

Homework 7

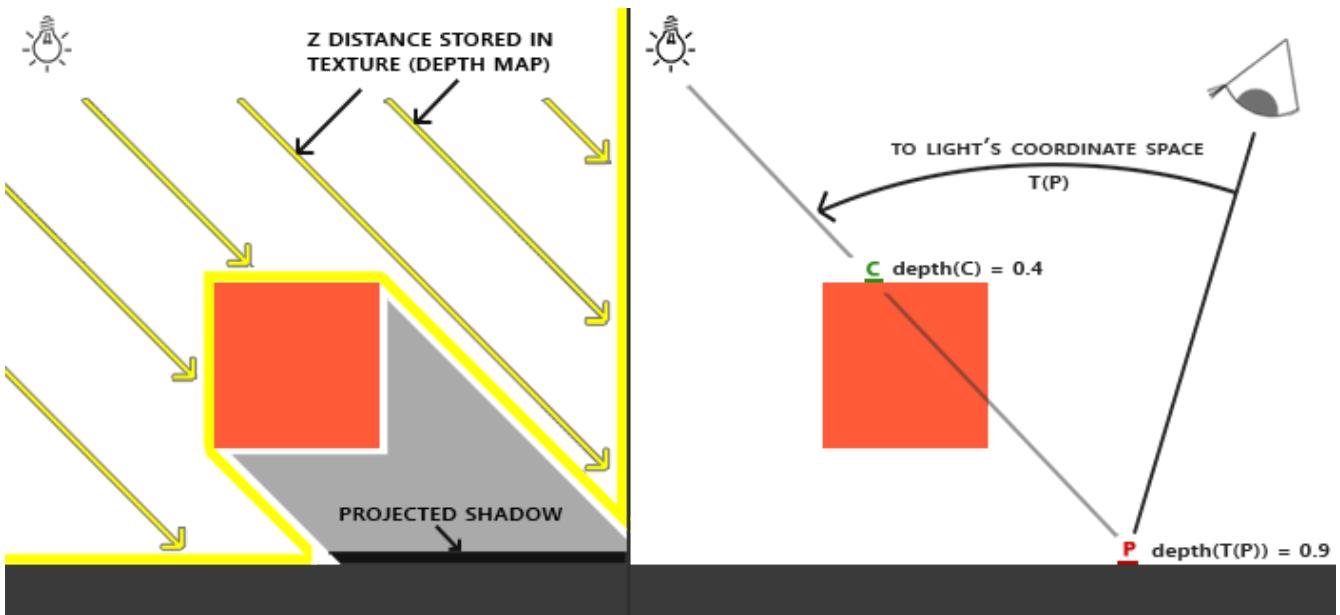
此次作业需要实现Shadow Mapping，并对其进行优化。

阴影渲染

阴影渲染有两大基本步骤：

- 以光源视角渲染场景，得到深度图（DepthMap），并存储为texture
- 以camera视角渲染场景，使用Shadow Mapping算法，决定某个点是否在阴影下。

如下左图，深度贴图会记录光源方向见到的第一个片元（黄色部分），深度在这些之外的片元就处于阴影中。



生成深度贴图，首先需要创建一个帧缓冲对象来存储场景的渲染结果：

```
GLuint depthMapFBO;  
 glGenFramebuffers(1, &depthMapFBO);
```

之后与使用纹理的方式相同，生成一个深度纹理作为帧缓冲的深度缓冲。此时颜色缓冲没有用，所以可以显示告诉OpenGL不进行颜色缓冲：

```
glDrawBuffer(GL_NONE);  
glReadBuffer(GL_NONE);
```

然后就开始生成深度贴图，实际就是使用glBindFramebuffer将帧缓冲绑定后对场景进行渲染。前后需要进行两次渲染，分别是对深度贴图的渲染和使用深度贴图后对场景的渲染，所以要注意glViewport的管理，因为阴影贴图与场景渲染通常有着不同的解析度。阴影贴图的解析度调高可以使阴影的锯齿更少（小）。

在渲染过程中还涉及一个光源空间的变换，可以设置光源的投影方式：透视或者正交。这里因为使用的是一个平行光，所以先使用正交投影，此时透视图将没有任何变形：

```
GLfloat near_plane = 1.0f, far_plane = 7.5f;
glm::mat4 lightProjection = glm::ortho(-10.0f, 10.0f, -10.0f, 10.0f, near_plane,
far_plane);
```

将物体变换到光源视角的空间需要用到变换矩阵，同样用到lookAt函数，这次从光源看向坐标原点处：

```
glm::mat4 lightView = glm::lookAt(glm::vec(-2.0f, 4.0f, -1.0f), glm::vec3(0.0f),
glm::vec3(1.0));
glm::mat4 lightSpaceMatrix = lightProjection * lightView;
```

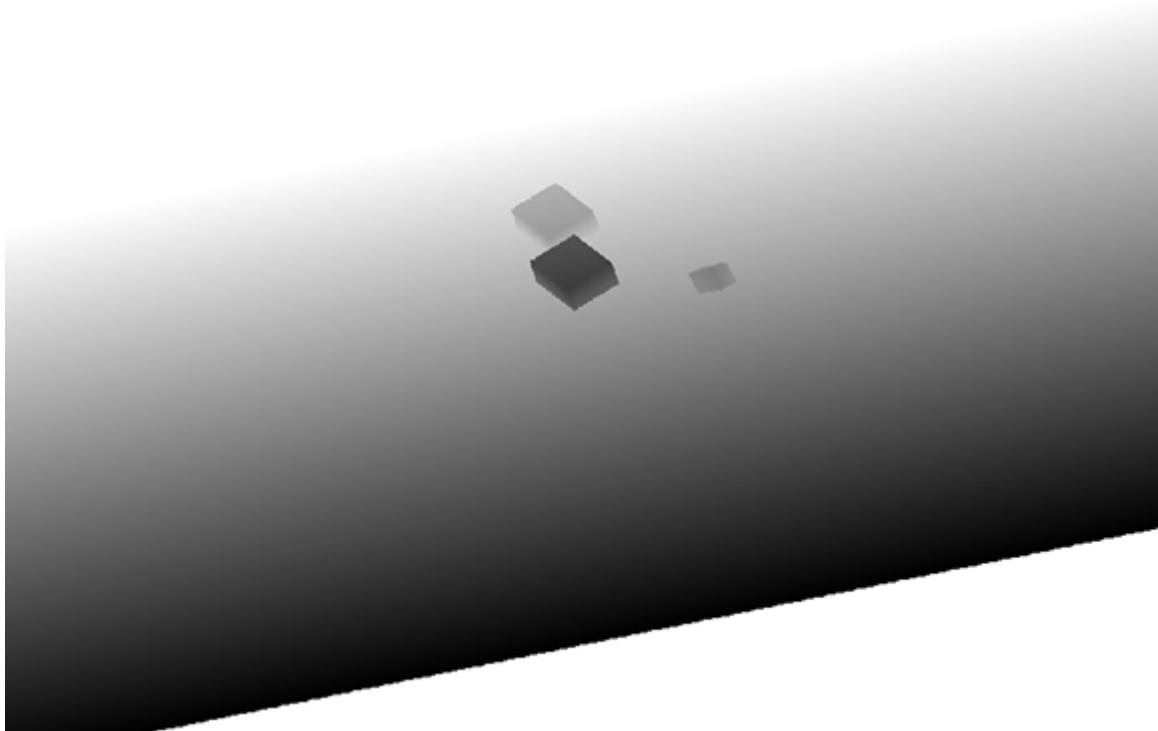
在深度贴图渲染对应的顶点着色器中，顶点位置的设置就要用到lightSpaceMatrix进行转换：

```
#version 330 core
layout (location = 0) in vec3 position;

uniform mat4 lightSpaceMatrix;
uniform mat4 model;

void main()
{
    gl_Position = lightSpaceMatrix * model * vec4(position, 1.0f);
}
```

最后，如果将深度缓冲贴图纹理投射到一个2D四边形上进行显示，其效果类似：



接着进行场景中的阴影渲染。当一个fragment在阴影中时，计算其shadow值为1.0，反之为0.0，然后在进行光照渲染时将其乘上漫反射光照与镜面反射光照的颜色值（因为阴影不会是全黑的）：

```
vec3 lighting = (ambient + (1.0 - shadow) * (diffuse + specular)) * color;
```

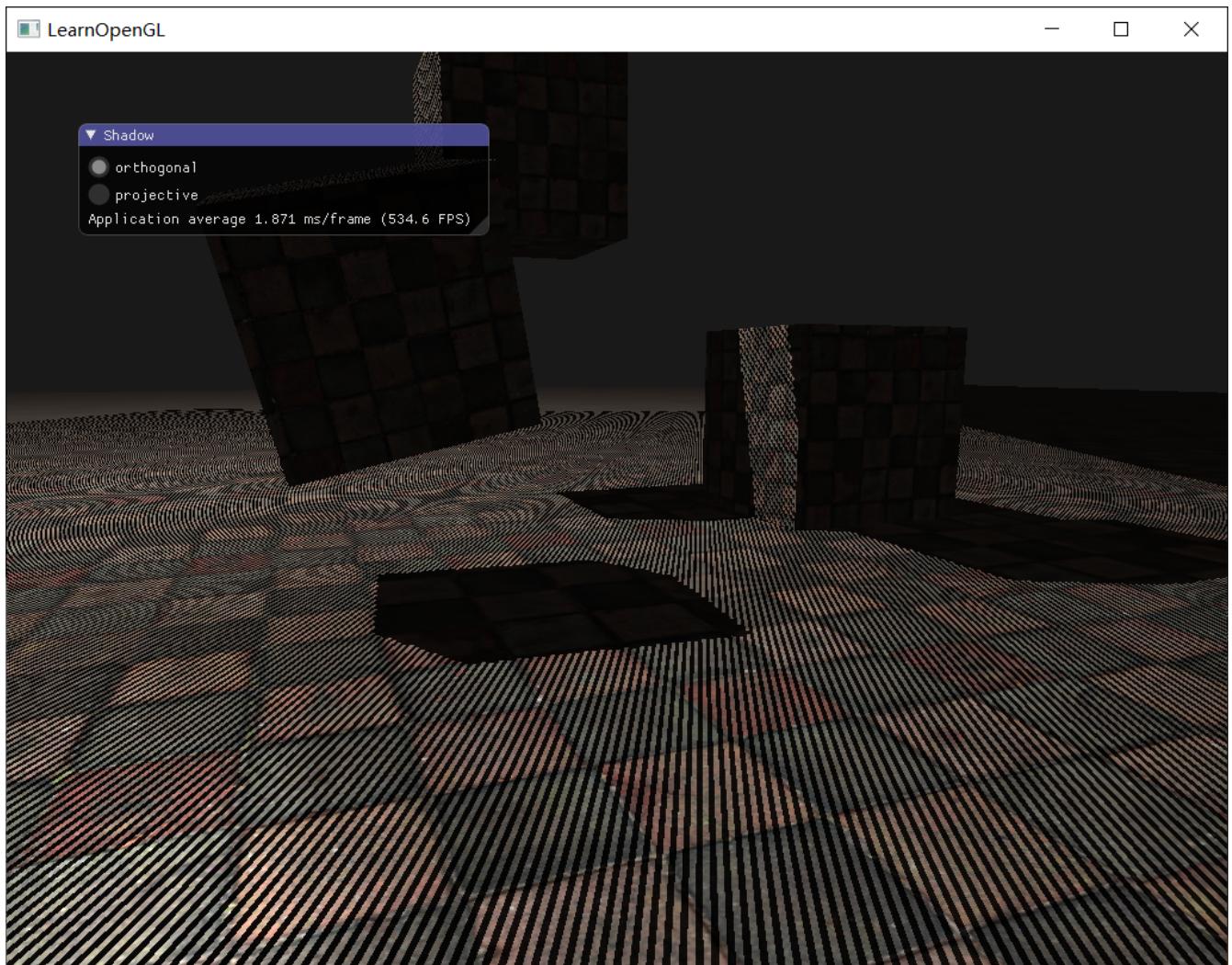
阴影的计算由透视线除法开始，首先将坐标转为标准设备空间坐标的范围[-1,1]，然后转到[0,1]（小于0对于深度贴图的采样来说没有意义）：

```
vec3 projCoords = fragPosLightSpace.xyz / fragPosLightSpace.w;
projCoords = projCoords * 0.5 + 0.5;
```

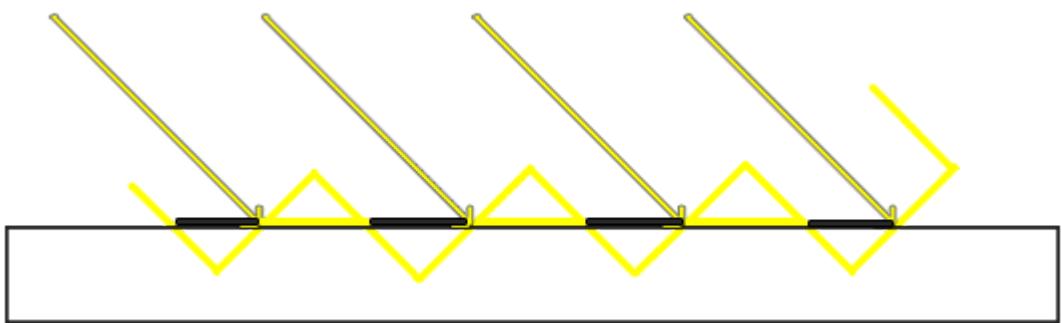
之后得到光源方向下片元的最近深度，检查当前片元的深度是否高于最近深度，如果是则片元在阴影中：

```
float closestDepth = texture(shadowMap, projCoords.xy).r;
float currentDepth = projCoords.z;
float shadow = currentDepth > closestDepth ? 1.0 : 0.0;
```

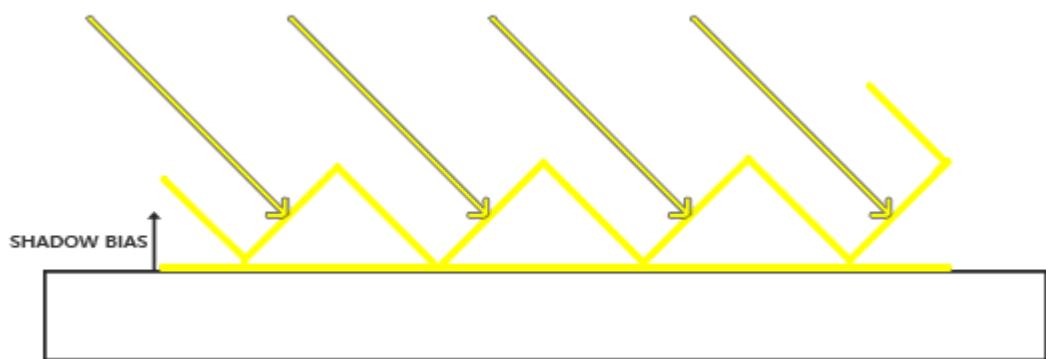
现在得到的结果为：



可以看到有明显的阴影失真，这是由于阴影贴图受限于解析度，在距离光源较远的情况下，多个片元可能从深度贴图的同一个值中去采样，如图：



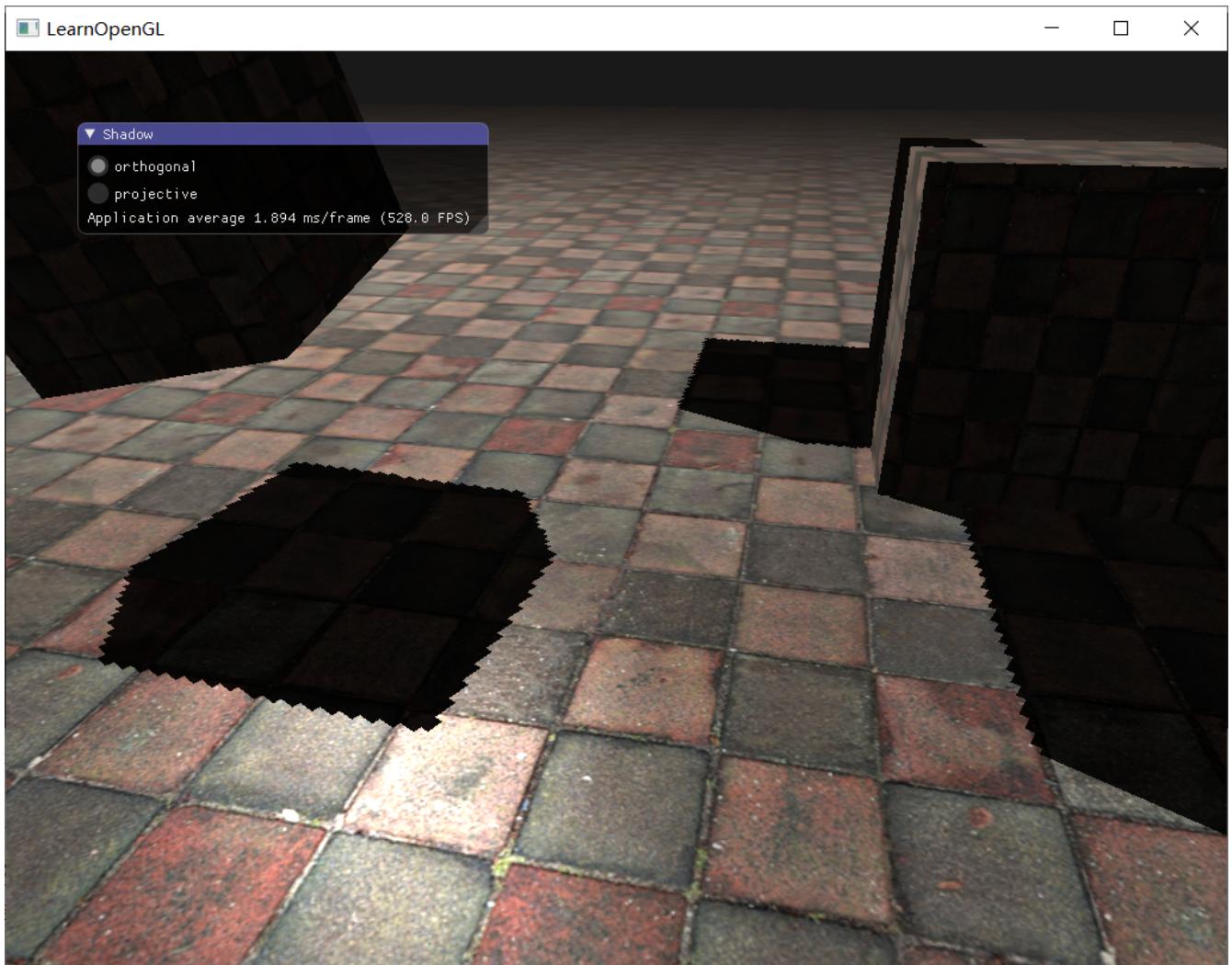
因为我们的光源是以一个角度射向平面的，这时就可能出现这样的阴影失真的问题。可以用一个叫做阴影偏移的技巧来解决这个问题，如图：



即计算阴影的时候增加一个偏移量，去除表面坡度的影响。但这样做当坡度很大时仍会产生阴影失真，单纯增加一个偏移量时不好确定它的值。有一个更加可靠的办法是根据表面朝向光线的角度更改偏移量：

```
float bias = max(0.05 * (1.0 - dot(normal, lightDir)), 0.005);
float shadow = currentDepth - bias > closestDepth ? 1.0 : 0.0;
```

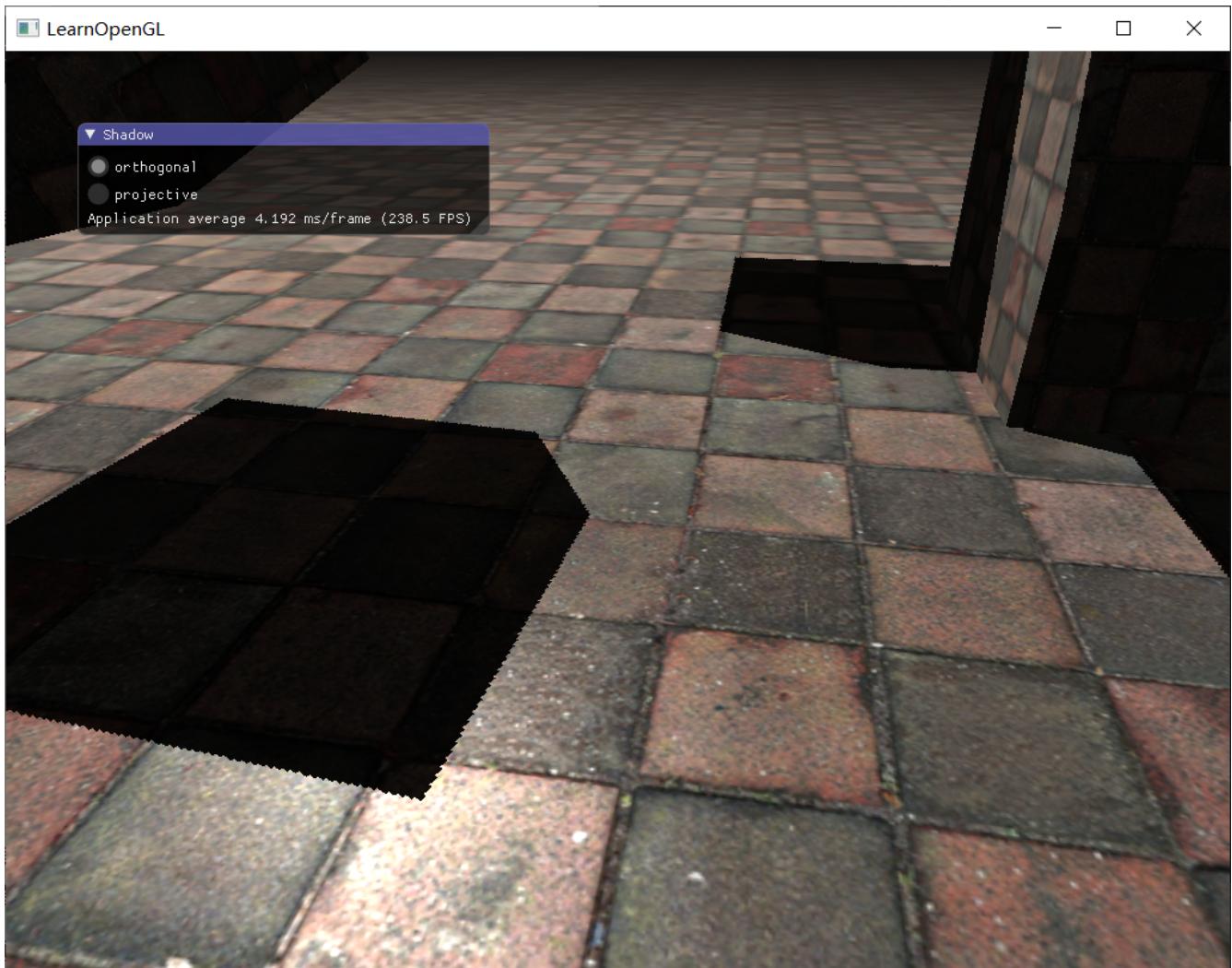
此时的结果为：



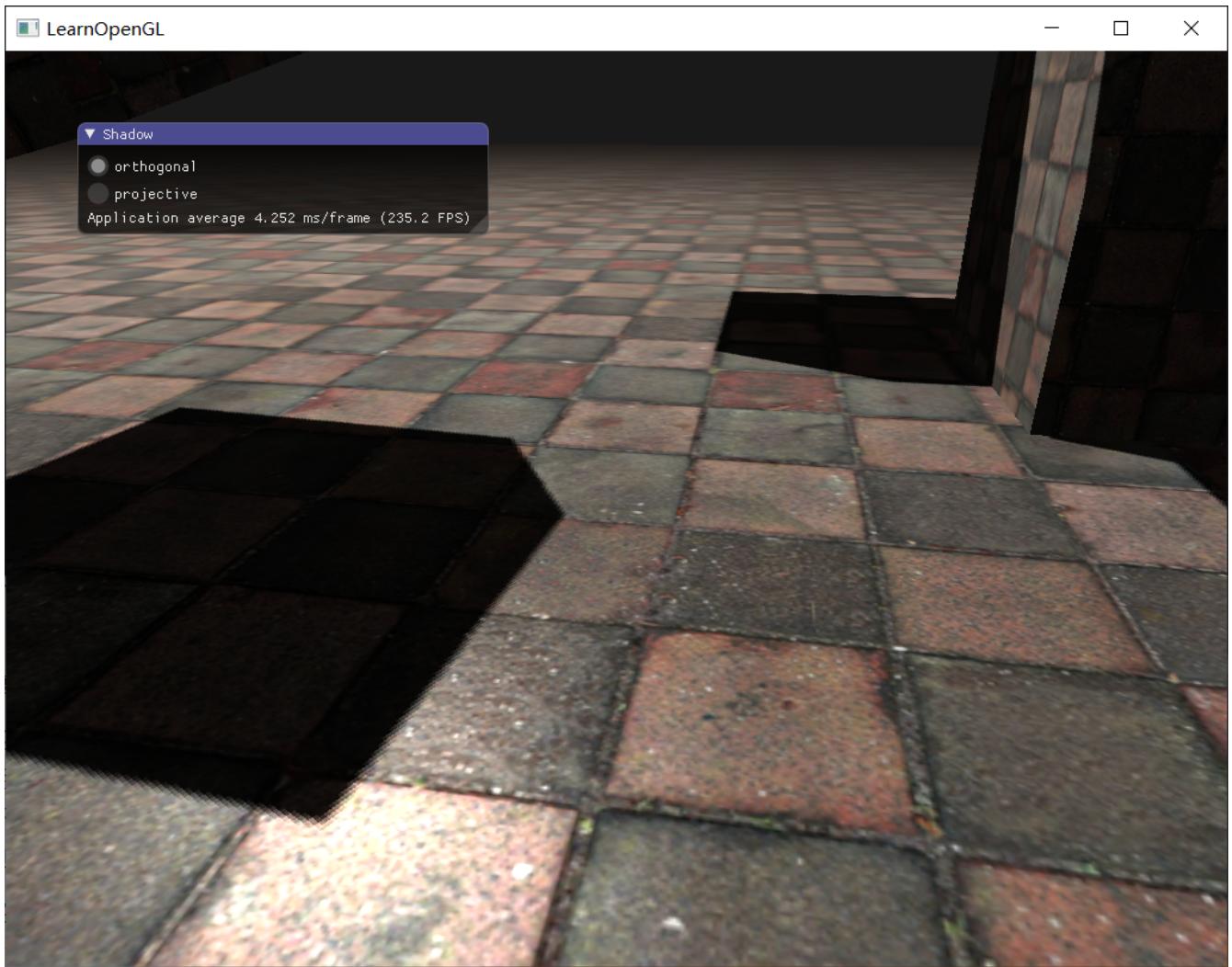
阴影优化

此时阴影的渲染效果已大致成型，但还有一点是不太令人满意的，可以看到，阴影边上的锯齿较为严重。这是因为深度贴图有一个固定的解析度，多个片元对应于一个纹理像素，则多个片元会从同一个深度值进行采样，得到同一个阴影，就产生了锯齿。

首先尝试提高深度贴图的解析度，从 1024×1024 提高到 4096×4096 ：

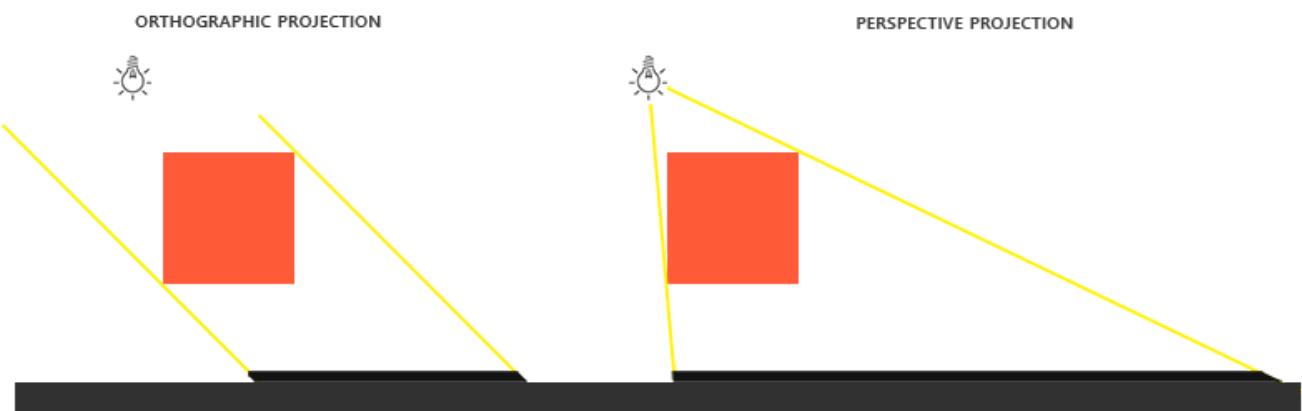


可以看到锯齿变小了许多。还可以对阴影进行一个PCF (percentage-closer filtering)处理，使得锯齿部分变得模糊。实际上就是从纹理像素四周对深度贴图进行采样，然后把结果平均起来。这样的处理效果为：



这样的效果就好多了。

最后进行正交与透视两种投影下的阴影比较，如下图：



透视投影对于点光源来说更合理，这里使用的是平行光，所以得到的结果会较为奇怪（左为正交投影，右为透视投影）：

