《计算机图形学》实验 6 实验报告

学生	刘沅昊	学号	15331220
学院	数据科学与计算机学院	年级专业	16 级软件工程 (数字媒体技术)

实验内容：

Homework

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

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。

实验结果：

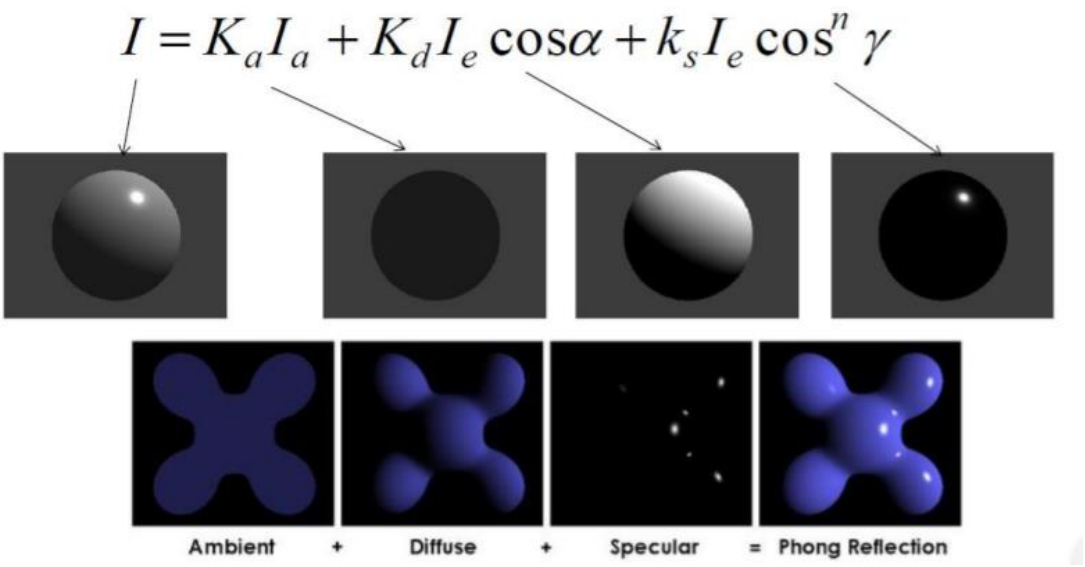
具体效果请查看 record.gif

Basic：

1. **实现 Phong 光照模型：**

   冯光照明模型主要结构由 3 个分量组成：环境光（Ambient），漫反射（Diffuse）和镜面反射（Specular）。

   模型如下：



Phong shading 的实现主要发生在片段着色器，步骤如下：

1. 计算环境光照分量 ambient，由光的颜色乘以一个很小的常量环境因子获得。

```
25          float ambientStrength = 0.1f;
26          vec3 ambient = ambientStrength * lightColor;
```

2. 计算漫反射分量 diffuse,先对 cube 各顶点的法向量 norm 和指向光源的方向向量 light Dir 进行标准化，接着对 norm 和 light Dir 点乘获得漫反射影响（保证反射分量不为负数），最后以光的颜色获得 diffuse。

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

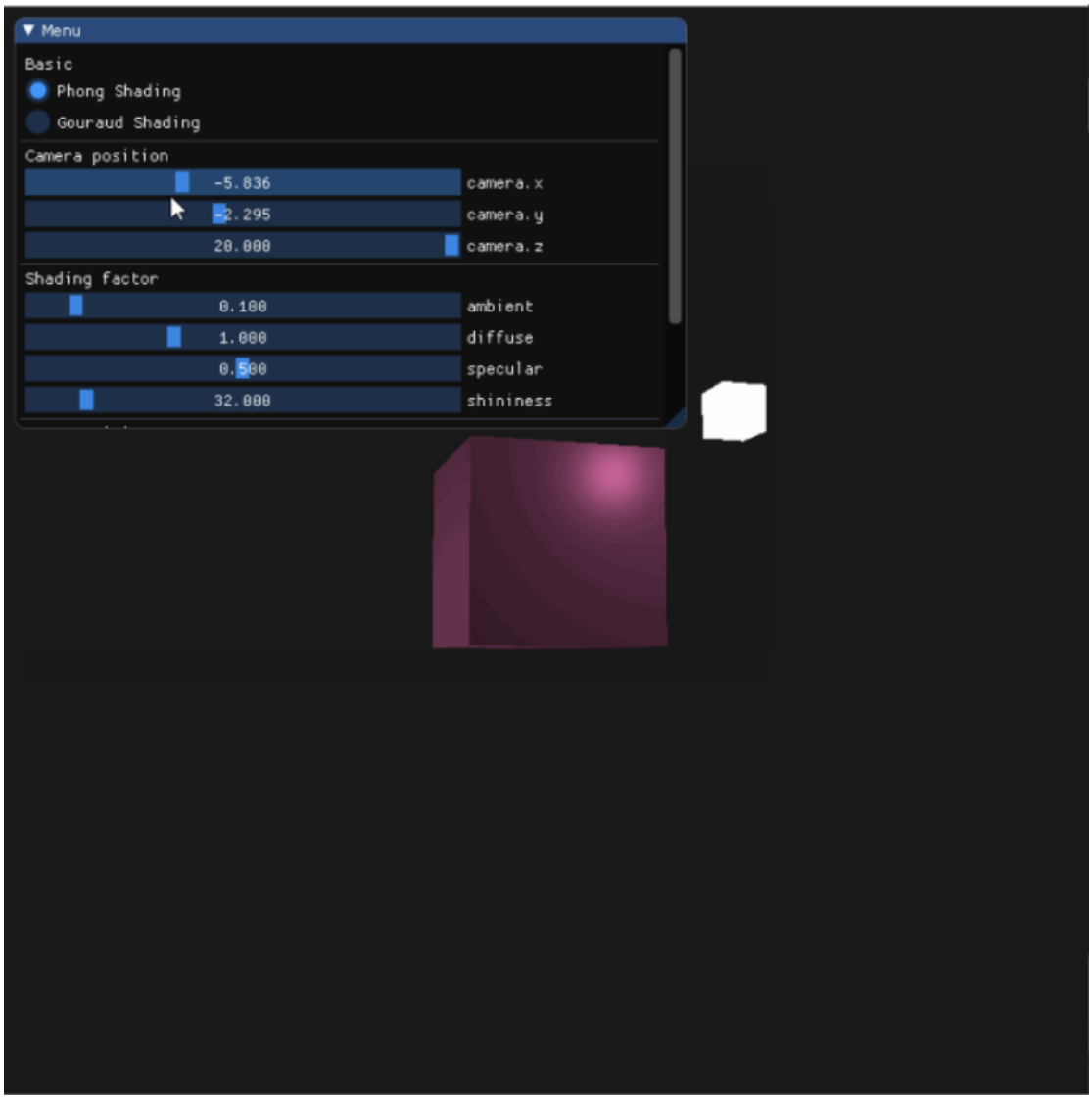
3. 计算镜面关照分量 specular，首先设置镜面强度为 0.5，避免过度影响，接着计算视线方向向量 viewDir 和光沿着顶点法向量的反射向量 reflectDir，然后计算 viewDir 与 reflectDir 点乘并取多次幂，最后与光的颜色和镜面强度相乘得到 specular

```
33          float specularStrength = 0.5f;
34          vec3 viewDir = normalize(viewPos - FragPos);
35          vec3 reflectDir = reflect(-lightDir, norm);
36          float spec = pow(max(dot(viewDir, reflectDir), 0.0), shininessFactor);
37          vec3 specular = specularStrength * spec * lightColor;
```

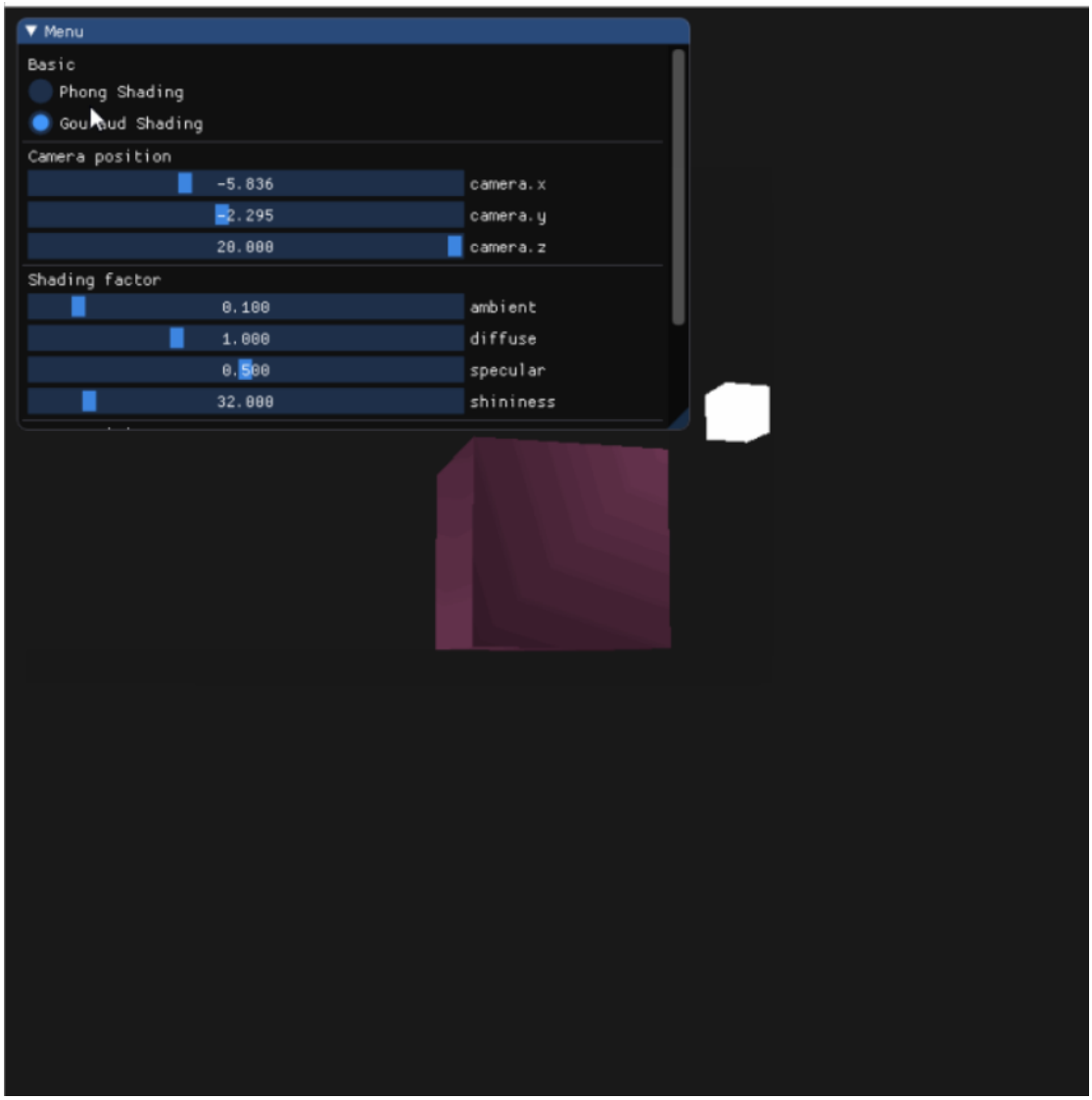
4. 将三个分量因子相加后乘以物体本身的颜色，就得到了最终的着色效果

```
39          vec3 result = (ambient*ambientFactor + diffuse*diffuseFactor + specular*specularFactor) * objectColor;
40          FragColor = vec4(result, 1.0);
```

Gouraud shading 与 Phong shading 相似，但主要发生在顶点着色器，同样计算三个分量因子，再作为光照颜色传给片段着色器



图表 1 Phong shading 结果



图表 2 Gouraud shading 效果

2. 使用 GUI，使参数可以调节，效果实时更改

```
184         {
185             ImGui::Begin("Menu");
186             ImGui::Text("Basic");
187             ImGui::RadioButton("Phong Shading", &shading_choose, 1);
188             ImGui::RadioButton("Gouraud Shading", &shading_choose, 2);
189             ImGui::Separator();
190             ImGui::Text("Camera position");
191             ImGui::SliderFloat("camera.x", &camPos[0], -20.0f, 20.0f);
192             ImGui::SliderFloat("camera.y", &camPos[1], -20.0f, 20.0f);
193             ImGui::SliderFloat("camera.z", &camPos[2], -20.0f, 20.0f);
194             ImGui::Separator();
195             ImGui::Text("Shading factor");
196             ImGui::SliderFloat("ambient", &ambient, 0.0f, 1.0f);
197             ImGui::SliderFloat("diffuse", &diffuse, 0.0f, 3.0f);
198             ImGui::SliderFloat("specular", &specular, 0.0f, 1.0f);
199             ImGui::SliderFloat("shininess", &shininess, 0.0f, 256.0f);
200             ImGui::Separator();
201             ImGui::Text("Lamp position");
202             ImGui::Checkbox("Lamp Moving", &is_lamp_moving);
203             ImGui::SliderFloat("lamp.x", &light_pos[0], -20.0f, 20.0f);
204             ImGui::SliderFloat("lamp.y", &light_pos[1], -20.0f, 20.0f);
205             ImGui::SliderFloat("lamp.z", &light_pos[2], -20.0f, 20.0f);
206 
207             ImGui::End();
208         }
```

设置 GUI 可以选择是 Phong shading 还是 Gouraud shading，可以调节相机位置，渲染的参数和光源的位置。

Bonus：

```
243         if (is_lamp_moving) {
244             light_pos[0] = lightPos.x = 1.0f + sin(glfwGetTime()) * 2.0f;
245             light_pos[1] = lightPos.y = sin(glfwGetTime() / 2.0f) * 1.0f;
246         }
```

通过简单的函数让光源可以来回移动。