**NASA Technical Report:Euler Angles, Quaternions and Transformation Matrices**

Author: 练孙鸿

**0 引言**

因为最近要给Noise3D集成一个RigidTransform之类的class，动机是做SH Lighting Preview Utility的时候想要用鼠标旋转球体，结果发现如果只能控制Euler Angle的话实现delta rotation很麻烦，干脆就把四元数也加进来然后实现一个RigidTransform的类，把几种旋转表示的转换都封装一下。这里考虑了几种旋转表示方法：Euler Angle，Quaternion，Rotation Matrix，（还有个擦边的Lookat转Euler Angle）。网上很多blog的旋转表示转换方法都是基于XYZ的欧拉角顺规的，这就很不清真了，因为D3D是y轴向上的左手系而且顺规也不一样，那么很多公式就不能直接copy了。这篇笔记玩意主要参考了一篇NASA 1977的Technical Report[1]，看起来好像给出了12种欧拉角顺规的euler-to-matrix conversion，以及与quaternion的转换等。所以我觉得可以看一手，并且记下一点与D3D相关的旋转转换公式。

**1 欧拉角(Euler Angle)与旋转矩阵(Rotation Matrix)**

D3D和OpenGL不同，用的坐标系是Y轴竖直向上的左手系，所以欧拉角Yaw-Pitch-Roll的顺规是跟广大blog、OpenGL不一样的，那么博客上、甚至维基百科[2]上的各种基于右手系xyz顺规(分别对应roll, pitch,yaw)的看起来就不太能随随便便直接用了。虽然有一部分转换是DirectXMath支持的，例如XMQuaternionRotation()，XMMatrixRotationQuaternion()等。但是为了不要用得稀里糊涂，以及实现一些接口不提供的东西，我还是决定自己实现几种Rotation Representation的conversion吧…

首先欧拉角旋转序列(Euler Angle Rotational Sequence)一共有12种顺规，6种绕三条轴的旋转(XYZ,XZY,YXZ,YZX,ZXY,ZYX)，另外6种绕两条轴的旋转(XYX,YXY,XZX,ZXZ,YZY,ZYZ)。如果相邻两次旋转是绕同一条轴，例如XXY，那么其实可以坍缩成XY。那么只绕一条轴旋转就根本不够自由度就不需要说了。所以一共是12种旋转顺规（可以表示所有旋转的集合），其中基于D3D的Noise3D采用的是Z(roll)X(pitch)Y(Yaw)顺规，写成列向量矩阵时应该是：（与维基百科Euler Angles[3]给出的结果一致）

其中是绕y轴旋转的yaw，是绕x轴旋转的pitch，是绕z旋转的roll，即：

注意两种不同的旋转顺规想要把同一物体旋转到同一姿态，那么他们的欧拉角Yaw-Pitch-Roll角度值是不一样的。

从欧拉角构建旋转矩阵就很简单啦，按照顺规把三个Elemental Rotation Matrix连乘起来就好了。但是由于(2018.9.18)现在要实现用鼠标拖拽一个sphere进行delta Rotation的功能，就是在当前姿态上再加上一个旋转，然后还要update其他的旋转表示。直接给Euler Angle加上一个delta值是不能得到正确的结果的，一个反例就是，如果物体的正面面对着用户时，鼠标向上拖，物体的pitch增加就好了。但是如果物体的背面面对着用户，鼠标向上拖，pitch应该是要减少的。总之得用其他的旋转表示来做delta Rotation比较合适，欧拉角不适合做delta rotation。那么四元数或者旋转矩阵其实也是可以的，这两种形式可以很方便地加上一个delta rotation，而且delta rotation也都可以很方便地由Axis-Angle的方法构造出来。给旋转矩阵乘多一个delta rotation matrix然后再提取出欧拉角，就可以实现各个rotation representation的同步。所以这一节接下来再讨论一下从旋转矩阵提取欧拉角的方法。

参考NASA technical Report[1]的Appedix-A6和[5]，我们可以用旋转矩阵元素的相乘、相除、反三角函数等操作去提取出欧拉角。[5]给出了从XYZ顺规提取欧拉角的方法、步骤、思路，[1]则给出了全部12种顺规的欧拉角提取公式，但是没有给一些细节注意事项。所以总结一下，根据[1]、[5]、[7]《Real Time Rendering 3rd Edition》4.2.2和自己的推导，Z(roll)X(pitch)Y(Yaw)顺规的欧拉角提取公式（[1]原文下标似乎有点小问题）：

* Y axis yaw angle:
* X axis pitch angle:
* Z axis roll angle:

要注意一个很关键的问题，注意到矩阵的每一个元素都是pitch angle 的函数…所以如果当公式无意义时，也就是，这时候其他的欧拉角Extraction formula都跟着无意义了….在ZXY顺规下，pitch angle 恰好就是Gimbal Lock的位置。在Gimbal Lock的时候，旋转矩阵会退化为：

所以在的gimbal lock corner case时，甚至连函数都无济于事了，因为也是无意义的。这个时候，我们不按《Real Time Rendering》[7]的比较粗暴的解决方法来，而按[5]的思路和自己的推算，gimbal lock corner case应该分两种情况处理：

* 时，，

其中和可以先随便固定一个，计算出另外一个。

* 时，，

其中和可以先随便固定一个，计算出另外一个。

**2 四元数(Quaternion)与旋转矩阵**

众所周知的是，欧拉角和旋转矩阵表示法是有万向锁(Gimbal Lock)的问题的。幸好我们有四元数这种东西。

我们可以用把一个旋转写成四元数的形式。如果我们设是旋转角，单位旋转轴为，由轴角Axis-Angle方式构造四元数方式如下：

注意：

这里按右手定则来定义旋转方向（笔者疑问：左右手系和左右手法则有没有什么影响呢？），那么给定一个四元数，可以构造旋转矩阵[1][4][8]（参考Wikipedia -Rotation Matrix: Conversion- Quaternion）：

用四元数的分量直接构造旋转矩阵是非常高效的，因为只有乘和加减法，而不需要三角函数。

然后从旋转矩阵提取四元数，也是可以像提取欧拉角那样，用正向构造的矩阵表达式凑出来。参考[8]《RTR》，我们观察一下的元素，可以发现：

意思是我们只需要再凑出个，那么四元数的四个分量都可以求出来了。于是我们又发现了一个恒等式：

其中是矩阵的迹(trace)。于是四元数就可以从旋转矩阵里面提取出来了：

**3 轴-角(Axis-Angle)**

轴-角顾名思义就是绕某条顶轴旋转一定角度，从这个意义上看，它构造四元数是非常和谐的，毕竟几何意义有一点点类似，轴角构造四元数具体构造在Chap2有提到，绕单位轴旋转的四元数是：

因为已经有了欧拉角/旋转矩阵/四元数来表示当前姿态了，所以Axis Angle我就打算用来构造其他表示的delta Rotation。

首先是Axis Angle转Rotation Matrix。可以使用罗德里格斯公式Rodrigues Rotation Formula：

**引用**

[1]Henderson, D.M.. Euler angles, quaternions, and transformation matrices for space shuttle analysis[C]//NASA, Jun 09, 1977.

[2] https://en.wikipedia.org/wiki/Conversion\_between\_quaternions\_and\_Euler\_angles#Tait%E2%80%93Bryan\_angles

[3] <https://en.wikipedia.org/wiki/Euler_angles>

[4] <https://en.wikipedia.org/wiki/Rotation_matrix>

[5] Slabaugh G G. Computing Euler angles from a rotation matrix[J]. 1999.

[6] Mike Day, Converting a Rotation Matrix to a Quaternion. <https://d3cw3dd2w32x2b.cloudfront.net/wp-content/uploads/2015/01/matrix-to-quat.pdf>

[7] Tomas K.M. , Eric H., Naty H.. Real Time Rendering 3rd Edition , p68-p69, 2008.

[8] Tomas K.M. , Eric H., Naty H.. Real Time Rendering 3rd Edition , p76-p77, 2008.