

计算机图形学实验

实验3、三维模型显示

姓 名：\_\_\_\_黄子安\_\_\_\_\_

学 号：22920212204396\_

学 院：\_\_\_\_信息学院\_\_\_

专 业：\_\_\_\_软件工程\_\_\_

年 级：\_\_\_\_2021级\_\_\_\_

2023年 4月 30 日

目录

[Task1: 纯色小人绘制 3](#_Toc133867586)

[1.1文件输入 3](#_Toc133867587)

[1.2 绘制模型 5](#_Toc133867588)

[Task2：绘制光暗小人 6](#_Toc133867589)

[2.1绘制立体模型 6](#_Toc133867590)

[2.2 光源旋转 7](#_Toc133867591)

[2.3 实现视角转动 9](#_Toc133867592)

[Task 3：明暗立方体 11](#_Toc133867593)

[3.1 修改绘制代码 11](#_Toc133867594)

[3.2 改变光源和材质 12](#_Toc133867595)

[附加题：滑动条 15](#_Toc133867596)

[4.1 思路 15](#_Toc133867597)

[4.2 初始化滑动条 15](#_Toc133867598)

[4.3绘制滑动条 16](#_Toc133867599)

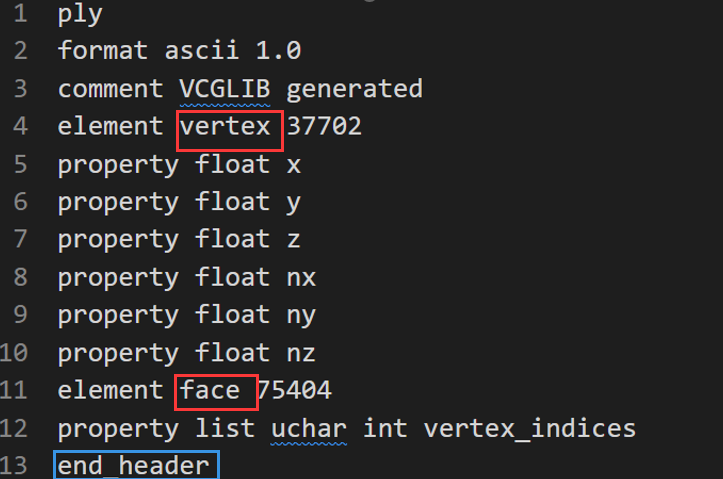
[4.4实现滑动功能 17](#_Toc133867600)

# Task1: 纯色小人绘制

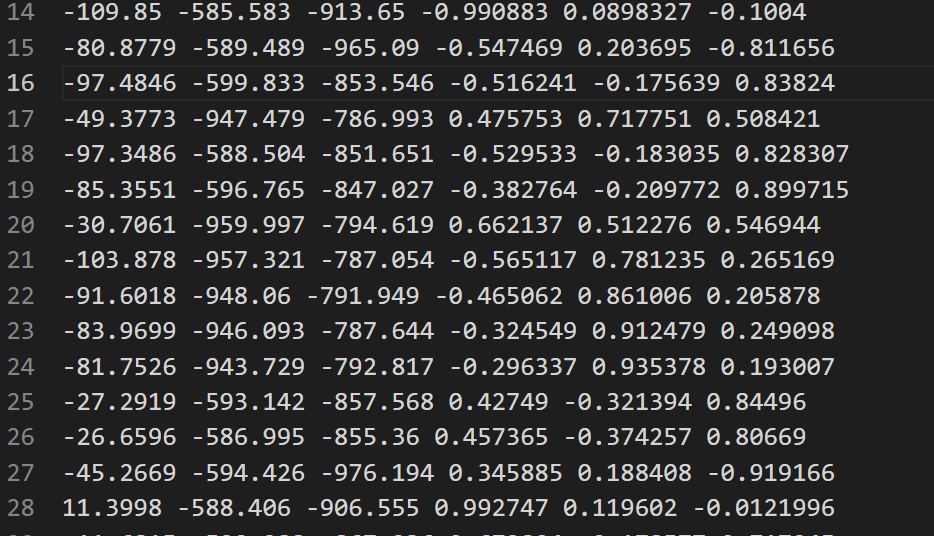
## 1.1文件输入

先利用vscode打开所给ply文件观察对应的内容格式，可以发现一个ply文件由三个部分组成：

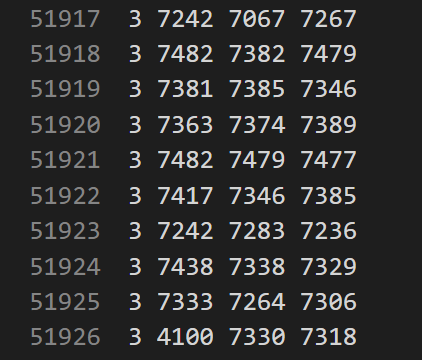
1.文件头，其中说明了该文件中一个顶点含有的信息个数是6，分别为点坐标三个坐标加上对应法向量的三个坐标，总共有37702个顶点，75404个面元。



2.顶点，每一个顶点含有6个数据信息，分别表示顶点本身3个坐标和法向量的3个坐标，以及它们的数据类型都为float型



3.面元，可知每一个面元都是三角形，其中的数字代表了对应顶点的数组下标



关键代码：

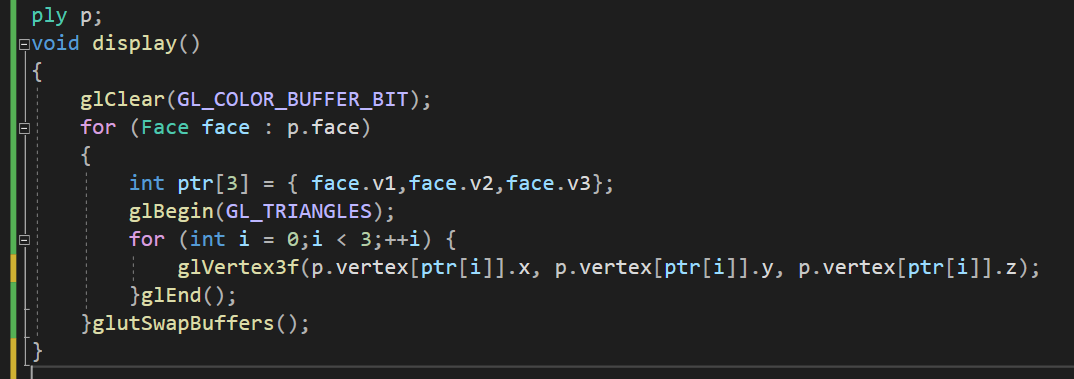
为了使代码不过于冗余，对所读取的文件给出以下要求：每个顶点的参数个数都为6且为float类型，面元都为三角形且给出三个点对应的数组下标。

读取数据代码使用c++的文件流，通过对ply文件头的特征获取顶点数和面元个数，之后进行读取每一个顶点和面元信息，保存在vector中。

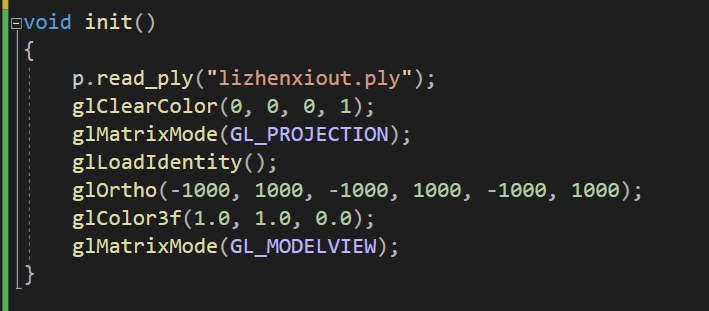


## 1.2 绘制模型

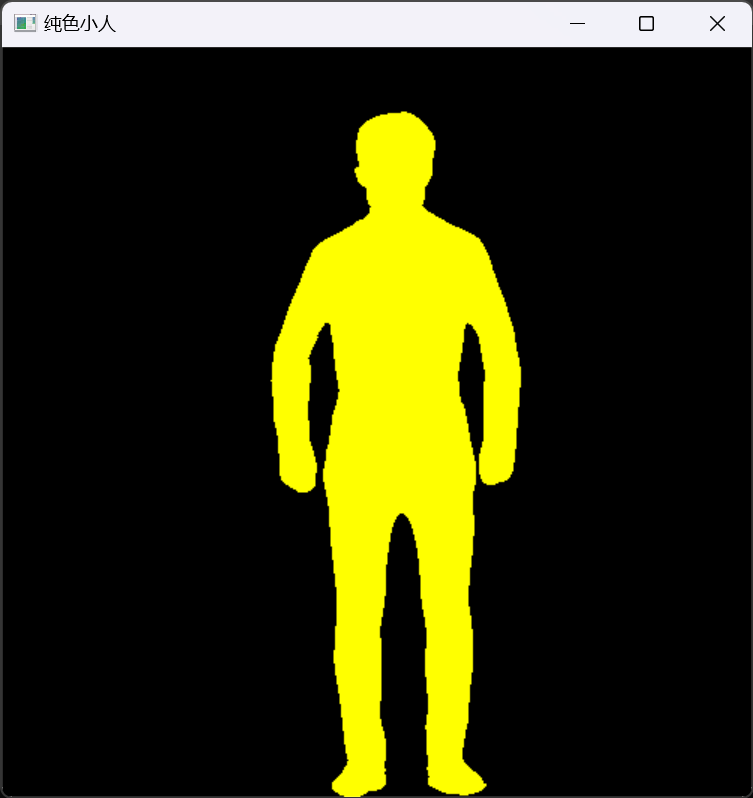
完成对ply文件的读取之后利用glVerrtex3f即可完成点的绘制。



但是此时是无法找到模型，原因在于模型点坐标过大，直接超出了投影矩阵的范围，因此需要对投影矩阵进行修改，通过对ply文件的分析可以获取适当的参数



运行效果：

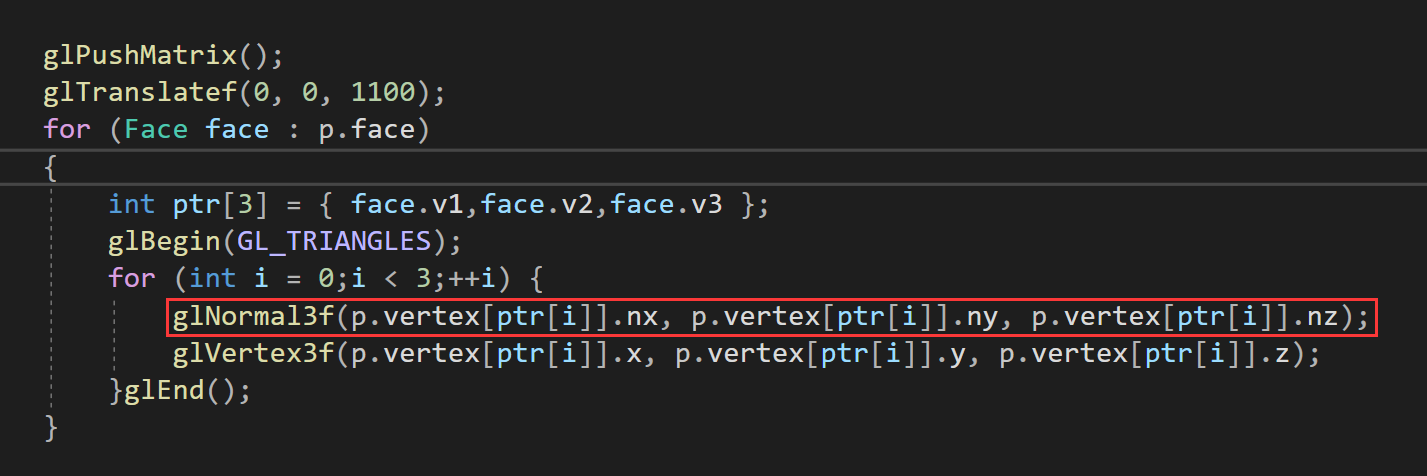


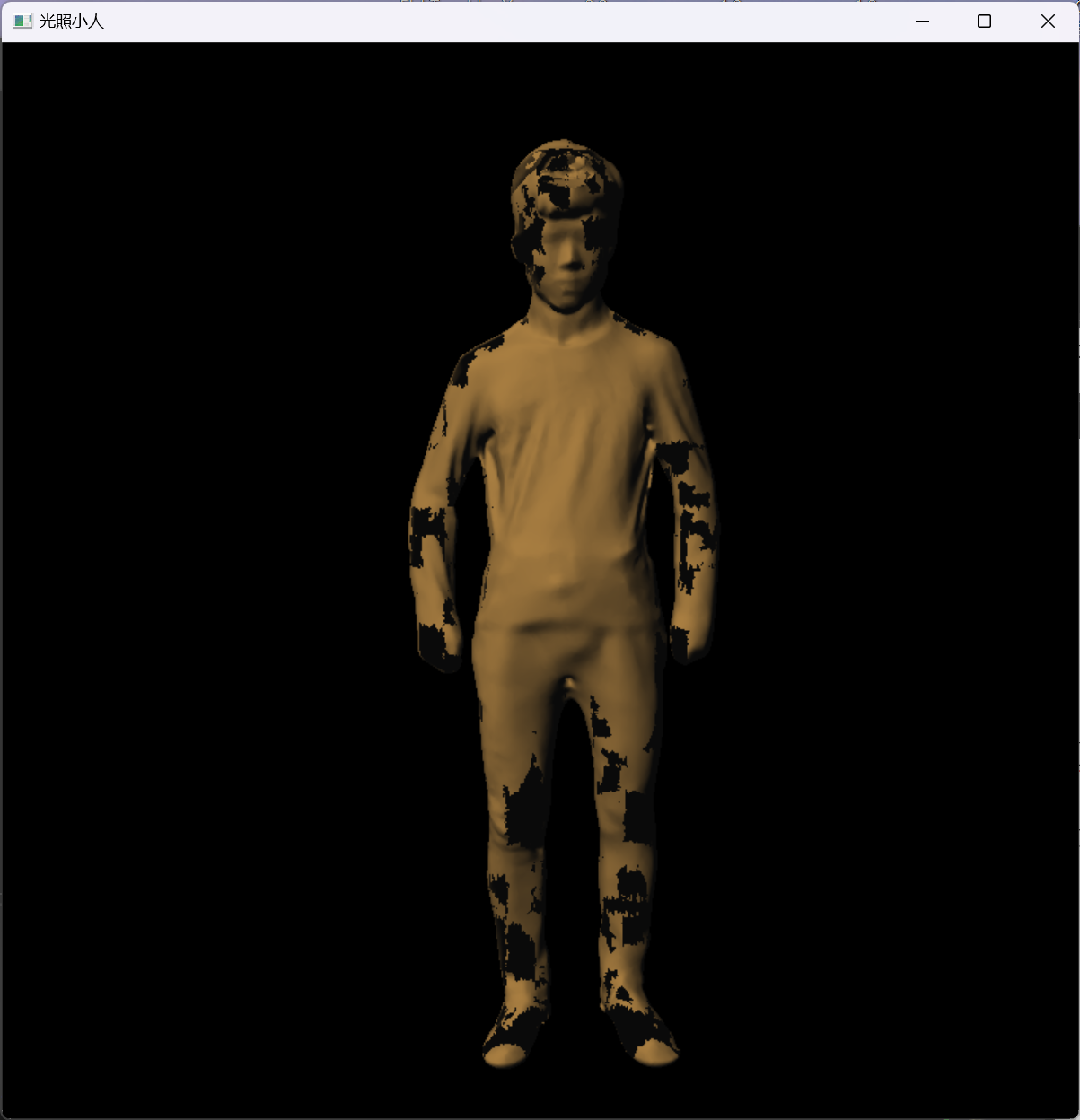
# Task2：绘制光暗小人

## 2.1绘制立体模型

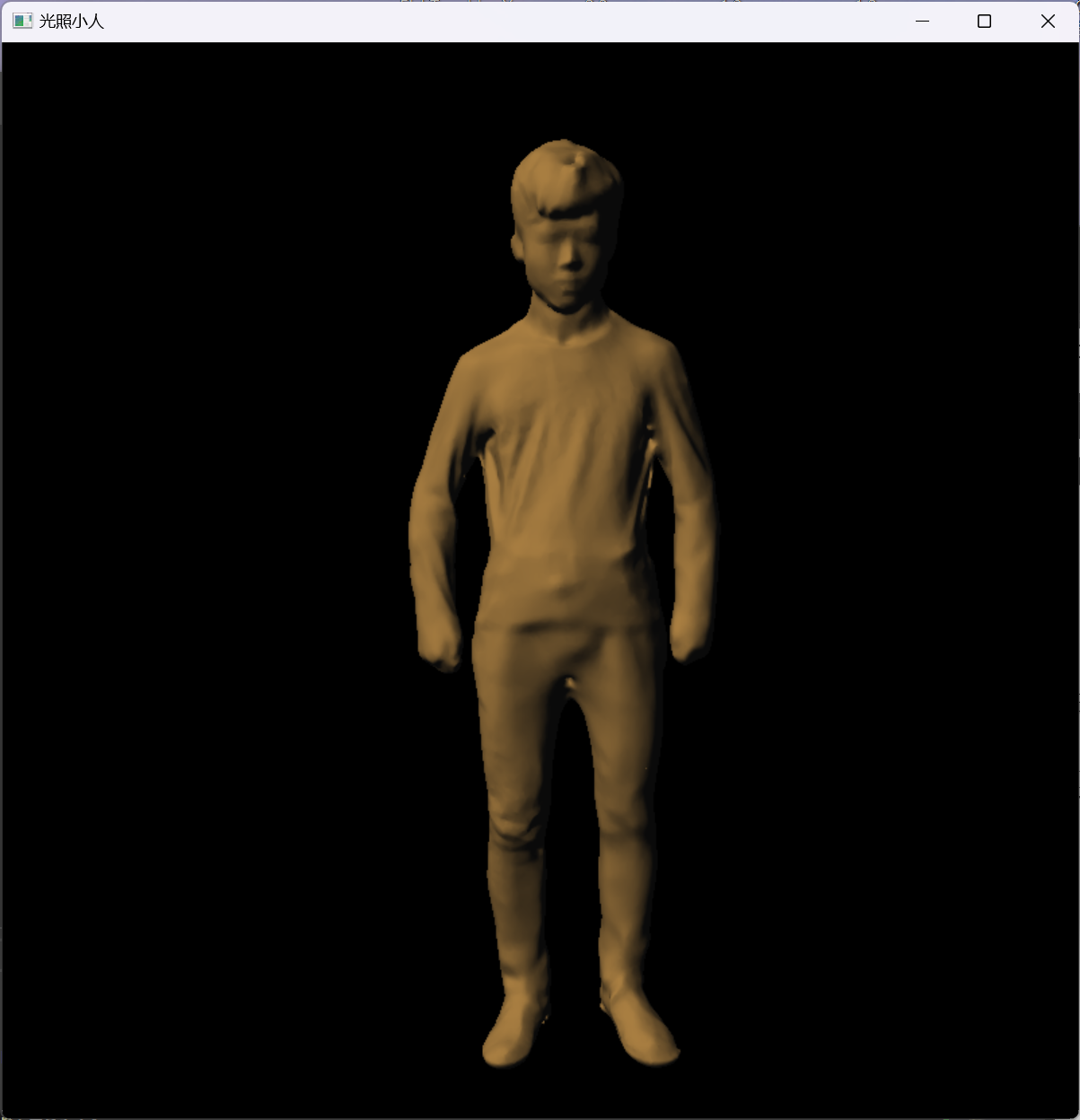
为了绘制立体模型，需要开启openGL的光照功能，此时glColor函数将不再发挥作用，模型的颜色将由光的颜色和物体的材质决定，此时绘制模型要给出每一个点的法向量坐标，之后由openGL使用phong模型进行绘制





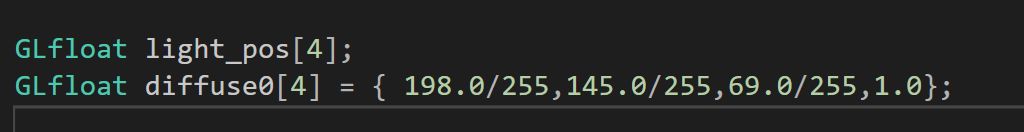


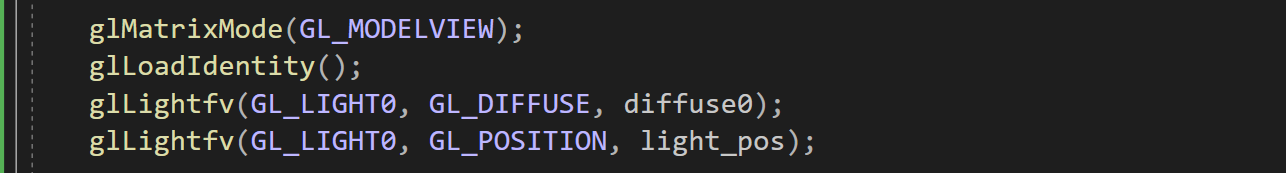
观察生成的结果可以发现存在明显问题，经过研究可知是没有启用深度检测，打开深度检测之后可以看到模型效果符合我们的预期

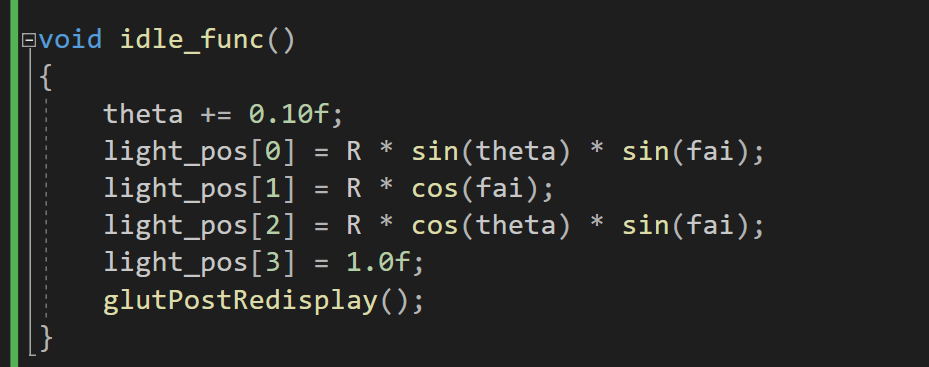


## 2.2 光源旋转

光源由四个属性组成，包括位置、镜面光、环境光和漫反射光，在之前的2.1中已经修改了模型的颜色，通过的就是修改光源的漫反射光属性，要实现光源的旋转同理修改位置参数即可，和之前的实验的操作类似，使用球坐标系描述光源位置，之后注册一个gluIdleFunc的回调函数，改变方位角的大小即可实现光源的旋转

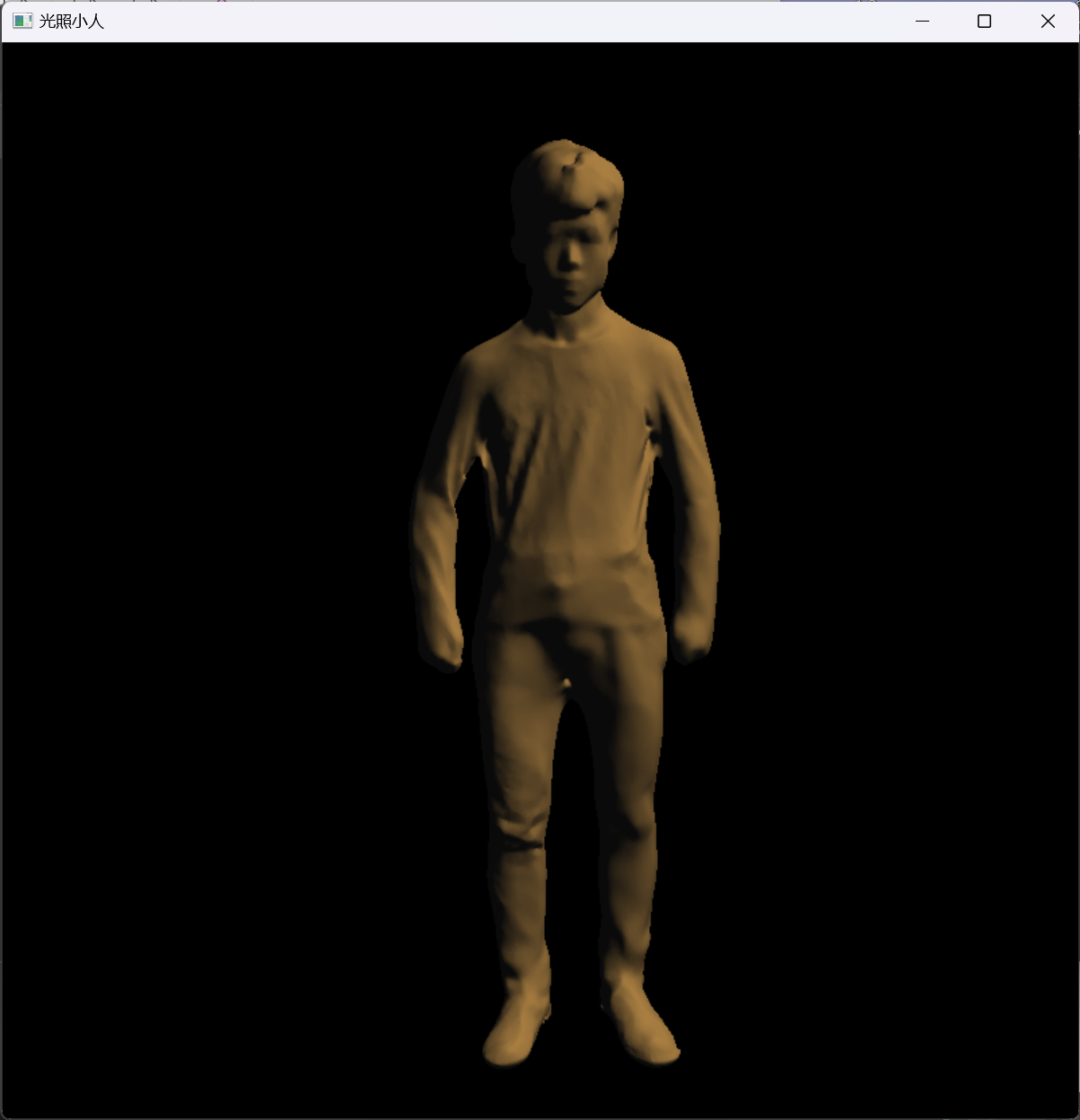
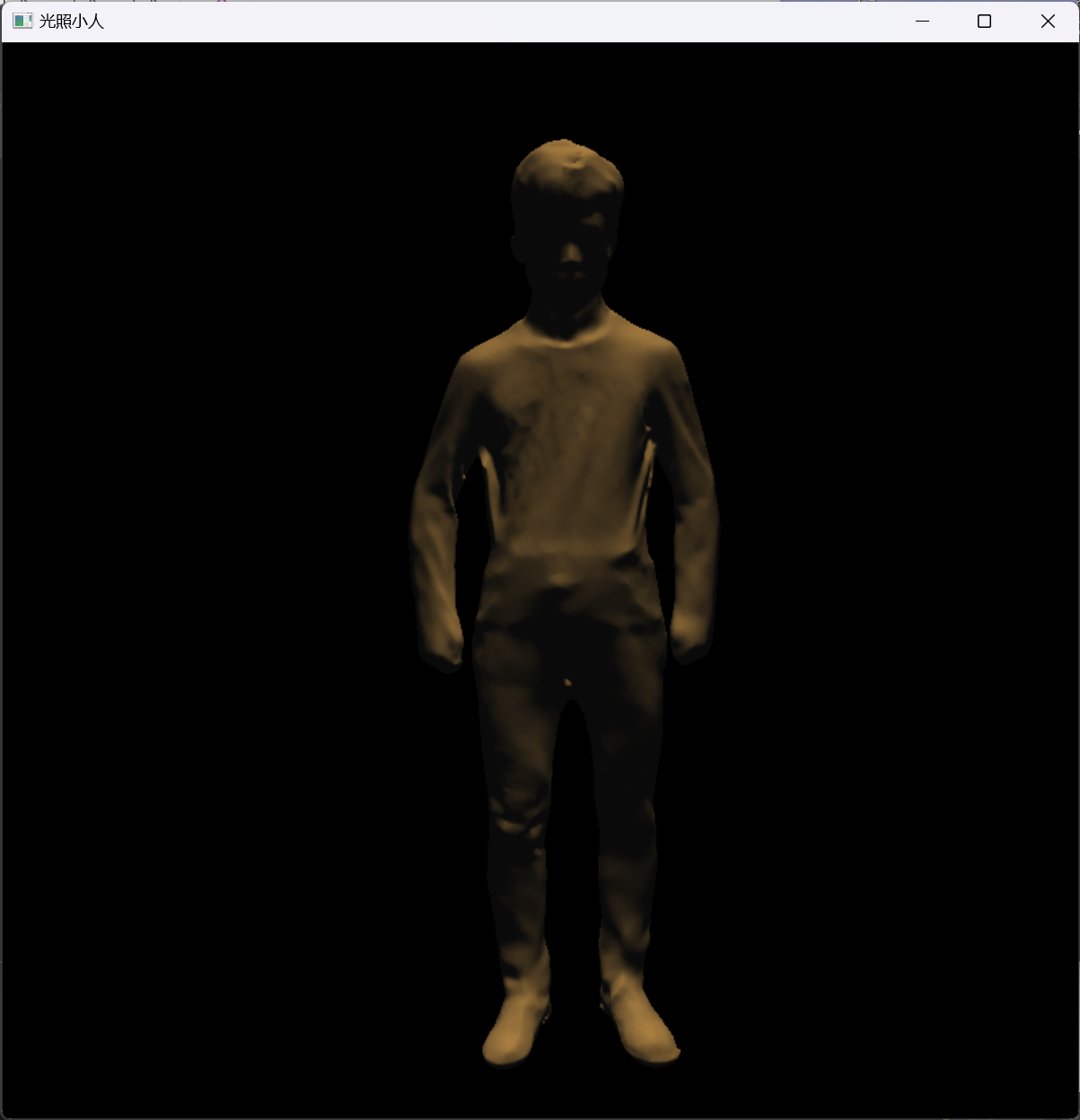


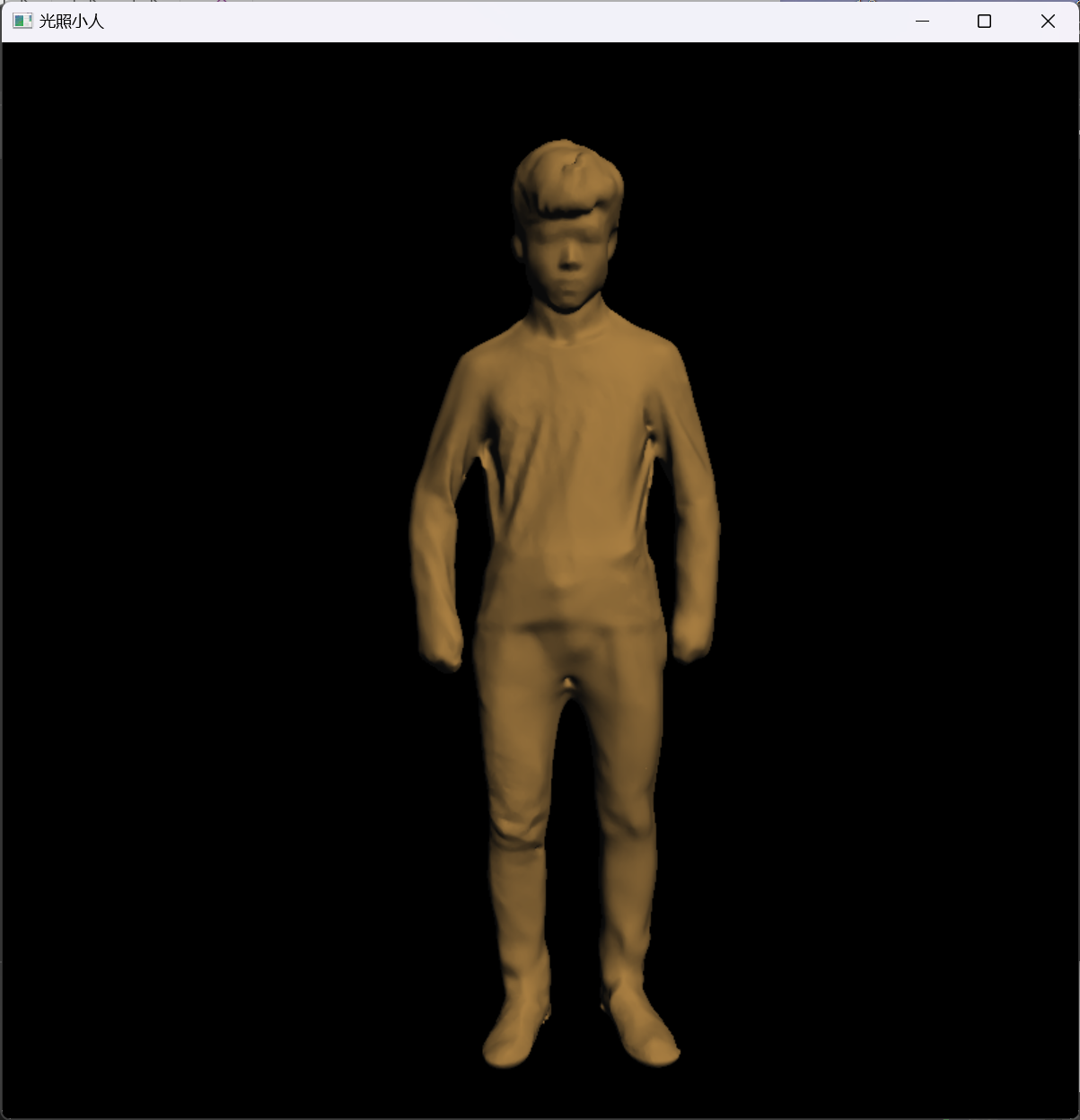
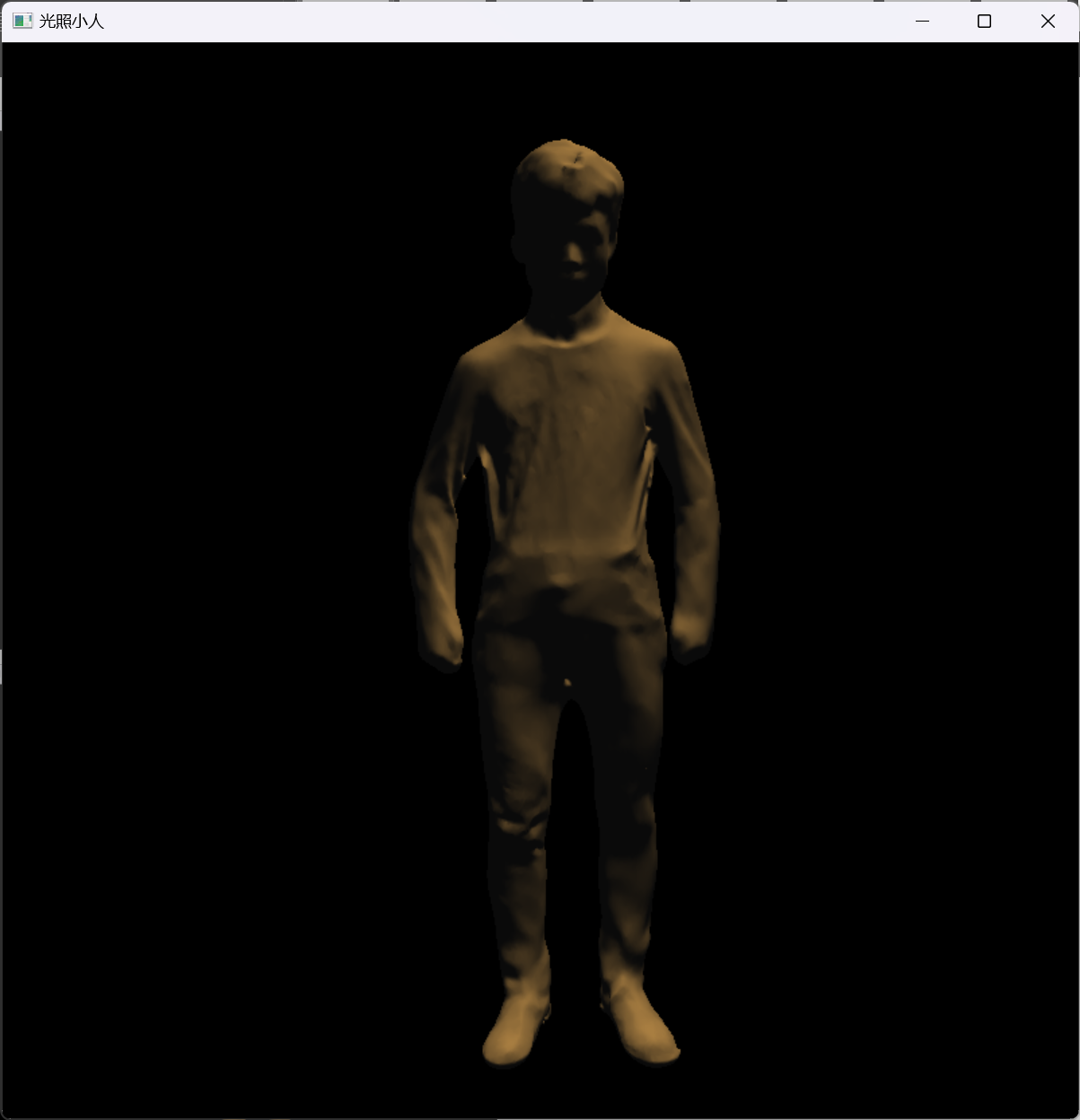




运行效果：

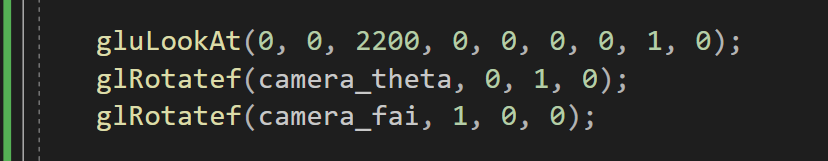
光源围绕小人模型转动

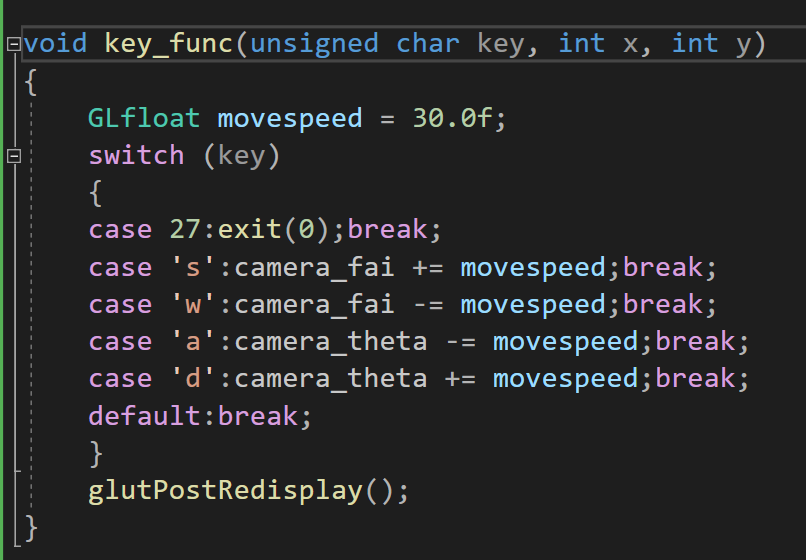


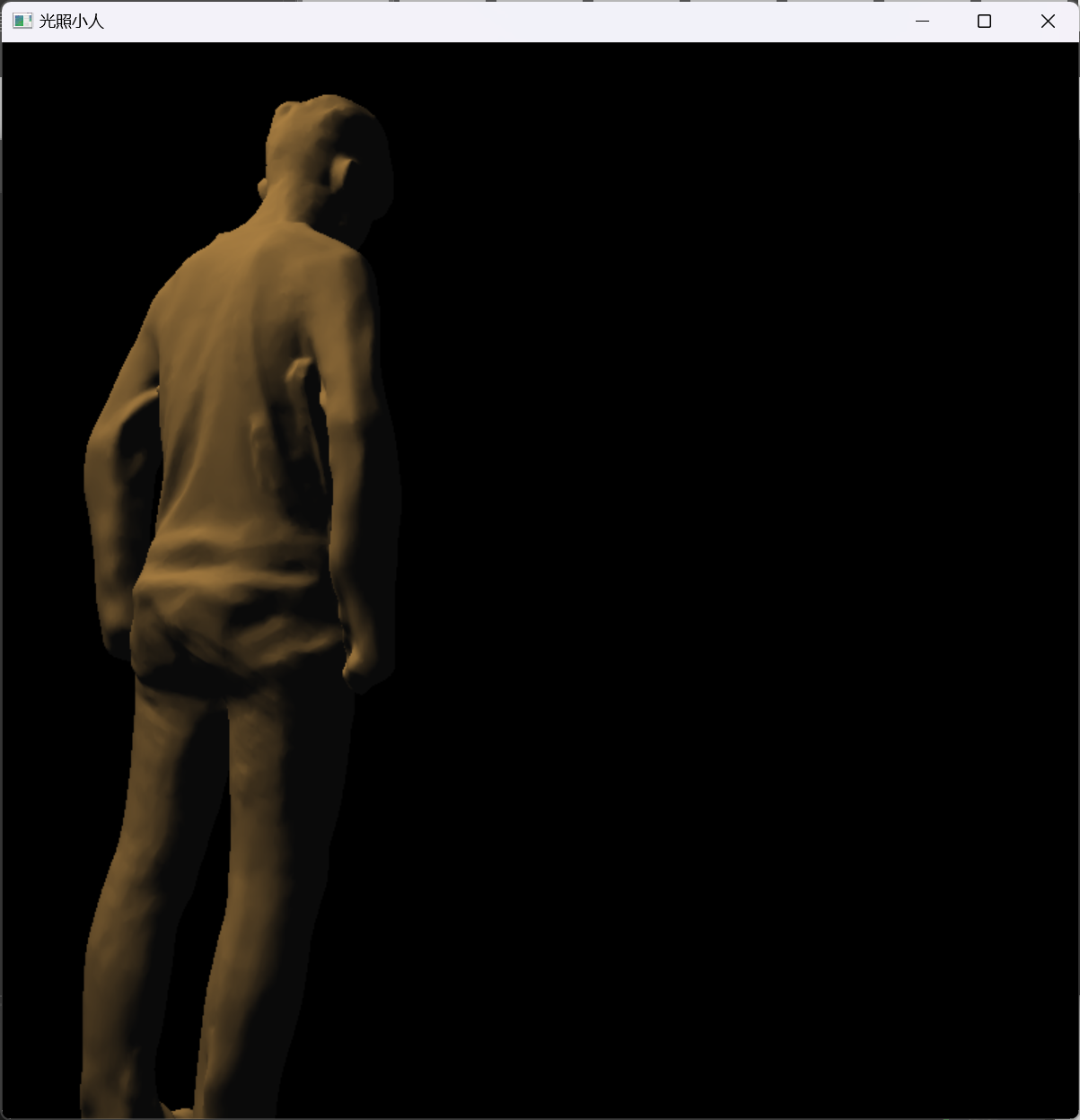
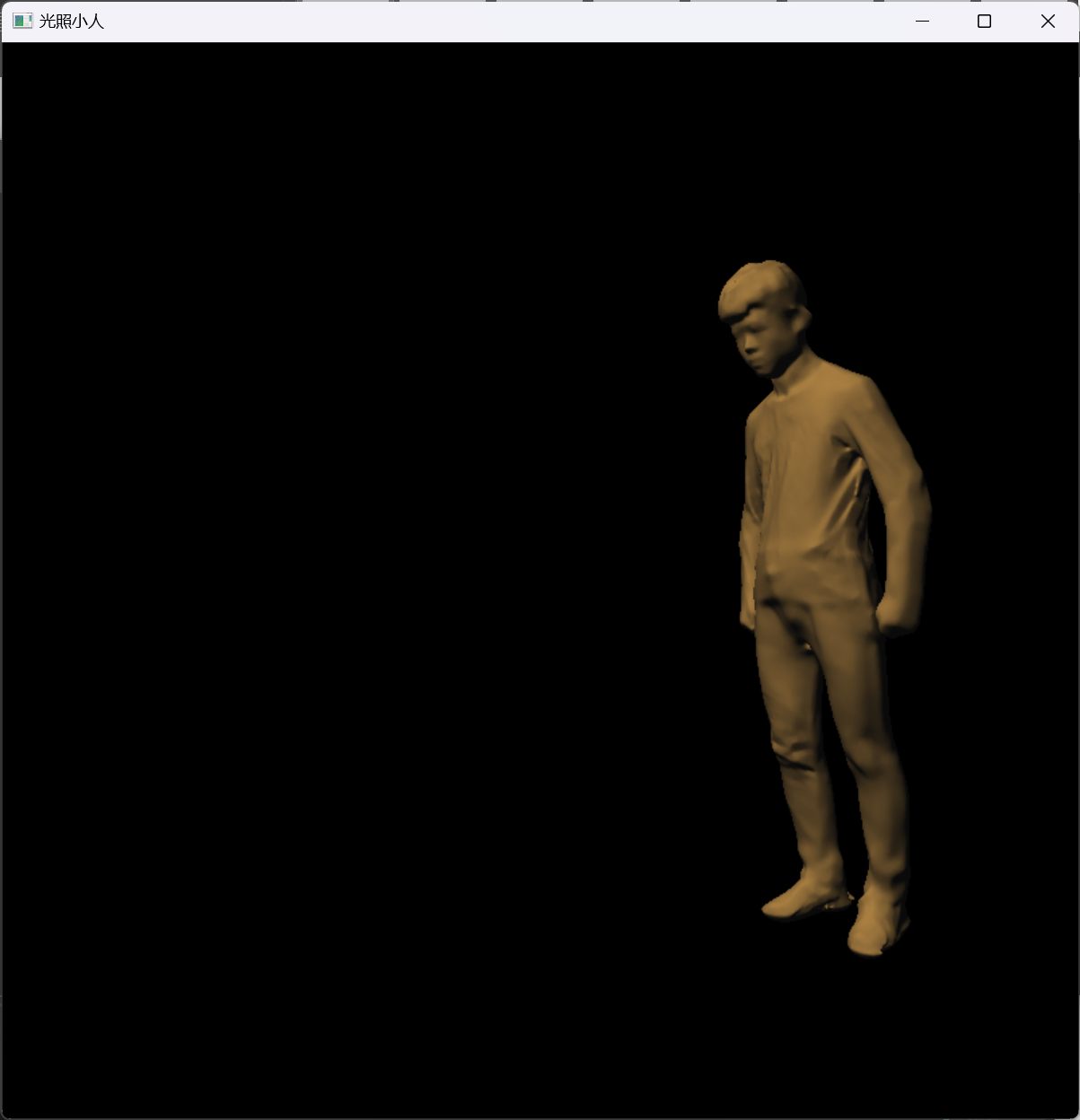
## 2.3 实现视角转动

为了方便观察小人，增加使用键盘对视角进行控制，视角的变换可以等价于对小人进行坐标变换，为了方便起见直接使用glRotatef对小人进行转动

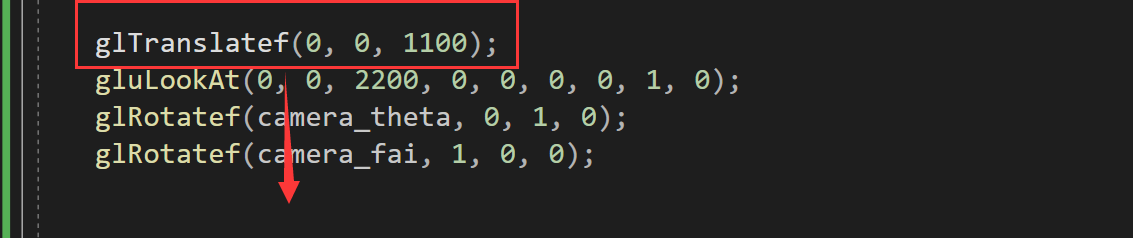




可是当我们进行旋转出现了预期之外的效果，根据代码的思想可以确定小人应该位于窗口中央，但实际旋转时发现小人产生了预期外的旋转效果

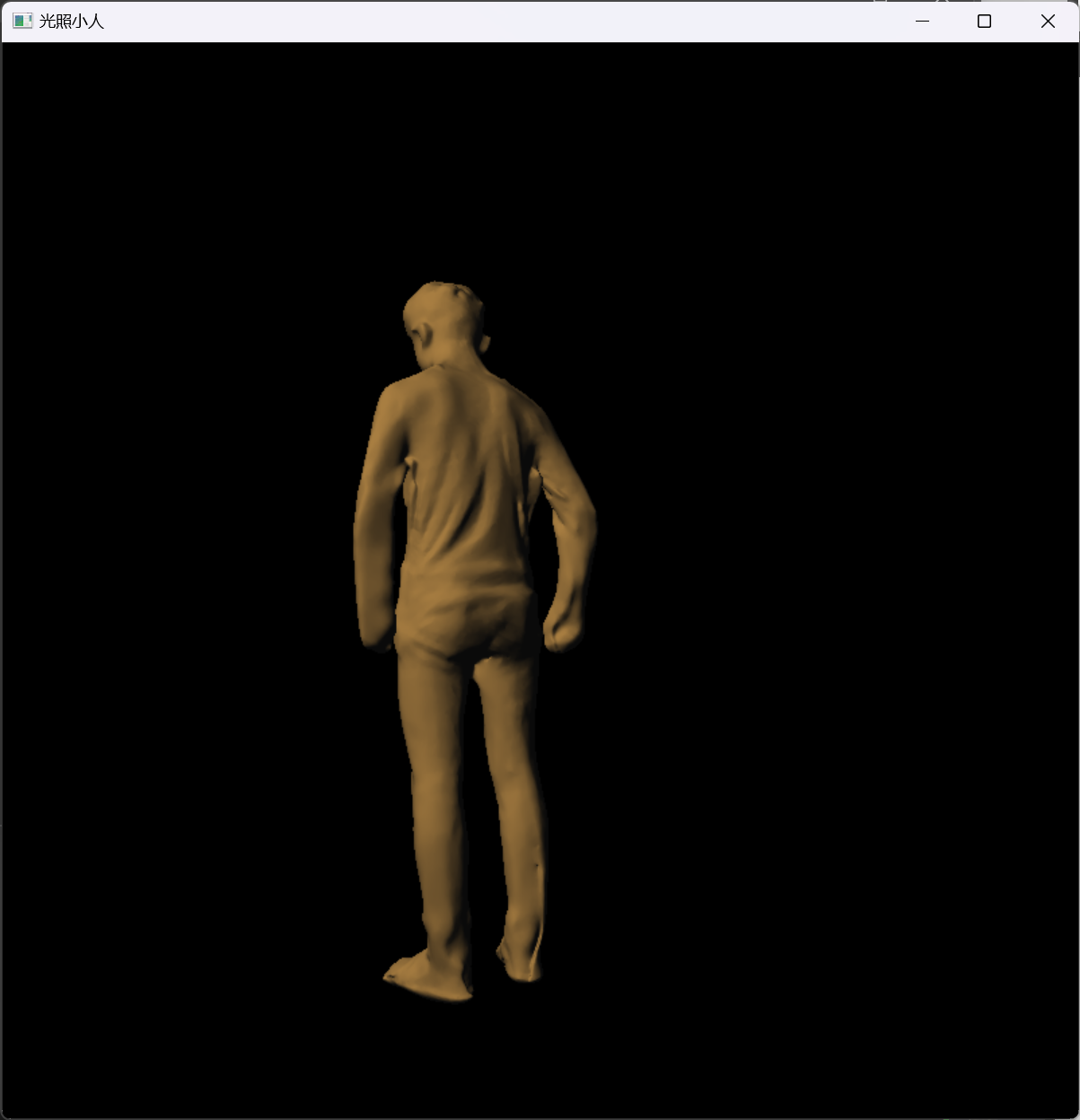
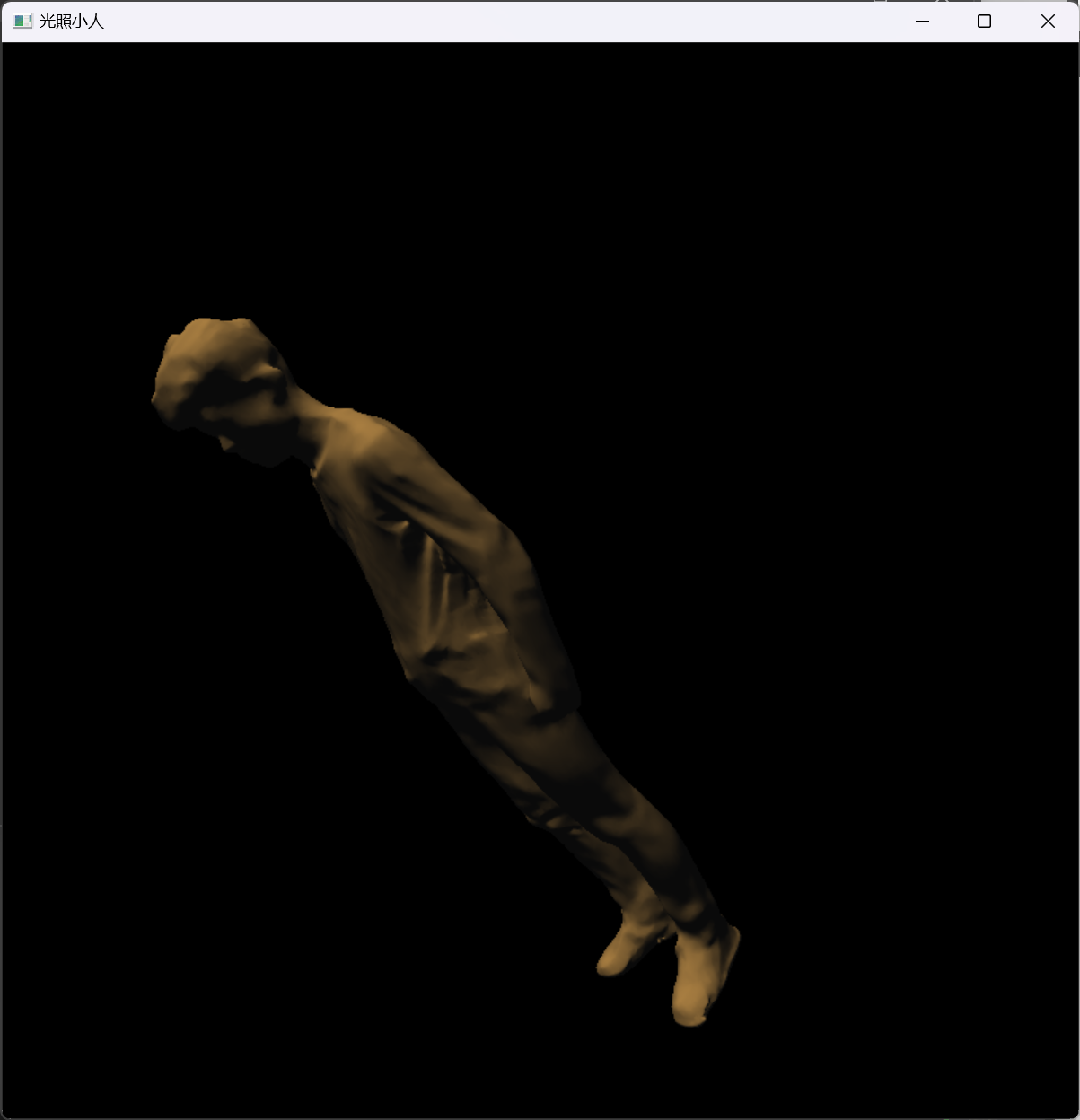
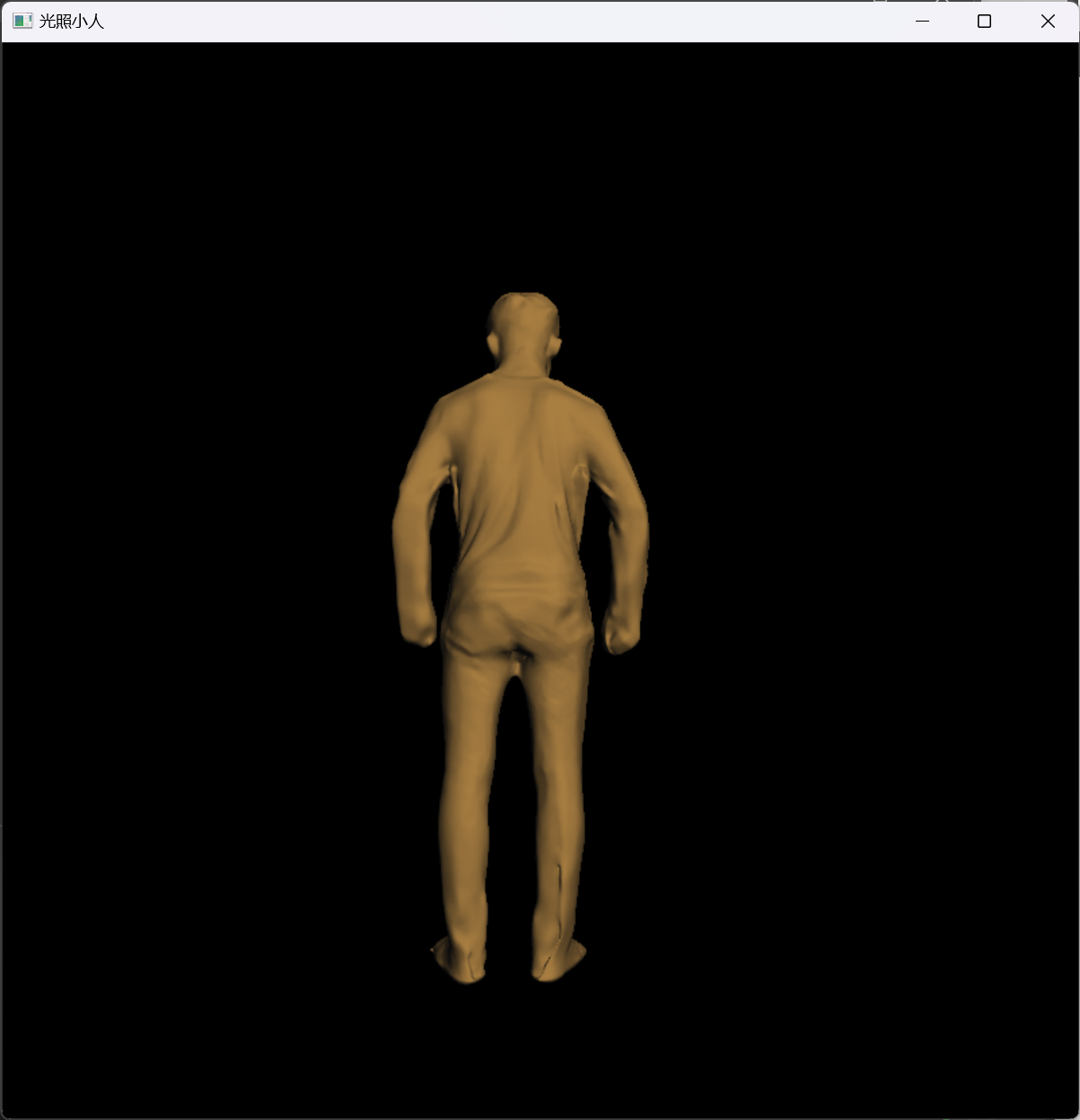


通过分析可以知道该情况主要的原因是小人的中心并不位于坐标轴的原点，为了解决该问题最简单的方法就是将小人向前平移，但此时发现该代码和原来效果还是一样，经过研究发现如果将平移矩阵放置在视角矩阵前面平移矩阵会对之后的所有物体包括摄像机也产生作用，在世界坐标系会产生不会预期的效果



因此需要将物体的平移矩阵移动到视角矩阵之后，避免产生干扰导致无法正确绘制

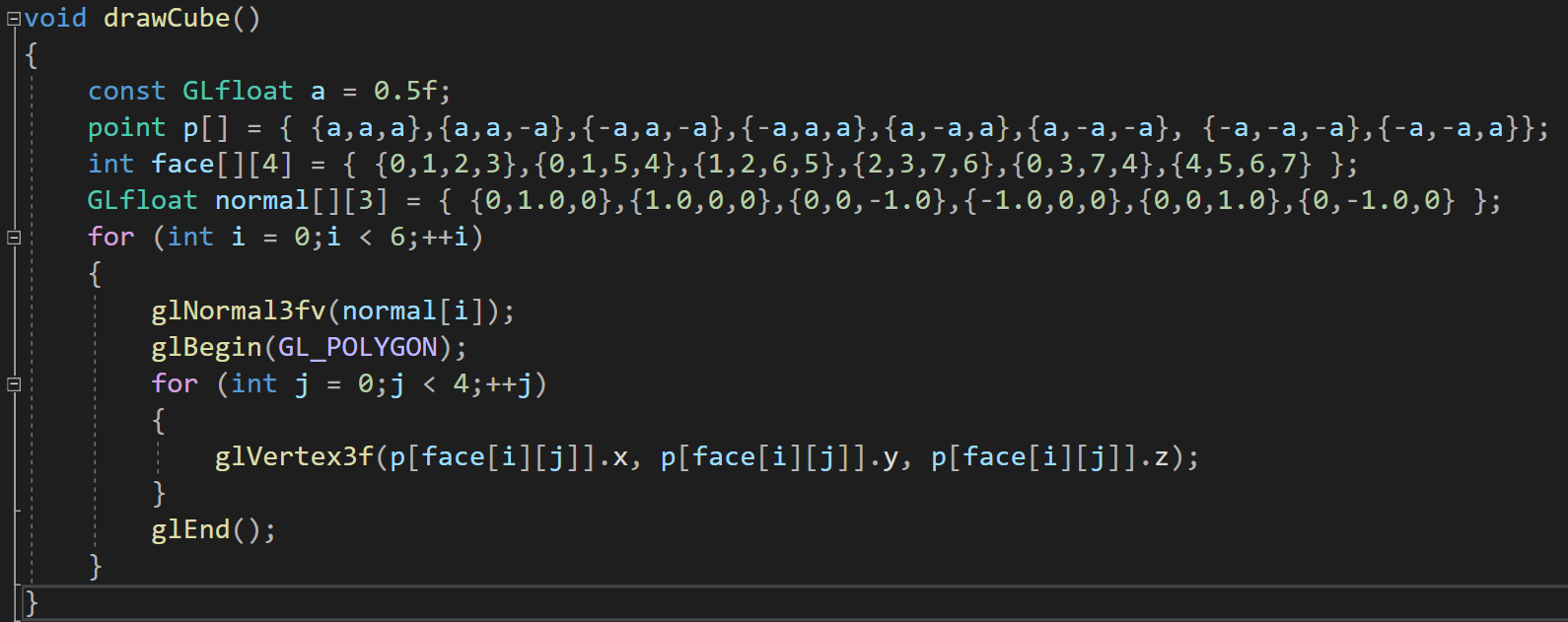
运行效果：



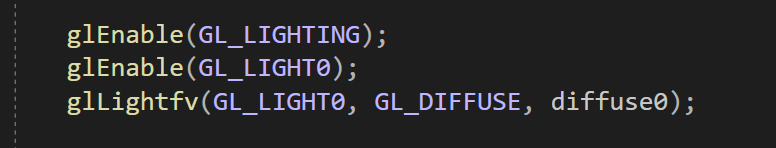
# Task 3：明暗立方体

## 3.1 修改绘制代码

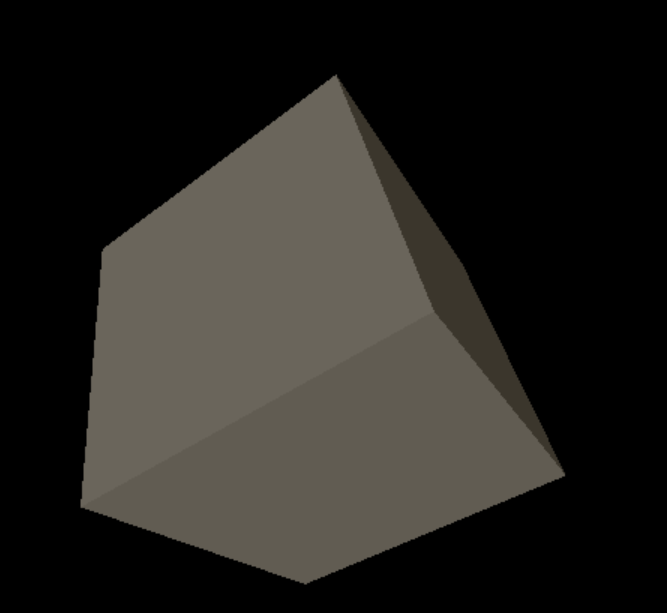
openGL的光照计算需要使用顶点的法向量，对于该立方体的法向量可以直接通过观察法写出，而不必使用叉乘



之后开启光照即可实现明暗计算功能

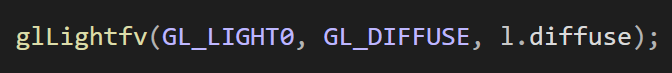


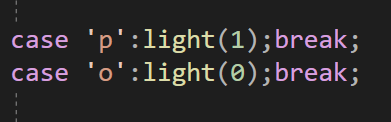
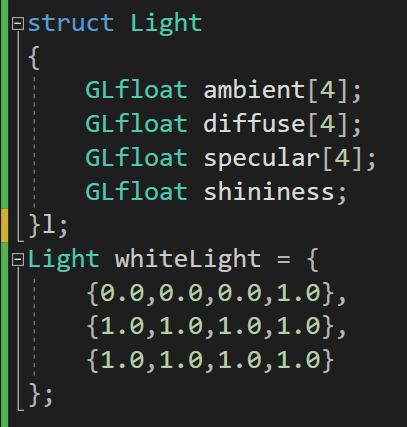
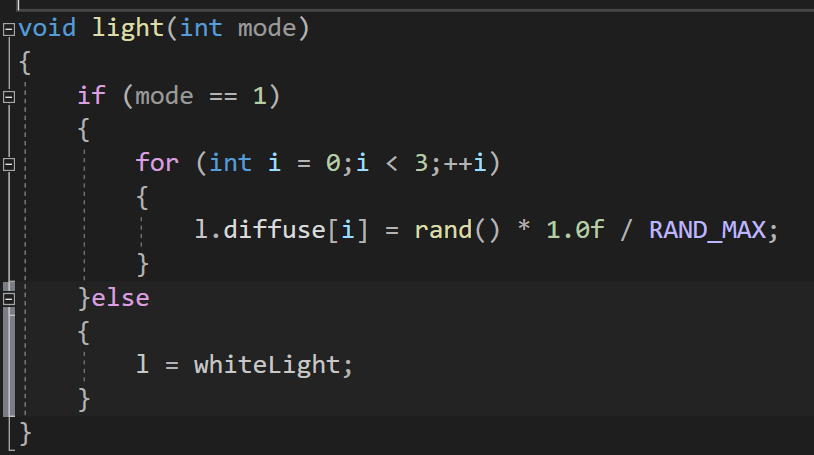
运行效果：



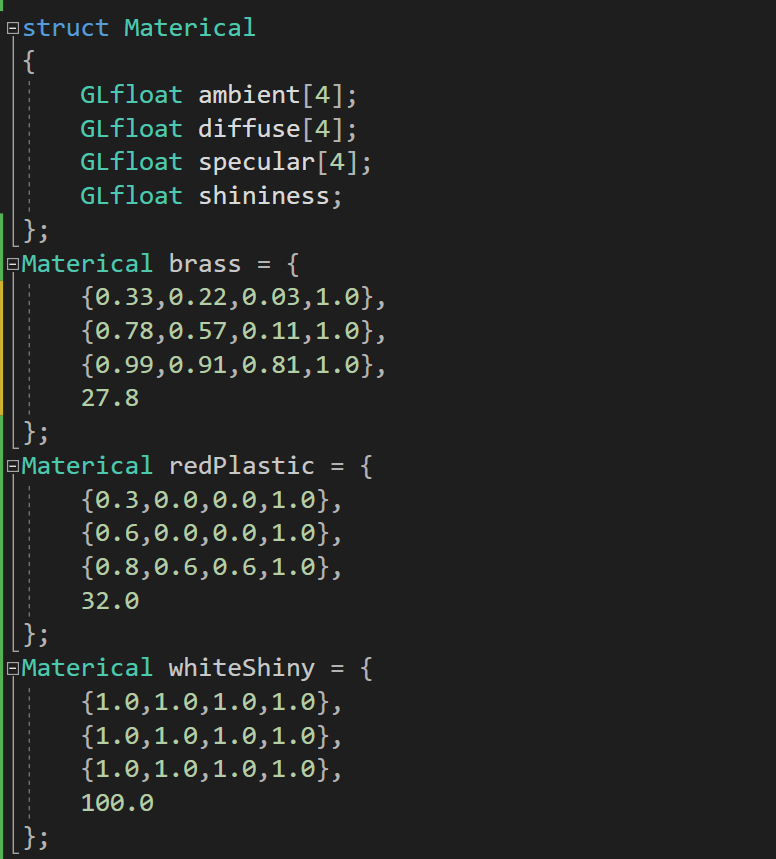
## 3.2 改变光源和材质

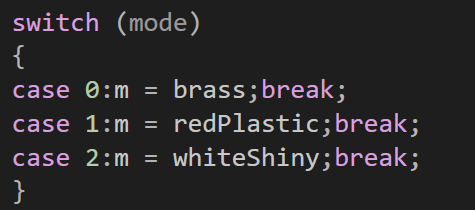
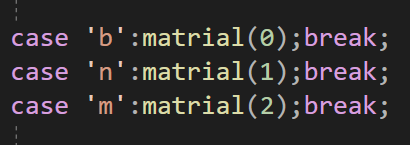
通过phong光照模型可以知道光源由三个部分构成，在代码中使用结构体保存光源的各个属性，之后通过按键实现光源的更改

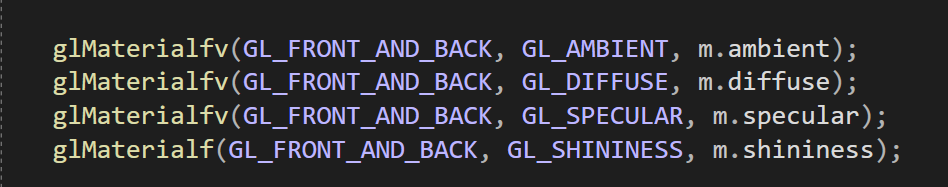




物体表面的材质有四个部分构成，可以采用同样的方法进行描述





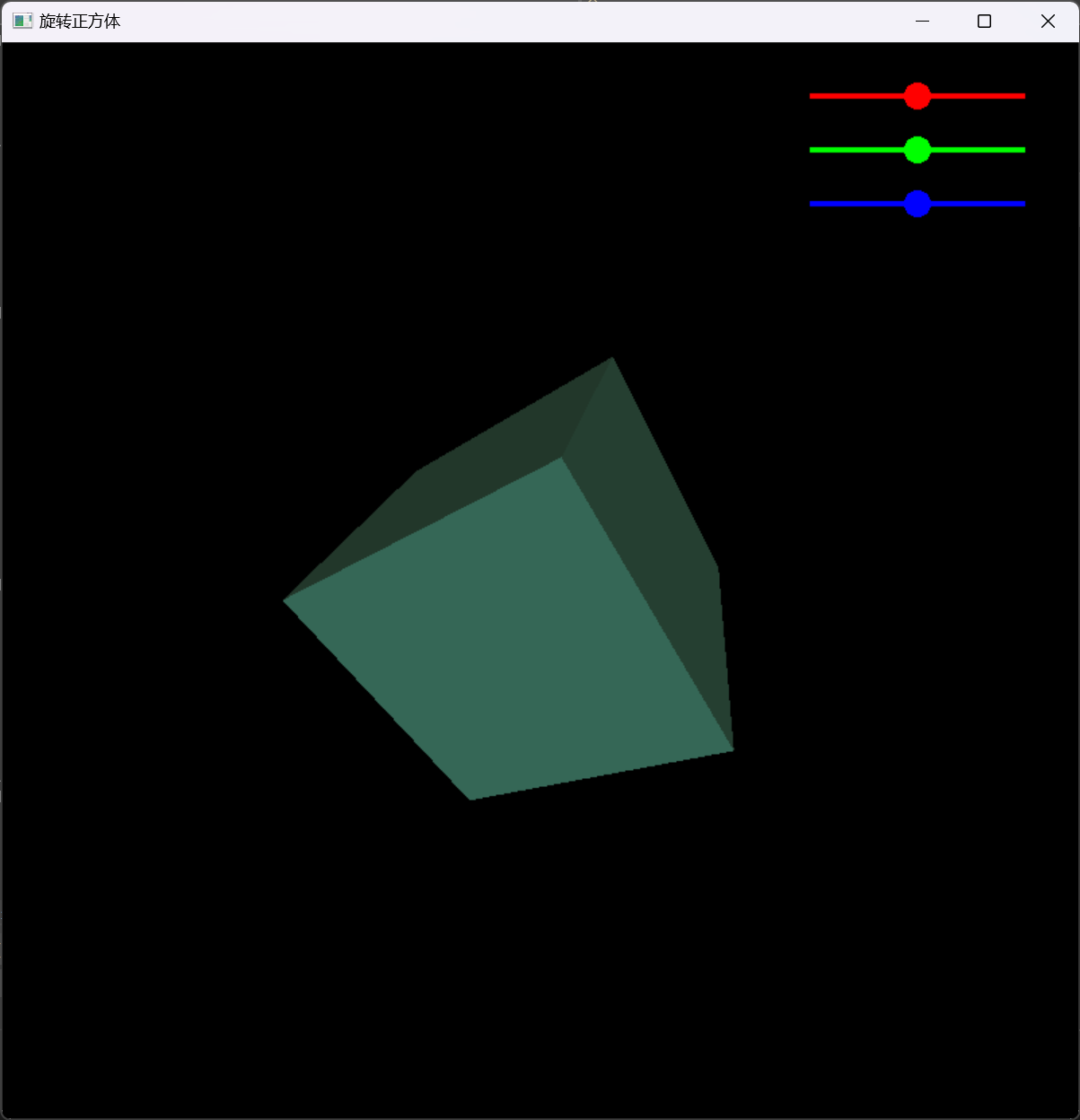


运行效果：

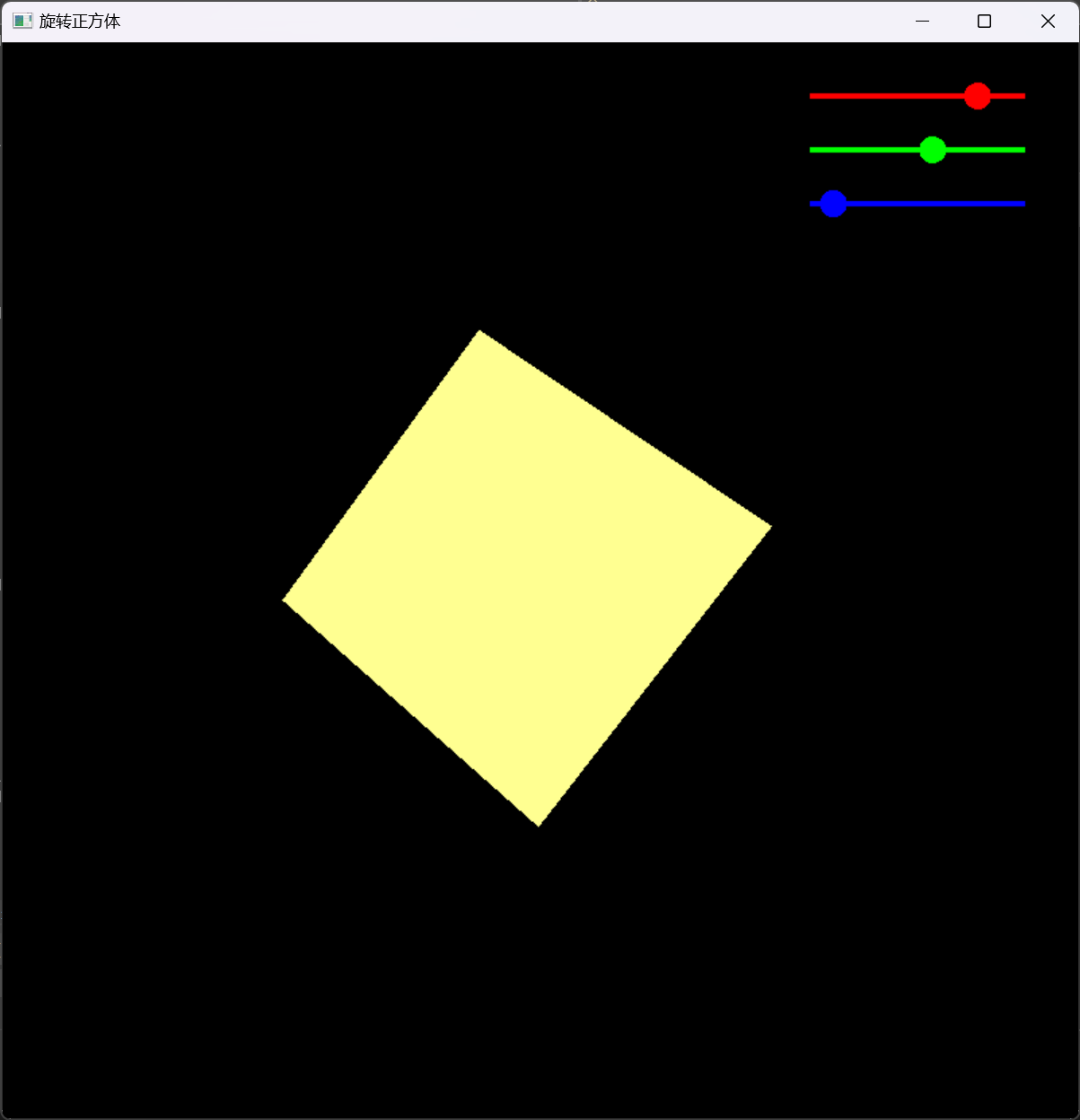
白光



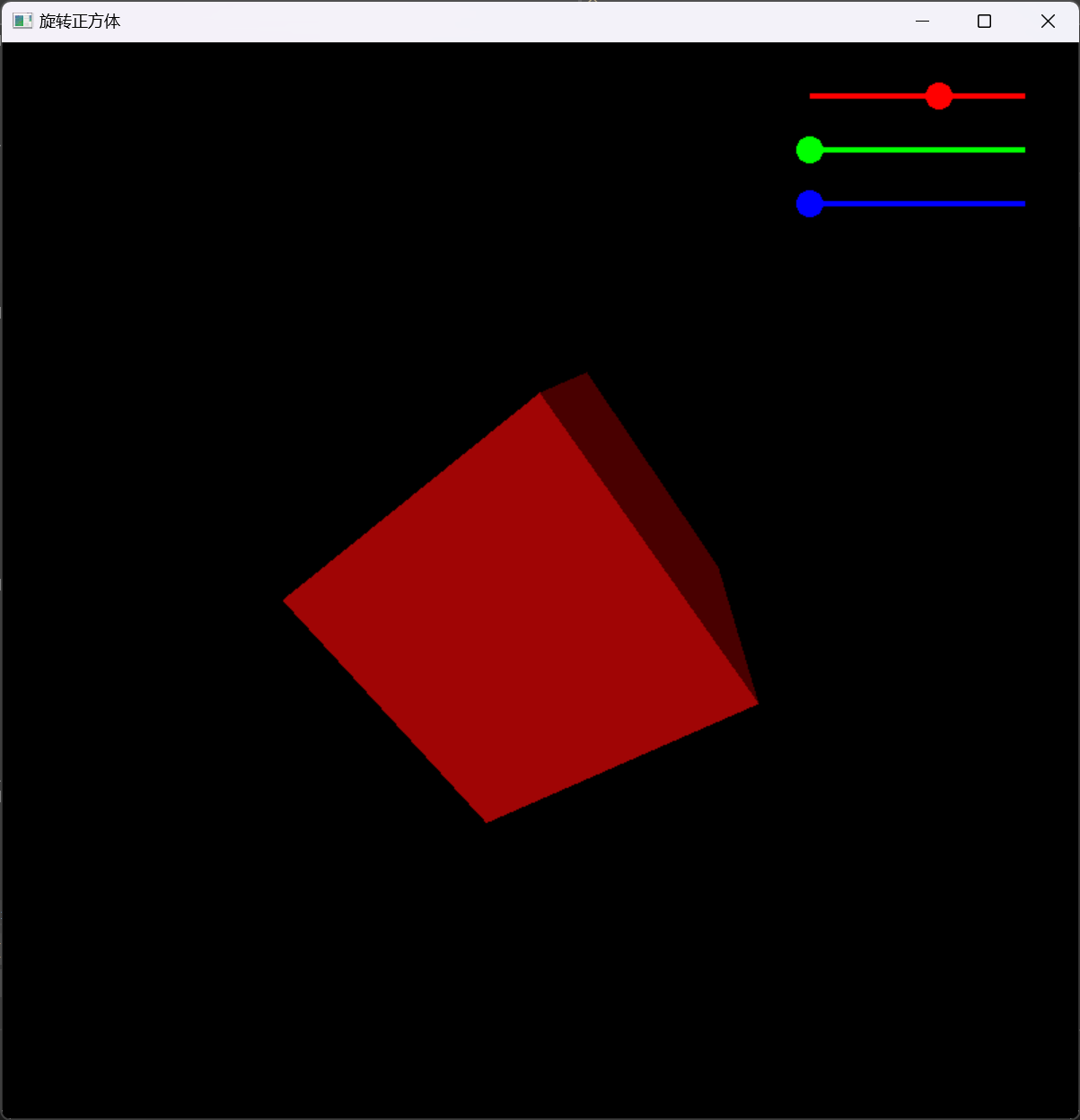
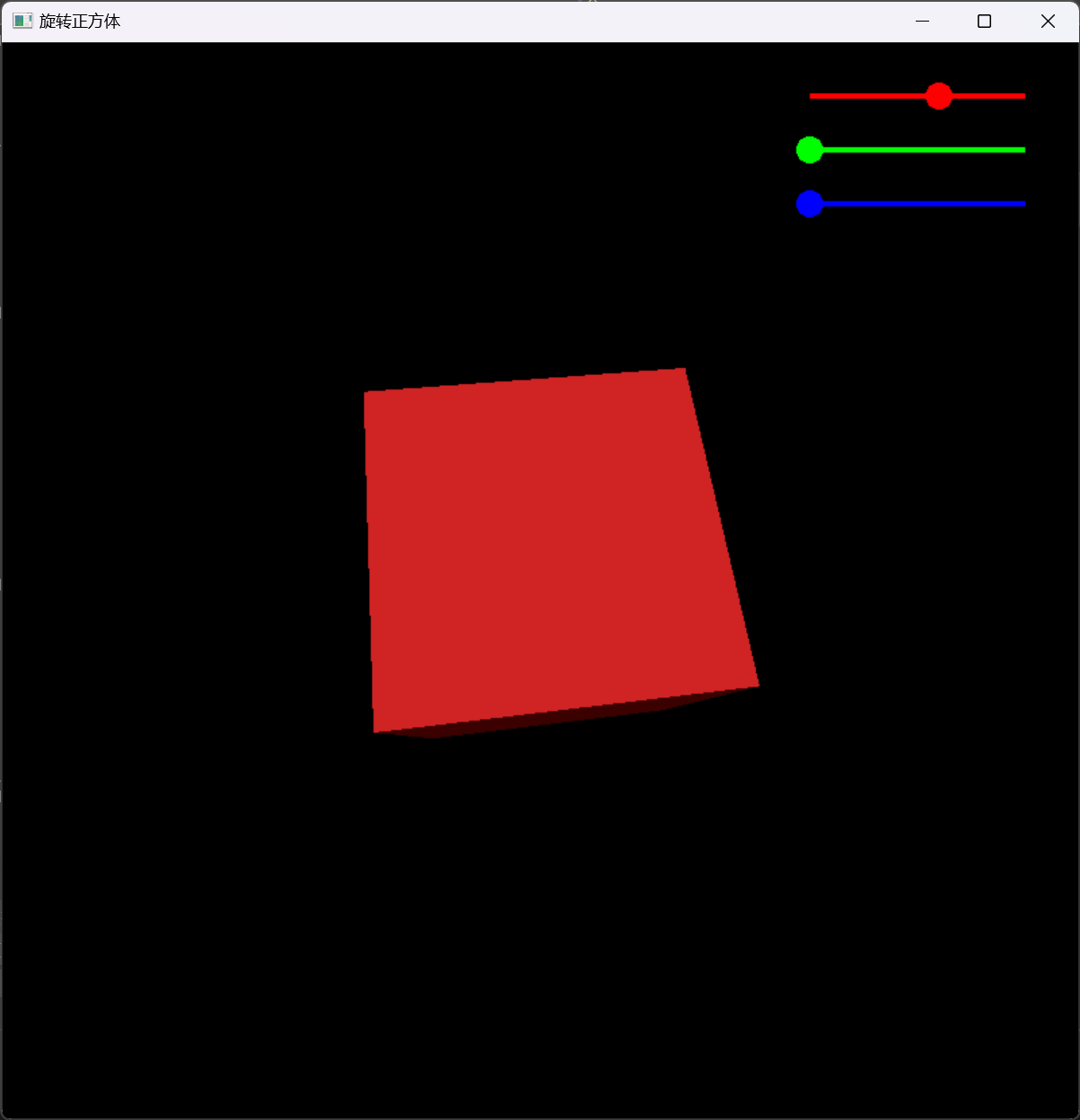
彩色光（颜色随机）



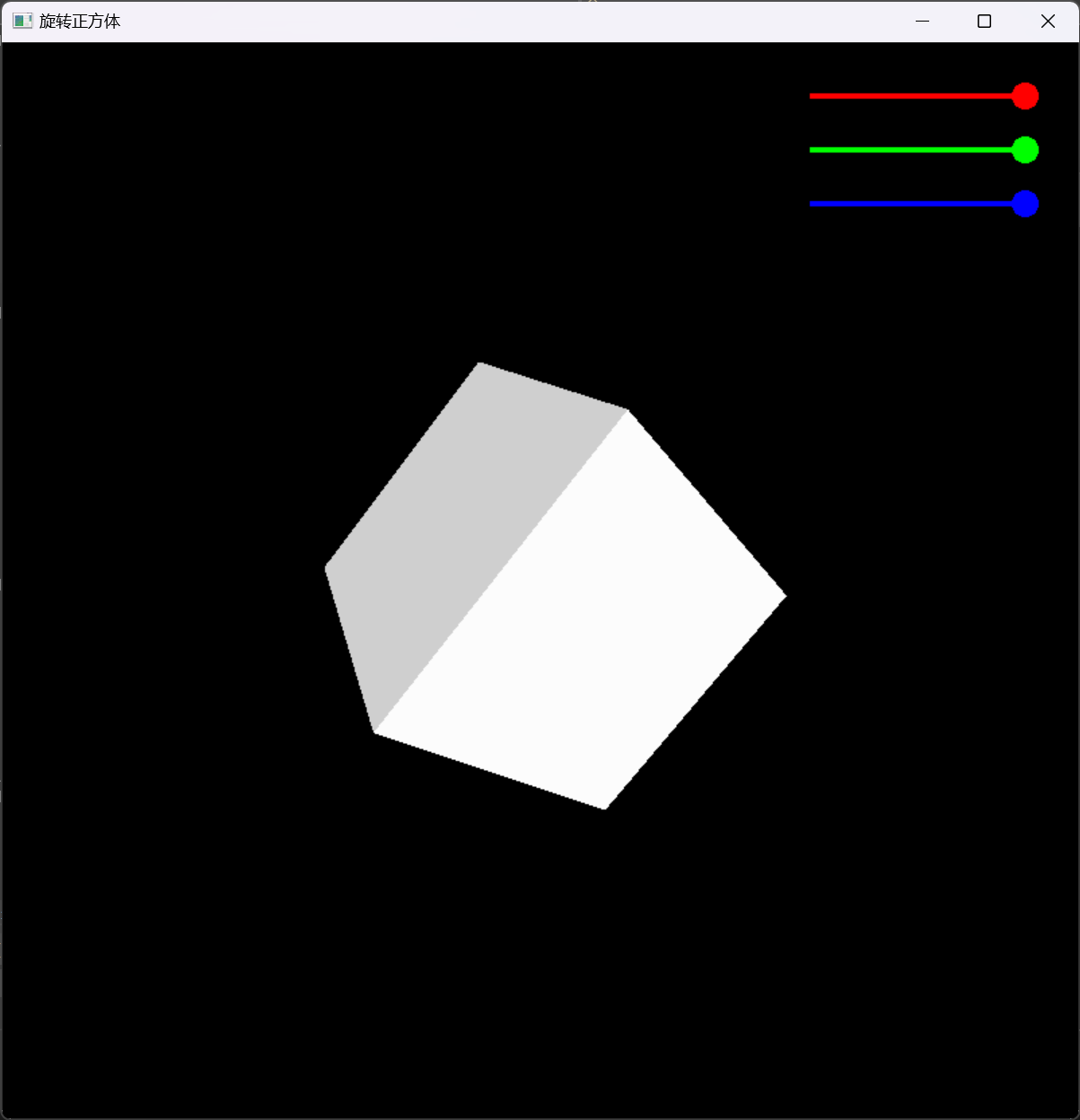
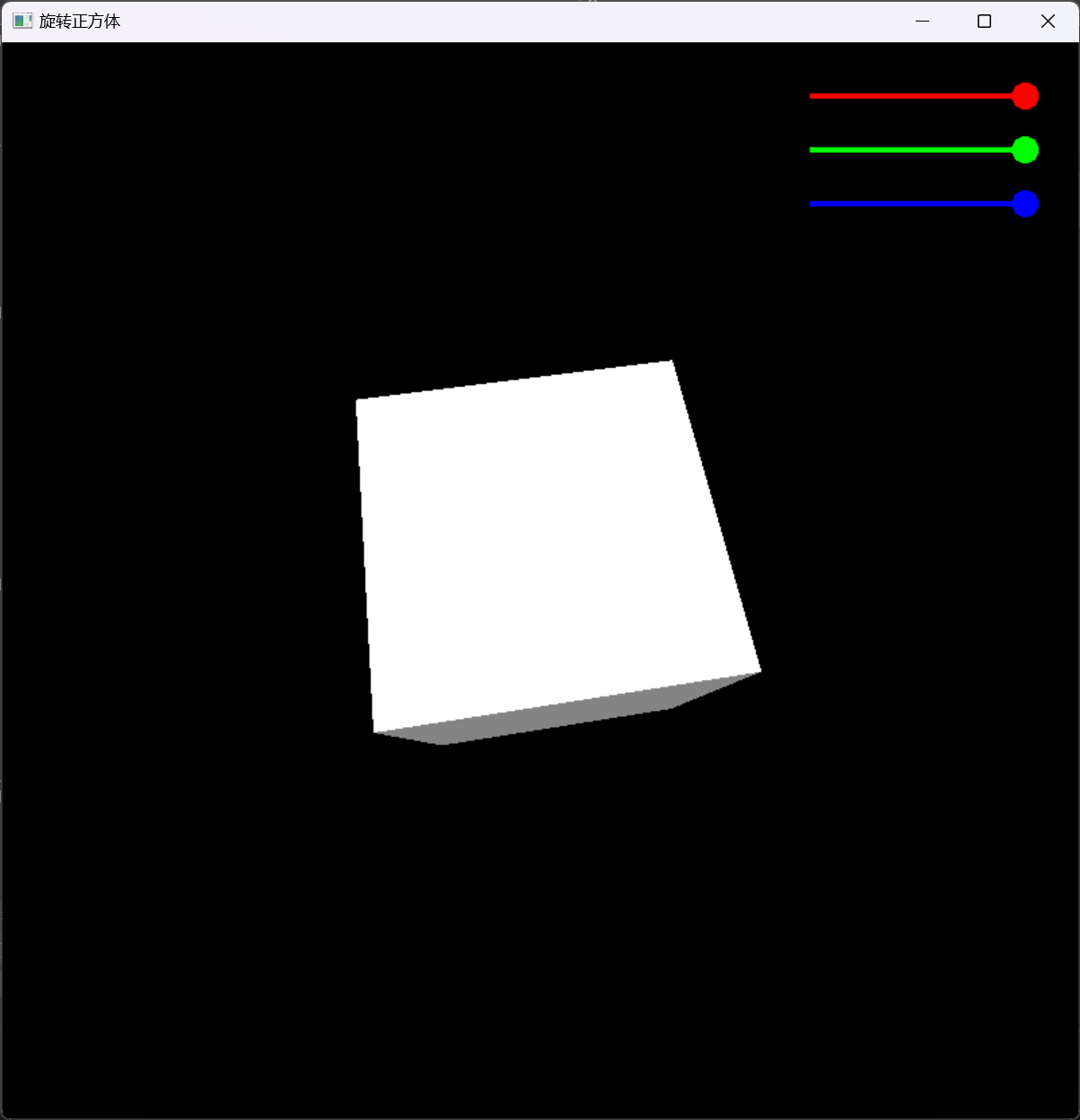
黄铜材质



红色塑料材质



亮白色材质



# 附加题：滑动条

## 4.1 思路

滑动条细节设计比较多，会出现较大的代码量，且会出现大量代码复用，因此为了清晰可以采用类封装

类包括以下属性：滑动条位置，滑动条的属性值，滑动条颜色

要实现以下函数接口：

1.初始化滑动条：根据所给坐标和颜色初始化滑动条

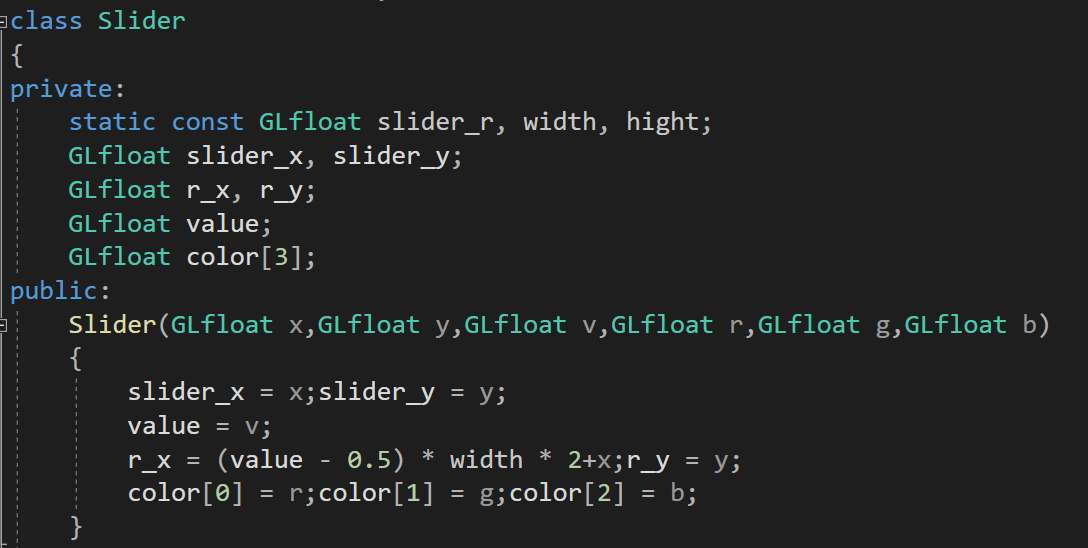
2.绘制滑动条：利用openGL在屏幕上显示滑动条，该函数需要被放置在display当中

3.滑动：需要根据鼠标的坐标修改滑动体的值，因此该函数需要和openGL的鼠标移动回调函数相结合

4.获取滑动值：需要根据滑动条的值去修改其他参数，因此需要获取滑动条的当前值

## 4.2 初始化滑动条

利用构造函数在创建滑动条对象时设定其颜色和位置



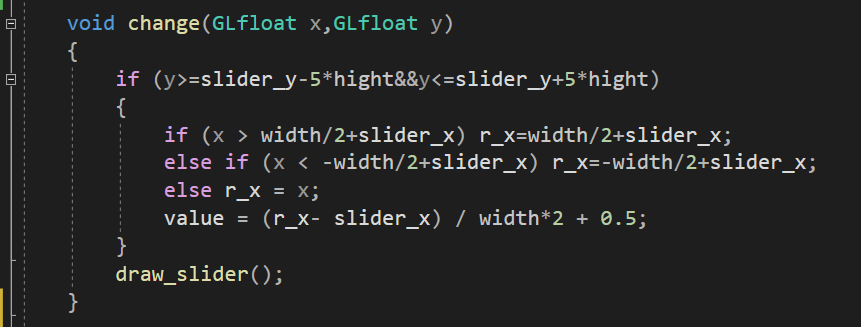
## 4.3绘制滑动条

绘制滑动条时希望使用颜色而不是光照，因此我们需要关闭光照，除此之外滑动条应该时固定在窗口的固定位置当中，不应该随着视角转动或者物体坐标变换而发生改变，因此我们需要将之前用于世界坐标变换的矩阵全部清空，避免对滑动条产生影响，同时为了保存之前的矩阵，可以使用矩阵堆栈同时实现这两个功能，在开始绘制滑动条前将世界坐标的矩阵入栈消除影响，在绘制完成后重新出栈对后续物体绘制继续发生作用

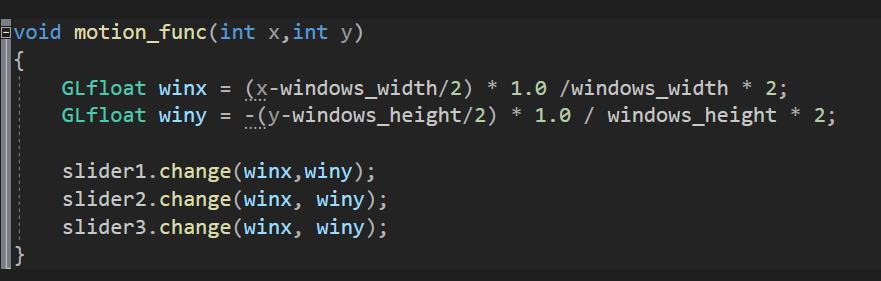


## 4.4实现滑动功能

滑动功能分为两个部分，第一个部分是实现value和鼠标的UI界面坐标相匹配，该部分中x，y为UI界面的坐标，也就是无论窗口大小如何，UI界面的横纵坐标都是从-1.0到1.0，该函数对坐标进行简单判断，确认在滑动条上则根据坐标修改value值



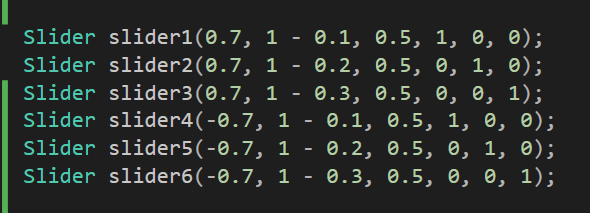
第二个部分为获取鼠标输入和将窗口坐标转换为UI界面坐标，需要注册一个鼠标点击移动的回调函数，之后进行窗口坐标转换，最后调用第一部分的函数



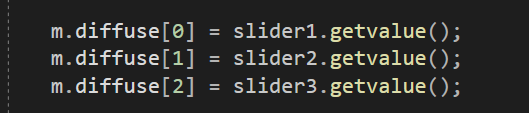
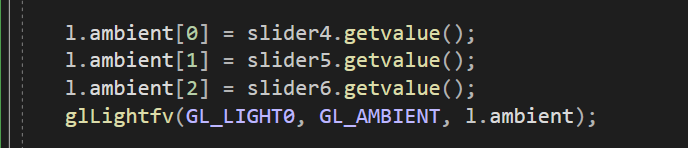
## 4.5 使用滑动条

创建并且绘制一个滑动条需要以下几个步骤

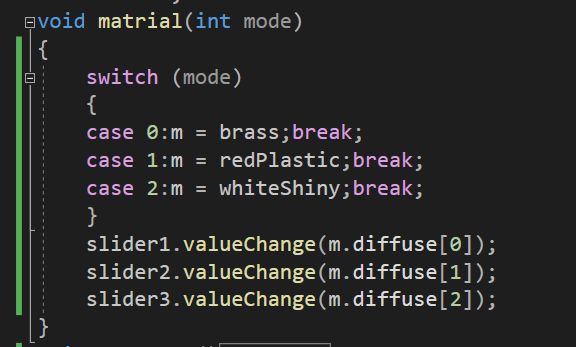
1.创建滑动条：需要实例化一个滑动条对象，给出其坐标和颜色，在代码中为了方便可以使用全局对象来实现，创建了总共6个滑动条，前三个表示物体材质漫反射表面的RGB属性，后三个代表光源环境光的RGB属性

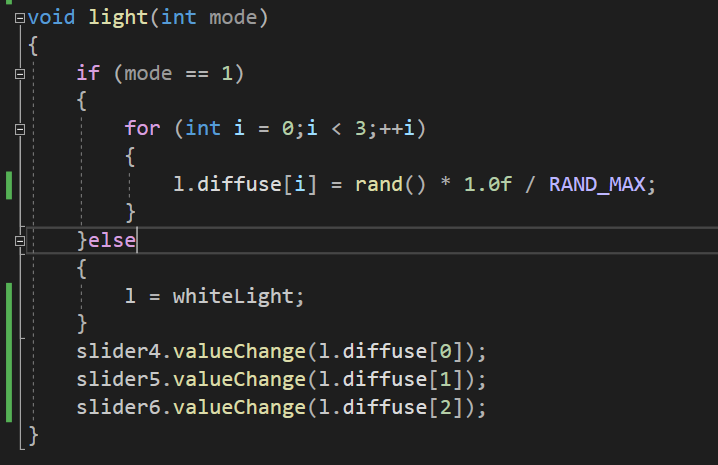


2.绑定滑动条和所需要修改的参数：该代码会在每次绘制时通过滑动条当前的值修改光源和材质的属性，需要将value和对应的参数先进行绑定，该部分一般放置在display中

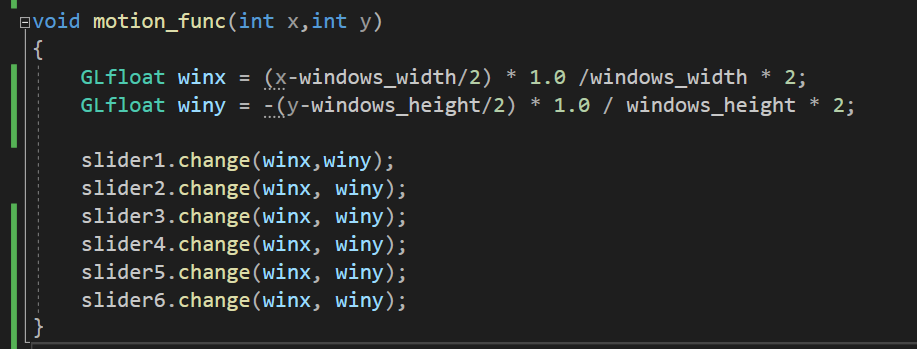


除此之外代码中由出现使用键盘直接控制材质和光源，因此也要进行反向绑定，即通过材质和光照的值修改value值，该部分一般放置在键盘控制修改材质和光照后

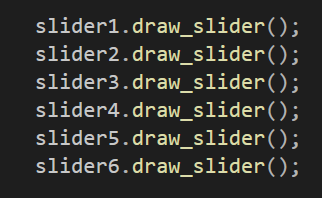




3.注册滑动条拖动：通过鼠标的坐标修改滑动条的值，该部分一般放置在鼠标移动的回调函数

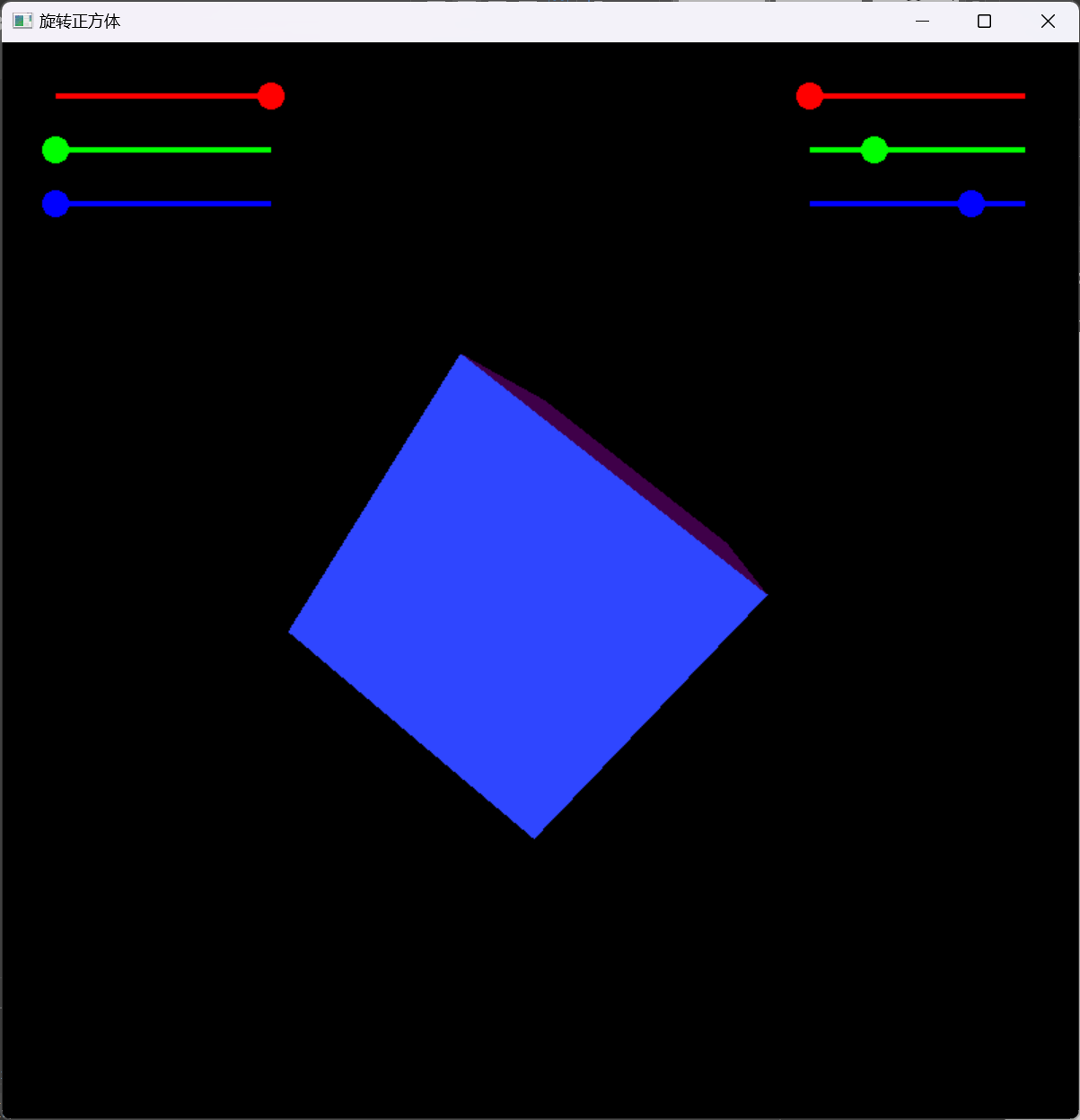
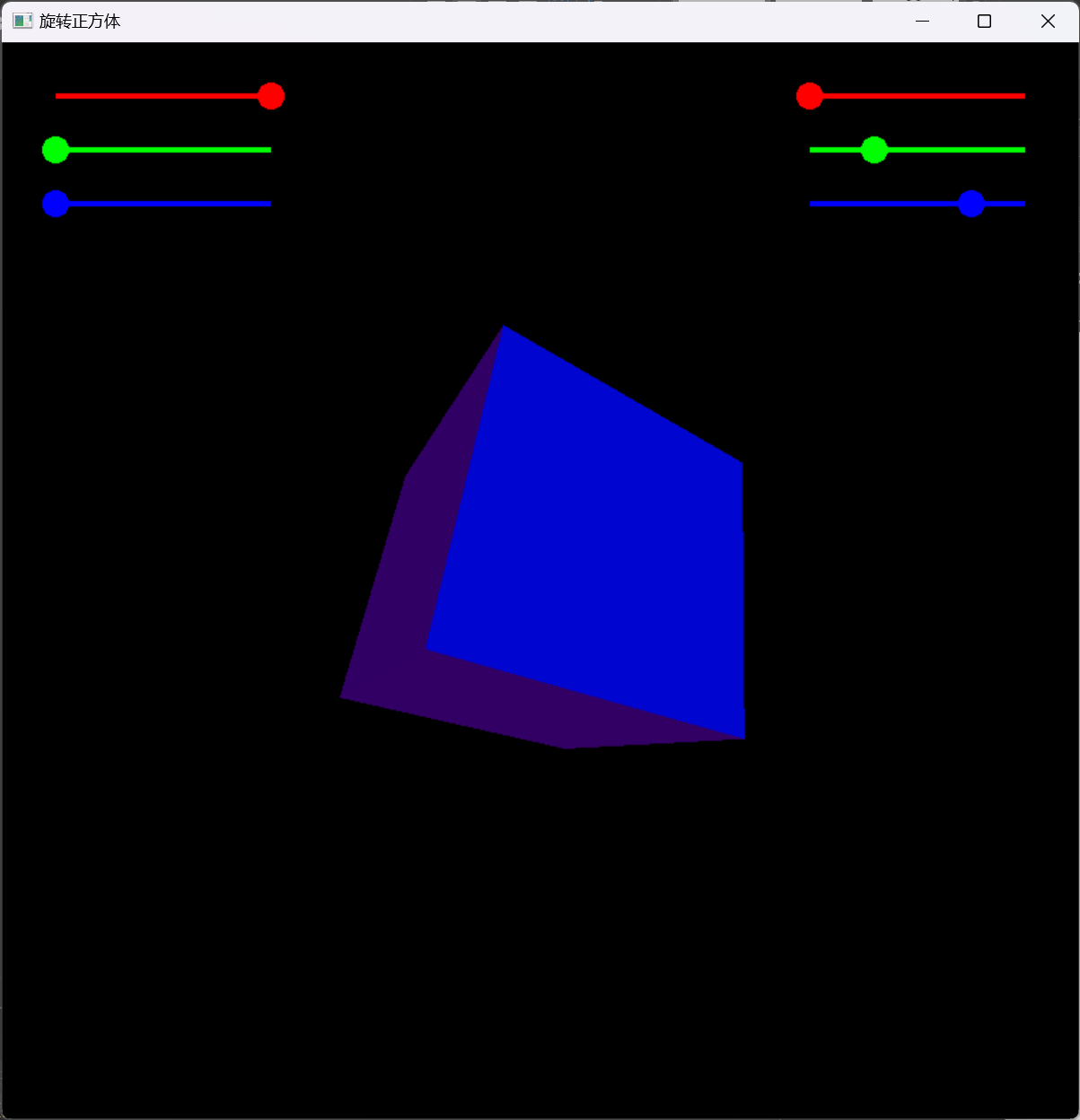


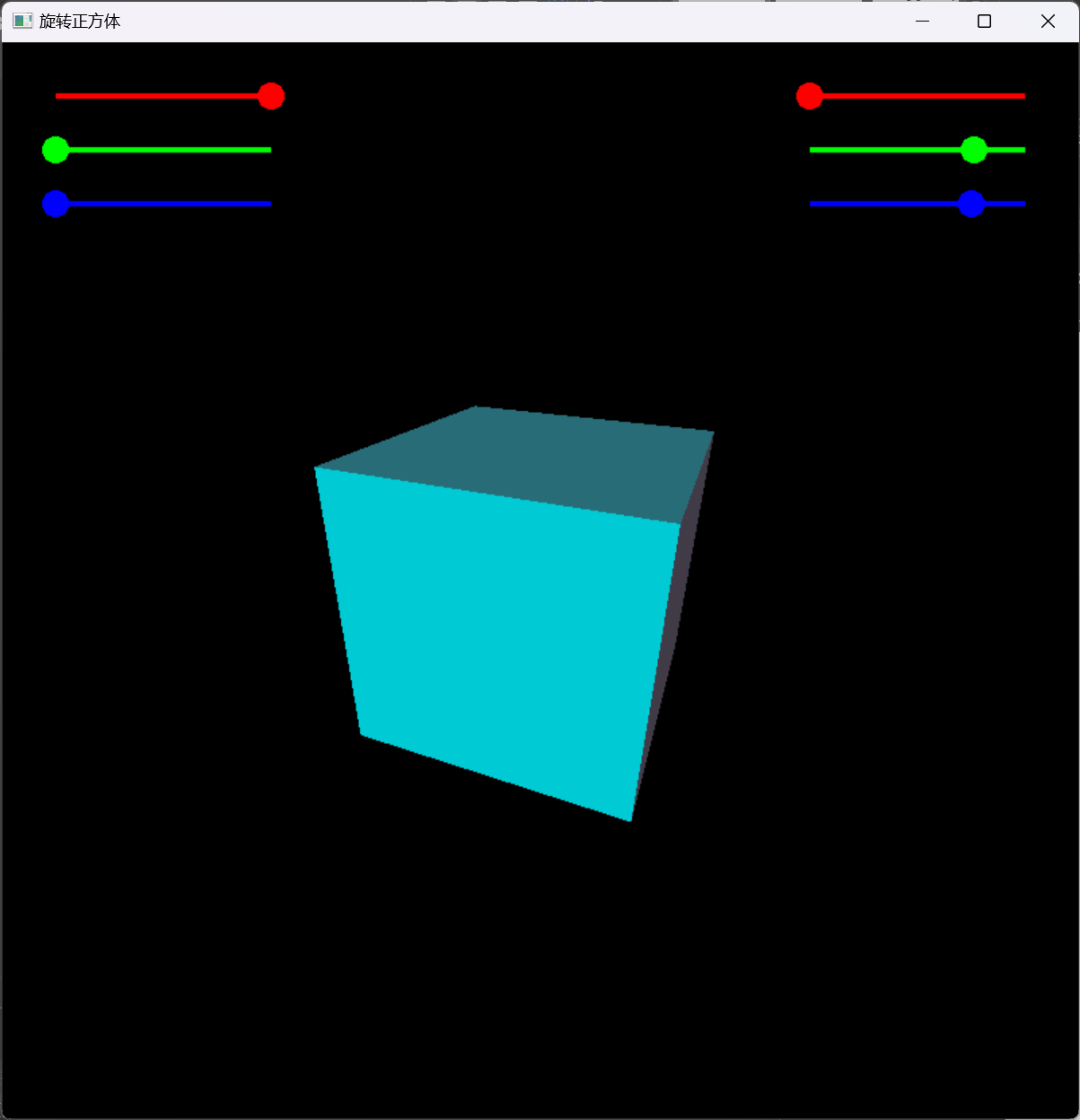
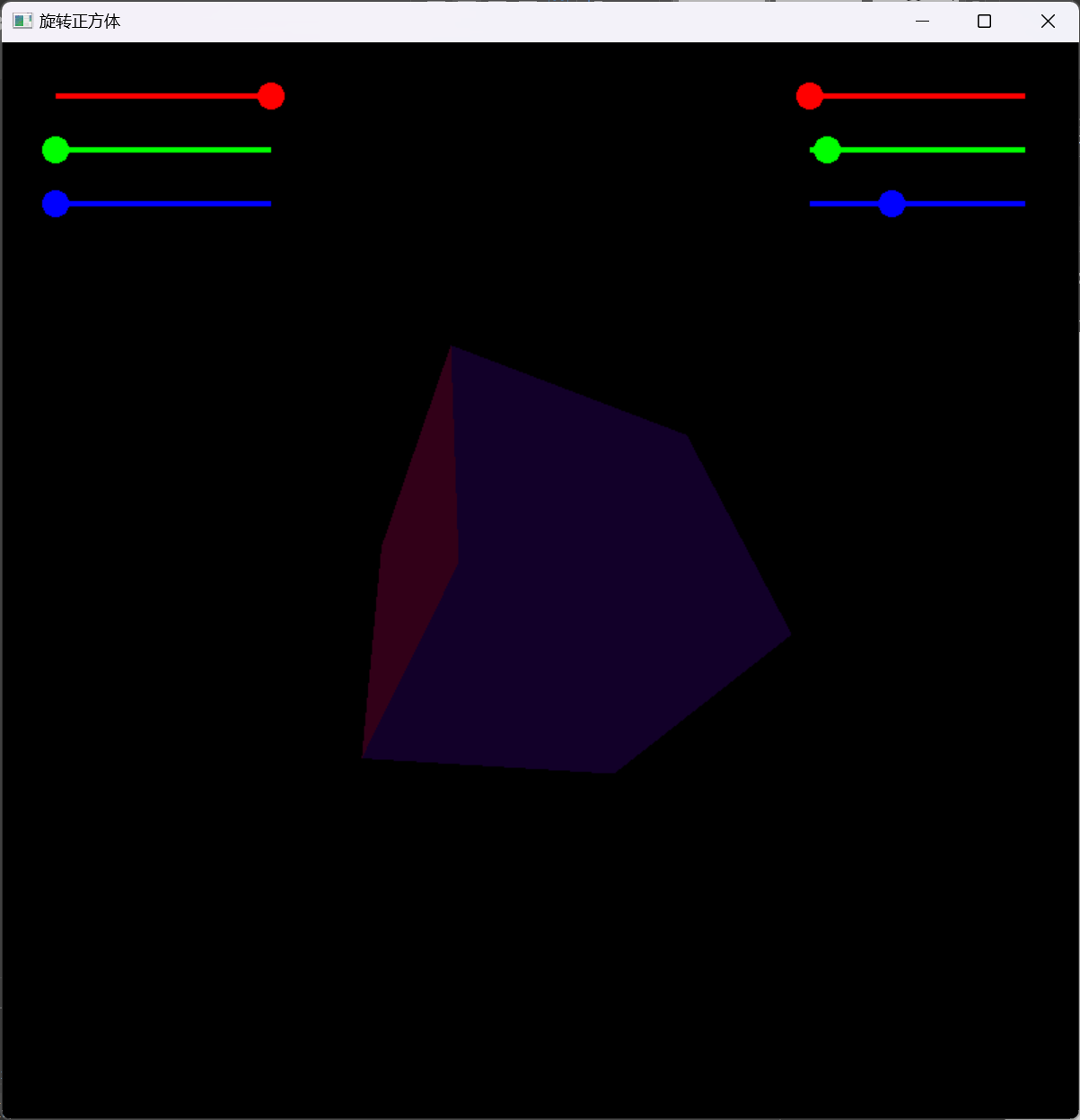
4.绘制滑动条：放置在display中即可



实现效果：左边三个滑动条控制环境光，右边三个控制漫反射表面

红光下切换物体材质：可以看到面向光源的一侧（背面）有泛红





黄铜材质在不同色光下呈现不同颜色：

