Calculation of Euler angles

Prokopi Nikolaev

www.geom3d.com
geom3d@geom3d.com

The common transformation task using Euler angles consists of 3 rotations (Fig. 1):

1. Rotate around Z1 axis local coordinate system (LCS) by angle α.
2. Rotate around transformed X1 axis (X' on Fig. 1) by angle β.
3. Rotate around transformed Z1 axis by angle γ.

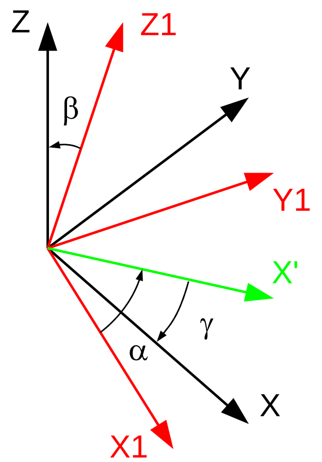

Fig. 1

The angles α, β, γ are called precession, nutation and rotation respectively.

The presented algorithm allows obtaining Euler angles for transformation zero-based orthogonal LCS into world coordinate system (WCS).

X, **Y**, **Z** are axis of WCS:

X = (1, 0, 0), **Y** = (0, 1, 0), **Z** = (0, 0, 1).

It is implied that axis **X1**, **Y1**, **Z1** of LCS are defined in WCS:

X1 = ($X1_x$, $X1_y$, $X1_z$), **Y1** = ($Y1_x$, $Y1_y$, $Y1_z$), **Z1** = ($Z1_x$, $Z1_y$, $Z1_z$).

The first step is to find α. We should calculate the rotation angle around **Z1** axis to place **X1** axis into XY plane of WCS. Let's see how to do it. The new position of **X1** is **X'** on Fig.1 should satisfy two conditions: it should be perpendicular to **Z** axis because it lies in XY plane. And it should be perpendicular to **Z1** axis because it is still part of LCS. So the best candidate to the role of **X'** is a vector product of **Z1** and **Z**:

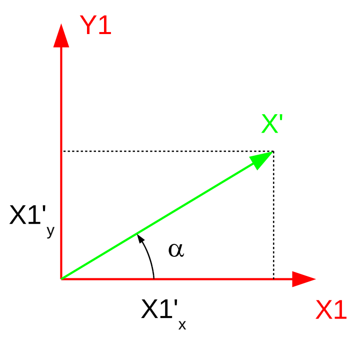

Fig. 2

X' = **Z1**×**Z** = ($Z1_y$, $-Z1_x$, 0).

Now, when the vector **X'** is found it is easy to find α. It is the angle between **X1** and **X'**. To calculate it let's move into plane X1Y1 of LCS (Fig. 2). This plane contains **X'**.

Angle α can be obtained using standard library function atan2(y,x). This function calculates the arctangent of *y*/*x* and is very convenient for our purpose. To use it we need projections of **X'** on **X1** and **Y1** axes. These projections ($X1'_x$ and $X1'_y$) are scale products of **X'** to **X1** and **Y1**:

$X1'_x = \mathbf{X'}\cdot\mathbf{X1} = X'_x \cdot X1_x + X'_y \cdot X1_y + X'_z \cdot X1_z = Z1_y \cdot X1_x - Z1_x \cdot X1_y$,

$X1'_y = \mathbf{X'}\cdot\mathbf{Y1} = X'_x \cdot Y1_x + X'_y \cdot Y1_y + X'_z \cdot Y1_z = Z1_y \cdot Y1_x - Z1_x \cdot Y1_y$.

$\alpha = \text{atan2}(X1'_y, X1'_x)$.

The next step – find β – the angle of rotation around **X'**.

The picture (Fig. 3 a) shows the plane containing **Z** and **Z1**. This plane is perpendicular to **X'**, so the rotation of **Z1** to **Z** will take place in this plane. The required β is the angle between **Z1** and **Z**. Again we can use atan2 to calculate it. The projection of **Z1** to **Z** is defined as $Z1_z$. $Z1_{xy}$ is the projection of **Z1** to XY plane of WCS (Fig. 3 b):

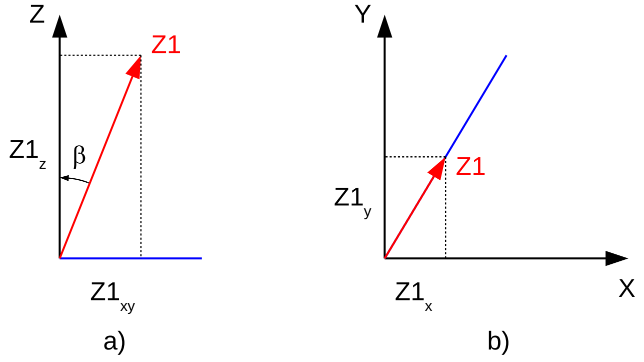

Fig. 3

$$Z1_{xy} = \sqrt{Z1_x^2 + Z1_y^2}.$$

And the angle is:

β = atan2 ($Z1_{xy}$, $Z1_z$).

The last angle is γ. This is the angle between **X'** and **X** (Fig.4).

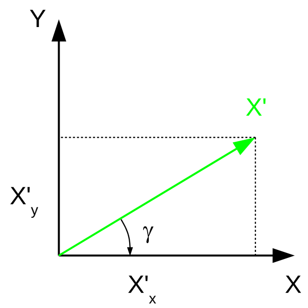

Fig. 4

It can be found in the same way:

γ = –atan2 (X'_y, X'_x) = –atan2 ($-Z1_x$, $Z1_y$).

There is one special case. This algorithm does not work when **Z** and **Z1** are collinear. In this case we should use different approach. In fact algorithm becomes trivial:

$\alpha = 0$, $\beta = 0$ for $Z1_z > 0$, $\beta = \pi$ otherwise, γ = –atan2 ($X1_y$, $X1_x$).

To distinguish this special case we can use the value of $Z1_{xy}$ – projection length of **Z1** to XY plane. For special case it will be 0.

The final remark: in order to get Euler angles for back transformation of WCS to LCS we can simply change signs of angles and exchange α with γ.

In conclusion there is a C-cod implementation of the described algorithm.

```
#include <math.h>
#include <float.h>

#define PI 3.141592653589793

void LCS2Euler (
    double X1x, double X1y, double X1z,
    double Y1x, double Y1y, double Y1z,
    double Z1x, double Z1y, double Z1z,
    double *pre, double *nut, double *rot)
    {
    double Z1xy = sqrt (Z1x*Z1x + Z1y*Z1y);
    if (Z1xy > DBL_EPSILON)
        {
        *pre = atan2 (Y1x*Z1y - Y1y*Z1x, X1x*Z1y - X1y*Z1x);
        *nut = atan2 (Z1xy, Z1z);
        *rot = -atan2 (-Z1x, Z1y);
        }
    else
        {
        *pre = 0.;
        *nut = (Z1z > 0.) ? 0. : PI;
        *rot = -atan2 (X1y, X1x);
        }
    }
```