CS250 Affine & Orientation

By Sungmin Moon

KMU-Digipen Sophomore

Affine Transformation

When controlling 3D objects, a 4 X 4 matrix is usually used.

The most simple form of transform matrix is the identity matrix, which takes the form of

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

This applies to matrices of other sizes as well, in the form of $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ in 2X2, and $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ in 3X3.

If we multiply a vector to this matrix, the vector is left completely unharmed, like the following process.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 \\ 1 \cdot 2 \\ 1 \cdot 3 \\ 1 \cdot 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Based on this identity matrix, we change some of these values to control three aspects of 3D transformation, scale, translation and rotation.

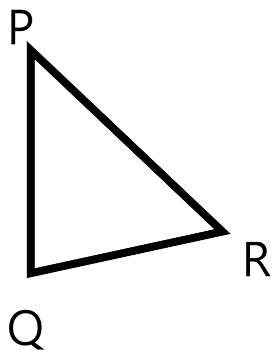
The fourth component - w - is known as a homogeneous coordinate. It is the key to controlling the translations of 3D vectors, and is the main reason why we use a 4 X 4 matrix.

Orientation

An orientation vector, also called as the normal vector, has the following attributes.

- Orthogonal to the plane containing the triangle.
- Points towards the front of the triangle.

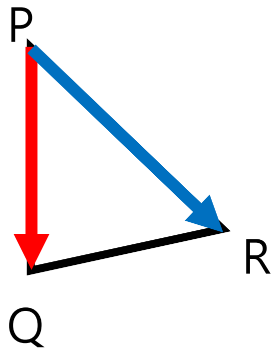
So, it is used to differentiate the front and back face of a triangle.



If the triangles' vertexes are defined in a clockwise/counterclockwise order, we can use cross products to find an orientation vector.

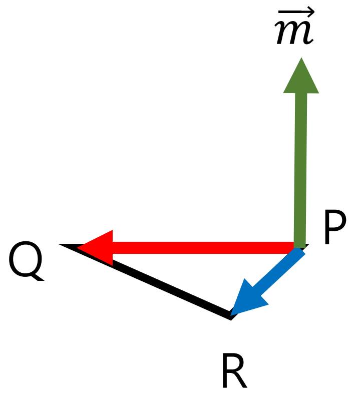
For example, if a triangle's vertexes P, Q, R are defined in a counterclockwise order like the picture, we find the orientation vector $\vec{m}$ by using the method of $\vec{m} = (Q - P) \times (R - P)$.

The basic visualization for this should look like this.



Based on the right-hand rule of cross product, the orientation vector $\vec{m}$'s direction is predictable.

Looking at the triangle from a different viewpoint, the vector $\vec{m}$ would look something like this.



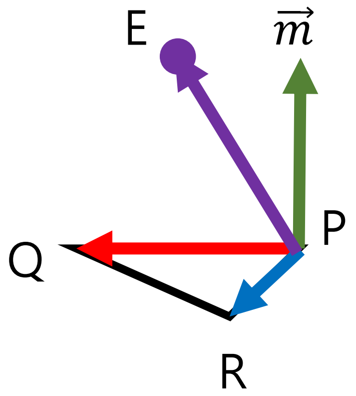
So, the face facing at the direction of orientation vector would be the front face.

Using this orientation vector, we are capable of calculating if a surface is visible from a certain point or not.

Visibility Checking

Basically, it's calculating if the front of the triangle is visible from a certain point E. From a triangle composed of vertices P, Q, R and orientation vector of $\vec{m}$, the front face of triangle is visible if and only if

$(E - P) \cdot \vec{m} \geq 0$.



This is a basic picture demonstrating the method.

The purple vector is the vector (E - P), and it is pointing at a direction where the dot product between (E − P) and the orientation vector $\vec{m}$ would not be negative, showing that the front face will be visible in the point E.

On the other hand if the dot product is negative, it would mean that the vector (E - P) points to an opposite direction, and the point E is going to be facing the back face of the triangle.

Applying this to each surfaces in a 3D model, it is easy to cull the surfaces of an object that should not be visible to the camera.