

计算机图形学实验

实验3 三维模型显示

姓 名：\_\_\_\_\_潘宇科\_\_\_\_\_\_\_\_\_\_

学 号：\_\_\_\_37320222204170 \_

学 院：\_\_\_ 信息学院\_\_\_\_ \_\_\_\_

专 业：\_\_\_\_软件工程\_\_ \_\_\_\_\_

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

2024年 5 月 5 日

目录

[Task1/2：ply小人 3](#_Toc97636928)

[1.Task1:设置模型颜色 3](#_Toc97636929)

[2.Task2:设置光照 3](#_Toc97636930)

[Task3: 旋转立方体 3](#_Toc97636931)

[1.旋转立方体的修改 3](#_Toc97636932)

[2.附加题 3](#_Toc97636933)

# Task1/2：ply小人

## 1.Task1：设置模型颜色

**学习ply文件格式，读入文件内容，将模型显示出来，要求将模型设置为纯色，并显示出来，类似下图效果（颜色不限）**

**操作流程：**

使用所提供的源cpp代码完成ply文件的读取与显示，得到模型的展示，然后通过对DrawVertices函数进行修改以设置模型的颜色：

在 DrawVertices 函数中设置颜色： 我们需要在 DrawVertices 函数中找到绘制每个三角形的地方，并在绘制之前设置颜色。在代码中，我们在绘制每个三角形之前调用 glColor3f 函数来设置颜色。

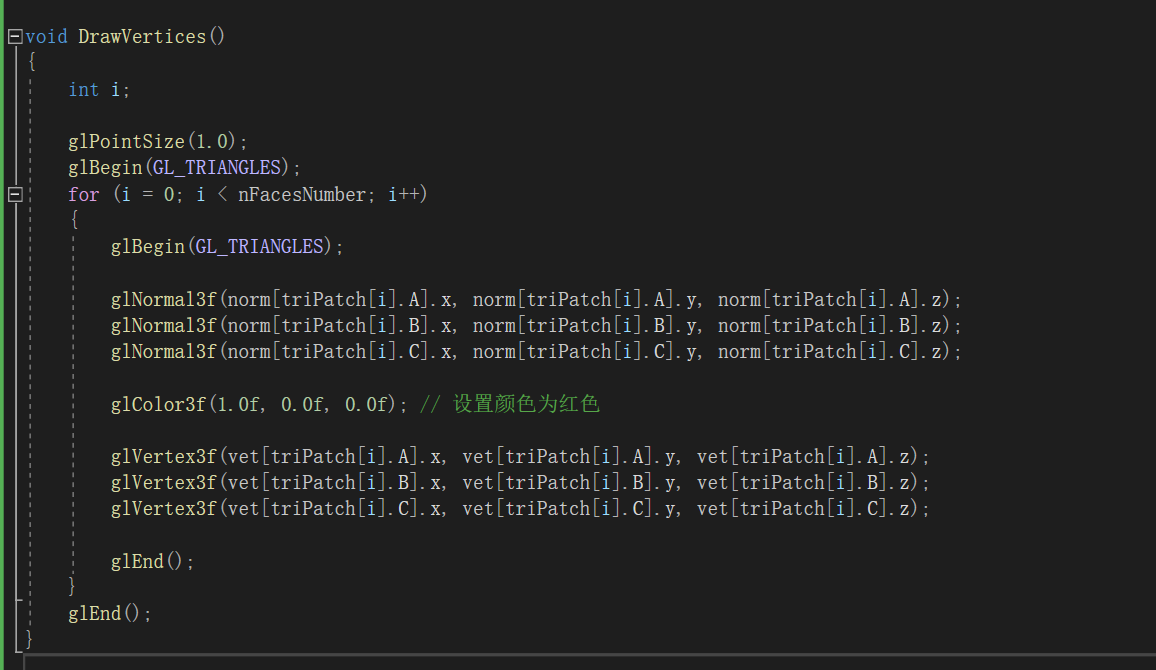
设置颜色为红色： 我们将颜色设置为红色，即 (1.0f, 0.0f, 0.0f)。

绘制三角形： 在设置完颜色后，我们绘制当前三角形的三个顶点。

重复直到所有三角形都绘制完毕： 我们重复这个过程，直到所有的三角形都被绘制完毕。

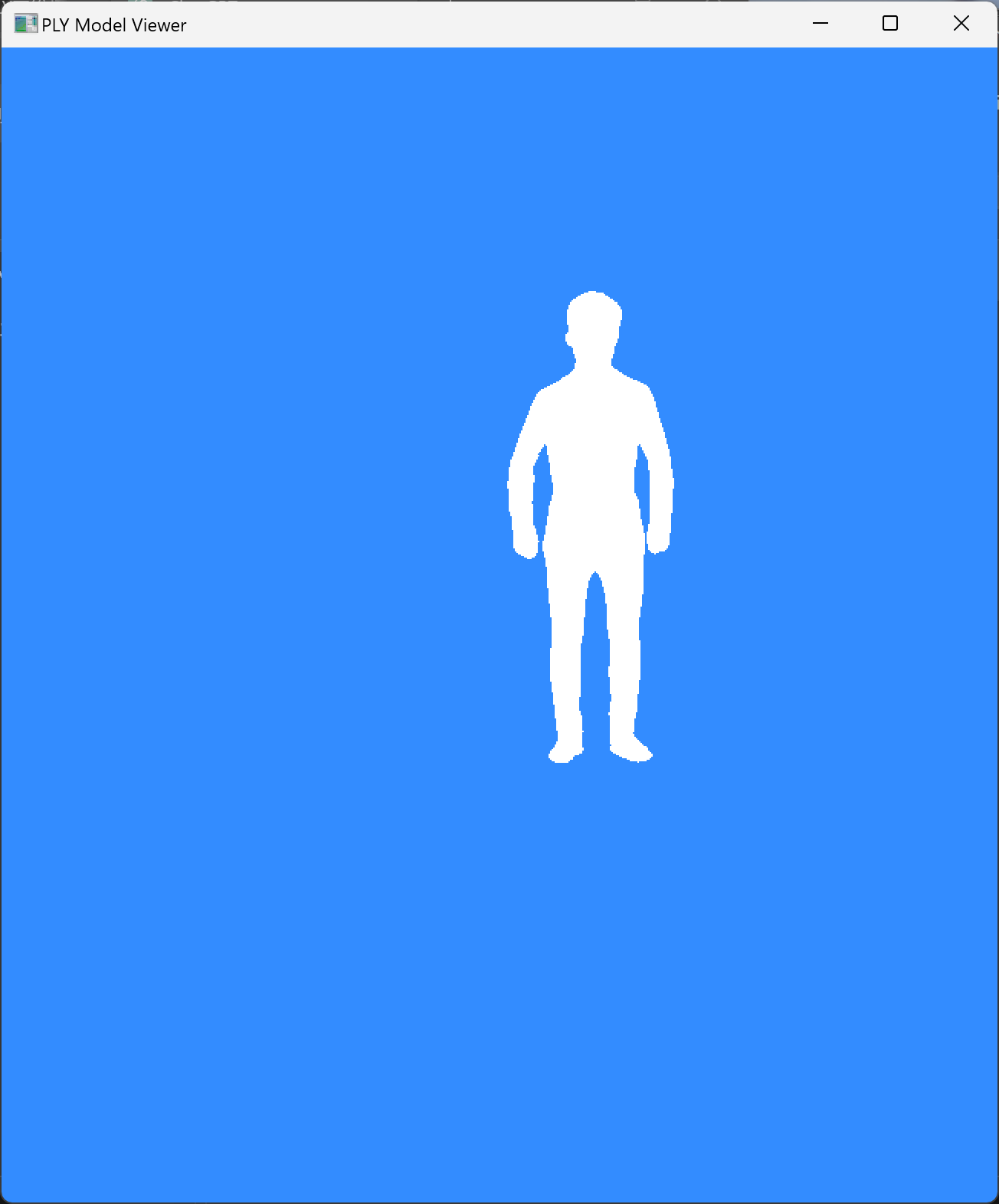
**关键代码截图：**

设置模型为红色后的函数

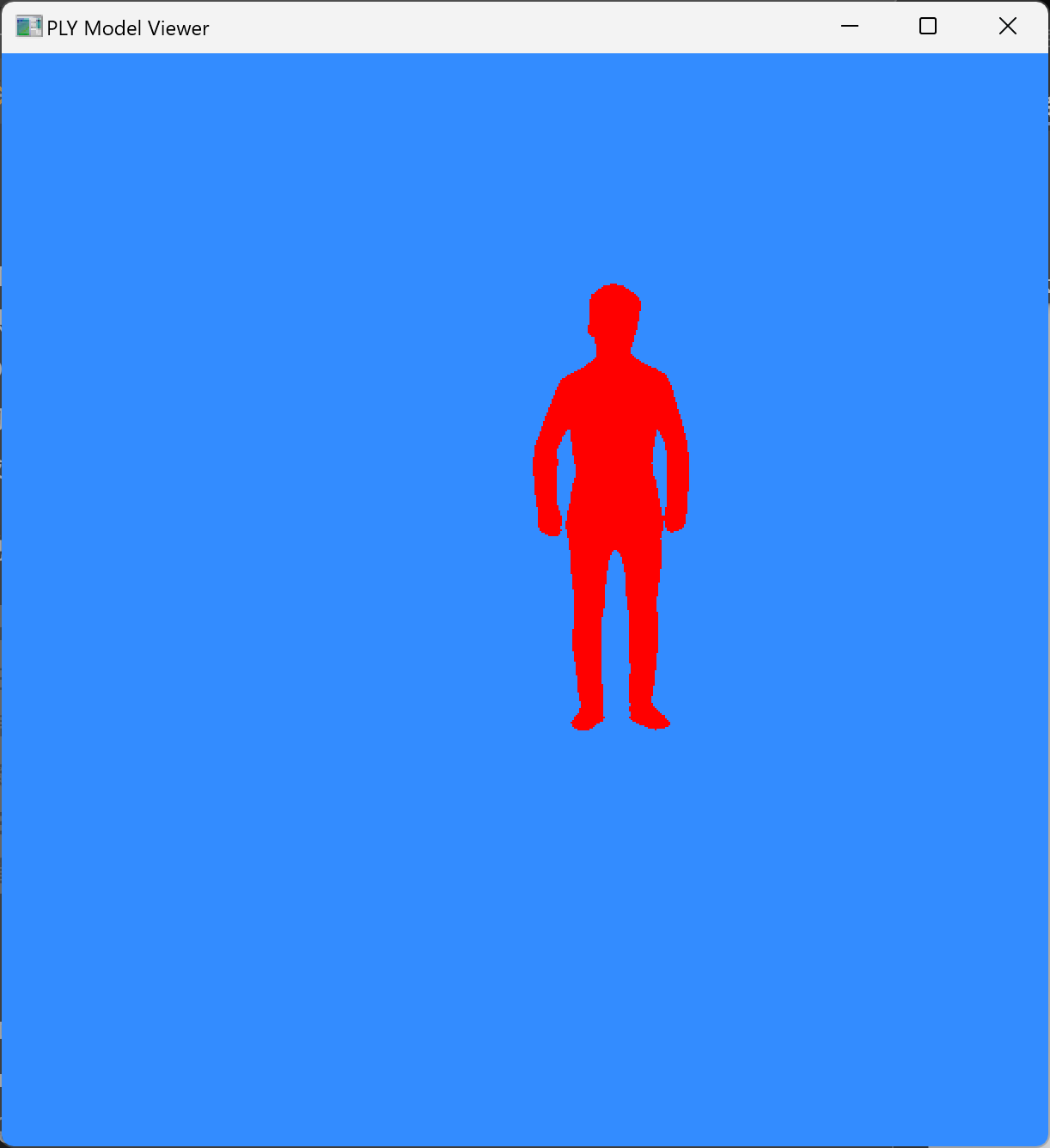


**运行结果截图**：

使用源代码直接对ply文件进行读取展示



设置颜色后



## 2.Task2：设置光照

**设置光照使小人显示出明暗效果，类似下图效果。调整光源位置，使其围绕小人旋转，展示出动画效果。**

**操作流程：**

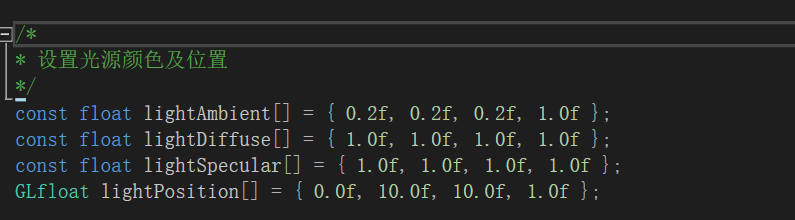
在程序初始化阶段，启用光照功能，并设置光源的位置、环境光、漫反射光、镜面反射光等属性。

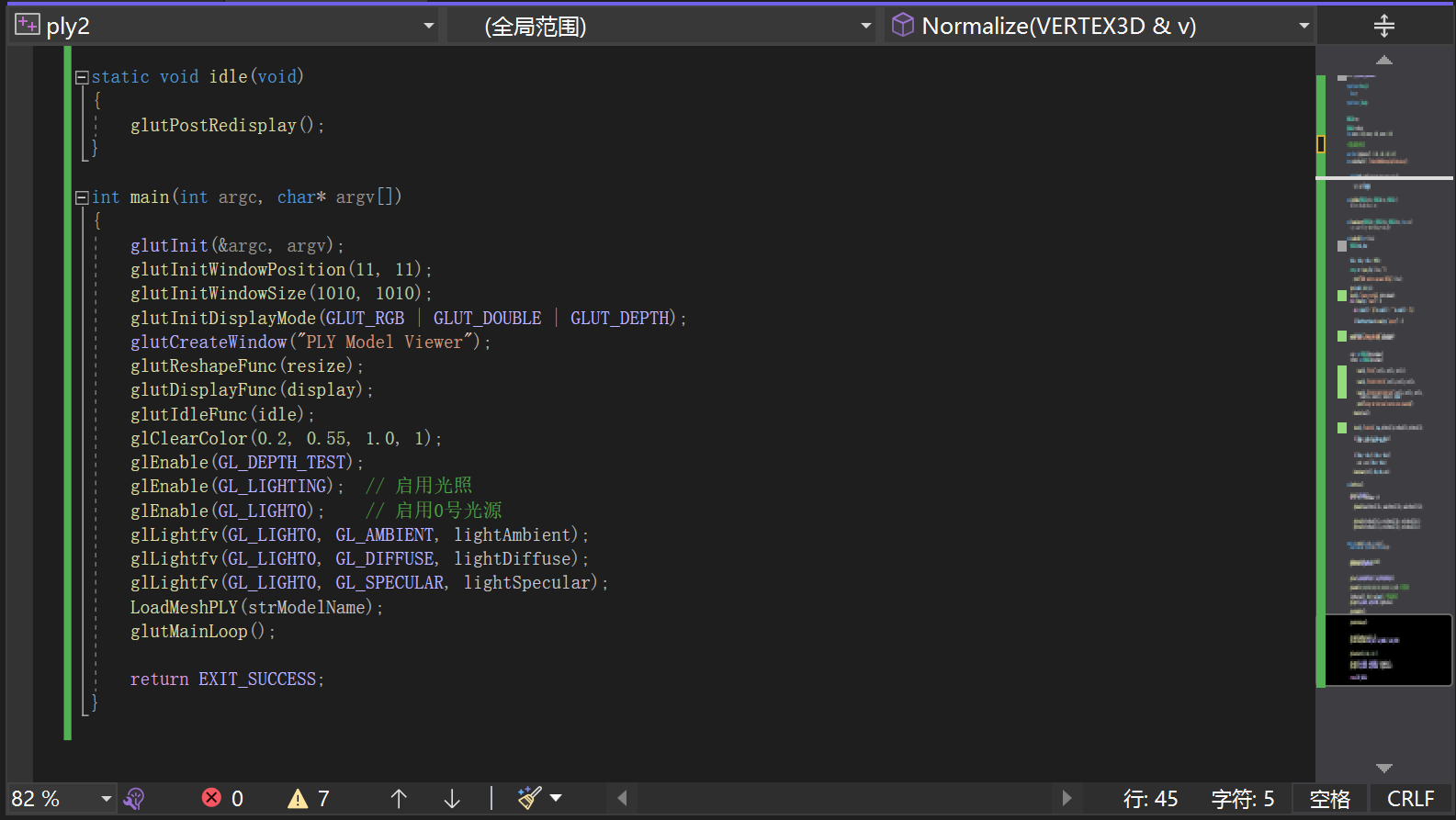
在绘制小人的过程中，对每个顶点设置法向量和颜色，以便模拟明暗效果。

在每一帧渲染时，更新光源的位置，使其围绕小人旋转。

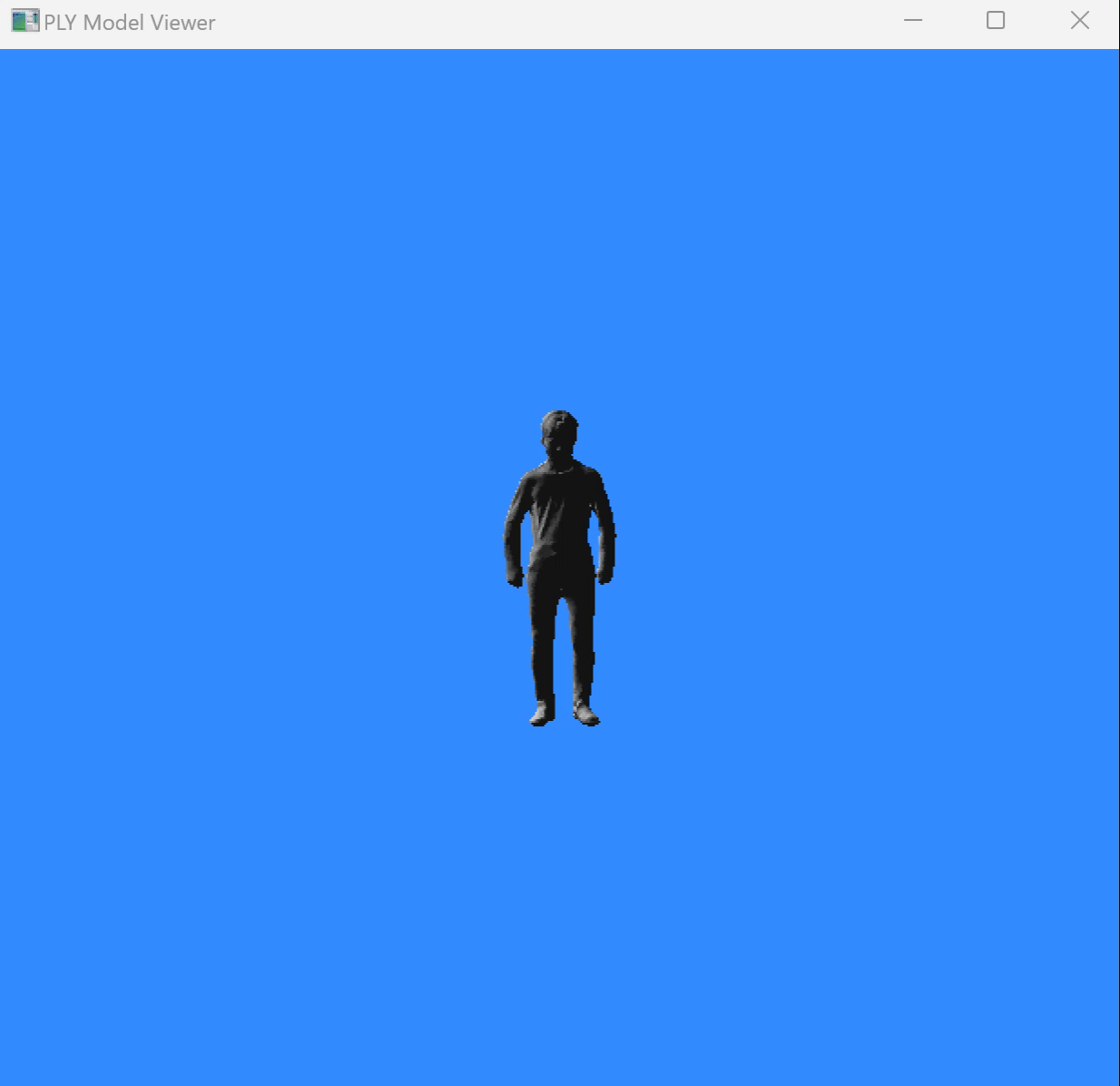
通过循环渲染每一帧，实现动画效果。

**关键代码截图：**





**运行结果截图：**



# Task2: 旋转立方体

## 1.旋转立方体的修改

**对实验2的旋转立方体程序进行修改，实现明暗计算。用键盘实现材质的修改（b为黄铜材质，n为红色塑料材质，m为白色光亮材质，具体参数设置可参考参考书）。用键盘实现光源的修改（o为白色光，p为彩色光）。**

**操作流程：**

1.初始化OpenGL环境

设置显示模式：使用 glutInitDisplayMode 设置双缓冲和颜色深度模式，以支持更平滑的动画和颜色渲染。

创建窗口：通过 glutCreateWindow 创建一个窗口。

初始化光源：配置初始光源属性，包括环境光、漫反射光、镜面反射光和光源位置。

启用深度测试：为了正确处理深度信息，开启深度测试以支持前后物体的遮挡关系。

2. 设置光源和材质

光源初始化：在初始化函数中设置一个默认的白色光源。

材质属性：设置默认的材质属性，包括颜色和光泽度，确保物体可视。

3. 处理键盘输入

响应材质更改：根据用户按键（如 b, n, m）更改材质属性，这些属性影响物体如何反射光线。

响应光源更改：通过按键（如 o, p）更改光源的颜色和性质，这影响场景的整体照明。

4. 绘制和更新立方体

模型变换：使用 glRotate, glTranslate, glScale 等函数调整物体在场景中的位置和方向。

绘制几何体：使用 glBegin 和 glEnd 绘制立方体的各个面。

动画：通过修改视角或物体的位置来创建动画效果，例如使立方体围绕某个轴旋转。

5. 渲染循环

清除屏幕和深度缓冲：每个新帧开始时，清除颜色和深度缓冲。

加载和修改矩阵：加载适当的矩阵来设置视点和投影。

调用绘制函数：通过 glutDisplayFunc 设置的函数来进行绘制。

交换缓冲：使用双缓冲模式，完成绘制后交换前后缓冲，显示新帧。

6. 用户交互和反馈

注册回调函数：对于键盘输入和窗口变化，注册回调函数以处理这些事件。

实时更新：根据用户的输入实时更新光源和材质设置，以及立方体的状态和位置。

7. 结束和清理

退出程序：当用户按下退出键（如 ESC），正确关闭OpenGL窗口和程序

添加键盘与材质和光源的交互

默认光源（白色光）

环境光：通常设置为较低的值，因为它是在整个场景中均匀分布的光。

漫反射光：模拟从特定方向上光源发出并在表面上均匀散射的光。

镜面反射光：模拟从光源发出并在物体表面反射的光，通常用于创建高光效果。

GLfloat light\_ambient[] = {0.2, 0.2, 0.2, 1.0};

GLfloat light\_diffuse[] = {1.0, 1.0, 1.0, 1.0};

GLfloat light\_specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat light\_position[] = {0.0, 0.0, 2.0, 1.0}; // 点光源

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

材质设置

材质属性决定了物体表面对光线的反应方式，包括环境反射、漫反射、镜面反射和发光系数。

黄铜材质

环境光反射：较暗的金黄色。

漫反射光反射：鲜艳的金黄色。

镜面反射光反射：较亮的黄色，模拟金属的反光。

光泽度：较高的值表示光滑的表面。

GLfloat brass\_ambient[] = {0.33f, 0.22f, 0.03f, 1.0f};

GLfloat brass\_diffuse[] = {0.78f, 0.57f, 0.11f, 1.0f};

GLfloat brass\_specular[] = {0.99f, 0.94f, 0.81f, 1.0f};

GLfloat brass\_shininess = 27.8f;

红色塑料材质

环境光反射：暗红色。

漫反射光反射：明亮的红色。

镜面反射光反射：略带白色的高光。

光泽度：较低的值，表示较粗糙的表面。

GLfloat red\_plastic\_ambient[] = {0.3f, 0.0f, 0.0f, 1.0f};

GLfloat red\_plastic\_diffuse[] = {0.6f, 0.0f, 0.0f, 1.0f};

GLfloat red\_plastic\_specular[] = {0.8f, 0.6f, 0.6f, 1.0f};

GLfloat red\_plastic\_shininess = 32.0f;

白色光亮材质

环境光反射：亮灰色。

漫反射光反射：亮白色。

镜面反射光反射：强烈的白色高光。

光泽度：非常高的值，表示非常光滑的表面。

c

Copy code

GLfloat shiny\_white\_ambient[] = {0.3f, 0.3f, 0.3f, 1.0f};

GLfloat shiny\_white\_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};

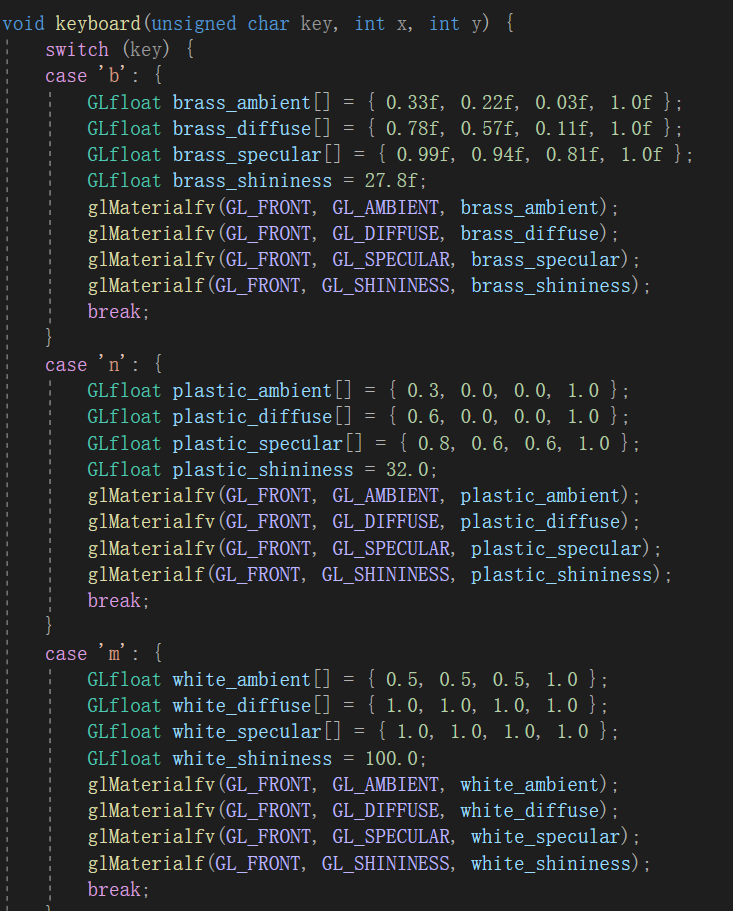
GLfloat shiny\_white\_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};

GLfloat shiny\_white\_shininess = 100.0f;

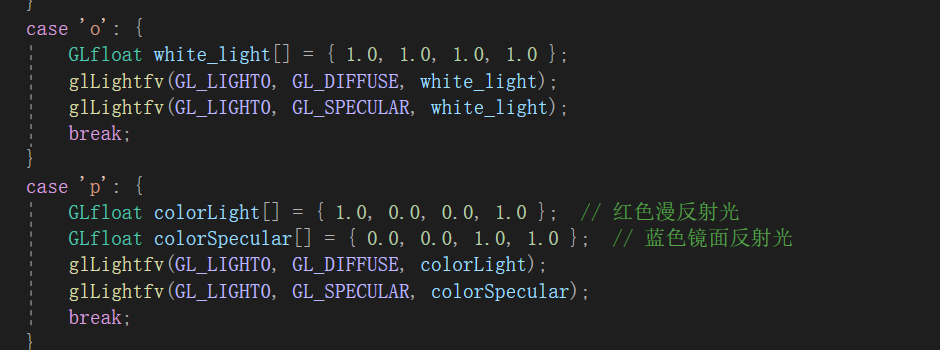
这些设置通过编写键盘事件处理函数，在运行时动态应用，从而改变场景的视觉效果。通过合理配置光源和材质参数，可以达到非常真实的视觉效果，使场景中的对象在不同光照条件下呈现不同的视觉效果。

**关键代码截图：**

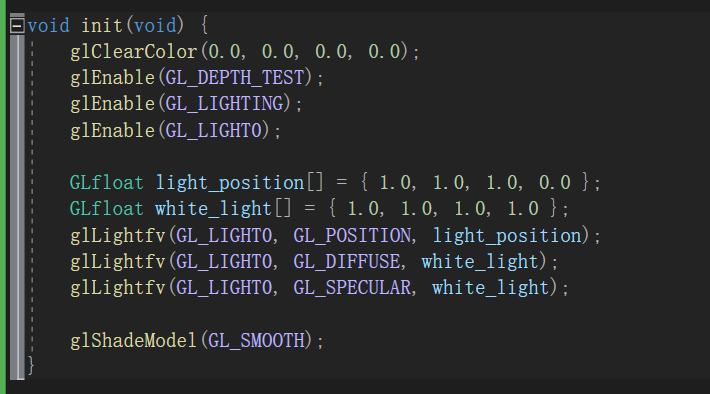
材质



光源

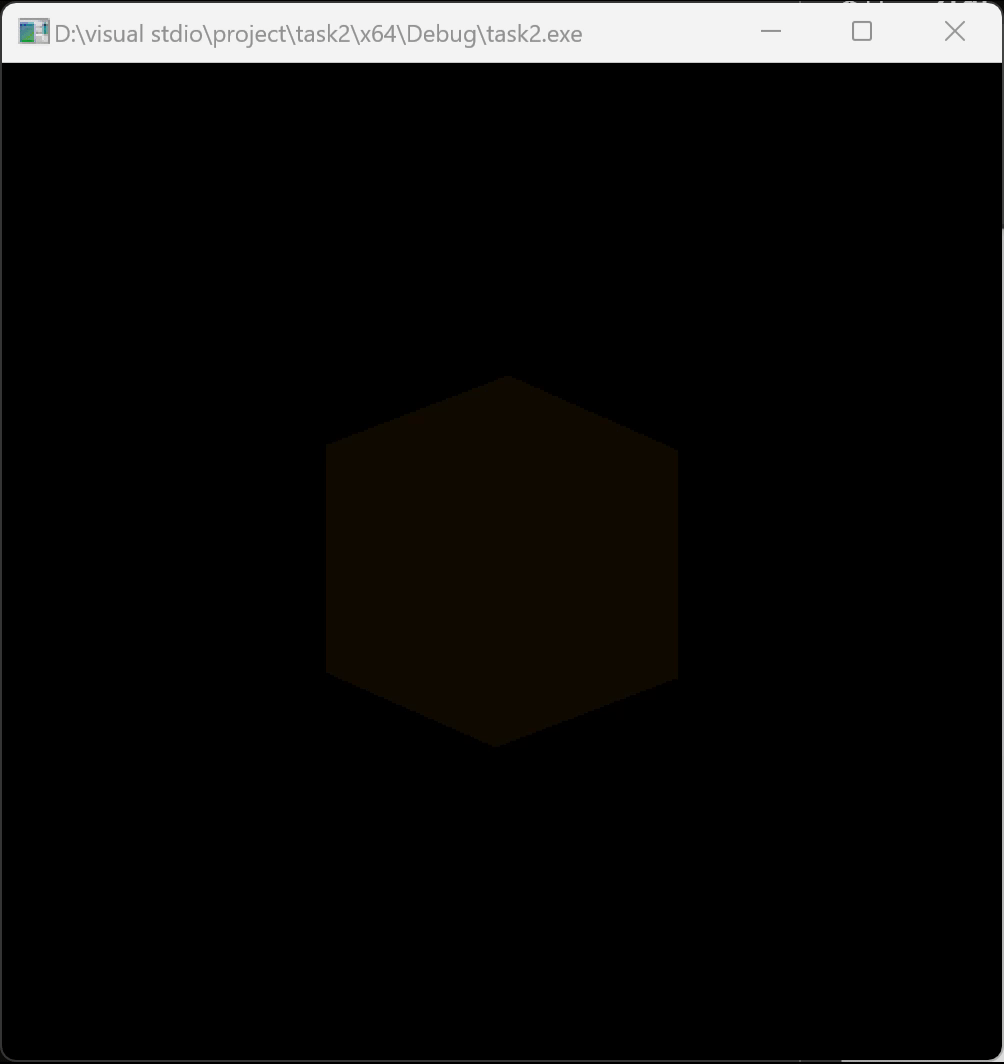


初始化：

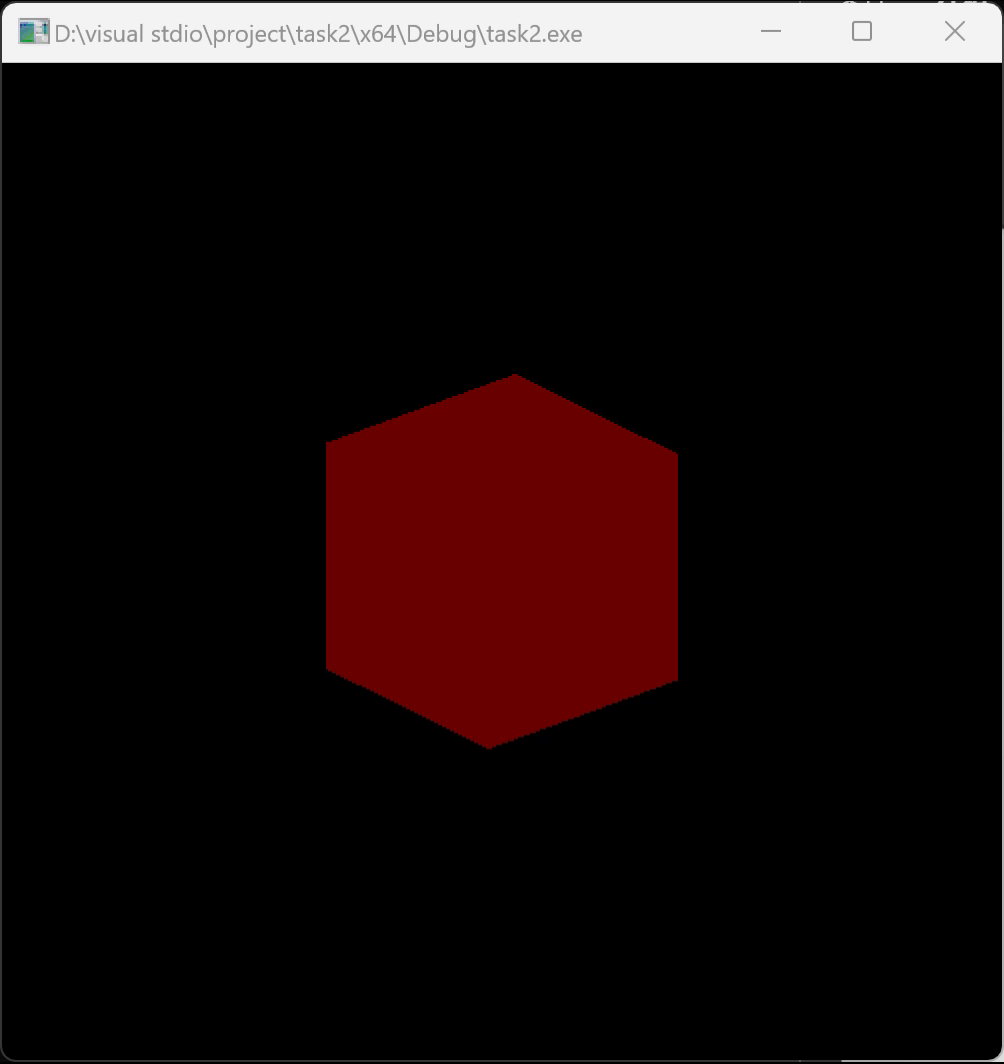


运行结果截图：

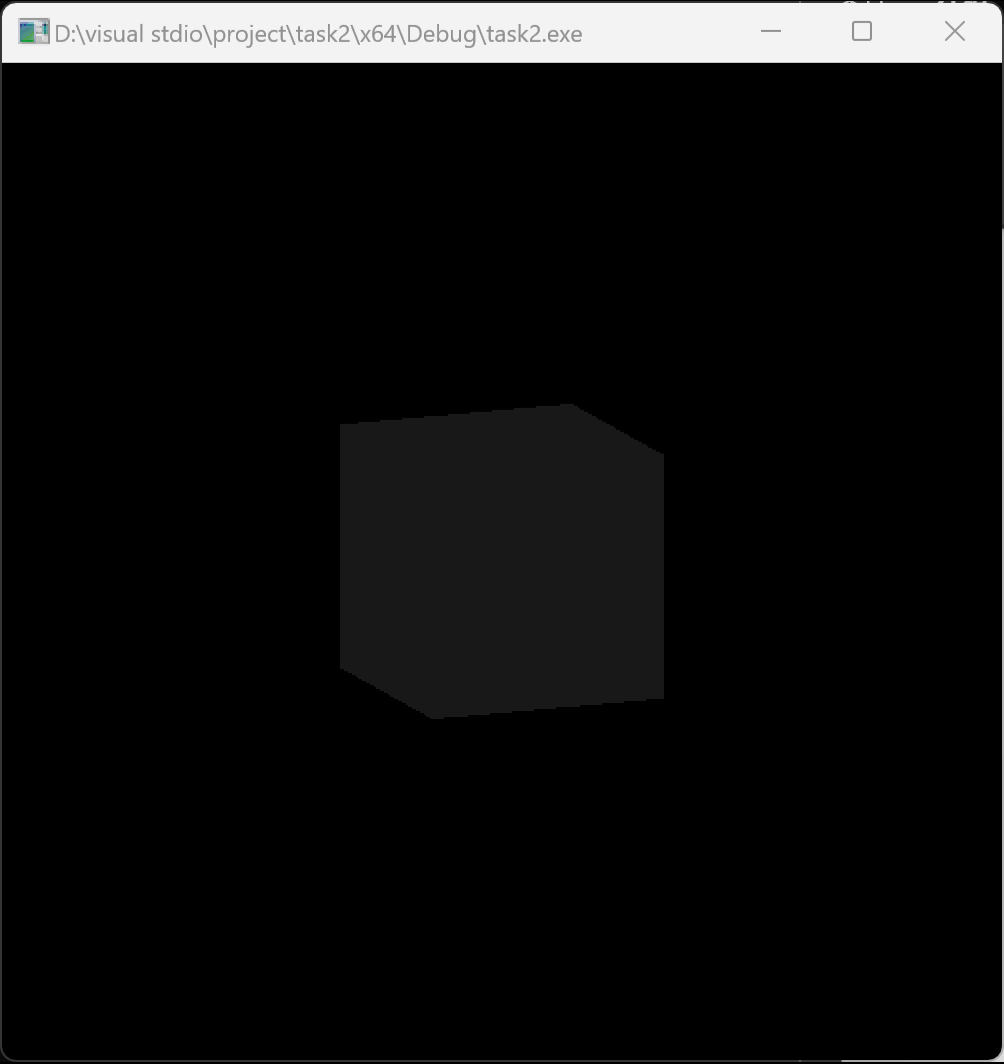
黄铜材质



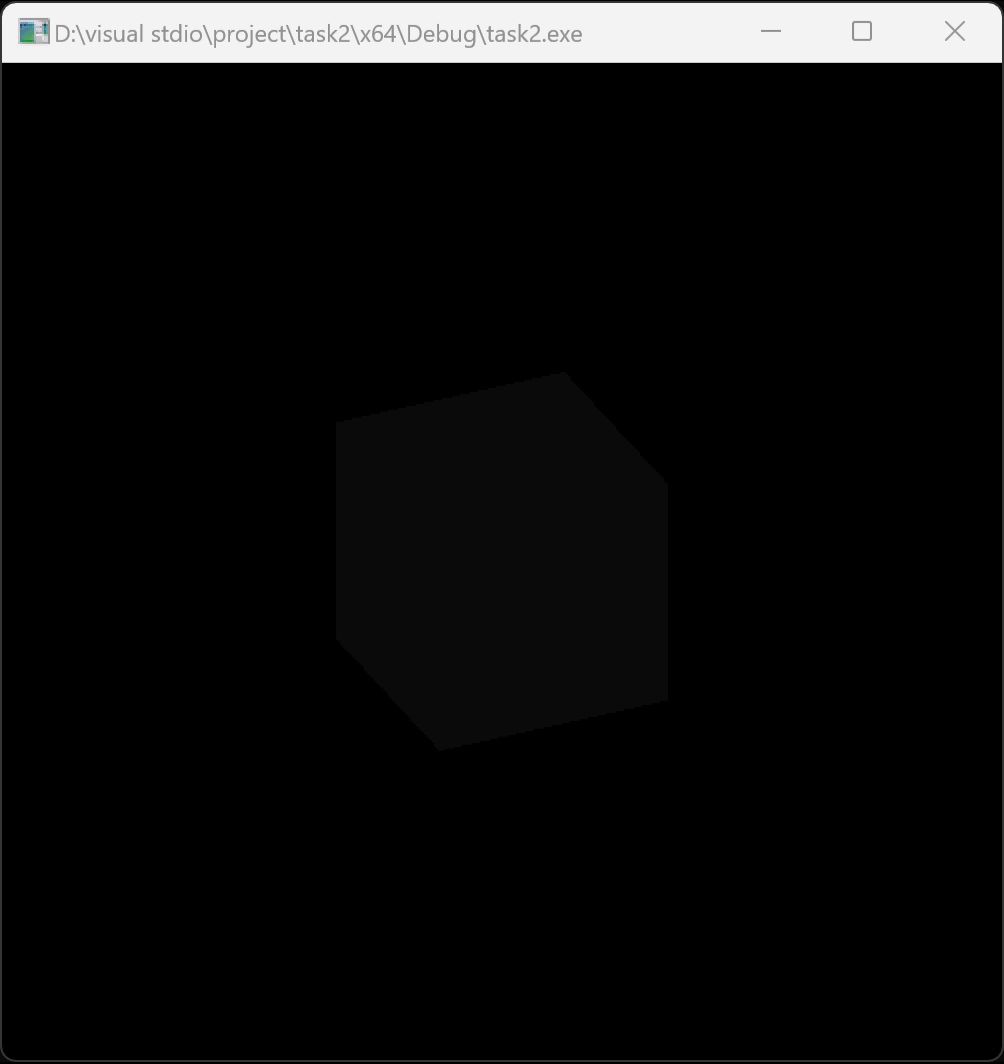
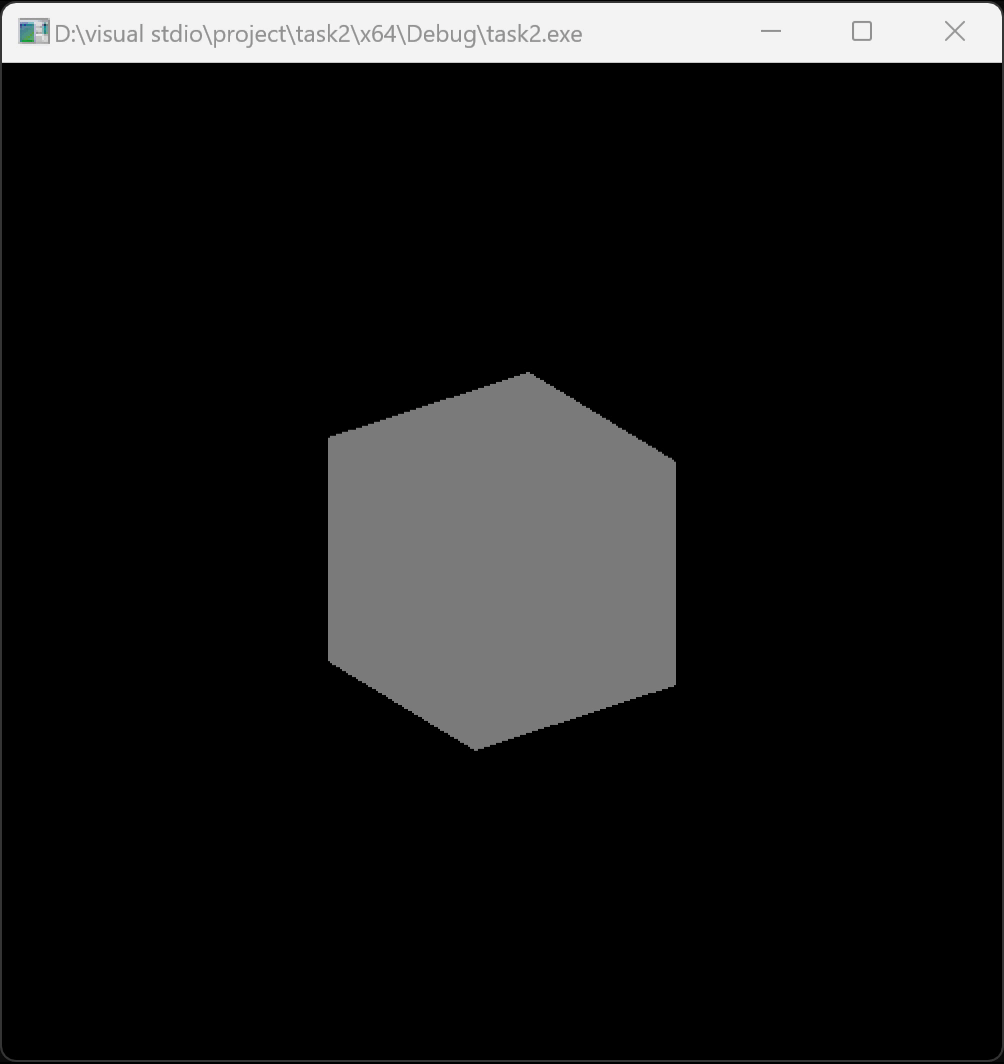
红色塑料材质



白色亮光材质



光源对照，以初始立方体为例



## 2.附加题

**实现交互式明暗计算（用滑动条来调整光照和材质参数）**

操作流程：/

关键代码截图：/

运行结果截图： /