Basic:

1. 绘制一个 cube，vertex 的数据格式如下所示：

```
float vertices[] = {
    -2.0f, -2.0f, -2.0f,  0.4f, 0.5f, 0.8f,  0.0f, 0.0f, -1.0f,
    2.0f, -2.0f, -2.0f,  0.4f, 0.5f, 0.8f,  0.0f, 0.0f, -1.0f,
    2.0f,  2.0f, -2.0f,  0.4f, 0.5f, 0.8f,  0.0f, 0.0f, -1.0f,
    2.0f,  2.0f, -2.0f,  0.4f, 0.5f, 0.8f,  0.0f, 0.0f, -1.0f,
    -2.0f,  2.0f, -2.0f,  0.4f, 0.5f, 0.8f,  0.0f, 0.0f, -1.0f,
    -2.0f, -2.0f, -2.0f,  0.4f, 0.5f, 0.8f,  0.0f, 0.0f, -1.0f,
```

前三个为顶点坐标，中间 3 个为面的颜色（本作业中忽略掉），最后为法向量；

2.分别实现 Phong Shader 和 Gouraud Shader：

```
// shader
Shader phongShader("phong.vs", "phong.fs");
Shader gouraudShader("gouraud.vs", "gouraud.fs");
Shader lightShader("light.vs", "light.fs");
```

Phong Shading：

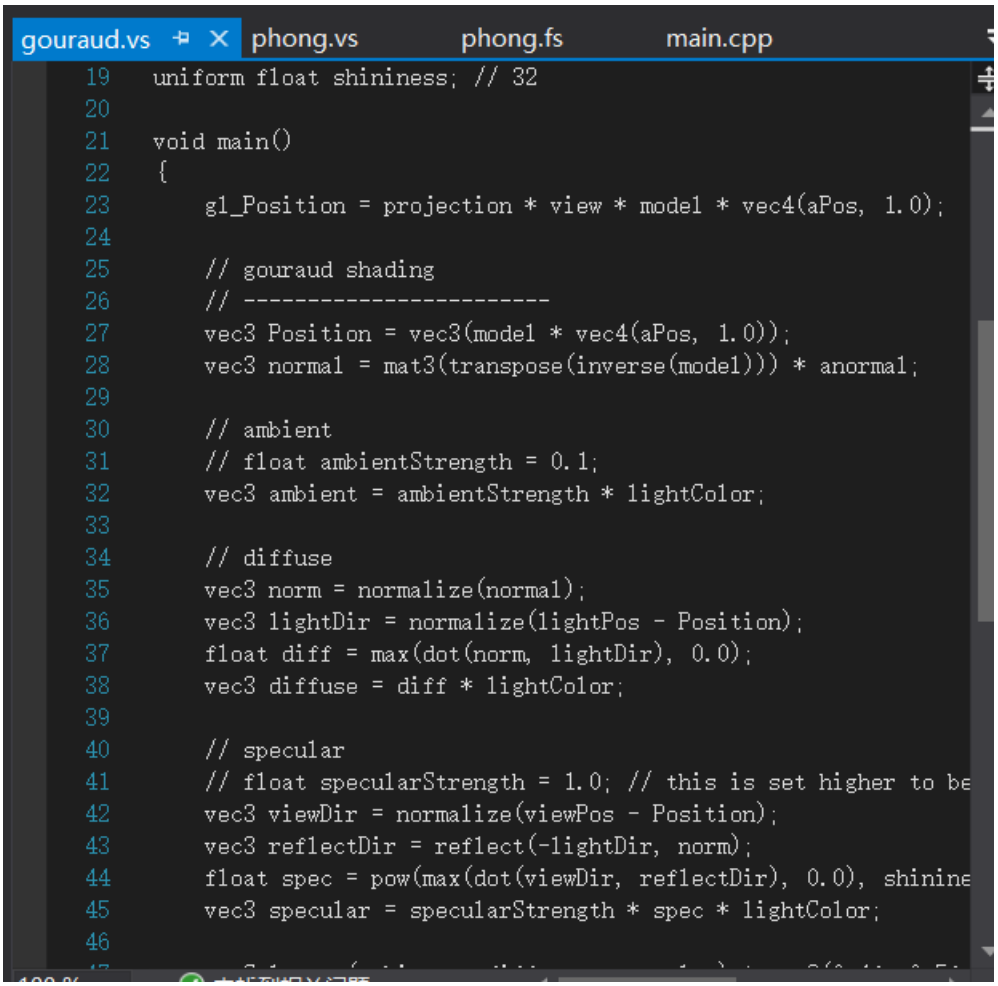
按照教程中的写法完成 vs 和 fs 文件即可，其中，phongShading 是在 frag 着色器中进行光照颜色的计算的：

vs　phong.fs　main.cpp　camera.hpp

```
uniform float ambientStrength;
uniform float specularStrength;
uniform float shininess; // 32

void main()
{
    // FragColor = texture(texture1, TexCoord);
    // ambient
    // float ambientStrength = 0.1;
    vec3 ambient = ambientStrength * lightColor;

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

    // specular
    // float specularStrength = 0.5;
    vec3 viewDir = normalize(viewPos - FragPos);
    vec3 reflectDir = reflect(-lightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), shinine
    vec3 specular = specularStrength * spec * lightColor;

    vec3 result = (ambient + diffuse + specular) * outColor;
    FragColor = vec4(result, 1.0);
}
```

Gouraud Shading：

Gouraud Shading 的颜色计算发生在顶点着色器中：



计算完 color 之后传给 frag 着色器：

```
outColor = (ambient + diffuse + specular) * vec3(0.4f, 0.5f, 0.8f);
```

实现原理：

Gouraud shading 是先计算顶点的颜色，然后根据顶点信息，对中间的颜色进行二维的插值。因此其代码直接在顶点着色器中实现。

Phong shading 对每个片段（fragment）计算光照，点的法向量通过顶点的法向量插值得到。因此其代码在片段着色器中实现，其开销会比 Gouraud 要大。

3. 设置 GUI 以修改参数：

```
ImGui::SliderFloat("ambient", &ambient, 0.0f, 1.0f);
ImGui::SliderFloat("specular", &specular, 0.0f, 2.0f);
ImGui::SliderInt("shininess", &shininess, 10, 50);
```

因为 diffuse 是计算求出来的，因此无需对其进行改变调节。

用一个引用来切换两种 Shader：

```
Shader& shader = phongShader;
```

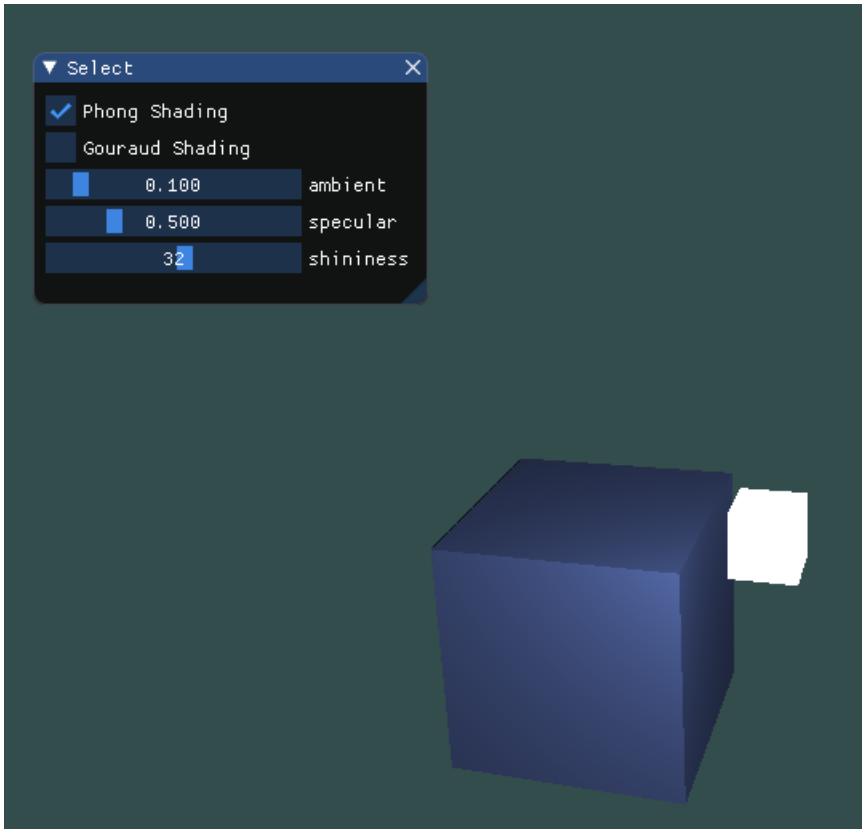
```
if (select1) {
    shader = phongShader;
}
else if (select2) {
    shader = gouraudShader;
}
```

Bonus:

更改 lightPos 的值来使光源移动，使其绕中间的 cube 旋转：

```
lightPos.x = 6 * cos(glfwGetTime() * 1.0f);
lightPos.z = 6 * sin(glfwGetTime() * 1.0f);
```

结果截图：



通过 GUI 可以切换两种 shading，并且调节参数。