Perspective Projection

by Patrick Rutkowski, Columbia University www.patrick-rutkowski.com

[Summary]

The projection matrix is

$$\begin{bmatrix} \cot\left(\frac{\gamma}{2}\right)/\sigma & 0 & 0 & 0 \\ 0 & \cot\left(\frac{\gamma}{2}\right) & 0 & 0 \\ 0 & 0 & \frac{\alpha+\beta}{\alpha-\beta} & \frac{2\alpha\beta}{\alpha-\beta} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

In this matrix γ is the vertical field of view angle, σ is the width-to-heigh ratio, and α and β are the distances of the clipping planes from the origin. The values of α and β are always positive, though the clipping planes themselves sit on the negative z-axis. For this matrix to work the incoming vector must have its fourth component set to 1.

[Discovery]

In a typical graphics system all geometry must end up inside of the normalized space

$$\text{NDC} = [-1, 1] \times [-1, 1] \times [-1, 1]$$

The acronym NDC is short for "normalized device coordinates." The range $[-1, 1]$ on the x-axis is mapped to the viewport's horizontal pixel space, and $[-1, 1]$ on the y-axis is mapped to the vertical pixel space. The range $[-1, 1]$ on the z axis is mapped to the depth buffer.

We need to find a method of projecting all visible geometry into this small NDC cube, and in a perspective-correct way. Let us say that the eye sits at the origin and looks down the negative z-axis. We will place the near clipping plane down the negative z-axis at a distance of α from the origin, and the far clipping plane at a distance of β. Note that the clipping planes are located on the negative z-axis, but the values α and β are always taken to be positive.

A given point $\mathbf{Q}$ has a vector that connects it to the eye. We need to intersect that vector with the near and far clipping planes. We will consider the near clipping plane first. Let $\mathbf{A}$ be the point at which $\mathbf{Q}$ intersects the near clipping plane. It will then have to be the case that

$$A_z = -\alpha$$
$$\mathbf{Q} \times \mathbf{A} = 0$$

The first equation expresses that $\mathbf{A}$ lies on the near clipping plane, and the second expresses that $\mathbf{A}$ is collinear with $\mathbf{Q}$. If we expand out the second equation we get

$$Q_y A_z - Q_z A_y = 0$$
$$Q_z A_x - Q_x A_z = 0$$
$$Q_x A_y - Q_y A_z = 0$$

But we already know that $A_z = -\alpha$, so let us rewrite the above with this taken into account:

$$\begin{aligned} -Q_y\alpha - Q_zA_y &= 0 \\ Q_zA_x + Q_x\alpha &= 0 \\ Q_xA_y - Q_yA_x &= 0 \end{aligned}$$

We can now rearrange these to solve for the other components of $\mathbf{A}$. We will drop the third equation, because we can solve for $\mathbf{A}$ with just the first two:

$$\begin{aligned} A_y &= -\alpha \cdot Q_y/Q_z \\ A_x &= -\alpha \cdot Q_x/Q_z \end{aligned}$$

Recall that we have to map these x and y values into the range $[-1, 1]$. Let's start with A_y. Think of the near clipping plane as being cut into two pieces, one above the xz-plane and one below. If γ is the field of view angle then the height h of either half can be computed by treating the value $\tan(\gamma/2)$ as a slope, and taking

$$h = \alpha \cdot \tan(\gamma/2)$$

We need only divide A_y by this value in order to map it into the range $[-1, 1]$. We will call the mapped result N_y (N for NDC):

$$N_y = A_y/h = \frac{-\alpha \cdot Q_y/Q_z}{\alpha \cdot \tan(\gamma/2)} = -\frac{Q_y \cdot \cot(\gamma/2)}{Q_z}$$

Unless the viewport is a square A_x will have to be divided by a different value. Let us give the name σ to the width-to-height ratio of the viewport. The relevant width will then be

$$w = \sigma h$$

The relevant N_x value will then be

$$N_x = A_x/w = \frac{-\alpha \cdot Q_y/Q_z}{\sigma \cdot \alpha \cdot \tan(\gamma/2)} = -\frac{Q_x \cdot \cot(\gamma/2)}{\sigma \cdot Q_z}$$

At this point we can't help but notice that this relation isn't linear, and so it can't be represented as a matrix. The division by $-1/Q_x$ is the key problem. We fix this non-linearity in the following way. Let us define a new intermediate coordinate system. We will say the transformation of $\mathbf{Q}$ into this coordinate system is called $\mathbf{C}$, where $\mathbf{C}$ is given by the following transformation (ignore the unknown elements for now):

$$\mathbf{C} = \begin{bmatrix} \cot\left(\frac{\gamma}{2}\right)/\sigma & 0 & 0 & 0 \\ 0 & \cot\left(\frac{\gamma}{2}\right) & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ ? \end{bmatrix}$$

We will then say that the final point in NDC space $\mathbf{N}$ is given by

$$\mathbf{N} = \mathbf{C}/C_w$$

Notice, of course, that the matrix we used ensures that $C_w = -Q_z$. This forced non-linear operation is called "perspective division," and it is widely supported across many graphics systems.

Next let us focus on calculating N_z. We have to find some expression for C_z which will divide sensibly by $-Q_z$ to yield a depth value in the range $[-1, 1]$. On the extreme ends we want to obtain -1 when $Q_z = -\alpha$ and $+1$ when $Q_z = -\beta$. Thus C_z must be in the range $[-\alpha, \beta]$. Observe that our point $\mathbf{Q}$ can be expressed as

$$\mathbf{Q} = \mathbf{A} + d \cdot (\mathbf{B} - \mathbf{A})$$

where d is some value between 0 and 1. Let us consider this the z-components of this equation so that we can solve for d:

$$\begin{aligned} Q_z &= A_z + d \cdot (B_z - A_z) \\ Q_z &= -\alpha + d \cdot (-\beta + \alpha) \\ Q_z + \alpha &= d \cdot (-\beta + \alpha) \\ d &= \frac{Q_z + \alpha}{\alpha - \beta} \end{aligned}$$

Since we desire that C_z be in the range $[-\alpha, \beta]$ we can set

$$\begin{aligned} C_z &= -\alpha + d\,(\alpha + \beta) \\ &= -\alpha + (\alpha + \beta) \cdot \frac{Q_z + \alpha}{\alpha - \beta} \\ &= \frac{-\alpha\,(\alpha - \beta) + (\alpha + \beta)\,(Q_z + \alpha)}{\alpha - \beta} \\ &= \frac{-\alpha^2 + \alpha\beta + Q_z\,(\alpha + \beta) + \alpha\beta + \alpha^2}{\alpha - \beta} \\ &= \frac{\alpha\beta + Q_z\,(\alpha + \beta) + \alpha\beta}{\alpha - \beta} \\ &= Q_z \frac{\alpha + \beta}{\alpha - \beta} + \frac{2\alpha\beta}{\alpha - \beta} \end{aligned}$$

Here we have again derived an expression that doesn't look like a typical matrix-vector multiplication. In a general matrix-vector multiplication every term will have one element from the matrix and one element from the vector. Yet here we have this stray term $2\alpha\beta/(\alpha - \beta)$. Fortunately we can fit this term into the matrix multiplication if we require that $Q_w = 1$. Our final result is thus

$$\mathbf{C} = \begin{bmatrix} \cot\left(\frac{\gamma}{2}\right)/\sigma & 0 & 0 & 0 \\ 0 & \cot\left(\frac{\gamma}{2}\right) & 0 & 0 \\ 0 & 0 & \frac{\alpha + \beta}{\alpha - \beta} & \frac{2\alpha\beta}{\alpha - \beta} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix}$$

There is one aspect of this matrix which we have so far neglected to mention, which is important in some graphics systems. Recall that the expression for A_x was

$$A_x = -\alpha \cdot Q_x / Q_z$$

As we continued on from this we chose to construct the matrix such that we would have

$$C_w = -Q_z$$

We could have just as easily set flipped the -1 on the last row to a 1, in which case we would have

$$C_w = Q_z$$

In this case we would have to negate terms like $\cot(\gamma/2)/\sigma$ to get the negative sign back into the expression for A_x. Now, recall that visible points will have negative values for Q_z, since the eye looks down the negative z-axis. This will mean that the choice $C_w = -Q_z$ will make C_w positive. This is important because OpenGL will automatically clip away points for which any of following inequalities fail:

$$\begin{aligned} -C_w &\leq C_x \leq C_w \\ -C_w &\leq C_y \leq C_w \\ -C_w &\leq C_z \leq C_w \end{aligned}$$

These inequalities would be false by definition if C_w were to be assigned a negative value. This is exactly why we chose to have -1 on the fourth row of our matrix, as opposed to 1.