Vertex Projection onto a Plane

Wojciech Sterna

2 November, 2008

Projection is important graphics operation. We use projection to make our 3-dimensional world appear on a 2-dimensional screen, we use dot product to project one vector onto another. In this article we will derive matrices that project a 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 generating shadows) or from a specific point (a point light generating shadows). We will consider both cases.

1 Directional Projection

Let's start with some formulas:

$$\begin{aligned} x &= x_0 + at \\ y &= y_0 + bt \\ z &= z_0 + ct \end{aligned}$$

$$Ax + By + Cz + D = 0$$

What we have here is the straight line equation with projection direction $[a, b, c]$, vertex (x_0, y_0, z_0) we are projecting and the 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 the plane equation and solve for t. In other words we want to find such t for which a point on our straight line also lies on the plane. We have:

$$Ax + By + Cz + D = 0$$

$$A(x_0 + at) + B(y_0 + bt) + C(z_0 + ct) + D = 0$$

$$Ax_0 + Aat + By_0 + Bbt + Cz_0 + Cct + D = 0$$

$$t(Aa + Bb + Cc) = -(Ax_0 + By_0 + Cz_0 + D)$$

$$t = -\frac{Ax_0 + By_0 + Cz_0 + D}{Aa + Bb + Cc}$$

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 (x_0, y_0, z_0) 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 use 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 the projection matrix we need to solve x, y and z equations with substituting t and rearrange them a bit. Let's first solve x for t:

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

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

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

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:

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

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.

2 Point Projection

Point projection is very similar to directional. The only thing that is different is that the projection vector is unique for every object's vertex (x_0, y_0, z_0) and depends on projector's (point) position (d, e, f). Initial formulas:

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

$$Ax + By + Cz + D = 0$$

So as you can see, the equations are very similar to those used in directional projection. The only change is that we consider projector's position.

Just as in direction projection we must first find t by substituting x, y and z to the plane equation:

$$Ax + By + Cz + D$$

$$A(d + (x_0 - d)t) + B(e + (y_0 - e)t) + C(f + (z_0 - f)t) + D = 0$$

$$(Ad + Ax_0 t - Adt) + (Be + By_0 t - Bet) + (Cf + Cz_0 t - Cft) + D = 0$$

$$t(Ax_0 - Ad + By_0 - Be + Cz_0 - Cf) = -(Ad + Be + Cf + D)$$

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

Solving x for t yields:

$$\begin{aligned}
x &= d + (x_0 - d)t \\
x &= d - \frac{(x_0 - d)(Ad + Be + Cf + D)}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)} \\
x &= \frac{d(Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf))}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)} - \frac{(x_0 - d)(Ad + Be + Cf + D)}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)} \\
x &= \frac{d(Ax_0 + By_0 + Cz_0) - d(Ad + Be + Cf) - x_0(Ad + Be + Cf + D) + d(Ad + Be + Cf) + dD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)} \\
x &= \frac{dAx_0 + dBy_0 + dCz_0 - x_0 Ad - x_0 Be - x_0 Cf - x_0 D + dD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)} \\
x &= \frac{x_0(-Be - Cf - D) + y_0(dB) + z_0(dC) + dD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}
\end{aligned}$$

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

$$\begin{aligned}
y &= \frac{x_0(eA) + y_0(-Ad - Cf - D) + z_0(eC) + eD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)} \\
z &= \frac{x_0(fA) + y_0(fB) + z_0(-Ad - Be - D) + fD}{Ax_0 + By_0 + Cz_0 - (Ad + Be + Cf)}
\end{aligned}$$

And so we have — the matrix form is now easy to be written down. You may be worried about the denominator, as 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 division. Dividing by w will be necessary here. The final matrix:

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

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

3 Acknowledgments

I would like to thank to Krzysztof Kluczek for helping me in developing these formulas (via gamedev.pl forum).