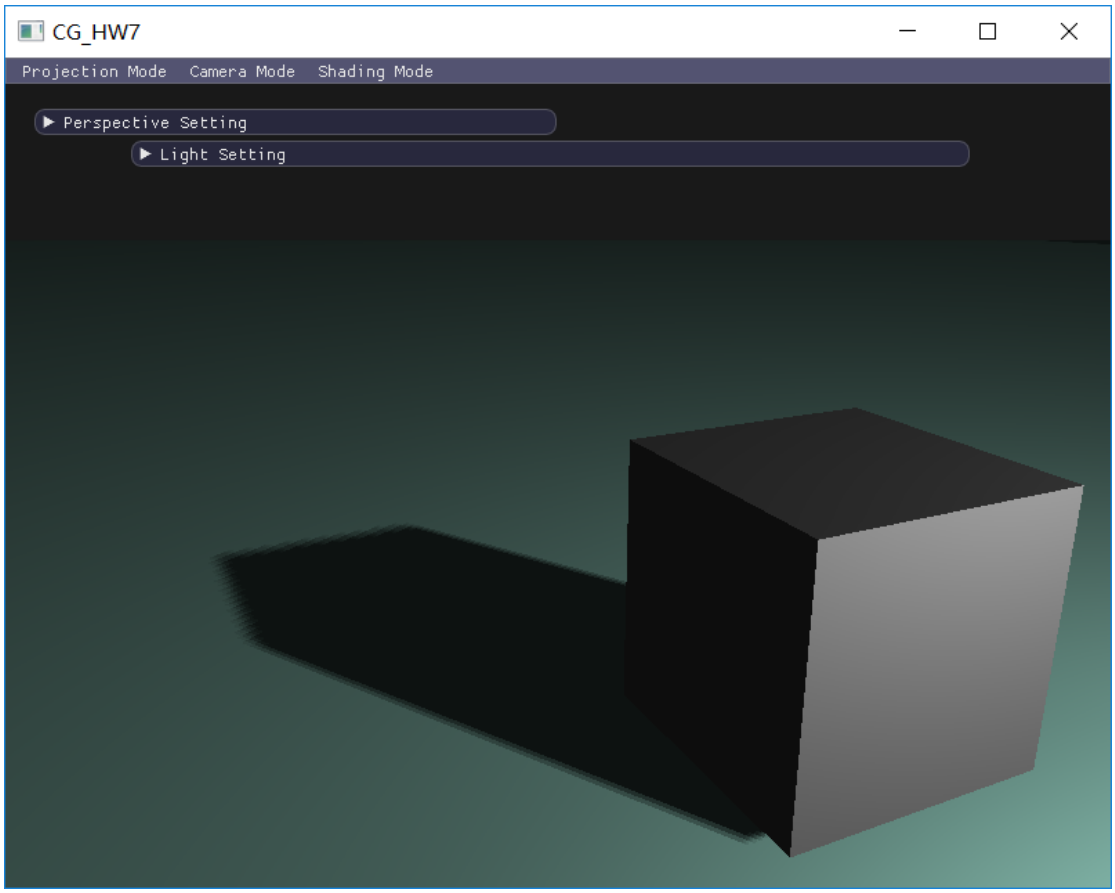
计算机图形学 HW7 报告

16340049 陈泽键

Basic:



shadow mapping 解释:

在现实世界中，如果光线无法透过物体，则光线只会点亮它最先遇到的物体表面，而在未照射到的部分则会形成阴影。shadow mapping 算法是一种描述该现象的一个简单的方法。这个算法的主要思路是以光的位置为视角进行渲染，能看见的部分，即光线最先遇到的部分将会被点亮，而剩余部分则认为是在阴影中。

实现的步骤:

① 生成深度贴图

```
//深度贴图
    GLuint depthMapFBO;
    glGenFramebuffers(1, &depthMapFBO);
    GLuint shadow_width = 1024, shadow_height = 1024;
    GLuint depthMap;
    glGenTextures(1, &depthMap);
    glBindTexture(GL_TEXTURE_2D, depthMap);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT,
        shadow_width, shadow_height, 0, GL_DEPTH_COMPONENT, GL_FLOAT, NULL);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
```

```
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    float borderColor[] = { 1.0, 1.0, 1.0, 1.0 };
    glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, borderColor);
    glBindFramebuffer(GL_FRAMEBUFFER, depthMapFBO);
    glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, GL_TEXTURE_2D,
depthMap, 0);
    glDrawBuffer(GL_NONE);
    glReadBuffer(GL_NONE);
    glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

这里生成了一个用于存储深度贴图的帧缓存，深度贴图着色器的结果将会放在里面。

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

uniform mat4 lightSpaceMatrix;
uniform mat4 model;

void main() {
    gl_Position = lightSpaceMatrix * model * vec4(aPos, 1.0);
}

//simpleDepthShader.fs
#version 330 core

void main() {

}
```

② 渲染场景

```
float ShadowCalculation(vec4 fragPosLightSpace)
{
    //齐次坐标转普通坐标
    vec3 projCoords = fragPosLightSpace.xyz / fragPosLightSpace.w;
    //归一化
    projCoords = projCoords * 0.5 + 0.5;
    //得到最近光照透视的深度值
    float closestDepth = texture(shadowMap, projCoords.xy).r;
    //得到目前片段的深度值
    float currentDepth = projCoords.z;
    //计算偏差
    vec3 normal = normalize(Normal);
    vec3 lightDir = normalize(lightPos - FragPos);
    float bias = max(0.05 * (1.0 - dot(normal, lightDir)), 0.005);
    //检查目前片段位置是否在阴影中
```

```
    //float shadow = currentDepth - bias > closestDepth  ? 1.0 : 0.0;
    // PCF
    float shadow = 0.0;
    vec2 texelSize = 1.0 / textureSize(shadowMap, 0);
    //周围阴影系数的平均值
    for(int x = -1; x <= 1; ++x)
    {
       for(int y = -1; y <= 1; ++y)
        {
            float pcfDepth = texture(shadowMap, projCoords.xy + vec2(x, y) *
texelSize).r;
            shadow += currentDepth - bias > pcfDepth  ? 1.0 : 0.0;
        }
    }
    shadow /= 9.0;

    // keep the shadow at 0.0 when outside the far_plane region of the light's
frustum.
    if(projCoords.z > 1.0)
        shadow = 0.0;

    return shadow;
}
```

此着色器的基本实现思路是拿每一点的深度与同一光线上最近点的深度比较，如果深度大，则认为改点在阴影中，shadow 系数为 1，否则 shadow 系数为 0。