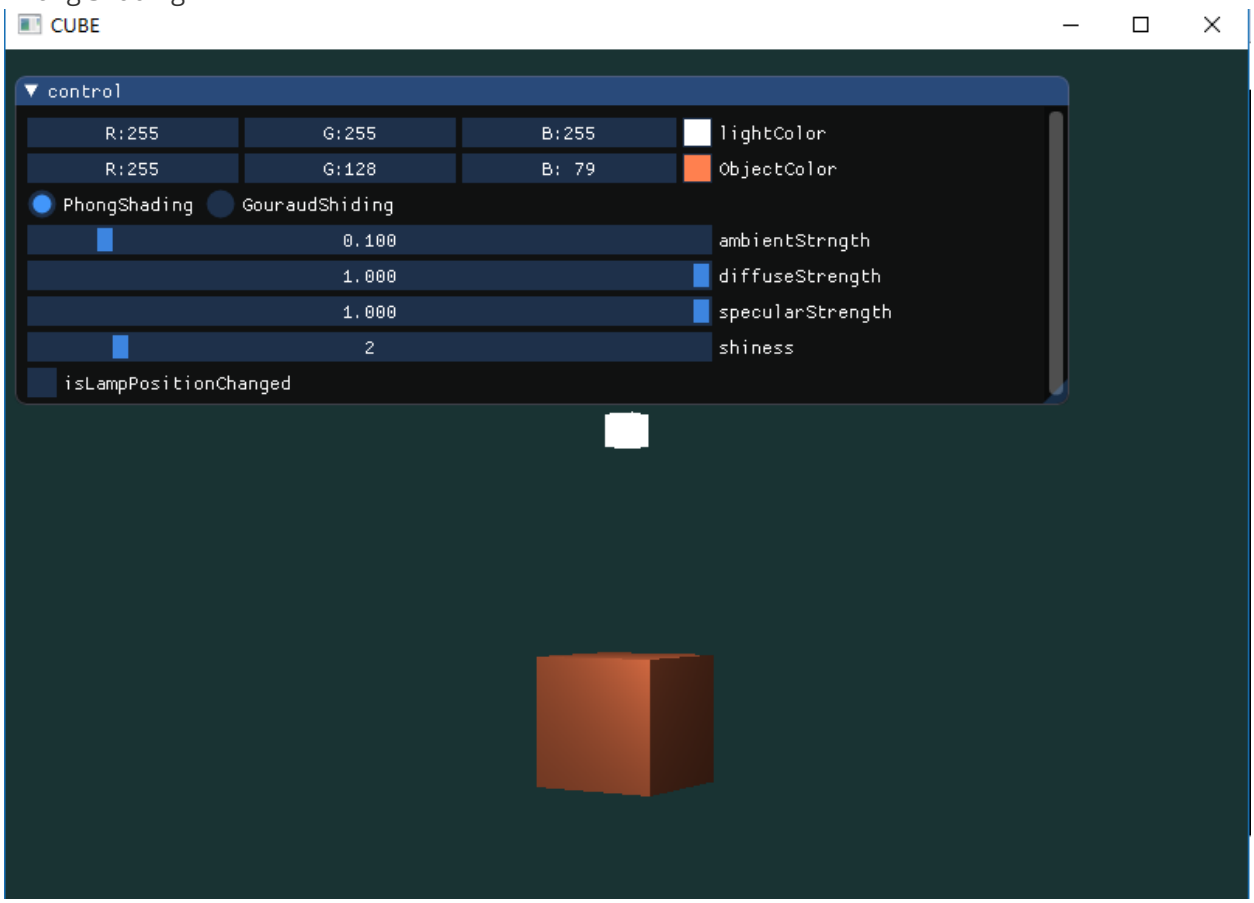
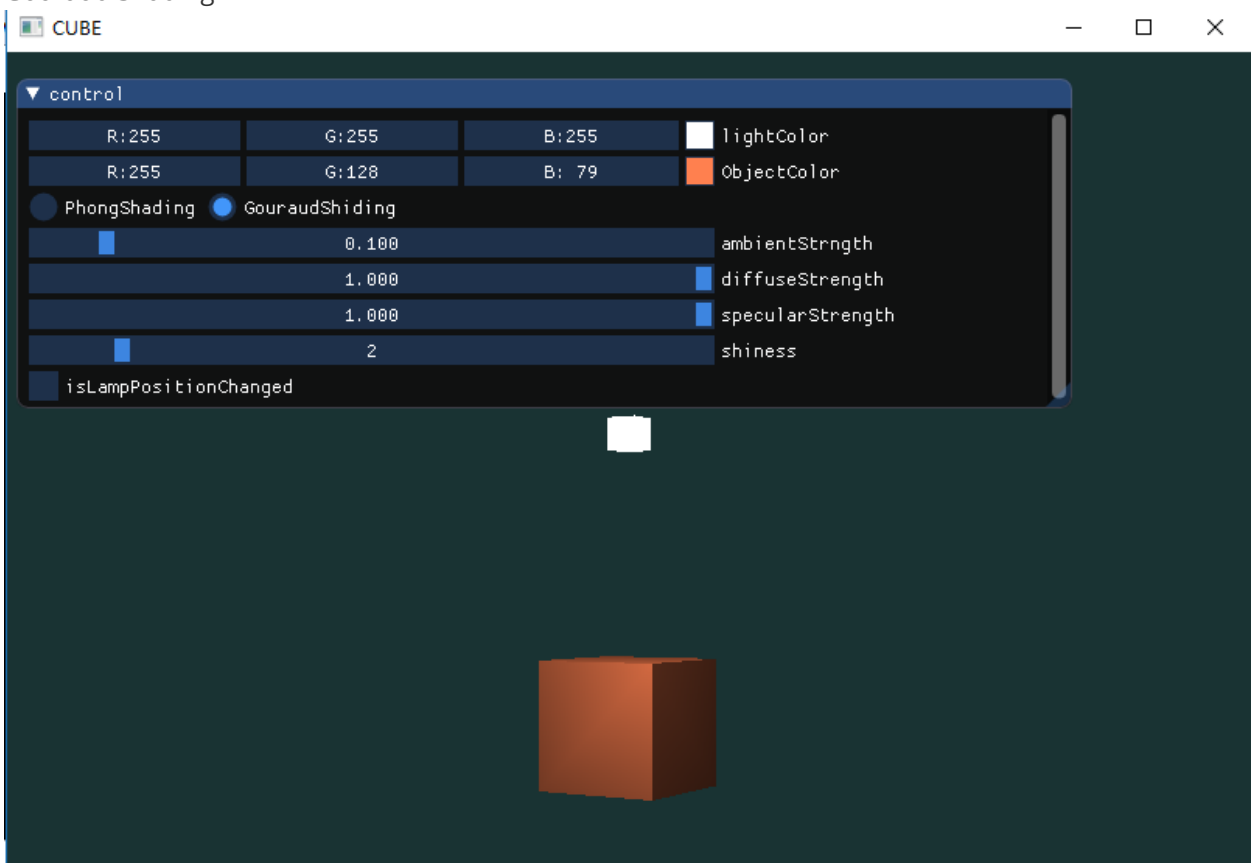
计算机图形学-平时作业六

一、运行结果

- Phong Shading



- Gouraud Shading



二、两种Shading的实现原理

这两个Shading的实现都是基于 **Phong光照明模型** 。其中Phong Shading在 **FragmentShader** 中实现Phong光照明模型，而Gouraud Shading在 **VerticesShader** 中实现Phong光照明模型。

Phong光照明模型

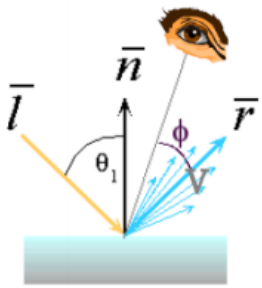
Phong光照明模型中的光照是指 **环境光+漫反射光+镜面反射光**的结果。最后将得到的光的结果与物体颜色做点积的到物体应该显示的颜色。

1. **环境光**
   计算公式： $I_{ambient} = k_{ambientStrength} I_{incidentLight}$
2. **反射**

   根据**朗博余弦定理**进行漫反射光照结果计算。
   计算公式：$I_{diffuse} = k_{diffuseStrength} I_{incidentLight} cos < l, n >$ 其中l为所求光照的物体表面的点指向光源的方向的单位向量，n为物体表面的单位法向量。

3. **镜面反射**
   计算公式如下图所示：

$$I_{specular} = k_s I_{light} (\bar{v} \cdot \bar{r})^{n_{shiny}}$$



- V is the unit vector towards the viewer
- r is the ideal reflectance direction
- K_s: specular component
- I_{light}: incoming light intensity
- n_{shiny}: purely empirical constant, varies rate of falloff(材质发光常数，值越大，表面越接近镜面，高光面积越小。)

4. **物体颜色显示**
   物体的颜色显示为前面三个的计算结果的和与物体颜色的点积。

三、相关函数

1. 实现两种shading的切换
   创建两个shaderProgram分别对应Phong Shading和Gouraud Shading，要是用哪一个 shading 就使用相对应的shaderProgram。为了简化创建shaderProgram的重复代码，实现函数**createShader**。

```
unsigned int createShader(const char *vertexShaderSource, const char *fragmentShaderSource);
```

   此函数返回shaderProgram。
2. 为了简化对shader中生命的 **uniform** 变量进行赋值实现下面的三个函数

```
void setShaderMat4(unsigned int ID, const std::string &name, const glm::mat4 &value);
void setShaderVec3(unsigned int ID, const std::string &name, const glm::vec3 &value);
void setShaderFloat(unsigned int ID, const std::string &name, const float &value);
```

   上述三个函数中输入的参数 **ID** 是指 **ShaderProgram** ;而 **name** 代表的是 **uniform** 变量的变量名; **value** 是指将对应变量赋值的值。
3. 实现光源随着时间旋转

```
    lightPos.x =  sin(glfwGetTime()) * 3.0f;
    lightPos.z = cos(glfwGetTime()) * 3.0f;
```

   根据时间的变换修改 **lightPos** 的值，之后再 **model** 矩阵中进行操作来修改光源的位置。

四、演示视频地址

视频链接