Bonus: Gimbal Lock

Krasjet

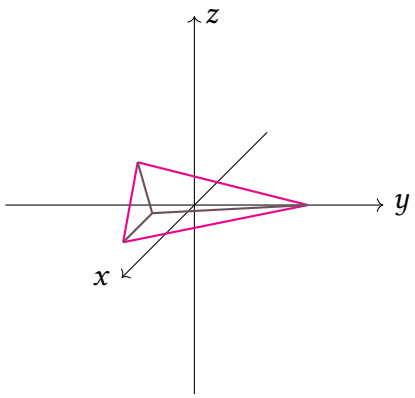
（这个 Bonus 章节是由我很久之前的笔记修改而成，所以有些概念没有从头开始介绍，但对理解应该不会造成太大的困难．如果你想了解的是四元数的相关知识，请点击这里）

在这里，我会简单介绍一下使用欧拉角表示 3D 朝向 (Orientation) 或者旋转时 Gimbal Lock（通常译为**万向锁**或者**万向节死锁**）的产生原因．虽然这方面的资料很多，但是某些解释可能会让人比较费解（比如 Wikipedia 上那个球型 Gimbal 的动画），有时候直接从数学上来理解反而会更直观一点．和四元数的教程一样，我们在这里使用的也是右手坐标系．

0.1 欧拉角

在介绍 Gimbal Lock 之前，我们需要先了解朝向和旋转的欧拉角表示方法．因为欧拉角的资料有很多，我不会从它的最基础讲起．如果你不了解什么是欧拉角，在继续阅读之前可以先读一下欧拉角的 Wikipedia.

首先，假设我们有一个物体，我们使用三个互相正交的坐标轴对它建立一个坐标系：



现在，我们的目标就是能改变这个物体的朝向．有一点需要注意的是，**朝向** (Orientation) 和**方向** (Direction) 虽然看起来很像，但是它们是完全不同的两个概念．朝向并不是使用一个向量定义的，我们通常会将朝向定义为将某一个「正朝向」**旋转**

至当前朝向所进行的**变换**，所以当你思考朝向的时候，你需要想到的其实是一个旋转（想象一下，我们需要对物体的每一个顶点都施加一个旋转的变换，将它变换到当前的朝向）.

在很多年以前，欧拉就证明了，3D 空间中的任意一个旋转都可以拆分成为沿着物体自身三个正交坐标轴的旋转（注意，这里指的是沿着物体**自己的**坐标轴旋转）. 而欧拉角就规定了这三次旋转的角度，我们分别称他们为 yaw、$pitch$ 和 $roll$（具体什么含义请去查看 Wikipedia）. 这也就是说，3D 空间中任意的旋转都可以由三个旋转矩阵相乘的方式得到：

$$E(yaw, pitch, roll) = R_z(yaw)R_y(roll)R_x(pitch)$$

其中 R_x、R_y、R_z 是沿着 x、y、z 轴旋转的旋转矩阵（它们的推导很简单，线性代数的课本上应该都有）：

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

因为我们将一个旋转拆分成了三个 3D 旋转矩阵的复合，而且 3D 旋转矩阵的相乘一般是不可交换的（即一般 $R_x(\theta)R_z(\phi) \neq R_z(\phi)R_x(\theta)$），所以进行这三次旋转的**顺序**非常重要. 在上面的例子中，我们是按照 x-y-z 轴的顺序进行旋转的. 当然，这并不是唯一的选择，如果你想用 y-x-z 的顺序进行旋转也是完全可以的.

然而，要注意的是，一般情况下，我们一般只会选择其中**固定的一个**顺序来编码旋转或者朝向，而这正是导致 Gimbal Lock 的原因.

0.2 Gimbal Lock

还记得，我们将 3D 空间中任意一个旋转写成了三个矩阵相乘的形式，在下面的讨论中，我们将固定使用这一套旋转顺序：

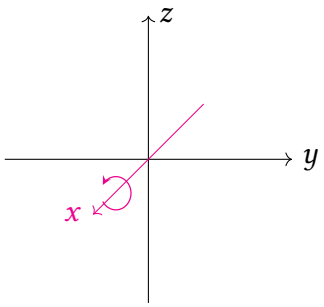
$$E(yaw, pitch, roll) = R_z(yaw)R_y(roll)R_x(pitch)$$

其中，每一次旋转变换就代表着有一个 Gimbal（通常译为**万向节**或者**平衡环架**）. 所以，在这里我们一共有**固定的**三个 Gimbal. 我们可以改变每次旋转的角度，但是不论怎么变化角度这个旋转的顺序都是不能改变的.

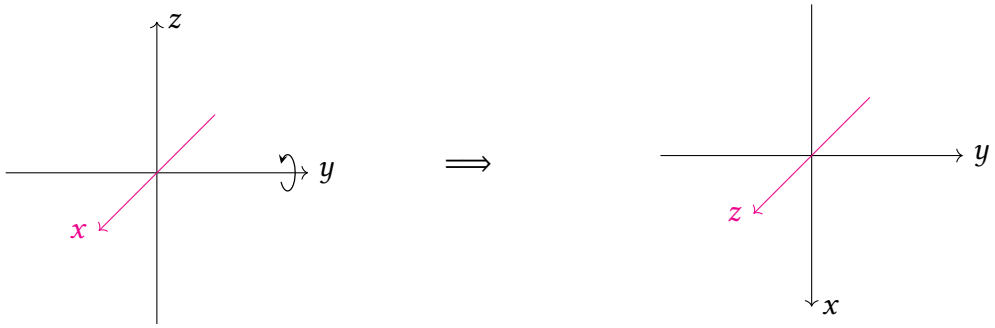
大部分情况下，这一套顺序都能够对**三个**不同的轴分别进行三次旋转. 然而，在某一些特殊情况下，它其中的某两个旋转变换可能变换的是同一个轴（这里指的不是物体自身的轴，而是外部或者世界的轴），这样就导致我们丧失了一个轴的自由度，从而导致 Gimbal Lock 的产生. 虽然看起来有点不可思议，但是下面我们会用实际的例子来说明.

假设我们使用 x-y-z 的顺序旋转任意一个点. 其中，我们沿 x 轴旋转任意度数，沿 y 轴旋转 $\pi/2$ 弧度，最后，我们再沿着 z 轴旋转任意的度数. 我们来仔细研究这个过程.

首先，我们沿着 x 轴旋转任意的度数：



接下来，我们沿 y 轴旋转 $\pi/2$ 弧度：



注意，经过这一次的变换之后，我们将 z 轴变换到**原来** x 轴方向，而 x 轴变到**原来**

$-z$ 的方向.

这个变换执行完毕后，我们仅仅只剩下一个 z 轴的旋转矩阵了．然而，**当前的** z 轴与**原来的** x 轴重合，也就是说，最后 z 轴的旋转与 x 轴的旋转其实操纵的是同一个轴（同样，这里指的是外部的旋转轴，在图中用红色标出）．三次旋转变换仅仅覆盖了两个外部轴的旋转，一个自由度就这样丢失了，这也就导致了 Gimbal Lock 的现象.

现在，如果我们想要操纵第三个轴的旋转（也就是在 y 轴变换之前 z 轴所表示的竖直方向），就需要对 y 轴变换**之后的** x 轴进行旋转．然而，唯一的 x 轴 Gimbal（或者变换）优先级在最外部，我们**当时**操纵它的时候，它操纵的外部旋转轴还与现在的 z 方向相同，并不能补回这个自由度.

这个变换用公式来理解的话，则是这样（你可以自己来代入验证一下）：

$$
\begin{aligned}
E(\alpha, \frac{\pi}{2}, \beta) &= R_z(\beta)R_y(\frac{\pi}{2})R_x(\alpha) \\
&= \begin{bmatrix} 0 & \cos(\beta)\cdot\sin(\alpha) - \cos(\alpha)\cdot\sin(\beta) & \sin(\alpha)\cdot\sin(\beta) + \cos(\alpha)\cdot\cos(\beta) \\ 0 & \sin(\alpha)\cdot\sin(\beta) + \cos(\alpha)\cdot\cos(\beta) & \cos(\alpha)\cdot\sin(\beta) - \cos(\beta)\cdot\sin(\alpha) \\ -1 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & \sin(\alpha-\beta) & \cos(\alpha-\beta) \\ 0 & \cos(\alpha-\beta) & -\sin(\alpha-\beta) \\ -1 & 0 & 0 \end{bmatrix} \\
&= R_y(\frac{\pi}{2})R_x(\alpha-\beta)
\end{aligned}
$$

利用一些三角恒等式，我将原本由三个旋转矩阵所组成的变换化简成了两个变换矩阵．即便我们分别对 x-y-z 三轴进行了旋转，实际上这个矩阵仅仅旋转了 x-y 两轴，它并没有对（初始的）z 轴进行变换．$R_z(\beta)$ 与 $R_x(\alpha)$ 这两个变换被合并为单独的一个 $R_x(\alpha-\beta)$ 变换（因为化简完之后变换顺序不一样了，严格来说并不是合并，只不过是能够使用一步来完成）.

注意，我们在这里化简的并不是单独的一个变换，而是**一系列**变换．因为它对任意 α 和 β 角都是成立的．也就是说，一旦 y 轴上的变换角将这两个旋转轴对齐，我们就没有任何办法对最初的 z 轴进行旋转了．无论 x 轴与 z 轴的旋转角是多少，变换都会丧失一个自由度.

当然这也不是说现在没有办法对最初的 z 轴进行变换．我们只需要在最内部添加一个 x 轴 Gimbal，或者说在变换序列的最后再添加一个 x 轴的旋转变换，就能解决问

题了（注意下面经过 R_y 变换之后 R_x 操纵的其实是原本的 z 轴）：

$$\begin{aligned} E(\alpha, \frac{\pi}{2}, \beta, \gamma) &= R_x(\gamma)R_z(\beta)R_y(\frac{\pi}{2})R_x(\alpha) \\ &= R_x(\gamma)R_y(\frac{\pi}{2})R_x(\alpha - \beta) \end{aligned}$$

然而，我们之前说过，我们选择的是固定的三个旋转顺序，实际上并没有这个第四个 Gimbal，所以没办法解决这个问题．虽然你在当初选择顺序的时候也可以选择采用四个 Gimbal 的设计，但是这仍不能完全解决问题，因为除了 y 轴之外其它轴的旋转仍然可能将其中的两个轴或者三个轴对齐，只不过现在 Gimbal Lock 产生的几率可能会变小而已．Gimbal Lock 问题的核心还是在于我们采用了固定的旋转顺序.