

Vertex Projection onto a Plane

Wojciech „maxest” Sterna

2/11/2008

Vertex Projection onto a Plane

Projection is an important 3D graphics operation. We use projection to make our 3D world appear in 2D screen, we use vector's dot product to project one of them onto another. In this article we will derive matrices that project vertex onto a plane.

If you still do not know what it is for I will give you a brief example. Imagine you want your object to cast a shadow onto a ground plane. You could achieve this by projecting all object's vertices onto that plane. What you get is your object flattened on the plane just as it would cast a shadow. Such object could be projected from a specific direction (a directional light casting shadow) or from a specific point (a point light casting shadow). We will consider both cases.

Directional projection

Let's start with some formulas:

$$\begin{aligned}x &= x_0 + at \\y &= y_0 + bt \\z &= z_0 + ct \\Ax + By + Cz + D &= 0\end{aligned}$$

What we have here is a straight line equation with projection direction [a, b, c], vertex (x₀, y₀, z₀) we are projecting and a plane equation [A, B, C, D] we are projecting onto. Our main task is to find t.

To find t we need to substitute x, y and z equations to plane equation. In other words we want to find such t for which a point on our straight line also lies on a plane. We have:

$$\begin{aligned}Ax + By + Cz + D &= A(x_0 + at) + B(y_0 + bt) + C(z_0 + ct) + D = 0 \\Ax_0 + Aat + By_0 + Bbt + Cz_0 + Cct + D &= 0 \\t(Aa + Bb + Cc) &= -(Ax_0 + By_0 + Cz_0 + D) \\t &= -\frac{Ax_0 + By_0 + Cz_0 + D}{Aa + Bb + Cc}\end{aligned}$$

At this point we actually have the solution we were looking for. If we substitute this just calculated t value to our initial straight line equation we will have a position (x, y, z) of vertex (x₀, y₀, z₀) projected onto a plane [A, B, C, D] along [a, b, c] direction.

Although we have a proper solution it would be much better to have it in a matrix form. Such a matrix form is convenient, for example, when we are using vertex shaders. What we do is just passing the computed matrix to a shader and letting it do the projection in a fast GPU way.

To derive a projection matrix we need to solve x, y and z equations with substituting t and rearrange them a bit. Let's first solve the equations:

$$\begin{aligned}x &= x_0 + at \\x &= x_0 - a\left(\frac{Ax_0 + By_0 + Cz_0 + D}{Aa + Bb + Cc}\right) \\x &= x_0 - \frac{aAx_0}{Aa + Bb + Cc} - \frac{aBy_0}{Aa + Bb + Cc} - \frac{aCz_0}{Aa + Bb + Cc} - \frac{aD}{Aa + Bb + Cc} \\x &= x_0\left(1 - \frac{aA}{Aa + Bb + Cc}\right) + y_0\left(-\frac{aB}{Aa + Bb + Cc}\right) + z_0\left(-\frac{aC}{Aa + Bb + Cc}\right) + \left(-\frac{aD}{Aa + Bb + Cc}\right)\end{aligned}$$

Solving y and z for t in the same way yields:

$$\begin{aligned}y &= x_0\left(-\frac{bA}{Aa + Bb + Cc}\right) + y_0\left(1 - \frac{bB}{Aa + Bb + Cc}\right) + z_0\left(-\frac{bC}{Aa + Bb + Cc}\right) + \left(-\frac{bD}{Aa + Bb + Cc}\right) \\z &= x_0\left(-\frac{cA}{Aa + Bb + Cc}\right) + y_0\left(-\frac{cB}{Aa + Bb + Cc}\right) + z_0\left(1 - \frac{cC}{Aa + Bb + Cc}\right) + \left(-\frac{cD}{Aa + Bb + Cc}\right)\end{aligned}$$

Now let's introduce a helper variable k such that:

$$k = -\frac{1}{Aa + Bb + Cc}$$

Now:

$$\begin{aligned}x &= x_0(1 + kaA) + y_0(kaB) + z_0(kaC) + kaD \\y &= x_0(kbA) + y_0(1 + kbB) + z_0(kbC) + kbD \\z &= x_0(kcA) + y_0(kcB) + z_0(1 + kcC) + kcD\end{aligned}$$

You may not see it yet but we have just derived our matrix form! Take a look:

$$[x \ y \ z \ w] = [x_0 \ y_0 \ z_0 \ 1] \begin{bmatrix} 1 + kaA & kbA & kcA & 0 \\ kaB & 1 + kbB & kcB & 0 \\ kaC & kbC & 1 + kcC & 0 \\ kaD & kbD & kcD & 1 \end{bmatrix}$$

And that's it. Our projection matrix has been derived. Note that the 4th column is made of [0, 0, 0, 1] vector. That means this matrix doesn't do any perspective transformation so we may be sure that w will always be equal to 1. Dividing x , y and z by w is not necessary.

Point projection

Point projection is very similar to directional projection. The only thing that is different is that a projection vector is unique for every object's vertex and depends on projector (point) position. Initial formulas:

$$\begin{aligned}l &= [d \ e \ f] \\x &= d + (x_0 - d)t \\y &= e + (y_0 - e)t \\z &= f + (z_0 - f)t \\Ax + By + Cz + D &= 0\end{aligned}$$

So as you can see, the equations are very similar to those used in directional projection. The only change is that we consider projector position. Note that projector position (d, e, f), vertex being projected (x_0, y_0, z_0) and projected vertex (x, y, z) all lie on the same line. So in fact we could alternatively write:

$$\begin{aligned}x &= x_0 + (x - x_0)t \\y &= y_0 + (y - y_0)t \\z &= z_0 + (z - z_0)t\end{aligned}$$

Or even:

$$\begin{aligned}x &= x_0 + (x - x_0)t \\y &= y_0 + (y - y_0)t \\z &= z_0 + (z - z_0)t\end{aligned}$$

However, both variants make deriving a matrix form very hard if not impossible. Try it yourself!

Let's go back to our initial formulas. Just as in direction projection we must first find t by substituting x , y and z to plane equation:

$$\begin{aligned}Ax + By + Cz + D &= A(d + (x_0 - d)t) + B(e + (y_0 - e)t) + C(f + (z_0 - f)t) + D = 0 \\Ad + Ax_0t - Adt + Be + By_0t - Bet + Cf + Cz_0t - Cft + D &= 0 \\t(Ax_0 - Ad + By_0 - Be + Cz_0 - Cf) &= -(Ad + Be + Cf + D) \\t &= -\frac{Ad + Be + Cf + D}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}\end{aligned}$$

Now we are solving x for t :

$$x = d + (x_0 - d)t = d - \frac{(x_0 - d)(Ad + Be + Cf + D)}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}$$

$$x = \frac{d(Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)) - (x_0 - d)(Ad + Be + Cf + D)}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}$$

$$x = \frac{d(Ax_0 + By_0 + Cz_0) - d(Ad + Be + Cf) - x_0(Ad + Be + Cf + D) + d(Ad + Be + Cf) + dD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}$$

$$x = \frac{dAx_0 + dBy_0 + dCz_0 - x_0Ad - x_0Be - x_0Cf - x_0D + dD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}$$

$$x = \frac{x_0(-Be - Cf - D) + y_0(dB) + z_0(dC) + dD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}$$

Solving y and z in a similar way yields:

$$y = \frac{x_0(eA) + y_0(-Ad - Cf - D) + z_0(eC) + eD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}$$

$$z = \frac{x_0(fA) + y_0(fB) + z_0(-Ad - Be - D) + fD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}$$

And so we have – a matrix form is now easy to write. You may be worried about the denominator. Vertex (x_0, y_0, z_0) appears there so you may think it is not correct. If so you probably forgot about one – perspective transformation.

Dividing by w will be necessary here. Final matrix:

$$[x \ y \ z \ w] = [x_0 \ y_0 \ z_0 \ 1] \begin{bmatrix} -(Be + Cf + D) & eA & fA & A \\ dB & -(Ad + Cf + D) & fB & B \\ dC & eC & -(Ad + Be + D) & C \\ dD & eD & fD & -(Ad + Be + Cf) \end{bmatrix}$$

After transformation all you have left is to divide x , y and z by w .

Acknowledgments

I'd like to thank Krzysiek K., member of <http://forum.gamedev.pl> forums, for his contribution in the following thread <http://forum.gamedev.pl/index.php/topic,1168.0.html>.