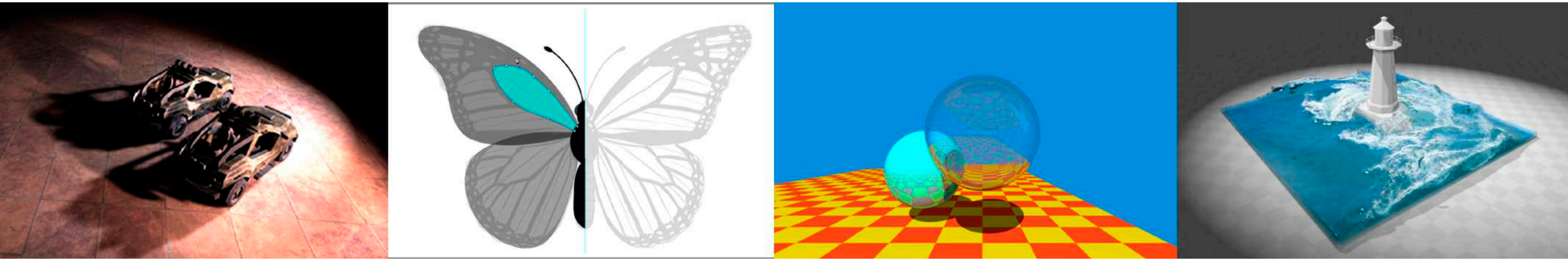
Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 4: Transformation Cont.



http://www.cs.ucsb.edu/~lingqi/teaching/games101.html

Announcement

- Homework 0 will be released TODAY
- This lecture will be difficult :)

Last Lecture

- Transformation
  - Why study transformation
  - 2D transformations: rotation, scale, shear
  - Homogeneous coordinates
  - Composite transform
  - 3D transformations

Today

- 3D transformations
- Viewing (观测) transformation
  - View (视图) / Camera transformation
  - Projection (投影) transformation
    - Orthographic (正交) projection
    - Perspective (透视) projection

3D Transformations

Use homogeneous coordinates again:

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

In general, (x, y, z, w) ($w \neq 0$) is the 3D point:

$$(x/w, y/w, z/w)$$

3D Transformations

Use 4×4 matrices for affine transformations

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

What's the order?

Linear Transform first or Translation first?

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

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

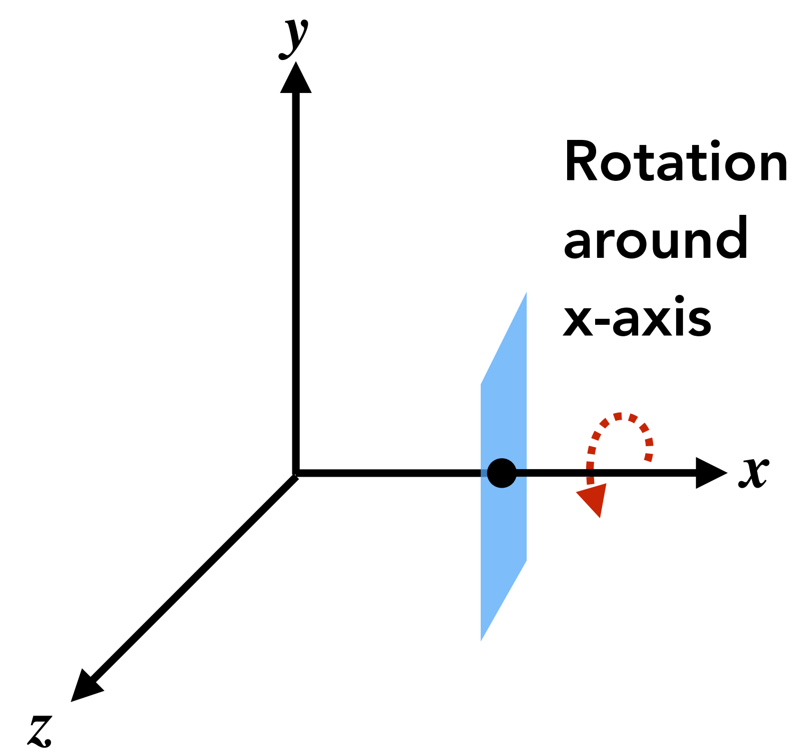
3D Transformations

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

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

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

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



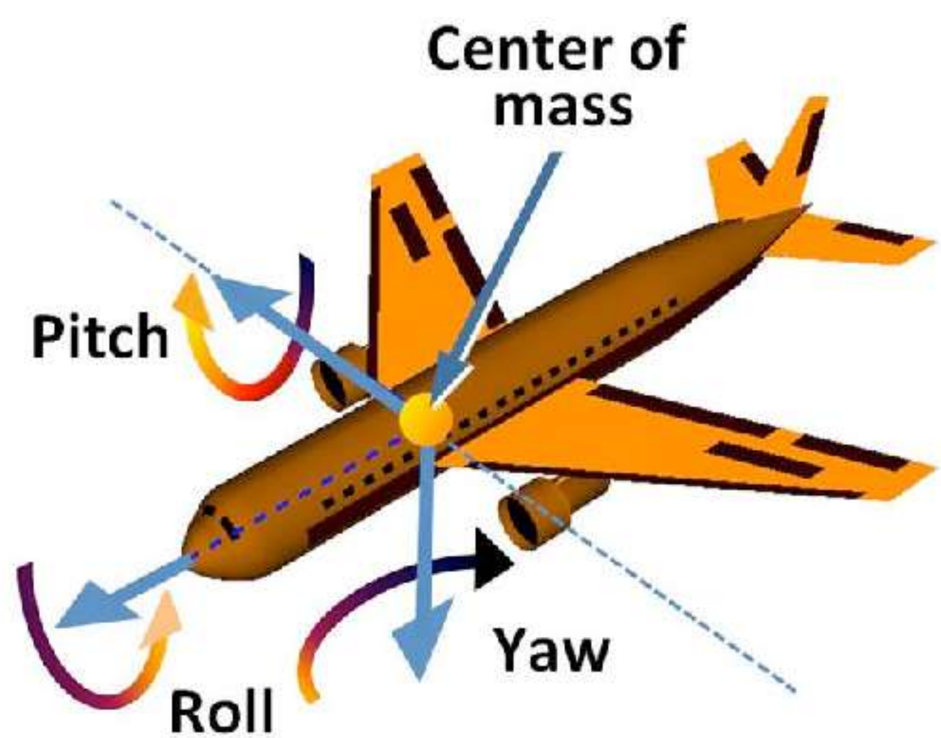
Anything strange about R_y?

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



Rodrigues' Rotation Formula

Rotation by angle α around axis $\mathbf{n}$

$$\mathbf{R}(\mathbf{n}, \alpha) \;=\; \cos(\alpha)\,\mathbf{I} \;+\; (1 - \cos(\alpha))\,\mathbf{n}\mathbf{n}^T \;+\; \sin(\alpha)\,\underbrace{\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}}_{\mathbf{N}}$$

How to prove this magic formula?

Check out the supplementary material on the course website!

Today

- 3D transformations
- Viewing transformation
  - **View / Camera transformation**
  - Projection transformation
    - Orthographic projection
    - Perspective projection

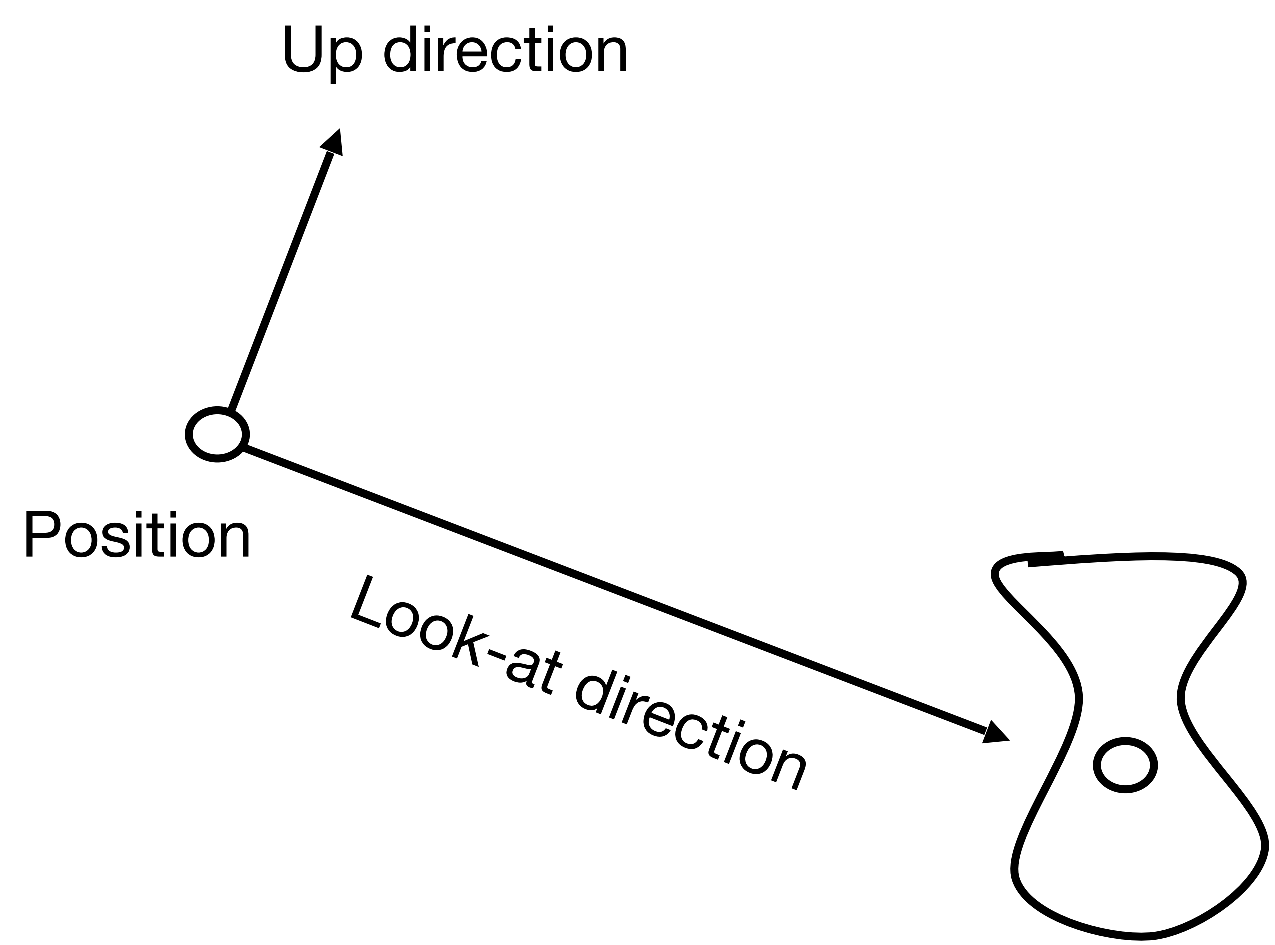
View / Camera Transformation

- What is view transformation?

- Think about how to take a photo
  - Find a good place and arrange people (**model** transformation)
  - Find a good “angle” to put the camera (**view** transformation)
  - Cheese! (**projection** transformation)

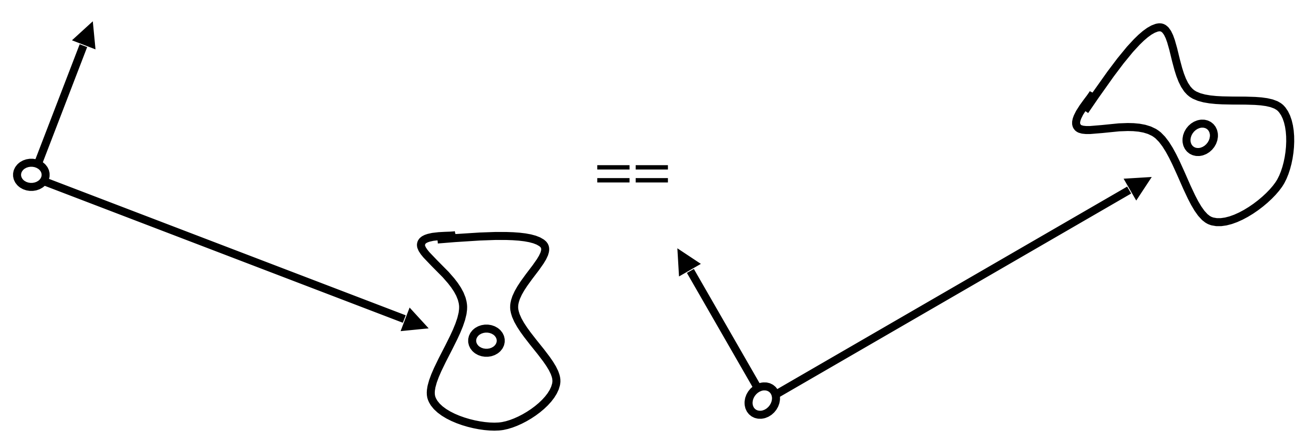
View / Camera Transformation

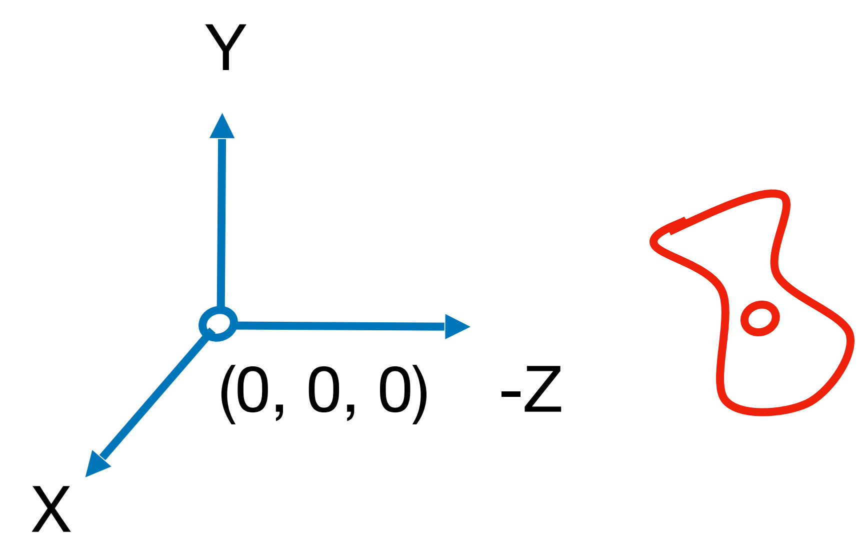
- How to perform view transformation?
- Define the camera first
  - Position $\vec{e}$
  - Look-at / gaze direction $\hat{g}$
  - Up direction $\hat{t}$
    (assuming perp. to look-at)



View / Camera Transformation

- Key observation
  - If the camera and all objects move together, the "photo" will be the same

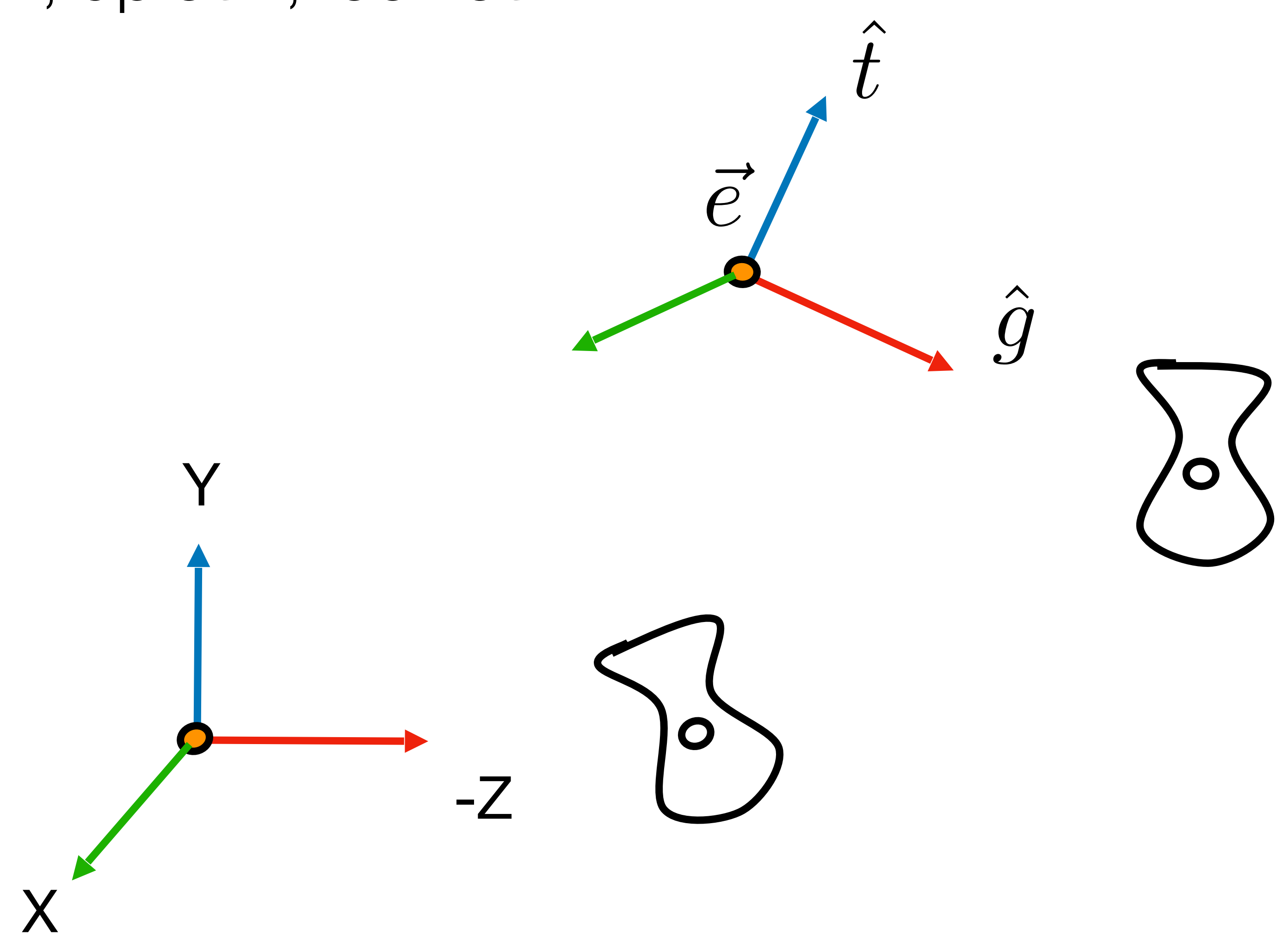




- How about that we always transform the camera to
  - The origin, up at Y, look at -Z
  - And transform the objects along with the camera

View / Camera Transformation

- Transform the camera by M_{view}
  - So it's located at the origin, up at Y, look at -Z
- M_{view} in math?
  - Translates e to origin
  - Rotates g to -Z
  - Rotates t to Y
  - Rotates (g x t) To X
  - Difficult to write!



View / Camera Transformation

- M_{view} in math?
  - Let's write $M_{view} = R_{view}T_{view}$
  - Translate e to origin

$$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Rotate g to -Z, t to Y, (g x t) To X
  - Consider its inverse rotation: X to (g x t), Y to t, Z to -g

$$R_{view}^{-1} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & x_t & x_{-g} & 0 \\ y_{\hat{g}\times\hat{t}} & y_t & y_{-g} & 0 \\ z_{\hat{g}\times\hat{t}} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{WHY?}} R_{view} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t & y_t & z_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

View / Camera Transformation

- Summary
  - Transform objects together with the camera
  - Until camera's at the origin, up at Y, look at -Z
- Also known as ModelView Transformation
- But why do we need this?
  - For projection transformation!

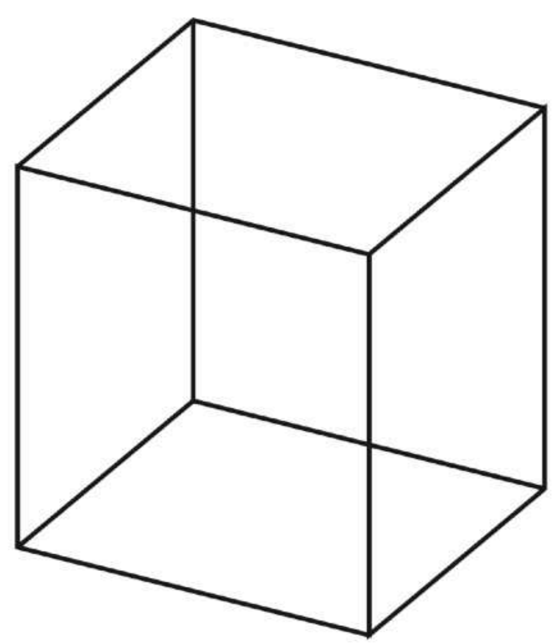
Today

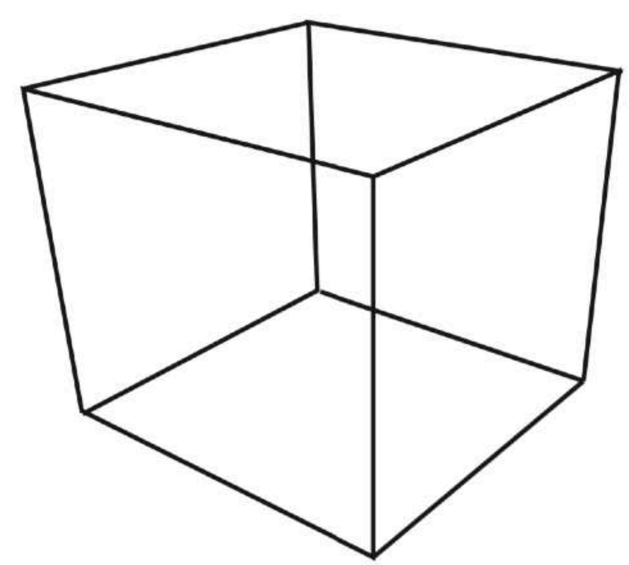
- 3D transformations
- Viewing transformation
  - View / Camera transformation
  - **Projection transformation**
    - Orthographic projection
    - Perspective projection

Projection Transformation

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

Orthographic projection



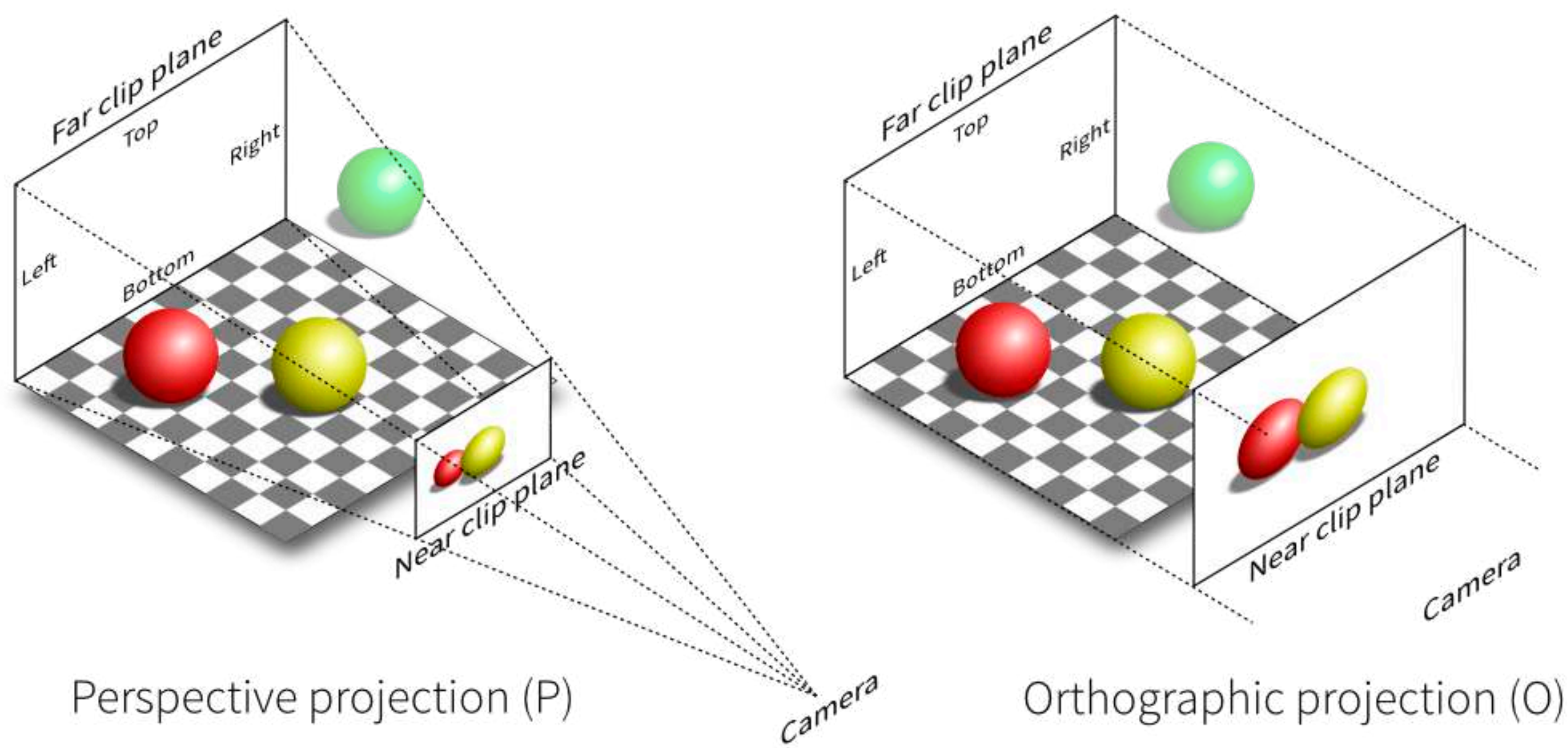


Perspective projection

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

Projection Transformation

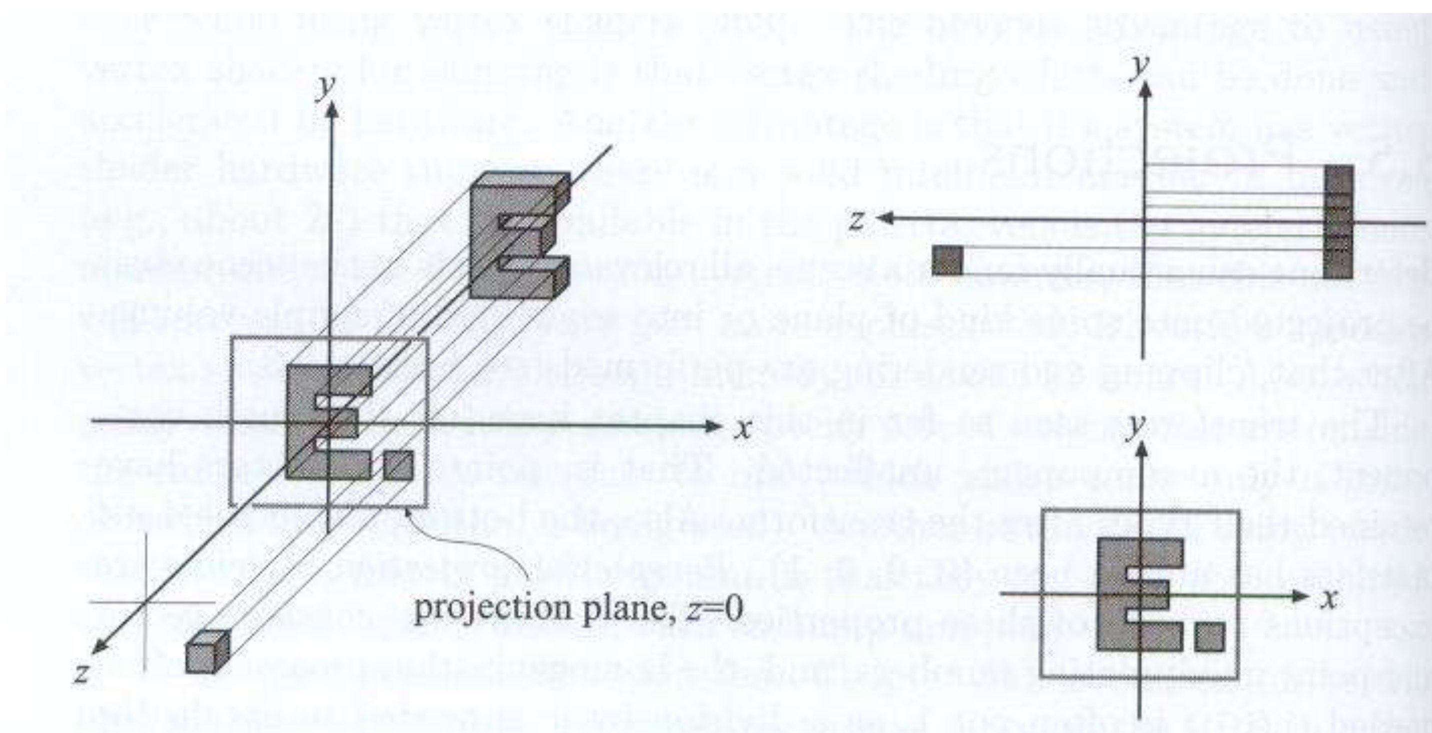
- Perspective projection vs. orthographic projection



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

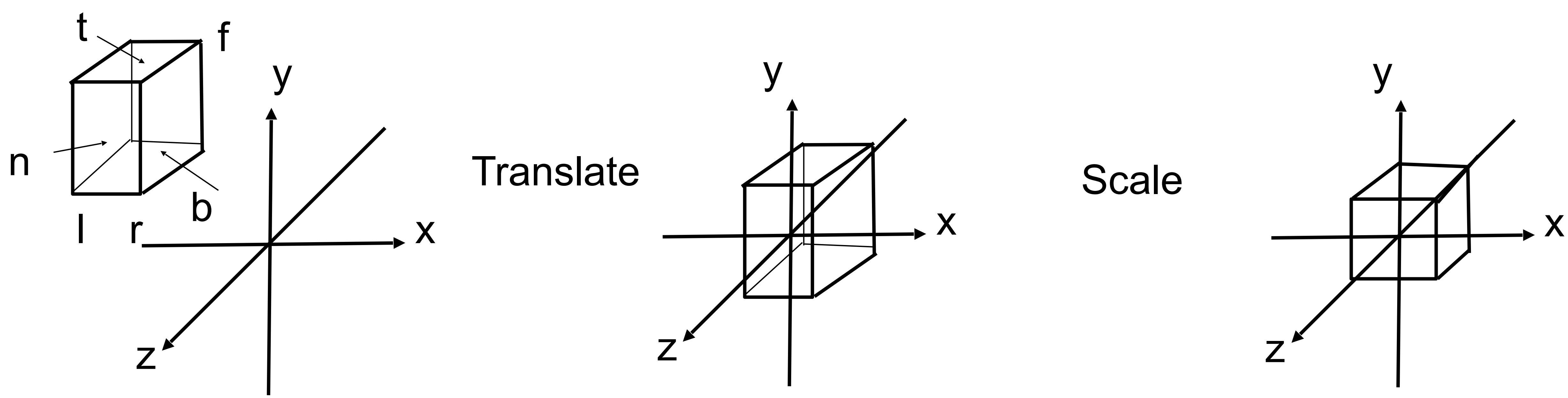
Orthographic Projection

- 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 $[-1, 1]^2$



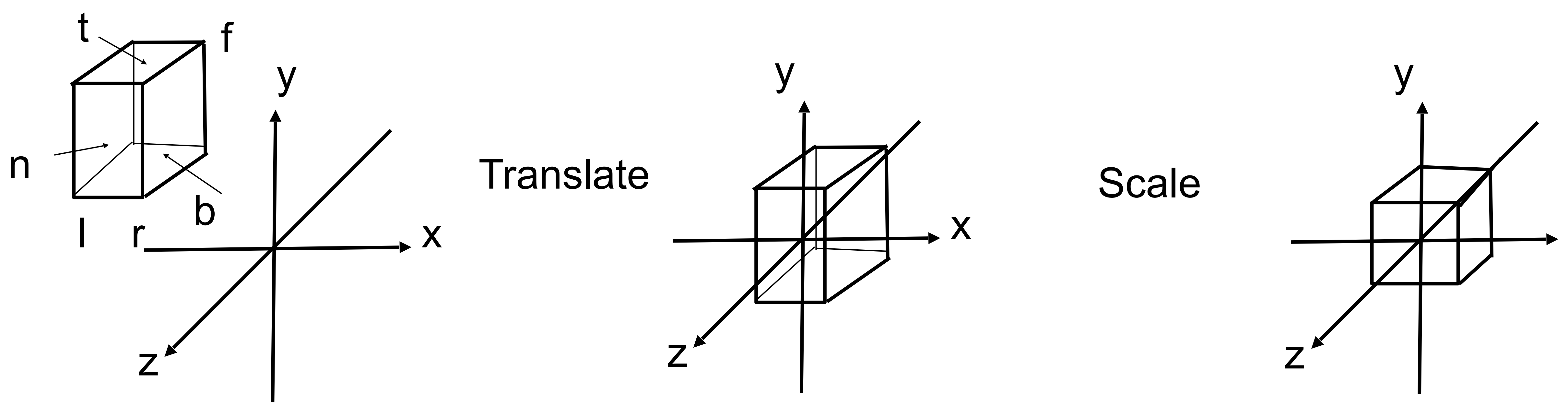
Orthographic Projection

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



Orthographic Projection

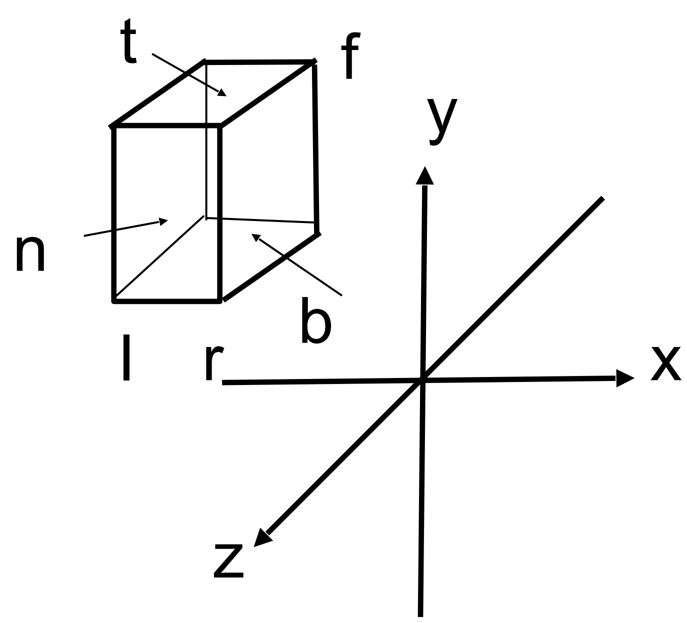
- Slightly different orders (to the “simple way”)
  - Center cuboid by translating
  - Scale into “canonical” cube



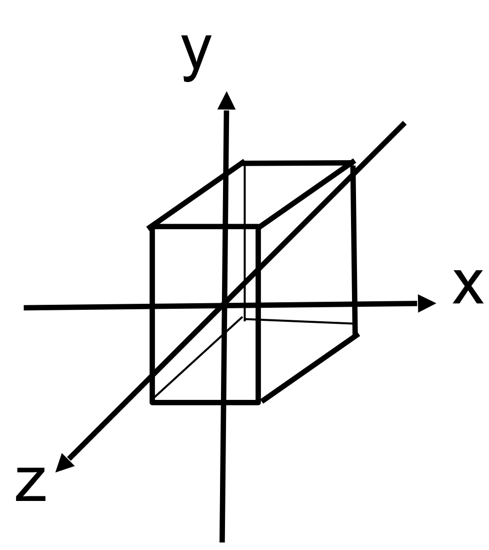
Orthographic Projection

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

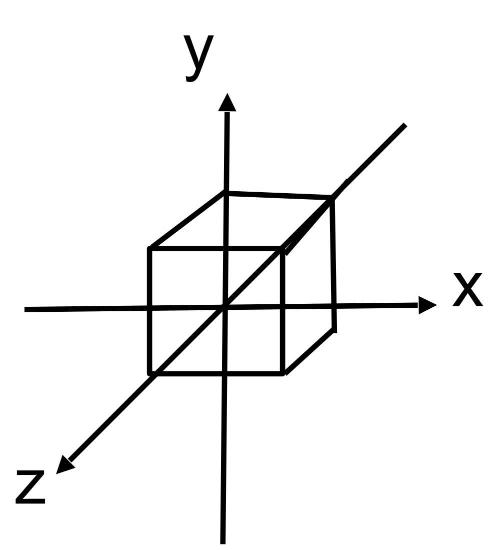
$$M_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Translate

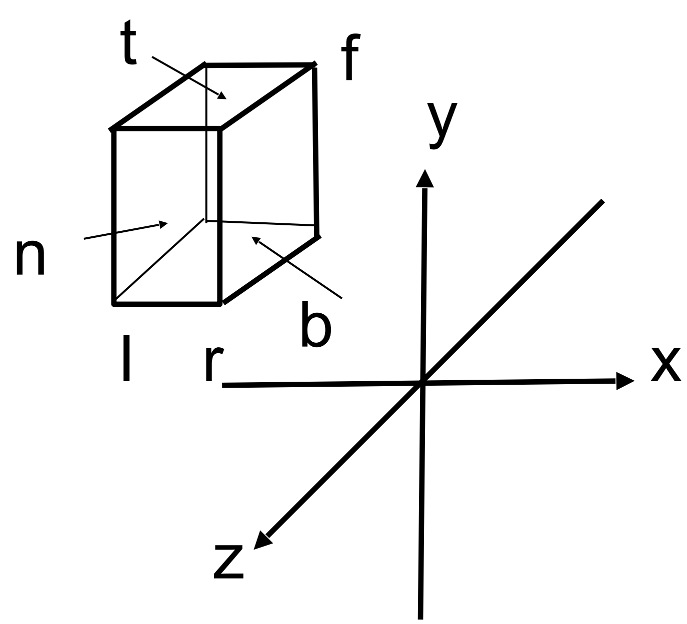


Scale

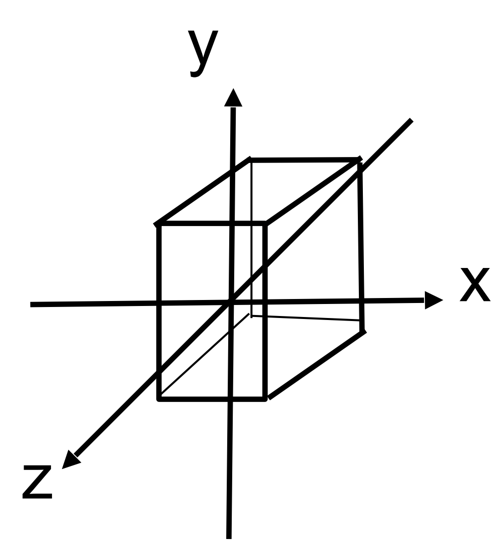


Orthographic Projection

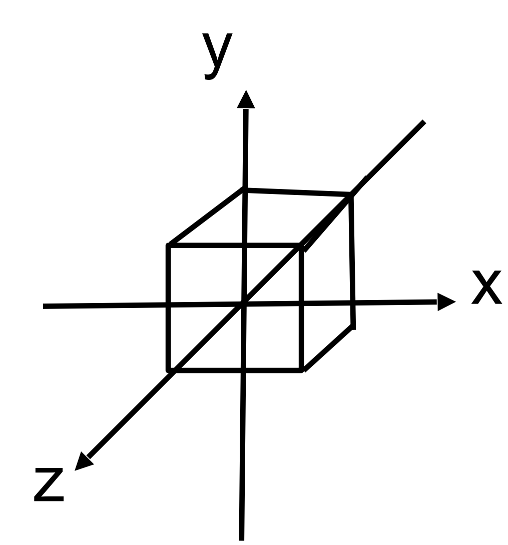
- Caveat
  - Looking at / along -Z is making near and far not intuitive (n > f)
  - FYI: that's why OpenGL (a Graphics API) uses left hand coords.



Translate

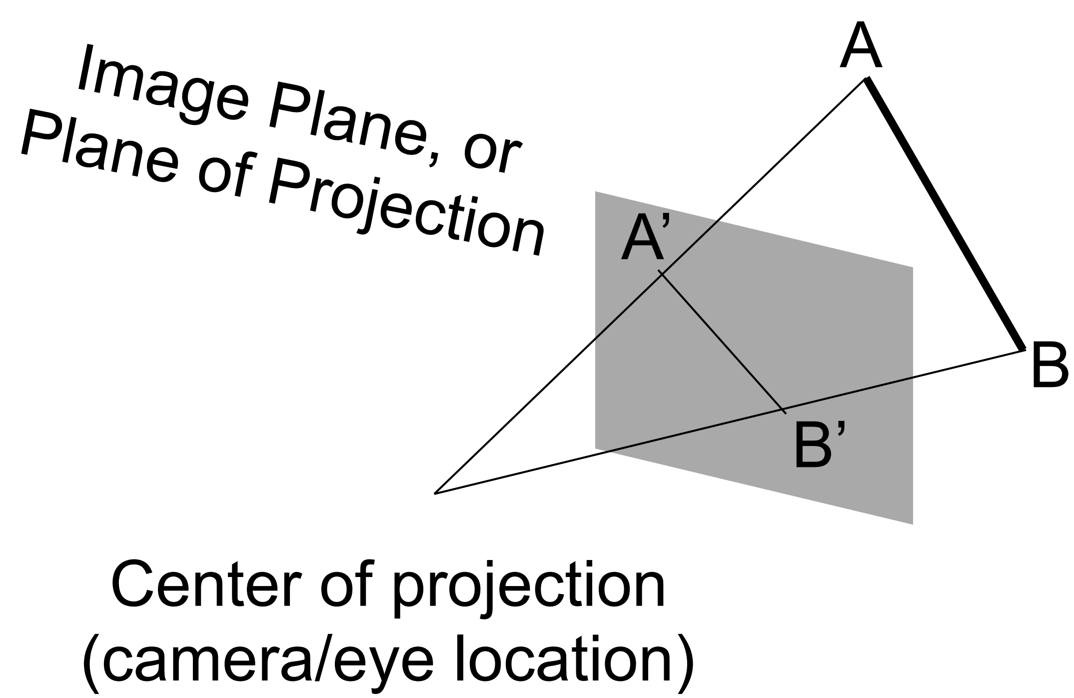


Scale



Perspective Projection

- Most common in Computer Graphics, art, visual system
- Further objects are smaller
- Parallel lines not parallel; converge to single point



Perspective Projection

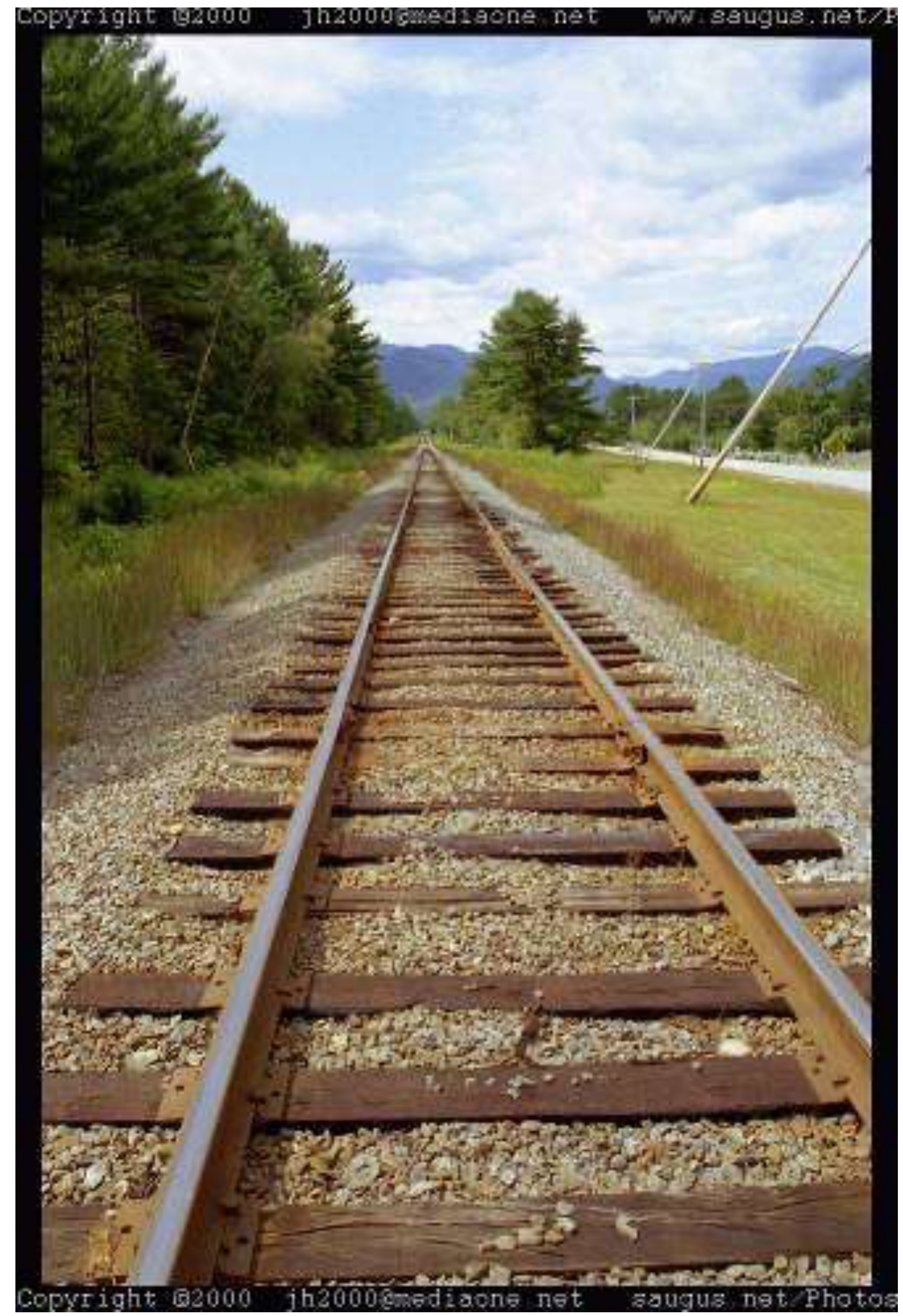
- Euclid was wrong??!!

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

- Before we move on
- Recall: property of homogeneous coordinates
  - (x, y, z, 1), (kx, ky, kz, k != 0), **(xz, yz, z^2, z != 0)** all 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)
- Simple, but useful

Perspective Projection

- How to do perspective projection
  - First "squish" the frustum into a cuboid (n -> n, f -> f) ($M_{persp->ortho}$)
  - Do orthographic projection (M_{ortho}, already known!)

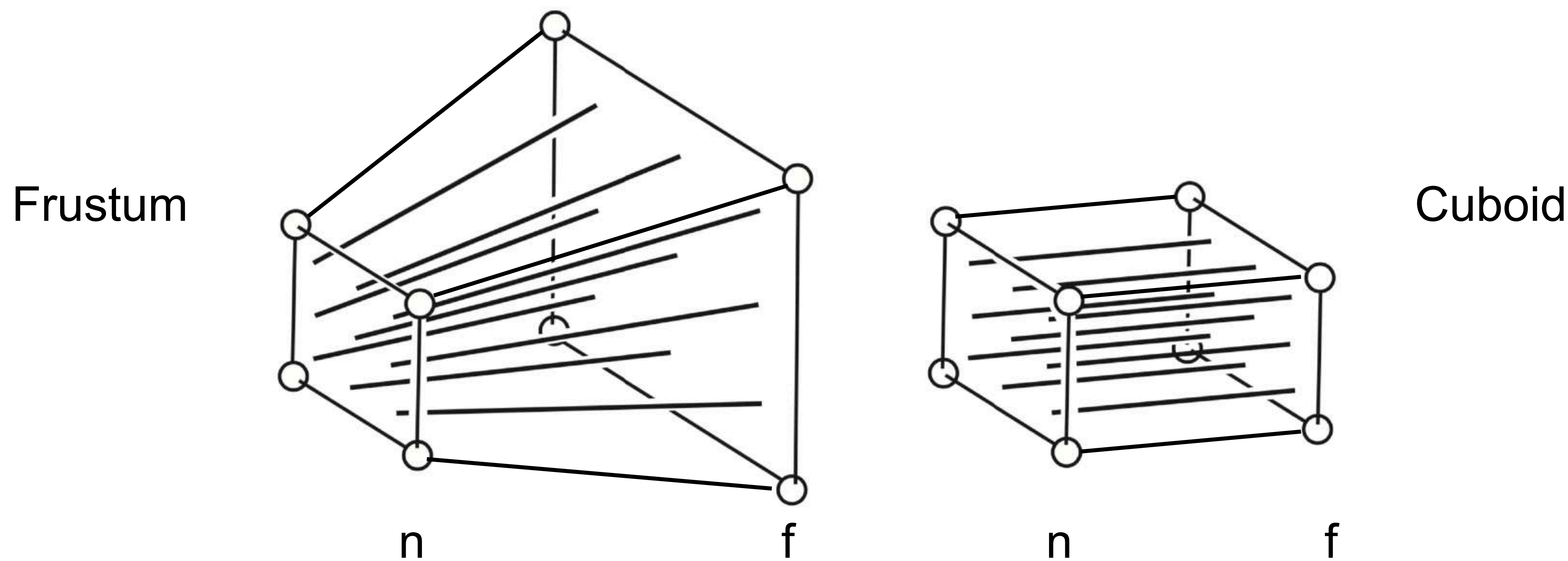
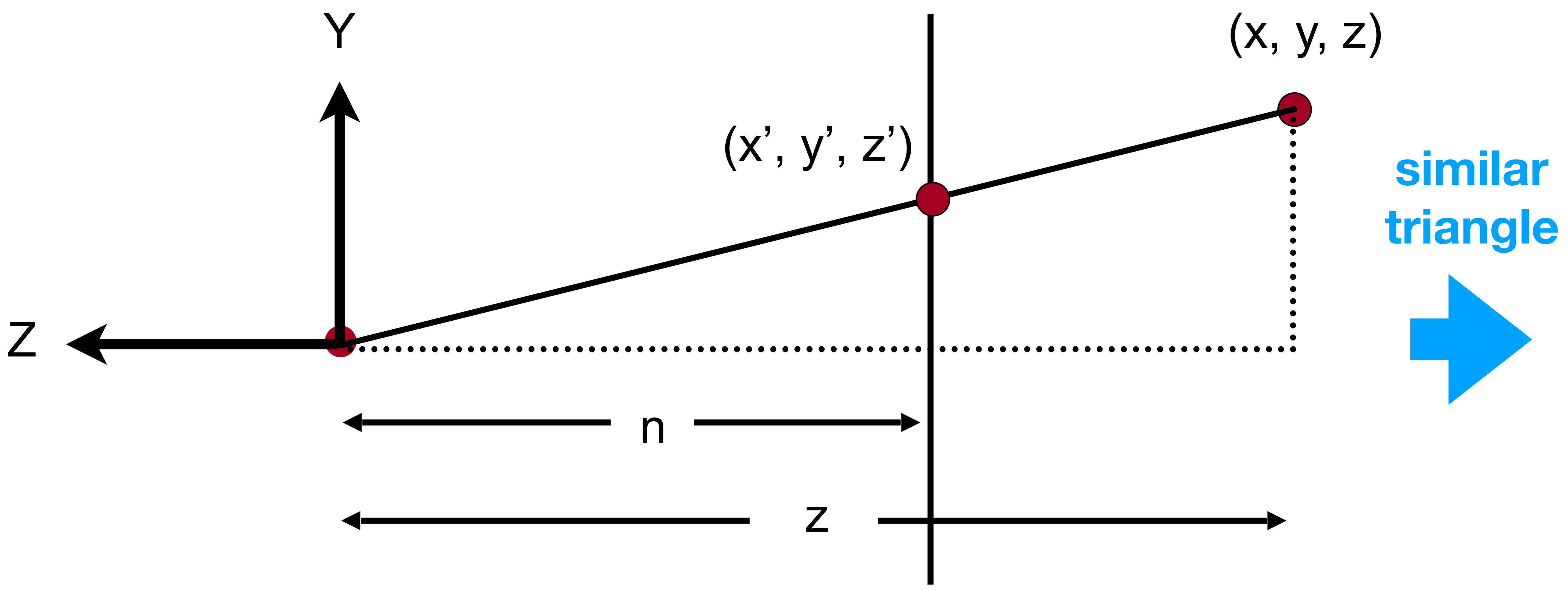


Fig. 7.13 from *Fundamentals of Computer Graphics, 4th Edition*

Perspective Projection

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



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

Perspective Projection

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

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

- In homogeneous coordinates,

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

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 $M_{persp\text{->}ortho}$

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

WHY?

Perspective Projection

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

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

- 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

Perspective Projection

- Any point on the near plane will not change

$$M_{persp\to ortho}^{(4\times4)}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}nx\\ny\\\text{unknown}\\z\end{pmatrix}$$

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)

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

n^2 has nothing to do with x and y

Perspective Projection

- What do we have now?

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

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

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

Perspective Projection

- Solve for A and B

$$An + B = n^2 \qquad A = n + f$$
$$Af + B = f^2 \qquad B = -nf$$

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

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

Thank you!