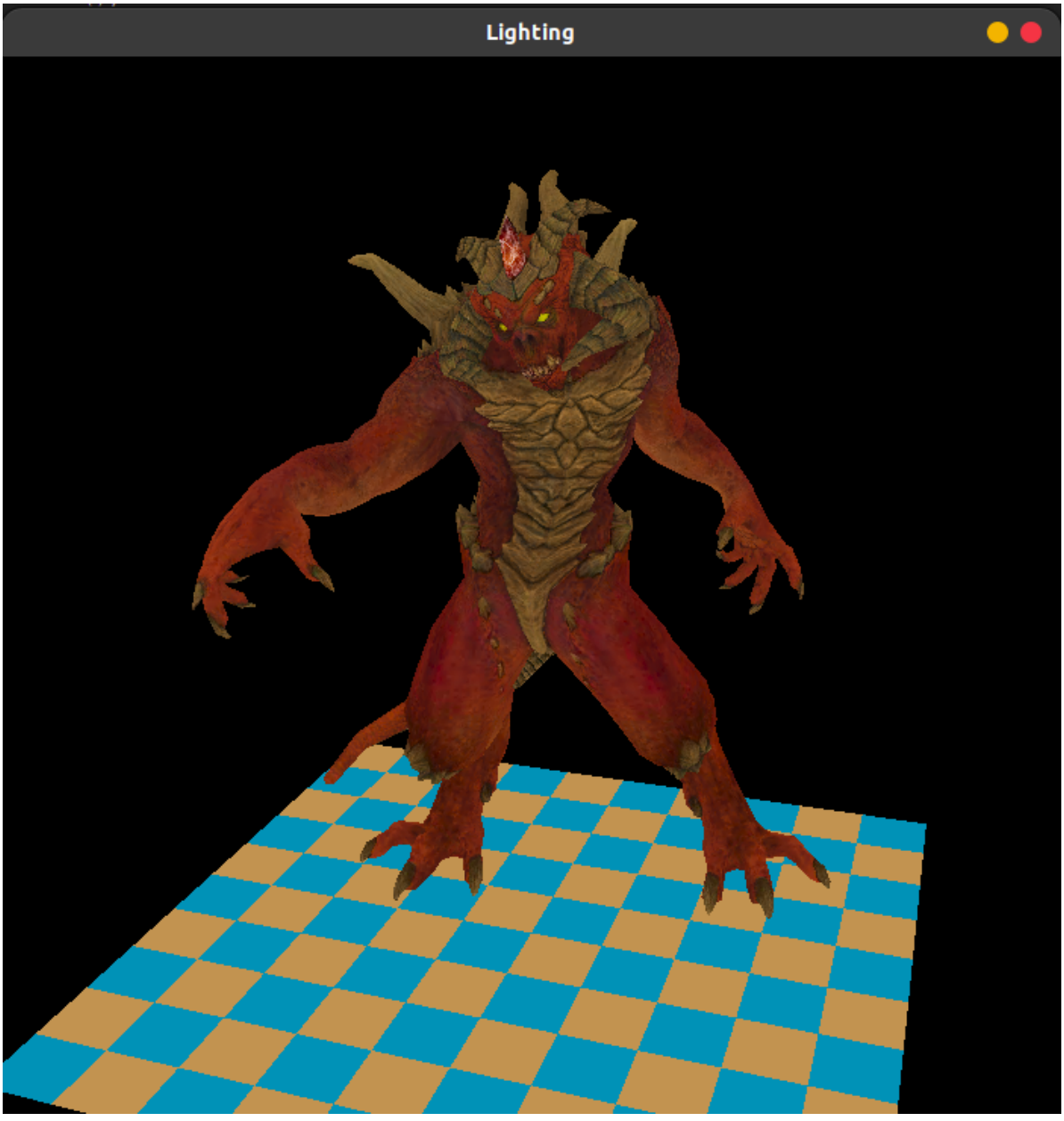
计算机图形学实验4

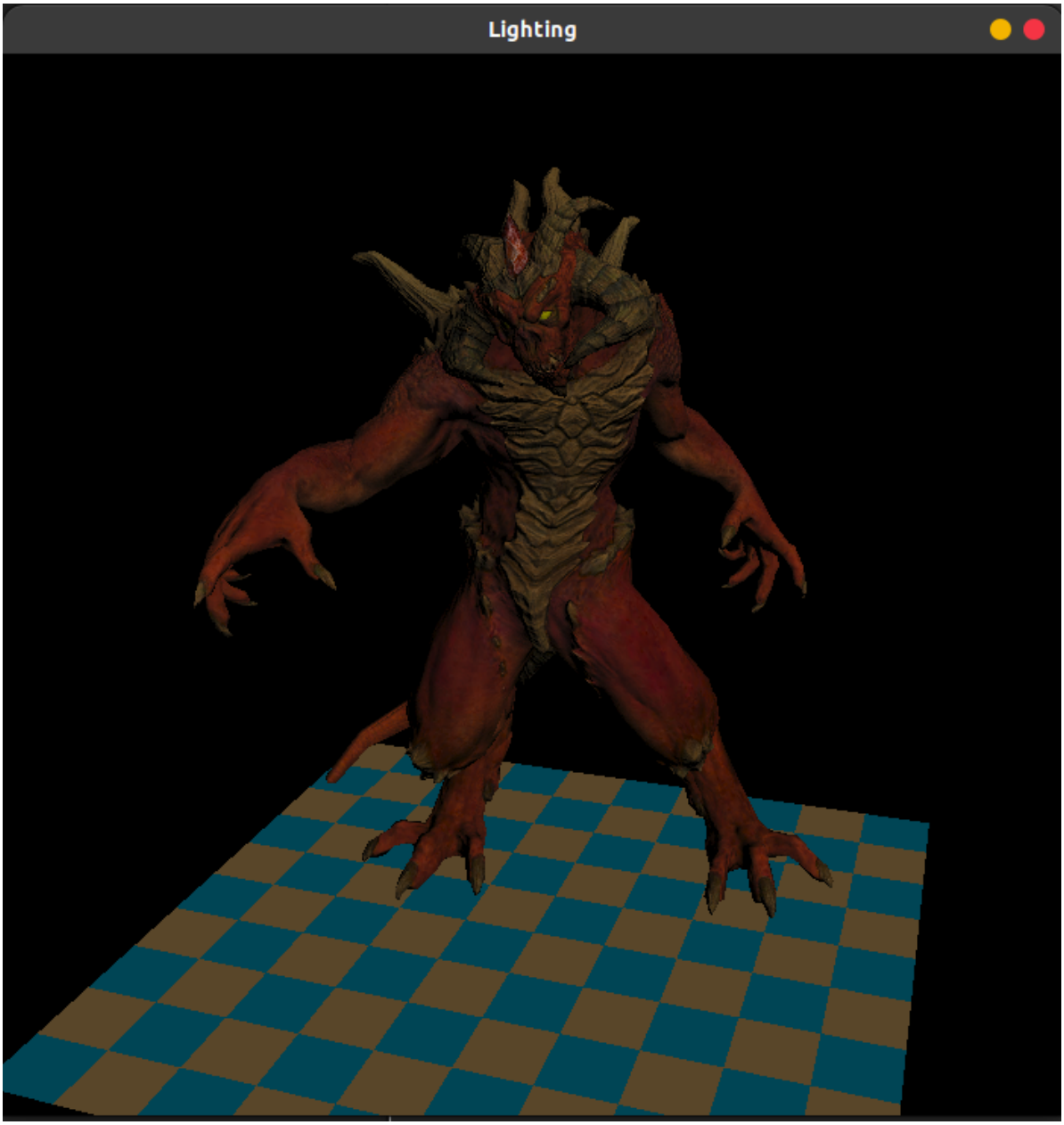
姓名： 杨伯宇　　学号：18340189

刚开始的运行结果为



Task 1、在 Shader 类中的 fragment 函数中实现漫反射光照的计算逻辑。

结果为



实现思路为

计算法线和入射光线夹角的余弦值，和0取最大值，再乘漫反射系数$k_d = 0.8$，得到最终结果，代码如下

```
        // TODO 1:
        // calculate the diffuse coefficient using Phong model, and then save it
to diff.
        {
            diff = normal * lightDir;
            diff = (diff > 0) ? diff : 0;
            diff *= 0.8;
        }
```

Task 2、在 Shader 类中的 fragment 函数中实现Phong的镜面光照计算逻辑。

首先要计算出反射光线，通过几何关系可以得到，
$reflectDir - normal * \cos(\theta) = normal * \cos(\theta) - lightDir$,其中$\cos(\theta) = normal * lightDir$

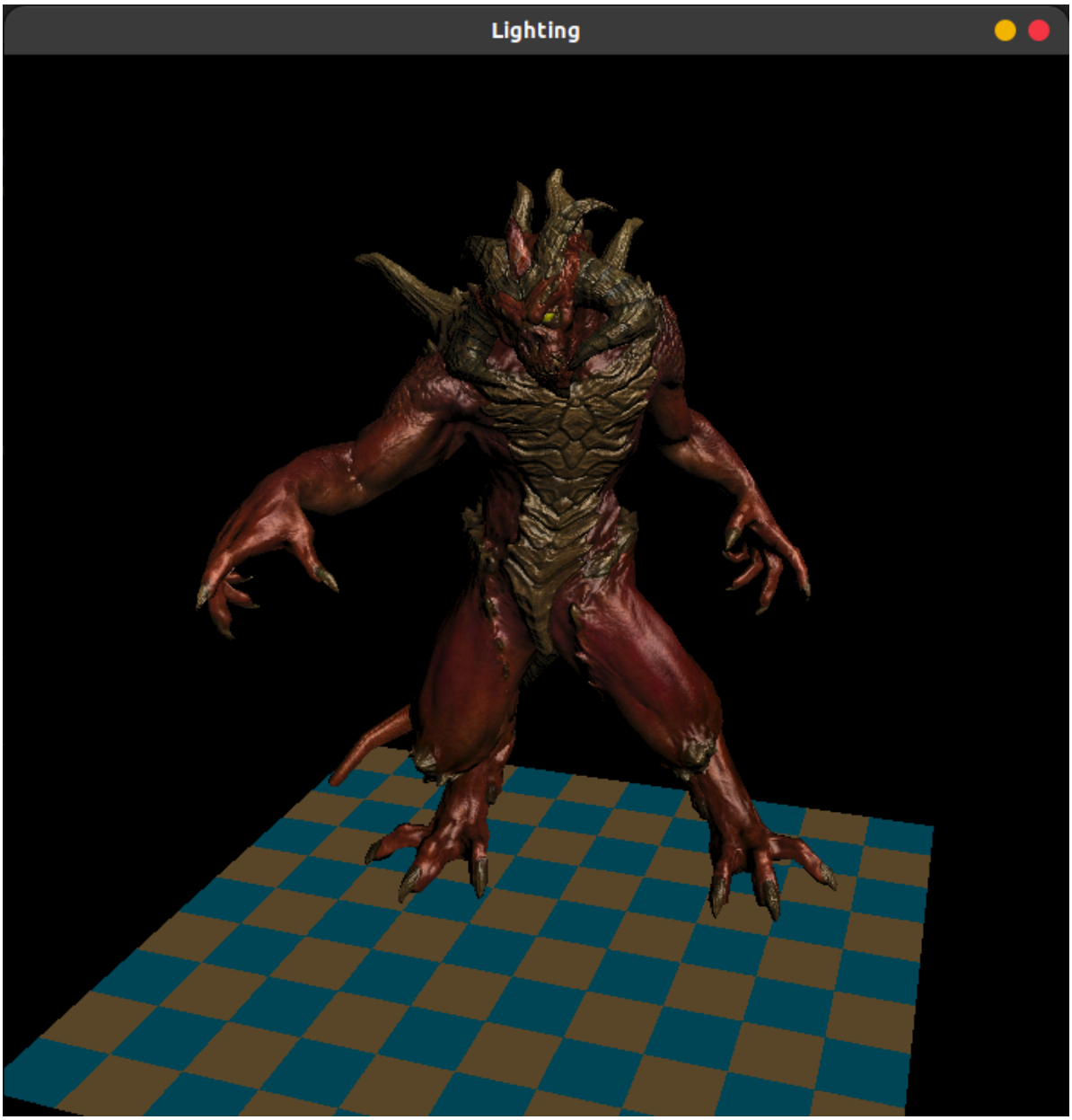
再加上高光系数p和镜面反射系数，就可以得到结果

代码如下

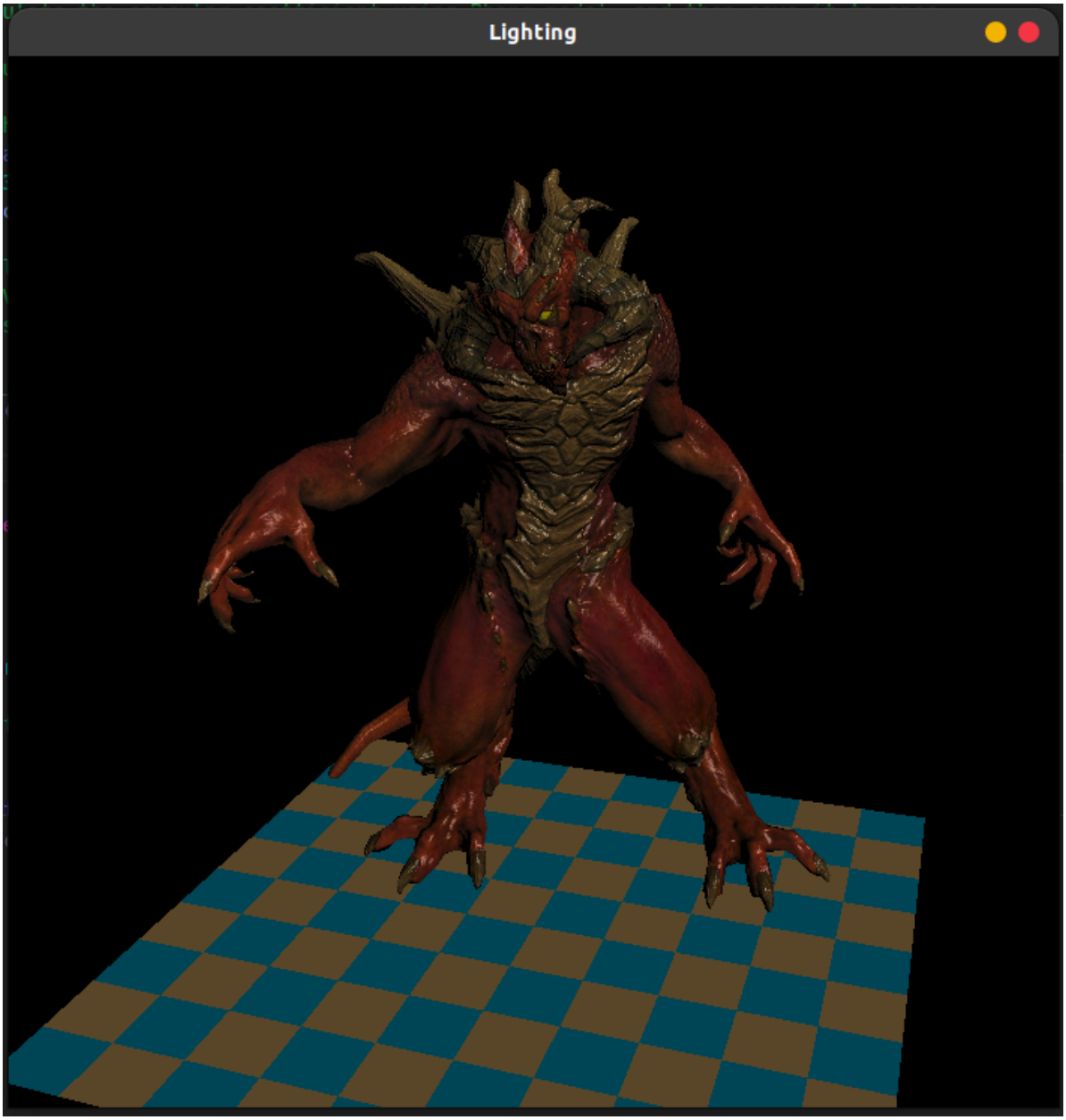
```
float cos_theta = normal * lightDir;
Vec3f reflectDir = normal * 2 * cos_theta - lightDir;
spec = reflectDir.normalize() * viewDir;
spec = (spec > 0) ? 0.4 * pow(spec, 64) : 0;
```

使用不同的高光系数p来观察结果

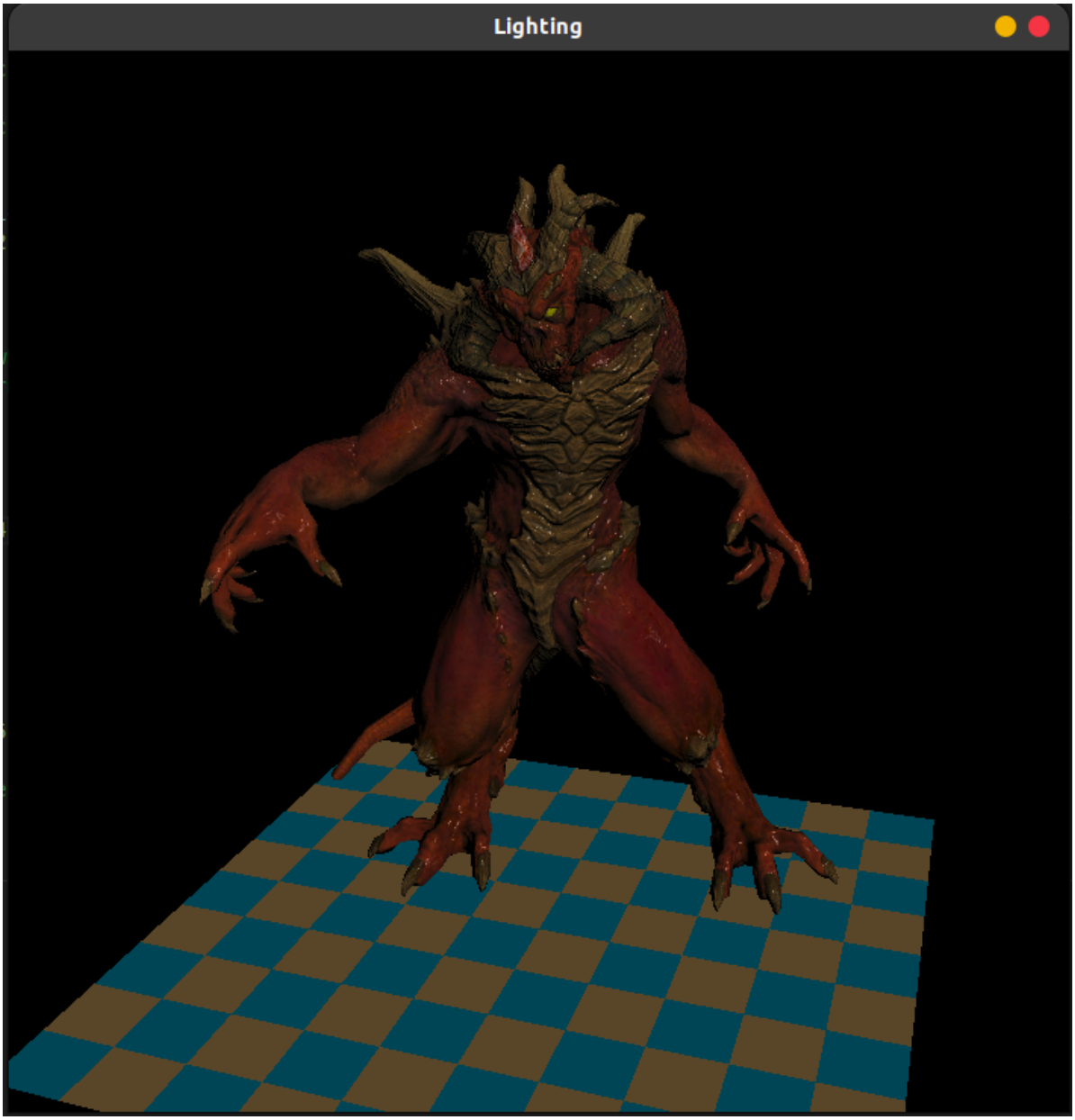
p=4时



p=16时



p=64时



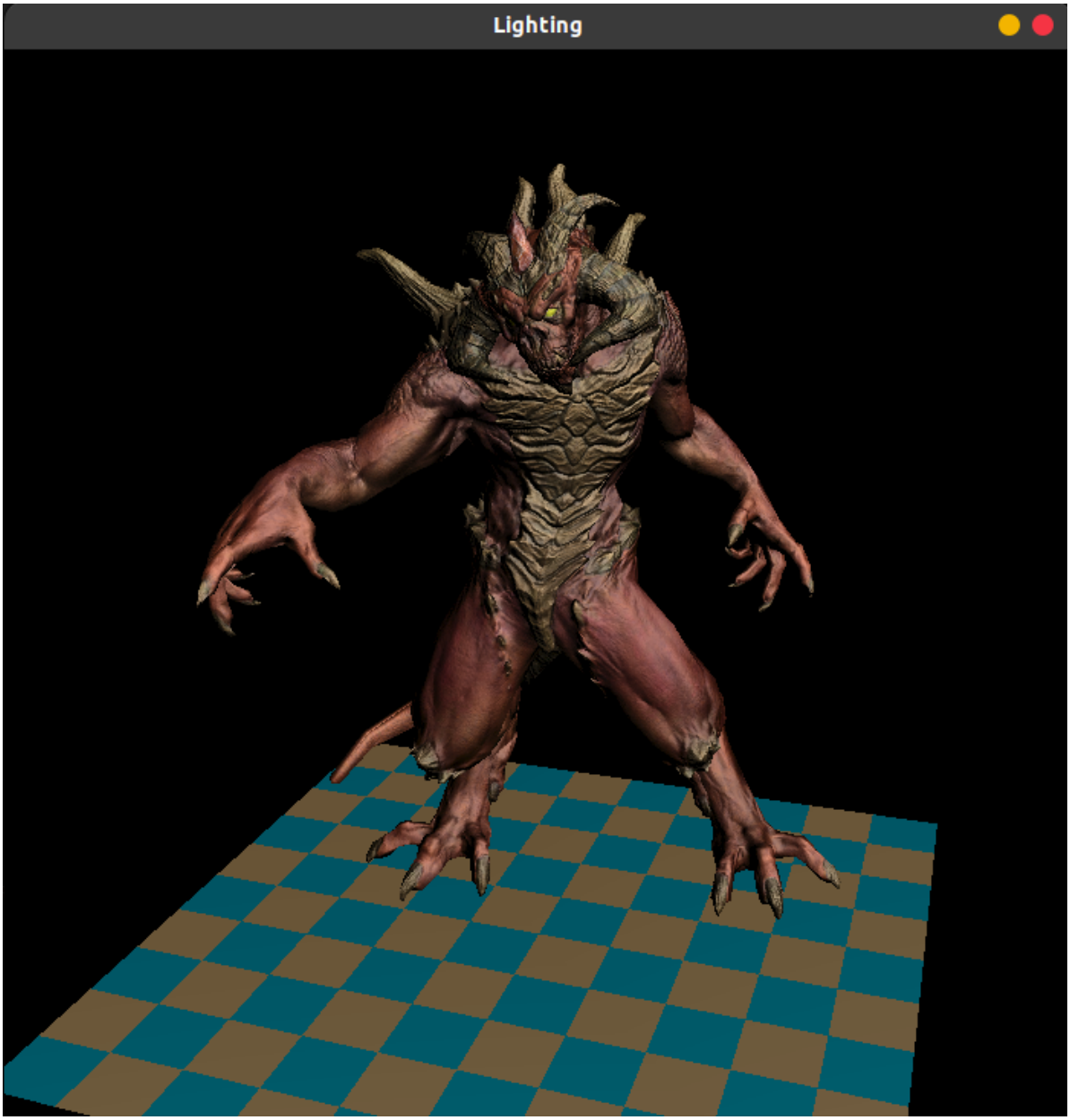
可见p越小，高光区域越大，因为$1 \geq \max\{\cos < normal, reflectDir >, 0\} \geq 0$，所以$p$越大，整体亮度就越小，这时说任何地方的亮度都会越小，所以，高光区域就越小。

Task 3、在 Shader 类中的 fragment 函数中实现Blinn-Phong的镜面光照计算逻辑

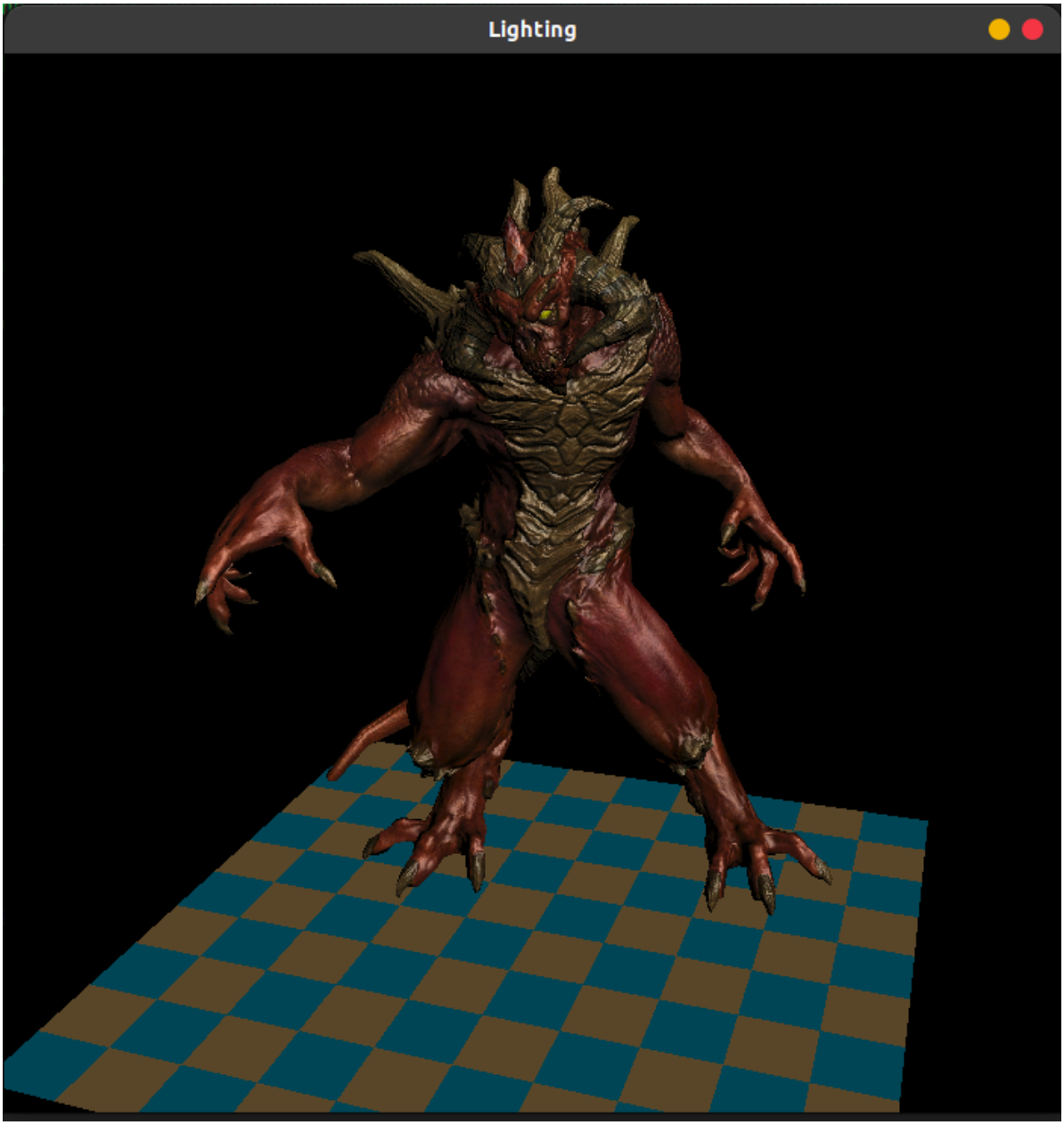
只需要计算中值，单位化后再计算余弦值即可。

```
Vec3f H = (lightDir + viewDir);
spec = H.normalize() * normal;
```

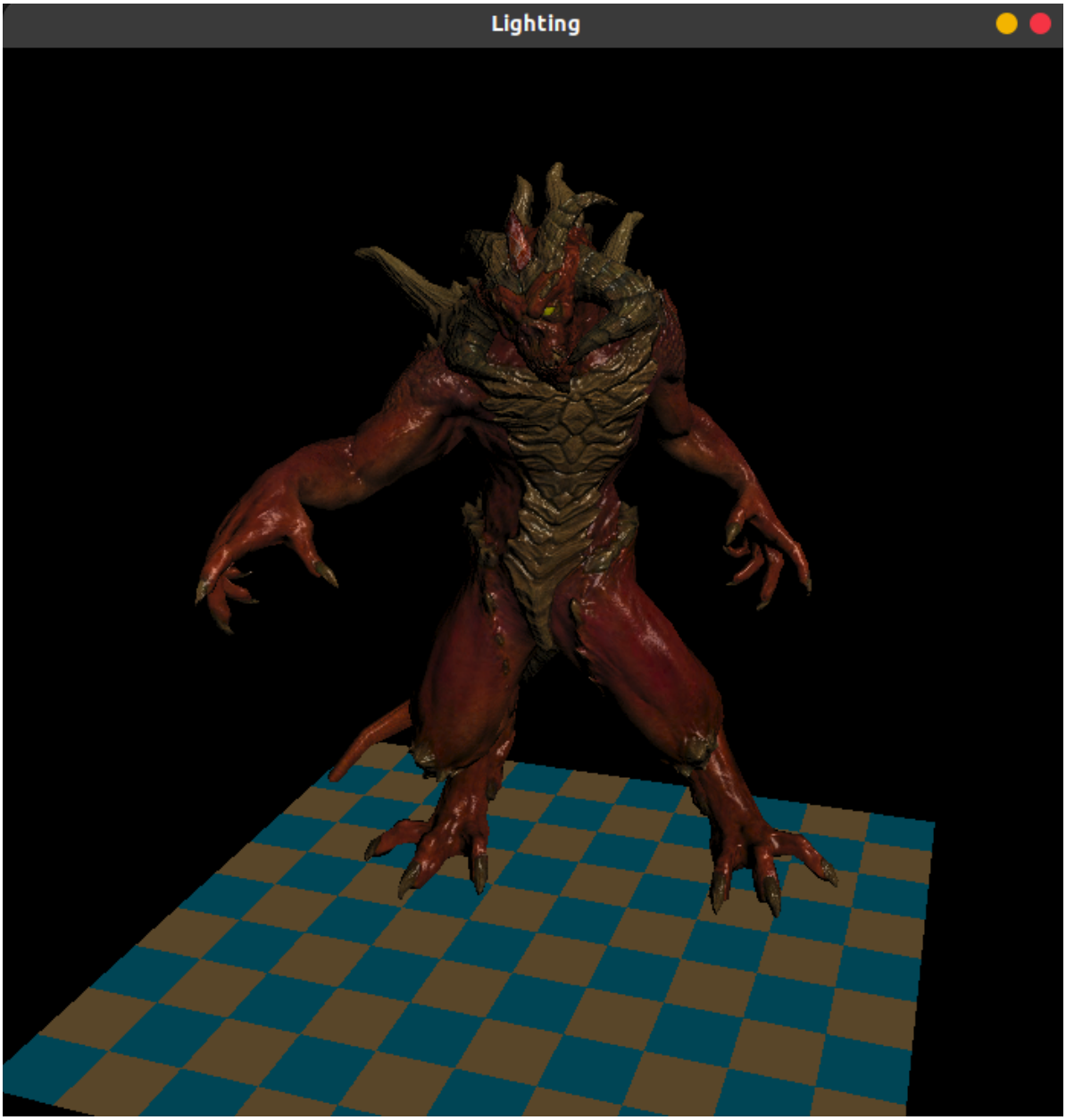
p=4时



p=16时



p=64时



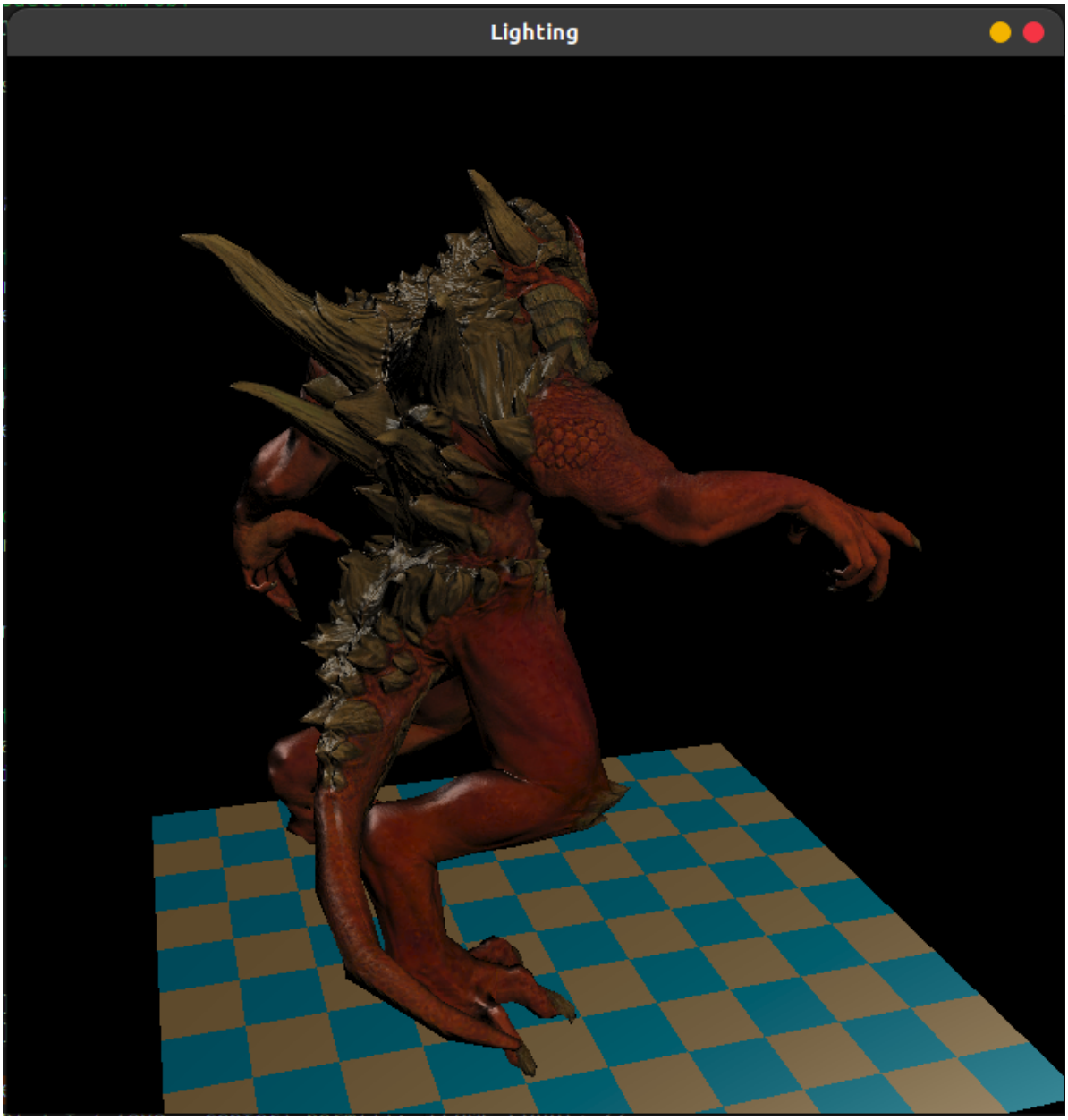
对比可知，bling-phong模型的高光处和暗处的对比没有phong模型那么突兀。而最重要的是bling-phong模型计算快，只学要把入射光线向量和视线向量相加再标准化最后和法线向量求内积即可

Task 4、在 main 函数中实现 eye 绕着 y 旋转的动画

只需使用选装矩阵乘一下就行，其实也可以手动计算

```
// TODO 4: rotate the eye around y-axis.
{
    // do something to variable "eye" here to achieve rotation.
    float theta = 5 * 3.1415926535 / 180.0f;
    Vec3f temp_eye = eye;
    eye[0] = cos(theta) * temp_eye[0] - sin(theta) * temp_eye[2];
    eye[2] = sin(theta) * temp_eye[0] + cos(theta) * temp_eye[2];
}
```

结果如下


Lighting