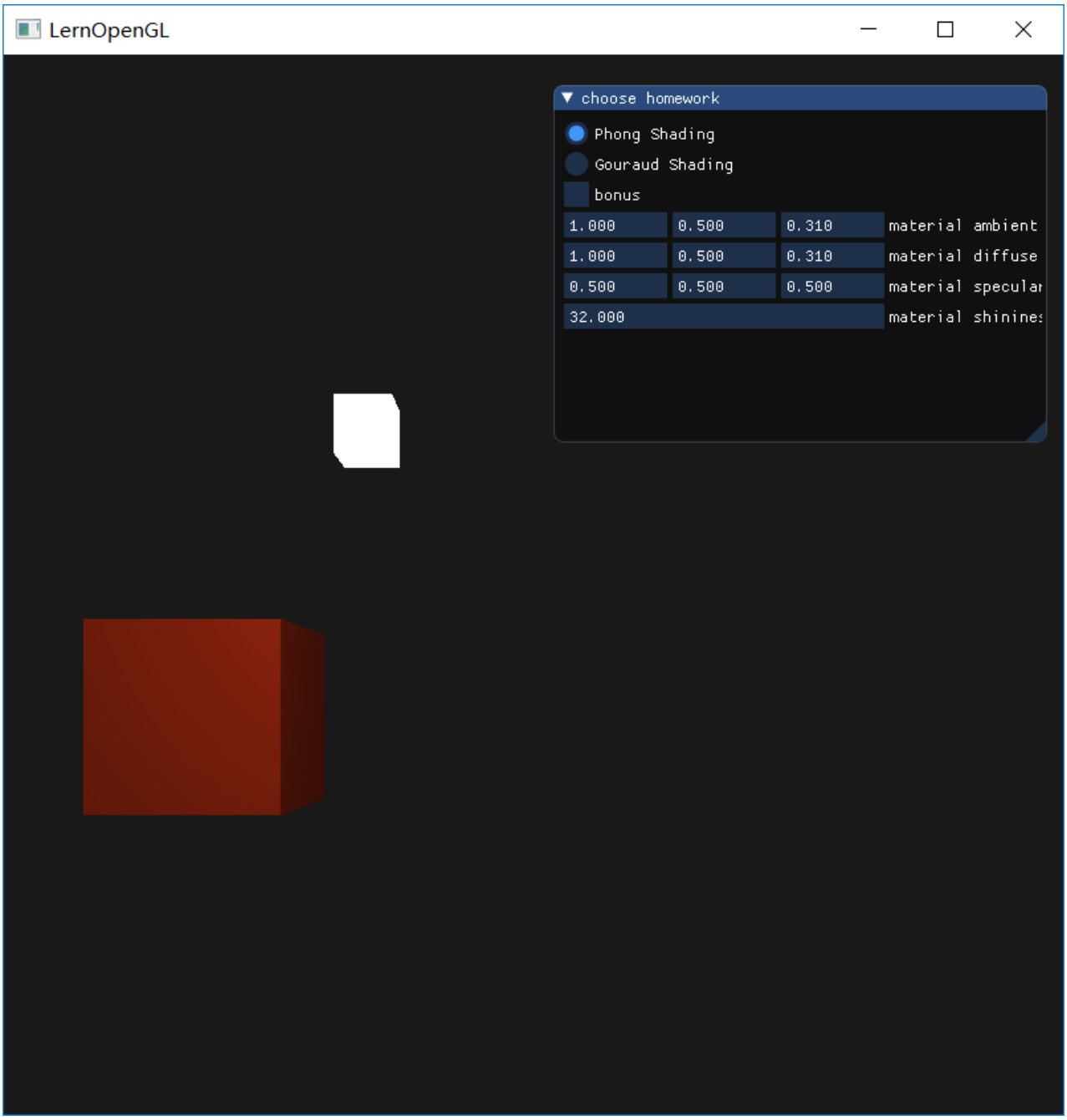
实验报告

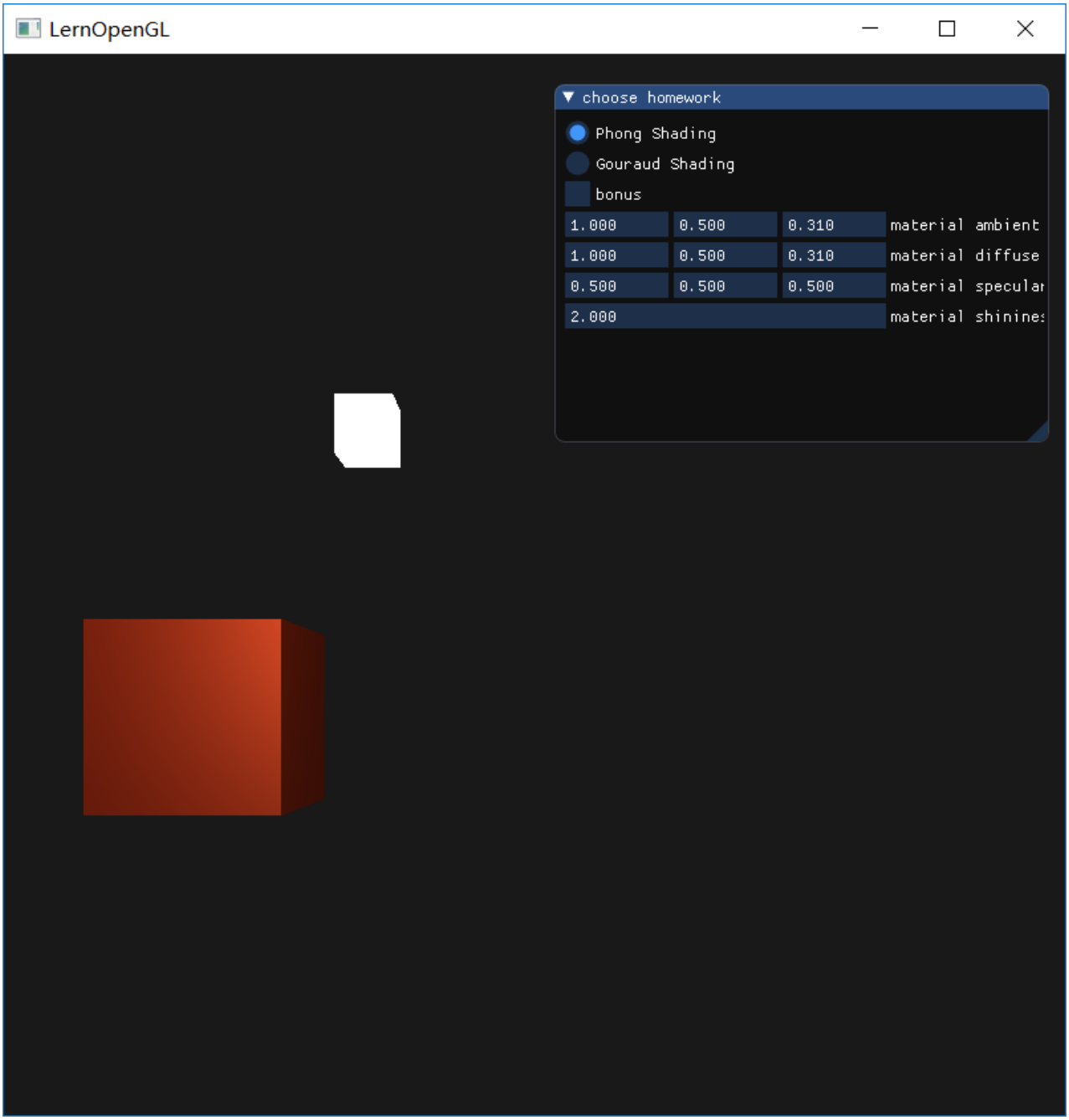
1. Basic

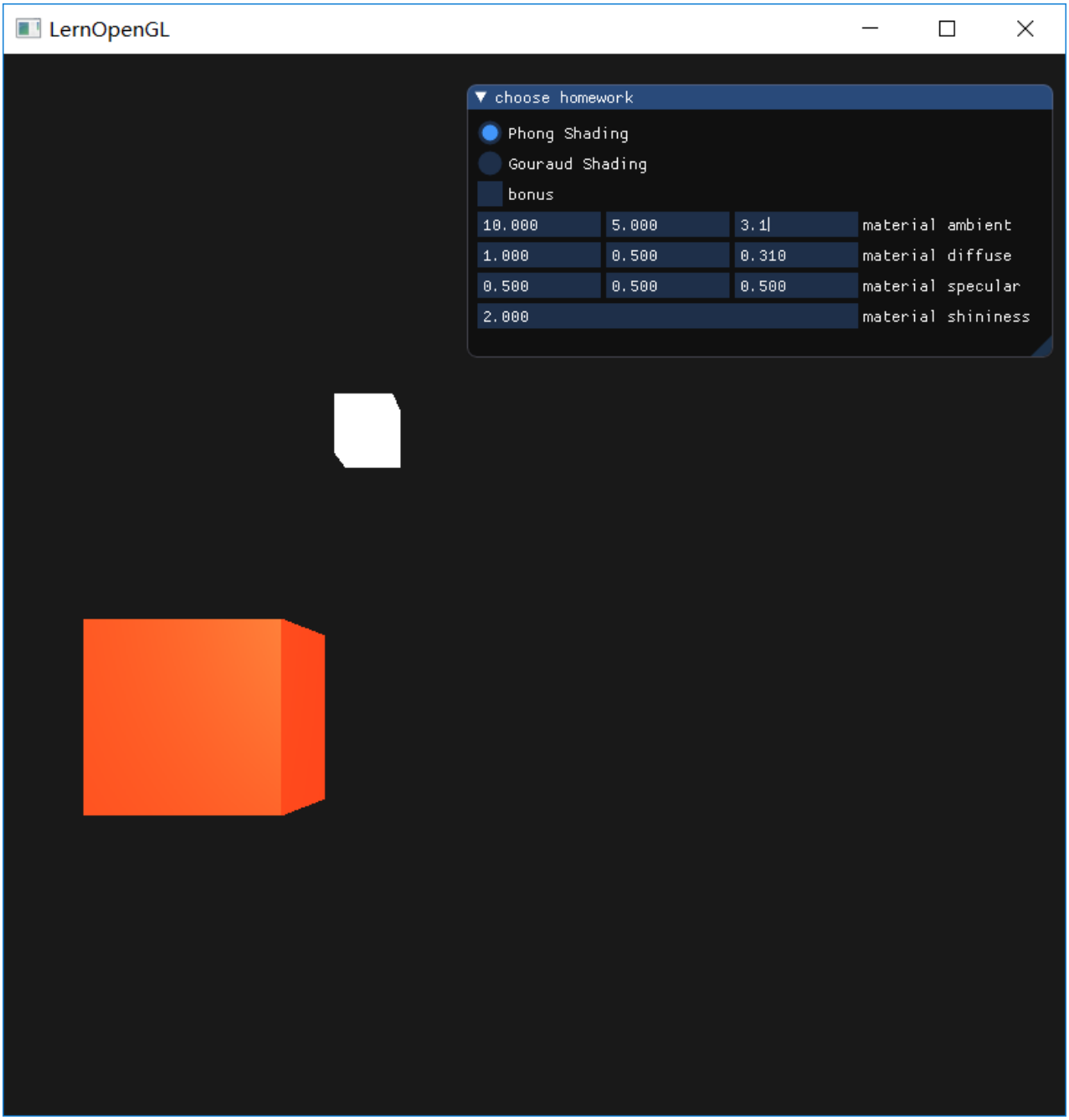
1. Phong Shading

1. 实验结果



修改参数:


LernOpenGL
choose homework
Phong Shading
Gouraud Shading
bonus
1.000
0.500
0.310
material ambient
1.000
0.500
0.310
material diffuse
0.500
0.500
0.500
material specular
2.000
material shinines



2. 实现原理
    1. 更改顶点数据：在数组内增加法向量数据
    2. 存入缓存：顶点数据和法向量数据分别用函数 glVertexAttribPointer 传入缓存
    3. 编写物体顶点着色器：
        1. 传入顶点位置和法向量的值
        2. 通过 `mat3(transpose(inverse(model))) * aNormal` 获得法向量，并将这个值作为结果传给片段着色器

            ```
            Normal = mat3(transpose(inverse(model))) * aNormal;
            ```

        3. 其余内容和之前编写的顶点着色器一样
    4. 编写物体的片段着色器

1. 传入内容包括：
    1. 顶点着色器传入的位置信息和法向量
    2. 用户传入的视点位置、物体的散射光材质系数K_a 、物体的漫反射系数K_d 、物体的镜面反射系数K_s 、物体的材质发光系数n_{shiny}
2. 通过公式 $I_{reflected} = K_a I_a$ 获得散射后的强度

```
vec3 ambient = lightAmbient * materialAmbient;
```

3. 通过公式 $I_{diffuse} = K_d I_{light}(\vec{n} \cdot \vec{l})$ 获得漫反射后的强度，其中 $\vec{n}$ 是法向量，$\vec{l}$ 是入射光的方向

```
vec3 norm = normalize(Normal);
vec3 lightDir = normalize(lightPosition - FragPos);
float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = lightDiffuse * (diff * materialDiffuse);
```

4. 通过公式 $I_{specular} = K_s I_{light}(\vec{v} \cdot \vec{r})^{n_{shiny}}$ 获得镜面反射的强度，其中 $\vec{v}$ 是观察者和物体的角度，$\vec{r}$ 是理论上反射角，可以直接使用自带的reflect函数得到

```
vec3 viewDir = normalize(viewPos - FragPos);
vec3 reflectDir = reflect(-lightDir, norm);
float spec = pow(max(dot(viewDir, reflectDir), 0.0), materialShininess);
vec3 specular = lightSpecular * (spec * materialSpecular);
```

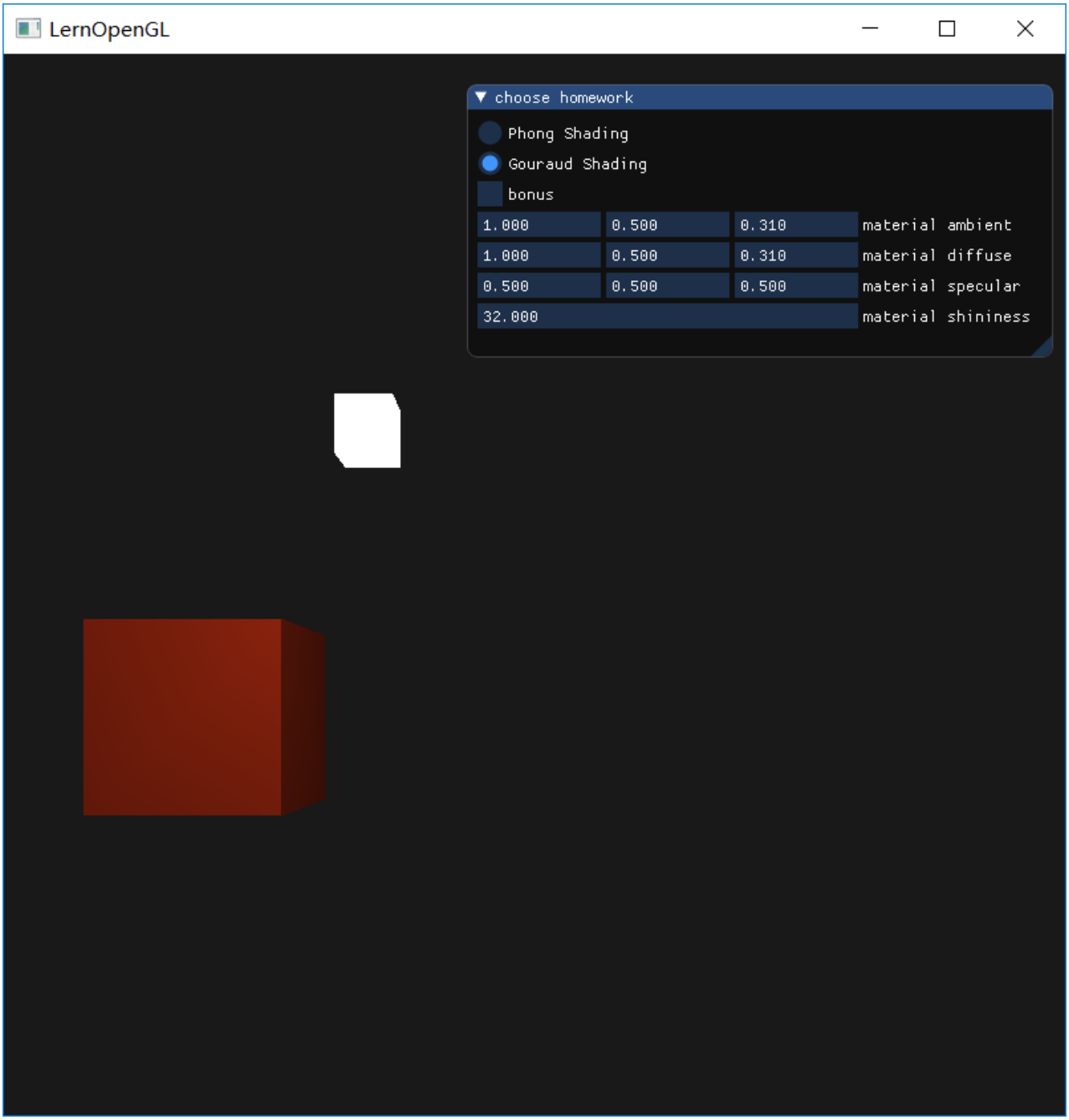
5. 结果为$I = I_{reflected} + I_{diffuse} + I_{specular}$

```
vec3 result = (ambient + diffuse + specular) * objectColor;
```

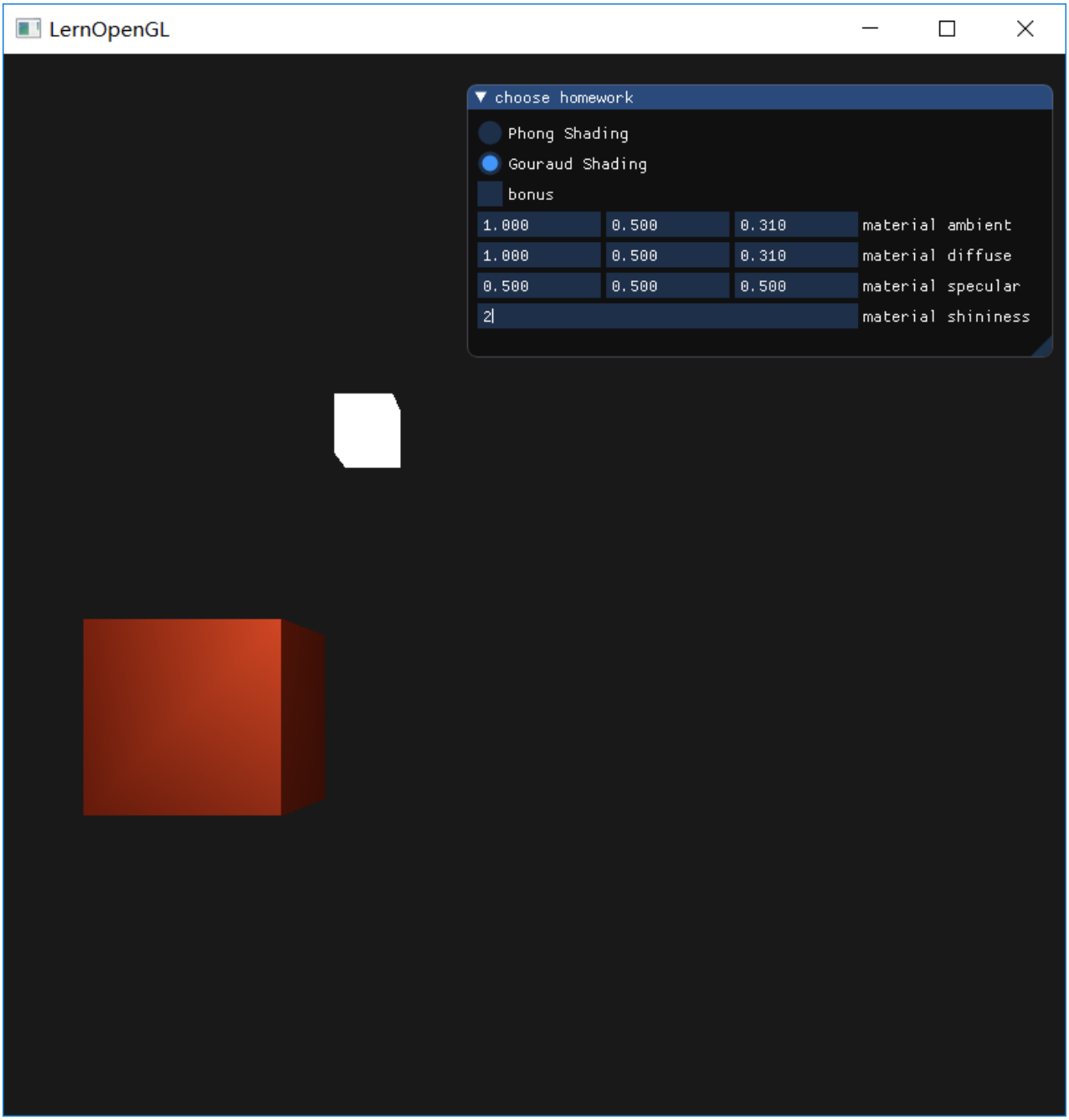
5. 编写光源的着色器：和之前作业的着色器相同。

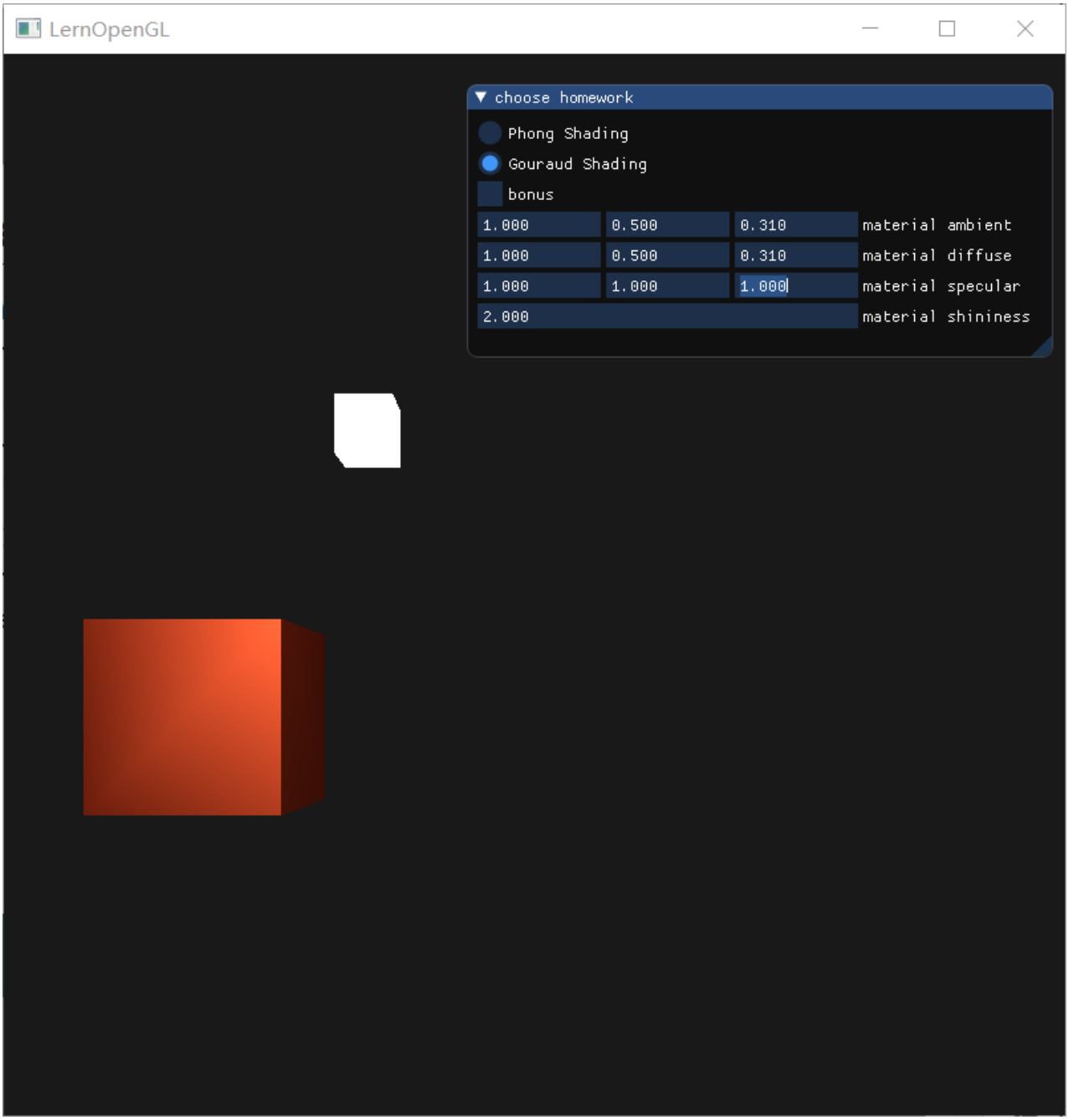
2. Gouraud Shading

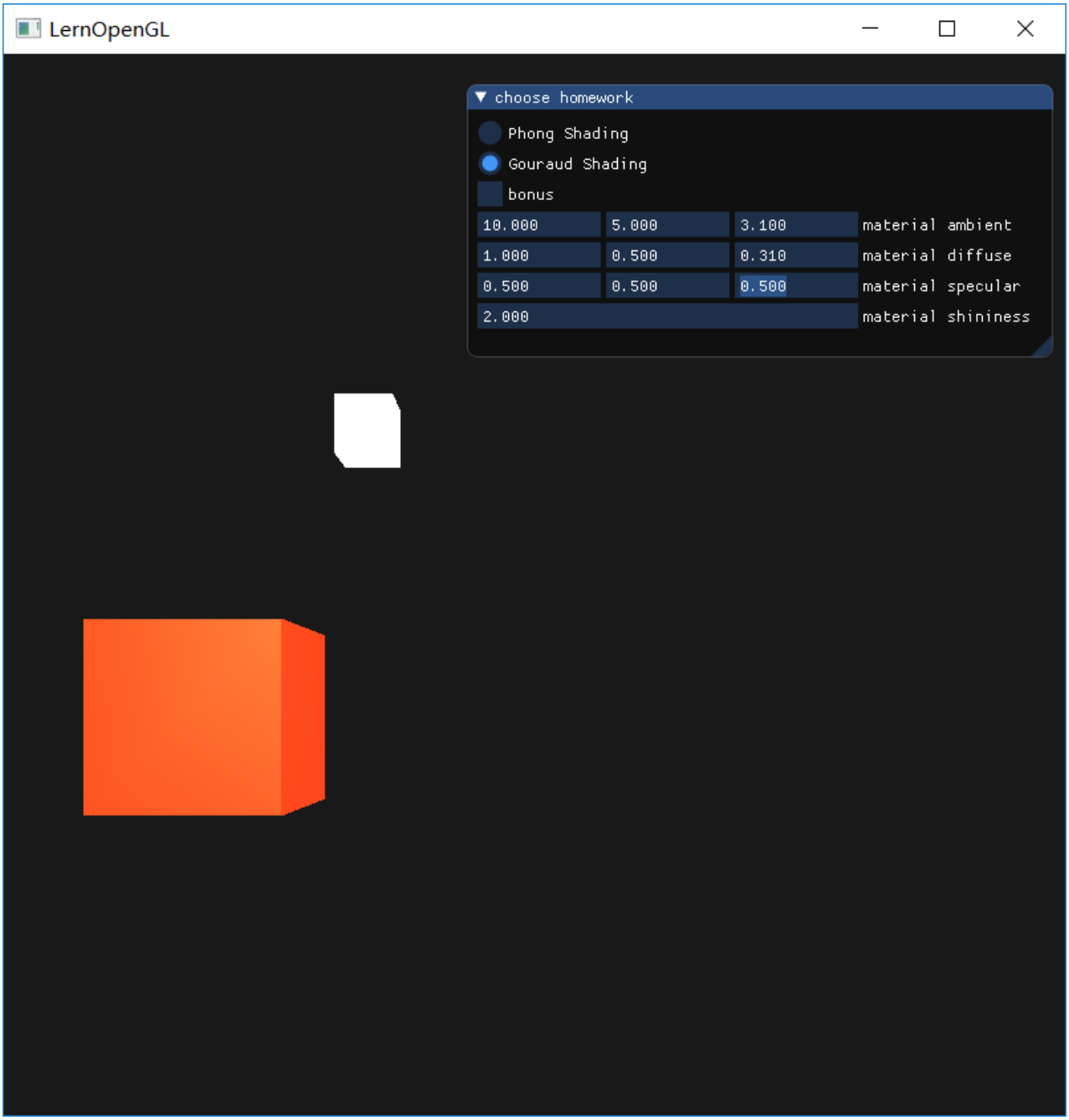
1. 实验结果


LernOpenGL
choose homework
Phong Shading
Gouraud Shading
bonus
1.000
0.500
0.310
material ambient
1.000
0.500
0.310
material diffuse
0.500
0.500
0.500
material specular
32.000
material shininess

更改参数:


LernOpenGL
choose homework
Phong Shading
Gouraud Shading
bonus
1.000
0.500
0.310
material ambient
1.000
0.500
0.310
material diffuse
0.500
0.500
0.500
material specular
2
material shininess


LernOpenGL
choose homework
Phong Shading
Gouraud Shading
bonus
1.000
0.500
0.310
material ambient
1.000
0.500
0.310
material diffuse
1.000
1.000
1.000
material specular
2.000
material shininess



在这个状态下不容易看出两种shading的区别，在bonus有更容易看出区别的角度

2. 实现原理
    1. 顶点数据等和前一个一样
    2. 编写顶点着色器
        1. 传入内容包括：
            1. 用户传入的视点位置、物体的散射光材质系数K_a、物体的漫反射系数K_d、物体的镜面反射系数K_s、物体的材质发光系数n_{shiny}
        2. 通过 `mat3(transpose(inverse(model))) * aNormal` 获得法向量

            ```
            Normal = mat3(transpose(inverse(model))) * aNormal;
            ```

        3. 通过公式 $I_{reflected} = K_a I_a$ 获得散射后的强度

```
vec3 ambient = lightAmbient * materialAmbient;
```

4. 通过公式 $I_{diffuse} = K_d I_{light}(\vec{n} \cdot \vec{l})$ 获得漫反射后的强度，其中 $\vec{n}$ 是法向量，$\vec{l}$ 是入射光的方向

```
vec3 norm = normalize(Normal);
vec3 lightDir = normalize(lightPosition - FragPos);
float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = lightDiffuse * (diff * materialDiffuse);
```

5. 通过公式 $I_{specular} = K_s I_{light}(\vec{v} \cdot \vec{r})^{n_{shiny}}$ 获得镜面反射的强度，其中 $\vec{v}$ 是观察者和物体的角度，$\vec{r}$ 是理论上反射角，可以直接使用自带的reflect函数得到

```
vec3 viewDir = normalize(viewPos - FragPos);
vec3 reflectDir = reflect(-lightDir, norm);
float spec = pow(max(dot(viewDir, reflectDir), 0.0), materialShininess);
vec3 specular = lightSpecular * (spec * materialSpecular);
```

6. 结果为$I = I_{reflected} + I_{diffuse} + I_{specular}$

```
vec3 result = (ambient + diffuse + specular) * objectColor;
```

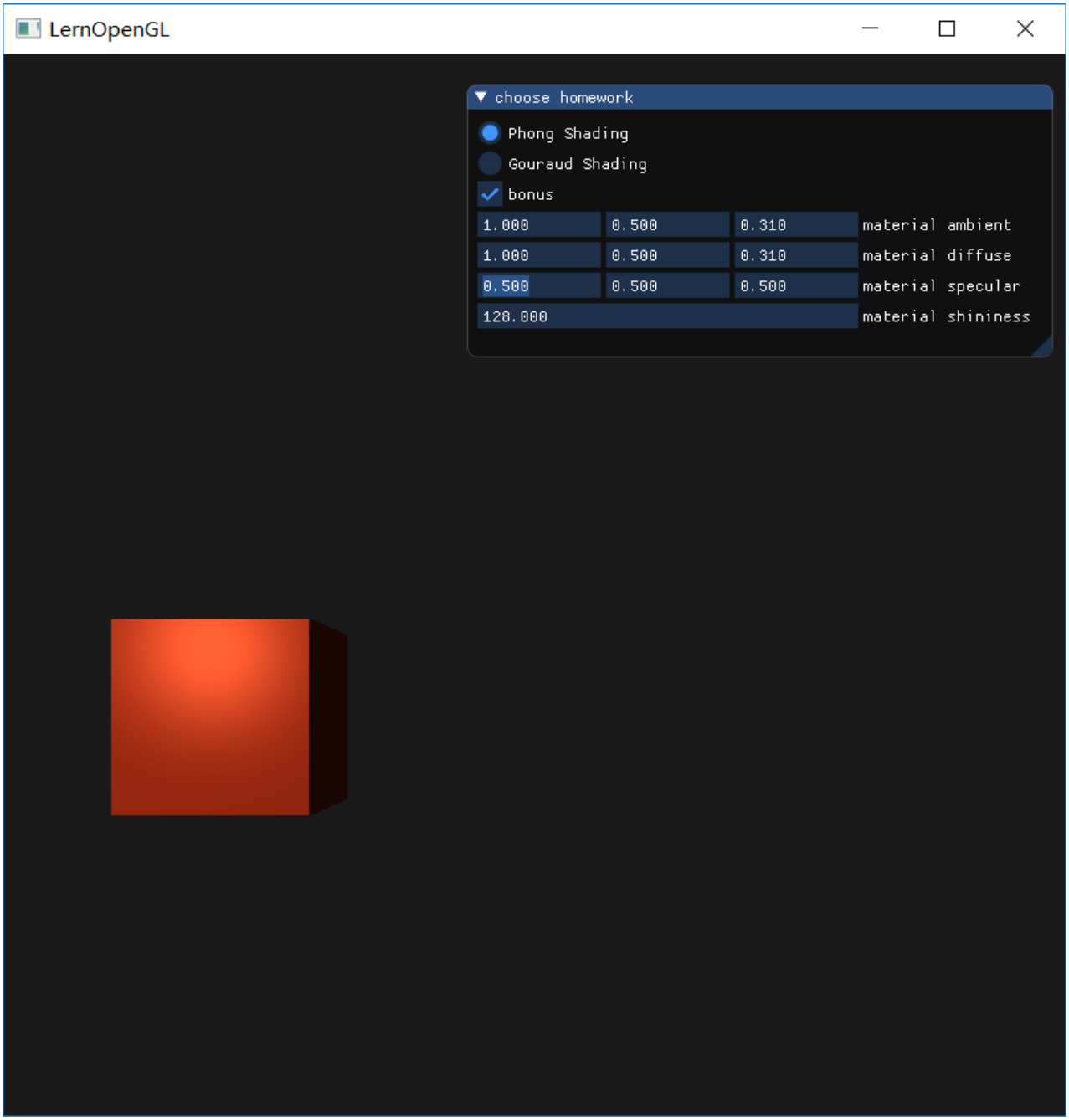
3. 编写片段着色器

```
FragColor = vec4(LightingColor * objectColor, 1.0);
```

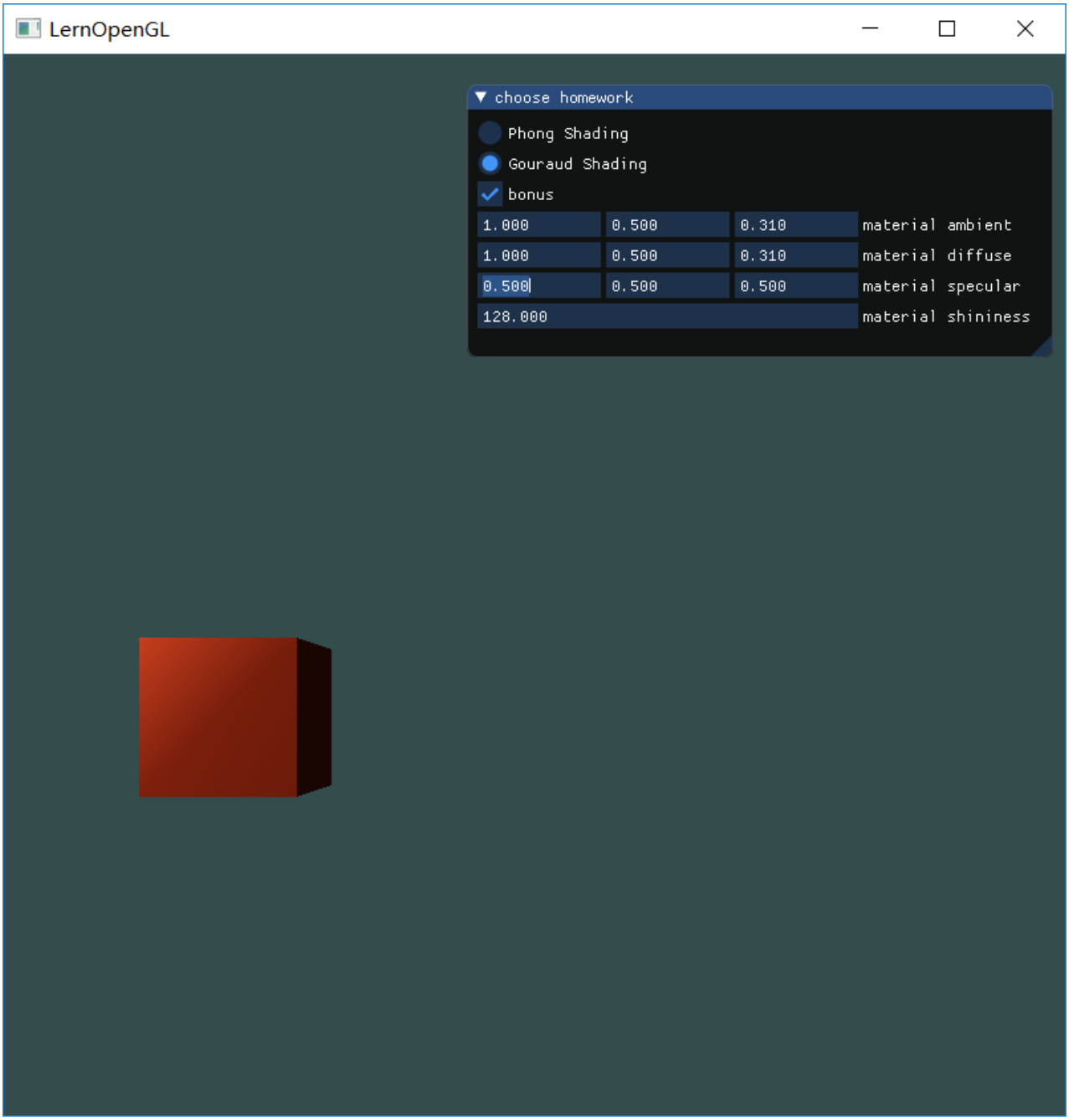
2. Bonus

1. 实验结果

   phong shading



Gouraud Shading



这里出现黑色的原因是环境光的系数很小导致的

2. 实现过程

    1. 设置光源的位置

```
lightPos.x = 1.0f + sin(glfwGetTime()) * 2.0f;
lightPos.y = sin(glfwGetTime() / 2.0f) * 1.0f;
```