歐拉角定義的是三次基本旋轉的三個角度,旋轉順序和旋轉參考軸隨不同領域有不同,在飛機上常用的是Z-X-Y規則,三次順序旋轉角度是偏航,俯仰,橫滾,具體還要指明是旋轉的
參考系還是旋轉的體坐標系通過這三個基本旋轉角度可以得到坐標變換矩陣,也叫姿態矩陣,一個3*3矩陣

歐拉角法是直接迭代歐拉角微分方程,但是當轉動規則中的第二次基本旋轉接近90度的時候,第一次和第三次基本旋轉將變得作用相同,稱為**萬向鎖**,因此受到一定限制

方向餘弦法實際上就是直接迭代姿態矩陣微分方程求解姿態,但是是一個3*3矩陣,計算起來比較多

而四元數是將旋轉描述為一次旋轉,四元數微分方程也比較簡潔,迭代起來相對方便快捷
一般傳感器應該是採用的四元數迭代,然後由四元數可以算出姿態矩陣,最後算出歐拉角

一，什麼是歐拉角？
用一句話說，歐拉角就是物體繞坐標系三個坐標軸(x,y,z軸）的旋轉角度。

在這裡，坐標系可以是世界坐標系，也可以是物體坐標系，旋轉順序也是任意的，可以是
xyz,xzy,yxz,zxy,yzx,zyx中的任何一種，甚至可以是xyx,xyy,xzz,zxz等等等等。。。。。。
所以說歐拉角多種多樣。歐拉角可分為兩種情況：

1，靜態：即繞世界坐標系三個軸的旋轉，由於物體旋轉過程中坐標軸保持靜止，所以稱為靜態。

2，動態：即繞物體坐標系三個軸的旋轉，由於物體旋轉過程中坐標軸隨著物體做相同的轉動，所以稱為動態。

對於分別繞三個坐標軸旋轉的情況，下述定理成立：

物體的任何一種旋轉都可分解為分別繞三個軸的旋轉，但分解方式不唯一。如：

假設繞y軸旋轉為heading，繞x軸旋轉為pitch，繞z軸旋轉為bank，則先heading45°再pitch90°等價於先pitch90°再bank45°。

二，萬向鎖是一種什麼現象？
對於動態歐拉角，即繞物體坐標系旋轉。（靜態不存在萬向鎖的問題）無論heading和bank為多少度，只要pitch為±90°（即繞第二個軸的旋轉），就會出現萬向鎖現象。

為了對這一現象有一個感性的認識，請拿起自己的手機（沒有？不會吧）和一支筆（用作旋轉軸），親手做如下的幾個旋轉。

首先確定手機的物體坐標系朝向，為了方便記憶，我們假設z軸與手機屏幕垂直（手機平放於桌面）指向上方，手機較短的一條邊為x軸，較長的一條邊為y軸

（方向由手機尾部指向頭部），物體坐標系的原點是手機左下角的頂點。（注意旋轉順序為zyx）繞z軸旋轉任意角度（注意x和y軸也跟著一起旋轉），再繞y軸旋轉90°，
再繞x軸旋轉任意角度。通過多次嘗試，你會發現一個共同點：z軸永遠是水平的，通俗的說，手機永遠也不會立起來！本來我們以為手機會指向任何方向，
但實際上手機好像是被鎖在桌面上，只能指向水平的某個方向，這個現象就稱為萬向鎖。
而如果繞y軸旋轉不等於90°（1°也好89°也好），只要選擇適當的繞x和z的角度，就可以讓手機指向三維空間中的任何一個方向，手機是自由的，也就不會遇到萬向鎖現象。

三，遊戲動畫中遇到萬向鎖時會發生什麼？

之所以會出現萬向鎖現象，本質原因是，當第二次旋轉角度為90°時，第三個軸就被轉到了與第一個軸相同的方向，因此手機缺失了一個自由度（豎直方向的自由度缺失），
只剩下第一個軸和第二個軸的自由度。而只有兩個自由度意味著手機的運動被限制在了二維空間中，因此手機永遠立不起來。

在遊戲中，當角色旋轉的動畫觸發時，角色就會做一系列連續的旋轉變換，每一個變換都要用一組歐拉角來表示，但是不可能吧每一個方位的歐拉角都存儲起來，
因此動畫師定義了一系列關鍵幀，指定關鍵幀處角色的方位（用一組歐拉角描述），然後計算機根據時間t對這幾組歐拉角進行插值，得到一系列歐拉角。

如果pitch不是±90°，就不會出現萬向鎖現象，插值後的一系列歐拉角完全可以刻畫出我們所期望的角色旋轉路徑。

如果某個關鍵幀的pitch即繞第二個軸的旋轉為90°，就會遇到萬向鎖，這時手機只能在水平面內旋轉，如果動畫師指定下一個關鍵幀手機的方位不是立起來的，
沒有任何問題，但如果指定的下一個關鍵幀的方位是立起來的，會出現什麼情況呢？

為了能有一個感性的認識，還是以手機為例,下面我指定了四個關鍵幀，四個關鍵幀處手機的方位分別用R0,R1,R2,R3四組歐拉角表示(逆時針為正方向，右手法則）：

（注意R0處的物體坐標系與世界坐標系的指向是相同的，我假定z軸向上，x軸向右，y軸指向自己的胸口）

繞z軸旋轉角度 繞y軸旋轉角度 繞x軸旋轉角度

R0: 0 0 0

R1: 90 0 0

R2: 90 -90 0

R3: 0 0 90

請先分別對手機做這四個變換，然後記住手機的這四個方位，想像一下你「期望」的連續的動畫應該是什麼樣子的。

簡單說明如下：初始時刻，手機位於桌面上，屏幕朝上，手機頭部指向你的胸口，然後，手機在桌面內逆時針旋轉90°，知道手機頭部指向右側，然後手機的右邊
開始高起來，直到與桌面垂直，此時手機頭部仍然指向右側，由於R2的第二次旋轉是90°，因此手機進入萬向鎖模式。然後手機應該背對著你，垂直於桌面站立起來，
直到手機是豎直的背對你。

但實際情況是否是這樣的呢？

你可以自己嘗試對這個四個方位角插值，然後進行旋轉，看看得到的路徑是否和上述我們所期望的相同。

以下是我的嘗試：

求出R0 到R1以及R1到R2的插值，然後旋轉，完全符合上面的路徑。但是再求出R2到R3的幾個插值後，旋轉得到的路徑與期望不符。比如這兩個插值：

z：60 y：60 x：30

z: 45 y:45 x:45.

做這兩個旋轉，你會發現手機與桌面不垂直，也就是R2到R3的路徑與期望的發生了偏移。這就是《3D數學基礎：圖形與遊戲開發》中提到的「路徑偏移」和「攝像機抖動」。

總結：如果動畫師在某個關鍵幀處指定了會引發萬向鎖的方位，則下一個關鍵幀的方位一旦超出了萬向鎖的約束範圍，則這兩個關鍵幀之間的路徑就會發生偏移，反映
在角色動畫上是旋轉偏移，反映在鏡頭控制上就是鏡頭抖動。

要獲得路徑偏移的感性認識，可以參考這個視頻：這個視頻和我的描述有些不同，該視頻使用一個稱為萬向節的奇怪裝置解釋的，而我是直接用的物體坐標系
但路徑偏移都是一樣的。

四，怎樣解決萬向鎖這個問題？

出現這個問題的根本原因是在萬向鎖情況下對歐拉角的插值不是線性的，因此旋轉路徑發生偏移。

解決方法是：

將歐拉角轉換為四元數，對四元數進行slerp即球麵線性插值，再將這一系列四元數轉換為對應的歐拉角，而後作用於角色。缺點是耗費一定的內存，但角色可以任意旋轉，
靈活度高。