

计算机图形学实验

实验2、OpenGL绘制基础

姓 名：\_\_\_\_高远昊\_\_\_\_\_\_\_\_\_\_\_

学 号：\_\_\_\_\_37320222204082\_\_\_\_\_\_\_\_\_\_

学 院：\_\_\_\_\_信息学院\_\_\_\_\_\_\_\_\_\_

专 业：\_\_\_\_\_数字媒体技术\_\_\_\_\_\_\_\_\_\_

年 级：\_\_\_\_\_\_2022\_\_\_\_\_\_\_\_\_

2023年 5月 16 日

目录

[Task1：实现镂垫程序 3](#_Toc166944382)

[1.理解并实现镂垫程序 3](#_Toc166944383)

[2. 为不同三角形设置不同的颜色 3](#_Toc166944384)

[3. 生成动画 5](#_Toc166944385)

[Task2: 完成三维程序 6](#_Toc166944386)

[1.设置合适的相机位置和相机投影矩阵，使得Opengl相机能够看到 6](#_Toc166944387)

[2.改写程序 8](#_Toc166944388)

[Task3: 完成三维程序 13](#_Toc166944389)

[1．生成动画 13](#_Toc166944390)

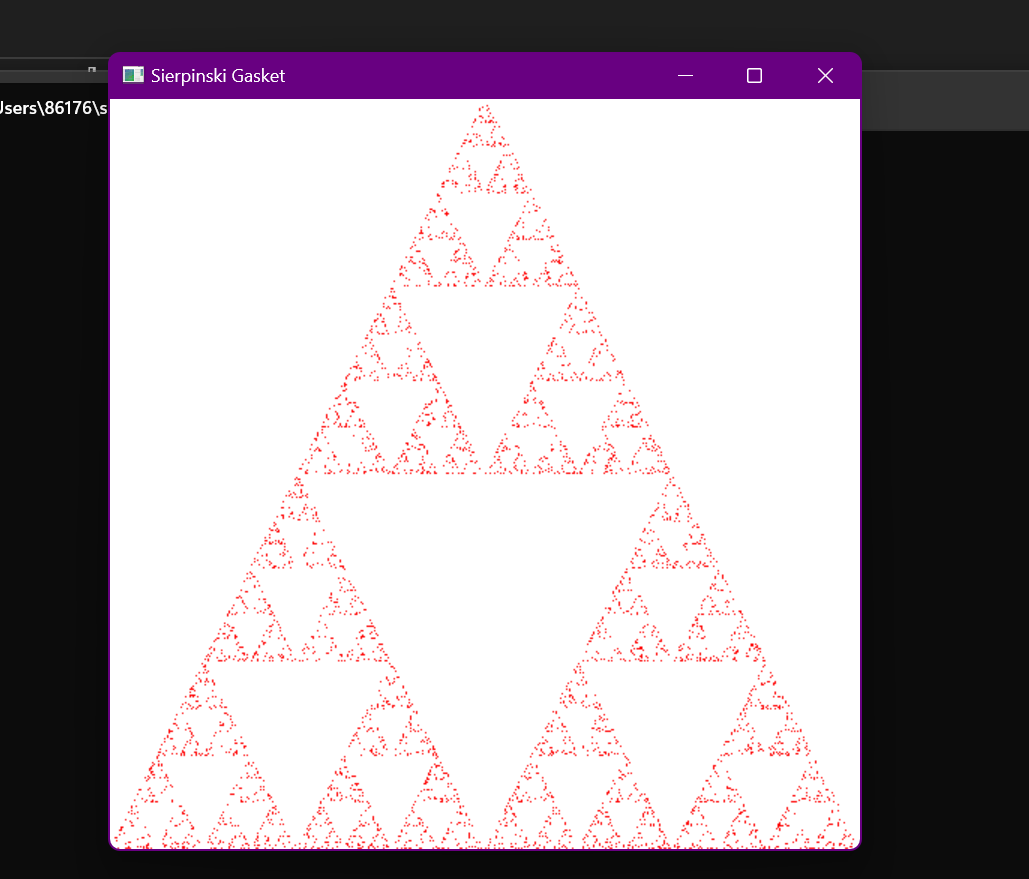
# Task1：实现镂垫程序

## 1.理解并实现镂垫程序

操作流程：直接运行

关键代码截图：略

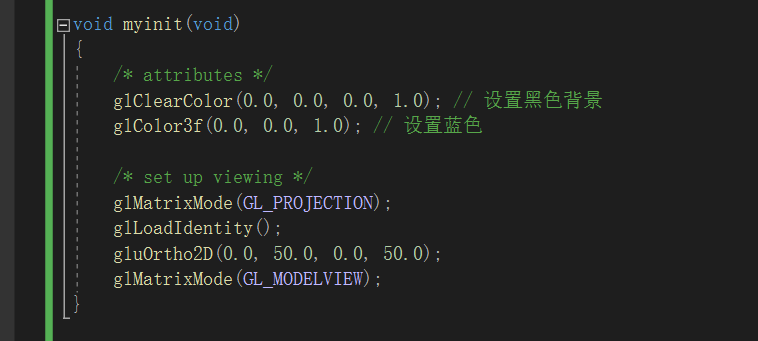
运行结果截图：



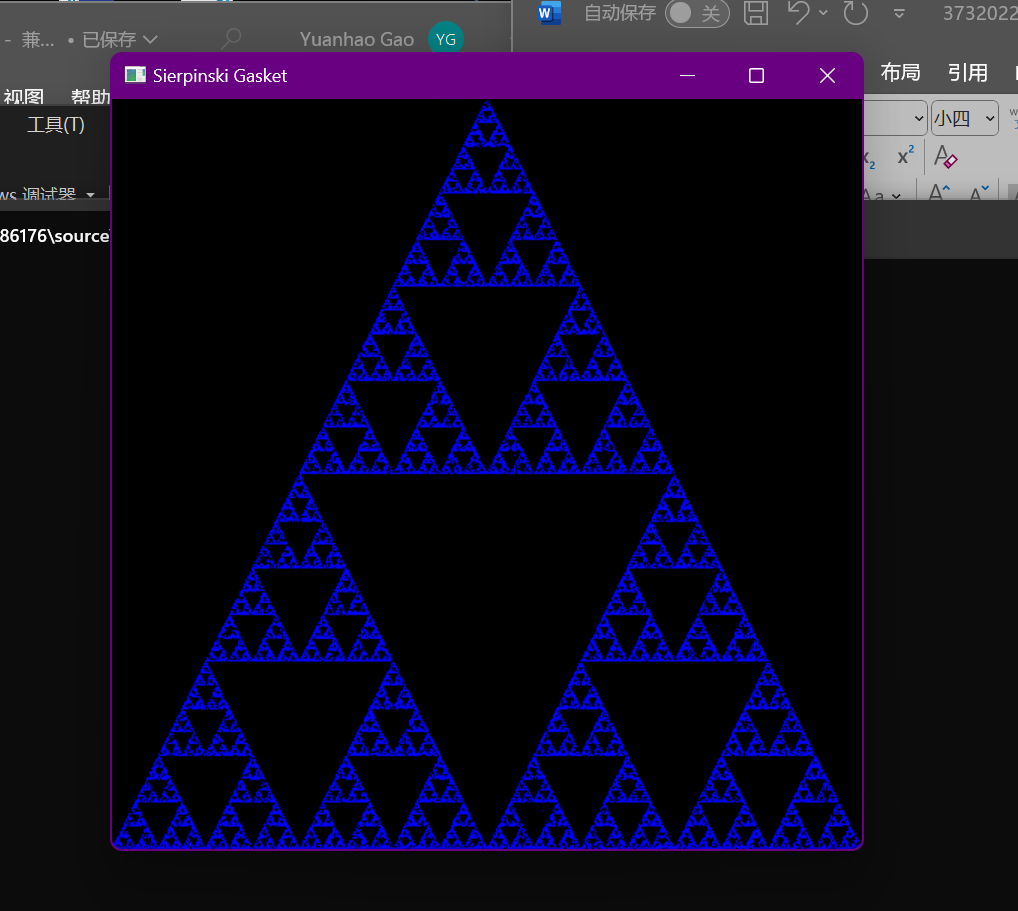
## 2. 为不同三角形设置不同的颜色

操作流程：用glClearColor设置背景，用glcolor3f更改颜色

关键代码截图：



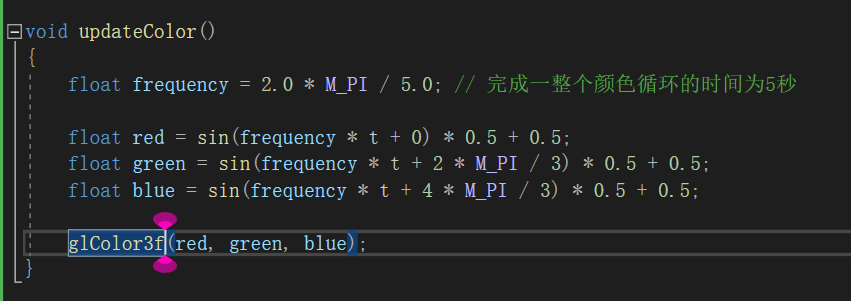
运行结果截图：

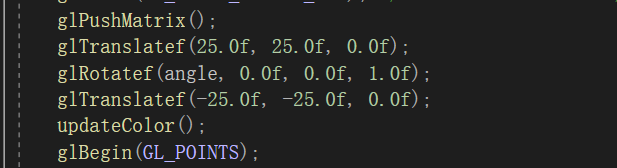


## 3. 生成动画

操作流程：

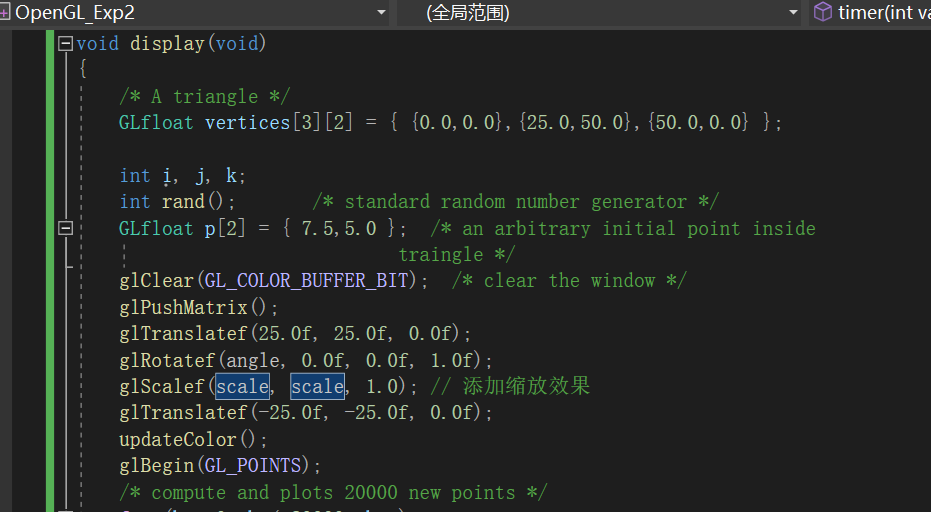
颜色变化：定义计时器函数，调用计时器更改颜色，重新绘制；



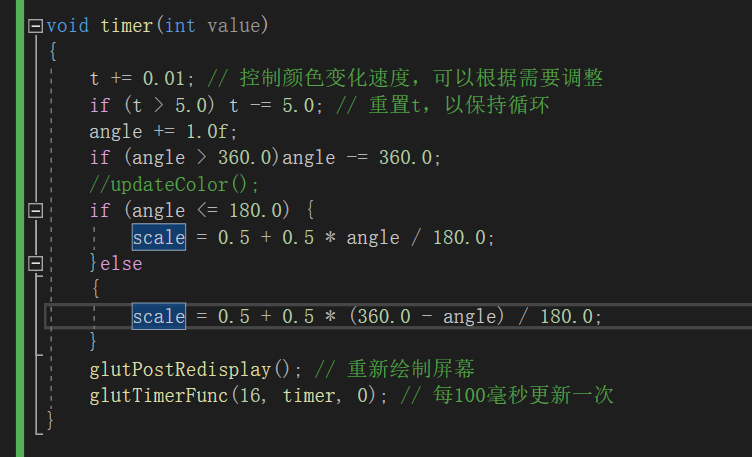
旋转效果：在绘制点集之前用glTranslate()将旋转中心固定，glRotatef()实现旋转；

缩放效果：

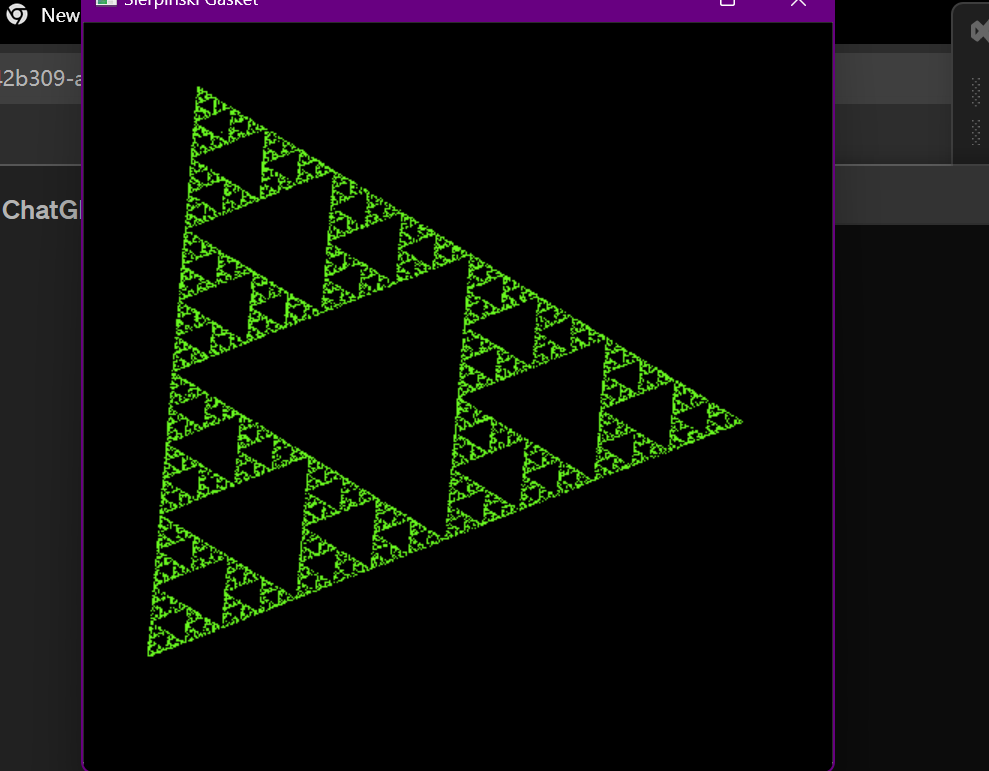
用glScalef()控制缩放，scale在timer中变化，



如下图，scale先从0.5开始，随着角度增大而变大，等到旋转半圈时，随着角度增大而变小；



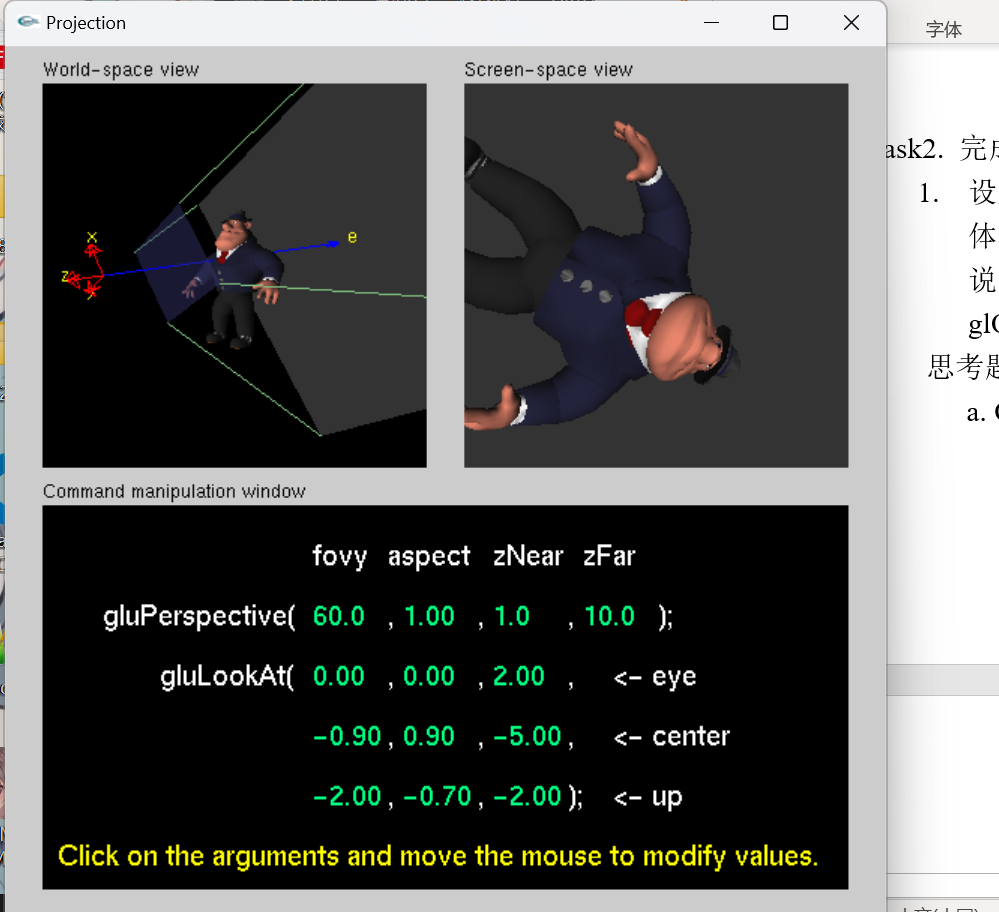
效果：



# Task2: 完成三维程序

## 1.设置合适的相机位置和相机投影矩阵，使得Opengl相机能够看到

运行结果：



思考题：

1. Opengl中，三维空间的坐标系是怎么样的？

答：包含世界坐标系，照相机坐标系，屏幕坐标系，原点在左下角

顶点坐标在对象坐标系或世界坐标系中定义；

1. OpenGL中，相机的方位是怎么样的？如何调整相机的朝向呢？

照相机包含6个自由度，由相机的位置（位置）和朝向（方向）来定义；

gluLookAt()来设置相机的位置和朝向；

1. OpengL中，相机可见范围是如何设置的？

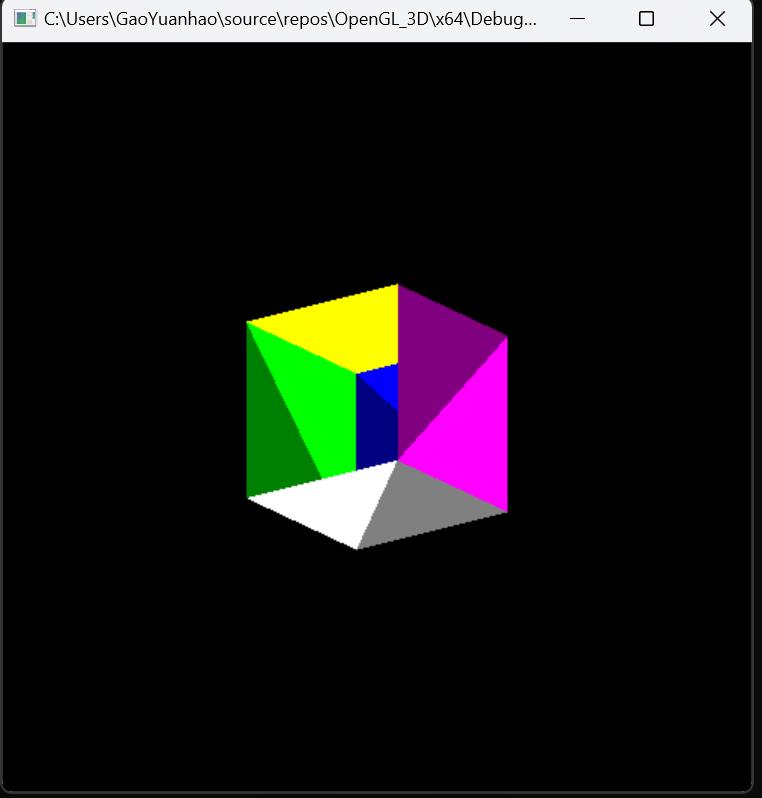
视景体，相机的投影矩阵来设置的；

## 2.改写程序

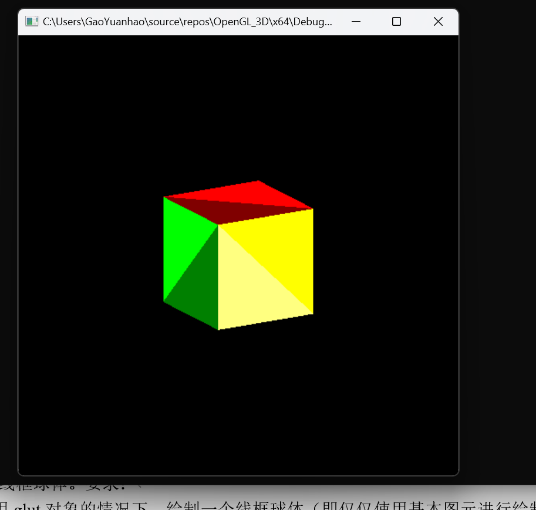
操作流程：

1. 比较开启/不开启深度缓冲区 glEnable(GL\_DEPTH\_TEST) 的效果;理解深度缓冲区的作用、用法

运行结果截图：不开：



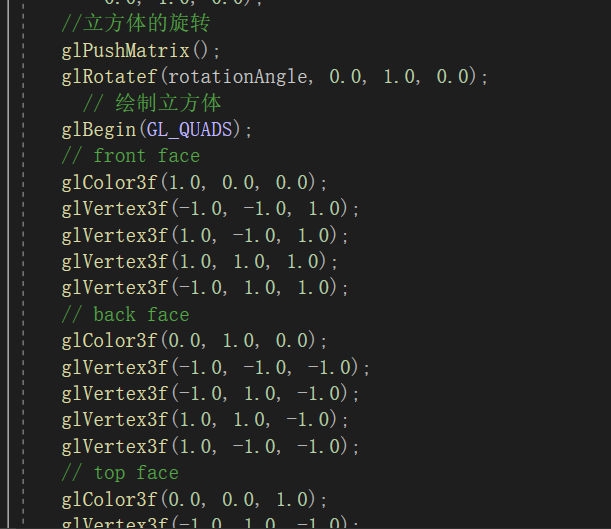
开：



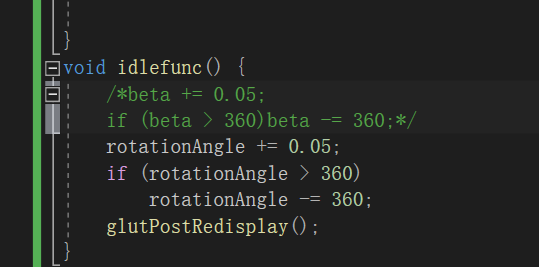
深度缓冲区：用于存储场景中每个像素的深度值的缓冲区。在渲染过程中确定哪些像素应该被绘制在其他像素之前，以便正确处理遮挡关系。

1. 让正方体自行旋转，而不是相机旋转；

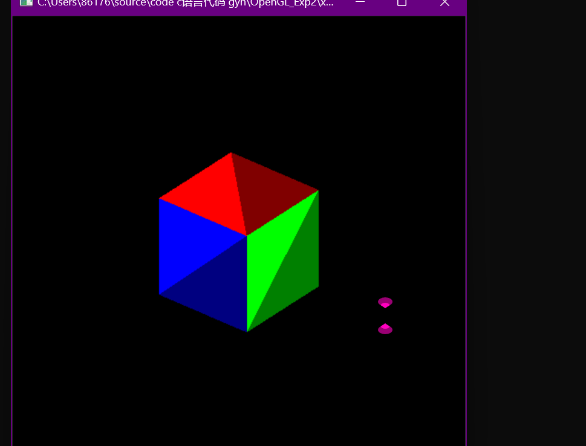
用glRotatef确定旋转轴为y轴



用rotationAngle取代beta作为旋转角；



效果截图：

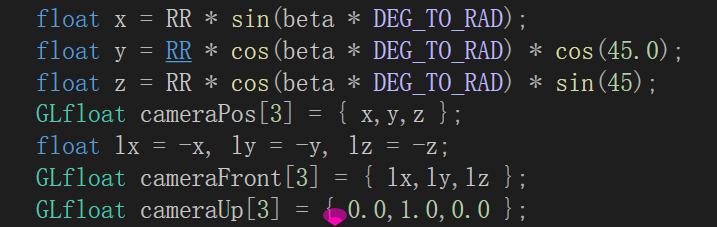


1. 实现交互式的相机控制（wasd控制相机的前进后退左右移动，qe实现相机的升降，使用鼠标调整相机的朝向，L锁定相机的移动和旋转）

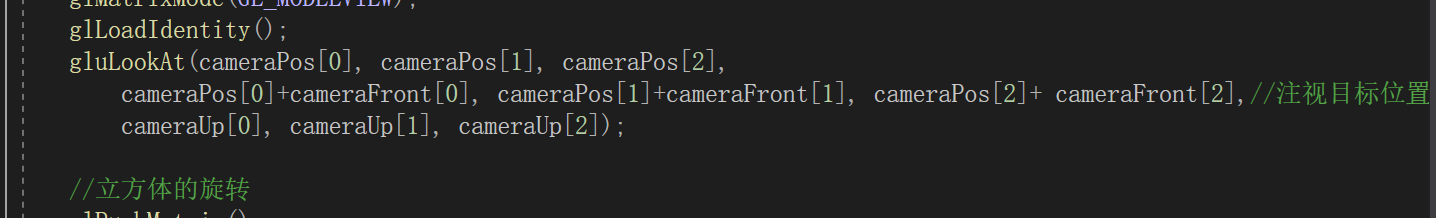
相机包括相机位置cameraPos,朝向cameraFront,上向量cameraUp;

gluLookAt的第一个参数为cameraPos,第三个参数为cameraUp,但是第二个参数是目标位置，因此要替换为cameraPos+cameraFront;

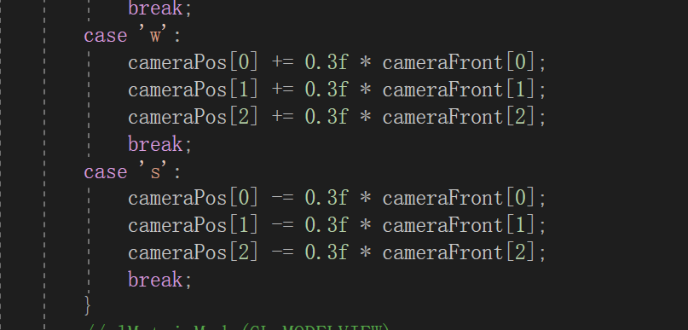
由助教所给的源程序可知，



因此gluLookAt对应替换



对相机的控制也就是对glulookAt的控制，准备工作结束，下面实现移动



W就是位置cameraPos向朝向cameraFront增加，也就是更靠近朝向的物体；S相反；

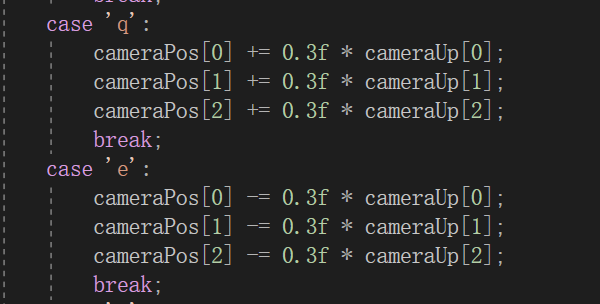
对于D，要计算了相机正前方向量 cameraFront 和上方向向量cameraUp 的叉乘；

叉乘操作返回了一个垂直于这两个向量的新向量，即相机的右方向向量。

A左方向向量就是上方向向量与朝向叉乘；(代码用的减)；



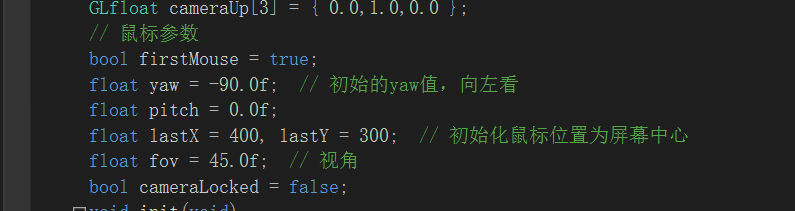
QE实现与WS类似，相机位置Pos向上方向向量增加；



L定义了bool cameraLocked = false;来判定是否调用keyboard回调函数；

鼠标需要用 glutPassiveMotionFunc(mouse\_callback);来实现；

先初始化：



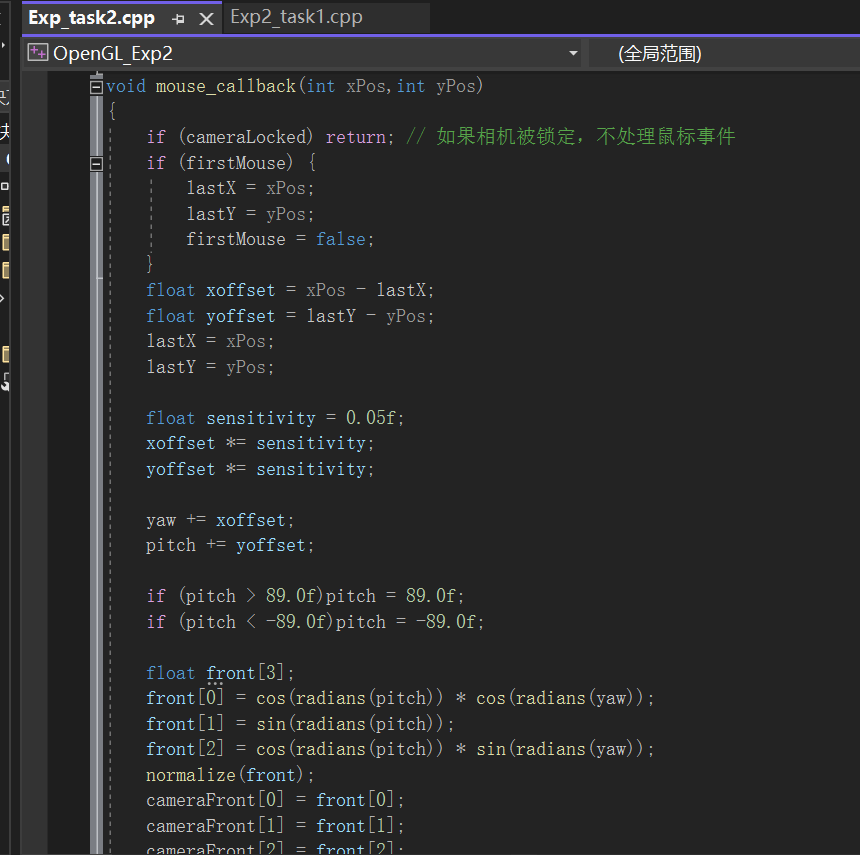
首先用firstMouse捕获上一帧的鼠标位置，存到lastX,lastY中；

计算当前帧鼠标位置和上一帧的偏移量，用xoffset和yoffset表示；

Sensitivity为鼠标灵敏度；把偏移量加到yaw角和pitch角上；

yaw：水平角，从X轴正方向起，绕Z轴旋转的角度（也称为方位角）。pitch：垂直角，从X-Y平面起，绕X轴旋转的角度（也称为仰角）。

从球面坐标转换为笛卡尔坐标，计算计算最终方向向量front[3];



计算原理：

X分量 (front[0])：cos(pitch) 将仰角压缩到XY平面上。cos(yaw) 将结果沿X轴投影。

因此， front[0] = cos(pitch) \* cos(yaw)。

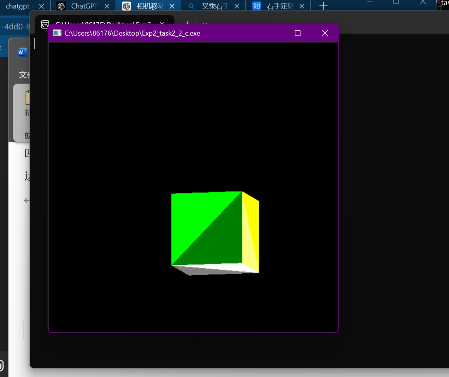
Y分量 (front[1])：sin(pitch) 直接表示在垂直方向上的投影。

因此， front[1] = sin(pitch)。

Z分量 (front[2])：cos(pitch) 将仰角压缩到XY平面上。sin(yaw) 将结果沿Z轴投影。

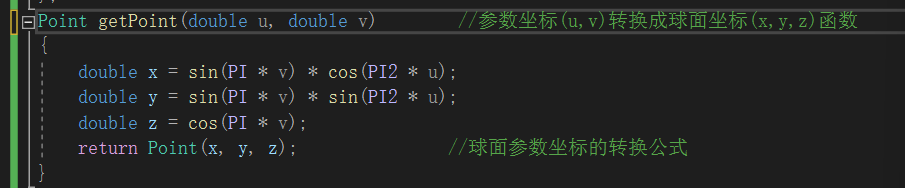
因此， front[2] = cos(pitch) \* sin(yaw)。

运行截图：

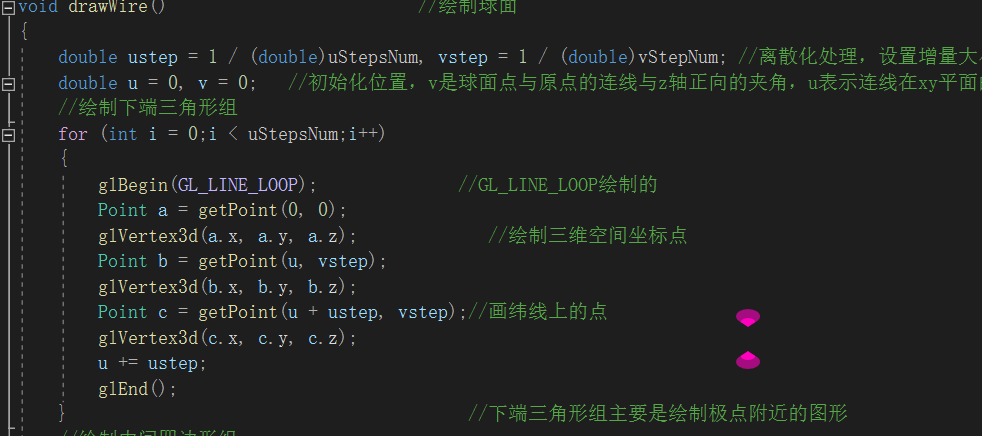


# Task3: 绘制一个线框球体

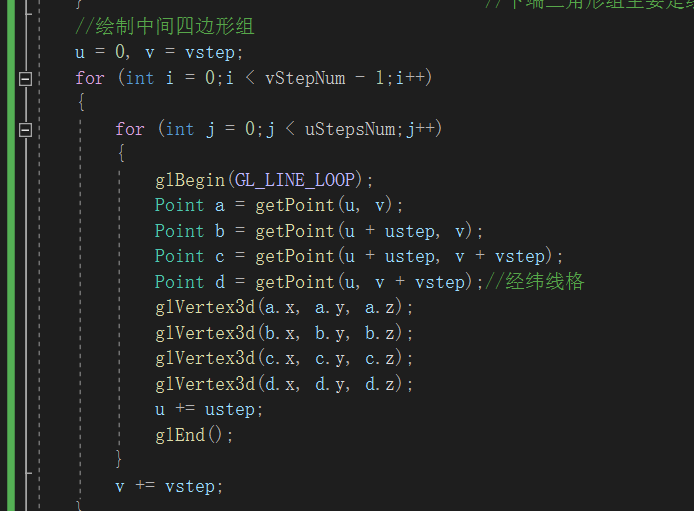
## 使用基本图元绘制

将参数坐标转化为球面坐标

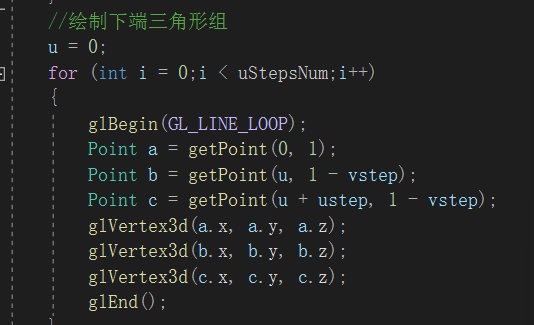
先画极点，极点三角形，vstep画第一根纬线上的点；u+=ustep遍历经线上的点



绘制经纬线格四个点，中间的纬线

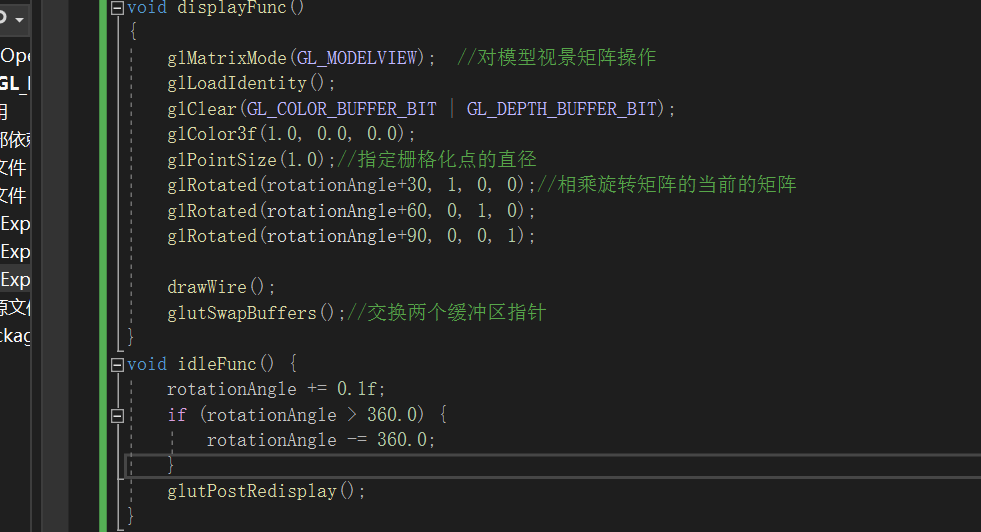


绘制下端三角形组：



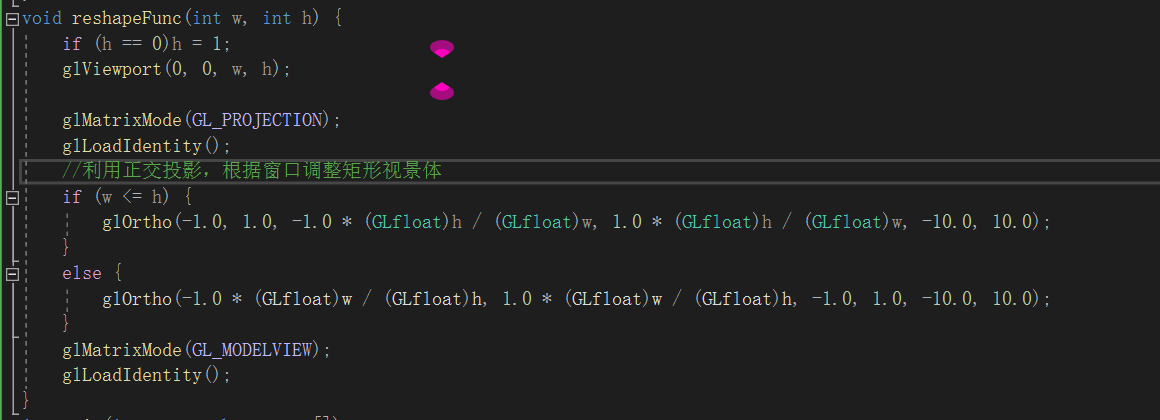
## 添加动画

由于球的绘制自身运用了旋转，在此基础上用rotationAngle来增加角度，实现旋转效果；

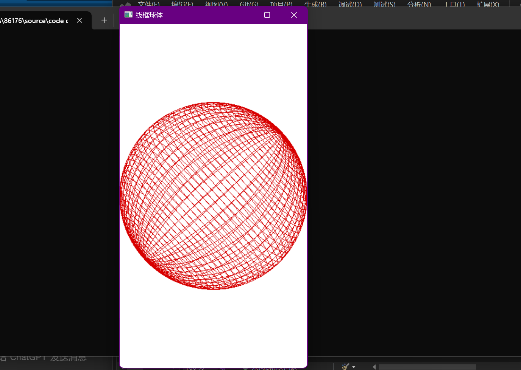


## Reshape

glOrtho()调整，太瘦了就调整bottom和top,使视景体上下变大；



效果：



## 交互

直接抄task的代码，略

