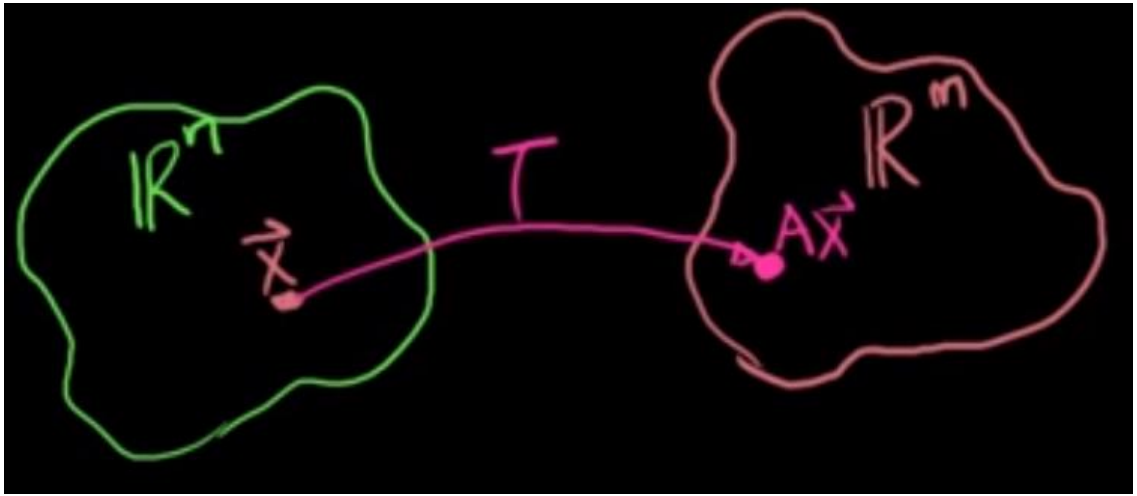
GRK 2

Dr W Palubicki

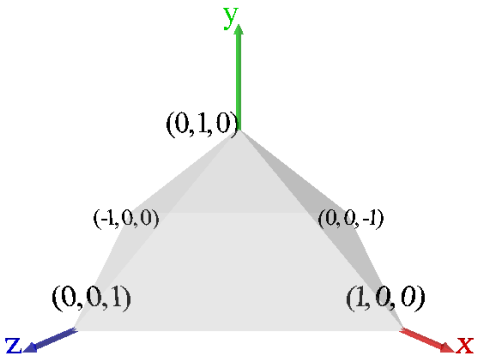
Matrix vector product as a transformation

$T: \mathbb{R}^n \to \mathbb{R}^m$

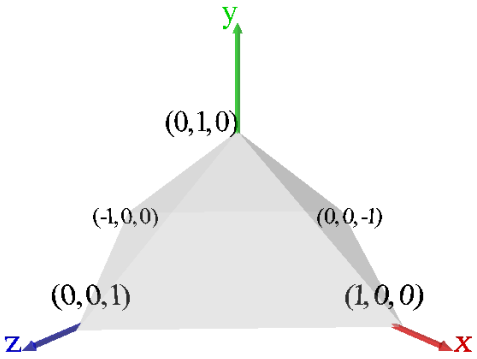
$T(\vec{x}) = A\vec{x}$



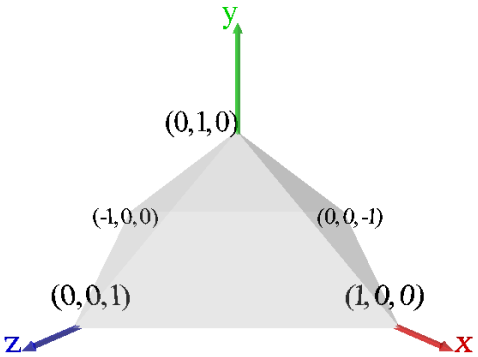
Fundamental Transformations



Scaling



Rotation



Translation

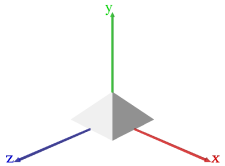
Simplified Rendering Pipeline

Model Matrix →

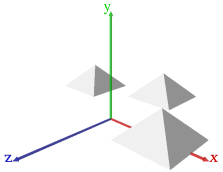
View Matrix →

Projection Matrix →

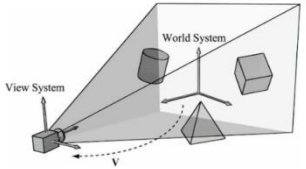
Viewport Transform →



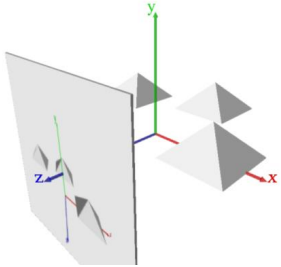
Object Space



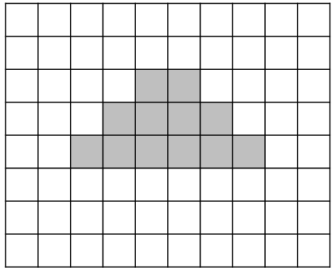
World Space



View Space

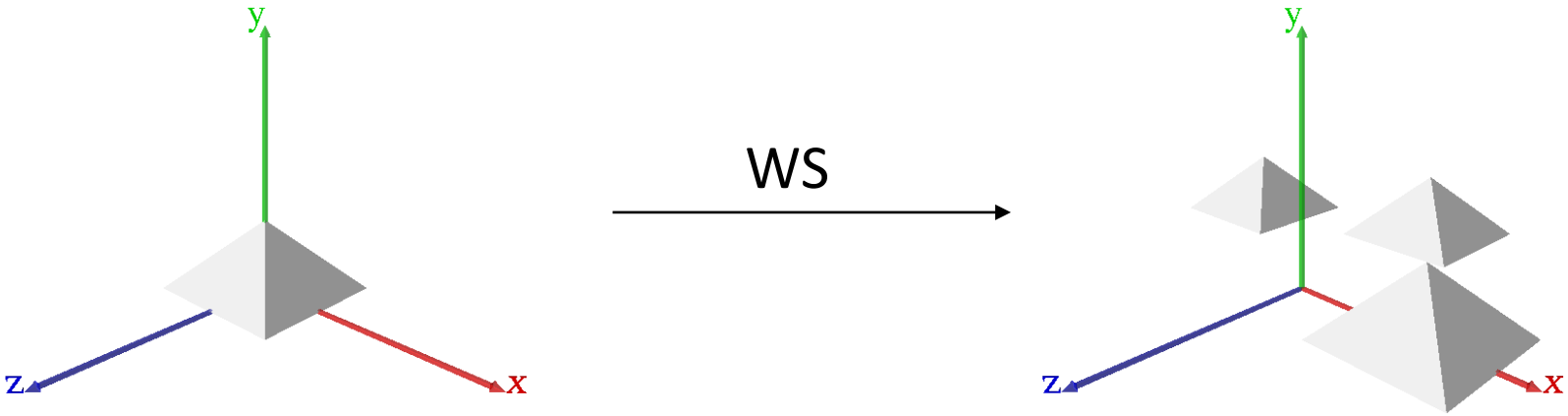


Clip Space



Screen Space

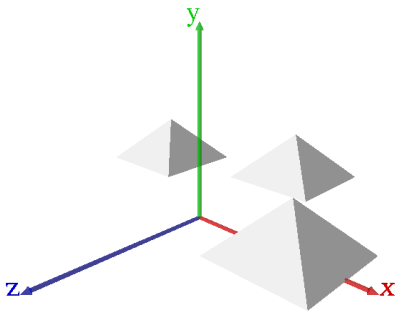
World space transformation

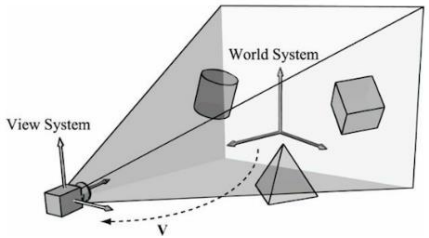


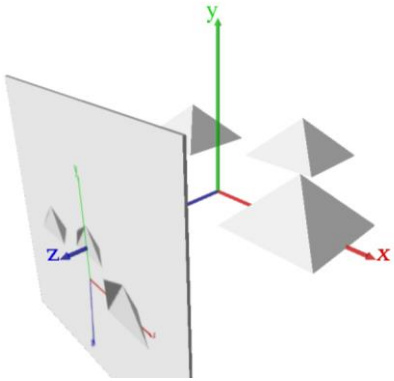
Object Space

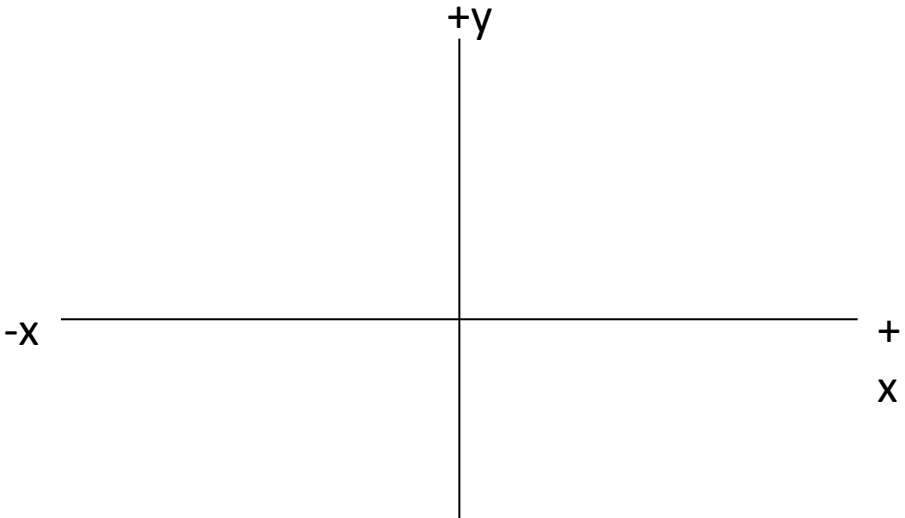
World Space

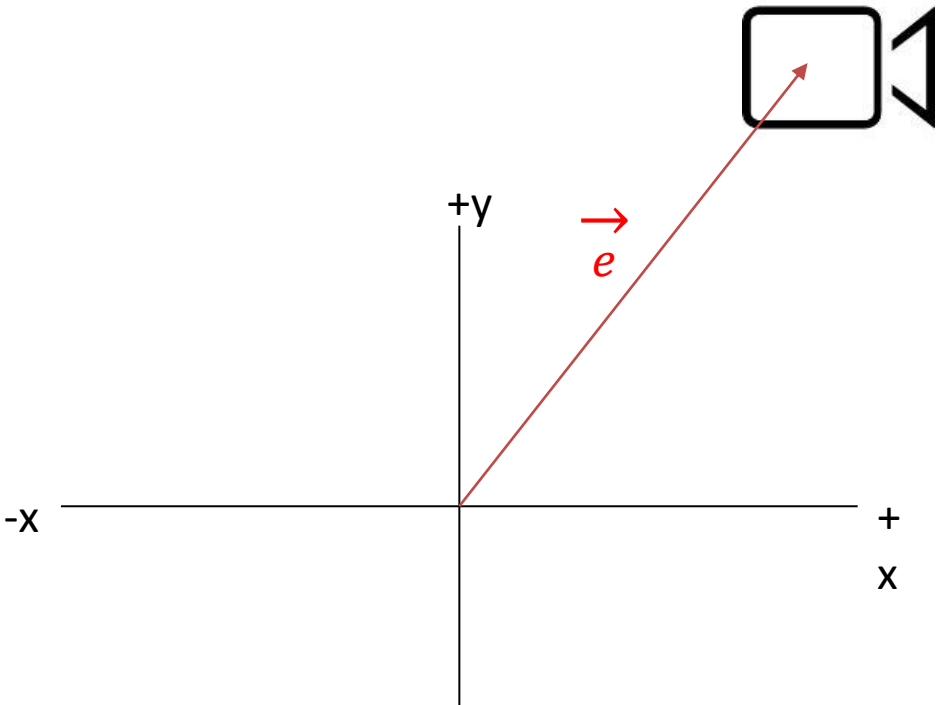
World space → 2D

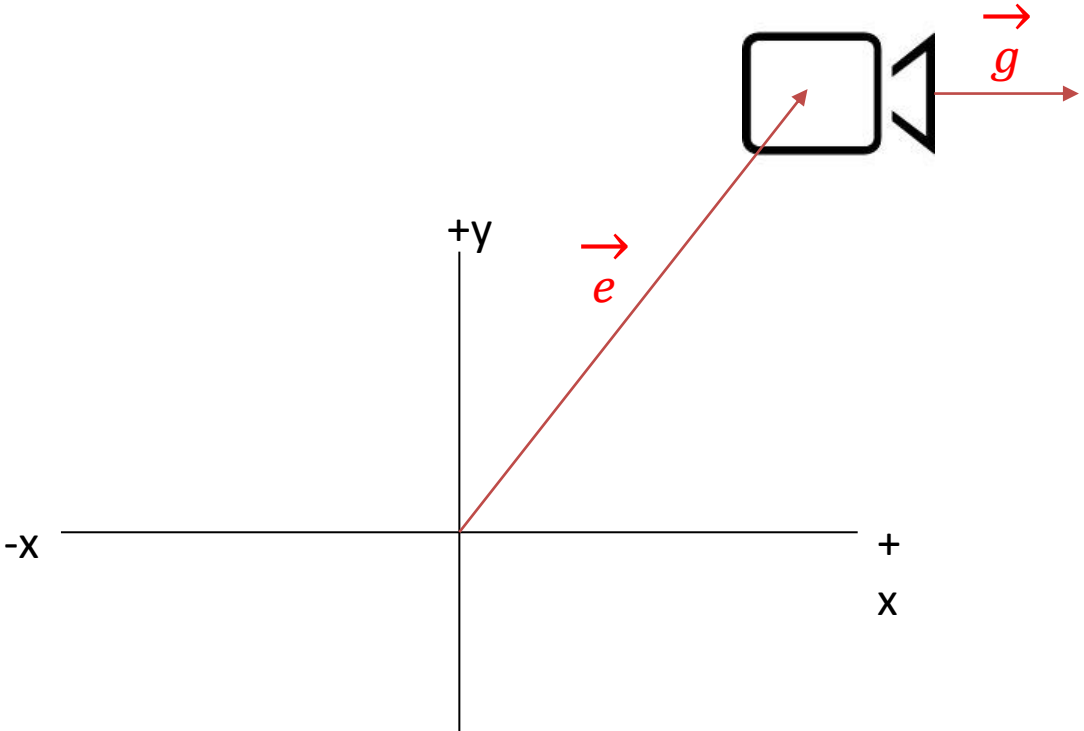


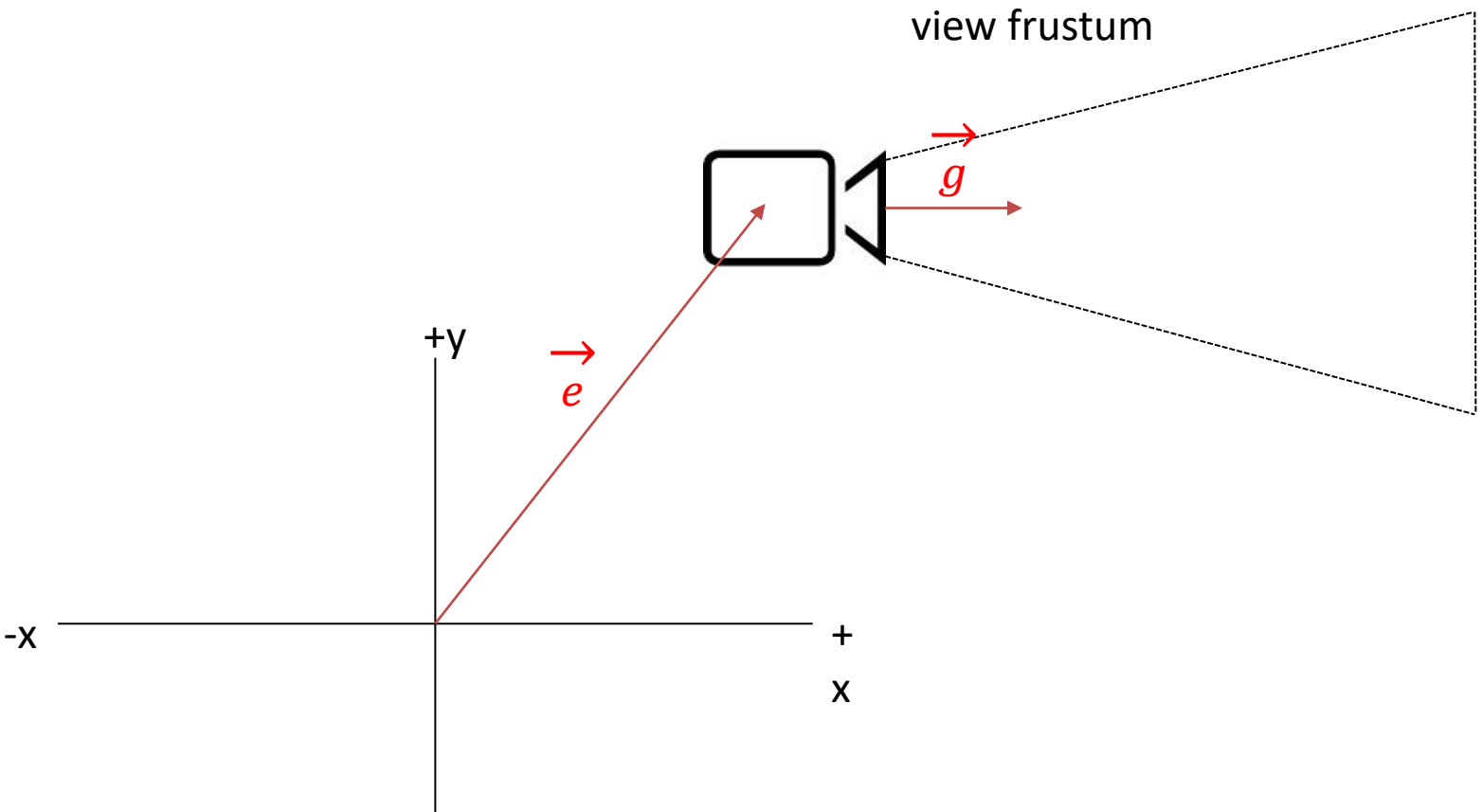


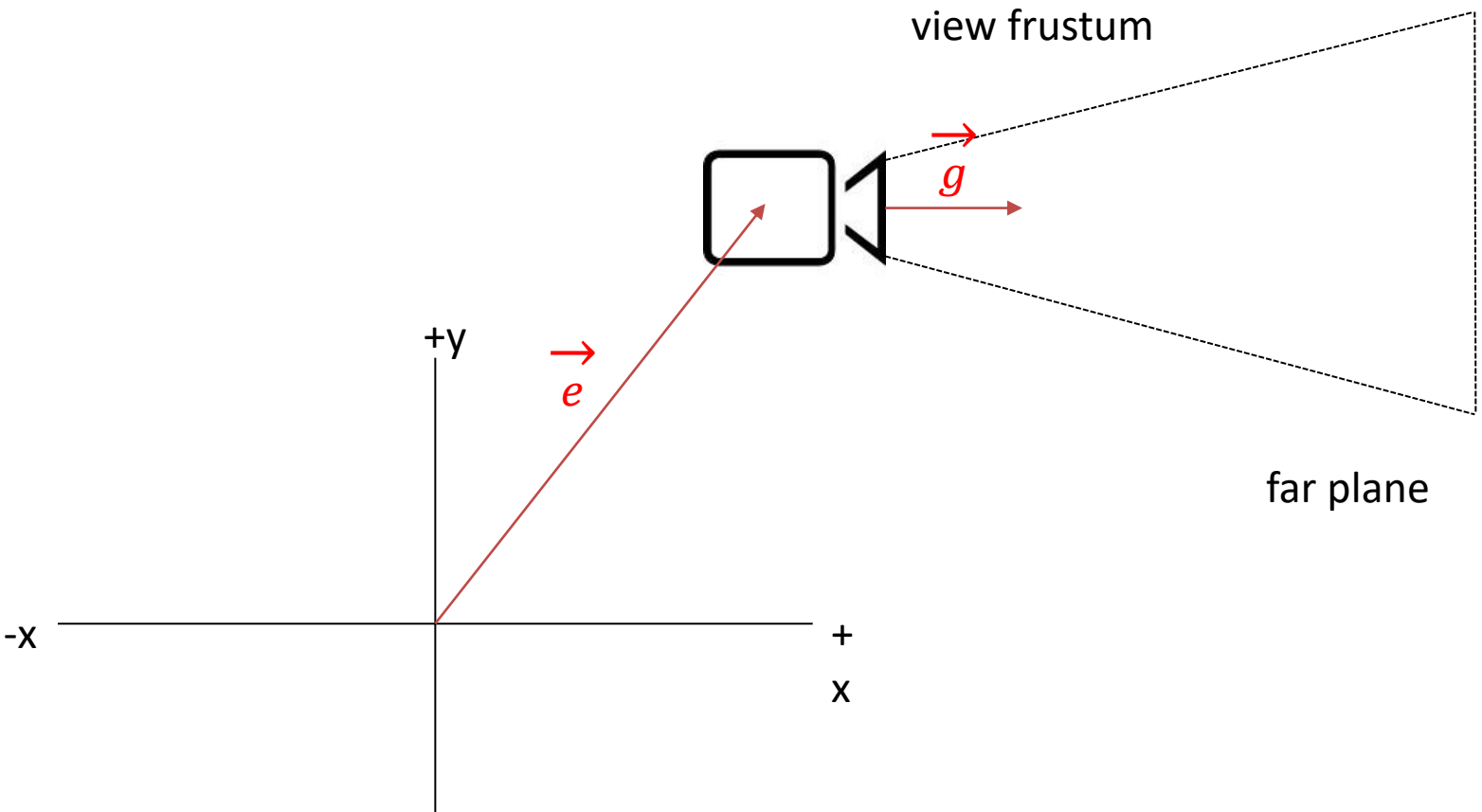


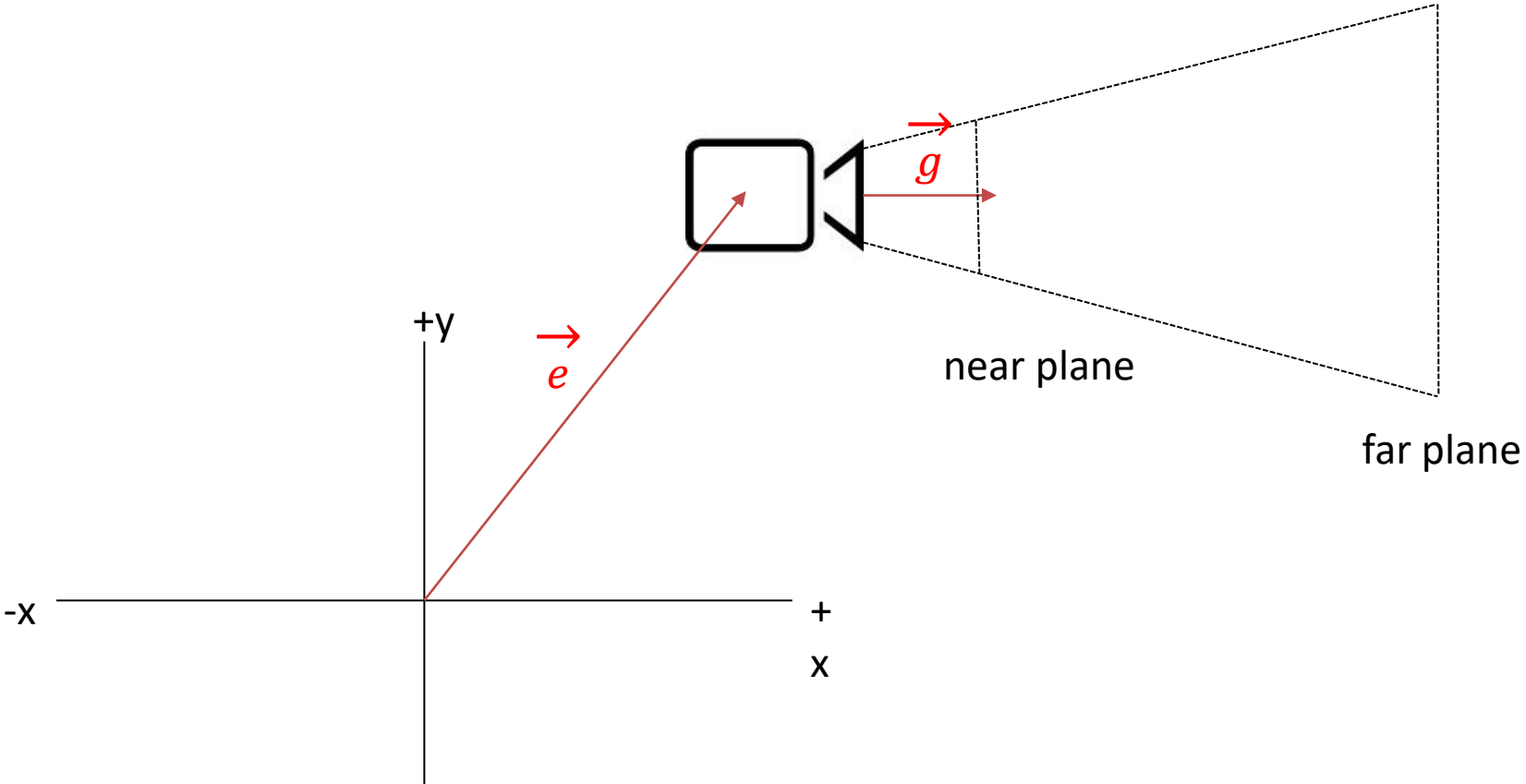

+y
-x
+
x

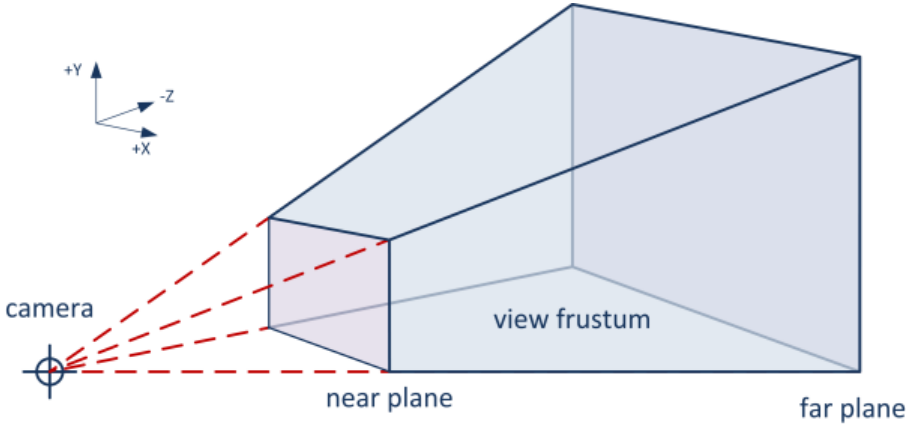

+y
$\vec{e}$
-x
+
x

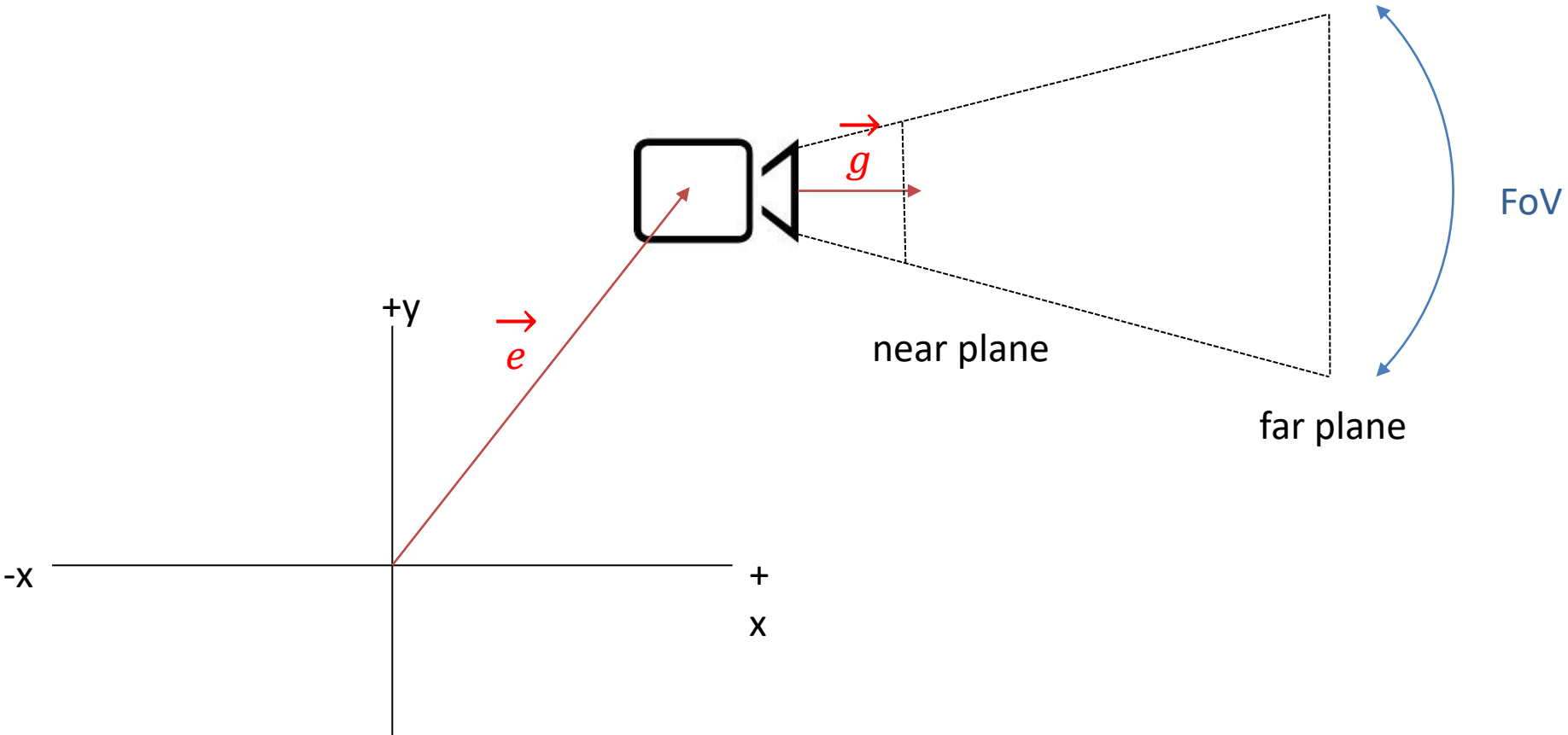

$\vec{g}$
+y
$\vec{e}$
-x
+
x


view frustum
$\vec{g}$
$\vec{e}$
+y
-x
+
x

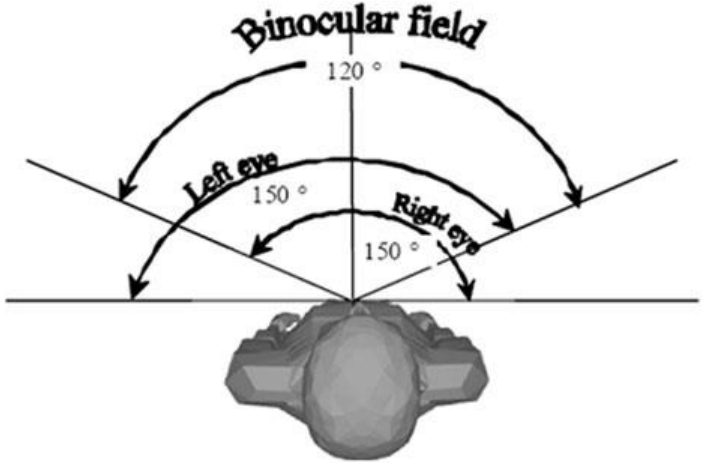

view frustum
$\vec{g}$
$\vec{e}$
+y
-x
+
x
far plane

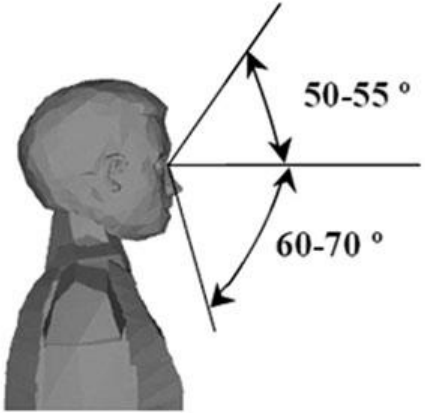

$\vec{g}$
near plane
far plane
+y
$\vec{e}$
-x
+
x


+Y
-Z
+X
camera
view frustum
near plane
far plane

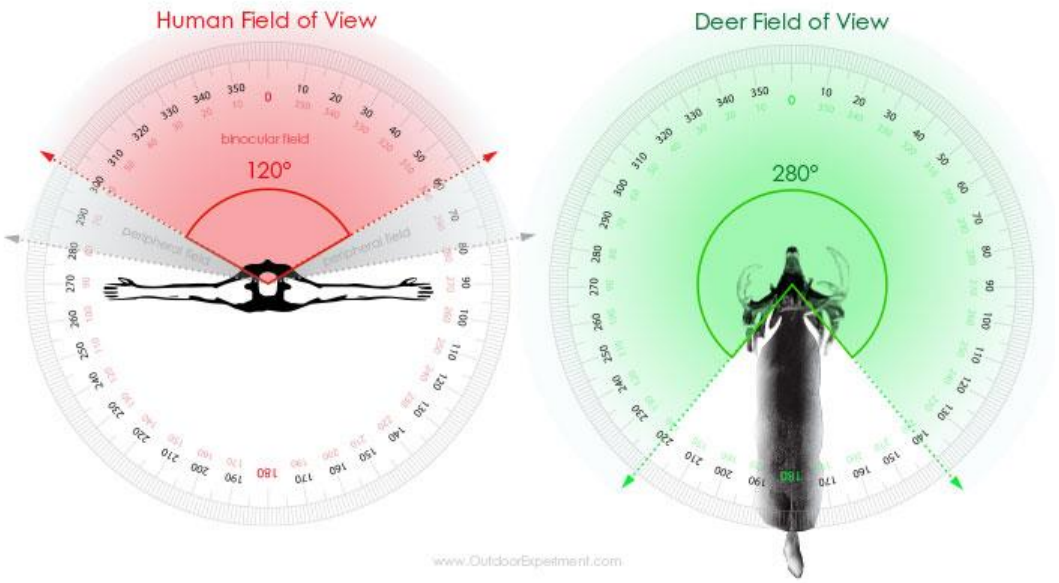

$\vec{g}$
FoV
+y
$\vec{e}$
near plane
far plane
-x
+
x

Field of View - FoV



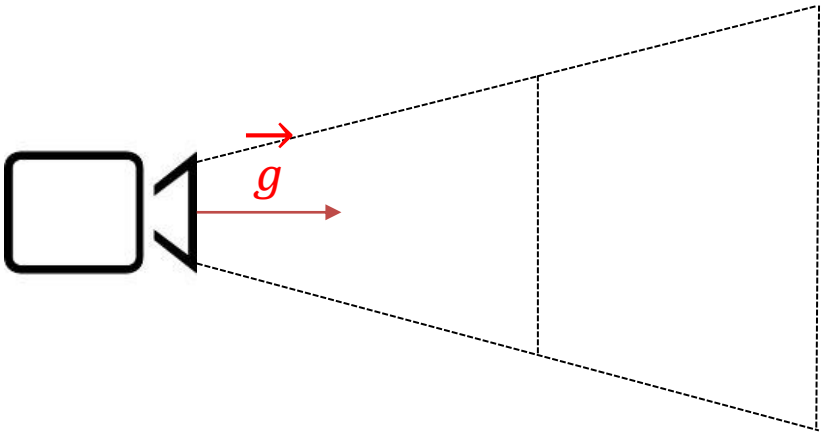


Field of View - FoV



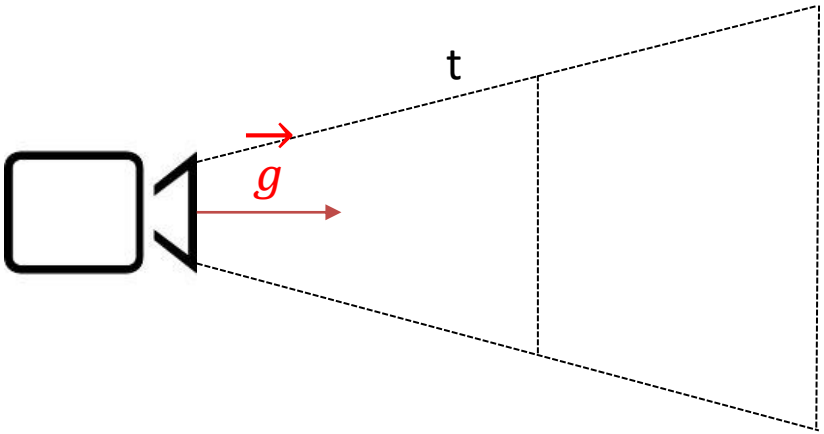
Camera Transformation

- View frustum:



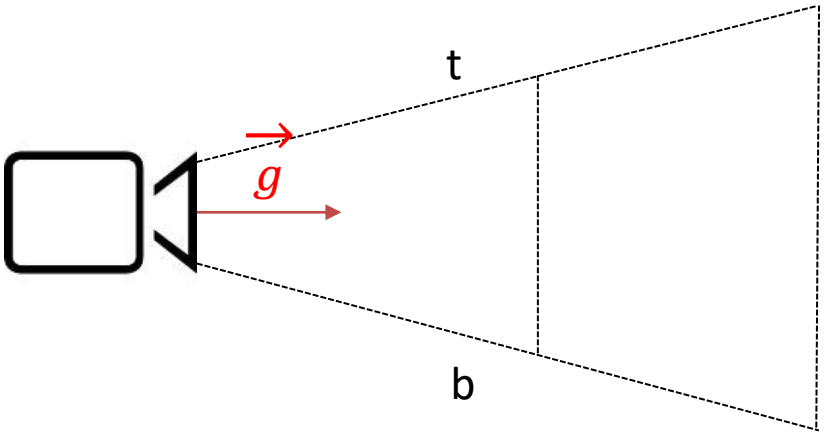
Camera Transformation

- View frustum:
  - Top plane



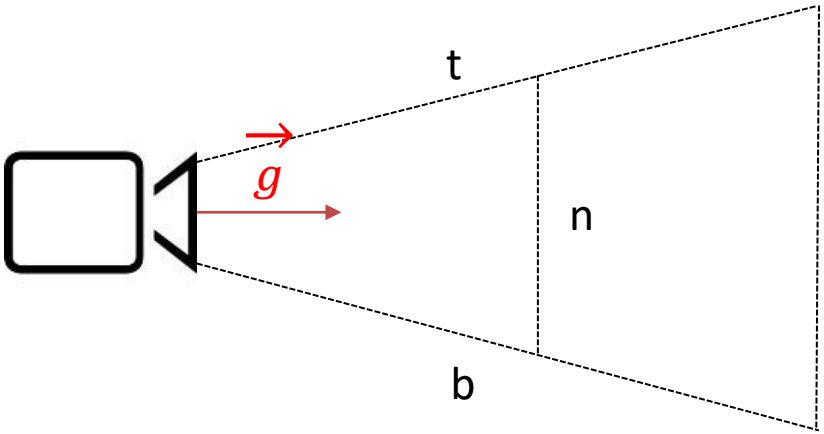
Camera Transformation

- View frustum:
  - Top plane
  - Bottom plane



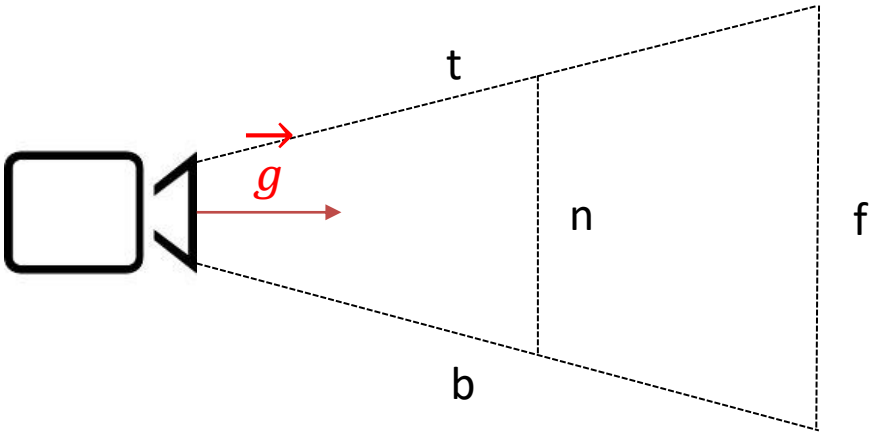
Camera Transformation

- View frustum:
  - Top plane
  - Bottom plane
  - Near plane



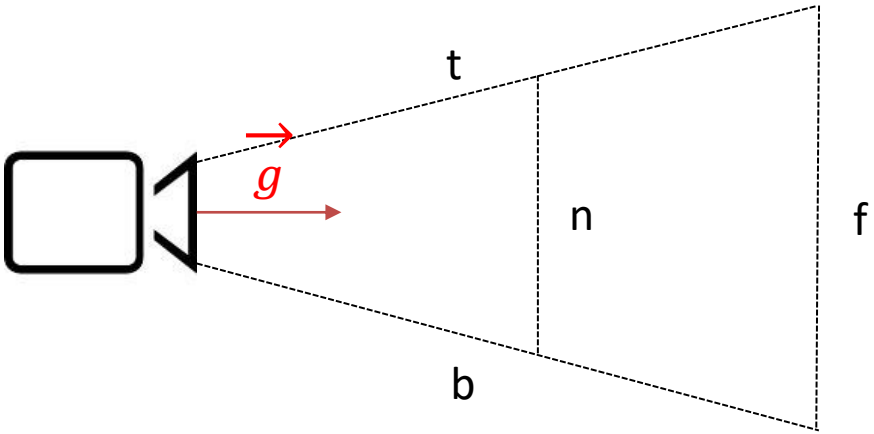
Camera Transformation

- View frustum:
  - Top plane
  - Bottom plane
  - Near plane
  - Far plane

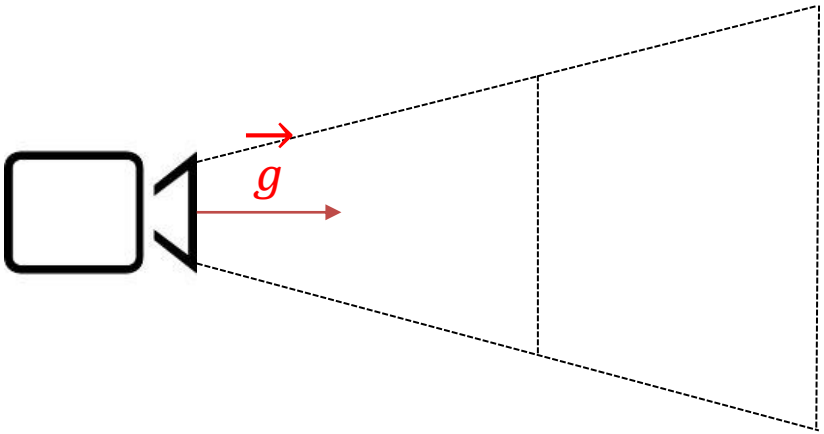


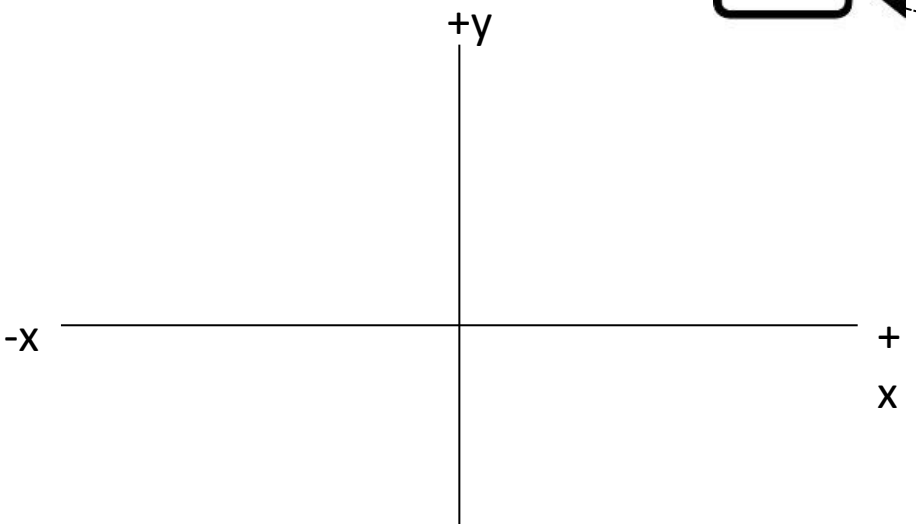
Camera Transformation

- View frustum:
  - Top plane
  - Bottom plane
  - Near plane
  - Far plane
  - Left plane l
  - Right plane r

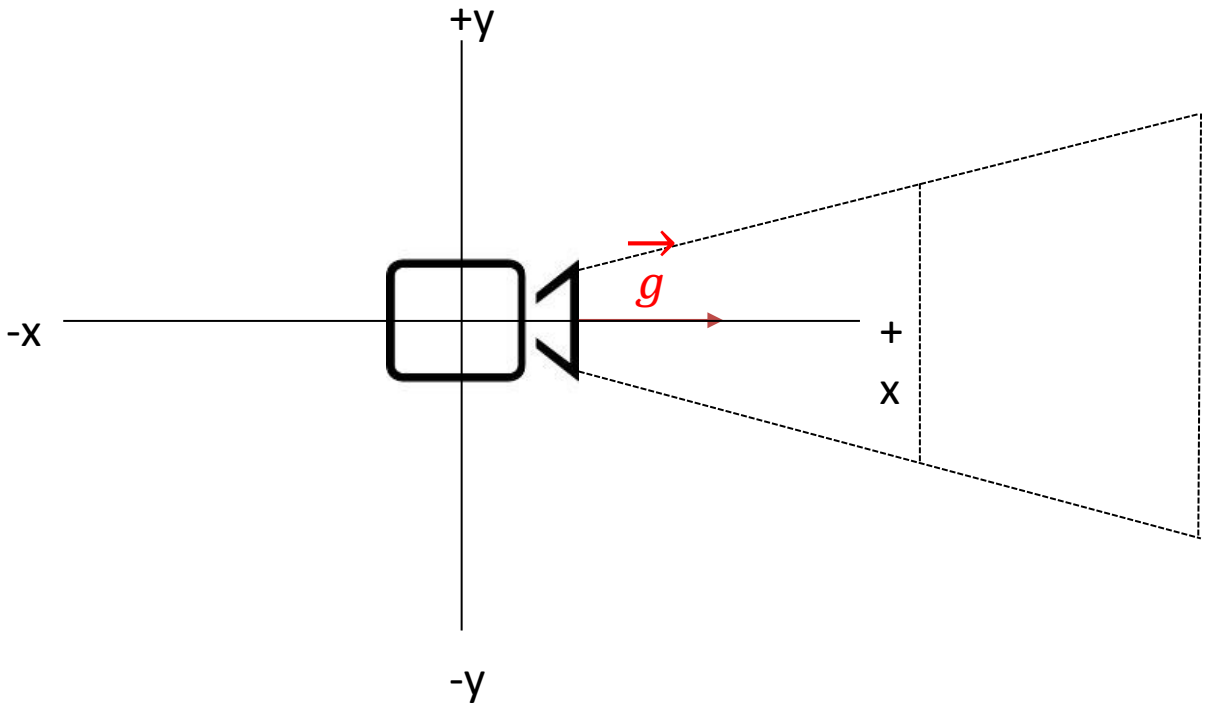


Camera Transformation

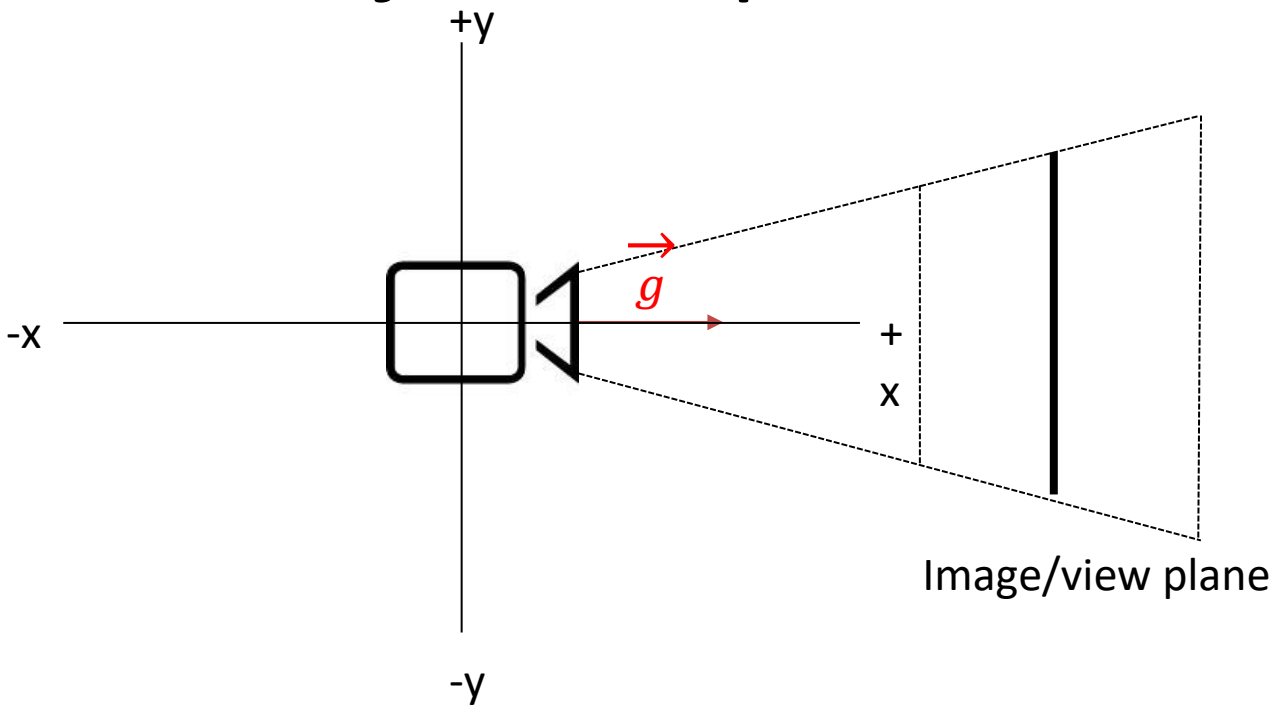




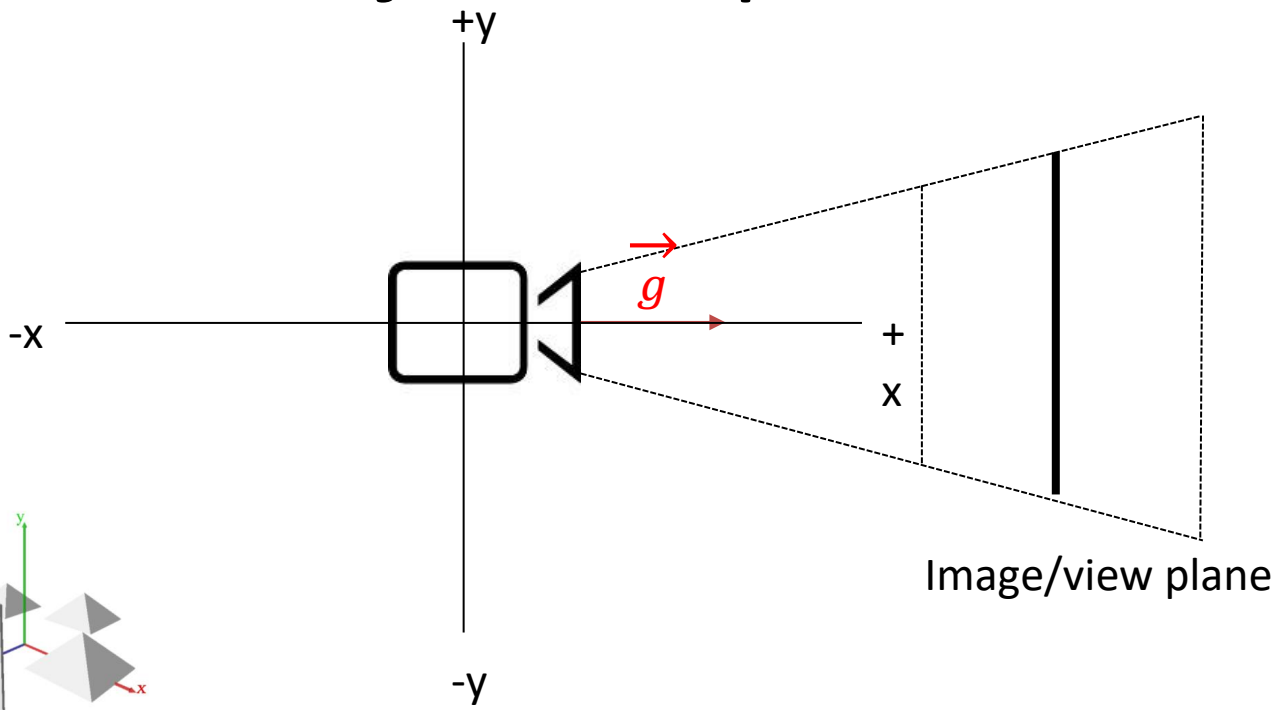
Camera Transformation

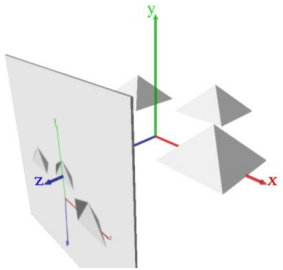


Projection plane

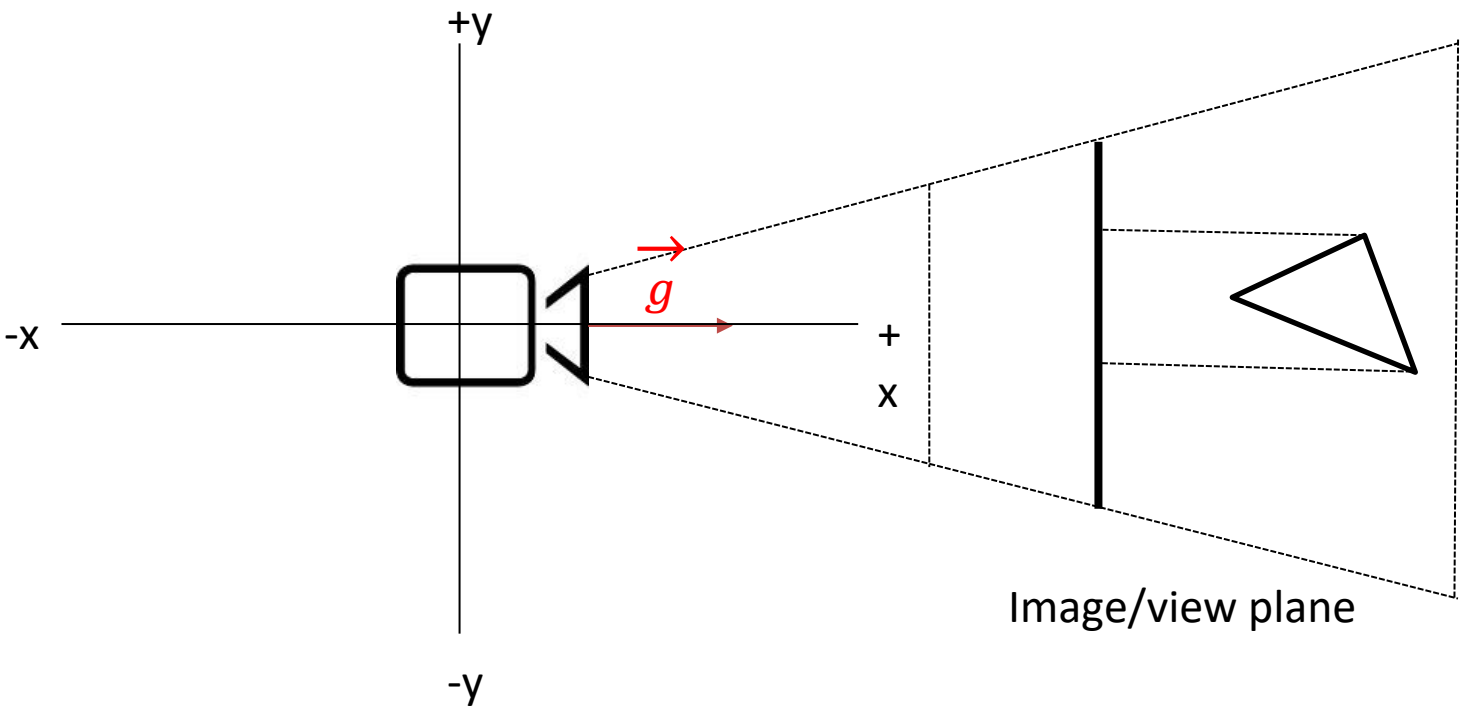


Projection plane

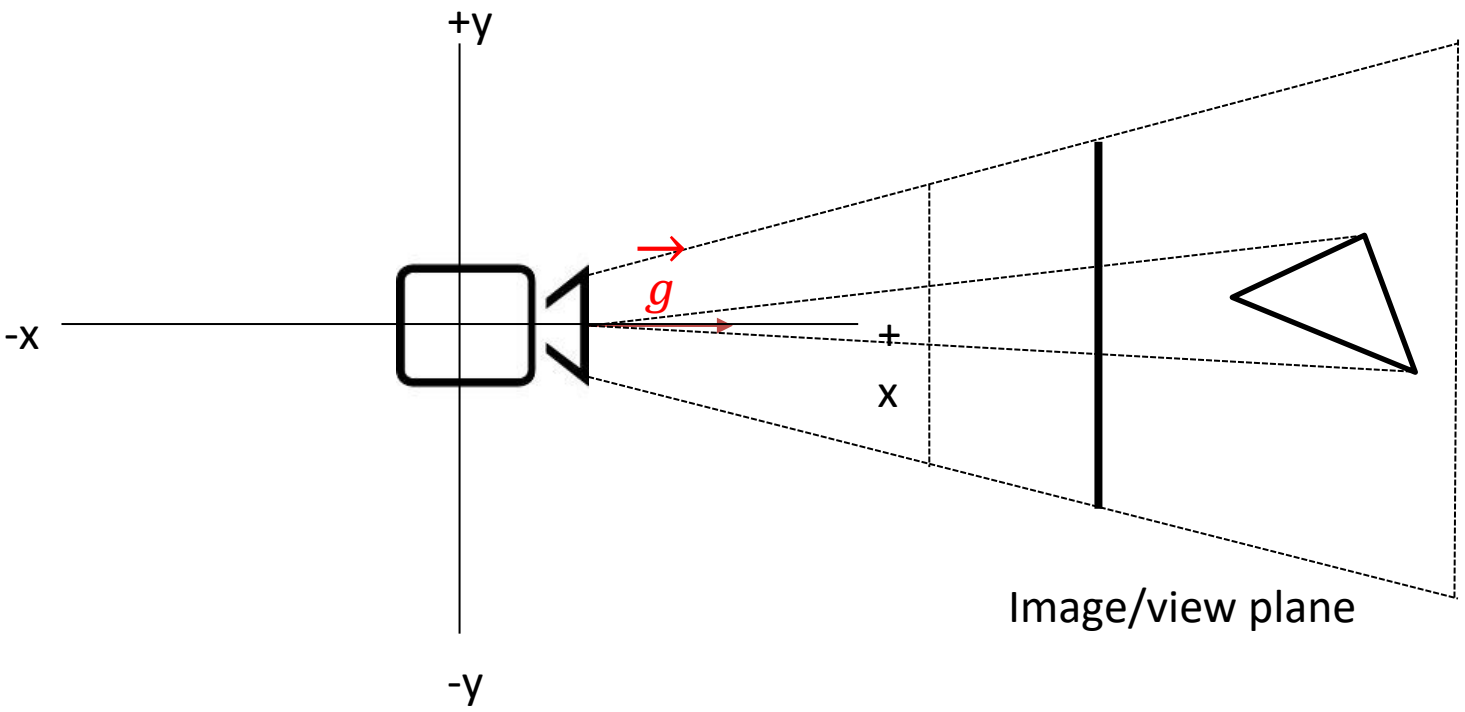


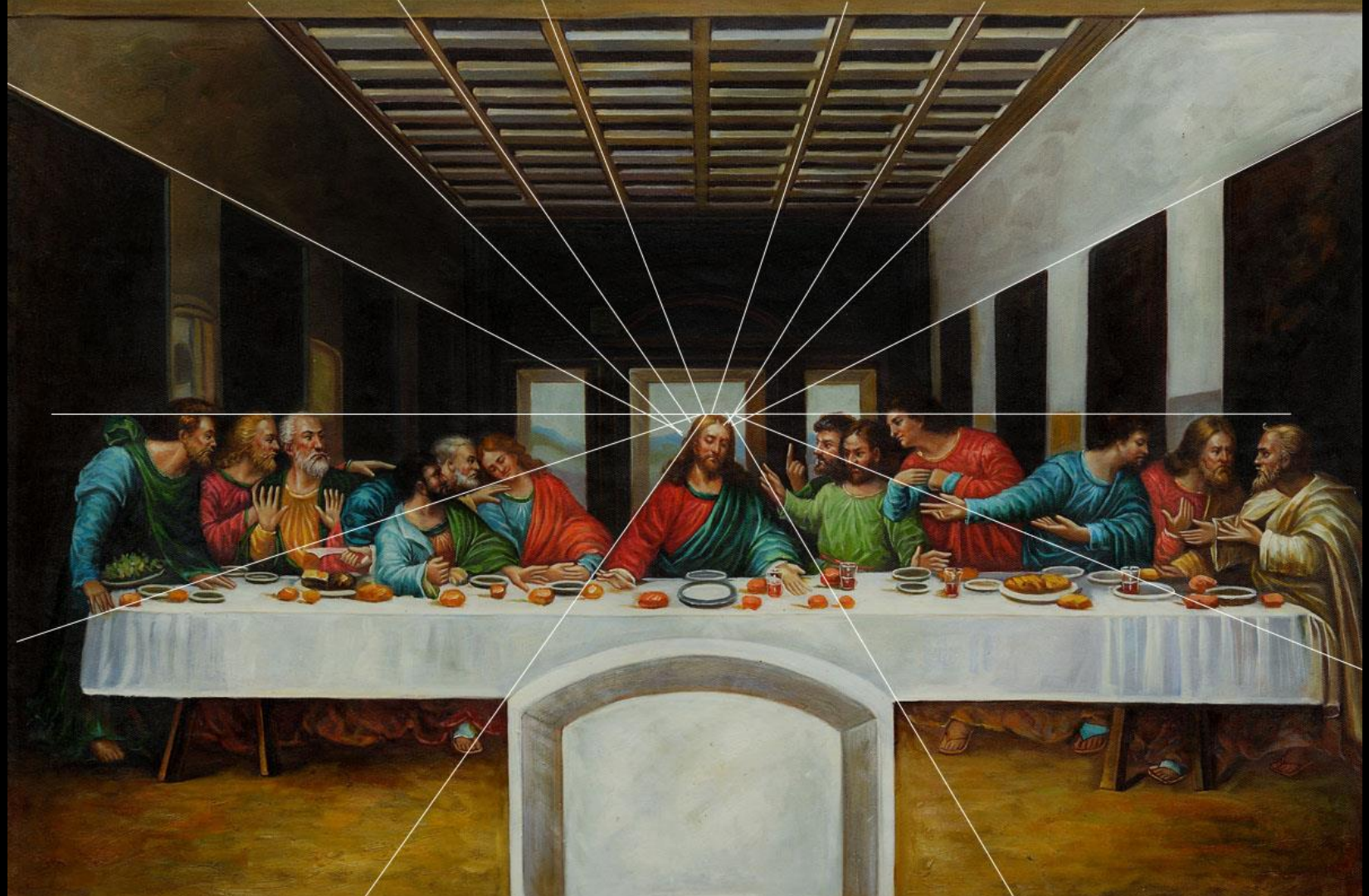


Parallel projection



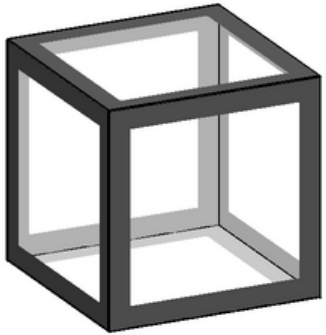
Perspective projection





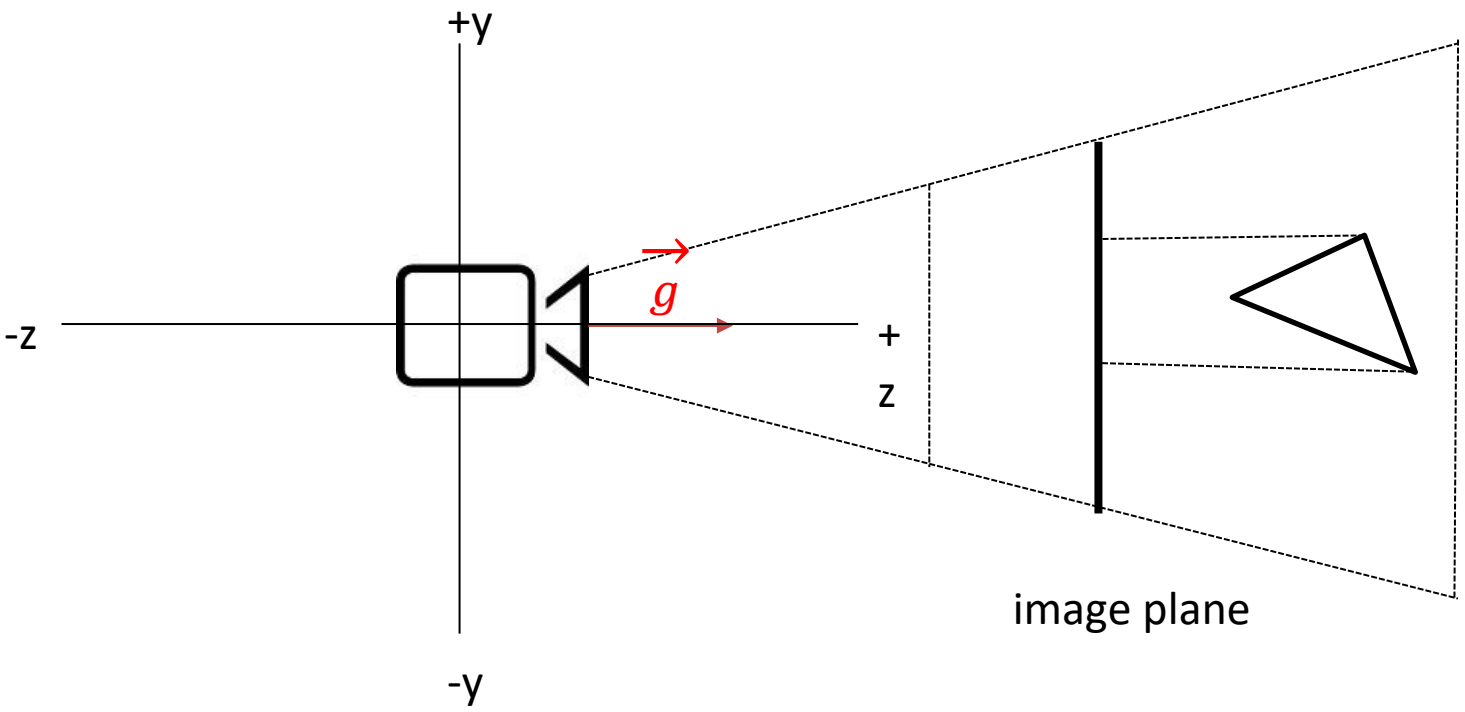
Projections

Perspective projection

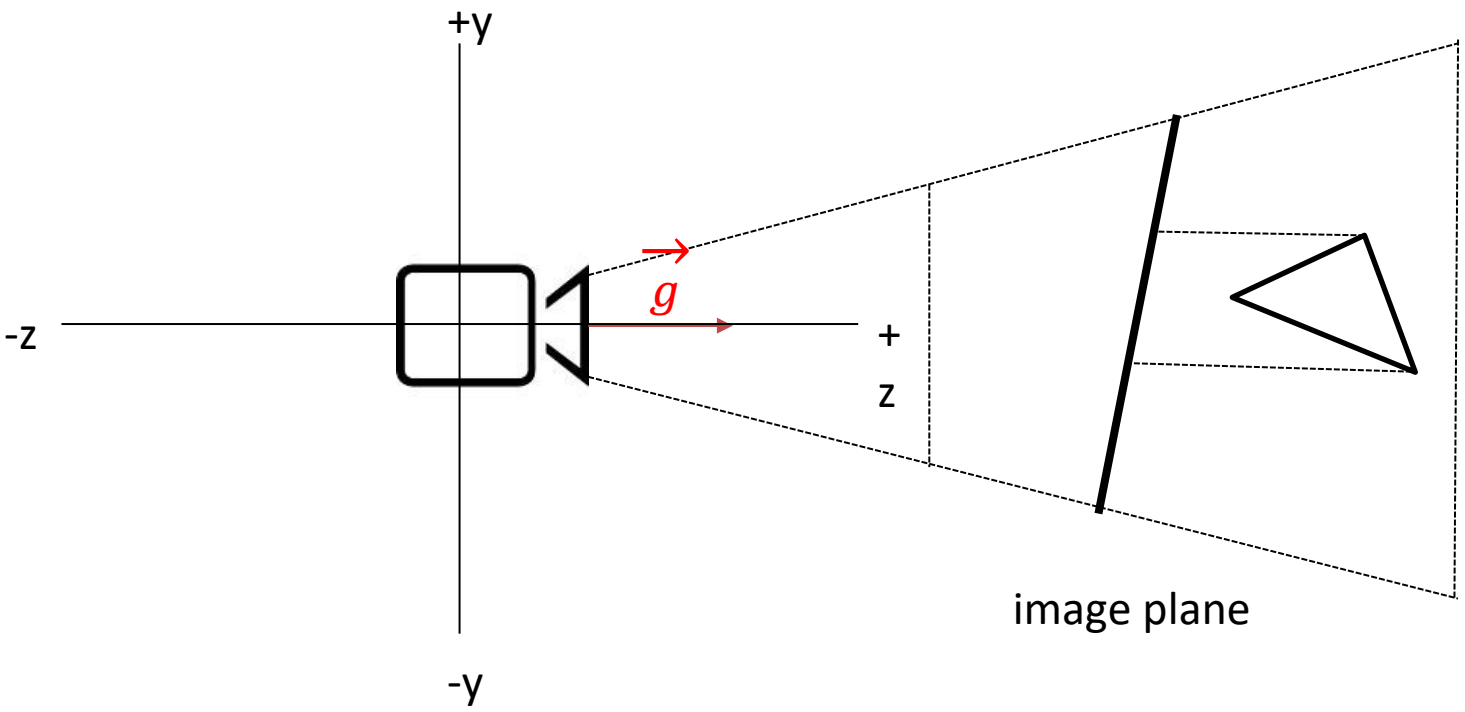


Orthographic projection

Orthographic projection

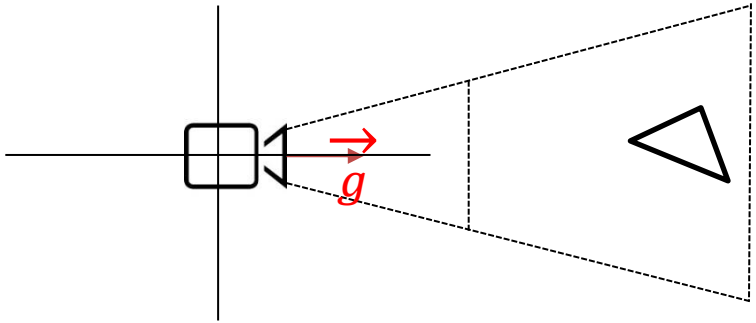


Non-orthographic projection

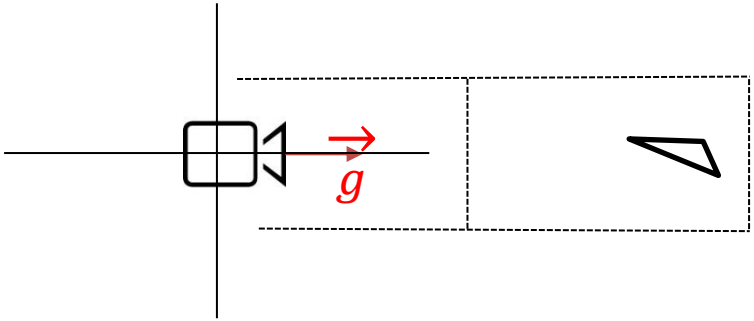


Orthographic projection

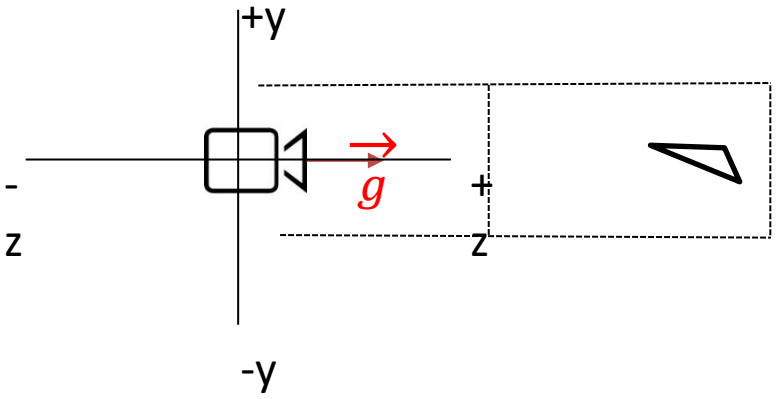
View frustum



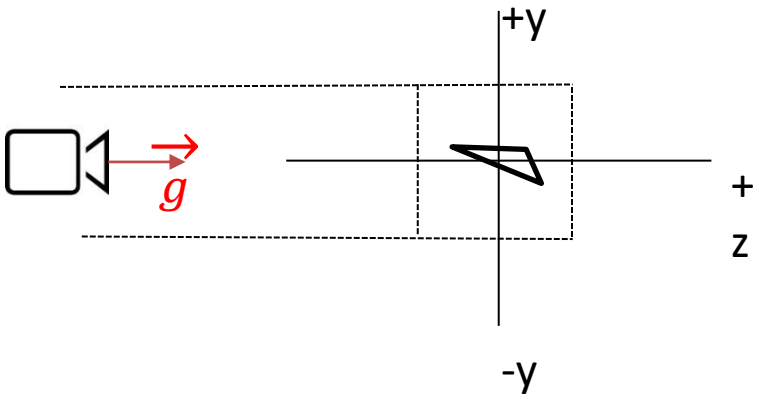
Orthographic view volume



Canonical view volume

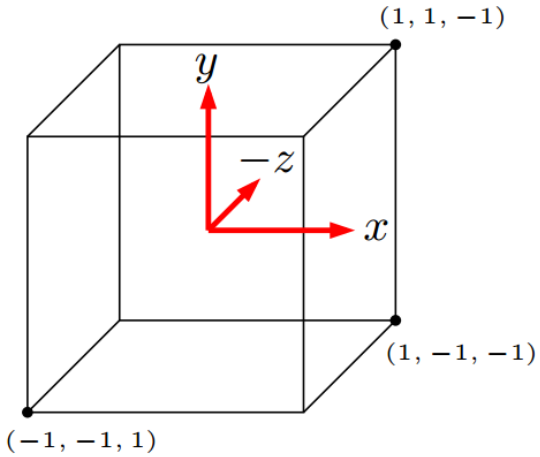


Orthographic view volume

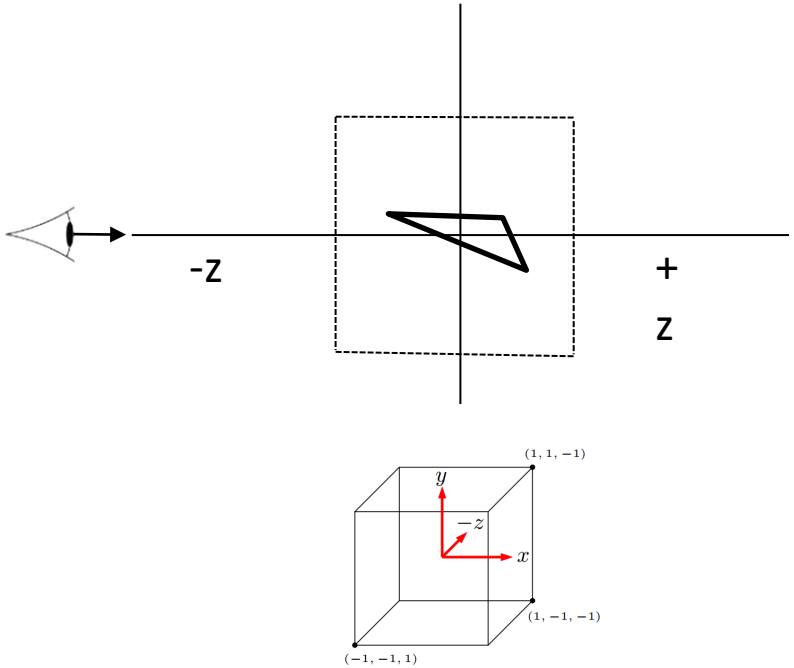


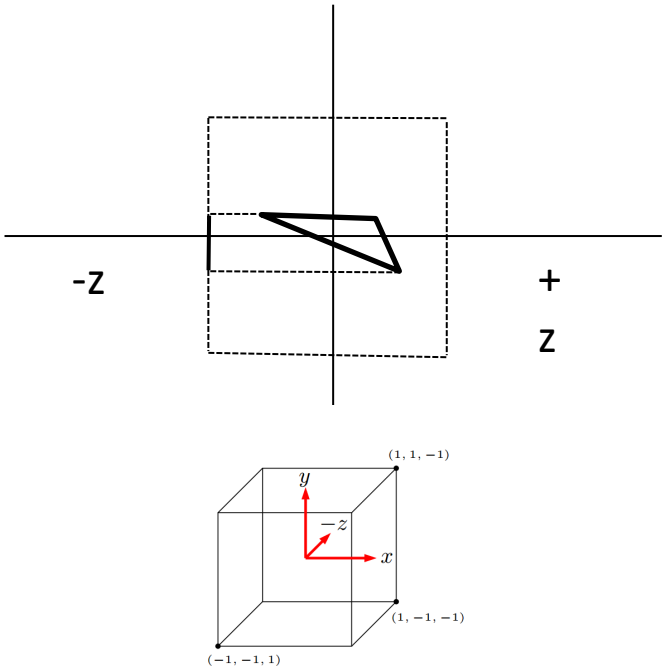
Canonical view volume

Canonical view volume

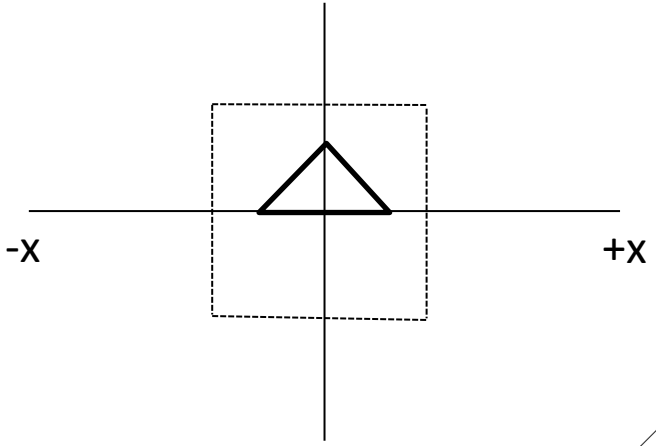


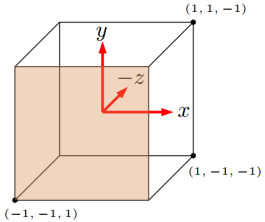
Viewport transformation



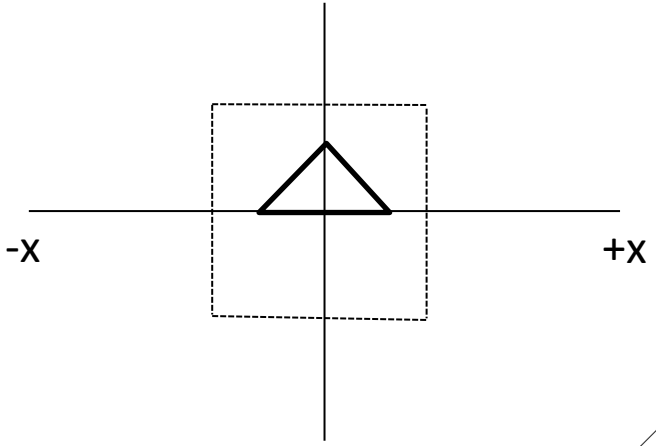


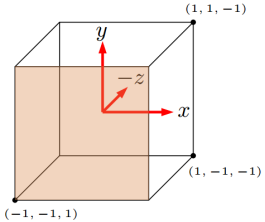
Viewport transformation

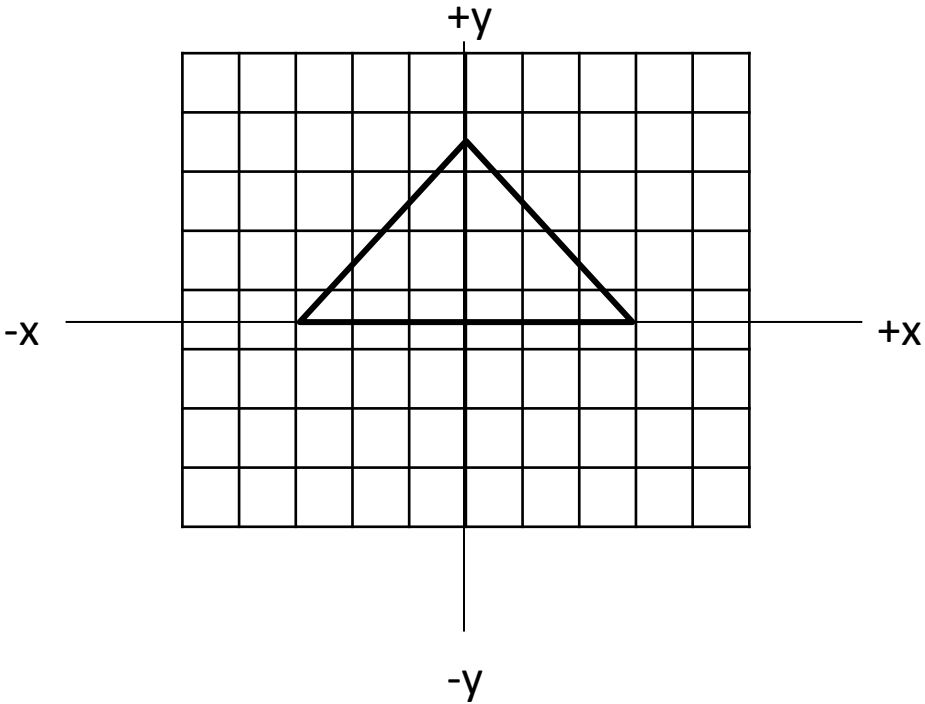




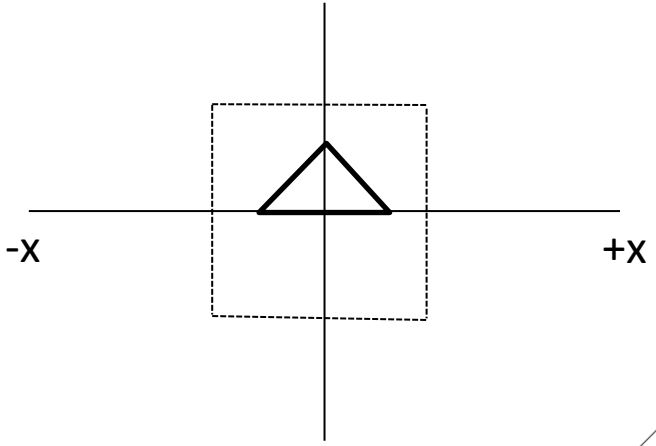
Viewport transformation

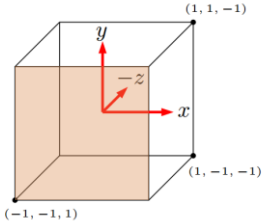


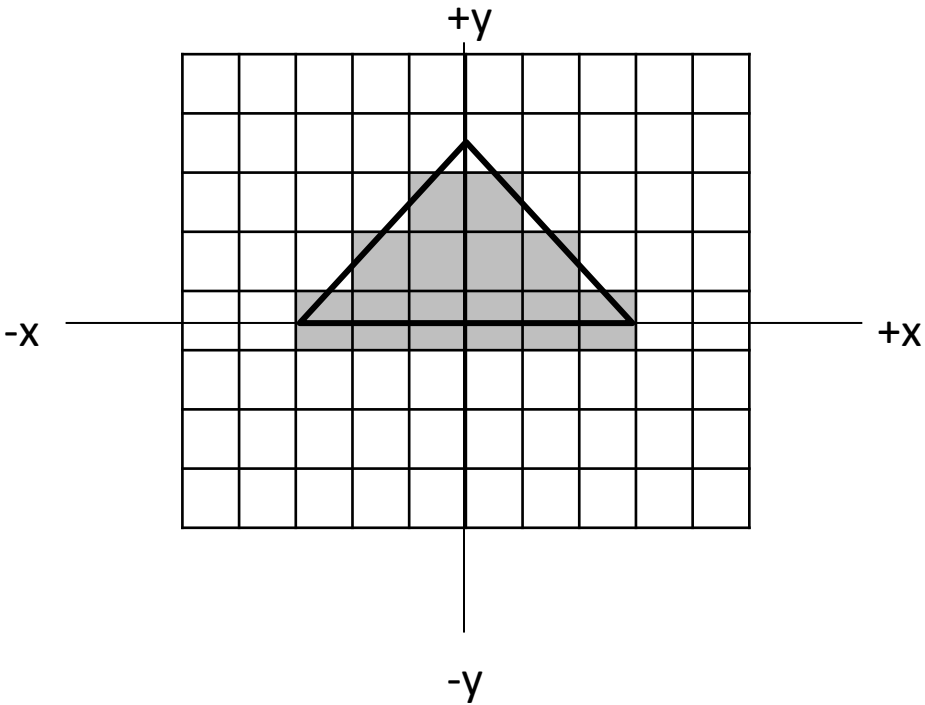


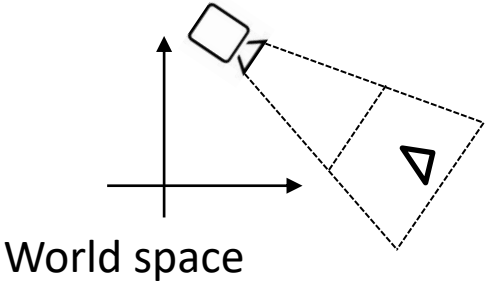


Viewport transformation





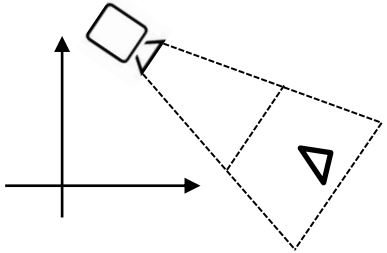


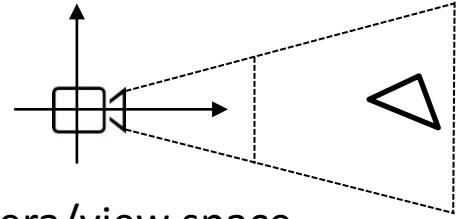


Rendering pipeline

- Every step we express as a matrix transformation. The whole pipeline can then be defined with **one matrix operation.**

Rendering pipeline

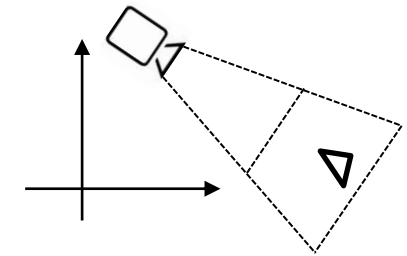


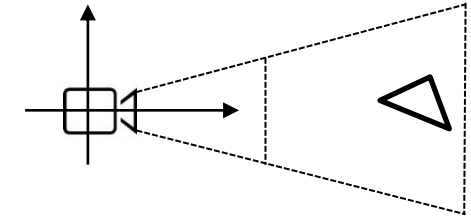


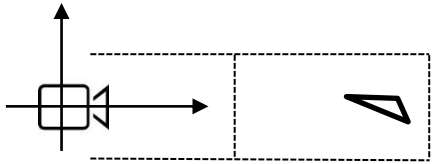
Camera/view space

- Every step we express as a matrix transformation. The whole pipeline can then be defined with **one matrix operation.**

Rendering pipeline



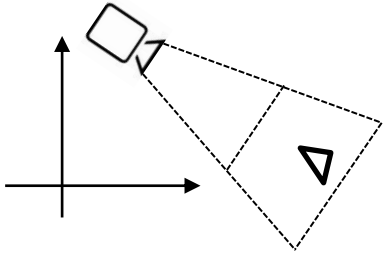


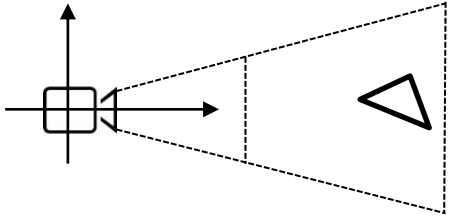


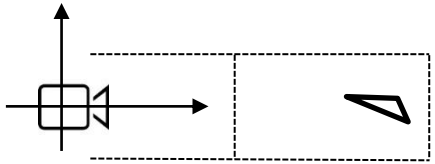
Orthographic view volume

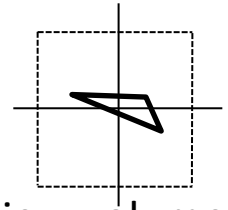
- Every step we express as a matrix transformation. The whole pipeline can then be defined with **one matrix operation.**

Rendering pipeline





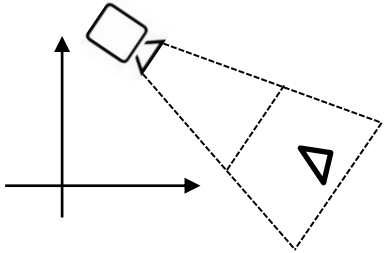


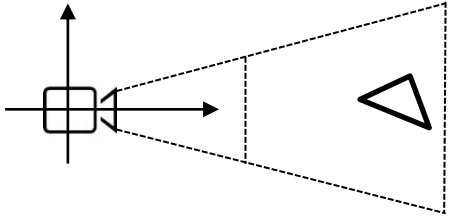


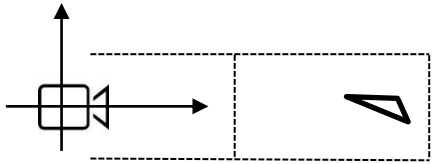
Canonical view volume

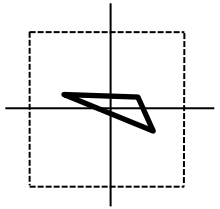
- Every step we express as a matrix transformation. The whole pipeline can then be defined with **one matrix operation.**

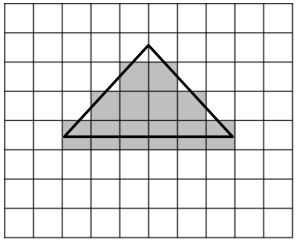
Rendering pipeline







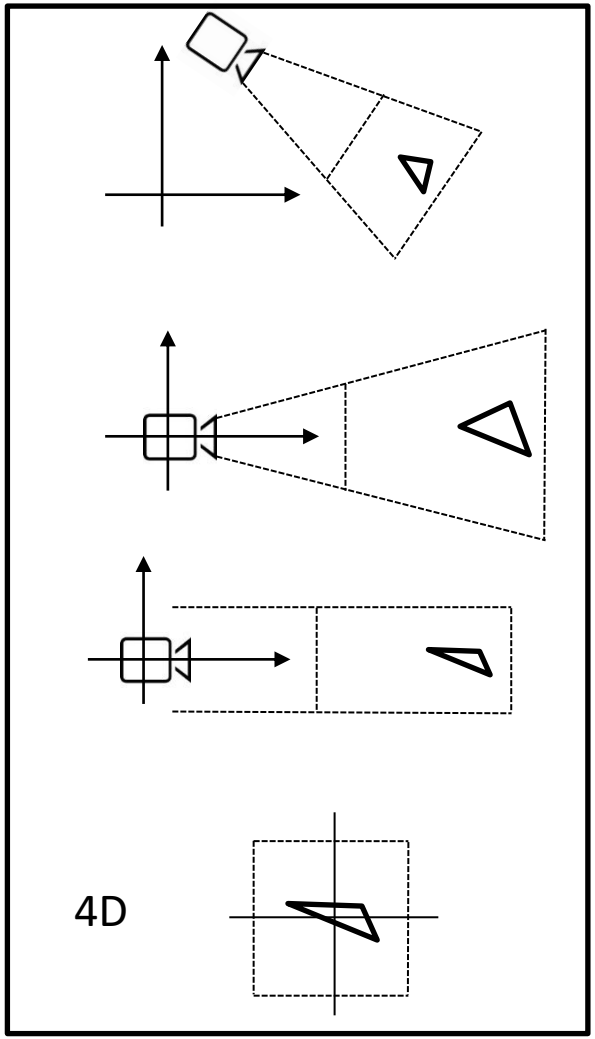


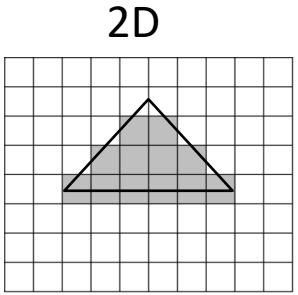


Window space

- Every step we express as a matrix transformation. The whole pipeline can then be defined with **one matrix operation.**

Rendering pipeline

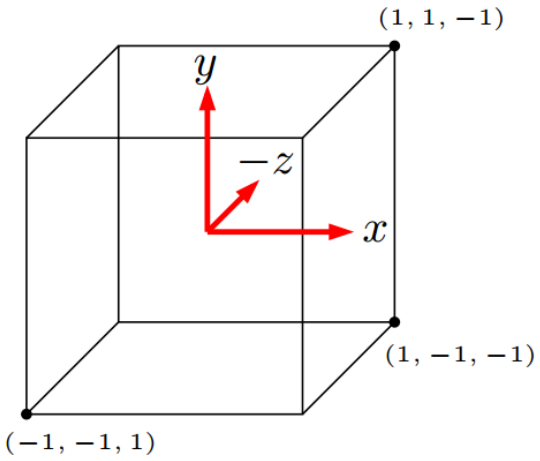




- Every step we express as a matrix transformation. The whole pipeline can then be defined with **one matrix operation.**

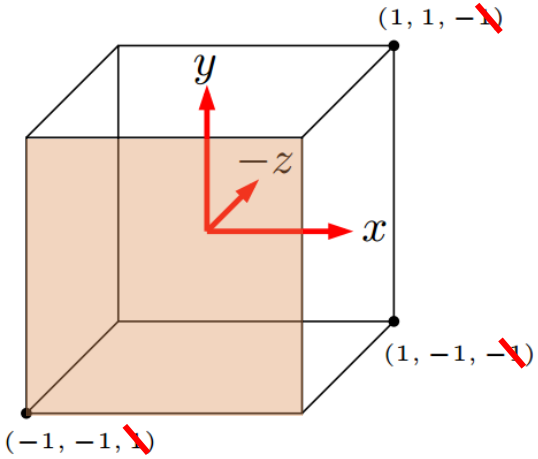
Canonical view volume

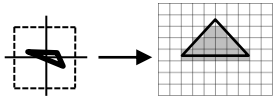
- The canonical view volume is a 2 x 2 x 2 cube with its center at the origin
- Assume that the view frustum has been transformed to such a shape



Canonical view volume

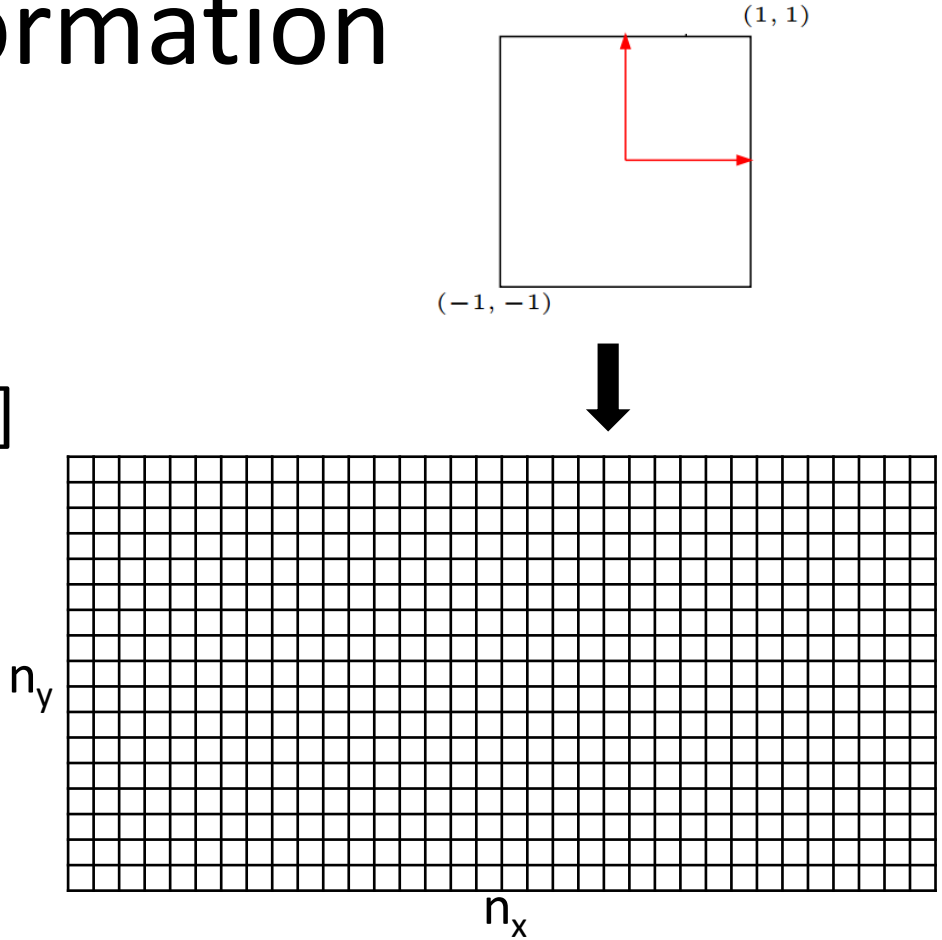
- The canonical view volume is a 2 x 2 x 2 cube with its center at the origin
- Assume that the view frustum has been transformed to such a shape
- We cut the z-coordinate

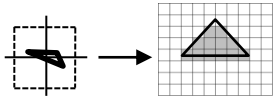




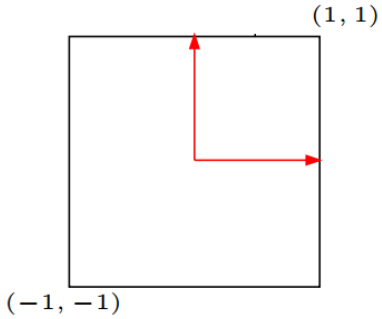
Transformation

- We have to map the square $[-1, 1]^2$ to a rectangle $[0, n_x] \times [0, n_y]$



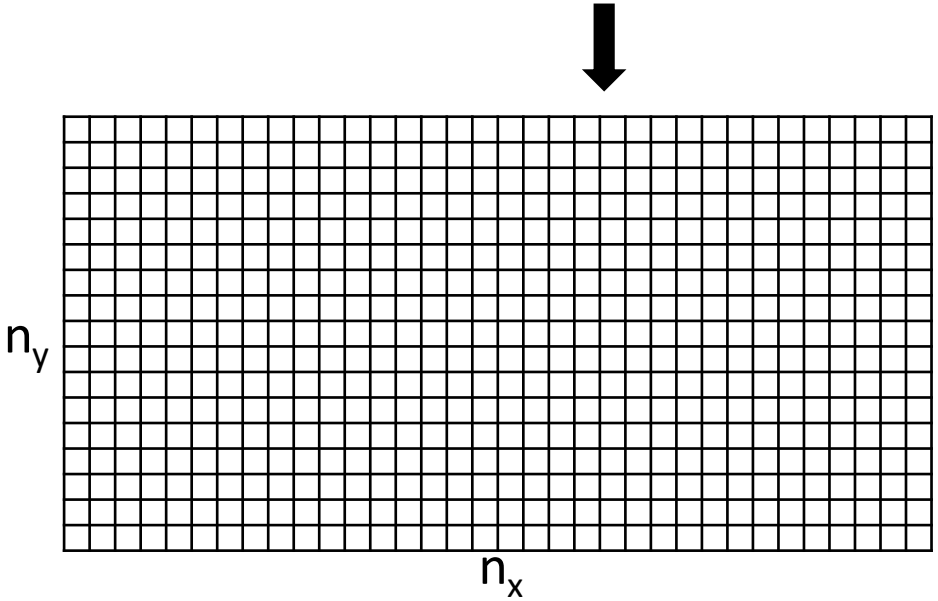


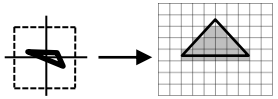
Transformation



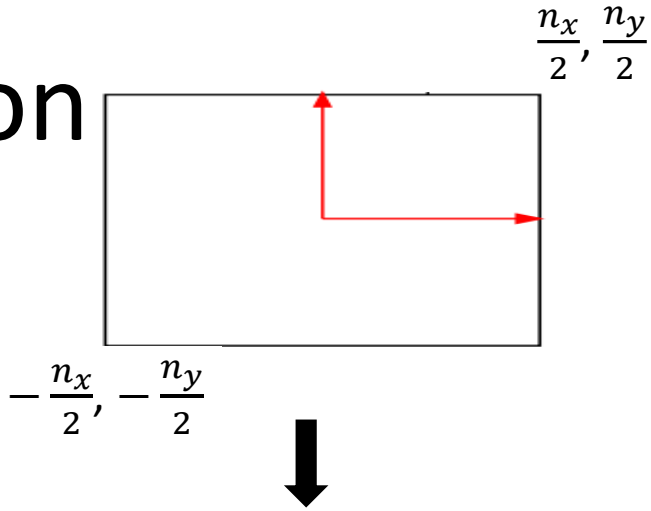
- We have to map the square $[-1, 1]^2$ to a rectangle $[0, n_x] \times [0, n_y]$
- The following scaling matrix accomplishes it:

$$\begin{bmatrix} \frac{n_x}{2} & 0 & 0 \\ 0 & \frac{n_y}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



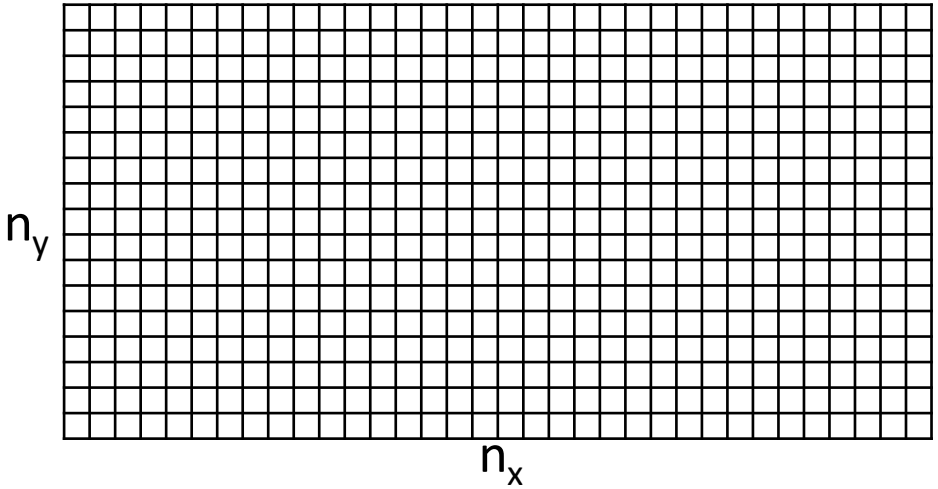


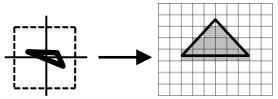
Transformation



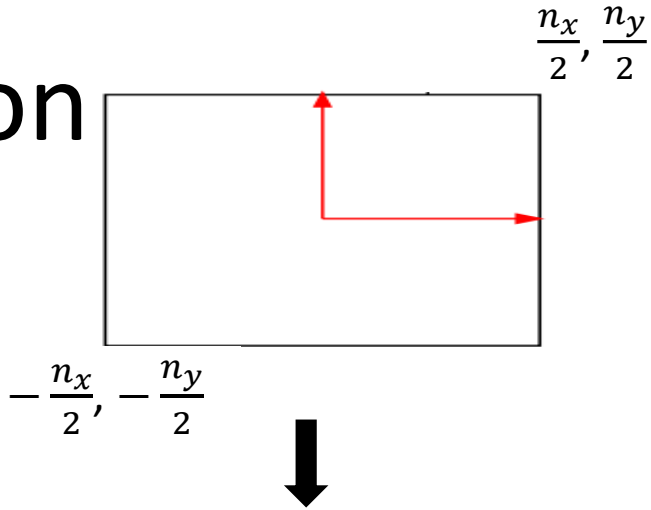
- We have to map the square $[-1, 1]^2$ to a rectangle $[0, n_x] \times [0, n_y]$
- The following scaling matrix accomplishes it:

$$\begin{bmatrix} \frac{n_x}{2} & 0 & 0 \\ 0 & \frac{n_y}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



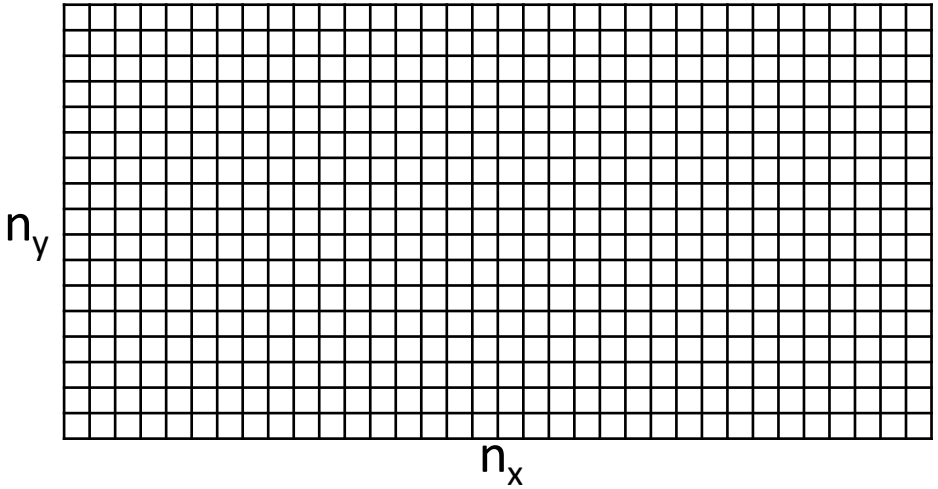


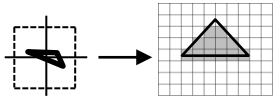
Transformation



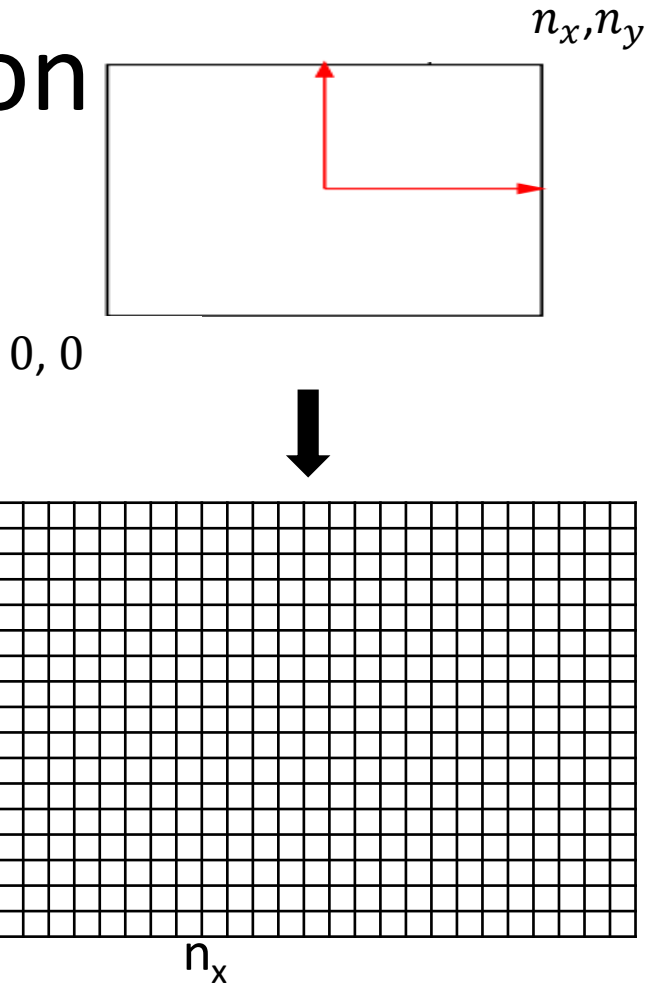
- We have to map the square $[-1, 1]^2$ to a rectangle $[0, n_x] \times [0, n_y]$
- The following matrix translates to the correct position:

$$\begin{bmatrix} 1 & 0 & \frac{n_x}{2} \\ 0 & 1 & \frac{n_y}{2} \\ 0 & 0 & 1 \end{bmatrix}$$



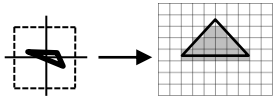


Transformation



- Resulting in this transformation

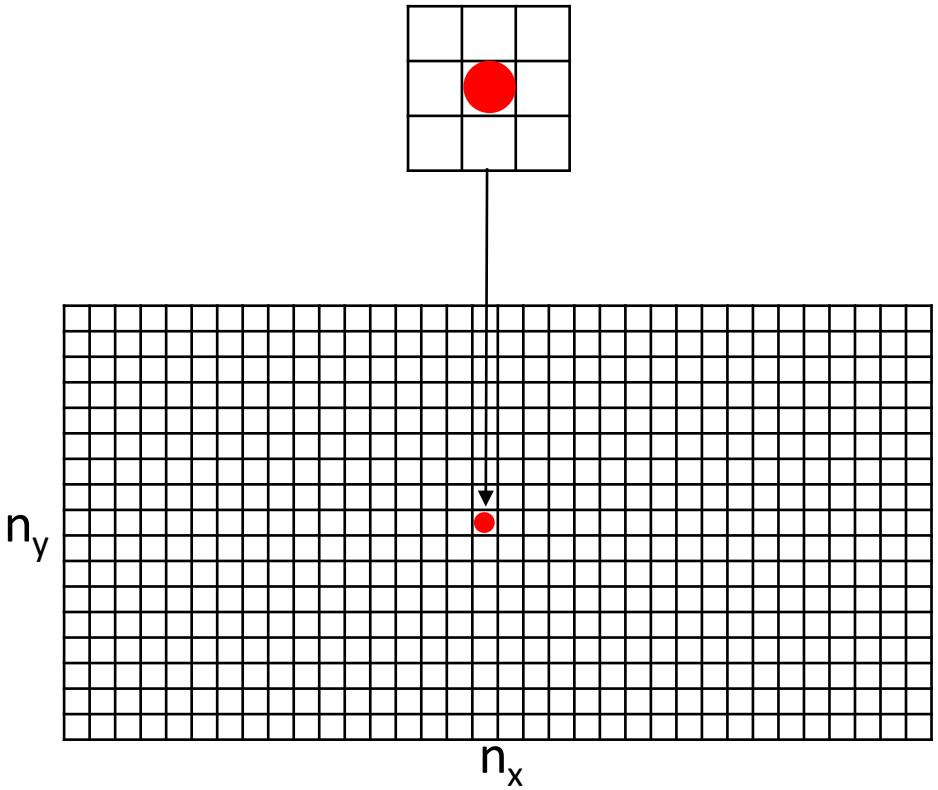
$$\begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

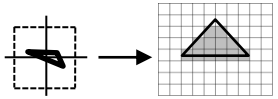


Transformation

- In practice pixels represent coordinates in integer units therefore we have to translate $[-\frac{1}{2}, n_x - \frac{1}{2}] \times [-\frac{1}{2}, n_y - \frac{1}{2}]$
- Then:

$$\begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x}{2} - \frac{1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y}{2} - \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

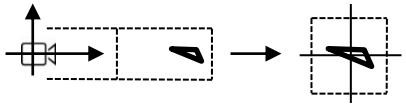




Transformation

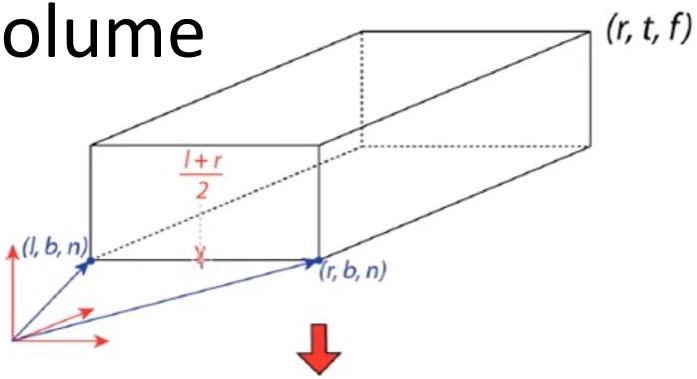
- The projection is simply the „cutting” of the ***z-coordinate***
- However we still need to be able to concatenate the matrix with other matrices, so the final 4D matrix is:

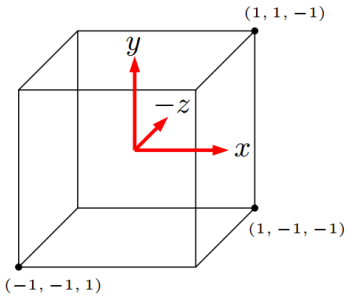
$$M_{vp} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x}{2} - \frac{1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y}{2} - \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

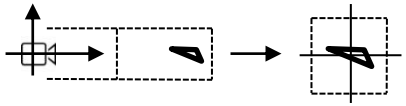


Orthographic view volume

- First we align the center with the origin

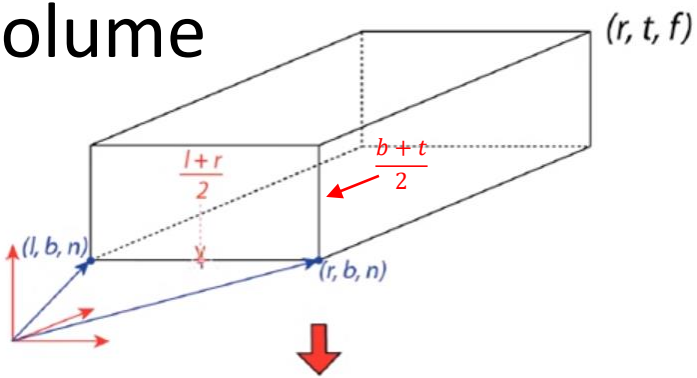


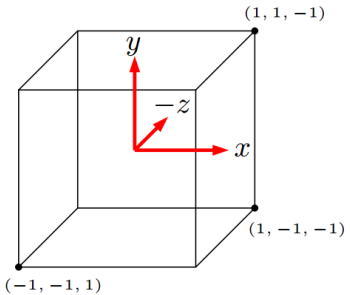


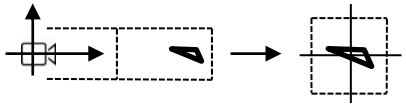


Orthographic view volume

- First we align the center with the origin

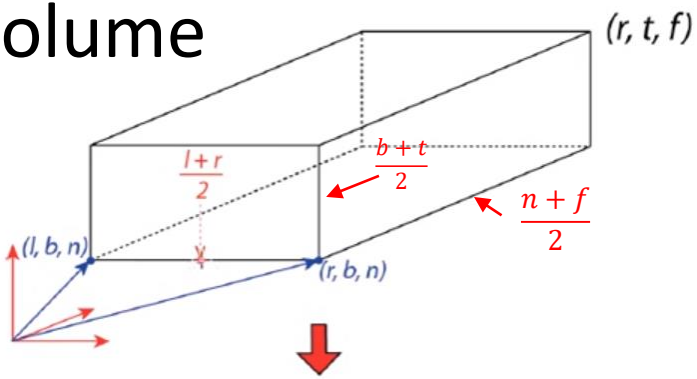


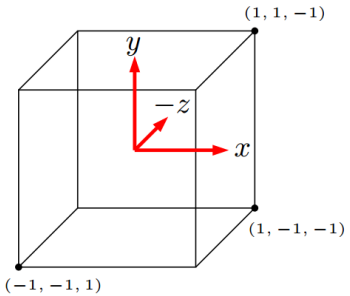


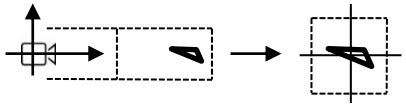


Orthographic view volume

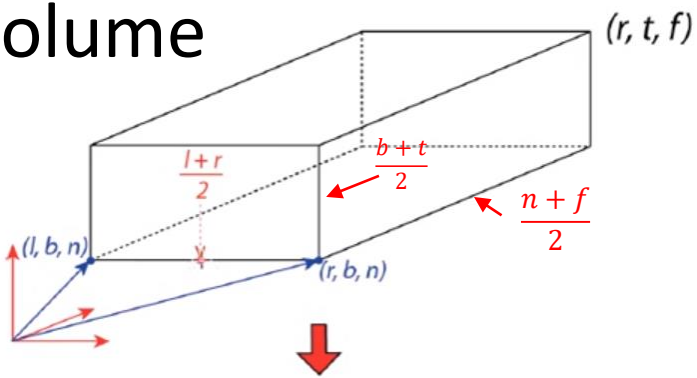
- First we align the center with the origin





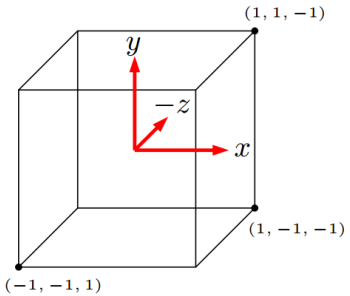


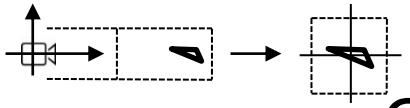
Orthographic view volume



- First we align the center with the origin

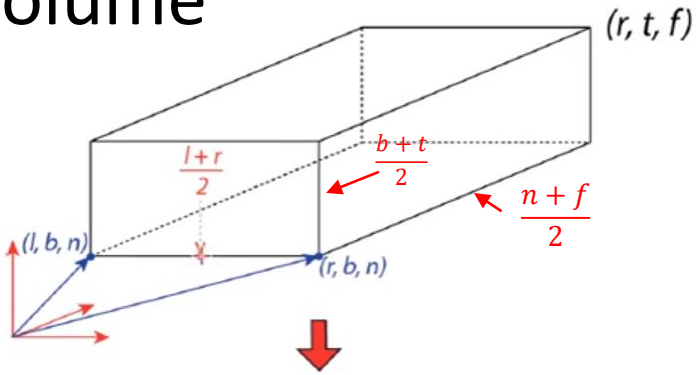
- $$\begin{bmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{b+t}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

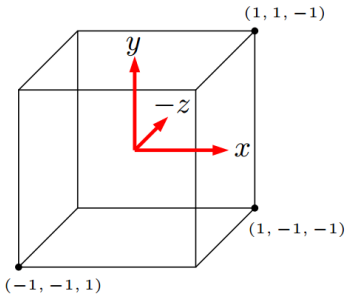


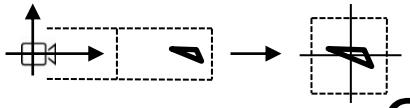


Orthographic view volume

- Then we have to scale the volume to the extents of [-1, 1]:

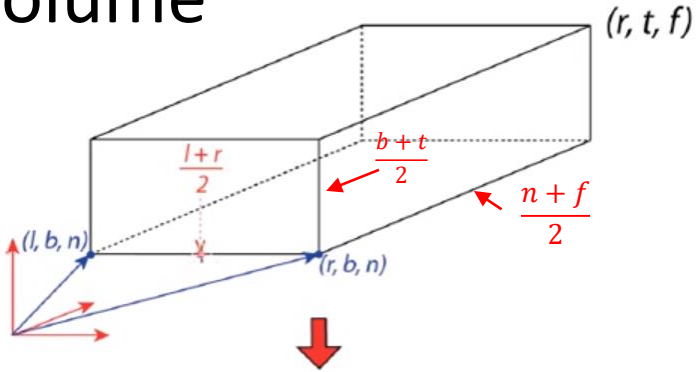




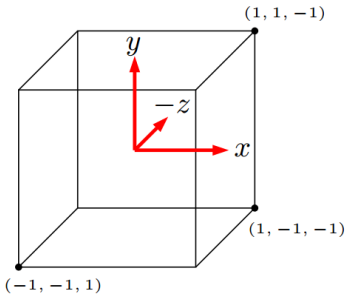


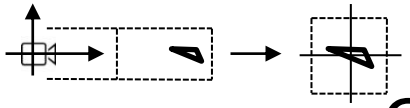
Orthographic view volume

- Then we have to scale the volume to the extents of [-1, 1]:



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

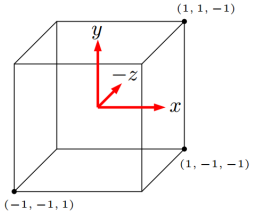


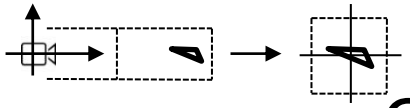


Orthographic view volume

- We multiply the matrices together to obtain a single matrix expressing the orthographic view volume:

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



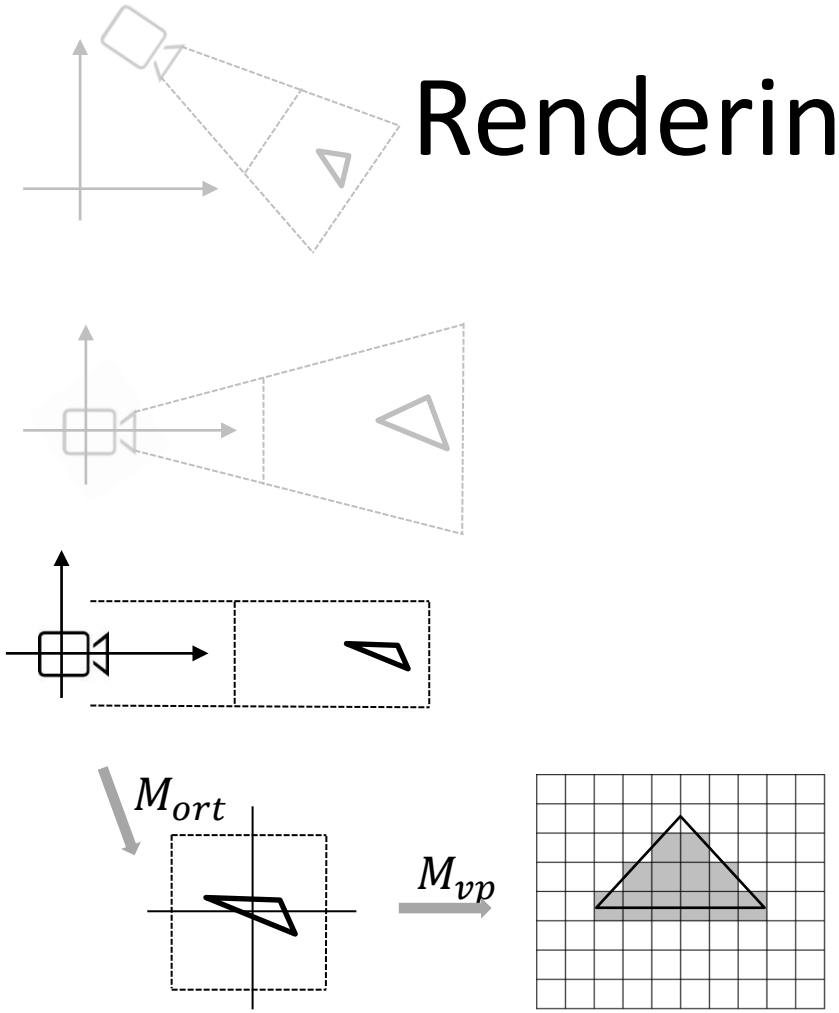


Orthographic view volume

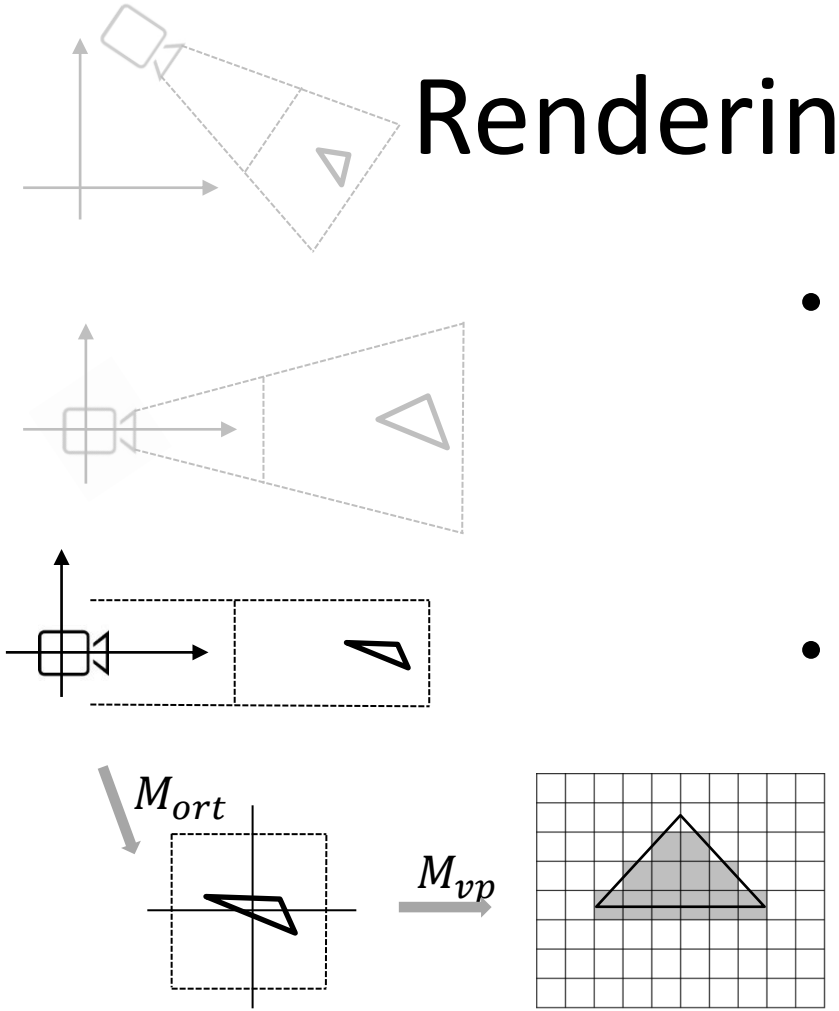
- We multiply the matrices together to obtain a single matrix expressing the orthographic view volume:

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

Rendering pipeline

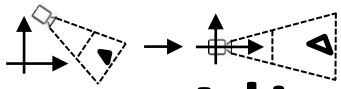


Rendering pipeline



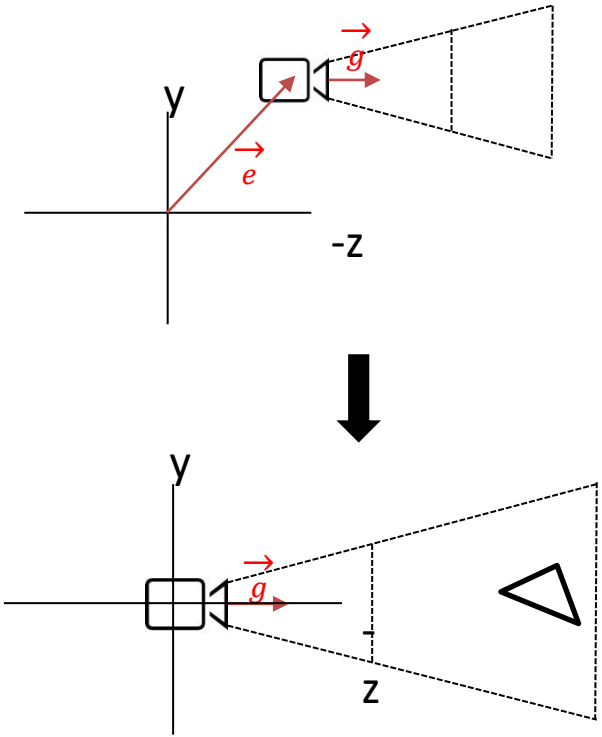
- Transformation:

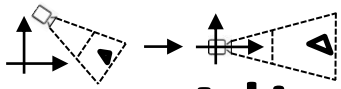
- $$\begin{bmatrix} x_{pixel} \\ y_{pixel} \\ z_{canonical} \\ 1 \end{bmatrix} = M_{vp} M_{ort} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Aligning coordinate systems

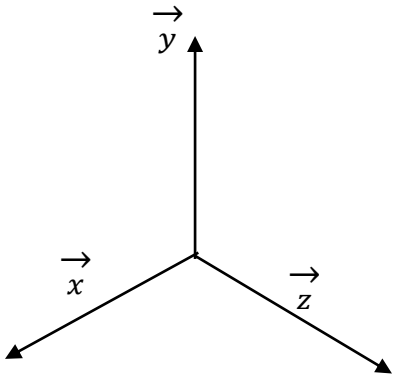
- World space:
  - Vectors, $\vec{x}, \vec{y}, i\ \vec{z}$
- Camera space:
  - Vectors $\vec{e}, i\ \vec{g}$



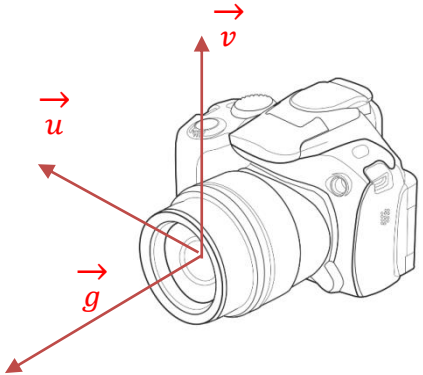


Aligning coordinate systems

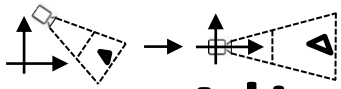
- To transform one space into another we need a single coordinate system for both



World space

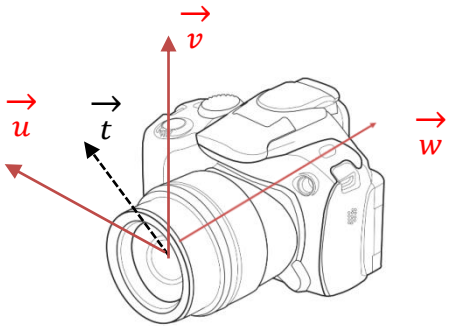


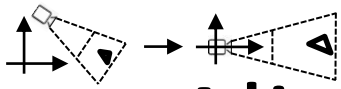
Camera space



Aligning coordinate systems

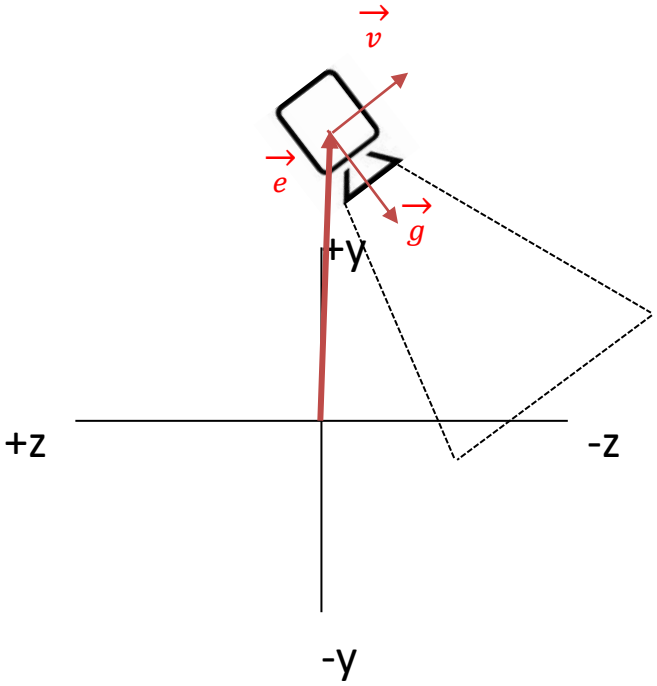
- First basis vector:
  - $\vec{t} \times \vec{g} = \vec{u}$
- Second basis vector:
  - $\vec{g} \times \vec{u} = \vec{v}$
- Third basis vector:
  - $-\vec{g} := \vec{w}$

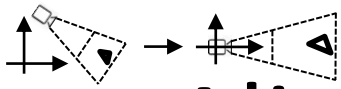




Aligning coordinate systems

- Align the origins
- Align the basis vector

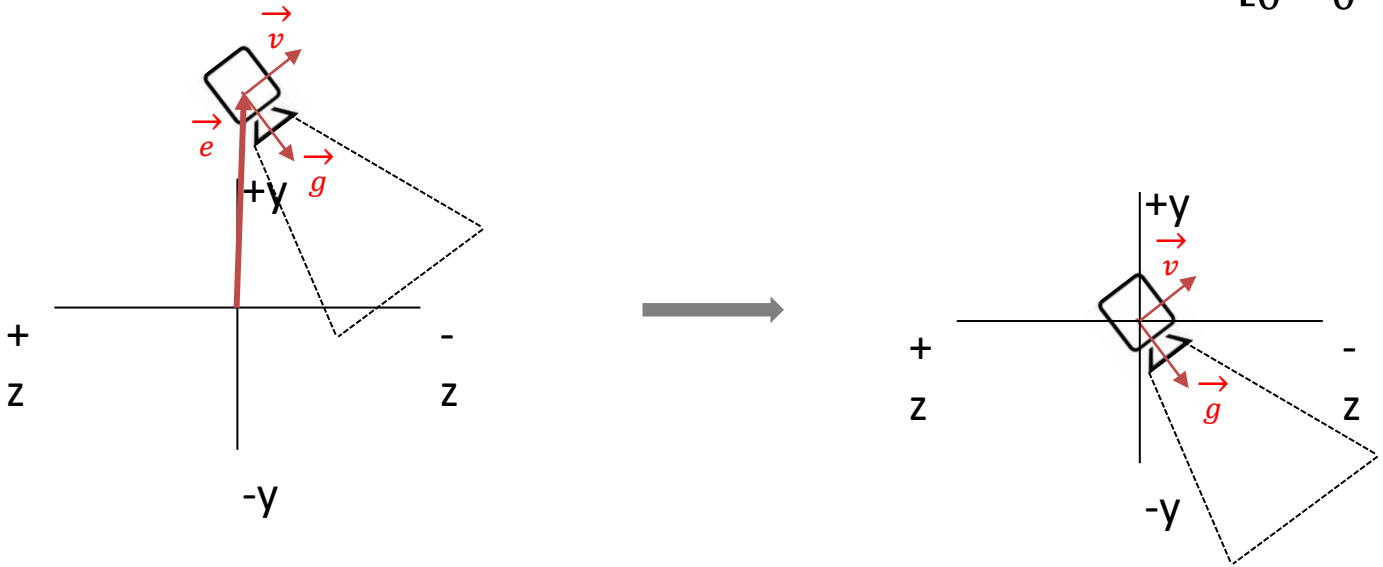


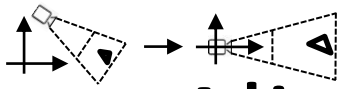


Aligning coordinate systems

- Align the origins via translation:

$$\begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

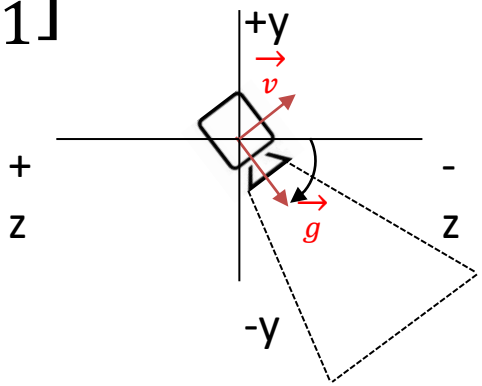


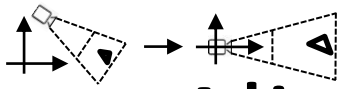


Aligning coordinate systems

- Rotation in the opposite direction is trivial:

$$\begin{bmatrix} u_x & v_x & w_x & 0 \\ u_y & v_y & w_y & 0 \\ u_z & v_z & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

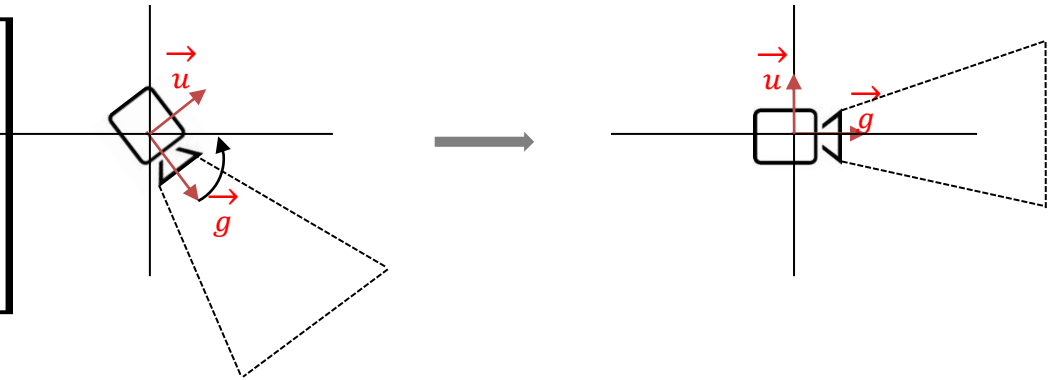


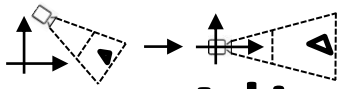


Aligning coordinate systems

- In case of an orthonormal matrix the inverse is the transpose matrix (vectors u,v,w have to be normalized):

$$-\begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



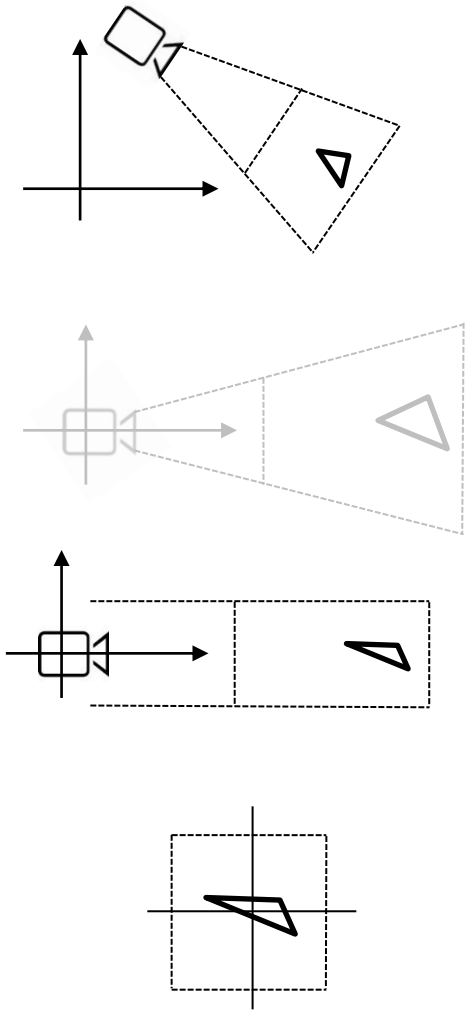


Aligning coordinate systems

- The transformation matrix from world space to view space is then:

$$M_{cam} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic projection

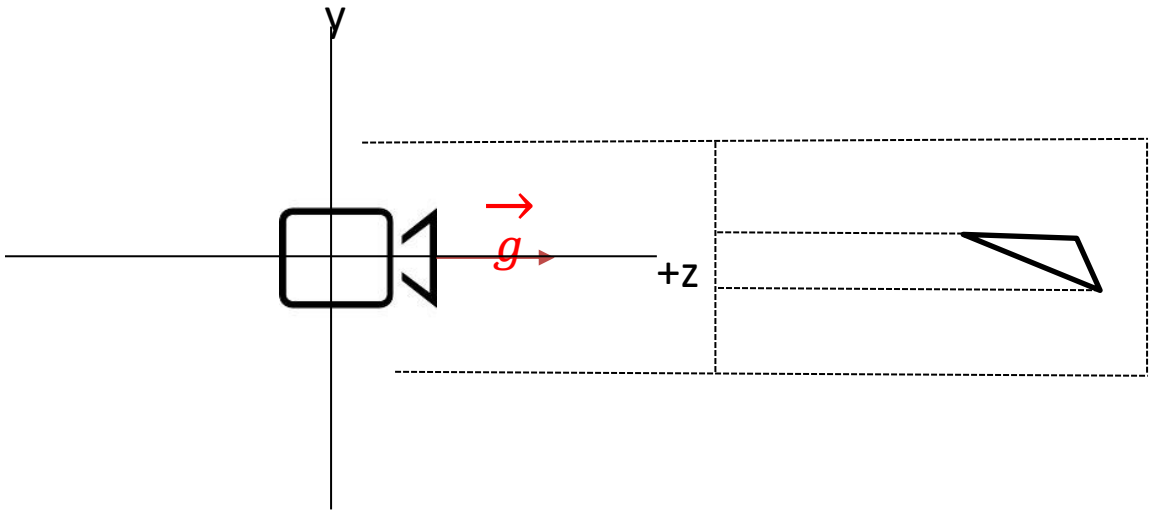


$$\begin{bmatrix} x_{piksel} \\ y_{piksel} \\ z_{kanoniczny} \\ 1 \end{bmatrix} = M_{vp} M_{orth} M_{cam} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Parallel projection

The Simpons Tapped Out

Parallel projection



In OpenGL

```
glUniformMatrix4fv(glGetUniformLocation(program, "transformation"), 1, GL_FALSE, (float*)&transformation);
```

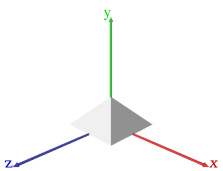
In OpenGL

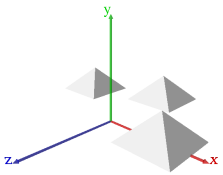
```
glUniformMatrix4fv(glGetUniformLocation(program, "transformation"), 1, GL_FALSE, (float*)&transformation);
```

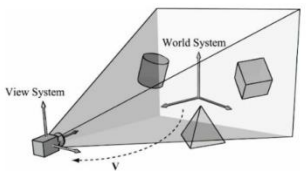
Model Matrix →

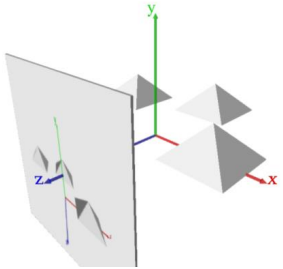
View Matrix →

Projection Matrix →









Object Space

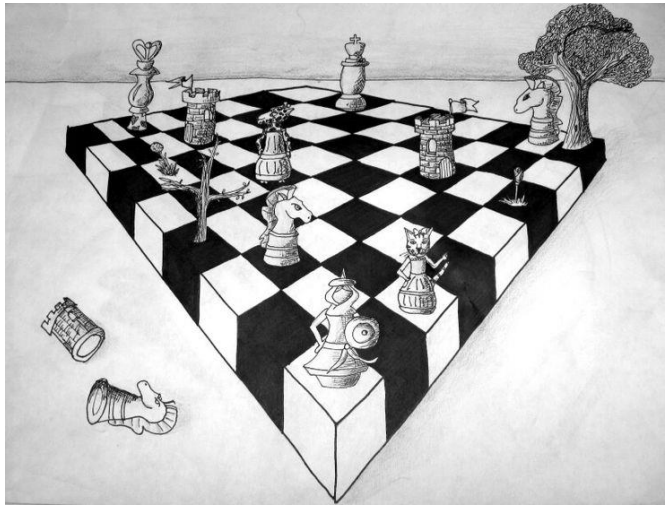
World Space

View Space

Clip Space

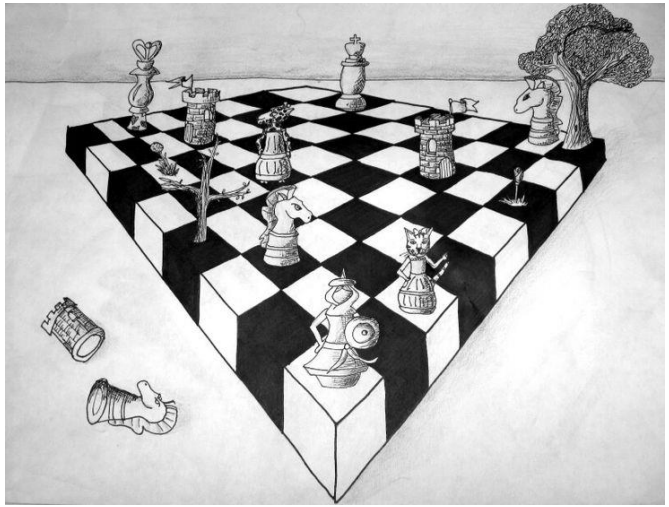
Perspective projection

- What is perspective?

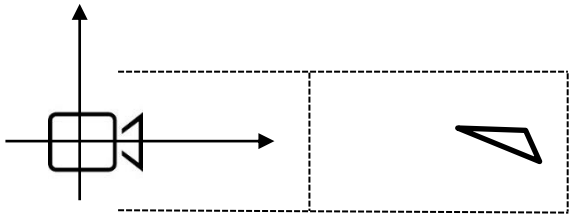


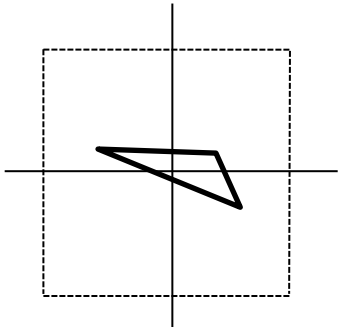
Perspective projection

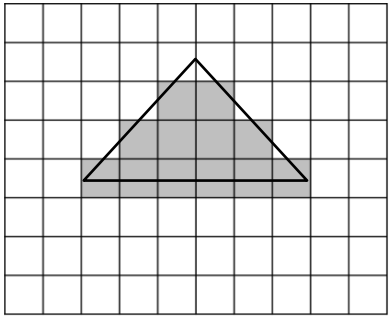
- What is perspective?
- The size of an object is proportional to its distance from the viewpoint.



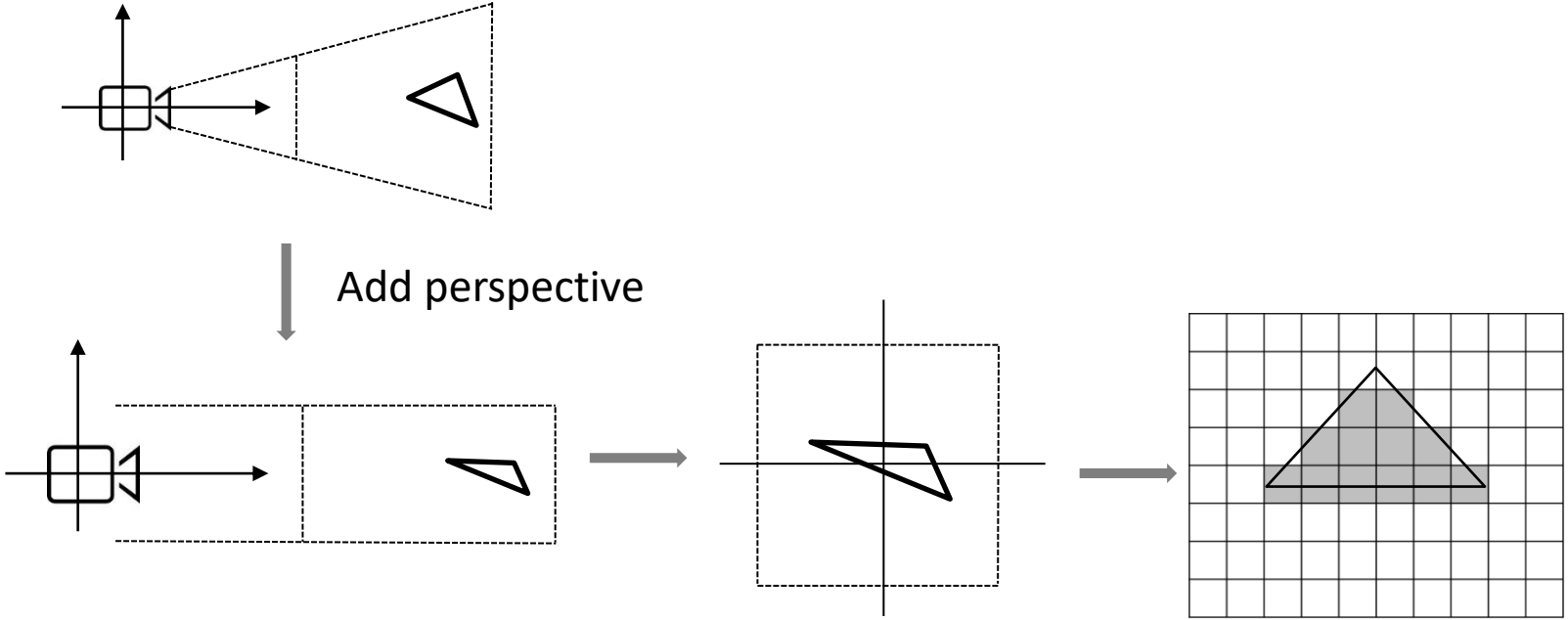
Rendering pipeline



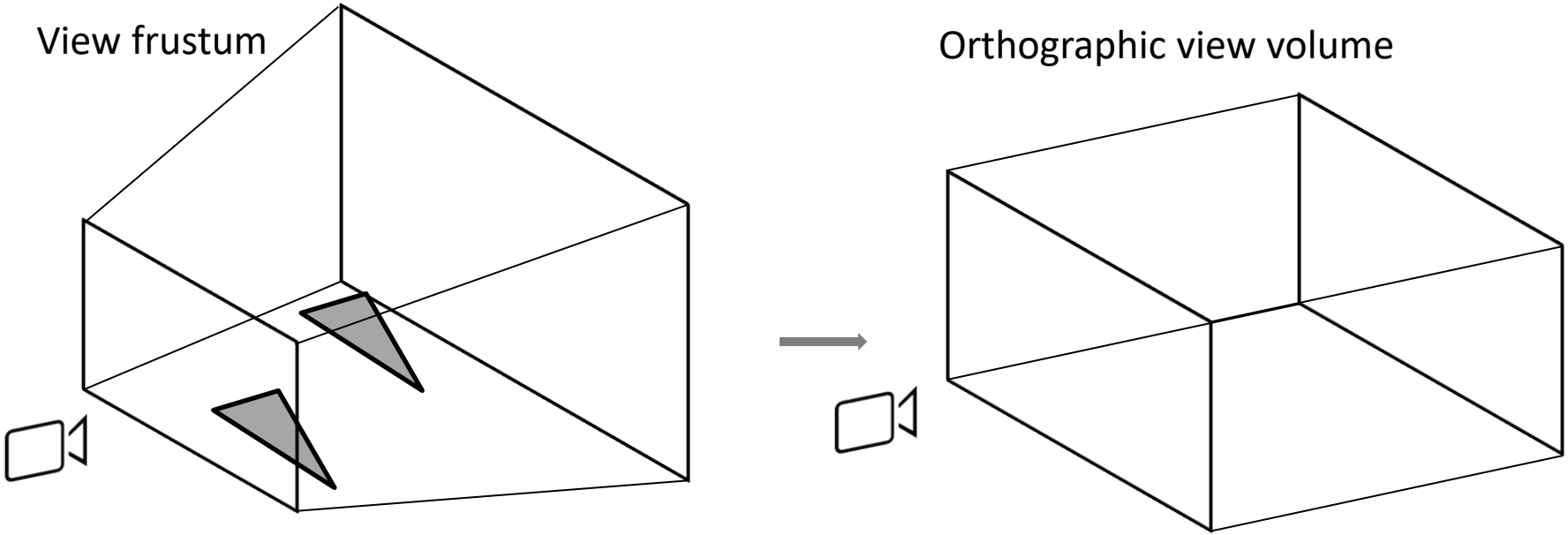




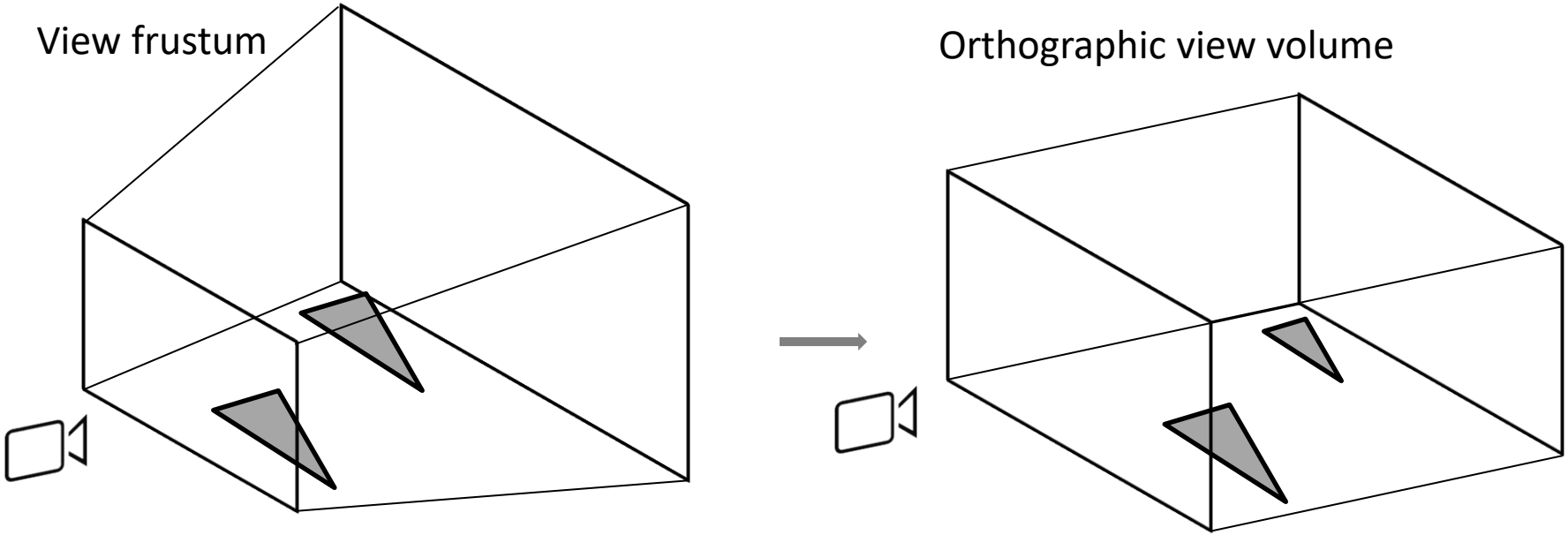
Rendering pipeline

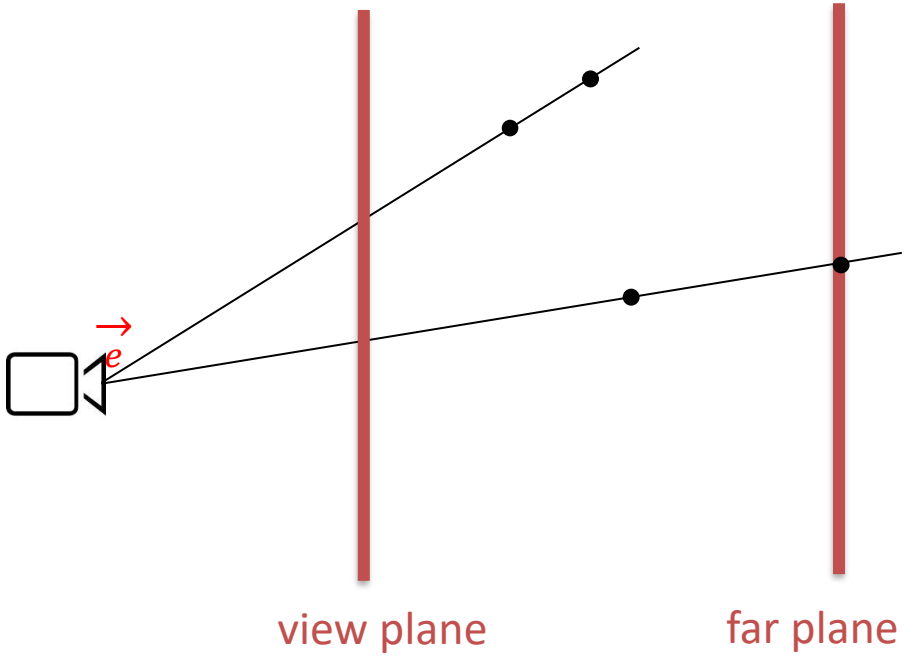


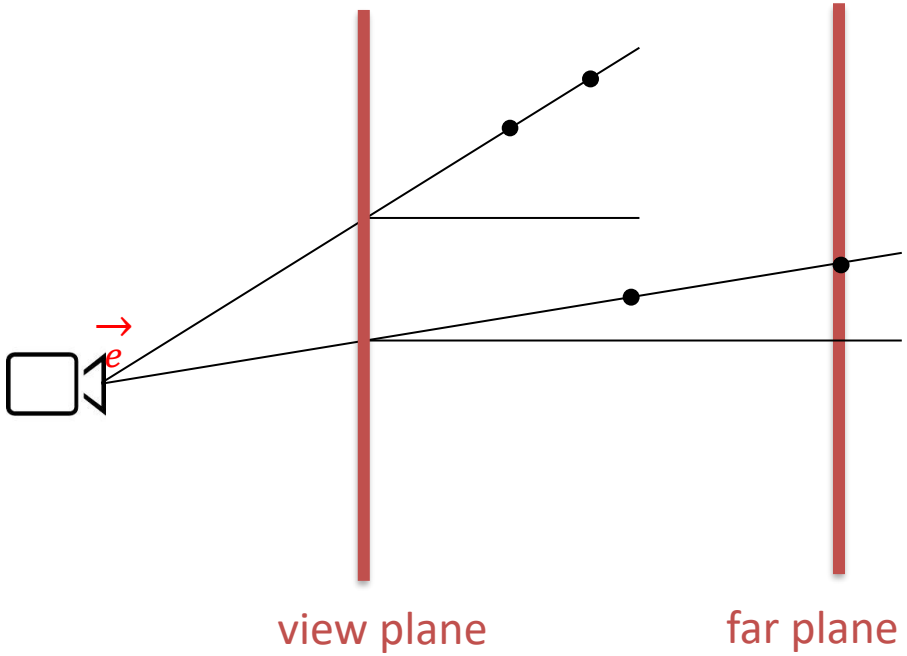
Transformation of the view frustum

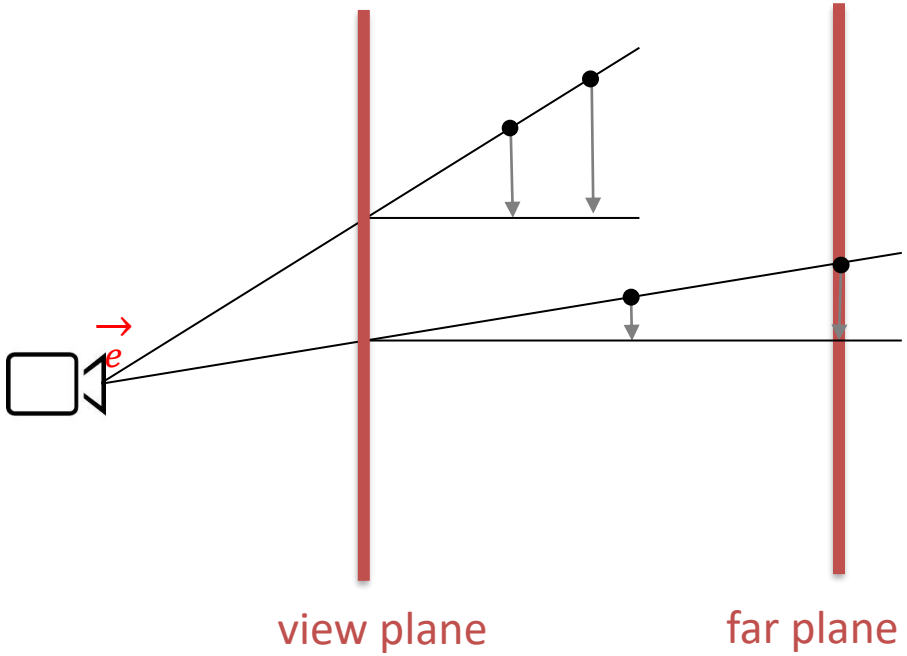


Transformation of the view frustum

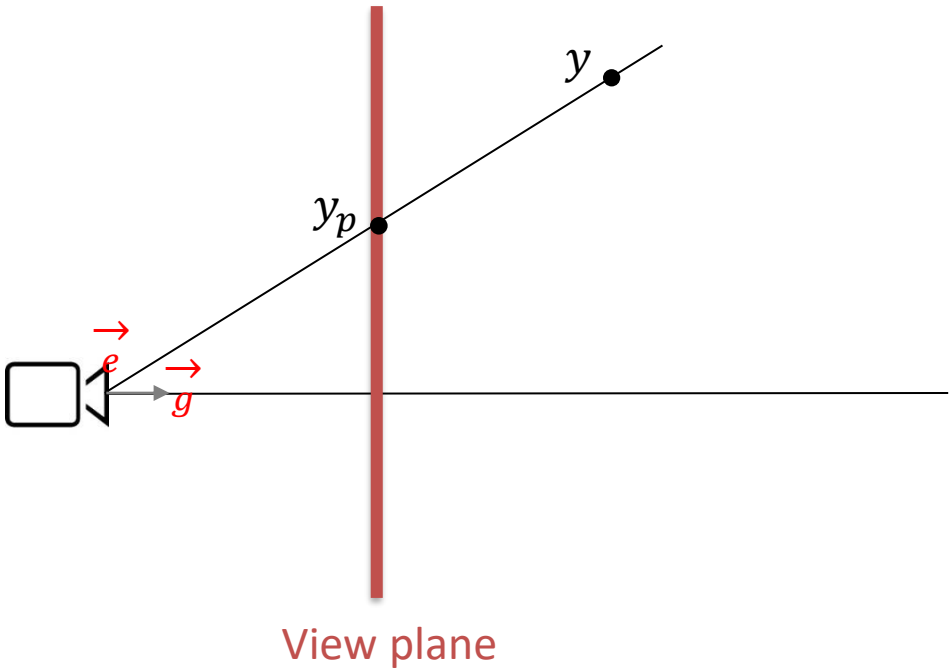



$\vec{e}$
view plane
far plane

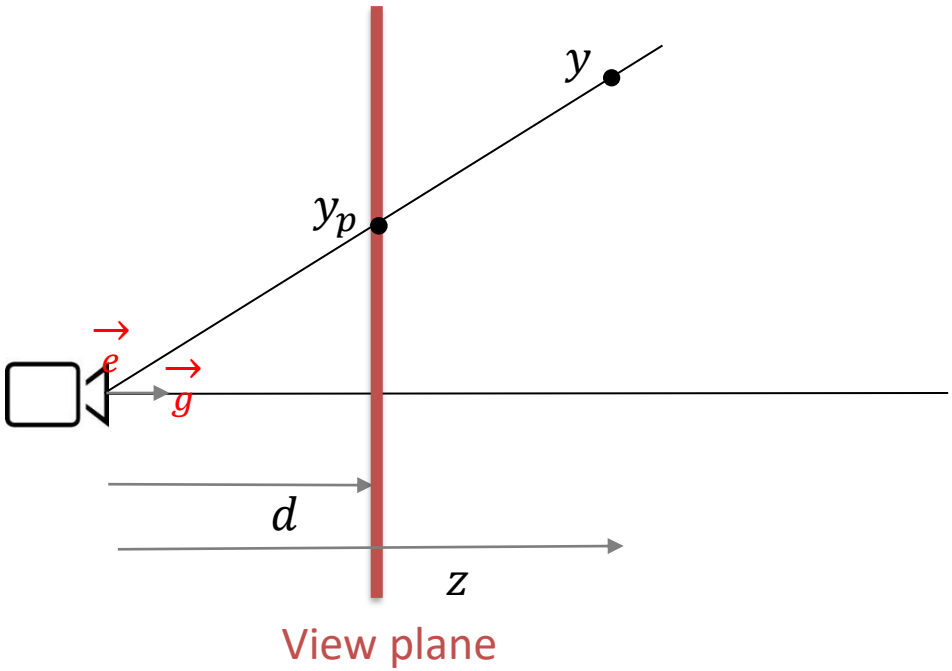

$\vec{e}$
view plane
far plane


$\vec{e}$
view plane
far plane

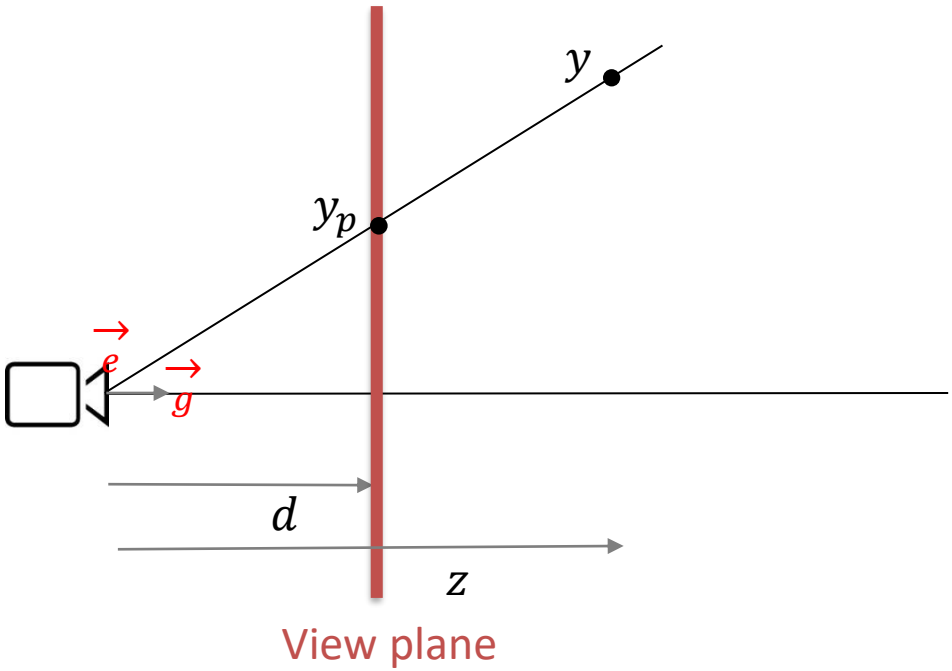
Projection on a plane



Projection on a plane

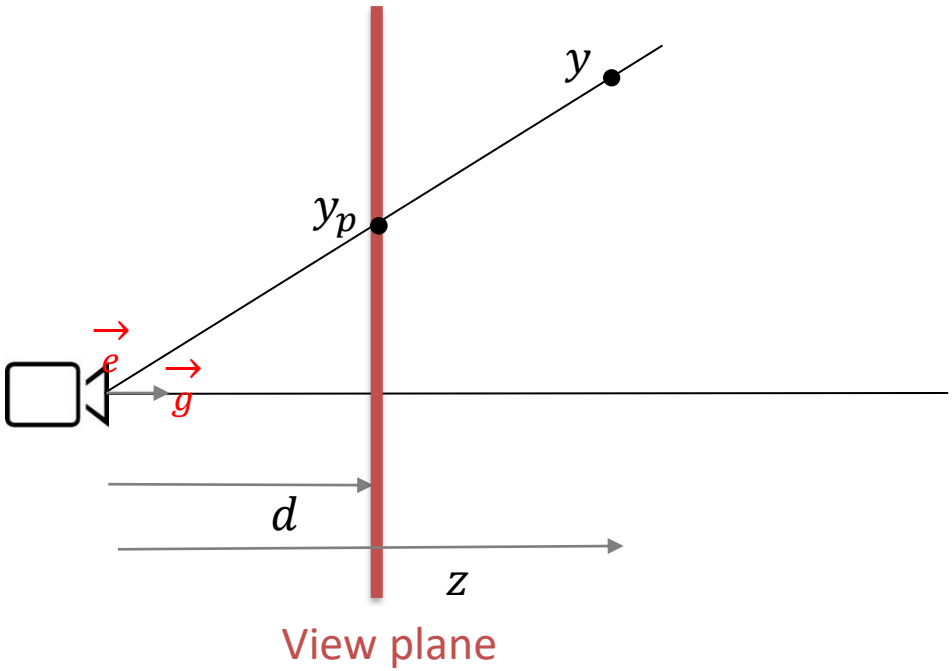


Projection on a plane



$$\frac{y_p}{y} = \frac{d}{z} \quad or$$

Projection on a plane



$$\frac{y_p}{y} = \frac{d}{z} \quad or$$

$$y_p = \frac{d}{z} y$$

Assumptions

- We assume that:
  - We are looking down the negative z-axis
  - We project on the near plane

- So distance d = -n, which means:

  - $x_p = \frac{d}{-z} x$
  - $y_p = \frac{d}{-z} y$

Assumptions

- We assume that:
  - We are looking down the negative z-axis
  - We project on the near plane

- So distance d = -n, which means:

  - $x_p = \frac{d}{-z} x = \frac{n}{z} x$
  - $y_p = \frac{d}{-z} y = \frac{n}{z} y$

Assumptions

- We assume that:
  - We are looking down the negative z-axis
  - We project on the near plane

- So distance d = -n, which means:

  - $x_p = \frac{d}{-z}x = \frac{n}{z}x$
  - $y_p = \frac{d}{-z}y = \frac{n}{z}y$

$$M\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ z \\ 1 \end{bmatrix}$$

Homogeneous coordinates

So far matrix multiplications were linear transformations, meaning we could only obtain transformations such as:

$$x' = a_1 x + b_1 y + c_1 z$$

Homogeneous coordinates

So far matrix multiplications were linear transformations, meaning we could only obtain transformations such as:

$$x' = a_1 x + b_1 y + c_1 z$$

Homogeneous coordinates allow representing points (x, y, z) as (x, y, z, 1) and allow us to use affine transformations, e.g.:

$$x' = a_1 x + b_1 y + c_1 z + d_1$$

Homogeneous coordinates

- In homogeneous coordinates

(x, y, z, 1) represents the point **(x, y, z)**

Additionally, we now allow other points

(x, y, z, w) which specifies the point $\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$

homogeneous

cartesian

Homogeneous coordinates

- Then matrix transformations become:

- $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & T_x \\ a_2 & b_2 & c_2 & T_y \\ a_3 & b_3 & c_3 & T_z \\ a_4 & b_4 & c_4 & w \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 x + b_1 y + c_1 z + T_x \\ a_2 x + b_2 y + c_2 z + T_y \\ a_3 x + b_3 y + c_3 z + T_z \\ a_4 x + b_4 y + c_4 z + w \end{bmatrix}$$

Homogeneous coordinates

- Then matrix transformations become:

- $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & T_x \\ a_2 & b_2 & c_2 & T_y \\ a_3 & b_3 & c_3 & T_z \\ a_4 & b_4 & c_4 & w \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_1x + b_1y + c_1z + T_x \\ a_2x + b_2y + c_2z + T_y \\ a_3x + b_3y + c_3z + T_z \\ a_4x + b_4y + c_4z + w \end{bmatrix}$$

- $$\begin{bmatrix} a_1x + b_1y + c_1z + T_x \\ a_2x + b_2y + c_2z + T_y \\ a_3x + b_3y + c_3z + T_z \\ a_4x + b_4y + c_4z + w \end{bmatrix} \xrightarrow{homogenization}$$

Homogeneous coordinates

- Then matrix transformations become:

- $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & T_x \\ a_2 & b_2 & c_2 & T_y \\ a_3 & b_3 & c_3 & T_z \\ a_4 & b_4 & c_4 & w \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_1x + b_1y + c_1z + T_x \\ a_2x + b_2y + c_2z + T_y \\ a_3x + b_3y + c_3z + T_z \\ a_4x + b_4y + c_4z + w \end{bmatrix}$$

- $$\begin{bmatrix} a_1x + b_1y + c_1z + T_x \\ a_2x + b_2y + c_2z + T_y \\ a_3x + b_3y + c_3z + T_z \\ a_4x + b_4y + c_4z + w \end{bmatrix} \xrightarrow{homogenization}$$

Homogeneous coordinates

- Then matrix transformations become:

- $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & T_x \\ a_2 & b_2 & c_2 & T_y \\ a_3 & b_3 & c_3 & T_z \\ a_4 & b_4 & c_4 & w \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_1x + b_1y + c_1z + T_x \\ a_2x + b_2y + c_2z + T_y \\ a_3x + b_3y + c_3z + T_z \\ a_4x + b_4y + c_4z + w \end{bmatrix}$$

- $$\begin{bmatrix} a_1x + b_1y + c_1z + T_x \\ a_2x + b_2y + c_2z + T_y \\ a_3x + b_3y + c_3z + T_z \\ a_4x + b_4y + c_4z + w \end{bmatrix} \xrightarrow{homogenization} \begin{bmatrix} \frac{a_1x+b_1y+c_1z+T_x}{a_4x+b_4y+c_4z+w} \\ \frac{a_2x+b_2y+c_2z+T_y}{a_4x+b_4y+c_4z+w} \\ \frac{a_3x+b_3y+c_3z+T_z}{a_4x+b_4y+c_4z+w} \\ 1 \end{bmatrix}$$

Homogeneous coordinates

- Then matrix transformations become:

- $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & T_x \\ a_2 & b_2 & c_2 & T_y \\ a_3 & b_3 & c_3 & T_z \\ a_4 & b_4 & c_4 & w \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_1x + b_1y + c_1z + T_x \\ a_2x + b_2y + c_2z + T_y \\ a_3x + b_3y + c_3z + T_z \\ a_4x + b_4y + c_4z + w \end{bmatrix}$$

- $$\begin{bmatrix} a_1x + b_1y + c_1z + T_x \\ a_2x + b_2y + c_2z + T_y \\ a_3x + b_3y + c_3z + T_z \\ a_4x + b_4y + c_4z + w \end{bmatrix} \xrightarrow{homogenization} \begin{bmatrix} \frac{a_1x+b_1y+c_1z+T_x}{a_4x+b_4y+c_4z+w} \\ \frac{a_2x+b_2y+c_2z+T_y}{a_4x+b_4y+c_4z+w} \\ \frac{a_3x+b_3y+c_3z+T_z}{a_4x+b_4y+c_4z+w} \\ 1 \end{bmatrix}$$

Homogeneous coordinates Cartesian coordinates

Transformation of the view frustum

Calculation: $$\begin{bmatrix} \dfrac{a_1x + b_1y + c_1z + T_x}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_2x + b_2y + c_2z + T_y}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_3x + b_3y + c_3z + T_z}{a_4x + b_4y + c_4z + w} \\ 1 \end{bmatrix}$$

Goal: $$\begin{bmatrix} \dfrac{nx}{z} \\ \dfrac{ny}{z} \\ z \\ 1 \end{bmatrix}$$

Transformation of the view frustum

Calculation: $$\begin{bmatrix} \dfrac{a_1x + b_1y + c_1z + T_x}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_2x + b_2y + c_2z + T_y}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_3x + b_3y + c_3z + T_z}{a_4x + b_4y + c_4z + w} \\ 1 \end{bmatrix}$$

Goal: $$\begin{bmatrix} \dfrac{nx}{z} \\ \dfrac{ny}{z} \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Fourth row

Transformation of the view frustum

Calculation: $$\begin{bmatrix} \dfrac{a_1x + b_1y + c_1z + T_x}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_2x + b_2y + c_2z + T_y}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_3x + b_3y + c_3z + T_z}{a_4x + b_4y + c_4z + w} \\ 1 \end{bmatrix}$$

Goal: $$\begin{bmatrix} \dfrac{nx}{z} \\ \dfrac{ny}{z} \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Fourth row

Transformation of the view frustum

Calculation: $$\begin{bmatrix} \dfrac{a_1x + b_1y + c_1z + T_x}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_2x + b_2y + c_2z + T_y}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_3x + b_3y + c_3z + T_z}{a_4x + b_4y + c_4z + w} \\ 1 \end{bmatrix}$$

Goal: $$\begin{bmatrix} \dfrac{nx}{z} \\ \dfrac{ny}{z} \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

First and second row

Transformation of the view frustum

Calculation: $$\begin{bmatrix} \dfrac{a_1x + b_1y + c_1z + T_x}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_2x + b_2y + c_2z + T_y}{a_4x + b_4y + c_4z + w} \\ \dfrac{a_3x + b_3y + c_3z + T_z}{a_4x + b_4y + c_4z + w} \\ 1 \end{bmatrix}$$

Goal: $$\begin{bmatrix} \dfrac{nx}{z} \\ \dfrac{ny}{z} \\ z \\ 1 \end{bmatrix}$$

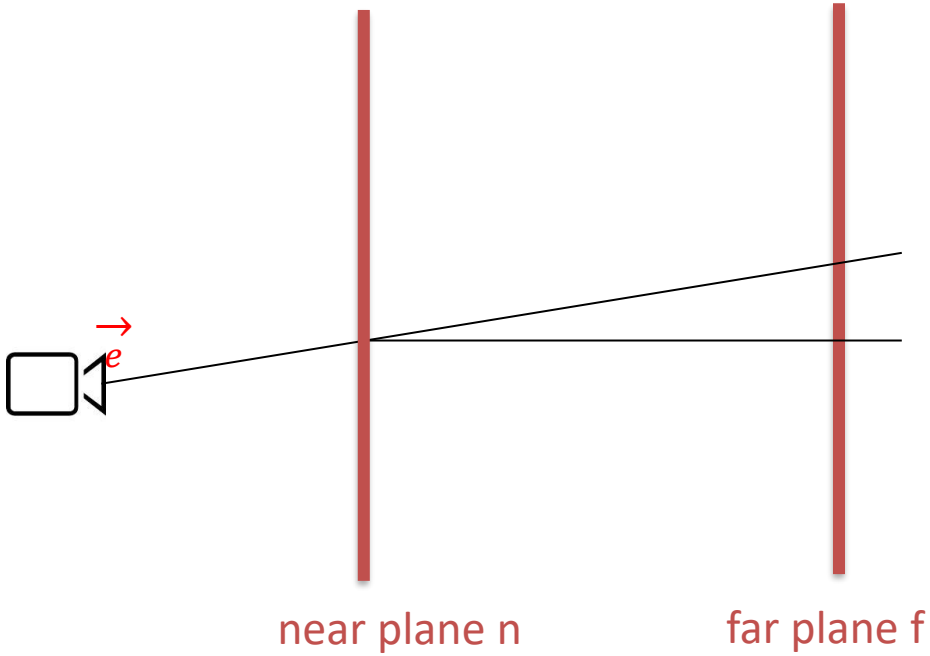
$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

First and second row

What about the third row

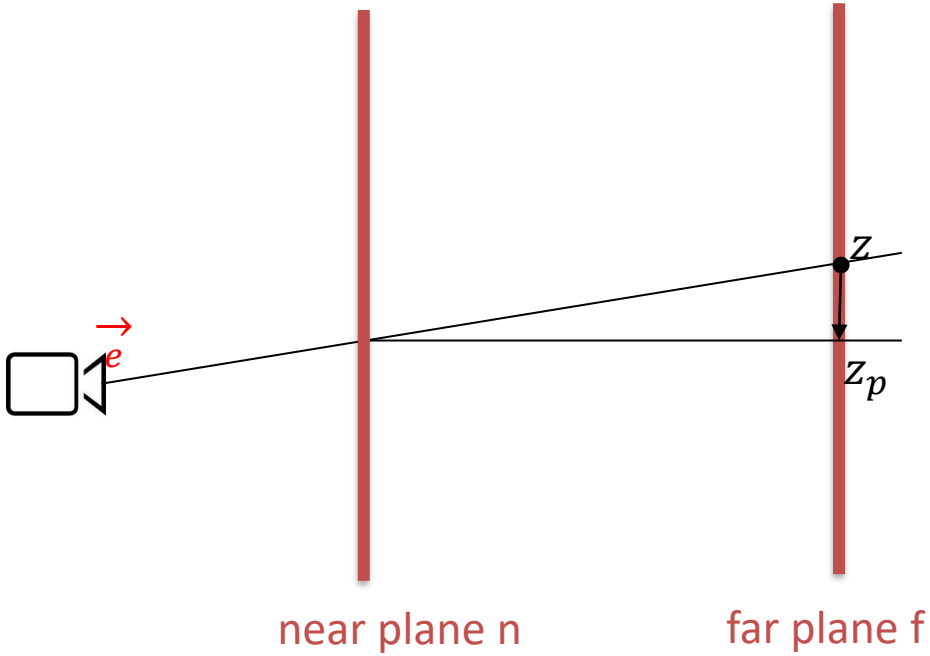
$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ z \\ 1 \end{bmatrix}$$

Transformation of the view frustum



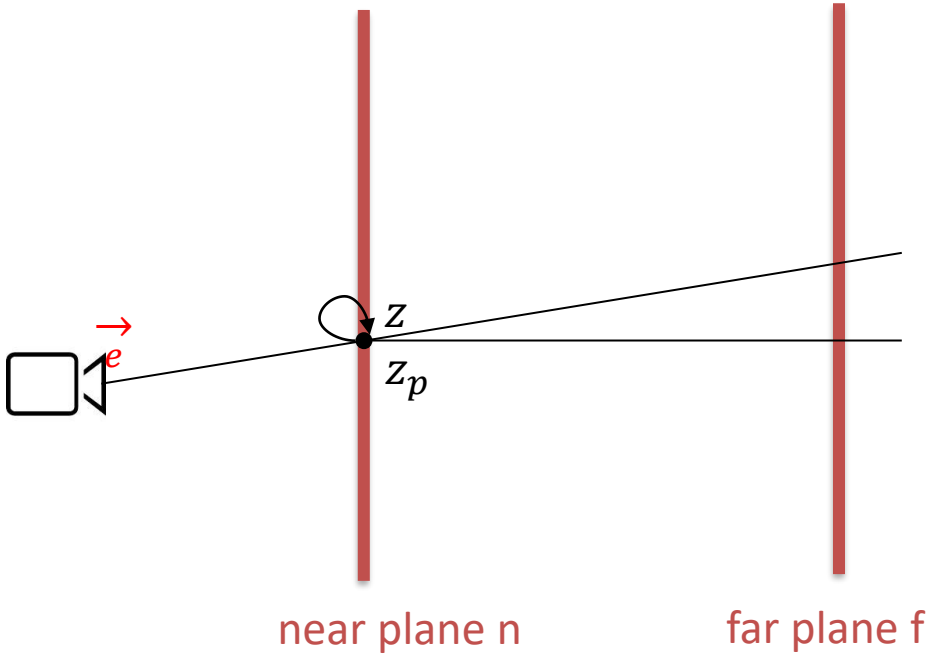
Transformation of the view frustum

$$z_p = f$$



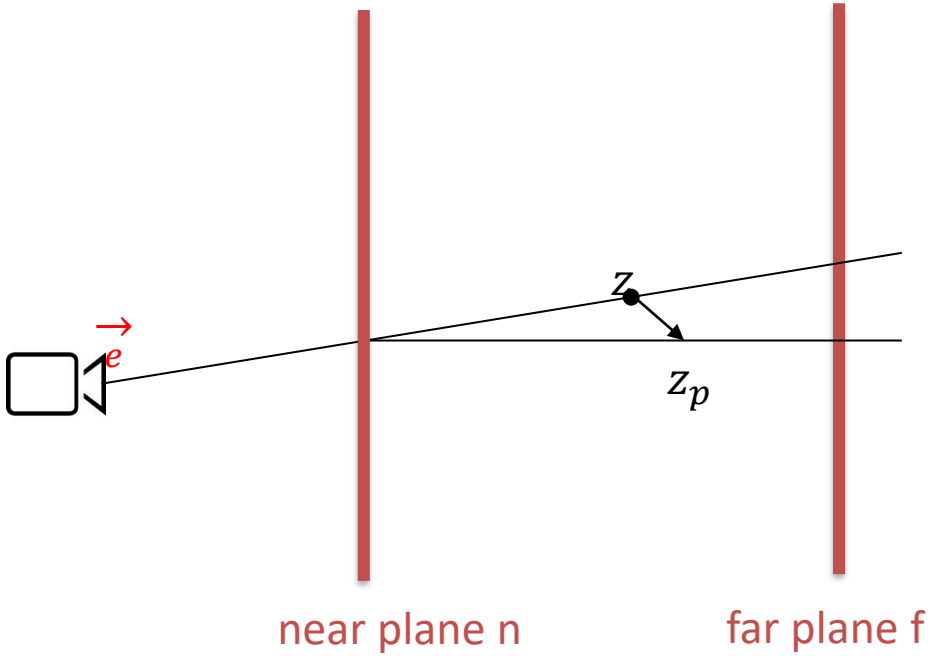
Transformation of the view frustum

$$z_p = n$$



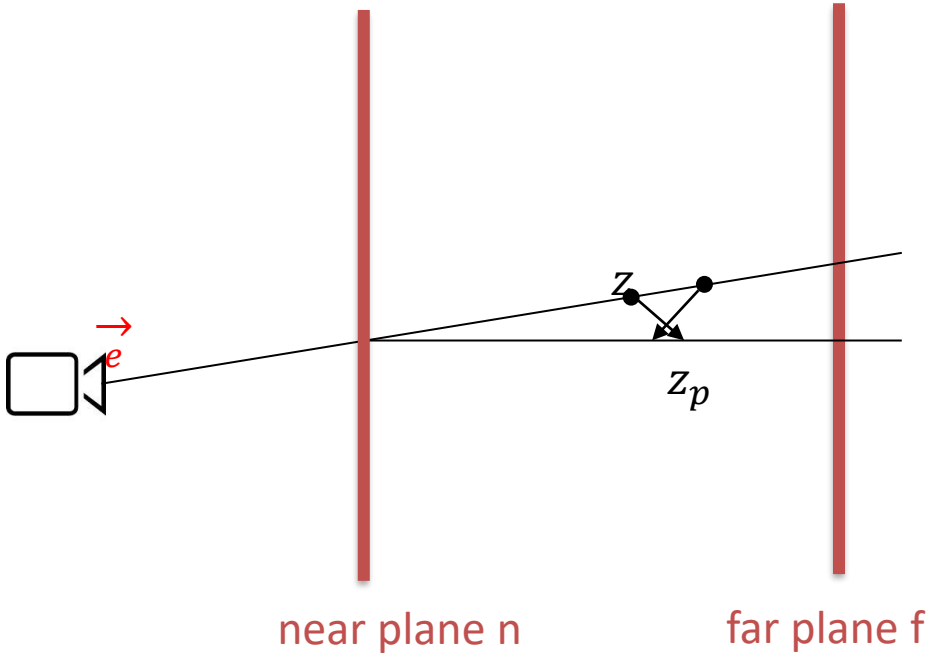
Transformation of the view frustum

$z_p \in [n, f]$



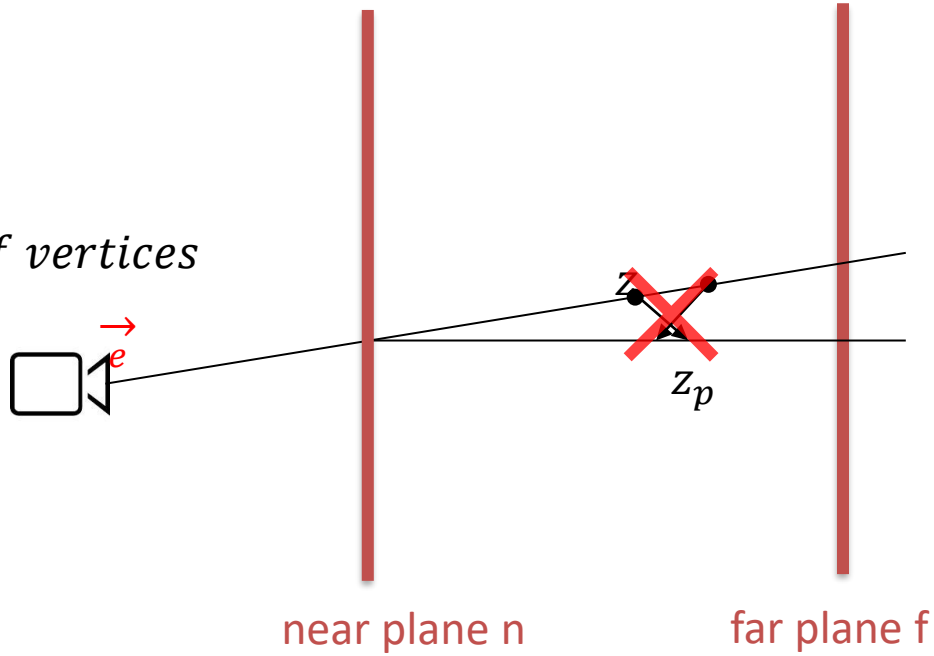
Transformation of the view frustum

$z_p \in [n, f]$



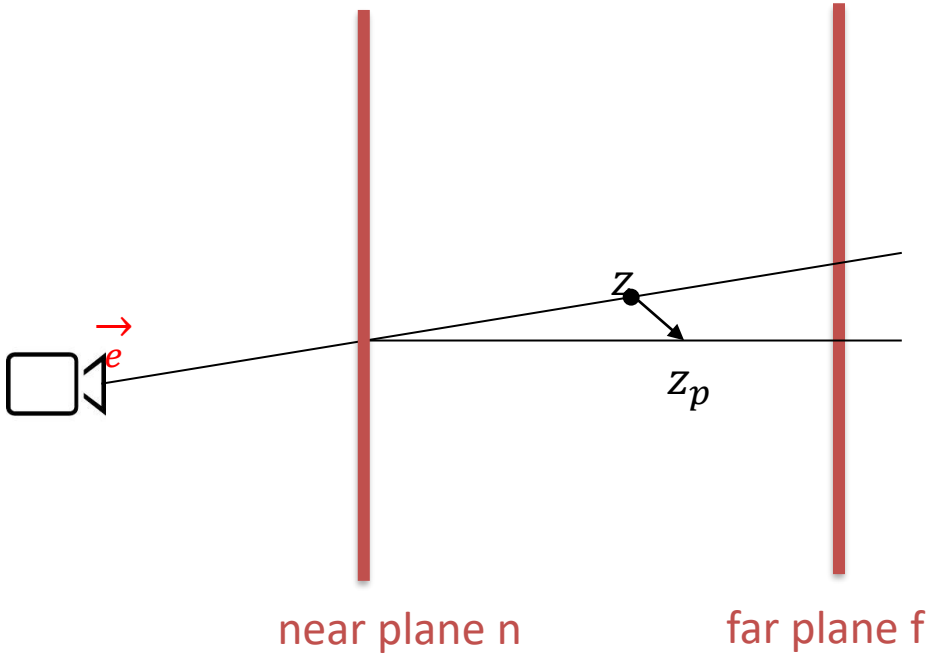
Transformation of the view frustum

z_p *has to preserve order of vertices*



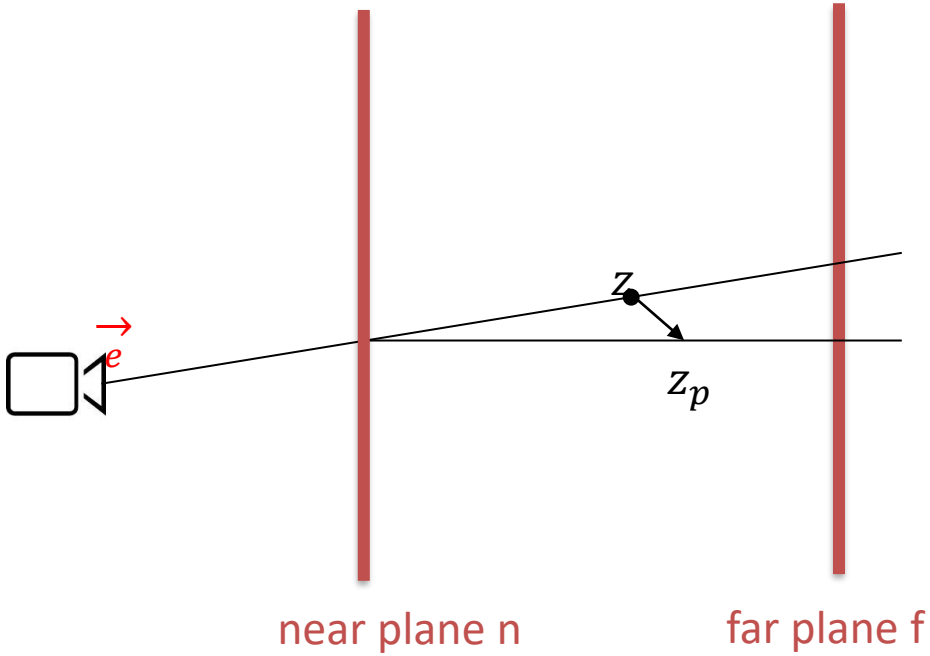
Transformation of the view frustum

$$z_p = \frac{nf}{z} \in [n, f]$$



Transformation of the view frustum

$$z_p = n + f - \frac{nf}{z}$$



Preserving order of vertices

- $z_p = n + f - \frac{fn}{z}$
- Order along the z-axis is preserved if $0 > n \geq z_1 > z_2 \geq f$ to $z_{1p} > z_{2p}$

Preserving order of vertices

- $z_p = n + f - \frac{fn}{z}$
- Order along the z-axis is preserved if $0 > n \geq z_1 > z_2 \geq f$ to $z_{1p} > z_{2p}$

- $z_{1p} - z_{2p} = \frac{fn}{z_2} - \frac{fn}{z_1}$

Preserving order of vertices

- $z_p = n + f - \frac{fn}{z}$
- Order along the z-axis is preserved if $0 > n \geq z_1 > z_2 \geq f$ to $z_{1p} > z_{2p}$

- $z_{1p} - z_{2p} = \frac{fn}{z_2} - \frac{fn}{z_1} = \frac{(z_1 - z_2)fn}{z_1 z_2}$

Preserving order of vertices

- $z_p = n + f - \frac{fn}{z}$
- Order along the z-axis is preserved if $0 > n \geq z_1 > z_2 \geq f$ to $z_{1p} > z_{2p}$
- $z_{1p} - z_{2p} = \frac{fn}{z_2} - \frac{fn}{z_1} = \frac{(z_1 - z_2)fn}{z_1 z_2}$
- Because $f, z_1, z_2, n < 0$ it follows $\frac{fn}{z_1 z_2} > 0$ and $z_1 > z_2$ meaning $z_1 - z_2 > 0$

Preserving order of vertices

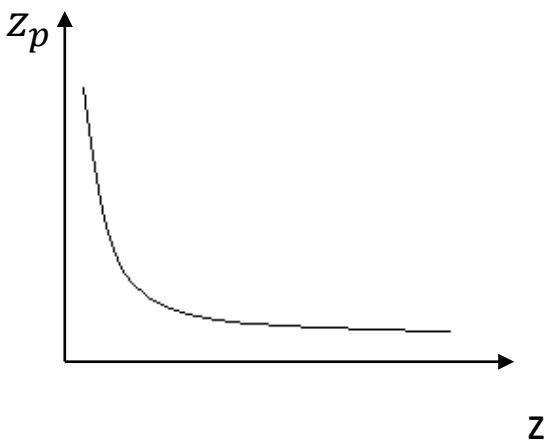
- $z_p = n + f - \frac{fn}{z}$
- Order along the z-axis is preserved if $0 > n \geq z_1 > z_2 \geq f$ to $z_{1p} > z_{2p}$
- $z_{1p} - z_{2p} = \frac{fn}{z_2} - \frac{fn}{z_1} = \frac{(z_1 - z_2)fn}{z_1 z_2}$
- Because $f, z_1, z_2, n < 0$ it follows $\frac{fn}{z_1 z_2} > 0$ and $z_1 > z_2$ meaning $z_1 - z_2 > 0$
- So $z_{1p} > z_{2p}$

Transformation of the view frustum

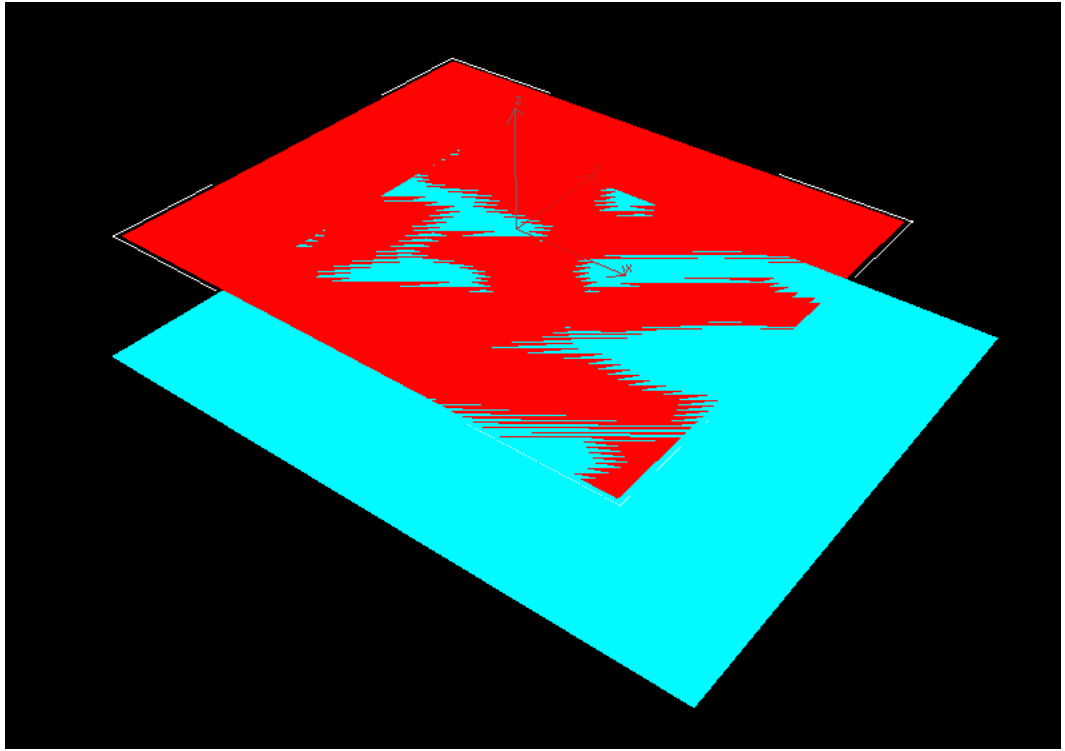
$$z_p = n + f - \frac{fn}{z} \quad \rightarrow \quad \boldsymbol{z_p \sim \frac{1}{z}}$$

Transformation of the view frustum

$$z_p = n + f - \frac{fn}{z} \quad \rightarrow \quad \boldsymbol{z_p \sim \frac{1}{z}}$$



Z-fighting



Matrix P

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

Matrix P

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n+f-\frac{nf}{z} \\ 1 \end{bmatrix}$$

Perspective transformation matrix

- Our final perspective transformation matrix M_{pers} is then:

- $M_{per} = M_{orth}P$

Perspective transformation matrix

- Our final perspective transformation matrix M_{pers} is then

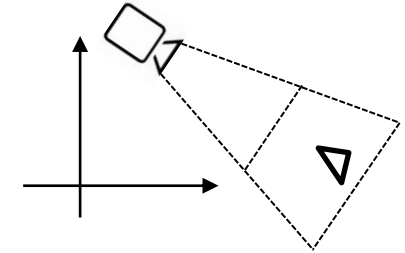
- $M_{per} = M_{orth}P = M_{orth}\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$

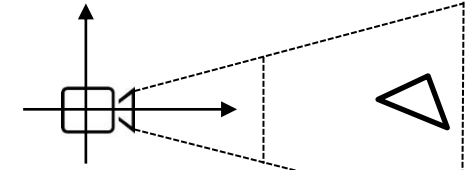
Perspective transformation matrix

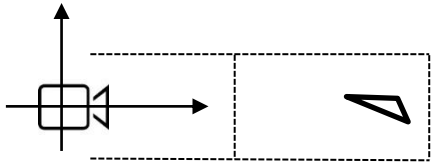
- Our final perspective transformation matrix M_{pers} is then

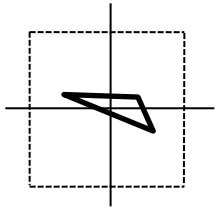
- $$M_{per} = M_{orth}P = M_{orth}\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\ 0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

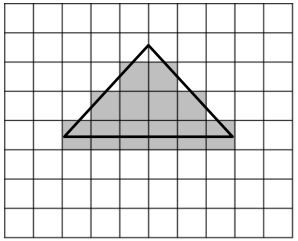
Rendering pipeline





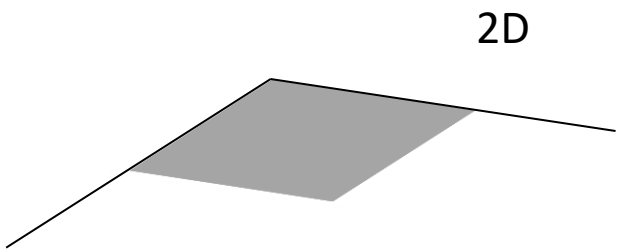




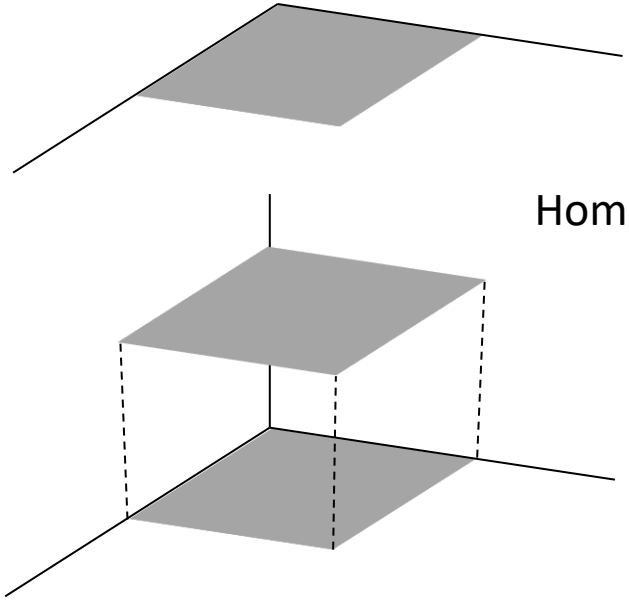


- Rendering pipeline with **orthographic** projection:
- $M_{vp}M_{ort}M_{cam}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$
- Rendering pipeline with **perspective** projection:
- $M_{vp}M_{per}M_{cam}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$

HC enable Translation

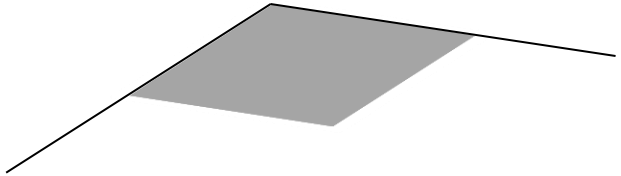


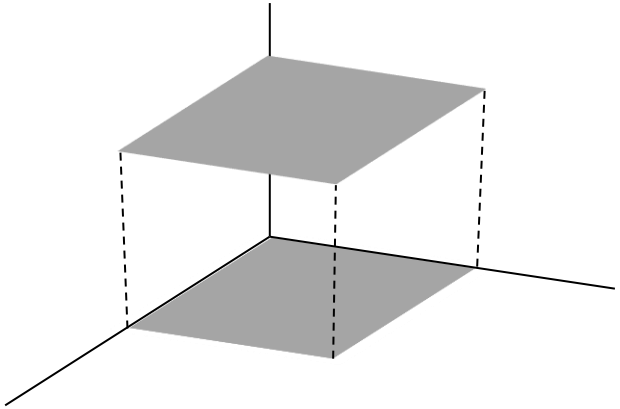
HC enable Translation



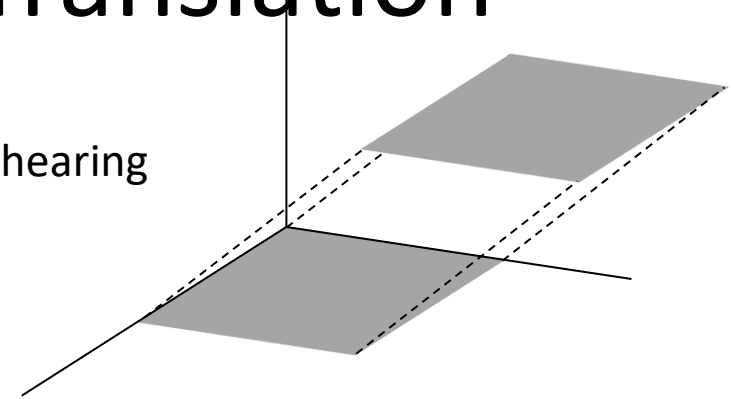
Homogeneous coordinates in 3D

HC enable Translation

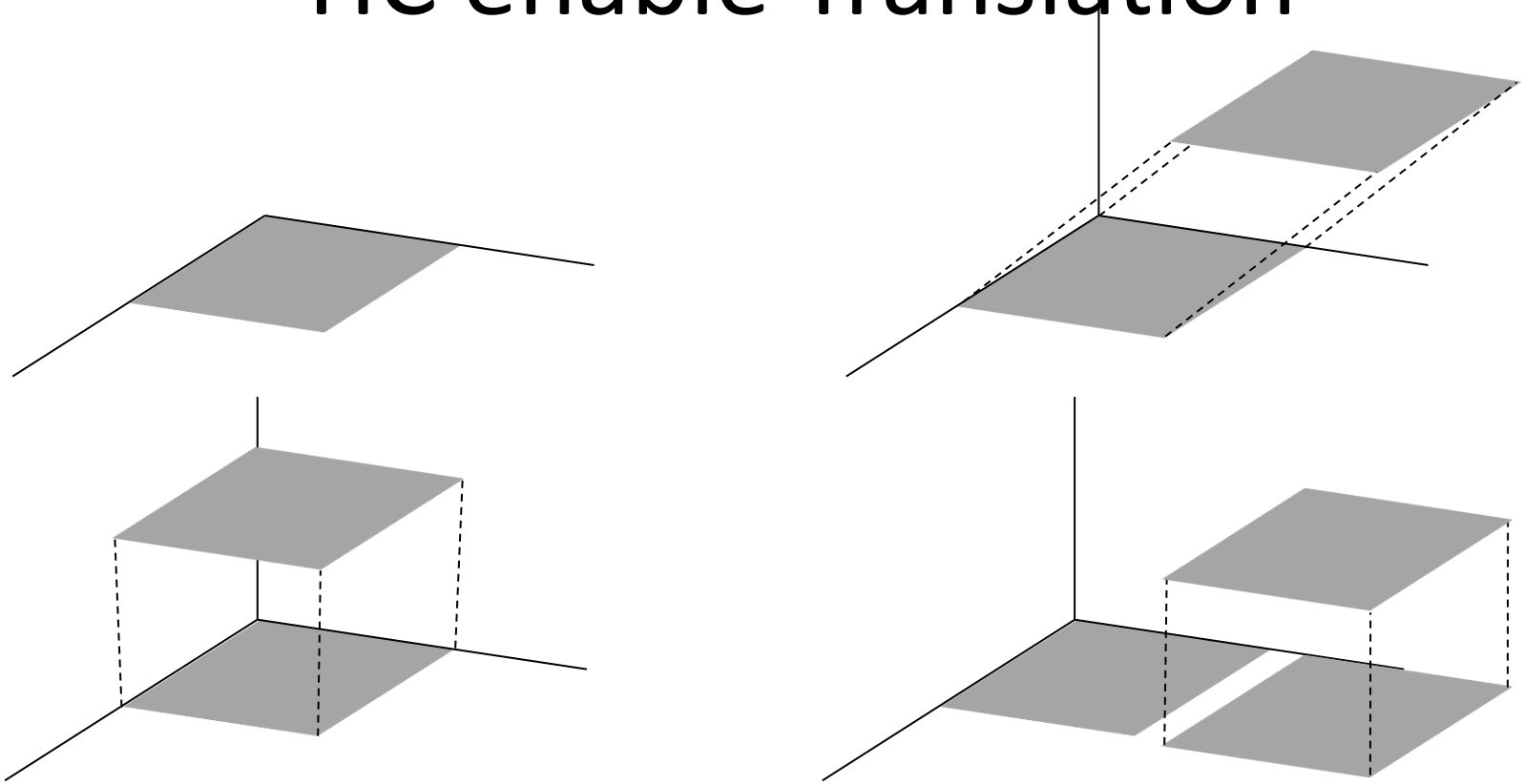




Shearing

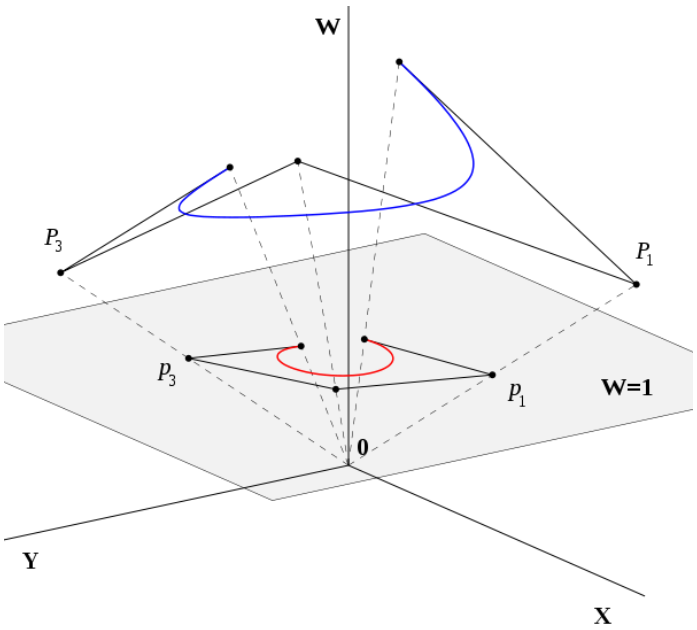


HC enable Translation



Translation in 2D

HC enable Perspective



https://en.wikipedia.org/wiki/Homogeneous_coordinates

Drawing on the Display

- The following pseudo-code illustrates how to draw a line between two points a and b

```
compute M_vp
compute M_per
compute M_cam
M = M_vp M_per M_cam

for each line segment (a, b) do
    p = M a
    q = M b
    drawline(p_x/p_w, p_y/p_w, q_x/q_w, q_y/q_w)
```