计算机图形学 Homework 6 - Lights and Shading

陈铭涛
16340024

April 24, 2019

1 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

   Cube 的实现方法与前两次作业相近，在本次作业中实现了`LightingCube`类，其中包含了 Phong 着色器和 Gouraud 着色器。

   Phong 着色法和 Gouraud 着色法都是由环境光照、漫反射光照和镜面光照组合而成的，着色器中除接收变换矩阵外的变量，还接收如下与光照相关的变量：

```
uniform vec3 lightPos;
uniform vec3 lightColor;
uniform vec3 objectColor;
uniform vec3 viewPos;
uniform float ambientStrength;
uniform float specularStrength;
uniform float diffuseStrength;
uniform int shininess;
```

   其中`lightPos`为光源的位置，`lightColor`为光照的颜色，`objectColor` 为物体的颜色，`viewPos`为摄像机的位置，`ambientStrength`为环境光照因子，`diffuseStrength`为漫反射因子，`specularStrength` 为镜面光照因子，`shininess`为镜面的反光度。

   顶点着色器需要输入顶点的法向量数据：

```
layout (location = 1) in vec3 aNormal;
```

Phong 着色法的原理是根据物体顶点法向量，对表面各个像素进行双线性插值，决定像素的颜色。而 Gouraud 着色法则根据物体顶点的法向量对顶点的颜色采用 Phong 模型进行插值，片段着色器填充像素的颜色时则仅根据顶点的颜色和距离进行插值填充。

由于法向量是一个方向向量，当应用了不等比缩放时会导致其在世界坐标中效果不正确，解决的方法是使用法线矩阵乘以原法向量来消除影响，法线矩阵为：

$$N = (M^{-1})^T \tag{1}$$

其中 M 为模型矩阵。

Phong 着色法与 Gouraud 着色法相比，真实度更高，镜面反射的效果更加准确，但是需要对每个像素点进行计算，所需的计算量更大。

Gouraud 着色法在渲染与位置相关度较大的光照效果时效果不佳，比如当镜面光照的高光位于顶点上时，Gouraud 着色法会使高光在表面不自然地扩散。如 Figure1，2：

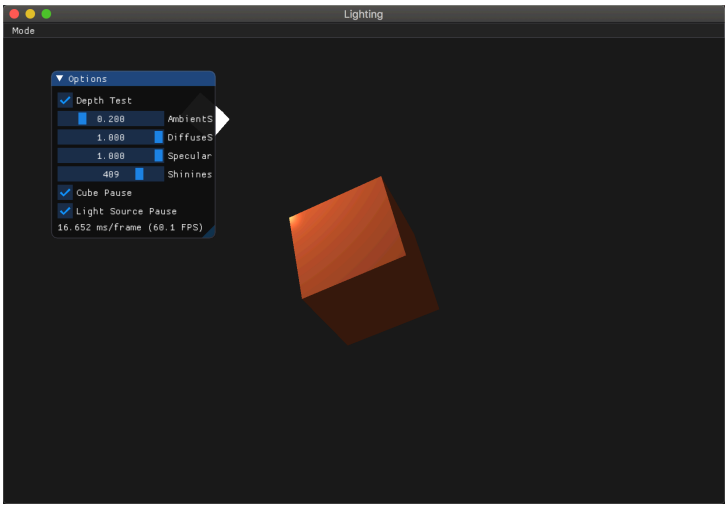

Figure 1: Phong 着色法，高光位于顶点上

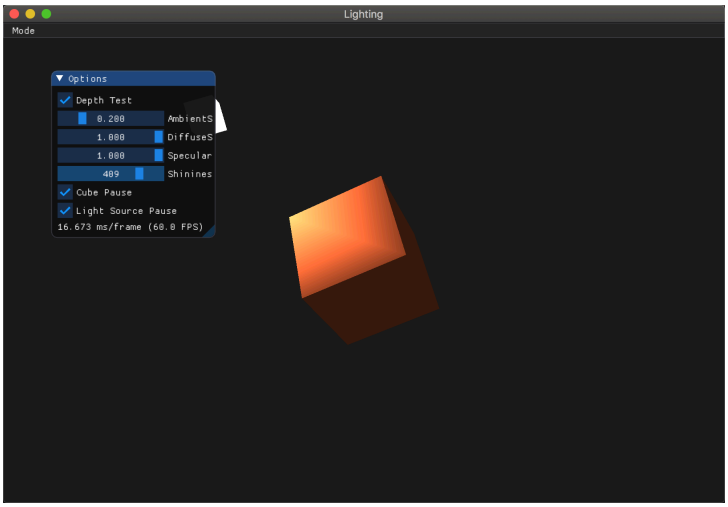

Figure 2: Gouraud 着色法，高光位于顶点上

当高光不扩散到任何顶点上时，Gouraud 着色法将不会渲染出其效果，如 Figure3，4：

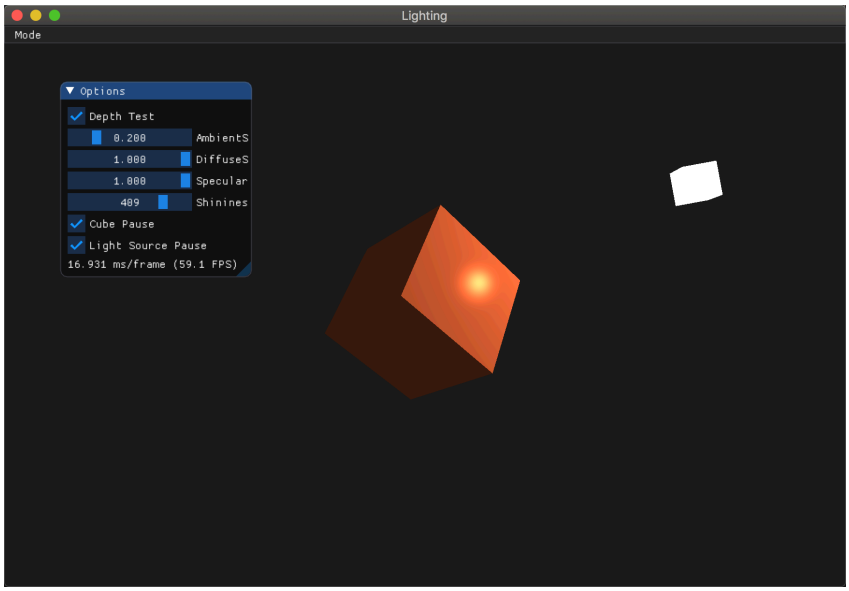

Figure 3: Phong 着色法，高光不位于任何顶点上

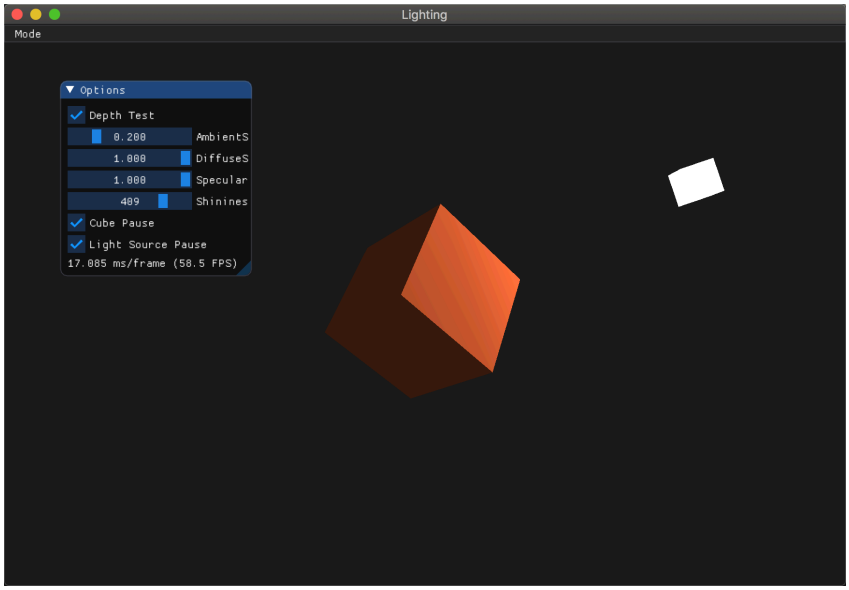

Figure 4: Gouraud 着色法，高光不位于任何顶点上

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

使用 GUI 在右上角添加了用于切换着色方法的菜单，并添加了可以调节 ambient, diffuse 以及 specular 因子和反光度参数的滑块，效果如图：

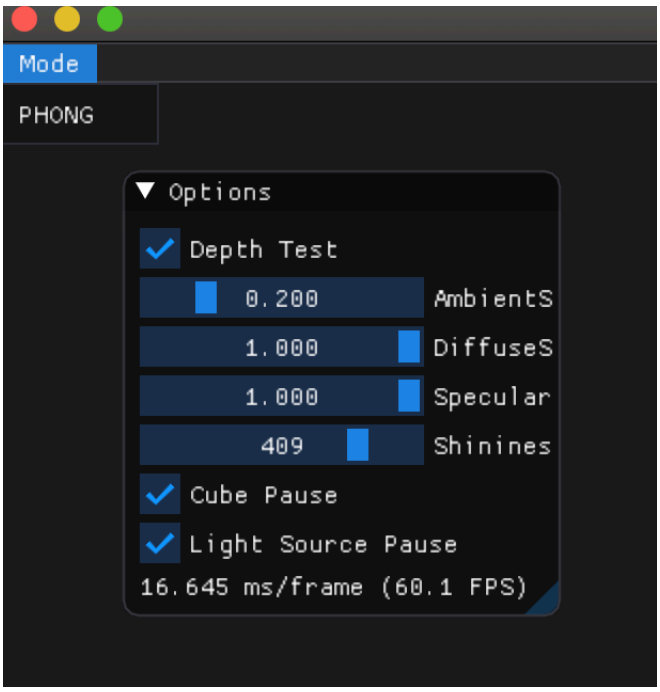

Figure 5: GUI

2 Bonus

1. 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。

   采用了在先前作业中实现的椭圆运动变换使光源在场景中可以椭圆形绕物体移动，将光源用一个纯白的立方体表示，每次在循环中获取立方体所处的位置作为光源坐标。

   物体自身也以一定的速度进行自转，在 GUI 中对光源的运动和物体自转提供了暂停的选项，方便寻找某一特定角度进行截图。

   具体动态效果请见 demo 视频。