CompSci 372 – Tutorial

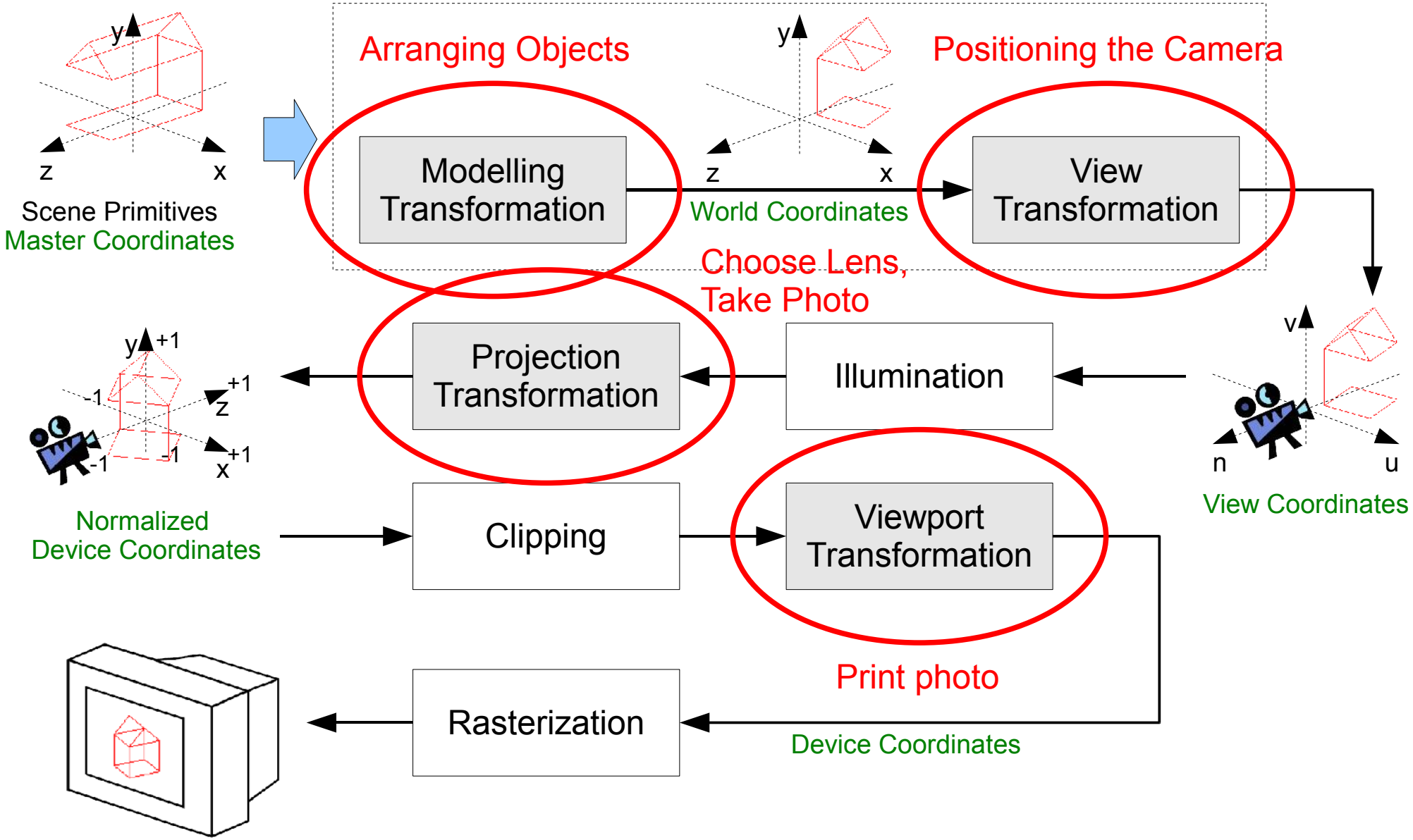
Part 7

Viewing

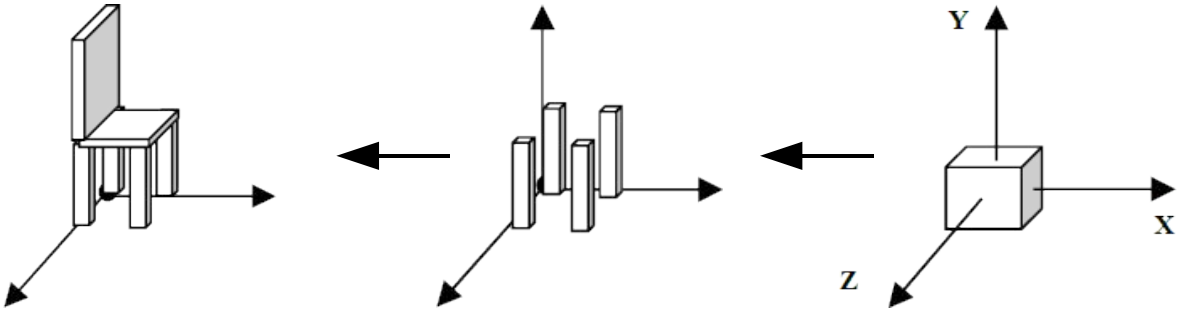
Rendering Steps

- Model Transformations
  - Arranging objects in a scene
- View Transformation
  - Positioning the camera
- Projection
  - Choosing a lens & taking a photo
- Viewport Transformation
  - Printing a photo

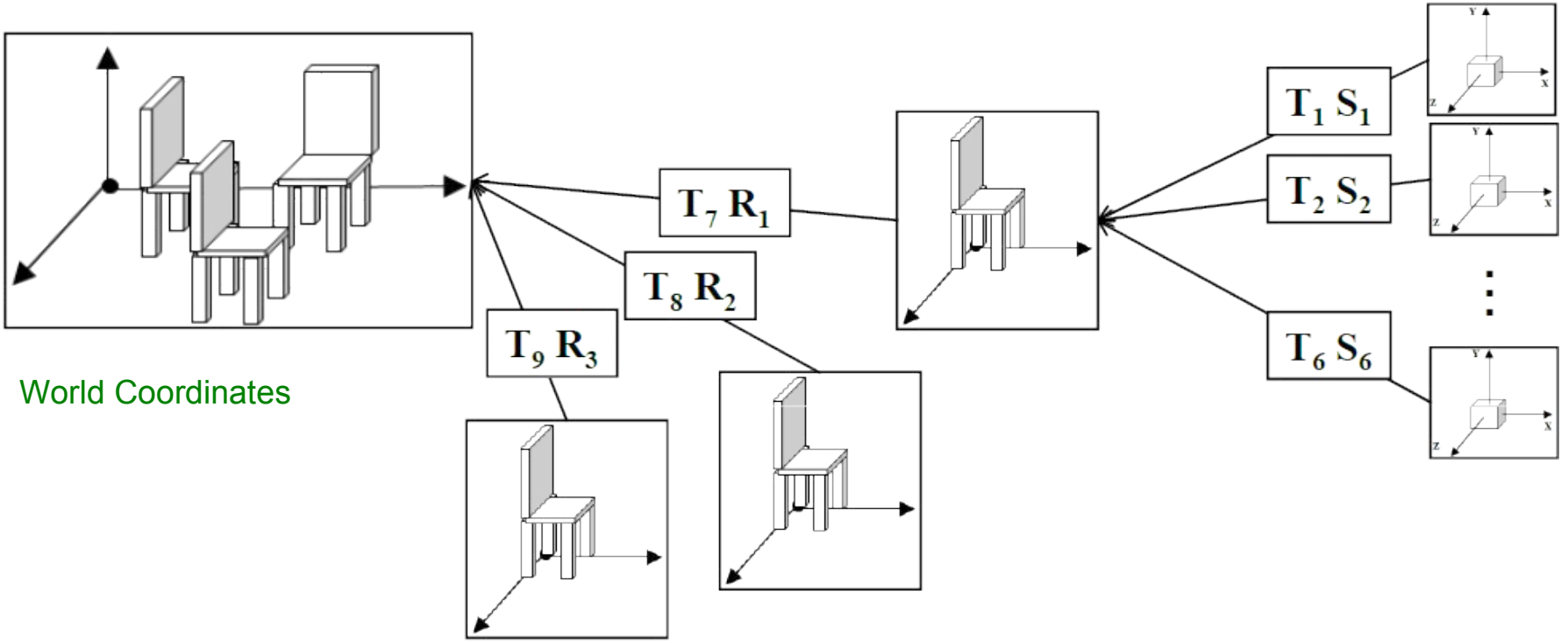
OpenGL Render Pipeline



Arrange Objects

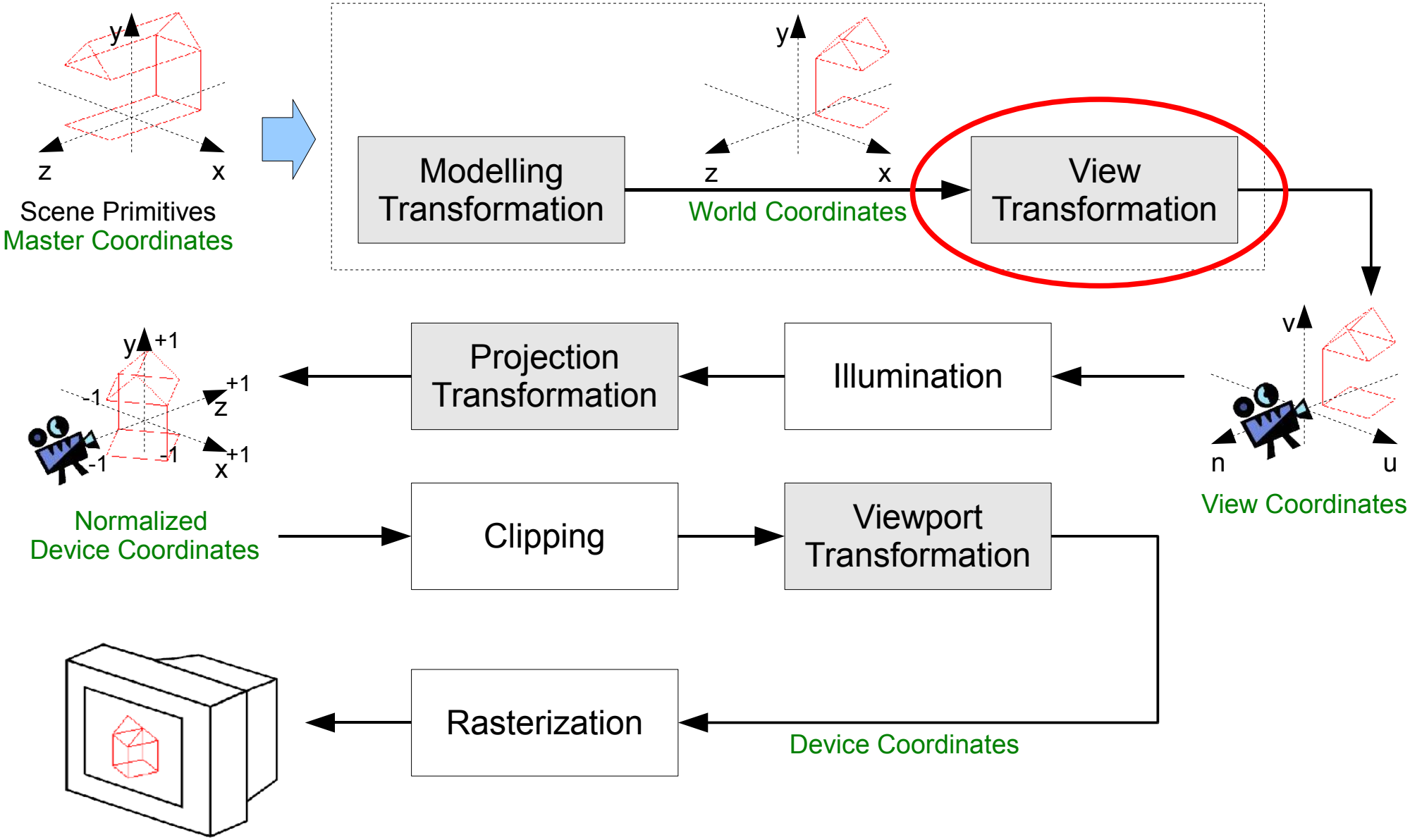


Master Coordinates



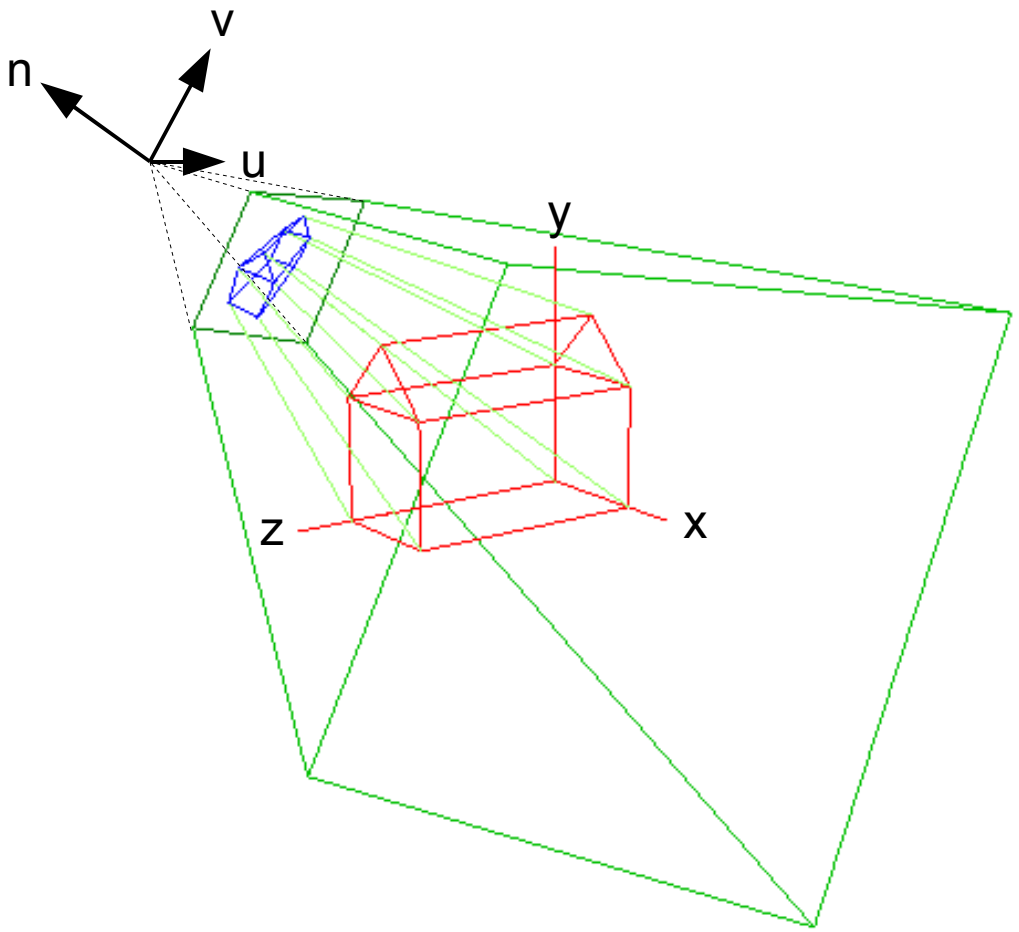
World Coordinates

OpenGL Render Pipeline



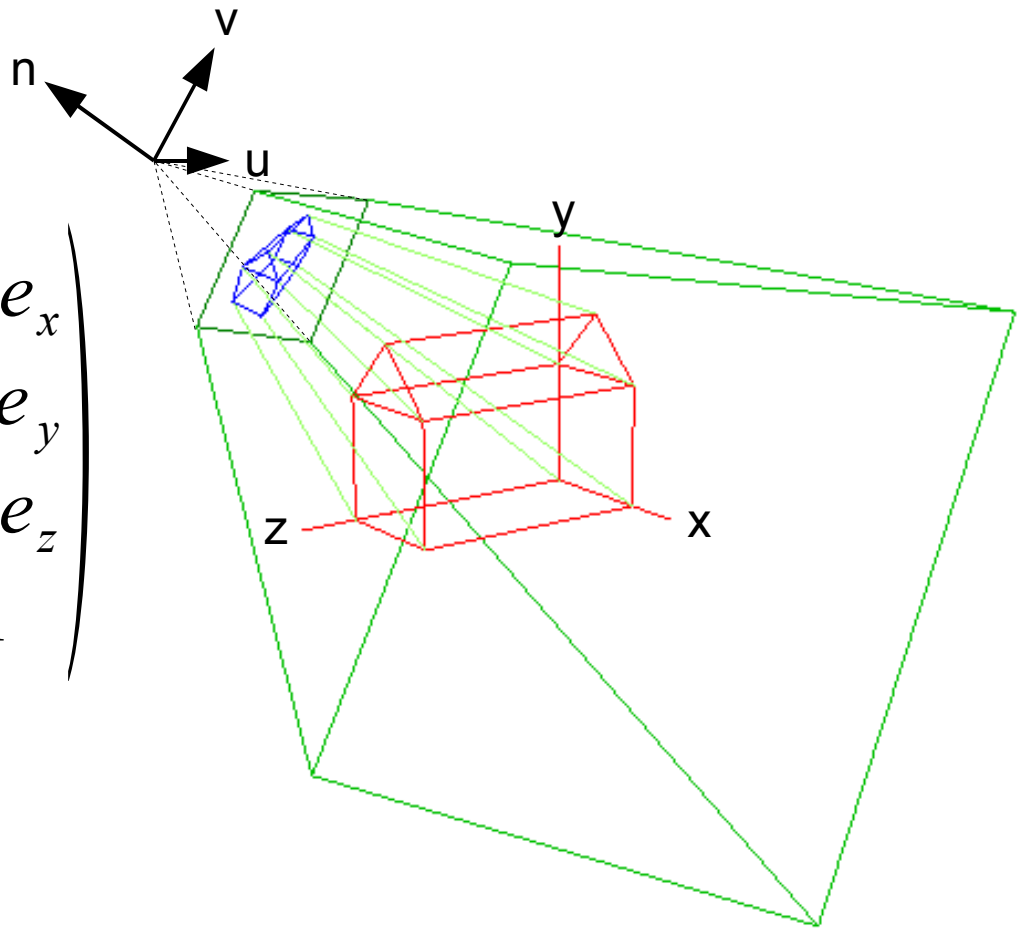
Position Camera

- Position
- Orientation
  - gluLookAt
  - Pitch/Yaw/Roll
  - Azimuth/Elevation
  - ...



Position Camera

$$\boldsymbol{M}_{Cam} = \begin{pmatrix} u_x & v_x & n_x & eye_x \\ u_y & v_y & n_y & eye_y \\ u_z & v_z & n_z & eye_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Position Camera

- Don't move the camera, move the scene! → Apply $\boldsymbol{M}_{\text{Cam}}^{-1}$

$$\boldsymbol{M}_{Cam} = \begin{pmatrix} u_x & v_x & n_x & eye_x \\ u_y & v_y & n_y & eye_y \\ u_z & v_z & n_z & eye_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & eye_x \\ 0 & 1 & 0 & eye_y \\ 0 & 0 & 1 & eye_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \boldsymbol{M}_T \boldsymbol{M}_R$$

Position Camera

- Don't move the camera, move the scene! → Apply $\boldsymbol{M}_{\text{Cam}}^{-1}$

$$\boldsymbol{M}_{Cam}^{-1} = \boldsymbol{M}_{R}^{-1}\,\boldsymbol{M}_{T}^{-1}$$

$$= \boldsymbol{M}_{R}^{T}\,\boldsymbol{M}_{T}^{-1}$$

Only, because $|\boldsymbol{u}|$, $|\boldsymbol{v}|$, $|\boldsymbol{n}|$ = 1 and orthogonal → Orthogonal Matrix

$$= \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Position Camera

- Don't move the camera, move the scene! → Apply M_{cam}^{-1}

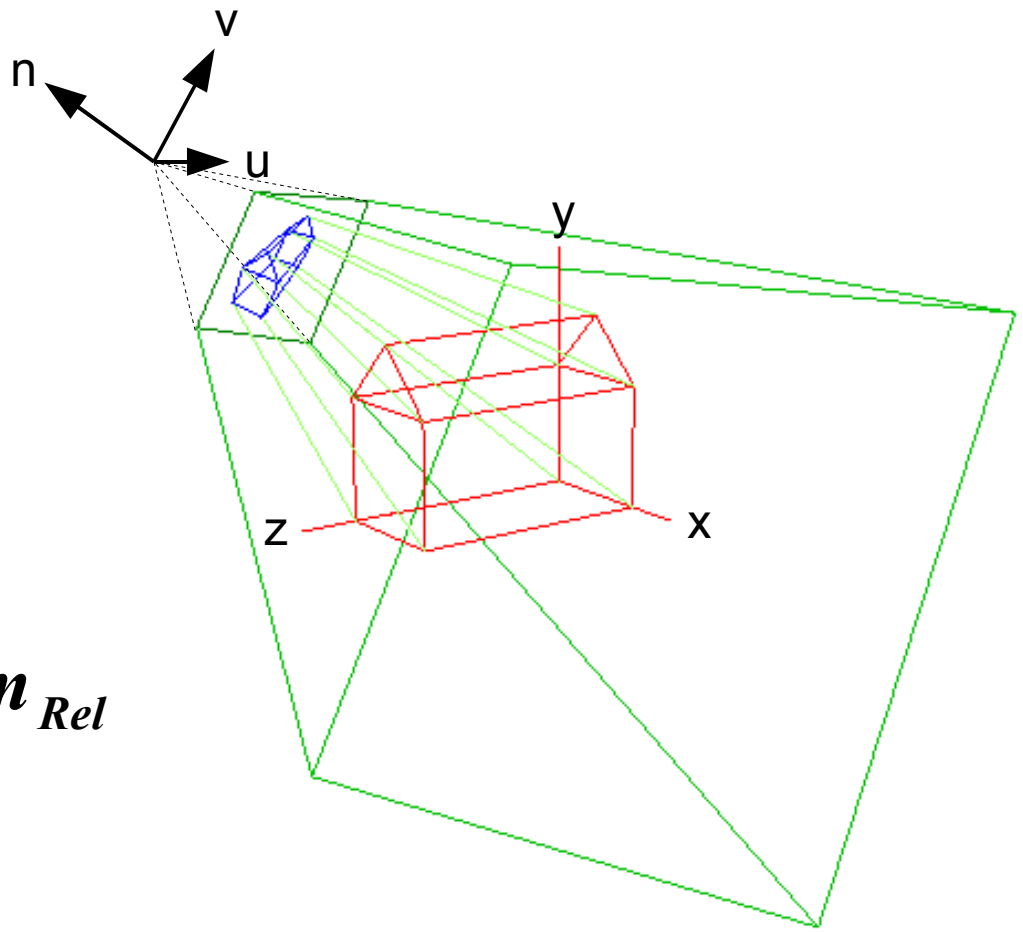
$$M_{Cam}^{-1} = \begin{pmatrix} u_x & u_y & u_z & -eye \cdot u \\ v_x & v_y & v_z & -eye \cdot v \\ n_x & n_y & n_z & -eye \cdot n \\ 0 & 0 & 0 & 1 \end{pmatrix} = V$$

- This is the last applied transformation → First lines in the program

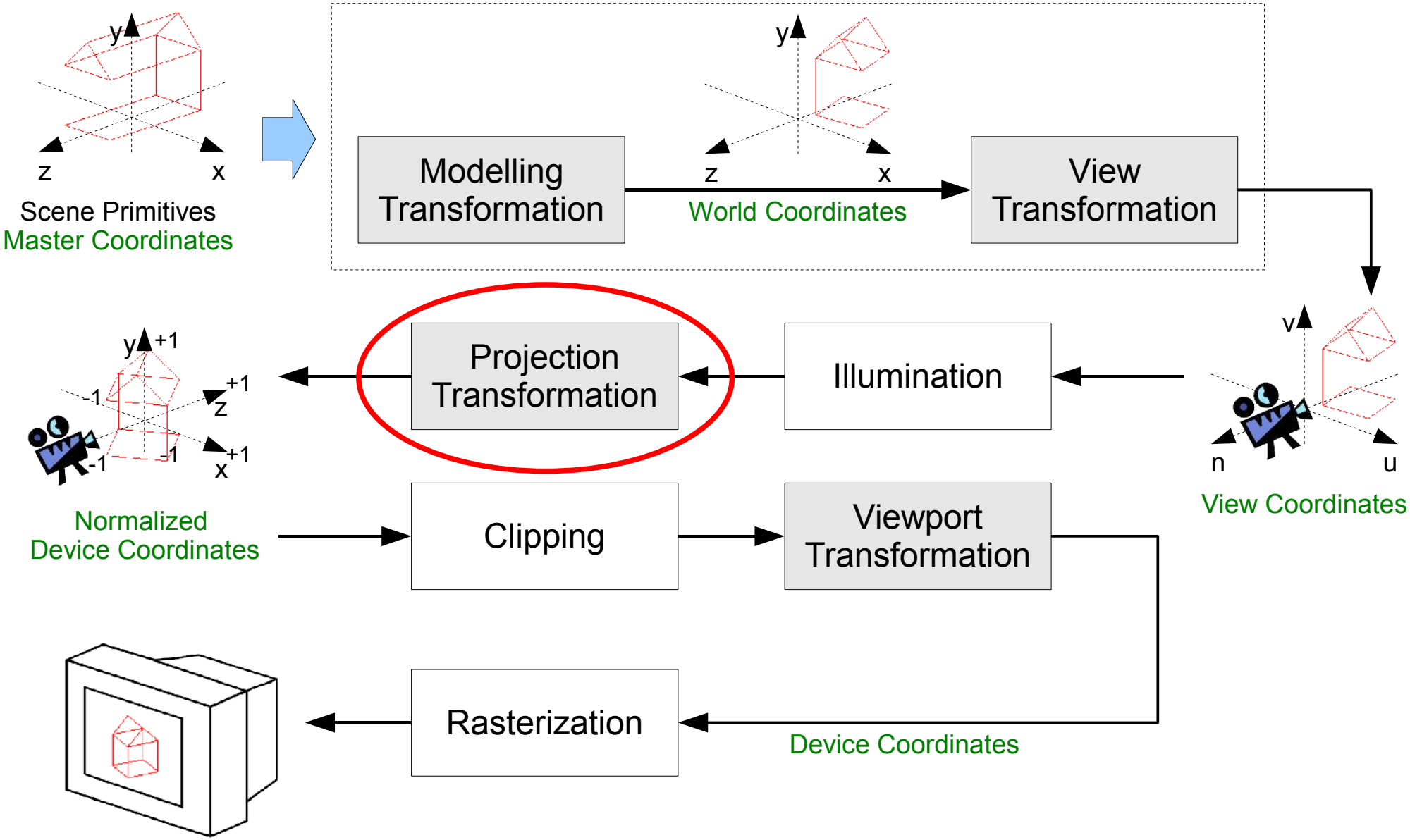
Move Camera

$$\boldsymbol{m}_{Abs} = \boldsymbol{M}_T \boldsymbol{m}_{Rel}$$

$$= \begin{pmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{m}_{Rel}$$

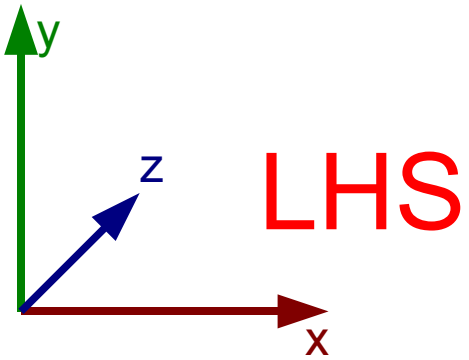


OpenGL Render Pipeline



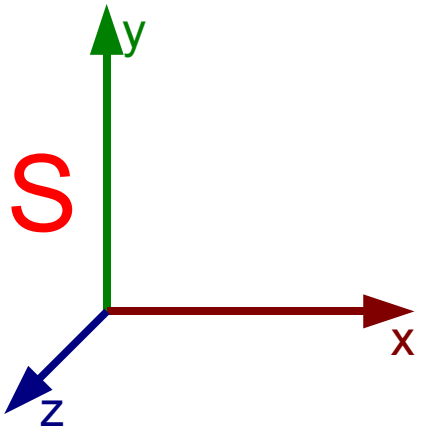
Projection

- "Flattening" the 3D scene onto the film
- Normalized Device Coordinates:
  - Example: 24x36mm film
- 3D Scene: Right Handed System
- NDC: Left Handed System



LHS

RHS

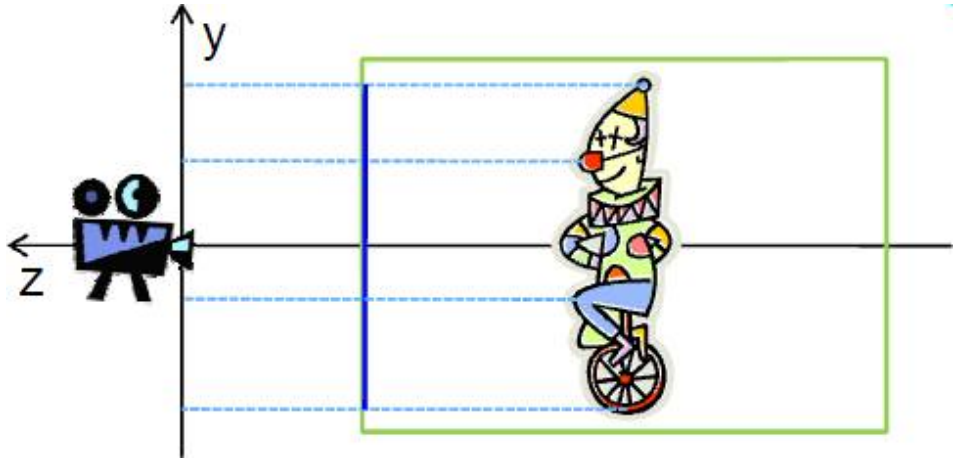


Projection

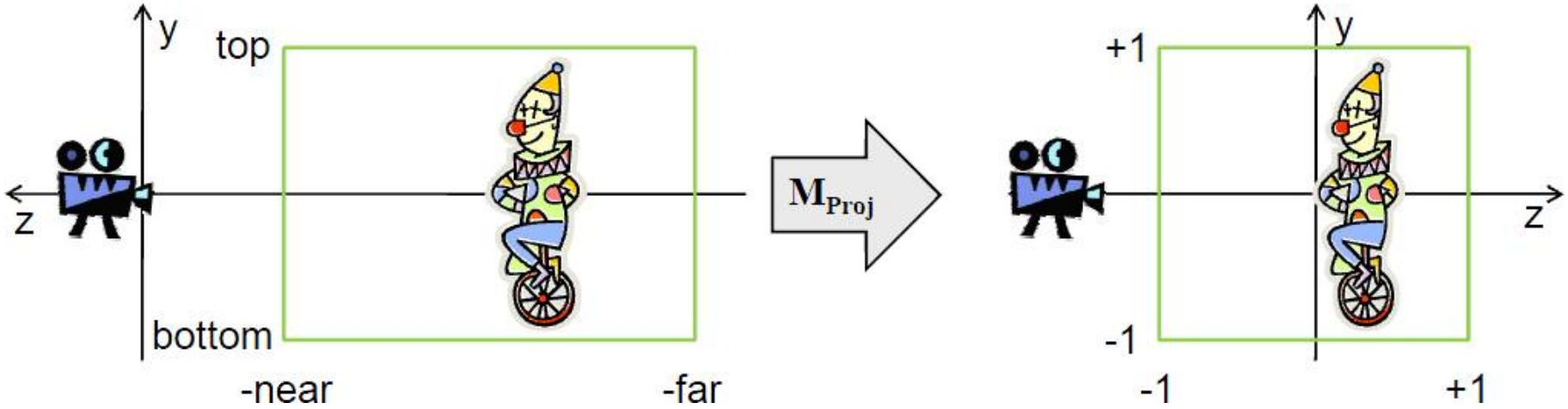
- Far/Near are positive values
  along the negative Z axis,
  which is the direction the camera is pointing at
- Top/Bottom / Left/Right / Far/Near are all relative to the eye point

Orthographic Projection

- Same size 3D → Same size 2D

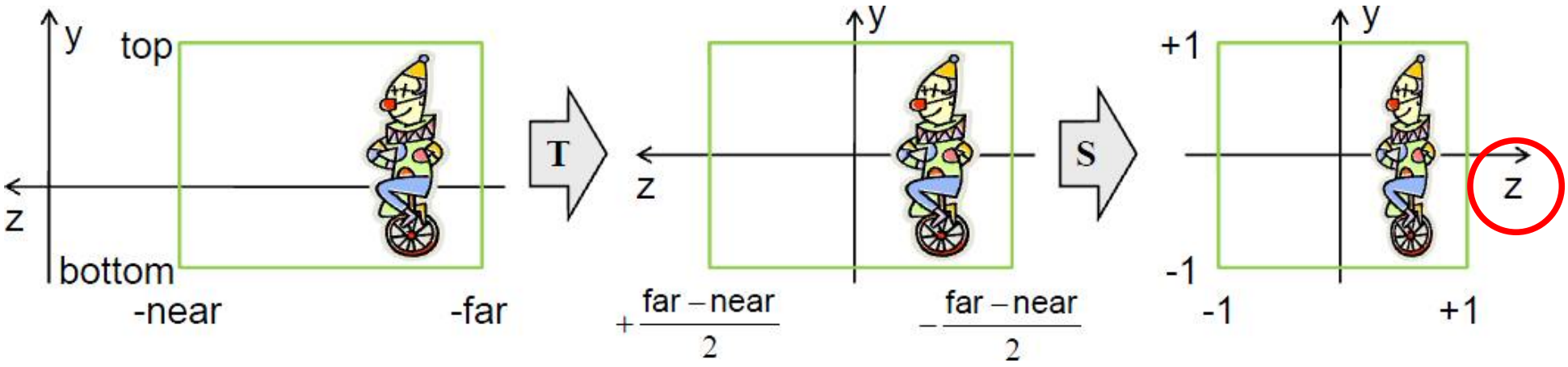


Orthographic Projection



- After the projection,
  simply use the x and y coordinates.

Orthographic Projection

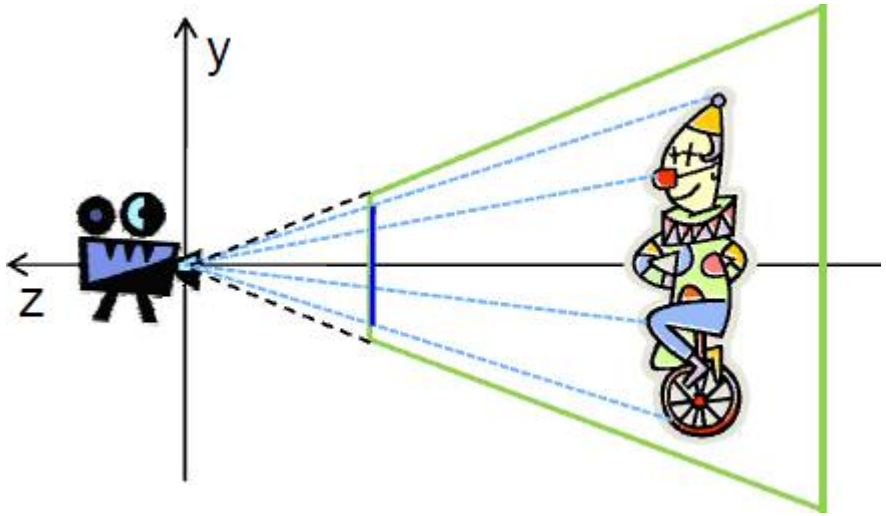


$$M_{Proj} = S\,T$$

$$= \begin{pmatrix} \frac{2}{right-left} & 0 & 0 & 0 \\ 0 & \frac{2}{top-bottom} & 0 & 0 \\ 0 & 0 & \frac{-2}{far-near} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & \frac{-(left+right)}{2} \\ 0 & 1 & 0 & \frac{-(top+bottom)}{2} \\ 0 & 0 & 1 & \frac{+(far+near)}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Perspective Projection

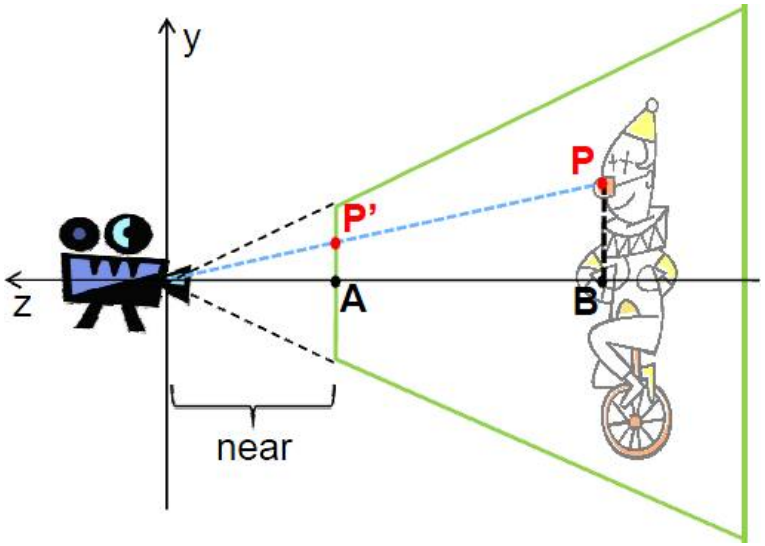
- Same size 3D → Not same size 2D, when distance is different



- Objects far away appear smaller

Perspective Projection

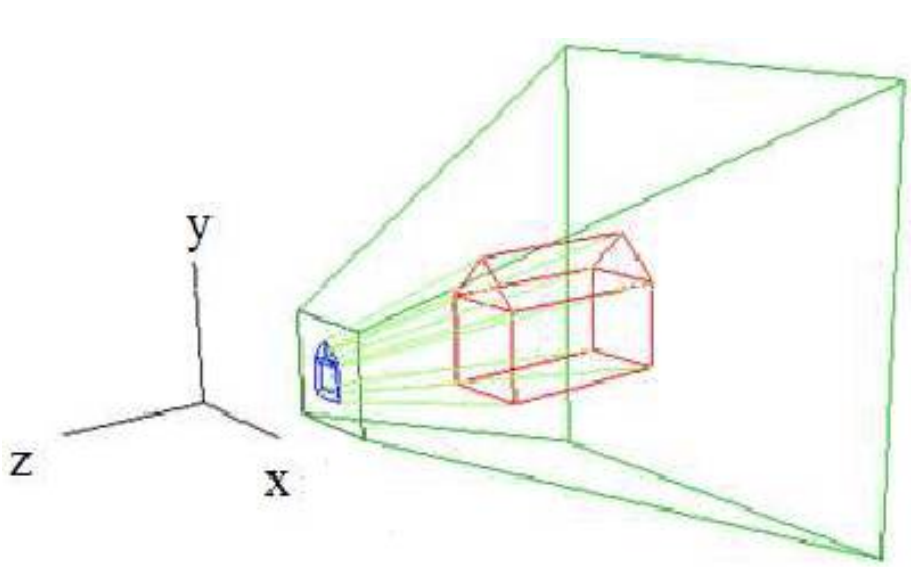
- $\frac{p'_{x/y}}{near} = \frac{p_{x/y}}{-p_z}$
- $p'_{x/y} = S_{persp}\, p_{x/y} \Leftrightarrow S_{persp} = \frac{near}{-p_z}$

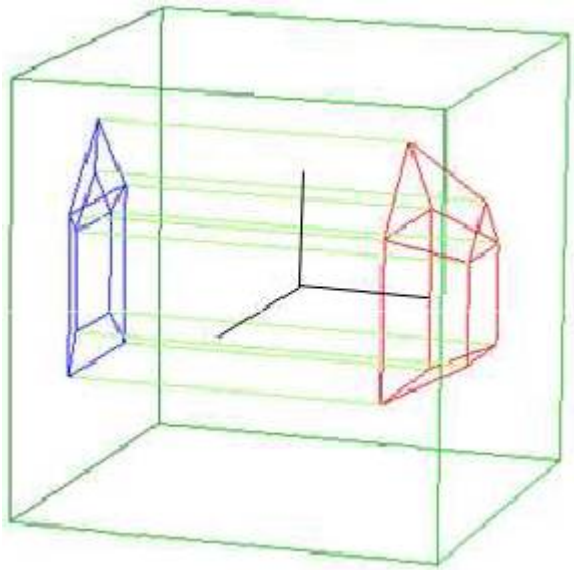


- Cannot scale Z:
  - Missing depth information for Rasterization stage (hidden surface removal)

Perspective Projection

- Solution: 2 Stages
  - Perspective Transformation
  - Orthographic Projection





Perspective Projection

- Perspective Matrix:
  - Scale x and y by according to S_{persp}
  - Maintain Z ordering and scale to -1....+1

$$\boldsymbol{p}' = \boldsymbol{M}_{persp}\,\boldsymbol{p} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} p_x \dfrac{near}{-p_z} \\ p_y \dfrac{near}{-p_z} \\ f(p_z) \\ 1 \end{pmatrix}$$

Perspective Projection

- Solution:
  - The magic of Homogeneous Coordinates

$$\boldsymbol{a} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \Rightarrow \begin{pmatrix} wx \\ wy \\ wz \\ w \end{pmatrix} \qquad \boldsymbol{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ w \end{pmatrix} \Rightarrow \begin{pmatrix} \frac{p_x}{w} \\ \frac{p_y}{w} \\ \frac{p_z}{w} \\ 1 \end{pmatrix}$$

Perspective Projection

- Solution:
  - The magic of Homogeneous Coordinates

$$\boldsymbol{p}' = \boldsymbol{M}_{persp}\,\boldsymbol{p} = \begin{pmatrix} near & 0 & 0 & 0 \\ 0 & near & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} near \cdot p_x \\ near \cdot p_y \\ a \cdot p_z + b \\ -p_z \end{pmatrix}$$

$$a = -\frac{far + near}{far - near}$$

$$b = -\frac{2 \cdot far \cdot near}{far - near}$$

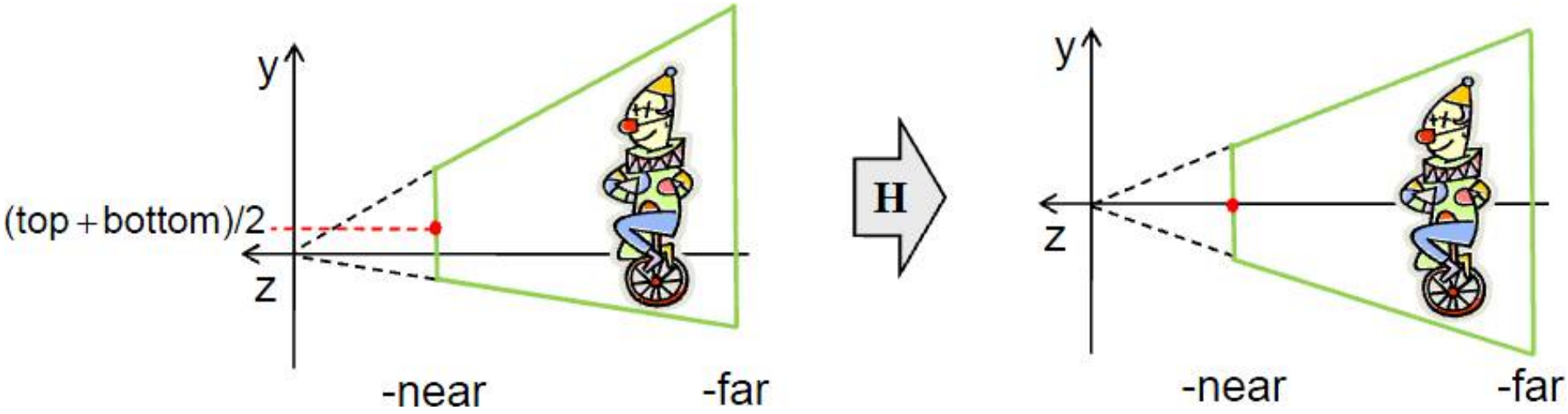
Perspective Projection

- Solution:
  - The magic of Homogeneous Coordinates

$$\boldsymbol{p}' = \begin{pmatrix} near \cdot p_x \\ near \cdot p_y \\ a \cdot p_z + b \\ -p_z \end{pmatrix} = \begin{pmatrix} p_x \dfrac{near}{-p_z} \\ p_y \dfrac{near}{-p_z} \\ \dfrac{a \cdot p_z + b}{-p_z} \\ 1 \end{pmatrix}$$

Perspective Projection

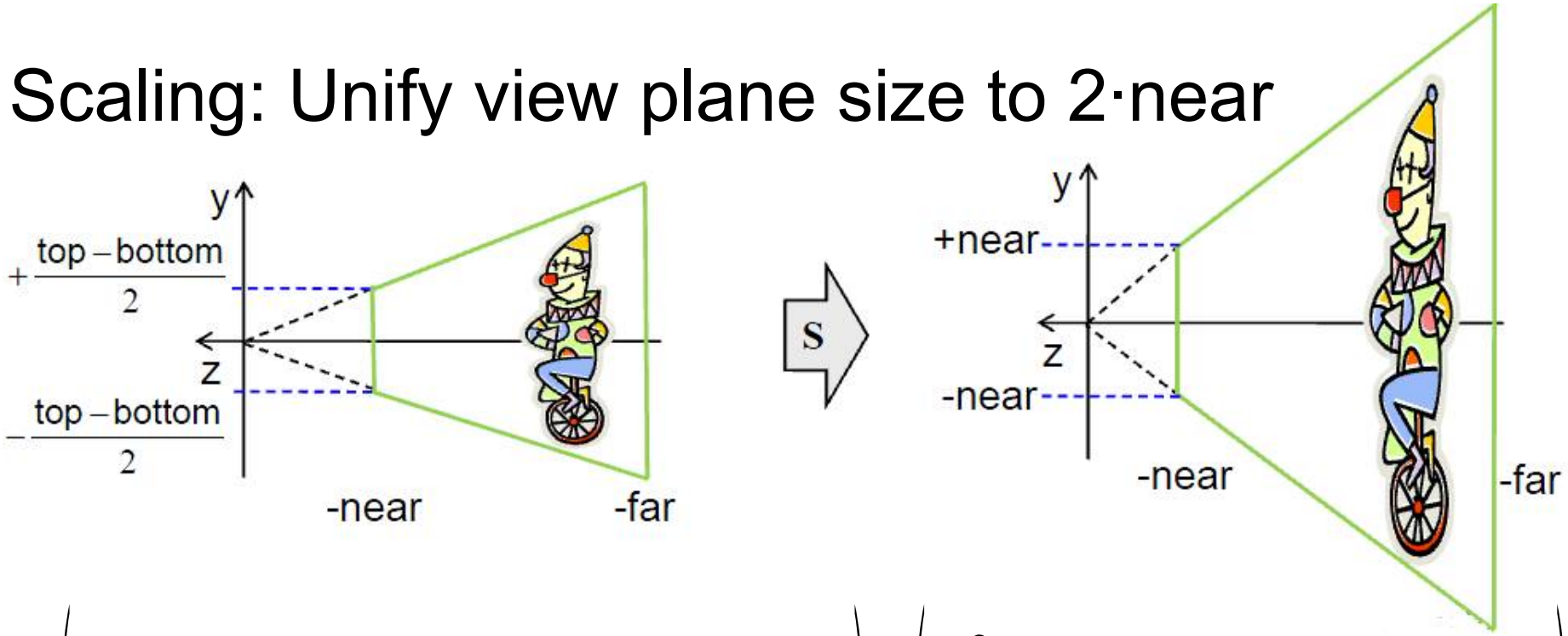
- Shearing: Get Z-Axis into middle of frustum



$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & \dfrac{-(left+right)/2}{-near} & 0 \\ 0 & 1 & \dfrac{-(top+bottom)/2}{-near} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dfrac{left+right}{2 \cdot near} & 0 \\ 0 & 1 & \dfrac{top+bottom}{2 \cdot near} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Perspective Projection

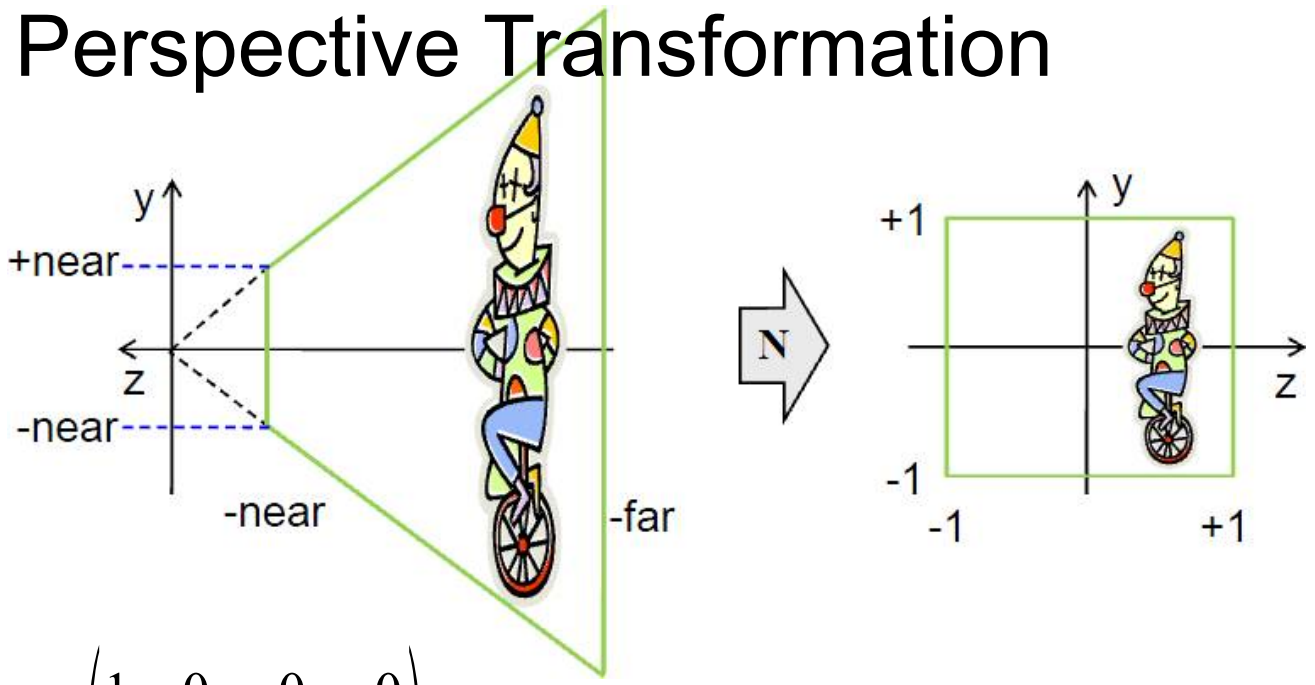
- Scaling: Unify view plane size to 2·near



$$\boldsymbol{S} = \begin{pmatrix} \frac{near}{(right-left)/2} & 0 & 0 & 0 \\ 0 & \frac{near}{(top-bottom)/2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{2 \cdot near}{right-left} & 0 & 0 & 0 \\ 0 & \frac{2 \cdot near}{top-bottom} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Perspective Projection

- Perspective Transformation



$$N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix}, \quad a = -\frac{far + near}{far - near}, \quad b = -\frac{2 \cdot far \cdot near}{far - near}$$

Perspective Projection

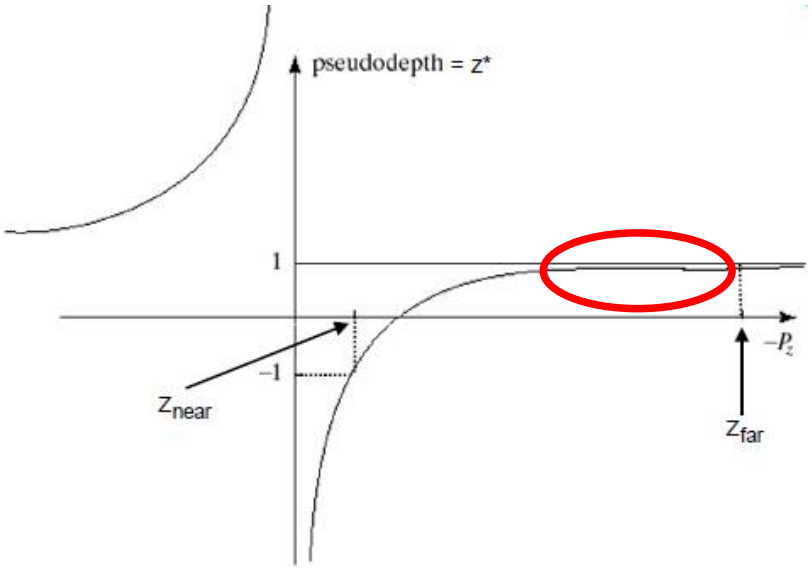
- Put it all together:

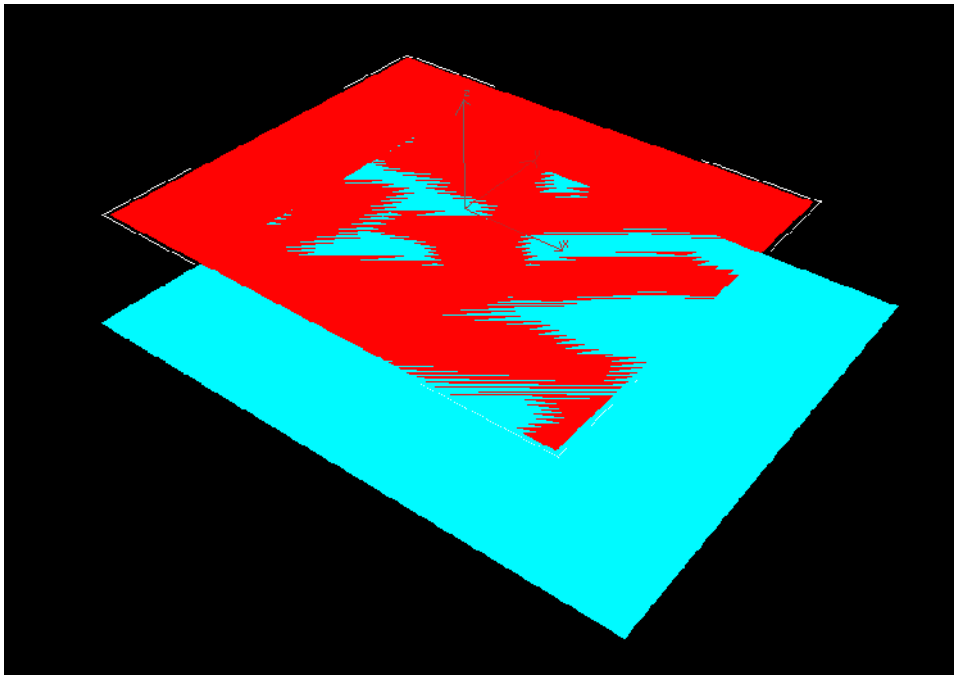
$$\boldsymbol{M}_{persp} = \boldsymbol{N}\,\boldsymbol{S}\,\boldsymbol{H}$$

Perspective Projection

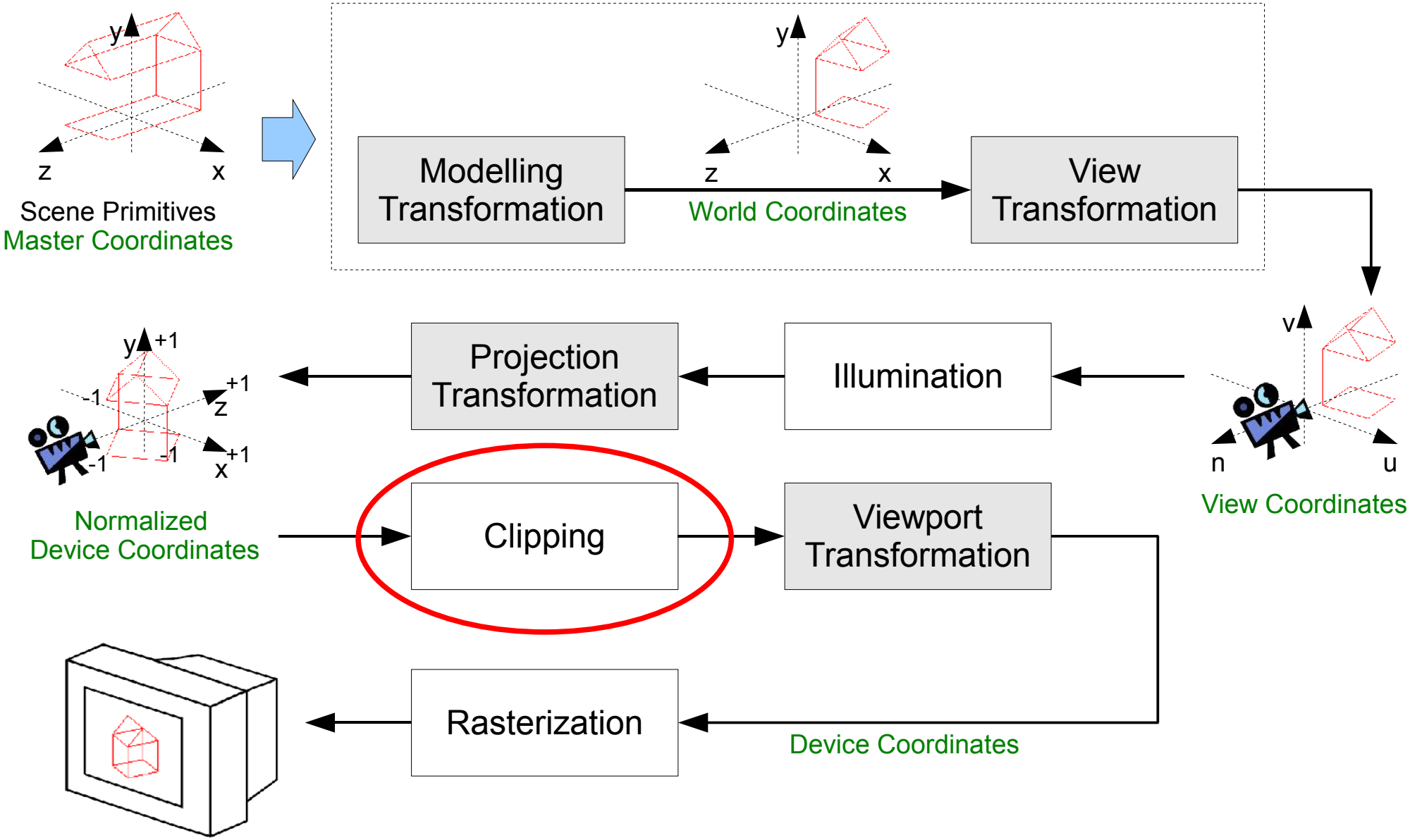
- Use of *near* and *far*:
  - Large range → Loss of precision → Artefacts

$$z^* = \frac{(far + near)\cdot z + 2\cdot far\cdot near}{(far - near)\cdot z}$$



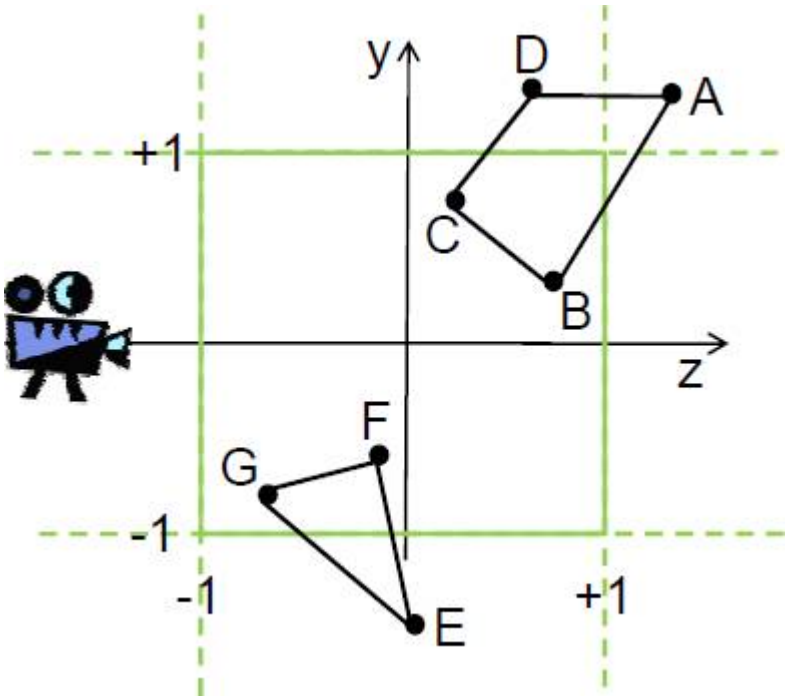


OpenGL Render Pipeline



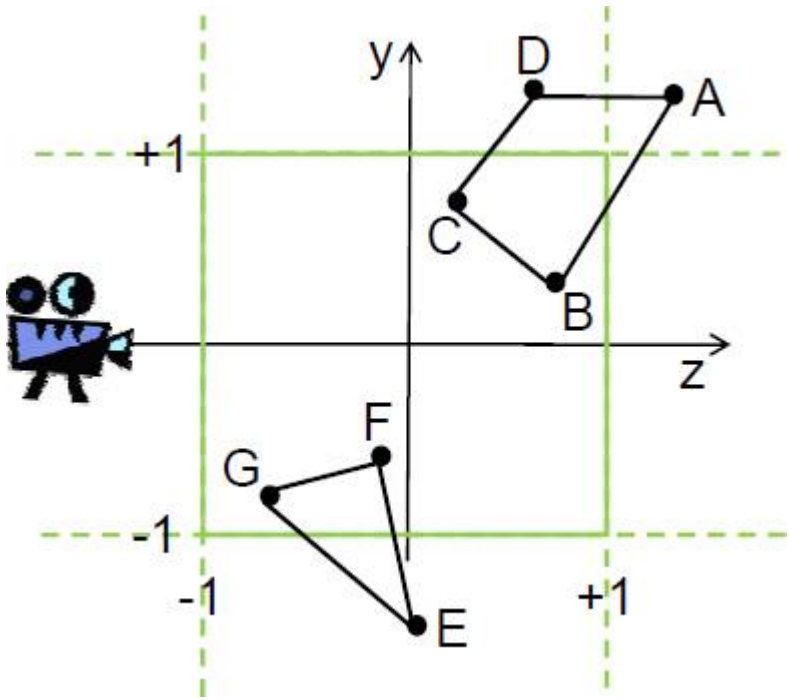
Clipping

- Trivial cases
  - Both points inside the view volume (CB, GF)
  - Both point outside the view volume (DA)
- Nontrivial case
  - One point inside, the other outside (GE, FE, AB, CD)



Clipping - Trivial

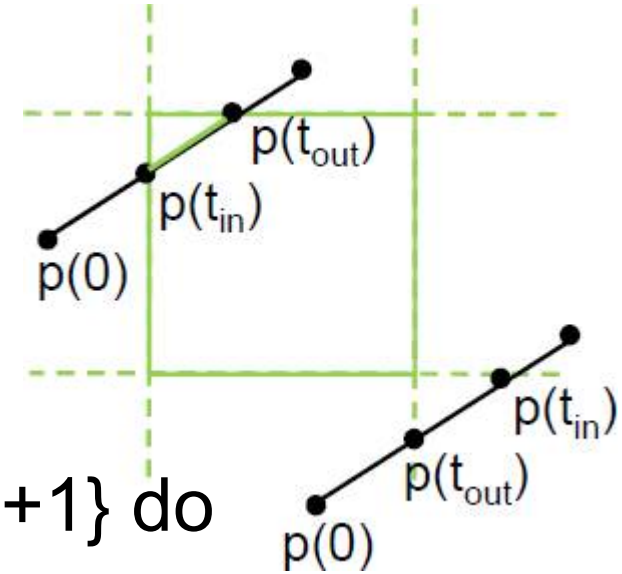
- Outcode table
  - outcode P1 & outcode P2 != 0 → outside
  - outcode P1 | outcode P2 == 0 → inside



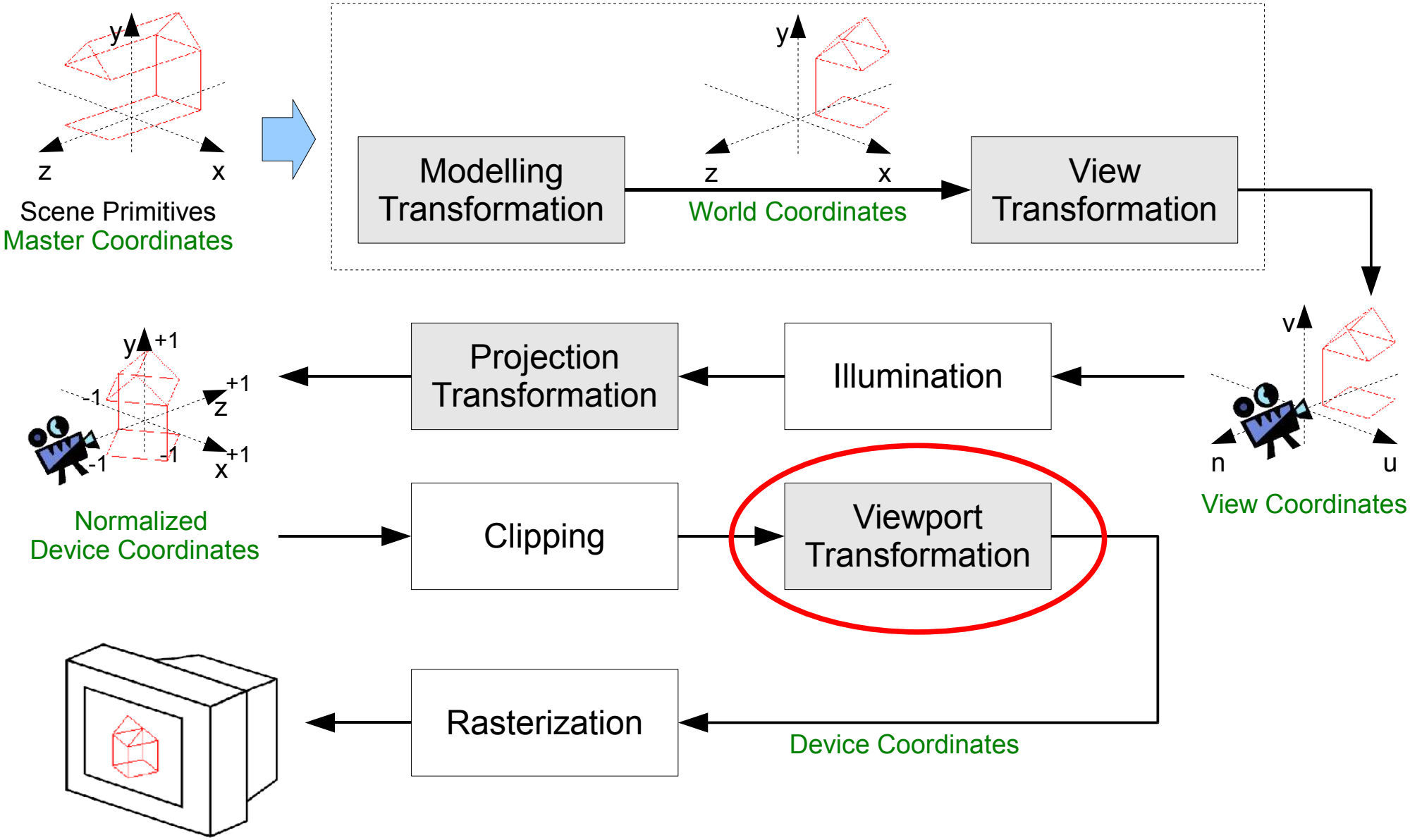
	L	R	B	T	N	F
A	0	0	0	**1**	0	**1**
B	0	0	0	0	0	0
C	0	0	0	0	0	0
D	0	0	0	**1**	0	0
E	0	0	**1**	0	0	0
F	0	0	0	0	0	0
G	0	0	0	0	0	0

Clipping – Non Trivial

- Liang-Barsky Clipping Algorithm
  1. Trivial tests first
  2. $t_{in}=0$, $t_{out}=1$
  3. For all halfspaces {x/y/z>– 1, x/y/z<+1} do
     1. Compute t where (extended) line crosses halfspace
     2. If entering half-space: $t_{in} = \max(t_{in}, t)$
        else: $t_{out} = \min(t_{out}, t)$
     3. Stop if $t_{in} > t_{out}$
  4. $t_{in} > t_{out}$: line is outside viewing volume
     else: line is inside VV from $p(t_{in})$ to $p(t_{out})$

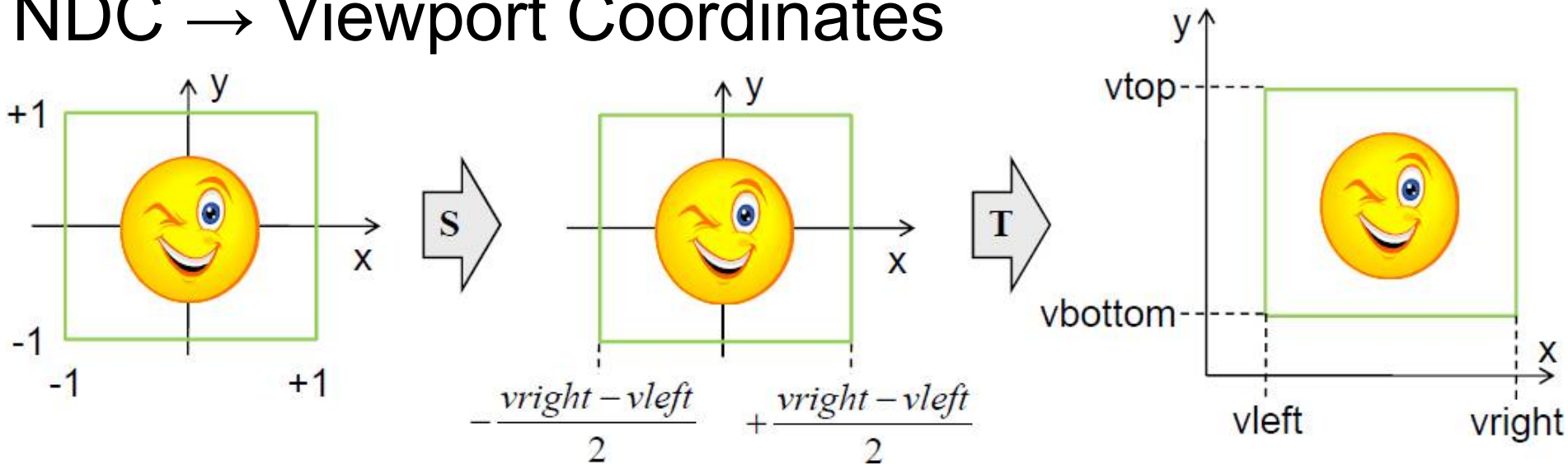


OpenGL Render Pipeline



Viewport Transformation

- NDC → Viewport Coordinates



$$M_{Viewport} = T\,S = \begin{pmatrix} 1 & 0 & 0 & \frac{v_{Right}+v_{Left}}{2} \\ 0 & 1 & 0 & \frac{v_{Top}+v_{Bottom}}{2} \\ 0 & 0 & 1 & \frac{max_z}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{v_{Right}-v_{Left}}{2} & 0 & 0 & 0 \\ 0 & \frac{v_{Top}-v_{Bottom}}{2} & 0 & 0 \\ 0 & 0 & 1 & \frac{max_z}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Viewport Transformation

- Take care of aspect ratio and window size

```
void reshape(int width, int height)
{
    // Set the viewport to be the entire window
    glViewport(0, 0, width, height);

    // Calculate new viewport parameters
    ...
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho(left, right, bottom, top, near, far);
}
```

```
int main(int argc, char** argv)
{
    ...
    glutReshapeFunc(reshape); // reshape function
    ...
}
```

Viewport Transformation

- Multiple Viewports
  - Repeat **`numViewport`** times
    - Setup viewport
    - Setup projection type/parameters
    - Setup camera
    - Draw scene