的 url，接着是加载 MySQL 驱动，获取连接，执行查询，对返回的结果进行 while 遍历，最终在页面中显示符合查询条件的记录，如图 4-11 所示，连接遍历代码如下：

```
public class DBManager {
    private static ArrayList<StudentBean> queryStu (String sql){
        ArrayList<StudentBean> list = new ArrayList <StudentBean>();
        String url = "jdbc:mysql://localhost:3306/school?";
        Connection con = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection(url, "root", "123456");
        if(con != null){
            Statement st=con.createStatement();
            ResultSet rs=st.executeQuery(sql);
```

```
            while(rs.next()){
                StudentBean bean= new StudentBean();
                String sname= rs.getString("sname");
                String sno= rs.getString(" sno");
                String sdepartment= rs.getString("sdepartment");
                String sintro= rs.getString("sintro");
                bean.setSname(sname);
                bean.setSno(sno);
                bean.setSage(rs.getInt("sage"));
                bean.getSdepartment();
                bean.setSintro(sintro);
                list.add(bean);
            }
        }else{
            return null;
            }
        } catch (ClassNotFoundException e) {

            e.printStackTrace();
            return null;
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            return null;
        }
        return list;
    }
}
```

图 0-11 查询结果显示

4.7 本章小结

本章主要介绍了系统的实现要知道模型如何建立，场景搭建后页面才能进行模型加载，对系统进行场景交互、对象交互和灯光交互实现。

5 系统测试

本章主要是对系统的主要功能模块进行了黑盒测试，通过系统测试可以发现系统在分析和设计过程中出现的错误，可以知道对应功能模块是否达到预期要求。下面将描述系统的测试环境并进行模拟测试，经过测试分析得出测试结论。

5.1 测试环境

在 Window10 的操作系统上借助系统显卡在 Chrome 浏览器上执行加载 JavaScript 程序，从而实现三维场景和模型展示，运行环境如下表 5-1 所示。

表 0-1 运行环境

环境	详情
浏览器	Chrome
操作系统	Window10
显卡	Intel(R)Iris(TM)Pro Graphics 5200,NVIDIA GeForce GTX950M
数据库	MySQL
开发工具	WebStorm，MyEclipse

系统采用了 WebStorm 进行 3D 开发，在 MyEclipse 上实现了与 MySQL 数据库的连接。

5.2 系统功能模块测试

5.2.1 系统展示

下面将对系统展示模块的室外、室内、房间和家具模型进行展示测试，从而证明模型加载功能是否得以实现，测试结果如图 5-1 至图 5-4 所示。对系统模型切换显示功能进行测试用例设计如表 5-2 所示。

表 0-2 模型切换展示测试用例设计

用例设计	用例测试	预期结果	实际结果
进入展示页面，执行整体模型切换操作	点击整体模型下的添加室外、添加室内后查看显示结果是否是对于模型	在点击添加室外时，显示室外模型，再点击添加室内时会删除室外模型显示室内模型。	同预测结果

续表 0-2

用例设计	用例测试	预期结果	实际结果
进入展示页面，执行房间模型切换操作	在房间模型下点击添加不同房间名称后，查看显示模型是否与点击名称相对应	在点击添加卧室时，显示卧室模型，再点击添加客厅时会删除卧室模型显示客厅模型，同理，点击厨房、书房、卫生间、餐厅也是如此。	同预测结果
进入展示页面，执行家具模型切换操作	点击家具模型下的添加沙发、添加餐桌后查看显示结果是否是对于模型	在点击添加沙发时，显示室外模型，再点击添加餐桌时会删除沙发模型显示餐桌模型。	同预测结果

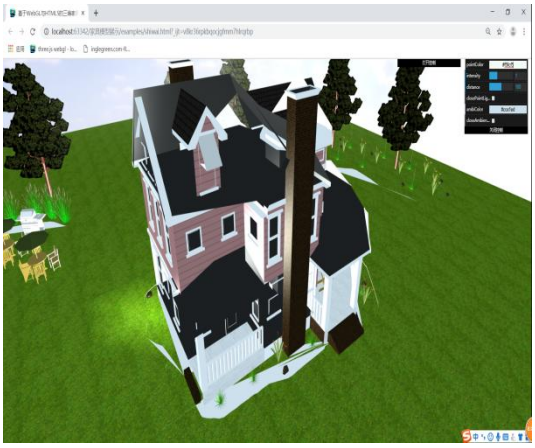

图 0-1 室外场景效果图

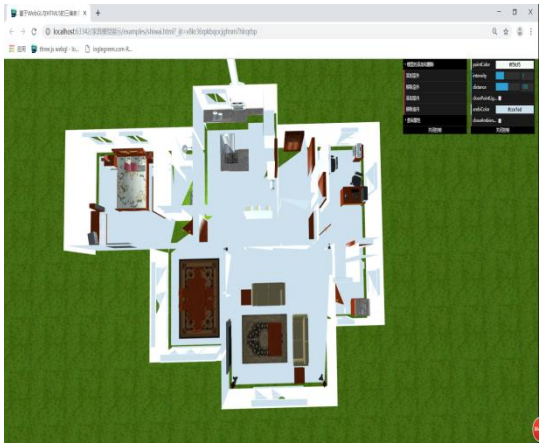

图 0-2 室内整体效果图

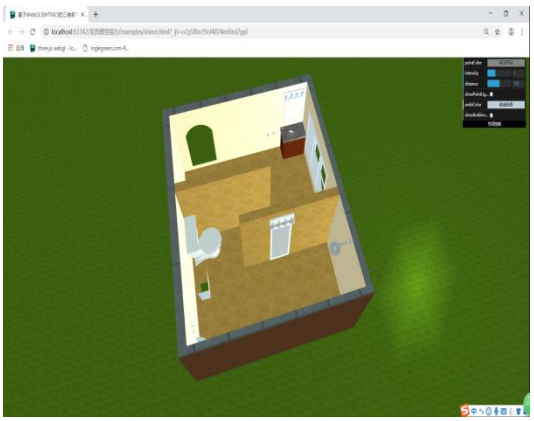

图 0-3 卫生间展示效果

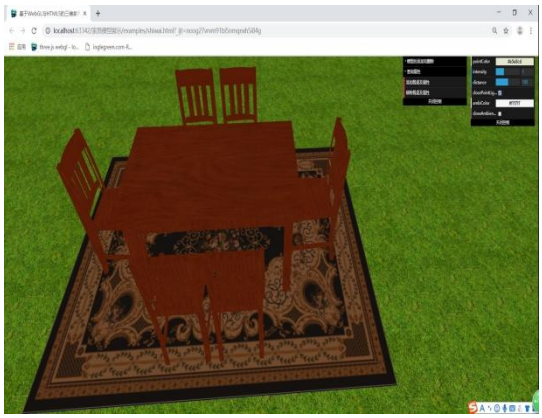

图 0-4 餐桌展示效果

以上测试结果表明，无论是室外、室内、房间还是家具模型能够加载成功，证明了该系统的模型加载显示功能已实现。

5.2.2 场景交互

（1）场景移动测试

下面将在展示卧室模型时对整个场景进行上下左右四个方向进行测试。如果能实现四个方向的移动则表示场景移动成功，反之，则表示移动失败，测试结果如图 5-6 至图 5-9 所示。对场景移动功能进行测试用例设计如下表 5-3 所示。

表 0-3 场景移动测试用例设计

用例设计	用例测试	预期结果	实际结果
进入展示页面，执行场景移动操作	按住鼠标右键进行拖动或按键盘上的“Z”、“C”、“S”、“X”、“→”、“↓”、“←”、“↑”键，查看是否实现上下左右移动	按住鼠标右键进行拖动可实现上下左右四个方向的移动；按住“X”或“↑”实现场景上移；按住“S”或“↓”实现场景下移；按住“C”或“←”实现场景左移；按住“Z”或“→”实现场景右移；	同预测结果

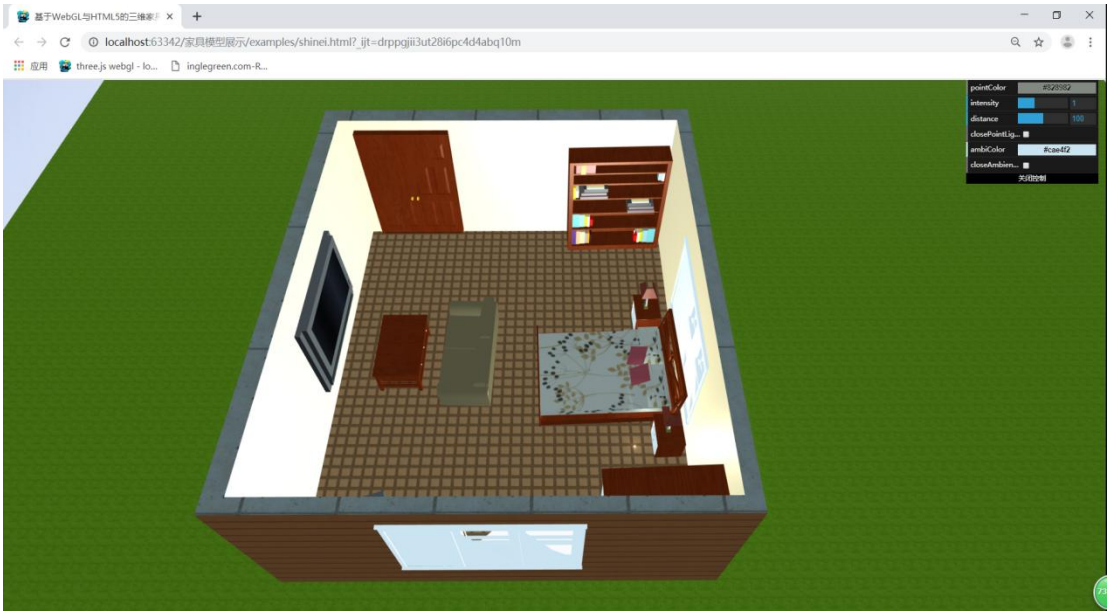

图 0-5 移动前

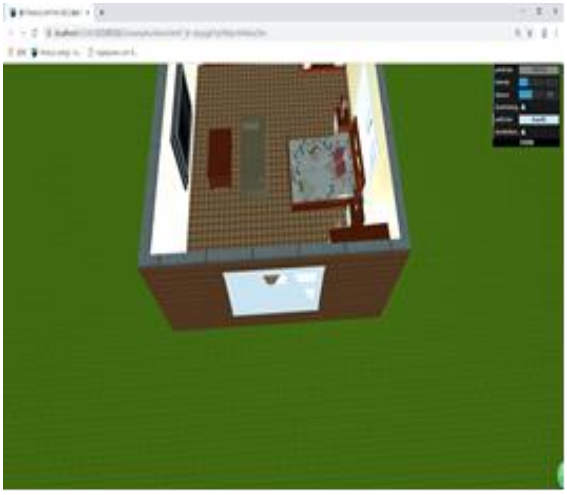

图 0-6 按住 X 向上移动

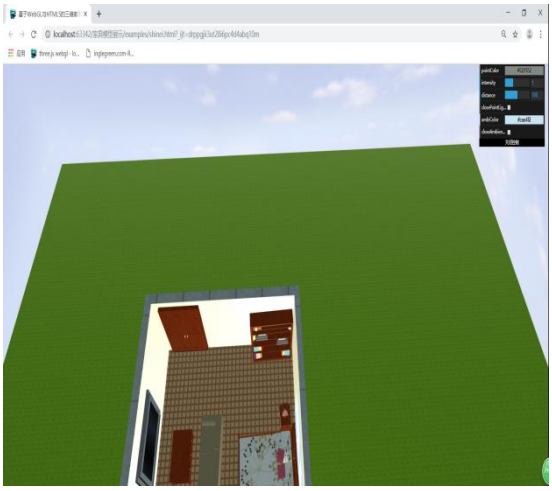

图 0-7 按住 S 向下移动

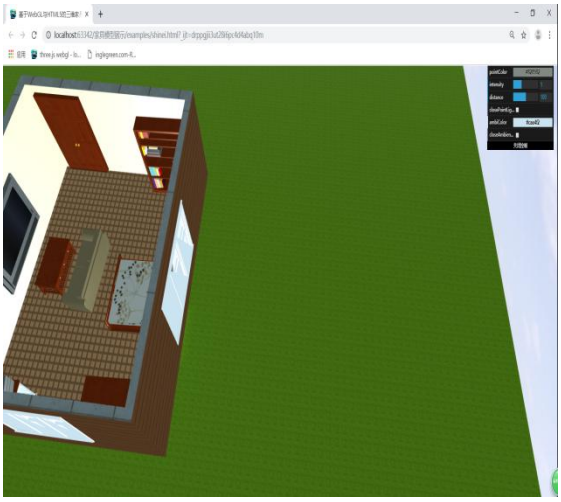

图 0-8 按住 C 向左边移动

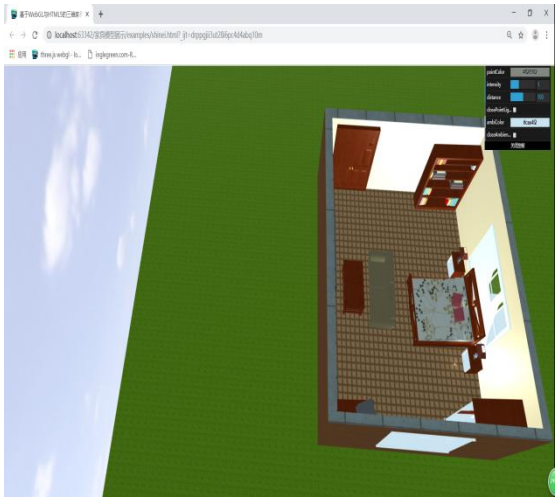

图 0-9 按住 Z 向右边移动

以上测试结果表明，系统能够通过按住鼠标右键进行拖动实现场景移动，也可以通过方向键盘“Z”、“C”、“S”、“X”、“→”、“↓”、“←”、“↑”键控制移动。

（2）场景缩放测试

下面选择展示客厅模型时对场景进行放大和缩小的测试。如果场景能够放大或者缩小则表示系统实现了缩放功能，反之，则系统缩放功能存在问题，测试结果如图 5-10 至图 5-11 所示。对场景缩放功能进行测试用例设计如下表 5-4 所示。

表 0-4 场景缩放测试用例设计

用例设计	用例测试	预期结果	实际结果
进入展示页面，执行场景缩放操作	上下滚动住鼠标滚轮进行拖动，查看场景是否被放大或者缩小	鼠标滚轮向前滚动时，场景将会被放大；鼠标滚轮向后滚动时，场景将会被缩小；	同预测结果

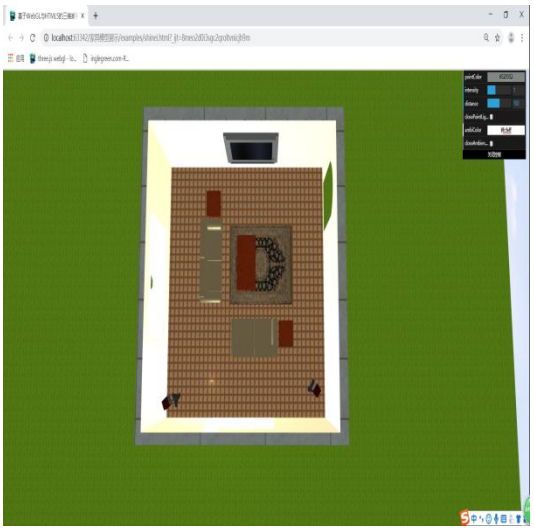

图 0-10 缩小状态效果图

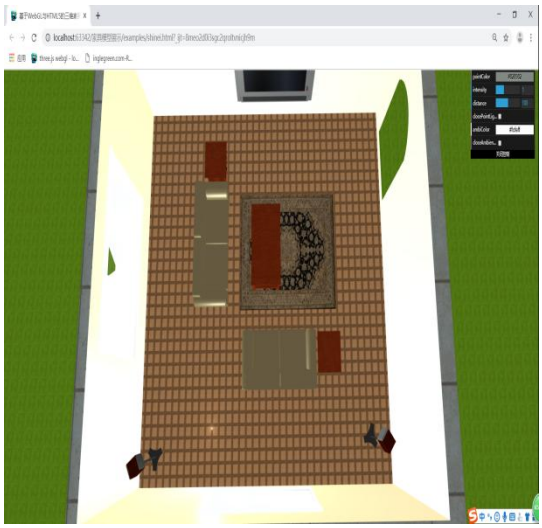

图 0-11 放大状态效果图

以上测试结果表明，系统能够通过滚轮滚动来实现场景的放大或缩小。系统的缩放功能是成功的。

（3）场景旋转测试

下面选择展示书房模型时对场景进行旋转测试。如果场景能够从当前展示角度到另一个角度进行展示则表示系统实现了旋转功能，反之，则系统旋转功能存在问题，测试结果如图 5-12 至图 5-13 所示。对场景旋转功能进行测试用例设计如下表 5-5 所示。

表 0-5 场景旋转测试用例设计

用例设计	用例测试	预期结果	实际结果
进入展示页面，执行场景旋转操作	按住鼠标左键进行拖到，查看页面显示结果是否为旋转后的效果	按住鼠标左键进行拖动可实现上下 360 度旋转	同预测结果

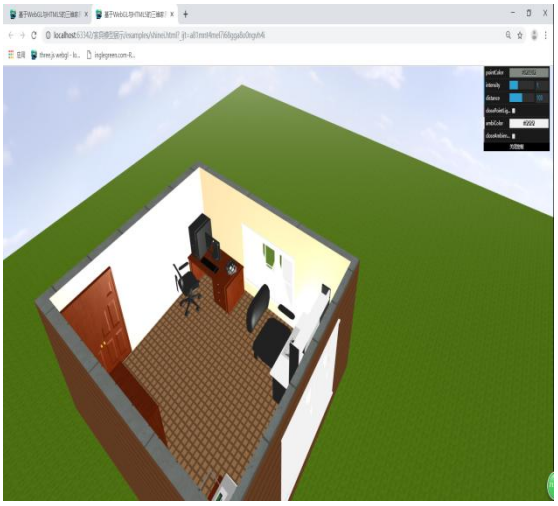

图 0-12 旋转状态 1 效果图

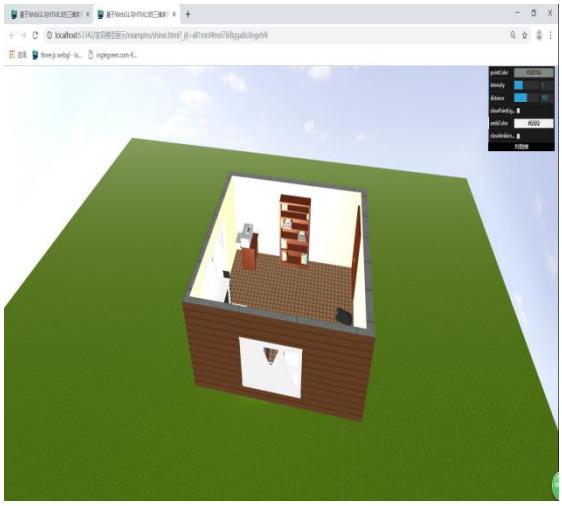

图 0-13 旋转状态 2 效果图

以上测试结果表明，系统能够通过按住鼠标左键来实现场景的旋转。系统的旋转功能是成功的。

5.2.3 对象交互

上面进行了对整个场景的交互控制测试，现在将进行个体对象的拉伸、移动交互测试。这边选用的对象是客厅内的家具进行测试，测试结果如图 5-14 至图 5-22 所示。对家具对象的交互功能进行测试用例设计如下表 5-6 所示。

表 0-6 对象交互测试用例设计

用例设计	用例测试	预期结果	实际结果
进入展示页面，执行家具对象缩放操作	调整拉伸条时，家具对象的压缩情况会随着拉伸条当前值而改变	调整拉伸控制条时，上下拉伸家具对象改变对象的伸缩比例。	同预测结果
进入展示页面，执行家具对象移动操作	调整控制条时，家具对象的位置会随着控制条当前值而改变	调整前后移动控制条时，家具对象实现前后移动；调整左右移动控制条时，家具对象实现左右移动；调整上下移动控制条时，家具对象实现上下移动。	同预测结果

图 0-14 客厅家具加载显示

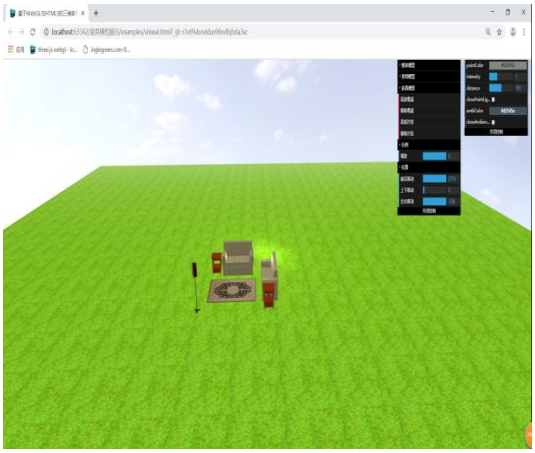

图 0-15 客厅家具拉伸值为 8 时的效果图

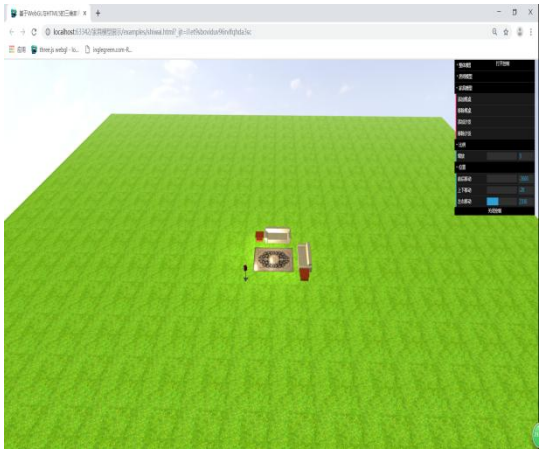

图 0-16 客厅家具拉伸值为 3 时的效果图

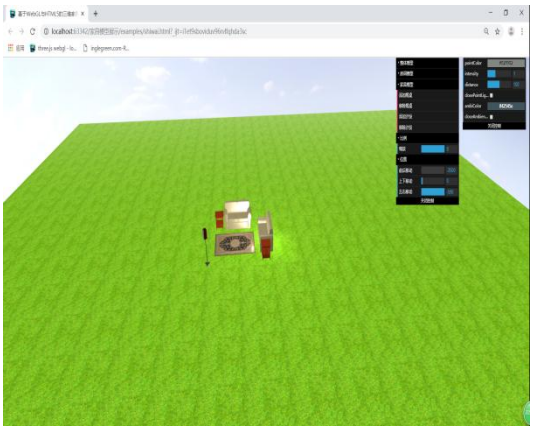

图 0-17 向前移动时效果图

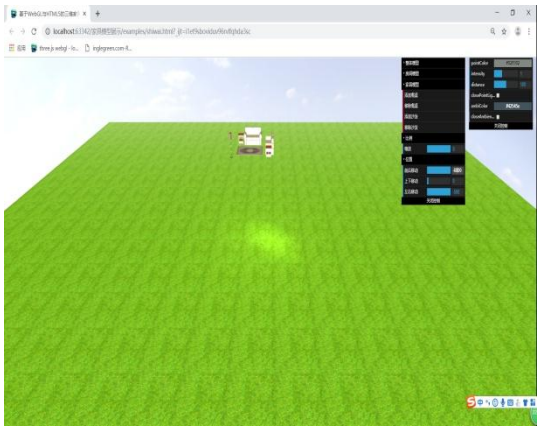

图 0-18 向后移动时效果图

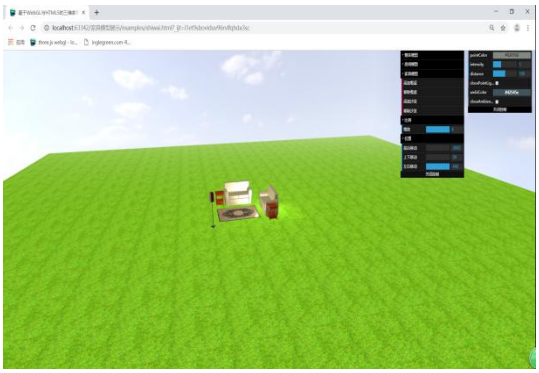

图 0-19 向下移动效果

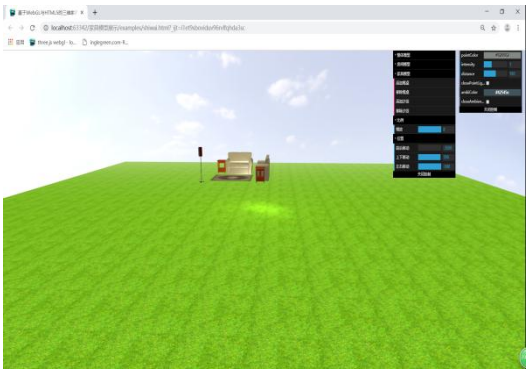

图 0-20 向上移动效果

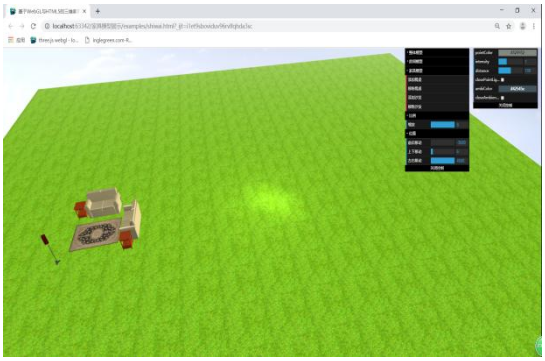

图 0-21 向左移动时的效果图

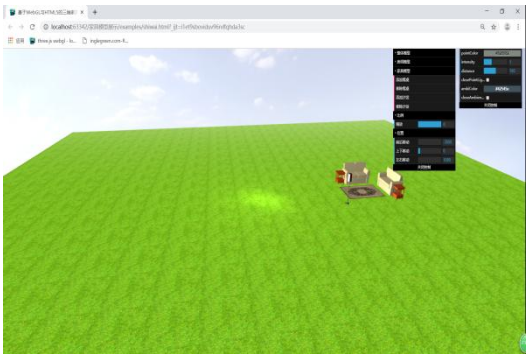

图 0-22 向右移动时的效果图

以上测试结果可知，该系统实现了对家具模型的上下拉伸和上下左右前后移动控制，表明系统能够实现对象的交互功能。

5.2.4 查询实现

在首页面的室内图片上进行按钮遮罩处理，通过点击按钮跳转到数据库查询显示页面，点击不同的房间会查询出不同的结果，测试结果如图 5-23 至图 5-29 所示。对房间信息查询功能进行测试用例设计如下表 5-7 所示。

表 0-7 房间信息查询测试用例设计

用例设计	用例测试	预期结果	实际结果
打开首页面，执行房间信息查询操作	在室内模型图片中点击不同的房间按钮，查看页面中是否查找到该房间的对应信息	当点击卧室时，跳转显示卧室的相关信息，同理，点击其它房间时也会查询显示出对应房间号的相关信息。	同预测结果

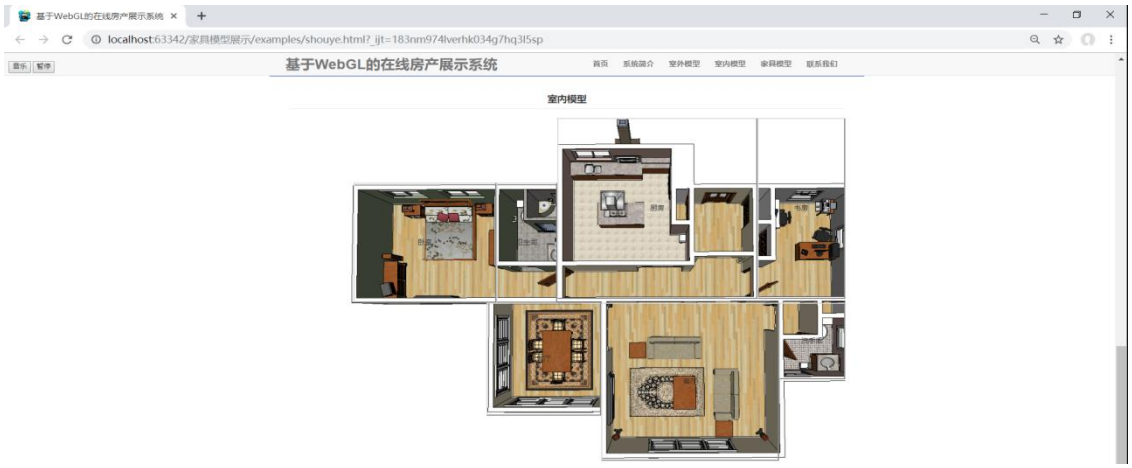

图 0-23 室内遮罩处理效果

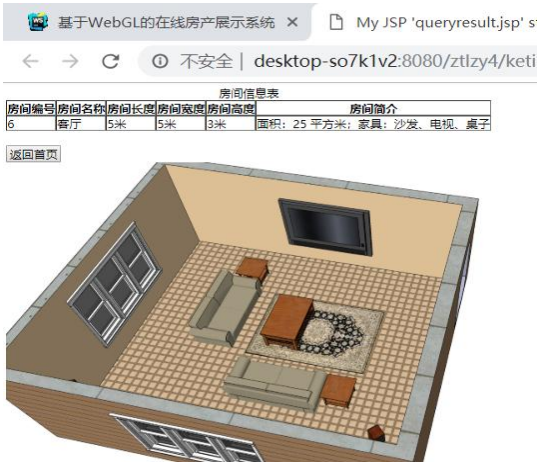

图 0-24 查询客厅信息

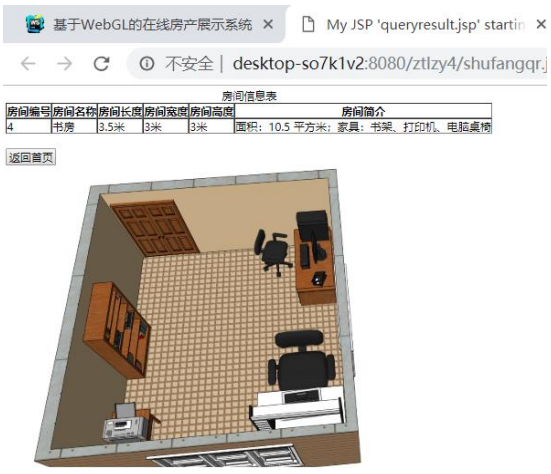

图 0-25 查询书房信息

图 0-26 查询卧室信息

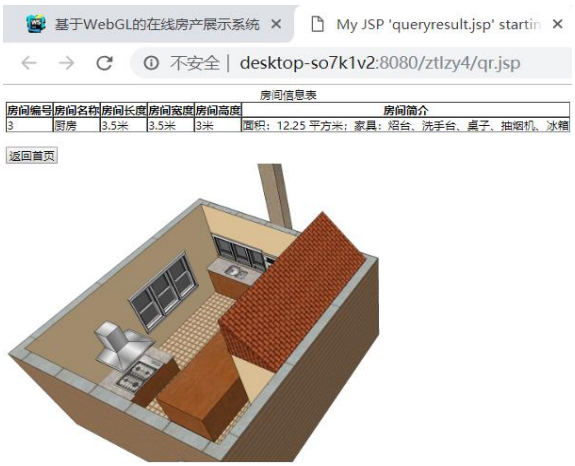

图 0-27 查询厨房信息

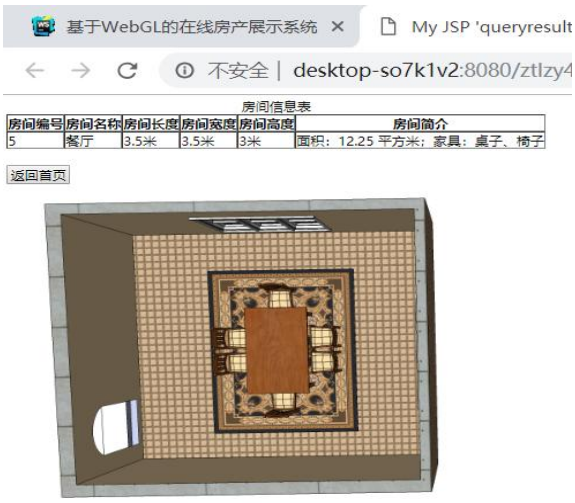

图 0-28 查询餐厅信息

图 0-29 查询卫生间信息

以上测试结果表明，系统能够实现在点击不同房间按钮时执行查出对应房间信息，实现了数据库查询功能。

5.3 测试结果

由以上测试结果可知，在首页对室内模型图片进行遮罩处理，点击按钮跳转到数据库连接查询部分,实现房间属性的查询。该系统能实现对任意 obj 格式的模型进行加载展示，不仅能实现整个场景的缩放、移动和旋转功能的交互控制，而且还能使用 guiconture()函数中进行对象的上下拉伸控制、前后、上下、左右移动控制，各项功能指标能够达到预期的要求。

下面是将本次系统测试中关键模块的测试结果列举如下表 5-8 所示。

表 0-8 测试结果

系统模块	功能模块	测试结果
系统展示	室外模型展示	房屋能正常加载显示，屋顶处有反射现象
	室内模型展示	模型能加载成功，但是由于模型材质有些没有导出成功，导致墙与地面间存在间隙
	房间模型展示	房间模型能正常加载并显示
	家具模型展示	家具模型能正常加载并显示
系统交互	场景交互	能够实现场景的上下左右前后移动、放大缩小控制和旋转控制，能从不同角度观察场景内的模型。
	对象交互	能实现家具的上下拉伸、上下左右前后移动控制
	房间属性查询	能够实现点击不同房间（房间包括：卧室、卫生间、书房、客厅、厨房、餐厅）时，查询到对应房间模型的基本信息

5.4 本章小结

本章主要是对系统进行测试。系统实现了模型加载功能和数据库查询，场景移动、缩放和旋转功能，对象实现拉伸和移动控制功能。

结　论

本文介绍了一款基于 WebGL 的在线房产展示系统的实现技术，采用 HTML5 和 WebGL 相结合的模式，将室内外场景以 3D 形式展示在网页当中。系统的模型主要通过 SketchUp 工具所建。首页面采用 Bootstrap 框架实现轮播显示和遮罩处理，JSP 与 MySQL 数据库连接实现属性查询功能。场景的交互采用 dat.gui 库来创建光照控制和模型展示切换界面组件，在 guiconture 模型显示控制界面上点击相应变量实现展示所选择的模型，隐藏此刻无需显示的模型在 addLightAndGUI 灯光控制界面中可调节光照的强度和颜色，选择打开或关闭光源，将会带来不同的光照效果，给人以环境正在变化一般触感。

系统实现了以下几个功能：

（1）本文采用了 Three.js 框架来定义场景、摄像机和渲染器对象，使用 Three.js 库来加载 OBJ 和 MTL 文本格式模型，实现了三维模型的渲染展示；

（2）系统实现了模型对象的拉伸、移动控制；系统场景的旋转、平移缩放控制；系统的灯光控制和模型切换控制；

（3）系统实现了数据库查询功能。

该系统的实现能够带来更好的用户体验，为购房者和售楼处提供一种全新的房产展示方式。

由于时间有限，该系统还存在以下需要改善之处：

（1）本系统采用的天空盒技术虽然在大部分角度观察都还算比较真实，但也存在一定的缺陷，在观察天空盒任意两个面的接缝处时，会存在有一个角度，没有那么真实，最好使用天空穹技术，可增强视觉体验效果。

（2）本文没有为场景添加人物对象。如若在室内添加一个人，并采用人物行走的漫游方式实现展示效果，带来的感觉会更好些。最后，此类系统展示时若加入了雨滴、雪花、烟雾粒子效果会更加真实。

（3）数据库查询功能不能实现对三维模型的加载显示，带来的效果不够真实，希望后续能够得到改进，从而实现点击三维模型直接查询。

参考文献

[1] 郑华,刘洋.基于 WebGL 的三维模型及其信息化技术研究[J].石家庄铁路职业技术学院学报,2017,16(1):64-70.

[2] 刘喆.基于 Web 的三维立体可视化房地产销售管理信息系统[D].天津:天津大学软件学院软件工程专业,2014.

[3] 吕杰英.三维全景在实训基地漫游系统中的应用研究[J].中国教育信息化,2016(18):78-80.

[4] 吴亚峰,于复兴,索依娜,等.H5 和 WebGL 3D 开发实战详解[M].北京:人民邮电出版社,2017:53-254.

[5] P Li,XQ Yu,J Wang,Progressive compression and transmission of 3D model with WebGL[J],International Conference on Audio,2017,2017(1):170-173.

[6] 松田浩一,李.WebGL 编程指南[M].谢光磊,译.北京:电子工业出版社,2014:10-12.

[7] 李兴田,张丽萍,基于 WebGL 的工程制图网络虚拟模型库的开发[J],图学学报,2016,37(6):836-841.

[8] 陈勇,张灿灿,刘洲,等.WebGL 在网页室内房型展示中的应用[J].物联网技术,2016(11):74-75,79.

[9] 唐路明.基于 WebGL 的大规模 3D 重建场景渲染系统的设计与实现[D].广州:华南理工大学软件学院软件工程专业,2016.

[10] 陈燕红,谢卫国,吕永杰,等.Web 页面三维动态展示技术研究与应用[J].现代电子技术,2018,41(20):24-28.

[11] 王磊,高珏,金野,等.基于 Web3D 无插件的三维模型展示的研究[J].计算机技术与发展,2015,25(4):217-220.

[12] 乔斯·德克森.Three.js 开发指南[M].杨芬,赵汝达,译.北京:机械工业出版社,2017:9-45.

[13] 赵菲.基于 WebGL 的古建筑 BIM 模型轻量化研究与实现[D].西安:西安建筑科技大学计算机技术专业,2018.