代码中变量名的意义:

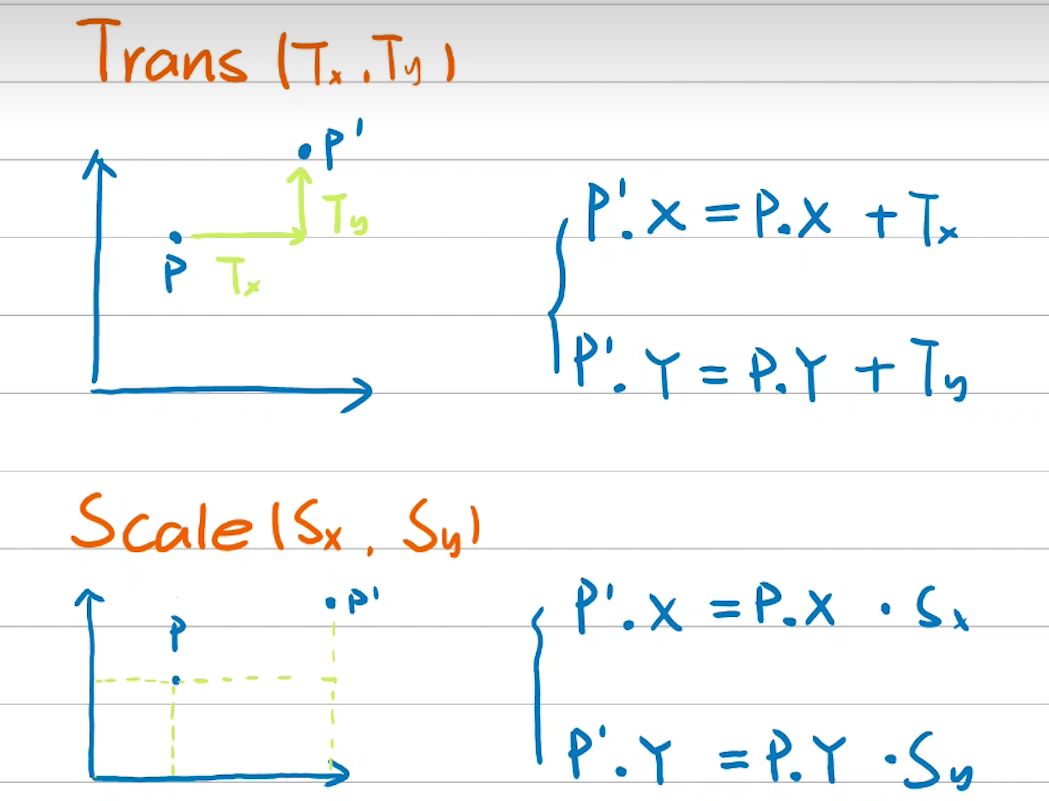
可能会用到的名词解释

[Override](https://learn.microsoft.com/zh-cn/dotnet/csharp/language-reference/keywords/override) [constexpr](https://learn.microsoft.com/zh-cn/cpp/cpp/constexpr-cpp?view=msvc-170) [template](https://learn.microsoft.com/zh-cn/cpp/cpp/templates-cpp?view=msvc-170)

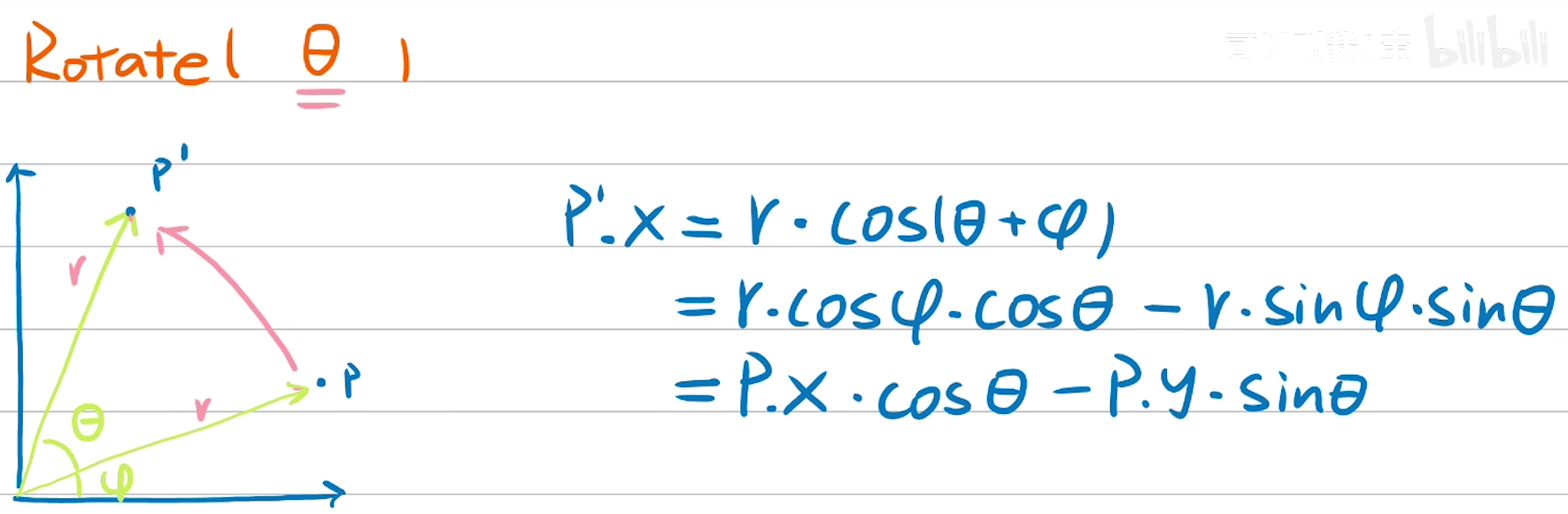
建议看了[OpenGL的坐标系统](https://learnopengl-cn.github.io/01%20Getting%20started/08%20Coordinate%20Systems/" \l "_2)后再往下看

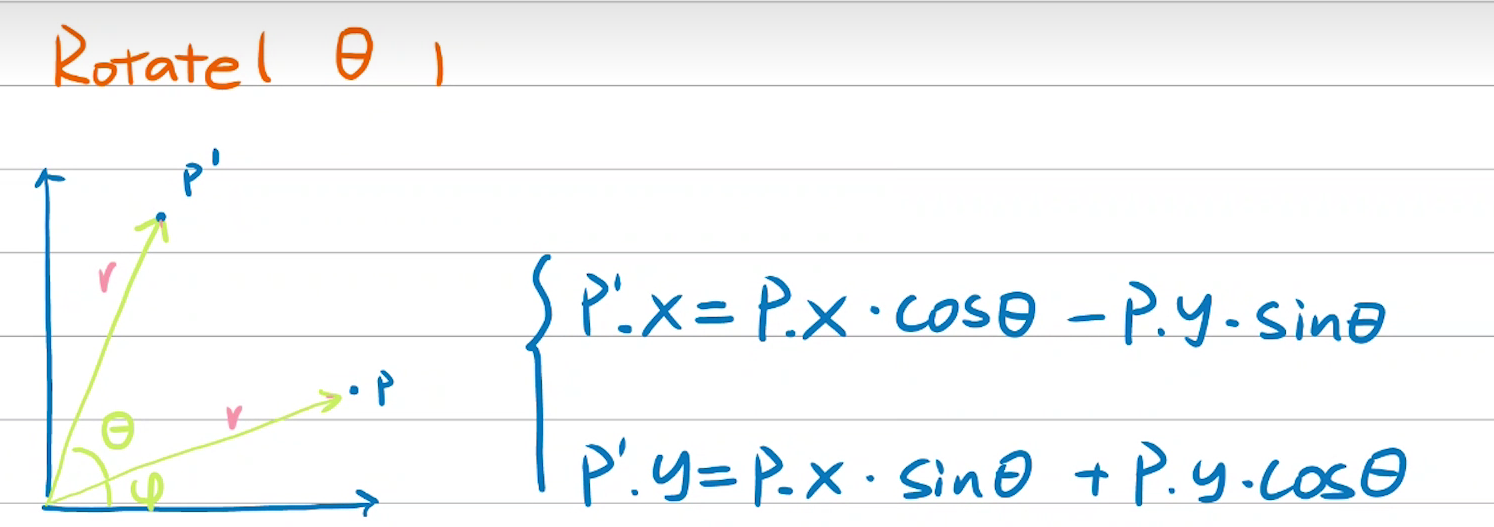
四.矩形变换

点到点:平移,拉伸,旋转



旋转解析:

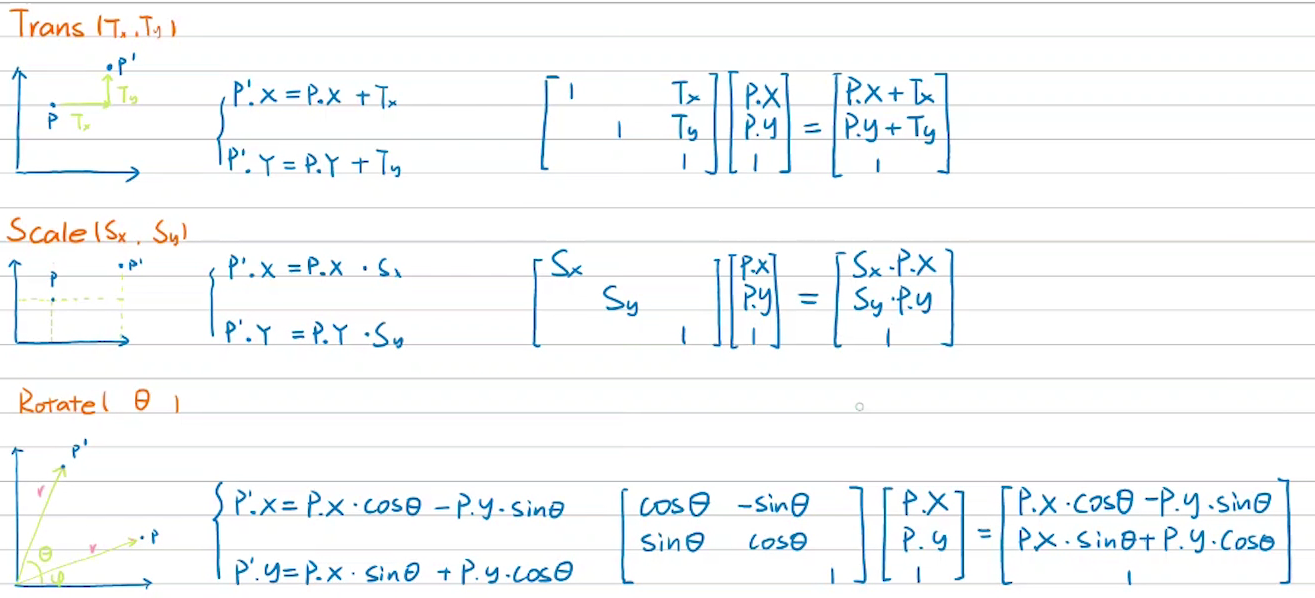




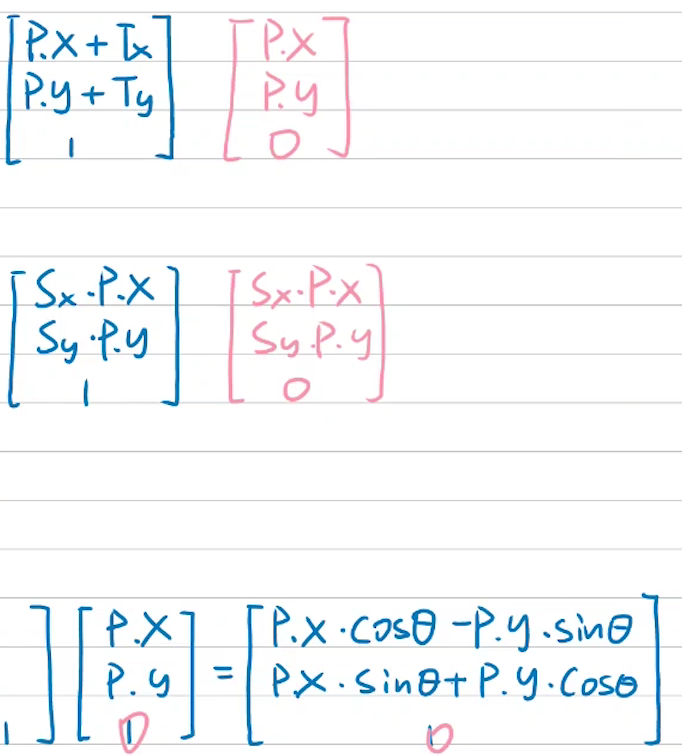
从结果来看,虽然引入了φ,但是结果式子和φ无关

虽然三种方法都很精简,但是有没有一种方法能将其统一起来?用一种表达形式来表示?

于是引入矩阵(Matrix)



那么为什么在每个矩阵的最后一行加上一个”1”,而不是”0”?



Ps:(那两个蓝色的’1’没写上’0’;前面蓝色的是p’的矩阵表示,后面粉色的是通过将’1’变为’0’后,通过计算得出的p’的矩阵表示)

观察结果得出:

拉伸和旋转并不受影响,但是平移受到了影响,发现得出的结果和p相同,从几何意义上看,意为p点没有发生变化,但将其看成空间中的起始点为原点的一个矢量,但平移本身对矢量来说并没有什么变化,

所以为了统一,并得出结论:

当最后一行为’1’时,矩阵可以表示为空间中的点;

当最后一行为’0’时,矩阵可以表示为空间中的矢量.

矩阵满足结合律:

几何意义:通过结合律,可以将一系列的几何变换矩阵运算为一个矩阵

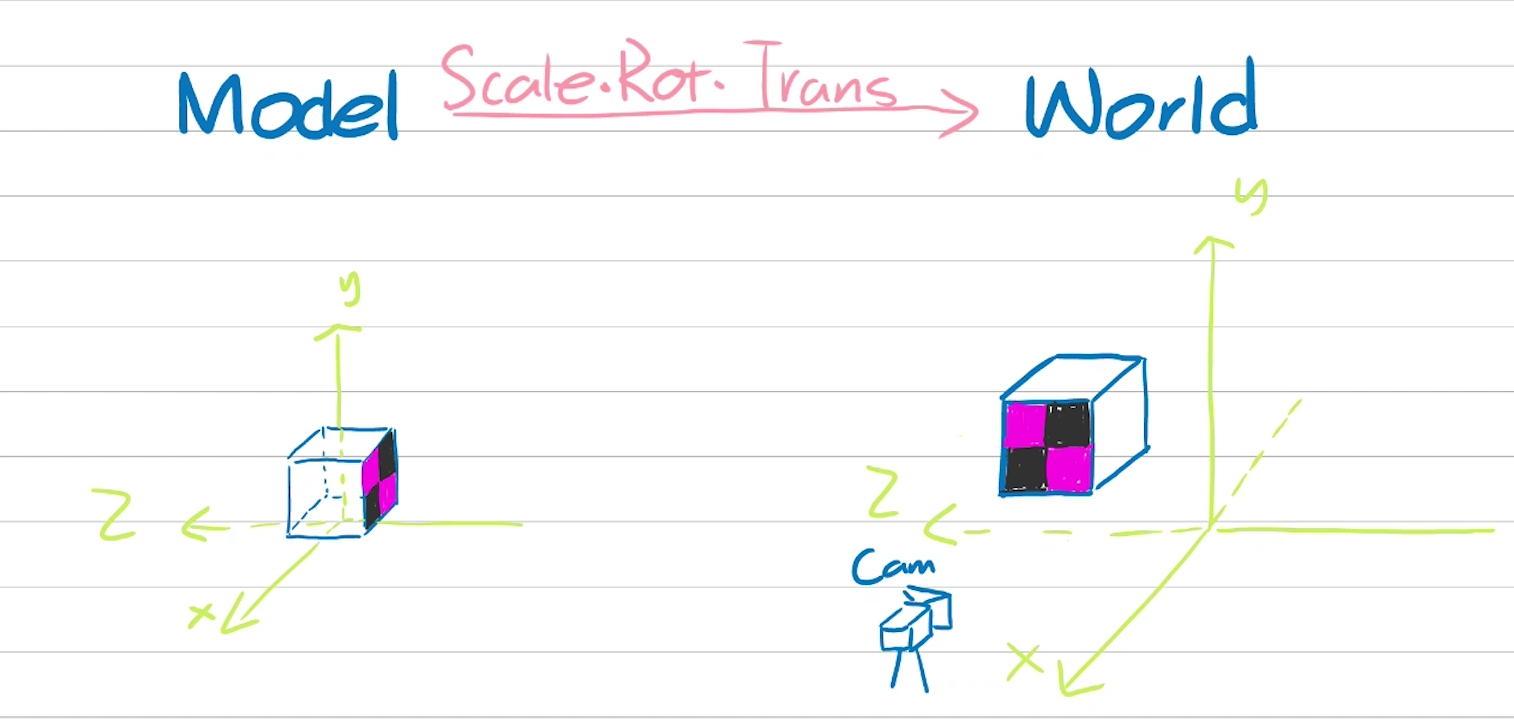
但是矩阵不满足交换律:

几何意义:例如,一个矢量先平移再绕原点旋转,和先然原点旋转再进行平移,大概率的结果是不一样的

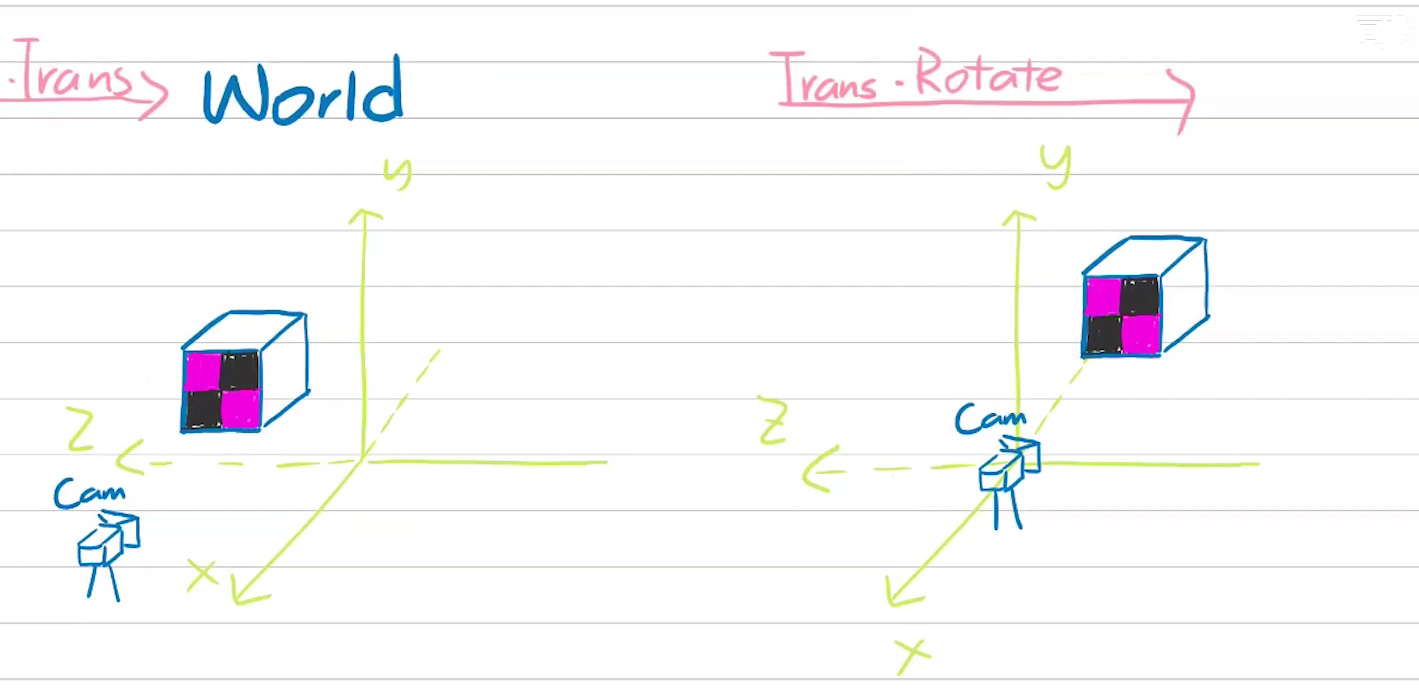
所以遇到矩阵相乘,通常会从右边矩阵开始,一个一个按矩阵顺序向左相乘(针对于列矩阵,行矩阵是向右)

当一个模型素材需要使用时,会经过三个空间:model,world,view

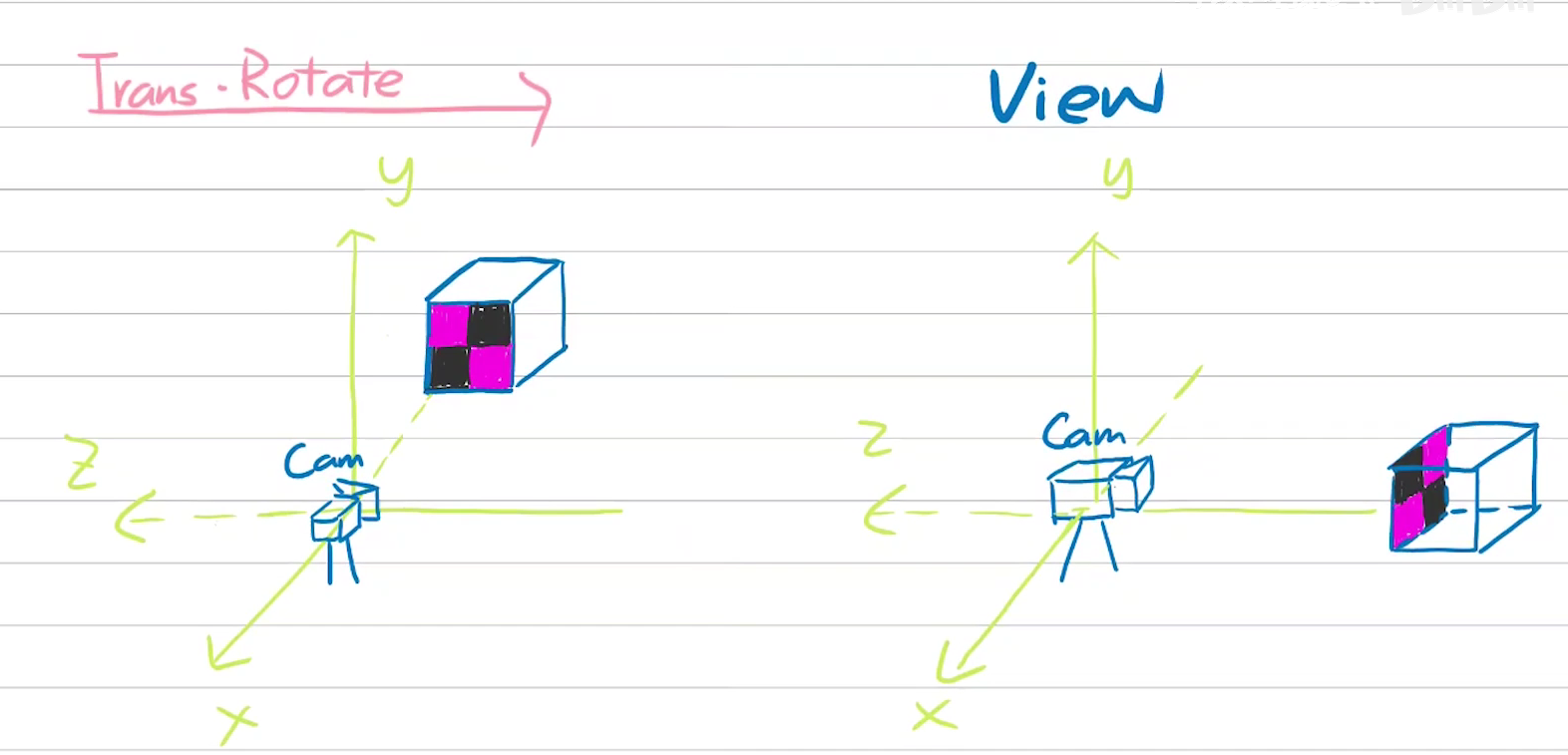
当模型被使用时,可能大小,朝向,方位等均不符合你的使用需求,所以需要经过拉伸,旋转,平移来调整模型,并在world空间(如Unity)中展示,此时便会多出一个实体camera(摄像机)与模型绑定



后将模型平移到世界的中心



然后通过旋转使得cam的朝向为-Z轴,右侧为X轴,上部为Y轴

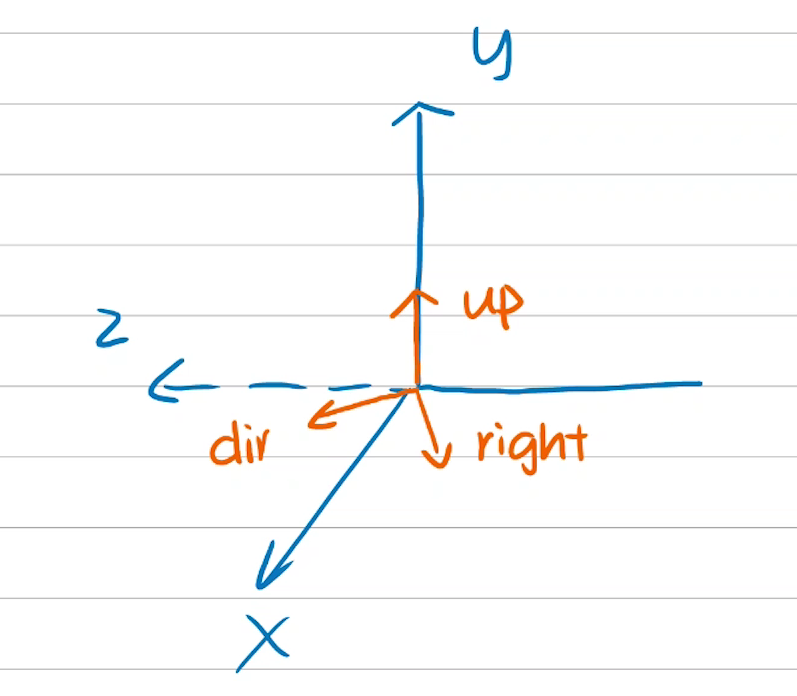


从model到world的模型参数都是我们自己定义的,所以其中的数据都知道,则拉伸,旋转,平移等操作都可以实现,最大的问题是从world到view的rotate矩阵该如何实现

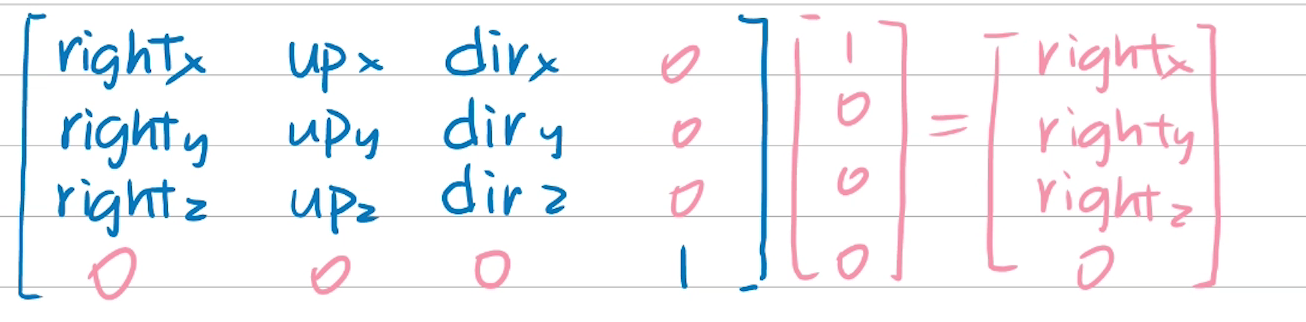
在此之前,我们要明白rotate矩阵的目的是什么?

先说结果,目的就是cam的方位与world的直角坐标系对齐(-Z轴为cam的正前方),当然与其绑定的模型也会一起转动

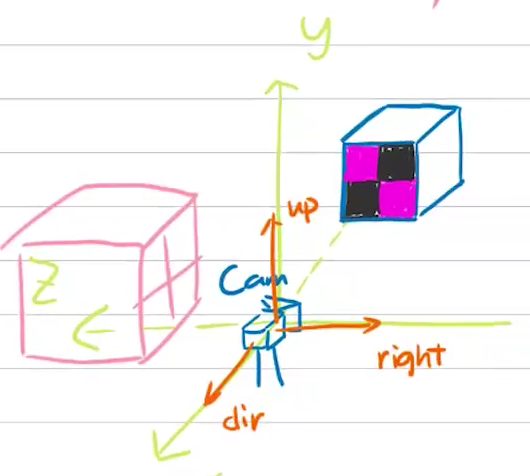
如下图,X,Y,Z轴为world的直角坐标系,up,dir,right为cam的单位向量,且两两垂直,其中dir为cam的正后方



当world的直角坐标系的单位矩阵与下面矩阵相乘,在几何上讲:相当于将X轴的朝向转向了right所对方向,

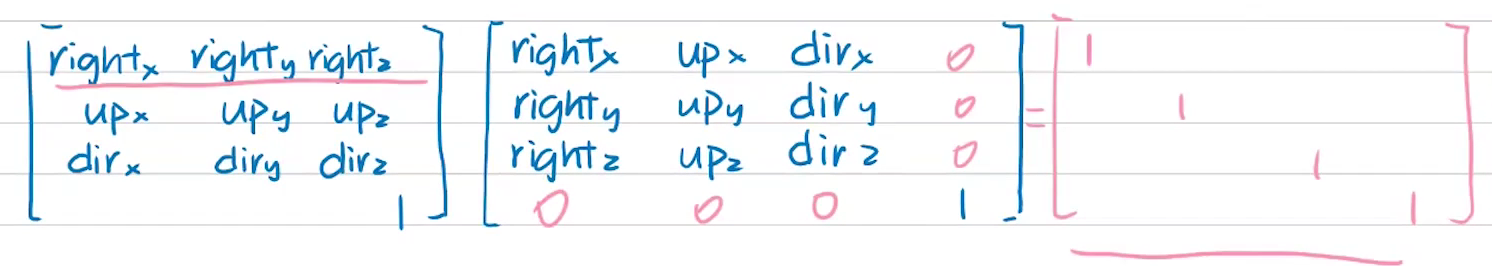


在空间中表示如下:

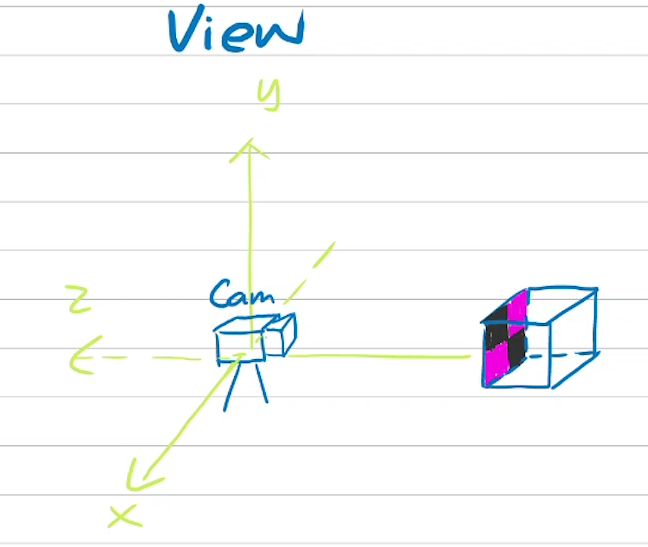


但这和我们想要的效果刚好转向相反,于是引入矩阵的转置

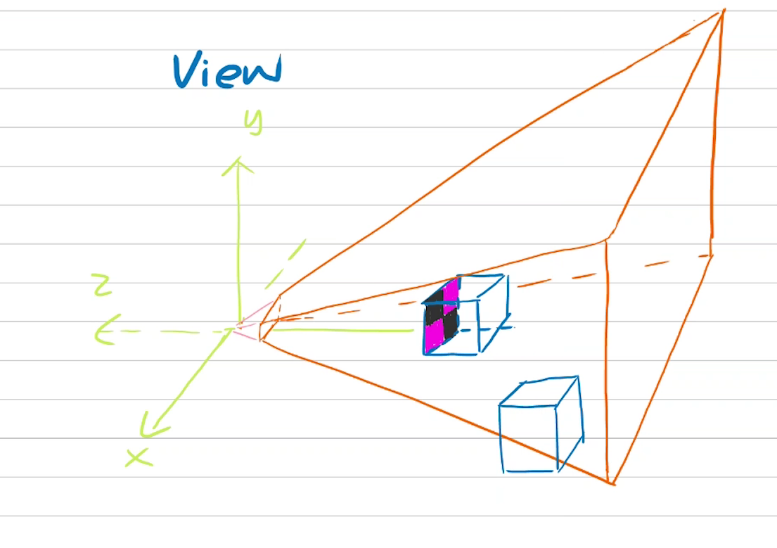
矩阵的转置就是该矩阵的逆,且一个矩阵乘它的转置等于单位矩阵



当world直角坐标系与以上提及的矩阵的转置相乘便实现了预想的效果

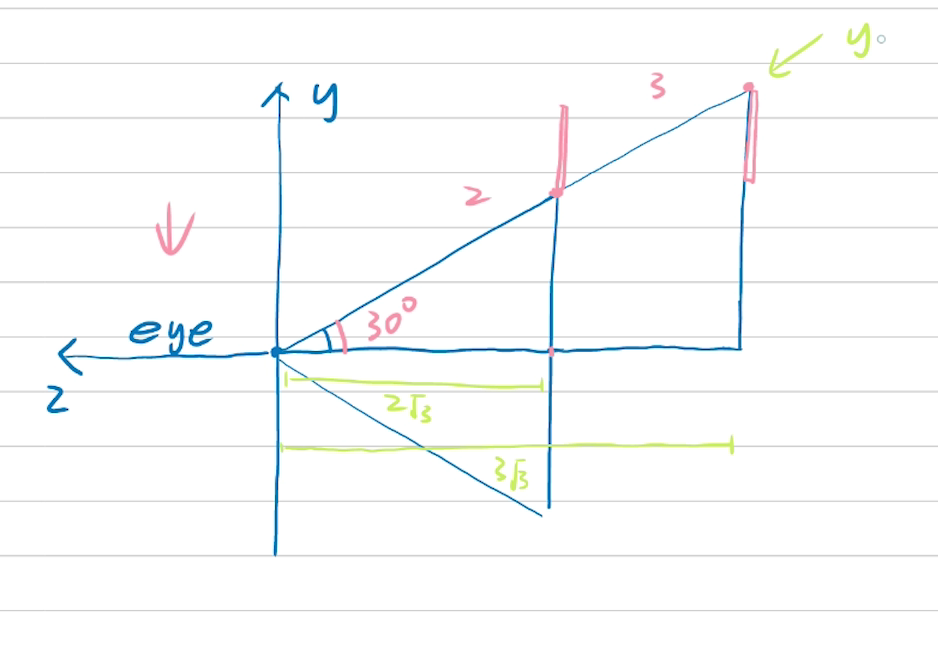


五.透视投影

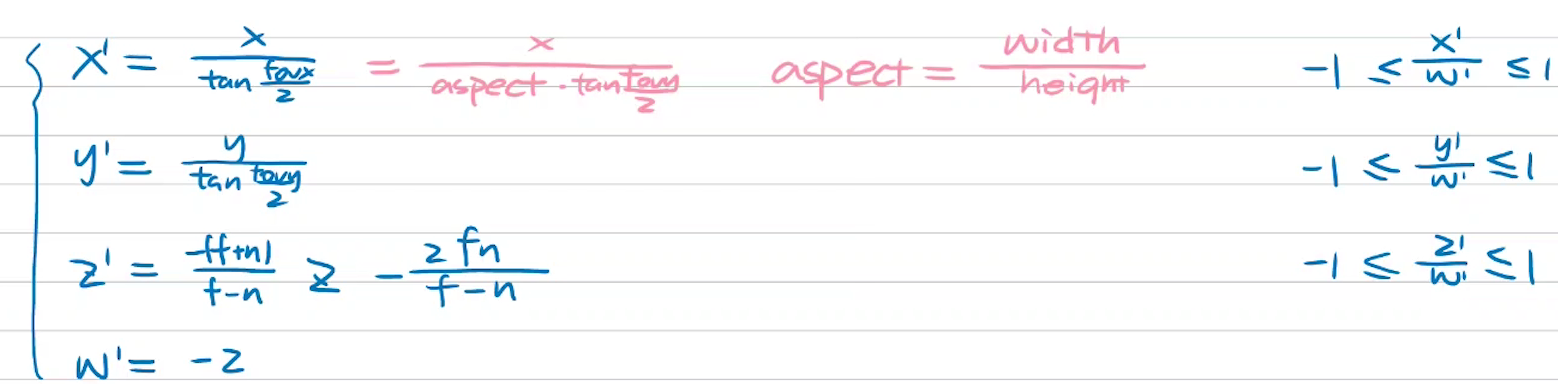
引入:

当空间中出现两个物体,且其中一个物体并不在camera视野当中时,便需要取舍(裁剪),但在此之前,首先要确认camera视野的范围

以y-z平面为例



懒得整了,上结论



以防看不清,写一个式子:x’=x/(tan(fovx/2)),fov代表cam在该平面上的视角

其中x,y,z代表点在view空间中的坐标,x’,y’,z’,w’代表点在裁剪空间中的值;

f(far)表示原点到远平面的距离,n(near)表示原点到近平面的距离,它们都有实际的物理意义,所以没有负值

当你算出x’,y’,z’,w’后,再带入后面的式子(x’/w’等),观察是否在范围内就可以判断它是否需要裁剪了

那得出这个结论有啥用呢?

在平时我们所观测的视野中,我们的眼睛会出现”近大远小”的现象,而这个结论就是为了解决物体在平面上模拟近大远小的问题,它保证了物体在不同视距的条件下的倍数问题

矢量的叉积:

复习:点积:**a · b =|a| |b|** cosθ



叉积: :**a × b =|a| |b|** sinθ



注:叉乘得出的结果是向量,点乘得出的结果是数值

