Introduction to Computer Graphics
(C S 6 0 2)
Lecture 18
3D Transformations II

b) Rotation

Rotation is the process of moving a point in space in a non-linear manner. More particularly, it involves moving the point from one position on a sphere whose center is at the origin to another position on the sphere. Why would you want to do something like this? As we will show in later section, allowing the point of view to move around is only an illusion – projection requires that the POV be at the origin. When the user thinks the POV is moving, you are actually translating all your points in the opposite direction; and when the user thinks the POV is looking down a new vector, you are actually rotating all the points in the opposite direction; and when the user thinks the POV is looking down a new vector, you are actually rotating all the points in the opposite direction.

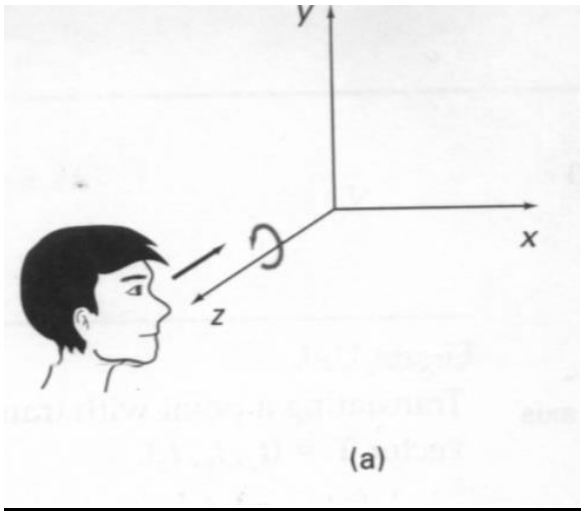
Normalization: Note that this process of moving your points so that your POV is at the origin looking down the +Z axis is called normalization.

Rotation a point requires that you know
1) the coordinates for the point, and
2) That you know the rotation angles.

You need to know three different angles: how far to rotate around the X axis(YZ rotation, or “pitch”); how far to rotate around the Y axis (XZ plane, or “yaw”); and how far to rotate around the Z axis (XY rotation, or “roll”). Conceptually, you do the three rotations separately. First, you rotate around one axis, followed by another, then the last. The order of rotations is important when you cascade rotations; we will rotate first around the Z axis, then around the X axis, and finally around the Y axis.

To show how the rotation formulas are derived, let’s rotate the point <x,y,z> around the Z axis with an angle of θ degrees.

ROLL:-



If you look closely, you should note that when we rotate around the Z axis, the Z element of the point does not change. In fact, we can just ignore the Z – we already know what it will be after the rotation. If we ignore the Z element, then we have the same case as if we were rotating the two-dimensional point <x,y> through the angle θ.

This is the way to rotate a 2-D point. For simplicity, consider the pivot at origin and rotate point P (x,y) where x = r cosΦ and y = r sinΦ

If rotated by θ then:

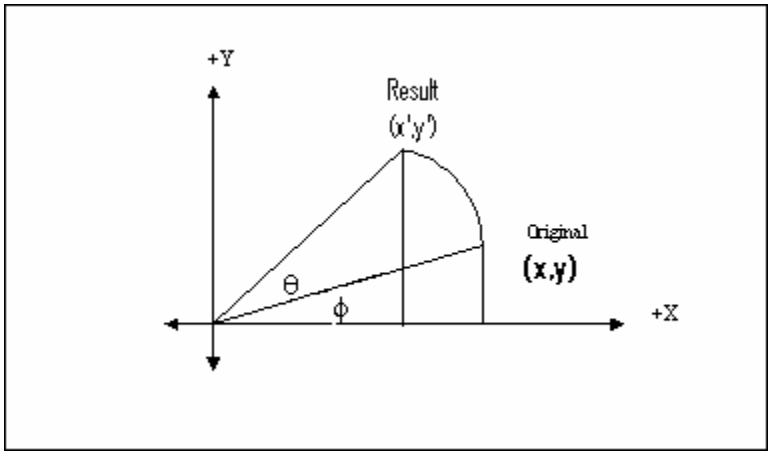
$x' = r\cos(\Phi + \theta)$

$= r\cos\Phi\cos\theta - r\sin\Phi\sin\theta$

and

$y' = r\sin(\Phi + \theta)$

$= r\cos\Phi\sin\theta + r\sin\Phi\cos\theta$



Replacing r cosΦ with x and r sinΦ with y, we have:

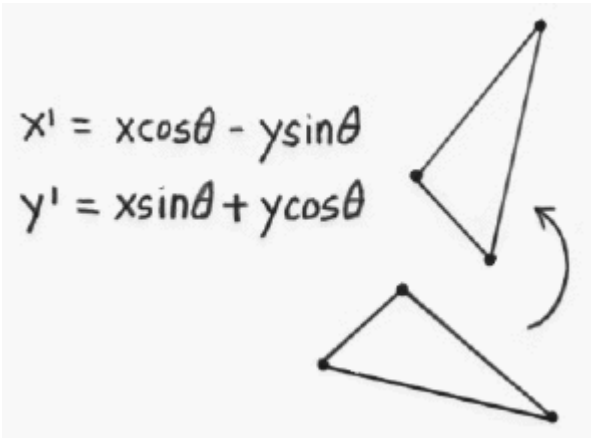
$x' = x\cos\theta - y\sin\theta$

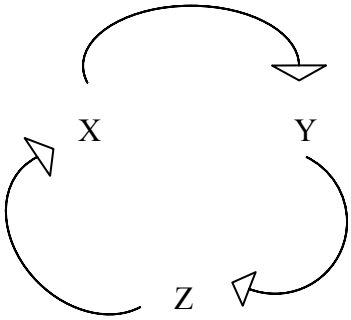
and

$y' = x\sin\theta + y\cos\theta$

and

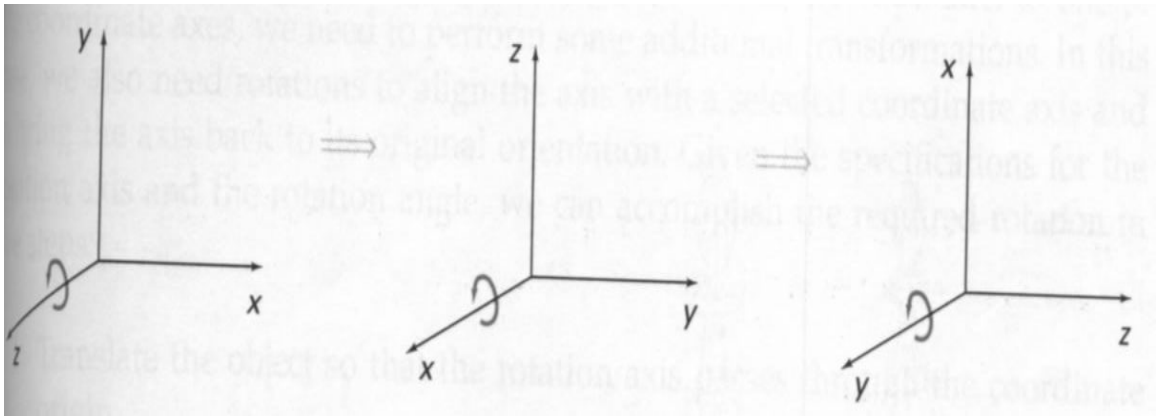
$z' = z$ (as it does not change when rotating around z-axis)





Now for rotation around other axes, cyclic permutation helps form the equations for yaw and pitch as well:

In the above equations replacing x with y and y with z gives equations for rotation around x-axis. Now in the modified equations if we replace y with z and z with x then we get the equations for rotation around y-axis.



Rotation about x-axis (i.e. in yz plane):

$x' = x$

$y' = y \cos\theta - z \sin\theta$

$z' = y \sin\theta + z \cos\theta$

Rotation about y-axis (i.e. in xz plane):

$x' = z \sin\theta + x \cos\theta$

$y' = y$

$z' = z \cos\theta - x \sin\theta$

Using Matrices to create 3D

A matrix is usually defined as a two-dimensional array of numbers. However, I think you will find it much more useful to think of a matrix as an array of vectors. When we talk about vectors, what it really mean is an ordered set of numbers (a tuple in mathematics terms). We can use 3D graphics vectors and points interchangeably for this, since they are both 3-tuples (or triples).

In general we work with "square" matrices. This means that the number of vectors in the matrix is the same as the number of elements in the vectors that comprise it. Mathematically, we show a matrix as a 2-D array of numbers surrounded by vertical lines. For example:

|x1 y1 z1|
|x2 y2 z2|
|x3 y3 z3|

we designate this as a 3*3 matrix (the first 3 is the number of rows, and the second 3 is the number of columns).
The "rows" of the matrix are the horizontal vectors that make it up; in this case, <x1, y1,z1>, <x2,y2,z2>, and <x3,y3,z3>. In mathematics, we call the vertical vectors "columns." In this case they are < x1,x2,x3>, <y1,y2,y3> and <z1,z2,z3>.
The most important thing we do with a matrix is to multiply it by a vector or another matrix. We follow one simple rule when multiplying something by a matrix: multiply each column by a multiplicand and store this as an element in the result. Now as I said earlier, you can consider each column to be a vector, so when we multiply by a matrix, we are just doing a bunch of vector multiplies. So which vector multiply do you use-the dot product, or the crosss product? You use the dot product.
We also follow on simple rule when multiplying a matrix by something: mubliply each ro by the multiplier. Again, rows are just vectors, and the type of ultiplicaiton is the dot product.
Let's look at some examples. First, let's assume that I have a matrix M, and I want to multiply it by a point < x,y,z>, the first ting I know is that the vector rows of the matrix must contain three elements (in other words, three columns). Why ? because I have to multiply those rows by my point using a dot product, and to do that, the two vectors must have the same number of element. Since I am going to get dot product for each row in M, I will end up with a tuple that has one element for each row in M. as I stated earlier, we work almost exclusively with square matrices, since I must have three columns, M will also have three rows. Lets see:

| 1 0 0|
< x,y,z> * |0 1 0| =**{** <x,y,z>*< 1,0,0> ,<x,y,z> **<**0,1,0**>,<**x,y,z**>** *<0,0,1> **}** = { x,y,z}
|0 0 1|

Using Matrices for Rotation

Roll (rotate about the Z axis):

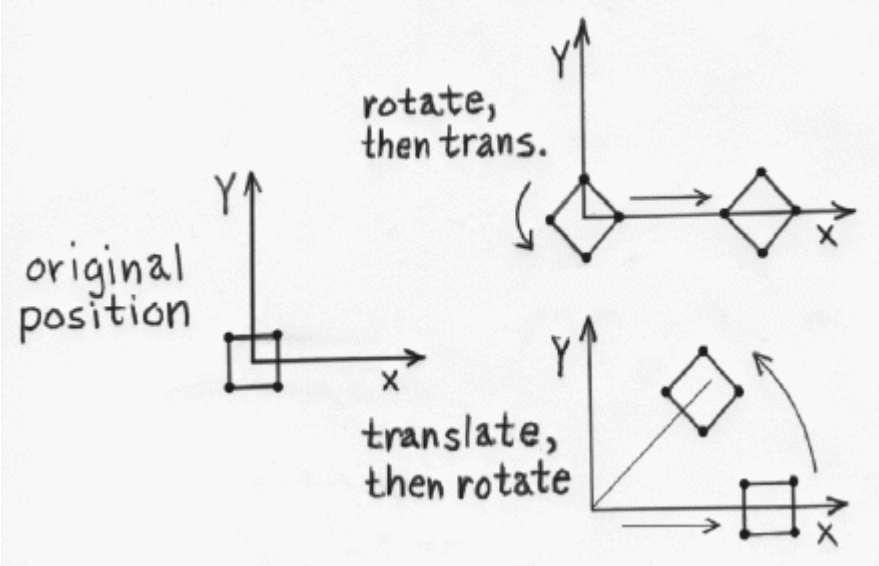
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Pitch (rotate about the X axis):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Yaw (rotate about the Y axis):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Example:

To show this happening, let's manually rotate the point <2,0,0> 45 degrees clockwise about the z axis.

$$\mathbf{v}' = R_z(45)\mathbf{v}$$

$$\mathbf{v}' = \begin{bmatrix} 0.707 & 0.707 & 0 \\ -0.707 & 0.707 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{v}' = \begin{bmatrix} 2 \times 0.707 + 0 \times 0.707 + 0 \times 0 \\ 2 \times -0.707 + 0 \times 0.707 + 0 \times 0 \\ 2 \times 0 + 0 \times 0 + 0 \times 0 \end{bmatrix}$$

$$\mathbf{v}' = \begin{bmatrix} 1.414 \\ -1.414 \\ 0 \end{bmatrix}$$

Now you can take an object and apply a sequence of transformations to it to make it do whatever you want. All you need to do is figure out the sequence of transformations needed and then apply the sequence to each of the points in the model.

As an example, let's say you want to rotate an object sitting at a certain point **p** around its z axis. You would perform the following sequence of transformations to achieve this:

$$\mathbf{v} = \mathbf{v}T(-\mathbf{p})$$

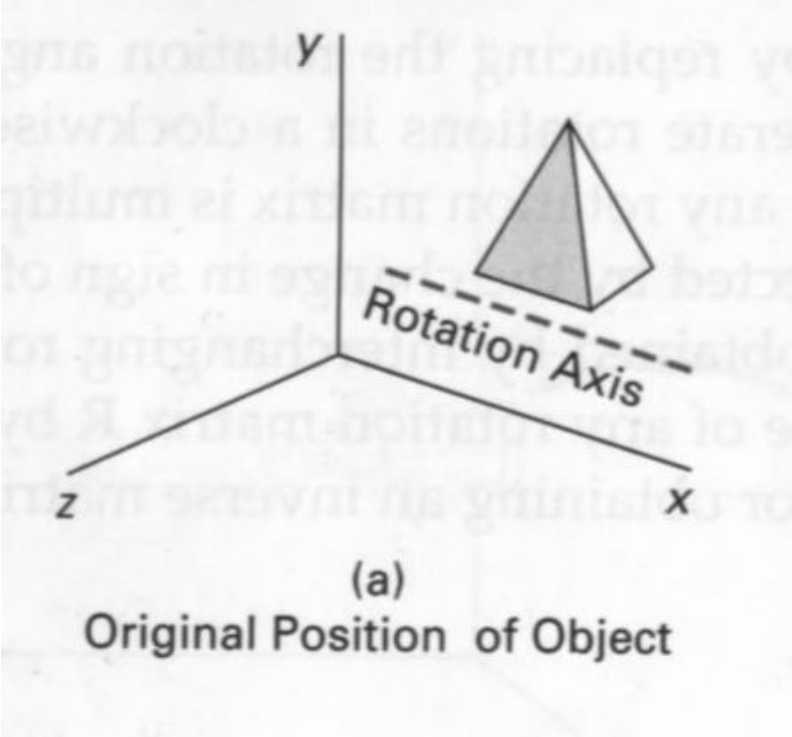
$$\mathbf{v} = \mathbf{v}R_z\left(\frac{\pi}{2}\right)$$

$$\mathbf{v} = \mathbf{v}T(\mathbf{p})$$

The first transformation moves a point such that it is situated about the world origin instead of being situated about the point **p**. The next one rotates it (remember, you can only rotate about the origin, not arbitrary points in space). Finally, after the point is rotated, you want to move it back so that it is situated about **p**. The final translation accomplishes this.

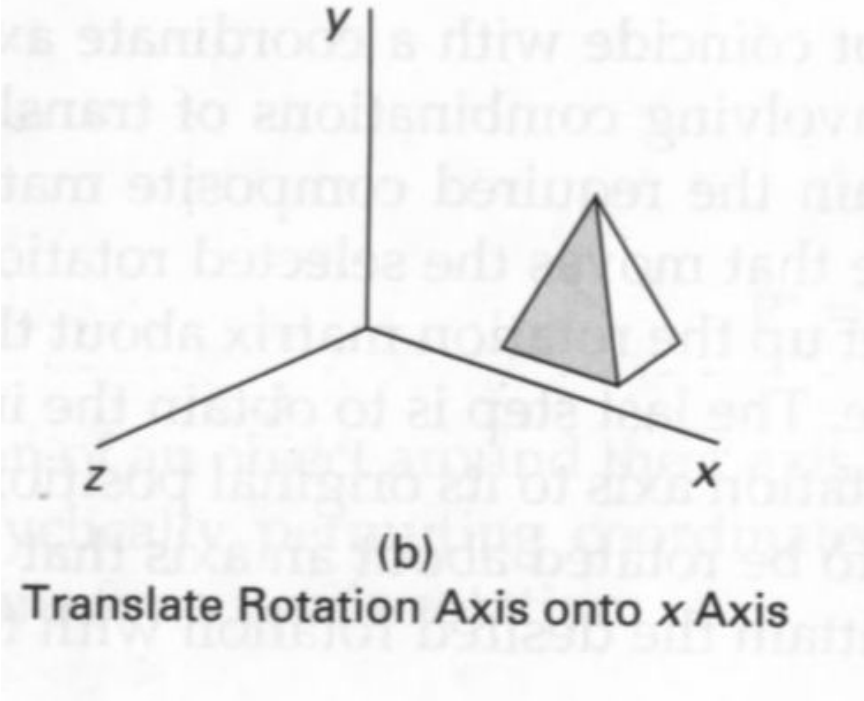
Rotation w.r.t. Arbitrary Axis:

If an object is required to be rotated with respect to a line acting as an axis of rotation, arbitrarily, then the problem is addressed using multiple transformations. Let us assume that such an arbitrary axis is parallel to one of the coordinate axes, say x-axis.



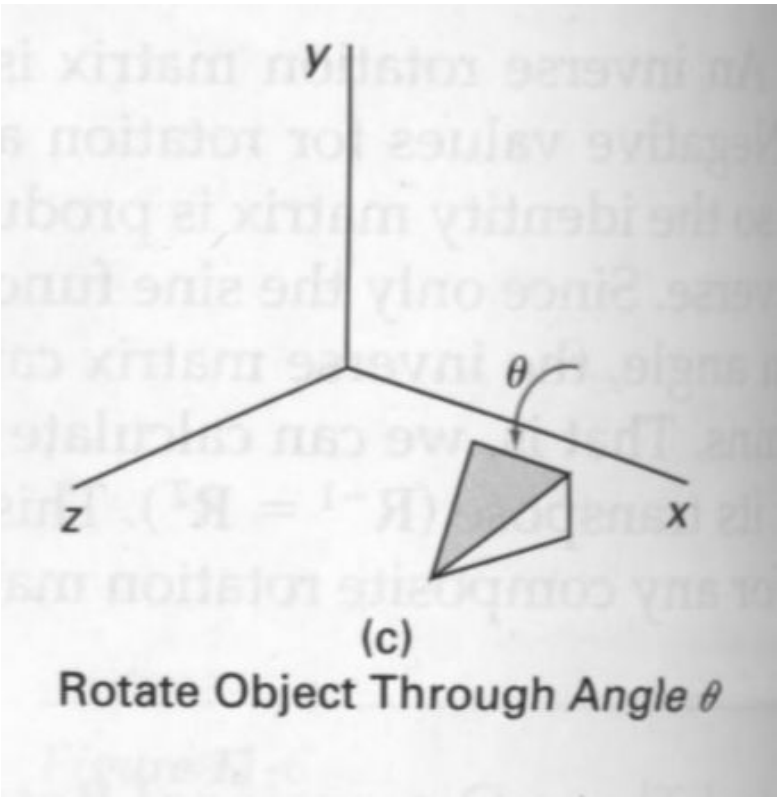
(a)
Original Position of Object

The first step in such case would be to translate the object such that the arbitrary axis coincides with the x-axis.



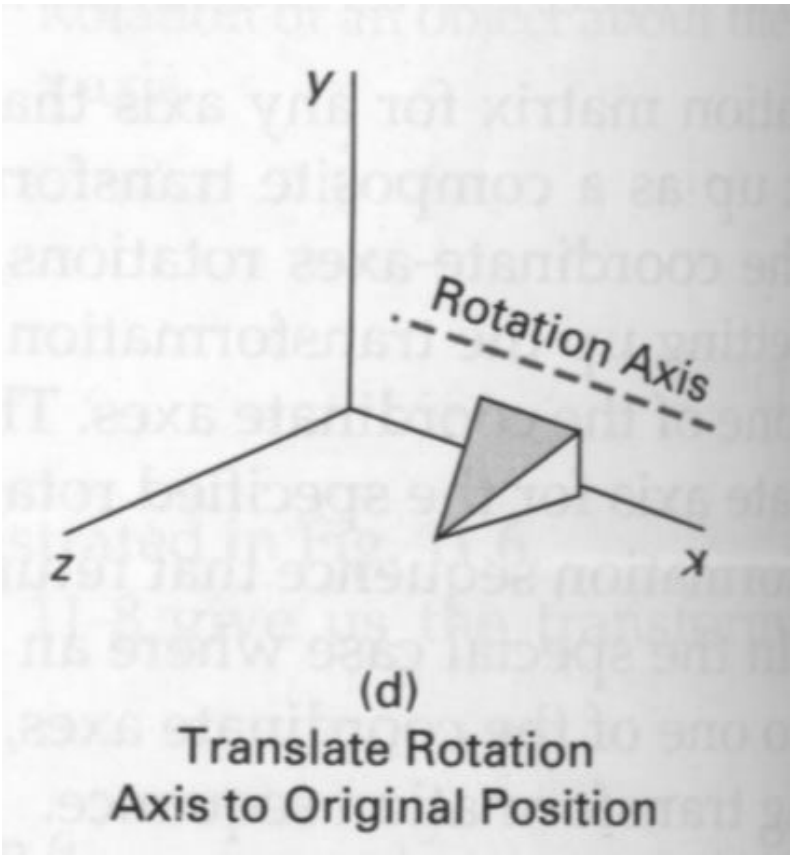
(b)
Translate Rotation Axis onto *x* Axis

The next step would be to rotate the object w.r.t. x-axis through angle θ.



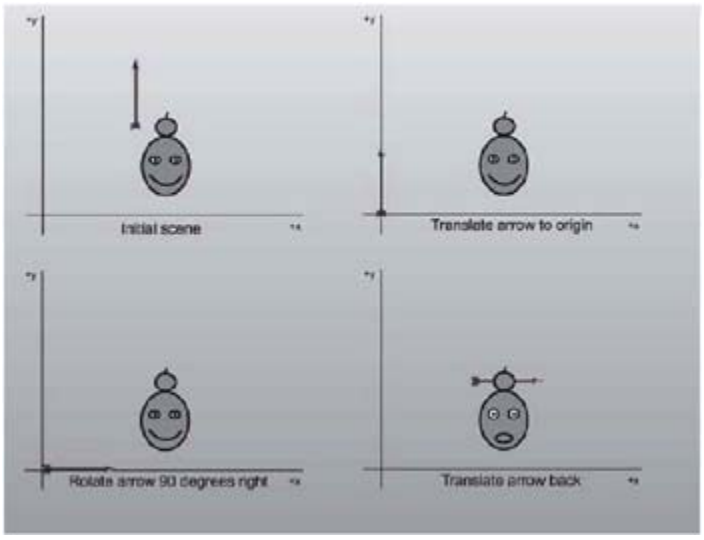
(c)
Rotate Object Through Angle θ

Then the object is translated such that the arbitrary axis gets back to its original position.

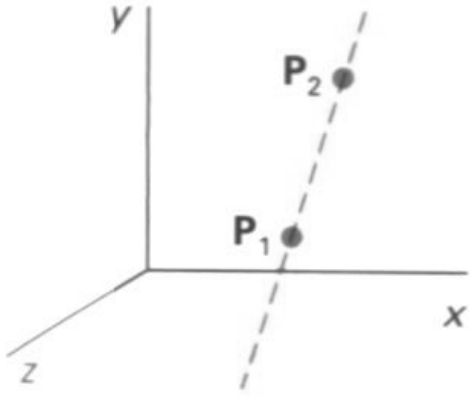


And thus the job is done.

An interesting usage of compound transformations:-

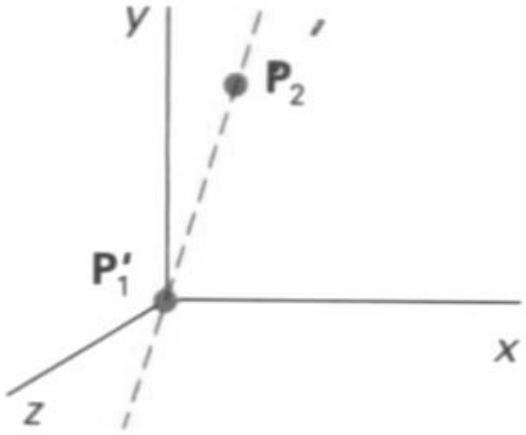


Now, if the arbitrary axis is not parallel to any of the coordinate axes, then the problem is slightly more difficult. It only adds to the number of steps required to get the job done. Let P1, P2 be the line arbitrary axis.



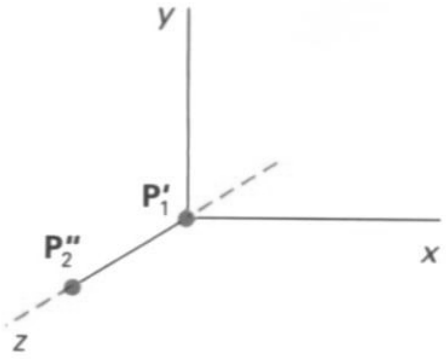
Initial Position

In the first step, the translation takes place that coincides the point P1 to the origin. Points after this step are P1' and P2'.

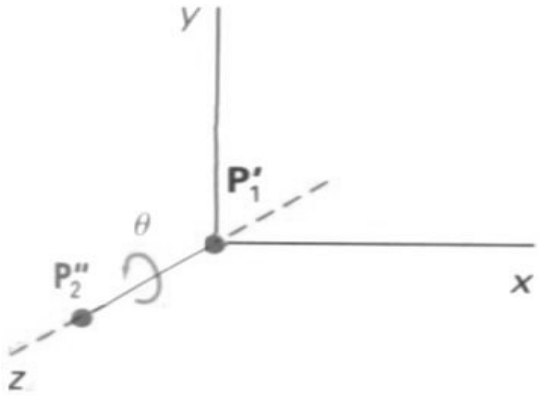


Step 1
Translate
P_1 to the Origin

Now the arbitrary axis is rotated such that the point P2' rotates to become P2'' and lies on the z-axis.

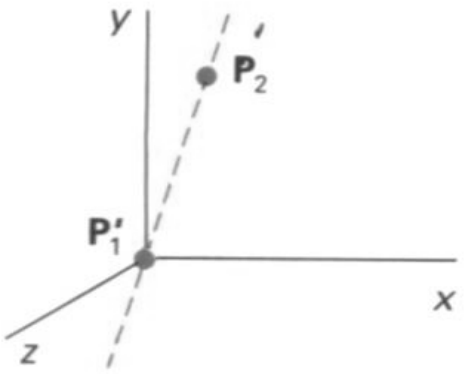


Step 2
Rotate P'_2
onto the *z* Axis



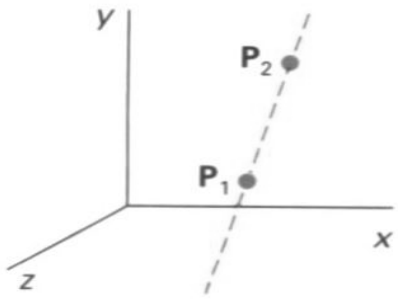
Step 3
Rotate the
Object Around the
z Axis

In the next step the object of interest is rotated around z-axis.



Step 4
Rotate the Axis
to the Original
Orientation

Now the object of interest is rotated about origin such that the arbitrary axis is poised like in above figure. Point P2'' gets back to its previous position P2'.



Step 5
Translate the
Rotation Axis
to the Original
Position

Finally the translation takes place to position the arbitrary axis back to its original position.

c) Scaling

Coordinate transformations for scaling relative to the origin are

X` = X . Sx
Y` = Y. Sy
Z` = Z. Sz

Scaling an object with transformation changes the size of the object and reposition the object relative to the coordinate origin. If the transformation parameters are not all equal, relative dimensions in the object are changed.

Uniform Scaling : We preserve the original shape of an object with a uniform scaling (Sx = Sy = Sz)

Differential Scaling : We do not preserve the original shape of an object with a differential scaling (Sx <> Sy <> Sz)

Scaling relative to the coordinate Origin:

Scaling transformation of a position P = (x, y, z) relative to the coordinate origin can be written as

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling with respect to a selected fixed position:

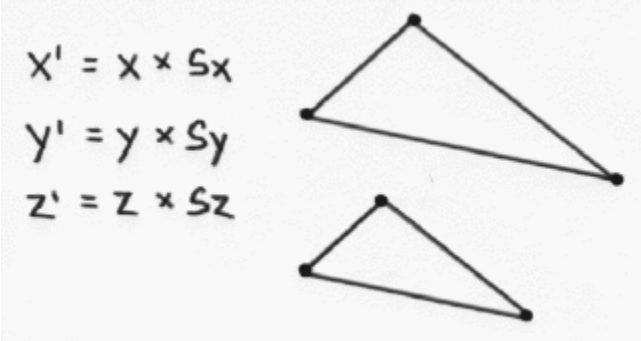
Scaling with respect to a selected fixed position (X_f,Y_f,Z_f) can be represented with the following transformation sequence:

1. Translate the fixed point to the origin.
2. Scale the object relative to the coordinate origin
3. Translate the fixed point back to its original position

For these three transformations we can have composite transformation matrix by multiplying three matrices into one

$$\begin{bmatrix} 1 & 0 & 0 & X_f \\ 0 & 1 & 0 & Y_f \\ 0 & 0 & 1 & Z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -X_f \\ 0 & 1 & 0 & -Y_f \\ 0 & 0 & 1 & -Z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} S_x & 0 & 0 & (1-S_x)X_f \\ 0 & S_y & 0 & (1-S_y)Y_f \\ 0 & 0 & S_z & (1-S_z)Z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



d) Reflection

A three-dimensional reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane. In general, three-dimensional reflection matrices are set up similarly to those for two dimensions. Reflections relative to a given axis are equivalent to 180 degree rotations.

The matrix representation for this reflection of points relative to the X axis

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

The matrix representation for this reflection of points relative to the Y axis

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

The matrix representation for this reflection of points relative to the xy plane is

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

e) Shears

Shearing transformations can be used to modify object shapes.
As an example of three-dimensional shearing, the following transformation produces a
z-axis shear:

$$\begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parameters a and b can be assigned and real values. The effect of this transformation matrix is to alter x and y- coordinate values by an amount that is proportional to the z value, while leaving the z coordinate unchanged.

y-axis Shear $\begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & c & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

x-axis Shear $\begin{bmatrix} 1 & 0 & 0 & 0 \\ b & 1 & 0 & 0 \\ c & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$