

计算机图形学实验

实验二、OpenGL绘制基础

姓 名：\_\_\_\_潘腾凯\_\_\_\_\_\_\_\_\_\_\_

学 号：\_\_\_37220232203786 \_\_\_

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

专 业：\_\_\_\_\_\_\_软件工程\_\_\_\_\_\_\_\_

年 级：\_\_ \_\_\_\_2023\_\_\_\_\_\_\_\_\_

2023年 4 月 3 日

目录

[Task1：题目名 3](#_Toc97636928)

[1.（情况一） 3](#_Toc97636929)

[2.（情况二） 3](#_Toc97636930)

[Task2: 题目名 3](#_Toc97636931)

[1.（情况一） 3](#_Toc97636932)

[2.（情况二） 3](#_Toc97636933)

# Task1：Sierpinski镂垫程序

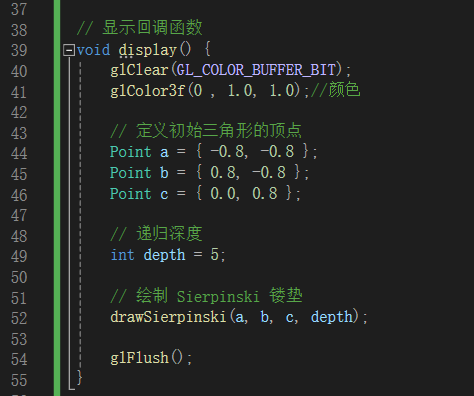
## 1.理解并实现课本程序

操作流程：

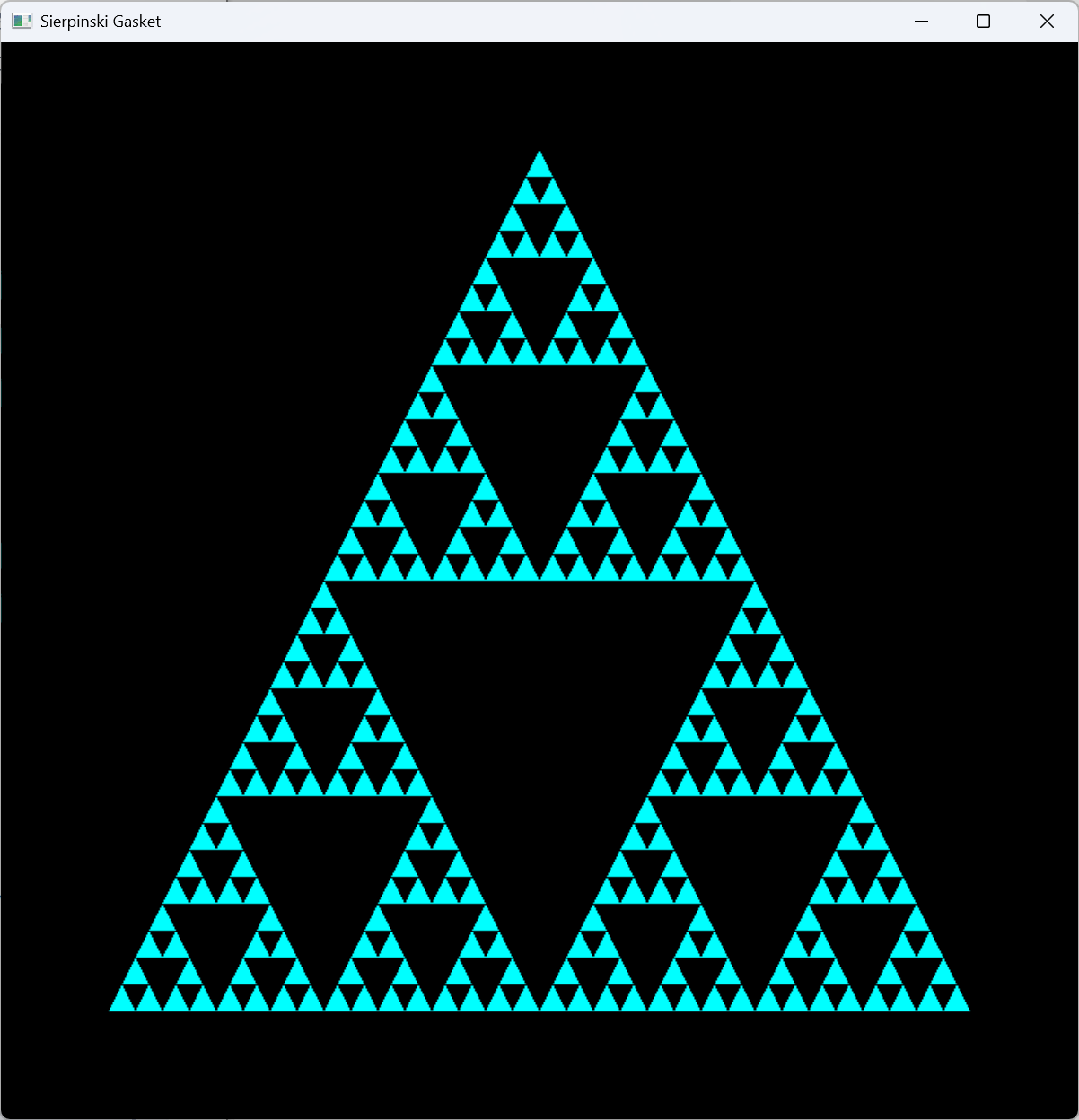
阅读课本教程和代码，尝试自主实现程序，要点在于递归绘制

关键代码截图：





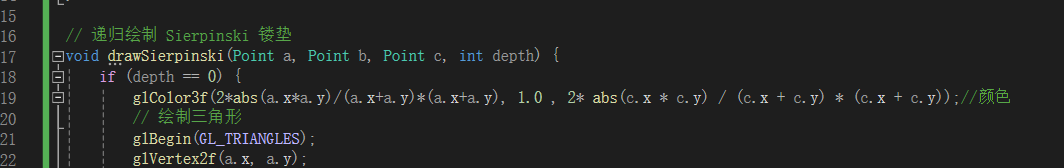
运行结果截图：



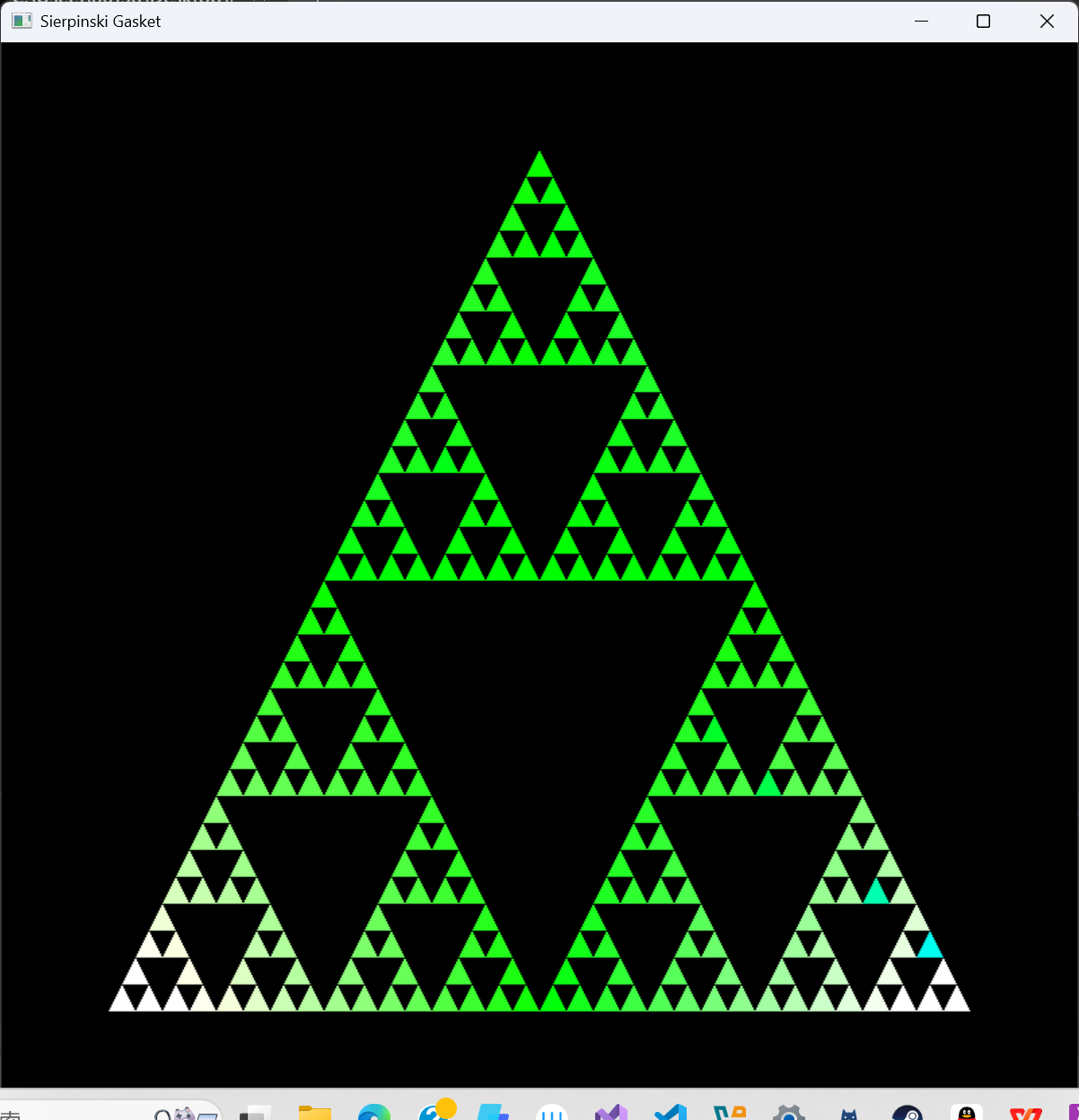
## 2.为不同三角形设置不同的颜色，使得看起来颜色更加美观

操作流程：可以在绘制函数里把三角形的颜色设置成与a、b、c的坐标相关联的函数，观感就会变化

关键代码截图：



运行结果截图：



## 为这个镂垫生成动画，①镂垫的颜色随时间不断变化。②在①的基础上增加旋转效果。③在②的基础上增加缩放效果。

操作流程：

用rotationAngle控制镂垫的旋转角度。scaleFactor控制镂垫的缩放比例。colorOffset控制颜色的变化。colorChangeSpeed、rotationSpeed 和 scaleSpeed分别控制颜色变化、旋转和缩放的速度。scaleRange控制缩放的范围。

对display 函数的修改：

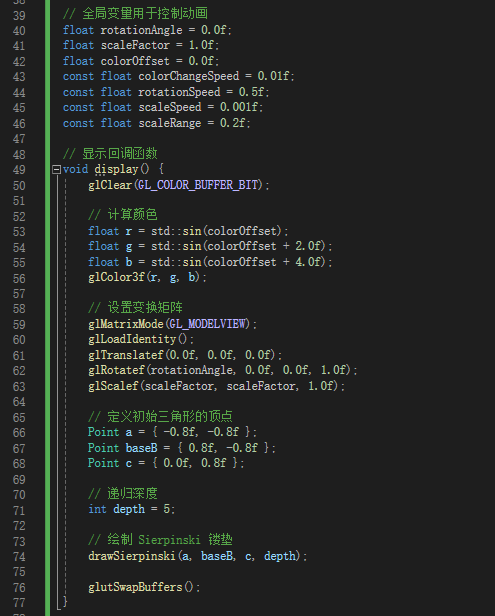
计算当前的颜色，使用正弦函数使颜色随时间平滑变化。

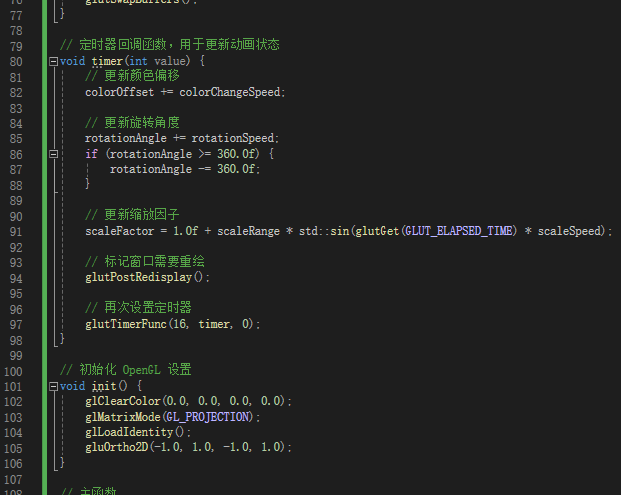
设置变换矩阵，包括平移、旋转和缩放。

设置timer 函数：

更新颜色偏移、旋转角度和缩放因子。标记窗口需要重绘。再次设置定时器，以实现连续的动画效果。

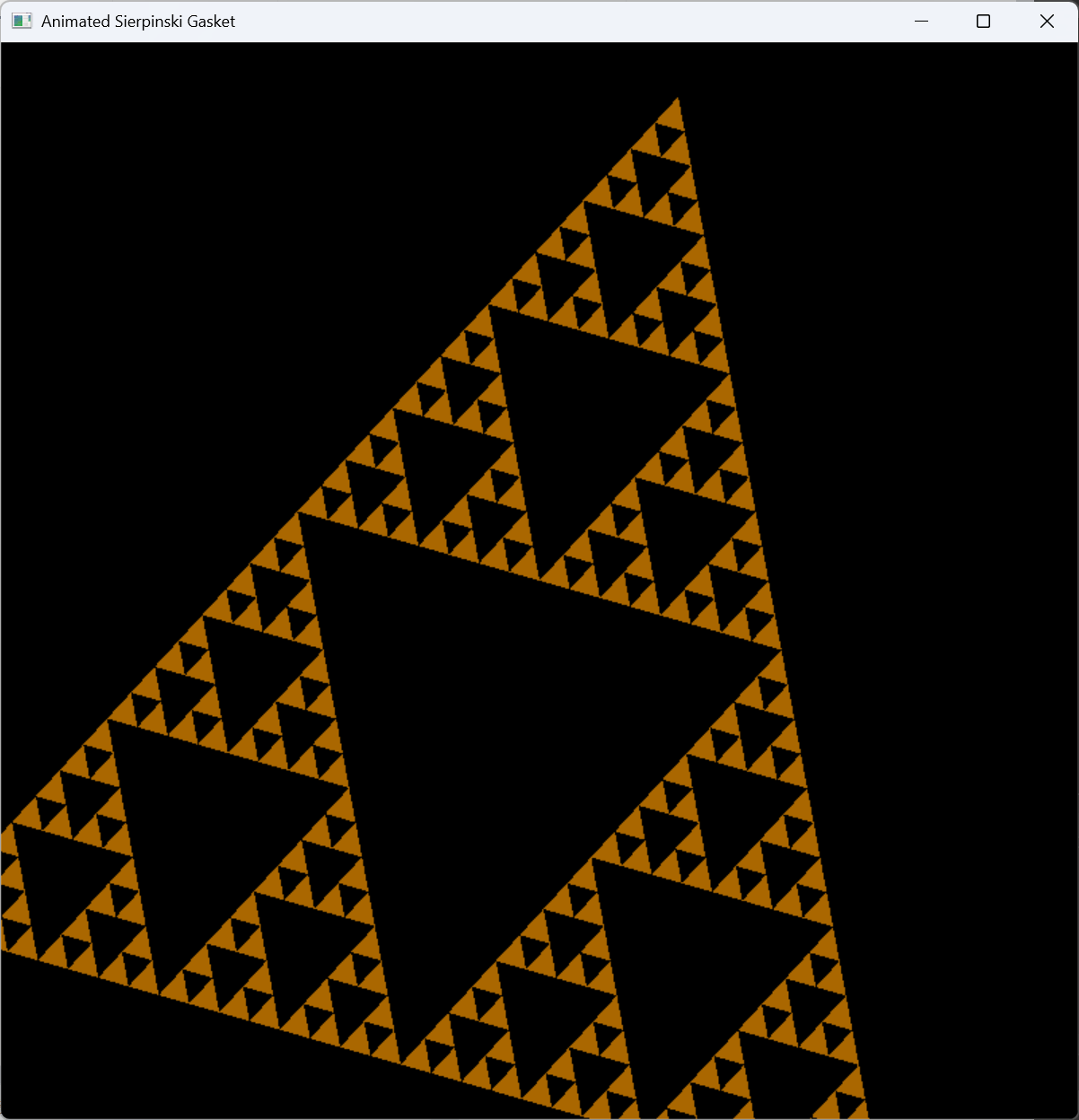
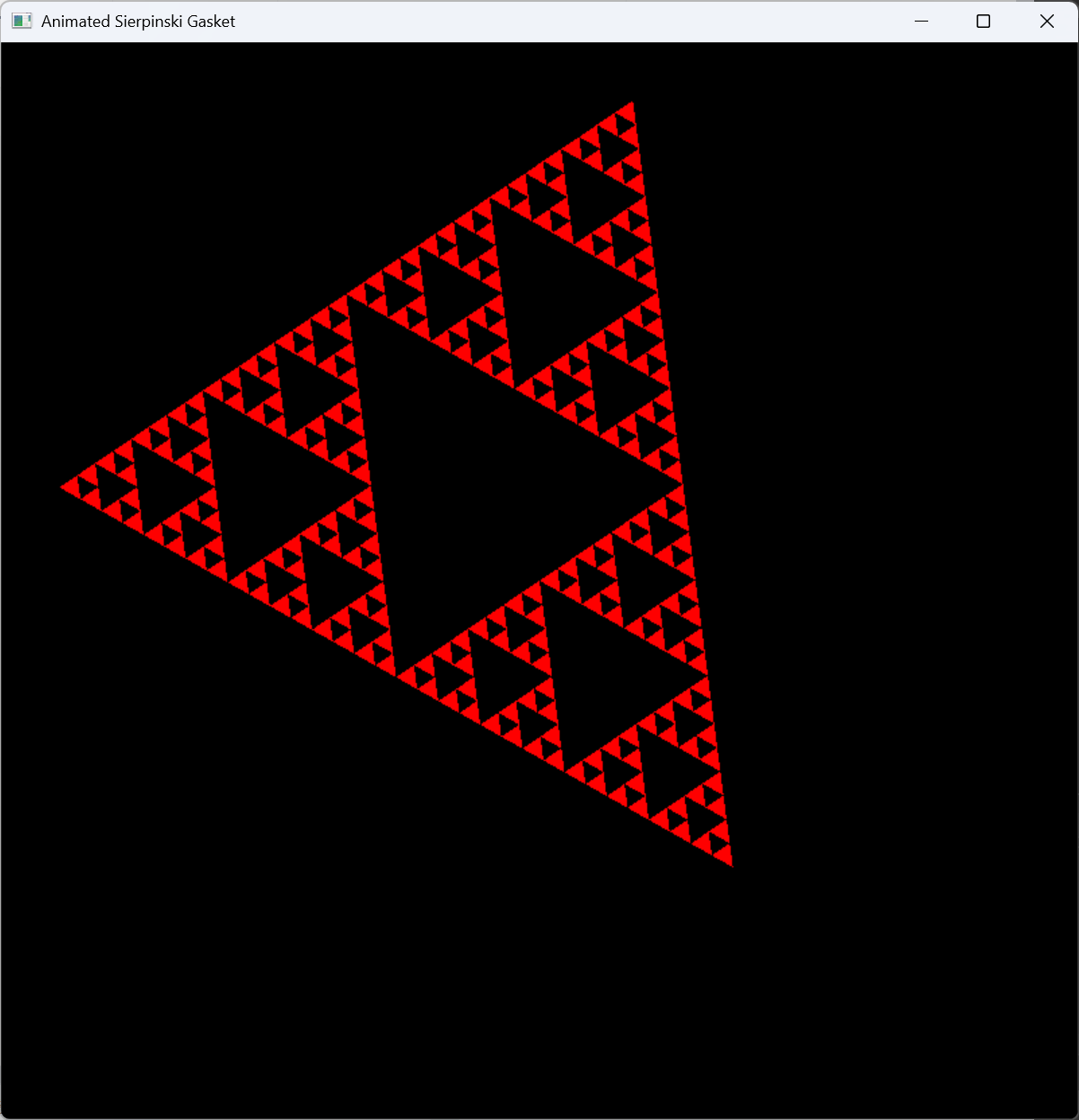
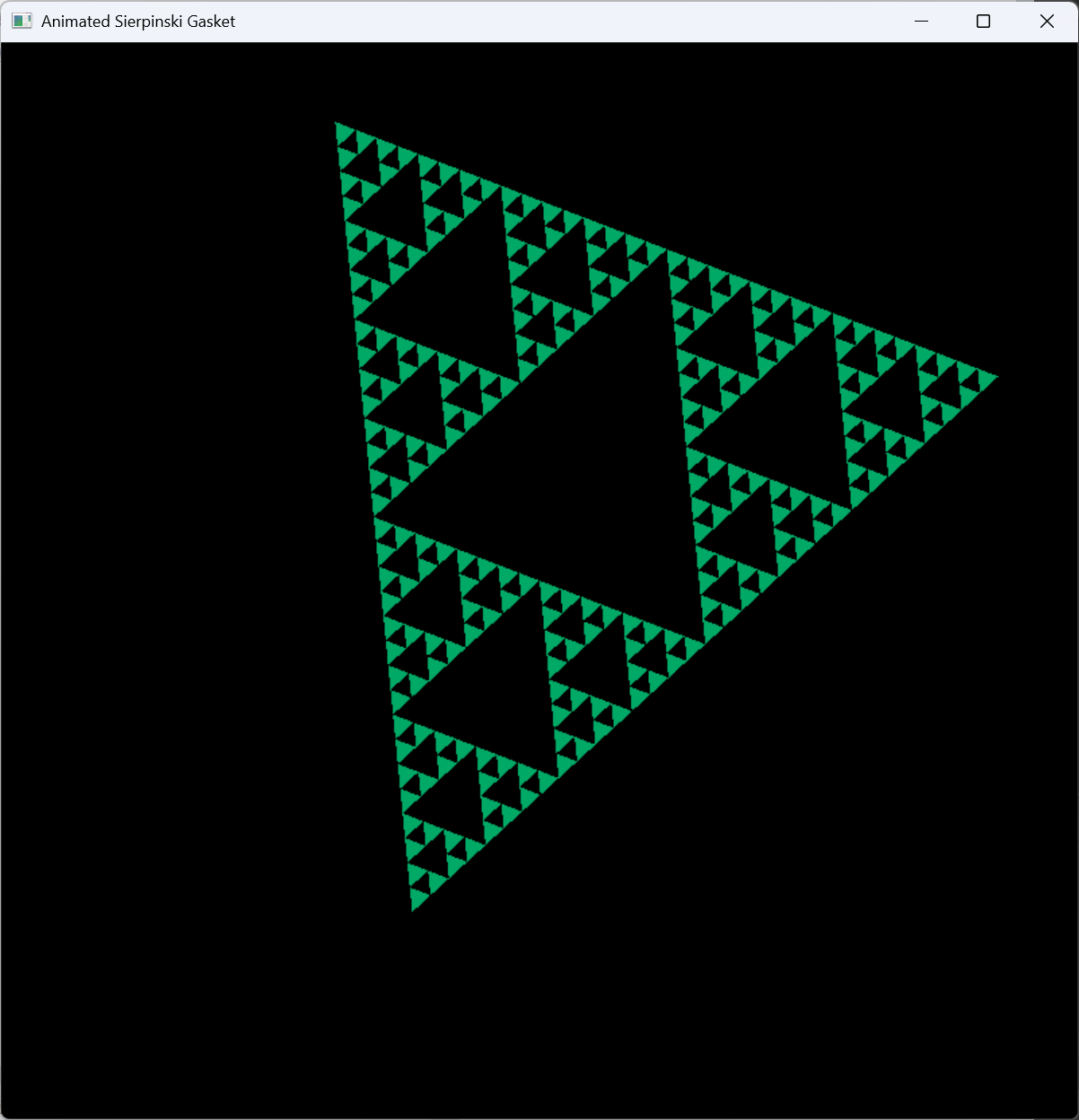
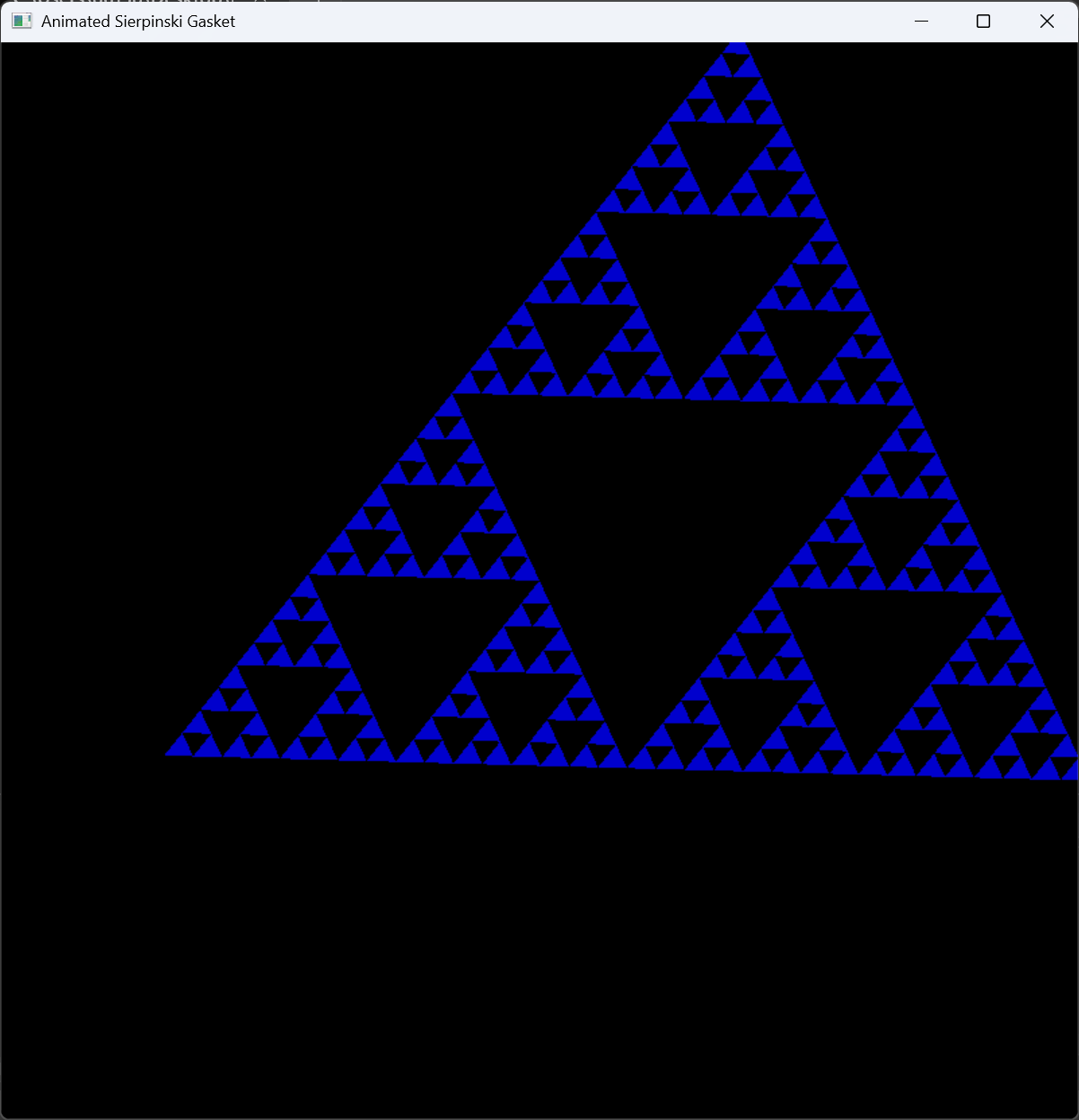
关键代码截图：





运行结果截图：

颜色变化、旋转和缩放效果都已实现。



# Task2: 完成一个三维的程序

## 1.projection

(1)运行提供的程序 projection/projection.exe



(2)调整glOrtho、 gluLookAt等函数的参数，了解各参数意义。

1)GluPerspective

fovy参数表示视角的缩放、aspect参数表示x方向上的伸缩变换、zNear是较近的投影面、zFar是较远的投影面

2)gluLookAt

对于eye部分，fovy参数表示水平方向移动视角、aspect参数表示竖直方向移动视角，并有缩放效果、zNear参数表示物体与投影面的距离

对于center，fovy参数表示水平方向移动视角、aspect参数表示竖直方向移动视角，不带有缩放效果、zNear参数表示视线e的方向

对于up，fovy参数和aspect参数表示旋转视角。

思考题：

1. OpenGL中，三维空间的坐标系是怎么样的？

答：采用右手坐标系，比如X轴指向屏幕右侧，Y轴指向屏幕上方，Z轴指向屏幕外方向。

1. OpenGL中，相机的方位是怎么样的？如何调整相机朝向呢？

答：相机的位置表示相机所在的位置，观察点表示相机观察的目标点，上方向表示相机视角的上方向。要调整相机的朝向，可以使用gluLookAt()函数。这个函数用来设置相机的位置、观察点和上方向，从而确定相机的方位。

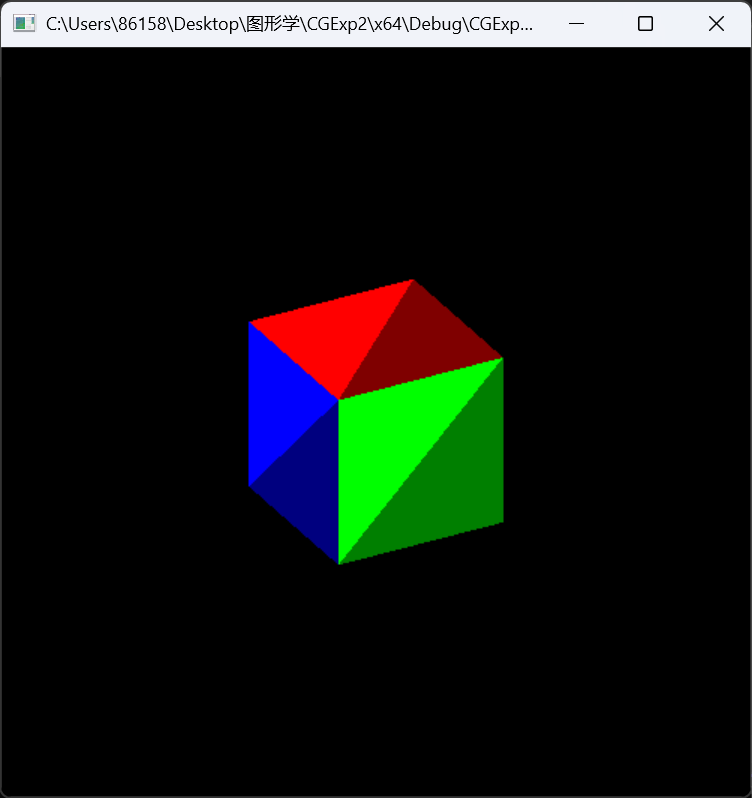
c. OpenGL中，相机可见范围是如何设置的？

答：通过设置投影矩阵实现。OpenGL提供了两种主要的投影方式：正交投影和透视投影。

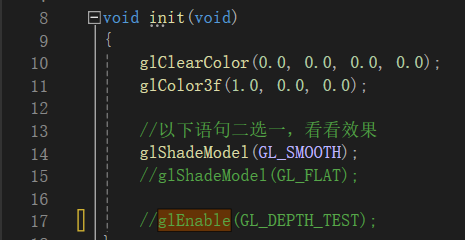
## 2.exp2-2-1

1.运行提供的示例程序(exp2-2-1.cpp)，对程序进行改写：

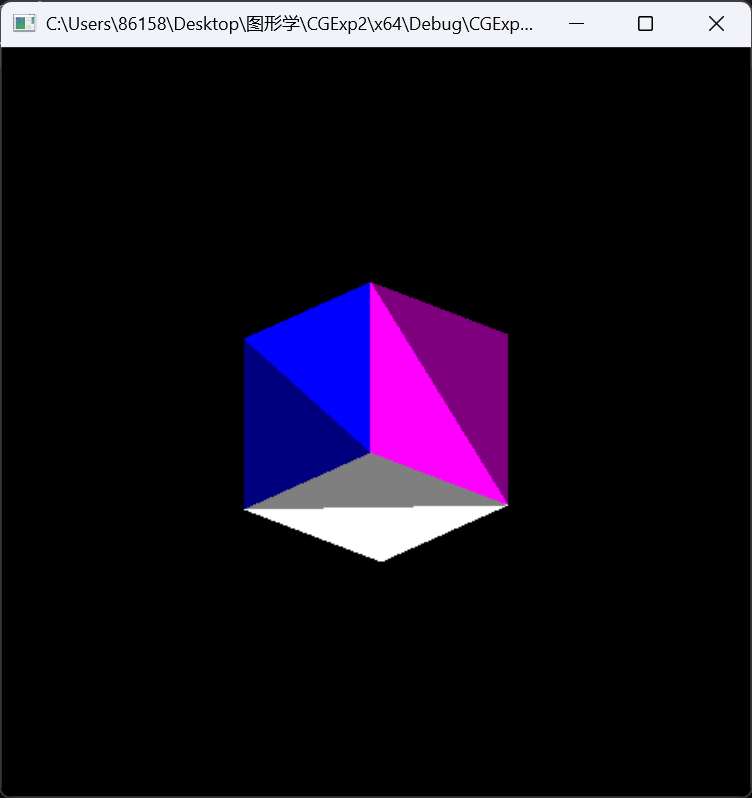
运行结果如下：

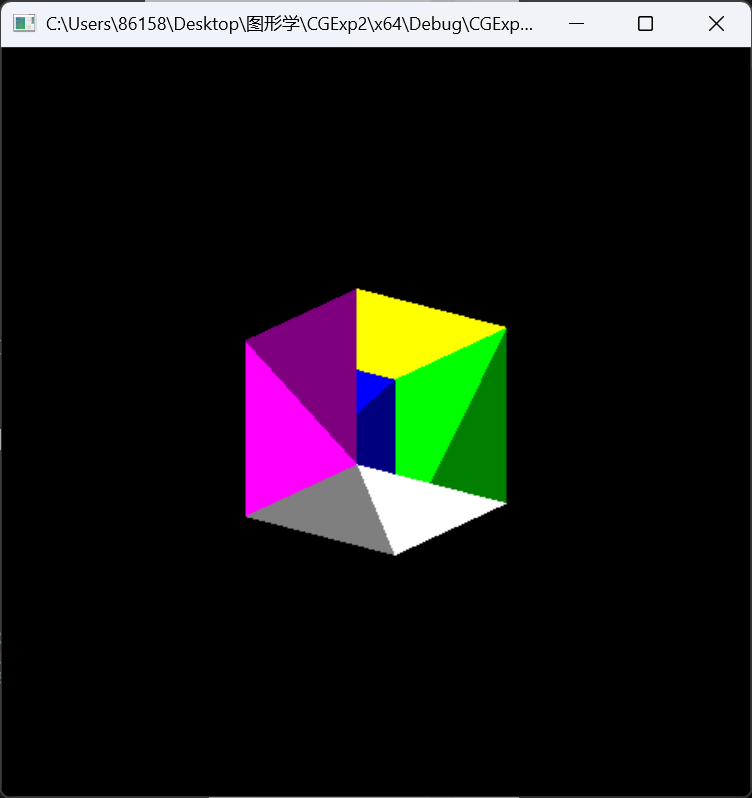


1. 比较开启/不开启深度缓冲区 glEnable(GL\_DEPTH\_TEST) 的效果;理解深度缓冲区的作用、用法



注释掉开启深度缓冲区的语句，观察生成的图像效果区别如下：





可以看到，正方体各个面的遮挡关系出错了，没办法正确展现。

原理分析：不开启深度缓冲区时，OpenGL 不会对物体之间的遮挡关系进行处理。在绘制物体时，后绘制的物体将会覆盖先绘制的物体，而不管它们在三维空间中的实际深度关系。这可能会导致在渲染复杂的三维场景时，出现不真实的效果，例如远处的物体覆盖了近处的物体。

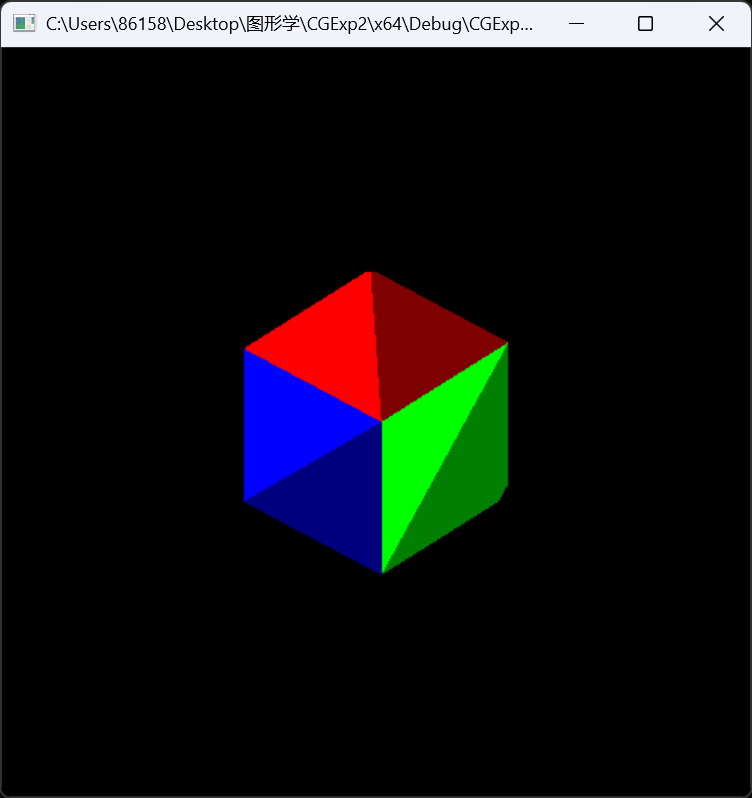
1. 让正方体自行旋转，而不是相机旋转

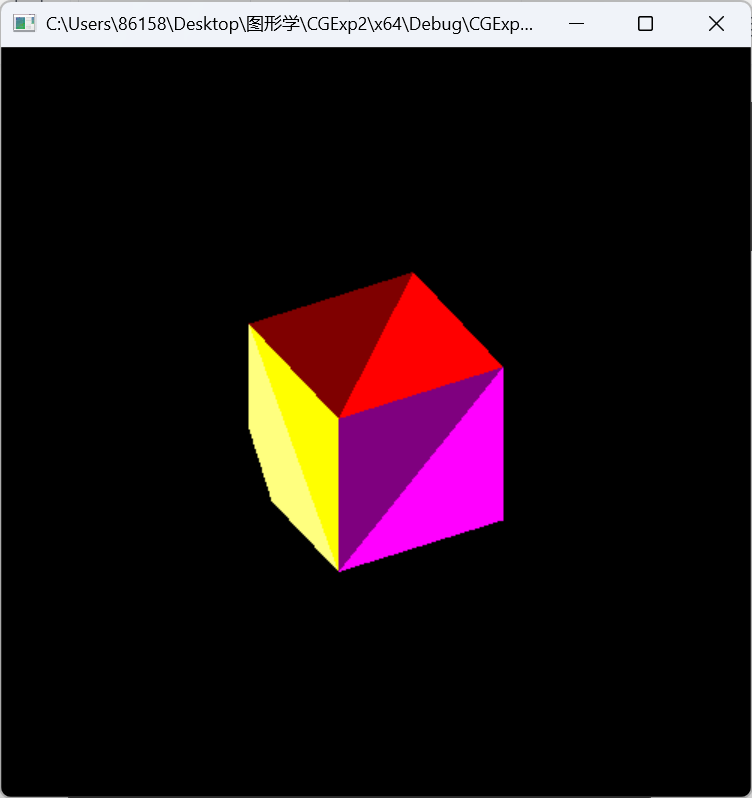
步骤流程：固定摄像机位置（gluLookAt保持不动），在display函数里添加旋转变换，保留 idlefunc() 中更新角度的逻辑，但作用在物体上。

关键代码：



运行结果：





1. 在正方体自行旋转的前提下，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）

实验流程：

（1）键盘控制相机位置

新增变量记录相机位置（camX, camY, camZ）和朝向角度（pitch, yaw）。在 keyboard() 函数中根据用户按键更新相机的位置：

W/S：沿着相机当前前方方向前进/后退。

A/D：向相机右侧/左侧平移。

Q/E：向上/向下移动。

利用三角函数从 yaw 和 pitch 计算方向向量，实现移动的方向感知。

（2）鼠标控制相机角度

使用 glutPassiveMotionFunc() 监听鼠标移动。

记录鼠标的变化量（x, y），并将其转换为相机的朝向角度变化（yaw：水平旋转，pitch：上下抬头低头）。

再次用三角函数，将 yaw 和 pitch 转换成视线方向向量，传递给 gluLookAt()，从而改变观察角度。

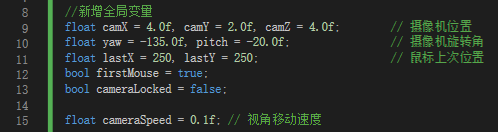
（3）实现 L 键锁定视角功能

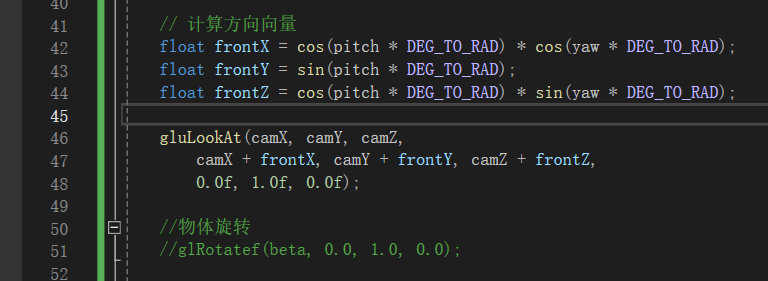
添加一个布尔变量 cameraLocked，控制是否响应用户输入。

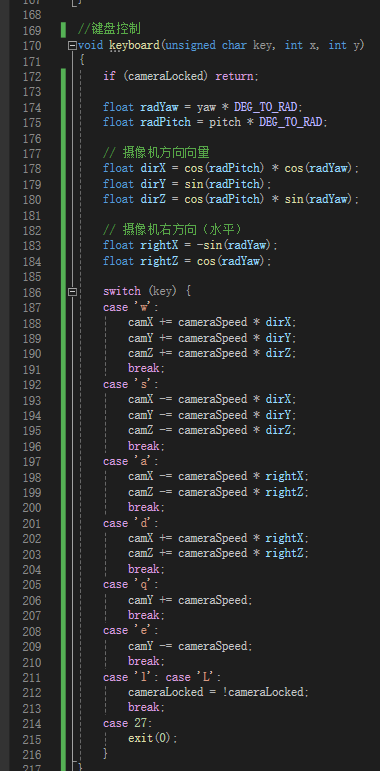
在键盘响应和鼠标响应函数中，如果锁定，就直接返回，不处理输入。

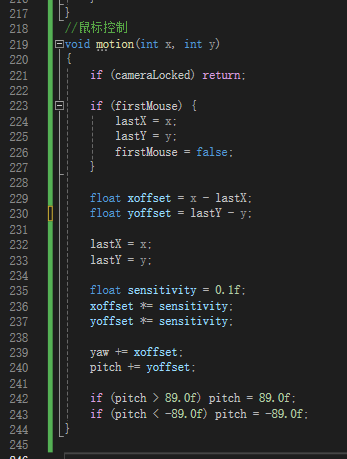
按下 L 键切换这个状态。

关键代码：

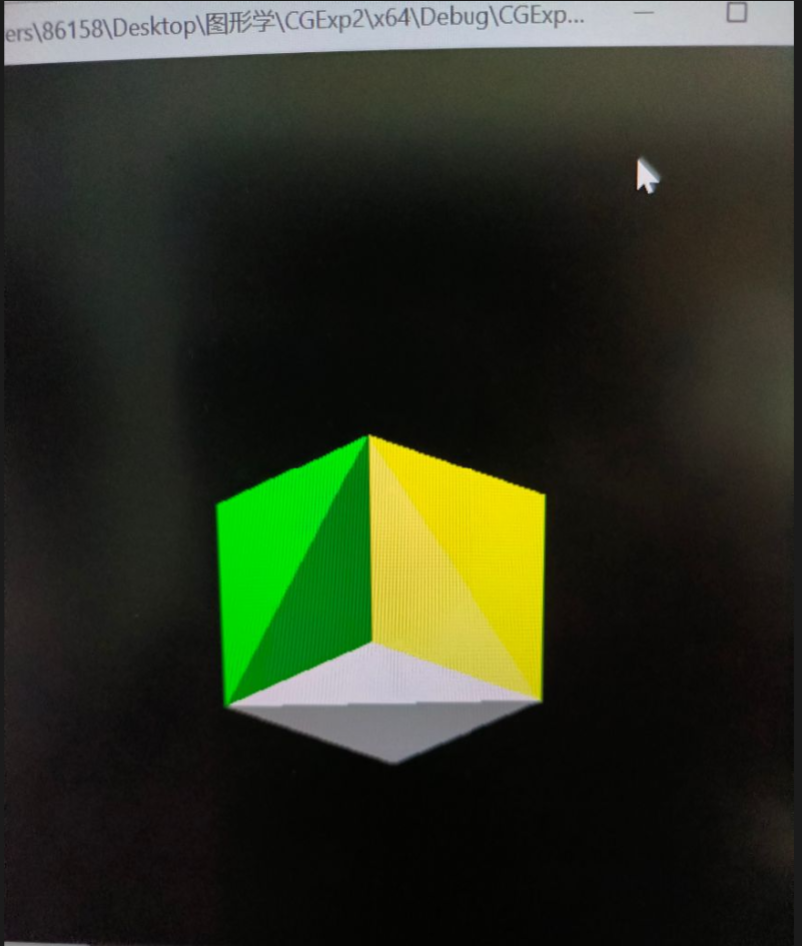
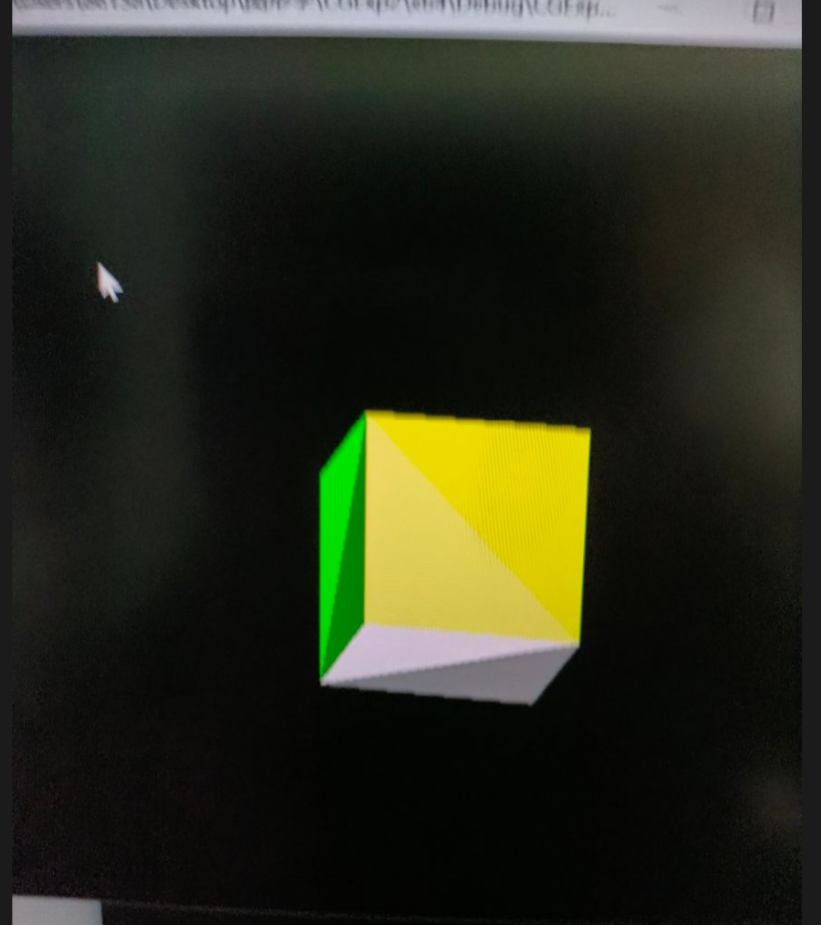




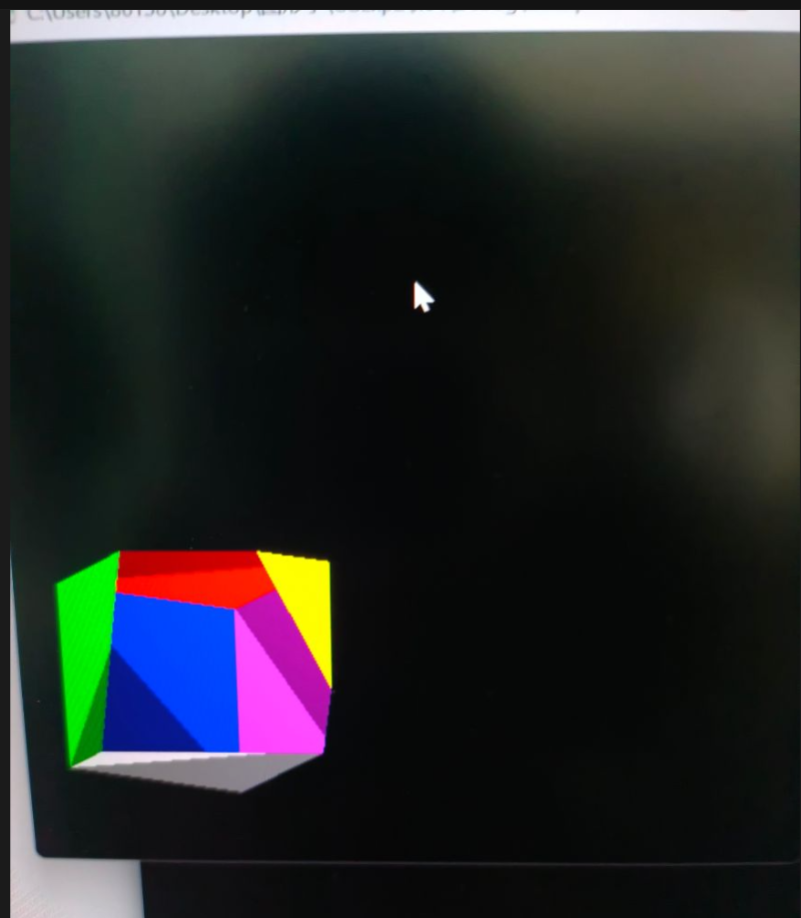
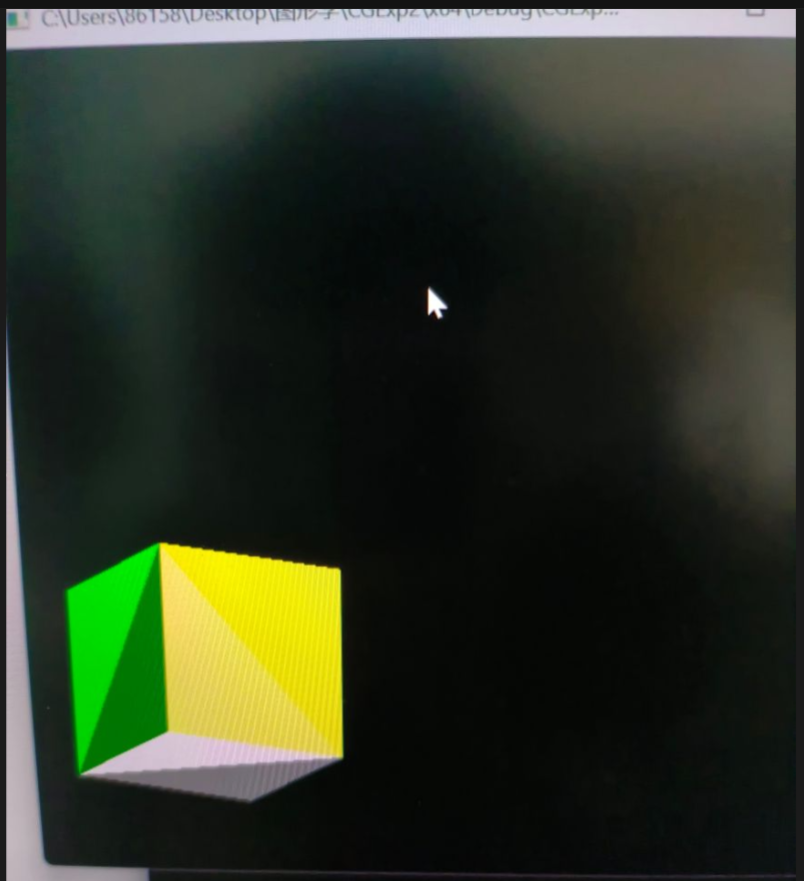




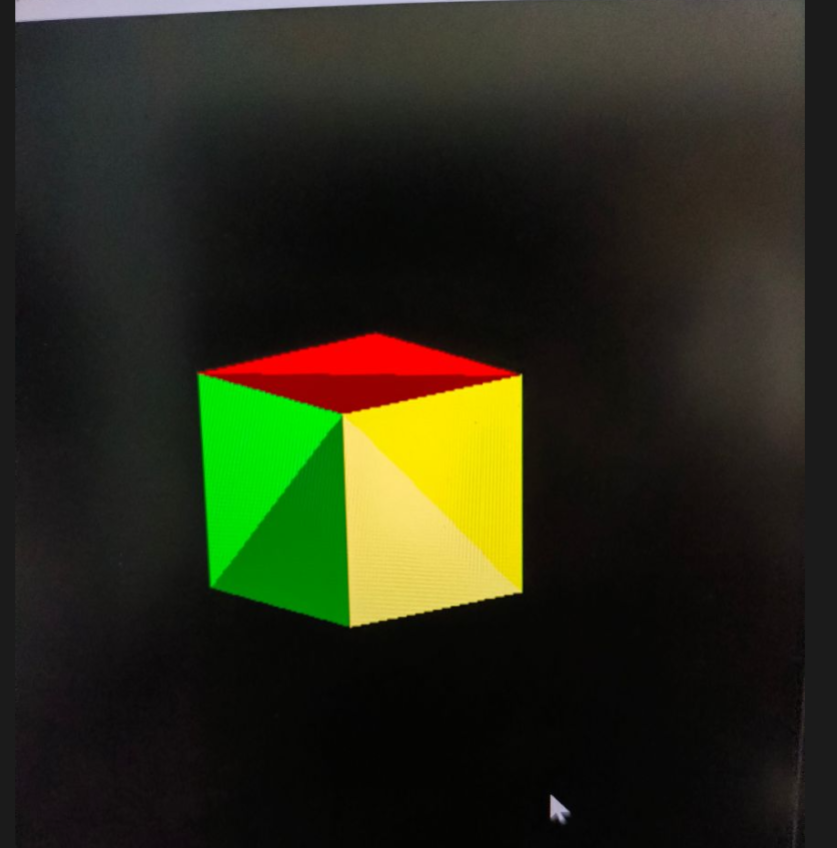
运行结果：



鼠标光标移动旋转视角。



键盘控制移动视角位置（三个维度），按L锁定视角。



思考题：如何实现前后面的遮挡？

答：

当启用深度测试并清除深度缓冲区后，OpenGL 在绘制每个像素时会进行如下操作：

计算当前像素对应的物体的深度值；将该深度值与深度缓冲区中该像素位置的深度值进行比较；根据比较结果和深度测试函数的设置，决定是否更新该像素的颜色和深度值。

通过这种方式，就能保证在渲染场景时，离相机近的物体能够遮挡离相机远的物体，从而实现前后面的遮挡效果。

# Task3: 绘制一个线框球体

## 1.情况1

操作流程：

1、在不使用glut对象的情况下，绘制一个线框球体（即仅仅使用基本图元进行绘制，不能使用glutWireSphere 绘制对象函数），

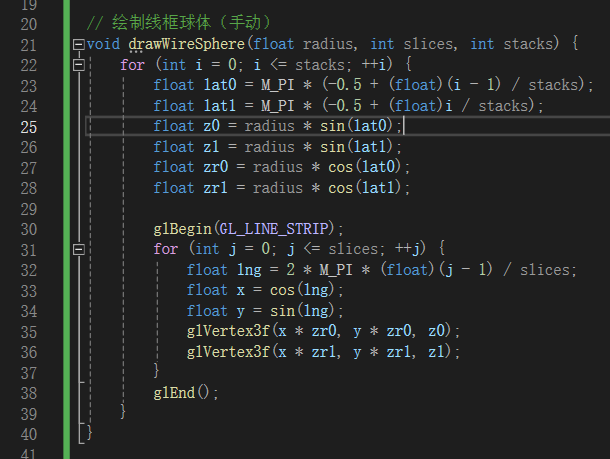
2、添加动画效果，让球体绕圆心旋转

3、添加reshape回调函数，使其保持比例，不会随着窗口大小的改变而产生形变。

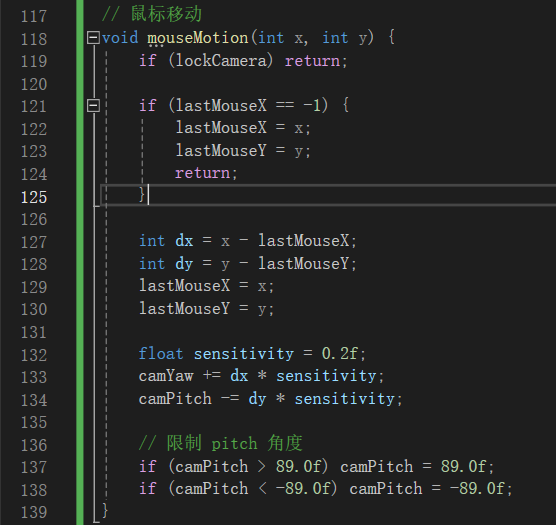
4、添加交互式的相机控制（同task2）

关键代码截图：

用两组圆环（纬线 + 经线）近似一个线框球体

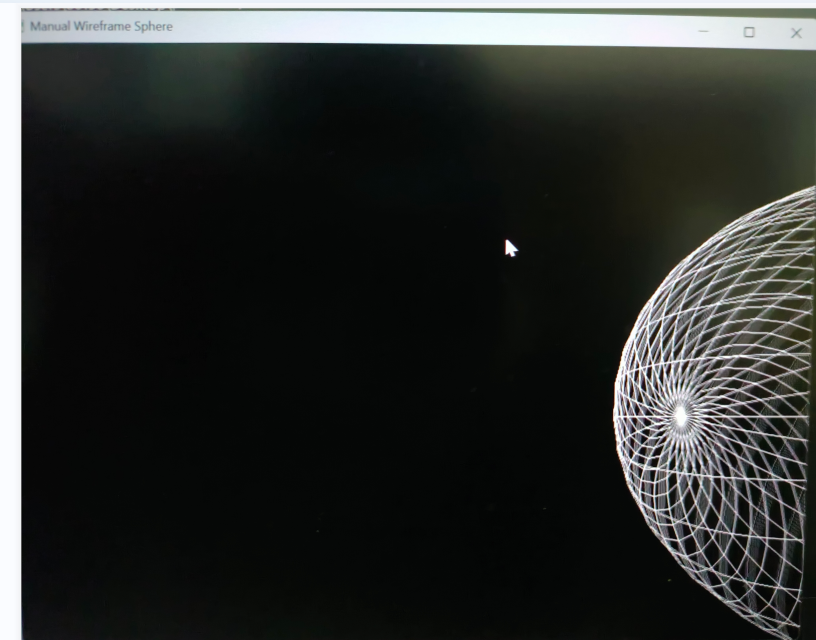


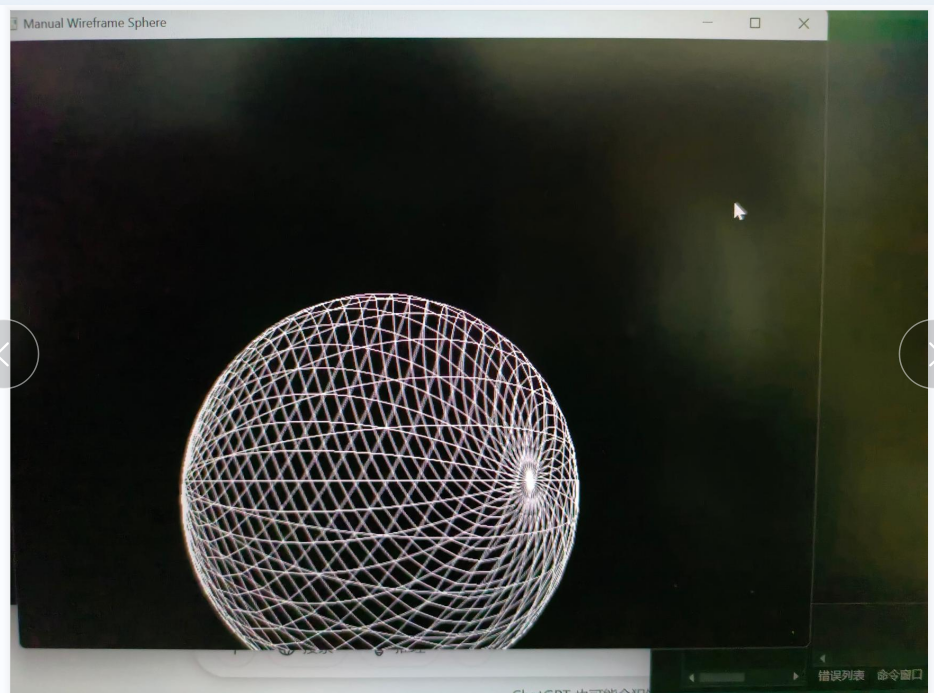


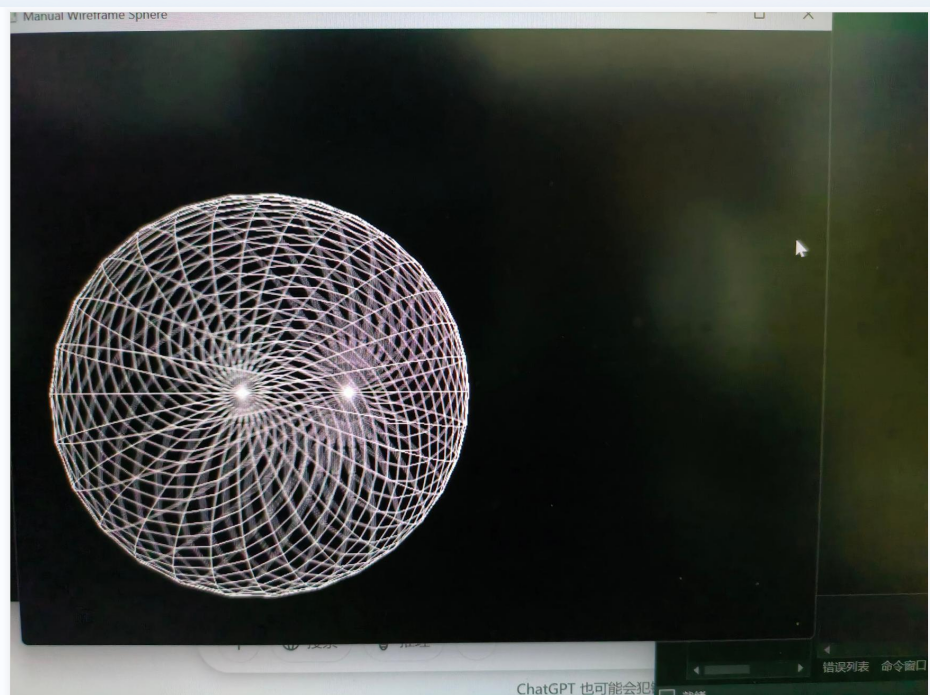


运行结果：

鼠标控制相机角度以及球体自主旋转：







键盘控制相机位置：

