Computer Graphics Homework 6

Basic

实现Phong光照模型

绘制Cube

和上次作业区别不大， 复用上一次的代码既可以绘制一个Cube。这次作业需要绘制两个Cube一个代表被光照的的物体，一个代表光源本身。由于二者对于vertex的解析相同，因此可以共享一个vertex shader，但是需要编译不同的fragment shader，因为被照亮的物体需要实现Phong光照模型，而光源六个表面都是发光颜色即可。需要两个VAO以及两个program一个用于绘制被光照的cube，另一个负责绘制光源。

Phong Shading和Gouraud Shading的实现

首先，这里由于要在一次作业中使用两种shader，即调用不同的shaderProgram，Homework类需要简要的重构，即在类变量中设置多个shader，并在构造函数中编译这些shader.

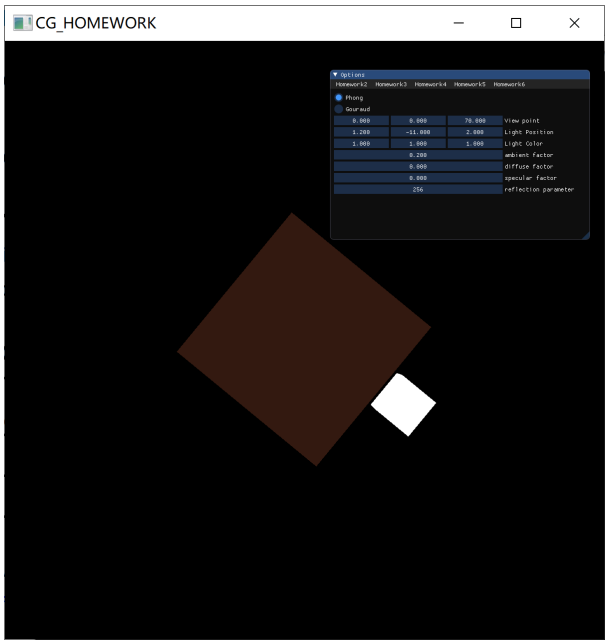
Phong Shading的实现

冯氏光照模型主要分为三个部分：环境光、漫反射和镜面反射。按照ppt上分别实现如下：

环境光

$I_{ambient} = I_a K_a$，即光源颜色乘以环境光因子（默认为0.1）

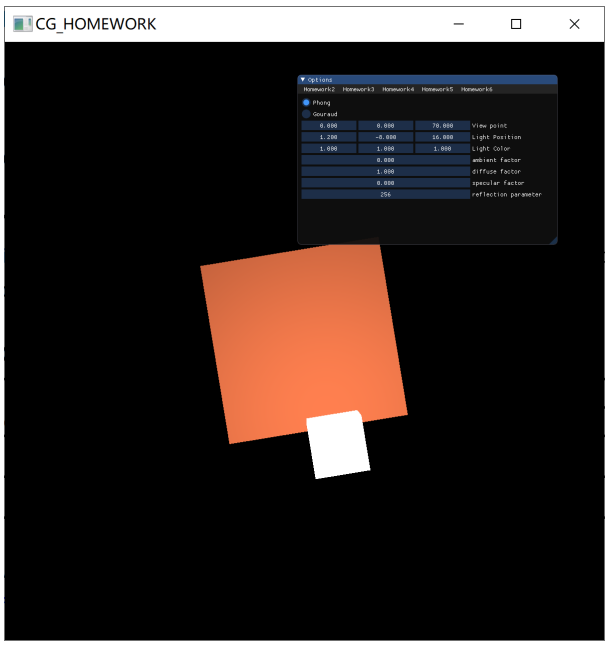
```
// phong_shading.fs
vec3 ambient = ambientFactor * lightColor;
```



漫反射

$I_{diffuse} = K_d I_{light} cos < \vec{n}, \vec{l} >$. 当$\vec{n}$和$\vec{l}$均为单位向量时：$cos < \vec{n}, \vec{l} >= \vec{n} \bullet \vec{l}$ 。因而在程序中先将法向量和光的入射向量标准化后计算夹角。

```
// phong_shading.fs
vec3 norm = normalize(Normal);
vec3 lightDir = normalize(lightPosition - FragPos);
float diff = max(dot(norm, lightDir), 0.0); // 处理夹角>90的情况
vec3 diffuse = diffuseFactor * lightColor * diff;
```



镜面反射

镜面反射遵循公式$I_{specular} = K_s I_{light} (\vec{v}, \vec{r})^{n_{shiny}}$。需要计算：

$\vec{v}$: 观察点到cube上的点的方向的单位向量

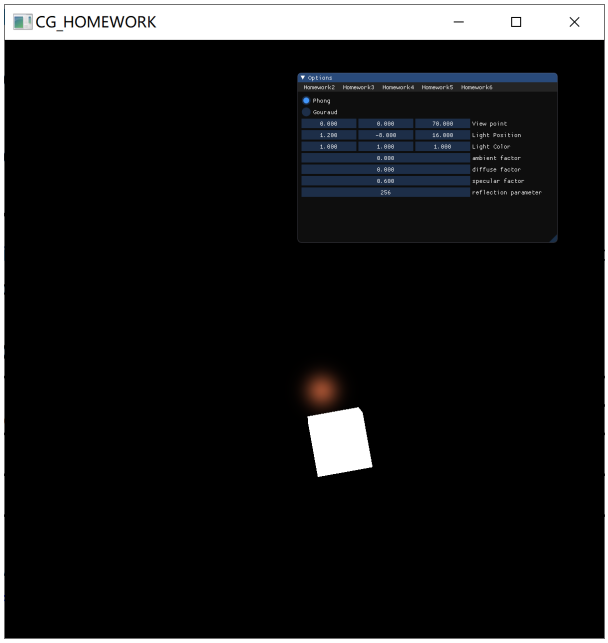
$\vec{r}$: 理想反射方向，调用 reflect 函数，通过$I_{light} - 2.0 * N \bullet I_{light} * N$计算的得到

K_s: 镜面反射银子

I_{light} : 入射光

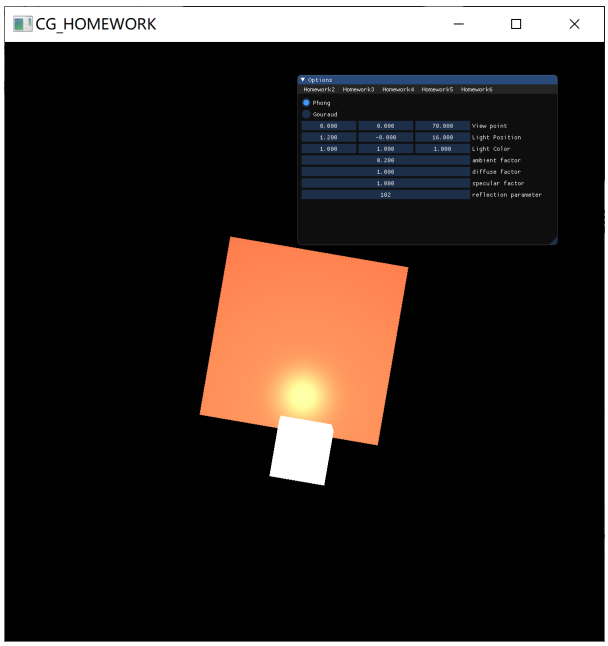
n_{shiny} : 材质发光常数

```
vec3 viewDir = normalize(viewPos - FragPos); // 视线方向
vec3 reflectDir = reflect(-lightDir, norm);
float spec = pow(max(dot(viewDir, reflectDir), 0.0), reflectionPara); // 32 - 反光度
vec3 specular = specularFactor * spec * lightColor;
```



最后将三者相加得到光对物体的所有作用，并结合物体的固有颜色`objectColor`进行**点积运算**，获得光作用在物体表面的环境光、漫反射、镜面光效果的叠加。

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

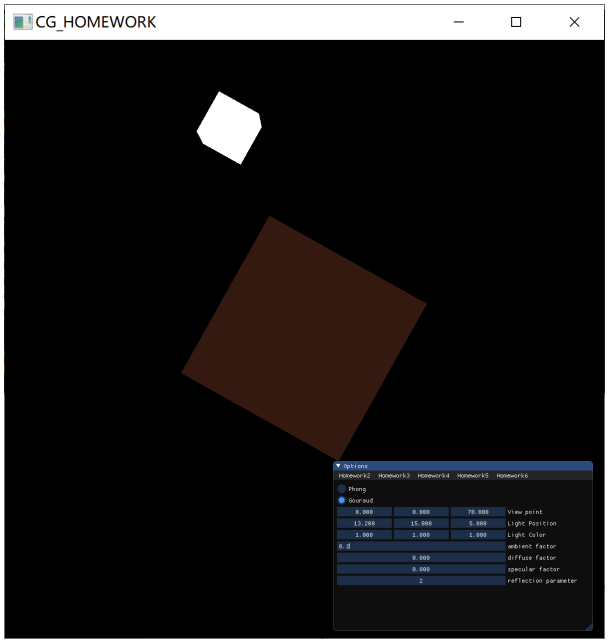


Gouraud Shading的实现

实现和phong Shading类似，只是将phong shading在fragment shader中实现的内容放在vertex shader中实现：

环境光

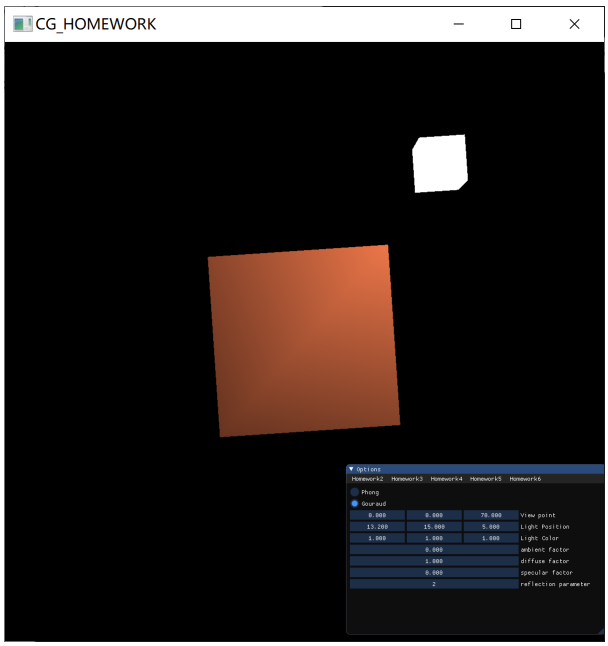
```
vec3 ambient = ambientFactor * lightColor;
```



漫反射

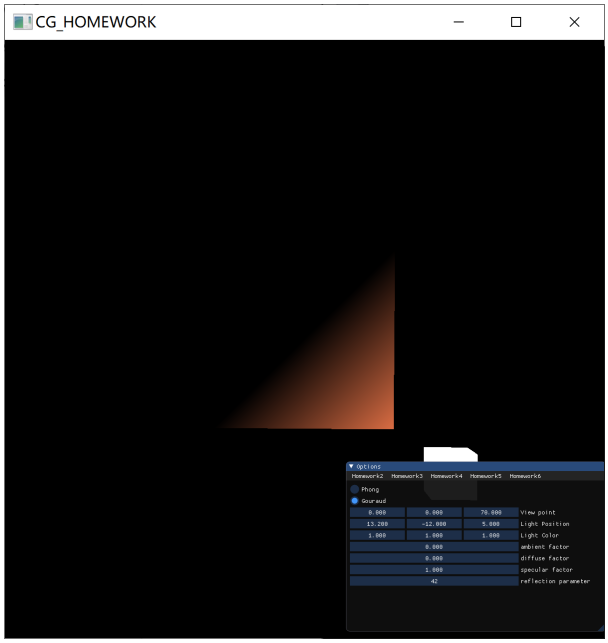
```
vec3 norm = normalize(Normal); // 标准化法向量
vec3 lightDir = normalize(lightPosition - Position); // 获得光照方向
float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = diffuseFactor * diff * lightColor; // 计算漫反射
```

> 注意这里的position是使用原来的顶点坐标乘以model矩阵的得到的， 即从局部坐标系中转化到世界坐标系，因为lightPosition也在世界坐标系中。



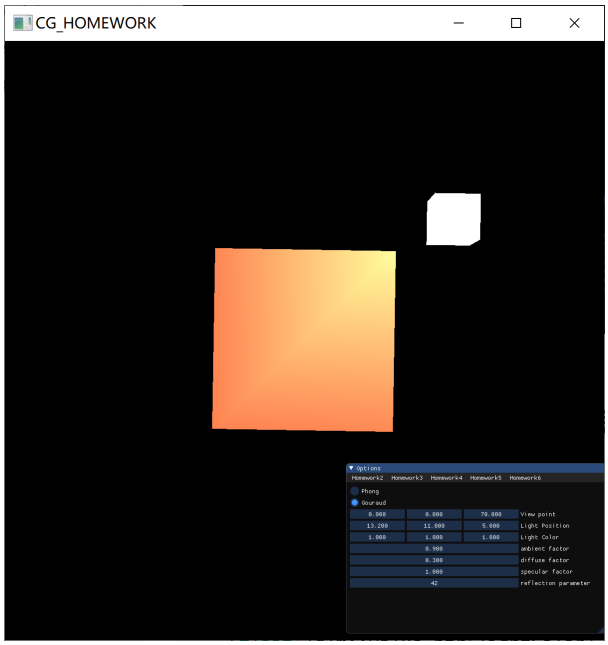
镜面反射

```
vec3 viewDir = normalize(viewPos - Position); // 从观察点到物体位置的方向
vec3 reflectDir = reflect(-lightDir, norm);  // 反射方向
float spec = pow(max(dot(viewDir, reflectDir), 0.0), reflectionPara);
vec3 specular = specularFactor * spec * lightColor;
```



在实验中，当光源移动到某些位置的时候，是没有镜面反射的，原因是在cube的顶点上并没有得到光照的效果，因此仅凭顶点的颜色插值只能插出全黑（在将环境光和漫反射的因子都设置为0的情况下）

将三种作用在顶点上的颜色效果叠加：



使用GUI，使得参数可调节，效果时时更改

修改光照模型相关参数

```
// 光照颜色
ImGui::DragFloat3("Light Color", (float*)glm::value_ptr(lightColor), 0.005f, 0.0f, 1.0f);
// ambient, diffuse, specular, reflectionRate
ImGui::DragFloat("ambient factor", &ambientFactor, 0.1f, 0.0f, 1.0f);
ImGui::DragFloat("diffuse factor", &diffuseFactor, 0.1f, 0.0f, 10.0f);
ImGui::DragFloat("specular factor", &specularFactor, 0.1f, 0.0f, 1.0f);
ImGui::DragInt("reflection parameter", &reflectionPara, 10, 2, 256 );
```

在Phong和Gouraud中二选一

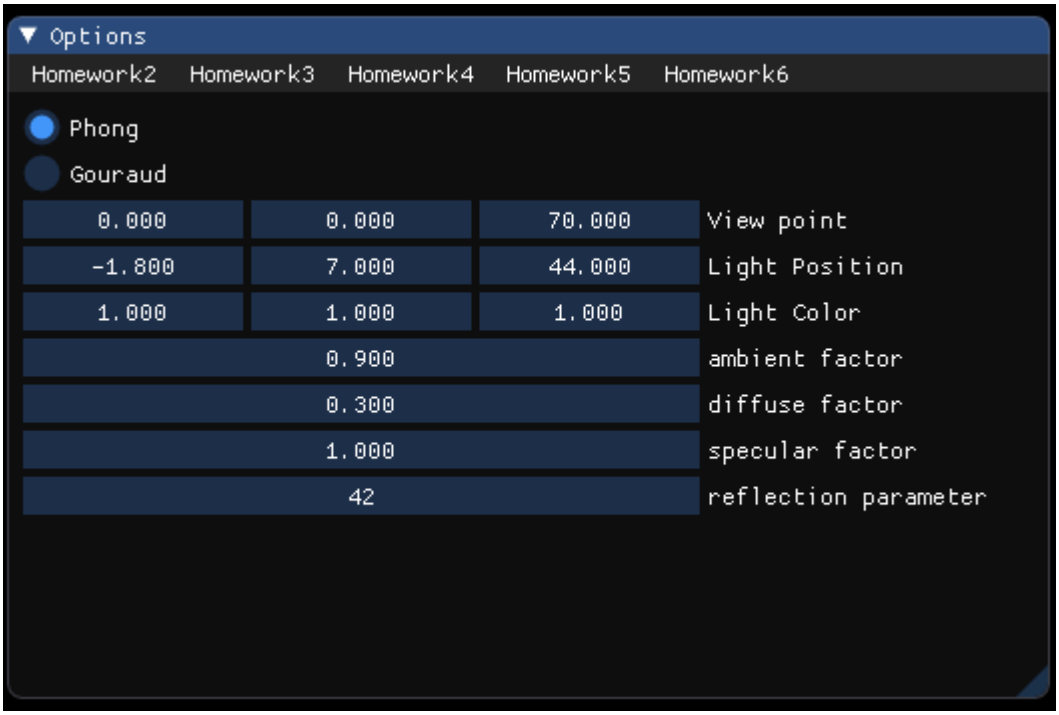
```
ImGui::RadioButton("Phong", &phong, 1);
ImGui::RadioButton("Gouraud", &phong, 0);
```

利用`radioButton`组件，并在不同选择的时候赋予不同的值即可，其中这里`phong`为int类型。

视点移动

由于这里设计的是既可以通过上一次作业的鼠标/键盘调节视点，也可以通过输入参数调节。为了避免二者相互干扰，默认设每次从鼠标键盘的角度获取当前视点。同时在每一帧存储当前视点为`lastViewPos`. 因此在本帧(`viewPos`)和上一帧(`lastViewPos`)不同的情况下，判定为通过在UI上设置数字修改了视点位置：

```
ImGui::DragFloat3("View point", (float*)glm::value_ptr(viewPoint), 1.0f);
// 视点 camera.position
if (lastViewPoint != viewPoint) {
    camera.setCamera(viewPoint);
    lastViewPoint = viewPoint;
}
else {
    lastViewPoint = camera.getPositon();
    viewPoint = camera.getPositon();
}
```



Bonus

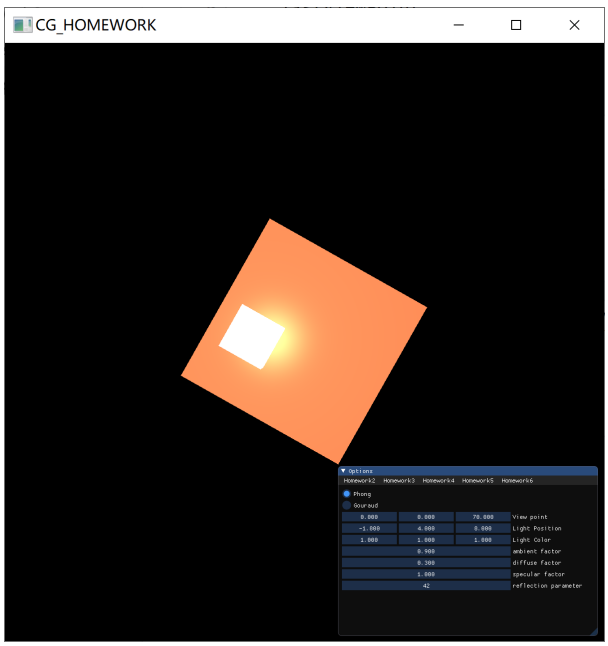
移动光源

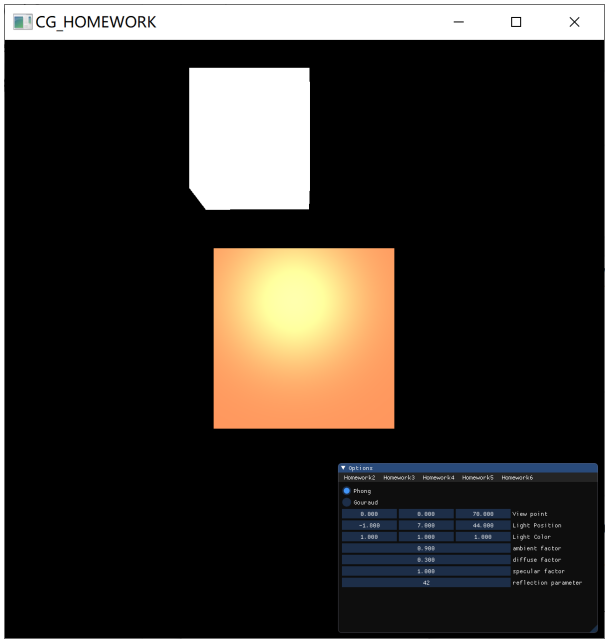
这里只需要在`imGUI`中修改类中的`lightPosition`变量，在每一次渲染时获得光源的最新位置，即可使得光源效果实时更改。

代码如下：

```
ImGui::DragFloat3("Light Position", (float*)glm::value_ptr(lightPos), 1.0f);
```

下面是将光源向视点方向（Z轴正方向）移动的结果：




CG_HOMEWORK
Options
Homework2 Homework3 Homework4 Homework5 Homework6
Phong
Gouraud
0.000 0.000 78.000 View point
-1.000 7.000 44.000 Light Position
1.000 1.000 1.000 Light Color
0.900 ambient factor
0.300 diffuse factor
1.000 specular factor
42 reflection parameter