韩宇潇 16340069

整体代码结构：

```
class HW6
{
public:
    HW6();
    ~HW6();
    void HW6LinkShader();
    unsigned int getShaderProgramPhone();
    unsigned int getShaderProgramGouraud();
    void draw();
    void Phong(Camera camera, bool, float ,float ,float, int, float);
    void Gouraud(Camera camera, bool, float, float, float, int, float);

private:
    unsigned int shaderProgramPhone, shaderProgramGouraud, shaderProgramLight;
    unsigned int VAO, lightVAO, VBO;
};
```

HW6LinkShader():

用来构建着色器，将在本次作业中用到的顶点着色器和片段着色器连接，在接下来的调用过程中方便使用。在本次作业中构建了 3 个着色器，Phong 光照模型、Gouraud 光照模型、光源模型。

getShaderProgramPhone():

getShaderProgramGouraud():

返回构建的着色器到 main 函数。

draw():

用于将画 cube 的数组存入缓存中，方便在接下来的画正方体使用。

Phong():

实现 Phong Shading，传入光源是否为动态、ambient 因子、diffuse 因子、specular 因子、反光度、光源静止时位置坐标

Gouraud():

实现 Gouraud Shading，传入光源是否为动态、ambient 因子、diffuse 因子、specular 因子、反光度、光源静止时位置坐标

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

Phong Shading：

实验思路：

实现 Phong Shading，主要在于构建顶点着色器和片段着色器

顶点着色器：实现的思路和之前的基本相同，传入位置，法向量，颜色，计算出顶点变换后的位置，把位置，法向量，颜色传递到片段着色器中。

```
const char *HW6vertexShaderSourcePhone =
"#version 330 core\n"
"layout (location = 0) in vec3 aPos;\n"
"layout (location = 1) in vec3 aNormal;\n"
"layout (location = 2) in vec3 aColor; \n"
"out vec3 ourColor;\n"

"uniform mat4 model;\n"
"uniform mat4 view;\n"
"uniform mat4 projection;\n"
"out vec3 Normal;\n"
"out vec3 FragPos;\n"
"void main()\n"
"{\n"
"   gl_Position = projection * view * model * vec4(aPos, 1.0);\n"
"   FragPos = vec3(model * vec4(aPos, 1.0));\n"
"   Normal = mat3(transpose(inverse(model))) * aNormal;\n"
"   ourColor = aColor;\n "
"}\n\0";
```

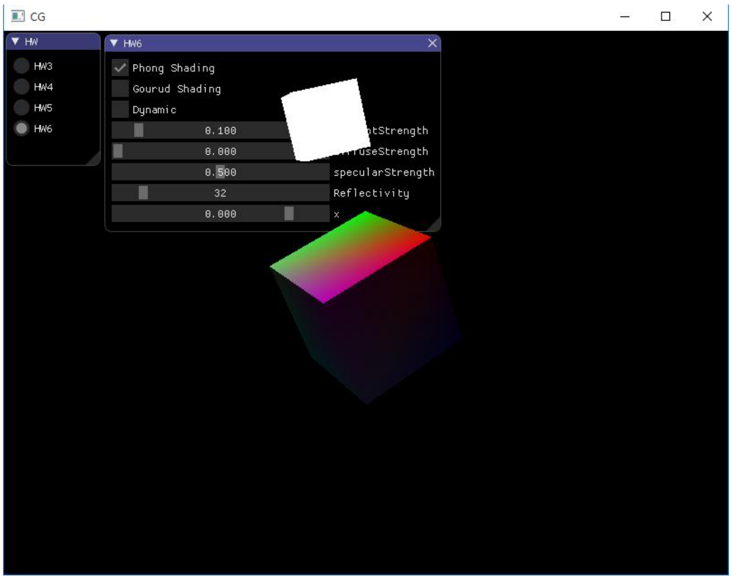
片段着色器：主要实现环境光照、漫反射光照、镜面光照。

```
const char *HW6vertexShaderSourceGouraud =
"#version 330 core\n"
"layout(location = 0) in vec3 aPos;"
"layout(location = 1) in vec3 aNormal;"
"layout (location = 2) in vec3 aColor; \n"
"out vec3 ourColor;\n"

"out vec3 LightingColor;"
"uniform float ambientStrength;"
"uniform float diffuseStrength;"
"uniform float specularStrength;"
"uniform int Reflectivity;"
"uniform vec3 lightPos;"
"uniform vec3 viewPos;"
"uniform vec3 lightColor;"

"uniform mat4 model;"
"uniform mat4 view;"
"uniform mat4 projection;"

"void main()"
"{"
"   gl_Position = projection * view * model * vec4(aPos, 1.0);"

"   vec3 Position = vec3(model * vec4(aPos, 1.0));"
"   vec3 Normal = mat3(transpose(inverse(model))) * aNormal;"

"   vec3 ambient = ambientStrength * lightColor;"

"   vec3 norm = normalize(Normal);"
"   vec3 lightDir = normalize(lightPos - Position);"
"   float diff = max(dot(norm, lightDir), diffuseStrength);"
"   vec3 diffuse = diff * lightColor;"

"   vec3 viewDir = normalize(viewPos - Position);"
"   vec3 reflectDir = reflect(-lightDir, norm);"
"   float spec = pow(max(dot(viewDir, reflectDir), 0.0), Reflectivity);"
"   vec3 specular = specularStrength * spec * lightColor;"

"   LightingColor = ambient + diffuse + specular;"
"   ourColor = aColor;"
"}";
```

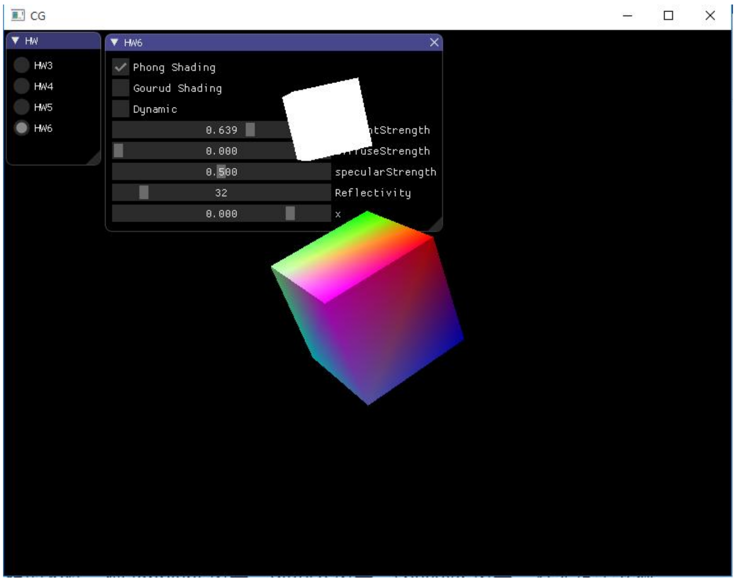
环境光照：是实现了一个简化的全局照明模型，为物体的最终颜色增加了很小的常量（光照）颜色，使场景中没有直接的光源也能看起来存在有一些发散的光。效果由 ambient 因子控制，调整增加的常量的大小。

漫反射光照：使物体上与光线方向越接近的片段能从光源处获得更多的亮度。通过计算和法向量间的夹角，判断亮度的显示，当夹角小于 diffuse 因子时，添加一个常量的漫反射光照。
镜面光照：添加镜面高光，是依据光的方向向量和物体的法向量来决定的，实现效果为当我们去看光被物体所反射的那个方向的时候，我们会看到一个高光。反光效果受 specular 因子影响，反光度受反光度参数影响。

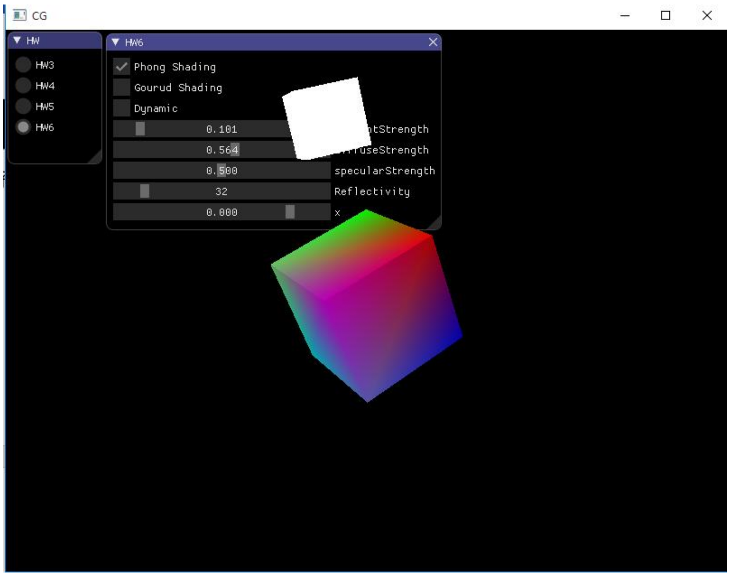
实验结果：
初始状态：



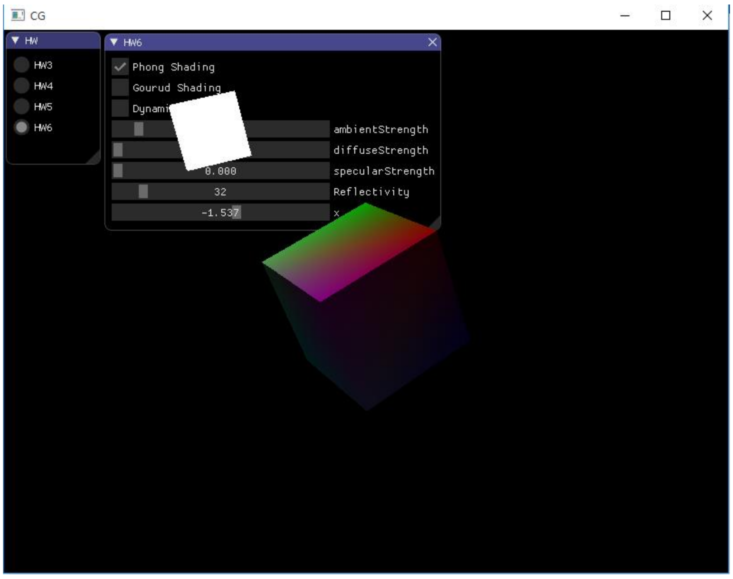
调整 ambient 因子：

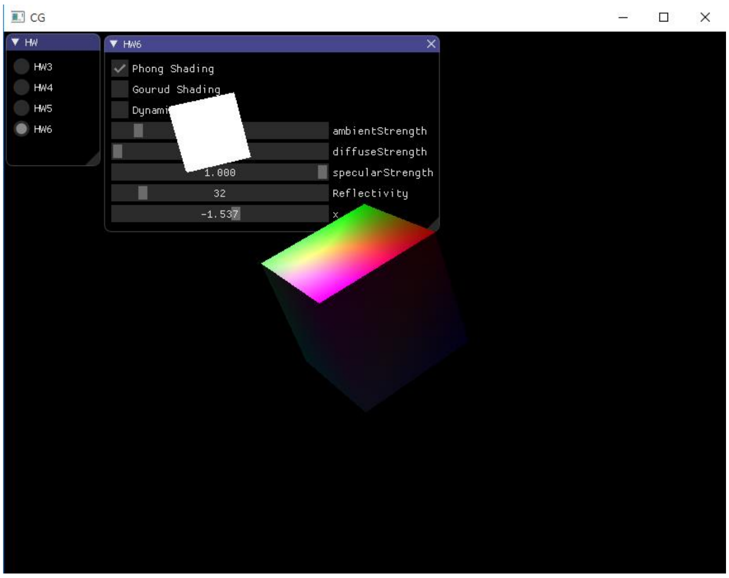


调整 diffuse 因子：

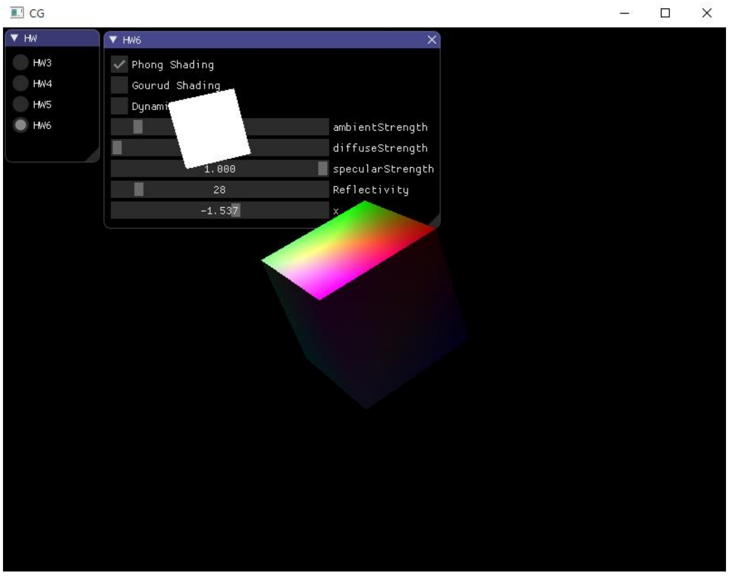


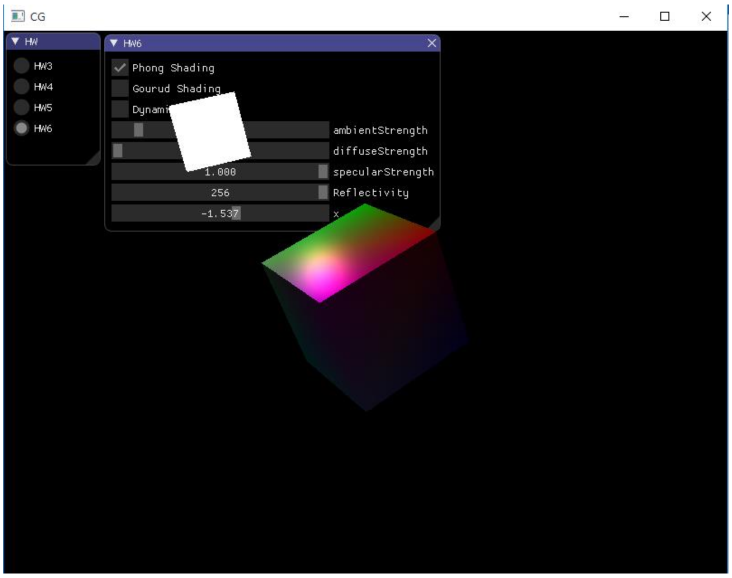
调整 specular 因子：





调整反光度：


CG
HW
HW3
HW4
HW5
HW6
HW6
Phong Shading
Gourud Shading
ambientStrength
diffuseStrength
1.000
specularStrength
28
Reflectivity
-1.537
x


CG
HW
HW3
HW4
HW5
HW6
HW6
Phong Shading
Gourud Shading
ambientStrength
diffuseStrength
1.000
specularStrength
256
Reflectivity
-1.537
x

Gouraud Shading：
实验思路：
实现 Gouraud Shading，主要在于构建顶点着色器和片段着色器，和 Phong Shading 的区别在于 Gouraud Shading 是在顶点着色器中实现光照效果，通过插值实现最终效果。优点是更高效，开销小，缺点是光照看起来不会非常真实，在反光度最高出现光斑时，会因为插值导致显示有问题。
实现方法基本相同，就不重复解释了。

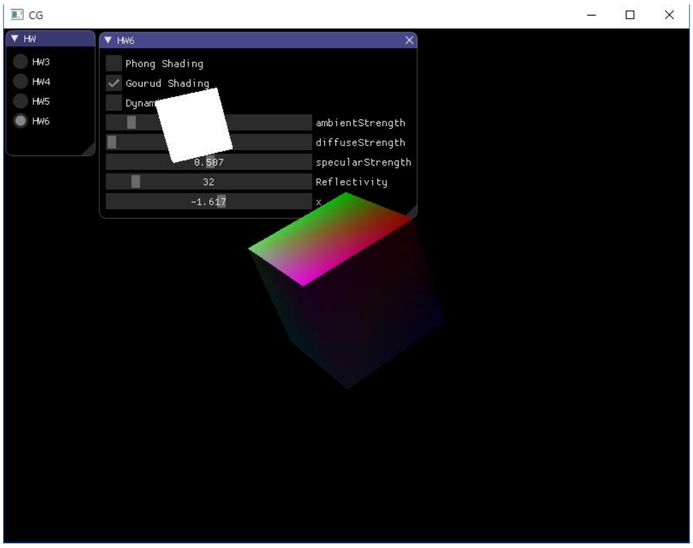
顶点着色器：
```
const char *HW6fragmentShaderSourcePhone =
"#version 330 core\n"
"in vec3 Normal;\n"
"in vec3 FragPos;\n"

"out vec4 FragColor;\n"

"uniform float ambientStrength;\n"
"uniform float diffuseStrength;\n"
"uniform float specularStrength;\n"
"uniform int Reflectivity;\n"

"uniform vec3 lightPos;\n"
"uniform vec3 viewPos;\n"

"in vec3 ourColor;\n"

"uniform vec3 lightColor;\n "

"void main()\n"
"{\n"
"   vec3 ambient = ambientStrength * lightColor;\n"

"   vec3 norm = normalize(Normal);\n"
"   vec3 lightDir = normalize(lightPos - FragPos);\n"
"   float diff = max(dot(norm, lightDir), diffuseStrength);\n"
"   vec3 diffuse = diff * lightColor;\n"

"   vec3 viewDir = normalize(viewPos - FragPos);\n"
"   vec3 reflectDir = reflect(-lightDir, norm);\n"
"   float spec = pow(max(dot(viewDir, reflectDir), 0.0), Reflectivity);\n"
"   vec3 specular = specularStrength * spec * lightColor;"

"   vec3 result = (ambient + diffuse + specular) * ourColor;"
"   FragColor = vec4(result, 1.0);\n"
"}\n\0";
```

片段着色器

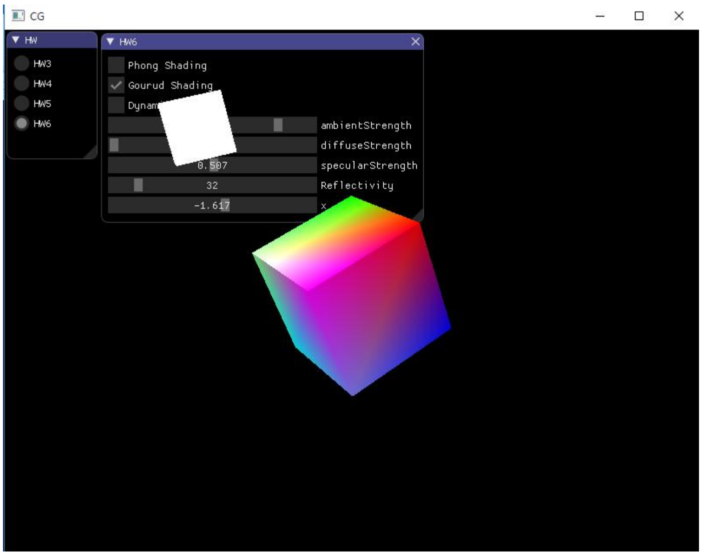
```
const char *HW6fragmentShaderSourceGouraud =
"#version 330 core\n"
"out vec4 FragColor;"

"in vec3 LightingColor;"

"in vec3 ourColor;\n"

"void main()"
"{"
"   FragColor = vec4(LightingColor * ourColor, 1.0);"
"}";
```

实验结果：

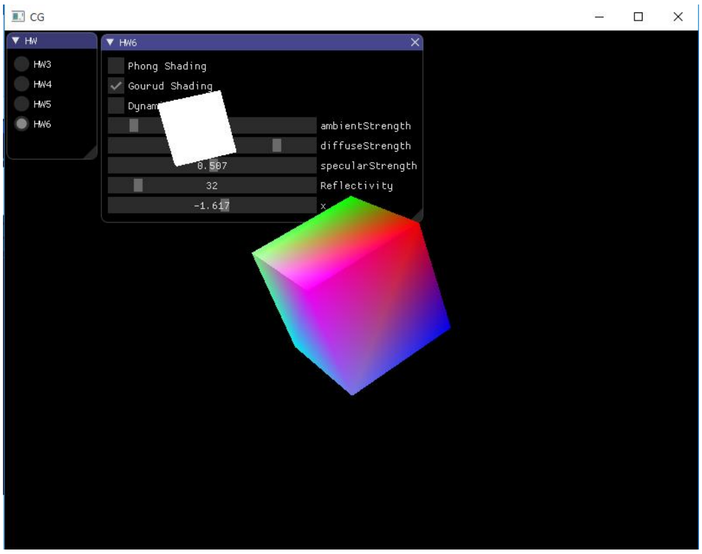
初始状态：



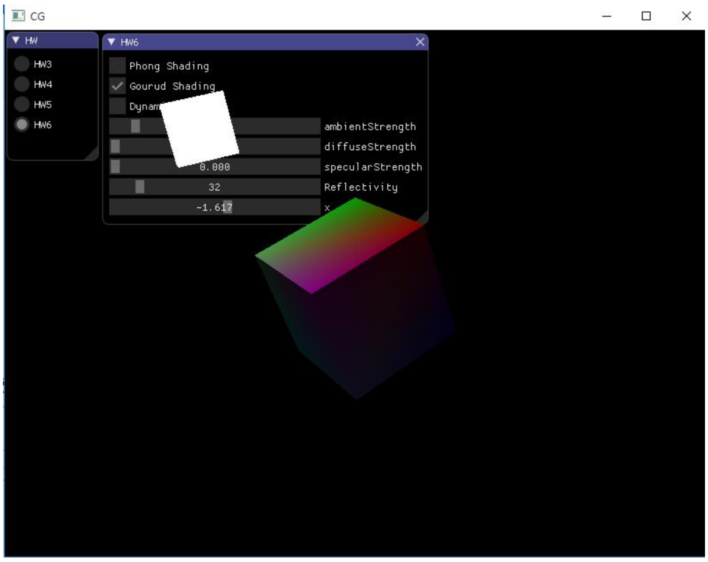
调整 ambient 因子：

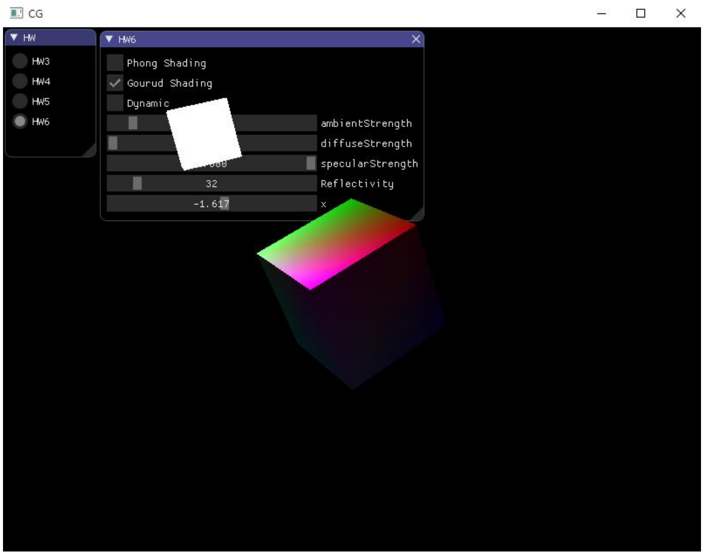


调整 diffuse 因子：

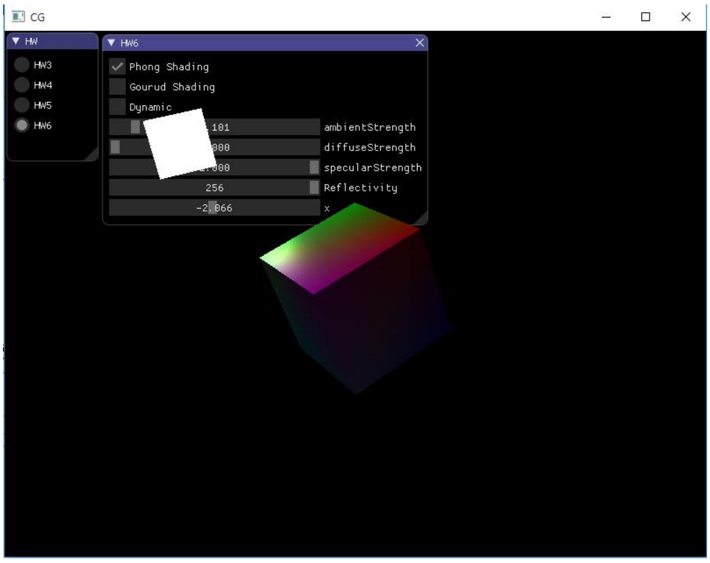


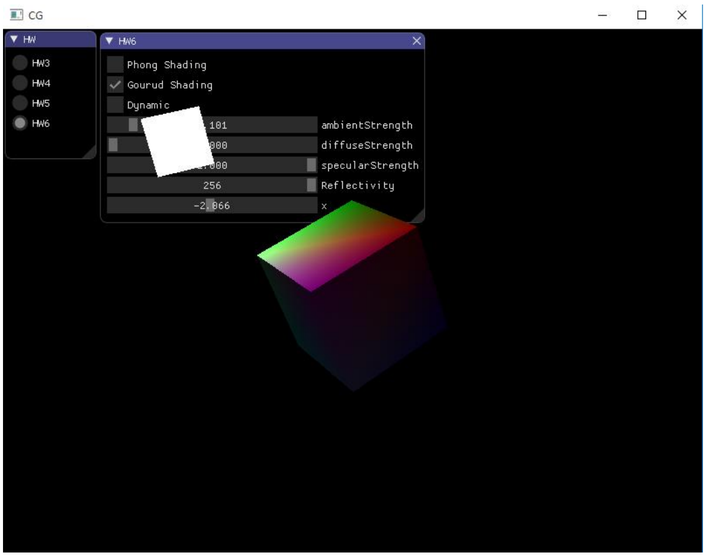
调整 specular 因子：



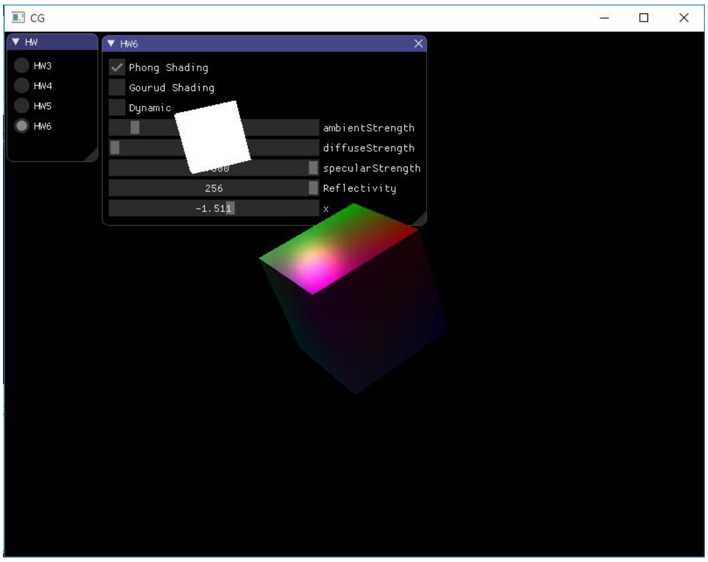


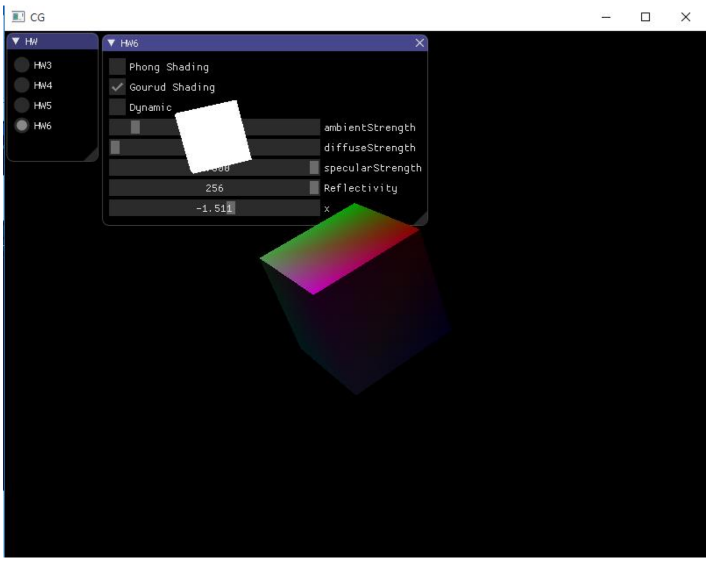
调整反光度，和 Phong Shading 进行对比：
光斑处于正方体顶点时，Gouraud Shading 由于通过顶点插值，会将光斑放大





光斑处于正方体中间时，Gouraud Shading 由于通过顶点插值，会导致没有光斑





Bonus:

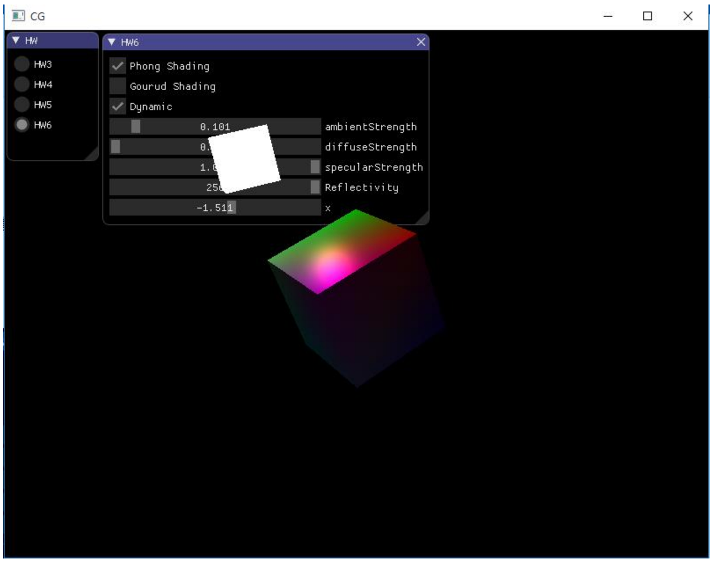
1. 当前光源为静止状态，尝试使光源在场景中来回移动，光照效果实时更改。

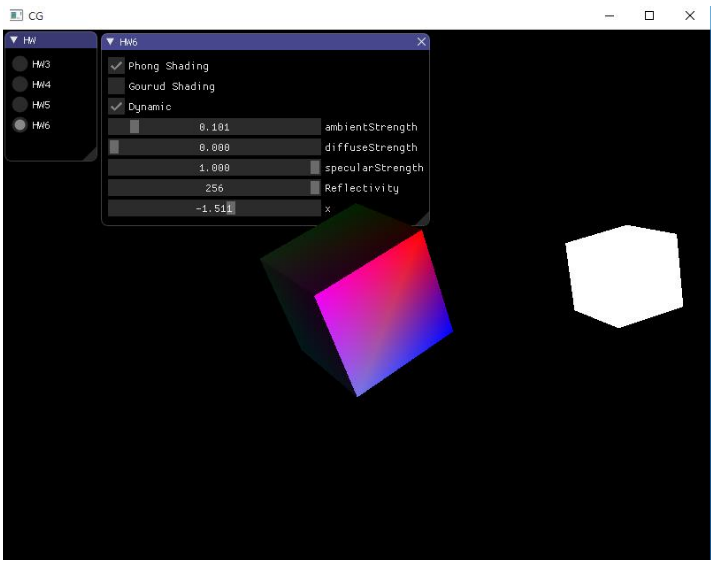
实验思路：

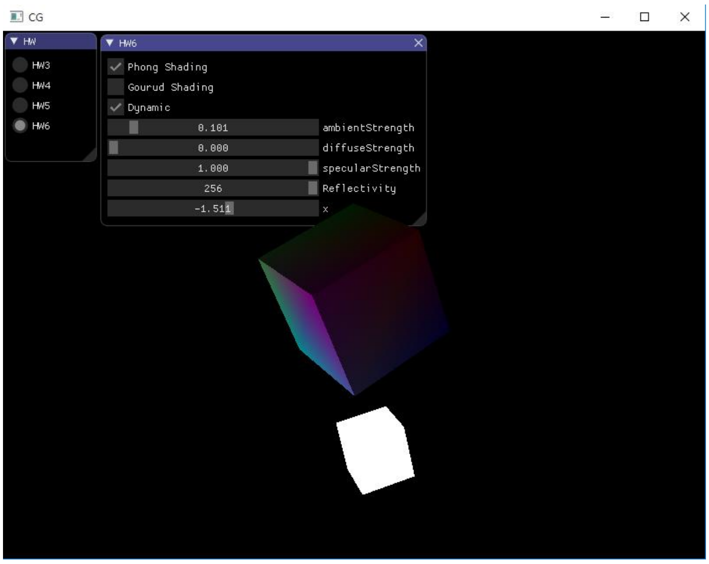
为光源添加移动，使光源绕物体旋转

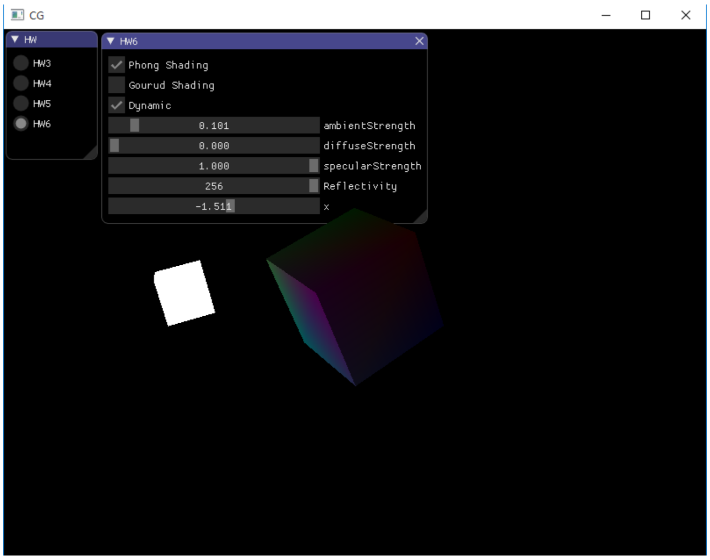
```
if (dynamic)
    glUniform3f(glGetUniformLocation(shaderProgramPhone, "lightPos"), sin(glfwGetTime()) * 3, cos(glfwGetTime()) * 3, sin(glfwGetTime()) * 3);
else
    glUniform3f(glGetUniformLocation(shaderProgramGouraud, "lightPos"), x, 3.0f, x);
if (dynamic)
    model = glm::translate(model, glm::vec3(sin(glfwGetTime())*3, cos(glfwGetTime()) * 3, sin(glfwGetTime()) * 3));
else
    model = glm::translate(model, glm::vec3(x, 3.0, x));
```

实验结果：






CG
HW
HW3
HW4
HW5
HW6
HW6
Phong Shading
Gourud Shading
Dynamic
0.101 ambientStrength
0.000 diffuseStrength
1.000 specularStrength
256 Reflectivity
-1.511 x


CG
HW
HW3
HW4
HW5
HW6
HW6
Phong Shading
Gourud Shading
Dynamic
0.101 ambientStrength
0.000 diffuseStrength
1.000 specularStrength
256 Reflectivity
-1.511 x