基于Three.js技术的太阳系3-D模型网站的设计与实现

作者姓名：李姿 专业班级：软件工程3班 指导老师：廖晓鹃

摘 要

随着技术的进步，普通的平面设计已经不能满足当代人对生活更进一步的追求。三维全景技术是一种用全新的视角，给用户带来身临其境般的直观感受，全方位展示产品的新技术。用户可通过鼠标、键盘等操作来控制观察全景的方向，可以在全方位地查看场景以及场景中的内容。

基于此背景以及个人兴趣，本次设计最终选定使用Three.js来进行太阳系三维全景的开发。实现通过全景漫游太阳系，查看太阳系的行星、恒星以及卫星的运行情况，通过与用户的简单交互，在页面中展示各星体的详细文字信息与图解。实验显示，太阳系场景搭建完整，用户可使用键鼠来漫游场景并查看具体信息，该课题基本完善了预设计的功能。

关键词：全景漫游；太阳系；Three.js

**Design and Implementation of Solar System 3-D Model Website Based on Three. js**

**Abstract:** With the advancement of technology, ordinary graphic design can no longer meet the contemporary people’s pursuit of life. 3D panoramic technology is a new technology that uses a new perspective to bring users an immersive and intuitive experience and display products in all directions. The user can control the direction of the panoramic view through the operation of the mouse, keyboard, etc., and can view the scene and the content in the scene in all aspects.

Based on this background and personal interests, this design finally chose to use Three. js to develop the 3D panorama of the solar system. To view the operation of planets, stars, and satellites of the solar system through the panoramic roaming, the detail text information and illustration of each star are displayed on the page with simple interaction. Experiments shows that the solar system scene is complete, users can use the mouse and keyboard to roam the scene and view specific information, the subject basically perfected the pre-designed function.

**Key words:** Panoramic roaming; Solar system; Three. js

目 录

[第1章 前言 3](#_Toc6065)

[1.1 课题背景与意义 3](#_Toc19623)

[1.2 课题目标 3](#_Toc5252)

[第2章 相关技术介绍 5](#_Toc6237)

[2.1 漫游空间的编辑 5](#_Toc2931)

[2.1.1使用Three.js渲染场景 5](#_Toc17277)

[2.1.2 为场景创建动画 8](#_Toc32749)

[2.2 太阳系具体信息的展示 9](#_Toc5028)

[第3章 系统分析 10](#_Toc22876)

[3.1 可行性分析 10](#_Toc12748)

[3.1.1 技术可行性 10](#_Toc26403)

[3.1.2 操作可行性 10](#_Toc6360)

[3.1.3 法律可行性 10](#_Toc9599)

[3.2 需求分析 10](#_Toc5483)

[3.2.1 功能需求 10](#_Toc21108)

[3.2.2 性能需求 11](#_Toc18848)

[3.2.3 运行需求 11](#_Toc6964)

[第4章 系统设计与模型分析 12](#_Toc31583)

[4.1 系统的总体设计 12](#_Toc11676)

[4.1.1 设计约束和原则 12](#_Toc27224)

[4.1.2 功能结构 12](#_Toc5821)

[4.1.3 系统架构 13](#_Toc4852)

[4.2 物理模型分析 14](#_Toc24203)

[4.2.1 物理模型概况 14](#_Toc11711)

[4.2.2 太阳模型 15](#_Toc4649)

[4.2.3 行星模型 15](#_Toc30477)

[第5章 具体实现 16](#_Toc14697)

[5.1 系统概述 16](#_Toc26863)

[5.2系统模型实现 16](#_Toc22905)

[5.2.1 相关描述 16](#_Toc8632)

[5.2.2 关键代码 17](#_Toc30168)

[5.3 背景模型实现 19](#_Toc17299)

[5.3.1 相关描述 19](#_Toc18946)

[5.3.2 关键代码 19](#_Toc15877)

[5.4 星球体系模型实现 21](#_Toc2425)

[5.4.1 相关描述 21](#_Toc14420)

[5.4.2 关键代码 22](#_Toc8466)

[5.5 系统数据交互实现 23](#_Toc19481)

[5.5.1 相关描述 23](#_Toc2993)

[5.5.2 关键代码 23](#_Toc31124)

[5.6 星体图解实现 24](#_Toc7638)

[5.6.1 相关描述 24](#_Toc19326)

[5.7 系统整合 24](#_Toc1709)

[5.7.1 相关描述 24](#_Toc3679)

[5.7.2 主要场景效果图 24](#_Toc31355)

[第6章 系统测试 26](#_Toc24509)

[6.1 功能测试 26](#_Toc22947)

[6.2 适配兼容测试 26](#_Toc15262)

[结论 27](#_Toc148)

[致谢 28](#_Toc9007)

[参考文献 29](#_Toc4187)

# 

# 前 言

在计算机技术高速发展的时代，现代浏览器的功能变得越来越强大，从以前的简单显示平面网页，到现在的成为了可以展示复杂应用和图形的平台。

## 课题背景与意义

大多数浏览器已经可以支持WebGL，不仅可以在Web创建二维应用以及平面图形，还可以通过GPU的运算功能创建具有优秀视觉效果及运行良好的三维应用。然而，使用WebGL编程是比较复杂的，为简化使用WebGL编程的过程，Three.js诞生了。

Three.js提供了很简单的关于WebGL特性的JavaScript API，使用者不需要学习复杂的着色语言和WebGL的底层细节，只需对JavaScript和HTML有大致了解，即可在浏览器中创建三维应用。由于近年来，用户对平面应用和图形开始感到乏味，渴望由新技术带来的不同以往的图形界面，如VR、AR、全景等技术开始大放异彩。通过在浏览器中创建三维动画，可以给用户带来一定视觉冲击效果，吸引用户的注意，在商业中形成独特的视觉营销，对企业的发展有一定的帮助，同时对教育等行业也有或多或少的促进效果。

## 国内外研究现状

Three.js由Mr.Doob创建，并同时在GitHub中提供了多个辅助库。Three.js库历次更新较大，且国内研究内容较少，参考文献等内容部分过时，仅能提供极为有限的参考价值，因此网络上提供的Three.js官方文档中英版以及由创建者等人提供的Examples在本次课题中起到了重要的作用。现阶段在网络上存在各种关于Three.js的例库以及样式库可供参考，实际使用的网站偏少。

## 课题目标

本次课题旨在通过学习Three.js的部分核心内容，实现一个模拟太阳系的三维场景的开发。

用户进入网站就可以观察到太阳系的整体场景，了解太阳系的基本结构以及运动规律。通过在场景中设置轨道控制器，用户可通过鼠标左键拖动、右键拖动和滚轮滚动对太阳系场景进行旋转、平移和缩放操作，从而实现对太阳系的全方位观察。

本次课题建立了太阳、八大行星和月球的十个模型，并且在屏幕左侧放置了缩略图图标，用户可以通过点击各个星球的模型或缩略图图标深入了解各个星球的具体信息。同时，在太阳系模型的周围设置了点云效果以及在六个面上设置了宇宙背景，丰富了用户的视觉体验。

本次课题将按照以上目标进行设计与实现，通过全面的三维展示效果、简洁的交互系统和真实可靠的信息吸引用户的眼球，激发用户的学习兴趣，使本系统成为一个真实有用的太阳系漫游应用。

# 相关技术介绍

## 2.1 漫游空间的编辑

Three.js可以理解为Three+js，Three可以表示为3D的意思，js自然就是JavaScript，那么Three.js顾名思义，就是用JavaScript来实现3D效果。在浏览器中使用JavaScript开发3D场景，具有可拓展性强、兼容性强和开发难度低等优点。

### 2.1.1使用Three.js渲染场景

使用Three.js创建三维场景，首先需要设置三个不可缺少的基本组件：场景（Scene），相机（Camera）以及渲染器（Renderer）。

场景（Scene）：

场景相当于一个容器，也可以称之为场景图，主要用于保存、跟踪所要渲染的物体和使用的光源，只有设置了THREE.Scene对象，Three.js才能渲染物体。场景不仅仅是一个对象数组，还包含了场景图树形结构中的所有节点，每个添加到Three.js场景的对象，甚至包括THREE.Scene本身，都是继承自一个名为THREE.Object3D的对象。THREE.Object3D对象可以有自己的子对象，可以使用它的子对象来创建一个Three.js能解释和渲染的对象树。

相机（Camera）：

相机对象决定了我们能够在场景看到什么，在Three.js中有两种相机：透视投影摄像机（PerspectiveCamera）和正交投影摄像机（OrthographicCamera）。正交投影摄像机中，对象相对于摄像机的距离对渲染是没有影响的，常用于二维游戏。因此，为了模仿人眼效果，本次课题使用的是透视投影摄像机。透视投影摄像机在Three.js中的参数如图2-1所示，fov表示相机的垂直视角角度，aspect表示相机的宽高比，near和far分别表示相机的近裁剪面和远裁剪面的距离，这两个参数需为正值，并且far的值应该大于near的值。

渲染器（Renderer）：

THREE.WebGLRenderer对象会基于相机的角度，并根据一定的规则来计算场景对象在浏览器中会渲染成什么样子，然后使用电脑显卡来渲染场景（Jos Dirksen，2015）。

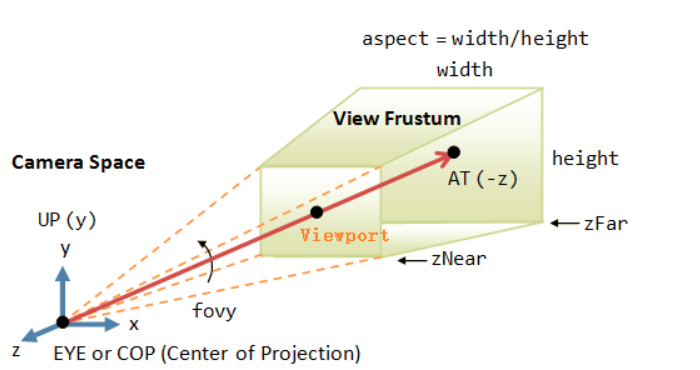


图2-1 透视投影摄像机的参数图解

以上则是构建场景所需要的主要三大组件，其关系如图2-2所示。

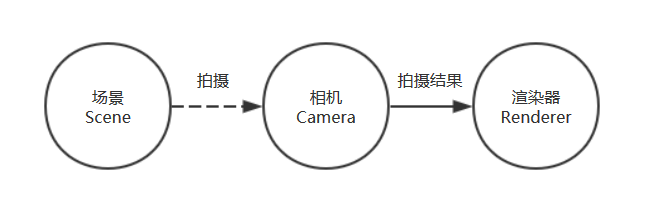


图2-2 三大组件的关系

在设置完以上三大组件后，开始设计场景的背景，即宇宙背景。全景图分为球型全景图、柱型全景图以及立方体全景图。其中，球型全景图的构建模式和人眼最为相似，但构建面多，性能消耗高，构建难度较大；柱型全景图垂直视野小，顶部与底部较难完成；立方体全景图最为常见，构建方便，贴图简单，因此本次课题采用立方体全景图来构建场景的背景。

构建立方体全景图，只需收集六个面的纹理素材，计算每个面的3D位置和旋转角度，并设置一定的透明度后即可放入场景。

设置好背景后，即可开始创建太阳系模型，本次一共设置了十个星体：太阳系中心的恒星——太阳，八大行星——水星、金星、地球、火星、木星、土星、天王星、海王星，以及地月系统中的卫星——月球。创建该模型只需创建十个球体（SphereGeometry），并为每个球体设置与之对应的贴图（本次课题用到的星体贴图均为网络收集），即可创建出与实际效果相差无几的星球。

为便于设置星体位置，在场景中添加THREE.AxesHelper对象以做参考，设置完成后需移除。Three.js使用的是右手坐标系，如图2-3所示。

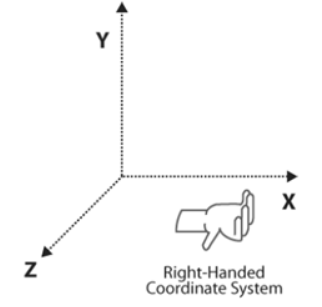


图2-3 Three.js中的坐标系

通过在网络上查询资料，获取星体的自转周期、公转周期以及距日距离，为各星体设置参数。由于距日距离差距过大，并且远大于星体大小，难以在模型中写实地表达，因此各星体的距日距离以及半径根据常见的太阳系图解的比例来设置。

星体的自转只需根据自转周期设置旋转属性中的y参数，操作简单，不加赘述。将星体的公转抽象成简易的圆周运动后，其y轴坐标必定为0，所以星体坐标为（x, 0, z），只需计算在平面上x值与y值的变化。此处通过设置每个星体的角速度θ，根据半径和三角函数来计算各星体的平面位置（如图2-4所示），最后在动画函数中运行自转函数与公转函数。

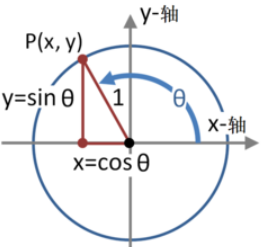


图2-4 单位圆表示的三角函数

为便于观察，在创建所有星体模型后，为所有行星添加运动轨迹线，设置以每个行星的半径-0.5为内径、每个行星的半径+0.5为外径的环（RingGeometry），由于Ring默认垂直于x轴，所以需要让每个Ring绕x轴旋转到xz平面上，然后将环添加到场景中。

由于土星具有土星环，所以还需在土星的位置处设置一个RingGeometry，为土星环贴图，然后旋转一定角度后添加到场景中，并在公转函数中设置和土星相同相同的运动轨迹。

宇宙中还有无数的星球，仅仅以上的模型仍然显得太单调，因此需要创建成千上万个SphereGeometry来作为背景中的星星，但显然电脑无法承受这么巨大的运算量，所以选择使用BufferGeometry。BufferGeometry是Geometry的高效替代，因为它使用buffer来保存所有的数据，包括顶点位置、面索引、法向量、颜色、UVs以及自定义属性。这大大节约了向GPU传递全部这些数据的成本，但同时也使BufferGeometry比Geometry更难处理，不是以对象的方式来访问，如使用Vector3来访问位置数据，以Color对象来访问颜色数据，需要从相应的attribute缓存中访问原始数据，这使得BufferGeometry很适合用来存储静态对象，创造完模型实例后不太需要去操作它。

此时场景中的模型已经基本构建完成，但是场景仍然无法显示，因为还没有往场景中添加光源，本次课题使用了点光源（PointLight）、环境光（AmbientLight）和平行光（DirectionalLight）。PointLight是一种单点发光、照射所有方向的光源，可以产生阴影，使用PointLight可模拟太阳光；AmbientLight不需要指定位置并且会应用到全局，所以只需指定颜色就可以添加到场景中，应用到整个场景。DirectionalLight可以看做是距离很远的光，发出的所有光线都是相互平行的，并且被平行光照亮的整个区域接收到的光强是一样的。点光源和平行光的照射方向如图2-5所示。

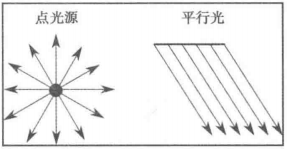


图2-5 点光源与平行光

### 2.1.2 为场景创建动画

为使场景中的模型运动起来，这里用到了Three.js中的实时渲染，即不停地对画面进行绘制渲染，这就需要使用HTML5提供的requestAnimationFrame()方法。在animate()中调用动画函数，在场景初始化完成只需调用一次render()，在animate()中调用一次render()，然后在animate()内部使用requestAnimationFrame()方法指向animate本身来触发下一次的场景渲染。这样，浏览器就可以在一定的时间间隔（通常是60次每秒）执行animate()方法，来达到实时渲染的目的。

## 2.2 太阳系具体信息的展示

为减轻GPU压力，选择使用HTML5 来展示星体的详细数据。使用Raycaster对象来创建鼠标左键点击事件，通过点击星体或场景的星体缩略图，在页面中显示星体的详细信息，包括文字简介、基本信息和图解。

# 系统分析

## 3.1 可行性分析

### 3.1.1 技术可行性

本次课题主要采用Three.js开发，并使用了少量简单的HTML5和CSS。由于Three.js的资料大多是英文，历次版本更新较大，中文资料跟进速度缓慢，但其本质是JavaScript，作者对JavaScript有一定学习基础，所以在基于官方文档使用Three.js上并无较大难度。

### 3.1.2 操作可行性

本次课题设计的目标是系统运行后，能够使用户对图形界面操作自如，因此不能有过于复杂的操作方式。随着电脑的普及，大部分人能够十分娴熟地使用鼠标操作，本次课题设计的系统只有简单的鼠标操作，系统界面简洁友好，所以能保证几乎所有用户都能够轻松使用。

### 3.1.3 法律可行性

本次系统开发所使用的软件均为正版软件或者开源免费软件，并在相关法律规定下进行开发，系统具有原创性，不会侵犯他人、集体或国家利益，在法律上是可行的。

## 3.2 需求分析

本系统是一个天文类的虚拟仿真系统，借助Three.js技术，虚拟展示了一个相对真实的太阳系：太阳在星系中心照亮整个星系，八大行星按照各自的轨道和周期围绕太阳永不停歇地旋转，月球作为地球的卫星也在自己的轨道上运动。

本系统主要用作教学演示使用。

### 3.2.1 功能需求

太阳系3-D模型系统设计具有以下功能：

1）B/S架构，只需通过浏览器访问，不需要额外安装任何插件；

2）浏览太阳系中太阳和各个行星、卫星的3D模型及其运行状况；

3）各个行星的运动轨迹可视化；

4）可以通过键盘以及鼠标进行自由漫游，对系统进行旋转、平移和缩放的操作，能够自由接近或远离各星体；

5）点击各星体时可显示该星体的基本信息；

6）点击各星体是可显示该星体的具体图解。

### 3.2.2 性能需求

1）模型准确度高，基本符合各项真实太阳系数据及运行情况；

2）有较强的沉浸感，3D模型清晰度高；

3）运行基本流畅，通过状态栏监控；

4）系统对用户的输入实时响应。

### 3.2.3 运行需求

1）可通过PC端各浏览器浏览；

2）服务器端：任一支持HTTP的Web服务器；

3）硬件需求：CPU：1GHz以上，内存：512MB以上；

4）Web浏览器：Firefox、Google Chrome、Microsoft Edge。

# 系统设计与模型分析

## 4.1 系统的总体设计

系统设计考虑的主要约束是客户端浏览器不需安装任何插件，运行流畅。

### 4.1.1 设计约束和原则

根据需求分析，设计原则主要有以下几点：

1. 基于浏览器，尽量降低复杂度以保证系统性能；
2. 尽量使用已有的框架、库，不重复造轮子；
3. 面向对象设计；
4. 便于扩展和调整维护。

### 4.1.2 功能结构

本系统基于B/S架构，包括服务器端和浏览器端两个部分，但由于系统主要功能集中在浏览器端，服务器端只需要满足HTTP的静态响应即可，无需动态技术，因此这里主要对浏览器端进行分析。根据需求分析，浏览器端可分为四部分：模型展示、场景漫游、运行控制和信息显示，如图4-1所示。

1. 模型展示

模型展示是系统的主要输出部分，通过利用Three.js库，可以将虚拟的太阳系各星体的3D模型和它们各自的运行情况准确地呈现出来。在显卡和Three.js的支持下，可以表现出和3D游戏相当的场景效果。

1. 场景漫游

本系统的场景漫游借助了3D技术中的相机控制原理，通过控制鼠标和键盘上的预定按键移动摄像机的位置来实现。这项功能可以带给用户较强的沉浸感，使用户仿佛置身宇宙中，自由地在星体间穿梭遨游。

场景漫游的控制主要包括位置控制和视角朝向控制两个部分。

位置控制可以根据用户的输入改变相机的坐标，可以通过键盘的“WASD”键位、空格键以及上下左右方向键来控制，其控制关系为：W/↑：前进；A/←：左移；S/↓：后退；D/→：右移；Space：回到初始位置。鼠标控制则是通过拖拽的形式来控制场景的展示。

视角方向控制则是控制用户看向的方向，即为相机的视向（lookAt）。考虑到用户可能拖拽过度，视向移到太阳系以外的未知方向，给用户带来不稳定、难以控制的感觉，因此在这里将视向固定在太阳系的中心天体——太阳所在的位置处。

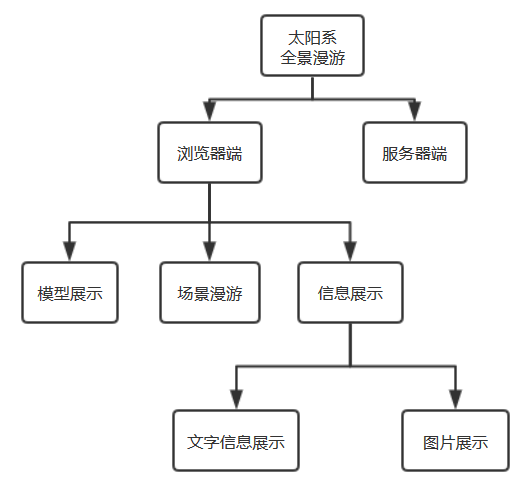


图4-1 浏览器端功能结构图

1. 信息展示

本系统中的信息展示是由用户触发点击事件产生的。由于太阳系的运动是永不停息的，部分用户可能有难以选中目标的问题，所以在界面中设置了一系列缩略图，使用户可以方便地点击查看目标星体信息。

信息展示主要分为两大部分：文字信息展示和图片展示。文字信息展示是详细地介绍星体已知的相关信息，并对星体的各个参数（如半径、公转周期等）进行罗列展示，使用户对星体有初步了解；图片展示则是使用图解的方式详细介绍星体的形态、形状以及内部构造等难以用文字阐述清楚的信息。

### 4.1.3 系统架构

系统的总体架构由四部分组成：交互层、模型层、框架层和数据层。

1）交互层

交互层处于系统的最前端，负责直接与用户交互。主要包括四个部分：

1. HTML提供界面的结构；
2. CSS对显示外观进行约定；
3. DOM Event处理用户输入；
4. JavaScript实现对模型层的调用。

2）模型层

模型层主要对系统整体和各个天体进行建模，这一层由各种模型类组成，大致分为五个部分：系统整体抽象、太阳模型、行星与卫星模型、轨道模型以及背景星空模型。

1. 系统整体抽象是指利用Three.js库中封装好的方法，建立整个虚拟太阳系能正常展示和运作的平台基础；
2. 太阳模型包括太阳本体和太阳光，是场景的中心和唯一的点光源；
3. 行星与卫星模型，包括了太阳系中八大行星的模型，以及地月系统中的月球模型、土星的土星环模型；
4. 轨道模型，即八大行星的公转运行轨道；
5. 背景星空模型是对太阳系的背景星空的建模，包括使用算法随机生成的大量星辰和宇宙全景图。

3）框架层

框架层是对3D引擎的封装，主要有第三方类库Three.js及其他相关类库组成。Three.js是对WebGL API的封装，提供了场景、光照、纹理、材质和模型的生成、加载、转换以及矩阵运算等强大功能，其他相关类库则是对控制器以及加载器等等优化装置进行一个补充。通过框架层对3D引擎的封装，为系统提供了优秀的3D展示服务。

4）数据层

数据层主要是通过angular.js将虚拟太阳系的数据以及系统进行了双向绑定，通过对星球的点击，使得信息展示框动态绑定星球数据，呈现相关信息，并且缩短了相关文档的加载时间，提高了系统性能。

## 4.2 物理模型分析

### 4.2.1 物理模型概况

太阳系由一个恒星、八个行星以及众多矮行星、小行星、彗星组成，本系统模拟的太阳系仅包括太阳、八大行星、背景星辰、背景宇宙和地月系统的卫星——月球。在现实太阳系中，太阳位于中心，其他行星与太阳的距离由近到远分别是水星、金星、地球、火星、木星、土星、天王星、海王星排列。系统除了需要按照此顺序排列行星外，还需要模拟出它们与太阳距离的比例，以及它们的体积比例和运行速度。由于实际数据相差过大，因此仅系统中的数据均做了一定调整，实际数据会在详细介绍中提到。

### 4.2.2 太阳模型

太阳位于太阳系中心，是太阳系中唯一发光的星体，在现实中太阳相对于宇宙微波背景辐射（CMB）的速度是370km/s，但在太阳系系统中，可看作是相对静止的。针对太阳的建模有以下要点：

1. 太阳为一规则球体；
2. 太阳位置固定于太阳系三维空间中心，即坐标（0,0,0处）；
3. 太阳是空间中唯一的光源，也就是说只能有一个点光源；
4. 太阳表面应设置为火红带斑纹的纹理。

### 4.2.3 行星模型

八大行星围绕太阳公转，它们的各项数据如图4-2所示。

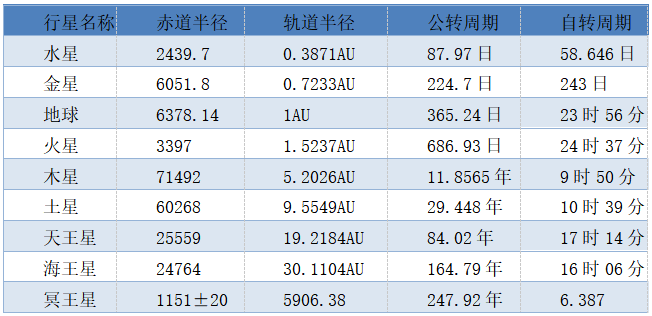


图4-2 八大行星数据

本系统仅设置了具有代表意义的卫星，即地月系统的卫星——月球，以地球为单位1，其相关数据如图4-3。



图4-3 地月系统数据

# 具体实现

## 5.1 系统概述

完成系统分析与设置之后，则进入编码实现阶段。本系统面向教师和学生开发，所以力求操作简单实用，界面干净简约，侧重点是寓教于乐。本系统遵循设计一个易用、美观、大方的系统的原则设计。

## 5.2系统模型实现

### 5.2.1 相关描述

系统模型是对系统运行基础的建模，也是系统的入口和主要工作模块，其主要功能为初始化和运行两部分，分别由init、animate、onWindowResize和onKeyDown函数实现。

在init函数中，实现对画布canvas、渲染器renderer、相机camera、轨道相机控制器orbitControls、环境光ambientLight、点光源pointLight等系统筑基的初始化，并添加到场景容器scene中。

在animate函数中，实现对控制器的实时更新，便于实时监控场景变化并作出相应反应，同时在其中调用动画函数，实现循环渲染，完成动画效果。

在onWindowResize函数中，监听了window对象的窗口改变的事件。当窗口改变大小，场景能根据目前窗口大小实时改变相机视野的大小，并调用渲染器进行重新渲染。

在onKeyDown函数中，监听了文档的键盘按下事件，当按下键盘时，获得事件返回的按键的键码值keycode，根据键码值的不同，实现不同的效果。这个函数主要是对场景中用键盘控制场景上下左右前后移动的实现。在函数中调用cameraMove函数，通过获得参数来改变相机的位置属性来达到控制场景的目的。这两个函数的结合，实现了键盘控制场景变化的功能。

系统模型实现的主要流程图如图5-1所示。

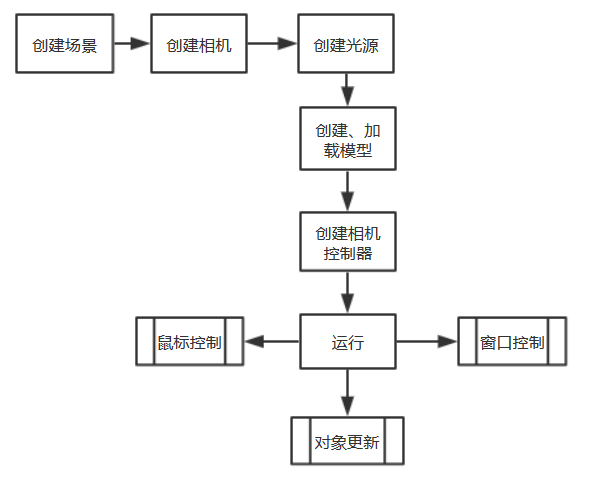


图5-1 系统模型实现流程图

### 5.2.2 关键代码

关键代码如下：

/--scene--/

scene = new THREE.Scene();

/--canvas--/

canvas.width = window.innerWidth;

canvas.height = window.innerHeight;

/--renderer--/

renderer = new THREE.WebGLRenderer({canvas: canvas, alpha: true, antialias: true});

renderer.shadowMap.enabled = true;

renderer.shadowMapSoft = true;

renderer.setPixelRatio(window.devicePixelRatio);

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

document.getElementById(‘WebGL-output’).appendChild(renderer.domElement);

/--camera--/

camera = new THREE.PerspectiveCamera(60, window.innerWidth / window.innerHeight, 1, 10000);

camera.position.set(500, 500, 0);

/--orbitControls--/

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

controls.enableKeys = true;

controls.rotateSpeed = 0.5;

controls.panSpeed = 0.5;

controls.ketPanSpeed = 3.5;

controls.enableDamping = true;

controls.dampingFactor = 0.25;

controls.screenSpacePanning = false;

controls.minDistance = 100;

sontrols.maxDistance = 1200;

controls.maxPolorAngle = Math.PI / 2;

/--ambiemtLight--/

var ambientLight = new THREE.AmbientLight(0xaaaaaa);

scene.add(ambientLight);

/--pointLight--/

var sunlight = new THREE.PointLight(0xddddaa, 5, 4000);

sunlight.castShadow = true;

scene.add(sunlight);

window.addEventListener(‘resize’, onWindowResize, false);

canvas.addEventListener(‘mousedown’, onDocumentMouseDown, false);

document.addEventListener(‘keydown’, onKeyDown, false);

## 5.3 背景模型实现

### 5.3.1 相关描述

为了使模拟太阳系系统更加逼真，使用了全景图来构建宇宙背景。在这里使用了立方体全景图来创建。因ShaderMaterial使用难度打，因此在这里创建了六个平面，为平面依次贴图后添加到场景中，并限制相机只能在立方体的场景内运行，即可完成背景空间的构造。

与此同时，为减轻GPU负担，采用BufferGeometry来创造有无数星辰的背景星空。由之前提到的BufferGeometry的使用方法可知，首先要用两个数组来分别确定一个星辰的位置和颜色信息。并用随机函数确定位置范围，然后用for循环为星辰随机赋予颜色，最后进行空间计算。这样创建出来的星空成本较低，但视觉效果很不错，在移动镜头的过程中还有层次感。

### 5.3.2 关键代码

关键代码如下：

var backPlane = creatBack(‘back.png’);

backPlane.rotation.y = Math.PI / 2;

backPlane.position.set(-2500, 0, 0);

scene.add(backPlane);

/--其他面创建方法同上，仅改变rotation和position的值--/

/--background stars--/

particles = 20000;

buffer = new THREE.BufferGeometry();

position = new Float32Array(particles \* 3);

colors = new Float32Array(particles \* 3);

color = new THREE.Color();

sizes = [];

/--星星出现的最近距离--/

gap = 1500;

/--随机星星出现的位置--/

for (var i = 0; i < position.length; i +=3){

/-- -2gap<x<2gap --/

var x = (Math.random() \* gap \* 2) \* (Math.random() < .5 ? -1 : 1);

var y = (Math.random() \* gap \* 2) \* (Math.random() < .5 ? -1 : 1);

var z = (Math.random() \* gap \* 2) \* (Math.random() < .5 ? -1 : 1);

/--选出绝对值最大的值--/

var max = Math.abs(x) > Math.abs(y) ? Math.abs(x) > Math.abs(z) ? ‘x’ : ‘z’ : Math.abs(y) > Math.abs(z) ? ‘y’ : ‘z’;

var pos = {x, y, z};

if (Math.abs(pos[max]) < gap) {

pos[max] = pos[max] < 0 ? -gap : gap;

}

/--定义位置--/

x = pos[‘x’];

y = pos[‘y’];

z = pos[‘z’];

position[i] = x;

position[i+1] = y;

position[i+2] = z;

/--70%的星星有随机颜色--/

var hasColor = Math.random() > 0.3;

var vx, vy, vz;

if (hasColor) {

vx = (Math.random() + 1) / 2;

vy = (Math.random() + 1) / 2;

vz = (Math.random() + 1) / 2;

} else {

vx = 1;

vy = 1;

vz = 1;

}

color.setRGB(vx, vy, vz);

colors[i] = color.r;

colors[i+1] = color.g;

colors[i+2] = color.b;

sizes.push(20);

}

buffer.addAttribute(‘position’, new THREE.BufferAttribute(positions, 3));

buffer.addAttribute(‘color’, new THREE.BufferAttribute(colors, 3));

buffer.addAttribute(‘size’, new THREE.Float32BufferAttribute(sizes, 1).setDynamic(true));

buffer.computeBoundingSphere();

/--计算后添加到场景中--/

var material = new THREE.PointsMaterial({size: 6, vertexColors: THREE.VertexColors});

particlesSystem = new THREE.Points(buffer, material);

scene.add(particlesSystem);

## 5.4 星球体系模型实现

### 5.4.1 相关描述

星球模型是整个太阳系系统的主要内容，其实现由初始化和更新构成，包括init、animate、createPlanet、revolution、rotation等函数。

在init函数中，实现了对各个行星、太阳以及行星运动轨道的创建和初始化。

在animate函数中，实现对星体的实时监控，使用户的每次操作都能让场景返回相应的动作。

在createPlanet函数中，通过init函数中创建并初始化的星体传入的参数，将各个星体的详细数据以及详细设计完善，使星体更加逼真，将函数单独提出来构造也达到了优化代码性能的目的。

在revolution函数中，实现了行星和卫星的公转，通过设置星体的每次转动角度，利用三角函数计算星体的位置并重新设置，进行实时渲染，这样就可以使星体动起来。

在rotation函数中，实现了各个星体的自转，只需通过改变星体的rotation的值，并进行实时渲染就可以使星体自转。

星球体系模型实现的主要流程如图5-2。

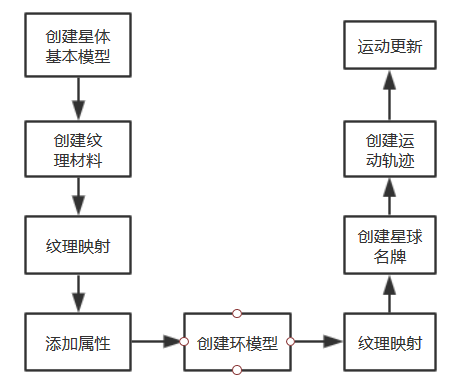


图5-2 星球体系模型实现流程图

### 5.4.2 关键代码

function createPlanet(name, distance, radius, revolutionSpeed,rotationSpeed,imageFile) {

var loader = new THREE.TextureLoader();

var texture = loader.load(“textures/” + imageFile);

var material = new THREE.MeshPhoneMaterial({side: THREE.DoubleSide});

material.map = texture;

var geometry = new THREE.SphereGeometry(radius, 32, 32);

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

mesh.position.x = distance;

mesh.receiveShadow = true;

mesh.castShadow = true;

mesh.name = name;

scene.add(mesh);

return mesh;

}

function revolution() {

deg += 1/20 \* Math.PI / 180;

Mercury.position.set(Mercury.distance \* Math.cos(Mercury.revolutionSpeed \* deg), 0, Mercury.distance \* Math.sin(Mercury.revolutionSpeed \* deg));

/--其余行星、卫星和土星环构造方法相同--/

function rotation(){

Earth.rotation.y += Earth.rotationSpeed;

/--其余星体构造方法相同--/

}

}

## 5.5 系统数据交互实现

### 5.5.1 相关描述

在太阳系系统中不仅是观看太阳系场景，更有意义的是获取星体相关信息。获取信息是通过对悬浮列表进行拾取处理而实现的。

### 5.5.2 关键代码

funtion clickSun() {

var visibility = document.getElementById(“sunInfo”).style.visibility;

if (visibility == “hidden”) {

document.getElementById(“sunInfo”).style.visibility = “visible”;

document.getElementById(“thumbnail”).style.visibility = “hidden”;

} else {

document.getElementById(“sunInfo”).style.visibility = “hidden”;

document.getElemetById(“thumbnail”).style.visibility = “visible”;

}

}

/--其余星体方法同上--/

## 5.6 星体图解实现

### 5.6.1 相关描述

为使信息展示不那么枯燥，在信息展示部分结合图片对星体进行讲解，更加激发用户兴趣，同时也可以具体解释文字展示中较为抽象的部分。该部分通过简单的HTML+CSS插入到信息面板中。

## 5.7 系统整合

### 5.7.1 相关描述

在实现了模型的构建和系统的友好交互后，将所有功能整合，模型和数据与页面文档对接，完成整个系统的运作。

### 5.7.2 主要场景效果图



图5-3 主界面

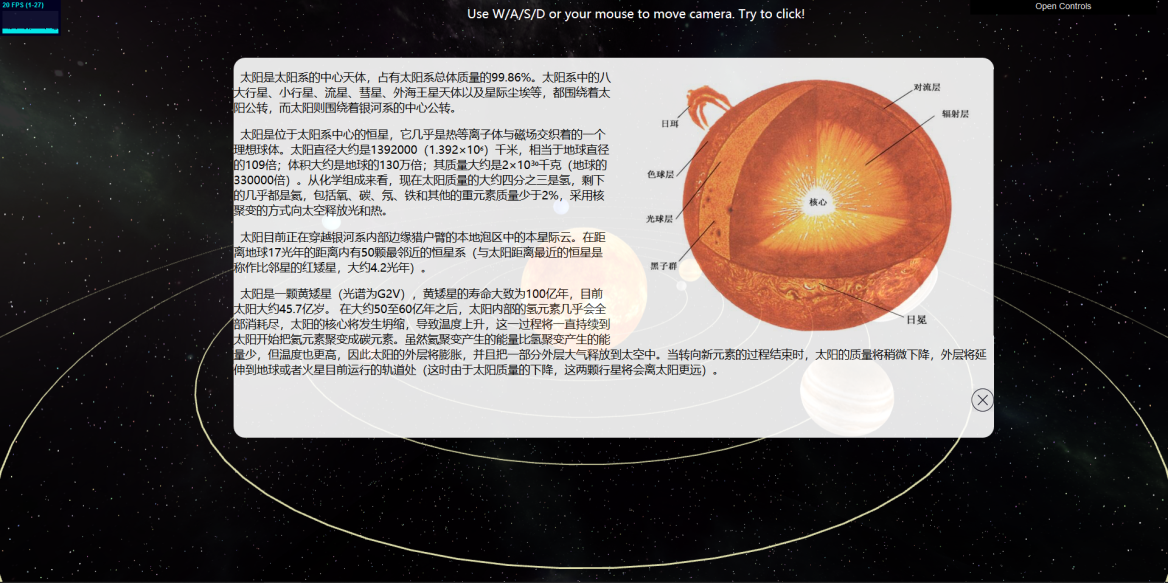


图5-4 信息及图解界面

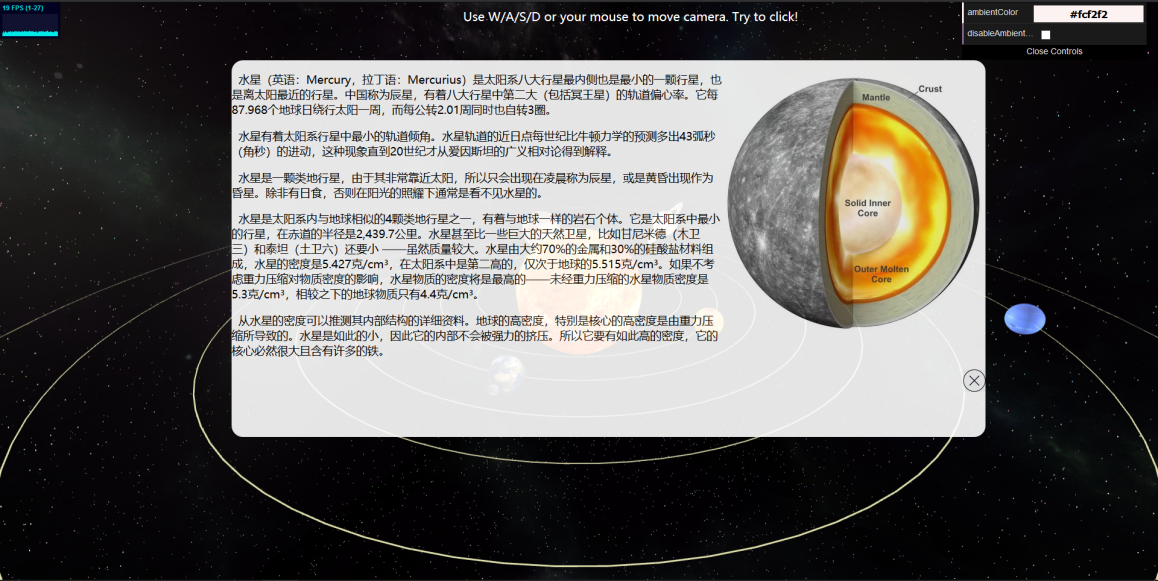


图5-5 其他信息及GUI界面

# 系统测试

## 6.1 功能测试

功能测试又称黑盒测试，通过列出测试用例来一一检测每个功能是否符合预期效果。本系统的功能测试用例如表6-1所示。

表6-1 功能测试用例表

|  |  |
| --- | --- |
| **测试操作** | **预期效果** |
| 鼠标拖拽太阳系 | 场景随鼠标操作发生相应变化 |
| 键盘控制太阳系 | 场景随键盘操作发生相应变化 |
| 点击星体 | 出现对应星体信息与图解面板 |
| 点击侧边栏导航 | 出现对应星体信息与图解面板 |
| 切换信息面板 | 信息更换 |
| 点击信息面板关闭按钮 | 关闭信息面板，返回场景 |

## 6.2 适配兼容测试

测试本系统在不同平台上是否适配，是否显示正常，功能是否符合需求，即在不同平台上测试上文中功能测试用例表中的操作是否正常。

# 结论

一转眼，大学生活即将结束，经过一个月的设计开发，本次毕业设计也大致开发完成。这次开发基本达到预期效果，实现设计目标所确定的功能，但也存在着有待改善的问题，未能实现的功能，将来仍可对该系统进行改进。

在确定题目时，对Three.js并无太多理解，在前端的学习上也并不深入，仅靠兴趣便确定了这个题目，并开始着手学习Three.js。由于中文材料极少，以及本人在英文方面的局限，对英文材料的阅读和理解进度较为缓慢，也因此导致了本次设计并不尽如人意。虽然本次毕业设计到此为止，但应继续加深关于Three.js的学习。

在开发过程中，遇到了较多困难，本人在图形学上也存在着问题，开发过程磕磕碰碰，但解决问题的过程仍是十分有意义的。这次开发让我学到了很多新知识，发现了自己的问题，在3D方面的了解也更加深入了。在独立开发的过程中，从需求分析、设计、开发到测试，整个过程使我受益匪浅。这次毕业设计靠兴趣挑战了一个从来没有使用过的新技术，提高了自己的学习和开发能力，希望能在这条路上越走越远。

道路很长，仍需不断努力。

## 

# 致谢

言至于此，心里惴惴不安。大学四年，一千余天，看似很长却如白驹过隙，还未来得及细细回忆便被紧凑仓促的毕业事宜推到了毕业前夕，四年前刚踏入大学的时刻仿佛就在昨天，此时站在四年旅程的终点，内心仍感茫然不知所措。

经历了大学四年的锤炼，我学会了迎难而上，学会了面对问题，从无知到慢慢成熟，感谢这四年陪伴我的所有人：永远支持我的父母、授业解惑的老师以及朝夕相处的同学，感谢你们在我人生途中的扶持与陪伴。

四年里，有打退堂鼓的时候，是父母给了我永远的避风港，回港休整后重振士气继续攻坚，感谢父母不变的关怀与支持。

毕业设计，是大学四年的最后一门课程。如今，这最后一门课也到了终点。在此十分感激廖晓娟老师在毕设过程中给予的支持和帮助，从选题到论文的完成，都离不开廖老师的指导。同时感谢学院所有老师在这四年中对我的教导，使我可以拥有一技之长。

在这一千四百多天中，与同学相处的时间居多，是你们丰富了我乏味的日常生活。一路的欢声笑语，鼓励安慰，扶持帮助，感谢你们一路相伴。

新的旅程即将开始，我将铭记四年时光，继续前行。

# 参考文献

Nicholas C.Zakas.2012.JavaScript高级程序设计[M]. 北京:人民邮电出版社.345-392.

Jos Dirksen.2015.Three.js开发指南[M]. 北京:机械工业出版社.207-237.

张亚飞.2011.Android、iPhone、Windows Phone手机网页及网站设计[M]. 北京:清华大学出版社.132-144.

刘西杰，柳林.2013.HTML、CSS、JavaScript网页制作从入门到精通[M]. 北京:人民邮电出版社.316-333.

连政.2014.基于HTML5技术的移动Web前端设计与开发[D]. 浙江:浙江工业大学.32-63.

郭伟伟.2008.CSS全程指南[M]. 北京:电子工业出版社.104-142.

张紫微.2010.Web前端性能优化的研究与应用[D]. 四川:电子科技大学.65-90.

David Flanagan.2007.JavaScript: The Definitive Guide[D]. State of California:O’Reilly.125-178.

Eric A.Mayer.2007.CSS: The definitive Guide [M]. State of California:O’Reilly.214-289.

Jos Dirksen.2015.three.js-cookbook[M]. Birmingham:Packt Publishing.18-56.