计算机图形学实验报告

学号：16340054

姓名：戴馨乐

学院：数据科学与计算机学院

作业：Homework6

Basic:

1. 实现 Phong 光照模型:
   - 场景中绘制一个 cube
   - 自己写 shader 实现两种 shading: Phong Shading 和 Gouraud Shading，并解释两种 shading 的实现原理
   - 合理设置视点、光照位置、光照颜色等参数，使光照效果明显显示
2. 使用 GUI，使参数可调节，效果实时更改:
   - GUI 里可以切换两种 shading
   - 使用如进度条这样的控件，使 ambient 因子、diffuse 因子、specular 因子、反光度等参数可调节，光照效果实时更改

实验部分:

Phong 光照模型包含以下 3 部分:

1) 环境光照 Ambient Lighting

模拟的是环境中向很多个方向散开的光，在现实生活中，光向各个

发散并反弹，模拟整个过程开销比较大，在这里使用了一个环境因子 ambientStrength，乘以光的颜色来表示

```
// 环境光照
vec3 ambient = ambientStrength * lightColor;
```

2) 漫反射光照 Diffuse Lighting

这也是模拟现实世界中的漫反射，但是也是趋近。通过光线向量和法向量点乘得到的 cos 值，可以实现不同片段和光源位置不同光照效果也不同，这样就有了漫反射的效果。这里光线向量和法向量都应该标准化，以简化获取 cos 值的计算。

另外，法向量在传入着色器时候，应该通过法向量矩阵来进行空间转换，以适应变换后的物体。

变换法向量:

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

计算漫反射:

```
// 漫反射
vec3 norm = normalize(Normal);
vec3 lightDir = normalize(lightPos - FragPos);
float diff = max(dot(norm, lightDir), 0.0f);
vec3 diffuse = diffuseStrength * diff * lightColor;
```

这里使用漫反射因子 diffuseStrength 来控制漫反射的程度

3) 镜面光照 Specular Lighting

镜面反射比起漫反射，并不是各个方向都有，而是固定和入射角度一样反射出去。那么假如人不在反射到的位置，这个光其实是投影

到了人看物体的视线上了，那么这个投影的分量就是我们要的镜面光照。

所以引入了观察者，通过观察者向量和光线向量点乘来得到结果。

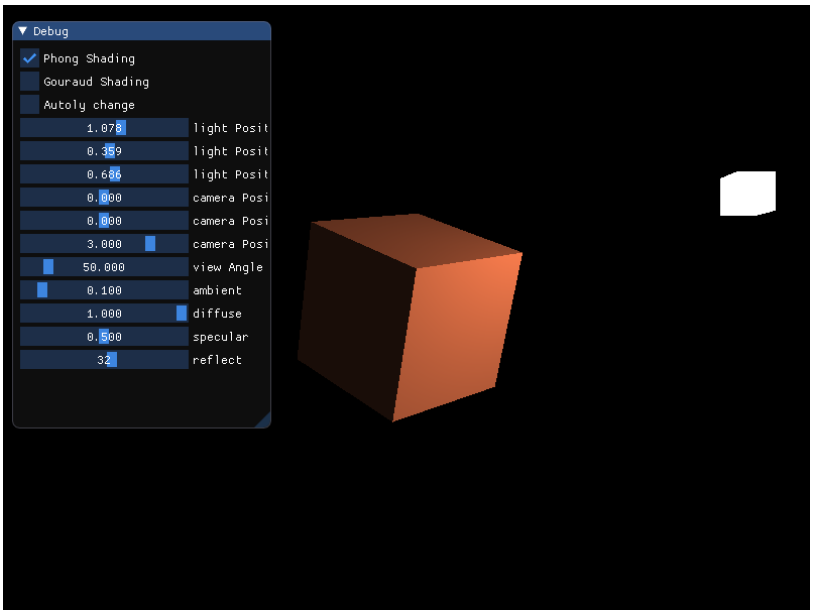
另外还引入了反光度，来表示镜面反射

```
// 镜面反射
vec3 viewDir = normalize(viewPos - FragPos);
vec3 reflectDir = reflect(-lightDir, norm);
float spec = pow(max(dot(viewDir, reflectDir), 0.0), reflectExtent);
vec3 specular = specularStregth * spec * lightColor;
```

使用 GUI 对 ambientStrength，diffuseStrength, specularStrength 以及反光度 reflectExtent 来进行控制，这里我还另外加入了对光源位置以及物体旋转角度的控制。

```
ImGui::NewFrame();
{
    ImGui::Checkbox("Phong Shading", &phong);
    ImGui::Checkbox("Gouraud Shading", &gouraud);
    ImGui::Checkbox("Autoly change", &autoChange);
    if (!autoChange) {
        ImGui::SliderFloat("light Position - x", &lightPosX, -5.0f, 5.0f);
        ImGui::SliderFloat("light Position - y", &lightPosY, -5.0f, 5.0f);
        ImGui::SliderFloat("light Position - z", &lightPosZ, -5.0f, 5.0f);
        ImGui::SliderFloat("camera Position - x", &cameraPosX, -5.0f, 5.0f);
        ImGui::SliderFloat("camera Position - y", &cameraPosY, -5.0f, 5.0f);
        ImGui::SliderFloat("camera Position - z", &cameraPosZ, -5.0f, 5.0f);
        ImGui::SliderFloat("view Angle", &viewAngle, 0.0f, 360.0f);
        ImGui::SliderFloat("ambient", &ambient, 0.0f, 1.0f);
        ImGui::SliderFloat("diffuse", &diffuse, 0.0f, 1.0f);
        ImGui::SliderFloat("specular", &specular, 0.0f, 1.0f);
        ImGui::SliderInt("reflect", &reflect, 10, 50);
    }
}
```

效果:

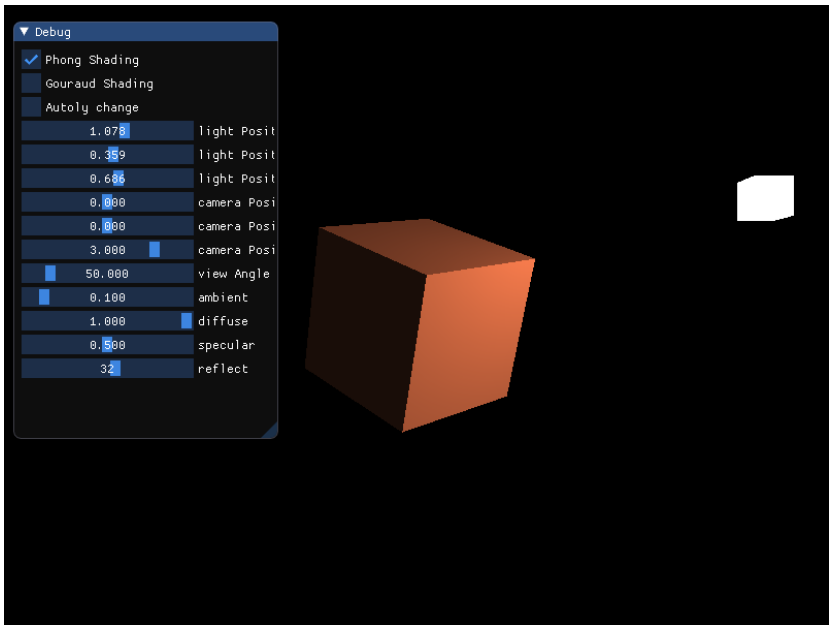


改变各个因子:

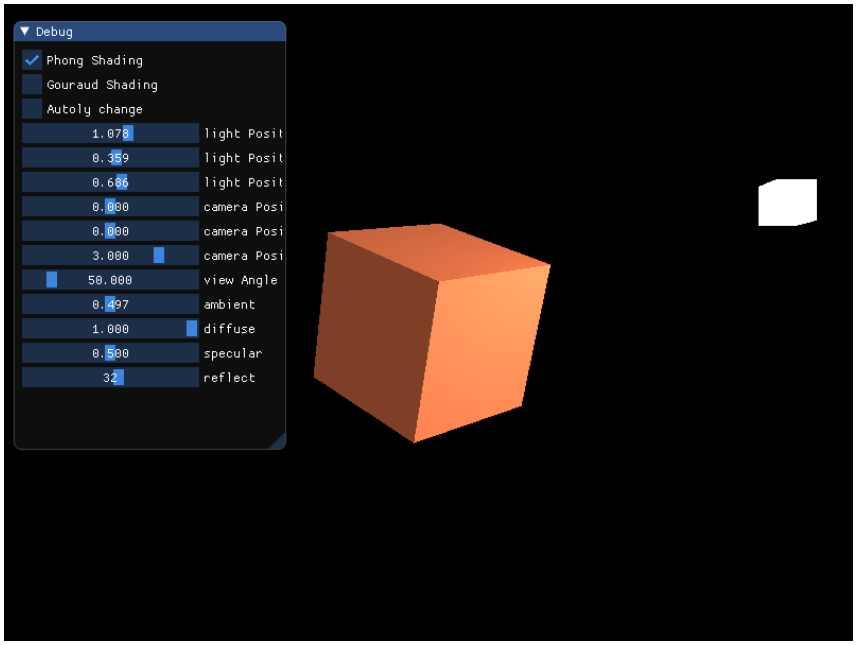
1) ambientStrength

保持其他变量不变，改变 ambientStrength

ambientStrength = 0.1 时候 ：



ambientStrength = 0.497 时候:

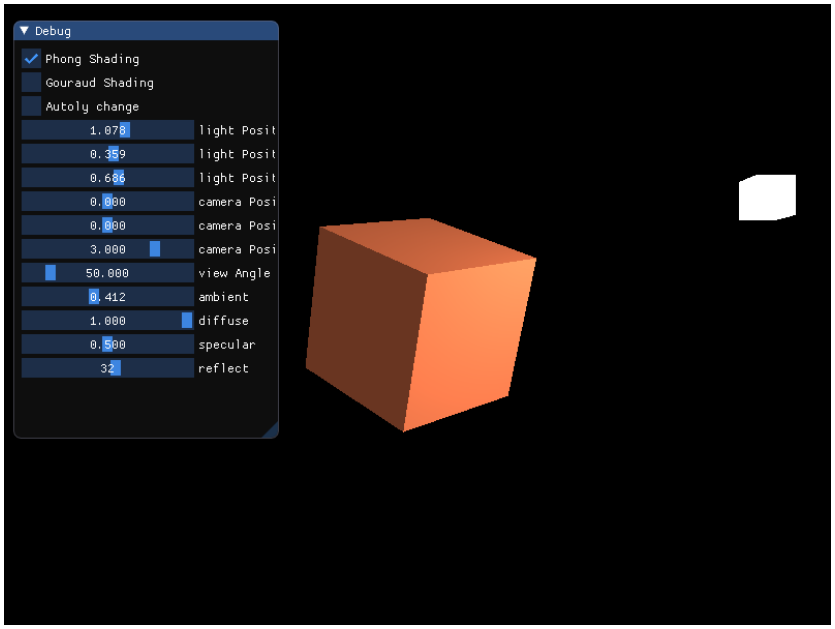


可以看出，提高环境光照因子，物体的亮度明显提高，这好比环境越来越亮，物体自然越来越清晰。

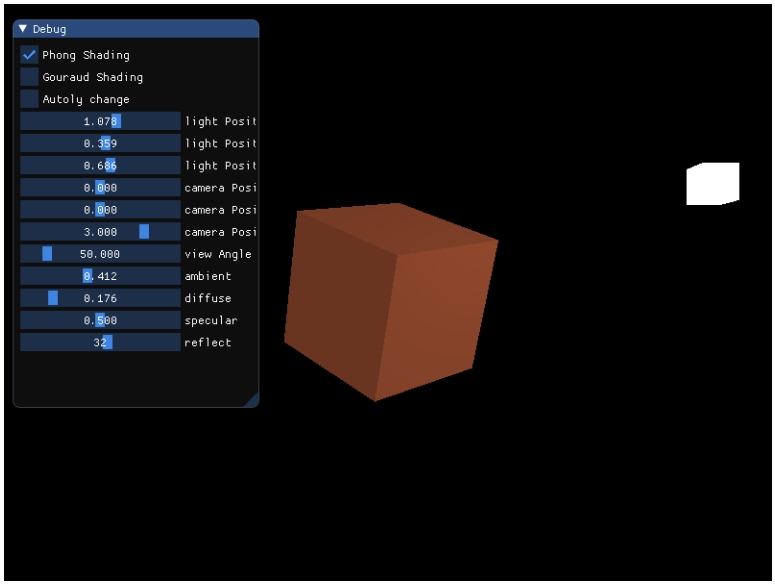
2) diffuseStrength

保持其他变量不变，改变 diffuseStrength

diffuseStrength = 1.0 时候：



减小 diffuseStrength 到 0.176：



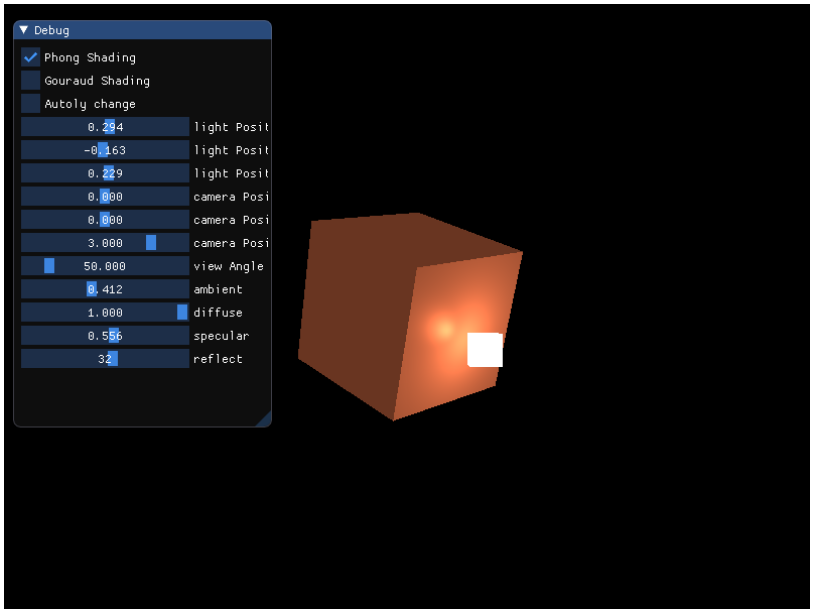
可以看到，漫反射减少，物体的亮度降低

3) specularStrength 和反光度

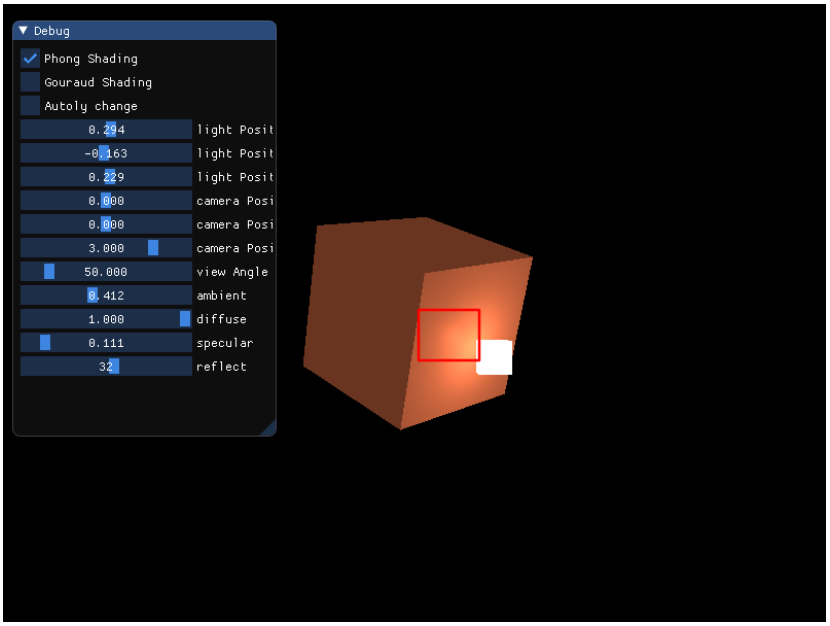
保持其他变量不变，改变 specularStrength 和反光度

为了效果明显，需要将光源移动到物体附近

此时 specularStrength = 0.556，反光度为 32

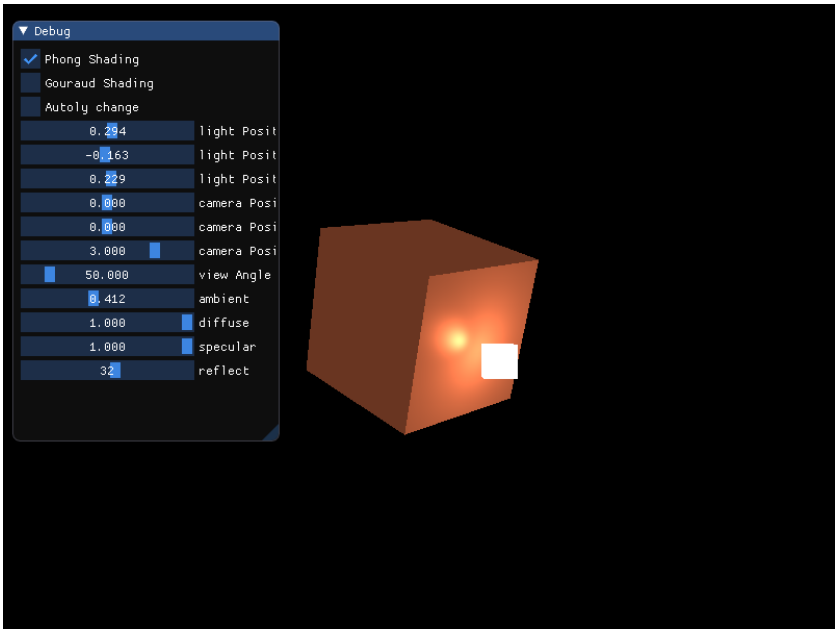


保持反光度和其他变量不变，减小 specularStrength = 0.111

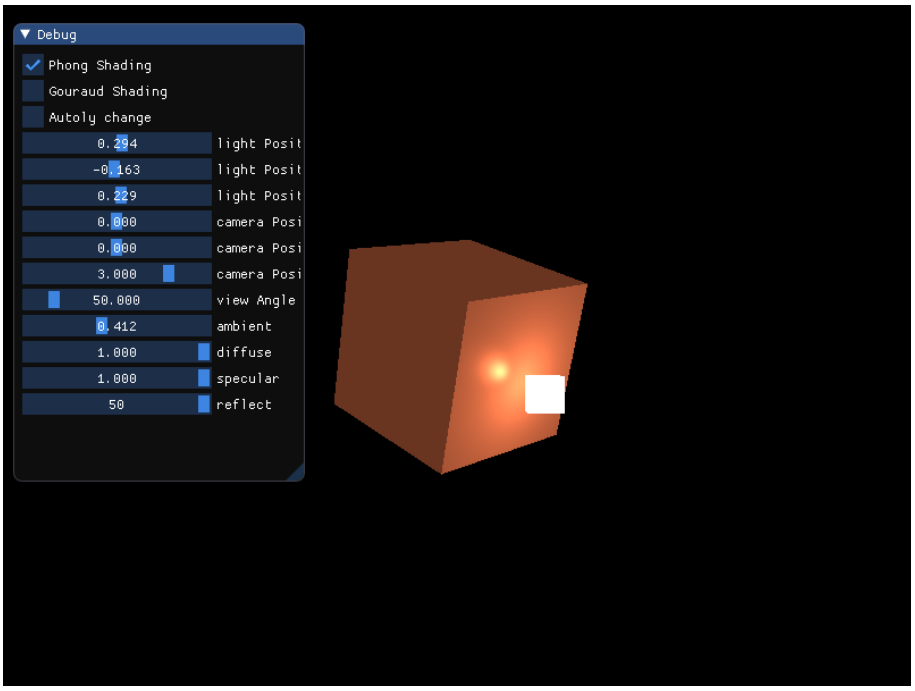


可以看到比起上图，镜面反射不明显了，镜面反射因子减小，镜面反射的影响降低

保持 specularStrength 不变，修改反光度，此时反光度 = 32

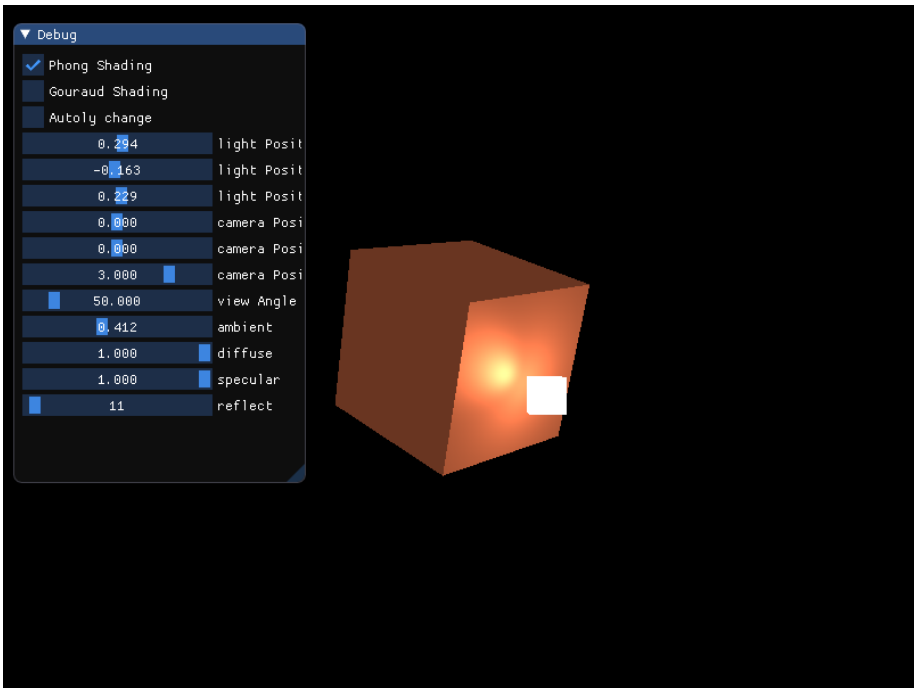


增大反光度至 50:



比起反光度为 32 时候，镜面反射产生的圆圈收缩，反射点集中

修改反光度为 11:



可以看到，反光度减小，镜面反射产生的圆圈向周围发散开来

Gouraud 光照模型：

和 Phong 模型类似，Gouraud 模型也由 3 个部分组成：

1）环境光照

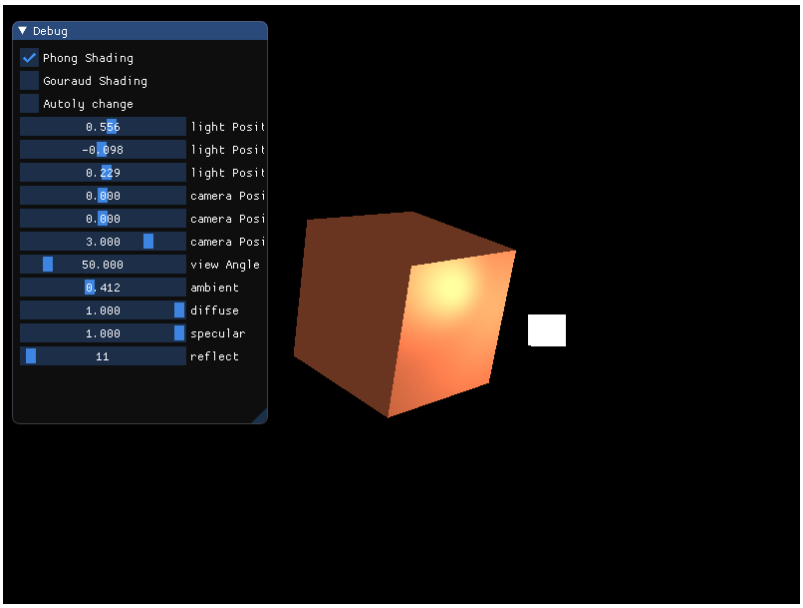
2）漫反射

3）镜面反射

不同的是，Phong 模型计算每个顶点的法向量，然后再片段中计算，所以前面 Phong 模型法向量的计算放在顶点着色器中，模型光照的计算放在片段着色器中，这样可以计算每个点的光强，保证了效果的真实。而 Gouraud 模型在顶点处采用上述的局部反射模型，计算了各个点的光照，然后在片段着色器中通过线性插值来得到结果，这样子计算量减小了，但是效果不如 Phong 真实。

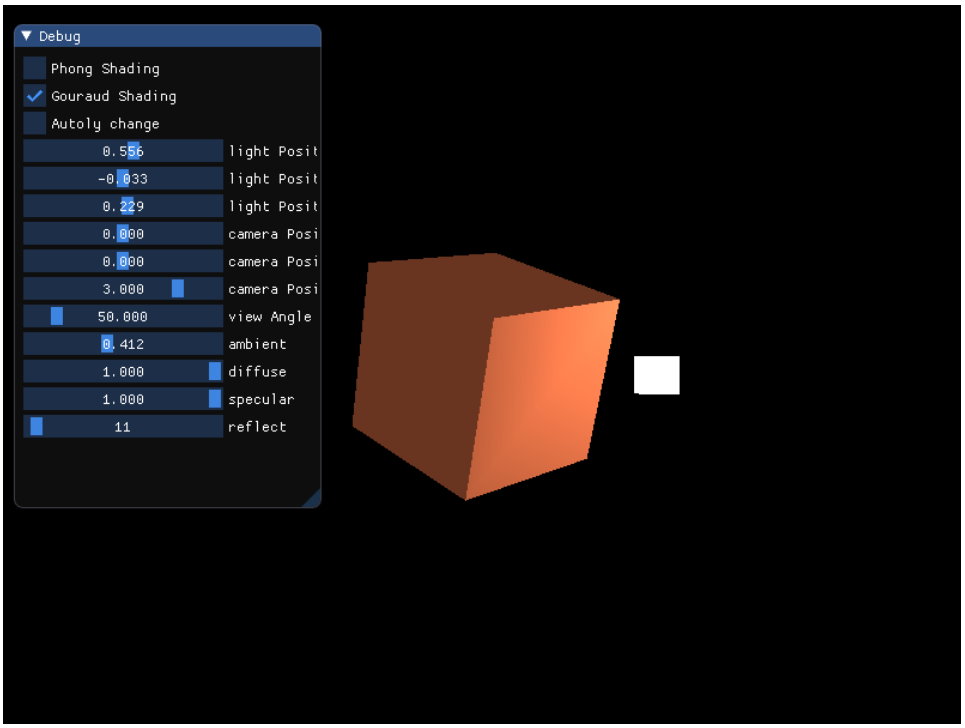
对比：

Phong 模型：



可以看到，采用 Phong 模型，镜面反射形成的圆圈是可见的，这是因为独立计算了每个点的光强

Gouraud 模型：



采用 Gouraud 模型之后，镜面反射形成的圆圈不见了，镜面反射的效果弱化了，这是因为各个点的光强在片段着色器中线性插值，使得效果不真实。

Bonus：

当前光源为静止状态，尝试使光源在场景中来回移动，光照效果实时更改。

这里让光源以（0，0，0）点为中心，在 x, y 平面绕着半径为 1.0f 的圆运动

```
if (autoChange) {
    // 实时改变光源位置
    lightPos.x = sin(glfwGetTime()) * 1.0f;
    lightPos.y = cos(glfwGetTime()) * 1.0f;
}
```

使用时间值作为角度值，使得光源实时变化

效果：