Computer Science and Application 计算机科学与应用, 2019, 9(5), 890-895
Published Online May 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.95100

Hans汉斯

Study on the Relationship between Euler Angle and Quaternion in Unity 3D

Chaofan Su, Renchun Guo

Shenyang University of Chemical Technology, Shenyang Liaoning
Email: 694001066@qq.com

Received: Apr. 30th, 2019; accepted: May 10th, 2019; published: May 17th, 2019

Abstract

In unity 3D game engine, the orientation of objects is stored in the form of quaternions, but the Euler angle is presented to the user on the interface. The user-set Euler angle is converted to quaternion first and finally to a new Euler angle, and each Euler angle can be expressed in two quaternions, which makes the problem very complicated. In this paper, two conventions of Euler angle and mathematical expressions of Euler angle and quaternion are given. By analyzing and studying the application of Euler angle and quaternion in unity 3D game engine, the transformation relationship between them is explained in detail.

Keywords

Unity 3D Game Engine, Euler Angle, Quaternion

Unity 3D中欧拉角与四元数关系的研究

苏超凡，郭仁春

沈阳化工大学，辽宁 沈阳
Email: 694001066@qq.com

收稿日期：2019年4月30日；录用日期：2019年5月10日；发布日期：2019年5月17日

摘　要

在Unity 3D游戏引擎中，物体的方位是用四元数的形式来存储的，但界面上给用户呈现的却是欧拉角。用户设置的欧拉角会先转化为四元数，最终转化为新的欧拉角，而每一个欧拉角都可以用两个四元数来表示，这样就会使问题变得非常复杂。本文给出了欧拉角的两种约定和欧拉角与四元数的数学表达式，通过对欧拉角和四元数在Unity3D游戏引擎中的实例应用的分析研究，将两者之间的转换关系做出了详

文章引用: 苏超凡, 郭仁春. Unity 3D 中欧拉角与四元数关系的研究[J]. 计算机科学与应用, 2019, 9(5): 890-895.
DOI: 10.12677/csa.2019.95100

细的说明。

关键词

Unity 3D游戏引擎，欧拉角，四元数

1. 引言

在 Unity3D 游戏引擎中，用方位来表征物体的旋转姿态，这是一个状态变量，方位可以采用旋转矩阵、欧拉角和四元数三种方法来描述[1]。在引擎中采用四元数来定义方位，但四元数不直观，设置困难，因此一般给用户呈现的是欧拉角。由于欧拉角可以任意设置，理论上同一个方位可以用无数个欧拉角来表示，也可以用两个四元数来表示。但 Unity 3D 引擎只用一个四元数来定义方位，这就造成了四元数与欧拉角一对多(1:n)的问题。采用四元数定义物体方位又带来另外一个问题，就是同一个方位可以用两个四元数来表示，这又形成了 1:2 的问题。因此同一个方位对应两个四元数和无数个欧拉角。用户为物体设置好的欧拉角会转化为系统内部的四元数，由系统内部的四元数又可以转化成一个新的欧拉角。因此转换过程是：用户设置欧拉角→系统内部四元数→新欧拉角。

虽然是同一个方位，但用户设置的欧拉角和系统生成的欧拉角在数据表现上并不相同，这会给学习 Unity 3D 游戏引擎带来巨大障碍，目前的各种文献都没有对此问题给出详细的介绍。本文从欧拉角和四元数的数学定义和转换公式入手，详细阐明其在 Unity 3D 中的基本原理，并通过具体实例讲解了两者的应用过程。

2. 欧拉角和四元数的概念

2.1. 欧拉角的概念

欧拉角是用来定义方位的一种表示，它有两种约定分别是“heading-pitch-bank”约定和“roll -pitch-yaw”约定。“heading-pitch-bank”约定是首先把物体坐标系和惯性坐标系对齐，物体旋转顺序为：$y \to x \to z$。heading 是绕竖轴方向旋转，在 Unity 3D 中 y 轴竖直向上，因此 heading 是绕物体坐标系的 y 轴旋转。pitch 是绕物体坐标系的 x 轴旋转，bank 是绕物体坐标系的 z 轴旋转，如图 1 所示。

“roll-pitch-yaw”约定只考虑惯性坐标系，旋转的顺序为“roll-pitch-yaw”，即：$z \to x \to y$，与 heading-pitch-bank ($y \to x \to z$)约定正好相反，如图 2 所示。

虽然 heading-pitch-bank ($y \to x \to z$)约定与 roll-pitch-yaw ($z \to x \to y$)约定相反，但这两种约定实质上是等价的。

2.2. 四元数的概念

四元数本身是一种超复数，用 4 个数来表示，形式为 $w + xi + yj + zk$，其中 w 为实部，x，y，z 为虚部，满足 $i^2 = j^2 = k^2 = ijk = -1$ [2]。其中 (x, y, z) 可以用一个三维向量来表示，即 $\boldsymbol{v} = (x, y, z)$，设 q 为四元数，可表示为 $q = (w, \boldsymbol{v})$。在 Unity 3D 中习惯将四元数的实部写在后面即 $q = xi + yj + zk + w$，或

$q=(\boldsymbol{v},w)$。

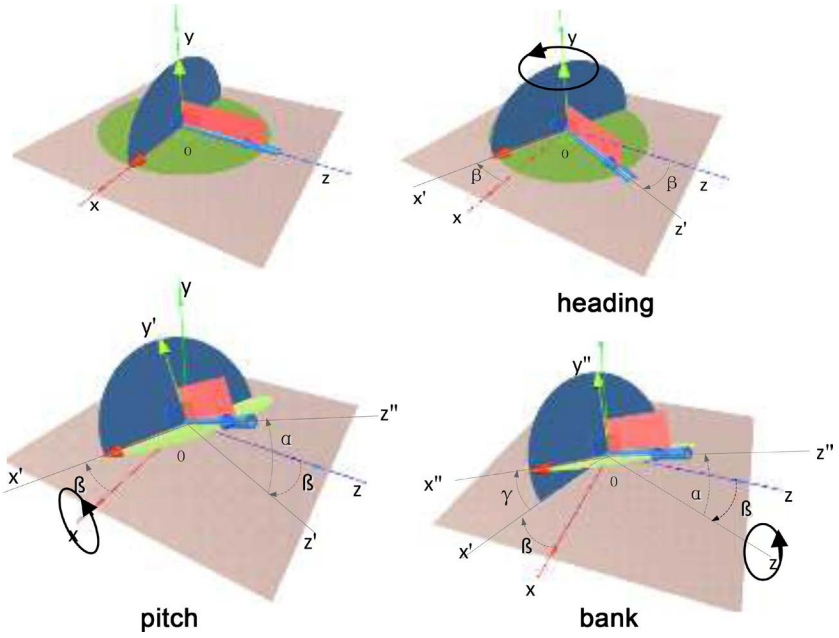

Figure 1. Heading-pitch-bank
图 1. 偏航角–俯仰角–滚转角

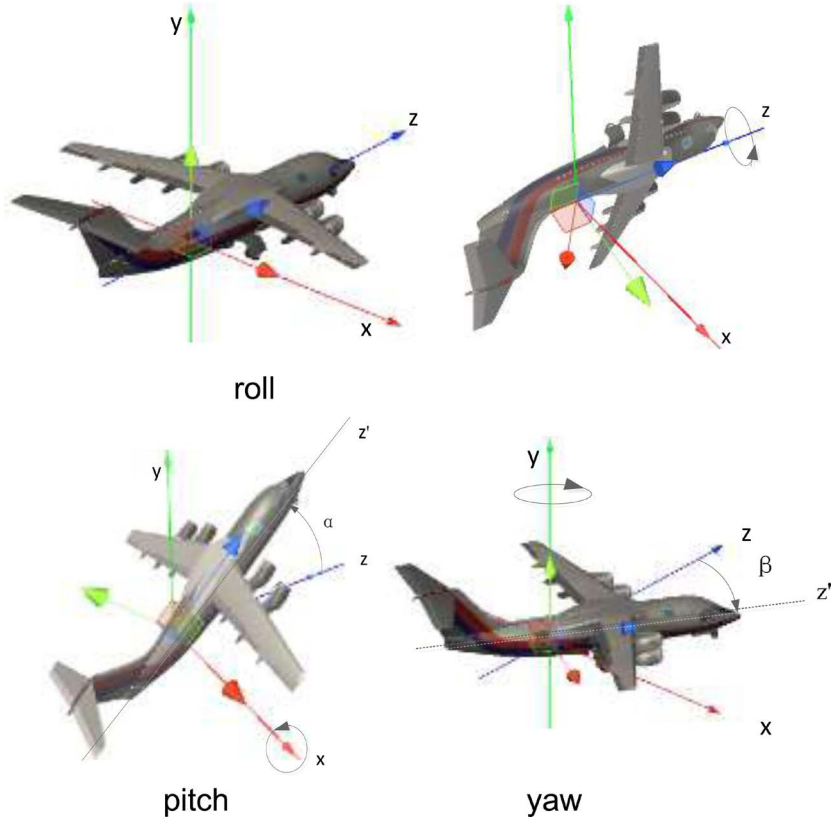

Figure 2. Roll-pitch-yaw
图 2. 滚转角–俯仰角–偏航角

3. 欧拉角与四元数的转换

3.1. 从欧拉角转换到四元数

物体某一方位可以用欧拉角 $(\theta_h,\theta_p,\theta_b)$ 来表示，也可以用一个四元数 $q=w+xi+yj+zk$ 来表示，该四元数 q 可以用 q_h,q_p,q_b 来表示。q_h,q_p,q_b 分别为绕轴 y，x，z 旋转的四元数。在此使用负旋转量。

$$q_h=\begin{bmatrix}\cos(\theta_h/2)\\ \begin{pmatrix}0\\ -\sin(\theta_h/2)\\ 0\end{pmatrix}\end{bmatrix},\ q_p=\begin{bmatrix}\cos(\theta_p/2)\\ \begin{pmatrix}-\sin(\theta_p/2)\\ 0\\ 0\end{pmatrix}\end{bmatrix},\ q_b=\begin{bmatrix}\cos(\theta_b/2)\\ \begin{pmatrix}0\\ 0\\ -\sin(\theta_b/2)\end{pmatrix}\end{bmatrix}$$

q_h,q_p,q_b，是三次旋转，最终需要形成一个四元数 q，将三者直接相乘即可，即 $q=q_hq_pq_b$，矩阵相乘的顺序采用“heading-pitch-bank”约定。

$$\text{则 } q=q_hq_pq_b=\begin{bmatrix}\cos(\theta_h/2)\cos(\theta_p/2)\cos(\theta_b/2)+\sin(\theta_h/2)\sin(\theta_p/2)\sin(\theta_b/2)\\ \begin{pmatrix}-\cos(\theta_h/2)\sin(\theta_p/2)\cos(\theta_b/2)-\sin(\theta_h/2)\cos(\theta_p/2)\sin(\theta_b/2)\\ \cos(\theta_h/2)\sin(\theta_p/2)\sin(\theta_b/2)-\sin(\theta_h/2)\cos(\theta_p/2)\cos(\theta_b/2)\\ \sin(\theta_h/2)\sin(\theta_p/2)\cos(\theta_b/2)-\cos(\theta_h/2)\cos(\theta_p/2)\sin(\theta_b/2)\end{pmatrix}\end{bmatrix} \tag{1}$$

3.2. 从四元数转换到欧拉角

根据上面的公式(1)，如果已知左侧的四元数 q 就可以求出右侧的欧拉角 $(\theta_h,\theta_p,\theta_b)$，结果如下：

$$\theta_p=\operatorname{asin}(-2(yz+wx))$$

$$\theta_h=\begin{cases}\operatorname{atan2}(xz-wy,1/2-x^2-y^2) & \text{若}\cos p\neq 0\\ \operatorname{atan2}(-xz-wy,1/2-y^2-z^2) & \text{其他}\end{cases} \tag{2}$$

$$\theta_b=\begin{cases}\operatorname{atan2}(xy-wz,1/2-x^2-z^2) & \text{若}\cos p\neq 0\\ 0 & \text{其他}\end{cases}$$

4. 在 Unity 3D 中欧拉角和四元数的应用

四元数本质上是一个角轴对，即物体绕着某一个轴旋转一定的角度[3]。如图 3 所示，是物体的所处的一个状态，轴向量为 $\boldsymbol{n}=(0.3,0.9,-0.2)$ 和角 $\theta=73.2^\circ$。

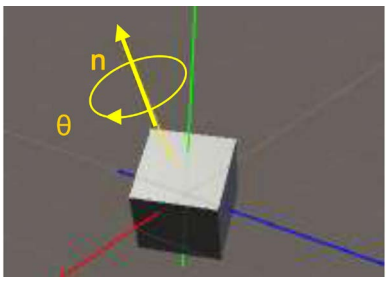

Figure 3. Determination of object orientation by angular pair
图 3. 角轴对决定物体方位

已知轴的方向 $\boldsymbol{n}$，那么绕这个轴旋转 θ 角，可以是 0˚、30˚、90˚、180˚、360˚、720˚等。0˚是指物体坐标系和惯性坐标系重合的状态。

轴可以是任意的，下面是四元数 $q=(x,y,z,w)$ 与角轴对 $(\theta,\boldsymbol{n})=(\theta,n_x,n_y,n_z)$ 之间的关系：

$$q=(x,y,z,w)=(n_x\sin(\theta/2),n_y\sin(\theta/2),n_z\sin(\theta/2),\cos(\theta/2))$$

当 $\theta=0$ 时，$q=(0,0,0,1)$。因为 $\sin(\theta/2)=0$，所以有 $\boldsymbol{n}=(n_x,n_y,n_z)$，可以是任意轴。

其中 $q=(0,0,0,1)$ 这个四元数的含义是不旋转。其地位相当于实数中的 1，或矩阵中的单位阵。任何四元数与该数相乘保持不变。在 Unity3D 中对这个四元数有专门的定义：

Quaternion.identity

因为四元数每个分量的取值范围是[−1, 1]，因此当用户给出不同的欧拉角时，系统会根据用户给出的欧拉角进行计算[4]，得到相应的四元数。例如当旋转轴保持为竖直方向的 y 轴时，即 $\boldsymbol{n}=(0,1,0)$。$\theta=0$ 时的四元数为 $q=(0,0,0,1)$，当 $\theta=360$˚时，四元数为：

$$q=(0\sin(360/2),1\sin(360/2),0\sin(360/2),\cos(360/2))=(0,0,0,-1)$$

从欧拉角来看，0 度和 360 度是同一状态，然而却对应着不同的四元数，如表 1 所示是欧拉角与四元数的一个对应关系。

Table 1. Initial Euler angle, quaternion and final Euler angle
表 1. 初始欧拉角，四元数与最终欧拉角

初始欧拉角	四元数	最终欧拉角	初始欧拉角	四元数	最终欧拉角
0, 0, 0	0, 0, 0, 1	0, 0, 0			
0, 90, 0	0, 0.7, 0, 0.7	0,90, 0	0, 450, 0	0, -0.7, 0, -0.7	0, 90, 0
0, 180, 0	0, 1, 0, 0	0, 180, 0	0, 540, 0	0, -1, 0, 0	0, 180, 0
0, 270, 0	0, 0.7, -0.7	0, 270, 0	0, 630, 0	0, -0.7, 0, 0.7	0, 270, 0
0, 360, 0	0, 0, 0, -1	0, 0, 0	0, 720, 0	0, 0, 0, 1	0, 0, 0

在 Unity 3D 中，由欧拉角生成四元数的代码：Quaternion q = Quaternion.Euler (0, 90, 0)；由四元数生成欧拉角的代码：Vector3 v = q.eulerAngles。

表中第一列是用户给定的欧拉角，由此欧拉角生成四元数列于第二列，根据四元数再生成欧拉角列在第三列，即 Vector3 v。从表中可以看出，从 0˚到 360˚再到 720˚，四元数的实部从 1 到 0 到−1 再到 0 再到 1，也就是欧拉角转两周，四元数回到初始值。因此同一个欧拉角会有两个四元数与之对应，因为同一个状态可以沿旋转轴正向旋转，也可以反转到同一状态，可以认为将轴的方向反向[5]。

5. 分析及总结

本文介绍了欧拉角和四元数的概念，以及两者之间的转换关系，又通过实例数据解释说明了欧拉角和四元数在 Unity3D 中的存在形式和联系，以利于 Unity3D 使用者更好的理解旋转操作。

参考文献

[1] 庞钦存. 四元数在游戏引擎 Unity3D 中的应用[J]. 产业与科技论坛, 2015, 14(17): 63-64.

[2] 李志伟, 李克昭, 赵磊杰, 王云凯, 梁晓庆. 基于单位四元数的任意旋转角度的三维坐标转换[J]. 大地测量与地球动力学, 2017, 37(1): 81-85.

[3] 鲁丹丹, 翟永翠, 周玉芳. 基于四元数的 DR 技术[J]. 指挥控制与仿真, 2019(2): 98-103.

[4] 张帆, 曹喜滨, 邹经湘. 一种新的全角度四元数与欧拉角的转换算法[J]. 南京理工大学学报(自然科学版), 2002(4): 376-380.

[5] 王彤, 马建仓, 秦涛, 柏会宁. 基于旋转四元数的姿态解算算法[J]. 弹箭与制导学报, 2014, 34(3): 15-16+20.