浙 江 理 工 大 学

毕业设计(论文)诚信声明

我谨在此保证：本人所写的毕业设计(论文)，凡引用他人的研究成果均已在参考文献或注释中列出。设计(论文)主体均由本人独立完成，没有抄袭、剽窃他人已经发表或未发表的研究成果行为。如出现以上违反知识产权的情况，本人愿意承担相应的责任。

声明人(签名)：

年 月 日

摘 要

随着网络技术的迅速发展、网络信息传输速度的进一步提高以及5G通信时代的到来，在游戏、购物等领域，单纯的二维交互已然不能满足人们的需要。近年来，虚拟现实技术在实际生活中已经实现广泛的应用。为此，本文提出构建基于WebGL的虚拟现实购物系统，在网页端进行商品的三维立体展示，模拟真实光照环境，展示商品的原貌，给用户以独特的购物视觉体验。

本文实现了虚拟现实购物平台的前端搭建，实现在网页中加载商品的三维网格模型并模拟在真实环境光照环境。为了优化商品模型的展示效果，实现了基于图像的后处理抗锯齿FXAA和SMAA算法。对于虚拟现实平台网页前端的搭建主要采用Bootstrap前端框架，实现对网页的优化和美化。采用建模软件搭建商品的同比例尺寸模型，并进行纹理设置，进行导出OBJ、FBX等格式文件。前端网页采用Three.js框架加载商品模型，完成灯光、相机设置，最后将商品模型渲染在网页，实现3D购物效果。对于绘制在屏幕中的商品图像出现的边缘锯齿化现象，采取了两种不同的后处理抗锯齿算法加以消除，实现FXAA算法和SMAA算法并加以比较。

关键词：WebGL；Three.js；三维网格模型；抗锯齿；FXAA；SMAA

**Abstract**

With the rapid development of network technology, the further increase in the speed of network information transmission, and the advent of 5G communications era, simple two-dimensional interactions cannot satisfy people's needs in the fields of games and shopping. In recent years, virtual reality technology has been widely used in our real life. For this reason, this paper proposes to construct a virtual reality shopping system based on WebGL, to perform three-dimensional display of goods on the web page, simulate the real lighting environment, display the original appearance of the goods, and give the user a unique shopping visual experience.

This article implements the front-end construction of the virtual reality shopping platform, implements the loading of a three-dimensional grid model of a commodity in a web page, and simulates the lighting environment in a real environment to display the goods. In order to optimize the display effect of the merchandise model, image-based post-processing antialiasing FXAA and SMAA algorithms are implemented. The Bootstrap front-end framework is mainly adopted for the construction of the front end of the virtual reality platform webpage to realize optimization and beautification of the webpage. Modeling software is used to build the same-size model of the product, and texture settings are performed to export OBJ, FBX and other format files. The front-end webpage uses the Three.js framework to load the product model, complete the lighting and camera settings, and finally render the product model on the webpage to realize the 3D shopping effect. For the phenomenon of edge aliasing appearing on the product image drawn on the screen, two different post-processing anti-aliasing algorithms are adopted to eliminate, and the FXAA algorithm and the SMAA algorithm are implemented and compared.

**Keywords:** WebGL, Three.js, 3D Mesh Model, Anti-aliasing, FXAA, SMAA

**目 录**

摘 要

Abstract

[第1章 绪论 1](#_Toc513890993)

[1.1 研究目的与意义 1](#_Toc513890994)

[1.2 国内外研究现状 1](#_Toc513890995)

[1.2.1 虚拟购物平台的应用现状 2](#_Toc513890996)

[1.2.2 WebGL的技术现状 2](#_Toc513890997)

[1.2.3三维展示存在问题和局限 3](#_Toc513890998)

[第2章 基于WebGL的虚拟现实购物展示系统的搭建 5](#_Toc513890999)

[2.1 虚拟现实系统的协议规则和框架 5](#_Toc513891000)

[2.1.1 WebGL绘图协议 5](#_Toc513891001)

[2.1.2 运行在网页端的3D引擎 ------ Three.js 6](#_Toc513891002)

[2.2 虚拟现实购物平台的搭建 7](#_Toc513891003)

[2.2.1 Bootstrap搭建前端展示页面 7](#_Toc513891004)

[2.2.2 Three.js 工作流程与实现 9](#_Toc513891005)

[第3章 基于图像的后处理抗锯齿算法 11](#_Toc513891006)

[3.1 基于图像的后处理算法FXAA 11](#_Toc513891007)

[3.1.1 FXAA算法的优势与不足 11](#_Toc513891008)

[3.1.2 FXAA算法的实现与效果比较 12](#_Toc513891009)

[3.2基于图像的后处理算法SMAA 13](#_Toc513891010)

[3.2.1 SMAA算法的优势和不足 13](#_Toc513891011)

[3.2.2 SMAA算法的实现与效果比较 14](#_Toc513891012)

[第4章 总结与展望 16](#_Toc513891013)

[4.1总结 16](#_Toc513891014)

[4.2展望 16](#_Toc513891015)

[参考文献 18](#_Toc513891016)

[致 谢 20](#_Toc513891017)

[附录 21](#_Toc513891018)

II

1. 绪论

## 研究目的与意义

随着计算机技术和网络技术的迅速发展，以及5G网络将在2020年的推广商用，在网络娱乐、购物等多领域，仅仅单纯的网络二维空间的交互性[1]已然不能满足人们的需要。而就目前的网络购物而言，小到铅笔、橡皮，大到车辆、家用电器均可在各大网购平台购买。2017年十一月十一日狂欢购物节，开场11秒，交易金额超十亿元人民币，3分01秒交易金额超百亿。网络购物[2]，不仅简化了购物模式，也大大节省了买卖双方的时间成本和空间成本。目前网络购物方便、快捷，但在购物的过程中，消费者只能在网页上浏览、查看商家的文字和图片信息来获取商品的各种参数信息，是无法直接感受和触摸到真实的商品，仅仅是片面上的接触，而且这些文字和图片信息是由商家提供的带有一定的诱惑性，从而导致买家无法获取商品的真实、完整的信息，而造成对商品的误判。由此导致的商品尺寸、颜色以及质感的差异，频繁的退还货等问题是目前网络购物无法避免的痛点问题，给买家和卖家都带来一定程度的损失。

因而，针对买卖双方信息不对称问题，提出希望建立虚拟现实购物[3]平台解决这一问题。平台旨在实现商品的三维立体展示[4]，尽量在浏览器中展现商品在真实环境下的状态，给人以身临其境的时空感；旨在更好的展现商品，提供视觉效果上的全部信息参数。致力于搭建的虚拟购物平台为了使顾客更好的全面了解商品，获取更好的购物视觉体验。从而减少因色差、尺寸问题导致的退换货发生率，减少交易双方的时间、空间成本，给顾客以良好的购物体验。

虚拟现实技术[5]，是利用计算机系统进行多种信息融合、可交互的三维动态实景和行为的系统仿真，使得用户沉浸到构建的虚拟环境[6]中。正是基于这种视觉现实性的体验，提出构建虚拟现实购物平台，提高顾客的购物现实性体验，减少因信息不对称问题而引起的一系列问题。

## 国内外研究现状

虚拟现实（Virtual Reality, VR）技术是使用各种计算机技术以及相应硬件交互技术模拟真实的环境[7]，获取人在真实的环境中的感知与行为从而进行交互的科学技术手段。自2014年，国际互联网公司纷纷加入到虚拟现实领域，争夺良好的发展契机。VR技术已然得到各个国家和技术组织的高度重视。Web3D[8]（又称网络三维），是建立在虚拟现实技术的基础上，把真实世界中的实体物品通过建模软件进行模型建立。而后使用计算机渲染技术进行虚拟的三维立体展示，实现在浏览器中对实体商品的缩放浏览等操作的一种虚拟现实技术。与当前的网络展示技术相比，Web3D使得用户用不同的角度去观察商品实体，可以进行许多的互动操作，使之更具有趣味性。

而网络三维立体展示系统仍存在很多的局限，具体来说，目前Web3D仍然没有统一的标准，每种解决方案都使用不同的格式标准和方案。浏览器插件也是比较棘手的问题，目前Web3D技术仍还需要浏览器插件的支持，正是因为标准的不同，各个厂商的插件也是不尽相同，从几百K到几十M不等。基于Web架构的3D内容渲染对本地计算机和带宽都有很高的要求，也是限制了用户的要求。

### 1.2.1 虚拟购物平台的应用现状

阿里巴巴是马云在1999年成立的网络科技公司，涉及网络购物、云计算服务平台等领域。作为引领潮流的网络销售公司，阿里巴巴在曾在2016年推出虚拟现实购物系统[9]BUY+，11月正式上线。而作为全球第一个VR购物系统，采用眼球动作捕捉技术，从而实现了虚拟环境下的实景购物体验，从进店到付款的全部过程，通过眼球的运动实现所有的购物操作。BUY+的起点是一间普通的房间，布置有七张壁画。这些壁画的地点就是你将要去的购物目的地。而随着眼前场景的不停转换，使用户足不出户，却可以实现在全世界各地的购物体验。

我国中视典公司积极运用先进的图形以及相关技术，研发出三维虚拟现实平台软件[10]，在一定程度上打破了国外垄断的局面，以较高的性价比优势受到大众的喜爱。该公司开发的虚拟现实平台软件已经有较高的市场占有率，不仅能够达到消费者的购物心理预期，还能为消费者带来较多的便利。

### 1.2.2 WebGL的技术现状

WebGL（Web Graphics Library），简单来说，是一种在网页端展示三维立体图像的规范或者标准[11]，WebGL将JavaScript和OpenGL ES 2.0巧妙的联通起来，通过使用JavaScript实现对OpenGL ES 2.0的绑定，WebGL也实现了在硬件层面为HTML5 Canvas[12]提供3D加速渲染的功能，如此便可以利用计算机显卡来在浏览器中更流畅地展现3D模型，数据的可视化当然也不再话下。由此，当我们使用WebGL标准便不再需要浏览器安装相应的插件，简化了用户操作，方便用户使用。对于普通用户来说，消除了不必要的麻烦，从而可以轻松的使用网页三维展示。

Three.js框架是采用JavaScript脚本语言编写的，可以在网页运行且基于WebGL的接口框架。随着Html5超文本语言的普遍应用，使得网页拥有更好的展现力，WebGL为网页提供了可用接口，可以通过调用对应的API进行3D图形的绘制，而Three.js框架则是基础接口之上进行了二次封装。因此它具有了掩盖3D渲染细节、面向对象、功能强大丰富、速度快、扩展性强以及同时支持Html5和Canvas。Three.js框架辅助我们导入不同的模型数据，采用不同材质，灵活使用不同灯光等等。

### 1.2.3三维展示存在问题和局限

目前仍有许多因素限制网络三维展示的发展、应用。目前为止，网络三维展示仍然没有制定统一的标准。目前流行的Web三维展示依然需要3D插件的支持。正是由于标准的不统一，不同的厂商生产的插件也是不尽相同，甚至是差异巨大，插件规模小到几百K到达到几十M皆有。基于Web架构的3D渲染对本地计算机和带宽都有很高的要求，也是限制了网络三维立体展示的推广应用。而WebGL和Html5的使用推广可以解决此类问题。

阿里推出的BUY+虚拟现实购物系统，必须使用穿着特定的三维可视化设备，增加了用户的消费负担，以及增加了用户使用的不便利性。在使用可穿戴设备时，屏蔽了真实世界的部分信息，因此需要较安全和安静的环境。避免与真实环境造成冲突，发生危险。

由于计算机的栅格显示系统以及离散取样显示问题，导致模型出现边缘锯齿化以及走样现象。这会导致模型的失真，与我们的真实物品相去甚远，这也就失去了虚拟现实的意义。

**研究内容与论文结构**

第1章主要介绍本次设计搭建的虚拟现实购物平台的研究目的与意义，并且介绍Web3D发展历程及分类，着重概括介绍网络3D存在问题和局限。

第2章主要介绍WebGL协议和Three.js框架以及基于WebGL虚拟现实购物展示系统的搭建。

第3章主要介绍FXAA和SMAA算法，首先讨论了两种反走样抗锯齿算法的优势和不足， 然后对两者分别实现并进行比较。

第4章对本文进行总结，对虚拟现实技术、WebGL、Three.js以及基于图像的后处理抗锯齿算法的展望。

1. 基于WebGL的虚拟现实购物展示系统的搭建

本章主要介绍如何使用Three.js框架快速搭建3D展示系统。为了实现在网页上实现3D模型的展示以及消除不同标准的插件的依赖，故采用WebGL标准，使用计算机操作系统的标准图形库，将三维模型进行转换，使之展现在二维的计算机屏幕上。通过编写JavaScript脚本语言，借用Three.js框架对WebGL的二次封装接口，实现在网页端进行真实环境的模拟，通过绘制阴影效果更好的表现商品的三维立体感。

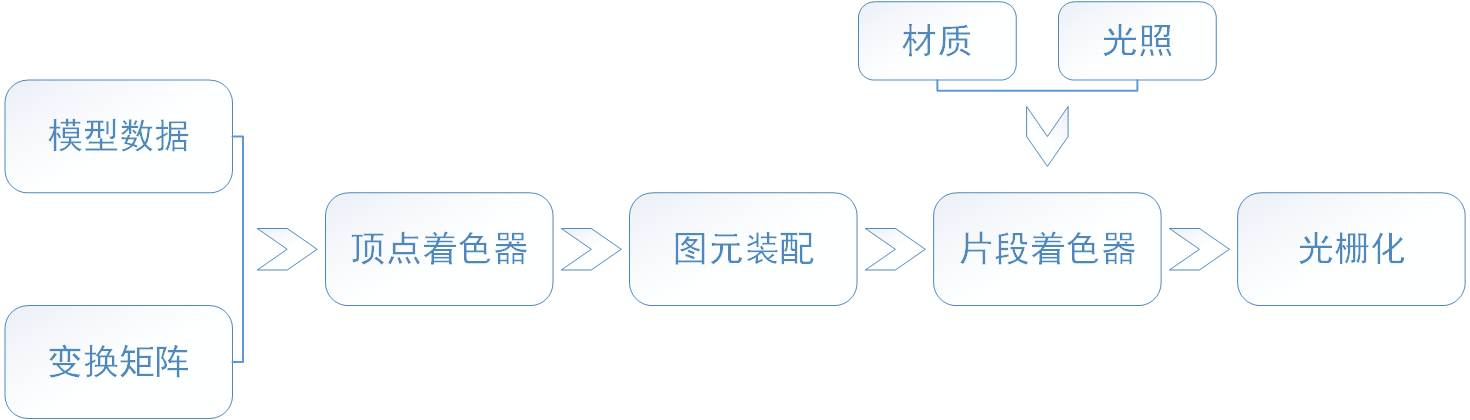
## 2.1 虚拟现实系统的协议规则和框架

### 2.1.1 WebGL绘图协议

WebGL（Web Graphics Library），简单来说，是一种在网页端绘制3D图形的规范或者标准[13]，WebGL将JavaScript和OpenGL ES 2.0巧妙的联通起来，利用JavaScript实现对OpenGL ES 2.0的绑定，WebGL也实现了在硬件层面为HTML5 Canvas提供3D加速渲染的功能，如此便可以使用计算机显卡来在浏览器中更流畅地展现3D模型，数据的可视化当然也不再话下。由此，使用WebGL标准便不再需要浏览器安装相应的插件。实际开发人员就可以使用计算机系统显卡在网页端流畅的进行3D场景和模型的展示，以及数据的可视化操作。WebGL网页3D标准的最明显的优势便是不在需要各个浏览器开发者进行网页渲染插件的开发，可以在本地计算机下构建实物的三维立体模型，然后采用WebGL标准渲染展示在网页端，甚至是直接开发web端3D网页游戏。

而于2010年发布的WebGL主要有两大优势：第一，通过网页编程语言，调用底层图形库，直接在网页端进行渲染，不再需要其他插件的支持。第二，它是通过统一、标准以及夸平台的OpenGL接口来调用底层的图形硬件加速功能而进行3D图形的渲染，实现在不同浏览器中进行实物的三维立体展示。

WebGL绘图标准主要是通过获取顶点坐标、图元装配（绘制三角形）以及光栅化进行商品的三维立体展示。我们使用从三维建模软件中导出的数据文件，如OBJ、FBX等三维格式文件。我们从中获取到模型的顶点信息进行图元装配，即进行三角形构建。使用三角形覆盖进行模型表面构建。在这个过程中，顶点着色器首先对顶点坐标进行转换处理。而后GPU（Graphics Processing Unit，图形处理器）进行图元装配。而后，计算机给3D模型“着色”，而帮这些工作的是运行在GPU上的“片元着色器”，它进行模型颜色、质地、光照效果以及阴影等处理。最后，WebGL进行光栅化处理，即依据每个已经确定的片元颜色，以及根据深度缓存区来判断该片元是否被遮挡，最终将片元信息存储到颜色缓存区实现三维模型的渲染绘制。渲染完毕的模型，将被展示在计算机屏幕中。WebGL的工作流程如图2-1所示。

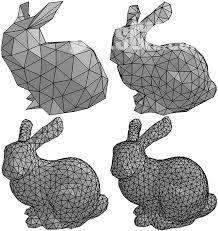


**图2-1 WebGL工作流程**

### 2.1.2 运行在网页端的3D引擎 ------ Three.js

Three.js框架是使用JavaScript脚本语言编写的基于WebGL的接口框架。众所周知，JavaScript脚本语言是在网页端运行的程序语言，故Three.js框架也是可以直接在浏览器中运行。OpenGL（Open Graphics Library，开放图形库）是可以调用底层图形库并且具有跨语言、跨平台的图形程序接口。WebGL绘图协议或标准是基于OpenGL而针对网页开发的图形标准。 Three.js框架则是对WebGL实现了二次封装，简化很多底层的图形调用接口，降低了对编程人员的要求，使用起来更加方便。

Three.js框架并没有规定统一的计量单位，本质上是无单位的，对于模型采用不同的计量单位，需要首先进行归一化处理，最后按照比例进行绘制成最终展现在Web端的3D图像。Three.js框架是遵循右手坐标系的，即以屏幕中心为原点，屏幕水平右侧为x轴正方向，屏幕竖直向上方向为y轴正方向，垂直屏幕向外为z轴正方向。对模型进行旋转操作时，rotation的正值代表逆时针旋转，负值表示顺时针旋转。Three.js框架主要有三大组件，场景（scene）、相机（camera）和渲染器（renderer）。场景即是物体容器，编程开发人员将自己需要的模型、物体等放入场景中。而相机的主要功能是面对场景，娶一个合适的角度进行拍摄。而渲染器（renderer）的功能则是将相机拍摄的图片放在网页中进行展示。在计算机系统中，绘制弧线是通过有限个点连接成线段获得的。当线段的数量多到一定的程度的时候，在我们的视觉系统中，折线段也就变成了圆滑的曲线了。类似于数学中极限，当折线段的数量趋于无穷时，那展现给我们的就是圆滑的曲线。而在计算机系统中的三维立体模型便是利用这一原理，标准做法是三维立体模型采用三角形构成的网格模型[14]来进行绘制，业界称为Mesh模型。



**图2-2 稀疏模型 🡪 致密模型**

如图2-2所示，随着三角形面片数量的增加，斯坦福兔子的形态变得更加形似和光滑。在Three.js中Mesh的函数构造器是：Mesh（geometry, material）。其中，geometry参数是设置模型的形状，而material参数是设置模型的材质。

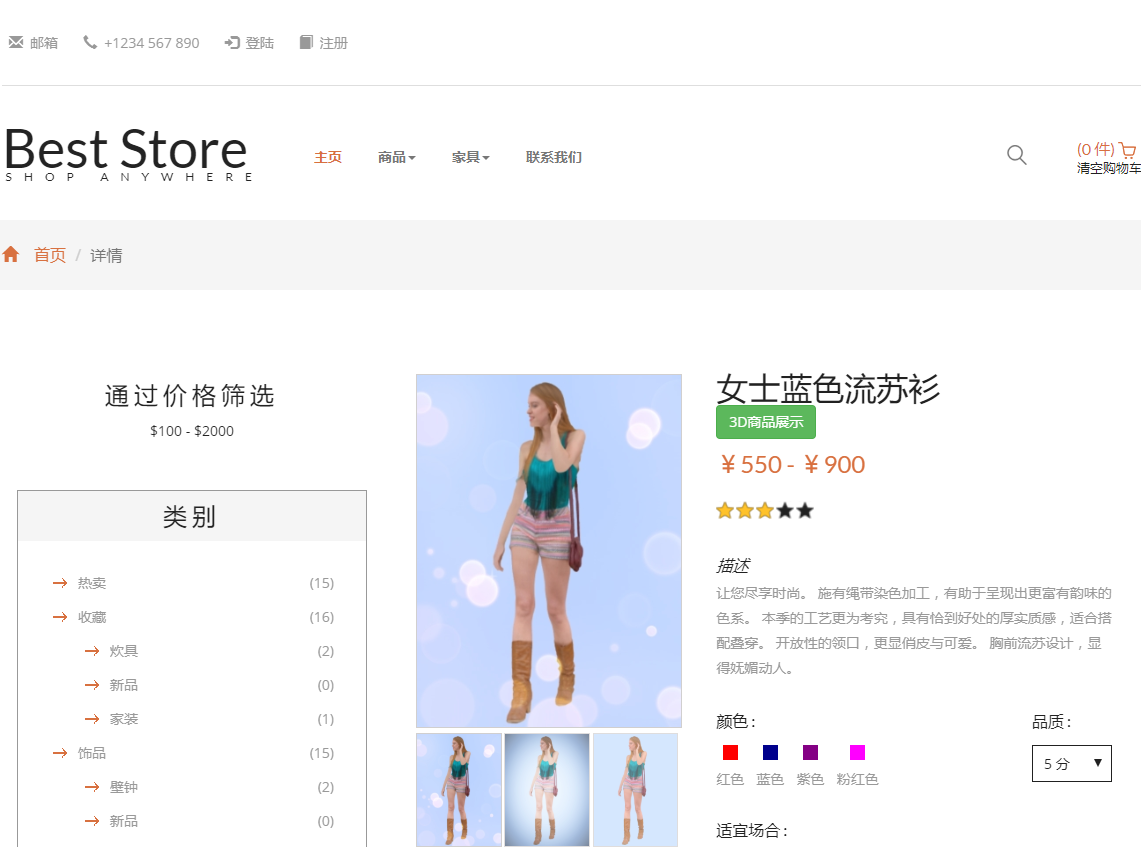
## 2.2 虚拟现实购物平台的搭建

本课题构建的虚拟现实购物平台[[1]](#footnote-1)，主要采用Bootstrap[15]前端框架进行样式格式控制。Bootstrap前端框架是目前最受欢迎的CSS/HTML开源框架之一。 Bootstrap框架基于HTML、CSS和JavaScript，提供了很好的HTML和CSS规范、示例以及jQuery插件，这些都为Bootstrap框架赋予了“生命”。

### 2.2.1 Bootstrap搭建前端展示页面

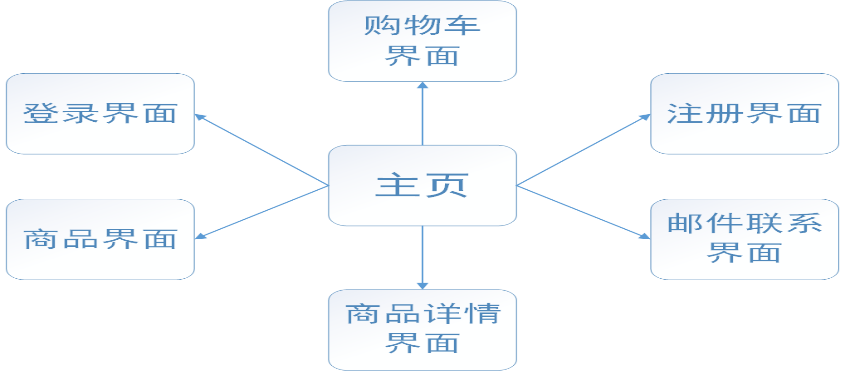
平台前端的页面采用延时加载，实时获取当前用户的操作，对页面进行分步加载，只有当用户浏览到当前网页位置时，进行图片等资源进行加载。

作为商品展示平台，采用图片加模型的展示方式，用户可在主页选中商品查看详情，在商品详情页面进行3D立体模型的展示，如图所示。主页设计为标题栏、内容栏以及版权声明栏。标题栏主要是进行商品类别的选择，进入对应类别的商品展示页面。



在商品展示页面，用户预览商品，获取商品图片、价格等信息。在商品的详情展示页面，用户对商品细节进行全方位的感知，获取最真实的商品感官数据，对商品做出最直接的判断，是否符合需要，避免盲目购买，从而减少退、换商品的发生，节约买卖双方的成本。商品3D模型是按照商品的真实参数和真实环境下的光照条件下在三维建模软件中制作完成的。而平台使用建模软件导出的OBJ、FBX等格式的3D文件进行加载渲染在网页端。平台兼具购物功能，具有登录、注册、浏览、加入购物车以及进行结算等购物平台的基本功能。

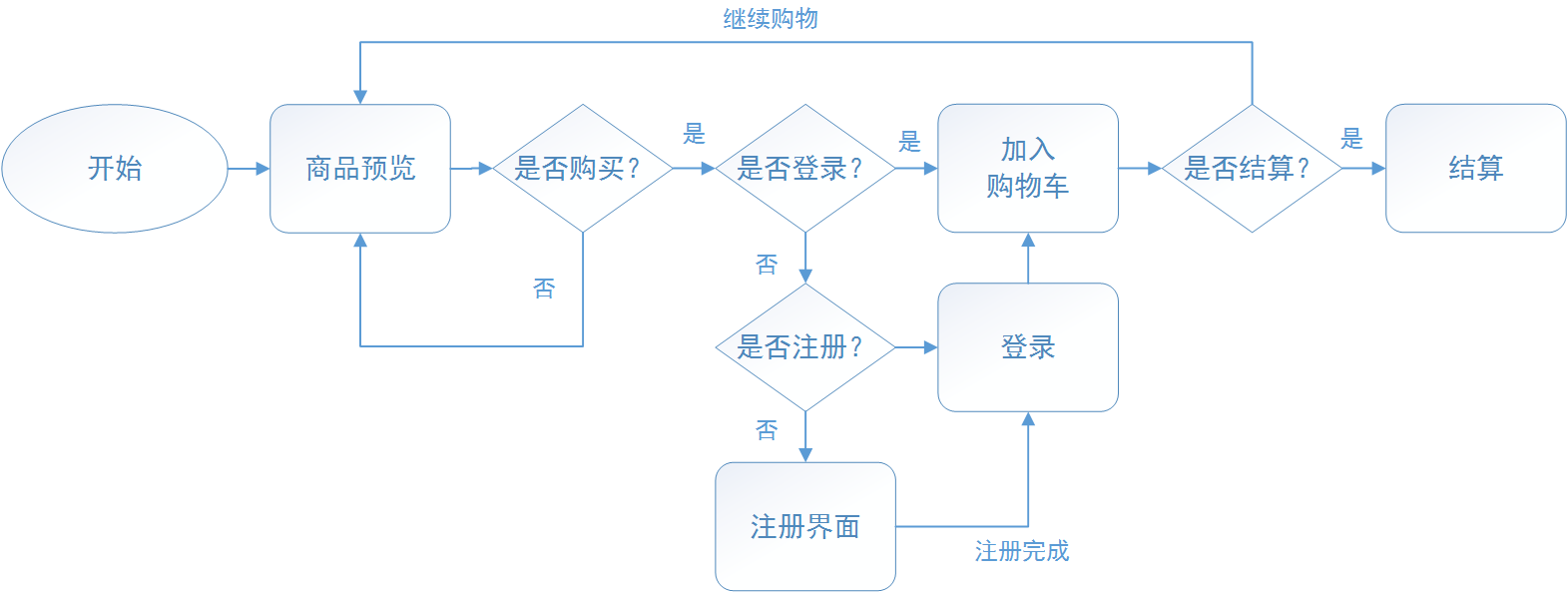
平台架构如图2-3所示：



**图2-3 虚拟购物平台系统架构**

登入虚拟现实购物平台，用户预览各类商品。通过商品展示图片链接进入商品详情页。在商品详情页，用户通过点击3D商品展示按钮，在页面弹出框中进行商品模型的三维立体展示，并可以进行旋转、缩放等交互操作，从多角度观察商品的全貌，获取更多的商品信息。用户购买商品时，必须进行首先进行用户注册，完善个人信息。之后才可进行正确的结算操作。

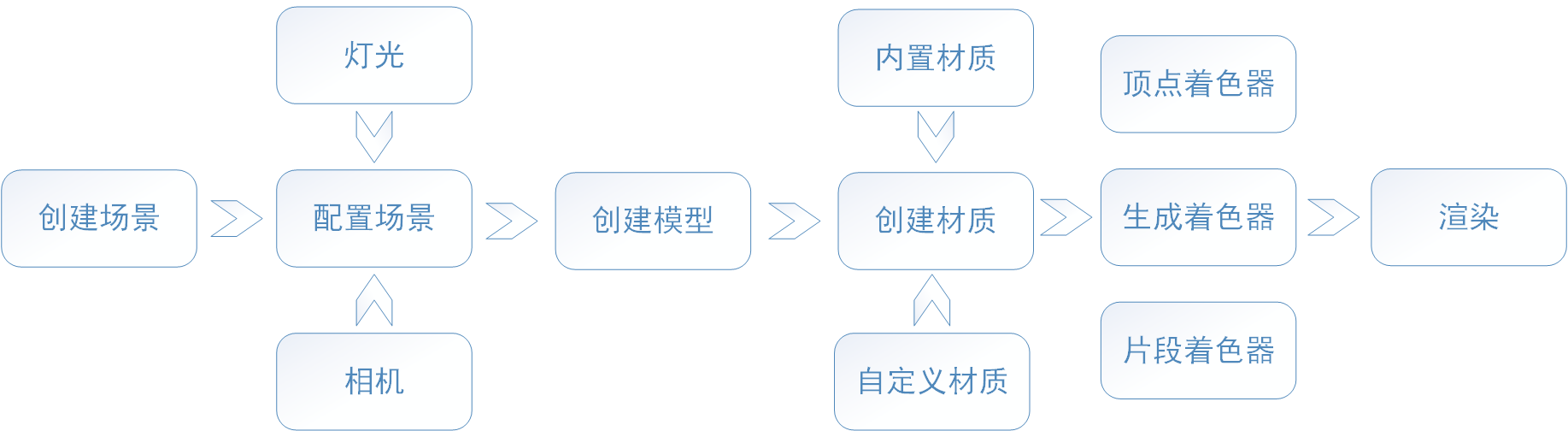
针对用户的平台使用流程如图2-4所示：



**图2-4 虚拟购物平台系统用户流程图**

### 2.2.2 Three.js 工作流程与实现

本虚拟现实购物平台的特色是采用Three.js框架进行商品三维模型的加载、渲染等操作。如图2-5所示，首先创建、配置场景，场景中将展示商品实体模型。利用Three.js框架的模型加载API，加载模型的数据文件，获得模型的顶点数，而后进行模型矩阵、视图矩阵以及投影矩阵的计算，在这一过程中，Three.js框架帮助我们在计算机CPU中完成了局部坐标到世界坐标到视图坐标再到屏幕坐标的计算。在计算机GPU中完成顶点着色器参数计算。最后，渲染在我们计算机的屏幕中。



**图2-5 Three.js工作流程图**

本文实现的虚拟购物平台，采用环境光与聚光灯光源相配合，配置透视相机作为主视角，使用OBJ、FBX等格式的商品模型文件，进行商品实体模型数据的传输和加载。在模型的材质方面，对于不同文件采用不同的纹理加载方式。针对于FBX文件，直接采用数据文件中的纹理参数进行设置。而对于OBJ文件采用对应的图片纹理加载，将图片纹理覆盖在模型表面，从而实现模型纹理加载，增强模型的真实感。

1. 基于图像的后处理抗锯齿算法

走样[16]是指图像在计算机中用离散量来表示连续量而引起的图像失真,出现图像变形的现象。计算机图形走样主要发生在图形光栅化[17]和纹理映射的过程中，称为几何走样和纹理走样。而相对应地，采用相应的计算机技术来消除走样现象的技术，则成为反走样技术[18]。对于反走样技术的基本思想是探测到图像边缘，对图像边缘和边缘周围的像素点颜色进行混合运算，采用新生成的像素点取代原位置的像素点，以达到对实物边缘进行锯齿消除和柔滑处理

当前计算机图形学中主要采用硬件方式、后处理方式以及混合方式三种。而对于硬件方式主要采用提高取样点数目，提高采样频率以达到消除锯齿，平滑边缘的目的。虽然可以很好的处理锯齿问题，但是对硬件资源的消耗比较大，并且对计算机的显示分辨率有一定的要求。后处理方式的基本思想是在完整图像绘制完毕之后，在屏幕显示之前进行边缘检测，对图像的边缘进行消锯齿操作，使图像边缘平滑。

为了获取商品模型更好的三维展示效果，消除计算机系统带来的边缘走样问题，本章主要介绍和实现基于图像后处理抗锯齿算法——快速近似反走样算法（Fast Approximate Anti-Aliasing, FXAA）和增强子像素形态学反走样算法（Enhanced Subpixel Morphological Antialiasing, SMAA）。

## 3.1 基于图像的后处理算法FXAA

### 3.1.1 FXAA算法的优势与不足

2009年Intel工程师成功将形态学引入计算机图形学中，提出了在计算机领域中具有时代意义的形态学反走样技术（Morphological Antialiasing，MLAA）。MLAA是依据相邻元素之间的颜色插值进行边缘提取，对于取到的轮廓与模式进行匹配，而后计算权值进行颜色混合处理。优点是独立于图形处理流水线，在CPU上进行边缘像素的抗锯齿处理[19]，算法易实现。而后，人们陆陆续提出了许多对于FXAA的改进算法。

快速近似反走样技术（Fast Approximate Anti-Aliasing, FXAA）与MLAA类似，运行于渲染后期处理阶段，快速近似抗锯齿[20]（Fast Approximate Anti-Aliasing, FXAA），对图像边缘进行抗锯齿处理，速度更快，而不会形成镜面模糊想想和亚像素模糊现象（表面渲染不足一个像素时的闪烁现象）。由于使用该方法进行边缘检测时，需要对整幅画面进行搜索，并与临近元素进行比较，最后需要加以计算进行边缘的平滑处理，故该方法在计算和存储上对计算机资源消耗比较大。该方法中元素的偏移量计算复杂繁琐导致对于一些模型的边缘反锯齿绘制效果并不是很好。尤其对于文字的处理，会导致文字的边缘模糊，不能得到良好的抗锯齿锐化效果。

### 3.1.2 FXAA算法的实现与效果比较

本虚拟购物平台搭建使用的Three.js框架良好的兼容了自定义算法的实现。FXAA算法是采用像素单元为基本单位，利用像素单元的亮度进行边缘检测。对获取到的边缘提取梯度。获取边缘的梯度方向（函数值或灰度值的最大增长方向），并将子偏移量初始于梯度值小的方向从而计算出该位置的颜色值。得到该像素位置的计算颜色值，并利用当前像素的颜色值和相邻像素的颜色值进行混合，得到最终的颜色值。混合效果图如下：



**图3-1-1输入图像与显著图**

算法流程：

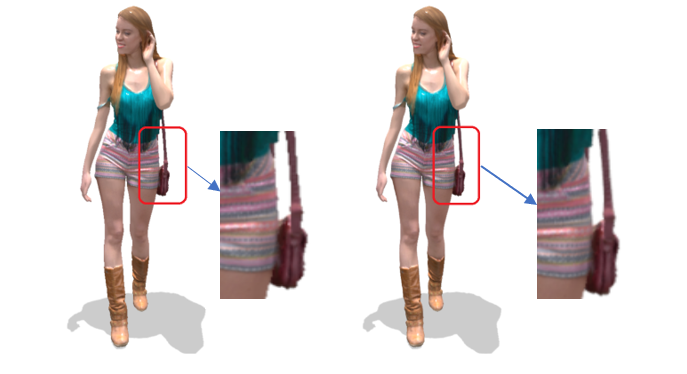
步骤一：模型加载，存储模型的顶点信息以及最后渲染绘制的纹理信息。

步骤二：判断是否为边缘，若是执行步骤三；否则执行步骤四。

步骤三：计算混合朝向和以及子偏移量，继而进行平滑处理，执行步骤四。

步骤四：绘制到计算机显示屏幕，程序结束。

通过上述步骤，借助Three.js框架将三维立体模型展示在屏幕，并通过FXAA算法对模型边缘进行了锯齿消除，使模型边缘平滑。效果如图3-1-2所示：



**图3-1-2 原始图像与FXAA消除锯齿过后的效果图**

## 3.2基于图像的后处理算法SMAA

基于图像的后处理抗锯齿算法存在普遍的缺陷，主要是：由于算法的分析方式并不能准确的判定边界以及获取到准确的边界特征信息；缺少图像必要的场景集合信息，会使得绘制图像的结果出现边缘阶梯状严重以及过度模糊等现象，丧失了处理算法的优越性。故已实现的增强子像素形态学反走样算法SMAA [21][22]在一定程度上解决上述问题，得到较好的绘制图像，对图像边缘的锯齿消除有较好的效果。

### 3.2.1 SMAA算法的优势和不足

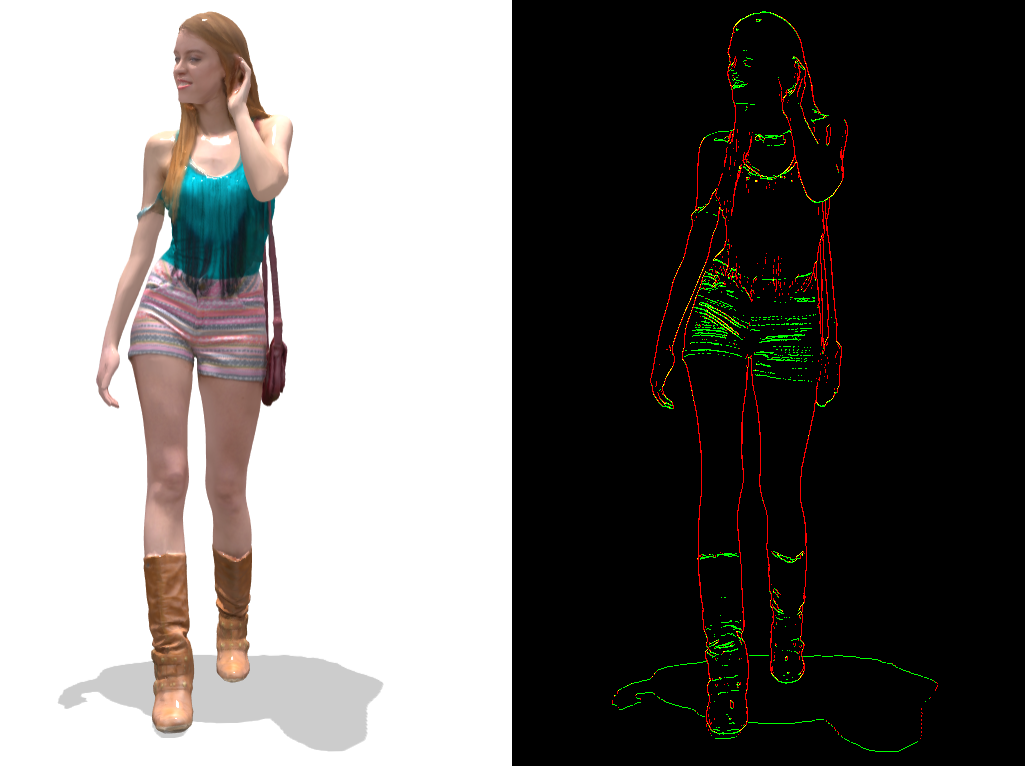
增强子像素形态学反走样SMAA算法也是一种后处理抗锯齿技术，SMAA算法流程是基于优化的MLAA（计算机形态学抗锯齿）算法。而在SMAA算法之前，后期抗锯齿处理技术仍存在一定的缺陷，主要有：没有很好的获取实物的边缘轮廓信息，甚至存在边缘过度模糊的现象；忽视对角线模式检测，影响了抗锯齿消除的效果；基于颜色的边缘检测，没有很好的消除高光和色彩上的锯齿现象。而SMAA算法为解决这些问题应运而生。SMAA算法的边缘检测技术采用精确的距离搜索算法并且增强了局部对比度，更加精准的处理图像。与此同时，SMAA算法强化尖锐几何特征检测和对角线检测，更加精确的展示实物的边缘特征，并获得良好的抗锯齿化效果。

SMAA算法相较于它之前的后处理抗锯齿算法有了明显的提升，但算法本身仍有一定的不足和缺陷，如在绘制图像的边缘提取中，SMAA算法采用了固定阈值，针对与不同的模型特征，不能进行适应性更正，导致图像边缘的获取信息较少，与此同时SMAA算法存储了许多不必要的面积纹理，导致消耗计算机的许多存储资源。

### 3.2.2 SMAA算法的实现与效果比较

本平台使用的基于Web GL的3D框架良好的性能借以实现SMAA算法。通过编写自定义SMAAShader.js进行实现算法，并进行性能测试。

算法概述：SMAA算法[23]可分为边缘提取、权值计算和颜色混合等三个步骤。进行边缘提取时，主要有三种方式：亮度检测、颜色检测和深度检测。SMAA算法对边缘提取时采用固定阈值，导致对某些集合边缘的提取不甚有效。而后，将检测到的边缘与预设的几何形状进行匹配和颜色混合，得到最终图像的绘制效果图，并绘制到计算机屏幕中加以显示。如图3-2-1所示。



**图3-2-1输入图像与通过SMAA获取的边缘轮廓图**

使用SMAA算法进行边缘锯齿的消除效果如图3-2所示：







**图3-2-2输入原始图像与通过SMAA消除锯齿图**

对于本文实现的两种不同抗锯齿算法FXAA和SMAA。对于同一模型的加载进行图像边缘锯齿化消除。如图3-2-3所示的局部放大图所示，SMAA对边缘锯齿的消除效果明显优于FXAA算法。这是因为SMAA算法采用了改进的边缘检测以及尖锐几何特征检测，使之花费较少的时间来获取更好的锯齿消除质量。

(a) (b) (c)

**图3-2-3 原始图像(a)、FXAA消除锯齿图像(b)、SMAA消除锯齿图像(c)**

第4章 总结与展望

## 4.1总结

虚拟现实技术是计算机仿真技术和计算机图形学等多种科学技术的集合，是多领域的交叉科学，拉开了新世界的大门。本文介绍了基于WebGL的虚拟购物平台的搭建，以及围绕更好的进行3D展示做了一些工作。采用了比较流行的前端框架Bootstrap搭建显示页面，使用动态模态框展示商品的3D模型。采用123D建模软件进行商品模型的建立，导出指定格式的模型文件以便加载使用。在网页端，使用JavaScript语言编写的第三方开发库-----Three.js。在编程之前，比较详细地了解了WebGL的前身今生，找到比较良好的框架进行开发。在众多的抗锯齿算法中，选择了两种比较有特点的算法进行实现与比较，较好地消除了图像边缘的锯齿化现象，使渲染效果更加真实。

本次主要实现了基于虚拟现实技术的3D商品展示系统，建立模型，完成3D模型在网页端的展示，并实现基于图像的后处理抗锯齿算法FXAA和SMAA算法以及效果比较。本文的总结工作如下：

1. 介绍了基于WebGL的虚拟现实购物平台的研究目的与意义，同时也对目前国内外的虚拟现实技术研究状况进行了总结，简介了虚拟现实技术的应用、技术现状以及存在的问题。
2. 本次设计使用Bootstrap前端框架构建虚拟购物平台展示前端。Bootstrap前端框架简洁、灵活，使得构建网页时非常的便捷。代码开源，接口简介统一，Web开发快捷。
3. 介绍了本次设计与实现过程中涉及到的理论知识，介绍构建虚拟现实购物平台的使用的Web GL协议，以及为便捷开发而使用的3D框架---Three.js。为了提高图像以及阴影的成像效果，实现FXAA和SMAA算法，消除边缘锯齿化，平滑边缘。

## 4.2展望

当今网络技术和计算机技术的迅猛发展，在一定程度上极大的改变了人们的生产、生活方式。虚拟现实技术的不断发展，带给人们新的体验，试图通过计算机等技术给人身临其境的真实体验感。WebGL协议的推广，实现了在浏览器内进行实物的三维立体展示，解除了对相应插件的依赖。开源的Three.js框架实现了的对WebGL的良好的封装，减轻了程序员的压力，又可以面向更多的普通人进行3D展示编程，虽然相关的文档还是比较稀少，相信伴随着该框架的不断发展，相关工作会逐步完善，更有利于Three.js的推广使用。由于我们计算机使用光栅化的图像显示技术，不可避免地产生走样问题，本体涉及的FXAA算法以及SMAA算法都是对渲染图像边缘锯齿化进行一定程度上的消除。伴随着计算机图形学和阴影技术的不断发展，FXAA算法以及SMAA算法会在继承的基础上加以改进和优化，越来越多的基于图像的后处理抗锯齿算法会相继问世，更好解决图像走样问题。

参考文献

1. 沈朝魁. 虚拟现实技术在网络购物中的应用研究[D]. 南京航空航天大学, 2012.
2. 谢文达. 建构基于VRML/X3D的网上虚拟商店[D]. 广东工业大学, 2006.
3. 徐静静. 基于虚拟现实技术的数字化商城原型系统的研究[D]. 杭州电子科技大学, 2013.
4. 齐东健, 解凯, 何翊卿, 等. 基于WebGL的3D购物平台的设计与实现. 北京印刷学院学报, 2016, 24(4):42-45.
5. 卢薇朵. 网络购物平台上的虚拟现实技术研究[J]. 信息与电脑, 2017(2):48-49
6. 赵沁平. 虚拟现实综述[J]. 中国科学:信息科学, 2009, 39(1):2-46.
7. 陈月华, 郑春辉, 韩纪庆. 虚拟现实技术在网络购物平台上的应用[J]. 哈尔滨工业大学学报, 2009(1):97-100.
8. 王磊, 高珏, 金野, 等. 基于Web3 D无插件的三维模型展示的研究. 计算机技术与发展, 2015(4):217-220.
9. 沈朝魁. 虚拟现实技术在网络购物中的应用研究[D]. 南京航空航天大学, 2012.
10. Matsuda K, Lea R. WebGL编程指南(第一版), 电子工业出版社. 2014.
11. 何龙, 杜鹏. 基于WebGL的三维可视化[J]. 科技资讯, 2015, 13(30):23-23.
12. Danchilla B. Beginning WebGL for HTML5. New York: A press，2012．
13. 徐新山, 张志华. 基于WebGL的地层体信息三维可视化[J]. 测绘与空间地理信息, 2016(11):65-68.
14. 杜文俊, 冯结青, 杨宝光，沈笠. 三角型重建的几何阴影算法[J]. 计算机辅助设计与图形学学报，2015.
15. http://www.bootcss.com/
16. 杜文俊, 冯结青.面向延迟着色的统一反走样算法[J].计算机辅助设计与图形学学报,2015.
17. Kaneda K, Shinya M, Nishita T. Advanced Rendering Techniques[J]. Journal of the Institute of Image Electronics Engineers of Japan, 2004, 33:620-631.
18. 杜文俊. 基于几何的实时绘制反走样[D]. 浙江大学, 2015.
19. Reeves W T, Salesin D H, Cook R L. Rendering antialiased shadows with depth maps[J]. Acm Siggraph Computer Graphics, 1987, 21(4):283-291.
20. 刘燕, 张建伟. 一种快速高效的反走样算法[J]. 计算机应用研究, 2012, 29(3):1138-1140.
21. Jimenez J, Echevarria J I, Sousa T, et al. SMAA: Enhanced Subpixel Morphological Antialiasing[J]. Computer Graphics Forum, 2012, 31(2pt1):355-364.
22. 刘镜荣, 杜慧敏, 杜琴琴. 子像素形态学反走样算法的改进[J]. 计算机应用, 2017, 37(10):2871-2874.
23. 刘镜荣, 杜慧敏, 杜琴琴. 形态学反走样算法的改进[J]. 计算机应用, 2016.

致 谢

经过大半年的学习研究，本科生的毕业设计已经基本完成。在这里我要感谢我的指导老师宋滢。从毕业设计的选题开始、开题答辩、中期答辩到最后的毕业论文的撰写以及毕业答辩，宋滢老师都细心地提供很多切实有效的指导，给予了我许多意见和帮助，尤其是在毕业设计后期遇到问题无法解决的时候，宋滢老师多次和我展开讨论，指出问题所在，给出解决问题的思路和可行的方案。正是宋滢老师的指导与帮助，才能让我一次次克服困难，最终完成毕业设计。宋滢老师面对问题的态度以及工作的认真以及严谨性都是我今后学习与工作的榜样。

与此同时，我要感谢我的母校，我的老师，我的同学。老师们的谆谆教导，给我今后的学习打下坚实的基础，让我有展望明天的自信。同学们的无私帮助，让我度过美好而快乐的大学四年。母校的无私培养让我能成长为一个不惧风雨自信自强的新世纪大学生。

最后感谢我的父母，感谢你们对我的无限包容与无私付出。你们是我人生的第一位导师，是你们的辛劳培养，才能成就今天的我。感谢我的父母，希望你们永远健康、快乐！

孙晓明

2018年4月30日

附录

1. FXAA算法测试程序

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>WebGL 测试objloder</title>

<script src="build/three.js"></script>

<script src="js/libs/stats.min.js"></script>

<script src="js/libs/dat.gui.min.js"></script>

<script src="js/loaders/OBJLoader.js"></script>

<script src="js/controls/OrbitControls.js"></script>

<script type="text/javascript" src="js/shaders/SMAAShader.js"></script>

<style type="text/css">

div#canvas-frame {

border: none;

cursor: pointer;

width: 100%;

height: 600px;

}

</style>

<script>

var scene;

function initScene() {

scene = new THREE.Scene();

}

var renderer;

function initThree() {

width = document.getElementById('canvas-frame').clientWidth;

height = document.getElementById('canvas-frame').clientHeight;

renderer = new THREE.WebGLRenderer({

antialias : true , // 抗锯齿

precision:"highp" //着色精度选择

});

//renderer.setSize(width, height);

renderer.setSize(window.innerWidth,window.innerHeight);

renderer.setClearColor(0xffffff,1);//设置背景颜色

renderer.shadowMap.enabled = true; // 设置阴影

renderer.shadowMapSoft = true; // 软阴影

renderer.shadowMapType=THREE.PCFSoftShadowMap; //边缘柔和 AA

document.getElementById('canvas-frame').appendChild(renderer.domElement);

}

var camera;

function initCamera() {

camera = new THREE.PerspectiveCamera(45, width / height, 0.1, 1000);

camera.position.x = 0;

camera.position.y = 0;

camera.position.z = 3;

camera.up.x = 0;

camera.up.y = 1;

camera.up.z = 0;

camera.lookAt(scene.position);

}

//初始化灯光

var light;

var amlight;

function initLight() {

amlight = new THREE.AmbientLight(0xFFFFFF,0.5);

amlight.position.set(100, 100, 100);

scene.add(amlight);

light = new THREE.SpotLight(0xFFFFFF,1);

light.position.set(2,10,10);

light.castShadow=true;

light.shadowMapHeight = 2048;

light.shadowMapWidth = 2048;

scene.add(light);

}

var model;

function initObject() {

var manager = new THREE.LoadingManager();

var texture = new THREE.Texture();

var loader = new THREE.ImageLoader( manager );

loader.load( 'models/texture/female.jpg', function ( image ) {

texture.image = image;

texture.needsUpdate = true;

} );

var material = new THREE.MeshBasicMaterial({map:texture});

var loader = new THREE.OBJLoader( manager );

loader.load( 'models/obj/Female.obj', function ( object ) {

var mesh = new THREE.Mesh( object,material );

object.traverse( function ( child ) {

if ( child instanceof THREE.Mesh ) {

child.material.map = texture;

child.position.set(0,-0.8,0);

child.castShadow = true;

child.receiveShadow = true;

}

} );

scene.add(object);//将导入的模型添加到场景

});

}

var plane;

function plane(){

var planeGeometry=new THREE.PlaneGeometry(20,20);//平面

var planeMaterial=new THREE.MeshLambertMaterial({color:0xffffff});

plane=new THREE.Mesh(planeGeometry,planeMaterial);

plane.rotation.x=-0.5\*Math.PI;//将平面沿着x轴进行旋转

plane.position.x=0;

plane.position.y=-0.8;

plane.position.z=0;

plane.receiveShadow=true;//平面进行接受阴影

scene.add(plane);

}

// 帧循环

function animate()

{

//mesh.rotation.y +=0.01;

//boxMesh.rotateY(0.01);

//controls.update();

renderer.render(scene, camera);

requestAnimationFrame(animate);

}

//用户交互插件 鼠标左键按住旋转，右键按住平移，滚轮缩放

var controls;

function initControls() {

controls = new THREE.OrbitControls( camera, renderer.domElement );

// 如果使用animate方法时，将此函数删除

//controls.addEventListener( 'change', render );

// 使动画循环使用时阻尼或自转 意思是否有惯性

controls.enableDamping = true;

//动态阻尼系数 就是鼠标拖拽旋转灵敏度

//controls.dampingFactor = 0.25;

//是否可以缩放

controls.enableZoom = true;

//是否自动旋转

controls.autoRotate = true;

//设置相机距离原点的最近距离

controls.minDistance = 2;

//设置相机距离原点的最远距离

controls.maxDistance = 100;

//是否开启右键拖拽

controls.enablePan = true;

}

function threeStart(){

initScene(); //初始化场景

initThree(); //初始化渲染器

plane(); //初始化平面

initCamera(); //初始化相机

initLight(); // 初始化灯光

initObject();

animate();

initControls();

}

</script>

</head>

<body onload="threeStart();">

<div style="margin-left:150px;">

<p>引用模型</p>

</div>

<div id="canvas-frame"></div>

</body>

</html>

1. 运行环境：win10 操作系统、CPU主频1.7 GHz、Chrome浏览器

   编辑环境：Tomcat服务器、IDEA编辑器 [↑](#footnote-ref-1)