Computer Graphic with Matrices
Math 308A Project

Student: Wei-Liang Chen — Student #: 0030292

Date: Dec 3rd, 2001 — Professor: James King

Imagine that you are drawing a picture by hand or computer. The picture we draw by computer are called computer graphic. Computer graphic can divide into 2 dimensional (2-D) and 3 dimensional (3-D). 2-D graphic is usually like anime, comic, or any kind of drawing that is on a paper. The games we see nowadays are usually 3-D, just like Diablo II or Baldur's Gate. The difference between 2-D and 3-D are 2-D is flat and 3-D is like solid, so as you look over them by a camera. 2-D graphic will just be the same; if the graphic is the front of the person, you won't able to see his back. However, in a 3-D graphic, you can move the camera around. You will be able to see the person's back if you move camera behind that person. You can even see from the top or the bottom of the person. For here, I am going to talk about the matrix relates to the 3-D computer graphic, but it will also include some detail about 2-D graphic too.

As we draw or look at 3-D graphic, the structure includes four basic things: polygon, pointer, vertex and edge. The relationship of these four is show as the table below:

Polygon list	Edge List		Vertex list
edges	Polygons	vertices	coordinates
p1: e1 e2 e3	e1: p1	v1 v4	v1: x_1 y_1 z_1
p2: e3 e4 e5	e2: p1	1 2	v2: x_2 y_2 z_2
	e3: p1 p2	v2 v4	v3: x_3 y_3 z_3
	e4: p2	v2 v3	v4: x_4 y_4 z_4
	e5: p2	v3 v4	

It might take some time to understand what this table means is, but what it means here is polygons are the pointers of the edge list and pointers are held in clockwise order from the front. For the containing polygons and vertices, edges are their pointers. Each edge is also in the lists for two polygons. For vertex, they can only be keep once for their coordinates. With all these, it means we are able to move the graph easily, which means we are able to change vertices, edge and polygons as we want to.

In a 3-D graph, even we have the structure of the thing we need to draw, we still need to do something we called transformations to make our 3-D model into the final destination. In order to make the 3-D model into our final destination, there are three parts to do. First is to scale the object into the correct size, second is rotate it to the correct orientation, then the last is to translate to the final destination. This is where 2-D or 3-D graphic involves in matrix. You might think that 2-D graph used 2x2 matrices and 3-D graph used 3x3 matrices. If you think of this, then you are totally wrong. 2-D graph actually used 3x3 matrices and 3-D graph used 4x4 matrices. The reason is we need the extra matrix to represent 2-D or 3-D translation, scaling and rotation. This solution or theory that is used in transformation is called <u>homogeneous coordinates</u>.

Scaling in 3-D:

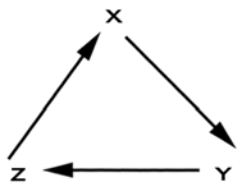
Assume we have a matrix for scaling s_x in direction x.

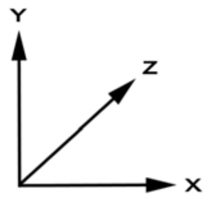
$$S = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{The inverse is } S^{-1} = \begin{pmatrix} 1/s_x & 0 & 0 & 0 \\ 0 & 1/s_y & 0 & 0 \\ 0 & 0 & 1/s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

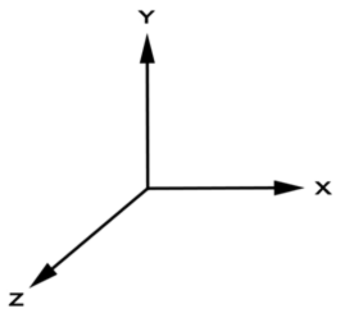
This is how we did scaling as long as the *s*s in the matrix is not zero. When one of s in the matrix is zero, this means the dimension of this object is lower.

Rotation in 3-D:

Remember we have learned the rotation in 2-D before in the class. However, 3-D rotation is much harder because we have another axis z. Let’s recall the rotation matrix for 2-D, which is $\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$. This matrix works well on 2-D but also 3-D. Just the rotation on 2-D is by point and 3-D is by axis. The hard thing when you rotate in 3-D is to determine which side is positive or negative. Here are three ways to determine the positive or negative on a 3-D object:







Left handed Right handed

I am not really sure about the first way. I have asked my friend who works on computer graphic. He also has no idea about it. Second one will a bit confused unless you really try with your left hand. The most popular one that is used by people was the right handed rule. It is also easy recognizable if you look at a 3-D perfect square too. Here is another list of what's the direction of positive when you rotate the axis you need:

Rotation axis	Direction of positive rotation
x	from y towards z
y	from z towards x
z	from x towards y

This table gives people a clear mind which way is positive and negative. With this direction table, we were able to find the rotational matrix for each axis. The matrices for these three main axes are:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_z = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

These three matrices are used on the rotation when there is a point on the axis and the angle it rotates. Here is an example of how these matrices work.

Example: Let P = (0, 0, 1) in a right-handed system. Apply a rotation about the y-axis of +90°.

$$R_y = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad M = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

We got the matrix, and then we are able to get the homogenous form:

$$PM = \begin{pmatrix} 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 1 \end{pmatrix}.$$

So the point P after rotation will be (-1, 0, 0).

Translation in 3-D:

Some of you must wonder why the fourth row is always zero except the last one is 1. This row is used for translation. If said we want to translate by (t_x, t_y, t_z), the matrix we are calculating will be

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{pmatrix}.$$

The point will be transform as

$$\begin{pmatrix} x & y & z & 1 \end{pmatrix} T = \begin{pmatrix} x + t_x y + t_y z + t_z 1 \end{pmatrix}$$

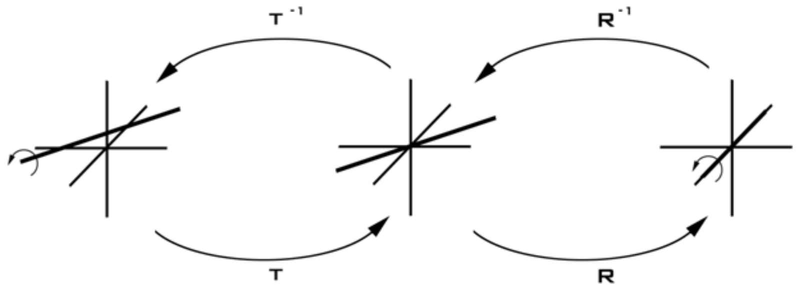
Now you can actually see the relationship between scaling, rotation and translation. We can combine them and form a new matrix:

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ t_x & t_y & t_z & 1 \end{pmatrix}.$$

The r_s in the matrix are used in scaling and rotation, and t_s are used for translation. Normally, scale is set to one so the r values will just be pure rotation, no need to worry about the scaling. As I mention before, the modeling process is scale, rotate and then translate. It can also do in the backward way for other situation. In the situation of viewing, the process is translated, rotate then scale. This situation only works on viewing because we are actually moving the viewer here, so the viewer's direction is actually opposite the object's direction.

Rotation about Arbitrary Axes:

This is the last process after you done modeling the object, what happened if the object you model is not on the x-axis you want to put on? The last thing we do here is rotate the object about Arbitrary axes. The step for rotating model in arbitrary axe is actually just like transformation, but we don't need scaling here. All we need is translation and rotation. First, we translate the model so it passes through the origin. Then we rotate it so it will lie on the axis you want, assume z-axis. This process can also be reverse if you want to find the original state of the modeling. Below is a picture of a short summary of rotation about arbitrary axes:



All of these transformation, rotation, translation, scaling are all how matrix applies on computer graphic. Arbitrary axes sometime can also give nasty answer due to the process you need to calculate can result a matrix that looks nasty. It is better you try to avoid this kind of situation by not using this rotation much.

Here is an example which will make everyone clear up on what I refer in this project. This examples show the calculation of transformation and rotation of 2D. (This example is taken out from the source I used to write up this project)

Example:

Find the 3x3 matrix that corresponds to the composite transformation of a scaling by 3, a rotation by 90^o, and finally a translation that adds (-5, 2) to each point of a figure.

Solution: If $\varphi = \pi/2$, then $\sin\varphi = 1$, and $\cos\varphi = 0$.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \xrightarrow{Scale} \begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\xrightarrow{Rotate} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\xrightarrow{Translate} \begin{pmatrix} 1 & 0 & -0.5 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

The matrix for the composite transformation is

$$\begin{pmatrix} 1 & 0 & -0.5 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & -0.5 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -0.3 & -0.5 \\ 0.3 & 0 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

This is the calculation, hopefully you can get it. If you are not familiar with 3-D calculation, example 7 from text book Linear Algebra and Its Applications in section 2.8 will be a good one to look at it, I hope the thing I explain here does make people understand the relationship between matrix and computer graphic. And I hope this can also help you understand more about the matrix we are learning in the class now.

Bibliography:

#1 Source: www.bath.ac.uk/~maspjw/ch3-surf-models.pdf, 10 pages.

#2 Source: Linear Algebra And Its Applications, David C. Lay, Section 2.8 (Applications to Computer Graphics), reprinted with correction, April 2000.