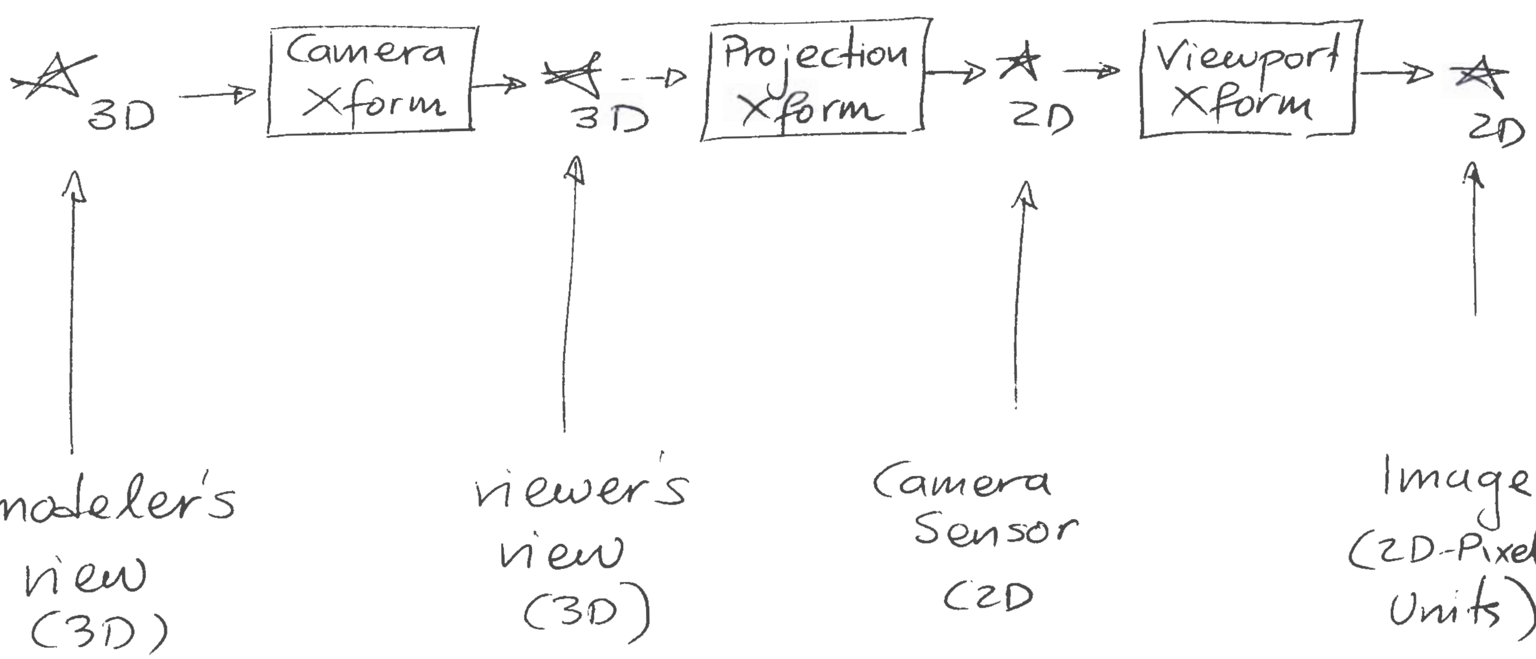
Viewing transformations

One of the most direct uses of transformations (and the algebra that comes along with them) can be appreciated in the process of <u>viewing</u>: the formation of a 2D <u>geometric</u> description from a 3D representation of the world.

(Note this is still about geometry ... i.e. generating 2D points, lines, etc. Forming discrete pixels, colors, etc is the focus of shading/rendering which comes later on)

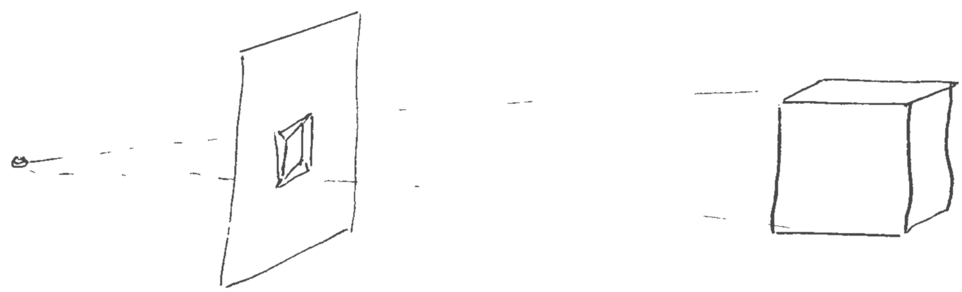
Conceptually this transformation comes in 3 stages:



Among them

→ Camera Xform : Change-of-coordinates transform (everything still stays in 3D, just a rigid body Xform) to reinterpret everything w.r.t. coordinate system affixed to camera

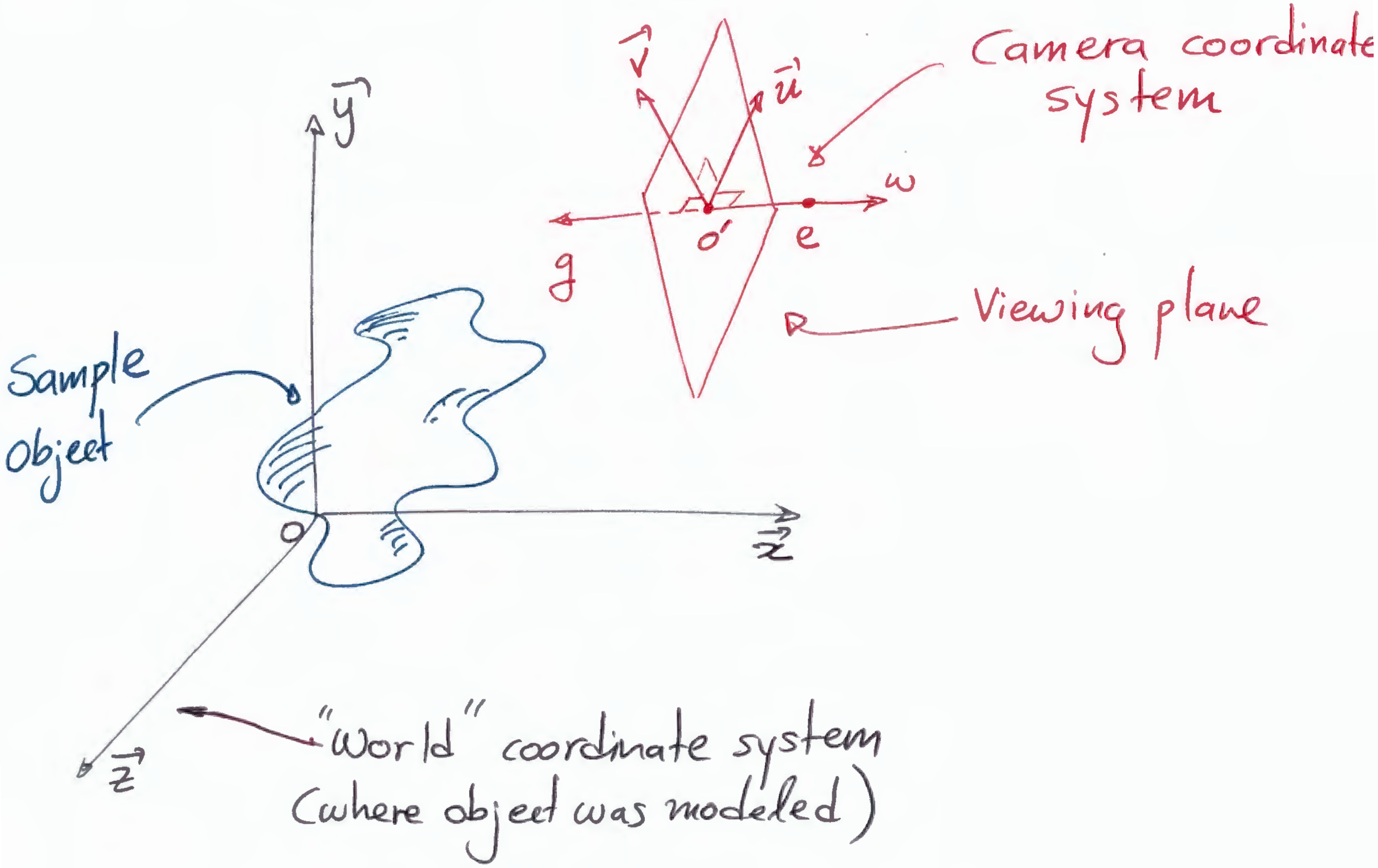
→ Projection Xform : A 2D "flattened" conversion of the 3D world



→ Viewport Xform : A 2D "adjustment of units & origin" transform, from "world" units to "display" units & re-centering.

(Also: depending on convention, sometimes the projection xform will yield "canonical" units that don't represent either world- or display-domain measurements)

Camera Transform



Geometry of camera transform

$O\vec{x}\vec{y}\vec{z}$: Reference (world/model) coordinate system

$O'\vec{u}\vec{v}\vec{w}$: Camera coordinate system

o' : Center of projection plane

e : ("eye") center of projection

$\vec{g}$: "Gaze" (or "lookat") vector

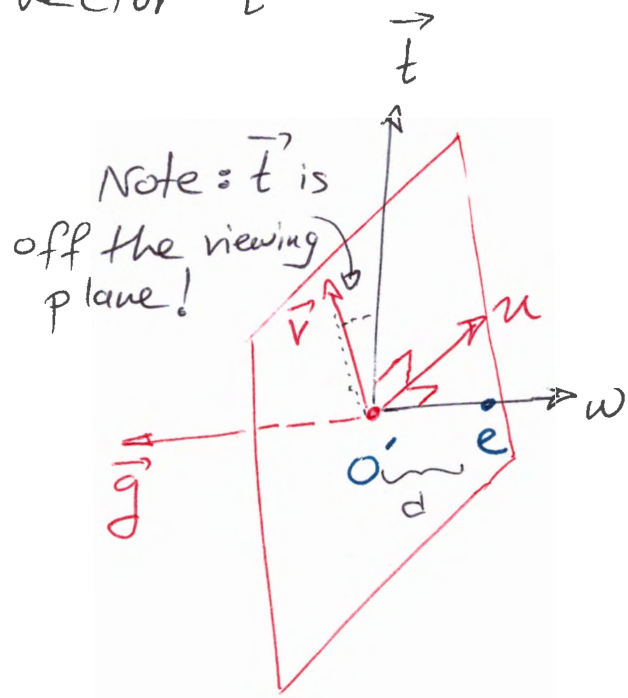
How to generate an orthonormal (& right-handed) camera coordinate system:

Start with: (user-provided)

→ Lookat/gaze vector $\vec{g}$

→ View-up ("top") vector $\vec{t}$

→ Eye location $\underline{e}$



* Invert & normalize $\vec{g}$ to get $\vec{w} = -\frac{1}{\|\vec{g}\|}\vec{g}$

* Construct $\vec{u}$ as mutually orthogonal to $\vec{t}$ & $\vec{w}$

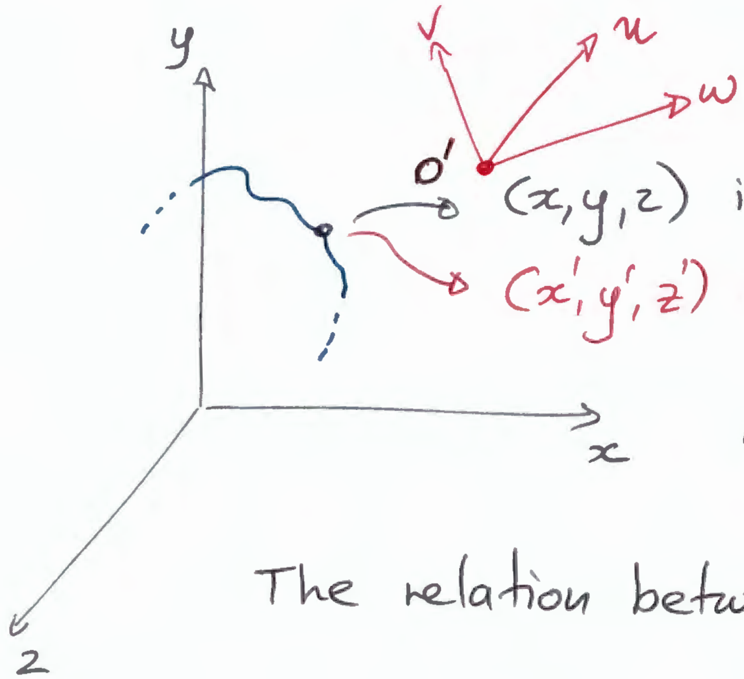
$$\vec{u} = \frac{\vec{t} \times \vec{w}}{\|\vec{t} \times \vec{w}\|}$$

* Build $\vec{v}$ as mutually orthogonal to $\vec{w}$, $\vec{u}$

$$\vec{v} = \vec{w} \times \vec{u}$$ (No need to normalize since $\vec{w} \perp \vec{u}$)

* Center of viewing plane (o') should be picked along the direction of g, starting from e ($o' = e - d \cdot \vec{w}$: d = distance).

Algebra of camera transform



(x, y, z) in world coordinates

(x', y', z') in camera coordinates

Let $o' = (o_x, o_y, o_z)$ in <u>world</u> coords

The relation between (x, y, z) and (x', y', z') is

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} o_x \\ o_y \\ o_z \end{pmatrix} = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

go homogeneous

$$\begin{bmatrix} 1 & & & -o_x \\ & 1 & & -o_y \\ & & 1 & -o_z \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} u_x & v_x & w_x & \\ u_y & v_y & w_y & \\ u_z & v_z & w_z & \\ & & & 1 \end{bmatrix}}_{\text{Rotation ; invers = transpose}} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

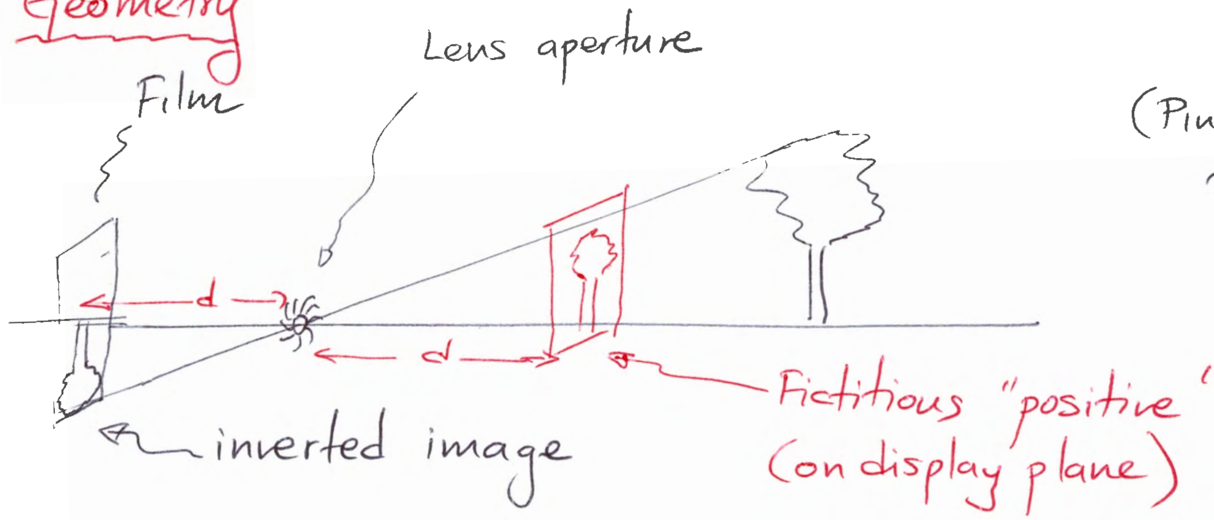
$$\rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & \\ v_x & v_y & v_z & \\ w_x & w_y & w_z & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & -o_x \\ & 1 & & -o_y \\ & & 1 & -o_z \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Projections

There are 2 steps to understanding those ... first, understanding the geometry, and then figuring out how the algebra can do the heavy lifting for you.

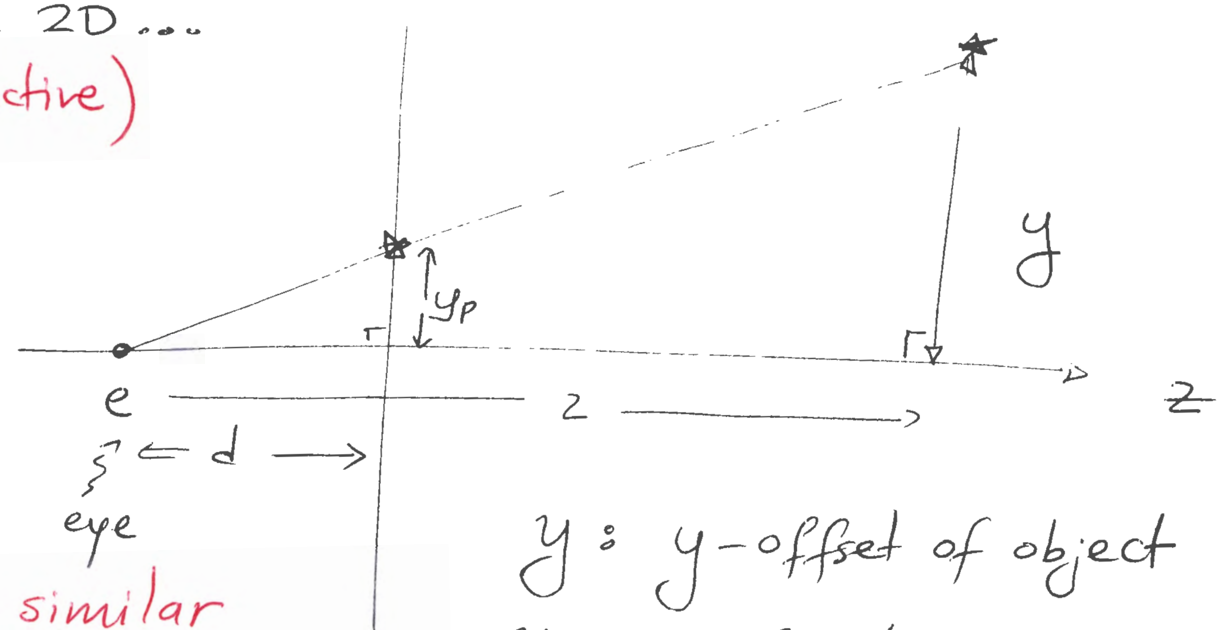
Geometry

(Pinhole camera model)



in 2D ...

(Perspective)



y : y-offset of object

y_p : y-offset along display plane

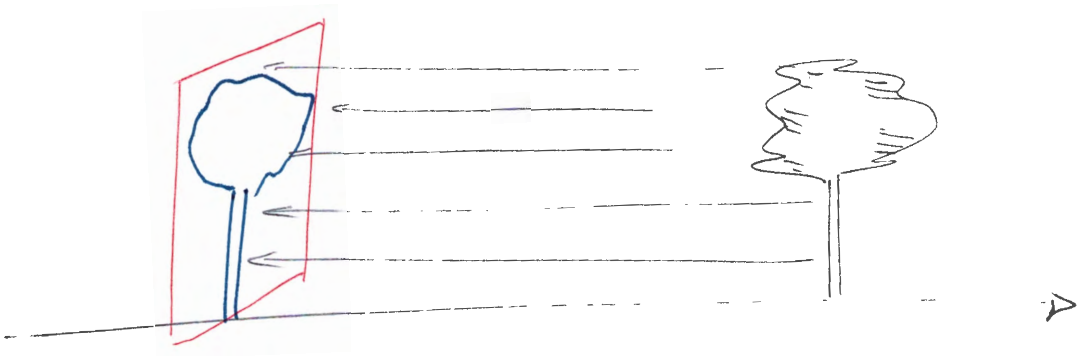
z : distance from eye

d : distance of new plane from eye.

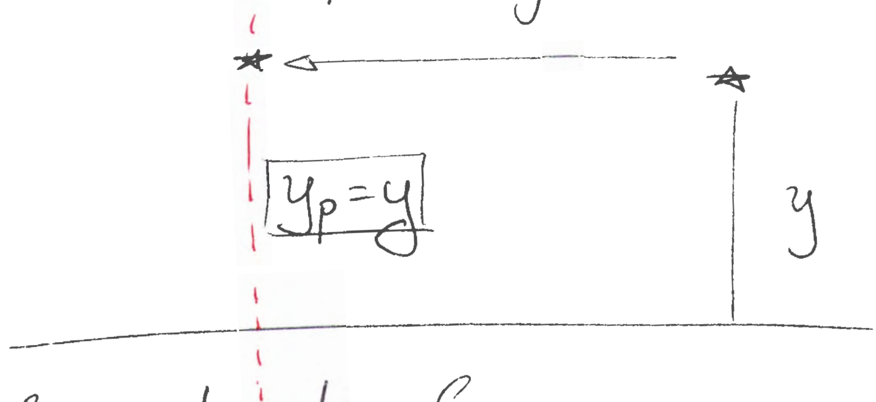
From similar triangles:

$$\frac{y_p}{d} = \frac{y}{z} \Rightarrow \boxed{y_p = d\,\frac{y}{z}}$$

Orthographic ...

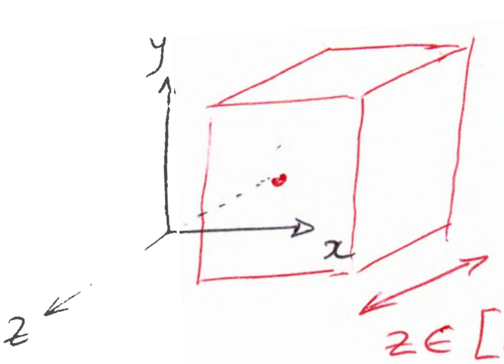


(equivalent to eye being far, far away...)



Utility of projective transforms:

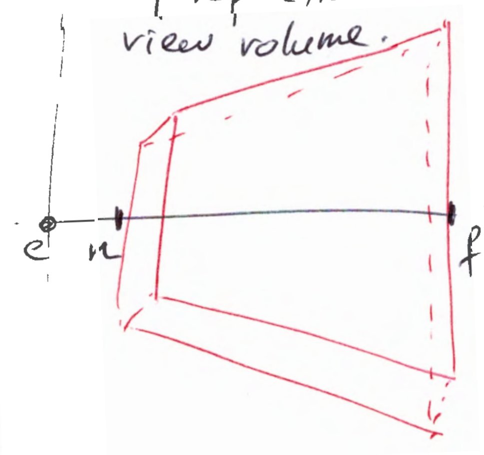
- ⤳ Encode depth foreshortening (persp) or lack thereof (orth
  i.e the relations $y_p = d\,(y/z)$ or $y_p = y$
- ⤳ Keep the z-coordinate around for easy clipping



Orthographic View volume

$z \in [n, f]$ (near-far)
$x \in [l, r]$ (left-right)
$y \in [b, t]$ (bottom-top)

Perspective view volume.



Algebra ...

A. Orthographic

least negative

most negative

Objective : Remap cube $[l,r] \times [b,t] \times [f,n]$

to $[-1,1]^3$

In homogeneous coords :

$$P_{orth} = \begin{bmatrix} \frac{2}{r-l} & & & \frac{r+l}{l-r} \\ & \frac{2}{t-b} & & \frac{t+b}{b-t} \\ & & \frac{2}{n-f} & \frac{n+f}{f-n} \\ & & & 1 \end{bmatrix}$$

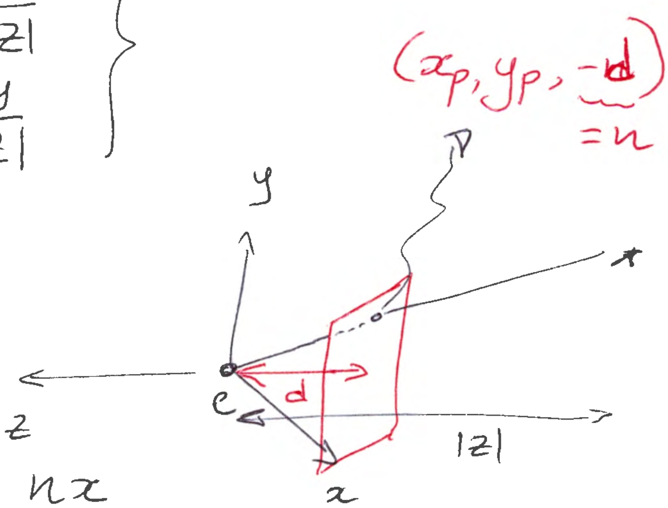
(no need to memorize just understand action)

B. Perspective

Objective :

$\rightarrow$ Remap $\left\{ \begin{array}{l} x_p = d \dfrac{x}{|z|} \\ y_p = d \dfrac{y}{|z|} \end{array} \right\}$

(initially)



Due to orientation

$\left. \begin{array}{l} d = -n \\ |z| = -z \end{array} \right\}$

$$x_p = \frac{nx}{z}$$

$$y_p = \frac{ny}{z}$$

$$\text{"}z_p\text{"} = n = -d$$

Problem : Transform no longer affine!
(we divide x/z ....)

Solution : equivalence in homogeneous coords

we allow $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ and $\begin{bmatrix} \alpha \cdot x \\ \alpha \cdot y \\ \alpha \cdot z \\ \alpha \end{bmatrix}$ to

represent the same point (x, y, z) in 3D.

(or if we are given an arbitrary $\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$ we re-factor it as:

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} w(x/w) \\ w(y/w) \\ w(z/w) \\ w \end{pmatrix} \sim \begin{pmatrix} x/w \\ y/w \\ z/w \\ 1 \end{pmatrix} \to \text{3D pt} \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix})$$

Try this now :

$$P_{persp.} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -\frac{1}{d} & 0 \end{bmatrix}$$ ← not "1"!

$$P_{persp} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -\frac{z}{d} = \frac{z}{n} \end{bmatrix} = \begin{bmatrix} \frac{z}{n} \cdot \frac{nx}{z} \\ \frac{z}{n} \cdot \frac{ny}{z} \\ \frac{z}{n} \cdot n \\ \frac{z}{n} \end{bmatrix} \sim$$

$$\sim \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ n \\ 1 \end{bmatrix}$$

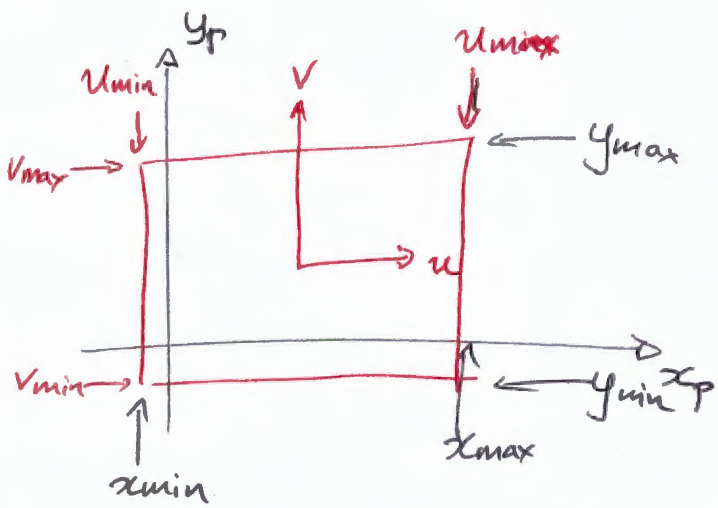
Be mindful of slight variations...

(Shirley)

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

essentially same... just keeps track of a <u>rescaled</u> z-coord to facilitate clipping between $z=n$ & $z=f$ planes:

<u>Viewport transform</u>



Easiest...

adjusts range & origin

$$\begin{bmatrix} u \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{U_{max}-U_{min}}{x_{max}-x_{min}} & & A \\ & \frac{V_{max}-V_{min}}{y_{max}-y_{min}} & B \\ & & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

$$A = \frac{U_{min} X_{max} - U_{max} X_{min}}{x_{max} - x_{min}}$$

$$B = \frac{V_{min} Y_{max} - V_{max} Y_{min}}{Y_{max} - Y_{min}} \quad \ldots$$