**计算机图形学 Homework5**

15331416 赵寒旭

**1. 运行结果**

1）正交投影

参数选择通过移动滑块自行调整。

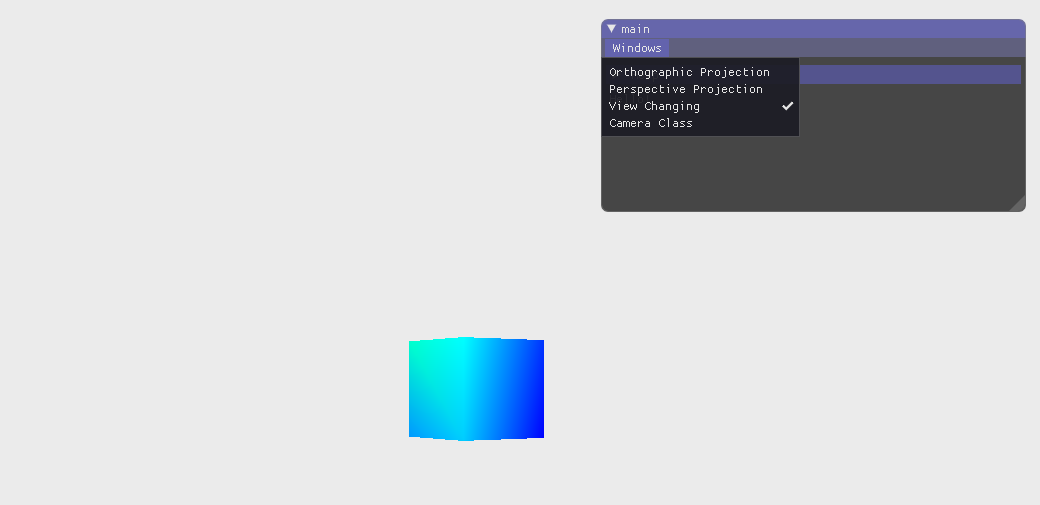


2）透视投影

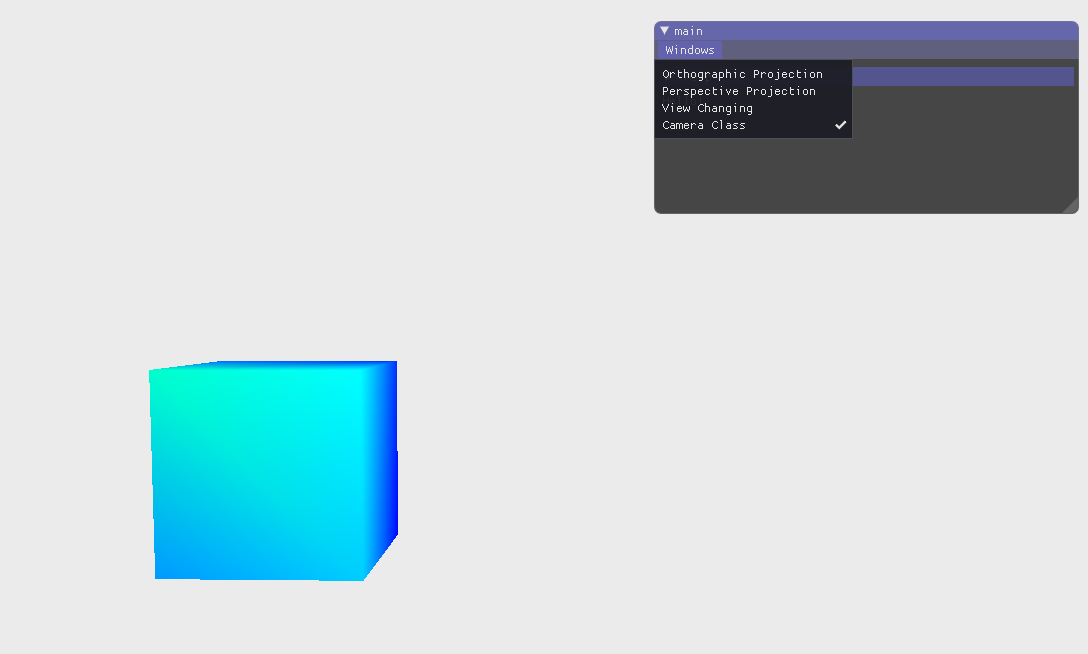
参数选择通过移动滑块自行调整。



3）视角变换



4）Camera类



另有视频演示在/doc文件夹下。

**2. 问题回答**

问：在显示生活中，我们一般将摄像机摆放的空间View matrix和被拍摄的物体摆设的空间Model matrix分开，但是在OpenGL中却将两个合二为一设为ModelView matrix，通过上面的作业启发，你认为是为什么呢？（你可能有不止一个摄像机）

答：我们获得物体最终的屏幕坐标是靠几次坐标变换得到的，从物体的局部坐标开始，先通过Model matrix将其转换到世界坐标，再由View matrix转换到观察空间坐标，考虑有多个摄像机的情况，如果我们将这两次操作合并为一个变换矩阵ModelView matrix，可以减少计算量，同时获得和分步操作完全相同的结果。

**3. 实现思路**

**3.1 投影**

把上次作业绘制的cube放置在(-1.5,0.5,-1.5)位置，要求6个面颜色不一致。

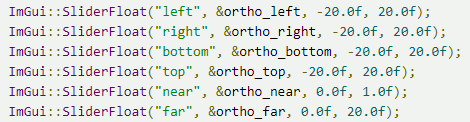
1）正交投影

实现正交投影，使用多组（left, right, bottom, top, near, far）参数，比较结果差异。

（1）cube放置在(-1.5,0.5,-1.5)位置



（2）用ImGui中的滑块修改参数值，比较不同参数组合下的结果差异



（3）正交投影



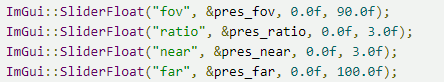
2）透视投影

实现透视投影，使用多组参数，比较结果差异。

（1）cube放置在(-1.5,0.5,-1.5)位置



（2）用ImGui中的滑块修改参数值，比较不同参数组合下的结果差异



（3）透视投影



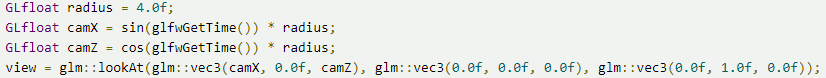
**3.2 视角变换**

把cube放置在(0,0,0)处，做透视投影，使摄像机围绕cube旋转，并且时刻看着cube中心。

（1）把cube放置在(0,0,0)处



（2）使摄像机围绕cube旋转，并且时刻看着cube中心



lookAt三个参数分别表示：摄像机位置，目标位置，表示上向量的世界空间中的向量（我们使用上向量计算右向量）。

为每一帧创建摄像机的x和z坐标确保摄像机在一个圆周上运动，通过重复计算camX和camZ来遍历所有圆圈上的点，这样摄像机就会绕着场景旋转。

预定义圆的半径radius为4.0f。

我们将目标位置设为(0,0,0)保证摄像机始终看着cube中心。

此时GLM会创建一个LookAt矩阵，我们可以把它当作我们的观察矩阵，每次迭代都会创建一个新的观察矩阵。

（3）做透视投影



**3.3 Camera类的实现**

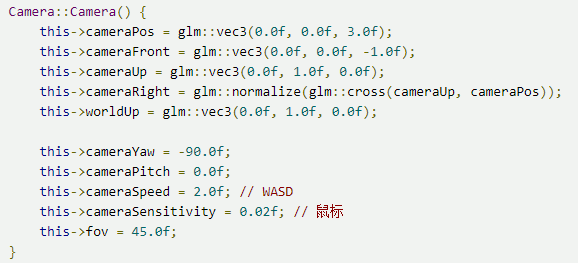
要求实现一个camera类，当键盘输入w，a，s，d，能够前后左右移动，当移动鼠标，能够视角移动，即类似FPS的游戏场景。

1）Camera类定义



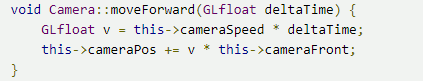


默认构造函数，初始化各个数据成员的值：



2）键盘控制移动

以向前移动函数为例，WASD其他操作同理：

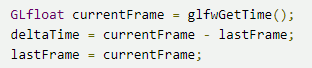


此处参数deltaTime存储上一帧所用的时间，把所有速度都去乘以deltaTime值，使摄像机的速度一直保持一致。

初始化：



每次渲染时更新：



具体按键及响应由输入控制函数processInput函数控制：

main函数中调用



函数具体定义：



响应修改对象camera中摄像机位置的值，下一次渲染时获取新的view矩阵：

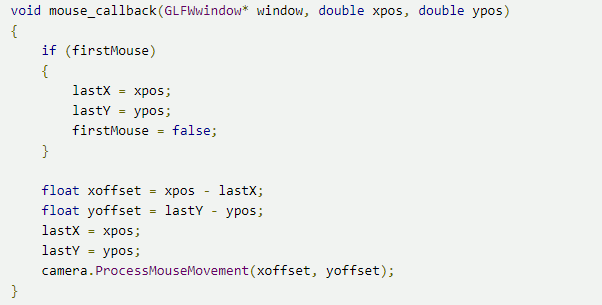


此函数返回一个LookAt矩阵作为观察矩阵：



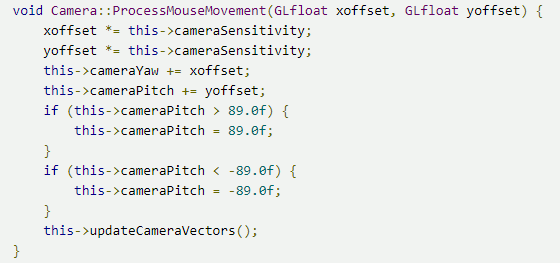
3）鼠标控制移动视角

响应鼠标移动的回调函数：

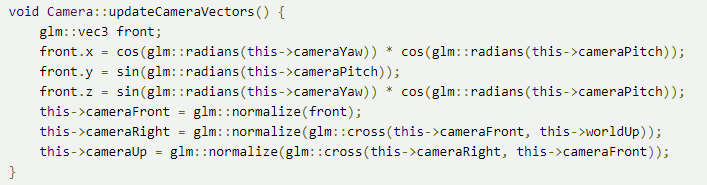


在回调函数中计算当前帧和上一帧鼠标位置的偏移量，传入Camera类的成员函数ProcessMouseMovement控制摄像机的视角变化。

把偏移量乘上预先定义的cameraSensitivity控制视角的变化程度，同时为了防止视角的过分变换和超越，限制俯仰角不能超过89度也不能低于-89度。



此时我们获得了新的俯仰角和偏航角，可以据此得到新的由相机位置到观察目标的front向量cameraFont并更新相关数据：



在下一次渲染时，重新获取LookAt矩阵作为view矩阵，就可以更新观察矩阵直接感受到视角的变化。