计算机图形学第六次作业

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

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。

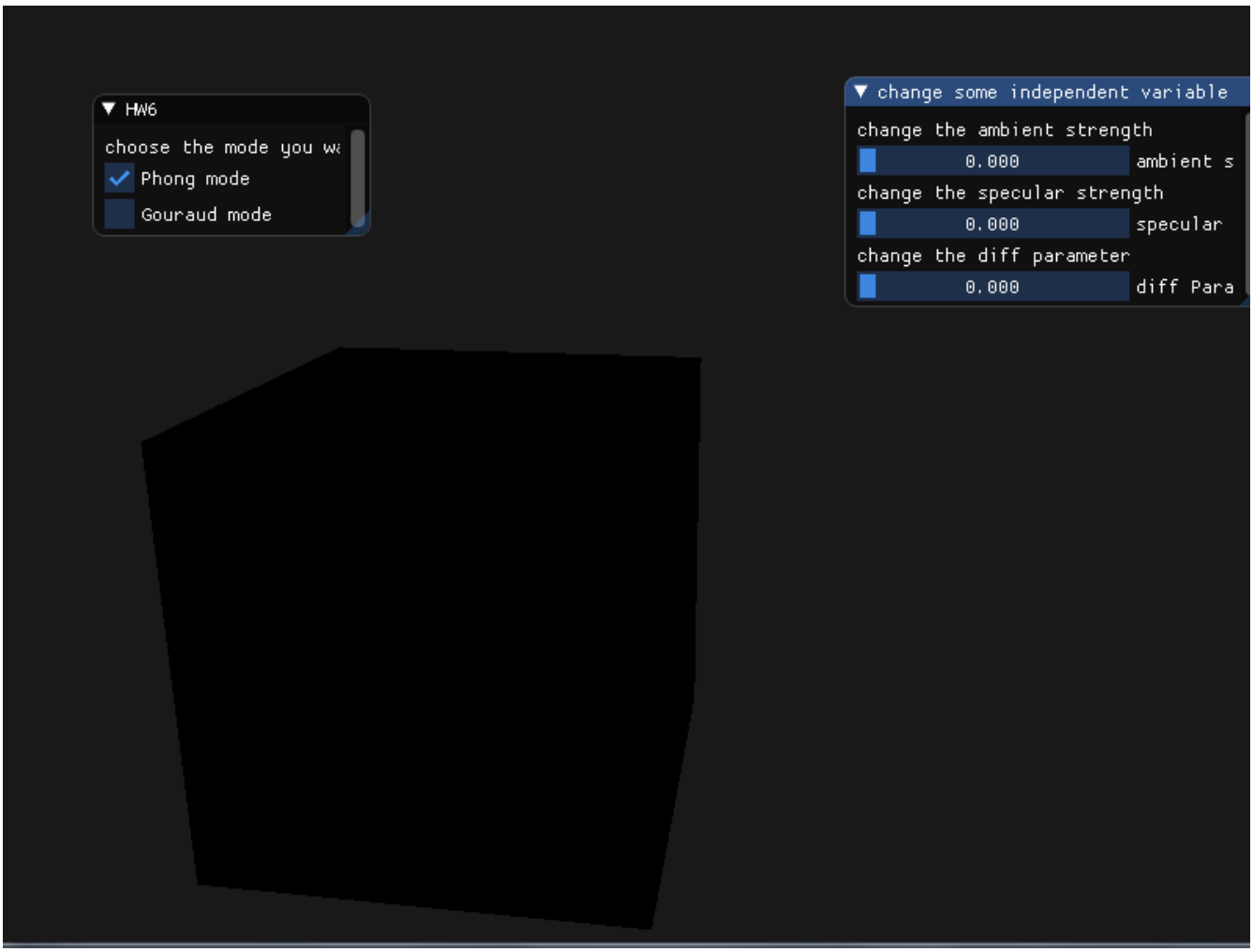
场景中绘制cube

这本次项目中，我在场景中绘制了两个cube，一个作为被光照的物体，另一个作为光源，由于之后要用到立方体的六个面的法向量，因此我提前在vertices数组中将立方体六个面的法向量标出来了：

```
float vertices[] = {
    -0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
    0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
    0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
    0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
    -0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
    -0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,

    -0.5f, -0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
    0.5f, -0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
    0.5f,  0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
    0.5f,  0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
    -0.5f,  0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
    -0.5f, -0.5f,  0.5f,  0.0f,  0.0f, 1.0f,

    -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f,  0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f, -0.5f,  0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,

    0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
    0.5f,  0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
    0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
    0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
    0.5f, -0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
```

```
    0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,

    -0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,
    0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,
    0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
    0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
    -0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
    -0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,

    -0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,
    0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,
    0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
    0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
    -0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
    -0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f
};
```

然后我继续使用了上一次作业的可移动摄像机类，将cube放到三维空间中显示，而物体的颜色可以直接在shader中设置，但这样的设置就只能显示一个单色调的立方体。



冯氏光照模型

冯氏光照模型的主要结构包括：环境光照，漫反射，镜面光照。

环境光照主要是使用简化的全局光照，即光通常都不是来自于同一个光源，而是来自于我们周围分散的很多光源，即使它们可能并不是那么显而易见。光的一个属性是，它可以向很多方向发散并反弹，从而能够到达不是非常直接临近的点。使用的方式是在shader中用光的颜色乘以一个很小的常量环境因子，再乘以物体的颜色，然后将最终结果作为片段的颜色：

```
float ambientStrength = ambient_Strength;
vec3 ambient = ambientStrength * lightColor;
```

漫反射需要的参数有两个，分别是物体表面的法向量和定向的光线，法向量可以直接用立方体中已经设置的法向量的值，而定向光线的获取需要物体的位置和光源的位置，光源的位置可以直接使用光源立方体的位置，通过uniform传入到片段着色器中，由于我们是在世界坐标中计算光照，所以我们需要一个世界坐标的片段位置：

```
FragPos = vec3(model * vec4(aPos, 1.0));
```

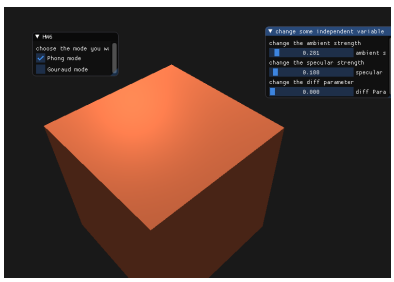
然后对法向量进行标准化后和光线方向进行点乘得到光源位置对物体光照的影响。

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

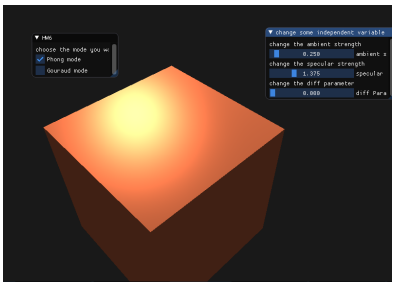
镜面反射实现：

```
//Specular
float specularStrength = specular_Strength;
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;
```

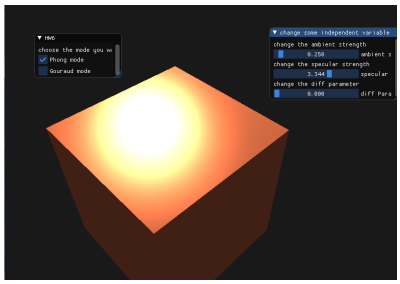
实验截图：



specular=0.188



specular=1.375



specular=3.344

从实验结果可以看出，随着镜面反射的specular因子增大，我们看到的反射亮度增大。

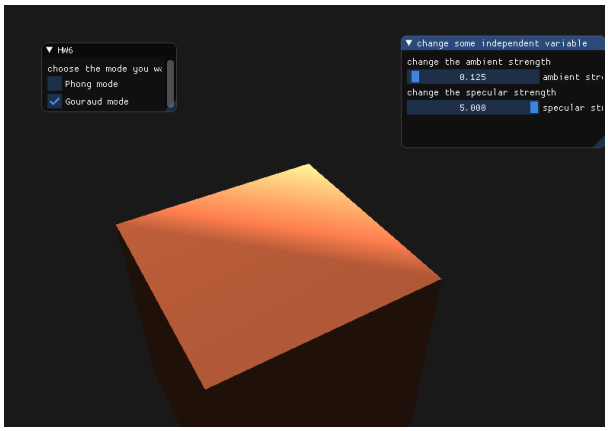
高洛德光照模型

对gouraud光照模型而言，他是在顶点着色器中计算光模型，然后在片段着色器中进行插值计算来渲染立方体：

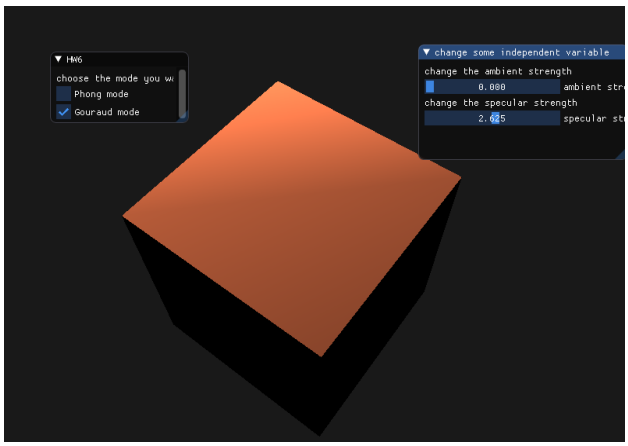
```
void main()
```

```
{
    gl_Position = projection * view * model * vec4(aPos, 1.0);

    // gouraud shading
    vec3 Position = vec3(model * vec4(aPos, 1.0));
    vec3 Normal = NormalMatrix * aNormal;

    // ambient
    float ambientStrength = ambient_Strength;
    vec3 ambient = ambientStrength * lightColor;

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

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

    LightingColor = ambient + diffuse + specular;
}
```

实验结果：



specular=5.0



specular=2.625

可以看出，相对于冯氏光照模型，gouraud光照模型的镜面反射没有光圈。同样的，随着specular因子的增大，反射亮度增大。

对比来说，Phong shading是Phong lighting model在每一个片元上进行实现，而gouraud shading是Phong lighting model在每一个顶点是实现然后通过插值获取到面中各个片元的光照颜色值，所以看起来会是顶点特别亮，然后亮度逐渐向其他片元延伸。如果在顶点处没有高光，那么就会出现shading miss的现象。

光源在场景中来回移动

加分项实现的是一个能随时改变光源移动的功能，这里我使用的是glfwGetTime()函数设置光源的位置：

```
lightShader.setVec3("lightPos", glm::vec3(1.0f, 2.0f, 2.0f*(float)sin(glfwGetTime())));
```

这样就保证了光源在z轴方向上来回移动。一旦改变了光源的位置，就会对漫反射下光线的方向产生影响，进而影响漫反射，而使用了摄像机类又会对镜面反射和漫反射的效果有影响，这样就能达到改变多种因素产生不一样的光照效果。