Homework 6 - Lights and Shading

黄树凯

16340085

作业要求

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

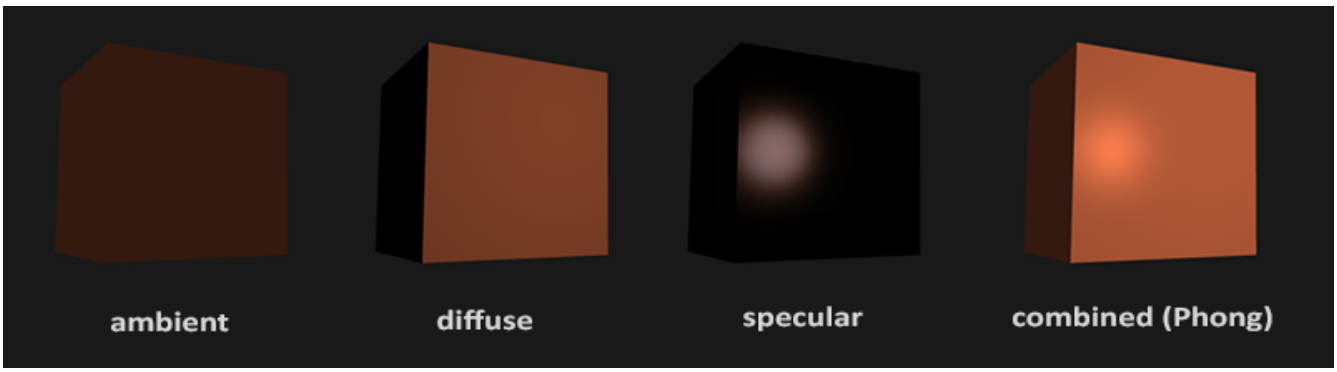
Bonus:

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。

实现及结果

1. Phong 光照模型

Phong光照模型主要有三个分量组成：环境(Ambient)、漫反射(Diffuse)、镜面(Specular)光照。



- 环境光照：使用一个环境光照常量永远给物体一些颜色

  使用一个很小的常量（光照）颜色，添加到物体片段的最终颜色中，这样子的话即便场景中没有直接的光源也能看起来存在有一些发散的光。用光的颜色乘以一个很小的常量环境因子，再乘以物体的颜色，然后将最终结果作为片段的颜色：

```
float ambientStrength = 0.1;
vec3 ambient = ambientStrength * lightColor;

vec3 result = ambient * objectColor;
FragColor = vec4(result, 1.0);
```

- 漫反射光照：模拟光源对物体的方向性影响，物体的某一部分越是正对着光源，它就会越亮

  计算漫反射需要知道物体表面的法向量和定向的光线（即光的位置向量和片段的位置向量）

```
// 光源的位置向量
uniform vec3 lightPos;

lightingShader.setVec3("lightPos", lightPos);

// 片段的位置向量
out vec3 FragPos;
out vec3 Normal;

gl_Position = projection * view * model * vec4(aPos, 1.0);
FragPos = vec3(model * vec4(aPos, 1.0));
Normal = mat3(transpose(inverse(model))) * aNormal; // 等比缩放

// 计算漫反射光照因子
in vec3 FragPos;

// 定义调整漫反射光照因子的参数 diffuseStrength
vec3 norm = normalize(Normal);
vec3 lightDir = normalize(lightPos - FragPos);
float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = diffuseStrength * diff * lightColor;
```

- 镜面光照：模拟有光泽物体上面出现的亮点，镜面光照的颜色相对于物体的颜色会更倾向于光的颜色。

  镜面光照也是依据光的方向向量和物体的法向量来决定的，但是它也依赖于观察方向，镜面光照是基于光的反射特性。通过反射法向量周围光的方向来计算反射向量。然后计算反射向量和视线方向的角度差，如果夹角越小，那么镜面光的影响就会越大。它的作用效果就是，当去看光被物体所反射的那个方向的时候，会看到一个高光。

  观察向量是镜面光照附加的一个变量，可以使用观察者世界空间位置和片段的位置来计算它。之后，计算镜面光强度，用它乘以光源的颜色，再将它加上环境光和漫反射分量。

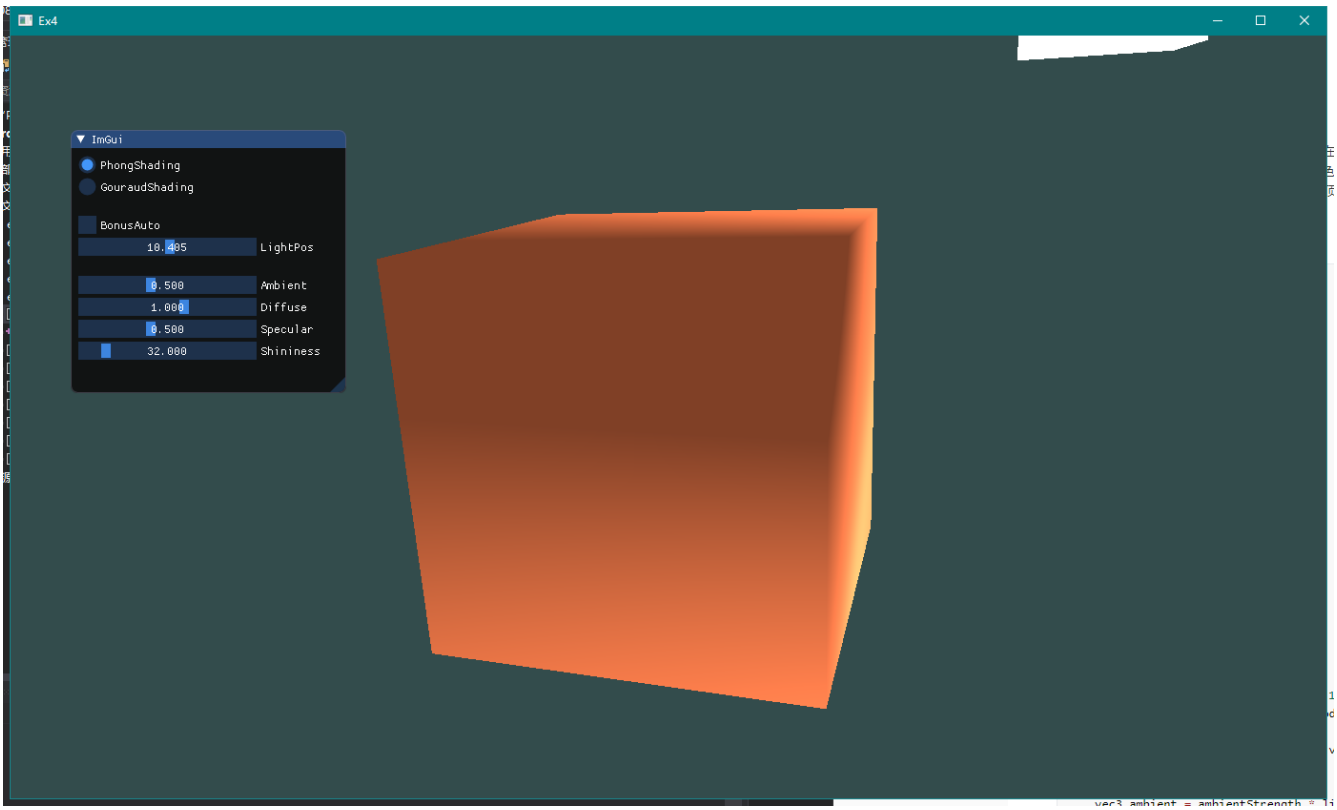
  为了得到观察者的世界空间坐标，可以使用摄像机对象的位置坐标代替。把另一个uniform添加到片段着色器，把相应的摄像机位置坐标传给片段着色器：

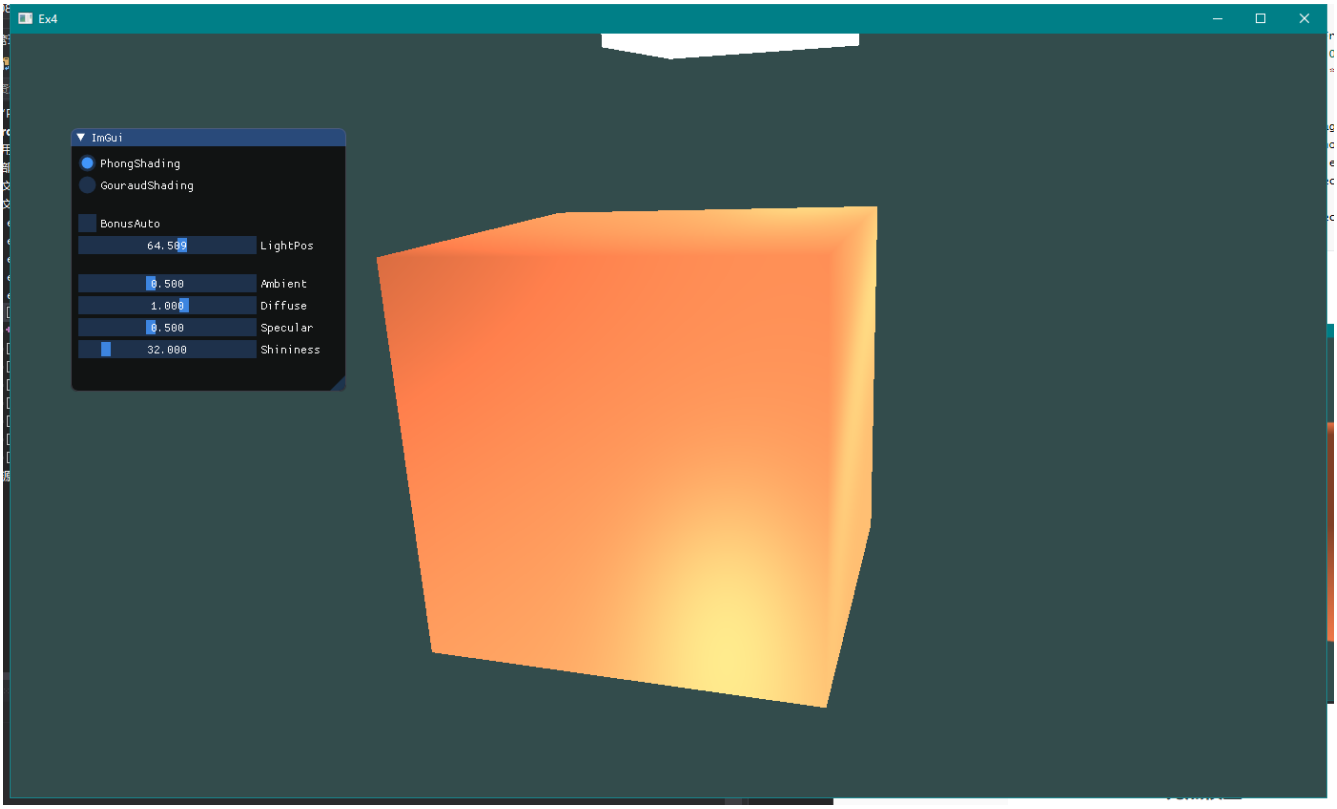
```
uniform vec3 viewPos;
lightingShader.setVec3("viewPos", camera.Position);

// 镜面强度变量
float specularStrength = 0.5;
// 计算镜面分量
vec3 viewDir = normalize(viewPos - FragPos);
vec3 reflectDir = reflect(-lightDir, norm);
float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
vec3 specular = specularStrength * spec * lightColor;
```

- 将三个光照分量整合得到物体的颜色

```
// 计算三个光照分量最后得到物体的颜色
// 三个 Strength 变量为可变参数
// ambient
vec3 ambient = ambientStrength * lightColor;

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

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

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

实验效果





2. Gouraud 光照模型

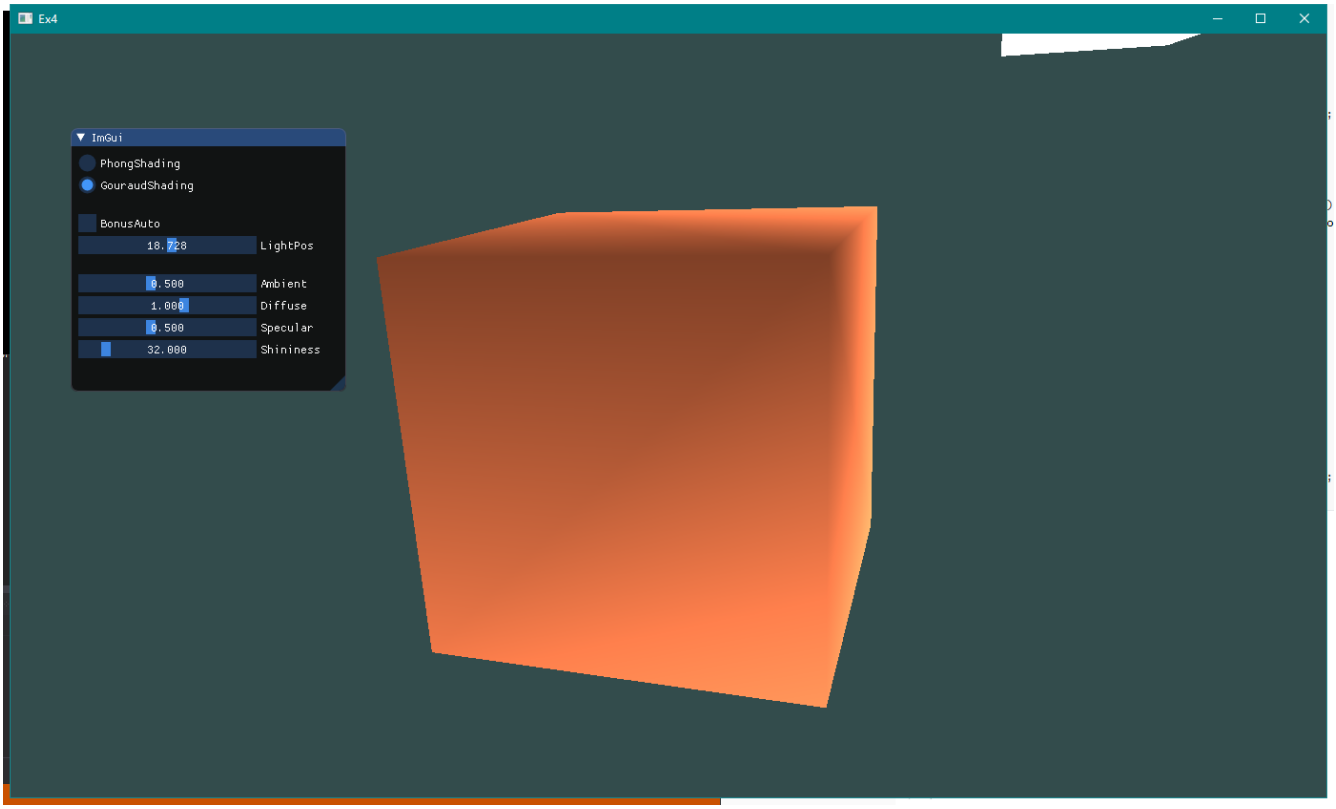
Gouraud 光照模型和 Phong 光照模型的区别在于： Gouraud光照模型是在顶点着色器中实现的光照模型，称之为 Gouraud 着色，而 Phong着色是在片段着色器中实现。 相比较之下 Gouraud 光照模型做光照会更高效，开销更低，但顶点着色器中的最终颜色值仅仅是该顶点的颜色值，片段的颜色值是由插值光照颜色得来的，因此光照效果会显得不够真实。

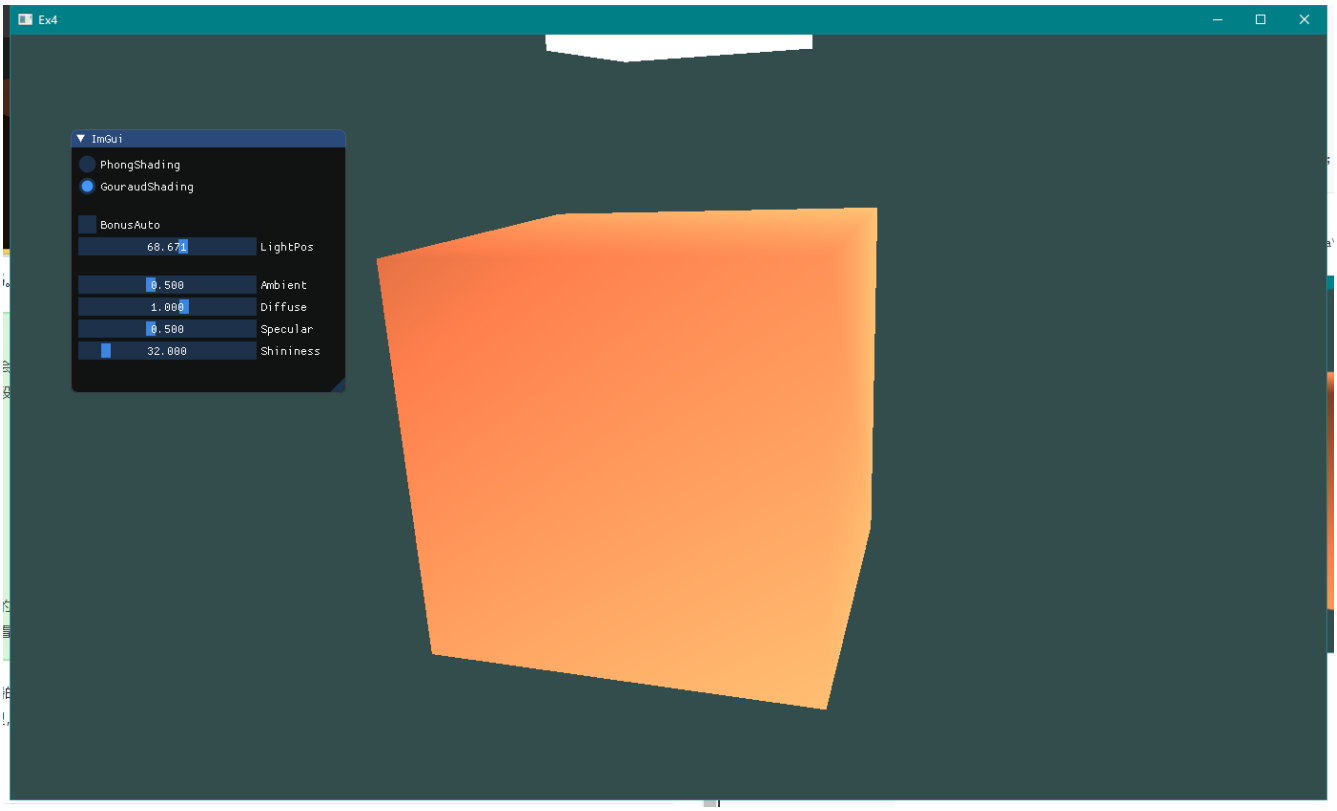
代码结构变化：

```
// light.vs
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;

out vec3 LightingColor;
out vec3 FragPos;
out vec3 Normal;

uniform vec3 lightColor;
uniform vec3 lightPos;
uniform vec3 viewPos;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
```

```
uniform float ambientStrength;
uniform float diffuseStrength;
uniform float specularStrength;
uniform float shininess;

void main() {
    FragPos = vec3(model * vec4(aPos, 1.0));
    Normal = mat3(transpose(inverse(model))) * aNormal;

    gl_Position = projection * view * vec4(FragPos, 1.0);

    // ambient
    vec3 ambient = ambientStrength * lightColor;

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

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

    LightingColor = ambient + diffuse + specular;
}

// light.fs
#version 330 core
out vec4 FragColor;

in vec3 LightingColor;

uniform vec3 objectColor;

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

实验效果：





整体光照效果不如 Phong Shading 平滑，反光点的亮度也不明显。

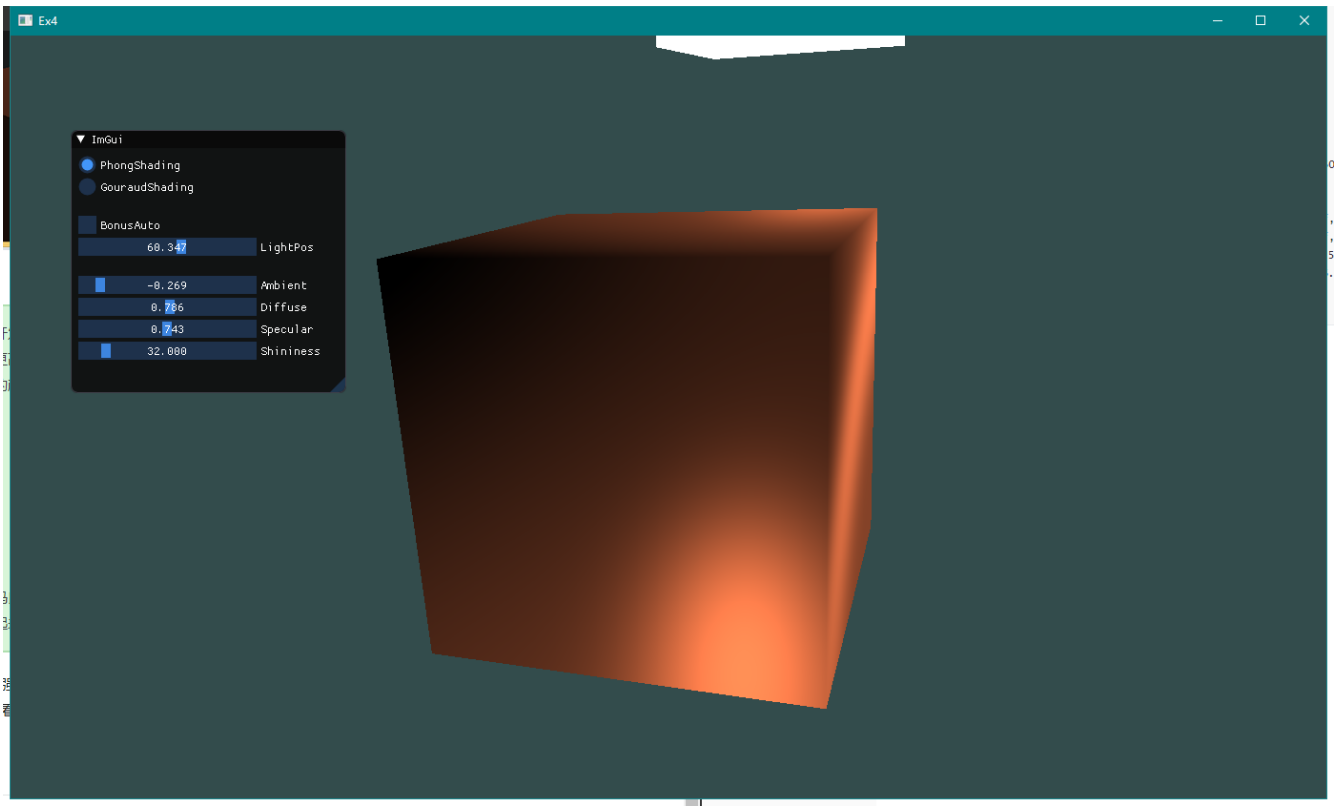
3. GUI 修改参数实时改变光照效果

```
// ImGui 菜单
{
```

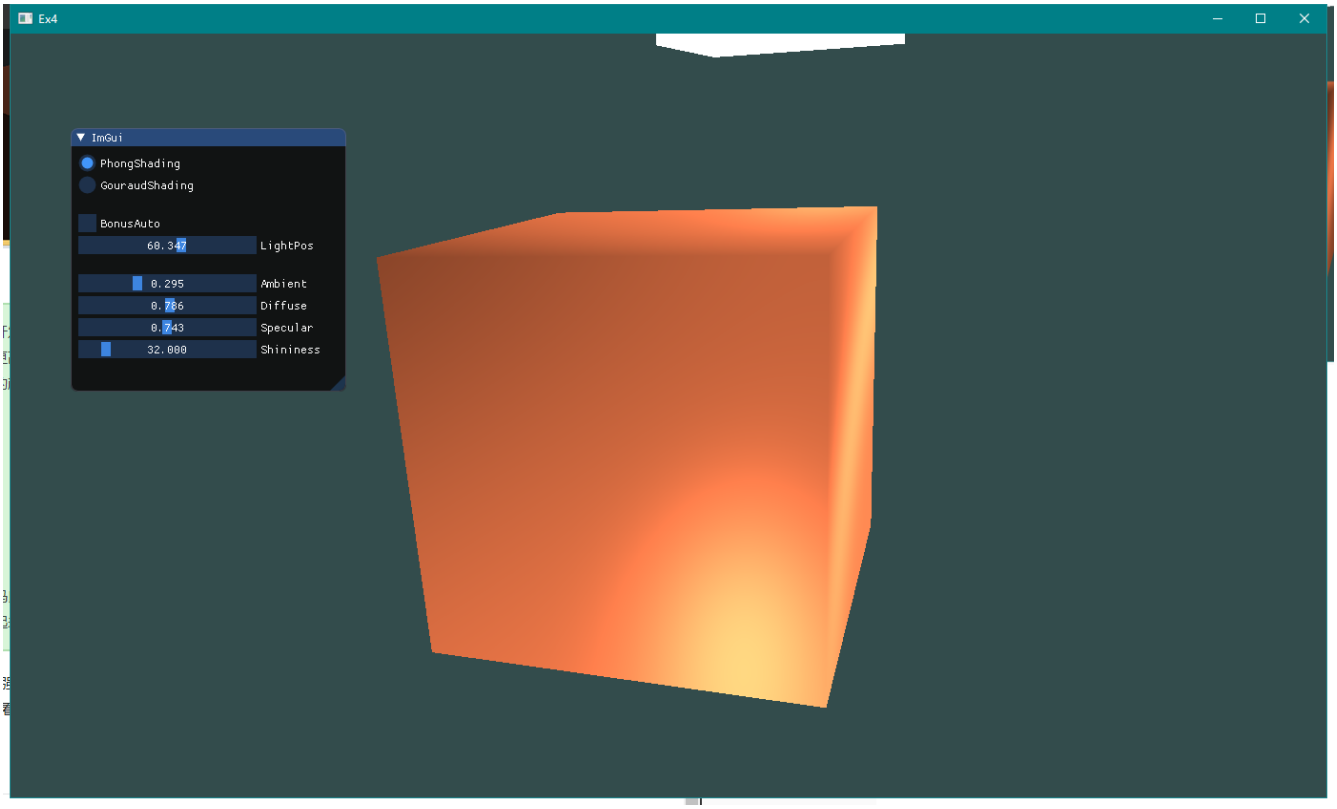
```
    ImGui::Begin("ImGui");
    ImGui::StyleColorsDark();
    // 功能选择
    ImGui::RadioButton("PhongShading", &mode, 0);
    ImGui::RadioButton("GouraudShading", &mode, 1);
    ImGui::NewLine();
    ImGui::Checkbox("BonusAuto", &bonus);
    ImGui::SliderFloat("LightPos", &bonusPos, -360.0f, 360.0f);
    ImGui::NewLine();
    // 参数
    ImGui::SliderFloat("Ambient", &ambientStrength, -0.5f, 2.0f);
    ImGui::SliderFloat("Diffuse", &diffuseStrength, -0.5f, 2.0f);
    ImGui::SliderFloat("Specular", &specularStrength, -0.5f, 2.0f);
    ImGui::SliderFloat("Shininess", &shininess, 0.0f, 256.0f);
    ImGui::End();
}
```

修改环境光照参数

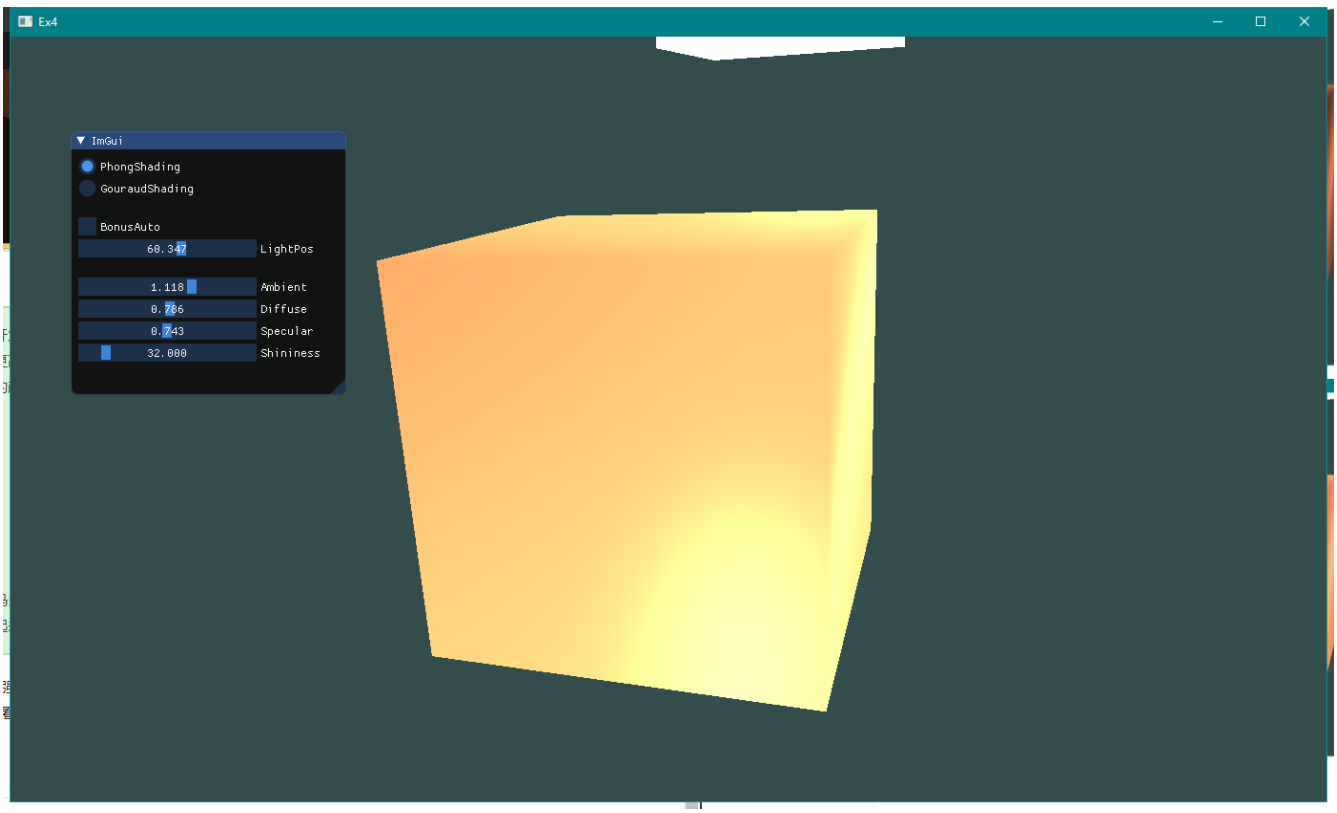
调低环境光照参数



适中的环境光照参数



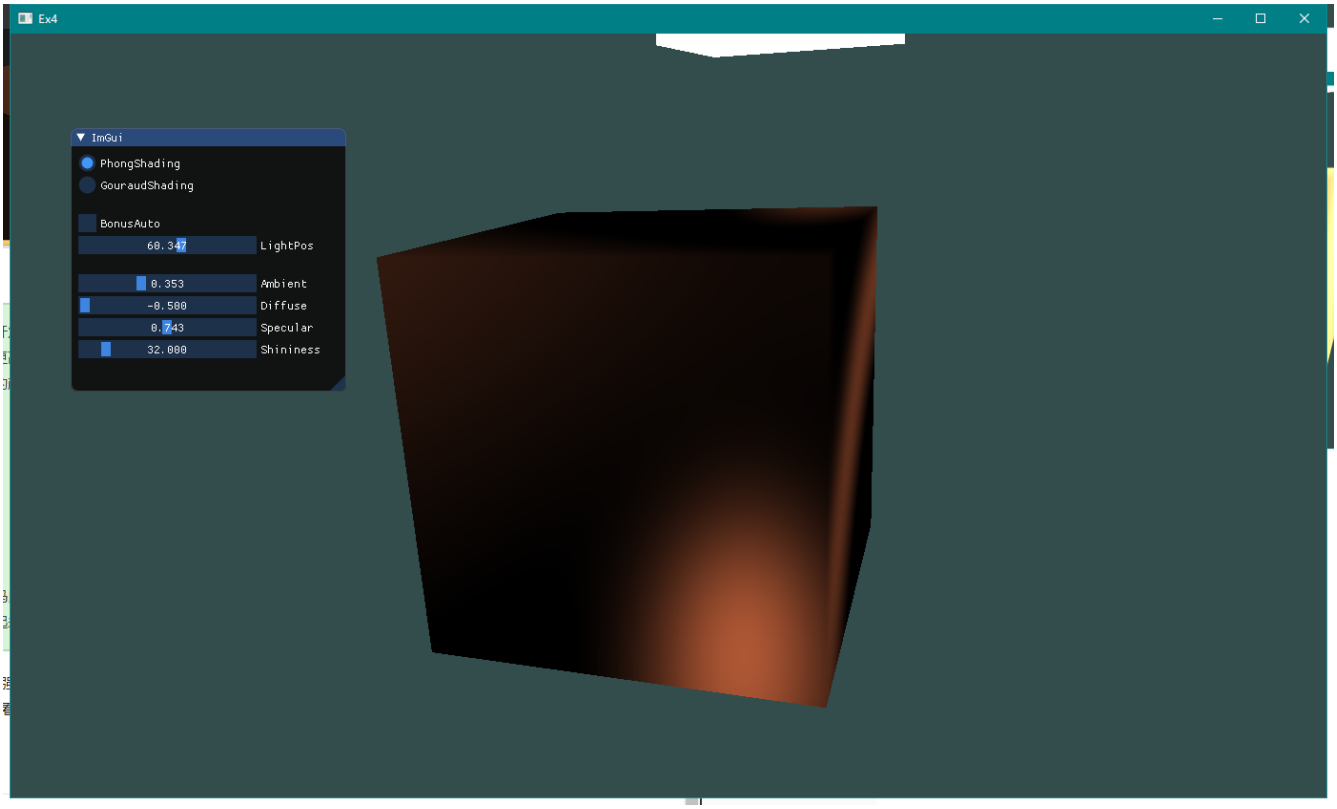
调高环境光照参数



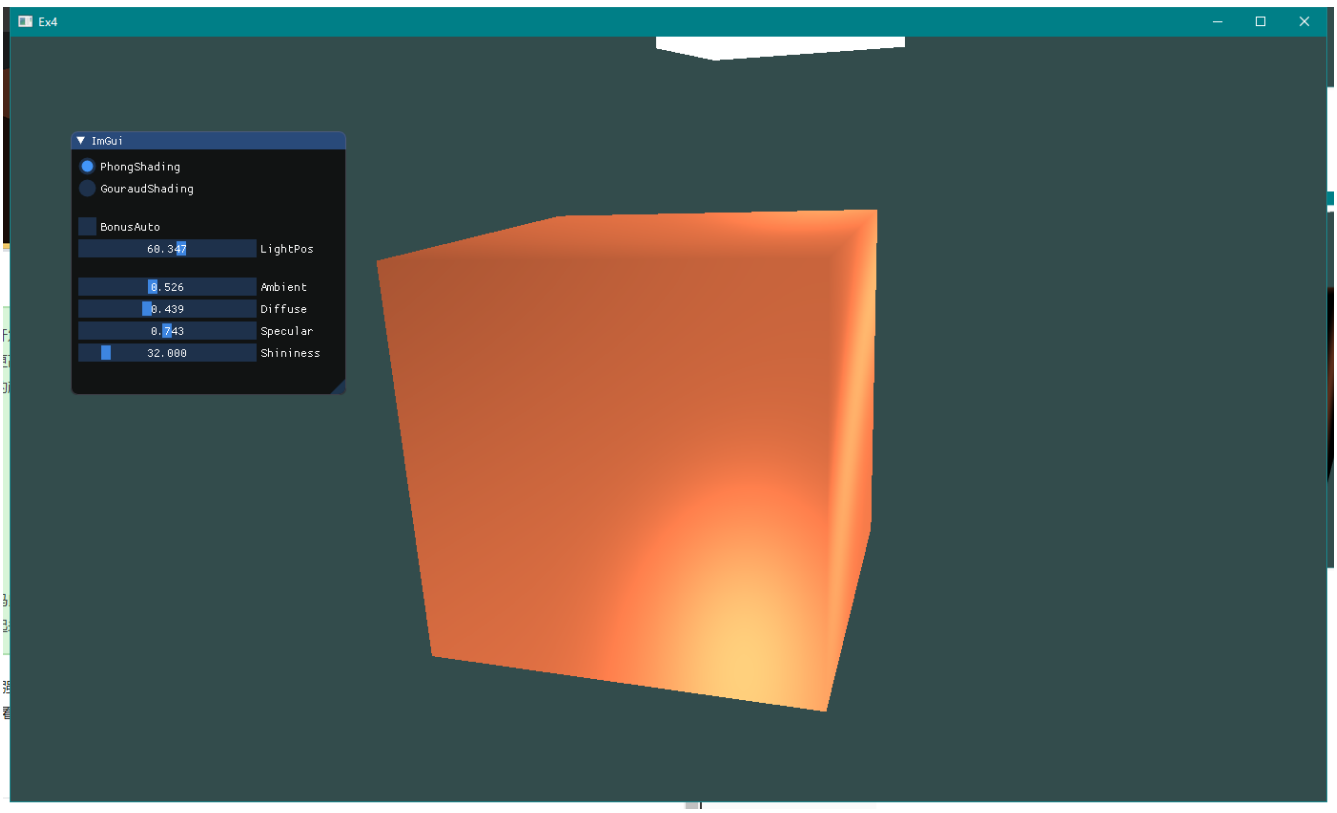
修改环境光照参数会使得物体的整个颜色色度发生变化。

修改漫反射光照参数

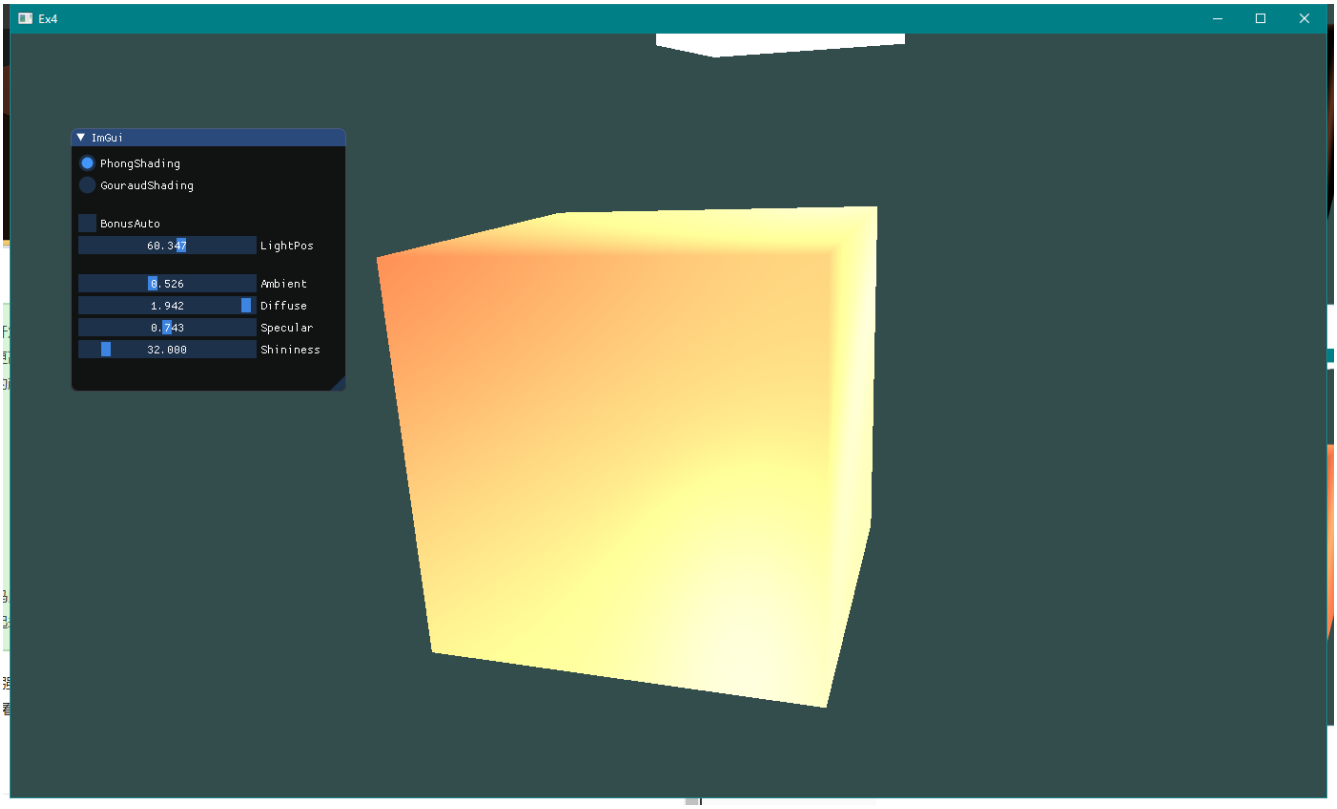
调低漫反射光照参数



适中的漫反射光照参数



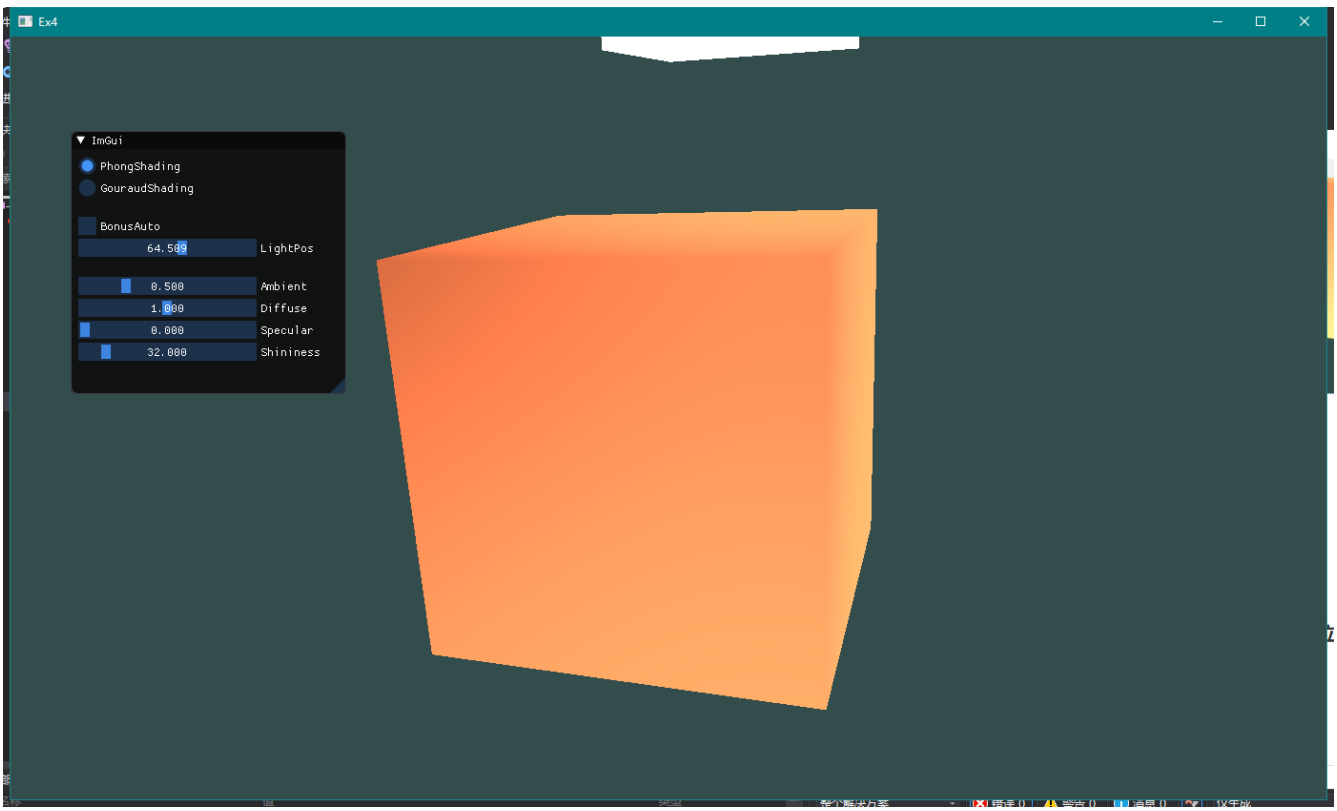
调高漫反射光照参数



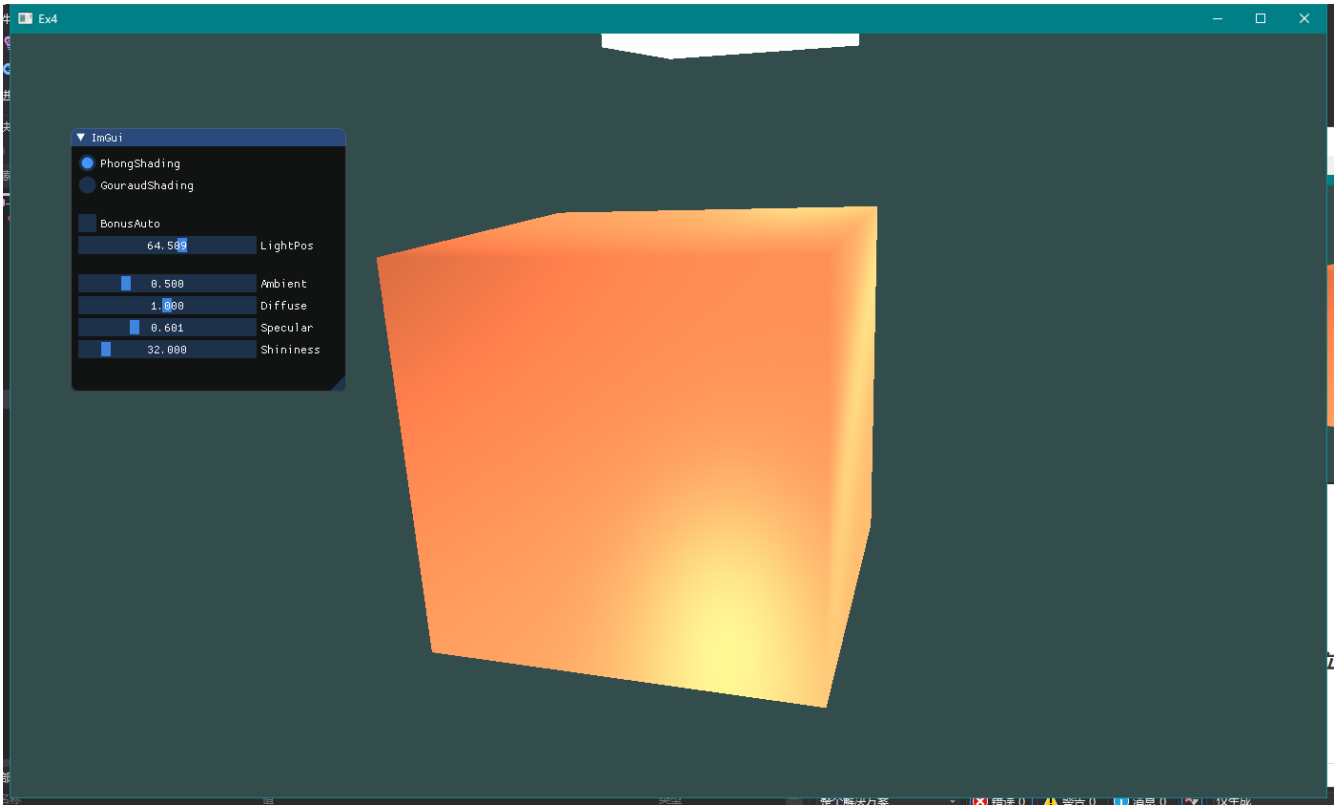
修改漫反射光照参数会使得光照在物体表面距离光线直射点不同位置的颜色变化程度改变

修改镜面反射光照参数

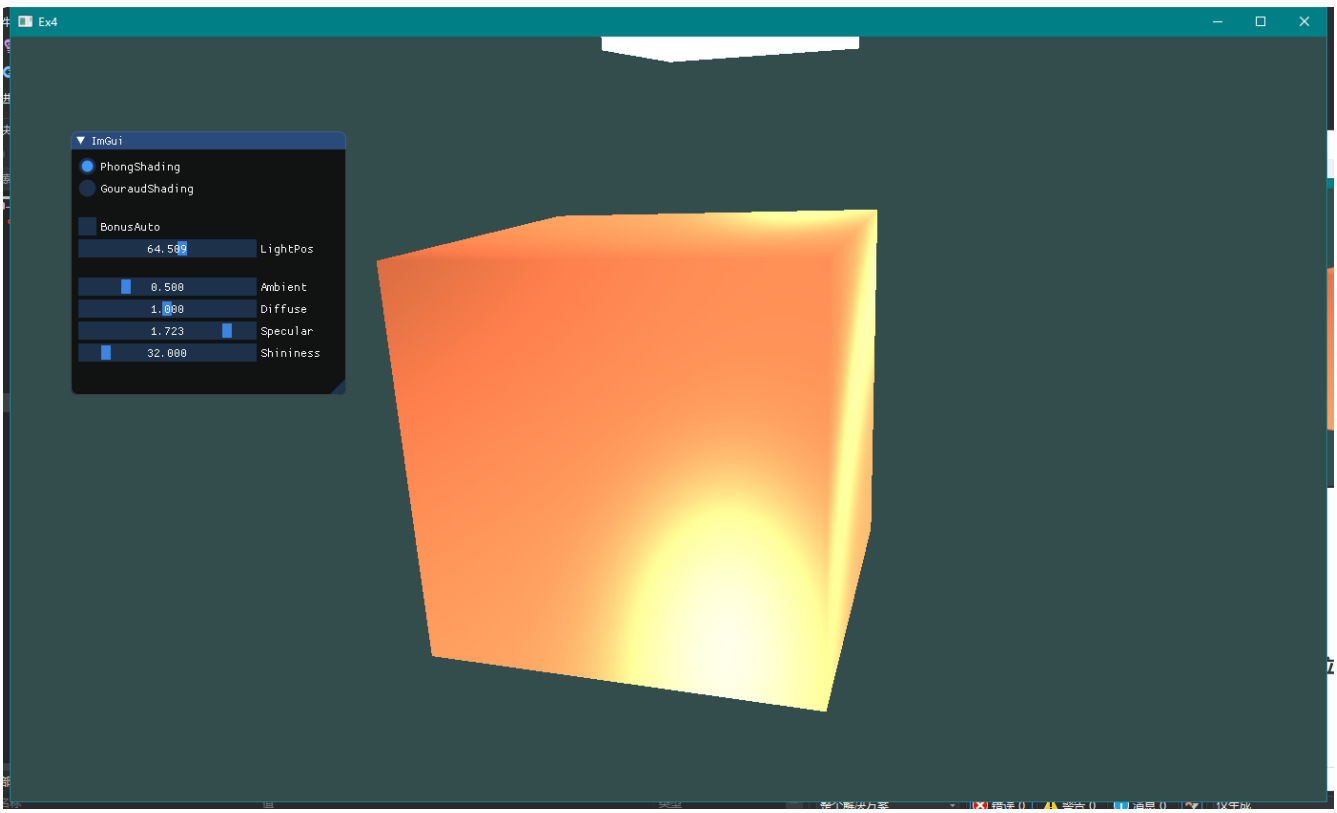
调低镜面反射光照参数



适中的镜面反射光照参数



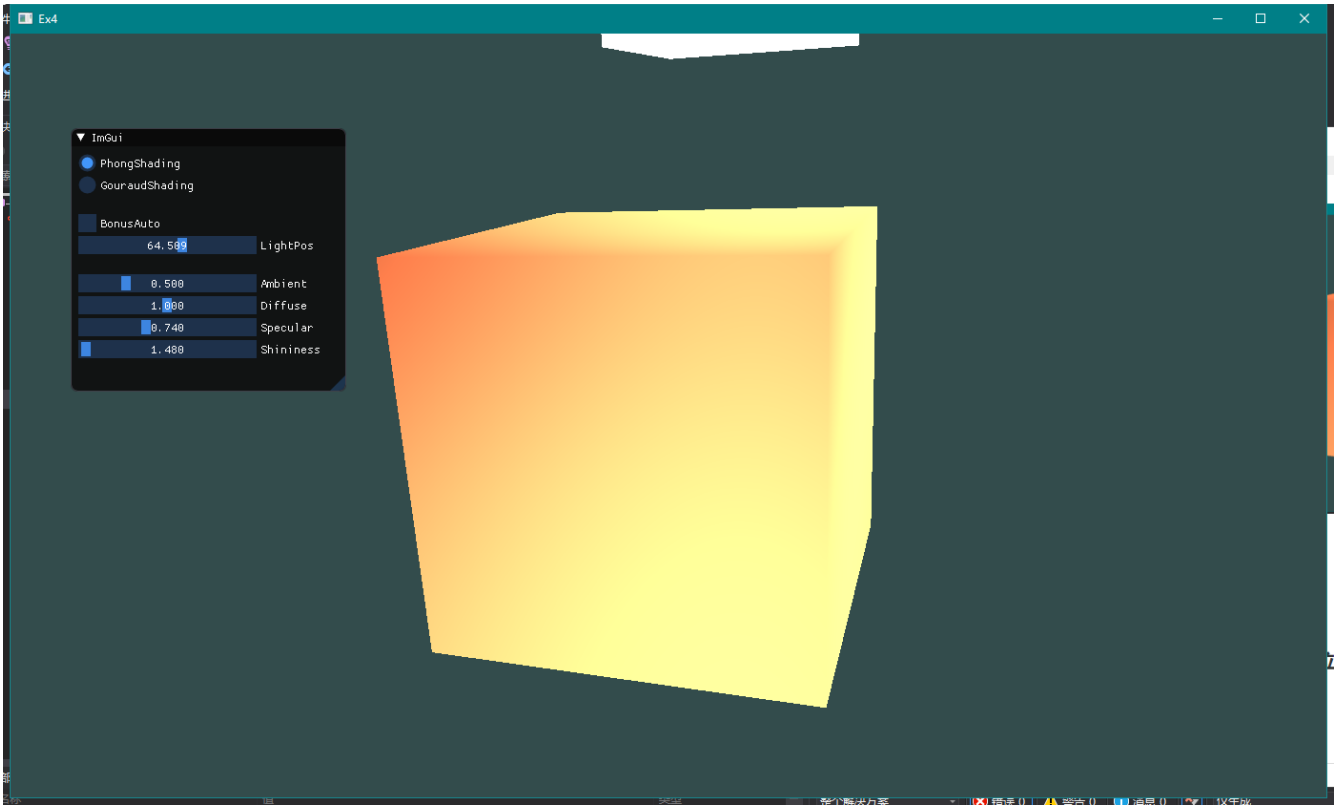
调高镜面反射光照参数



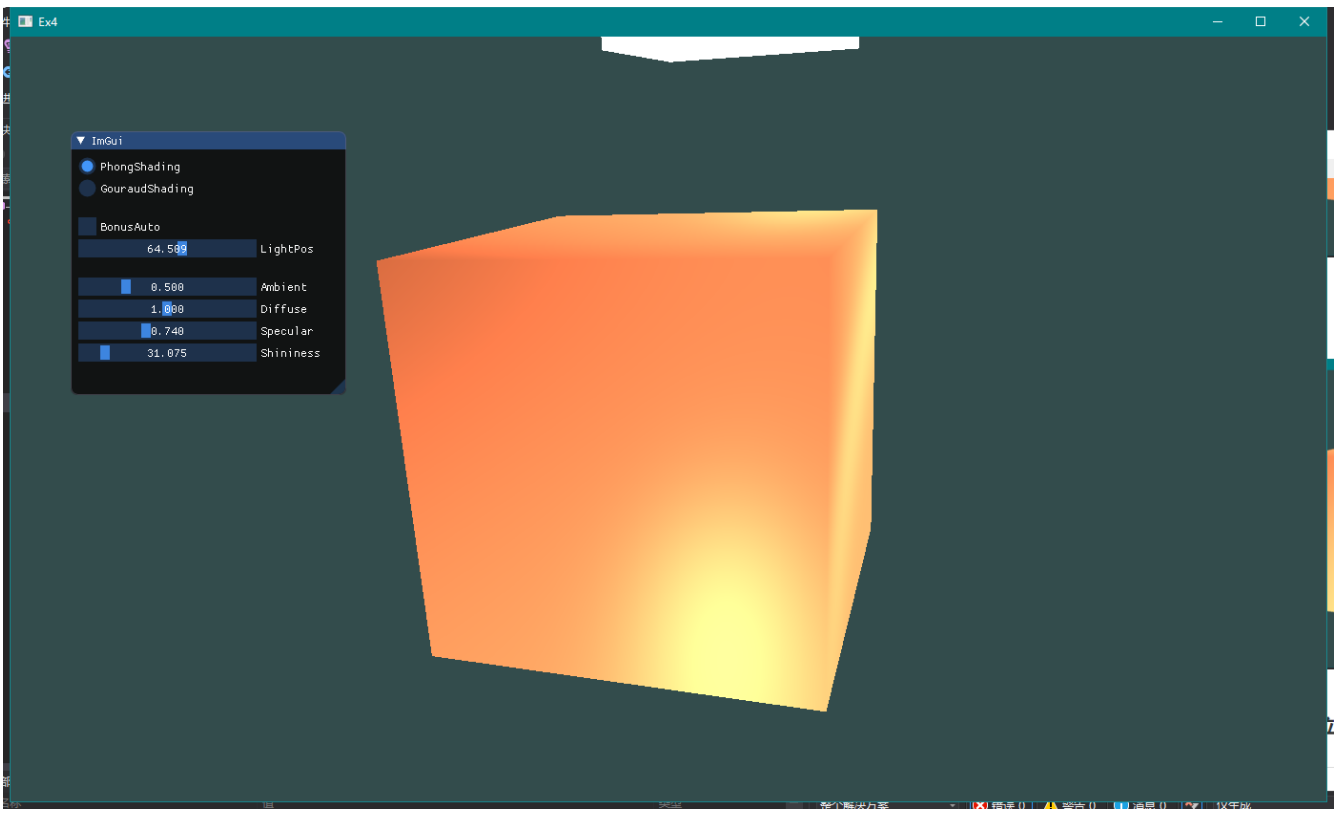
修改镜面反射光照参数会使光照点的亮度发生变化。

修改反光度

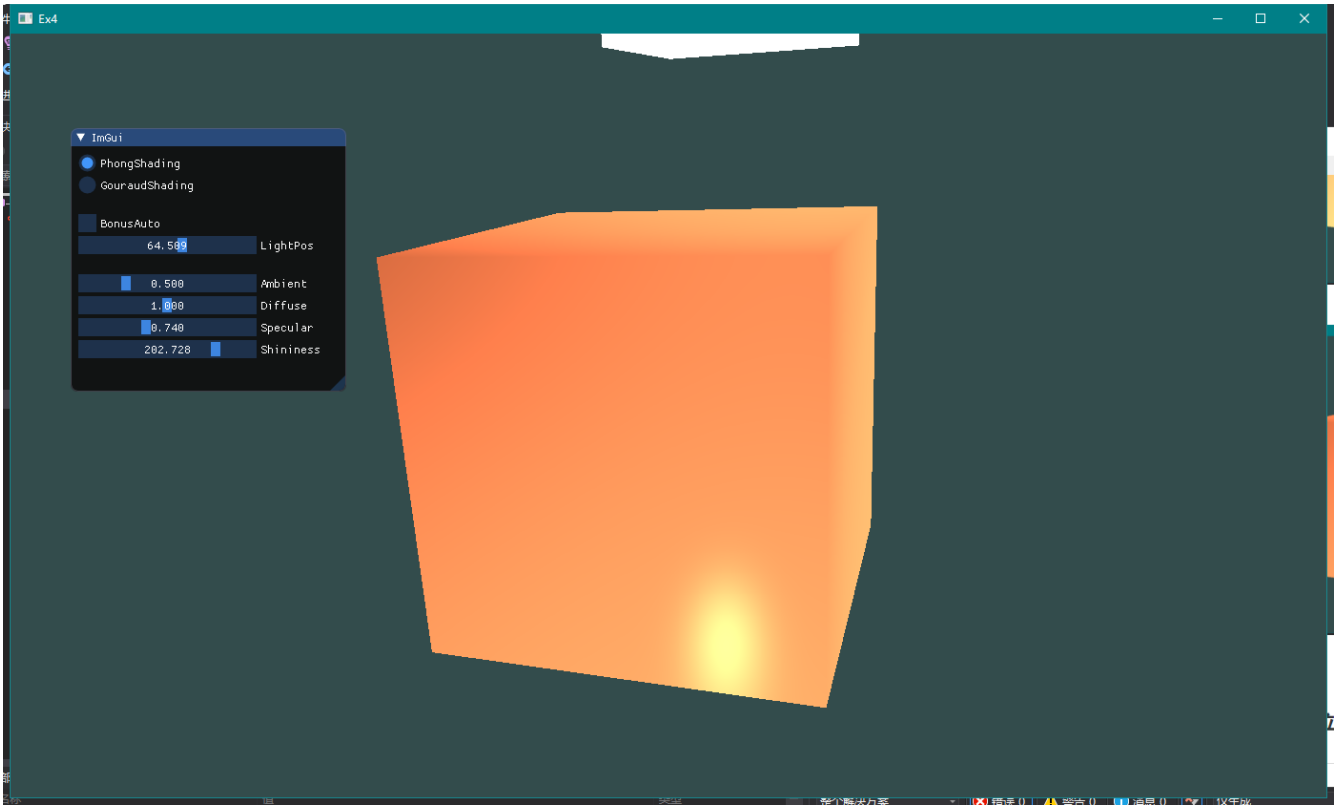
低反光度


Ex4
ImGui
PhongShading
GouraudShading
BonusAuto
64.500
LightPos
0.500
Ambient
1.000
Diffuse
0.740
Specular
1.480
Shininess

中反光度


Ex4
ImGui
PhongShading
GouraudShading
BonusAuto
64.500
LightPos
0.500
Ambient
1.000
Diffuse
0.740
Specular
31.075
Shininess

高反光度



修改反光度会影响物体光照点的范围大小。

4. Bonus 实时光源位置改变

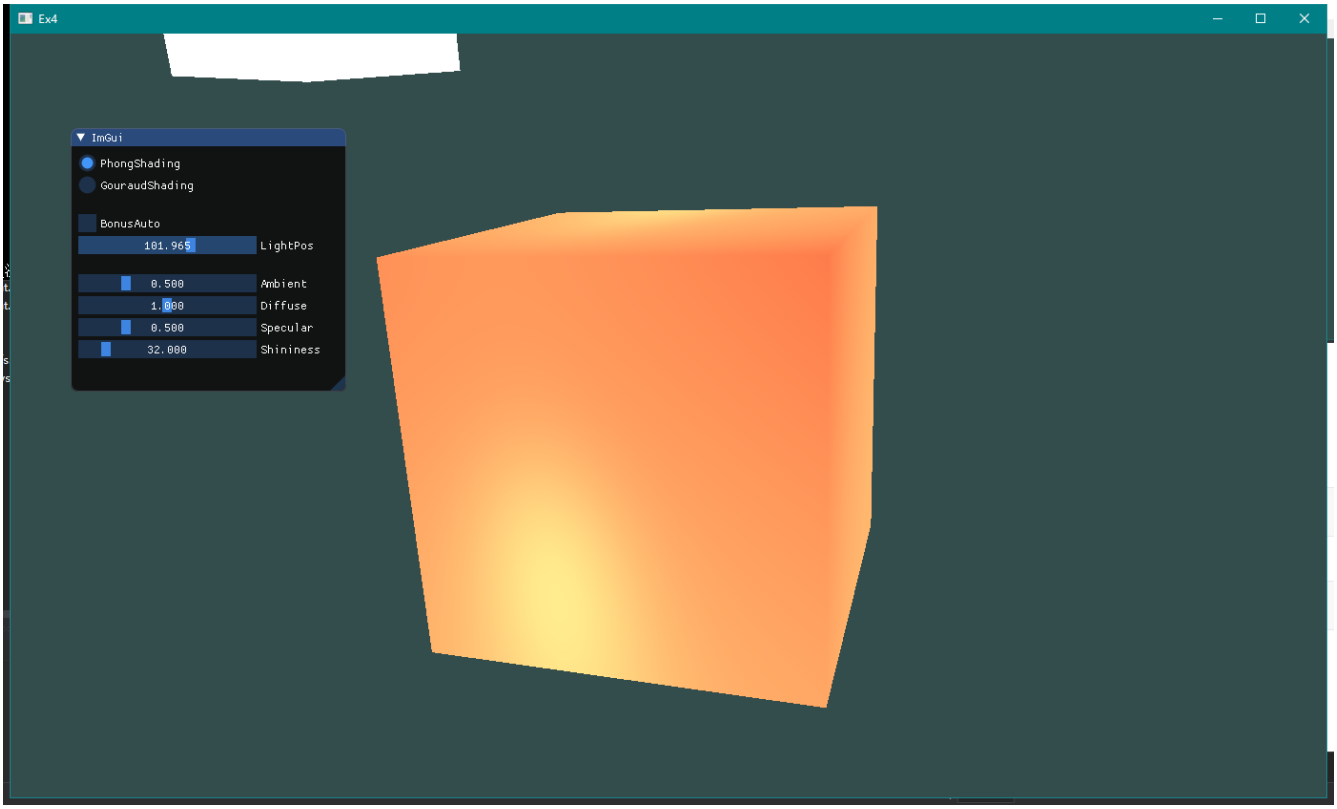
根据时间变化自动改变光源的Z，X轴方向上的坐标位置

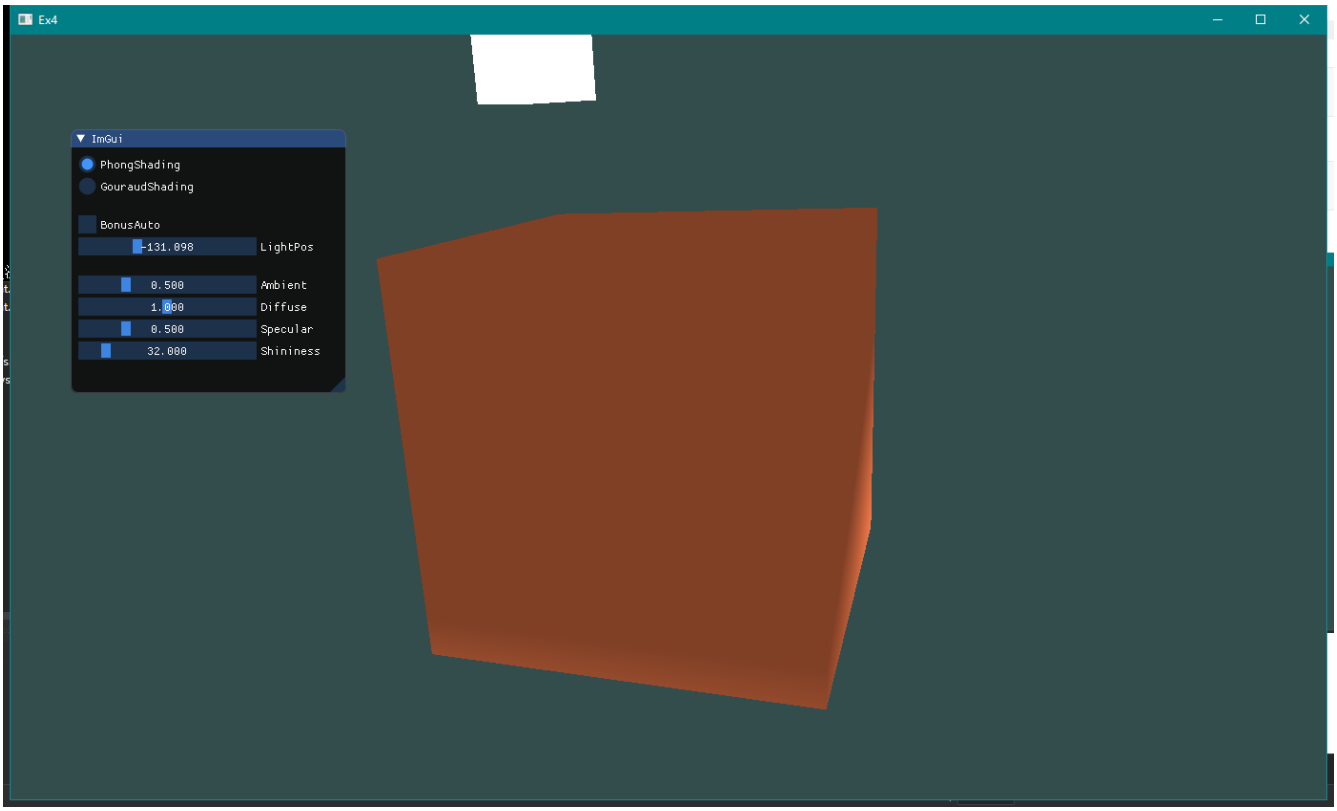
```
lightPos.x = cos(glfwGetTime()) * 1.0f;
lightPos.z = sin(glfwGetTime()) * 1.0f;
```

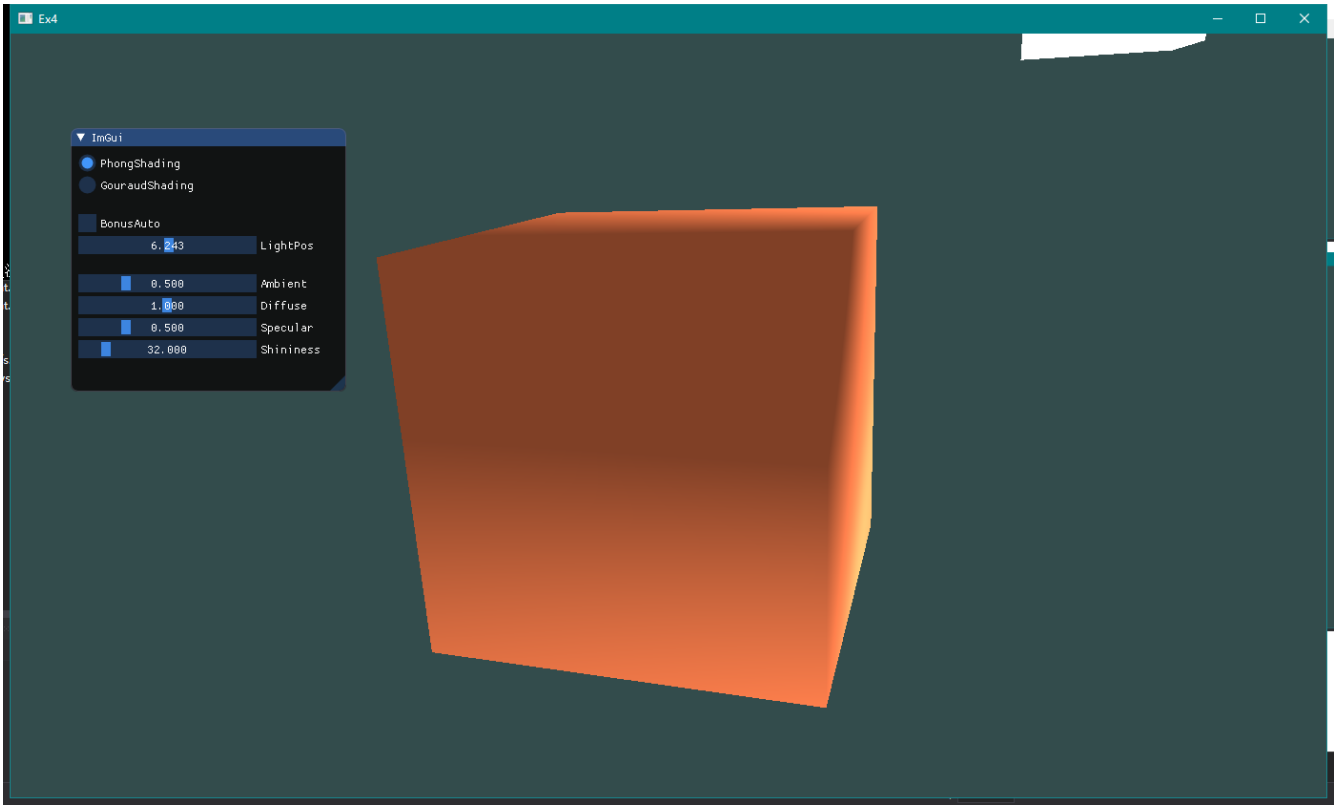
根据滑条参数修改光源的坐标位置

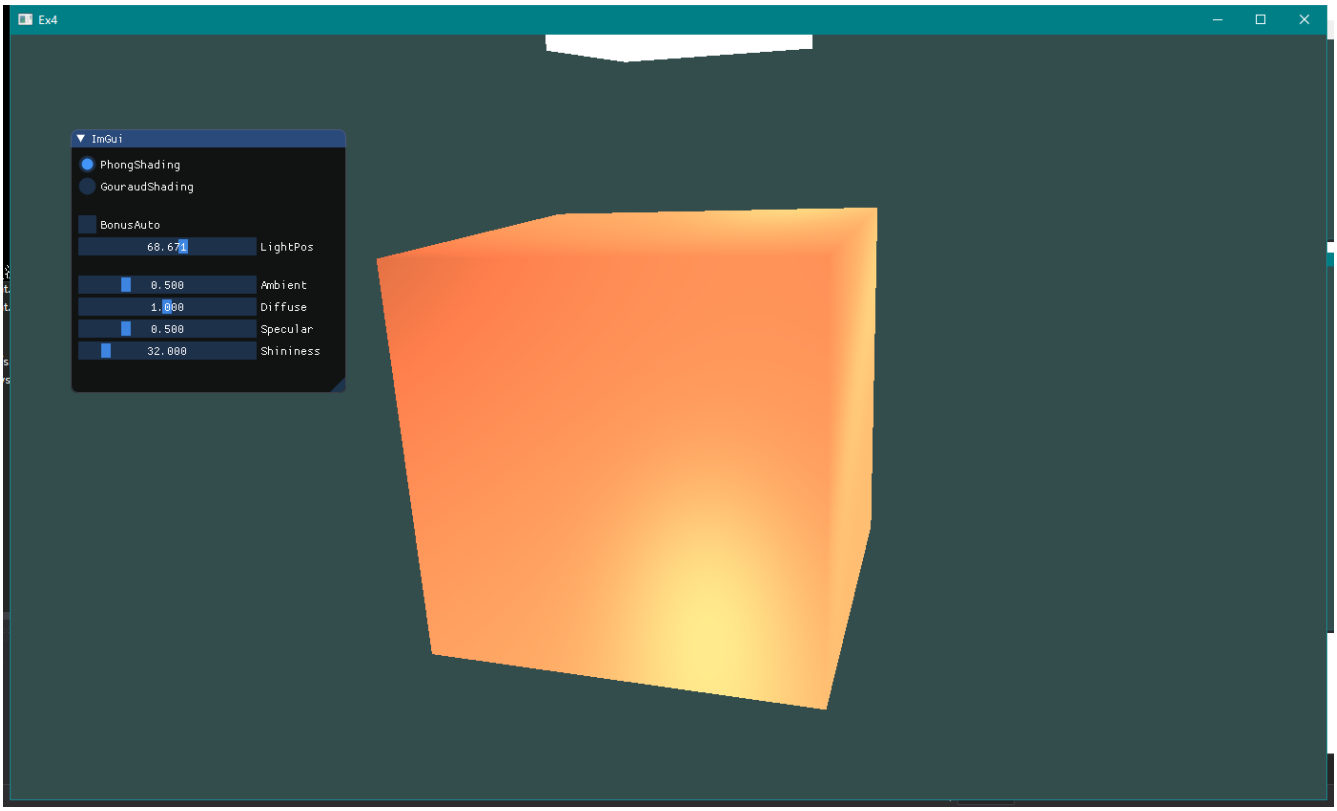
```
lightPos.x = cos(glm::radians(bonusPos)) * 1.0f;
lightPos.z = sin(glm::radians(bonusPos)) * 1.0f;
```

实验效果


Ex4
ImGui
PhongShading
GouraudShading
BonusAuto
101.965
LightPos
0.500
Ambient
1.000
Diffuse
0.500
Specular
32.000
Shininess


Ex4
ImGui
PhongShading
GouraudShading
BonusAuto
-131.098
LightPos
0.500
Ambient
1.000
Diffuse
0.500
Specular
32.000
Shininess


Ex4
ImGui
PhongShading
GouraudShading
BonusAuto
6.243
LightPos
0.500
Ambient
1.000
Diffuse
0.500
Specular
32.000
Shininess


Ex4
ImGui
PhongShading
GouraudShading
BonusAuto
68.671
LightPos
0.500
Ambient
1.000
Diffuse
0.500
Specular
32.000
Shininess