View Transformations

Acknowledgements: Daniele Panozzo

Linear Transformations

- Definition: $f: V \rightarrow W, \mathrm{u}, \mathrm{v} \in \mathrm{V}$

$$f(u+v) = f(u) + f(v)$$

$$f(cu) = cf(u)$$

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

2D Linear Transformations

- Each 2D linear map can be represented by a unique 2×2 matrix

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

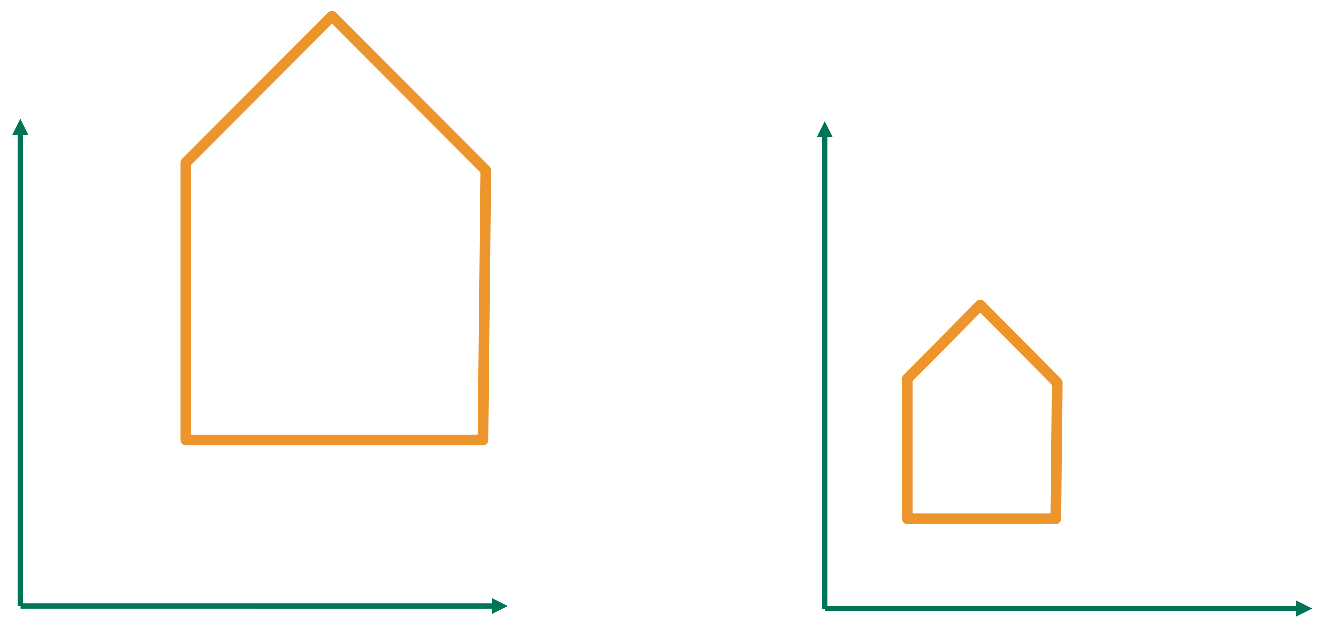
- Concatenation of mappings corresponds to multiplication of matrices

$$L_2(L_1(\mathbf{x})) \; = \; \mathbf{L}_2 \, \mathbf{L}_1 \, \mathbf{x}$$

- Linear transformations are very common in computer graphics!

2D Scaling

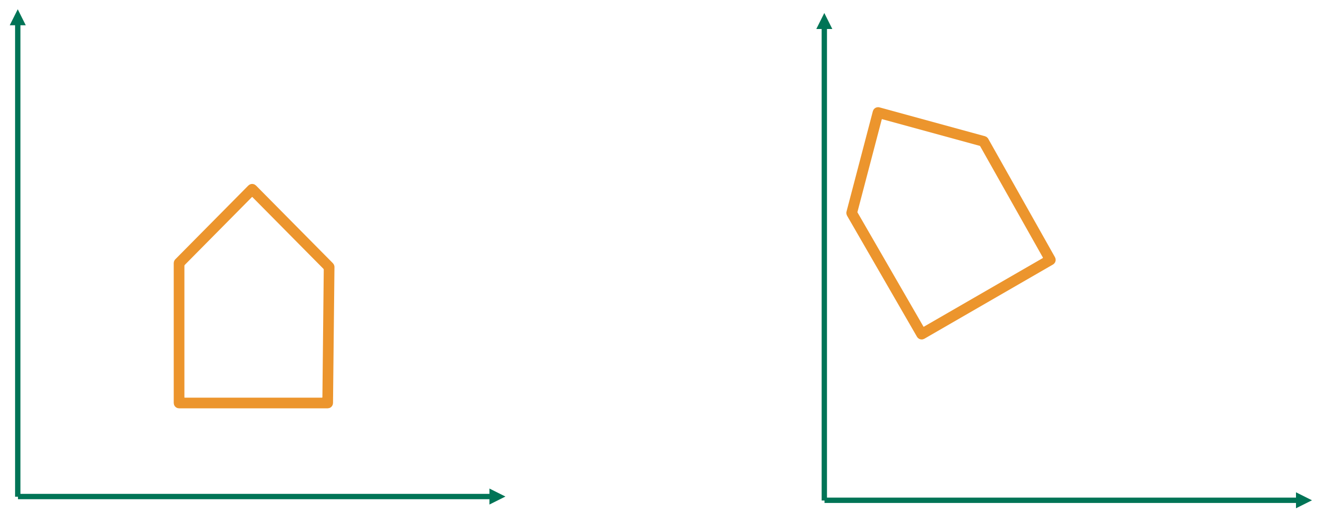
- Scaling $$\begin{pmatrix}x'\\y'\end{pmatrix} = \underbrace{\begin{pmatrix}s_x & 0\\ 0 & s_y\end{pmatrix}}_{\mathbf{S}(s_x,s_y)} \cdot \begin{pmatrix}x\\y\end{pmatrix}$$



$\mathbf{S}(0.5, 0.5)$

2D Rotation

- Rotation $\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}}_{\mathbf{R}(\alpha)} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$



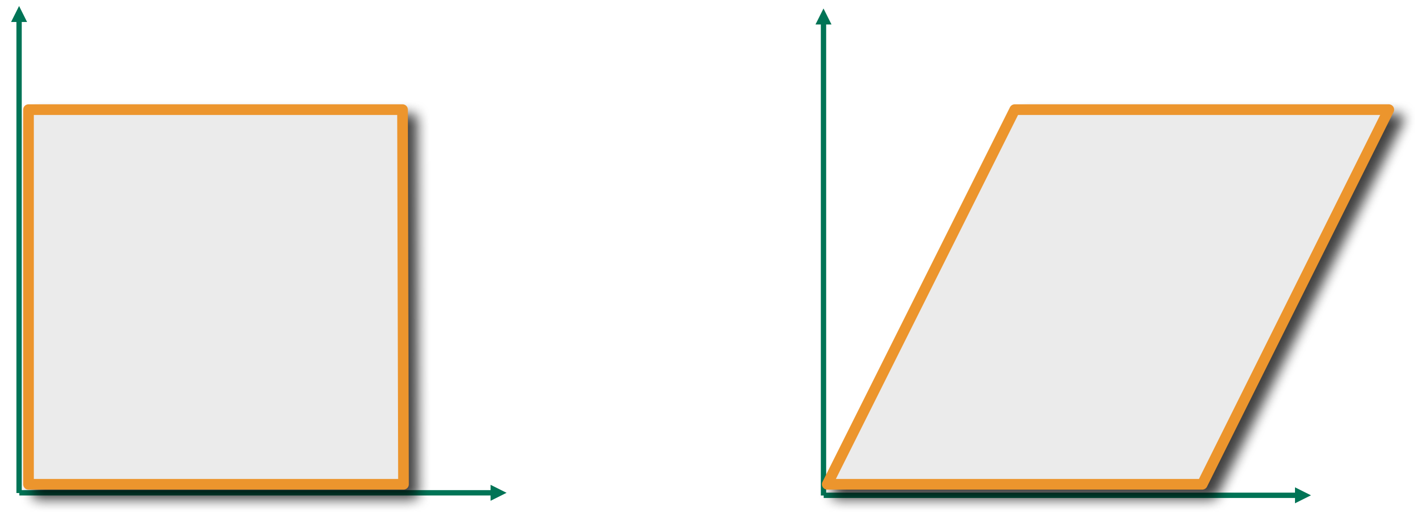
$\mathbf{R}(20°)$

Special case: $\mathbf{R}(90) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

2D Shearing

- Shear along x-axis

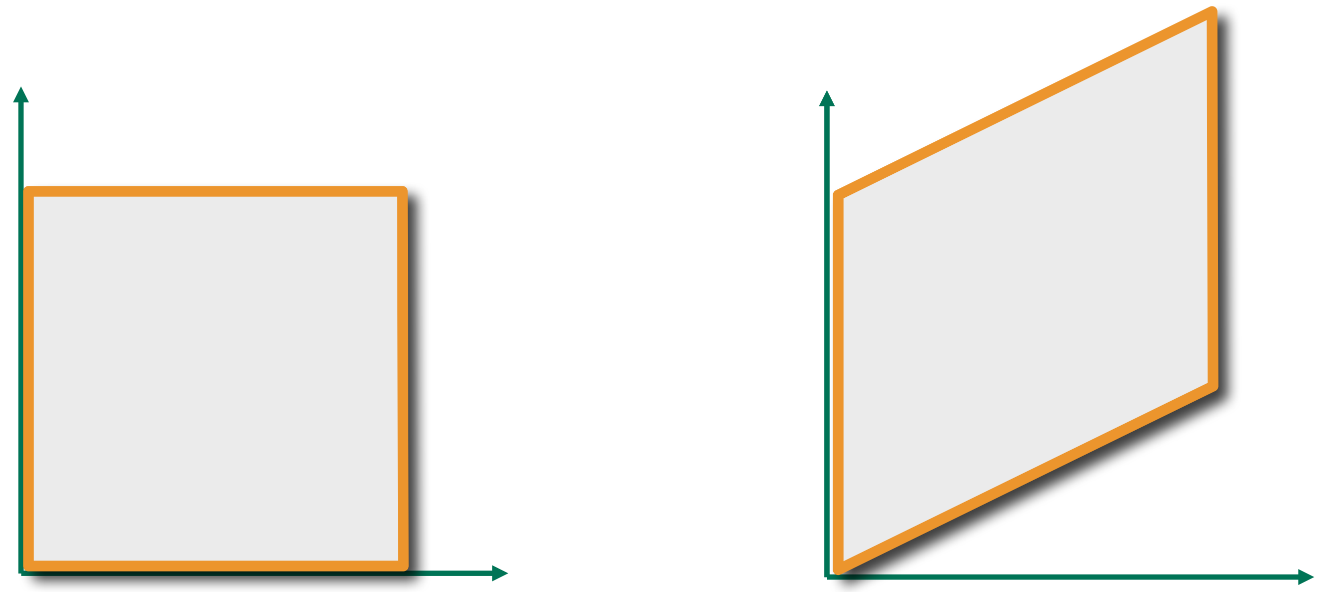
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}}_{\mathbf{H}_x(a)} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$



$\mathbf{H}_x(0.5)$

- Shear along y-axis

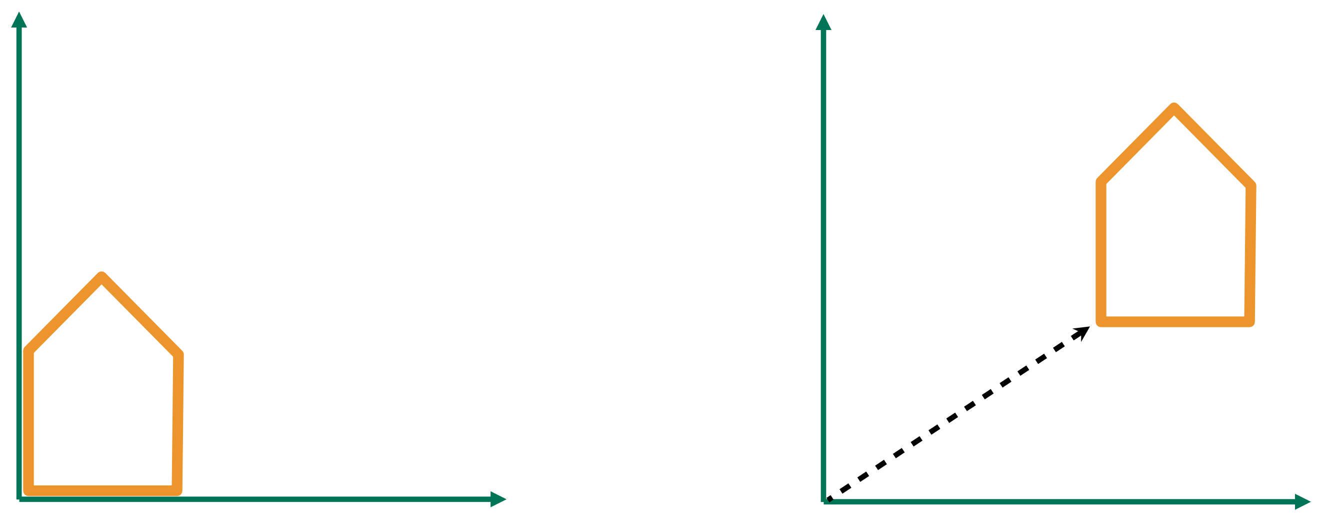
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ b & 1 \end{pmatrix}}_{\mathbf{H}_y(b)} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$



$\mathbf{H}_y(0.5)$

2D Translation

- Translation $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$



- Matrix representation? $\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{T}(t_x, t_y) \cdot \begin{pmatrix} x \\ y \end{pmatrix}$

Affine Transformations

- Translation is not linear, but it is ***affine***
  - Origin is no longer a fixed point
- Affine map = linear map + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \mathbf{Lx} + \mathbf{t}$$

- Is there a matrix representation for affine transformations?
  - We would like to handle all transformations in a unified framework -> simpler to code and easier to optimize!

Homogenous Coordinates

- Add a third coordinate (*w-coordinate*)
  - 2D point = $(x, y, 1)^T$
  - 2D vector = $(x, y, 0)^T$

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

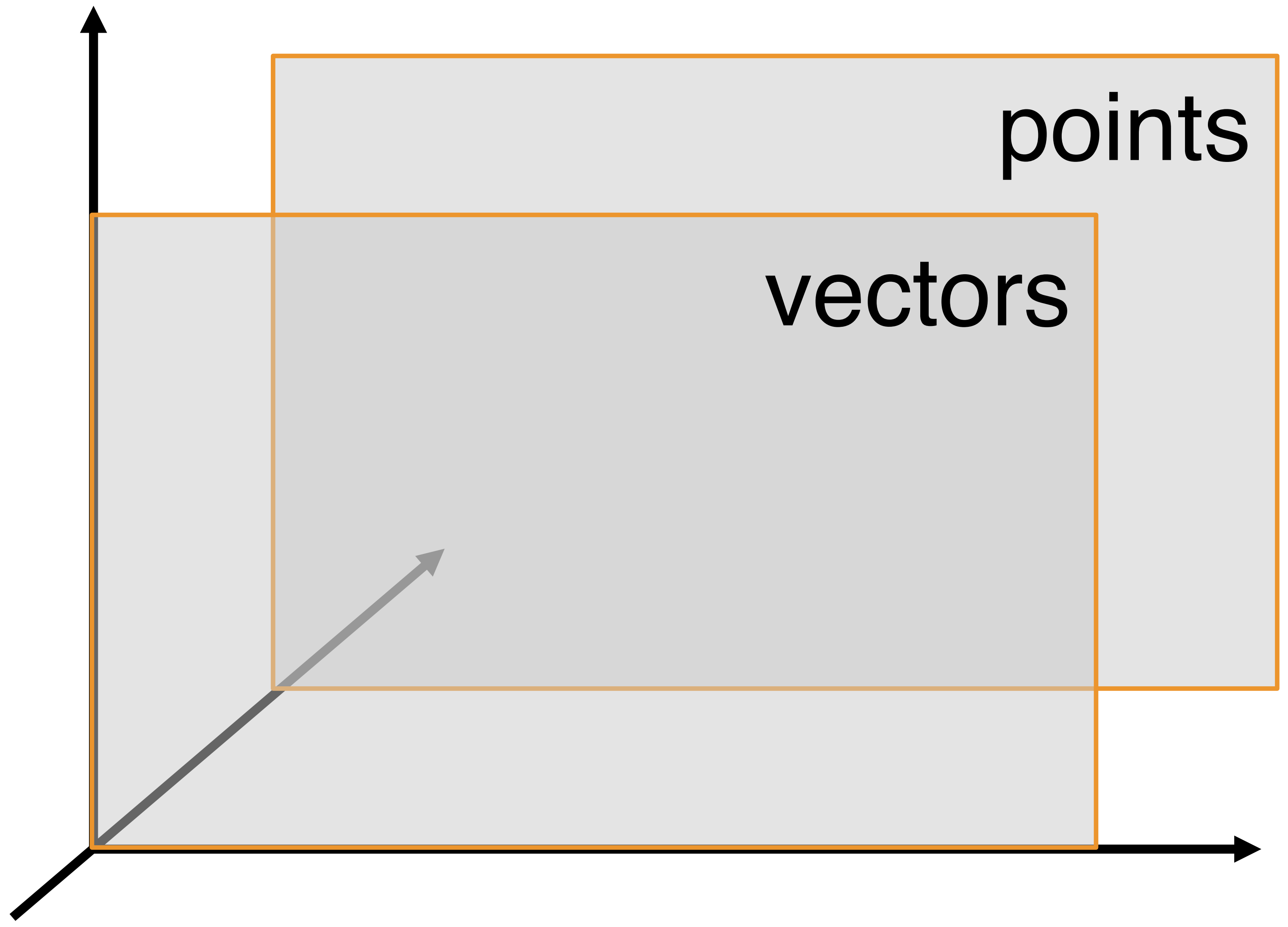
- Matrix representation of translations

Homogenous Coordinates

- Valid operation if the resulting w-coordinate is 1 or 0
    - vector + vector= vector
    - point - point = vector
    - point + vector= point
    - point + point = ???

Homogenous Coordinates

- Geometric interpretation: 2 hyperplanes in $\mathbf{R}^3$



Affine Transformations

- Affine map = linear map + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

- Using homogenous coordinates:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2D Transformations

- Scale

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

- Rotation

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

- Translation

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

Concatenation of Transformations

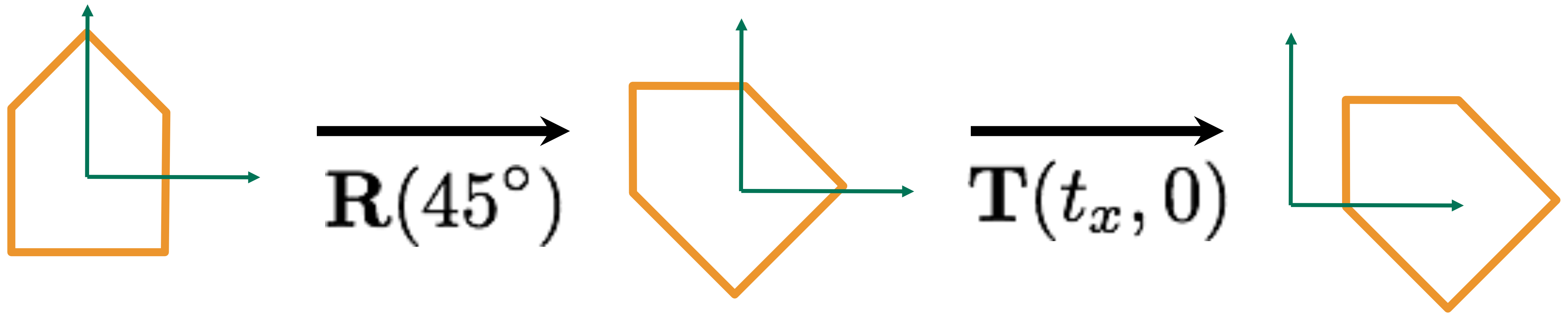
- Sequence of affine maps $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, ...
  - Concatenation by matrix multiplication

$$A_n(\ldots A_2(A_1(\mathbf{x}))) = \mathbf{A}_n \cdots \mathbf{A}_2 \cdot \mathbf{A}_1 \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

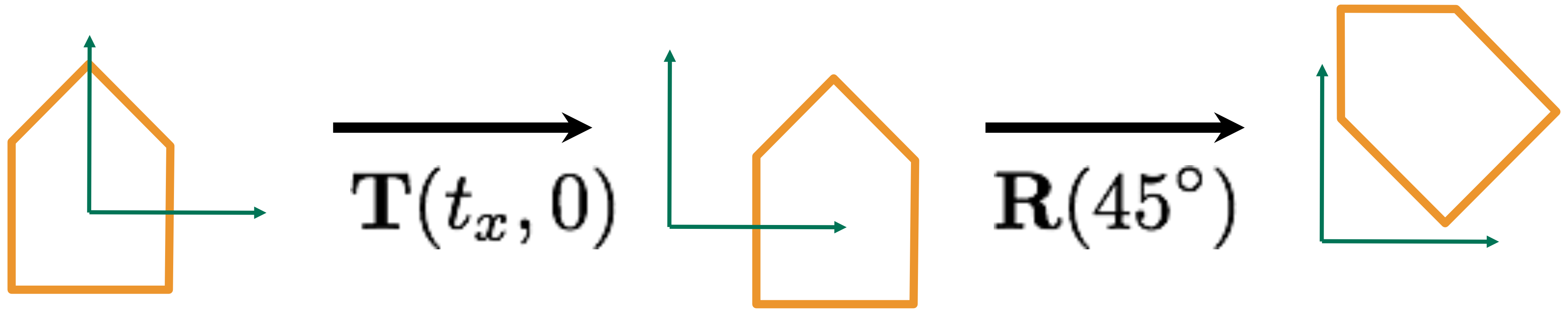
  - Very important for performance!
  - Matrix multiplication not commutative, ordering is important!

Rotation and Translation

- Matrix multiplication is not commutative!
  - First rotation, then translation

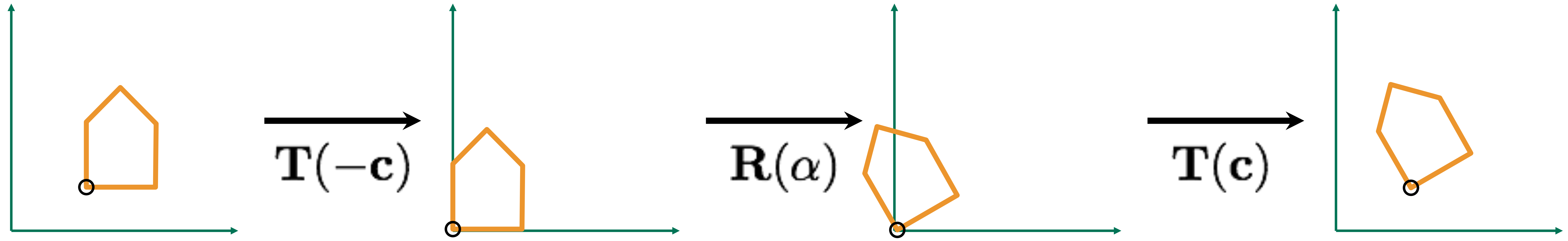


  - First translation, then rotation



2D Rotation

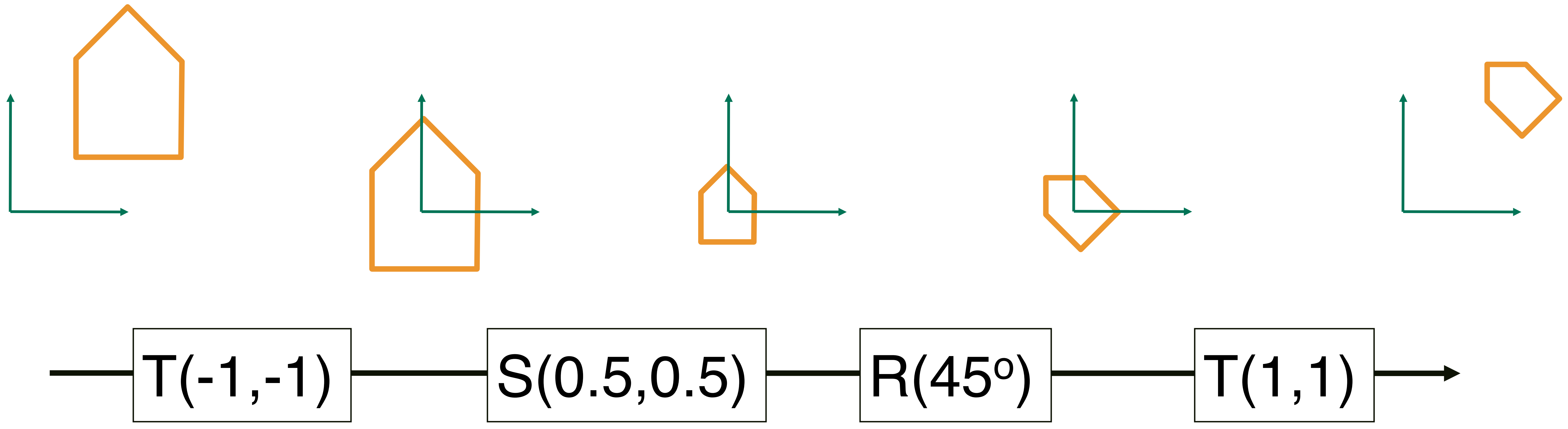
- How to rotate around a given point **c**?
  1. Translate **c** to origin
  2. Rotate
  3. Translate back

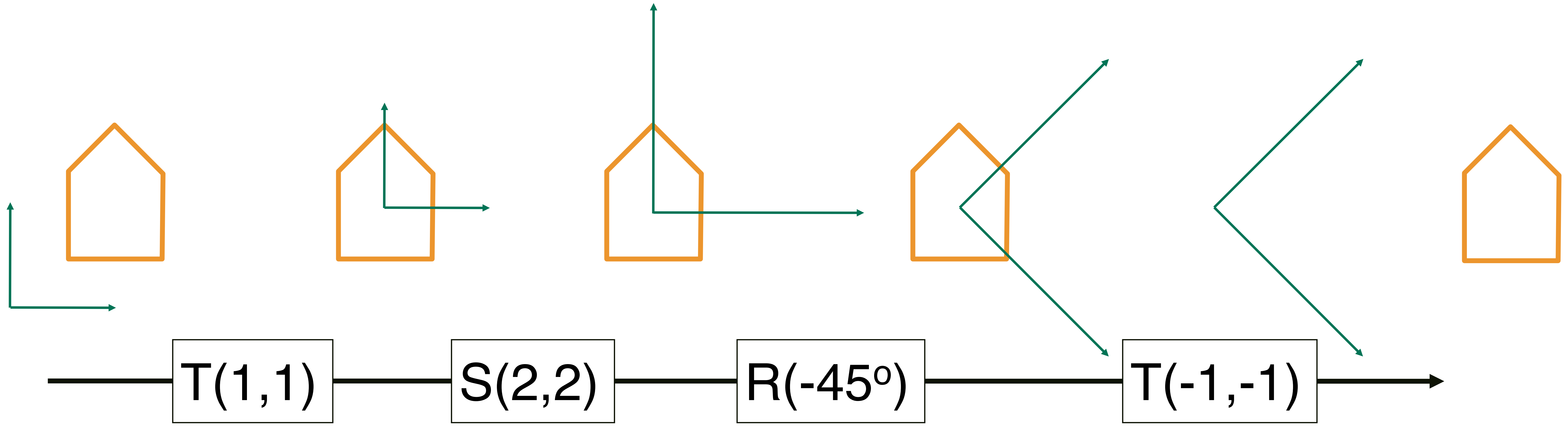


- Matrix representation?

$$\mathbf{T}(\mathbf{c}) \cdot \mathbf{R}(\alpha) \cdot \mathbf{T}(-\mathbf{c})$$

Transform Object or Camera?



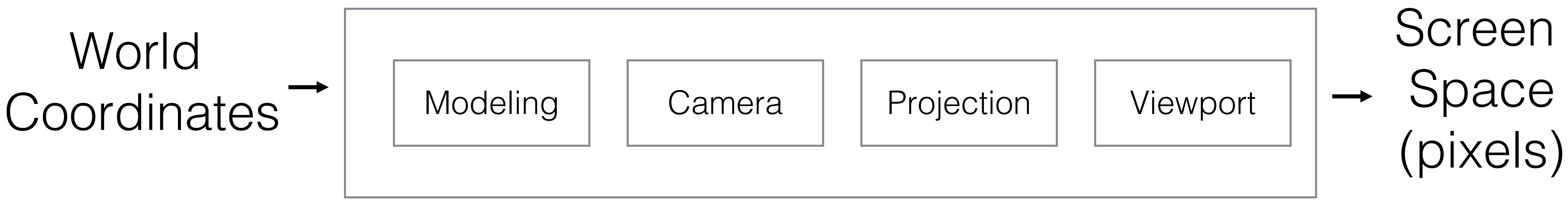


References

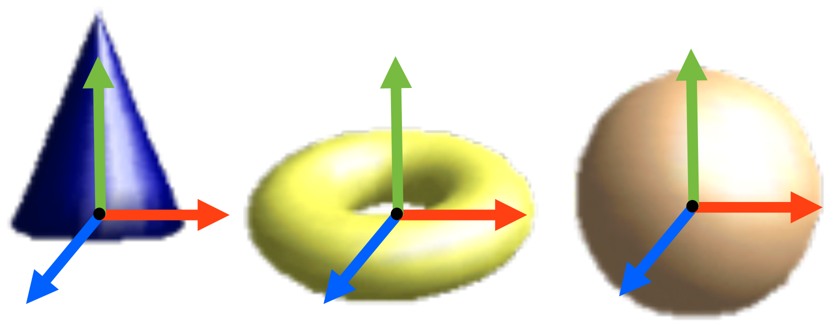
Fundamentals of Computer Graphics, Fourth Edition
4th Edition **by Steve Marschner, Peter Shirley**

Chapter 6

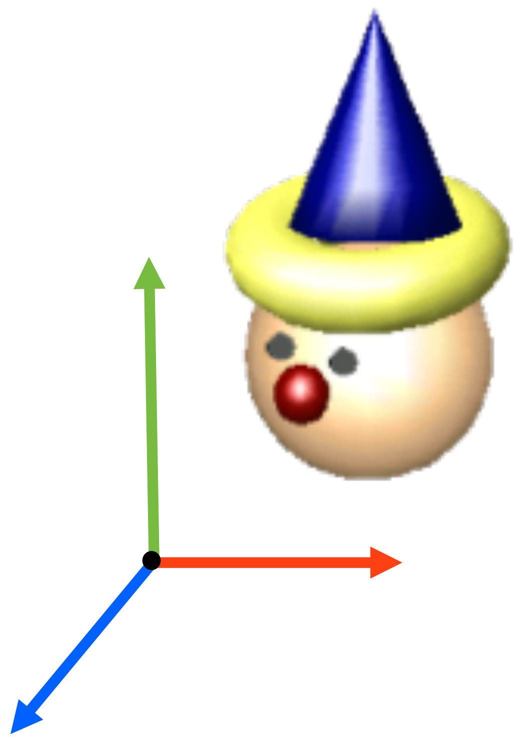
Viewing transformations



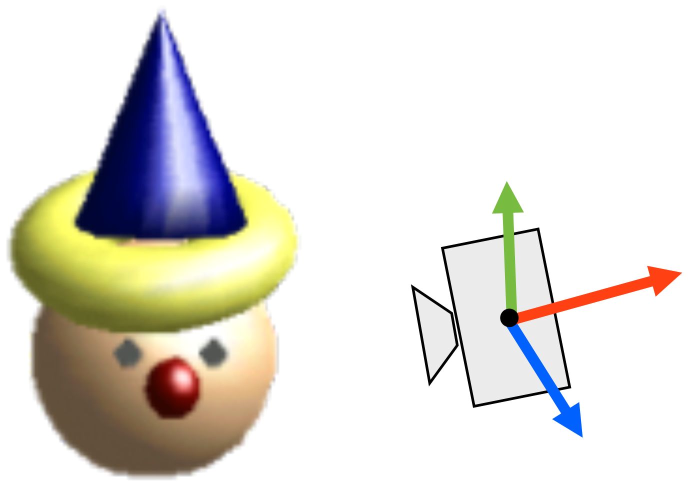
Coordinate Systems



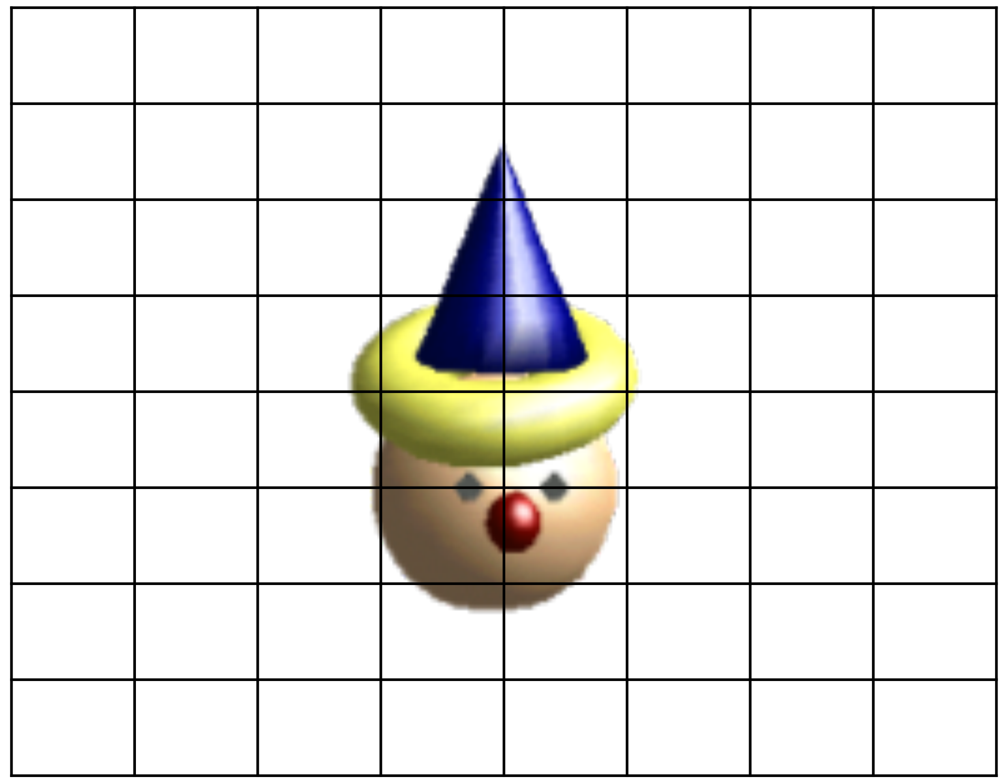
object
coordinates



world
coordinates

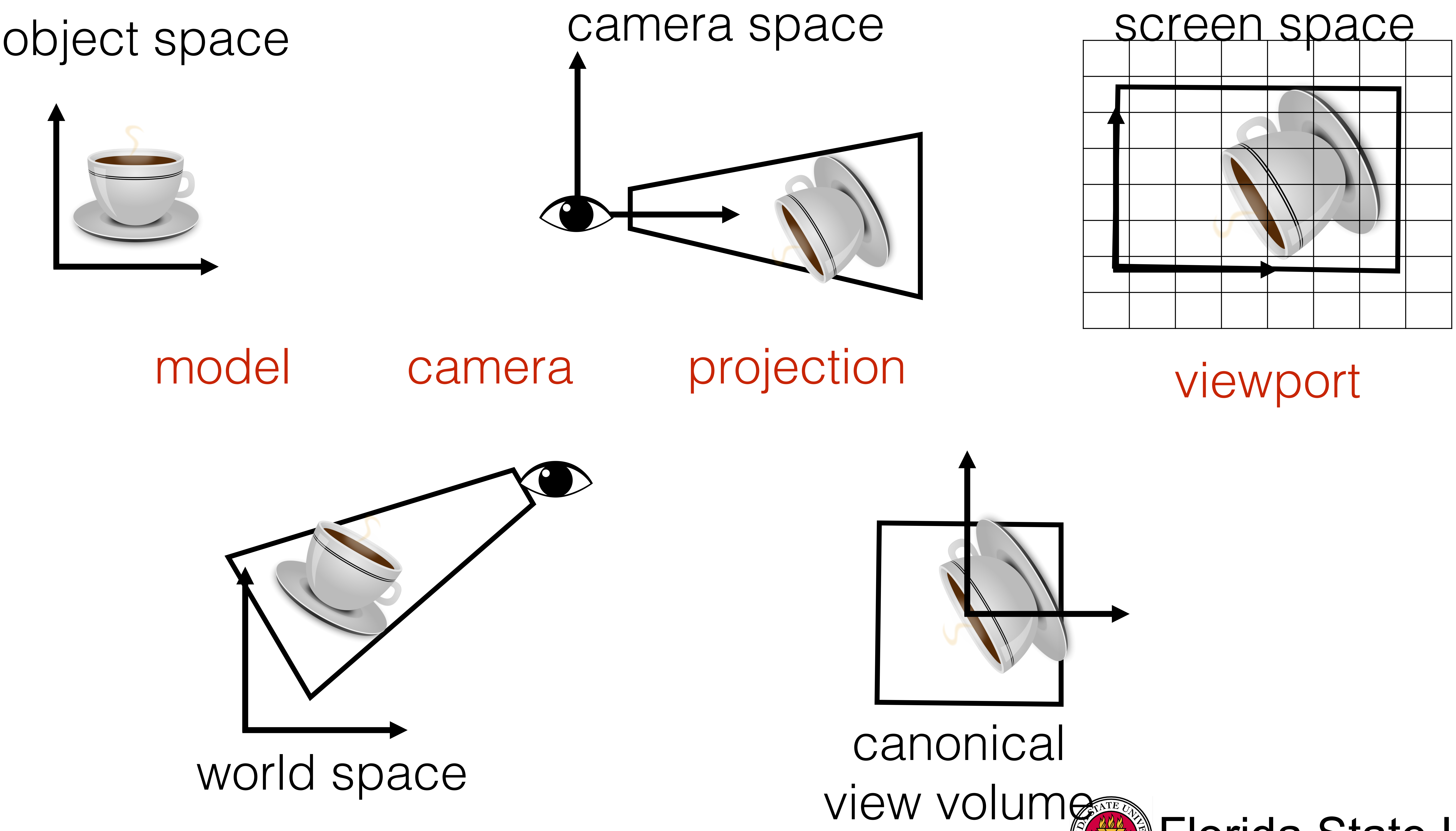


camera
coordinates

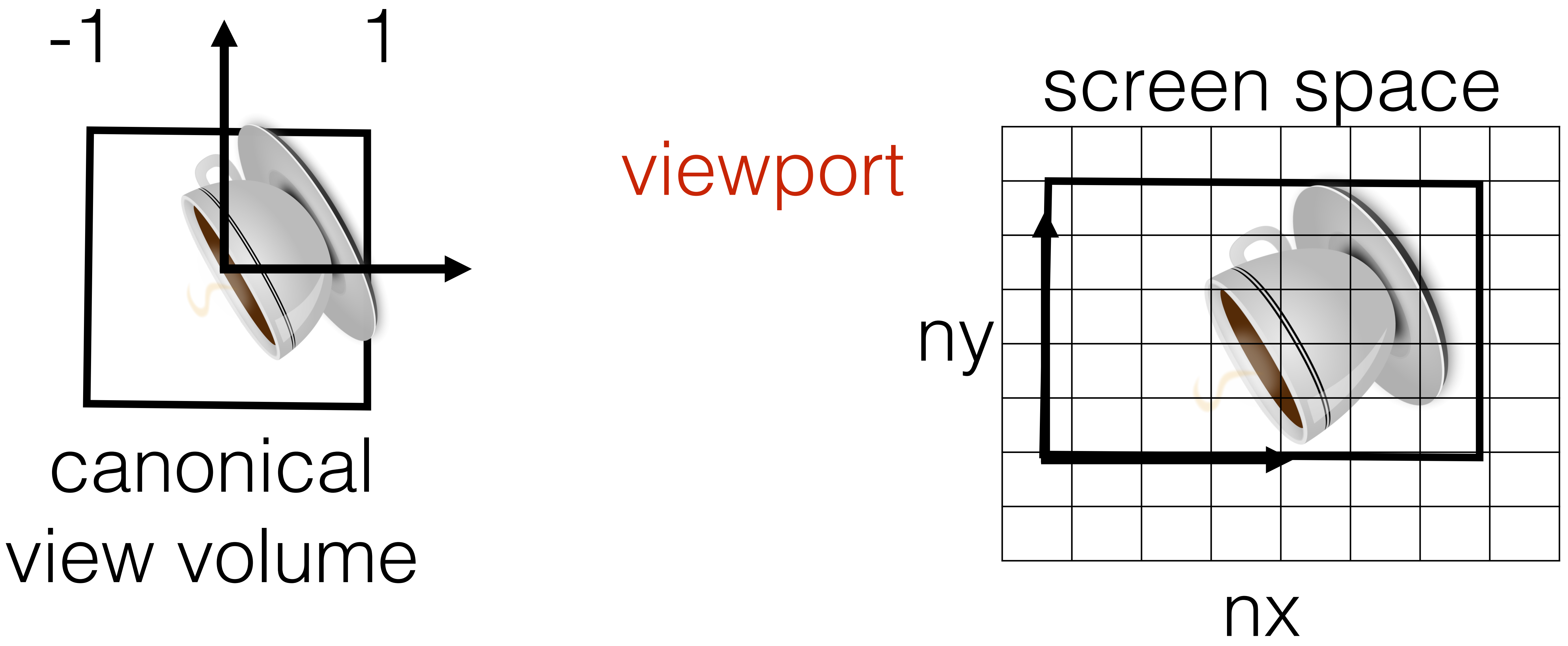


screen
coordinates

Viewing Transformation



Viewport transformation

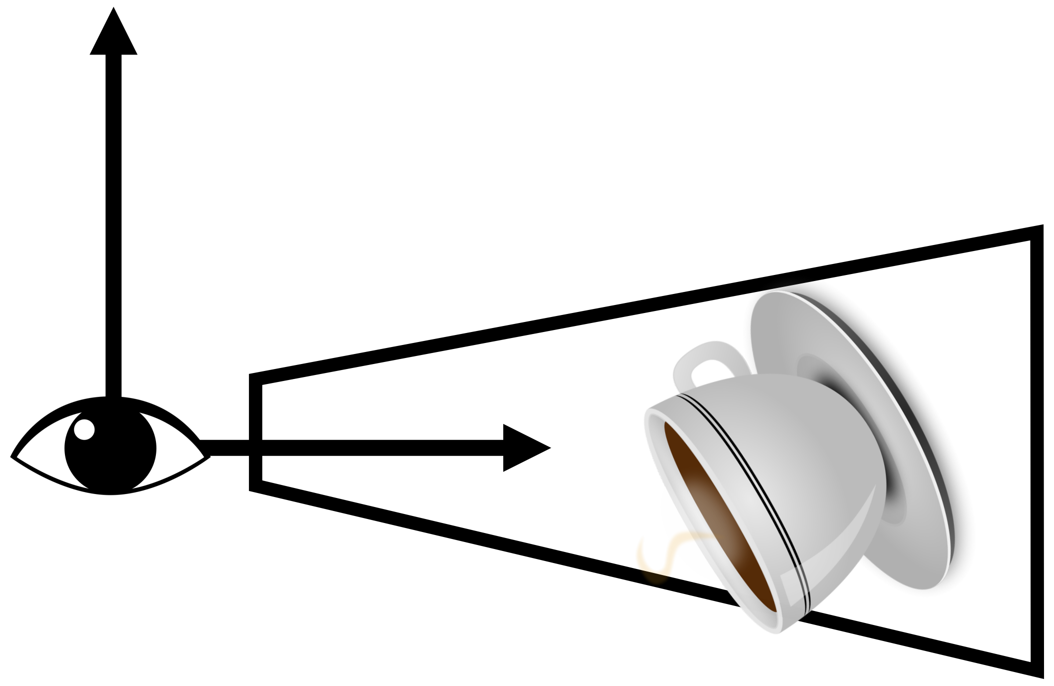


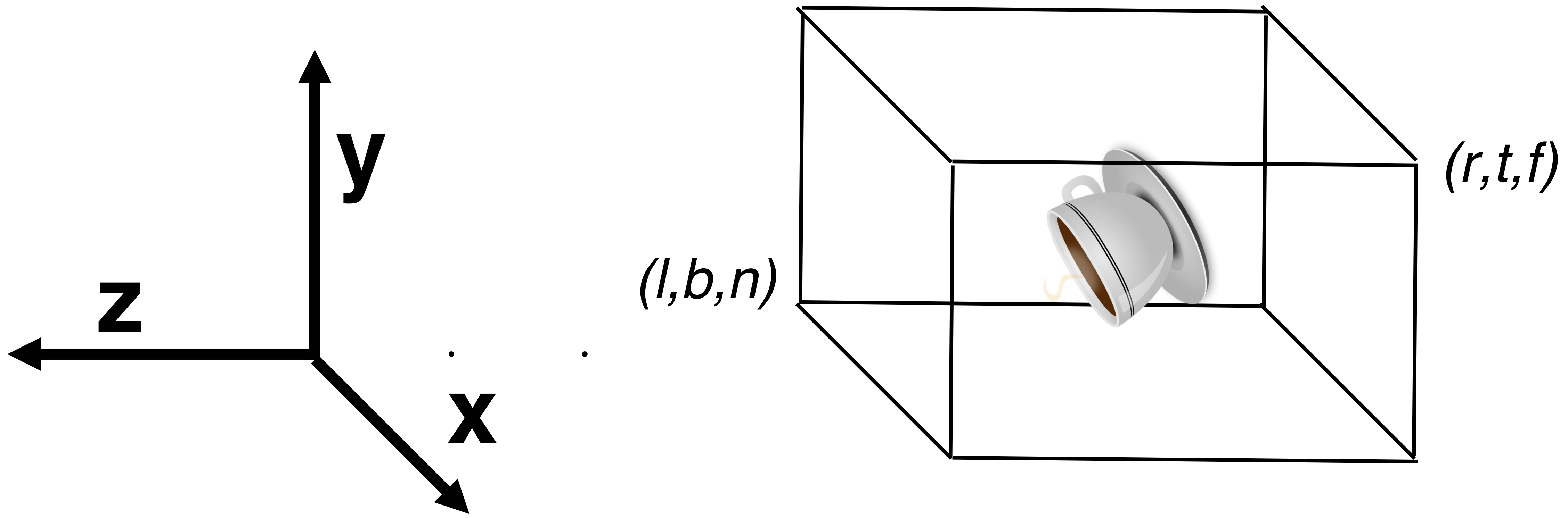
$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ 1 \end{bmatrix} = \begin{bmatrix} nx/2 & 0 & \frac{n_x-1}{2} \\ 0 & ny/2 & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ 1 \end{bmatrix}$$

How does it look in 3D?

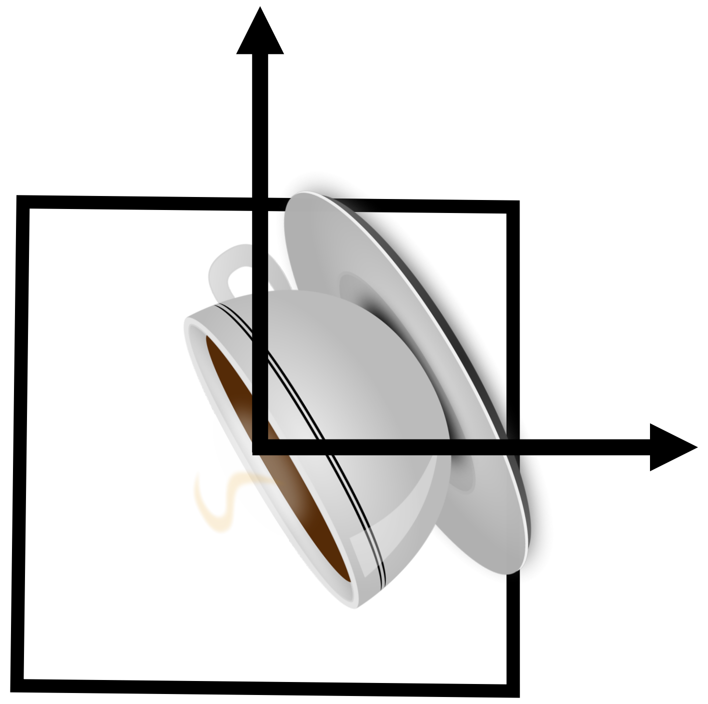
Orthographic Projection

camera space





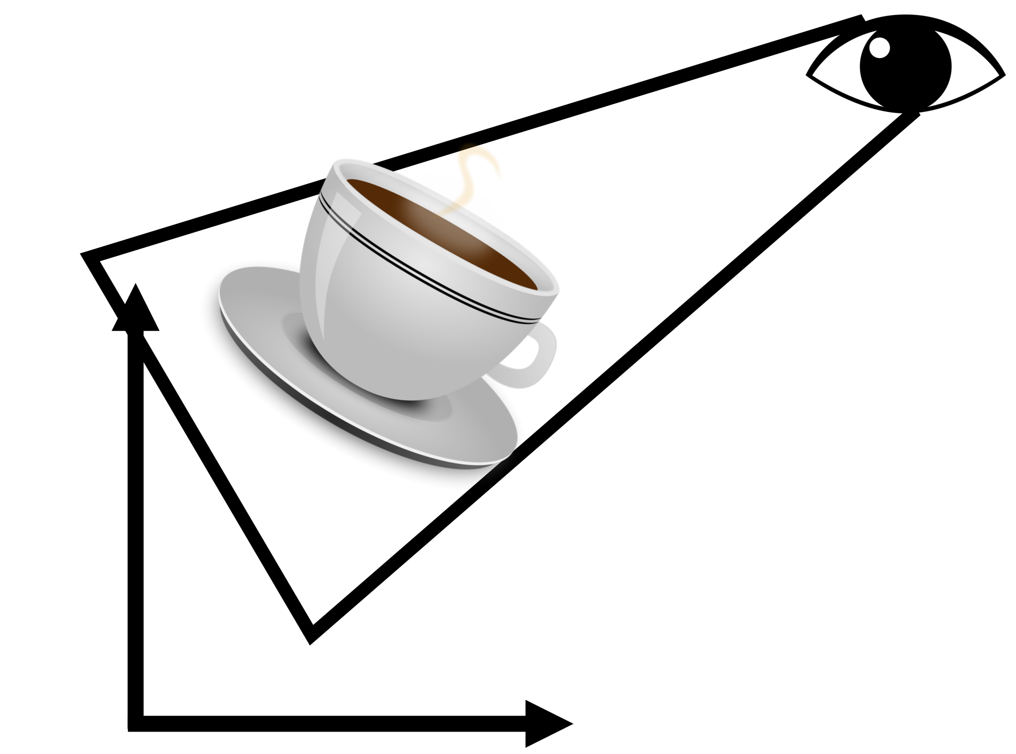
projection

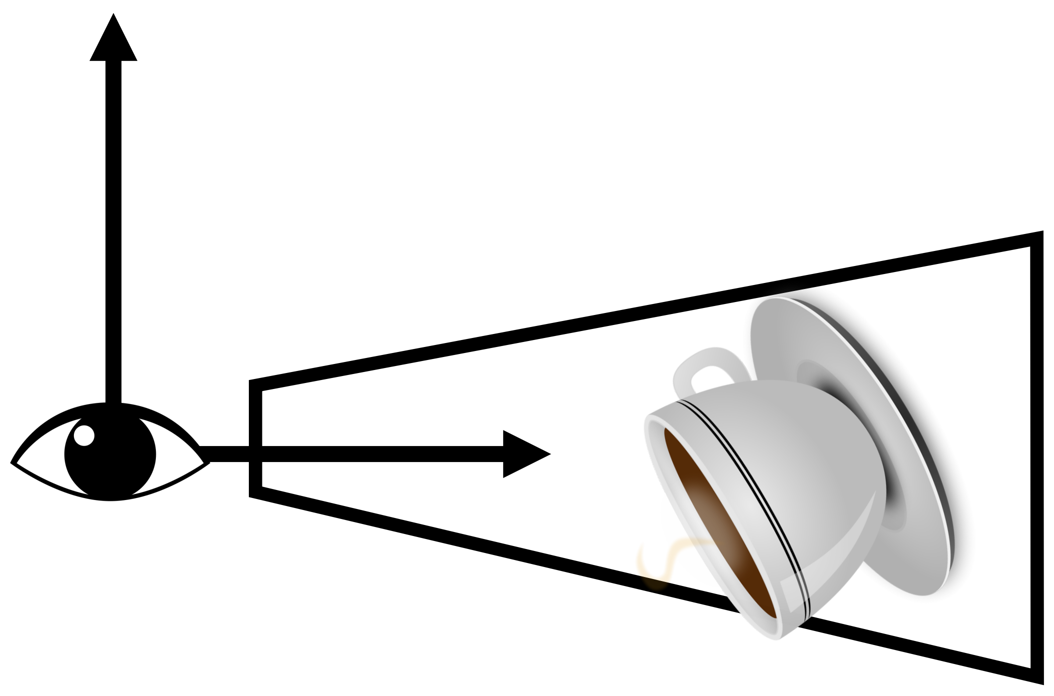


canonical
view volume

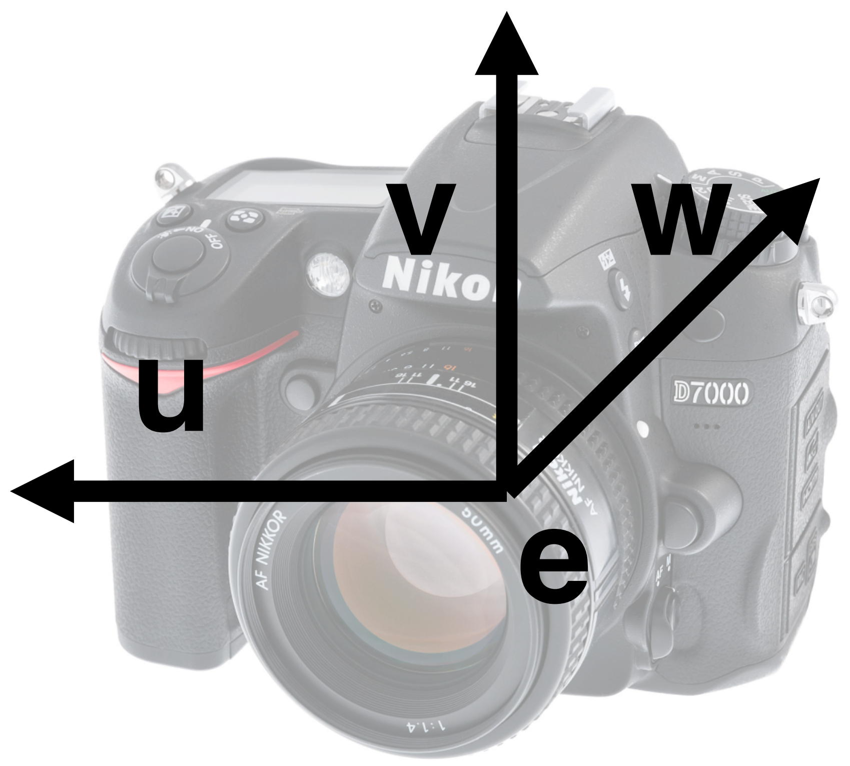
$$\mathbf{M}_{orth} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Camera Transformation





1. Construct the camera reference system given:
   1. The eye position **e**
   2. The gaze direction **g**
   3. The view-up vector **t**

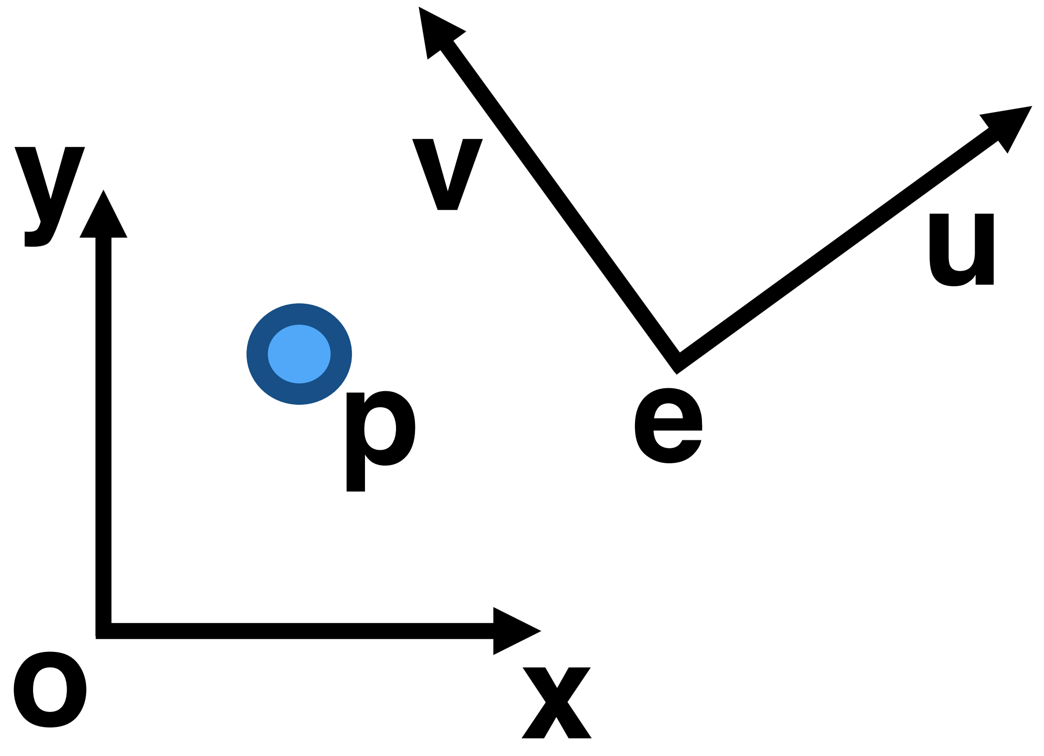


$$\mathbf{w} = -\frac{\mathbf{g}}{||\mathbf{g}||}$$

$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{||\mathbf{t} \times \mathbf{w}||}$$

$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$

Change of frame



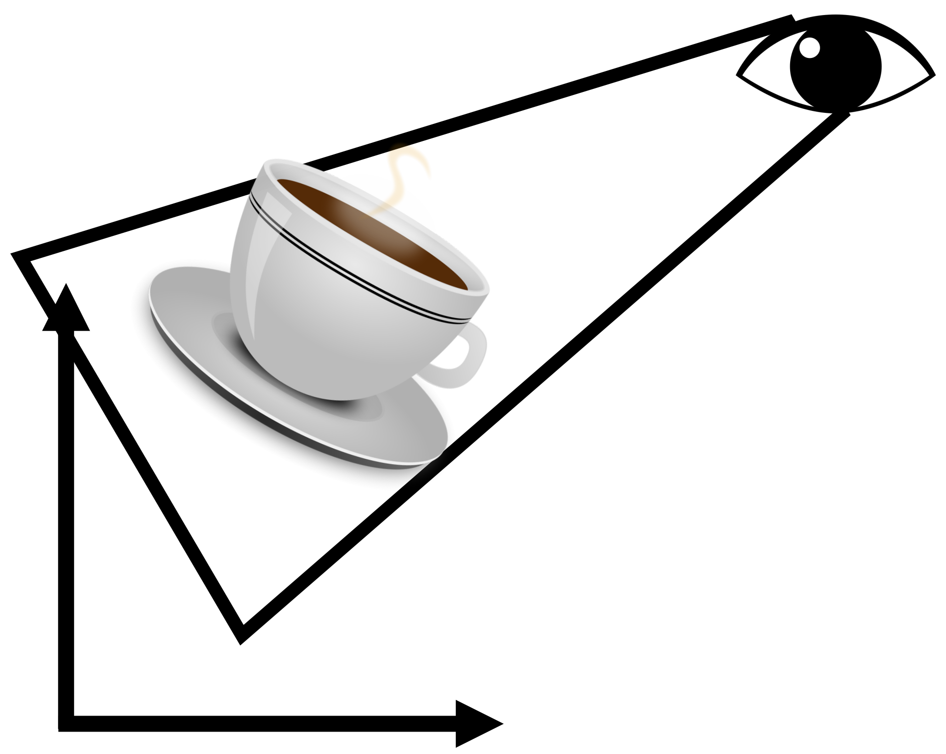
$$\mathbf{p} = (p_x, p_y) = \mathbf{o} + p_x\mathbf{x} + p_y\mathbf{y}$$
$$\mathbf{p} = (p_u, p_v) = \mathbf{e} + p_u\mathbf{u} + p_v\mathbf{v}$$

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & e_x \\ 0 & 1 & e_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & 0 \\ u_y & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & v_x & e_x \\ u_y & v_y & e_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix}$$

$$\mathbf{p}_{xy} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{e} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{uv} \qquad \mathbf{p}_{uv} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{e} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}_{xy}$$

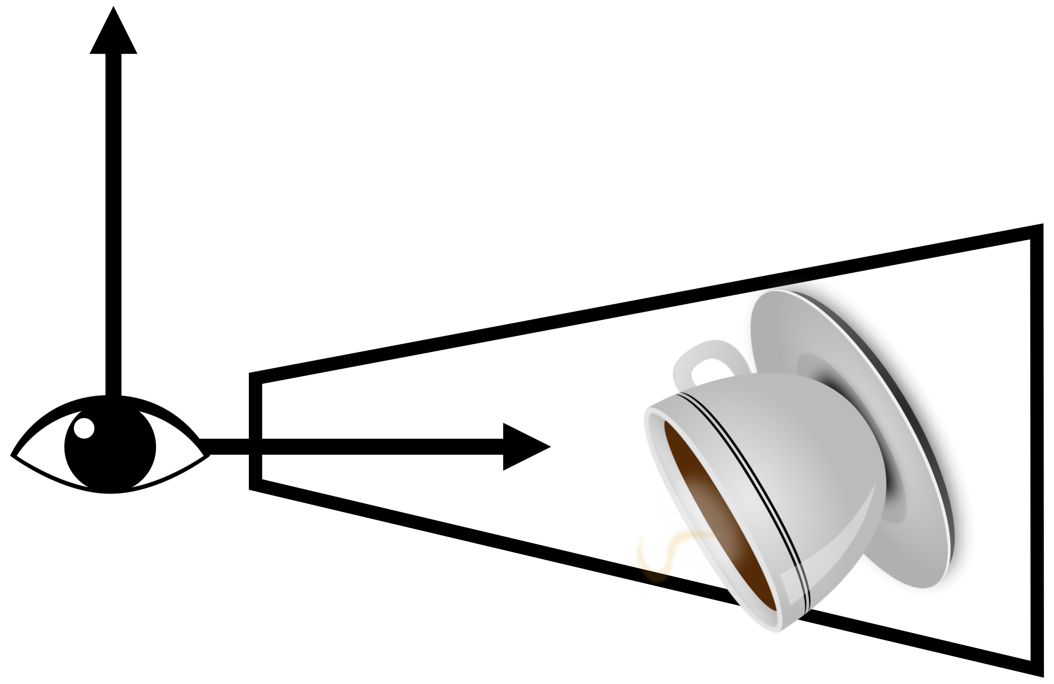
Can you write it directly without the inverse?

Camera Transformation



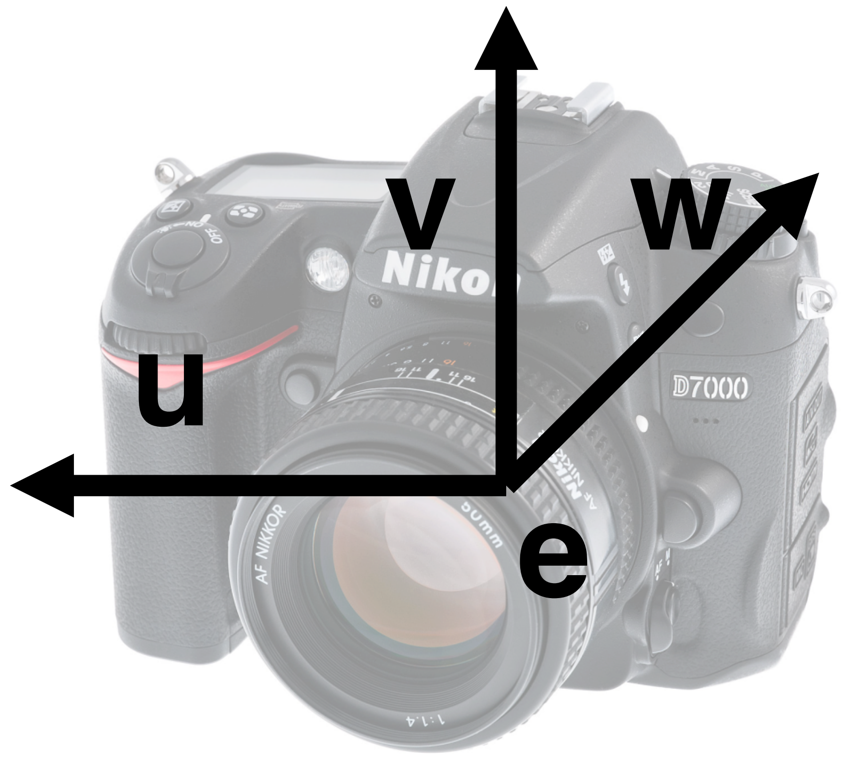
world space

camera



camera space

1. Construct the camera reference system given:
   1. The eye position **e**
   2. The gaze direction **g**
   3. The view-up vector **t**



$$\mathbf{w} = -\frac{\mathbf{g}}{||\mathbf{g}||}$$

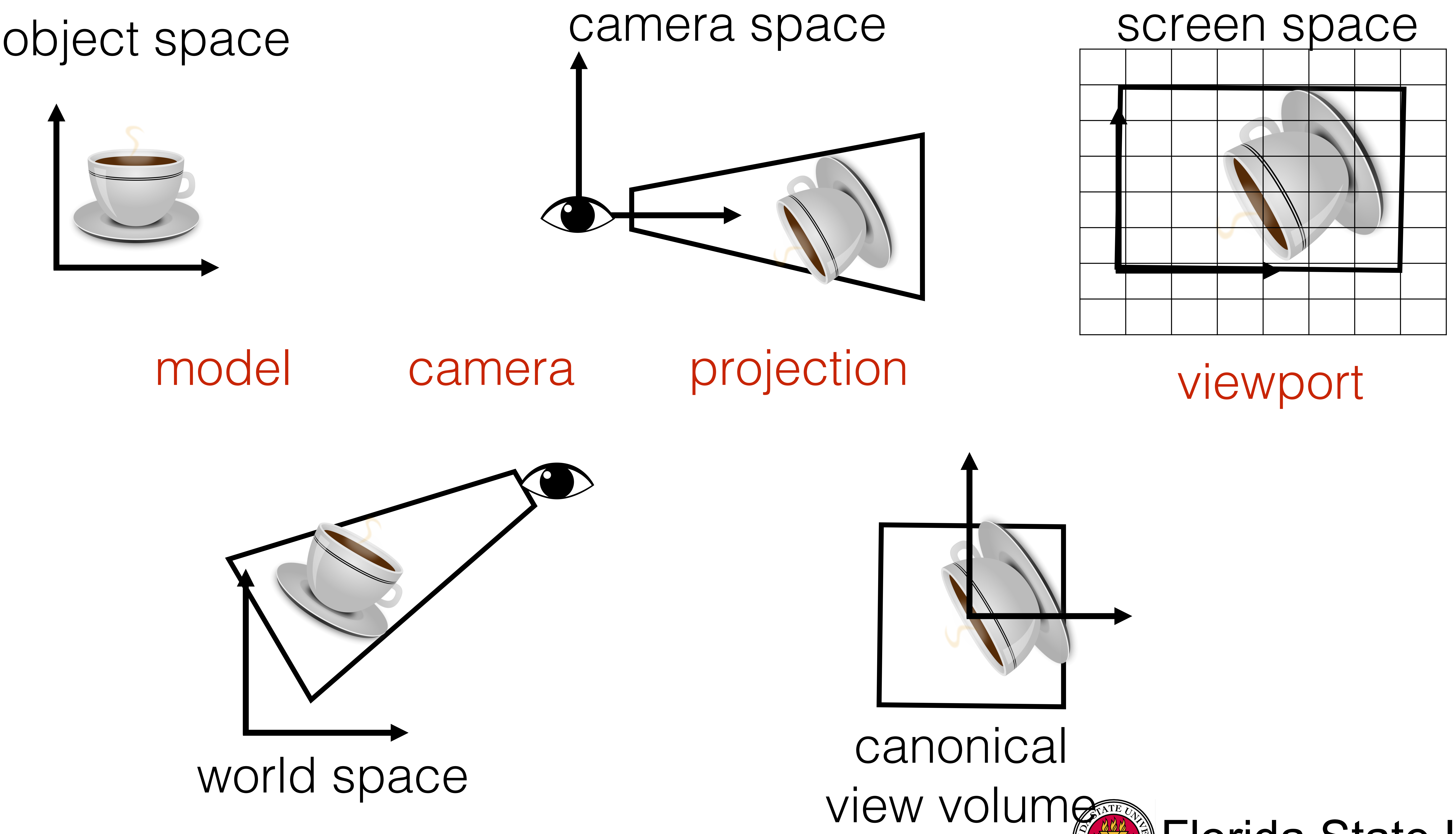
$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{||\mathbf{t} \times \mathbf{w}||}$$

$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$

2. Construct the unique transformations that converts world coordinates into camera coordinates

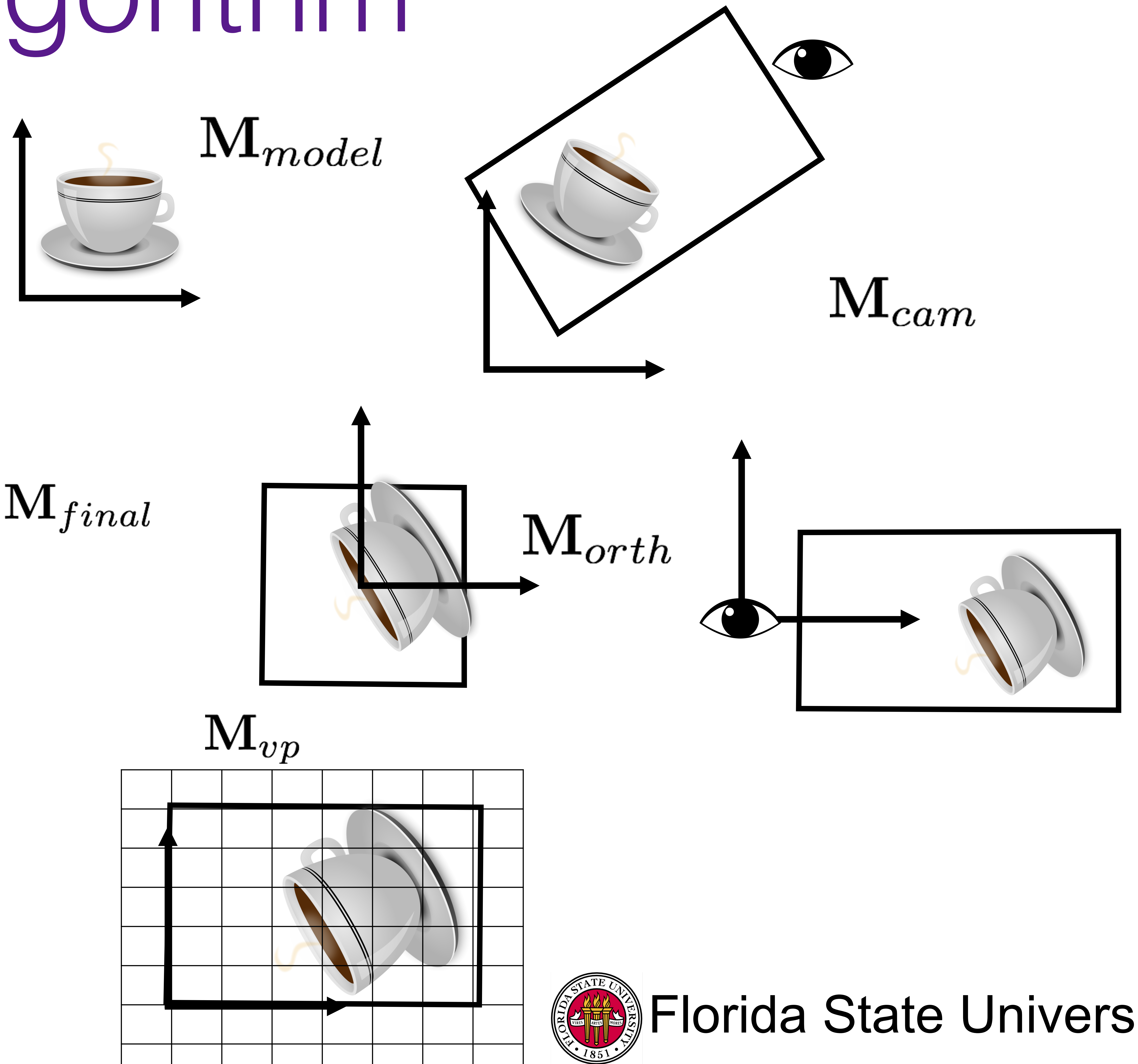
$$\mathbf{M}_{cam} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

Viewing Transformation



Algorithm

- Construct Viewport Matrix $\mathbf{M}_{vp}$
- Construct Projection Matrix $\mathbf{M}_{orth}$
- Construct Camera Matrix $\mathbf{M}_{cam}$
- $\mathbf{M} = \mathbf{M}_{vp}\mathbf{M}_{orth}\mathbf{M}_{cam}$
- For each model
  - Construct Model Matrix $\mathbf{M}_{model}$
  - $\mathbf{M}_{final} = \mathbf{M}\mathbf{M}_{model}$
  - For every point **p** in each primitive of the model
    - $\mathbf{p}_{final} = \mathbf{M}_{final}\mathbf{p}$
  - Rasterize the model



References

Fundamentals of Computer Graphics, Fourth Edition
4th Edition **by Steve Marschner, Peter Shirley**

Chapter 7