学号	姓名	论文规范性（10）	问题分析与调研（30）	方案创新性（20）	实验结果分析与讨论（40）	结课论文总成绩（100）
21301150	韩新阳	5	15	10	23	53

《计算机图形学》(24 春)期末课程论文

一. 摘要

通过三维图形的显示原理和方法，矩阵堆栈函数，三维观察变换绘制绘制太阳系线框球体绘制函数实现球体绘制，绘制相对大小的地球，太阳，和月球，通过矩阵来实现实现地球绕太阳的公转和自转

二. 正文

(一) 实验目的

1. 掌握三维图形的显示原理和方法，掌握三维观察的原理和方法；
2. 掌握 OpenGL 中矩阵堆栈函数的使用，会使用堆栈函数进行复杂场景的组装。
3. 掌握 OpenGL 中三维观察变换常用的函数的使用方法，了解三维模型的贴图方法；
4. 通过 OpenGL 绘制太阳系

(二)实验内容：

使用线框球体绘制函数实现球体绘制。当按下键盘“D”或“d”时，行星将实现自转；按下键盘“Y”或“y”时，行星将绕太阳公转。实现以下两个内容：

(1) 给行星加上卫星;

(2) 实现自动旋转功能，即卫星绕行星自动旋转、行星自动自传同时绕太阳公转。

(三)实验技术

1、模型变换和视图变换：

从“相对移动”的观点来看，改变观察点的位置与方向和改变物体本身的位置与方向具有等效性。在 OpenGL 中，实现这两种功能甚至使用的是同样的函数。由于模型和视图的变换都通过矩阵运算来实现，在进行变换前，应先设置当前操作的矩阵为“模型视图矩阵”。

2、投影变换：

投影变换就是定义一个可视空间，可视空间以外的物体不会被绘制到屏幕上。OpenGL 支持两种类型的投影变换，即透视投影和正投影。投影也是使用矩阵来实现的。

3、操作矩阵堆栈：

矩阵堆栈指的就是内存中专门用来存放矩阵数据的某块特殊区域。

一般说来，矩阵堆栈常用于构造具有继承性的模型，即由一些简单目标构成的复杂模型。矩阵堆栈对复杂模型运动过程中的多个变换操作之间的联系与独立十分有利

进行矩阵操作时，有可能需要先保存某个矩阵，过一段时间再恢复它。当我们需要保存时，调用 glPushMatrix 函数，它相当于把矩阵（相当于盘子）放到堆栈上。当需要恢复最近一次的保存时，调用 glPopMatrix 函数，它相当于把矩阵从堆栈上取下。OpenGL 规定堆栈的容量至少可以容纳 32 个矩阵，某些 OpenGL 实现中，堆栈的容量实际上超过了 32 个。因此不必过于担心矩阵的容量问题。通常，用这种先保存后恢复的措施，比先变换再逆变换要更方便，更快速

(四)实验场景

1. 制作的是一个三维场景，包括了太阳、地球和月亮。假定一年有 12 个月，每个月 30 天。每年，地球绕着太阳转一圈。每个月，月亮围着地球转一圈。即一年有 360

天。现在给出日期的编号（0~359），要求绘制出太阳、地球、月亮的相对位置示意图。

2. 认定这三个天体都是球形，且他们的运动轨迹处于同一水平面，建立以下坐标系：太阳的中心为原点，天体轨迹所在的平面表示了 X 轴与 Y 轴决定的平面，且每年第一天，地球在 X 轴正方向上，月亮在地球的正 X 轴方向。
3. 确立可视空间。太阳的半径要比太阳到地球的距离短得多。如果直接使用天文观测得到的长度比例，则当整个窗口表示地球轨道大小时，太阳的大小将被忽略。因此，只能成倍的放大几个天体的半径，以适应观察的需要。为了简单起见，把三个天体都想象成规则的球体。

(五)实验过程

1.地球公转&自转：

首先单实现地球绕太阳的公转和自转。使用线框球体绘制函数实现球体绘制。当按下键盘“D”或“d”时，行星将实现自转；按下键盘“Y”或“y”时，行星将绕太阳公转。

2.OpenGL 中并无绝对单位，只有相对大小

OpenGL 中所有的变化都是局部变化，也就是说 OpenGL 中所有物体的变换都以当前物体的中心为原点，只不过有时候他自己的中心恰好在默认的原点。

OpenGL 是先对物体进行操作（先罗列一堆变换等函数），再画这个物体

画球体的函数：glutWireSphere(半径，球体纬线的条数，球体经线的条数)

3.太阳系：

坐标变换过程：

太阳：不变，在中心即可。

太阳在坐标原点，不需要经过任何变换，直接绘制就可以。

地球：

1） 用循环改变 SolarAngle 的值，调用 glRotatef()函数实现公转；

2） 用循环改变 OwnAxisAngle 的值，调用 glRotatef()实现自转；

3） 调用 glTranslatef()函数将其平移至公转轨道，并随时间在轨道上改变位置。

```
glRotatef(day, 0, 0, –1);
/ * 地球公转是“自西向东”的，因此是饶着 Z 轴负方向进行逆时针旋转 */
glTranslatef(地球轨道半径, 0, 0);
glutSolidSphere(地球半径, 20, 20);
```

月球：

是最复杂的。因为它不仅要绕地球转，还要随着地球绕太阳转。但如果我们选择地球作为参考，则月亮进行的运动就是一个简单的圆周运动了。如果先绘制地球，再绘制月亮，则只需要进行与地球类似的变换：

glRotatef(月亮旋转的角度, 0, 0, –1);

glTranslatef(月亮轨道半径, 0, 0);

glutSolidSphere(月亮半径, 20, 20)

但这个“月亮旋转的角度”，并不能简单的理解为 day/一个月的天数 30*360 度。因为在绘制地球时，这个坐标已经是旋转过的。现在的旋转是在以前的基础上进行旋转，因此还需要处理这个“差值”。可以写成：day/30*360 – day，即减去原来已经转过的角度。

4. 投影、模型、视点变换的函数：

glMatrixMode(GL_PROJECTION);//设置成模型矩阵模式(投影变换)

glLoadIdentity();//载入单位化矩阵

//透视变换角度 75 度，长宽比 1：1，最近可视距离 1，最远 400000 两倍地球公转半径

gluPerspective(75,1,1,400000);

glMatrixMode(GL_MODELVIEW);//设置成模型矩阵模式（模型、视点变换）

glLoadIdentity();//载入单位化矩阵

//整体布局，视角位（这里是 45 度倾角），物体位置，z 轴正向

gluLookAt(0,−200000,−200000,0,0,0,0,0,1);

对象的变换都是通过矩阵来实现的，在进行矩阵操作前，需用 glMatrixMode()指定当前操作的矩阵对象，GL_PROJECTION 是进行投影变换，GL_MODELVIEW 告是进行模型、视点变换了。

（六）实验环境

OpenGL 版本：OpenGL 4.6

开发环境：Visual Studio 2019

编程语言：C++

OpenGL 库：OpenGL Mathematics (GLM)、OpenGL Extension Wrangler Library (GLEW)、Simple OpenGL Image Library (SOIL) 等

模型和纹理资源：采用常见的 3D 模型格式（如 OBJ）和图像格式（如 PNG）加载到程序中。

（七）实验总结

本次实验仍然是参照 LearnOpenGL 上的教程进行学习的，在提升了代码能力的同时，还让我加深了图像学相关知识的理解，深入学习了三维图形建模和渲染的整个过程。不仅提高了对 OpenGL 及其关键功能的理解，还掌握了如何制造真实感光照和使用纹理来增强视觉效果的方法。

三. 参考文献

【OpenGL 修行】三维变换讲解及实例_c++opengl 变换观察点-CSDN 博客

OpenGL——矩阵堆栈的入栈出栈详解及金字塔案例-CSDN 博客