

计算机图形学实验

实验5、Transformation变换

姓 名：\_\_\_\_黄子安\_\_\_\_\_

学 号：22920212204396\_

学 院：\_\_\_\_信息学院\_\_\_

专 业：\_\_\_\_软件工程\_\_\_

年 级：\_\_\_\_2021级\_\_\_\_

2023年 5月 10 日

目录

[Task1: myFrustum 3](#_Toc135643994)

[1.1流程分析 3](#_Toc135643995)

[1.2代码截图 5](#_Toc135643996)

[1.3运行效果 5](#_Toc135643997)

[Task2: myOrtho 6](#_Toc135643998)

[2.1流程分析 6](#_Toc135643999)

[2.2代码截图 6](#_Toc135644000)

[2.3运行效果 6](#_Toc135644001)

[Taks3:myLookAt 7](#_Toc135644002)

[3.1流程分析 7](#_Toc135644003)

[3.2代码截图 8](#_Toc135644004)

[3.3运行效果 8](#_Toc135644005)

[Task4: myViewport 10](#_Toc135644006)

[4.1流程分析 10](#_Toc135644007)

[4.2代码截图 10](#_Toc135644008)

[4.3运行效果 11](#_Toc135644009)

[Task5:myPerspective 12](#_Toc135644010)

[5.1流程分析 12](#_Toc135644011)

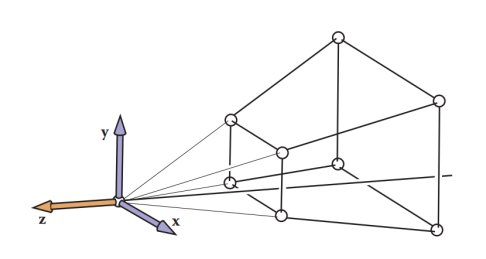
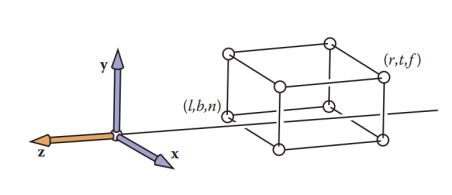
[5.2代码截图 13](#_Toc135644012)

[5.3运行效果 13](#_Toc135644013)

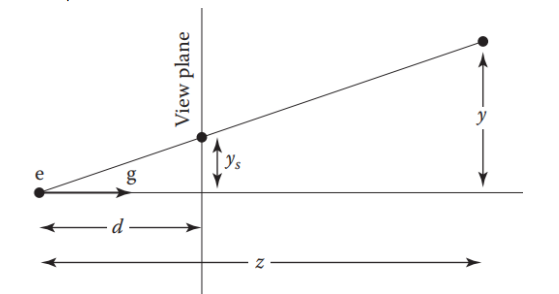
# Task1: myFrustum

## 1.1流程分析

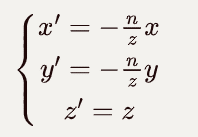
首先将图中的视锥体区域转换为一个正交投影区域，只要根据其中点的z坐标值进行对应的相似运算即可



对应的侧视图如图所示，因此可以得到几何关系为

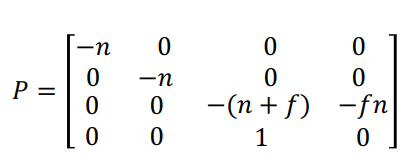


因此在相机齐次坐标系和myFrustum参数下一个对应的坐标变换公式为

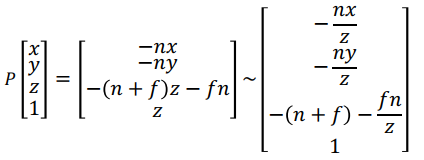


这里要带上负号是因为myFrustum所提供参数是正的，但是在相机坐标系中物体的为负号，因此要补上一个负号使得在变换前后符号不发生改变

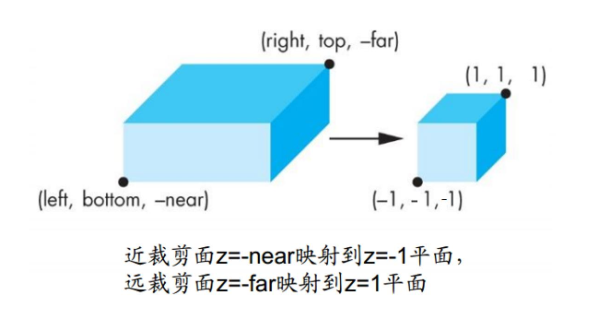
之后为实现矩阵运算，将其转换为齐次坐标表示，但是这样子依旧会有一个问题，就是从原来坐标系下的齐次坐标到现在的不是线性变换，无法用矩阵表示，所以刚才坐标变换公式下让保持不变是错误的。从变换后坐标出发，设变换后为，为了方便起见希望变换后的齐次化的坐标即的取值范围和之前的坐标范围保持不变，因此可以计算出k，b两个待定参数的值，因此得到变换后的坐标应该为，于是求得从视锥体到正交区域的变换矩阵为



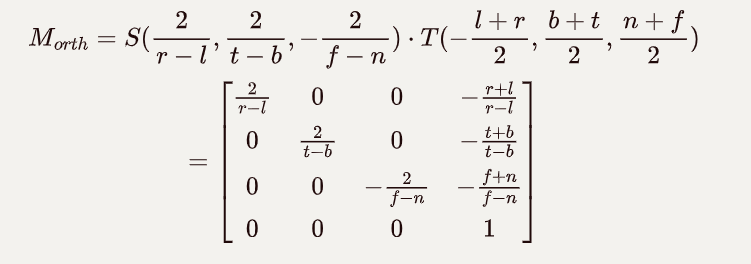
因此得到的坐标变换过程就是



接下来要做的就是将变换得到的正交投影区域变成规范化正交投影，即

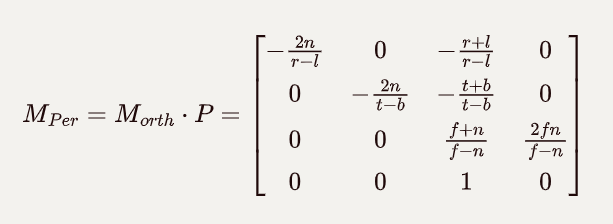


分为两步进行，先将中心移到原点，再进行坐标缩放，刚才得到的正交投影区域的两个对角线顶点坐标为和,所以对应的规范化矩阵为

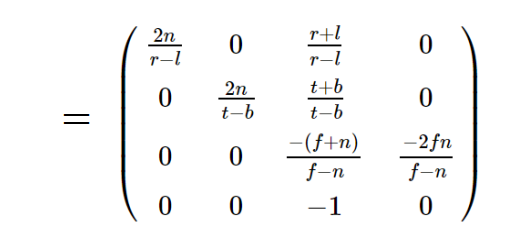


注意这里要把远裁剪平面映射到，而近裁剪平面映射到，相当于做了一个翻转操作，形参中的因此需要补上一个负号

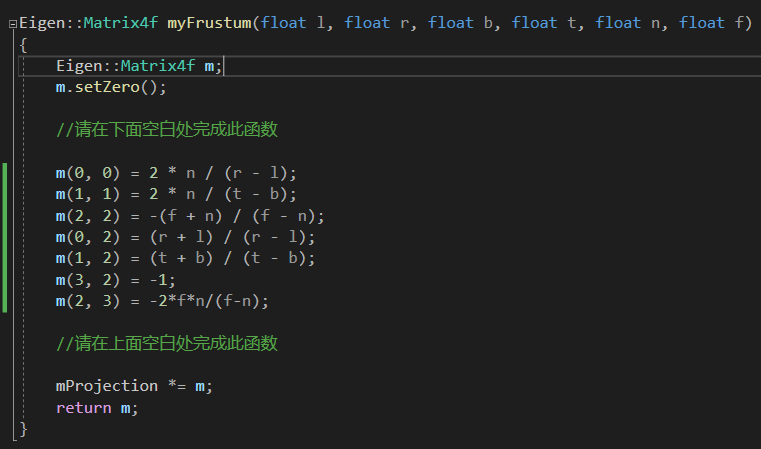
于是得到透视投影的矩阵为



注意到这是齐次坐标，可以所有列乘以-1，此时矩阵的变换效果不会发生改变，所以就可以得到PPT上对应版本的投影矩阵

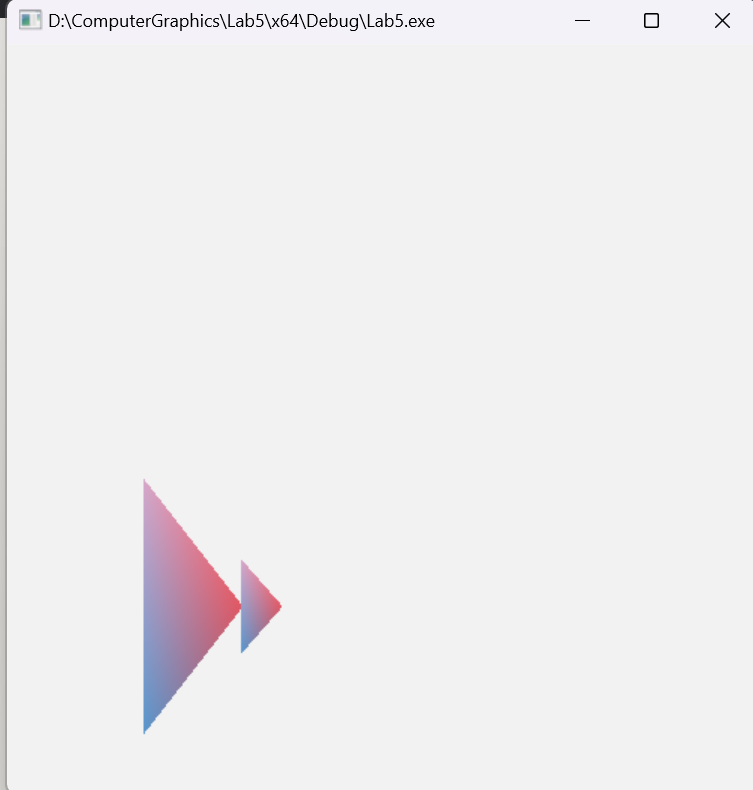
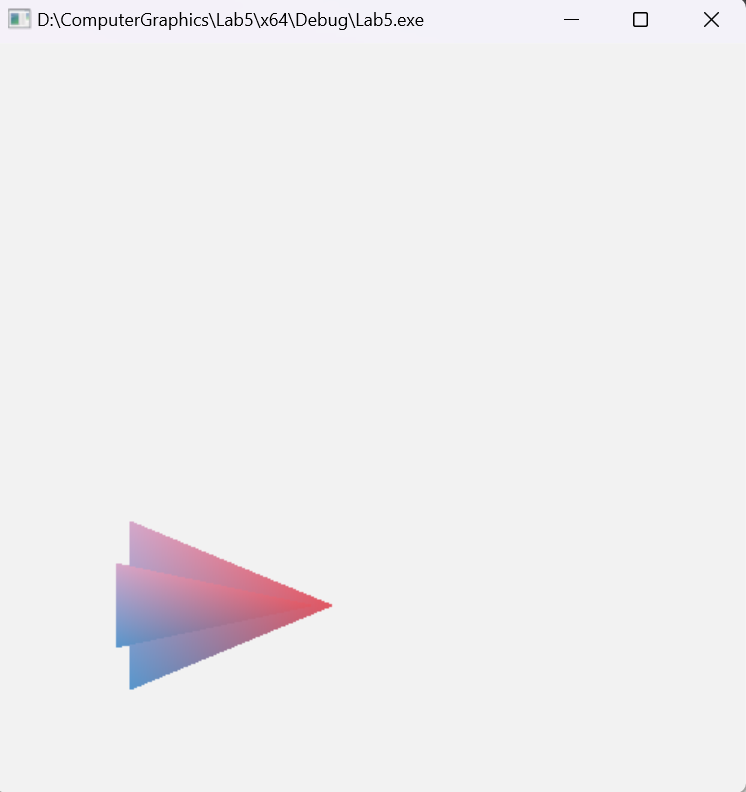


## 1.2代码截图



## 1.3运行效果

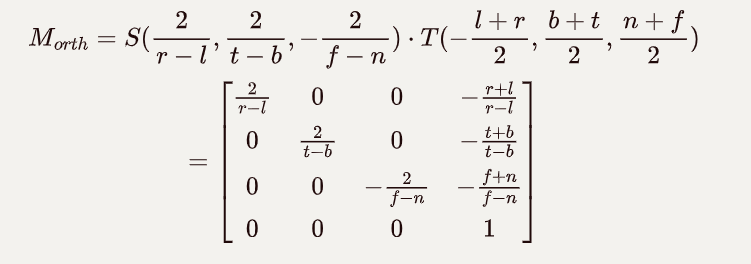
在窗口左下角观察到两个旋转的三角形



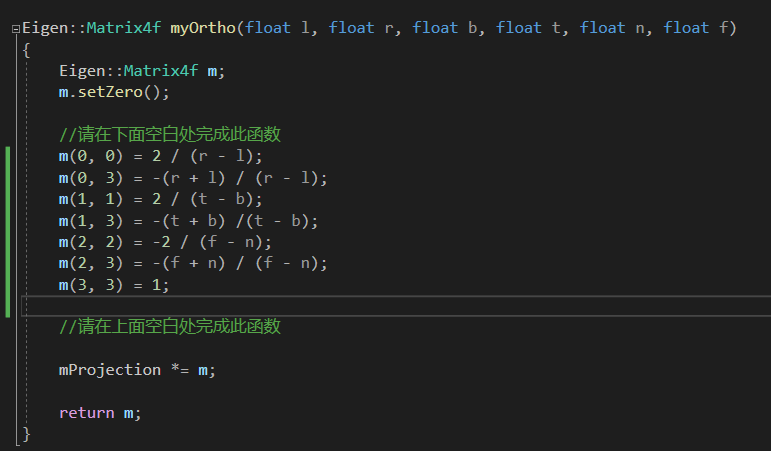
# Task2: myOrtho

## 2.1流程分析

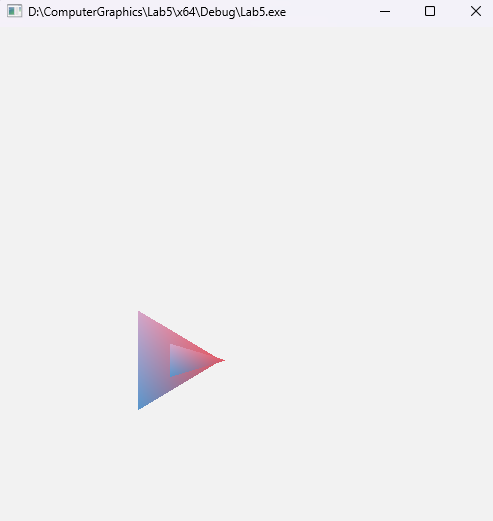
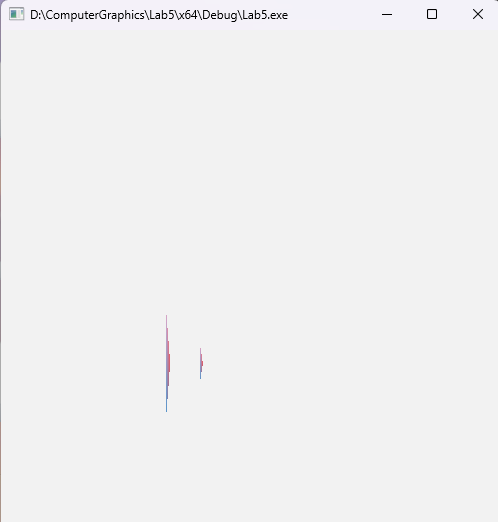
myOrtho的矩阵推导已经在上一个任务中实现



## 2.2代码截图

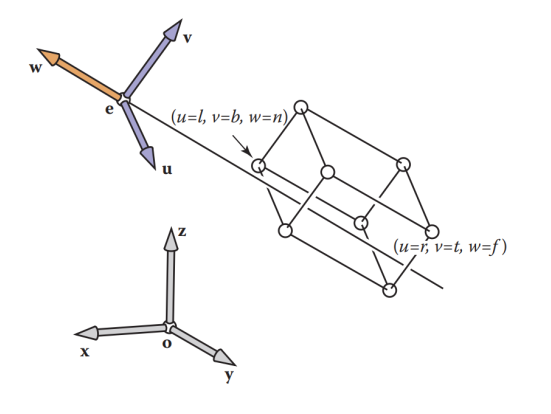


## 2.3运行效果



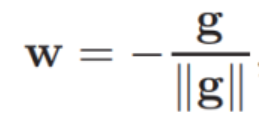
# Taks3:myLookAt

## 3.1流程分析

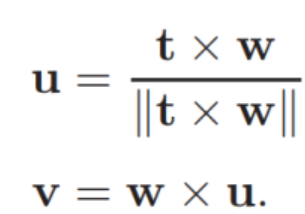


首先利用所提供的参数计算相机坐标的正交单位向量基，所提供的参数包含相机的位置，凝视的方向向量以及向上向量

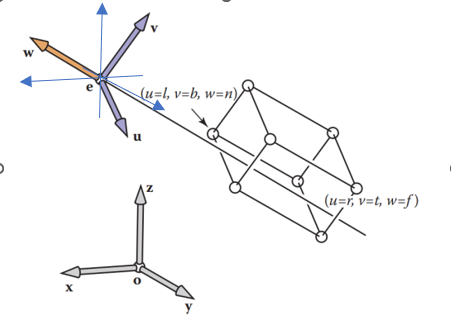
根据凝视方向可以得到相机坐标的第一个轴的向量，这里需要注意在openGL中相机是在相机坐标系的原点，面向轴的负半轴，因此需要加上一个负号



理论上所给向上向量会和凝视向量正交，但是实际openGL也能处理不正交的情况，这里可以先计算，再通过叉乘计算

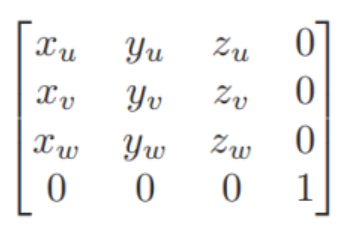


计算出基向量之后就可以计算对应的变换矩阵，分为两步，第一步先平移坐标，之后再进行坐标的旋转

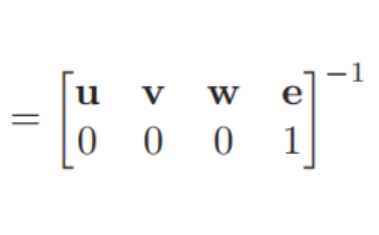
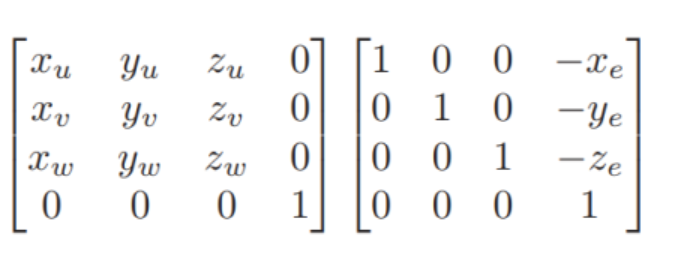
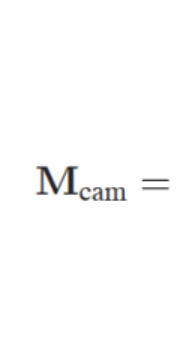


图中蓝色坐标系和世界坐标系三个轴平行，原点和相机坐标系的原点重合，要得到蓝色坐标系下点的坐标只需要根据运动的相对性左乘矩阵即可，之后进行坐标的旋转

旋转后的基向量为,因此过渡矩阵为,因为是正交矩阵，因此其逆矩阵就是原矩阵的转置，扩展成齐次坐标后得到



于是得到变换矩阵为



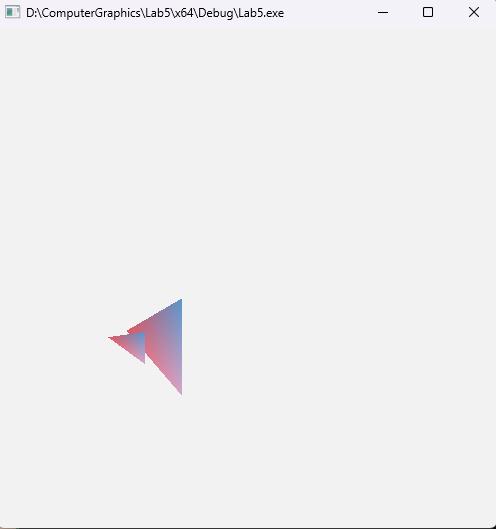
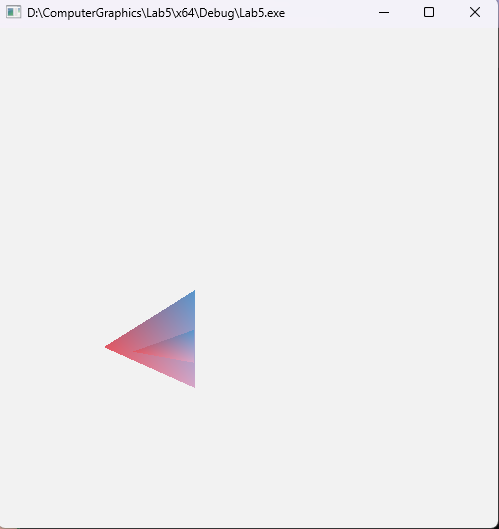
## 3.2代码截图

利用Eigen库自带的叉乘和矩阵乘法功能得到对应的变换矩阵，需要注意的是函数参数中的向量是看向的点坐标，所以要得到的凝视向量是



## 3.3运行效果

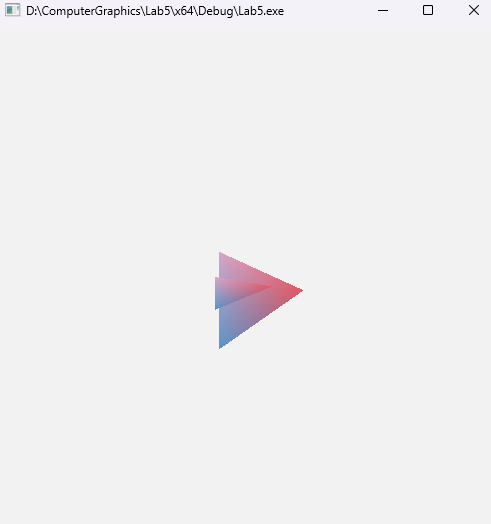




适当修改参数进行测试

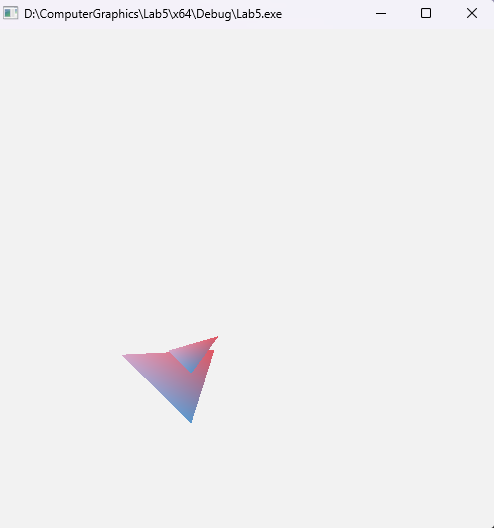


此时物体被拉到窗口中心





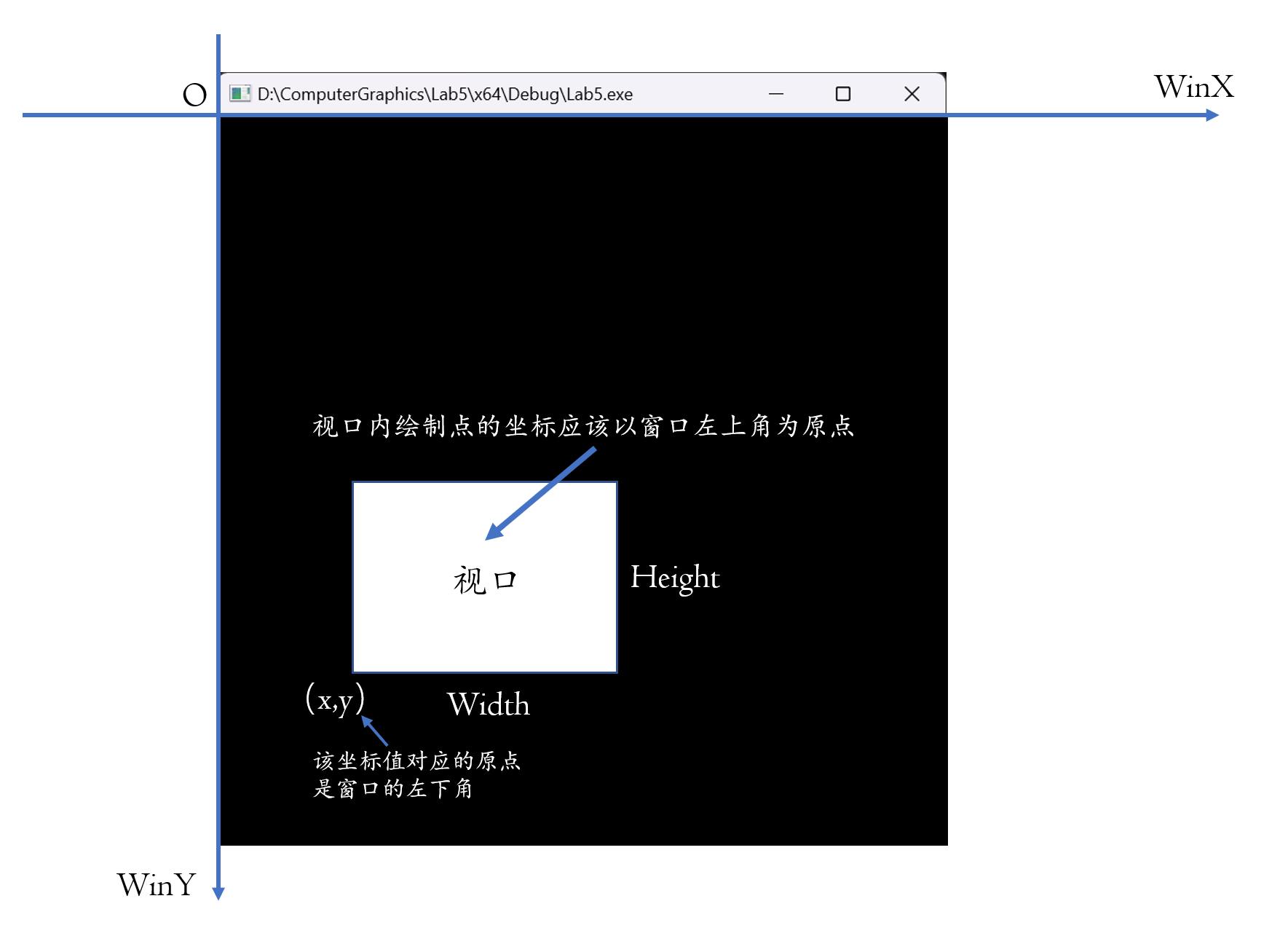
摄像机发生倾斜



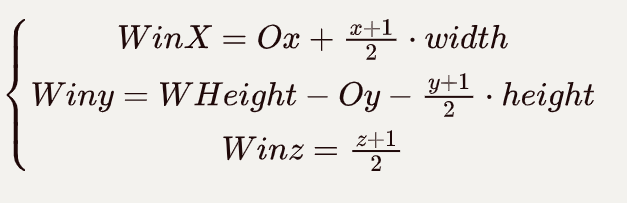
# Task4: myViewport

## 4.1流程分析

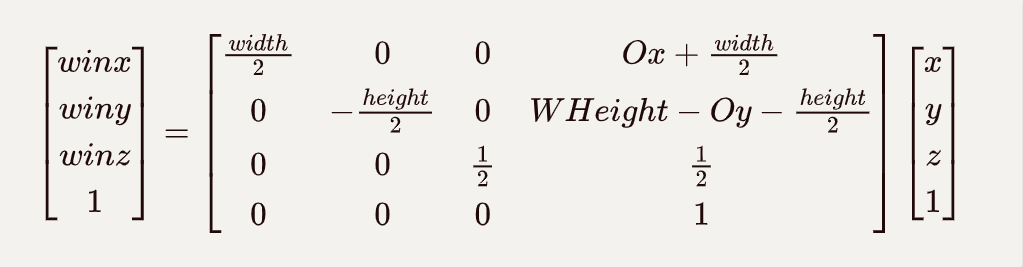
规范化正交变换将所有点的坐标转换为中心在原点边长为2的正方形，内，接下来要做的就是转换为窗口坐标



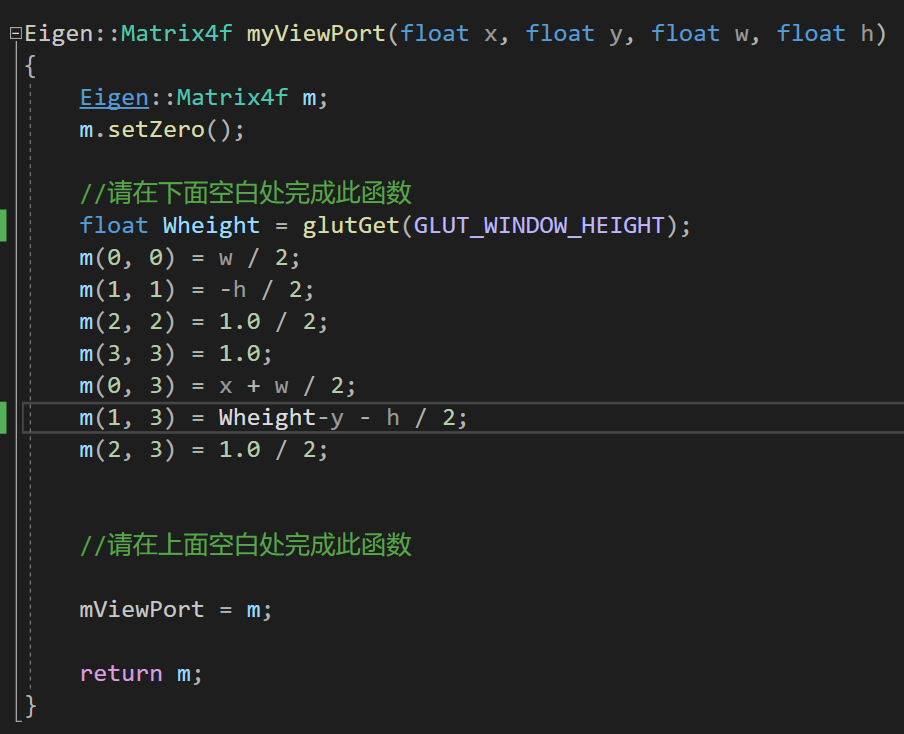
的四个参数中表示以窗口左下角为原点，视口左下角的坐标值，而这一组坐标值是以窗口左上角为原点的窗口坐标值，因此要注意坐标转换，这里的指的是窗口的高度



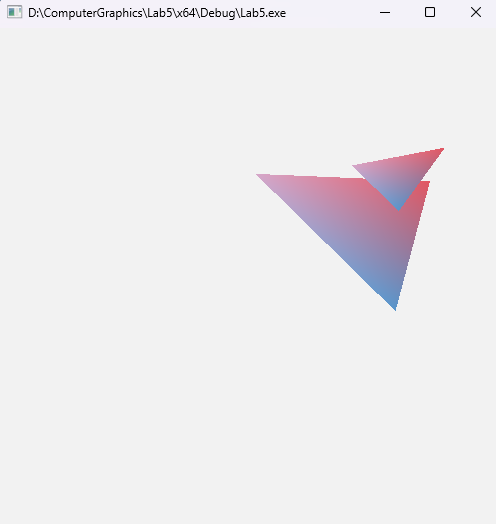
使用矩阵表示即为



## 4.2代码截图



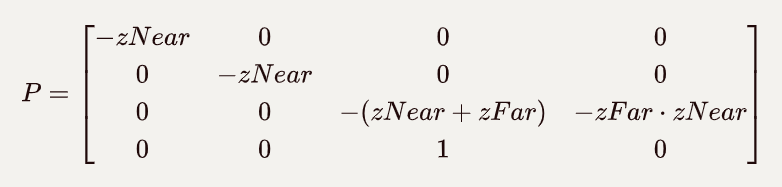
## 4.3运行效果

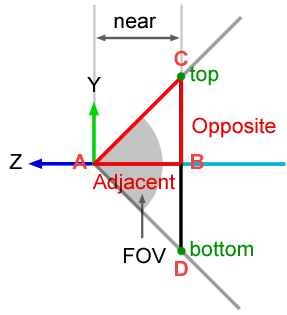


# Task5:myPerspective

## 5.1流程分析

同理于之前的myFrustum，先将视锥体转换为长方体的正交投影区域，变换的矩阵和之前的一样

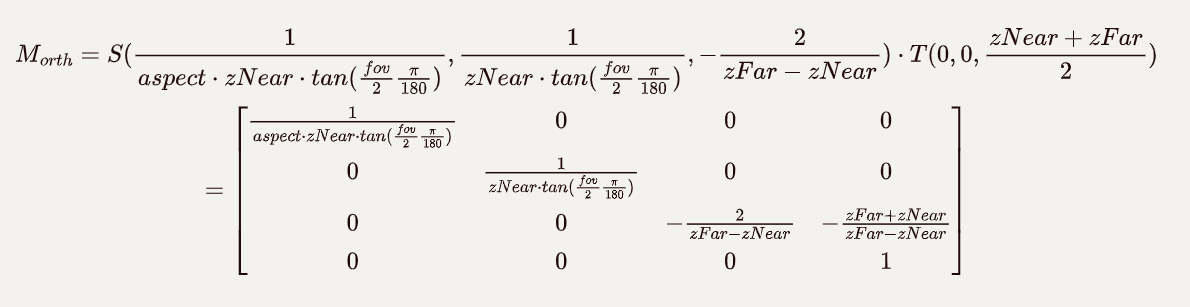




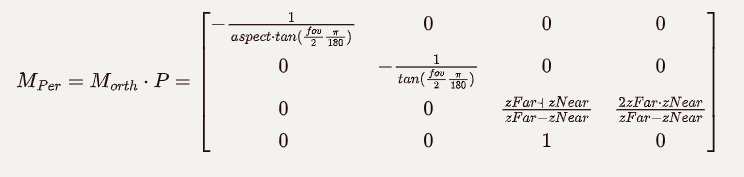
根据上图可知经过上述矩阵变换之后的得到的正交投影区域两个对角顶点的坐标为

*,*

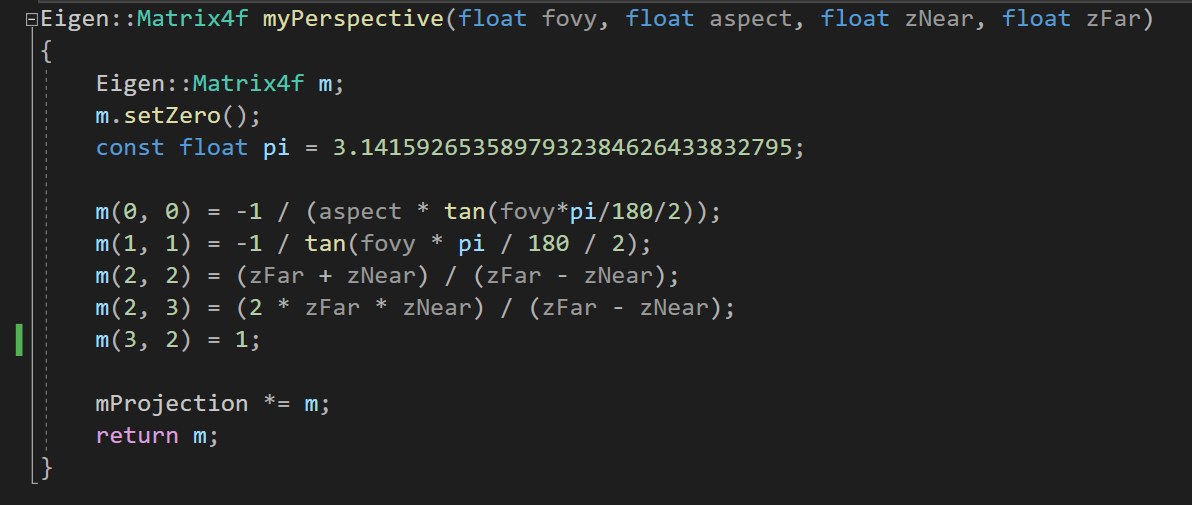
此时近裁剪平面的中心正好位于轴上，因此对应规范化正交投影矩阵为



因此得到最后的变换矩阵为



## 5.2代码截图



## 5.3运行效果

