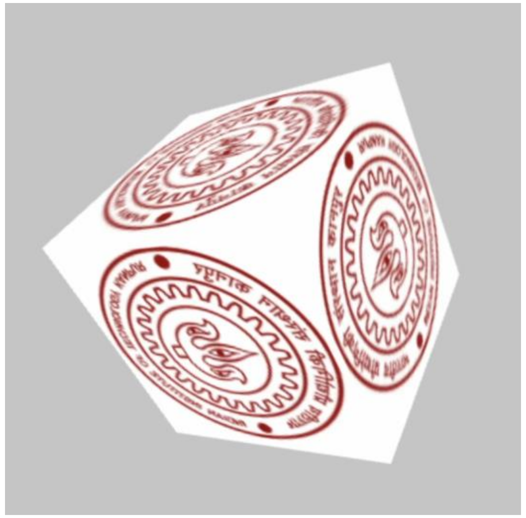


Introduction to Computer Graphics (CS360A)

Instructor: Soumya Dutta

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur (IITK)

email: soumyad@cse.iitk.ac.in

Acknowledgements

- A subset of the slides that I will present throughout the course are adapted/inspired by excellent courses on Computer Graphics offered by Prof. Han-Wei Shen, Prof. Wojciech Matusik, Prof. Frédo Durand, Prof. Abe Davis, and Prof. Cem Yuksel

Assignment 1 Doubts/Confusions?

IITK CS360A

Transformations in 3D

- Let’s say, we want to build a (3D) scene like the following

Affine Transformations

IITK CS360A

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

2D

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D

- 2D → 3D
- 3X3 → 4X4 Matrix (with Homogeneous Coordinate)

Translation in 3D

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2D 3D

- Translation matrix in 3D extends easily from 2D

Scaling in 3D

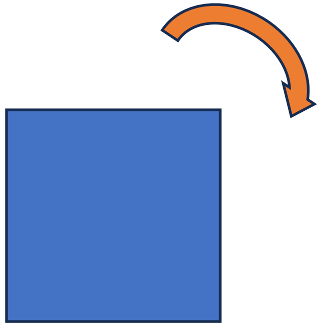
IITK CS360A

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2D 3D

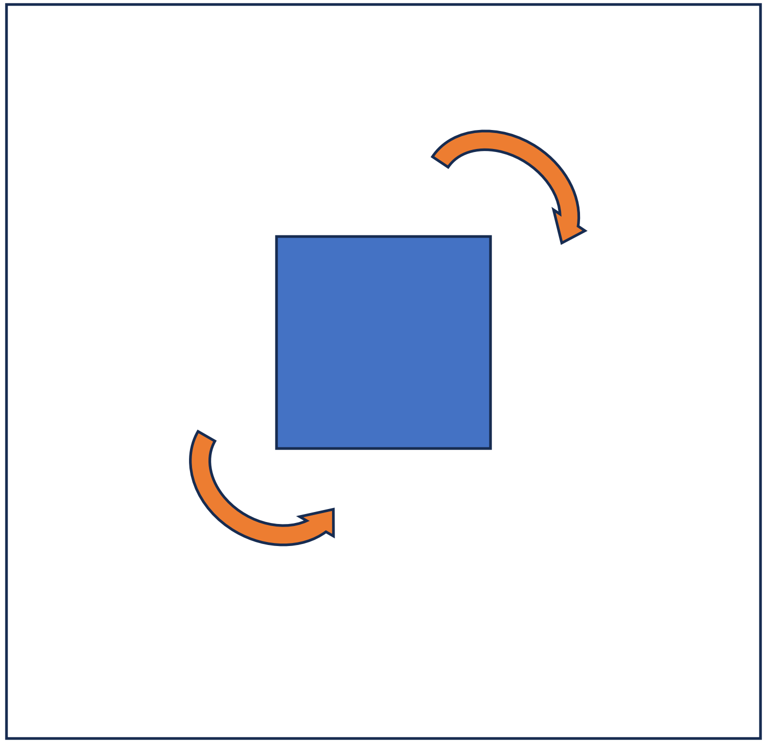
- Scaling matrix in 3D extends easily from 2D

Rotation from 2D to 3D



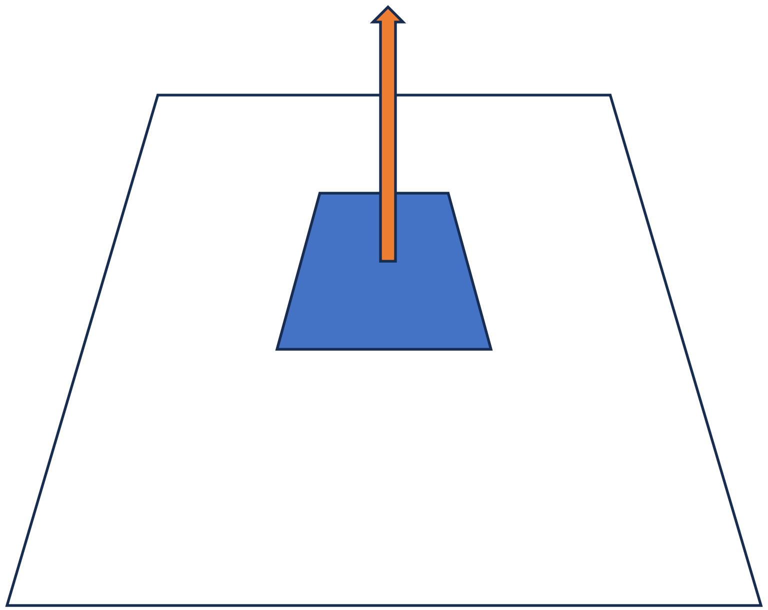
- Rotate an object in clockwise direction

Rotation from 2D to 3D



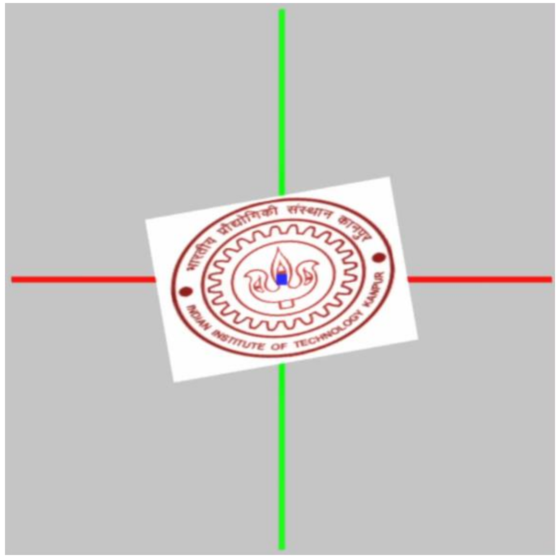
- Rotate an object in clockwise direction
- Rotate an object in counterclockwise direction
- 2D Rotation happens on a plane

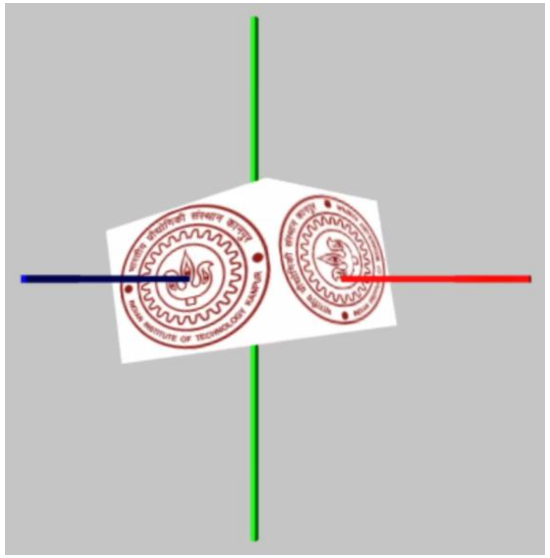
Rotation from 2D to 3D



- Rotate an object in clockwise direction
- Rotate an object in counterclockwise direction
- 2D Rotation happens on a plane
- In 2D, default axis of rotation is always Z-axis

Rotation from 2D to 3D

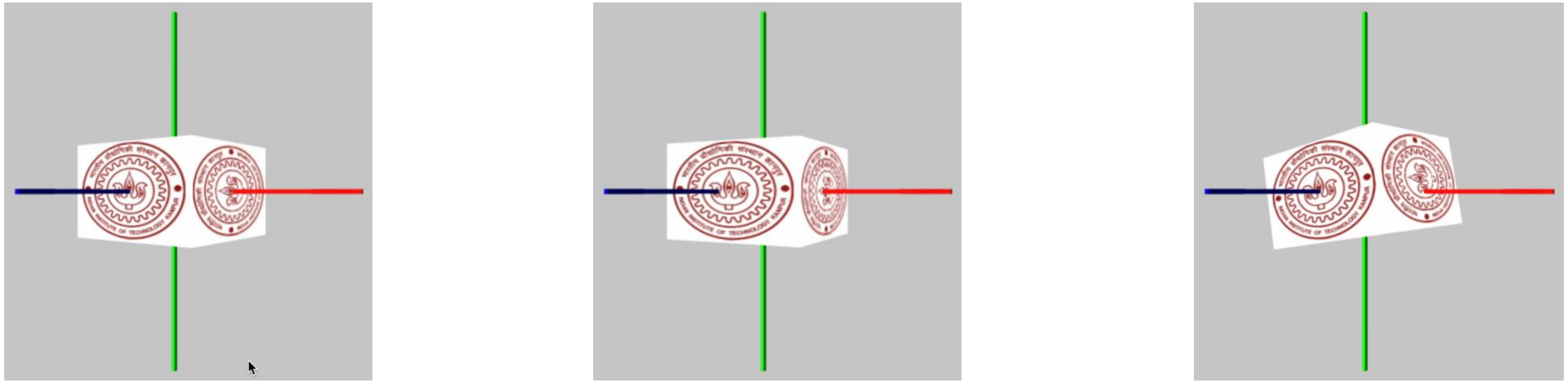




- 2D → 3D Rotation
- Rotation operation always happens with respect to a rotation axis
  - For 2D, the default rotation axis was Z axis
    - Pointing outward from the screen

Rotation in 3D

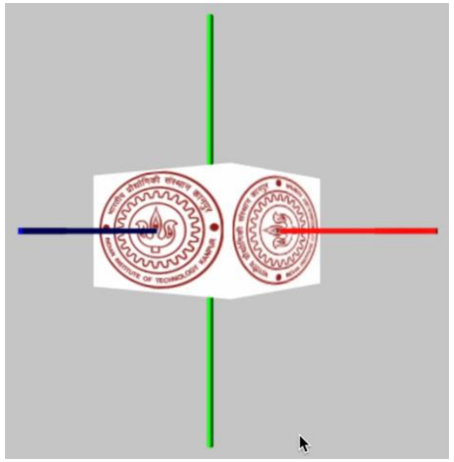
Z-axis
Y-axis
X-axis



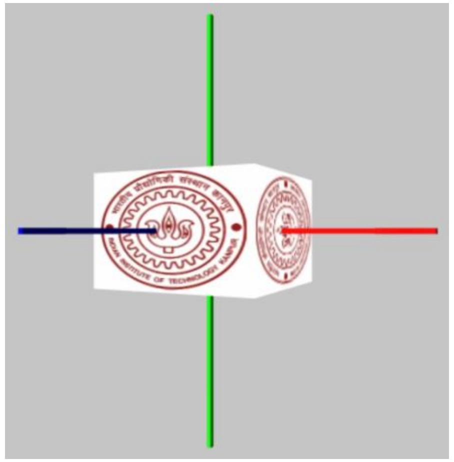
$$R_x: \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y: \begin{bmatrix} cos\theta & 0 & -sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z: \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation in 3D with Arbitrary Axis

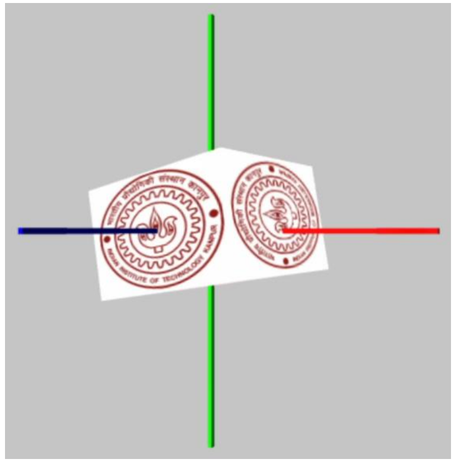
Z-axis
Y-axis
X-axis



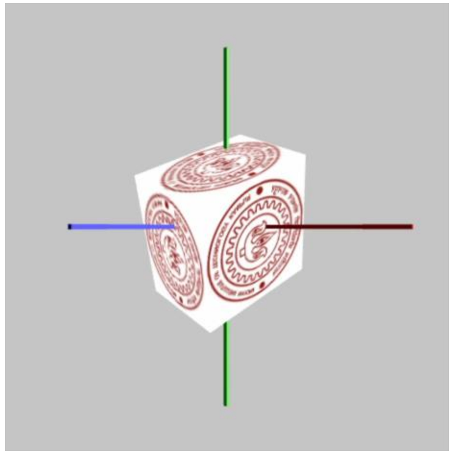
R_x



R_y



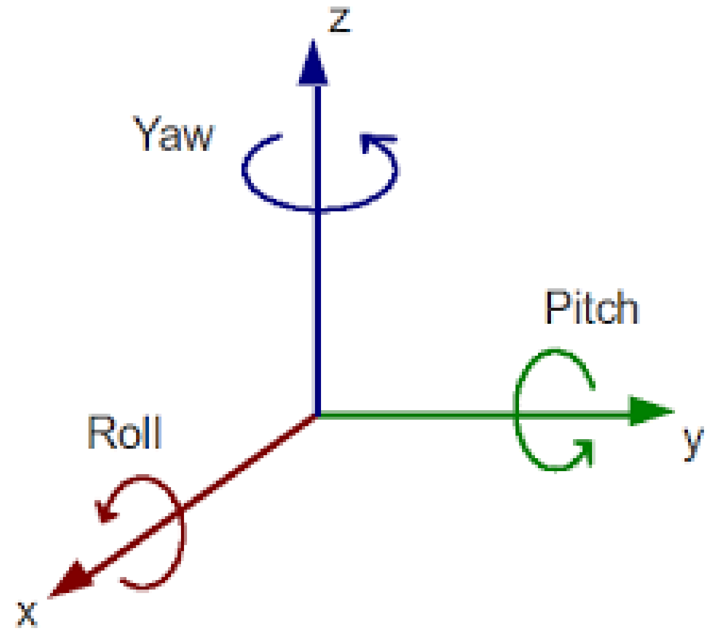
R_z



Rotation with respect to an arbitrary axis

Rotation About an Arbitrary Axis

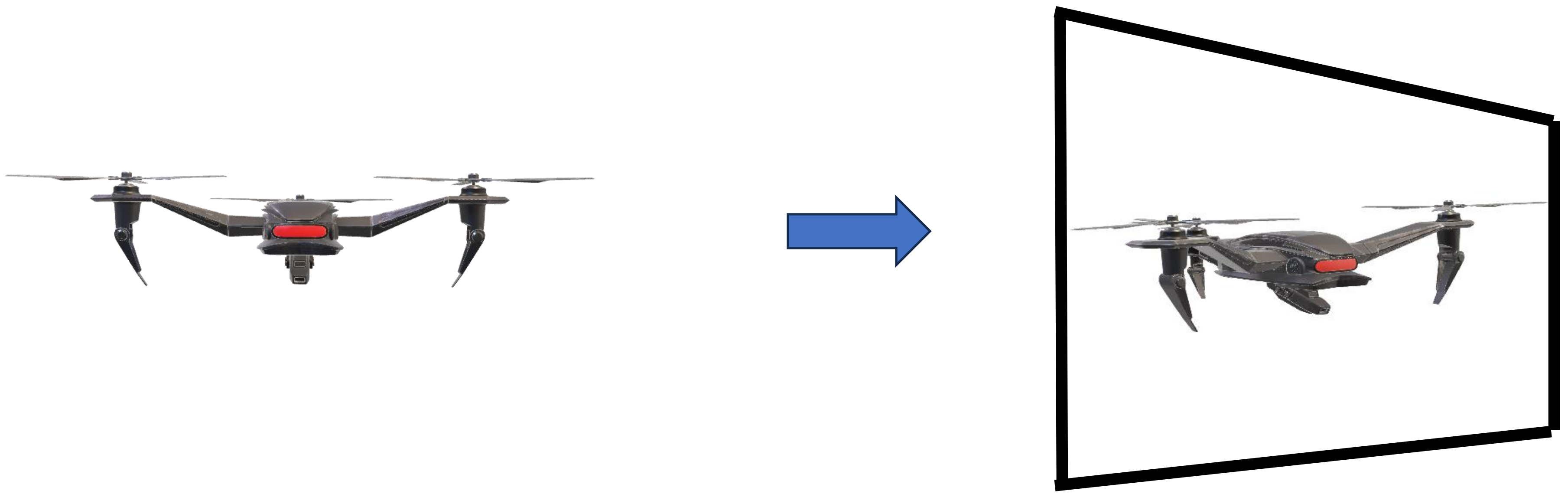
- Rotation about any arbitrary axis can be decomposed into rotation around X-axis, Y-axis, and Z-axis
- The order of applying the rotation is important to get the correct effect
- $\boldsymbol{R} = \boldsymbol{R_z(\alpha)}\boldsymbol{R_y(\beta)}\boldsymbol{R_x(\gamma)}$



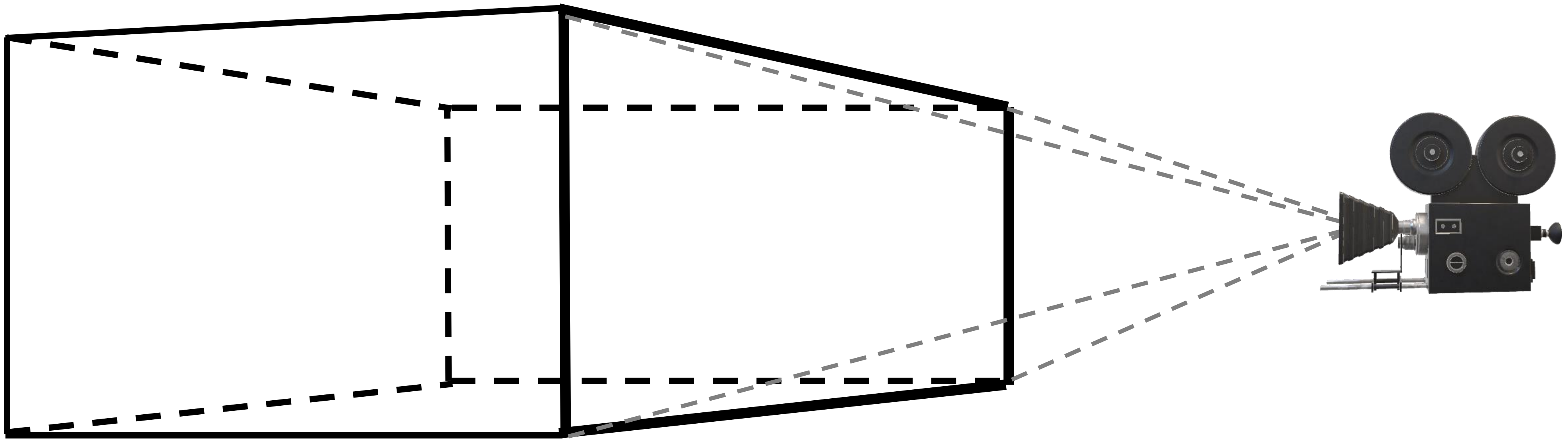
Yaw (Z-axis) **Pitch (Y-axis)** **Roll (X-axis)**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & sin\alpha & 0 \\ 0 & -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\beta & 0 & -sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ sin\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\gamma & sin\gamma & 0 & 0 \\ -sin\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

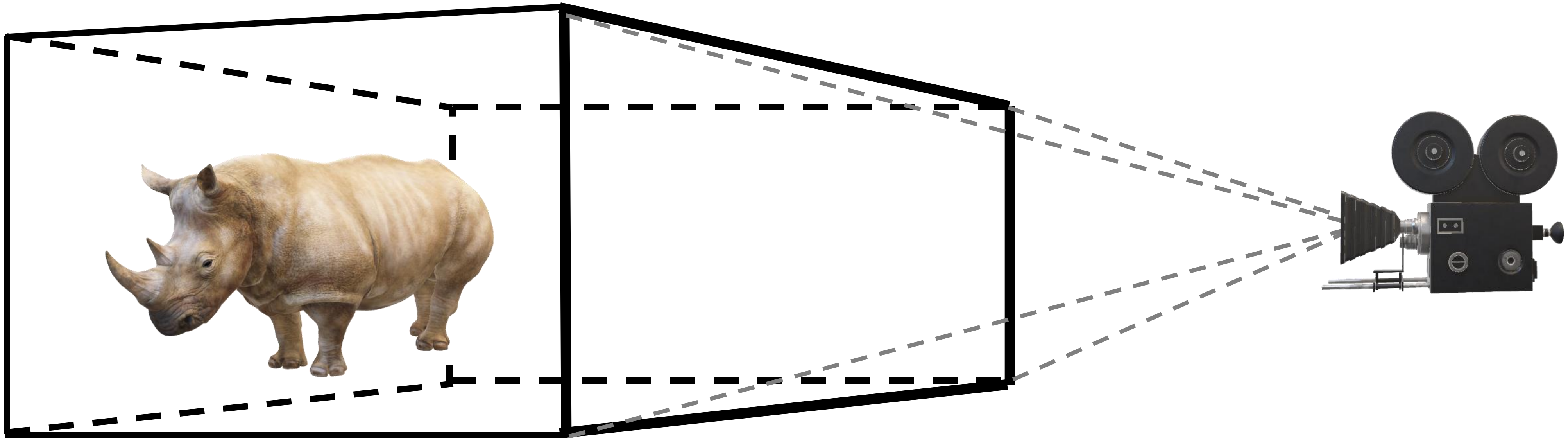
Viewing



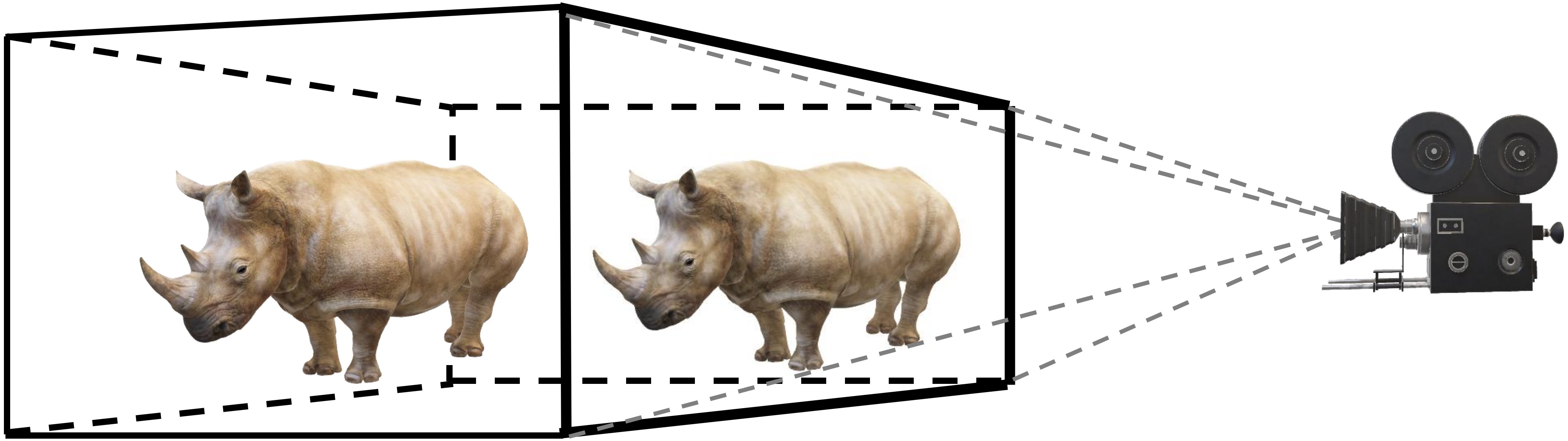
Viewing



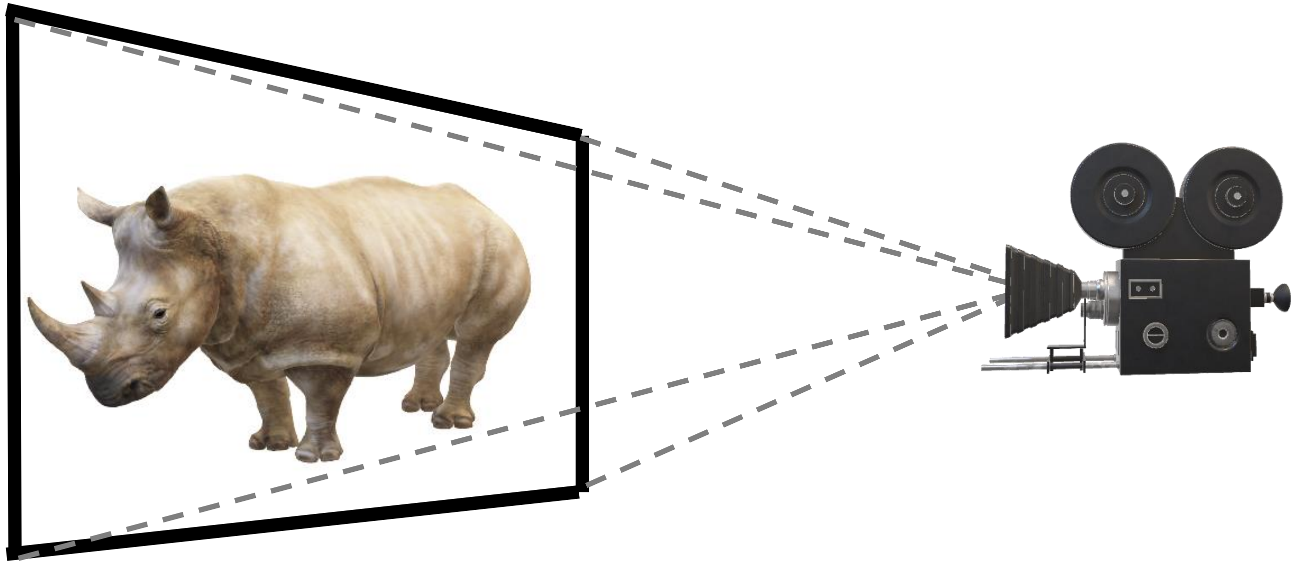
Viewing



Viewing

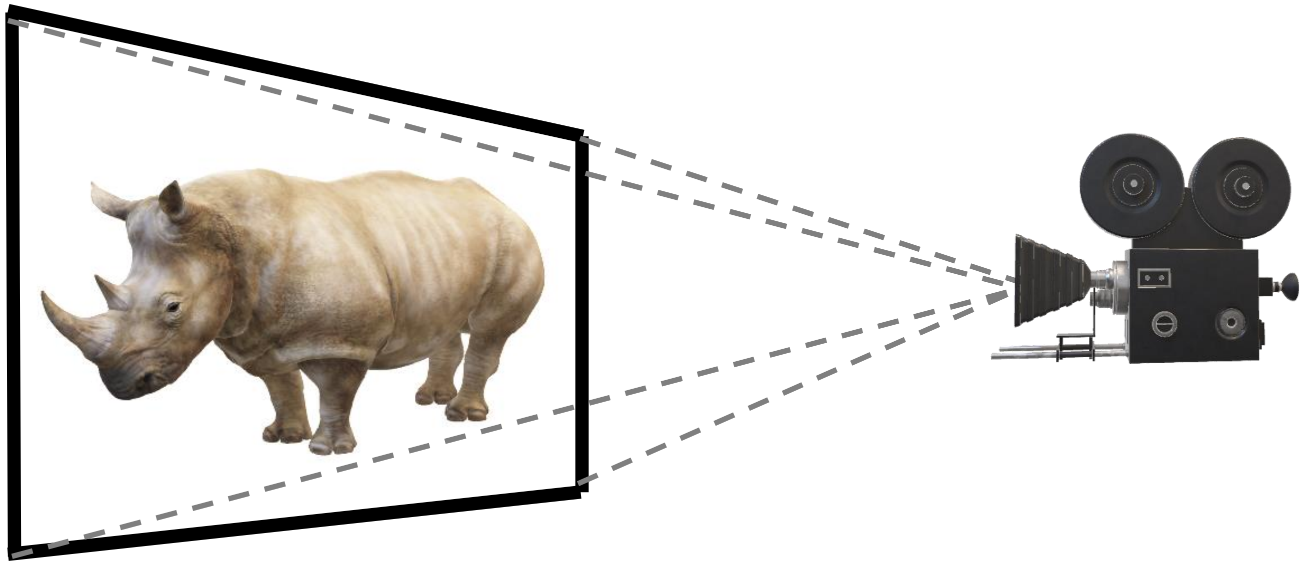


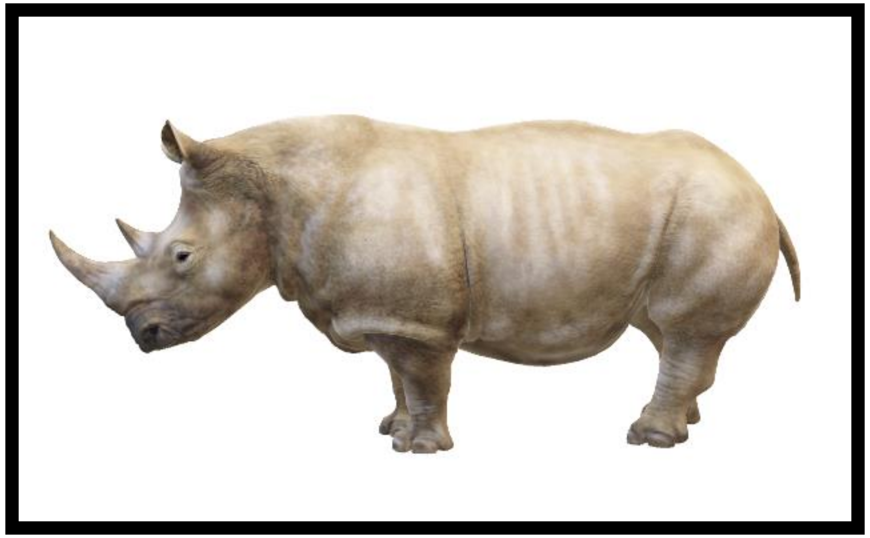
Viewing



We project the objects on the screen

Viewing

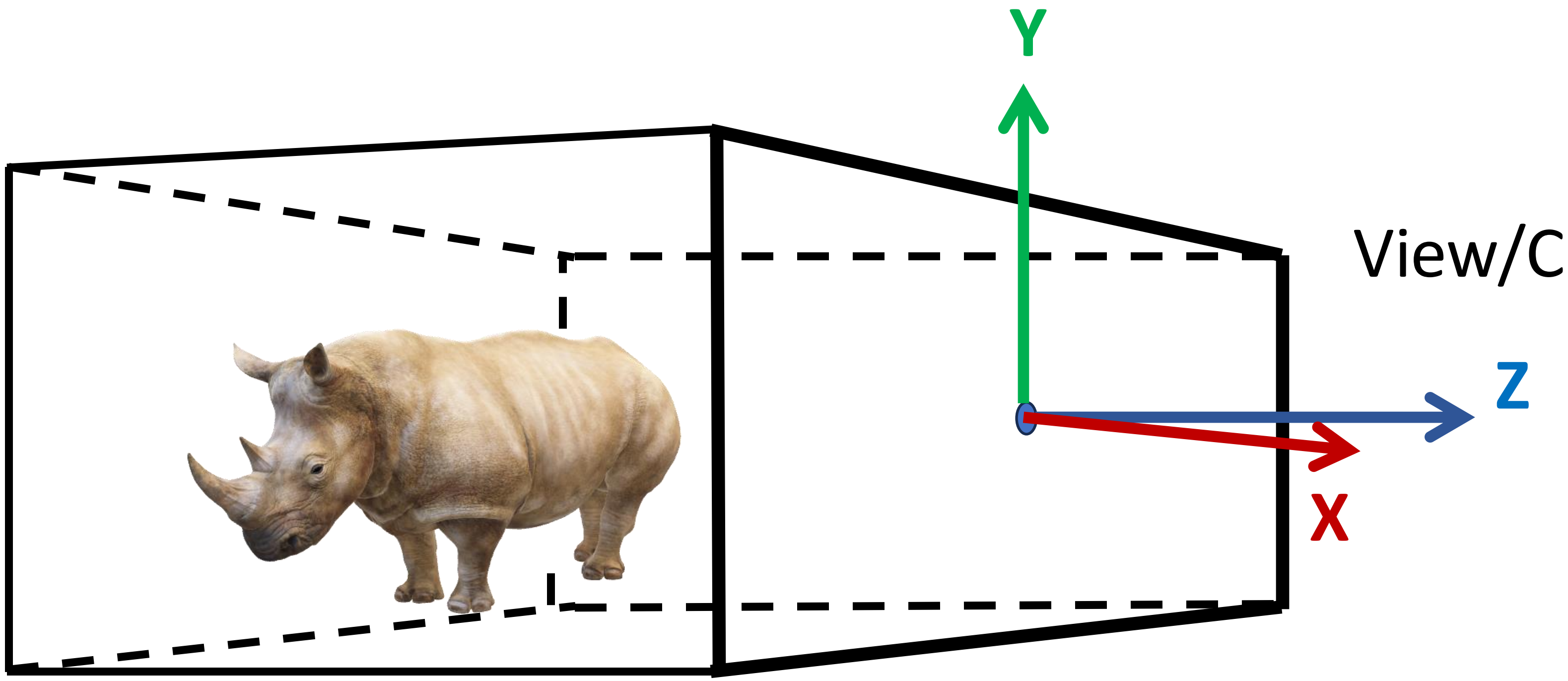




This is what we see on screen

Viewing Coordinate

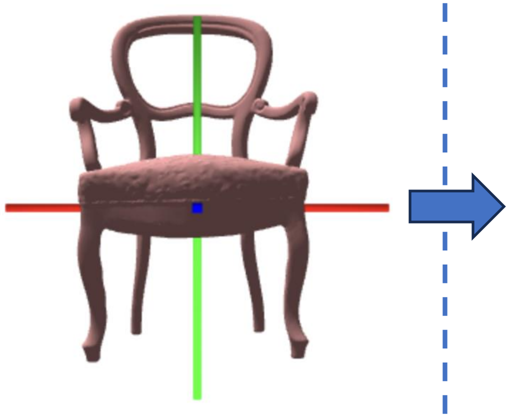
We need to define a coordinate frame for viewing



View volume

The scene objects will have negative z values

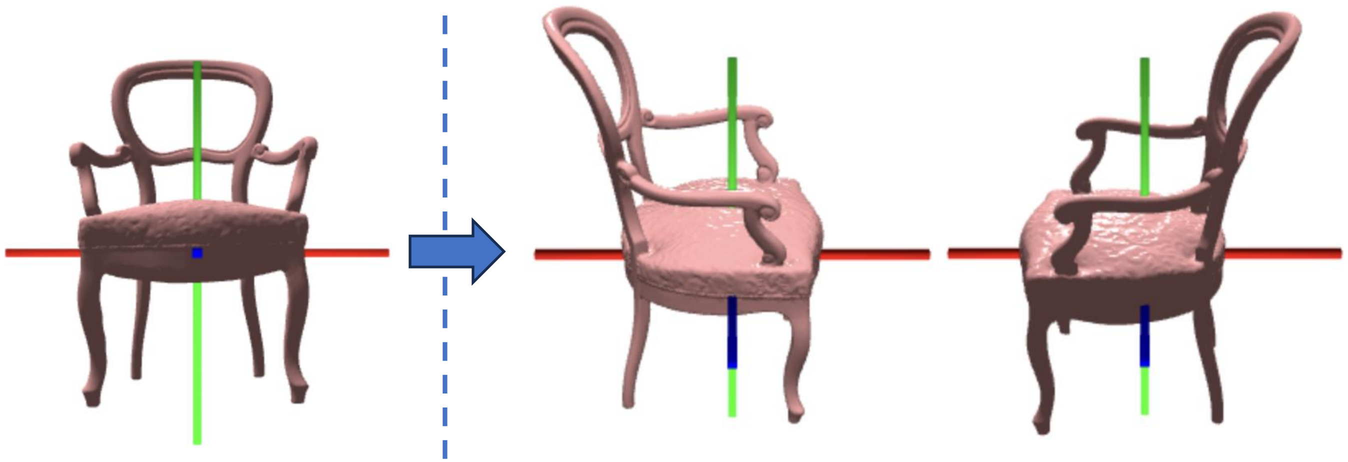
Viewing



Model/Object
Space

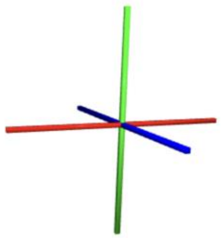
Viewing

Model Transformation
Matrix

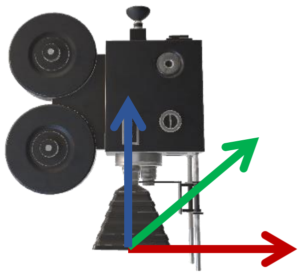


Model/Object
Space

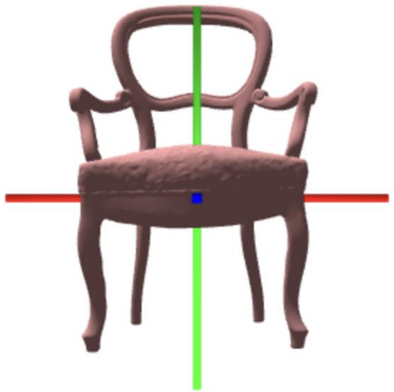
World Space

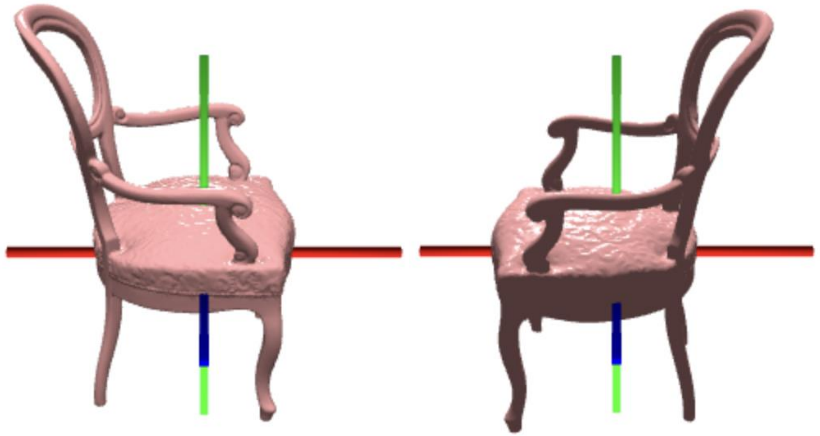


Viewing



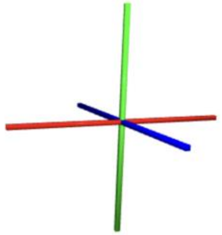
Model Transformation Matrix



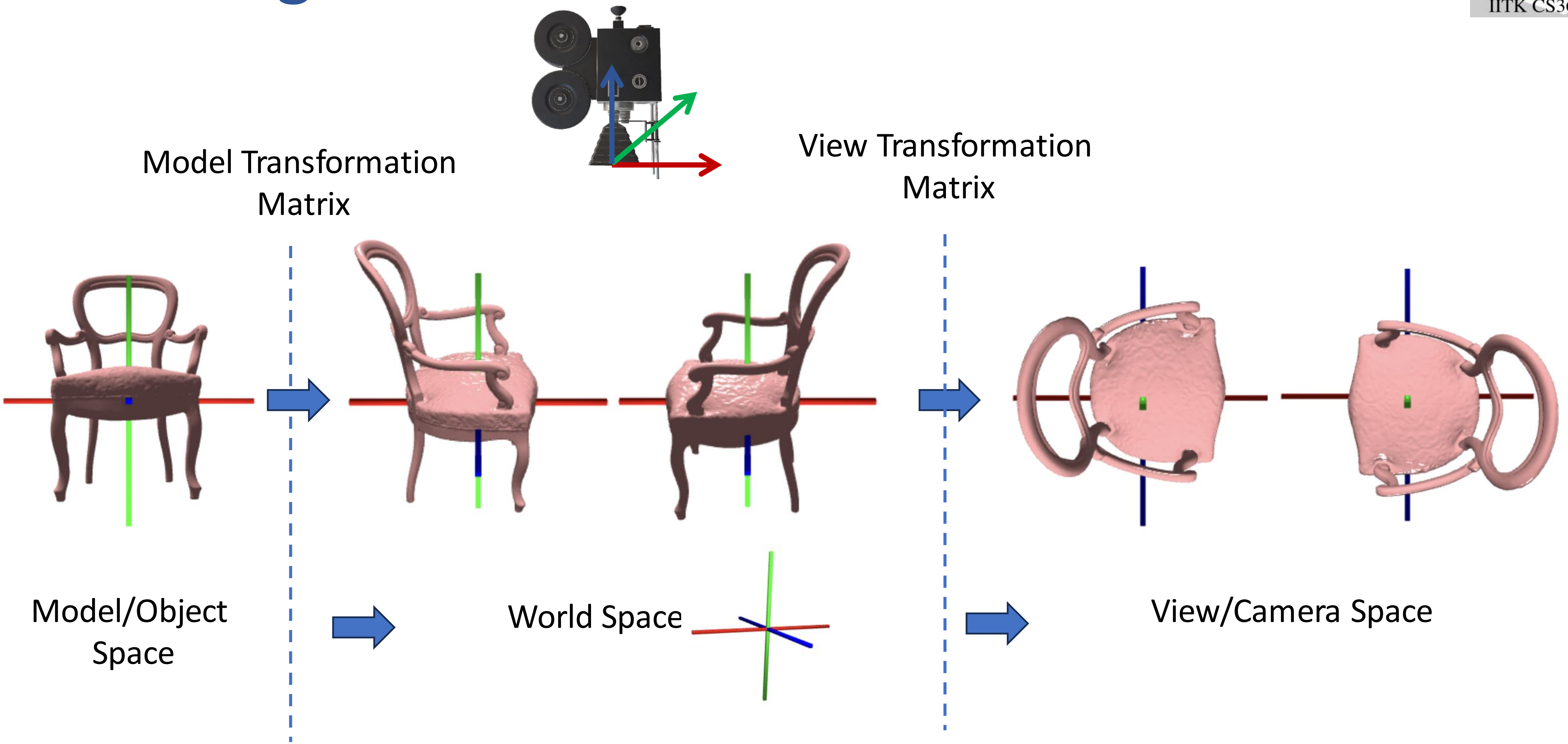


Model/Object Space

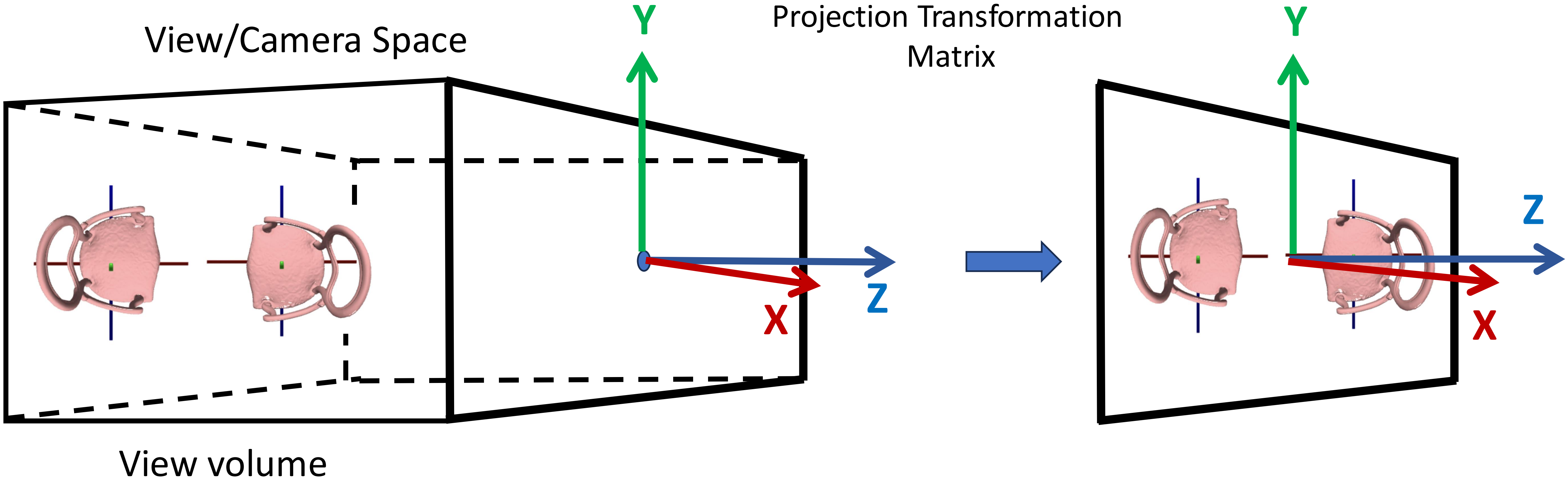
World Space



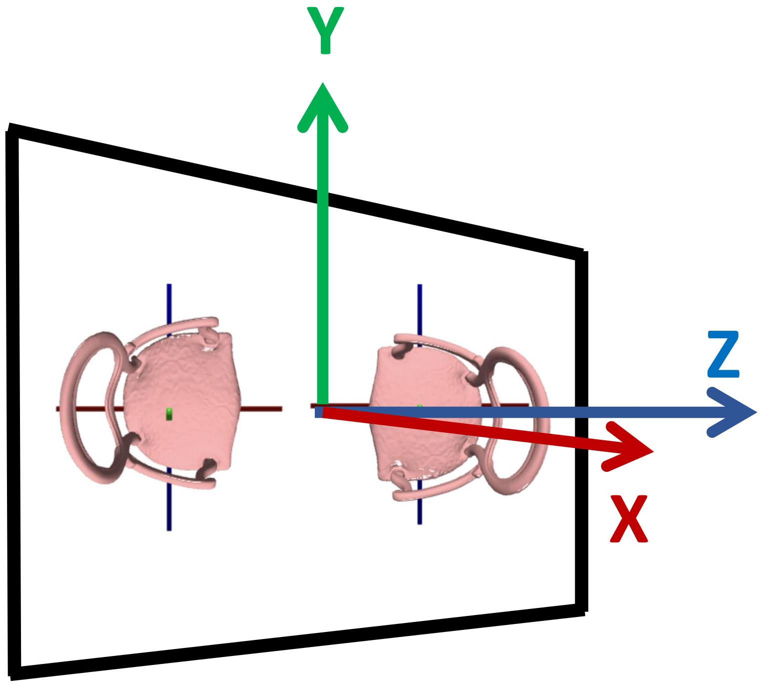
Viewing

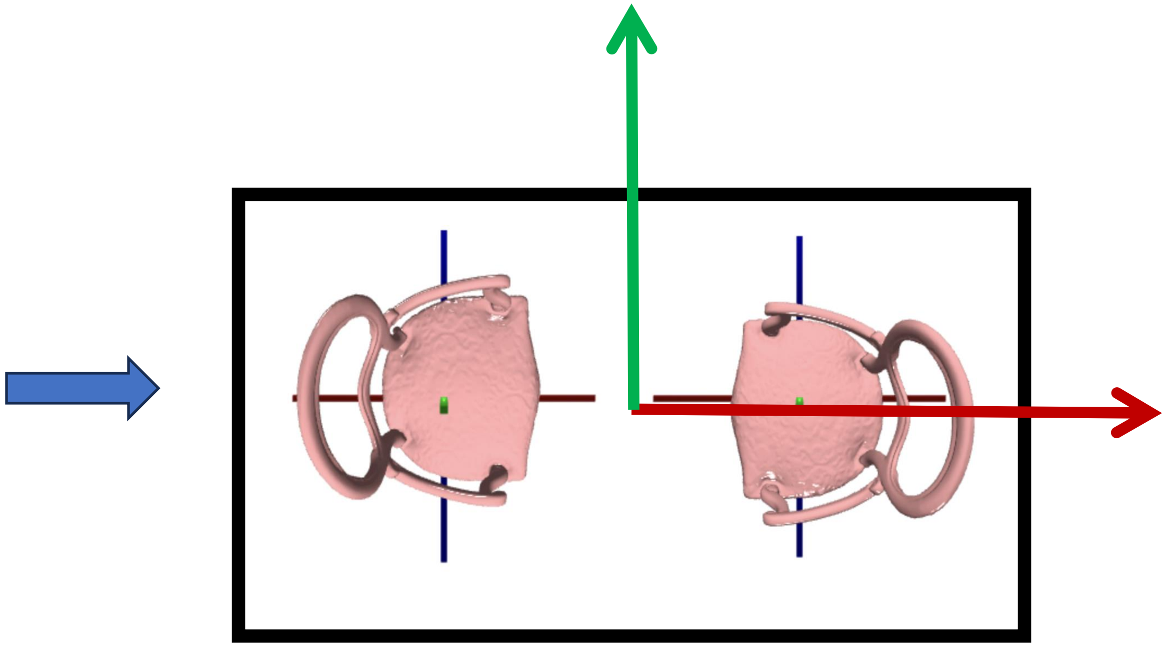

Model Transformation
Matrix
View Transformation
Matrix
Model/Object
Space
World Space
View/Camera Space

Viewing


View/Camera Space
Y
Projection Transformation Matrix
Y
Z
Z
X
X
View volume

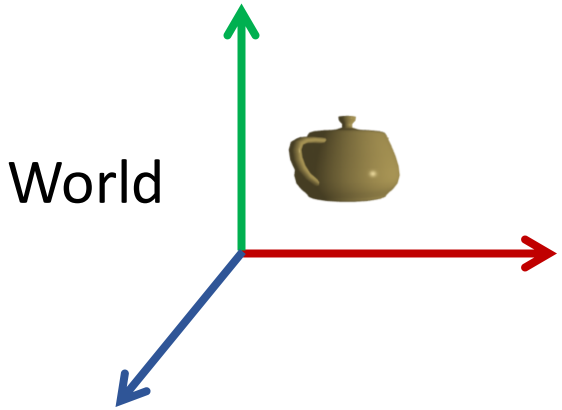
Viewing



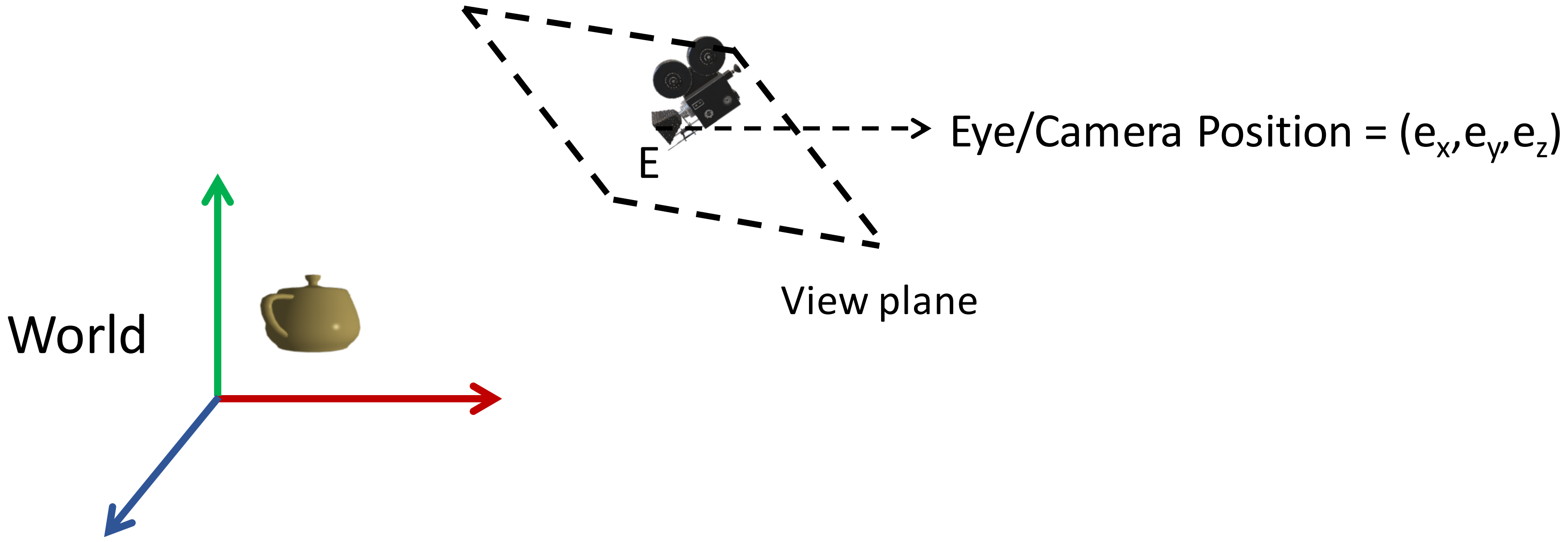


This is what we see in the screen

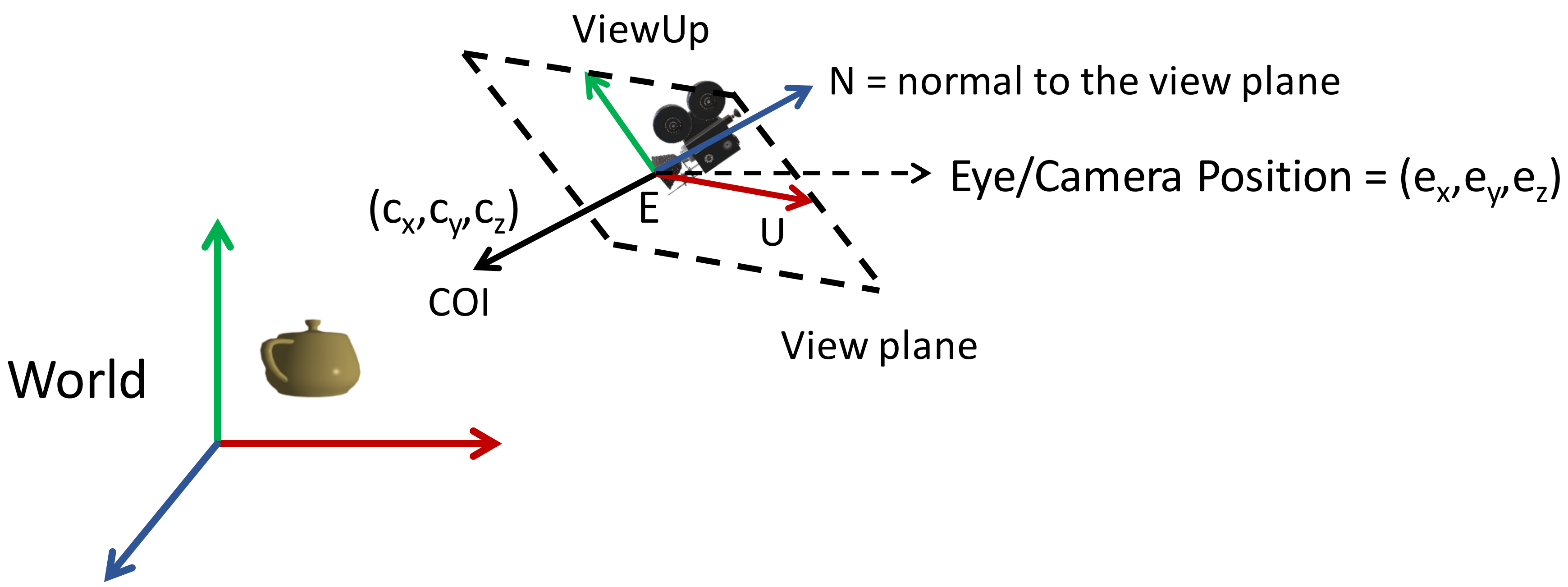
How Do We Form Camera Coordinate System



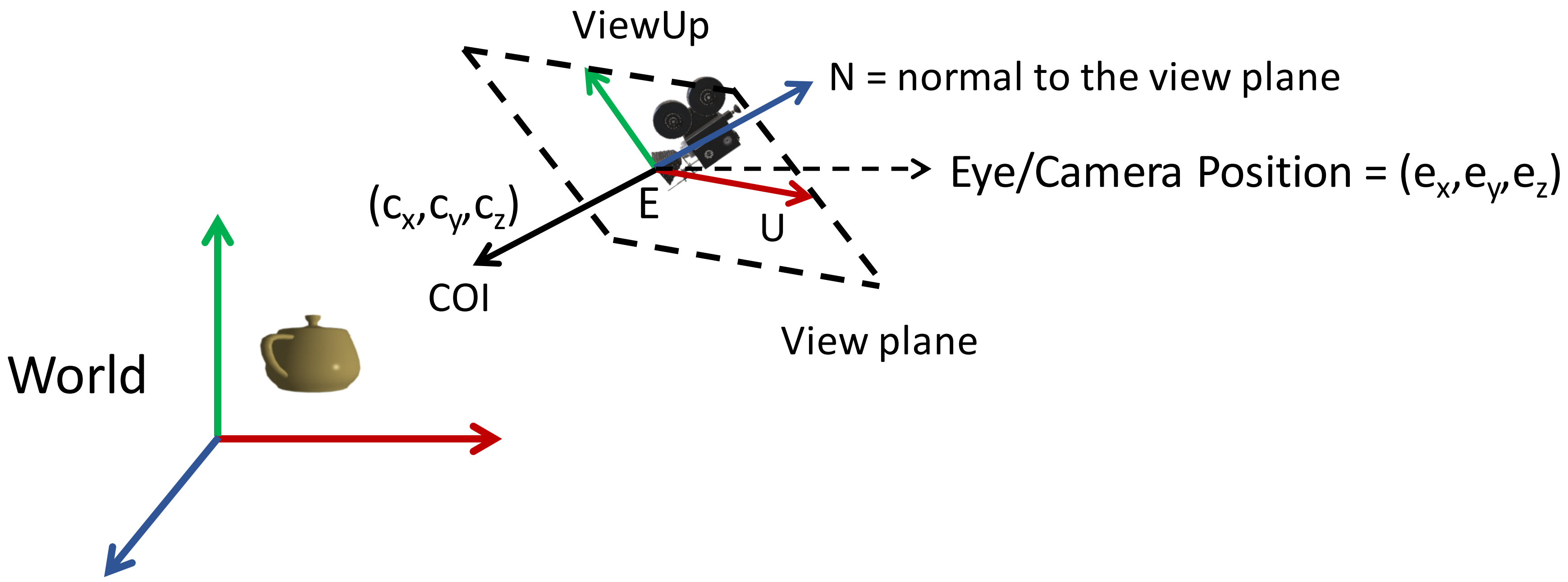
How Do We Form Camera Coordinate System



How Do We Form Camera Coordinate System



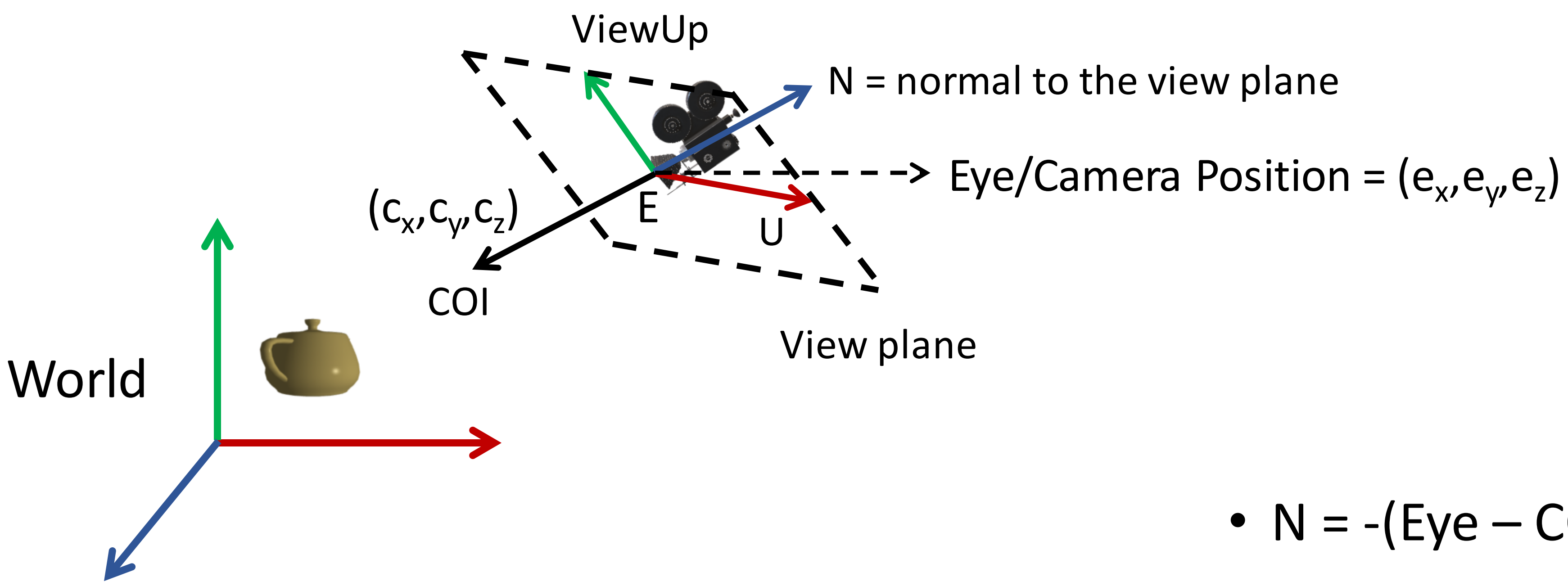
How Do We Form Camera Coordinate System



- Given:
  - COI (c_x,c_y,c_z) is the center of Interest
  - Eye (e_x,e_y,e_z)
- Vector N is normal vector to the viewing plane
  - Pointing away from world

How Do We Form Camera Coordinate System

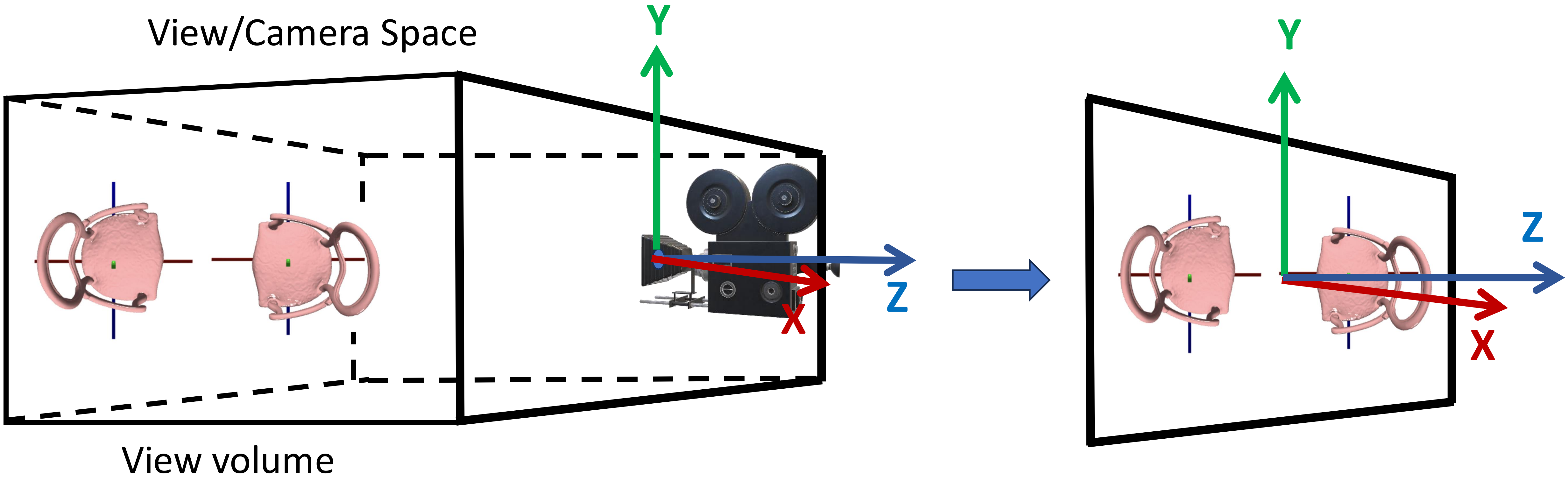
IITK CS360A



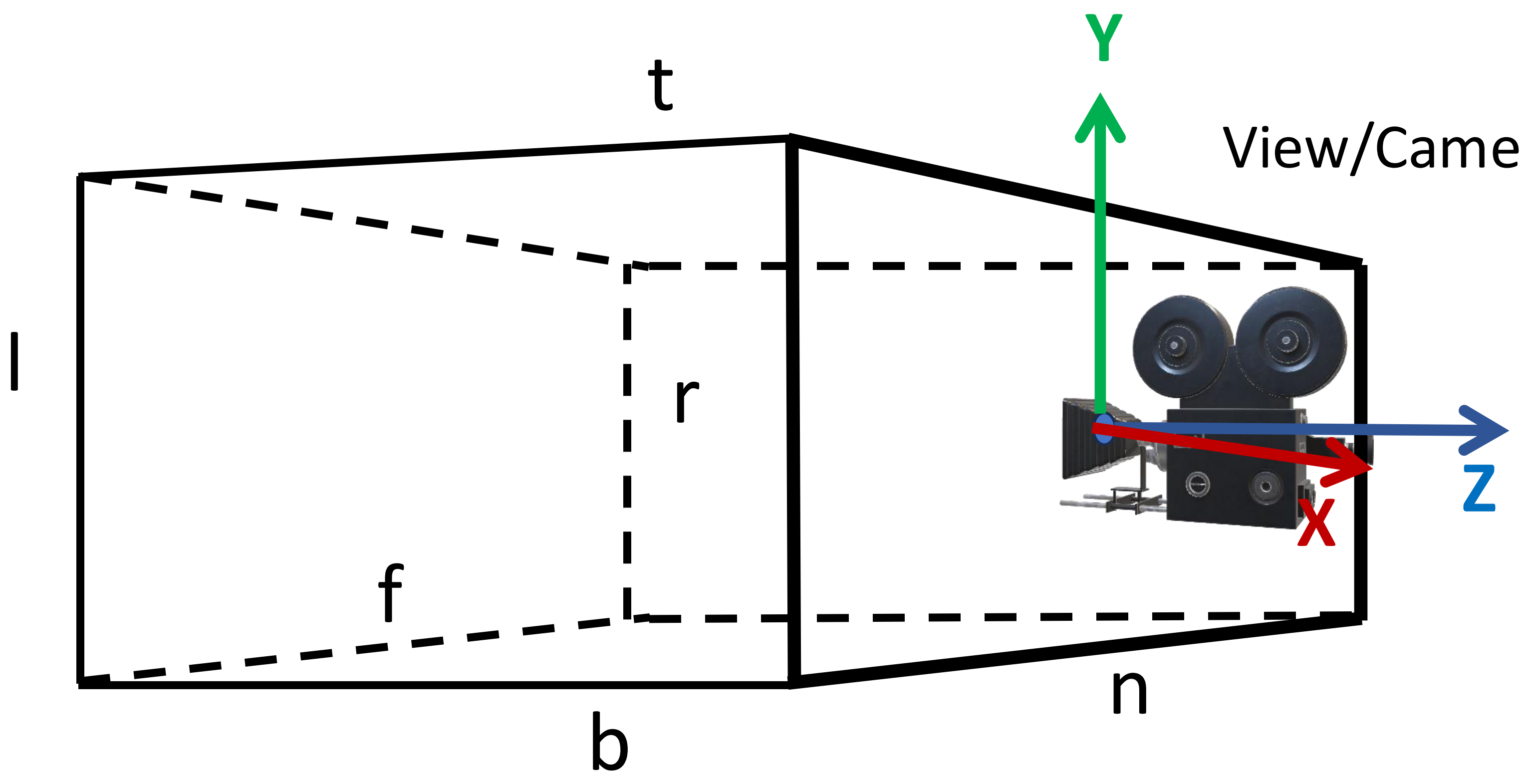
- Given:
  - COI (c_x,c_y,c_z) is the center of Interest
  - Eye (e_x,e_y,e_z)
- Vector N is normal vector to the viewing plane
  - Pointing away from world

- N = -(Eye – COI), n = N/|N|
- ViewUp: Direction of head up
- U = ViewUp X n, U = U/|U|

Projection



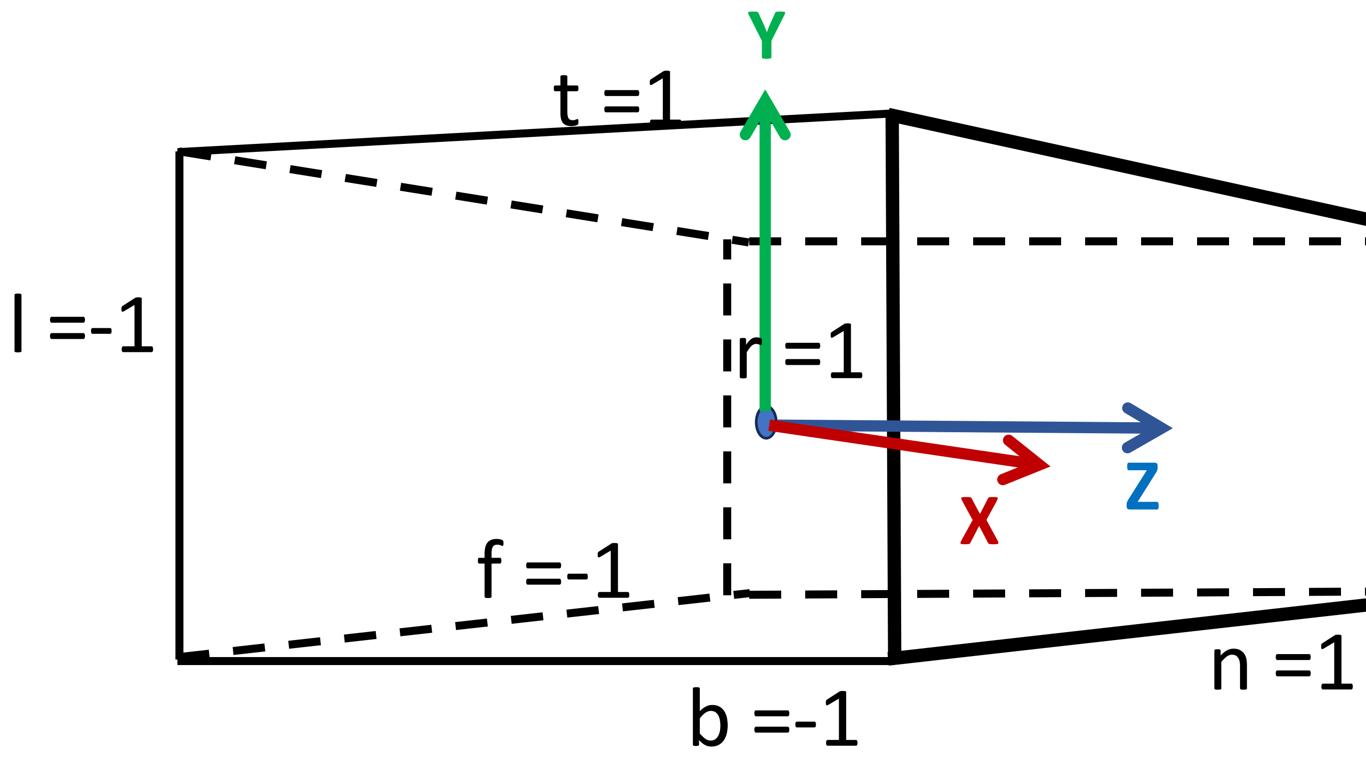
Projection: How Do We Define View Volume



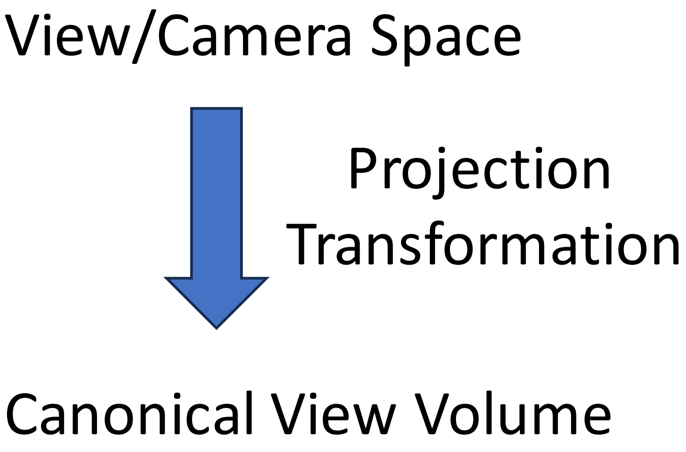
View volume

Projection: How Do We Define View Volume

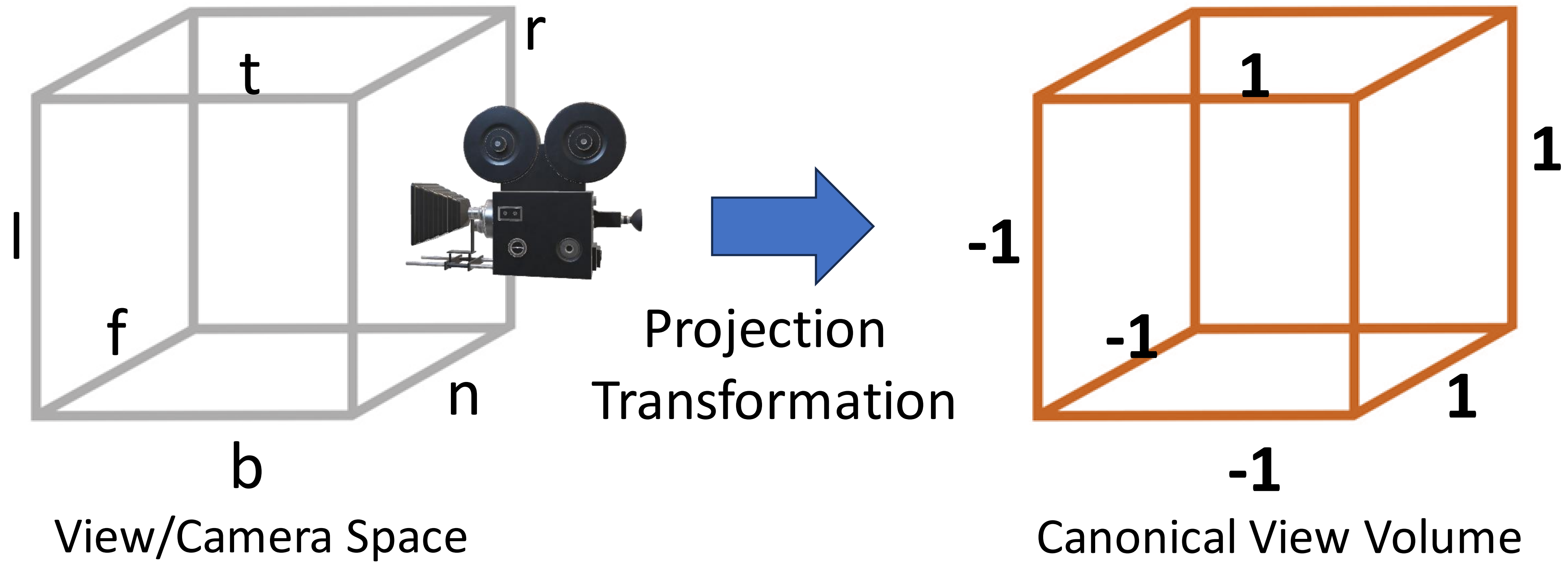
IITK CS360A



Canonical view volume

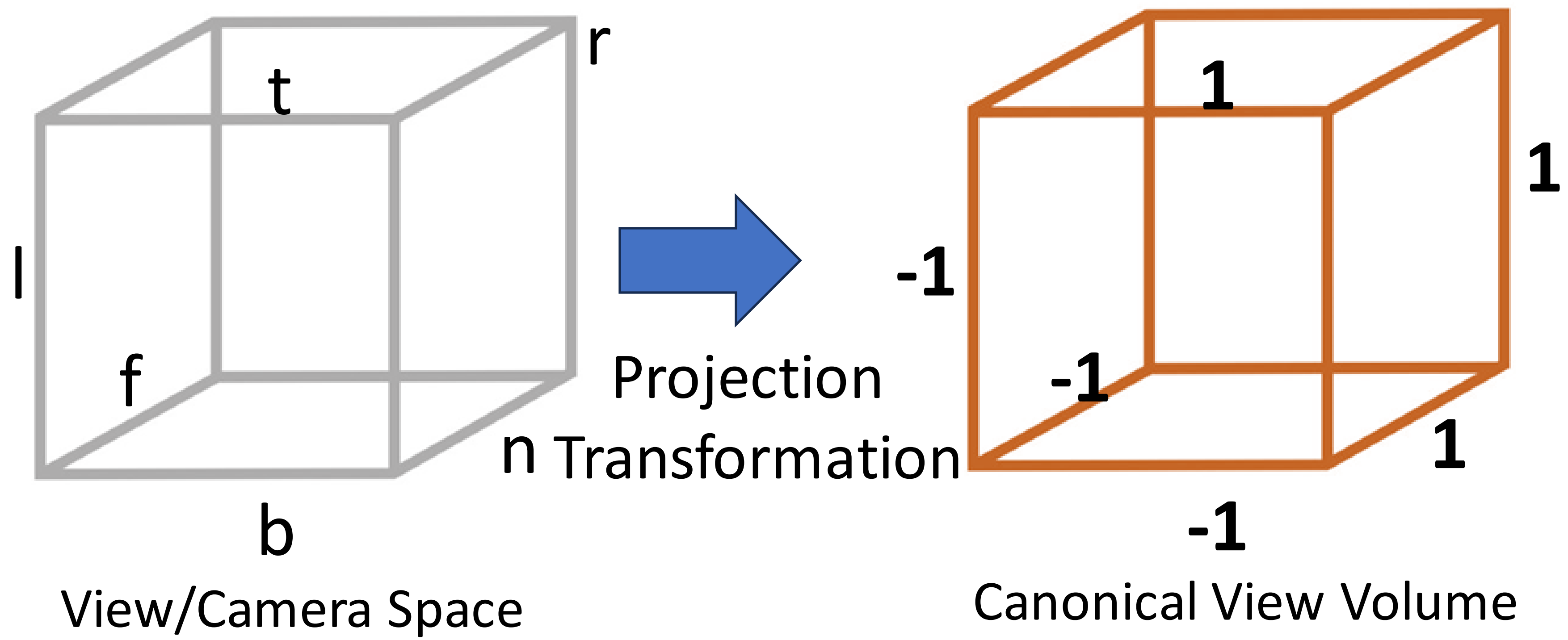


Orthographic Projection



We need a transformation matrix to transform objects from camera to canonical view volume space

Orthographic Projection

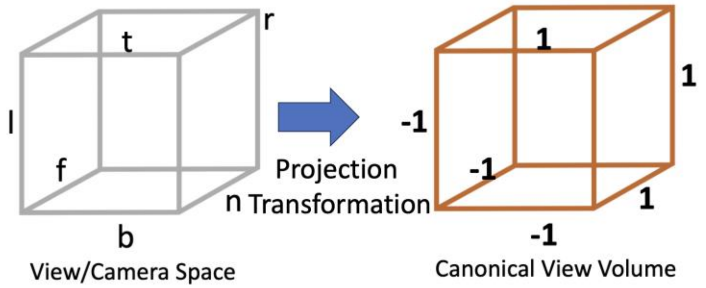


Orthographic Projection

In Canonical View Volume Space

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In Camera Space



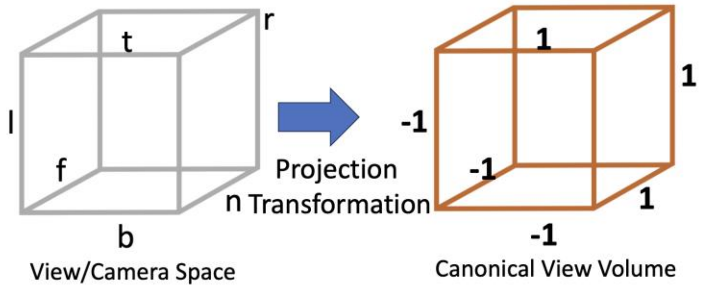
- Let's see how we can form the Orthographic projection matrix

Orthographic Projection

In Canonical View Volume Space

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In Camera Space



- Homogeneous coordinate system

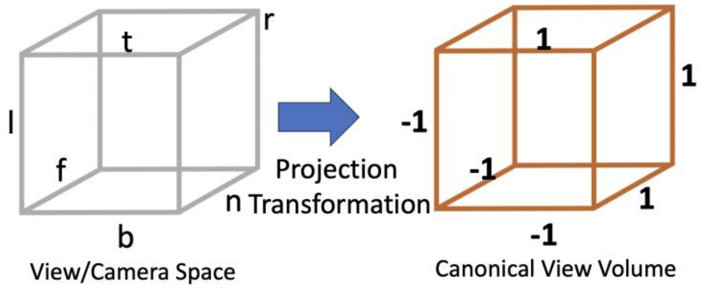
Orthographic Projection

IITK CS360A

In Canonical View Volume Space

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In Camera Space



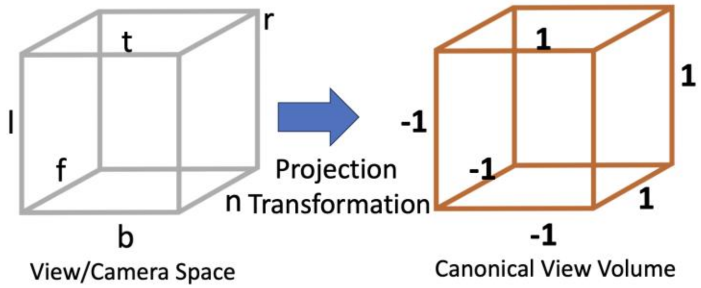
- X-Y-Z directions are aligned from Camera space to Canonical view volume
- There is no change in direction from view space to canonical view volume space
- Hence the first 3X3 elements in the matrix will not have any rotational components

Orthographic Projection

In Canonical View Volume Space

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} ? & 0 & 0 & ? \\ 0 & ? & 0 & ? \\ 0 & 0 & ? & ? \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In Camera Space



- X-Y-Z directions are aligned from Camera space to Canonical view volume
- There is no change in direction from view space to canonical view volume space
- Hence the first 3X3 elements in the matrix will not have any rotational components

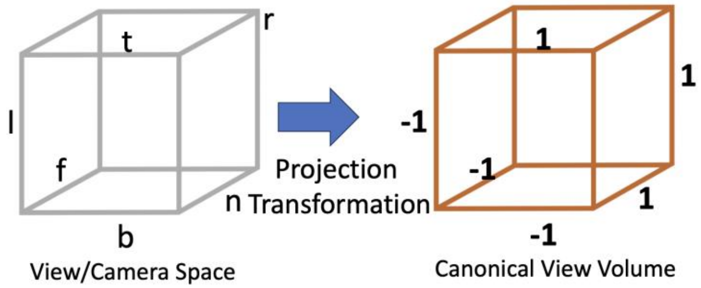
Orthographic Projection

IITK CS360A

In Canonical View Volume Space

$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix} = \begin{bmatrix}?&0&0&?\\0&?&0&?\\0&0&?&?\\0&0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

In Camera Space



- So, we will have some translation and scaling involved in the orthographic projection matrix

Orthographic Projection

IITK CS360A

In Canonical View Volume Space

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{f+n}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In Camera Space

Orthographic Projection

IITK CS360A

In Canonical View Volume Space

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\frac{2}{r-l}} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{f+n}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In Camera Space

- We are kind of normalizing by the length $(r-l)$ and then transforming the value to 0-2 range since length of a side in canonical view volume is 2
- Same logic applies for Y and Z scaling components

Orthographic Projection

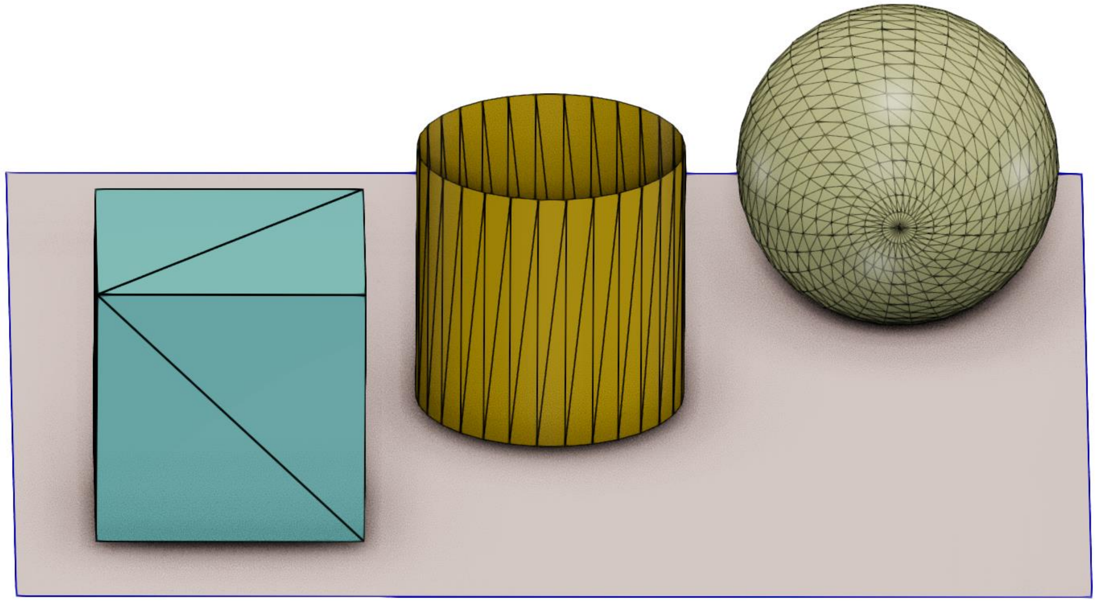
IITK CS360A

$$\begin{array}{c} \text{In Canonical View} \\ \text{Volume Space} \end{array} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{f+n}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{In Camera Space}$$

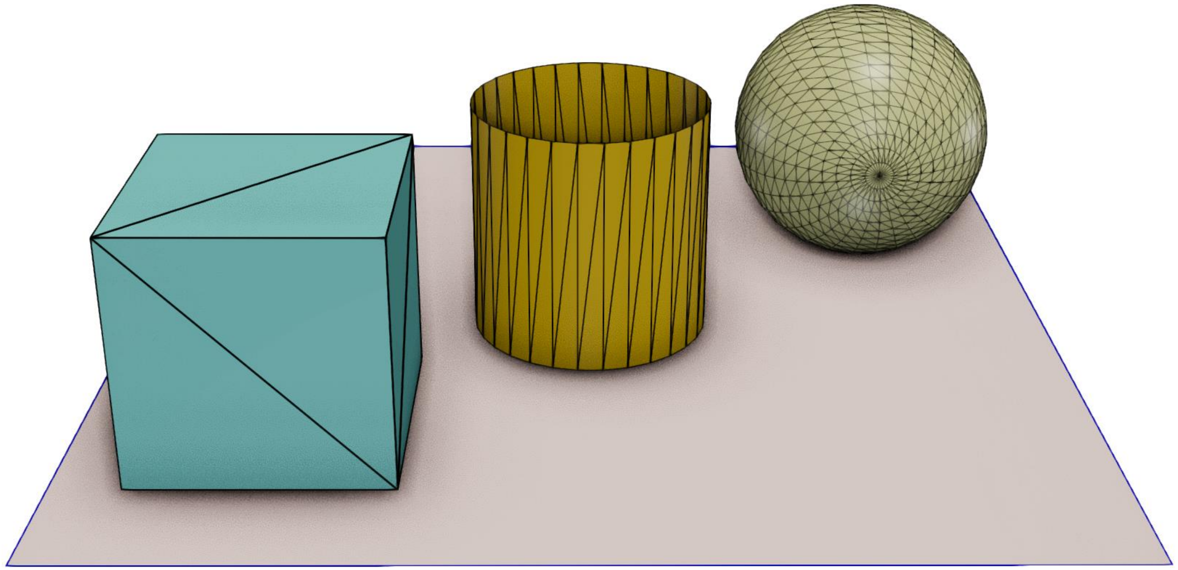
Now

- Now we have to shift the coordinate frame center at the center of canonical view volume
- Let's say, $x = l$, then the first term of the matrix becomes -1 which is what we want
- If $x = r$, then the first term of the matrix becomes 1 which is what we want

Projection



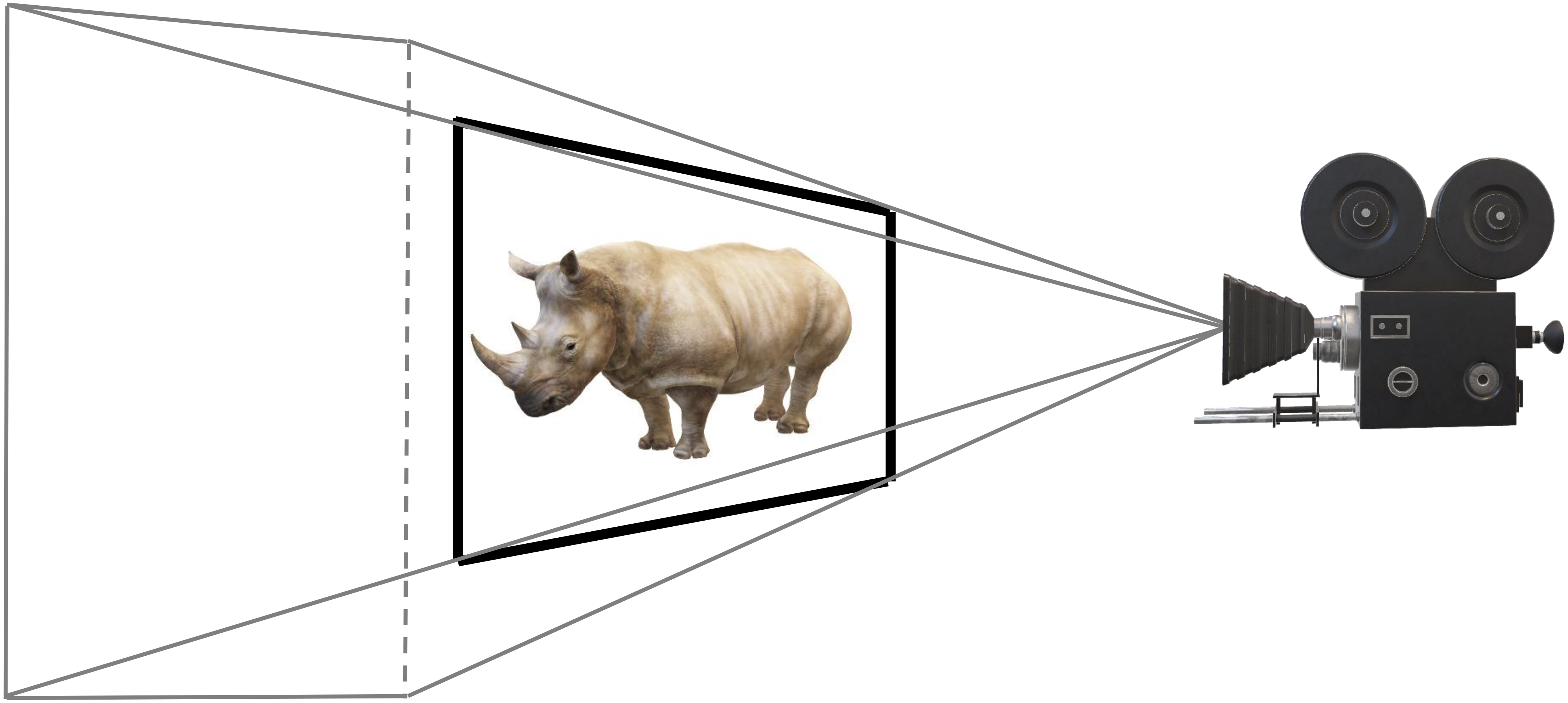
Orthographic Projection



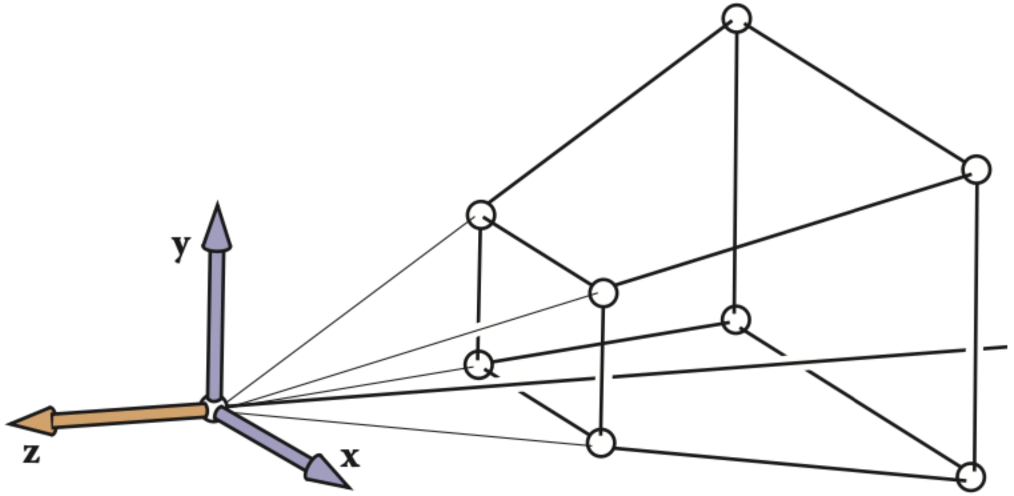
Perspective Projection

Perspective Projection

IITK CS360A

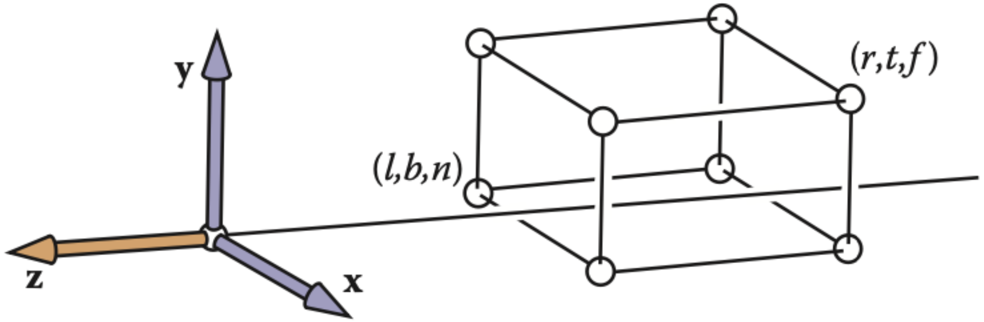


Perspective Projection

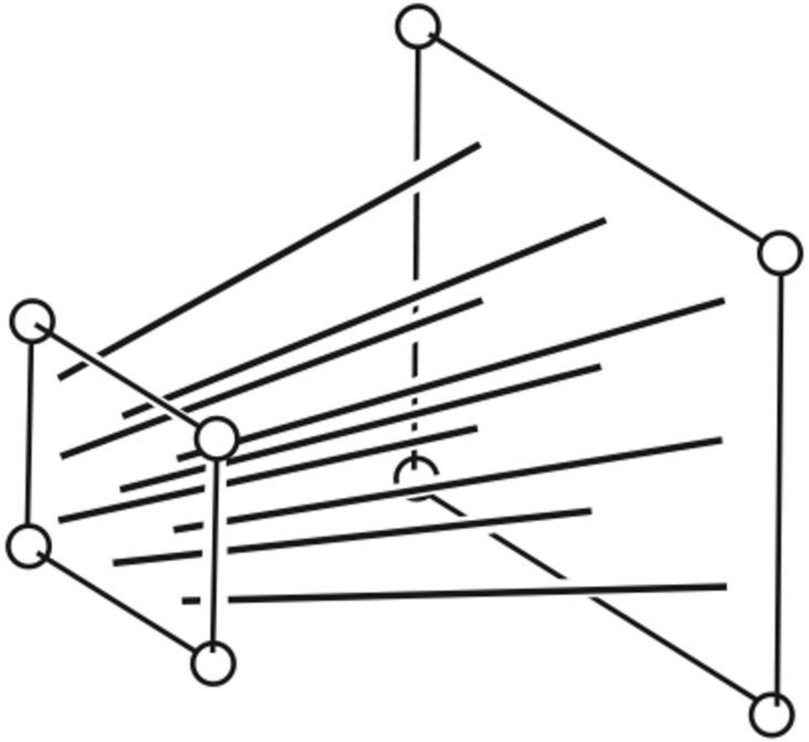


- Perspective transformation deforms the truncated frustum to the cube shape as shown below
- Then apply orthographic projection
- **So, perspective transformation followed by orthographic projection is perspective projection**

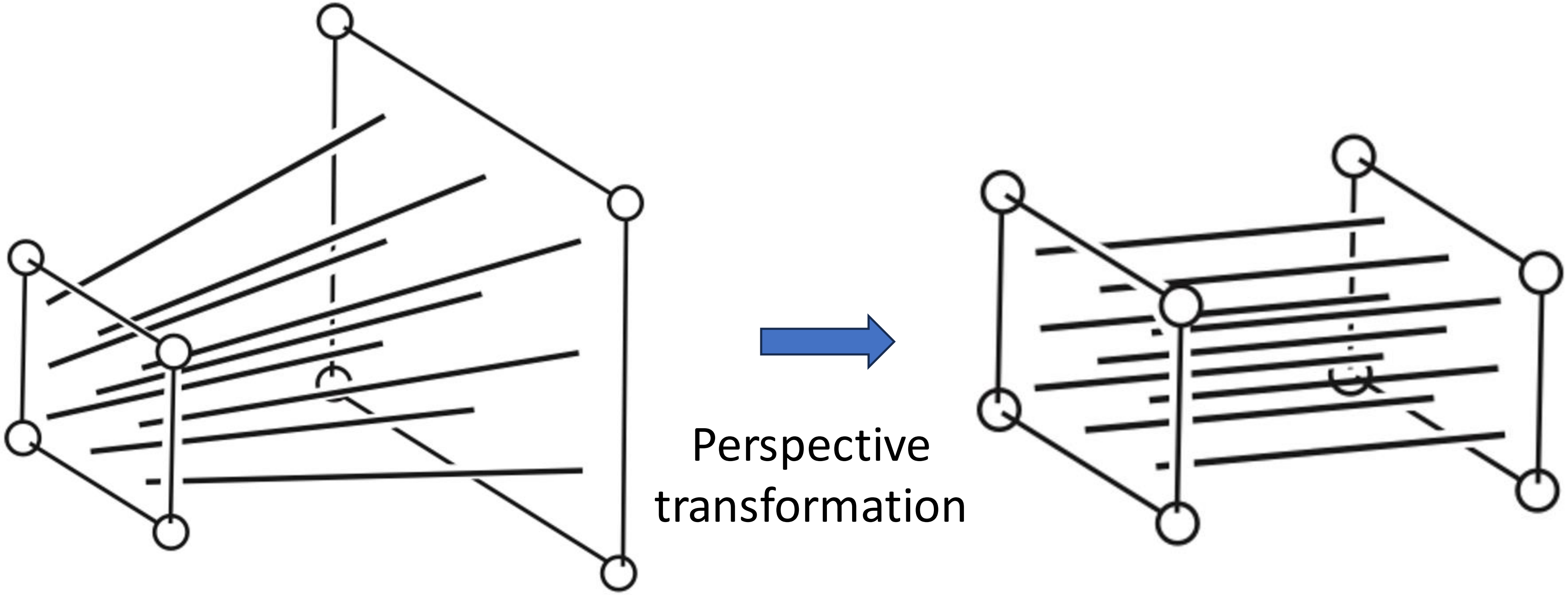
Perspective Transformation



Perspective Transformation

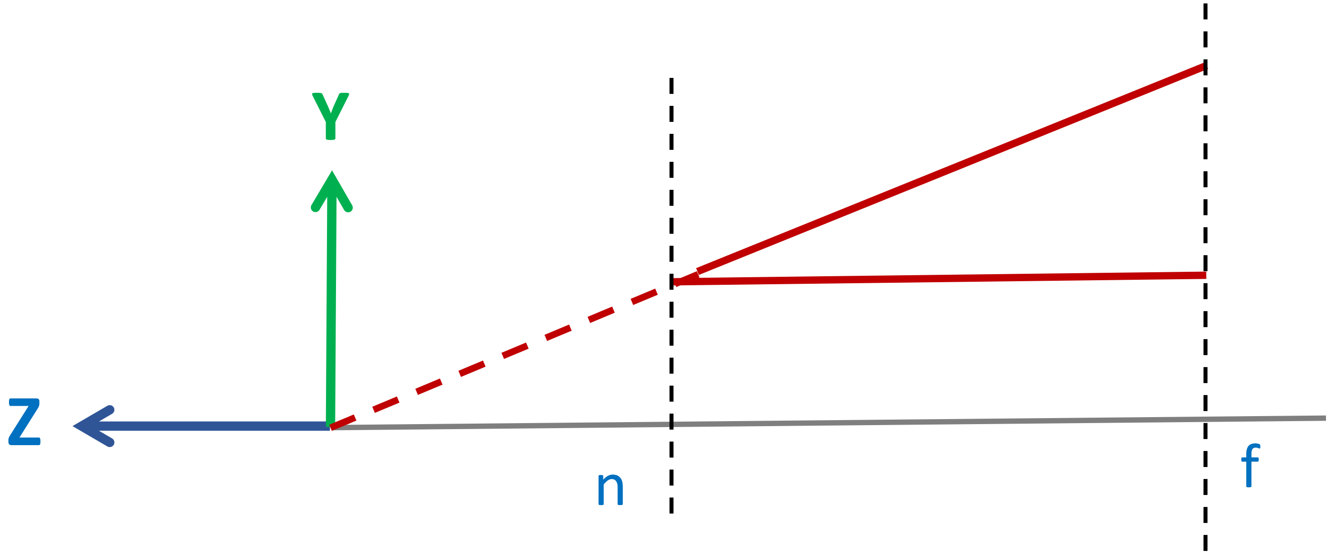


Perspective Transformation



Perspective Transformation

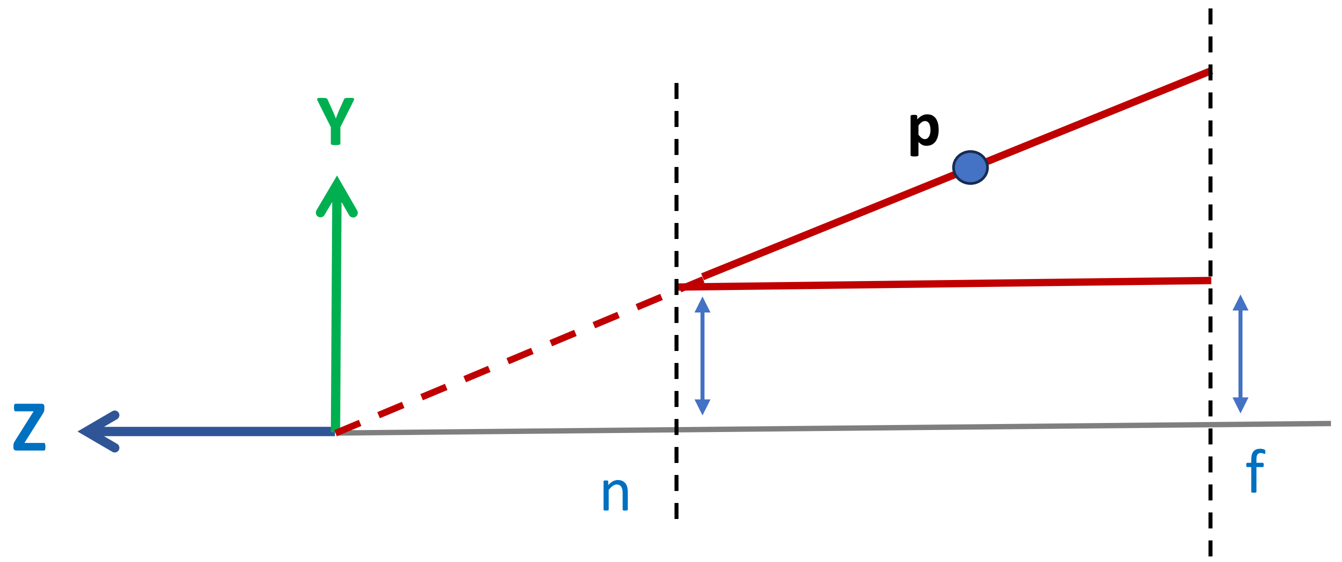
IITK CS360A



Looking at the Y-Z plane

- We want to deform the slanted red line into the straight red line

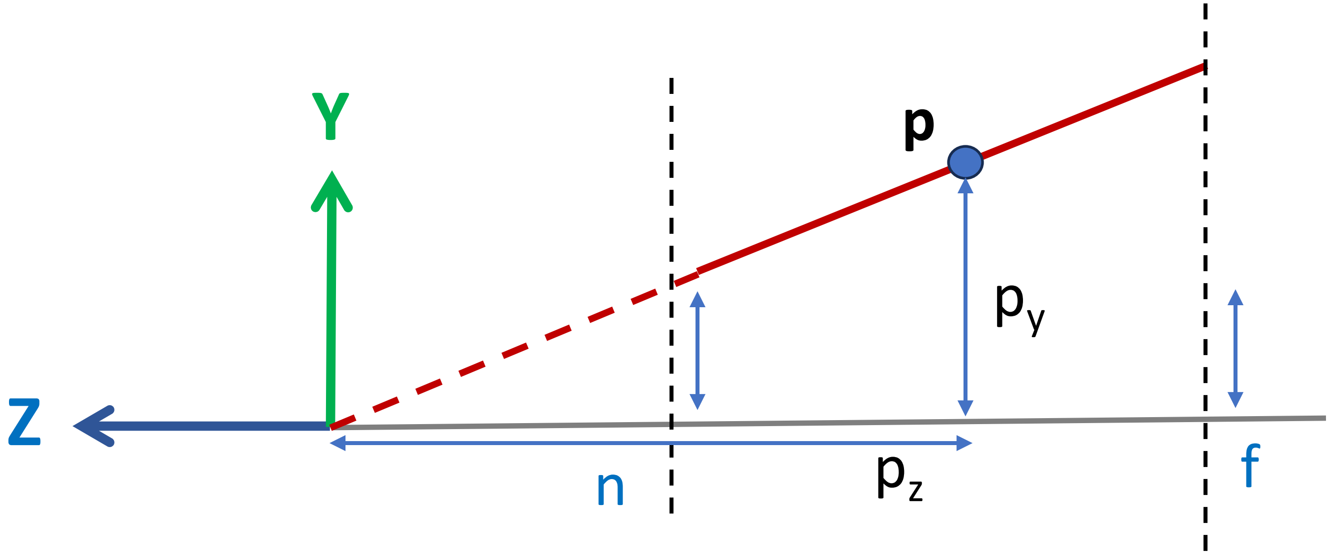
Perspective Transformation



Looking at the Y-Z plane

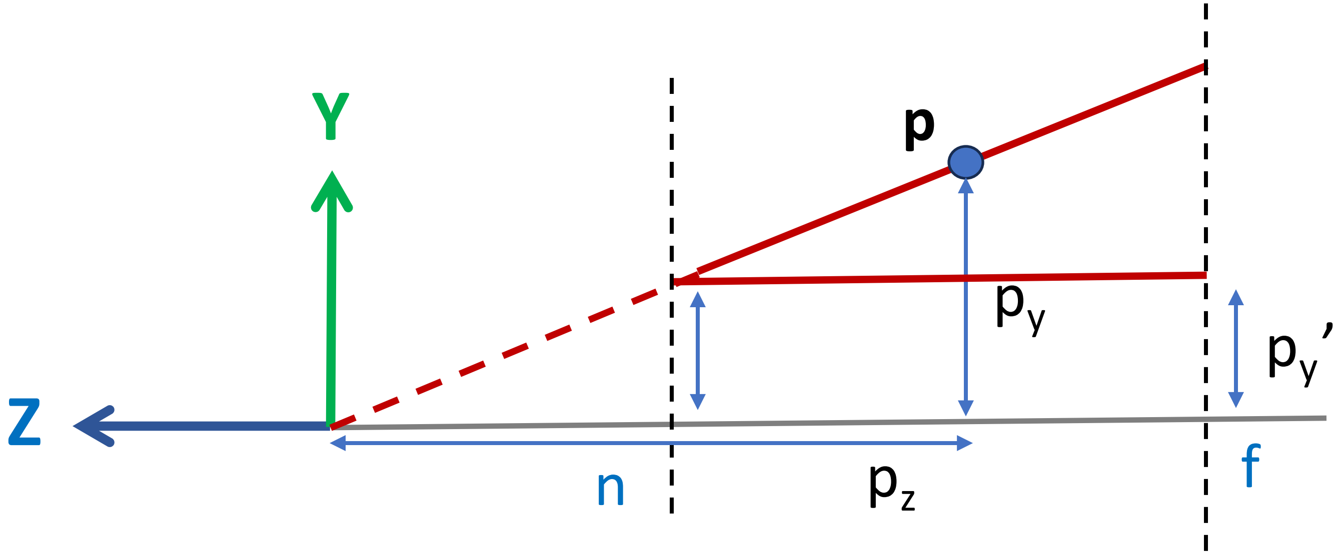
- We want to deform the slanted red line into the straight red line
  - Any point can be considered, p is just an example

Perspective Transformation



- For any arbitrary point p on the red line, **p_y/p_z is the same**

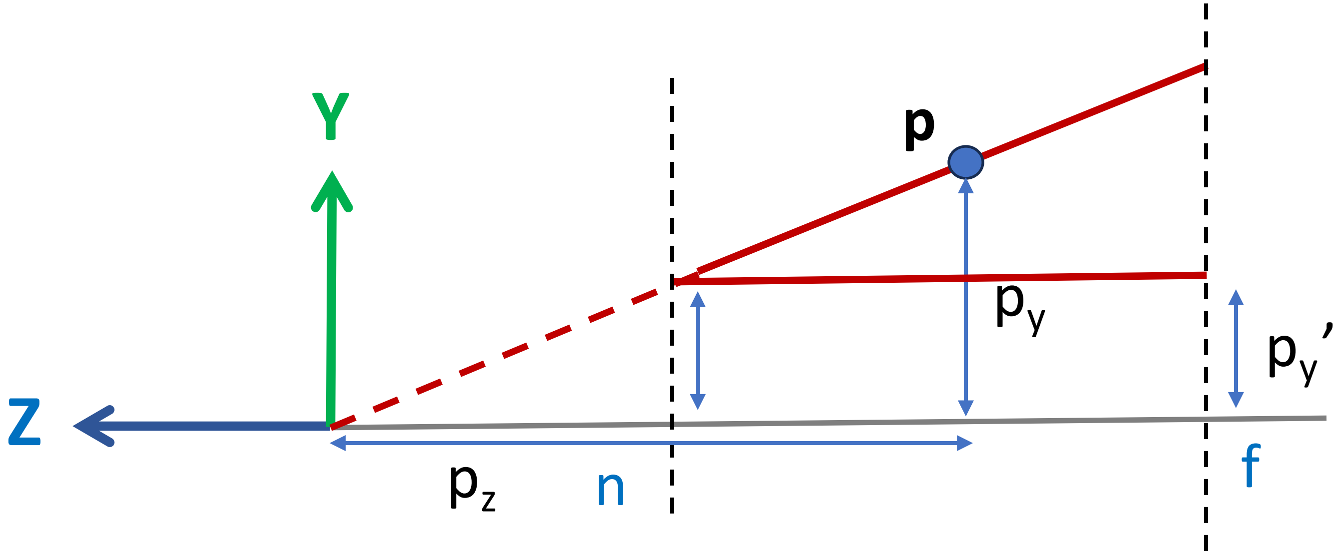
Perspective Transformation



- Let's say the length is $p_y' = (p_y/p_z)n$
- For the X axis, similarly we can write, $p_x' = (p_x/p_z)n$

Perspective Transformation

IITK CS360A



We can write the following:

$$\begin{bmatrix} p_x' \\ p_y' \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} \frac{n}{p_z}$$

- Observation: We are dividing by the z-coordinate
- This is hard to do using a matrix multiplication in this form

Revisit Homogeneous Coordinates

IITK CS360A

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \alpha p_x \\ \alpha p_y \\ \alpha p_z \\ \alpha \end{bmatrix}$$

We say that these two points are equivalent

- Observation: We are interested in 3D points and vectors
- But we are using 4D representations to achieve our goal
  - To make the math work out

So, here is what we want:

$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \begin{bmatrix} np_x/p_z \\ np_y/pz \\ ? \\ 1 \end{bmatrix} \equiv \begin{bmatrix} np_x \\ np_y \\ ? \\ p_z \end{bmatrix}$$

p_x and p_y will be divided by p_z

Perspective Transformation

IITK CS360A

$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \begin{bmatrix} np_x/p_z \\ np_y/pz \\ ? \\ 1 \end{bmatrix} \equiv \begin{bmatrix} np_x \\ np_y \\ ? \\ p_z \end{bmatrix}$$

- What we really want:
  - Keep the z values
  - Do not scale them significantly
  - Keep the points at near and far plan as it is

If we write in a matrix form, we get the following

$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & ? & ? \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Solution

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

Perspective Transformation

IITK CS360A

$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

If $p_z = n$, then $p'_z = \mathrm{n}$

If $p_z = f$, then $p'_z = \mathrm{f}$

$\Rightarrow$

- Implications
  - Z values do not get scaled significantly
  - Keeps the points at near and far plan as it is
  - The extent of the z values are preserved

Perspective Projection

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

Perspective Projection | Orthographic Projection | Perspective Transformation