1 概 要

1. **项目简介**

本项目中的摄像机和模型的旋转运动都是在以地球为原点的极坐标系中进行的。

模型方面，本项目中出现三个物体的模型（括号内表示该物体在代码中的对应命名）：地球(earth)、月球(moon)、星空贴图的天空盒(starBox)。

光源方面，光照模型采用取正值的Blinn-Phong模型，叠加环境光、镜面反射光和漫反射光。本项目只出现一个可移动的光源（sun），该光源只考虑作用在地球上产生的反射和阴影。月球上的光照和阴影不受到sun光源的移动影响。

初始状况下，地球、月球、摄像头、光源位于同一条直线上。投影模式为透视投影。

默认状况下，所有的三个物体（地球、月球以及天空盒）以地球为原点、在地月所在的水平面上一起自动做逆时针匀速旋转运动。

1. **交互操作**
   * + 物体旋转
       - 按下X键，所有物体立刻在地月所在的水平面上沿顺时针方向移动一小段距离
       - 按下V键，所有物体立刻在地月所在的水平面上沿逆时针方向移动一小段距离；
     + 摄像头移动
       - 按下W键，摄像头立刻在以地球为原点极坐标系中向上移动一小段距离
       - 按下A键，摄像头立刻在以地球为原点极坐标系中向左移动一小段距离；
       - 按下S键，摄像头立刻在以地球为原点极坐标系中向下移动一小段距离；
       - 按下D键，摄像头立刻在以地球为原点极坐标系中向右移动一小段距离；
       - 按下Z键，摄像头立刻在以地球为原点极坐标系中向后移动一小段距离；
       - 按下C键，摄像头立刻在以地球为原点极坐标系中向后移动一小段距离；
     + 光源移动
       - 按下Q键，光源立刻在地月所在的水平面上沿逆时针方向移动一小段距离；
       - 按下E键，光源立刻在地月所在的水平面上沿顺时针方向移动一小段距离；
     + 切换投影模式
       - 按下P键，投影模式在平行投影和透视投影之间来回切换。
     + 自动旋转
       - 按下T键，所有物体立刻停止自动的逆时针旋转，或是立刻开始自动的顺时针旋转。
     + 重置摄像头状态
       - 按下R键，重置摄像头的位置。

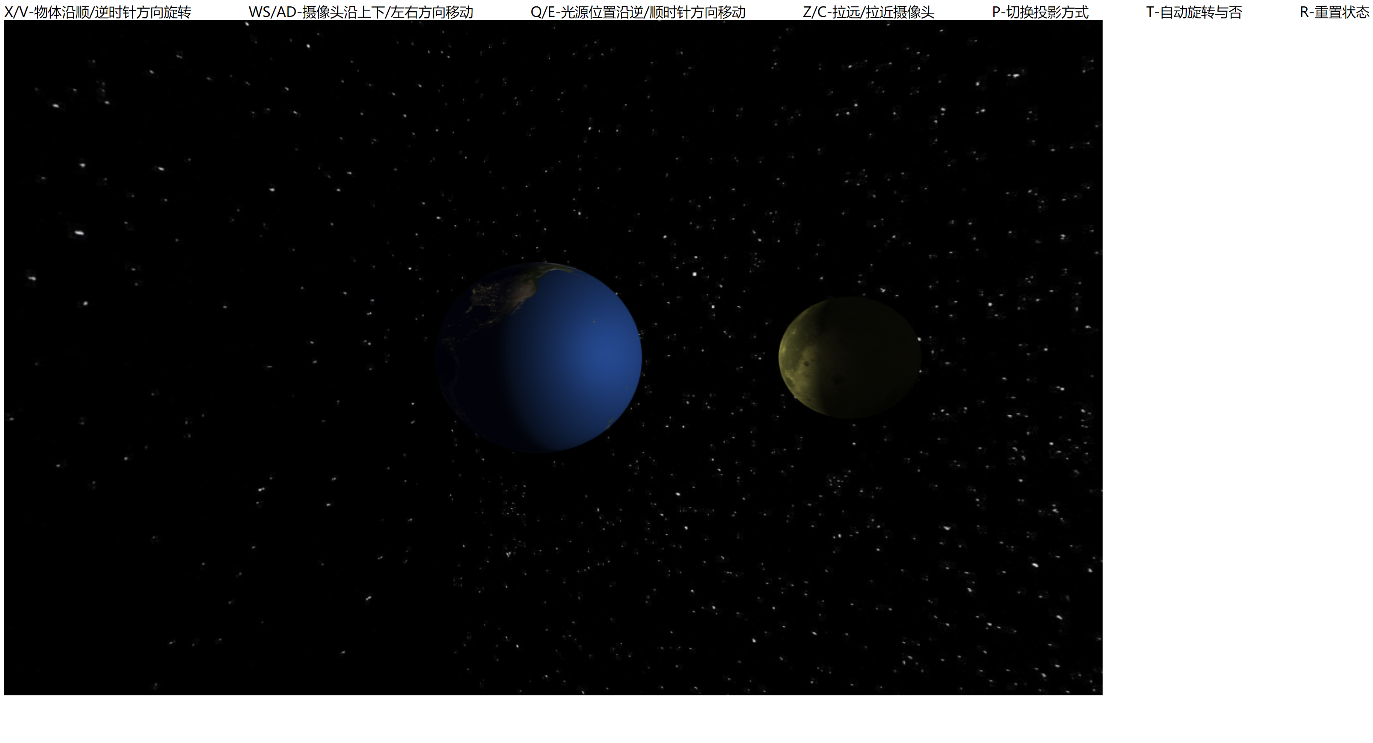
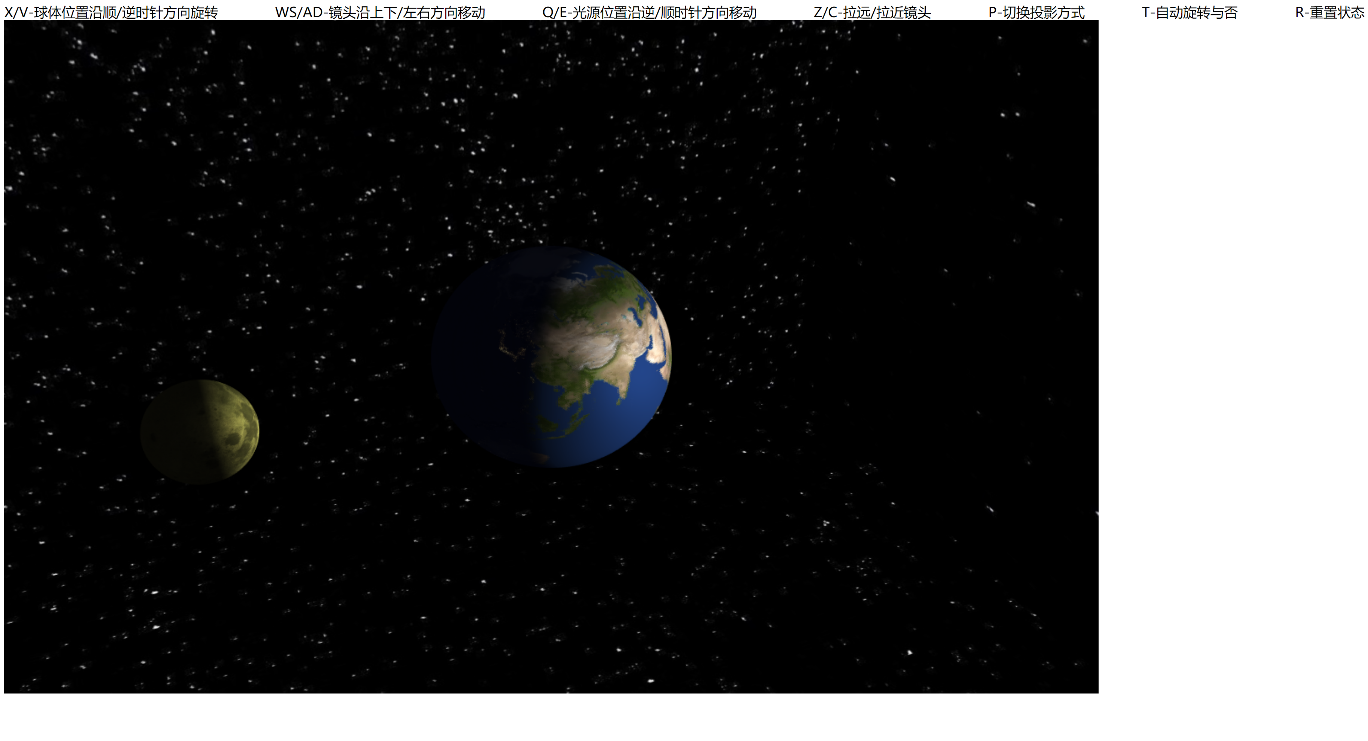


图1-1主要界面图

2 项目设计

1. **项目平台**

本项目基于WebGL绘图协议和GLSL ES 3.0版本，运行平台为64位的Chrome v87正式版，并配置为允许跨域文件。

1. **主要工作**

本项目中完成的主要工作是地球、月球、星空天空盒的建模、渲染和投影。

* 建模部分由球体建模和正方体建模这两大部分构成。

建模部分使用了多边形网格建模方法来完成地球、月球这两个球体的建模，以及天空盒这一个正方体的建模。

* 渲染部分由2D纹理贴图、多重纹理、光照模型、阴影贴图这四大部分构成。

渲染部分，2D纹理经由拓扑变换和拉伸变换映射到对应的球面上。使用多重纹理，用白天和黑夜两张纹理来同时渲染地球。光照模型采用了取正值的Blinn-Phong光照模型，搭建一个可移动的定向光光源来模拟太阳的平行光照射。阴影贴图通过计算纹理原来的颜色和光线强度（环境光、镜面反射光和漫反射之和），二者相乘来得到片元的实际颜色。

* 投影部分由模型变换、视点变换和投影切换这三大部分构成。

投影部分，通过键盘的操作交互方式来移动摄像头的位置或旋转地月系来观察模型。为了保证摄像头一直指向地球，摄像头的位置移动以三维极坐标系下的弧度来计算，当渲染每一帧前，需要将极坐标系下的角度转换为直角坐标系下的坐标。观察分为有平行投影、透视投影两种投影方式，为了符合真实感，默认状况下为透视投影。

1. **设计结构图**

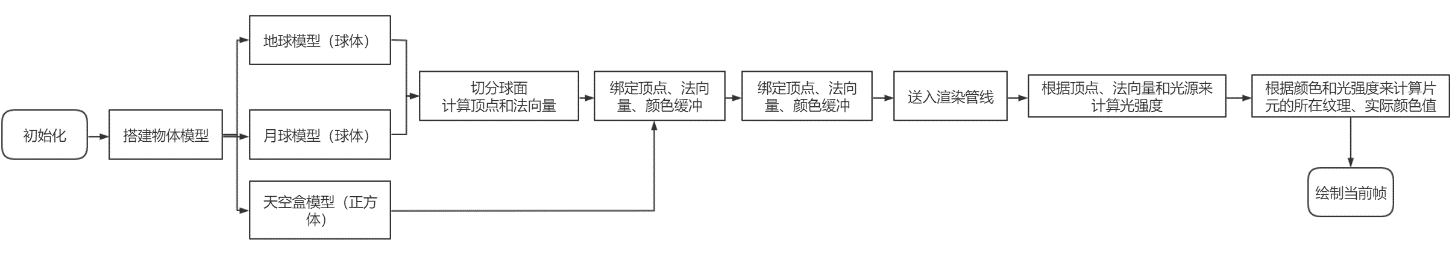
本项目的设计结构图如下：

图2-1 整体设计结构图

**2.1 场景物体模型设计**

本项目中一共出现了三个物体的模型（括号内表示该物体在代码中的对应命名）：地球(earth)、月球(moon)、星空贴图的天空盒(starBox)。其中，地球和月球都使用三位球体作为物体模型，而天空盒则使用正方体作为物体模型。本项目无导入模型。

1. **构建三维球体模型**

一个球心位于原点，半径为R的球面三维坐标系可以通过参数方程来表示，换句话说，球面上的每一个三角形的顶点坐标也可以通过参数方程来表示。[1]该方程如下。其中，

(1)

因此，球面的曲面可以按照经纬线的思路划分为若干个小四边形面片，而每个四边形又可以进一步划分为建模所需的两个小三角形，并通过参数方程来得到三角形的顶点坐标。所有的小三角形坐标共同构成了球面的顶点坐标。在做划分时，相邻两条纬线或经线之间的长度大小可以定义为步长。步长越小，球面的曲面越平滑。本项目中，根据实际需要，步长的大小被设定为1°。

1. **构建正方体模型**

正方形模型只需指定8个顶点的坐标即可。本项目中，正方体的边长被设定为150个单位，并依据这一长度来计算顶点的坐标。

**2.2 观察种的各种变换设计**

1. **物体变换（模型变换）**

本项目中的物体变换发生在三维极坐标系中，物体的变换以旋转为主，并且主要使用绕y轴（zox平面）的旋转变换。

rotateMatrix函数负责旋转的流程如下：本项目中，旋转角度（rotateAngle）会因为时间变化（自动旋转）或键盘输入（手动旋转）的变化而改变该变量的数值，再根据当前的旋转角度（rotateAngle）数值计算出当前表示标准旋转变换的齐次坐标矩阵（inputMatrix），最后在视点变换之前根据该矩阵进行模型变换，以实现物体位置的变换。

1. **相机变换（视点变换）**

场景中的物体变换都是在三维极坐标系中完成的，而模型的顶点坐标都是直角坐标系下的。因此，当进行相机变换或规范化投影变换时，需要将极坐标转换为直角坐标，其中，都以度数为单位，范围是。当达到360时，重新回到为0。极坐标和直角坐标之间转换的部分代码如下图所示。

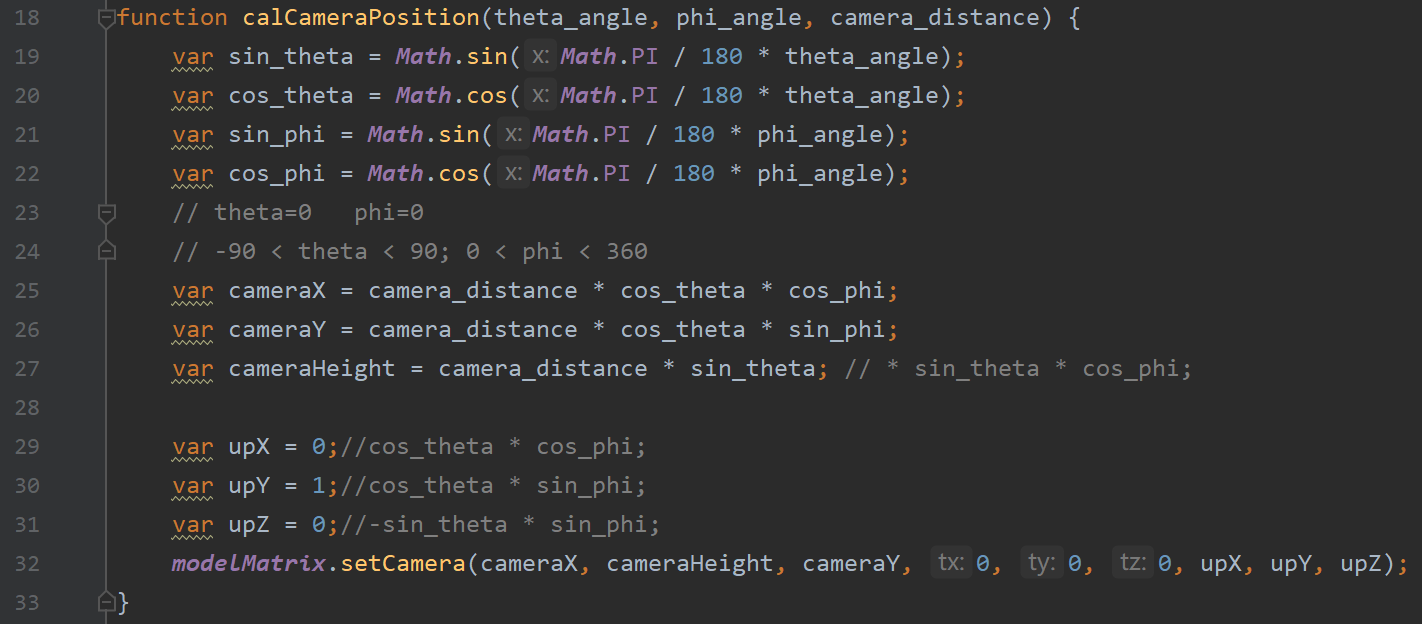


图2-2 视点变换代码图

1. **规范化投影变换**

与相机变换部分的思路类似，根据平行投影、透视投影两种投影方式得到不同的齐次坐标矩阵（inputMatrix），以实现投影变换。

**2.3真实感着色设计**

1. **光照**

光照模型采用了简化后的取正值Blinn-Phong光照模型，光照强度的计算公式如下：

(2)

(3)

其中，表示模型的光照强度。分别表示环境光、漫反射光和镜面反射光的强度。分别表示环境光、漫反射光和镜面反射光的反射系数。分别表示环境光强度和点光源强度。表示高光系数。表示光线的入射点处的单位法线向量。表示光的入射方向和视线方向。[2]

1. **纹理**

纹理方面采用了纹理映射、纹理拉伸和线形纹理采样，将2D贴图通过拓扑变换和拉伸变换的方式映射到对应的球面上再进行线形纹理采样。为了提高真实性和美观性，地球模型上采用了多重纹理（白天纹理和夜晚纹理）来额外模拟夜晚的灯光，并运用过程纹理的技术，在“晨昏线”处根据白天纹理和夜晚纹理二者的比例生成图形纹理，以实现平滑的过渡效果。[3]

项目中的2D贴图均来自Solar System Scope网站，使用了其中的” EARTH DAY MAP”, “EARTH NIGHT MAP”, “STARS”以及”MOON”共计四张贴图，并且经过必要的切割和大小调整。[4]

1. **阴影**

表面绘制（明暗处理）使用了简化后的Phong Shading，在构建三维球体模型时，球体上的每个多边形的顶点处的法向量都记录在Sphere.normalArray中。法向量被送入渲染管线后，先进行归一化后转为单位法向量，再按照光照模型来确定最终的光强，通过如下公式的计算得到阴影纹理的显示颜色。[5]

(4)

此外，地球使用了多重纹理，因此地球的阴影处的纹理会由单独的夜晚纹理来替代。当片元的漫反射光强度低于0.05时，就用夜晚的纹理来渲染，作为地球的阴影。反之，当漫反射光光强度高于0.25时，则用白天的纹理进行渲染。当漫反射光在0.05和0.25之间时，使用过程纹理技术，对白天纹理和夜晚纹理进行叠加处理，使视觉效果更为自然。

3 总结

1. **评价**

本项目较好地完成了所有的作业要求：

* 建模方面，完整搭建了球体和正方体模型；
* 观察方面，模型可旋转，移动摄像机视点位置，投影可在平行投影、透视投影之间切换；
* 着色方面，场景包含了一个可移动的太阳光源，并在地球上实现光照颜色的计算，模型的表面都有贴图，表面颜色是光照计算和纹理色的混合；
* 特效方面，实现了阴影和环境纹理，效果兼顾真实性和美观性；
* 交互方面，整体视觉效果较好，交互设计易懂明朗。

1. **不足**

不足之处是，地球的2D纹理在拓扑变换和拉伸变换的过程中变形较为严重，导致世界地图出现错误，影响观感。因此考虑使用其它更精确的映射方法来实现纹理的映射。

此外，星空贴图的天空盒因为使用的是正方体，导致棱的位置贴图过渡不平滑、不自然，而且星空的疏密程度在正方体上的每一面并不一致，影响美观性。因此考虑使用顶点数目更多的多面体乃至球体来替代天空盒，以实现更好的效果。

1. **问题和解决途径**

在编程过程中，遇到了很多问题，基本都是通过网上搜索相关资料和官方文档来解决的，少部分疑难杂症通过询问学长和同学予以了妥善的解决。问题主要分为如下的两类：

* 1. 技术层面上的问题，这一方面的问题数量虽然没有编程和语法方面的问题多，但是定位较为麻烦，而且解决方案也不容易找。例如，浏览器文件跨域问题导致无法加载纹理图片文件，后来通过配置Chrome浏览器快捷方式得到解决；GLSL语言版本的兼容性问题导致无法调用in和out变量类型，后来通过在着色器语言GLSL代码的首行标注并指定所需的es版本得以解决；[3]模型发生遮挡颜色显示顺序异常的问题，后来通过开启深度检测模式来解决，诸如此类问题；[6]
  2. 编程和语法方面的问题，例如各种爆栈、传参错误、状态初始化错误等问题，这些问题虽然多而零碎，不过，好在可以通过浏览器的检查页面的错误报告进行快速的定位，而且可以调用console.log进行输出和显示，再加上网上相关的前端解决方案较多，所以带来的困扰并不比技术层面上的问题更多。

1. **收获**

图形学是一门复杂的、涉及到多方面知识综合运用的学科，在编写项目的过程中我深切地体会到了这一点。这一过程中，不断地尝试理解和深入WebGL、JS、GLSL的整体技术框架，动手实践数学模型的转化和管线的渲染流程，在反复的修改与调试推翻原有的猜想和误解，拓展了原本无法接触到的知识面，获得了更高层面的认识和理解。尽管最终收获得到的图形效果较之前预想的更好，但与过程的学习所得相比，似乎显得不那么重要了。

从最初的一系列抽象的数值和公式，到最终结果的具体图形和场景，虽然图形学的实现过程十分艰辛，但也令人倍感值得。而这“柳暗花明又一村”的过程，就是最大的体验和收获。

**参考文献**

[1]张尤赛,陈福民.基于纹理映射与Phong光照模型的体绘制加速算法[J].中国图象图形学报,2003(09):81-87.

[2] 陈家新,周纬杰.一种改进的Phong光照模型及光线跟踪实现[J].系统仿真学报,2002(12):1642-1643+1646.

[3]Kouichi Matsuda,Rodger Lea.WebGL编程指南[M].电子工业出版社:北京,2014:69-393.

[4]INOVE.Solar Textures[EB/OL].https://www.solarsystemscope.com/textures/,2019-6.

[5] Edward Angel, Dave Shreiner.交互式计算机图形学[M].电子工业出版社:北京,2012:96-221.

[6] MDN contributors.WebGLRenderingContext[EB/OL].https://developer.mozilla.org/zh-CN/docs/Web/API/WebGLRenderingContext,2020-10.