**练 习 题 报 告**

|  |
| --- |
| **课程名称 计算机图形学**  **项目名称 投影和硬阴影**  **学 院 计算机与软件学院**  **专 业 软件工程（腾班）**  **指导教师 熊卫丹**  **报 告 人 黄亮铭 学号 2022155028** |

1. **练习目的**
2. 熟悉在OpenGL中实现正交投影变换。
3. 了解使用投影变换实现场景的硬阴影效果。

**二．练习完成过程及主要代码说明**

Task-1：完善lookAt函数、ortho函数和perspective函数

1. **完善lookAt函数**

**完善思路：**

* 1. 我们首先要初始化一个四维的单位矩阵。
  2. 然后根据相机的位置eye和物体中心（参考点）at计算VPN并归一化处理。使用数学公式表示为：
  3. 再通过at和VPN生成与两者均垂直的方向向量并对其归一化，同时计算得到up（相机胶片平面的上方）在相机胶片平面上的投影。使用数学公式表示为：
  4. 接下来我们将相机从坐标原点移动到视点，这一步骤使用实验2中的平移矩阵可以完成。使用数学公式表示为：

**代码实现：**首先初始化一个四维单位矩阵，然后计算VPN并归一化，再通过VUP和VPN计算与它们都垂直的方向向量，接下来计算VUP在照相机胶片平面上的投影，最后平移矩阵。

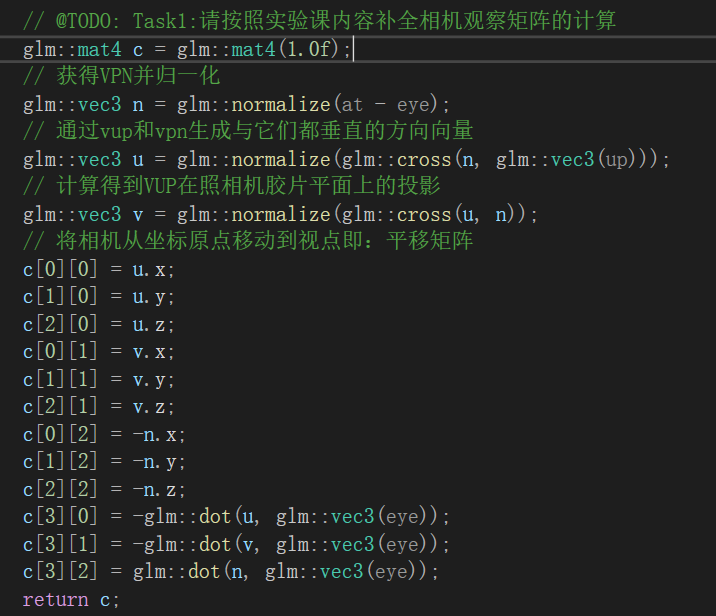


图1：lookAt函数

1. **完善ortho函数**

**完善思路：**

实验3.1实验文档中已经给出正交投影的投影矩阵，如图2所示。我们需要根据给定的投影矩阵对矩阵内的元素进行赋值。

****

图2：正交投影矩阵

**代码实现：**首先初始化一个四维矩阵为单位矩阵，然后将矩阵看成是二维数组，根据给定的投影矩阵填充四维矩阵，最后将四维矩阵作转置操作，因为在glm中矩阵的表示和二维数组的表示不同。

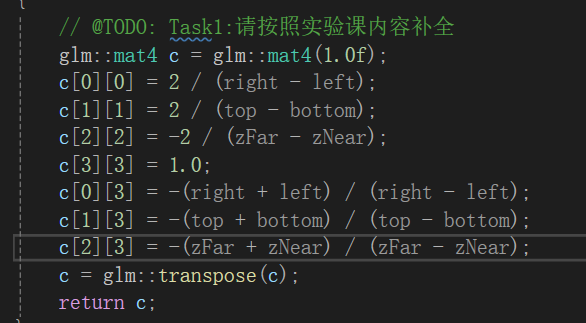


图3：ortho函数

1. **完善perspective函数**

**完善思路：**

实验3.1的实验文档中已经给出透视投影的投影矩阵，如下所示。我们需要根据给定的投影矩阵对矩阵内的元素进行赋值。

****

给定的透视投影矩阵为类似正交投影的棱台视见体。此外还有一种视域视见体，两者是等价的。

**代码实现：**首先初始化一个四维矩阵为单位矩阵，因为glm的矩阵表示和上述矩阵N的表示的转置相同，所以将矩阵N的数值填入四维矩阵时，手动进行转置。



图4：perspective函数

Task-2：完善display()函数

**完善思路：**在display函数中，我们需要实现绘制三角形阴影的代码。事实上，绘制三角形阴影等价于绘制三角形。但是，绘制三角形阴影比绘制三角形多了一些限制，例如需要考虑光源的位置和三角形的位置。因此，我们在三角形的变换矩阵前乘上一个投影矩阵即可，即变换矩阵左乘投影矩阵。

投影矩阵在实验3.2的实验文档中给出，如下所示。



**代码实现：**首先初始化一个四维矩阵为零矩阵，然后实验文档根据给定的投影矩阵填充数值，注意glm的矩阵和数学中的矩阵的表示的差异，最后投影矩阵右乘模型的变换矩阵。

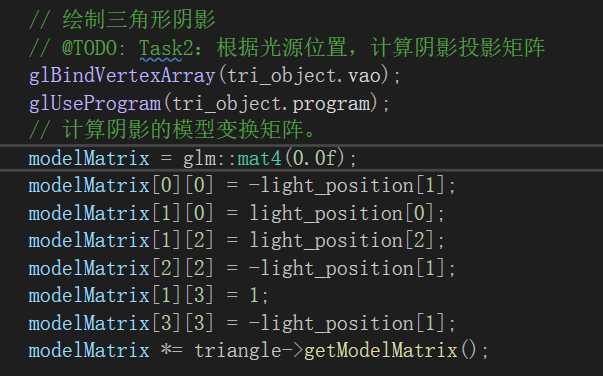


图5：display函数

其他：键盘控制光源

首先在main函数中判定键盘回调函数，然后在键盘回调函数中获取鼠标点击的位置，然后根据鼠标的点击位置设置新的l即光源的位置。具体代码如下图所示。

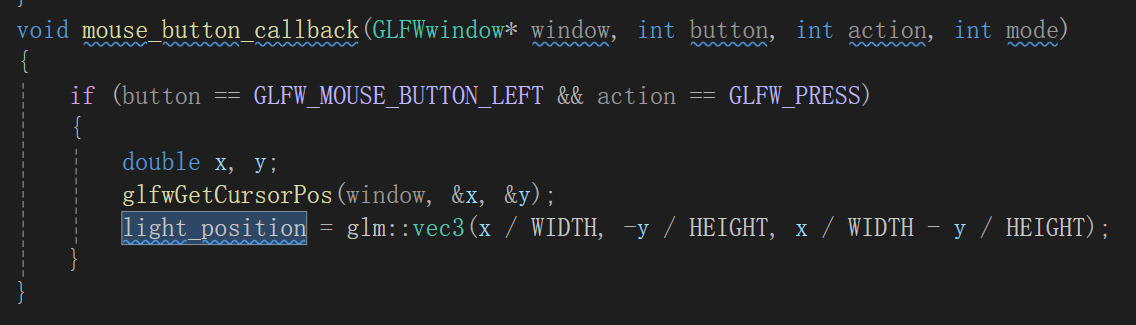


图6：键盘回调函数

结果展示

为了更加直观的展示阴影的移动，这里使用GIF动图进行展示，而非图片。（动图需要使用word文档打开）

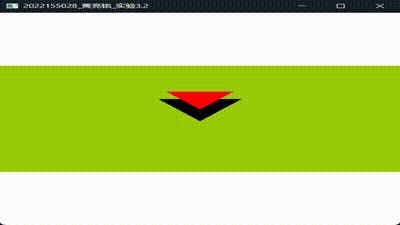


图7：结果展示

**三．实验总结**

* 1. 本次实验，我完成了对Camera.cpp文件中的lookAt函数、ortho函数和perspective函数的完善。同时，我也完成了对main.cpp文件中的display函数的完善。
  2. 运行程序后，通过键盘我观察到投影生成的三角形阴影受光照位置变化和三角形旋转而产生的移动符合预期，证明了代码的正确性。
  3. 通过本次实验，我熟悉了如何使用glm库实现正交投影变换。
  4. 通过本次实验，我了解了如何使用投影变换实现场景的硬阴影效果。