1. 实现 phong 光照模型

A． 绘制一个 Cube

B． 自己写 shader 实现两种 shading：phong 和 Gouraud，并解释两种 shading 的实现原理。

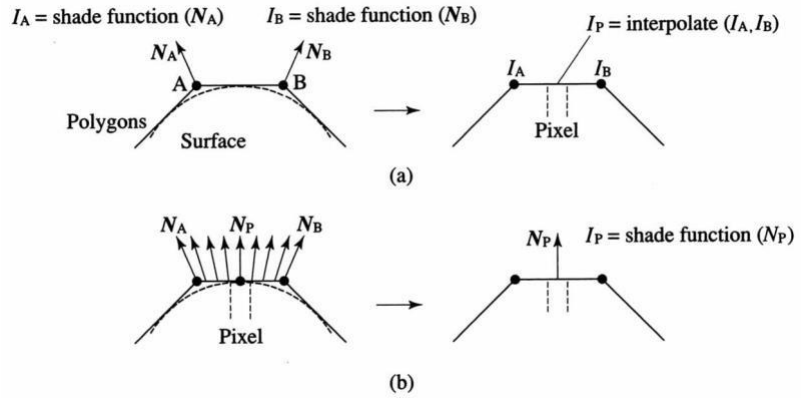
Phong Shading：对法向量进行插值后再对每个点用法向量和 lighting 模型计算光照强度。线性插值表面的法向量，在每个像素应用 phong 光照模型。

更加昂贵，但是 smooth-looking

1　顶点的法向量是所有汇聚于该面的法向量的平均

2 对于每一个多边形，被内插值占据的面法向量的值用在每个顶点的法向量的线性插值得到

Gouraud Shading：对光照强度直接进行插值。是内插算法，每个顶点的法向量是周围几个面的法向量的平均。每个顶点的光照强度计算时使用法向量和一个照明模型。对于每个多面体强度值是有周围顶点的值内插而得到。



这次基本重构了大部分之前代码（当屎堆积成山）。因为年轻犯下了错误，在主程序里手动暴力算出了法向量。实际上这一部分似乎应该在 shader 中完成。另外由于对渲染管线的理解不透彻，导致了 vertex 和 fragment 中的重复实现以及实现顺序的混乱。

Lighting 模型：环境光+漫反射+镜面反射

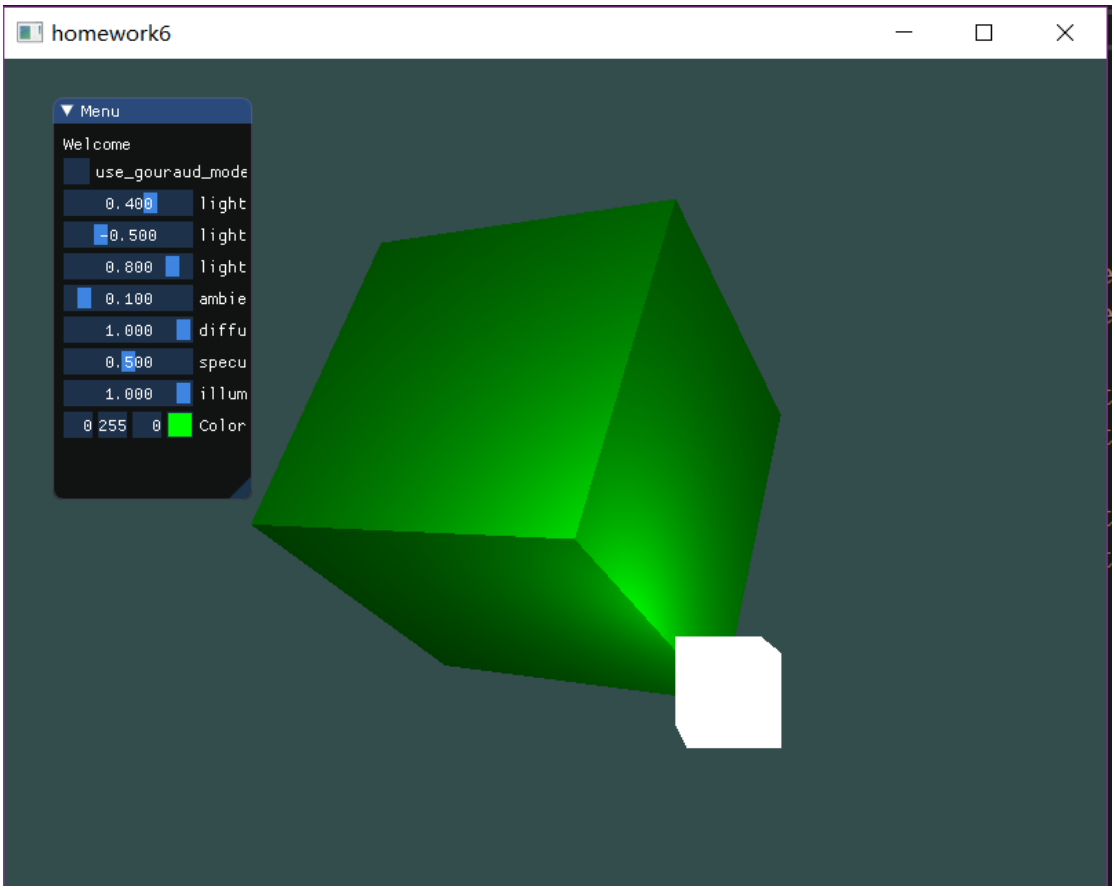
C． 设置合理的视点、光照位置、光照颜色等参数，使光照效果更加明显

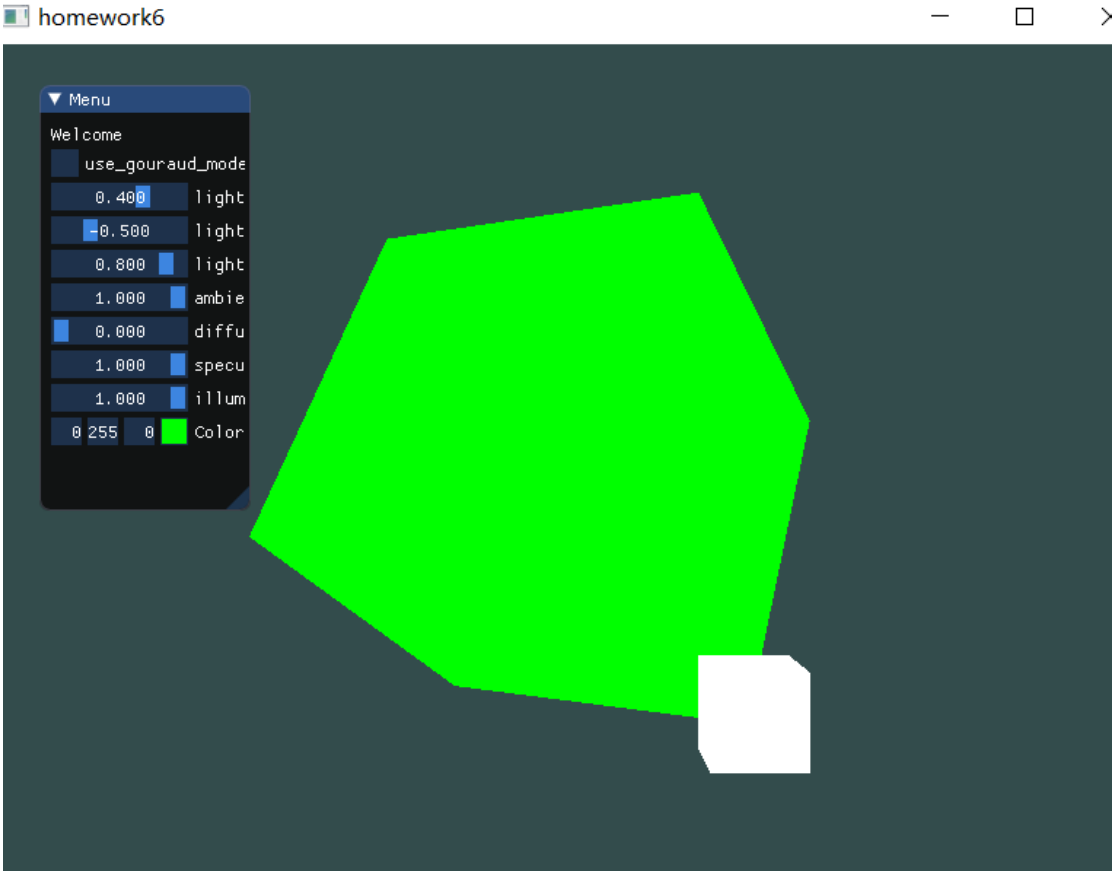
我们可以看到 gouraud 的光照效果不太适合处理 specular 的情况（高光），会出现三角形的亮区。

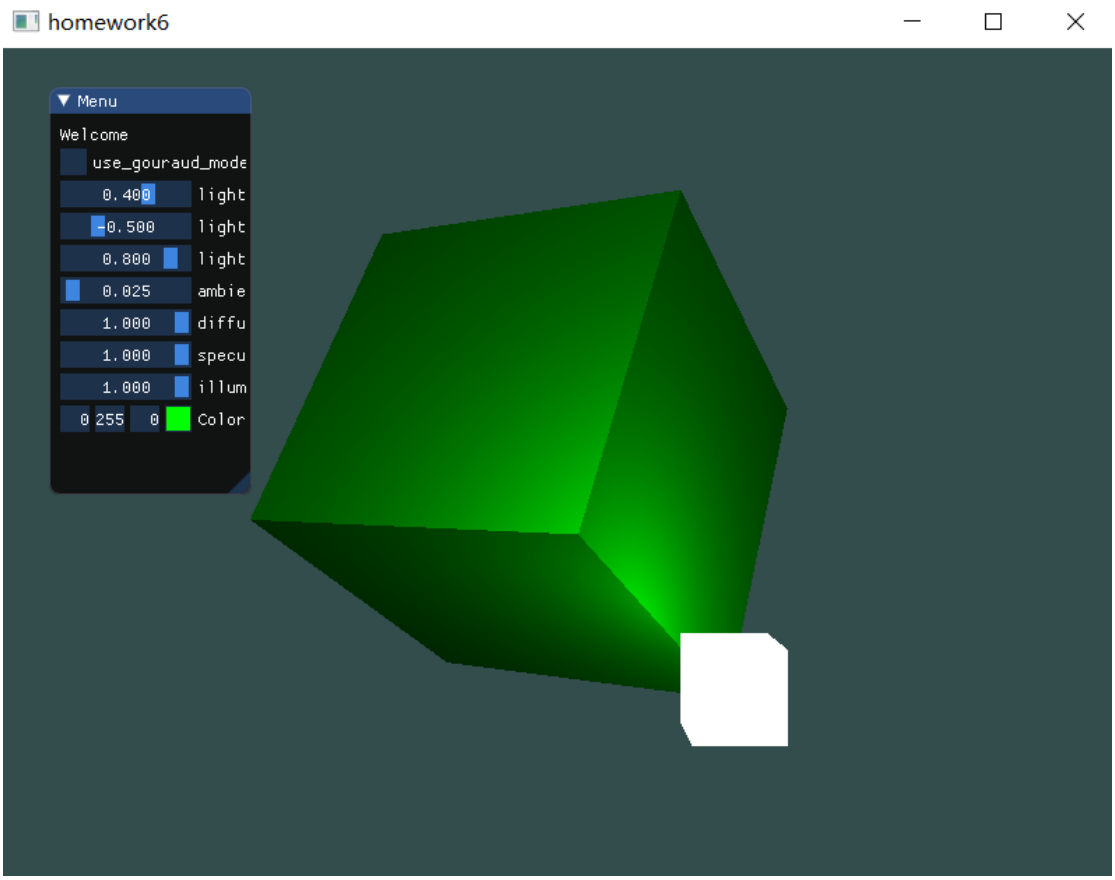
2　使用 GUI，使参数可调节，效果实时更改

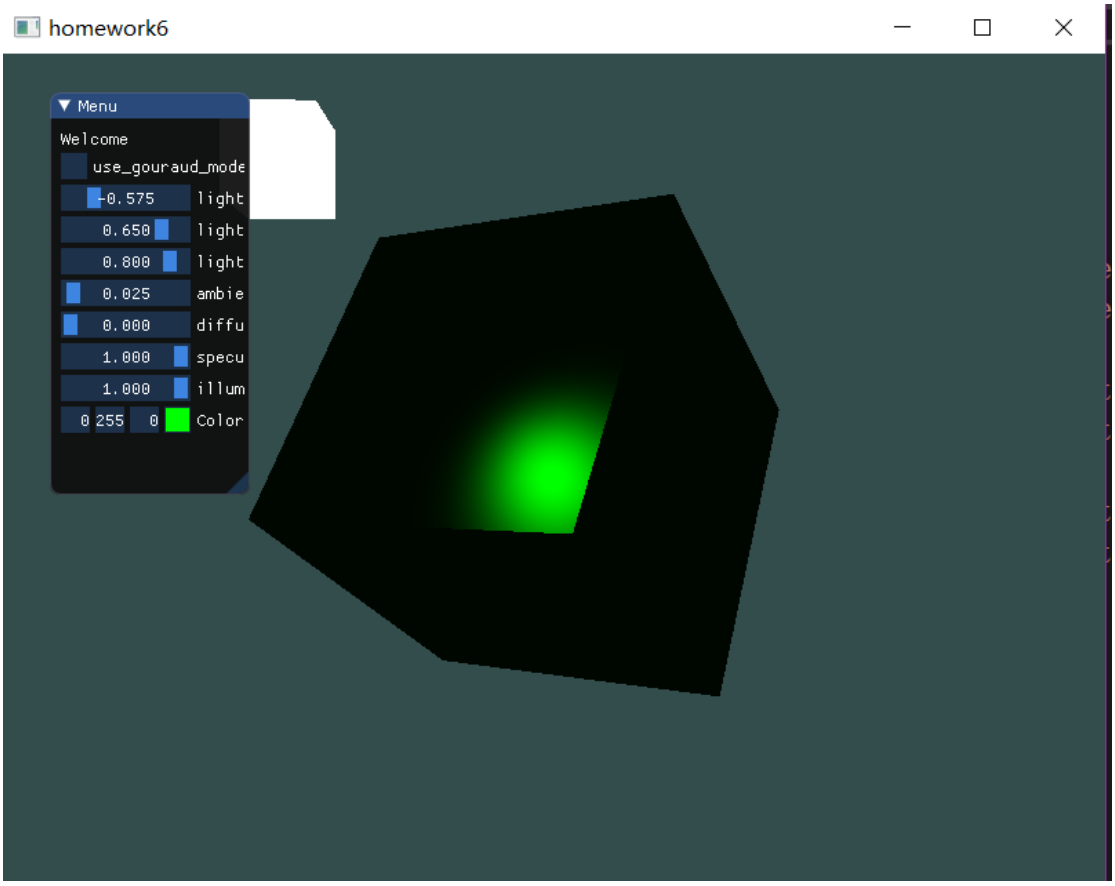
A． GUI 中可以切换两种 shading

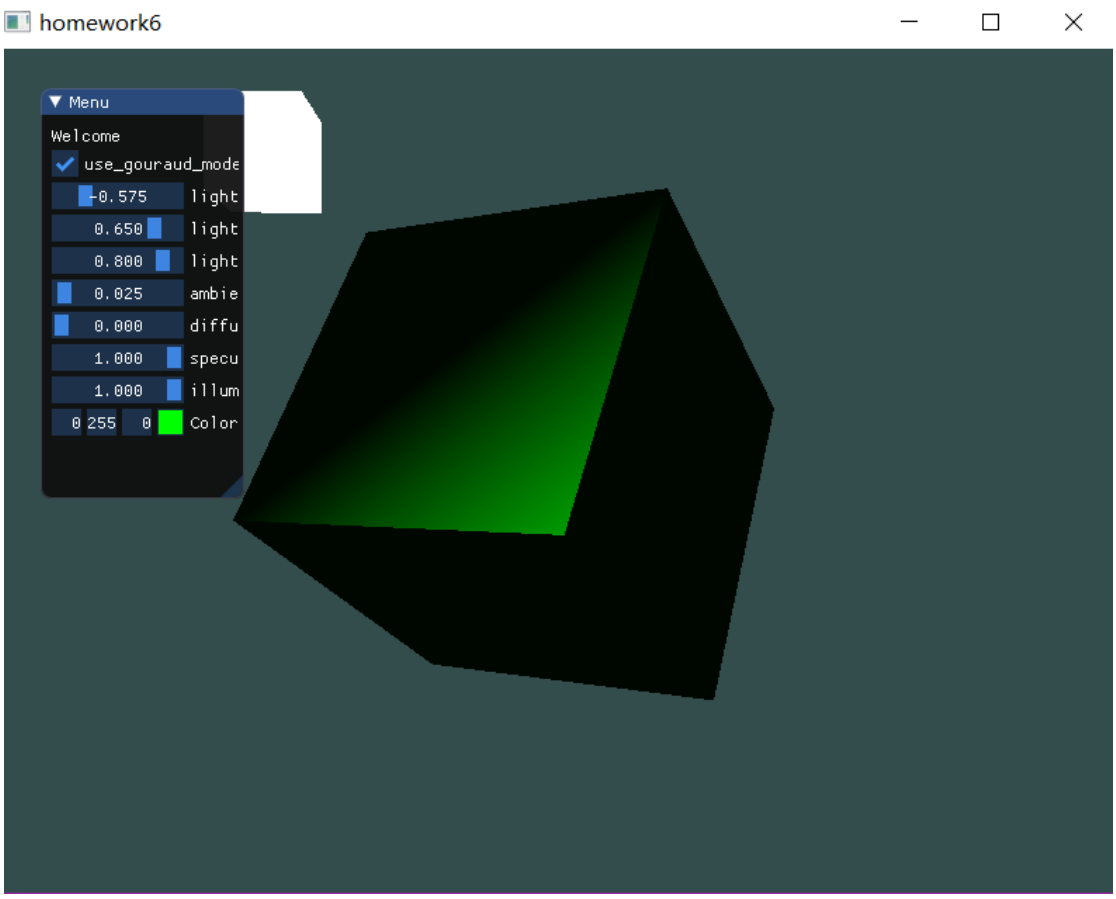
B． 使用进度条空间，使得 ambient、diffuse、specular、反光度可以调节，光照效果实时更改。


homework6
Menu
Welcome
use_gouraud_mode
0.400
light
-0.500
light
0.800
light
0.100
ambie
1.000
diffu
0.500
specu
1.000
illum
0 255 0
Color


homework6
Menu
Welcome
use_gouraud_mode
0.400
light
-0.500
light
0.800
light
1.000
ambie
0.000
diffu
1.000
specu
1.000
illum
0 255 0
Color


homework6
Menu
Welcome
use_gouraud_mode
0.400
light
-0.500
light
0.800
light
0.025
ambie
1.000
diffu
1.000
specu
1.000
illum
0 255 0
Color


homework6
Menu
Welcome
use_gouraud_mode
-0.575
light
0.650
light
0.800
light
0.025
ambie
0.000
diffu
1.000
specu
1.000
illum
0 255 0
Color



3　使光源在场景中来回移动，光照效果实时更改。

改变 lightcube 的位置就可以了。

分别得到三种光照 累加起来即可

```
#version 330 core
out vec4 FragColor;

in vec3 Normal;
in vec3 FragPos;

uniform vec3 objectColor;
uniform vec3 lightColor;
uniform vec3 lightPos;
uniform vec3 viewPos;
uniform float ambientStrength;
uniform float diffuseStrength;
uniform float specularStrength;
uniform float lightStrength;


void main() {
    vec3 ambient = ambientStrength * lightColor;

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

    vec3 viewDir = normalize(viewPos - FragPos);
    vec3 reflectDir = reflect(-lightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
    vec3 specular = specularStrength * spec * lightColor;

    vec3 result = lightStrength * (ambient + diffuse + specular) * objectColor;
    FragColor = vec4(result, 1.0);
}
```