Math For Computer Graphics

Harry Han

March 23, 2023

1 Linear Transformations

A point (or vertex) in three dimensional space can be represented by the following vector: $\boldsymbol{v} = [x_0, y_0, z_0, 1]^T$. The extra dimension is solely for the convenience of matrix manipulation.

1.1 Translation, Rotation

Translation of a vertex $\boldsymbol{v} = [x_0, y_0, z_0, 1]^T$ in x direction for t_x, y direction for t_y, and z direction for t_z is a linear transfomation, whose corresponding matrix is T:

$$T\boldsymbol{v} = \begin{bmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = [x_0 + x_t, y_0 + y_t, z_0 + z_t, 1]^T \tag{1}$$

The matrices R_x, R_y, R_z that correspond to the rotations of a vertex $\boldsymbol{v} = [x_0, y_0, z_0, 1]^T$ respect to x, y, z for θ degrees are:

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The Scaling matrix S is:

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

1.2 Projection

1.3 Perspective Projection

To transform a vertex $\boldsymbol{v} = [x_0, y_0, z_0]$ into its proper coordinate for the projection with viewing angle θ, use the projection matrix:

$$P = \begin{bmatrix} \frac{1}{z\tan\theta/2} & 0 & 0 & 0 \\ 0 & \frac{1}{z\tan\theta/2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

However, for implementation in OpenGL, it is useful to have a constant matrices. Since during rasterisation all coordinate of the vetices are divided by w, i.e., the value for the fourth dimension, we can implement the perspective projection matrix in openGL as follow:

$$P_{GL} = \begin{bmatrix} \frac{1}{\tan\theta/2} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan\theta/2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5}$$

Recall matrix under mutiplication is not an abelien group, i.e., matrix multiplication is associative, although not commutative. To tranform a vertex $\boldsymbol{v}$ with scaling, rotation, and translation, the order of the matrix is important:

$$P_{GL} T R_x R_y R_z S \boldsymbol{v} \tag{6}$$

OpenGL will normalize all the vertices before rendering to screen, which means if the setted screen is not a square, the final rendering will be distorted.

2 Colors in Fragment Shader

2.1 Interpolation

Assuming a triangle have three vertices A, B, C with arbitraily assigned coordinates $(1,0,0), (0,1,0), (0,0,1)$. Barycentric Coordinate system can assign every points inside the triangle with coordinates (a, b, c) such that $a+b+c = 1$ and value of a, b, c are inversely proportional to its distance from vertices A, B, C.

Algorithm 2.1 (Barycentric Coordinates).

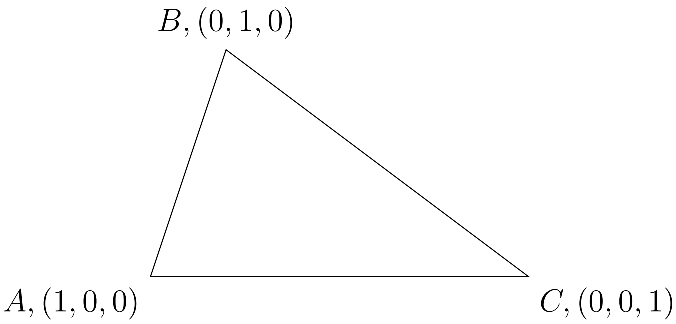

Figure 1: Barycentric Coordinate