←Back

OpenGL Projection Matrix

Related Topics: OpenGL Transformation, OpenGL Matrix

Updates: The MathML version is available here.

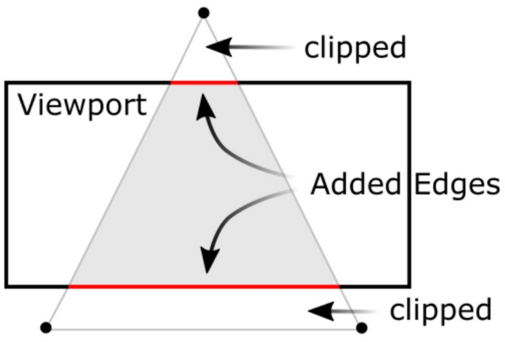
Overview

A computer monitor is a 2D surface. A 3D scene rendered by OpenGL must be projected onto the computer screen as a 2D image. GL_PROJECTION matrix is used for this projection transformation. First, it transforms all vertex data from the eye coordinates to the clip coordinates. Then, these clip coordinates are also transformed to the normalized device coordinates (NDC) by dividing with *w* component of the clip coordinates.

Therefore, we have to keep in mind that both clipping (frustum culling) and NDC transformations are integrated into **GL_PROJECTION** matrix. The following sections describe how to build the projection matrix from 6 parameters; *left*, *right*, *bottom*, *top*, *near* and *far* boundary values.

Note that the frustum culling (clipping) is performed in the clip coordinates, just before dividing by w_c. The clip coordinates, x_c, y_c and z_c are tested by comparing with w_c. If any clip coordinate is less than $-w_c$, or greater than w_c, then the vertex will be discarded.

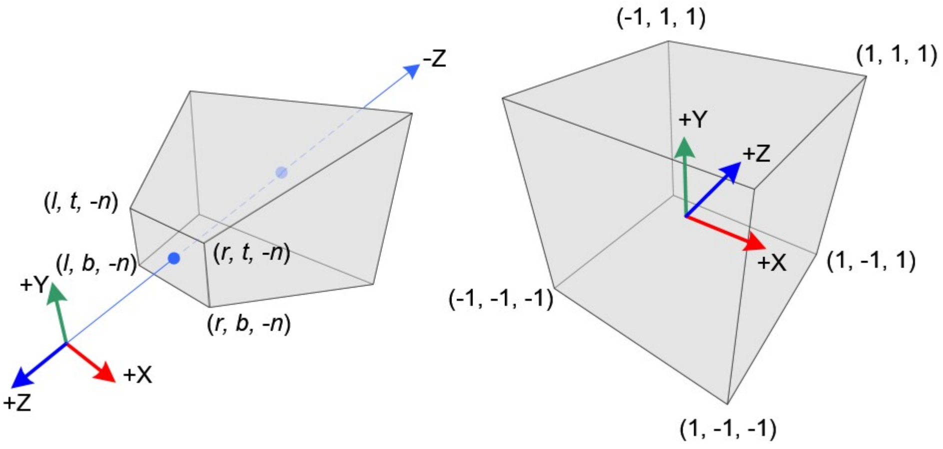
$$-w_c < x_c, y_c, z_c < w_c$$



A triangle clipped by frustum

Then, OpenGL will reconstruct the edges of the polygon where clipping occurs.

Perspective Projection

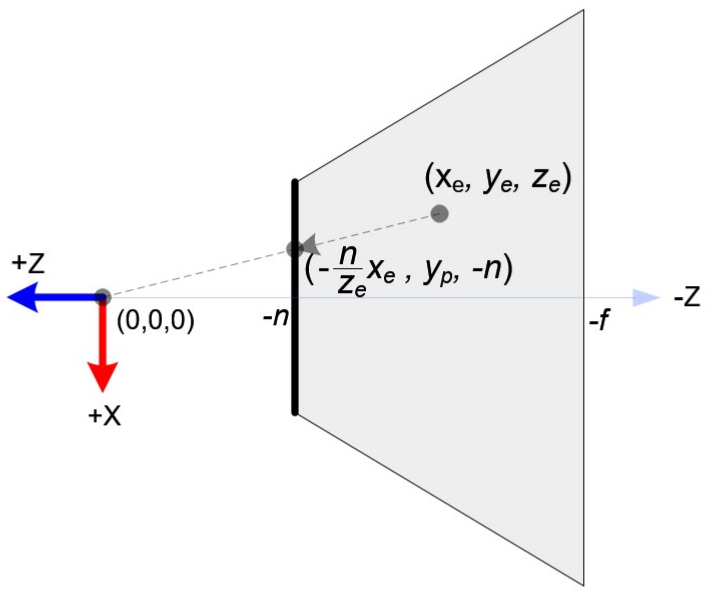


Perspective Frustum and Normalized Device Coordinates (NDC)

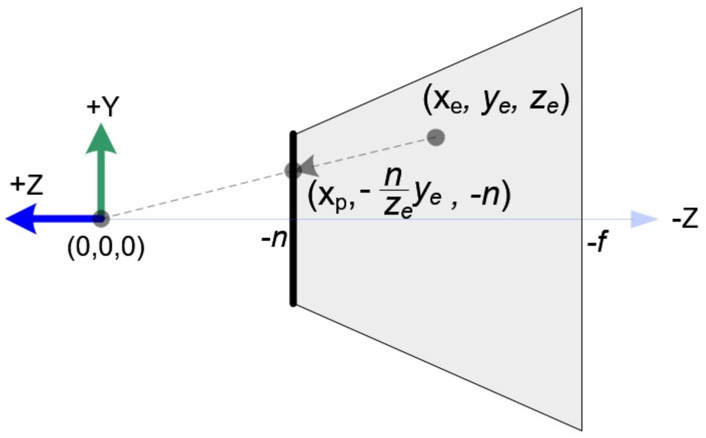
In perspective projection, a 3D point in a truncated pyramid frustum (eye coordinates) is mapped to a cube (NDC); the range of x-coordinate from [l, r] to [-1, 1], the y-coordinate from [b, t] to [-1, 1] and the z-coordinate from [-n, -f] to [-1, 1].

Note that the eye coordinates are defined in the right-handed coordinate system, but NDC uses the left-handed coordinate system. That is, the camera at the origin is looking along -Z axis in eye space, but it is looking along +Z axis in NDC. Since **glFrustum()** accepts only positive values of *near* and *far* distances, we need to negate them during the construction of GL_PROJECTION matrix.

In OpenGL, a 3D point in eye space is projected onto the *near* plane (projection plane). The following diagrams show how a point (x_e, y_e, z_e) in eye space is projected to (x_p, y_p, z_p) on the *near* plane.



Top View of Frustum



Side View of Frustum

From the top view of the frustum, the x-coordinate of eye space, x_e is mapped to x_p, which is calculated by using the ratio of similar triangles;

$$\frac{x_p}{x_e} = \frac{-n}{z_e}$$

$$x_p = \frac{-n \cdot x_e}{z_e} = \frac{n \cdot x_e}{-z_e}$$

From the side view of the frustum, y_p is also calculated in a similar way;

$$\frac{y_p}{y_e} = \frac{-n}{z_e}$$

$$y_p = \frac{-n \cdot y_e}{z_e} = \frac{n \cdot y_e}{-z_e}$$

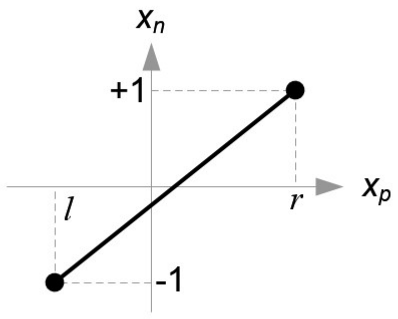
Note that both x_p and y_p depend on z_e; they are inversely propotional to $-z_e$. In other words, they are both divided by $-z_e$. It is a very first clue to construct GL_PROJECTION matrix. After the eye coordinates are transformed by multiplying GL_PROJECTION matrix, the clip coordinates are still a homogeneous coordinates. It finally becomes the normalized device coordinates (NDC) by divided by the w-component of the clip coordinates. (*See more details on OpenGL Transformation.*)

$$\begin{pmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{pmatrix} = M_{projection} \cdot \begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix}, \quad \begin{pmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{pmatrix} = \begin{pmatrix} x_{clip}/w_{clip} \\ y_{clip}/w_{clip} \\ z_{clip}/w_{clip} \end{pmatrix}$$

Therefore, we can set the w-component of the clip coordinates as $-z_e$. And, the 4th of GL_PROJECTION matrix becomes (0, 0, -1, 0).

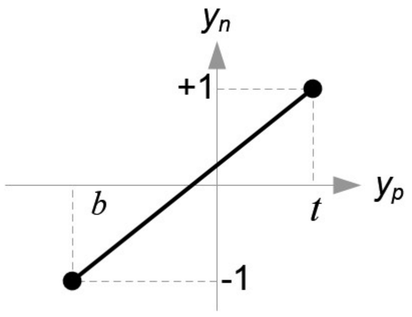
$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}, \qquad \therefore w_c = -z_e$$

Next, we map x_p and y_p to x_n and y_n of NDC with linear relationship; [l, r] ⇒ [-1, 1] and [b, t] ⇒ [-1, 1].



Mapping from x_p to x_n

$$\begin{aligned} x_n &= \frac{1-(-1)}{r-l} \cdot x_p + \beta \\ 1 &= \frac{2r}{r-l} + \beta \qquad \text{(substitute } (r,1) \text{ for } (x_p, x_n)) \\ \beta &= 1 - \frac{2r}{r-l} = \frac{r-l}{r-l} - \frac{2r}{r-l} \\ &= \frac{r-l-2r}{r-l} = \frac{-r-l}{r-l} = -\frac{r+l}{r-l} \\ \therefore x_n &= \frac{2x_p}{r-l} - \frac{r+l}{r-l} \end{aligned}$$



Mapping from y_p to y_n

$$\begin{aligned}
y_n &= \frac{1-(-1)}{t-b} \cdot y_p + \beta \\
1 &= \frac{2t}{t-b} + \beta \qquad \text{(substitute } (t, 1) \text{ for } (y_p, y_n)) \\
\beta &= 1 - \frac{2t}{t-b} = \frac{t-b}{t-b} - \frac{2t}{t-b} \\
&= \frac{t-b-2t}{t-b} = \frac{-t-b}{t-b} = -\frac{t+b}{t-b} \\
\therefore y_n &= \frac{2y_p}{t-b} - \frac{t+b}{t-b}
\end{aligned}$$

Then, we substitute x_p and y_p into the above equations.

$$\begin{aligned}
x_n &= \frac{2x_p}{r-l} - \frac{r+l}{r-l} \qquad (x_p = \frac{nx_e}{-z_e}) \\
&= \frac{2 \cdot \frac{n \cdot x_e}{-z_e}}{r-l} - \frac{r+l}{r-l} \\
&= \frac{2n \cdot x_e}{(r-l)(-z_e)} - \frac{r+l}{r-l} \\
&= \frac{\frac{2n}{r-l} \cdot x_e}{-z_e} - \frac{r+l}{r-l} \\
&= \frac{\frac{2n}{r-l} \cdot x_e}{-z_e} + \frac{\frac{r+l}{r-l} \cdot z_e}{-z_e} \\
&= \Bigg(\underbrace{\frac{2n}{r-l} \cdot x_e + \frac{r+l}{r-l} \cdot z_e}_{x_c}\Bigg) \Big/ - z_e
\end{aligned}$$

$$\begin{aligned} y_n &= \frac{2y_p}{t-b} - \frac{t+b}{t-b} \qquad \left(y_p = \frac{ny_e}{-z_e}\right) \\ &= \frac{2 \cdot \frac{n \cdot y_e}{-z_e}}{t-b} - \frac{t+b}{t-b} \\ &= \frac{2n \cdot y_e}{(t-b)(-z_e)} - \frac{t+b}{t-b} \\ &= \frac{\frac{2n}{t-b} \cdot y_e}{-z_e} - \frac{t+b}{t-b} \\ &= \frac{\frac{2n}{t-b} \cdot y_e}{-z_e} + \frac{\frac{t+b}{t-b} \cdot z_e}{-z_e} \\ &= \left(\underbrace{\frac{2n}{t-b} \cdot y_e + \frac{t+b}{t-b} \cdot z_e}_{y_c} \right) \Big/ -z_e \end{aligned}$$

Note that we make both terms of each equation divisible by $-z_e$ for perspective division (x_c/w_c, y_c/w_c). And we set w_c to $-z_e$ earlier, and the terms inside parentheses become x_c and y_c of the clip coordiantes.

From these equations, we can find the 1st and 2nd rows of GL_PROJECTION matrix.

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}$$

Now, we only have the 3rd row of GL_PROJECTION matrix to solve. Finding z_n is a little different from others because z_e in eye space is always projected to -n on the near plane. But we need unique z value for the clipping and depth test. Plus, we should be able to unproject (inverse transform) it. Since we know z does not depend on x or y value, we borrow w-component to find the relationship between z_n and z_e. Therefore, we can specify the 3rd row of GL_PROJECTION matrix like this.

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}, \qquad z_n = z_c/w_c = \frac{Az_e + Bw_e}{-z_e}$$

In eye space, w_e equals to 1. Therefore, the equation becomes;

$$z_n = \frac{Az_e + B}{-z_e}$$

To find the coefficients, *A* and *B*, we use the (z_e, z_n) relation; (-n, -1) and (-f, 1), and put them into the above equation.

$$\begin{cases} \dfrac{-An+B}{n} = -1 \\ \\ \dfrac{-Af+B}{f} = 1 \end{cases} \rightarrow \begin{cases} -An + B = -n & (1) \\ -Af + B = f & (2) \end{cases}$$

To solve the equations for *A* and *B*, rewrite eq.(1) for B;

$$B = An - n \qquad (1')$$

Substitute eq.(1') to *B* in eq.(2), then solve for A;

$$-Af + (An - n) = f \qquad (2)$$

$$-(f-n)A = f + n$$

$$A = -\frac{f+n}{f-n}$$

Put *A* into eq.(1) to find *B*;

$$\left(\frac{f+n}{f-n}\right)n + B = -n \qquad (1)$$

$$B = -n - \left(\frac{f+n}{f-n}\right)n = -\left(1 + \frac{f+n}{f-n}\right)n = -\left(\frac{f-n+f+n}{f-n}\right)n$$
$$= -\frac{2fn}{f-n}$$

We found *A* and *B*. Therefore, the relation between z_e and z_n becomes;

$$z_n = \frac{-\frac{f+n}{f-n}z_e - \frac{2fn}{f-n}}{-z_e} \qquad (3)$$

Finally, we found all entries of GL_PROJECTION matrix. The complete projection matrix is;

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

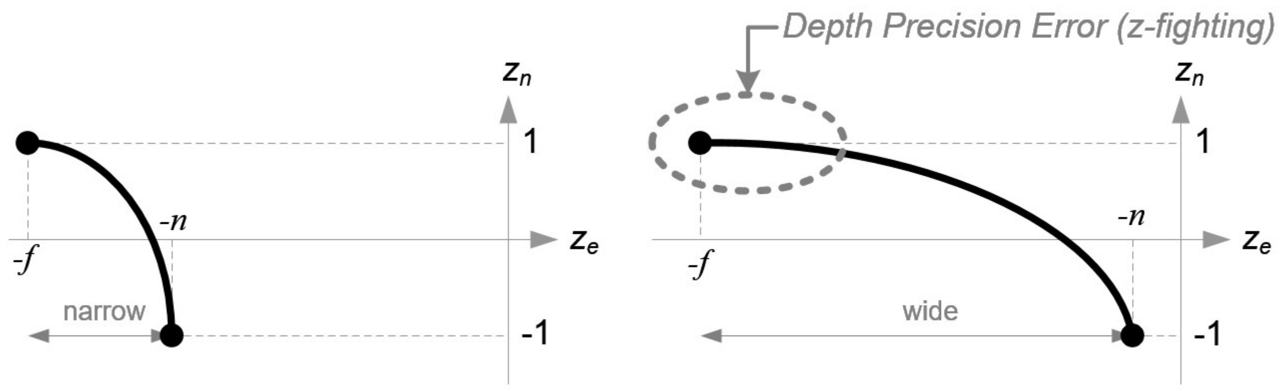
OpenGL Perspective Projection Matrix

This projection matrix is for a general frustum. If the viewing volume is symmetric, which is $r = -l$ and $t = -b$, then it can be simplified as;

$$\begin{cases} r + l = 0 \\ r - l = 2r \text{ (width)} \end{cases}, \quad \begin{cases} t + b = 0 \\ t - b = 2t \text{ (height)} \end{cases}$$

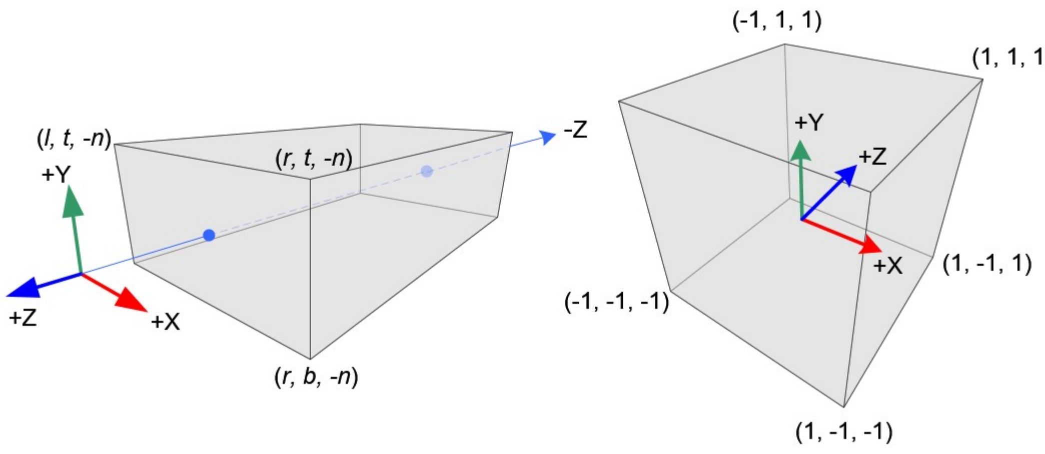
$$\begin{pmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Before we move on, please take a look at the relation between z_e and z_n, eq.(3) once again. You notice it is a rational function and is non-linear relationship between z_e and z_n. It means there is very high precision at the *near* plane, but very little precision at the *far* plane. If the range [-n, -f] is getting larger, it causes a depth precision problem (z-fighting); a small change of z_e around the *far* plane does not affect on z_n value. The distance between *n* and *f* should be short as possible to minimize the depth buffer precision problem.



Comparison of Depth Buffer Precisions

Orthographic Projection

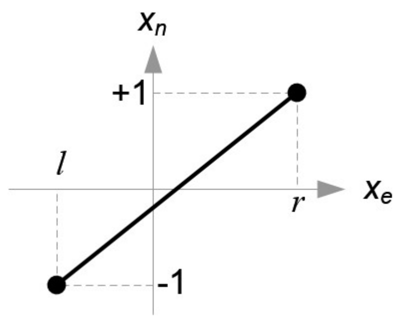


Orthographic Volume and Normalized Device Coordinates (NDC)

Constructing GL_PROJECTION matrix for orthographic projection is much simpler than perspective mode.

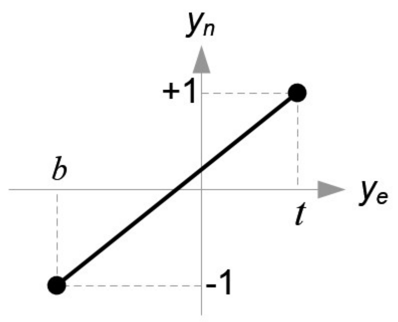
All x_e, y_e and z_e components in eye space are linearly mapped to NDC. We just need to scale a rectangular volume to a cube, then move it to the origin.

Let's find out the elements of GL_PROJECTION using linear relationship.



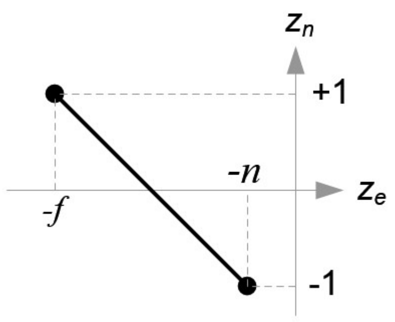
Mapping from x_e to x_n

$$\begin{aligned}
x_n &= \frac{1-(-1)}{r-l} \cdot x_e + \beta \\
1 &= \frac{2r}{r-l} + \beta \qquad (\text{substitute } (r, 1) \text{ for } (x_e, x_n)) \\
\beta &= 1 - \frac{2r}{r-l} = -\frac{r+l}{r-l} \\
\therefore x_n &= \frac{2}{r-l} \cdot x_e - \frac{r+l}{r-l}
\end{aligned}$$



Mapping from y_e to y_n

$$\begin{aligned}
y_n &= \frac{1-(-1)}{t-b} \cdot y_e + \beta \\
1 &= \frac{2t}{t-b} + \beta \qquad (\text{substitute } (t, 1) \text{ for } (y_e, y_n)) \\
\beta &= 1 - \frac{2t}{t-b} = -\frac{t+b}{t-b} \\
\therefore y_n &= \frac{2}{t-b} \cdot y_e - \frac{t+b}{t-b}
\end{aligned}$$



Mapping from z_e to z_n

$$\begin{aligned}
z_n &= \frac{1-(-1)}{-f-(-n)} \cdot z_e + \beta \\
1 &= \frac{2f}{f-n} + \beta \qquad (\text{substitute } (-f,1) \text{ for } (z_e, z_n)) \\
\beta &= 1 - \frac{2f}{f-n} = -\frac{f+n}{f-n} \\
\therefore z_n &= \frac{-2}{f-n} \cdot z_e - \frac{f+n}{f-n}
\end{aligned}$$

Since w-component is not necessary for orthographic projection, the 4th row of GL_PROJECTION matrix remains as (0, 0, 0, 1). Therefore, the complete GL_PROJECTION matrix for orthographic projection is;

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

OpenGL Orthographic Projection Matrix

It can be further simplified if the viewing volume is symmetrical, $r = -l$ and $t = -b$.

$$\begin{cases} r+l=0 \\ r-l=2r \ (\text{width}) \end{cases}, \begin{cases} t+b=0 \\ t-b=2t \ (\text{height}) \end{cases}$$

$$\begin{pmatrix} \frac{1}{r} & 0 & 0 & 0 \\ 0 & \frac{1}{t} & 0 & 0 \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

←Back