I. Camera.

1. For a camera. the following parameter matters:

① Position : $\vec{e} \xrightarrow{\text{Goes to}} \vec{0}$

② View Direction : $\hat{g} \xrightarrow{\text{Goes to}} -\hat{z}$

③ Up direction : $\hat{t} \xrightarrow{\text{Goes to}} \hat{y}$

We want to transform everything into the coord space where camera is at the origin, looking directly to $-\hat{z}$.

Step 1 Translate $-\vec{e}$ $\Rightarrow$ $T_{-\vec{e}} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Step 2. Rotate from $\hat{t}$ to $\hat{y}$, $\hat{g}$ to $-\hat{z}$.

Ex Arbitrary 3D Rotation.

- 3D Rotation Matrices are all orthonormal matrices, which means the 3 rows are mutually orthogonal unit vectors.

- Let $R_{uvw} = \begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{bmatrix}$, this represents $R_{uvw} = \begin{bmatrix} -\hat{u}- \\ -\hat{v}- \\ -\hat{w}- \end{bmatrix}$, for example, $\hat{u} = x_u\hat{x} + y_u\hat{y} + z_u\hat{z}$.

Notice that $R_{uvw}\,\hat{u} = \begin{bmatrix} \hat{u}\cdot\hat{u} \\ \hat{v}\cdot\hat{u} \\ \hat{w}\cdot\hat{u} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \hat{x}$

Similarly, $R_{uvw}\,\hat{v} = \hat{y}$, $R_{uvw}\,\hat{w} = \hat{z}$.

Thus. R_{uvw} takes the basis $\{u, v, w\}$ to the corresponding Cartesian axes.

On the other hand. since R_{uvw}^T is also the reverse of R_{uvw} (because it is orthonormal).

Thus. $R_{uvw}^T\,\hat{x} = \hat{u}$, $R_{uvw}^T\,\hat{y} = \hat{v}$, $R_{uvw}^T\,\hat{z} = \hat{w}$. → R_{uvw}^T rotates a x,y,z-based point to a u,v,w-based position.

If we wish to rotate about an arbitrary vector $\hat{a}$, we form an orthonormal basis with $\hat{w} = \hat{a}$.

Now, in the case of camera space transformation, both xyz and uvw are orthonormal so this transform must exist.

Consider its inverse rotation R^{-1} i.e. rotate $\hat{x}$ to $(\hat{g}\times\hat{t})$, $\hat{y}$ to $\hat{t}$, $\hat{z}$ to $-\hat{g}$.

$$R^{-1}(\hat{g}\times\hat{t}) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = x \quad \Rightarrow \quad Rx = \hat{g}\times\hat{t}.$$

i.e.

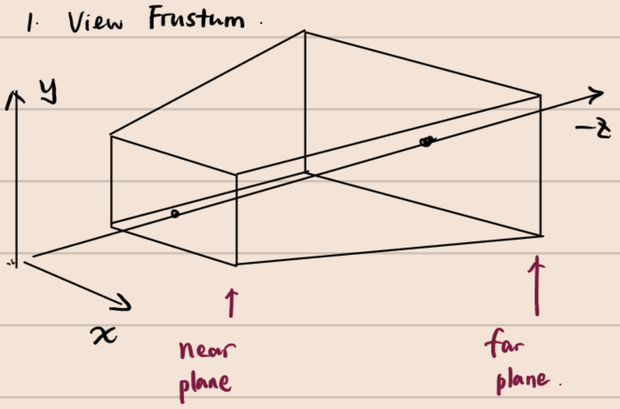
We call this the Canonical-to-Basis Matrix.

$$R^{-1} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & x_t & x_{-g} & 0 \\ y_{\hat{g}\times\hat{t}} & y_t & y_{-g} & 0 \\ z_{\hat{g}\times\hat{t}} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow R = (R^{-1})^{-1} = (R^{-1})^T = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t & y_t & z_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \hat{g}\times\hat{t} & \hat{t} & -\hat{g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

$$M_{cam} = R \cdot \begin{bmatrix} I & -\vec{e} \\ \vec{0} & 1 \end{bmatrix}$$

II. Projection.

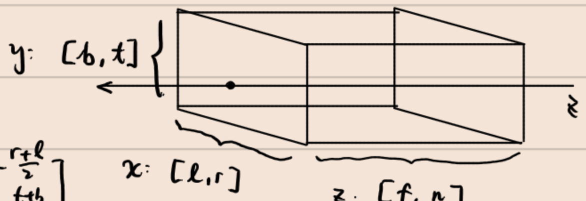
1. View Frustum.



The region that the camera can see.

We want to "compress" this frustum into a $[-1,1]^3$ cubic box. (which faciliates screen transform).

2. Orthographic Projections (Frustum is itself a box).



Step 1 Move center to the origin. (T_{center}).

Step 2 Scale to $[-1,1]^3$. (S).

$$M_{orth} = S \cdot T_{center} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & -\frac{r+l}{2} \\ & I & & -\frac{t+b}{2} \\ & & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

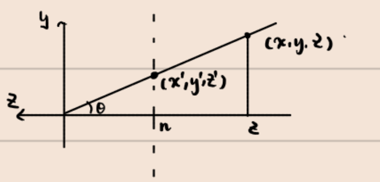
$(n \geq f)$.

Notice that since we are looking into $-z$, $n \geq f$.

3. Perspective Projection.

① Homogeneous Coord : $(x, y, z, 1) \mapsto (xz, yz, z^2, z)$. i.e. $[x\ y\ z\ w]^T \mapsto (x/w, y/w, z/w)$

② Transform.



$$\frac{y'}{y} = \frac{n}{z} \Rightarrow y' = \frac{ny}{z}$$

$$\frac{x'}{x} = \frac{n}{z} \Rightarrow x' = \frac{nx}{z}.$$

In homogeneous coord.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \xrightarrow{\text{transf.}} \begin{bmatrix} nx/z \\ ny/z \\ ? \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} nx \\ ny \\ z? \\ z \end{bmatrix}$$

i.e. $M_{projection} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ ? \\ z \end{bmatrix}$

$$\Rightarrow M_{proj} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Now, solve with particular points.

$$M_{proj} \begin{bmatrix} x \\ y \\ n \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ n \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} nx \\ ny \\ n^2 \\ n \end{bmatrix}$$

a point on the near plane stays on the near plane

3rd row. point on near plane.

i.e. $[A\ B\ C\ D]^T \begin{bmatrix} x \\ y \\ n \\ 1 \end{bmatrix} = n^2 \Rightarrow A = 0,\ B = 0.\quad Cn + D = n^2$

point on far plane.

$[A\ B\ C\ D]^T \begin{bmatrix} 0 \\ 0 \\ f \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 0 \\ 0 \\ f^2 \\ f \end{bmatrix} \Rightarrow Cf + D = f^2$

$\Rightarrow C = n + f.\quad D = -nf.$

The projection / perspective matrix is then

$$P = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

For any point $[x\ y\ z\ 1]^T$, P is transforming it to

$$P\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ (n+f)z - nf \\ z \end{bmatrix} \overset{\text{equivalent to}}{\sim} \begin{bmatrix} nx/z \\ ny/z \\ n+f-\frac{nf}{z} \\ 1 \end{bmatrix}$$

③ Note that after apply P, we will now have an orthogonal axis-aligned box.

Thus, the full set of matrices for perspective viewing is

$$M = M_{viewport}\ \boxed{M_{orth}\ P\ M_{cam}}$$ → they produce a canonical position.

$$= M_{viewport}\ \boxed{M_{perspective}\ M_{cam}}.$$

$[-1,1]^3$ ... e.g. $[0,1920] \times [0,1080] \times \{1\}$..

where $M_{viewport}$ transforms canonical coordinates to screen base coordinates

III. Field Of View.

We have determined the "depth" of the view frustum (n, f), and yet we haven't decide the horizontal and vertical "span" of the viewport.

These are determined by l, r, b, t.

1. Simplification.

① If we are always looking through the center of the window, we can assume $l = -r$, $b = -t$.

② If the pixels are square, then $n_x / n_y = r/t$. n_x, n_y are # pixels on x, y directions.

2. Vertical FOV θ.

Def $\tan\frac{\theta}{2} = \frac{t}{|n|}$. n is the depth of near clipping plane.