计算机图形学 Homework 6 - Lights and Shading

16340299 赵博然

场景中绘制cube.

顶点坐标数组.

float vertices[] = {

-2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 0.0f,

2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 0.0f,

2.0f, 2.0f, -2.0f, 1.0f, 0.0f, 0.0f,

2.0f, 2.0f, -2.0f, 1.0f, 0.0f, 0.0f,

-2.0f, 2.0f, -2.0f, 1.0f, 0.0f, 0.0f,

-2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 0.0f,

-2.0f, -2.0f, 2.0f, 0.0f, 1.0f, 0.0f,

2.0f, -2.0f, 2.0f, 0.0f, 1.0f, 0.0f,

2.0f, 2.0f, 2.0f, 0.0f, 1.0f, 0.0f,

2.0f, 2.0f, 2.0f, 0.0f, 1.0f, 0.0f,

-2.0f, 2.0f, 2.0f, 0.0f, 1.0f, 0.0f,

-2.0f, -2.0f, 2.0f, 0.0f, 1.0f, 0.0f,

-2.0f, 2.0f, 2.0f, 0.0f, 0.0f, 1.0f,

-2.0f, 2.0f, -2.0f, 0.0f, 0.0f, 1.0f,

-2.0f, -2.0f, -2.0f, 0.0f, 0.0f, 1.0f,

-2.0f, -2.0f, -2.0f, 0.0f, 0.0f, 1.0f,

-2.0f, -2.0f, 2.0f, 0.0f, 0.0f, 1.0f,

-2.0f, 2.0f, 2.0f, 0.0f, 0.0f, 1.0f,

2.0f, 2.0f, 2.0f, 0.0f, 1.0f, 1.0f,

2.0f, 2.0f, -2.0f, 0.0f, 1.0f, 1.0f,

2.0f, -2.0f, -2.0f, 0.0f, 1.0f, 1.0f,

2.0f, -2.0f, -2.0f, 0.0f, 1.0f, 1.0f,

2.0f, -2.0f, 2.0f, 0.0f, 1.0f, 1.0f,

2.0f, 2.0f, 2.0f, 0.0f, 1.0f, 1.0f,

-2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 1.0f,

2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 1.0f,

2.0f, -2.0f, 2.0f, 1.0f, 0.0f, 1.0f,

2.0f, -2.0f, 2.0f, 1.0f, 0.0f, 1.0f,

-2.0f, -2.0f, 2.0f, 1.0f, 0.0f, 1.0f,

-2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 1.0f,

-2.0f, 2.0f, -2.0f, 1.0f, 1.0f, 0.0f,

2.0f, 2.0f, -2.0f, 1.0f, 1.0f, 0.0f,

2.0f, 2.0f, 2.0f, 1.0f, 1.0f, 0.0f,

2.0f, 2.0f, 2.0f, 1.0f, 1.0f, 0.0f,

-2.0f, 2.0f, 2.0f, 1.0f, 1.0f, 0.0f,

-2.0f, 2.0f, -2.0f, 1.0f, 1.0f, 0.0f,

};

使用Homework 3的方法绘制cube.

Phong光照模型.

顶点着色器phong.vs. 在顶点着色器计算Frag的位置和FragPos和法向量Normal.

#version 330 core

layout(location = 0) in vec3 aPos;

layout(location = 1) in vec3 aNormal;

uniform mat4 model;

uniform mat4 view;

uniform mat4 projection;

out vec3 Normal;

out vec3 FragPos;

out vec3 result;

void main() {

FragPos = vec3(model \* vec4(aPos, 1.0));

gl\_Position = projection \* view \* model \* vec4(aPos, 1.0);

Normal = mat3(transpose(inverse(model))) \* aNormal;

}

片段着色器phong.fs. 在片段着色器计算ambient, diffuse, specular.

#version 330 core

in vec3 Normal;

in vec3 FragPos;

out vec4 FragColor;

uniform vec3 objectColor;

uniform vec3 lightColor;

uniform vec3 lightPos;

uniform vec3 viewPos;

uniform float ambientStrength;

uniform float specularStrength;

uniform float shininess;

uniform float diffuseMultiple;

void main() {

vec3 ambient = ambientStrength \* lightColor;

vec3 norm = normalize(Normal);

vec3 lightDirection = normalize(lightPos - FragPos);

vec3 viewDirection = normalize(viewPos - FragPos);

vec3 reflectDirection = reflect(-lightDirection, norm);

float diff = max(dot(norm, lightDirection), 0.0);

vec3 diffuse = diff \* lightColor;

float spec = pow(max(dot(viewDirection, reflectDirection), 0.0), shininess);

vec3 specular = specularStrength \* spec \* lightColor;

FragColor = vec4((ambient + diffuseMultiple \* diffuse + specular) \* objectColor, 1.0);

}

Gouraud光照模型. Gouraud在顶点着色器中计算光照模型, 然后在片段着色器中插值计算颜色. 光照模型的计算与Phong基本相同.

顶点着色器gouraud.vs.

#version 330 core

layout(location = 0) in vec3 aPos;

layout(location = 1) in vec3 aNormal;

uniform mat4 model;

uniform mat4 view;

uniform mat4 projection;

uniform vec3 lightColor;

uniform vec3 lightPos;

uniform vec3 viewPos;

uniform float ambientStrength;

uniform float specularStrength;

uniform float diffuseMultiple;

uniform float shininess;

out vec3 result;

void main() {

gl\_Position = projection \* view \* model \* vec4(aPos, 1.0);

vec3 FragPos = vec3(model \* vec4(aPos, 1.0));

vec3 Normal = mat3(transpose(inverse(model))) \* aNormal;

vec3 norm = normalize(Normal);

vec3 lightDirection = normalize(lightPos - FragPos);

vec3 viewDirection = normalize(viewPos - FragPos);

vec3 reflectDirection = reflect(-lightDirection, norm);

vec3 ambient = ambientStrength \* lightColor;

float diff = max(dot(norm, lightDirection), 0.0);

vec3 diffuse = diff \* lightColor;

float spec = pow(max(dot(viewDirection, reflectDirection), 0.0), shininess);

vec3 specular = specularStrength \* spec \* lightColor;

result = ambient + diffuseMultiple \* diffuse + specular;

}

片段着色器gouraud.fs.

#version 330 core

in vec3 result;

out vec4 FragColor;

uniform vec3 objectColor;

void main()

{

FragColor = vec4(result \* objectColor, 1.0);

}

使光源移动.

在ImGui中调节光源位置. 另外使光源可以随时间自动绕y轴旋转. 我还设置了调节摄像机位置方便观察.

ImGui::SliderFloat("light x", &lightPosition.x, -32.0f, 32.0f);

ImGui::SliderFloat("light y", &lightPosition.y, -32.0f, 32.0f);

ImGui::SliderFloat("light z", &lightPosition.z, -32.0f, 32.0f);.

ImGui::SliderFloat("view x", &viewPosition.x, -32.0f, 32.0f);

ImGui::SliderFloat("view y", &viewPosition.y, -32.0f, 32.0f);

ImGui::SliderFloat("view z", &viewPosition.z, -32.0f, 32.0f);

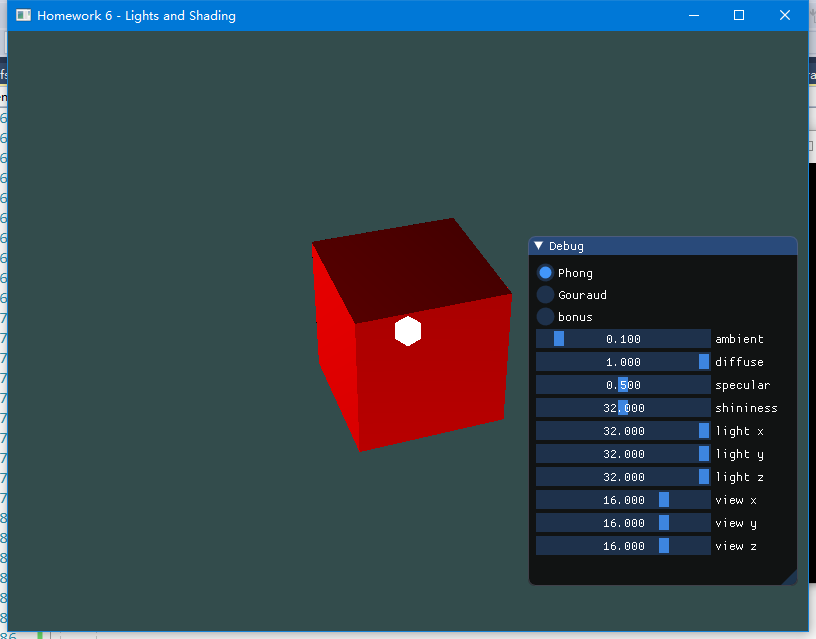
float radius = 32.0f;

float camX = sin(glfwGetTime()) \* radius;

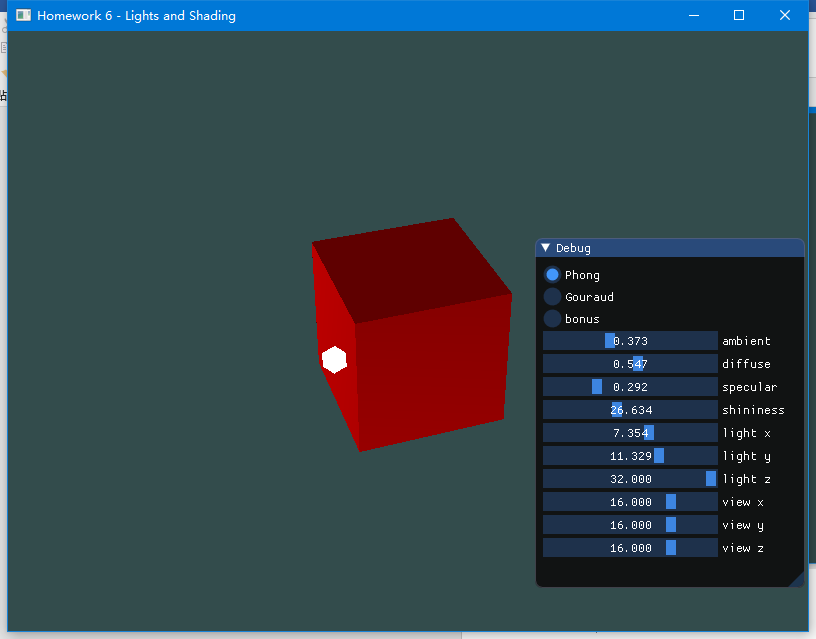
float camZ = cos(glfwGetTime()) \* radius;

截图.

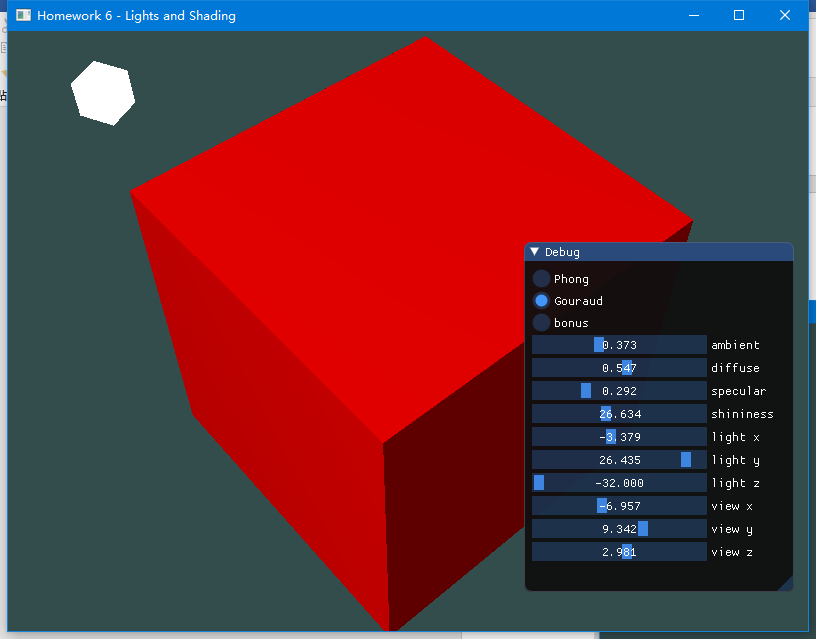
Phong光照模型.



移动光源, 改变参数.



Gouraud光照模型. 看上去与Phong没有太大区别.



bonus部分见video.mp4.

代码见src/main.cpp.

程序见doc/program/program.exe.

演示视频见doc/video.mp4.