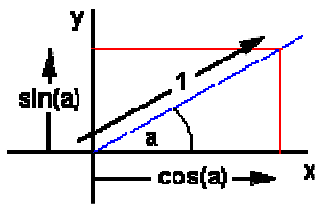


Maths - Rotation Matrices

Rotations can be represented by [orthogonal matrices](#) (there is an equivalence with quaternion multiplication [as described here](#))



First rotation about z axis, assume a rotation of 'a' in an anticlockwise direction, this can be represented by a vector in the positive z direction (out of the page).

If we take the point $(x=1, y=0)$ this will rotate to the point $(x=\cos(a), y=\sin(a))$

If we take the point $(x=0, y=1)$ this will rotate to the point $(x=-\sin(a), y=\cos(a))$

3D rotations

For an alternative we to think about using a matrix to represent rotation [see basis vectors here](#).

Rotation about the z axis

is given by the following matrix:

Rotation about z axis is: $R_z =$

$\cos(a)$	$-\sin(a)$	0
$\sin(a)$	$\cos(a)$	0
0	0	1

For example if we choose an angle of +90 degrees we get


0	-1	0
1	0	0
0	0	1

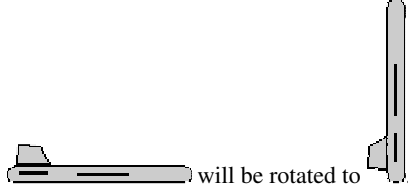
Prerequisites

Upto now we have used 3x3 matrices to represent rotations.

If you are not familiar with this subject you may like to look at the following pages first:

- [Matrices](#)
- [Rotations](#)

the direction of rotation is given by the right hand rule  where the thumb is in the +z direction (toward the viewer) and the fingers show the positive direction of rotation so



The conventions used are explained [here](#) and the relationship to other standards [here](#).

Other 90 degrees steps are shown [here](#).



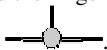
Similarly with rotation about y axis:

Rotation about y axis is: $R_y =$

$\cos(a)$	0	$\sin(a)$
0	1	0
$-\sin(a)$	0	$\cos(a)$

For example if we choose an angle of +90 degrees we get

0	0	1
0	1	0
-1	0	0

the direction of rotation is given by the right hand rule  where the thumb is in the +y direction (up) and the fingers show the positive direction of rotation so  will be rotated to .

And rotation about x axis:

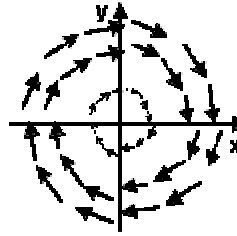
Rotation about x axis is: $R_x =$

1	0	0
0	$\cos(a)$	$-\sin(a)$
0	$\sin(a)$	$\cos(a)$

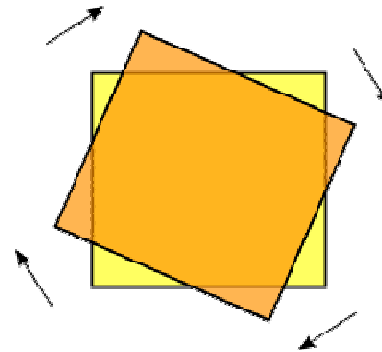
For example if we choose an angle of +90 degrees we get

1	0	0
0	0	-1
0	1	0

Transforms



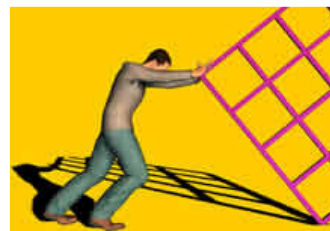
A transform maps every point in a vector space to a possibly different point.




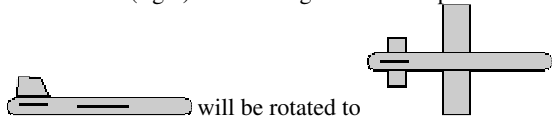
When transforming a computer model we transform all the vertices.

To model this using mathematics we can use [matrices](#), [quaternions](#) or other algebras which can represent multidimensional linear equations.

[This page](#) explains this.



the direction of rotation is given by the right hand rule  where the thumb is in the +x direction (right) and the fingers show the positive direction of rotation so



General rotation matrix:

Rotation about a general axis is:

r_{00}	r_{01}	r_{02}
r_{10}	r_{11}	r_{12}
r_{20}	r_{21}	r_{22}

This general rotation can be represented by a combination of the above rotations about Rz, Ry and Rx. As explained below the order of these rotations is important, there are different conventions for which order is assumed, as [explained here](#).

Rotation matrices are orthogonal as [explained here](#).

for Java and C++ code to implement these rotations [click here](#)

isRotationMatrix

This code checks that the input matrix is a pure rotation matrix and does not contain any scaling factor or reflection for example

```
/**
 *This checks that the input is a pure rotation matrix 'm'.
 * The condition for this is:
 * R' * R = I
 * and
 * det(R) = 1
 */
public boolean isRotationMatrix(matrix m) {
    double epsilon = 0.01; // margin to allow for rounding errors
    if (Math.abs(m[0][0]*m[0][1] + m[0][1]*m[1][1] + m[0][2]*m[1][2]) > epsilon) return false;
    if (Math.abs(m[0][0]*m[2][0] + m[0][1]*m[2][1] + m[0][2]*m[2][2]) > epsilon) return false;
    if (Math.abs(m[1][0]*m[2][0] + m[1][1]*m[2][1] + m[1][2]*m[2][2]) > epsilon) return false;
    if (Math.abs(m[0][0]*m[0][0] + m[0][1]*m[0][1] + m[0][2]*m[0][2] - 1) > epsilon) return false;
    if (Math.abs(m[1][0]*m[1][0] + m[1][1]*m[1][1] + m[1][2]*m[1][2] - 1) > epsilon) return false;
    if (Math.abs(m[2][0]*m[2][0] + m[2][1]*m[2][1] + m[2][2]*m[2][2] - 1) > epsilon) return false;
    return (Math.abs(det(m)-1) < epsilon);
    // det is defined here:
    // http://www.euclideanspace.com/maths/algebra/matrix/functions/determinant/threeD/
}
```

Successive Rotations

The order of Successive Rotations is significant, for example

Standards

There are a lot of choices we need to make in mathematics, for example,

- Left or right handed coordinate systems.
- Vector shown as row or column.
- Matrix order.
- Direction of x,y and z coordinates.
- Euler angle order
- Direction of positive angles
- Choice of basis for bivectors
- Etc. etc.

A lot of these choices are arbitrary as long as we are consistent about it, different authors tend to make different choices and this leads to a lot of confusion. Where standards exist I have tried to follow them (for example x3d and MathML) otherwise I have at least tried to be consistent across the site and have documented the choices I have made [on this page](#)

Standards

There are a lot of choices we need to make in mathematics, for example,

1. Rotate 90 degrees about x axis
2. Rotate 90 degrees about y axis
3. Rotate -90 degrees about x axis

This gives 90 degree rotation about Z axis,

whereas

1. Rotate 90 degrees about x axis
2. Rotate -90 degrees about x axis
3. Rotate 90 degrees about y axis

This gives 90 degree rotation about y axis (first 2 lines cancel out).

Successive rotations can be calculated by multiplying together the matrices representing the individual rotations. In the same way that the order of rotations are important, the order of matrix multiplication is important.

So with matrix algebra different rules apply than in the algebra of numbers.

In the first example the 3 rotations would be represented by:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

in the second case the rotations are represented by:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Axis Angle rotation

Any set of successive rotations can be replaced by a single equivalent rotation:

- Left or right handed coordinate systems.
- Vector shown as row or column.
- Matrix order.
- Direction of x,y and z coordinates.
- Euler angle order
- Direction of positive angles
- Choice of basis for bivectors
- Etc. etc.

A lot of these choices are arbitrary as long as we are consistent about it, different authors tend to make different choices and this leads to a lot of confusion. Where standards exist I have tried to follow them (for example x3d and MathML) otherwise I have at least tried to be consistent across the site. I have documented the choices I have made [on this page](#).

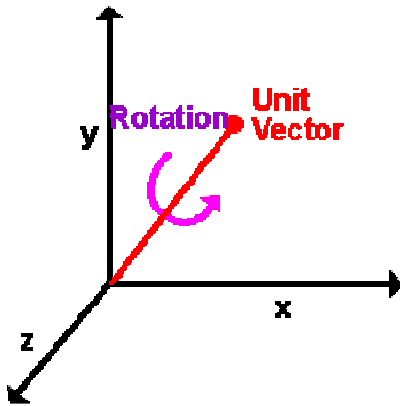
Rotation as Two Reflections

If we get two mirrors and put them at 90° to each other we can get a view that has been reflected in both mirrors. The object will appear to have been rotated by 180° which is twice the angle between the mirrors.



So 2 reflections in different planes are equivalent to a rotation.

[This page](#) explains this.



The matrix for this rotation is given by:

$$[R] = \begin{bmatrix} 1 + (1 - \cos(\text{angle})) * (x * x - 1) & -z * \sin(\text{angle}) + (1 - \cos(\text{angle})) * x * y & y * \sin(\text{angle}) + (1 - \cos(\text{angle})) * x * z \\ z * \sin(\text{angle}) + (1 - \cos(\text{angle})) * x * y & 1 + (1 - \cos(\text{angle})) * (y * y - 1) & -x * \sin(\text{angle}) + (1 - \cos(\text{angle})) * y * z \\ -y * \sin(\text{angle}) + (1 - \cos(\text{angle})) * x * z & x * \sin(\text{angle}) + (1 - \cos(\text{angle})) * y * z & 1 + (1 - \cos(\text{angle})) * (z * z - 1) \end{bmatrix}$$

where:

- x, y and z represent a unit vector
- angle is the angle of rotation about this unit vector.

[see axis-angle to matrix conversion](#)

[Also see discussion here](#)

Representing Rotation and Translation

If we want to represent rotation and translation using a single matrix we need to use a 4x4 matrix [as explained here](#).

Further Reading

You may be interested in other means to represent orientation and rotational quantities such as:

- [Other ways to represent rotation](#)
- [Rotation about a point](#)

metadata block

Origin Graphing Software Scientific/Eng Plotting & Analysis Multi-Y Plots, Contour/Surface Plot www.OriginLab.com
Become a Math Teacher The Most Affordable Option Online for Becoming a Math Teacher! www.WGU.edu
Math Practice - Ages 5-14 A math website kids LOVE — Win awards, certificates, have fun! www.IXL.com/math

see also:

- [Determinants](#)
- [Kinematics](#)
- [Dynamics](#)
- [Conversion Euler To Matrix](#)
- [Conversion Matrix to Euler](#)
- [Conversion Matrix to Quaternion](#)
- [Conversion Quaternion to Matrix](#)

Correspondence about this page

- [david](#)
- [Open Forum Discussion](#)

Book Shop - [Further reading.](#)

Where I can, I have put links to Amazon for books that are relevant to the subject, click on the appropriate country flag to get more details of the book or to buy it from them.



Mathematics for 3D game Programming - Includes introduction to Vectors, Matrices, Transforms and Trigonometry. (But no euler angles or quaternions). Also includes ray tracing and some linear & rotational physics also collision detection (but not collision response).

Commercial Software Shop

Where I can, I have put links to Amazon for commercial software, not directly related to the software project, but related to the subject being discussed, click on the appropriate country flag to get more details of the software or to buy it from them.

Can you help?

Please send me any improvements to [here](#). I would appreciate ideas to make the pages more useful including error correction, ideas for new pages, improvements to wording. It helps if you quote the full URL of the page.

[Terminology](#) and [Notation](#)

Specific to this page here:

program

<http://sourceforge.net/projects/mjbworld/>

I am working on a project which uses these principles, if you would like to help me with this you are welcome to join in, here:

This site may have errors. Don't use for critical systems.

Copyright (c) 1998-2010 Martin John Baker - All rights reserved - [privacy policy](#).

Ads by Google

[2011 Matrix](#)

[Vector Art 3D](#)

[Matrix Point](#)

[3D Scaling](#)