Perspective Projection,

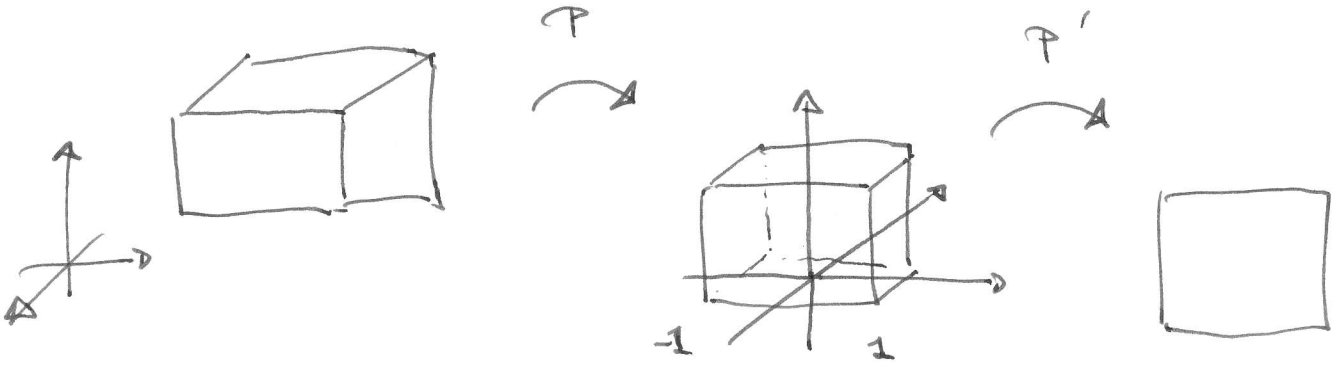
Derivation

Simple view : proj. : Wold space 3D

↓

Screen space 2D

However, to simplify clipping, we would like:



I.e., we are seeking a mapping $P : \mathbb{R}^3 \to \mathbb{R}^3$ that takes the view frustrum to the unit box $[-1, 1]^3$.

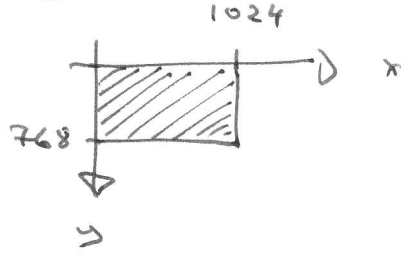
[Clipping can then be hardcoded on GPU (is same operation for all frustrums).]

Doing this for the ortographic proj. is simple, since the view box defined by (r, l, t, b, n, f) is already an axis aligned box. As seen on pages 645-647, simple translation and scaling is enough.

We here derive how to do it for the perspective proj. (ie., find P).

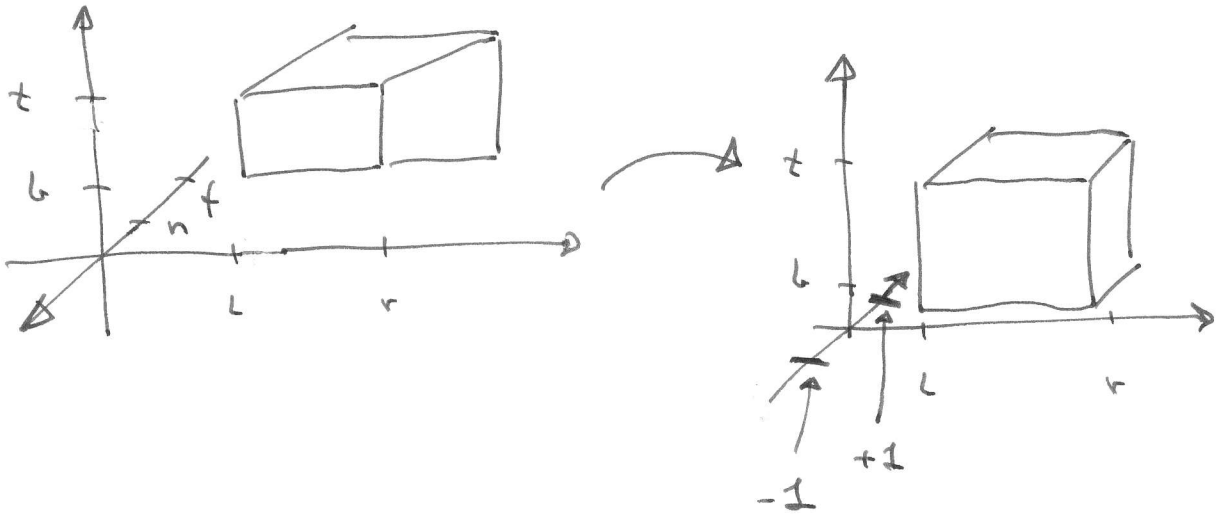
Note: the resulting mapping P is called perspective projection, even though the actual projection takes place in P'. Think of P as a "perspective rectifying" that allows the proj. P' to be ortographic. (P' just discards the z-coordinate)

Note: To actually get to screen space, scaling and translation (in 2D) is done after discarding the z-coordinate, to get to whatever coordinate system is used on the screen/raster, e.g.:



Note: The z-coord. is not used for screen positioning, but is used for z-buffer tests, eg. [so not literally discarded].

We start by creating a mapping $\bar{P}$ giving an axis-aligned box with $z \in [-1, 1]$ when applied to the view frustrum. This is the hard part. Later, we get $x, y \in [-1, 1]$ by simple translation and scaling (as for orthographic projection).



So we want

$$\bar{P}(\text{plane } z = n) = \text{plane } z = -1$$
$$\bar{P}(\text{plane } z = f) = \text{plane } z = +1$$

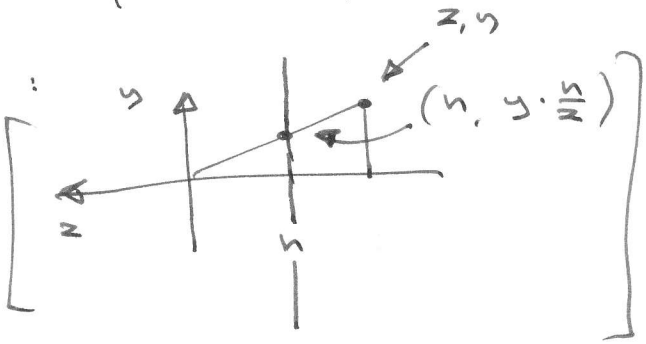
We also want

$$\bar{P}_x \begin{pmatrix} x \\ y \\ z \end{pmatrix} = x \cdot \frac{n}{z}$$
$$\bar{P}_y \begin{pmatrix} x \\ y \\ z \end{pmatrix} = y \cdot \frac{n}{z}$$

such that

discarding the z-values (using $\bar{P}'$) will leave the (x, y)-values req. by the perspective projections definition.

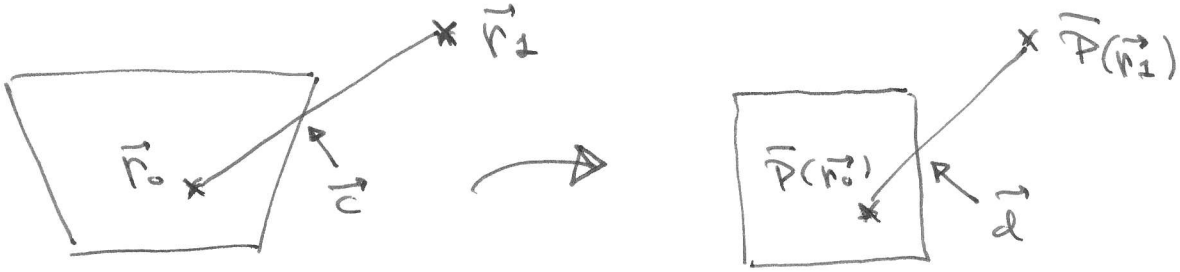
Cf. previous note-set:



We also want clipping after $\bar{P}$ transf. to be equivalent to clipping using the original frustrum's side planes.

To explain this (seen from the top, ie. only showing x and z axes — similar view for y and z axes, of course):

Mapping by $\bar{P}$ of a line segment is a mapping of its two endpoints:



After the mapping, the intersection $\vec{d}$ of the line segment $\vec{P}(\vec{r}_0)\vec{P}(\vec{r}_1)$ with the box is calculated (during clipping). We want of $\vec{P}$ that the intersection $\vec{c}$ of $\vec{r}_0\vec{r}_1$ with the view frustum fulfills $\vec{P}(\vec{c}) = \vec{d}$

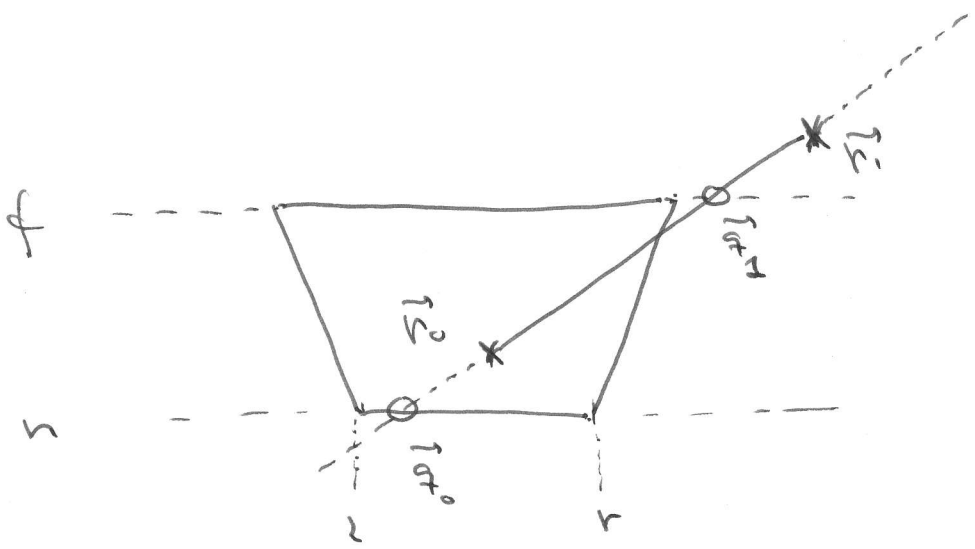
Else, clipping after $\vec{P}$ is not the same as before applying $\vec{P}$.
(and inside/outside decisions)

In previous note set, we developed the connection between a line segment before and after perspective projection $\left[\vec{f}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \cdot \frac{n}{z} \\ y \cdot \frac{n}{z} \end{pmatrix} \right]$.

We exploit it here.

First, extend $\vec{r}_0\vec{r}_1$ to the full line, and then represent this using its intersection

points $\vec{q}_0$, $\vec{q}_1$ with the near and far plane:



Full line: $\vec{q}(s) = \vec{q}_0 + (\vec{q}_1 - \vec{q}_0) \cdot s$, $s \in \mathbb{R}$

[this req. the line to not be parallel to near plane, but checking that the final mapping $\tilde{P}$ works also for clipping such lines is easy.]

From previous note set, we know

*) $\vec{f}(\vec{q}(s)) = \vec{f}(\vec{q}_0) + \left(\vec{f}(\vec{q}_1) - \vec{f}(\vec{q}_0)\right) \cdot t$

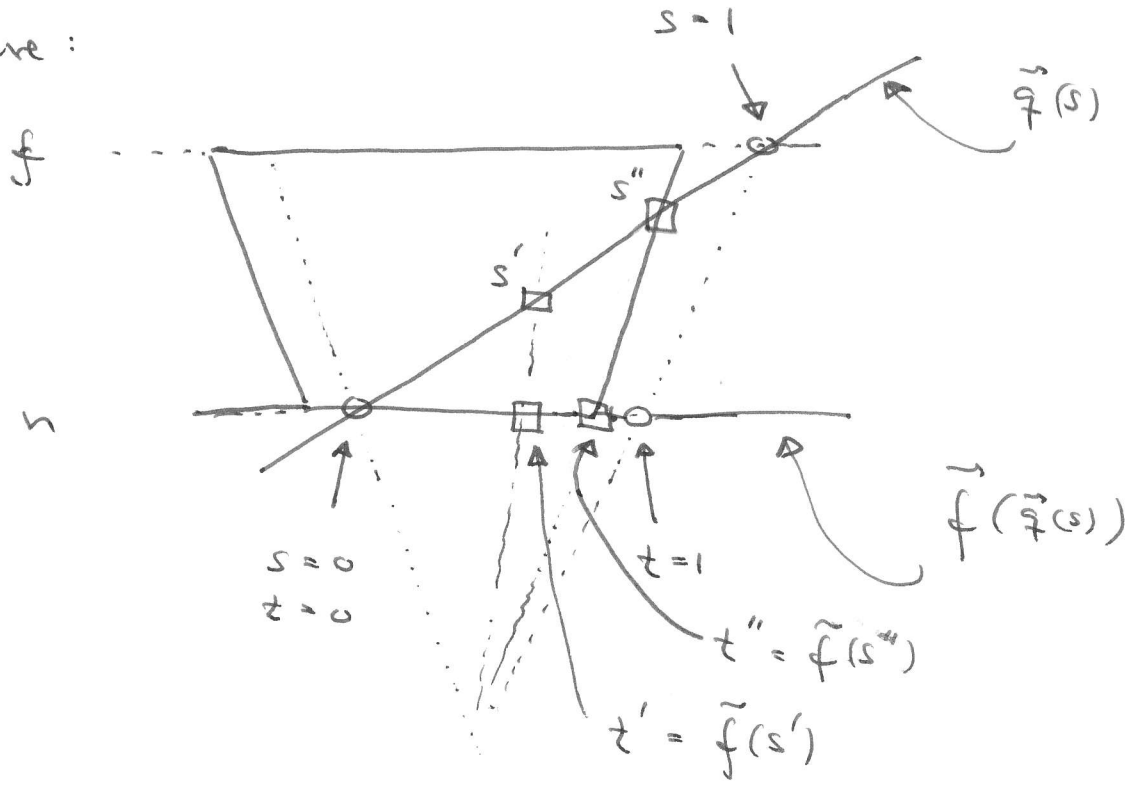
with $t = \tilde{f}(s)$

$= \dfrac{\omega_1 \cdot s}{\omega_0 + (\omega_1 - \omega_0) \cdot s}$

$\left(\vec{q}_0 = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}, \vec{f}(\vec{q}_0) = \begin{pmatrix} x_0' \\ y_0' \end{pmatrix}, \vec{q}_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}, \vec{f}(\vec{q}_1) = \begin{pmatrix} x_1' \\ y_1' \end{pmatrix} \right)$

$\left(\omega_0 = z_0/n, \quad \omega_1 = z_1/n \right)$

Figure:

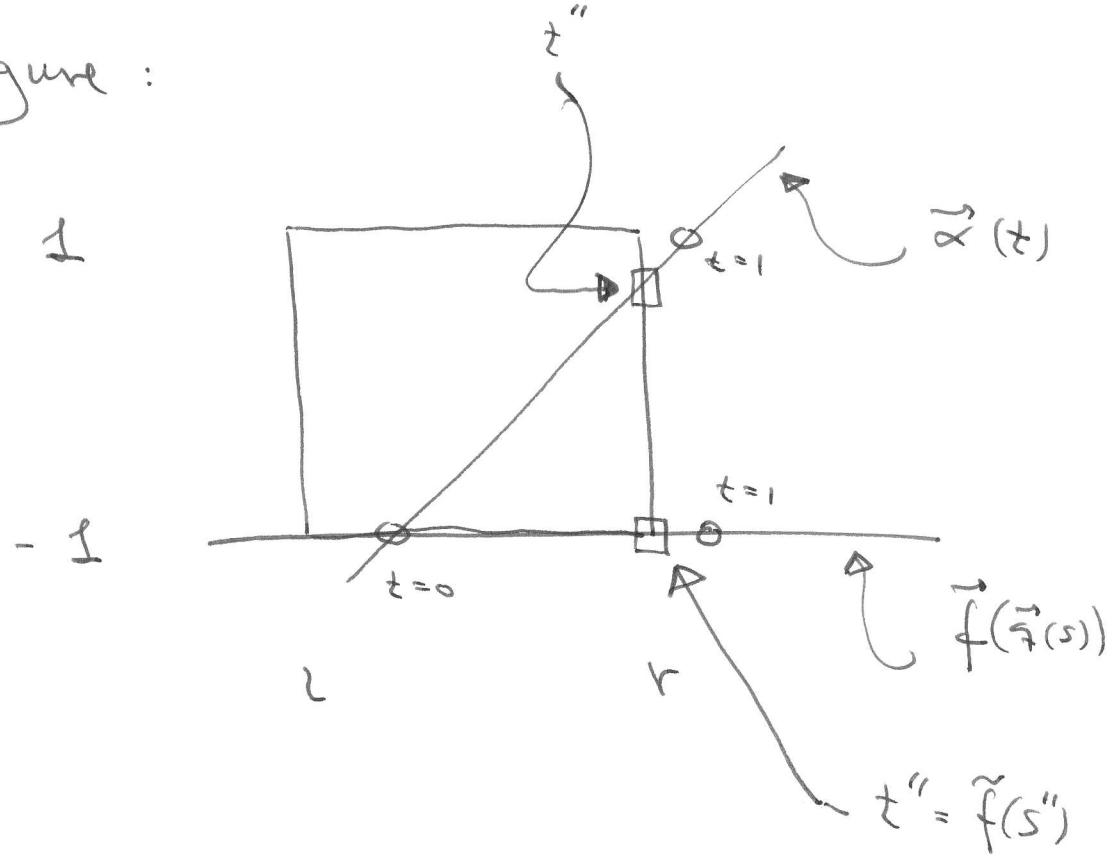


We now construct the following line $\vec{\alpha}(t)$ that we map $\vec{q}(s)$ to [keeping the connection $t = \tilde{f}(s)$ between line parameters]:

$$\vec{\alpha}(t) = \begin{pmatrix} x_0' \\ y_0' \\ -1 \end{pmatrix} + \begin{pmatrix} x_1' - x_0' \\ y_1' - y_0' \\ 1 - (-1) \end{pmatrix} \cdot t$$

Note: this is the same as the 2D line *), but with a z-coordinate added, which runs from -1 to 1 for $t \in [0;1]$.

Figure :



By design, the intersection of $\vec{\alpha}(t)$ with the box $[l;r] \times [b;t] \times [-1;1]$ happens at t''.

More generally,

$\vec{q}(s) \in$ frustrum

$\Updownarrow$

$s \in [0;1]$ and $\vec{f}(\vec{q}(s)) \in [l;r] \times [t;b]$

$\Updownarrow$

$\vec{\alpha}(\tilde{f}(s)) = \vec{\alpha}(t) \in$ box

Thus, for points $\vec{q}(s)$, mapping them to $\vec{\alpha}(\tilde{f}(s))$ is a mapping fulfilling what we want.

But all points in $\mathbb{R}^3$ are on such a line (on many, of course). As we will see below, we can express the above mapping without mentioning of the lines. Hence, the mapping is really one mapping $\mathbb{R}^3 \to \mathbb{R}^3$, and we know (by construction) that

i) clipping of lines ~~after~~ before mapping is the same as clipping against the frustum before.

ii) discarding z-coordinates leaves $\begin{pmatrix} x \cdot \frac{n}{z} \\ y \cdot \frac{n}{z} \end{pmatrix}$ so P' works as expected.

against the box $[l;r] \times [b;t] \times [-1;-1]$

We now express the mapping without the lines, i.e. in terms of x, y, z.

$$\left[\vec{q}(s) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right].$$

By construction, we already know

$$\vec{\alpha}_x(t) = \vec{f}_x\left(\vec{\gamma}(s)\right) = \vec{f}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = x \cdot \frac{n}{z}$$

$$\vec{\alpha}_y(t) = \text{---}''\text{---} = y \cdot \frac{n}{z}$$

Remaining is

$$\vec{\alpha}_z(t) = 2 \cdot t - 1$$

$$= 2 \cdot \tilde{f}(s) - 1$$

Since $z = \vec{\gamma}_z(s) = z_0 + (z_1 - z_0) \cdot s$

$$= n + (f - n) \cdot s,$$

we have $\dfrac{z-n}{f-n} = s$. Also: $w_0 = \dfrac{z_0}{n} = \dfrac{n}{n} = 1$

$$w_1 = \frac{z_1}{n} = \frac{f}{n}$$

Hence $\vec{\alpha}_z(t) = 2 \cdot \tilde{f}(s) - 1$

$$= 2 \cdot \frac{w_1 \cdot s}{w_0 + (w_1 - w_0) \cdot s} - 1$$

$$= 2 \cdot \frac{w_1}{\frac{w_0}{s} + (w_1 - w_0)} - 1$$

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

$$= 2 \cdot \frac{f/n}{\frac{f-n}{z-n} + \frac{f-n}{n}} - 1$$

$$= 2 \cdot \frac{f}{f-n} \cdot \frac{1}{\frac{n}{z-n} + 1} - 1$$

$$= 2 \cdot \frac{f}{f-n} \cdot \frac{1}{\frac{z}{z-n}} - 1$$

$$= 2 \cdot \frac{f}{f-n} \cdot \left(1 - \frac{n}{z}\right) - 1$$

$$= \frac{2f - \frac{2fn}{z} - (f-n)}{f-n}$$

$$= \frac{f+n-\frac{2fn}{z}}{f-n}$$

Thus, the sought mapping is:

$$\overline{P}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \cdot \frac{n}{z} \\ y \cdot \frac{n}{z} \\ \frac{f+n-\frac{2fn}{z}}{f-n} \end{pmatrix}$$

As a final step, we convert [translate and scale] x and y expressions to lie inside $[-1;1]$ instead of $[l;r]$ and $[b;t]$ (for points in frustrum).

Since scalings and translations are affine transformations, they maintain convex combinations, ie.:

$$T(\vec{r}_0 + (\vec{r}_1 - \vec{r}_0)\cdot s) = T(\vec{r}_0\cdot(1-s) + \vec{r}_1 \cdot s)$$

$$= T(\vec{r}_0)\cdot(1-s) + T(\vec{r}_1)\cdot s$$

$$= T(\vec{r}_0) + \left(T(\vec{r}_1) - T(\vec{r}_0)\right)\cdot s$$

(see algebraic manipulations (easy) above middle of p. 185)

Hence, intersection points calculated after this mapping will automatically be correct before the mapping [ie., same value of s gives the intersection point, so the issue discussed on page (4)-(5) above [$\vec{c}$ vs. $\vec{d}$] does not arise].

The actual mapping is as for the ortographic projection (p. 645-647): We center and scale each coordinate individually. For x:

$$x' \longrightarrow \left(x' - \underbrace{\frac{r+l}{2}}_{\text{center}}\right) \cdot \underbrace{\frac{2}{r-l}}_{\substack{\text{scale to} \\ \text{width two.}}}$$

Similar for y.

Hence, the final mapping P is

$$P\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \cdot \frac{n}{z} \cdot \frac{2}{r-l} - \frac{r+l}{r-l} \\ y \cdot \frac{n}{z} \cdot \frac{2}{t-b} - \frac{t+b}{t-b} \\ \frac{f+n-\frac{2fn}{z}}{f-n} \end{pmatrix}$$

From "OpenGL Programming Guide":

Perspective Projection

The call **glFrustum**(*l*, *r*, *b*, *t*, *n*, *f*) generates **R**, where

$$\mathbf{R} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \text{ and } \mathbf{R}^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$

R is defined as long as $l \neq r$, $t \neq b$, and $n \neq f$.

Note: n and f here are our n- and f-values negated (to be positive values, apparently believed nicer to work with).

The reader is encouraged to check that R, after homogenization, gives P.

I.e. find $\begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \\ w \end{pmatrix} = R \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ and check $\begin{pmatrix} \bar{x}/w \\ \bar{y}/w \\ \bar{z}/w \end{pmatrix} = P\begin{pmatrix} x \\ y \\ z \end{pmatrix}$.

[don't forget to negate n and f, cf. note 5].