**计算机图形学实验报告**

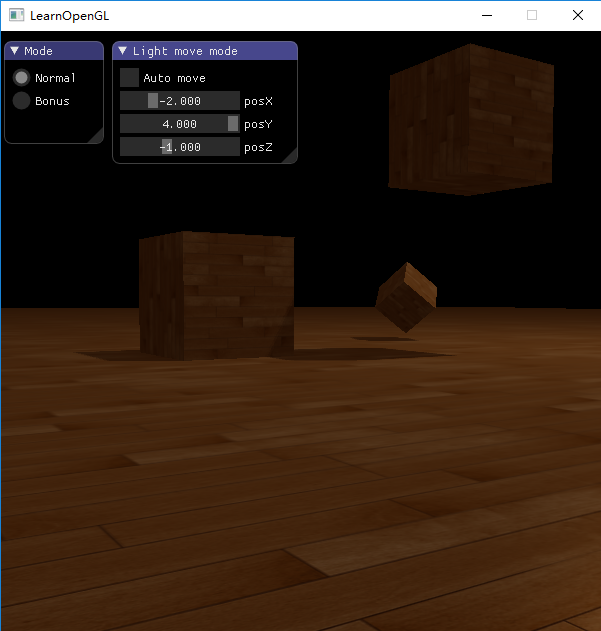
第一部分：实现思路和代码分析

|  |  |
| --- | --- |
| 代码截图 | 代码说明 |
|  | 将前面使用的立方体坐标代码稍作修改，将立方体中的每个顶点的坐标向量改为由位置坐标，法向量，纹理坐标组成的八维向量。 |
|  | 平面的顶点数组，将平面划分成两个三角形，共有六个顶点。每个顶点的同样使用一个八维坐标向量表示，与立方体顶点的表示方法相同。 |
|  | 将着色器Shader的配置独立成一个类，使主体代码更加简洁。这个类提供了着色器的初始化操作，同时支持不同类型的外部变量的绑定（数字，矩阵和向量）。 |
|  | 着色器Shader类的功能函数具体实现。这部分的代码主要实现了着色器代码的外部变量的绑定以及着色器程序的调用。 |
|  | 着色器Shader类的功能函数具体实现。这部分与前几次作业中的着色器生成程序一致，首先编译顶点着色器，然后编译片段着色器，最后创建着色器程序。 |
|  | 生成深度贴图所需的顶点着色器。这部分主要是将顶点坐标变换到光空间中。 |
|  | 生成深度贴图所需的片段着色器。由于深度贴图没有颜色缓冲，因此最后的片元不需要任何处理，因此使用一个空的片段着色器即可。 |
|  | 渲染阴影的顶点着色器。该顶点着色器使用同一个光空间转换矩阵，把世界空间顶点位置转换为光空间。顶点着色器传递一个普通的经变换的世界空间顶点位置和一个光空间的顶点位置给像素着色器。 |
|  | 渲染阴影的顶点着色器。本次作业的顶点着色器使用Blinn-Phong光照模型渲染场景。接着计算出一个shadow值，当片元在阴影中时shadow值为1.0，在阴影外是0.0。计算过程大致如下：把光空间片元位置转换为裁切空间的标准化设备坐标；将坐标向量都需要变换到[0,1]范围；得到光的位置视野下最近的深度；得到片元的当前深度；判断片元是否在阴影中。这部分还包括了一些阴影优化的代码。 |
|  | 开启深度测试，以及着色器和纹理贴图的初始化。 |
|  | 平面顶点的初始化。与立方体创建方式相同。 |
|  | 为渲染的深度贴图创建一个帧缓冲对象。然后，创建一个2D纹理，提供给帧缓冲的深度缓冲使用。接着，把生成的深度纹理作为帧缓冲的深度缓冲。 |
|  | 渲染深度贴图。使用深度着色器进行场景的渲染。这里需要初始化视窗的大小，因为阴影部分一般比原有的视窗大小要更大。 |
|  | 渲染一般场景贴图。使用常规着色器进行场景的渲染。这里首先要初始化视窗（viewport）的参数以适应阴影贴图的尺寸，否则将可能导致阴影部分显示不完整。 |
|  | 对待渲染的平面和立方体进行变换操作：平面不进行变换操作，三个立方体分别进行平移，旋转，缩放操作。 |
|  | 读取贴图并生成纹理。 |

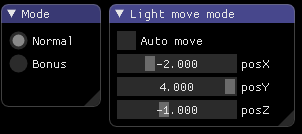
第二部分：实验截图分析

（1）基本要求

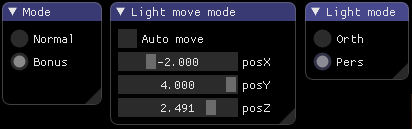
1. 实现方向光源的Shadowing Mapping：要求场景中至少有一个object和一块平面(用于显示shadow)；光源的投影方式任选其一即可；解释Shadowing Mapping算法。
2. 效果图：



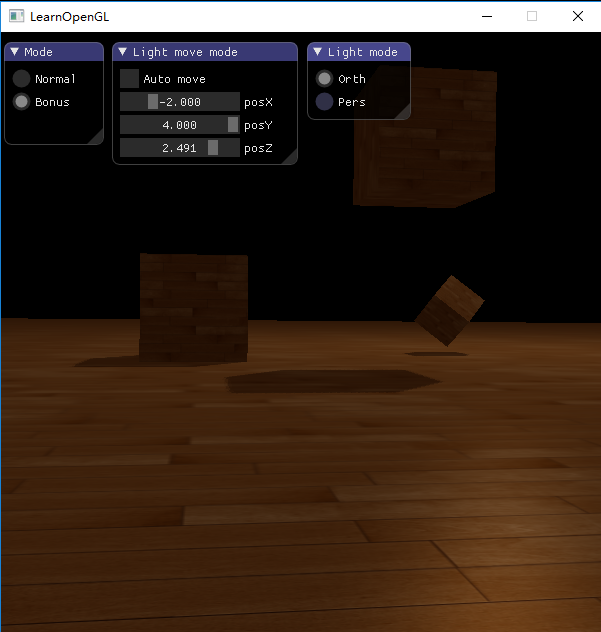
1. 解释Shadowing Mapping算法：
2. 修改GUI
3. 一般情况

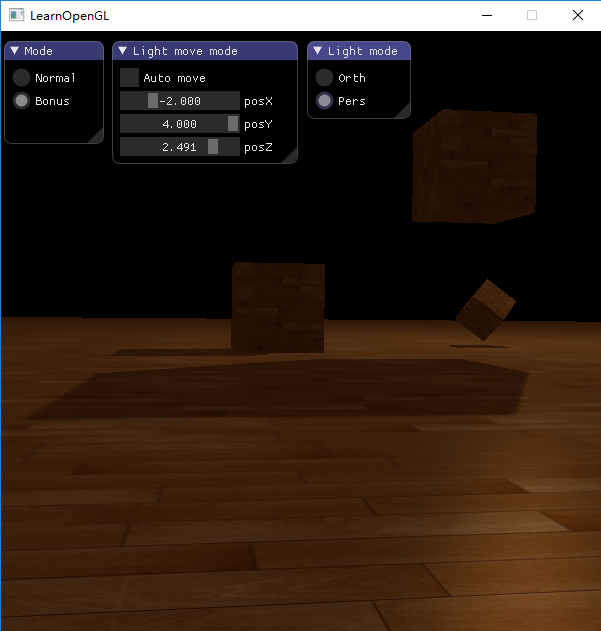


1. Bonus



1. Bonus
2. 实现光源在正交/透视两种投影下的Shadowing Mapping





1. 优化Shadowing Mapping