Computing $3D$ coordinate from a pixel

April 19, 2020

Introduction

Before we talk about the OpenGL pipeline, we should learn how to translate/rotate/scale points using matrices.

Translation matrix/vector

Let $p = (p_x, p_y, p_z)$ be some point in $\mathbb{R}^3$. and let $t = (t_x, t_y, t_z)$.
$q = p + t$ is a point that we get after translating (”moving”) p by t.
We can also define a matrix for translation (not vector) in the following way:

$$T = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In this case we can get that:

$$\begin{pmatrix} q_x \\ q_y \\ q_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}$$

Instead of calculating $q = p + t$ we used matrix multiplication. Using matrix multiplication can be useful when combining translation with other matrices like rotation matrix or scale matrix. If we want to translate and rotate a point we can just multiply a translation matrix with a rotation matrix.

Scaling matrix

Scaling matrix looks like the following:

$$S = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}$$

Given a point $p = (p_x, p_y, p_z)^T$,

$$S \cdot p = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} s_x p_x \\ s_y p_y \\ s_z p_z \end{pmatrix}$$

Example for a scaling matrix:

$$S = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

In this case,

$$S \cdot p = \begin{pmatrix} 2p_x \\ 2p_y \\ 2p_z \end{pmatrix}$$

Each coordinate was multiplied by 2. If we scale a mesh that contains many vertices by the same scaling matrix S we end up with a bigger mesh (every side of the mesh is twice bigger than the corresponding side in the original mesh).

Rotation matrix

Rotation matrix in d dimensions is a matrix of size $(d \times d)$ that rotates any d dimensional point p over a 2 dimensional plane that contains both p and the origin (or in other words, rotates p around a subspace of $(d-2)$ dimensions that passes through the origin).

Rotation matrix

In $\mathbb{R}^2$ $(d = 2)$ a subspace of $(d - 2)$ dimensions is a point. And since the subspace passes through the origin, the point must be the origin. Therefore, in $2D$ a rotation matrix is a matrix of size (2×2) that rotates a point around the origin point.

Rotation matrix

In $\mathbb{R}^3$ $(d = 3)$ a subspace of $(d - 2)$ dimensions is a line. Therefore, in $3D$ a rotation matrix is a matrix of size (3×3) that rotates a point around some axis line that passes through the origin.

Basic rotations in $3D$:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix} = \text{rotation about } x \text{ axis}$$

$$R_y(\theta) = \begin{pmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{pmatrix} = \text{rotation about } y \text{ axis}$$

$$R_z(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \text{rotation about } z \text{ axis}$$

Rotation matrix

What if we want to rotate a point p in angle θ around some arbitrary axis with the direction $u = (u_x, u_y, u_z)$, $\|u\| = 1$? Note that the axis must pass through the origin $(0, 0, 0)$ otherwise, we can't rotate the point around the axis using a rotation matrix only. We must use also a translation matrix (twice).

Rotation matrix

There are many ways to rotate a point around some axis u in the angle θ. For example, Rodrigues' rotation formula (Rodriques is for $3D$ only).
I'll show here one simple way (that can be generalized to d dimensions).

- We first need to calculate a rotation matrix R_u that rotates the point p such that u will coincide z axis.
- Then we rotate the point around z axis in the angle θ using the rotation matrix $R_z(\theta)$.
- And last, we rotate the point back using ${R_u}^T$ (which is the inverse of R_u) so that u will be back to its original direction.

How to calculate R_u:

Note that there are infinite possibilities for R_u. We just need one of them (doesn't matter which one).

Since R_u is a rotation matrix, it must satisfy the following:

- R_u is orthonormal.
- $\det(R_u) = 1$

$$R_u = \begin{pmatrix} w_x & w_y & w_z \\ v_x & v_y & v_z \\ k_x & k_y & k_z \end{pmatrix} = \begin{pmatrix} - & w & - \\ - & v & - \\ - & k & - \end{pmatrix}$$

Since R_u is orthonormal then

$\|w\| = 1, \|v\| = 1, \|k\| = 1, w \cdot v = 0, w \cdot k = 0, v \cdot k = 0.$

Rotation matrix

Since R_u rotates u to coincide z then:

$$R_u u = \begin{pmatrix} - & w & - \\ - & v & - \\ - & k & - \end{pmatrix} \begin{pmatrix} | \\ u \\ | \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Therefore, $w \cdot u = 0$, $v \cdot u = 0$, $k \cdot u = 1$.

Therefore, $k = u$ (since $k \cdot u = u \cdot u = \|u\|^2 = 1$).

Now we need to find w and v such that both perpendicular to u and also to each other (there are infinite possibilities. We just need one).

Rotation matrix

Note that $\|u\| = 1$.

if $u = (0,0,1)$ then we can say that $w = (1,0,0)$ and $v = (0,1,0)$.

Meaning that R_u is the identity matrix (we don't really need to rotate u to coincide z axis because u is already in z direction).

if $u = (u_x, u_y, u_z) \neq (0,0,1)$, we can say that:

- $w = \frac{(-u_y,u_x,0)}{\|(-u_y,u_x,0)\|}$ (note that we don't divide by zero).
- $v = w \times u$.

Rotation matrix

The matrix $R_u = \begin{pmatrix} - & w & - \\ - & v & - \\ - & k & - \end{pmatrix}$ is now orthonormal but we can get that $\det(R_u) = 1$ or $\det(R_u) = -1$.

Note that since R_u is a rotation matrix we must get that $\det(R_u) = 1$.

If $\det(R_u) = -1$ then we should negate w or v and then R_u should be the desired rotation matrix.

Rotation matrix

Now that we calculated R_u, the rotation matrix R that rotates any point in θ around an axis with the direction u that passes the origin should be the following rotation:

$$R = {R_u}^T R_z(\theta) R_u = \begin{pmatrix} | & | & | \\ w & v & u \\ | & | & | \end{pmatrix} \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} - & w & - \\ - & v & - \\ - & u & - \end{pmatrix}$$

Rotation matrix

satisfies the following:

- One entry in the list
- Another entry in the list

$$\begin{bmatrix} VP \\ matrix \end{bmatrix} = \begin{bmatrix} projection \\ matrix \end{bmatrix} \begin{bmatrix} view \\ matrix \end{bmatrix} \tag{1}$$

$$\text{clip coords} = \begin{bmatrix} x_c \\ y_c \\ z_c \\ w \end{bmatrix} = \begin{bmatrix} VP \\ matrix \end{bmatrix} \begin{bmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{bmatrix} \tag{2}$$

$$\text{ndc} = \begin{bmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{bmatrix} = \begin{bmatrix} \frac{x_c}{w} \\ \frac{y_c}{w} \\ \frac{z_c}{w} \end{bmatrix} \tag{3}$$

$$-1 \leq x_{ndc} \leq 1$$
$$-1 \leq y_{ndc} \leq 1$$
$$-1 \leq z_{ndc} \leq 1$$

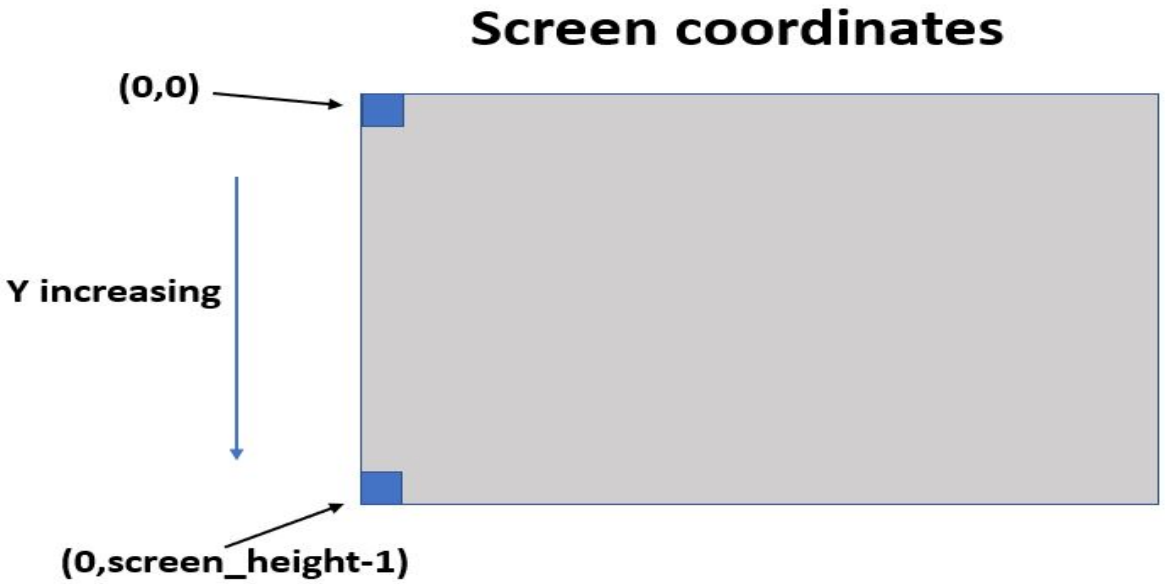
Suppose we know w from the beginning. We can change order and first divide by w and then multiply by VP matrix. The result will be the same.

$$\text{clip coords} = \begin{bmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \\ 1 \end{bmatrix} = \begin{bmatrix} VP \\ matrix \end{bmatrix} \begin{bmatrix} \frac{x_{world}}{w} \\ \frac{y_{world}}{w} \\ \frac{z_{world}}{w} \\ \frac{1}{w} \end{bmatrix} \tag{4}$$

We get the same ndc vector in both cases.

$$x_{pixel} = \frac{x_{ndc} + 1}{2} \cdot (SCREEN\_WIDTH - 1) \tag{5}$$

$$y_{pixel} = (SCREEN\_HEIGHT - 1) - \frac{y_{ndc} + 1}{2} \cdot (SCREEN\_HEIGHT - 1) \tag{6}$$



z is saved in depth buffer, but in the range (min_z, max_z) where min_z, max_z are defined by $glDepthRange$ (by default $min_z = 0$, $max_z = 1$).

$$z = \frac{z_{ndc} + 1}{2}(max_z - min_z) + min_z. \tag{7}$$

Note that if $glDepthRange(min_z, max_z)$ is called with $max_z < 1$, then $glClearDepth(max_z)$ should also be called (before calling $glClear$). Also, min_z, max_z should be in the range $[0, 1]$ otherwise each one of them is clamped to the closest edge. For example, if $min_z = -1$ then it is clamped to 0. Therefore $glDepthRange(-1, 1)$ will have the same effect as $glDepthRange(0, 1)$ which is the default case.

Given a pixel (x_{pixel}, y_{pixel}) and given z value that can be read from the depth buffer in the location (x_{pixel}, y_{pixel}), we want to calculate the $3D$ coordinate.

First, we calculate the ndc vector:

$$x_{ndc} = \frac{2 \cdot x_{pixel}}{SCREEN\_WIDTH - 1} - 1 \tag{8}$$

$$y_{ndc} = \frac{2 \cdot (SCREEN\_HEIGHT - 1 - y_{pixel})}{SCREEN\_HEIGHT - 1} - 1 \tag{9}$$

$$z_{ndc} = \frac{2 \cdot (z - min_z)}{max_z - min_z} - 1 \tag{10}$$

Where (8) is from (5), (9) is from (6) and (10) is from (7).

We now have:

$$\text{clip coords} = \begin{bmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \\ 1 \end{bmatrix} \tag{11}$$

By (21), we get:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} \frac{x_{world}}{w} \\ \frac{y_{world}}{w} \\ \frac{z_{world}}{w} \\ \frac{1}{w} \end{bmatrix} = \begin{bmatrix} VP \\ matrix \end{bmatrix}^{-1} \begin{bmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \\ 1 \end{bmatrix} \tag{12}$$

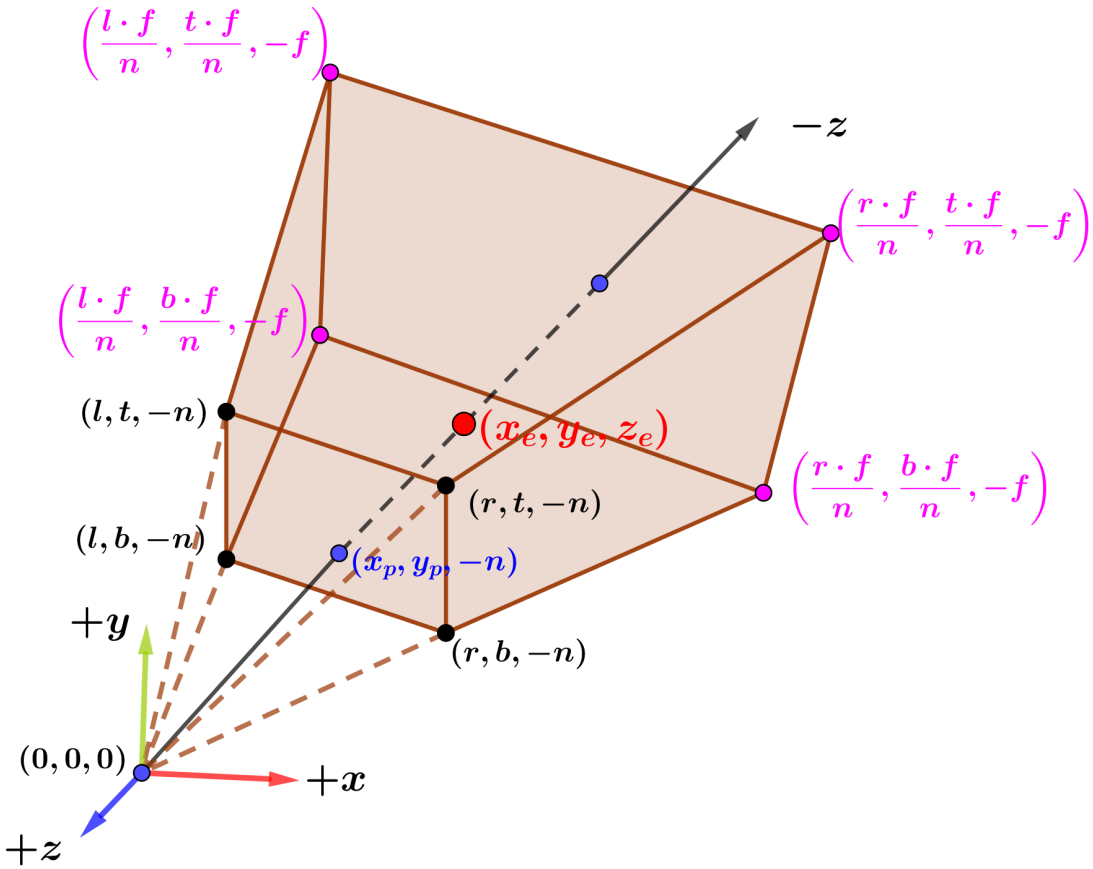
$v_4 = \frac{1}{w}$. Therefore $w = \frac{1}{v_4}$.

And finally:

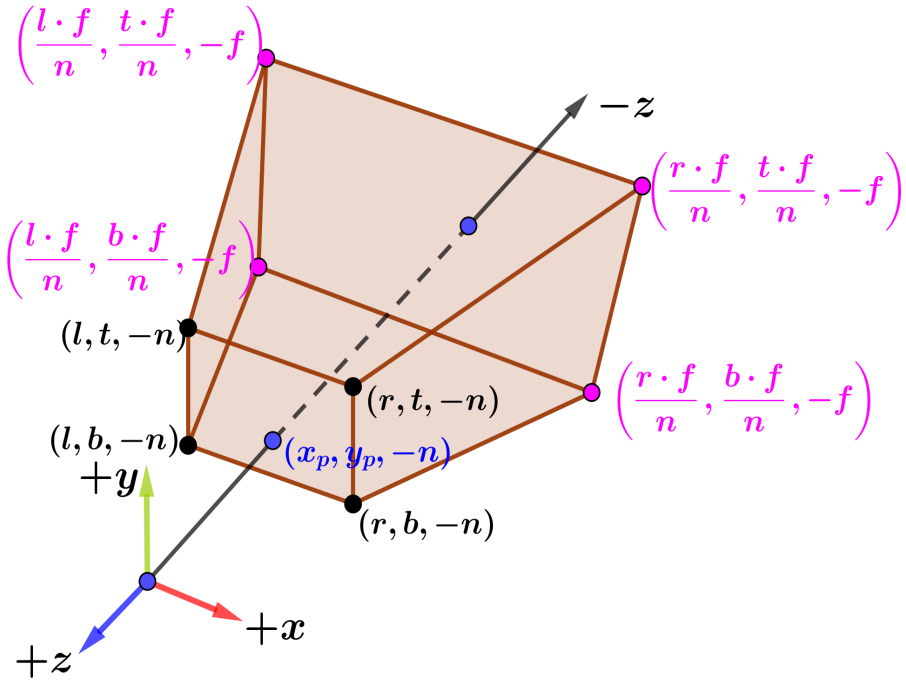
$$\begin{bmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{bmatrix} = w \cdot v = w \cdot \begin{bmatrix} \frac{x_{world}}{w} \\ \frac{y_{world}}{w} \\ \frac{z_{world}}{w} \\ \frac{1}{w} \end{bmatrix} \tag{13}$$

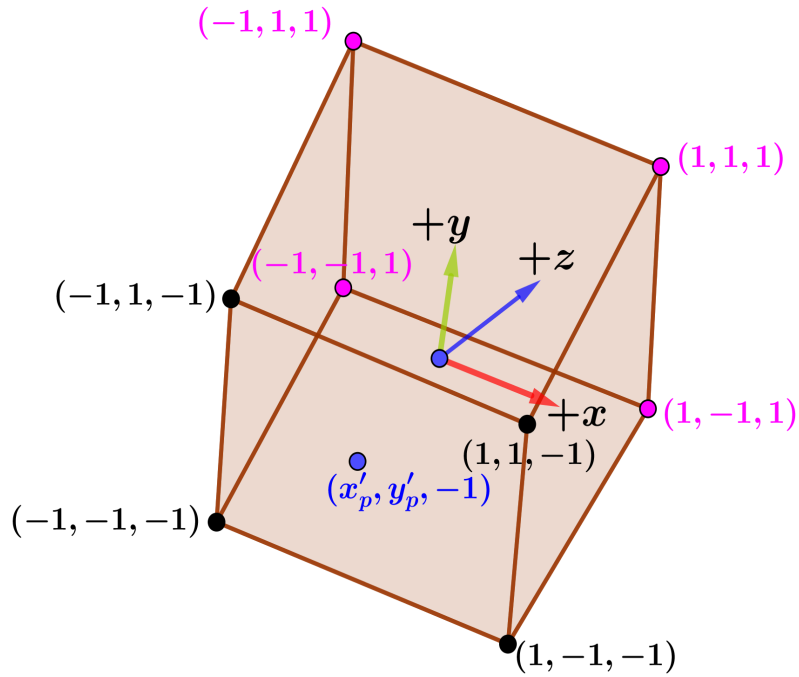
Projection matrix

Projection matrix is a matrix that projects a point in *eye* coordinate into a point in *clip* coordinate.



Projection matrix





Projection matrix

$(x_p, y_p, -n)$ is the projection of the point (x_e, y_e, z_e) on the near plane ($z = -n$). if (x_e, y_e, z_e) is inside the truncated pyramid then all the following conditions must be satisfied (assuming $l \leq r$, $b \leq t$, $n \leq f$):

$$l \leq x_p \leq r \implies \frac{-r \cdot z_e}{n} \leq x_e \leq \frac{-l \cdot z_e}{n} \tag{14}$$

$$b \leq y_p \leq t \implies \frac{-t \cdot z_e}{n} \leq y_e \leq \frac{-b \cdot z_e}{n} \tag{15}$$

$$-f \leq z_e \leq -n \quad (z_p = -n) \tag{16}$$

Projection matrix

$$\mathbf{p} = \left(\frac{-\mathbf{r}\cdot\mathbf{z}}{\mathbf{n}}, \frac{-\mathbf{y}\cdot\mathbf{z}}{\mathbf{n}}, \mathbf{z}\right)$$

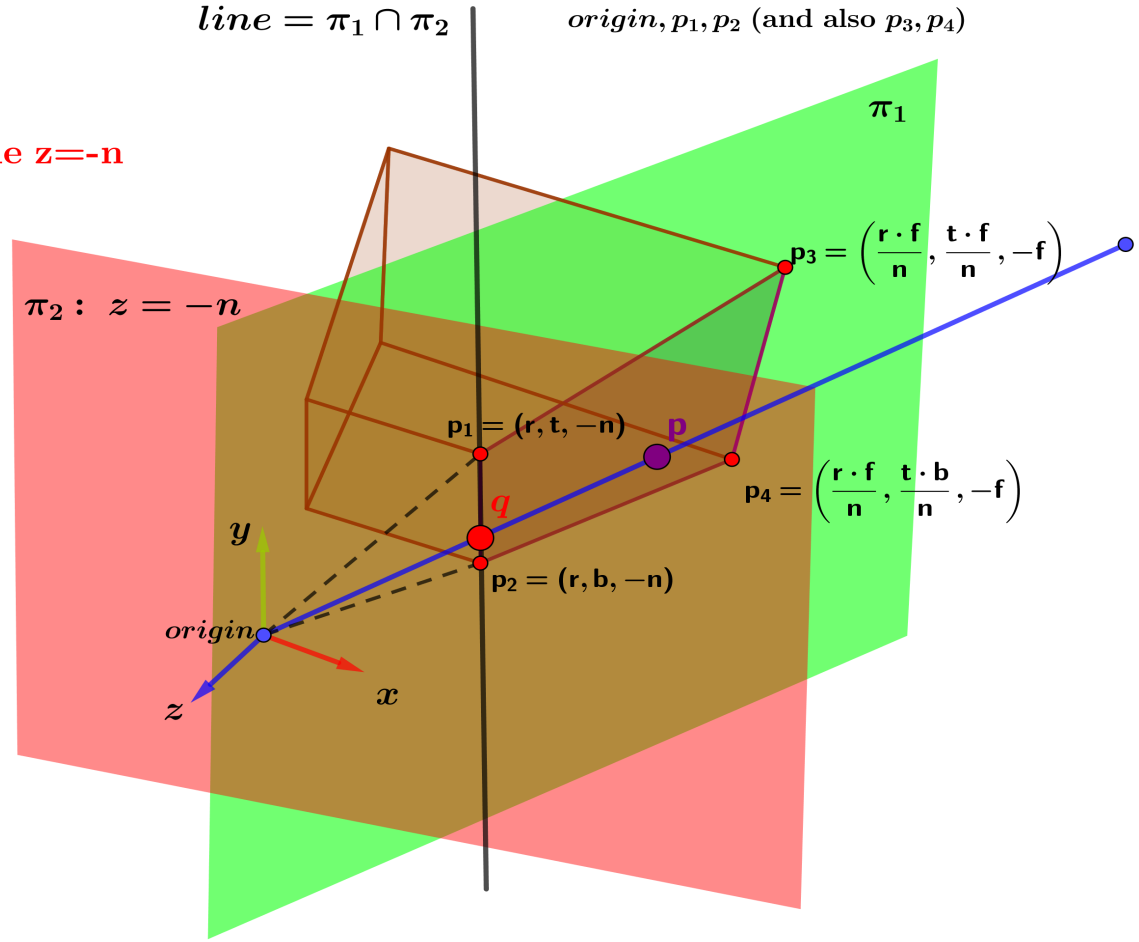
p is a general point on π_1

$q = (r, y, -n)$

q is the projection of p on plane z=-n

note that if y=t, we get $q = p_1$
and if y=b, we get $q = p_2$

π_1 : contains right face of truncated pyramid
π_1 is defined by 3 points:
$origin, p_1, p_2$ (and also p_3, p_4)

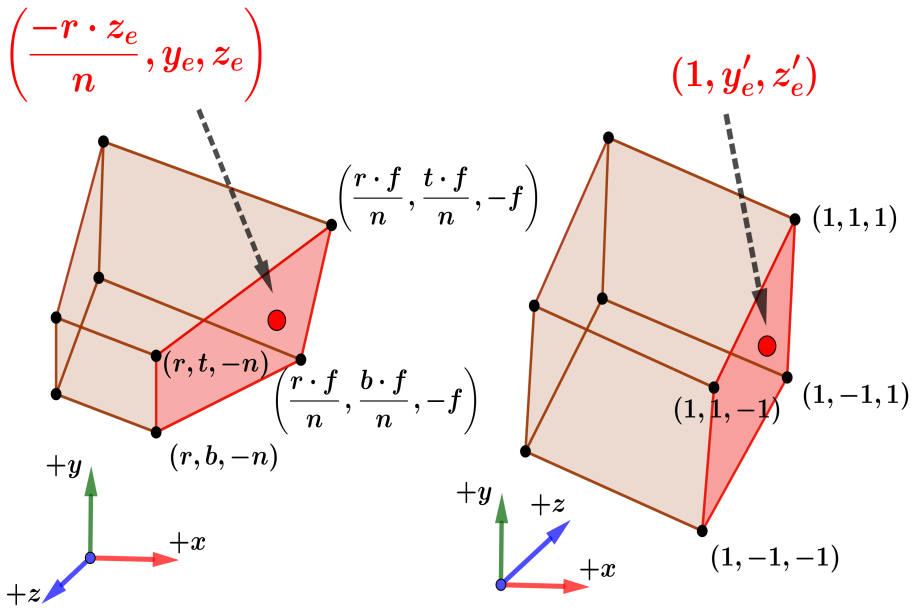


Projection matrix

$$\textbf{projection matrix} = \begin{bmatrix} \frac{2 \cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2 \cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2 \cdot f \cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{17}$$

Projection matrix

$$v_c = \begin{bmatrix} -z_e \\ . \\ . \\ -z_e \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{-r\cdot z_e}{n} \\ y_e \\ z_e \\ 1 \end{bmatrix} \tag{18}$$



$$v_c = \begin{bmatrix} z_e \\ \cdot \\ \cdot \\ -z_e \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{-l\cdot z_e}{n} \\ y_e \\ z_e \\ 1 \end{bmatrix} \tag{19}$$

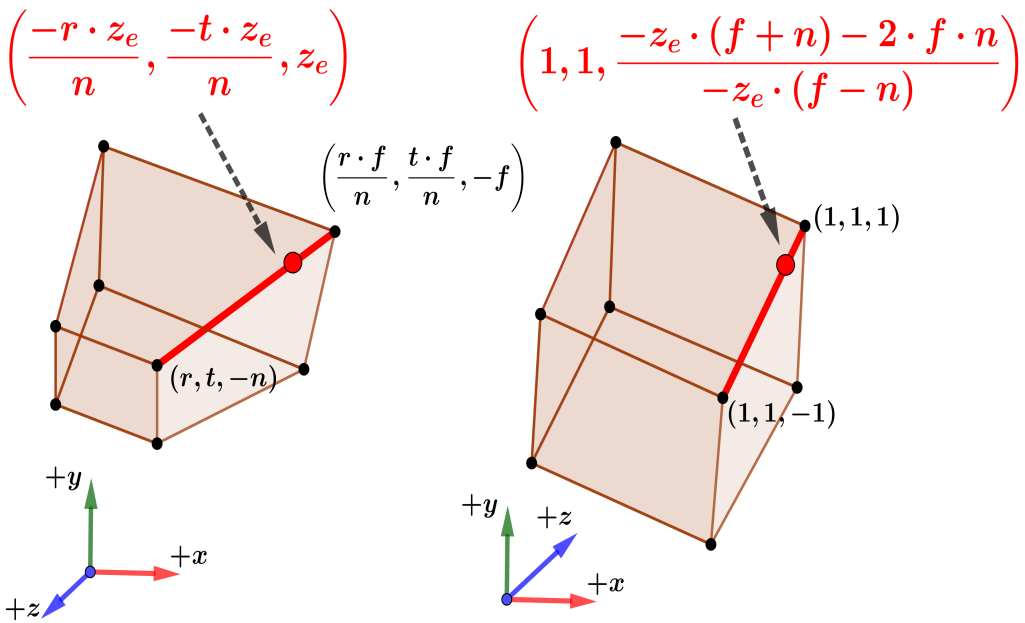
Projection matrix

$$v_c = \begin{bmatrix} \cdot \\ -z_e \\ \cdot \\ -z_e \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ \frac{-t\cdot z_e}{n} \\ z_e \\ 1 \end{bmatrix} \tag{20}$$

$$v_c = \begin{bmatrix} \cdot \\ z_e \\ \cdot \\ -z_e \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ \frac{-b\cdot z_e}{n} \\ z_e \\ 1 \end{bmatrix} \tag{21}$$

Projection matrix

$$v_c = \begin{bmatrix} -z_e \\ -z_e \\ \frac{-z_e(f+n)-2\cdot f\cdot n}{f-n} \\ -z_e \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{-r\cdot z_e}{n} \\ \frac{-t\cdot z_e}{n} \\ z_e \\ 1 \end{bmatrix} \tag{22}$$



substitute $z_e = -n$ in (22):

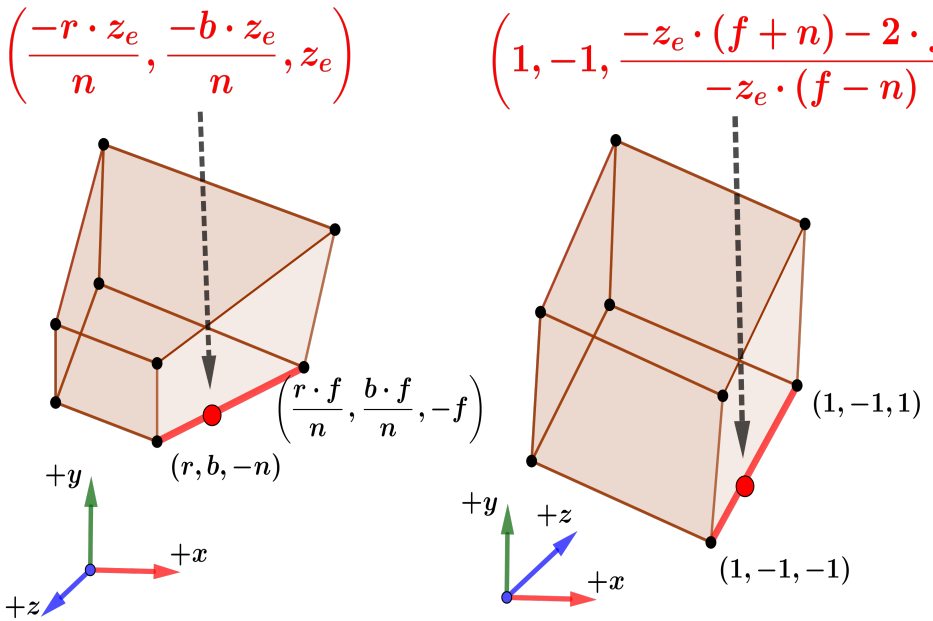
$$v_c = \begin{bmatrix} n \\ n \\ -n \\ n \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} r \\ t \\ -n \\ 1 \end{bmatrix} \implies ndc = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \tag{23}$$

substitute $z_e = -f$ in (22):

$$v_c = \begin{bmatrix} f \\ f \\ f \\ f \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{r\cdot f}{n} \\ \frac{t\cdot f}{n} \\ -f \\ 1 \end{bmatrix} \implies ndc = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{24}$$

Projection matrix

$$v_c = \begin{bmatrix} -z_e \\ z_e \\ \frac{-z_e(f+n)-2\cdot f\cdot n}{f-n} \\ -z_e \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{-r\cdot z_e}{n} \\ \frac{-b\cdot z_e}{n} \\ z_e \\ 1 \end{bmatrix} \tag{25}$$



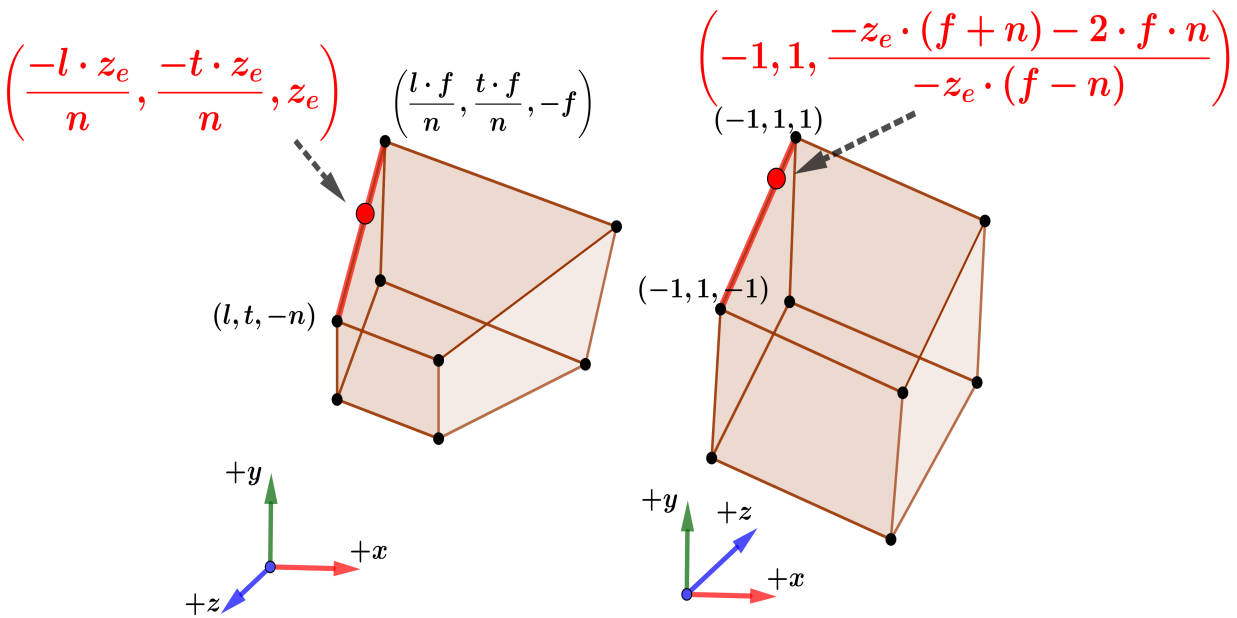
substitute $z_e = -n$ in (25):

$$v_c = \begin{bmatrix} n \\ -n \\ -n \\ n \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} r \\ b \\ -n \\ 1 \end{bmatrix} \implies ndc = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \tag{26}$$

substitute $z_e = -f$ in (25):

$$v_c = \begin{bmatrix} f \\ -f \\ f \\ f \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{r\cdot f}{n} \\ \frac{b\cdot f}{n} \\ -f \\ 1 \end{bmatrix} \implies ndc = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \tag{27}$$

$$v_c = \begin{bmatrix} z_e \\ -z_e \\ \frac{-z_e(f+n)-2\cdot f\cdot n}{f-n} \\ -z_e \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{-l\cdot z_e}{n} \\ \frac{-t\cdot z_e}{n} \\ z_e \\ 1 \end{bmatrix} \tag{28}$$



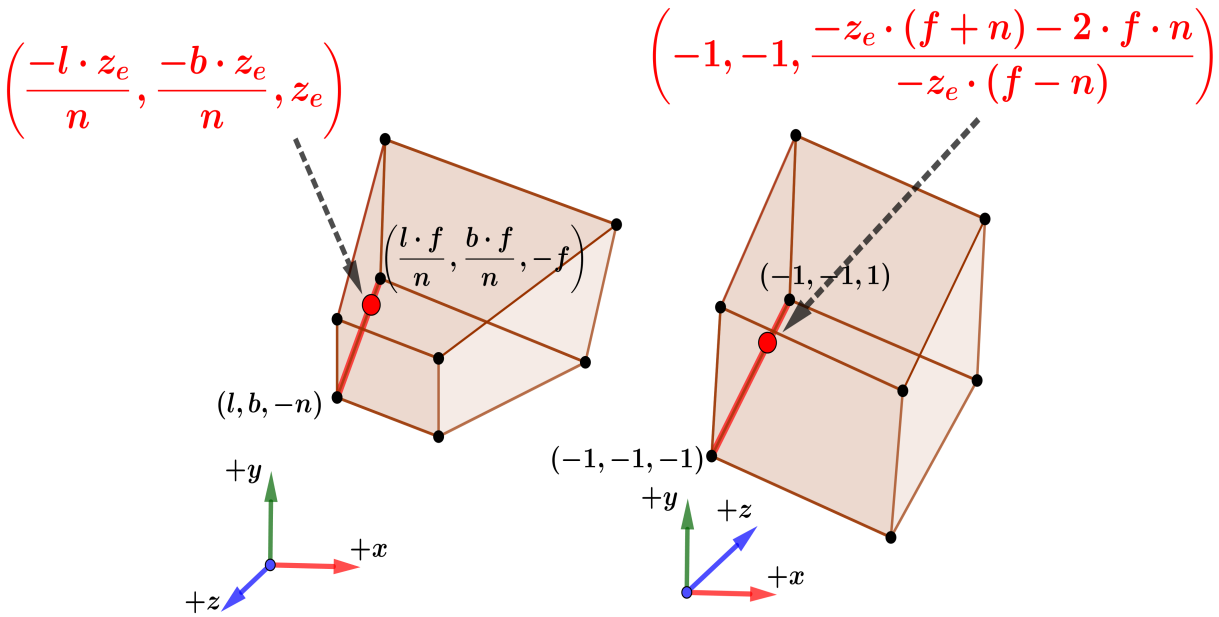
substitute $z_e = -n$ in (28):

$$v_c = \begin{bmatrix} -n \\ n \\ -n \\ n \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} l \\ t \\ -n \\ 1 \end{bmatrix} \implies ndc = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \tag{29}$$

substitute $z_e = -f$ in (28):

$$v_c = \begin{bmatrix} -f \\ f \\ f \\ f \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{l\cdot f}{n} \\ \frac{t\cdot f}{n} \\ -f \\ 1 \end{bmatrix} \implies ndc = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \tag{30}$$

$$v_c = \begin{bmatrix} z_e \\ z_e \\ \frac{-z_e(f+n)-2\cdot f\cdot n}{f-n} \\ -z_e \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{-l\cdot z_e}{n} \\ \frac{-b\cdot z_e}{n} \\ z_e \\ 1 \end{bmatrix} \tag{31}$$



substitute $z_e = -n$ in (31):

$$v_c = \begin{bmatrix} -n \\ -n \\ -n \\ n \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} l \\ b \\ -n \\ 1 \end{bmatrix} \implies ndc = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \tag{32}$$

substitute $z_e = -f$ in (31):

$$v_c = \begin{bmatrix} -f \\ -f \\ f \\ f \end{bmatrix} = \begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{l\cdot f}{n} \\ \frac{b\cdot f}{n} \\ -f \\ 1 \end{bmatrix} \implies ndc = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \tag{33}$$

We can see that the *projection* matrix is built such that the corners of the truncated pyramid will transform (after division by w) into the corners of a cube of $size = 2$ centered at the origin.
Therefore,

$$\text{ndc} = \begin{bmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{bmatrix} \tag{34}$$

satisfies:

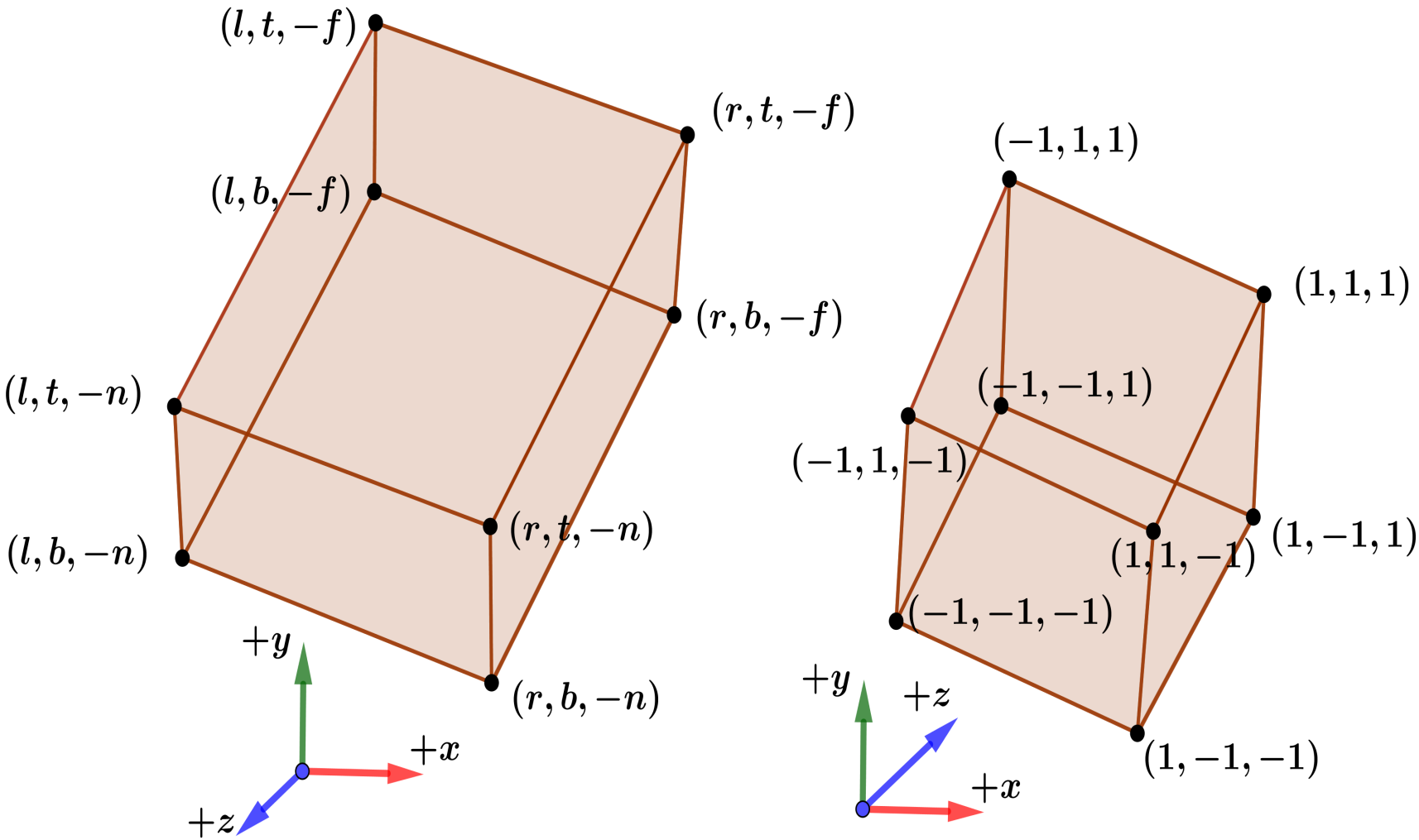
$-1 \leq x_{ndc} \leq 1$

$-1 \leq y_{ndc} \leq 1$

$-1 \leq z_{ndc} \leq 1$

$$\textbf{Orthographic projection matrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{35}$$

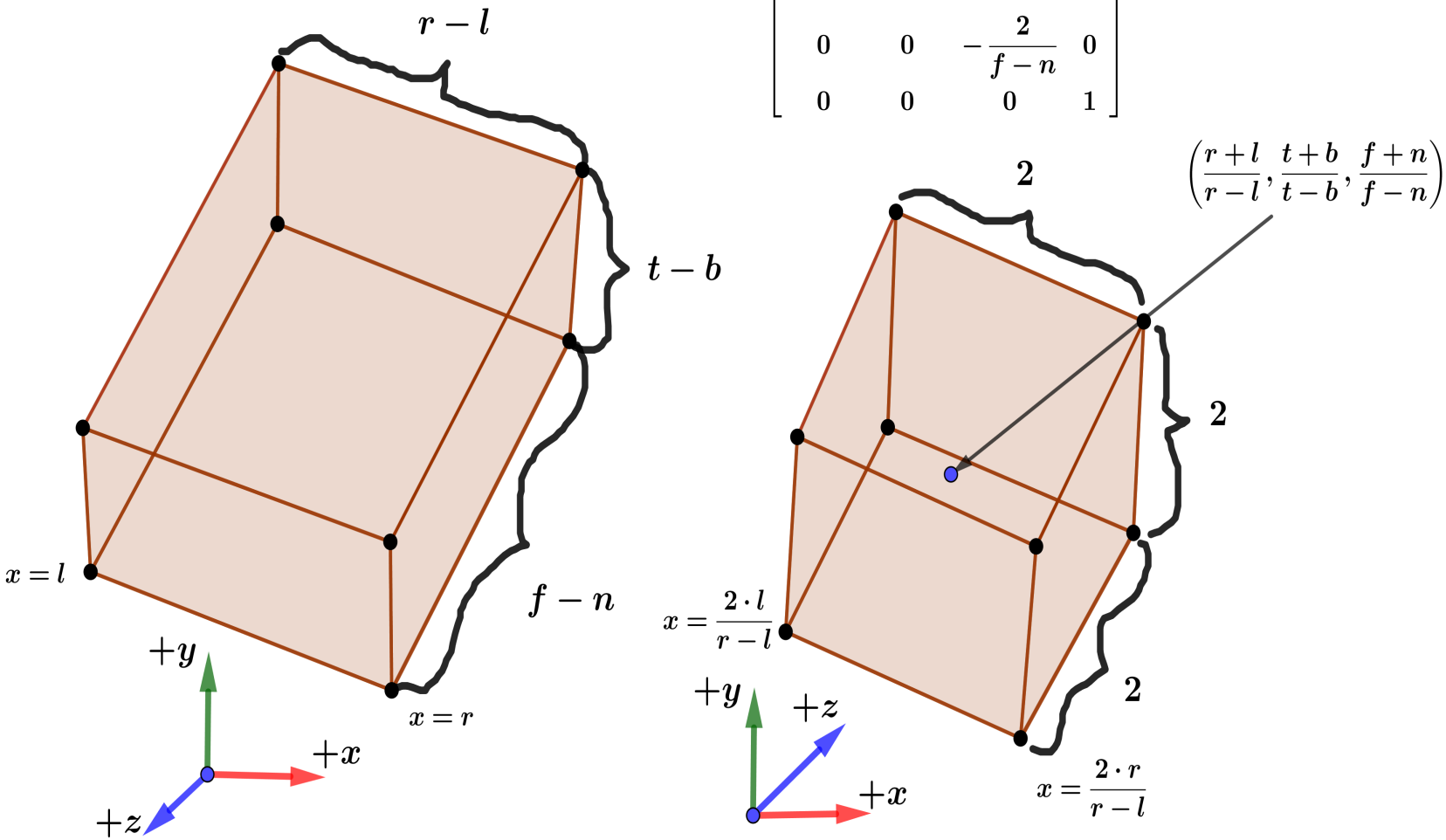
Orthographic projection matrix



Orthographic projection matrix

first we scale the box to cube

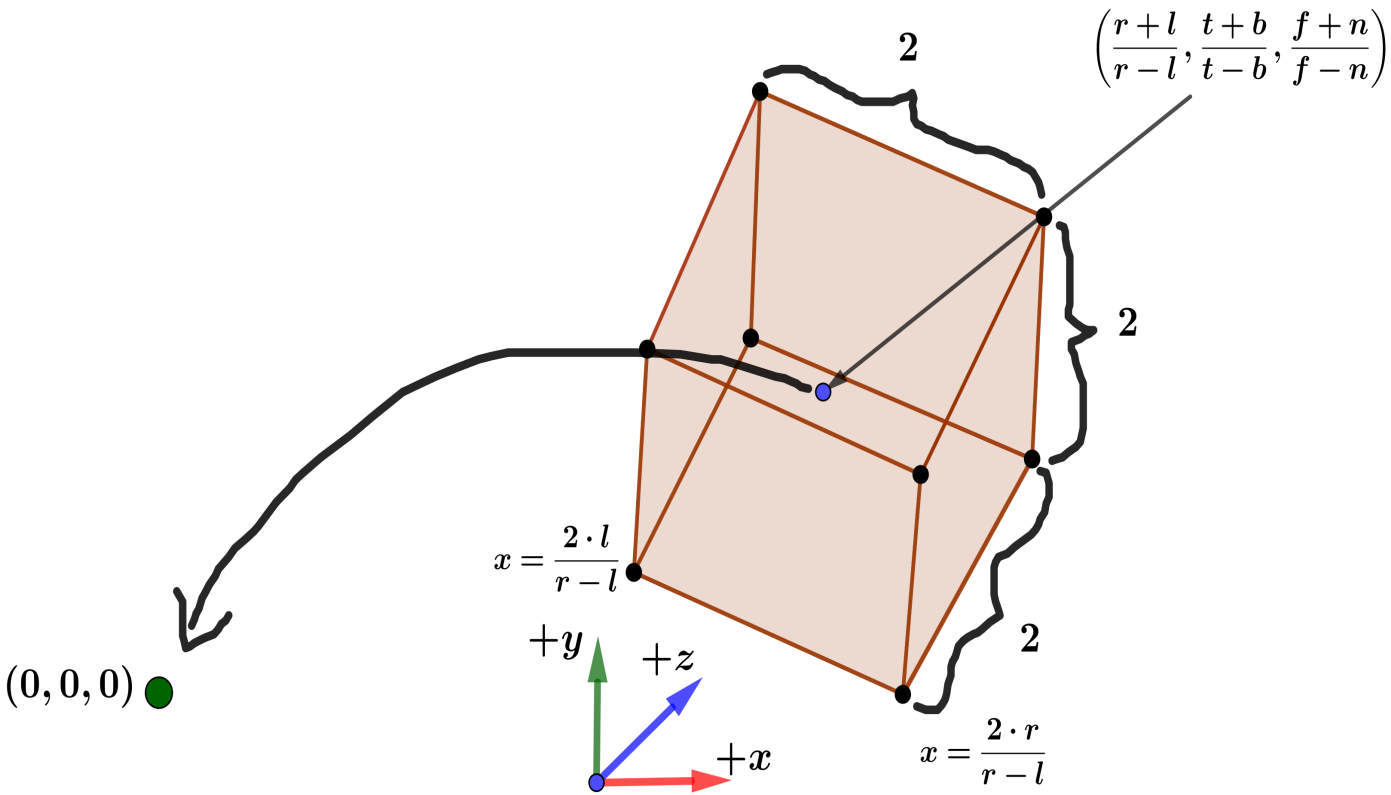
$$\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Orthographic projection matrix

then we translate the cube such that its center is at the origin

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



Orthographic projection matrix

All together the orthographic projection from a box to a cube centered at the origin is:

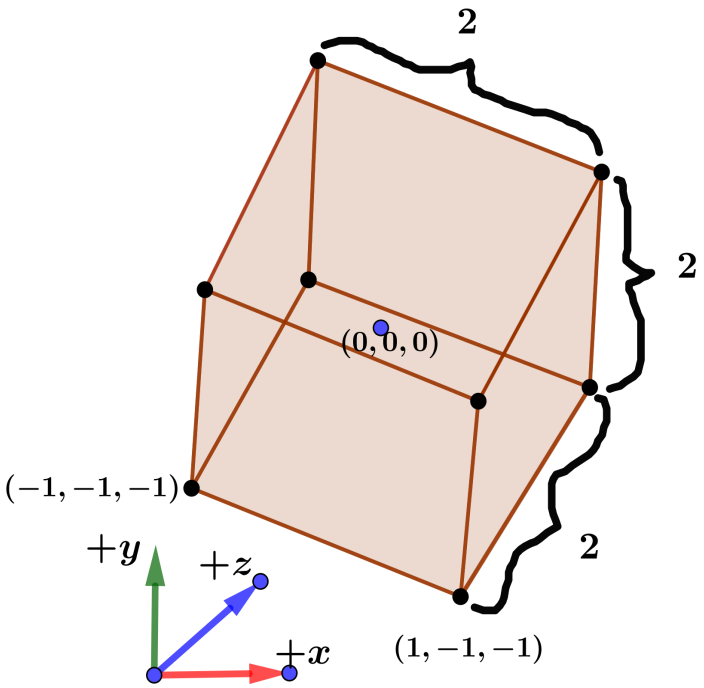
$$\begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & 1 & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & 1 & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{36}$$

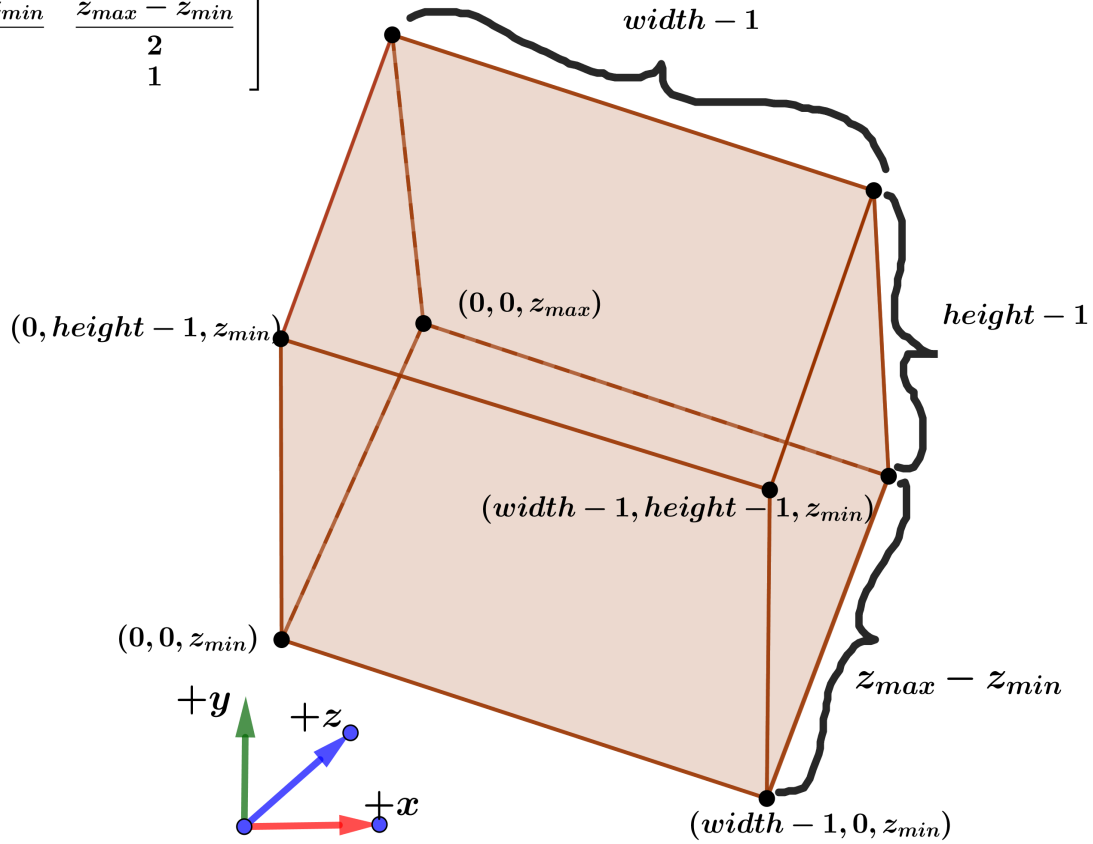
Orthographic projection matrix

From NDC to screen coordinates (scale and then translate)

$$\begin{bmatrix} \frac{width-1}{2} & 0 & 0 & \frac{width-1}{2} \\ 0 & \frac{height-1}{2} & 0 & \frac{height-1}{2} \\ 0 & 0 & \frac{z_{max}-z_{min}}{2} & \frac{z_{max}-z_{min}}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By default z_{max}=1 and z_{min}=0





Orthographic projection matrix

Over all, in orthographic projection, the calculations from object coordinates to pixel and depth coordinates can be done by scale/translate/rotate matrices multiplication without dividing by w on the way.

At the end, calculate $round(pixel_x)$ and $round(pixel_y)$.

Orthographic projection

In orthographic projection all the matrices are scale/translate/rotate:

$$\begin{bmatrix} \frac{width-1}{2} & 0 & 0 & \frac{width-1}{2} \\ 0 & \frac{height-1}{2} & 0 & \frac{height-1}{2} \\ 0 & 0 & \frac{z_{max}-z_{min}}{2} & \frac{z_{max}-z_{min}}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \tag{37}$$

$$\begin{bmatrix} & & & \\ & \textit{view} & & \\ & \textit{matrix} & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} & & & \\ & \textit{model} & & \\ & \textit{matrix} & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{model} \\ y_{model} \\ z_{model} \\ 1 \end{bmatrix} = \begin{bmatrix} pixel_x \\ pixel_y \\ depth \\ 1 \end{bmatrix} \tag{38}$$