NYU

Virtual and Augmented Reality

CS-GY 9223/CUSP-GX 6004

https://nyu-icl.github.io/courses/2022fall-vr-ar

Prof. Qi Sun
qisun@nyu.edu | www.qisun.me

Logistics

- More Google Cardboard on the way
- Assignment 1 releasing (closely related to today's lecture)
- Office hours and other materials on Brightspace (Meta Info)

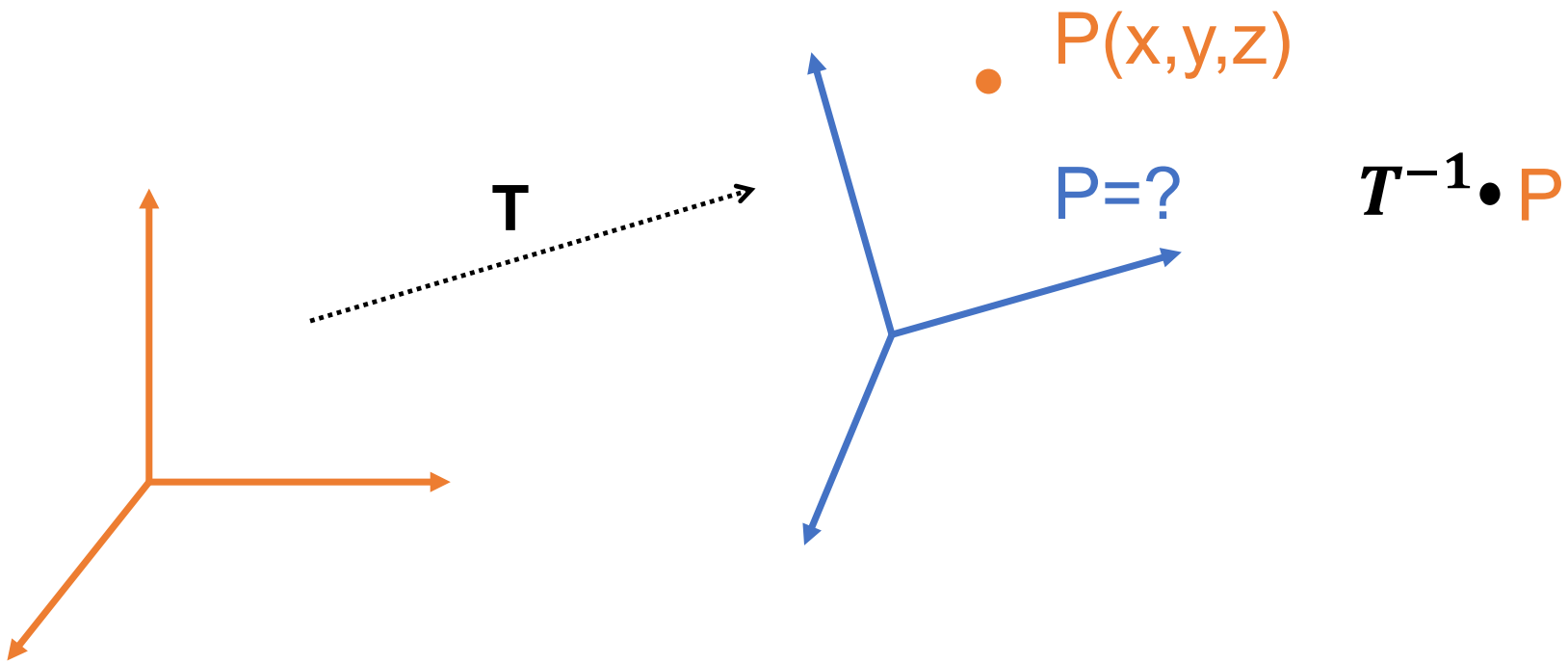
NYU

2

slide courtesy of Ren N

Problem Today: Show an Object on Screen

Coordinate Transformation (assignment!)



3D Transformations

Use homogeneous coordinates again:

- 3D point = (x, y, z, 1)^T
- 3D vector = (x, y, z, 0)^T

Use 4×4 matrices for affine transformations

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

slide courtesy of Ren Ng

3D Transformations

Scale

$$\mathbf{S}(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation

$$\mathbf{T}(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Coordinate Change (Frame-to-world)

$$\mathbf{F}(\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{o}) = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{o} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

slide courtesy of Ren Ng

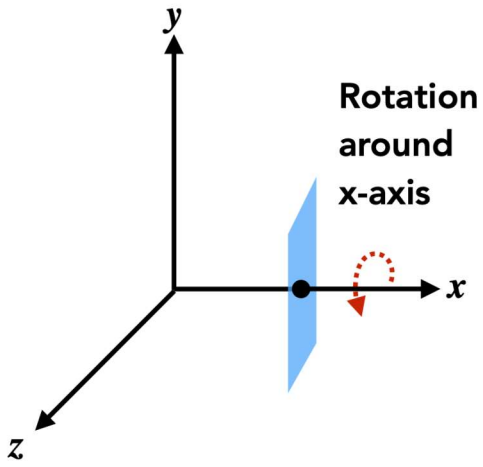
3D Transformations

Rotation around x-, y-, or z-axis

$$\mathbf{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



slide courtesy of Ren Ng

3D Rotations

Compose any 3D rotation from $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$?

$$\mathbf{R}_{xyz}(\alpha, \beta, \gamma) = \mathbf{R}_x(\alpha)\,\mathbf{R}_y(\beta)\,\mathbf{R}_z(\gamma)$$

- So-called *Euler angles*
- Often used in flight simulators: roll, pitch, yaw

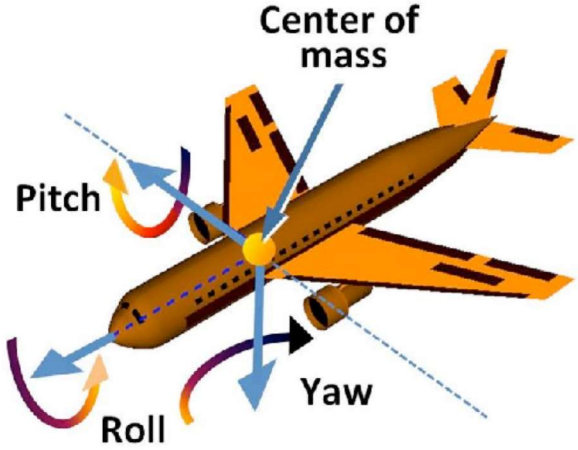
slide courtesy of Ren Ng

3D Rotations

Compose any 3D rotation from $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$?

$$\mathbf{R}_{xyz}(\alpha, \beta, \gamma) = \mathbf{R}_x(\alpha)\,\mathbf{R}_y(\beta)\,\mathbf{R}_z(\gamma)$$

- So-called *Euler angles*
- Often used in flight simulators: roll, pitch, yaw



slide courtesy of Ren Ng

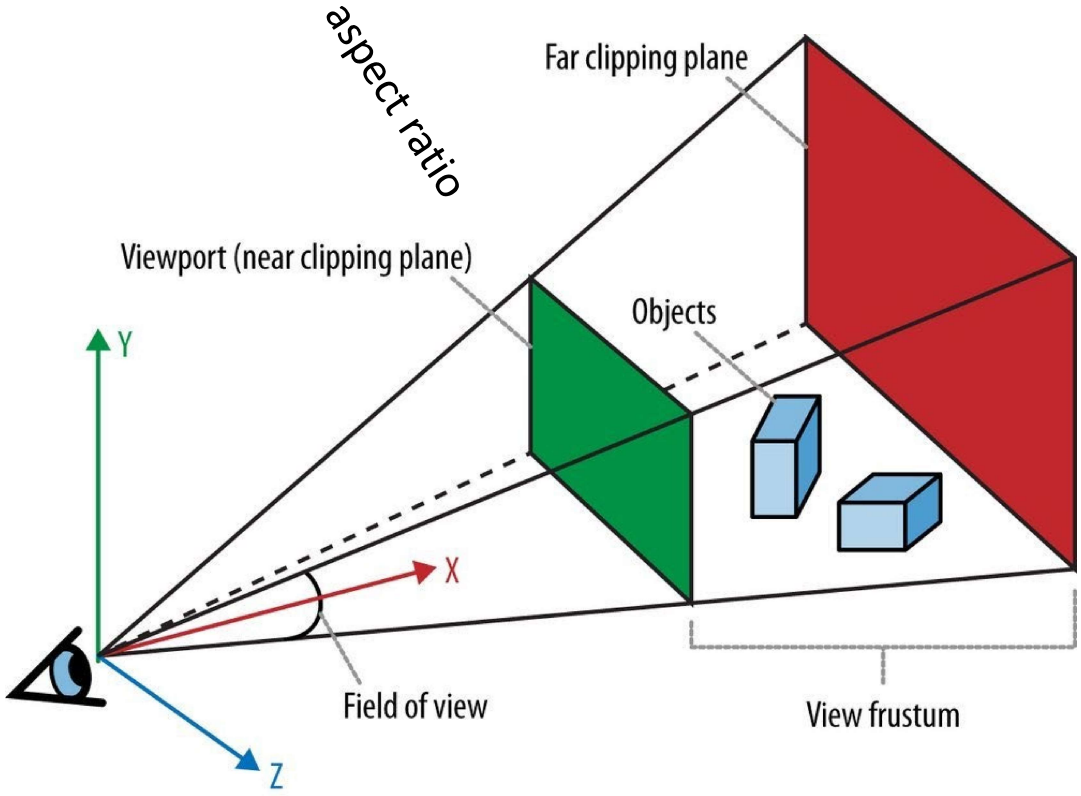
Inverse Transformation

- translation $T(d)=\begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$ inverse translation $T^{-1}(d)=T(-d)=\begin{pmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & -d_y \\ 0 & 0 & 1 & -d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- scale $S(s)=\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ inverse scale $S^{-1}(s)=S\left(\frac{1}{s}\right)=\begin{pmatrix} 1/s_x & 0 & 0 & 0 \\ 0 & 1/s_y & 0 & 0 \\ 0 & 0 & 1/s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- rotation $R_z(\theta)=\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ inverse rotation $R_z^{-1}(\theta)=R_z(-\theta)=\begin{pmatrix} \cos-\theta & -\sin-\theta & 0 & 0 \\ \sin-\theta & \cos-\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Inverse Transformation

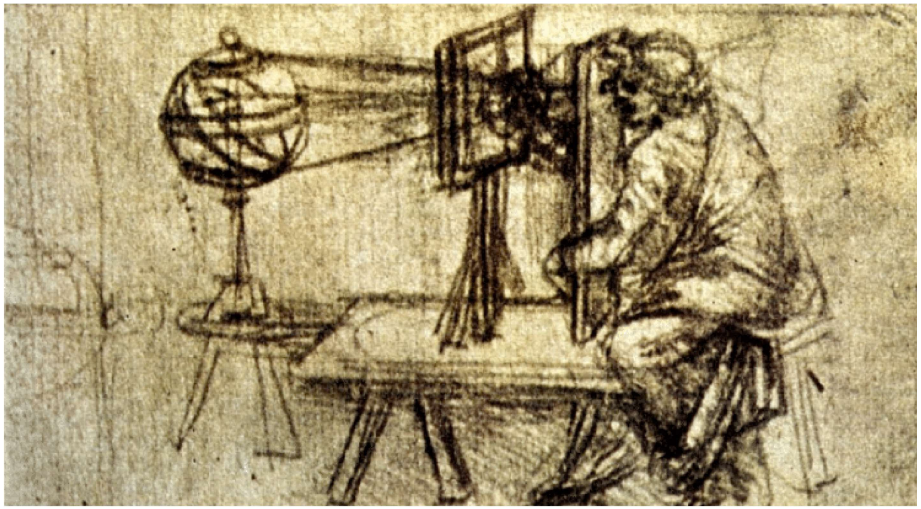
Coordinate Transformation (assignment!)

"Camera" in 3D



3D Transformation

Model-View-Projection



Leonardo da Vinci, 1515

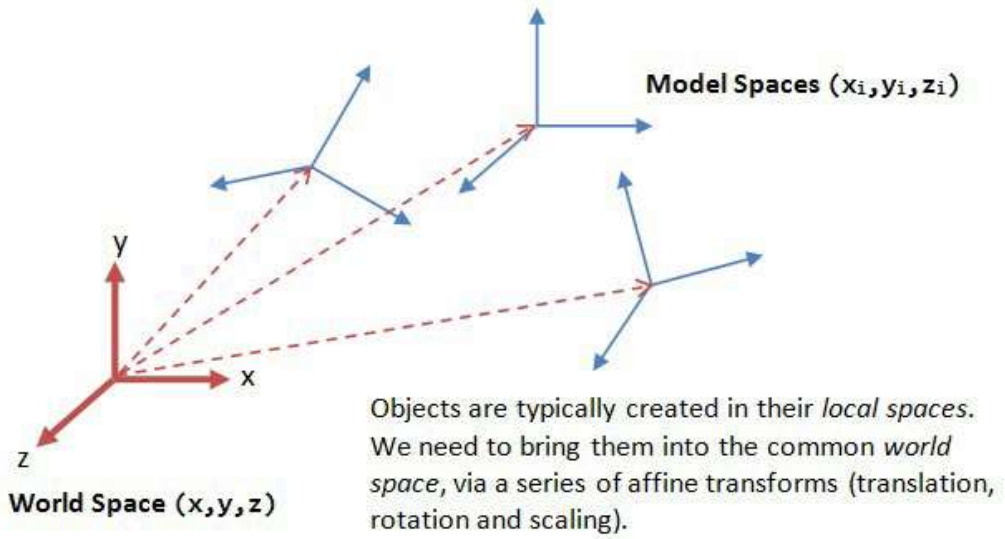
3D Transformation

Model-View-Projection

- While taking a selfie
  1. Find a good place to stand (**model** transformation)
  2. Find a good “angle” to place the camera (**view** transformation)
  3. Cheeese! (**projection** transformation)

Model Transform

- transform each point $v = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ from object coordinates to world coordinates



https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html

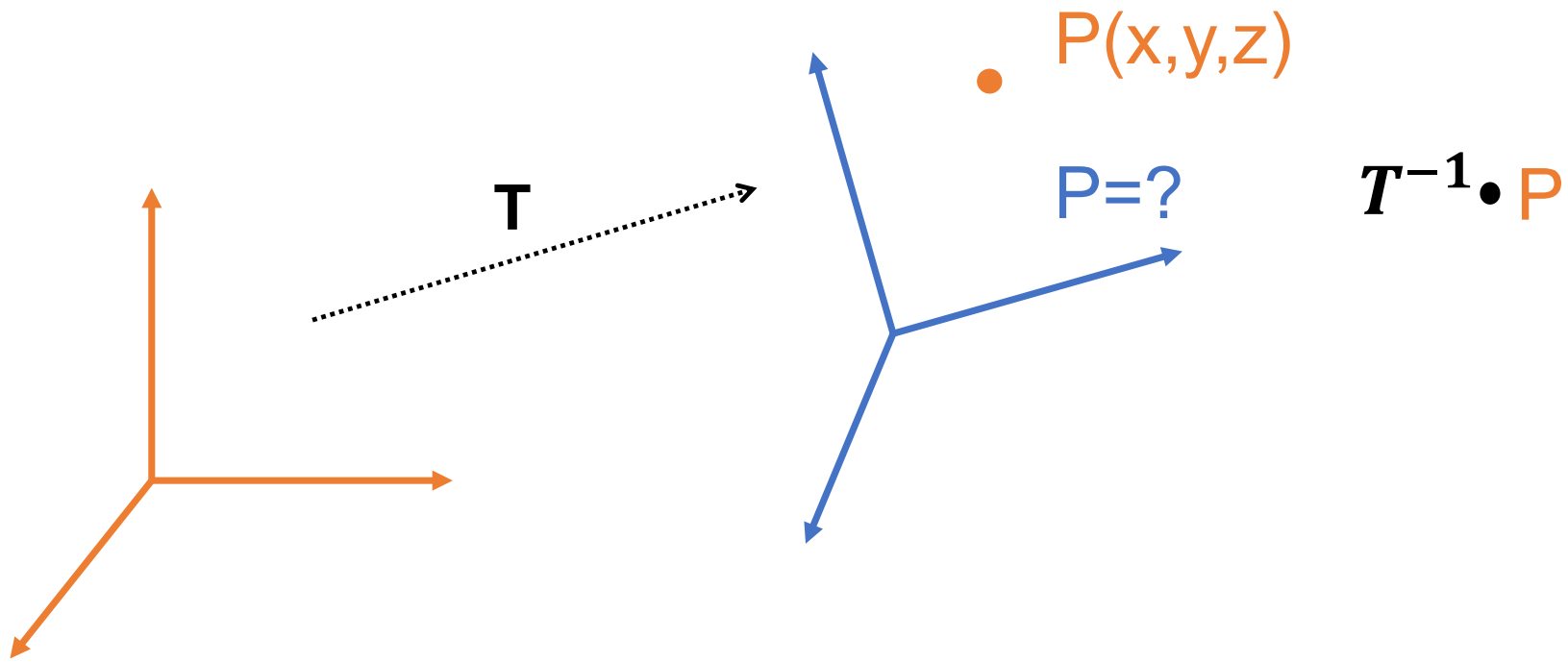
3D Transformation

Model-View-Projection

- While taking a selfie
  1. Find a good place to stand (**model** transformation)
  2. Find a good “angle” to place the camera (**view** transformation)
  3. Cheeese! (**projection** transformation)

Inverse Transformation

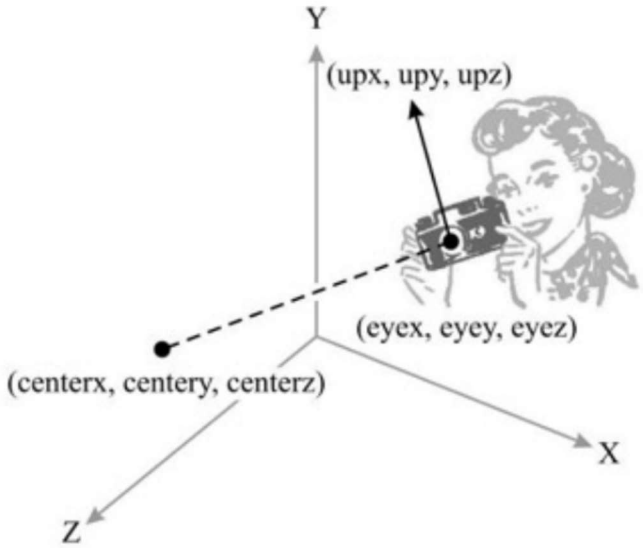
Coordinate Transformation (assignment!)



View Transform

specify camera by

- eye position $eye = \begin{pmatrix} eye_x \\ eye_y \\ eye_z \end{pmatrix}$
- reference position $center = \begin{pmatrix} center_x \\ center_y \\ center_z \end{pmatrix}$
- up vector $up = \begin{pmatrix} up_x \\ up_y \\ up_z \end{pmatrix}$



View Transform

specify camera by

- eye position $eye = \begin{pmatrix} eye_x \\ eye_y \\ eye_z \end{pmatrix}$
- reference position $center = \begin{pmatrix} center_x \\ center_y \\ center_z \end{pmatrix}$
- up vector $up = \begin{pmatrix} up_x \\ up_y \\ up_z \end{pmatrix}$

compute 3 vectors:

$$z^c = \frac{eye - center}{\|eye - center\|}$$

$$x^c = \frac{up \times z^c}{\|up \times z^c\|}$$

$$y^c = z^c \times x^c$$

View Transform

view transform M is translation into eye position,

followed by rotation

$$M = R \cdot T(-e) = \begin{pmatrix} x_x^c & x_y^c & x_z^c & 0 \\ y_x^c & y_y^c & y_z^c & 0 \\ z_x^c & z_y^c & z_z^c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

compute 3 vectors:

$$z^c = \frac{eye - center}{\|eye - center\|}$$

$$x^c = \frac{up \times z^c}{\|up \times z^c\|}$$

$$y^c = z^c \times x^c$$

3D Transformation

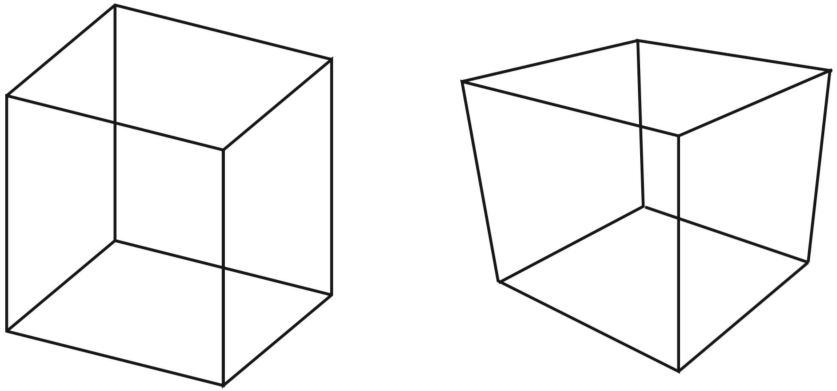
Model-View-Projection

- While taking a selfie
  1. Find a good place to stand (**model** transformation)
  2. Find a good “angle” to place the camera (**view** transformation)
  3. Cheeese! (**projection** transformation)

Projection Transformation

- Projection in Computer Graphics
  - 3D to 2D
  - Orthographic projection
  - Perspective projection

Orthographic projection

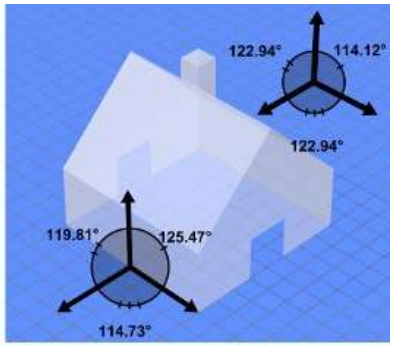


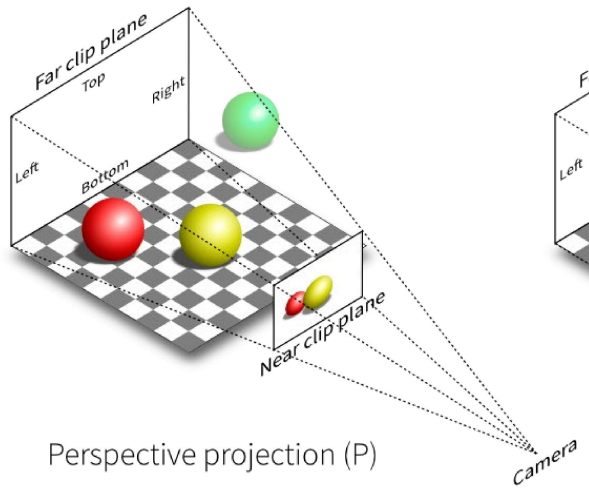
Perspective projection

Fig. 7.1 from *Fundamentals of Computer Graphics, 4th Edition*

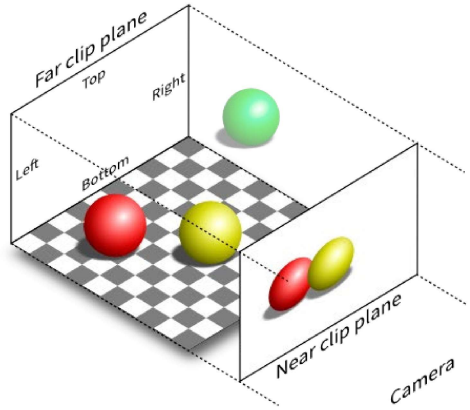
Projection Transformation

- Perspective projection vs. orthographic projection

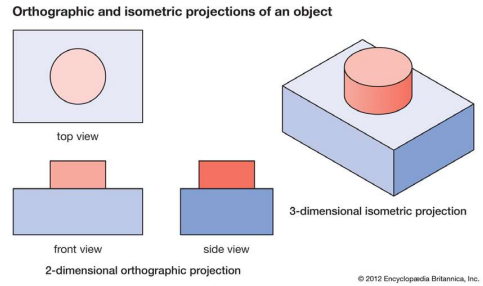




Perspective projection (P)



Orthographic projection (O)



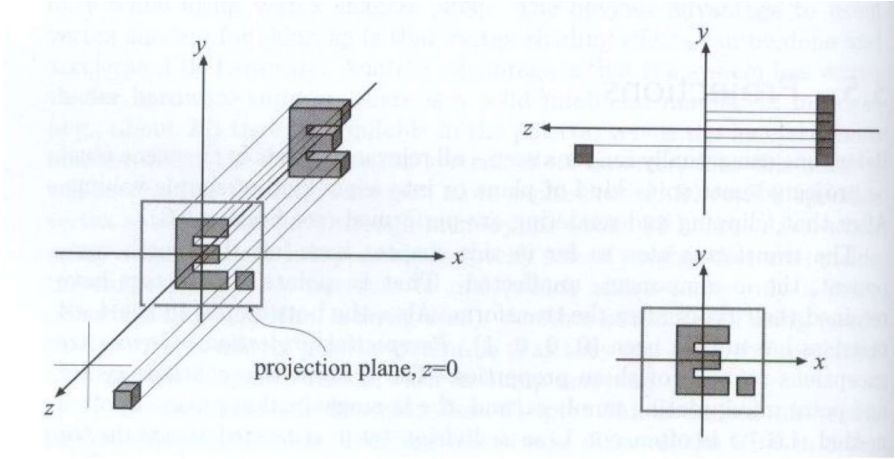
realistic visualization: graphics/arts/urban – human eye

scientific visualization & data design

https://stackoverflow.com/questions/36573283/from-perspective-picture-to-orthographic-picture

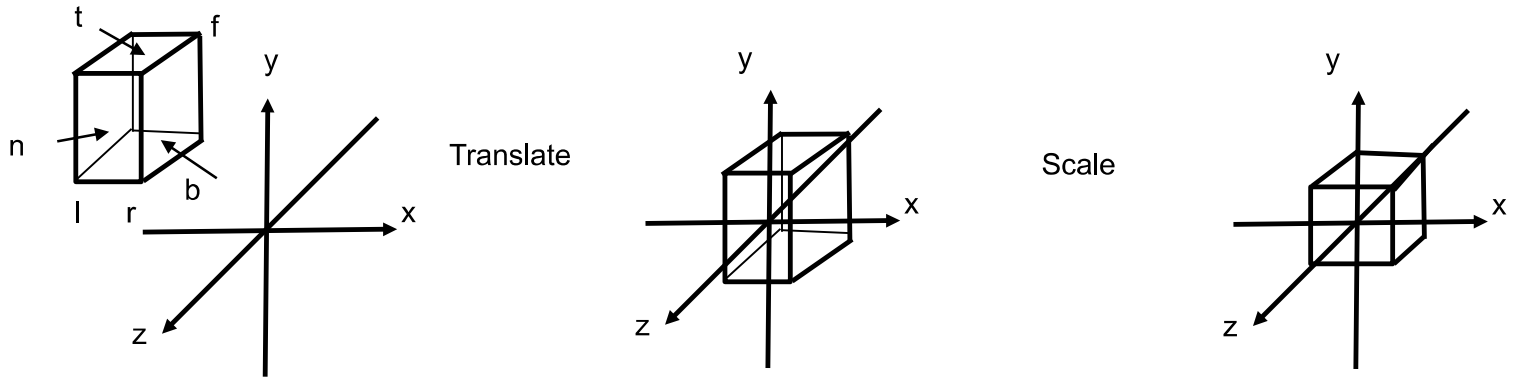
Orthographic Projection

- A simple way of understanding
  - Camera located at origin, looking at -Z, up at Y (looks familiar?)
  - Drop Z coordinate
  - Translate and scale the resulting rectangle to "canonical" view within $[-1, 1]^2$



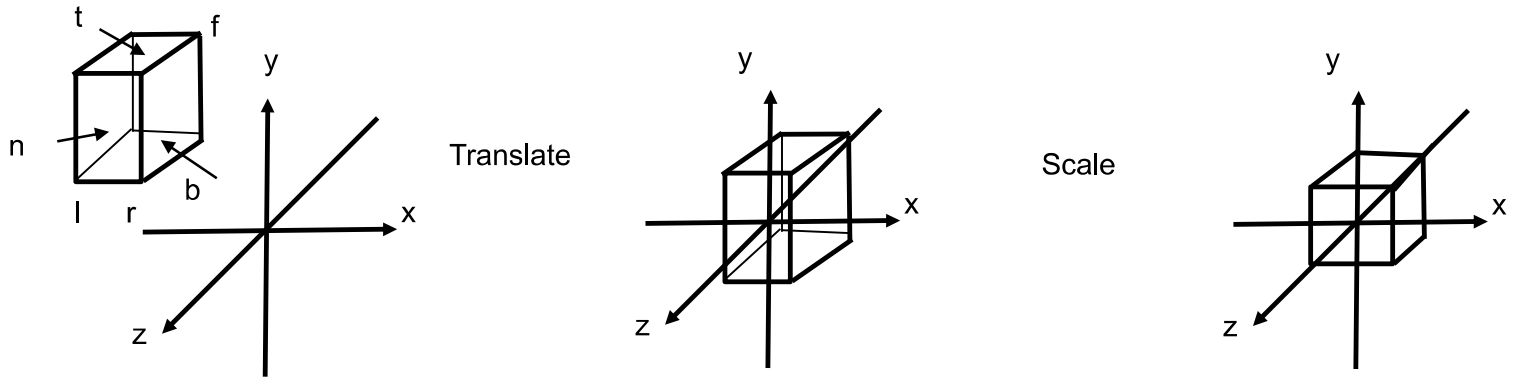
Orthographic Projection

- In general
  - We want to map a cuboid [l, r] x [b, t] x [**f, n**] to the “canonical” cube [-1, 1]^3



Orthographic Projection

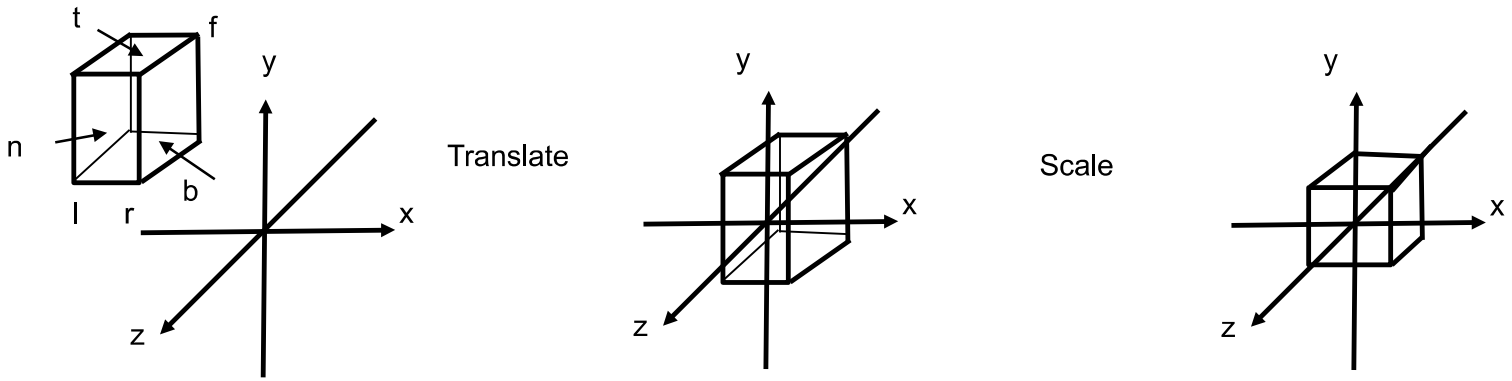
- First center cuboid by translating
- Then scale into “canonical” cube



Orthographic Projection

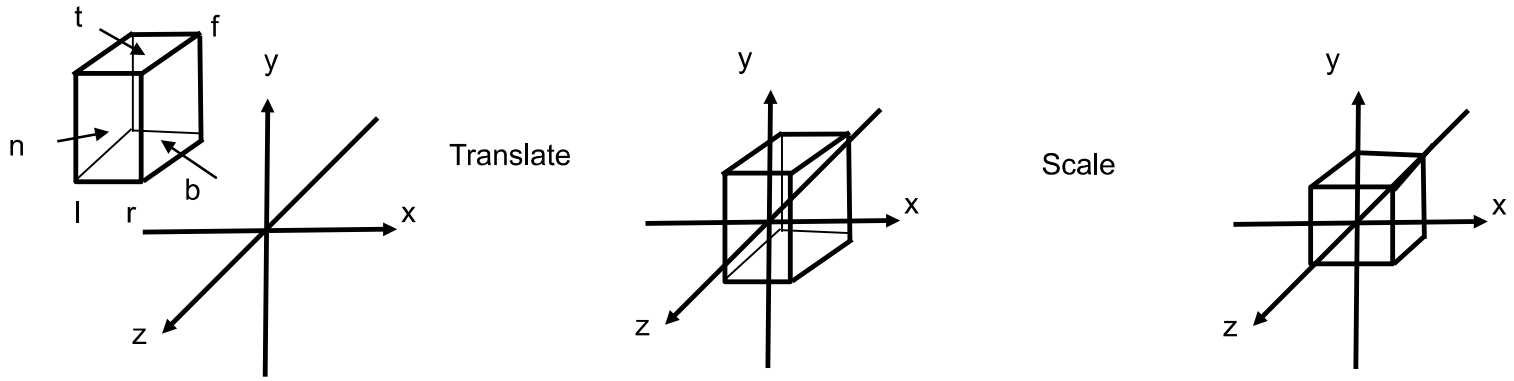
- Transformation matrix?
  - Translate (**center** to origin) first, then scale (length/width/height to **2**)

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



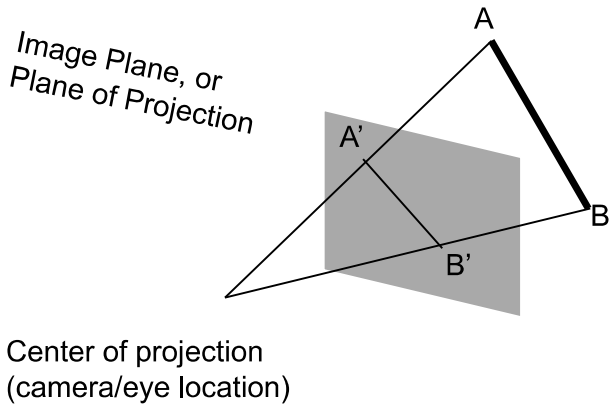
Orthographic Projection

- Caveat
  - Looking at / along -Z is making near and far not intuitive (n > f)
  - FYI: that's why OpenGL (a Graphics API) uses left hand coords.



Perspective Projection

- Most common computer graphics, VR, visual system
- Further objects are smaller
- Parallel lines not parallel; converge to single point



Perspective Projection

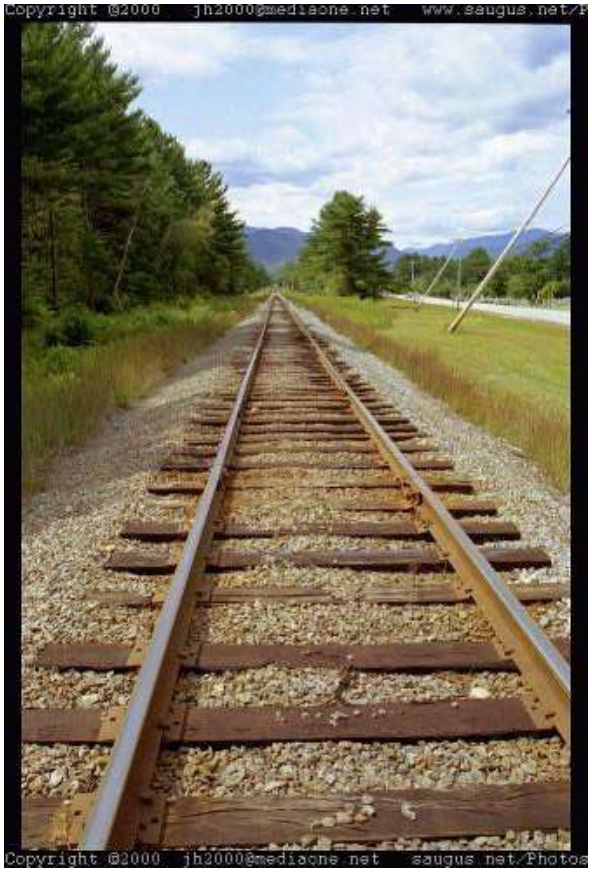
- Euclid was wrong??!!

In geometry, **parallel** lines are lines in a plane which do not meet; that is, two lines in a plane that do not intersect or touch each other at any point are said to be parallel. By extension, a line and a plane, or two planes, in three-dimensional Euclidean space that do not share a point are said to be parallel. However, two lines in three-dimensional space which do not meet must be in a common plane to be considered parallel; otherwise they are called skew lines. Parallel planes are planes in the same three-dimensional space that never meet.

Line art drawing of parallel lines and curves.

Parallel lines are the subject of Euclid's parallel postulate.[1] Parallelism is primarily a property of affine geometries and Euclidean geometry is a special instance of this type of geometry. In some other geometries, such as hyperbolic geometry, lines can have analogous properties that are referred to as parallelism.

https://en.wikipedia.org/wiki/Parallel_(geometry)



Perspective Projection

- A property of homogeneous coordinates
  - (x, y, z, 1) and (kx, ky, kz, k) represent the same point (x, y, z) in 3D
  - e.g. (1, 0, 0, 1) and (2, 0, 0, 2) both represent (1, 0, 0)

Perspective Projection

- How?
  - First “squish” the frustum into a cuboid (n -> n, f -> f) ($M_{persp->ortho}$)
  - Do orthographic projection (M_{ortho})

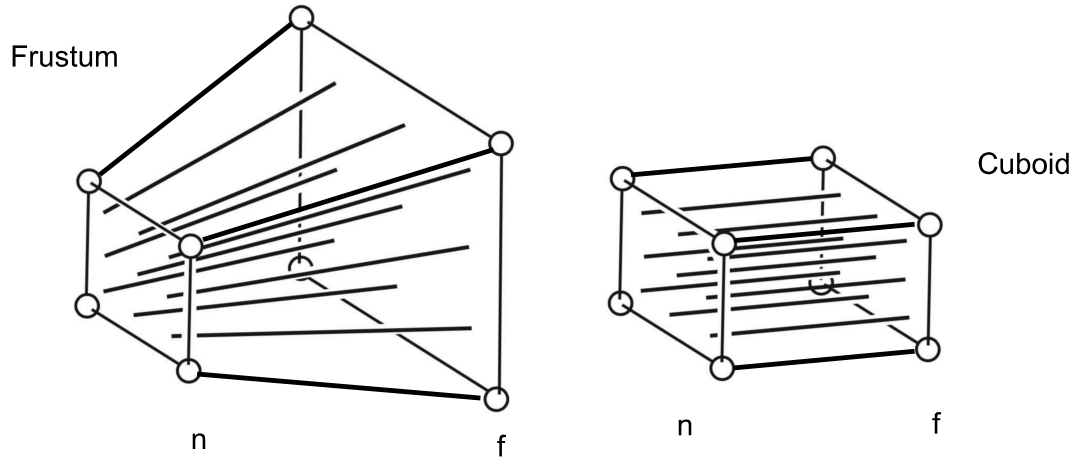
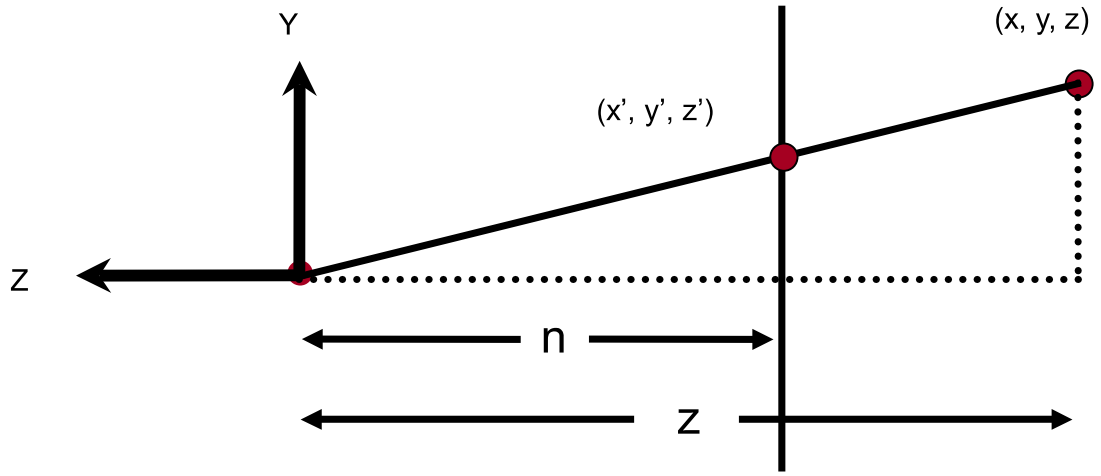


Fig. 7.13 from *Fundamentals of Computer Graphics, 4th Edition*

Perspective Projection

- In order to find a transformation
  - Find the relationship between transformed points (x', y', z') and the original points (x, y, z)



similar triangle

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

Perspective Projection

- In order to find a transformation
  - Find the relationship between transformed points (x', y', z') and the original points (x, y, z)

$$y' = \frac{n}{z}y \qquad x' = \frac{n}{z}x$$ (similar to y')

- In homogeneous coordinates,

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} nx/z \\ ny/z \\ \text{unknown} \\ 1 \end{pmatrix} \overset{\text{mult. by z}}{==} \begin{pmatrix} nx \\ ny \\ \text{still unknown} \\ z \end{pmatrix}$$

Slides courtesy: Prof. Ling-Qi Yan

Perspective Projection

- So the “squish” (persp to ortho) projection does this

$$M_{persp\to ortho}^{(4\times 4)} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} nx \\ ny \\ \text{unknown} \\ z \end{pmatrix}$$

- Already good enough to figure out part of $M_{\text{persp->ortho}}$

$$M_{persp\to ortho} = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Slides courtesy: Prof. Ling-Qi Yan

Perspective Projection

- How to figure out the third row of $M_{persp->ortho}$
  - Any information that we can use?

$$M_{persp\rightarrow ortho} = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- Observation: the third row is responsible for z'
  - Any point on the near plane will not change
  - Any point's z on the far plane will not change

Slides courtesy: Prof. Ling-Qi Yan

Perspective Projection

- Any point on the near plane will not change

$$M_{persp\to ortho}^{(4\times 4)}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}nx\\ny\\\text{unknown}\\z\end{pmatrix}$$

replace z with n

$$\begin{pmatrix}x\\y\\n\\1\end{pmatrix}\Rightarrow\begin{pmatrix}x\\y\\n\\1\end{pmatrix}==\begin{pmatrix}nx\\ny\\n^2\\n\end{pmatrix}$$

- So the third row must be of the form (0 0 A B)

$$\begin{pmatrix}0 & 0 & A & B\end{pmatrix}\begin{pmatrix}x\\y\\n\\1\end{pmatrix}=n^2$$

Slides courtesy: Prof. Ling-Qi Yan

Perspective Projection

- Besides, we have

$$\begin{pmatrix} 0 & 0 & A & B \end{pmatrix} \begin{pmatrix} x \\ y \\ n \\ 1 \end{pmatrix} = n^2 \quad \Rightarrow \quad An + B = n^2$$

- Any point's z on the far plane will not change

$$\begin{pmatrix} 0 \\ 0 \\ f \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 \\ 0 \\ f \\ 1 \end{pmatrix} == \begin{pmatrix} 0 \\ 0 \\ f^2 \\ f \end{pmatrix} \quad \Rightarrow \quad Af + B = f^2$$

Slides courtesy: Prof. Ling-Qi Yan

Perspective Projection

- Solve for A and B

$$An + B = n^2$$
$$Af + B = f^2$$

$$A = n + f$$
$$B = -nf$$

- Finally, every entry in $M_{persp->ortho}$ is known!

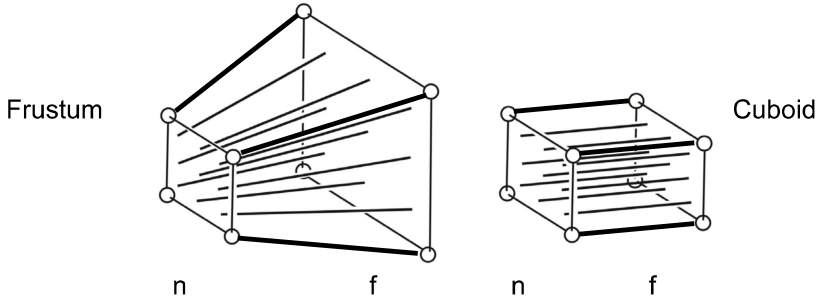
- What's next?
  - Do orthographic projection (M_{ortho}) to finish

$$M_{persp} = M_{ortho} M_{persp \rightarrow ortho}$$

Slides courtesy: Prof. Ling-Qi Yan

Perspective Projection

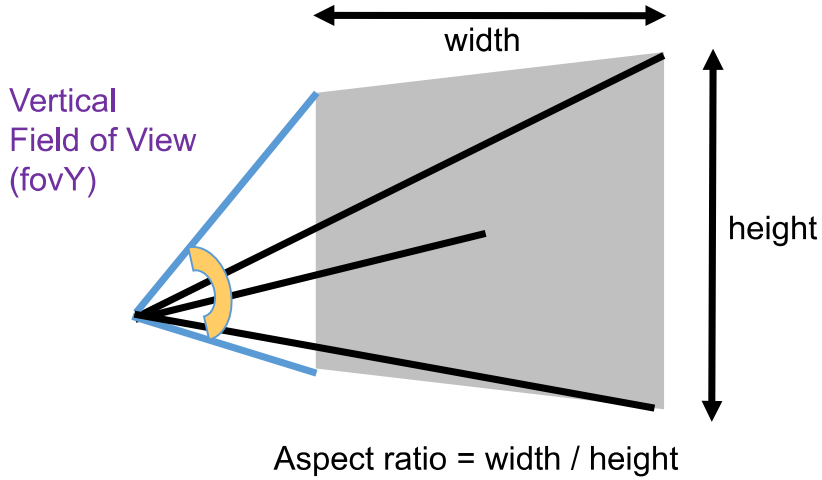
- What orthographic projection to perform?
  - Recall: just need the cuboid
  - [l, r] x [b, t] x [n, f]
  - n and f did not change
  - l, r, b, t are near plane's l, r, b, t



Slides courtesy: Prof. Ling-Qi Yan

Perspective Projection

- What's near plane's l, r, b, t then?
  - If explicitly specified, good
  - Sometimes people prefer defining them with vertical **field-of-view** (fovY) and **aspect ratio**, and assume symmetry i.e. l = -r, b = -t

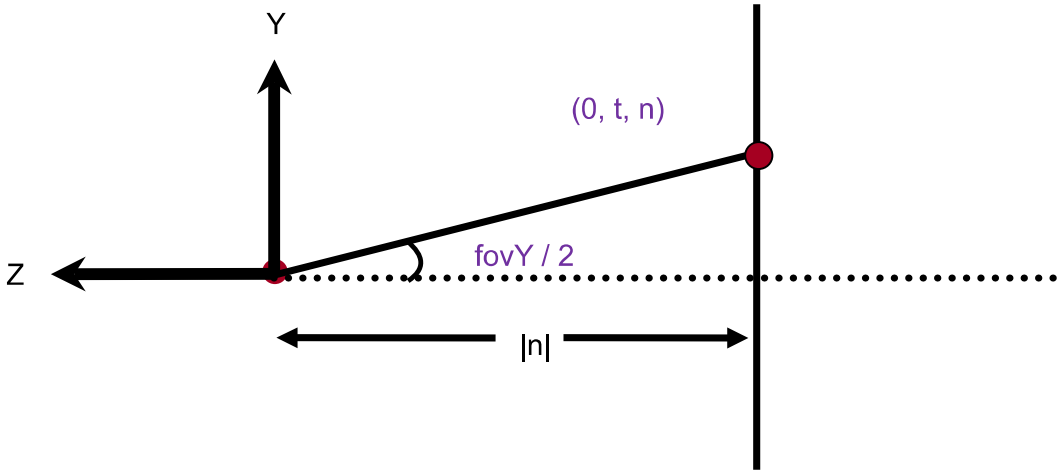


Aspect ratio = width / height

Slides courtesy: Prof. Ling-Qi Yan

Perspective Projection

- How to convert from fovY and aspect to l, r, b, t?
  - Trivial



$$\tan \frac{fovY}{2} = \frac{t}{|n|}$$

$$aspect = \frac{r}{t}$$

Slides courtesy: Prof. Ling-Qi Yan