Matrices in Computer Graphics

Ting Yip
Math 308A
12/3/2001

Ting Yip
Math 308A

Abstract

In this paper, we discuss and explore the basic matrix operation such as translations, rotations, scaling and we will end the discussion with parallel and perspective view. These concepts commonly appear in video game graphics.

Introduction

The use of matrices in computer graphics is widespread. Many industries like architecture, cartoon, automotive that were formerly done by hand drawing now are done routinely with the aid of computer graphics. Video gaming industry, maybe the earliest industry to rely heavily on computer graphics, is now representing rendered polygon in 3-Dimensions.

In video gaming industry, matrices are major mathematic tools to construct and manipulate a realistic animation of a polygonal figure. Examples of matrix operations include translations, rotations, and scaling. Other matrix transformation concepts like field of view, rendering, color transformation and projection. Understanding of matrices is a basic necessity to program 3D video games.

Graphics

(Screenshots taken from Operation Flashpoint)
Polygon figures like these use many flat or conic surfaces to represent a realistic human soldier.

Homogeneous Coordinate Transformation

Points (x, y, z) in $\mathbb{R}^3$ can be identified as a homogeneous vector $(x,y,z,h) \rightarrow \left(\frac{x}{h},\frac{y}{h},\overset{\text{The last coordinate is a scalar term.}}{\frac{z}{h}},1\right)$ with $h \neq 0$ on the plane in $\mathbb{R}^4$. If we convert a 3D point to a 4D vector, we can represent a transformation to this point with a 4 x 4 matrix.

EXAMPLE

Point (2, 5, 6) in R^3 $\mapsto$ Vector (2, 5, 6, 1) or (4, 10, 12, 2) in R^4

NOTE

It is possible to apply transformation to 3D points without converting them to 4D vectors. The tradeoff is that transformation can be done with a single matrix multiplication after the convertion of points to vectors. (More on this after Translation.)

Transformation of Points

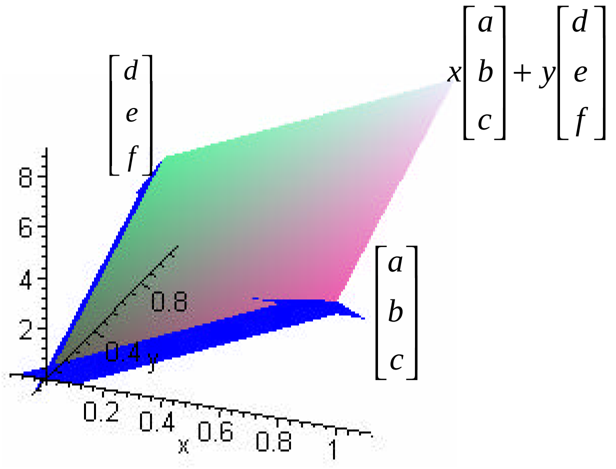
In general, transformation of points can be represented by this equation:

$$\text{Transformed Point} = \text{Transformation Matrix} \times \text{Original Point}$$

In a more explicit case, a plane spanned by two vectors can be represented by this equation:

$$\begin{aligned}\text{Transformed Plane} &= \text{Transformation Matrix} \times \text{Original Plane}\\ &= \text{Transformation Matrix} \times \text{span}\left\{\begin{bmatrix}a\\b\\c\end{bmatrix}, \begin{bmatrix}d\\e\\f\end{bmatrix}\right\}\\ &= \text{Transformation Matrix} \times \begin{bmatrix}a & d\\ b & e\\ c & f\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}\end{aligned}$$

x and y are scalars



Representation of a plane using matrices

Translation

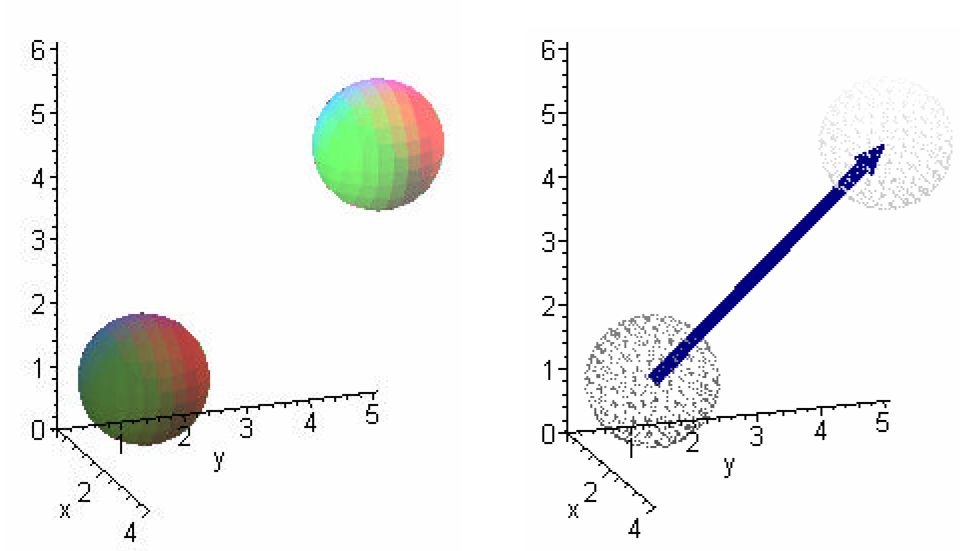
A translation basically means adding a vector to a point, making a point transforms to a new point. This operation can be simplified as a translation in homogeneous coordinate (x, y, z, 1) to (x + t_x, y + t_y, z + t_z, 1). This transformation can be computed using a single matrix multiplication.

Translation Matrix for Homogeneous Coordinates in R^4 is given by this matrix:

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Given any point (x, y, z) in R^3, the following will give the translated point.

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix}$$



For a sphere to move to a new position, we can think of this as all the points on the sphere move to the translated sphere by adding the blue vector to each point.

Graphics

(Screenshots taken from Operation Flashpoint)
In video game, objects like airplane that doesn't change its shape dynamically (rigid body) uses Translation to move across the sky. All the points that make up the plane have to be translated by the same vector or the image of the plane will appear to be stretched.

NOTE

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} x+i \\ y+j \\ z+k \end{bmatrix}$$

If we have more than one point, we would have to apply this addition to every point.

$$\begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \\ z_1 & z_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} x_1+i \\ y_1+j \\ z_1+k \end{bmatrix} \& \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} + \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} x_2+i \\ y_2+j \\ z_2+k \end{bmatrix} \mapsto \begin{bmatrix} x_1+i & x_2+i \\ y_1+j & y_2+j \\ z_1+k & z_2+k \end{bmatrix}$$

With homogeneous coordinate, we can use a single matrix multiplication.

$$\begin{bmatrix} 1 & 0 & 0 & i \\ 0 & 1 & 0 & j \\ 0 & 0 & 1 & k \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \\ z_1 & z_2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} x_1+i & x_2+i \\ y_1+j & y_2+j \\ z_1+k & z_2+k \\ 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} x_1+i & x_2+i \\ y_1+j & y_2+j \\ z_1+k & z_2+k \end{bmatrix}$$

As we can see, linear system is easier to solve with homogenenous coordinate transformation.

Scaling

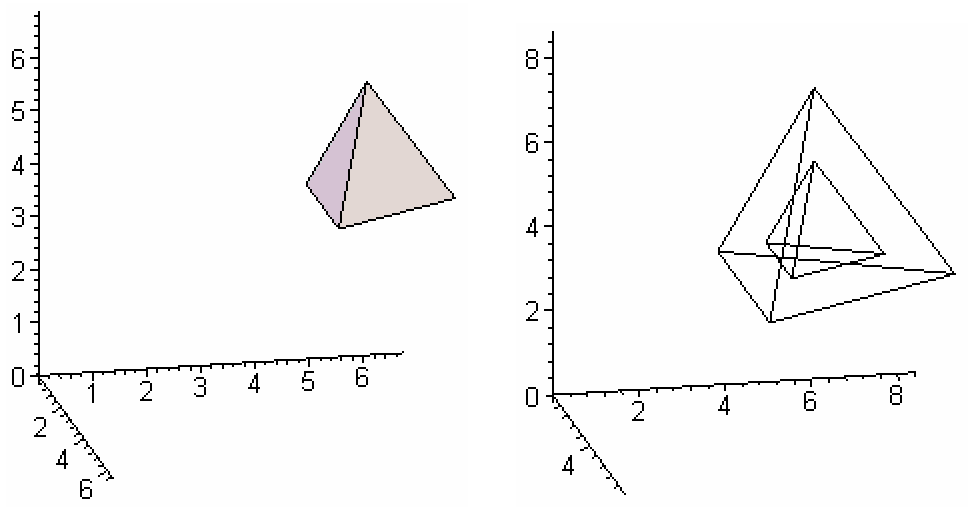
Scaling of any dimension requires one of the diagonal values of the transformation matrix to equal to a value other than one. This operation can be viewed as a scaling in homogeneous coordinate (x, y, z, 1) to (s_xx, s_yy, s_zz, 1). Values for s_x, s_y, s_z greater than one will enlarge the objects, values between zero and one will shrink the objects, and negative values will rotate the object and change the size of the objects.

Scaling Matrix for Homogeneous Coordinates in R^4 is given by this matrix:

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Given any point (x, y, z) in R^3, the following will give the scaled point.

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ s_z z \\ 1 \end{bmatrix}$$



If we want to scale the hexahedron proportionally, we apply the same scaling matrix to each point that makes up the hexahedron.

Rotations

Rotations are defined with respect to an axis. In 3 dimensions, the axis of rotation needs to be specified.

A rotation about the x axis is represented by this matrix:

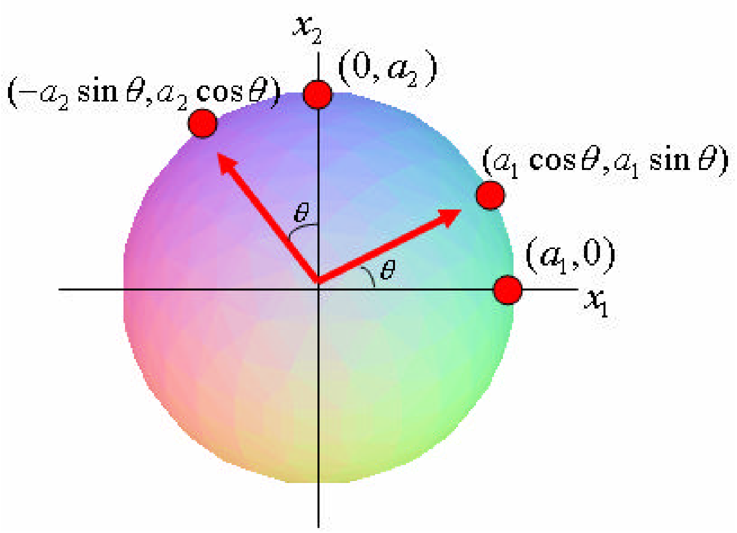
$$R_x(q) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos q & -\sin q & 0 \\ 0 & \sin q & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow R_x(q)\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos q & -\sin q & 0 \\ 0 & \sin q & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y\cos q - z\sin q \\ y\sin q + z\cos q \\ 1 \end{bmatrix}$$

A rotation about the y axis is represented by this matrix:

$$R_y(q) = \begin{bmatrix} \cos q & 0 & \sin q & 0 \\ 0 & 1 & 0 & 0 \\ -\sin q & 0 & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow R_y(q)\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos q & 0 & \sin q & 0 \\ 0 & 1 & 0 & 0 \\ -\sin q & 0 & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos q + z\sin q \\ y \\ -x\sin q + z\cos q \\ 1 \end{bmatrix}$$

A rotation about the z axis is represented by this matrix:

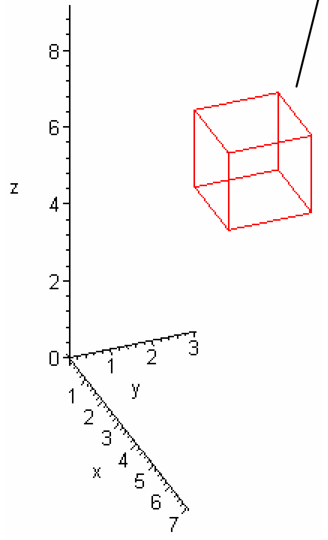
$$R_z(q) = \begin{bmatrix} \cos q & -\sin q & 0 & 0 \\ \sin q & \cos q & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow R_z(q)\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos q & -\sin q & 0 & 0 \\ \sin q & \cos q & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos q - y\sin q \\ x\sin q + y\cos q \\ z \\ 1 \end{bmatrix}$$


x_2
$(0, a_2)$
$(-a_2 \sin\theta, a_2\cos\theta)$
$(a_1\cos\theta, a_1\sin\theta)$
θ
θ
$(a_1, 0)$
x_1

3D rotation can be viewed as replacing x_1 and x_2 with two axes.

This wire polygon cube is represented by a matrix that contains its vertex point in every column.

$$\begin{bmatrix} 5 & 5 & 7 & 7 & 5 & 7 & 5 & 7 \\ 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 \\ 9 & 7 & 7 & 9 & 7 & 7 & 9 & 9 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

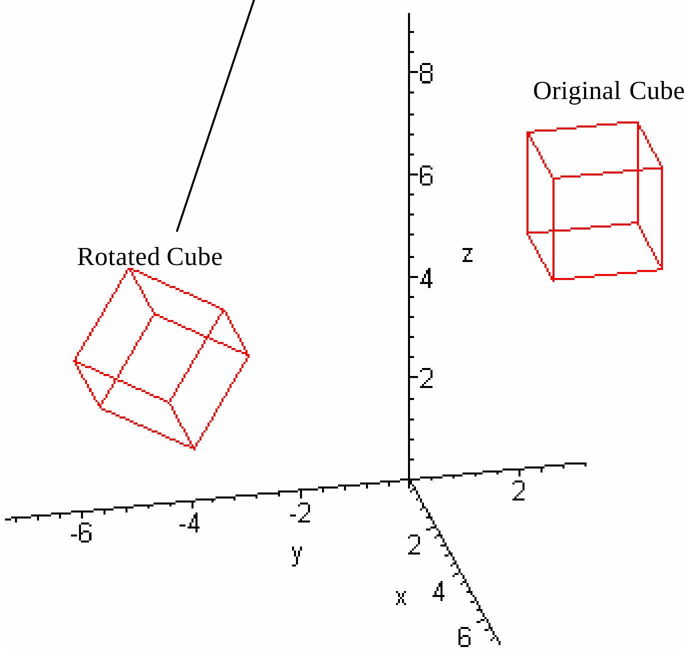


EXAMPLE

If we want to rotate this cube with respect to the x axis by $\left(\frac{p}{3}\right)$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{p}{3}\right) & -\sin\left(\frac{p}{3}\right) & 0 \\ 0 & \sin\left(\frac{p}{3}\right) & \cos\left(\frac{p}{3}\right) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 5 & 7 & 7 & 5 & 7 & 5 & 7 \\ 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 \\ 9 & 7 & 7 & 9 & 7 & 7 & 9 & 9 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 5 & 5 & 7 & 7 & 5 & 7 & 5 & 7 \\ \frac{1}{2}-\frac{9}{2}\sqrt{3} & \frac{1}{2}-\frac{7}{2}\sqrt{3} & \frac{1}{2}-\frac{7}{2}\sqrt{3} & \frac{1}{2}-\frac{9}{2}\sqrt{3} & \frac{3}{2}-\frac{7}{2}\sqrt{3} & \frac{3}{2}-\frac{7}{2}\sqrt{3} & \frac{3}{2}-\frac{9}{2}\sqrt{3} & \frac{3}{2}-\frac{9}{2}\sqrt{3} \\ \frac{9}{2}+\frac{1}{2}\sqrt{3} & \frac{7}{2}+\frac{1}{2}\sqrt{3} & \frac{7}{2}+\frac{1}{2}\sqrt{3} & \frac{9}{2}+\frac{1}{2}\sqrt{3} & \frac{7}{2}+\frac{3}{2}\sqrt{3} & \frac{7}{2}+\frac{3}{2}\sqrt{3} & \frac{9}{2}+\frac{3}{2}\sqrt{3} & \frac{9}{2}+\frac{3}{2}\sqrt{3} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



Projection Transformation

Even though we programmed objects in 3-Dimensions, we have to actually view the objects as 2-Dimensions on our computer screens. In another word, we want to transform points in R^3 to points in R^2.

Parallel Projection

In parallel projection, we simply ignore the z-coordinate. This operation can be viewed as a transformation in homogeneous coordinate (x, y, z, 1) to (x, y, 0, 1).

Parallel Matrix for Homogeneous Coordinates in R^4 is given by this matrix:

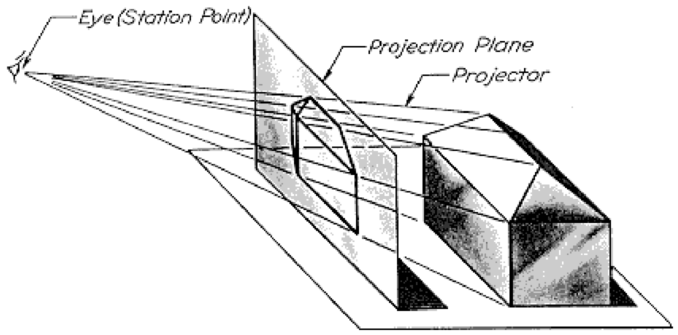
$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Given any point (x, y, z) in R^3, the following will give the parallel projected point.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

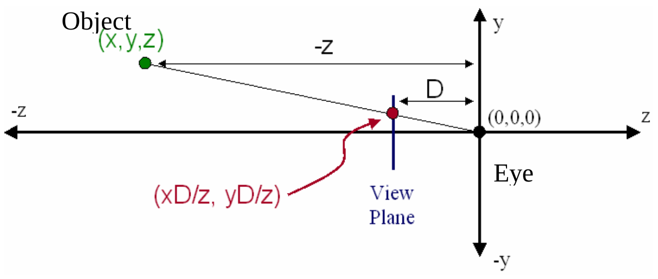
Perspective Projection

Video game tends to use perspective projections over other projections to represent a real world, where parallelism is not preserved. Perspective Projections is the way we see things, i.e. bigger when the object is closer.



http://mane.mech.virginia.edu/~engr160/Graphics/Perspective.html

Important:
1) Translate the eye to the Origin
2) Rotation until direction of eye is toward the negative z-axis



D is the distance of the eye to the view plane
z is the distance of the eye to the object
(Note: not your "eyes" but the eyes of the computer polygon person)

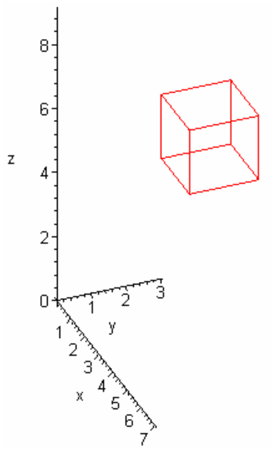
Perspective Matrix for Homogenous Coordinates in $\mathbb{R}^4$ is given by this matrix:

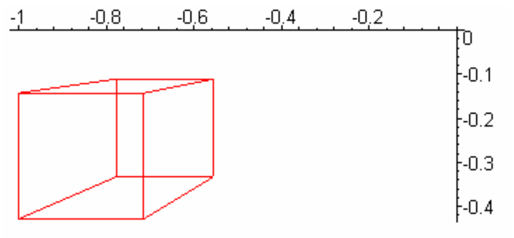
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}$$

Given any point (x, y, z) in $\mathbb{R}^3$, the following will give the parallel projected point.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ \frac{z}{d} \end{bmatrix} \Rightarrow \begin{bmatrix} x\frac{z}{d} \\ y\frac{z}{d} \\ 0 \\ 1 \end{bmatrix}$$

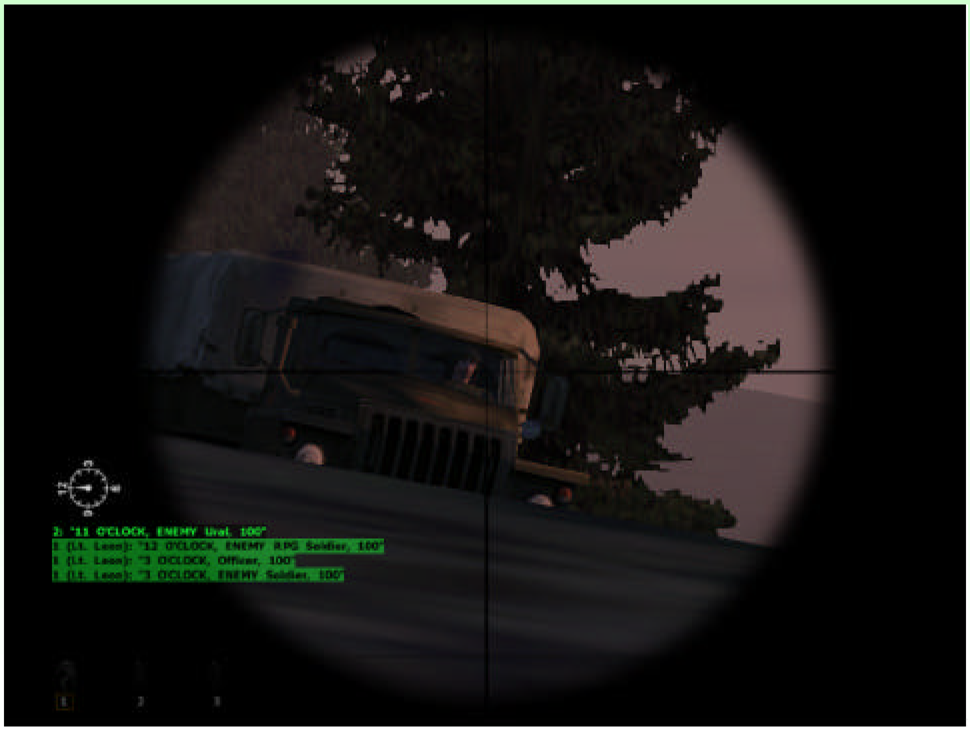
(Note: This matrix transformation does not give pixel coordinate on the monitor. The transformed coordinate is with respect to the object's coordinate. We have to translate the object's coordinate to pixel coordinate on the monitor.)





Perspective Projection of the cube

Graphics


2: "11 O'CLOCK, ENEMY Ural, 100"
1 (Lt. Leon): "12 O'CLOCK, ENEMY RPG Soldier, 100"
1 (Lt. Leon): "3 O'CLOCK, Officer, 100"
1 (Lt. Leon): "3 O'CLOCK, ENEMY Soldier, 100"

(Screenshots taken from Operation Flashpoint)
In sniping mode, the eye moves closer to the object.

Conclusion

I chose to do this project to show my curiosity in math and computer science. I had the chance to talk about video games and math that are often overlooked as unrelated. As shown in this project, *Linear Algebra* is extremely useful for video game graphics. Using matrices to manipulate points is a common mathematical approach in video game graphics.

References

Dam, Andries. "Introduction to Computer Graphics"
http://www.cs.brown.edu/courses/cs123/lectures/Viewing3.pdf

Holzschuch, Nicola. "Projectio ns and Perspectives"
http://www.loria.fr/~holzschu/cours/HTML/ICG/Resources/Projections/index.html

Lay, David. *Linear Algebra and its Applications.* Second Edition.

Runwal, Rachana. "Perspective Projection"
http://mane.mech.virginia.edu/~engr160/Graphics/Perspective.html