Matrix and Vector Manipulation for Computer Graphics

Mike Ducker

November 9, 2000

Contents

1 Summary

The intention of this paper is to provide an introduction to the concepts of matrix and vector manipulation for use in computer graphics systems.
I will work from first principles to give an encompassing view of the necessary techniques for the production and projection of 3D objects, from matrix and vector multiplication, the cartesian co-ordinate systems, point transformations to object, camera and world spaces.

2 An Introduction to Matrices

Matrices are a method of representing related elements in a table format. A matrix consists of a number of elements presented in rows and colums constituting a grid enclosed in brackets. Figure 1 shows a matrix consisting of two columns and three rows.

$$\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix}$$

Fig. 1 : A 3x2 Matrix

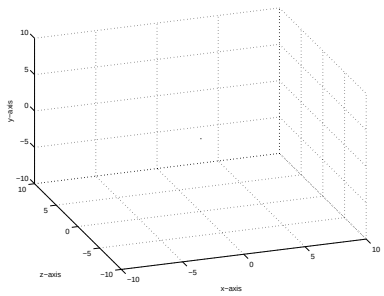
A more general representation for a matrix with m rows and n columns can be seen in figure 2. The standard notation for a matrix of this size is an mxn matrix.

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix}$$

Fig. 2 : An mxn Matrix

3 The Cartesian Coordinate System

The cartesian coordinate system represents any point in 3D space by use of three values to denote the relative position of the point along the x, y and z axes of the system. Each point is considered relative to an origin, which is the centre of cartesian space and located at (0,0,0). The three axes run perpendicular to one another. The x-axis runs from left to right, being negative to the left of the origin and positive to the right. The y-axis runs in the up-down plane, being positive above the origin and negative beneath it and the z-axis runs forward and back, with the space in front of the origin being positive and behind the origin being negative. This particular version of cartesian coordinates is known as the left hand rule. If you raise your left hand and move your digits so that your thumb points directly to your right, your index finger points straight up and your middle finger points forwards, you will be representing the positive axes of cartesian coordinates. There are other versions of the cartesian system which reverse the positive/negative directions of certain axes, most commonly with the z-axis being positive behind the origin, however, all versions keep x,y and z directions in the manner described. Using cartesian coordinates, any point in space can be defined by a unique set of values for each of these axes.



4 Vectors

A vector is a specific type of matrix containing only one column. It represents both the magnitude and direction of some quantity, for instance the velocity of an aircraft or the position of a point in cartesian space.
A standard vector denotes its values using the variables x, y and z to show its magnitude in these axes in the cartesian coordinate system. Figure 3 shows a standard vector.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Fig. 3 : A Standard Vector

Vectors can be manipulated mathematically for addition, subtraction and multiplication. Vector addition involves the separate addition of each axis to produce a new vector.

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 \\ y_1 + y_2 \\ z_1 + z_2 \end{pmatrix}$$

Fig. 4 : Vector addition

Vector subtraction similarly results in the separate subtraction of each axis.

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} - \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{pmatrix}$$

Fig. 5 : Vector subtraction

There are two methods of vector multiplication. One produces a scalar result (a single value containing only the magnitude of the multiplication), the other produces a vector result. The scalar result is obtained by a method known as the dot product and is produced by the addition of all axes once multiplication has occured between the corresponding values on each vector axis.

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = x_1 * x_2 + y_1 * y_2 + z_1 * z_2$$

Fig. 6 : The Dot Product

The second method of vector multiplication is the cross product. This is obtained using the following formula and produces a vector perpendicular to the plane defined by the two multiplied vectors. This 'normal' vector is used in lighting and other graphical, as well as physical, calculations.

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} X \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} y_1 * z_2 - z_1 * y_2 \\ z_1 * x_2 - z_1 * z_2 \\ x_1 * y_2 - y_1 * z_2 \end{pmatrix}$$

Fig. 7 : The Cross Product

5 Matrices and Vectors in the Context of Computer Graphics

The matrices and vectors in computer graphics are of a specialised form to allow for the various equations necessary for 3D manipulation of points.
The vector type used for computer graphics contains four values denoting the x, y and z axes and the homogenous value w. The w value is for use in transforming the world coordinates onto screen coordinates for output onto a 2D monitor. It is initially set to 1. The matrices used to transform points in 3D space are of size 4x4, each of the rows representing a homogenous vector (x,y,z,w). The first three rows contain the world space coordinates of the local x, y and z axes and the fourth row generally contains the values (0,0,0,1). Figure 8 shows the generalised forms of these specialised matrices and vectors.

$$\begin{pmatrix} x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

Fig. 8 : A Standard Matrix and Vector for Computer Graphics

6 Matrix Vector Transformations

To transform a vector by a matrix you have to multiply the vector by the matrix to produce a new vector. The method for doing this is by calculating each point of the new vector as the result of the dot product between that row in the matrix and the vector being transformed.

$$\begin{pmatrix} x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x_v \\ y_v \\ z_v \\ w_v \end{pmatrix} = \begin{pmatrix} x_1 * x_v + y_1 * y_v + z_1 * z_v + w_1 * w_v \\ x_2 * x_v + y_2 * y_v + z_2 * z_v + w_2 * w_v \\ x_3 * x_v + y_2 * y_v + z_3 * z_v + w_3 * w_v \\ 0 * x_v + 0 * y_v + 0 * z_v + 1 * w_v \end{pmatrix} = \begin{pmatrix} x_v^{'} \\ y_v^{'} \\ z_v^{'} \\ w_v^{'} \end{pmatrix}$$

Fig. 9 : Matrix Vector Tranformation

To explain the relationship more clearly figure 10 shows the equation for just the x' resulting from the transformation.

$$x_v^{'} = \begin{pmatrix} x_1 & y_1 & z_1 & w_1 \\ & & & \end{pmatrix} \cdot \begin{pmatrix} x_v \\ y_v \\ z_v \\ w_v \end{pmatrix}$$

$$= \begin{pmatrix} x_1 * x_v + y_1 * y_v + z_1 * z_v + w_1 * w_v \end{pmatrix}$$

7 Matrix Multiplication

While a single point may be transformed by a series of matrix transformations, it is often useful to compose many transformations into one matrix. This allows the same overall transformation to be performed on many points without the computational cost of multiplying many matrices again and again. For instance, instead of having a series of matrices which scale a point by a half in the z plane then move it across by 10 in the x plane before rotating it by 20 degrees in the y plane, you can multiply these matrices together. This creates a single matrix which carries out all the functions simultaneously and which can then be used for any future points needing transforming in the same way. This reduces the overhead from three matrix vector operations to one.
Multiplying two matrices is easily considered if you treat each column of the matrix to be multiplied as a vector. It is then simply the case of multiplying each of these 'vectors' by the matrix to create four new vectors which make up the four columns of the new matrix. The matrix to be multiplied (the multiplicand) is on the right and the matrix to be multiplied by (the multiplier) is on the left of figure 11.

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{pmatrix} * \begin{pmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{pmatrix} = \begin{pmatrix} y'_{11} & y'_{12} & y'_{13} & y'_{14} \\ y'_{21} & y'_{22} & y'_{23} & y'_{24} \\ y'_{31} & y'_{32} & y'_{33} & y'_{34} \\ y'_{41} & y'_{42} & y'_{43} & y'_{44} \end{pmatrix}$$

$$\begin{pmatrix} y'_{11} \\ y'_{21} \\ y'_{31} \\ y'_{41} \end{pmatrix} = \begin{pmatrix} x_{11} * y_{11} + x_{12} * y_{21} + x_{13} * y_{31} + x_{14} * y_{41} \\ x_{21} * y_{11} + x_{22} * y_{21} + x_{23} * y_{31} + x_{24} * y_{41} \\ x_{31} * y_{11} + x_{32} * y_{21} + x_{33} * y_{31} + x_{34} * y_{41} \\ x_{41} * y_{11} + x_{42} * y_{21} + x_{43} * y_{31} + x_{44} * y_{41} \end{pmatrix}$$

$$\begin{pmatrix} y'_{12} \\ y'_{22} \\ y'_{32} \\ y'_{42} \end{pmatrix} = \begin{pmatrix} x_{11} * y_{12} + x_{12} * y_{22} + x_{13} * y_{32} + x_{14} * y_{42} \\ x_{21} * y_{12} + x_{22} * y_{22} + x_{23} * y_{32} + x_{24} * y_{42} \\ x_{31} * y_{12} + x_{32} * y_{22} + x_{33} * y_{32} + x_{34} * y_{42} \\ x_{41} * y_{12} + x_{42} * y_{22} + x_{43} * y_{32} + x_{44} * y_{42} \end{pmatrix}$$

$$\begin{pmatrix} y'_{13} \\ y'_{23} \\ y'_{33} \\ y'_{43} \end{pmatrix} = \begin{pmatrix} x_{11} * y_{13} + x_{12} * y_{23} + x_{13} * y_{33} + x_{14} * y_{43} \\ x_{21} * y_{13} + x_{22} * y_{23} + x_{23} * y_{33} + x_{24} * y_{43} \\ x_{31} * y_{13} + x_{32} * y_{23} + x_{33} * y_{33} + x_{34} * y_{43} \\ x_{41} * y_{13} + x_{42} * y_{23} + x_{43} * y_{33} + x_{44} * y_{43} \end{pmatrix}$$

$$\begin{pmatrix} y'_{14} \\ y'_{24} \\ y'_{34} \\ y'_{44} \end{pmatrix} = \begin{pmatrix} x_{11} * y_{14} + x_{12} * y_{24} + x_{13} * y_{34} + x_{14} * y_{44} \\ x_{21} * y_{14} + x_{22} * y_{24} + x_{23} * y_{34} + x_{24} * y_{44} \\ x_{31} * y_{14} + x_{32} * y_{24} + x_{33} * y_{34} + x_{34} * y_{44} \\ x_{41} * y_{14} + x_{42} * y_{24} + x_{43} * y_{34} + x_{44} * y_{44} \end{pmatrix}$$

Fig. 11 : Matrix Multiplication

There are some important points to keep in mind when dealing with matrix multiplication. Firstly, the multiplication is non-commutative. This means that

$$M * N \neq N * M$$

so the order of multiplication is very important and changing this order will result in a different final matrix. The order in which the matrices are multiplied is from right to left.

$$A * B * C * D = A * B * (C * D)$$

$$C * D = D'$$

$$A * B * C * D = A * B * D'$$

In this example, as shown before, C is the multiplier and D is the multiplicand

An important matrix to consider is the identity matrix. This is a matrix that, when used as the multiplyer or multiplicand, results in the matrix it is being mulitplied with.

$$M * I = I * M = M$$

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 12 : The Identity Matrix

8 Transformation Matrices

Now we have covered the underlying algebra for matrix and vector calculations, it's time to start looking at the different ways matrices can transform points in the 3D world.
There are a number of different matrices, each of which cover a specific type of transformation.

8.1 The Translation Matrix

This matrix moves a point in 3D space by d_x, d_y and d_z relative to the origin of the world coordinates. 'd_x' literally means 'a change in x'. A translation of 50 metres in the x-axis ($d_x = 50$) would move the point 50 metres in the x-axis, independant of where it was relative to the origin. The form of the matrix is

$$\begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 13 : The Translation Matrix

8.2 The Scaling Matrix

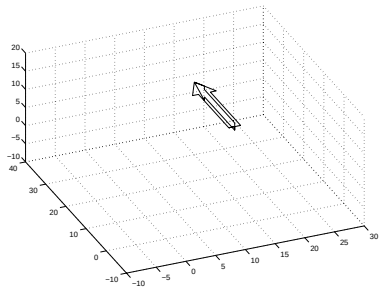
This matrix scales an object relative to the origin of the world coordinates. This means that, if the object is only one metre across but 500 metres from the world origin, it would be scaled relative to the 500 metres in distance from the origin. A scaling of two in all axes would increase the objects dimensions not by a metre but by 500 metres. To avoid this you must translate the object to the origin, scale it and translate it back to its original position.

$$\begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

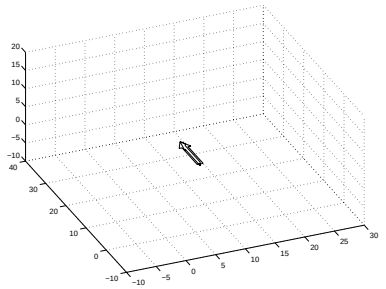
Fig. 14 : The Scaling Matrix

To show the effect of scaling without translation to the world origin the following shows two scalings of an arrow in 2D.

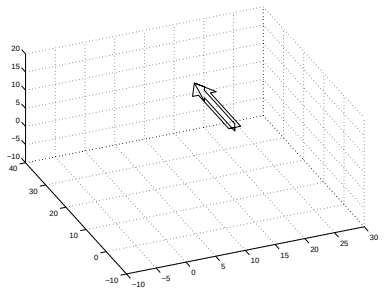
This is the arrow untransformed and centred around position (15,20,10).



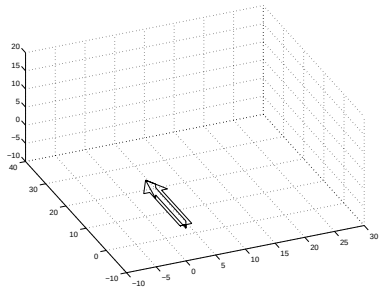
This shows the arrow scaled by 0.5 in all axes. Notice how it has scaled in size correctly but that it's position has changed significantly to (7.5,10,5).



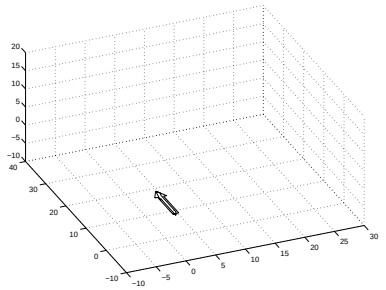
Here the arrow is replaced at its original position and size.



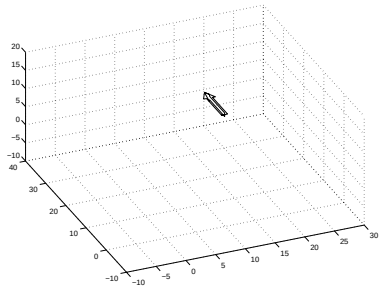
It is now translated to the origin.



Scaled by 0.5.



And translated back to its original coordinates the correct size and in the correct position.



While it is not inconceivable that certain problems may require scaling relative to the world origin, for the majority of situations it is advisable to be careful and to remember the effects of the objects' position when regarding the scaling transformation.

8.3 The Rotation Matrices

There are seperate matrices to rotate points about the x, y and z axes. If you imagine the x-axis to run from your left to your right, then a rotation in the x-axis will result in the object tipping back for positive rotations and forward for negative rotations (imagine holding on to a bar running in the x-axis and how your body rotates when it rotates about that bar). A rotation in the y-axis would result in the object turning around clockwise for positive rotations and anti-clockwise for negative rotations, if viewed from below. A rotation in the z-axis would be similar to tipping the object to its left for negative rotations and to its right for positive ones. If you imagine you are looking down the positive direction of any axis then the positive rotation is in the clockwise direction.
Remember that these rotations are non-commutative, rotating an object in the y-axis before rotating it in the x-axis will have a different result to rotating it in the x-axis before rotating it in the y-axis.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cosx & -sinx & 0 \\ 0 & sinx & cosx & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 15 : Rotation about the X-Axis

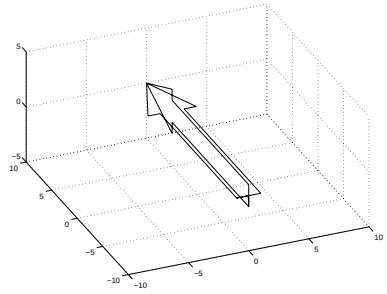
$$\begin{pmatrix} cosy & 0 & siny & 0 \\ 0 & 1 & 0 & 0 \\ -siny & 0 & cosy & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 16 : Rotation about the Y-Axis

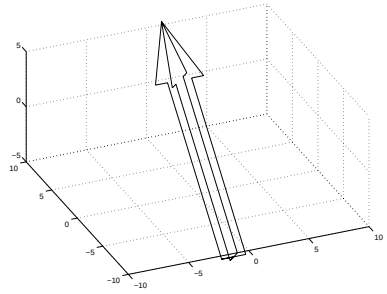
$$\begin{pmatrix} cosz & -sinz & 0 & 0 \\ sinz & cosz & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 17 : Rotation about the Z-Axis

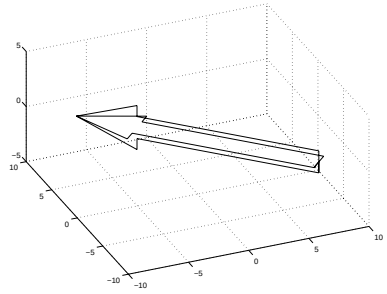
This is the arrow untransformed



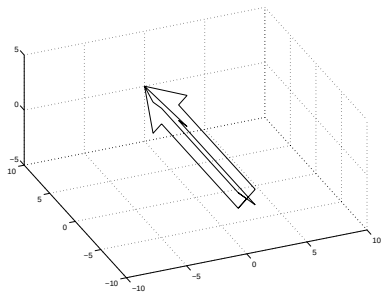
This is the arrow rotated by 45 degrees in the x-axis



This is the arrow rotated by 45 degrees in the y-axis



This is the arrow rotated by 45 degrees in the z-axis



Rotations are carried out with respect to the origin. This means that a rotation about the x-axis will be centred on the origin and the distance of the object from the x-axis i.e. along the y and z axes, will have a profound effect on the final position of the object. The rotation will be the same but if the object is 100 metres along the z-axis and is then transformed by a 90 degree rotation about the x-axis it will end up at -100 metres along the y-axis. You can envisage this by holding you arm out in front of you. Your shoulder is the origin and your hand the object to be rotated. If you rotate your arm by 90 degrees, so that it now points towards the ground, your hand is correctly rotated by 90 degrees but its position compared to the origin is significantly changed. You can overcome this by translating the object to the origin before rotation then translating it back to its starting point once rotated.

8.4 The Perspective Matrix

The perspective matrix is used to transform objects into a three dimensional view relative to their distance from the origin. This takes objects from being flat with a size irrespective of distance along the z-axis and transforms them into realistic, distance dependant objects.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix}$$

Fig. 18 : The Perspective Matrix

The value of d in the perspective matrix is the distance between the point the viewer is looking from (the centre of projection) and the view plane (the vertical plane in front of the viewer which separates the displayed world from the undisplayed world, all objects between the viewer and the view plane are not displayed as if they were behind the viewer).

9 Three State Spaces

There are three internal state spaces that make up the overall geometry of the system. They are object space, world space and camera space.

9.1 Object Space

Each object in the world may have its own subjective object space. The points making up an object are often created relative to a local origin and the parts that constitute the object may have orientations and positions relative to other objects in the hierarchy of that object. This may cause them to be rotated and translated around other objects in the hierarchy relative to the object space of the relative part.
An example of this would be the human arm. Your arm consists of an upper arm, a forearm, a hand and five digits, each of which are made up of two or three joints.
This is a hierarchy as, when you rotate your upper arm around your shoulder joint, the rest of the parts that make up your arm rotate with it. You can separate the parts of your arm into a hierarchy by considering which parts are moved (translated or rotated) due to the movement of other parts. Movement in your upper arm results in movement in your lower arm, your hand and your digits. Movement in your lower arm affects your hand and its digits but not your upper arm. Your hand affects the position and rotation of your digits which have their own individual hierarchies, the last parts of which are the tips of your digits which have no effect on any other part of the hierarchy.
Before any overall transformations can be considered on the arm as a whole, each part must be transformed into the object space of the part above it in the hierarchy. To take your index finger as an example, you must first rotate the tip of that finger into the object space of the middle part of that finger. As the tip can only move within a limited rotation forward and back along the axis of the finger, you have to transform it in that direction by the amount it is rotated relative to the middle part (this would probably be only a few degrees). Following this you must rotate the finger tip along with the middle part around the middle joint into the object space of the first part of the finger. All three parts of your index finger are then rotated around the knuckle into hand space. Your hand and all its digits (notice how the rotation of your index finger did not change the rotation of any other fingers although they are equally low in the hierarachy) are translated into forearm space. This can involve all the parts being rotated, within limits, in all three axes of rotation. Your forearm and hand are then rotated around the elbow joint (only in one direction again) into upper arm space and finally your entire arm is rotated around the shoulder joint into body space.

9.2 World Space

World space is the absolute space in which every object and camera in the system lies. Its central origin is the central origin about which rotation and scaling transformations take place. To continue with the above example, once all object transformations have taken place, the entire of the object (in this case your body) is transformed into world space. Imagine if you were seated in the cockpit of a plane. Once each part of your body had been rotated so that it was at the 'sitting' orientation, it would then further have to be translated and rotated into the cockpit seat. The cockpit (and the rest of the plane) could then be translated and rotated to its position in world space i.e. a particular portion of the sky, the runway or a hanger. World space is itself subjective to the needs of the world being modelled. You could continue to transform the pilot, the plane, the runway and even the Earth into solar system space. This in turn could be transformed into Galaxy space before everything was transformed into Universe space.
World space simply represents the highest possible point in a hierarchy about which all other objects and spaces are relative. In some simulations this can be a human body, a single cell or a group of atoms. The important point is that world space is not relative to anything else. Everything is relative to world space.

9.3 Camera Space

Camera space represents the eye of the viewer. This is a subjective view of the world and it is relative to world space. However, it is the last transformation necessary before the image is put to screen so I have left it until here to explain. Once all other transformations have taken place and all objects are in world space, the entire world is transformed into camera space. All objects are now relative to the viewer and the final stages of image projection can take place. In terms of computation, the way camera space is dealt with is that any movement of the camera, for instance a +10 translation in the z-axis, results in a multiplication to the camera matrix of the opposite magnitude i.e. -10 in the z-axis. This is so that, when objects are transformed into camera space, they are moved relative to the camera, in the opposite direction, to give the impression of the camera moving towards them. Because there is no actual camera in the world, the world is moved inversely relative to the origin where the final image is taken from.
To envisage this, hold an object 30cm in front of you. Now move your head towards the object by 10cm. This is the same as moving the object towards you by 10cm. As there is no physical camera (the equivilant of your eye) because the image is taken from the world origin, we have to move the object (in fact, the entire world) instead. This results in the inverse transformations being carried out on the object, rather than the actual transformations being carried out on the camera.

9.4 Finally

Firstly objects are transformed into their object space and, incrementally, into the object space of all objects higher on their hierarchies. Then the object, in it's entirety, is transformed into world space and, finally, all of world space is transformed into camera space. Once this is carried out a perspective transformation is applied (which may be part of the camera transformation) and the scene is placed onto the screen. This is where the vectors' homogenous coordinate comes into play. The perspective projection alters the value of w from 1. To change the point in space from 3D to 2D you simply divide the x and y coordinates by w to give their 2D vector. Before each point is placed onto screen a process called culling is carried out. This decides which points can be seen by the screen and which points are outside it, too far away in the distance or behind the camera. This is done automatically in some rendering systems (where you simply pass the transformed object) but most systems, if asked to plot an object which is partially offscreen, will happily oblige. In general, calls to display a pixel which is offscreen will simply return from the function with no ill effect (in fact, no effect whatsoever). The main point in culling objects which are out of view is to reduce computation, as it is more computationally expensive to draw an object offscreen than it is to calculate that it is offscreen and ignore it.

10 Example Questions

10.1 Matrix Transformations

1) Produce a matrix which will scale an object to half its current size.
2) Create a single matrix which will rotate an object about the x-axis by 45 degrees before translating by (30,40,0).

10.2 Matrix Vector Transformations

3) Rotate a point (10,20,15) by -30 degrees around the z-axis followed by 90 degrees in the x-axis.
4) Rotate a point (10,20,15) by 90 degrees in the x-axis followed by -30 degrees in the z-axis.
5) Translate a point at the origin to (0,0,50), now rotate it by -45 degrees in the x-axis.
6) How could you achieve the same result using only translation matrices. Write out the matrix.

11 Answers To Example Questions

1)

$$\begin{pmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2)

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(45) & -sin(45) & 0 \\ 0 & sin(45) & cos(45) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.7071 & -0.7071 & 0 \\ 0 & 0.7071 & 0.7071 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T = \begin{pmatrix} 1 & 0 & 0 & 30 \\ 0 & 1 & 0 & 40 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R * T = \begin{pmatrix} 1 & 0 & 0 & 30 \\ 0 & 0.7071 & -0.7071 & 40 \\ 0 & 0.7071 & 0.7071 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3)

$$V = \begin{pmatrix} 10 \\ 20 \\ 15 \\ 1 \end{pmatrix}$$

$$RZ = \begin{pmatrix} cos(-30) & -sin(-30) & 0 & 0 \\ sin(-30) & cos(-30) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.866 & 0.5 & 0 & 0 \\ -0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$RX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(90) & -sin(90) & 0 \\ 0 & sin(90) & cos(90) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$RZ' = RX * RZ = \begin{pmatrix} 0.865 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.5 & 0.865 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$V' = RZ' * V \begin{pmatrix} 18.649 \\ 15.0 \\ 12.299 \\ 1 \end{pmatrix}$$

4)

$$V = \begin{pmatrix} 10 \\ 20 \\ 15 \\ 1 \end{pmatrix}$$

$$RX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(90) & -sin(90) & 0 \\ 0 & sin(90) & cos(90) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$RZ = \begin{pmatrix} cos(-30) & -sin(-30) & 0 & 0 \\ sin(-30) & cos(-30) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.866 & 0.5 & 0 & 0 \\ -0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$RX' = RZ * RX = \begin{pmatrix} 0.865 & 0 & 0.5 & 0 \\ -0.5 & 0 & 0.865 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$V' = RX' * V \begin{pmatrix} 16.149 \\ 7.975 \\ 20.0 \\ 1 \end{pmatrix}$$

5)

$$V = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 50 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(45) & -sin(45) & 0 \\ 0 & sin(45) & cos(45) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.7071 & 0.7071 & 0 \\ 0 & -0.7071 & 0.7071 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T' = R * T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.7071 & 0.7071 & 35.354 \\ 0 & -0.7071 & 0.7071 & 35.354 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$V' = RZ' * V \begin{pmatrix} 0 \\ 35.354 \\ 35.354 \\ 1 \end{pmatrix}$$

6) As the translation is along the z-axis at 45 degrees in the x you can use trigonometry to calculate its position. Sin (45) * length gives you the magnitude in the y plane and Cos (45) * length gives you the magnitude in the z plane. $\sin(45) = 0.7071$ and $\cos(45) = 0.7071$ So the magnitude in the y plane = 35.354 as is the magnitude in the z plane. The translation matrix to achieve this is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 35.354 \\ 0 & 0 & 1 & 35.354 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

12 Further Reading

For more information on the fundamentals of computer graphics "Introduction to computer graphics" [1] covers all aspects from the basic matrix-vector transformations shown here to fully rendered, lit, shaded scenes. To test your knowledge of matrix multiplication try http://www.mkaz.com/math/matrix.html for a matrix calculator.

13 Notes

In the production of this piece of coursework I extensively used the tutorial by Z. Mortensen [2] as an outline for content. The majority of my knowledge of matrices and their use in computer graphics was procurred from [1].
I used LaTeX to produce the textual content and MatLab for the arrow graphics.

References

[1] D. Foley et al. :
Introduction to computer graphics
(1993)
Addison-Wesley Pub Co; ISBN: 0201609215

[2] Z. Mortensen :
Matrix transformation tutorial
(1995)
http://www.strangecreations.com/library/graphics/matrix.txt