Lecture 3: Transformations of 3D Worlds

In the last lecture we defined objects by specifying the coordinates of their vertices in some Cartesian space. We also introduced a canonical form for perspective projection, namely that we view the scene from the origin looking along the z axis. However, we need to be able to view our graphical objects from any arbitrary position that we choose. This means that we would like to change the coordinates of every point in the scene, such that some chosen viewpoint **C** = [Cx,Cy,Cz] is the origin and some view direction **d** = [dx,dy,dz] is the Z axis. Frequently, we may want to transform the points of a graphical scene for other purposes such as generation of special effects in pictures, like rotating objects. Transformations of this kind are achieved by multiplying every point of the scene by a transformation matrix. Unfortunately however, we cannot do all we need to do using normal Cartesian coordinates, and for that reason we now introduce a system called *homogeneous coordinates*. Three dimensional points expressed in homogeneous form have a fourth ordinate:

P = [p_x, p_y, p_z,s]

The fourth ordinate is a scale factor, and conversion to Cartesian form is achieved by dividing it into the other ordinates, so

[p_x, p_y, p_z,s] has Cartesian coordinate equivalent [p_x/s, p_y/s, p_z/s].

In most cases, s will be 1. The point of introducing homogenous coordinates is to allow us to translate the points of a scene by using matrix multiplication. This is achieved as follows:

$$[\,x, y, z, 1\,] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ tx & ty & tz & 1 \end{pmatrix} = [\,x+tx, y+ty, z+tz, 1\,]$$

The matrix for scaling a graphical scene is also easily expressed in homogenous form:

$$[x, y, z, 1] \begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = [sx*x, sy*y, sz*z, 1]$$

Notice that these two transformations are not commutative, and it is essential that they are carried out in the correct order. Diagram 3.1 illustrates the problem for a simple picture.

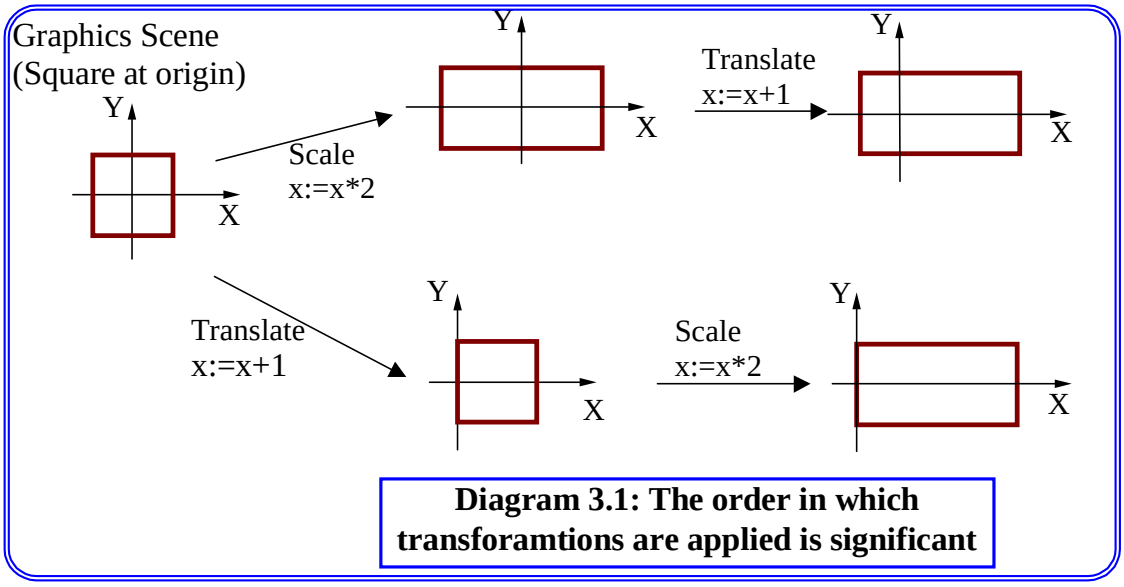


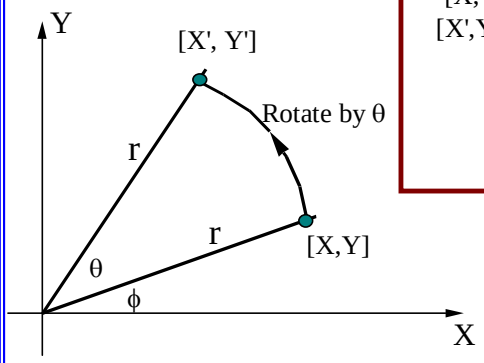
Diagram 3.1: The order in which transforamtions are applied is significant

Rotation however has to be treated differently since we need to specify an axis. The matrices for rotation about the three Cartesian axes are:

$$\boldsymbol{R}x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \text{Cos}(\theta) & \text{Sin}(\theta) & 0 \\ 0 & -\text{Sin}(\theta) & \text{Cos}(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \boldsymbol{R}y = \begin{pmatrix} \text{Cos}(\theta) & 0 & -\text{Sin}(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \text{Sin}(\theta) & 0 & \text{Cos}(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{R}z = \begin{pmatrix} \text{Cos}(\theta) & \text{Sin}(\theta) & 0 & 0 \\ -\text{Sin}(\theta) & \text{Cos}(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Some care is required with the signs. The above formulation obeys the conventions of a left hand axis system. That is, if the positive y-axis is taken as vertical, and the positive x-axis horizontal to the right, the positive z-axis is into the page. In these cases, rotation is in a clockwise direction



$$\begin{aligned} [X,Y] &= [r\,\text{Cos}\phi,\ r\,\text{Sin}\phi] \\ [X',Y'] &= [\ r\,\text{Cos}(\theta+\phi)\ ,\ r\,\text{Sin}(\theta+\phi)\ ] \\ &= [\ r\,\text{Cos}\phi\,\text{Cos}\theta - r\text{Sin}\phi\,\text{Sin}\theta,\ \ r\text{Sin}\phi\text{Cos}\theta + \ r\text{Cos}\phi\text{Sin}\theta\ ] \\ &= [\ X\,\text{Cos}\theta - Y\,\text{Sin}\theta,\ Y\text{Cos}\theta + X\text{Sin}\theta] \\ &= [\ X\ \ Y\ ] \begin{pmatrix} \text{Cos}\theta & \text{Sin}\theta \\ -\text{Sin}\theta & \text{Cos}\theta \end{pmatrix} \end{aligned}$$

Diagram 3.2: Derivation of the Rotation Matrix

when viewed from the positive side of the axis, or vice versa, anti-clockwise when viewed from the negative side of the axis. The derivation of the ***Rz*** matrix is shown in Diagram 3.2 the others may be proved similarly.

Inversions of these matrices can be computed easily, without recourse to Gaussian elimination, by considering the meaning of each transformation. For scaling, we substitute $1/s_x$ for s_x, $1/s_y$ for s_y and $1/s_z$ for s_z to invert the scaling. For translation we substitute $-t_x$ for t_x, $-t_y$ for t_y and $-t_z$ for t_z. For the rotation matrices we note that:

$$\text{Cos}(-\theta) = \text{Cos}(\theta) \text{ and } \text{Sin}(-\theta) = -\text{Sin}(\theta)$$

Hence to invert the matrix we simply change the sign of the Sin terms.

Flying Sequences

We will now consider the most important application of scene transformation. In any viewer centered application, such as a flight simulator or a computer game, we need to view the scene from a moving position. As the viewpoint changes we transform all the coordinates of the scene such that the viewpoint is the origin and the view direction is the z axis, before drawing the scene. Let us suppose that, in the coordinate system in which the scene is defined we wish to view it from

the point **L** = [Lx,Ly,Lz], looking along the direction **d** = [dx,dy,dz]. The first step is to move the origin to **L** for which we use the transformation matrix ***A***. Following this, we wish to rotate about the y-axis so that **d** lies in the plane x=0. Using the fact that **d** is defined by the co-ordinates [d_x d_y d_z] and using the notation $v^2 = d_x^2 + d_z^2$ this is done by matrix ***B***:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -L_x & -L_y & -L_z & 1 \end{pmatrix} \qquad \boldsymbol{B} = \begin{pmatrix} d_z/v & 0 & d_x/v & 0 \\ 0 & 1 & 0 & 0 \\ -d_x/v & 0 & d_z/v & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Notice that we have avoided computing the Cos and Sin functions for this rotation by use of the direction cosine. To get the direction vector lying along the z axis a further rotation is needed. This time it is about the x axis using matrix ***C***. The different steps of the process are illustrated by

$$\boldsymbol{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & v & d_y & 0 \\ 0 & -d_y & v & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

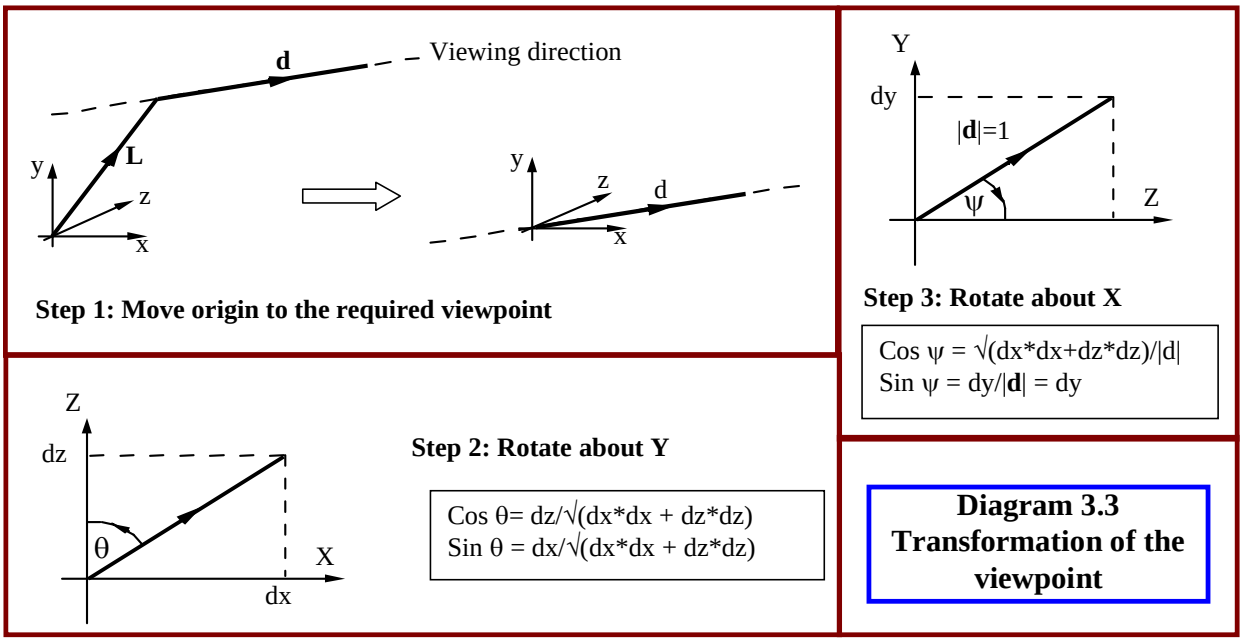


Diagram 3.3

Finally the transformation matrices are combined into one, and each point of the scene is transformed.

T = ***A*** * ***B*** * ***C***
and so for all the points
P = **P** * ***T***

T = ***A*** * ***B*** *C***

Projection by Matrix Multiplication

If we use homogeneous co-ordinates then it is also possible to express projection by the multiplication of a projection matrix. Placing the centre of projection at the origin and using z=f as the projection plane gives us matrix $\boldsymbol{M}_{\mathbf{p}}$ for perspective projection. Matrix $\boldsymbol{M}_{\mathbf{o}}$ is for orthographic projection:

$$\boldsymbol{M}_{\mathbf{p}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/f \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{M}_{\mathbf{o}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It is not immediately obvious that matrix $\boldsymbol{M}_{\mathbf{p}}$ produces the correct perspective projection. Let us transform an arbitrary point **V** with homogeneous co-ordinates [x, y, z, 1] by using matrix multiplication. For the projected point P we get

$$\mathbf{P} = \mathbf{V} * \boldsymbol{M}_{\mathbf{p}} = [x, y, z, z/f]$$

This point may be all right in four dimensions but it is not in a correct form for a 3D point. Homogeneous co-ordinates are defined as the perspective projection in 4D space to the h=1 *sphere* where h is the fourth co-ordinate (**P** = [x, y, z, h]). We use the same projection rules for h we used for z in 3D, i.e., we divide all three co-ordinates by the value of the h co-ordinate and we get:

$$\mathbf{P}_{H} = [x*f/z, y*f/z, f, 1]$$

which is the correct answer. It is interesting to note that the projection matrix is obviously a singular matrix (it has a row of zeros) and, therefore, it has no inverse. This must be so because it is impossible to reconstruct a 3D object from its 2D projection without other information. Projections can of course be combined with the other matrices. Indeed the popularity of the orthographic projection is that it simplifies the amount of calculations since the z row and column fall to zero. Although a simplification applies when the perspective projection is used, there is also the need to normalise the resulting homogenous coordinates, and this adds to the computation time.