Advanced Computer Graphics

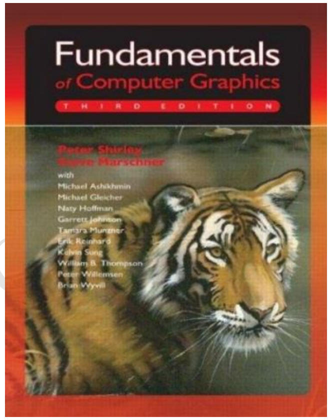
Lecture-05 Viewing and coordinate transformation

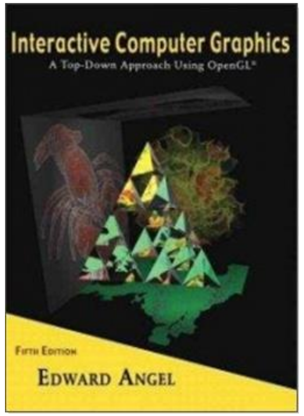
Tzung-Han Lin
National Taiwan University of Science and Technology
Graduate Institute of Color and Illumination Technology
e-mail: thl@mail.ntust.edu.tw

TAIWAN TECH National Taiwan University of Science and Technology
CIT 色彩與照明科技研究所 Graduate Institute of Color and Illumination Technology

Content in textbook

- Fundamentals of Computer Graphics, Chapter 7
- Interactive Computer Graphics, 5th edition, Chapter 5





Content outline

- 2D coordinate transformation
- 3D coordinate transformation
- 3D camera coordinate
- 3D viewing – projection (to observe the objects in world coordinate)

Coordinate transformation

- Consider two different situations:

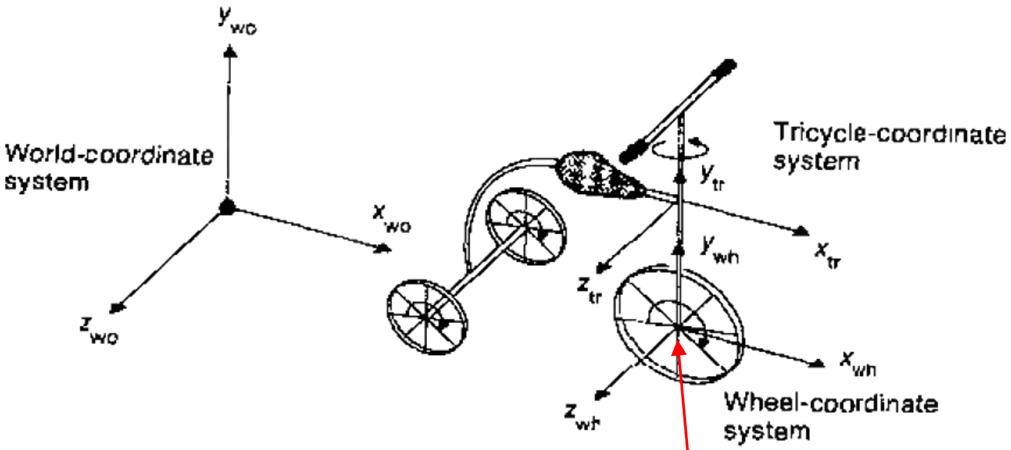
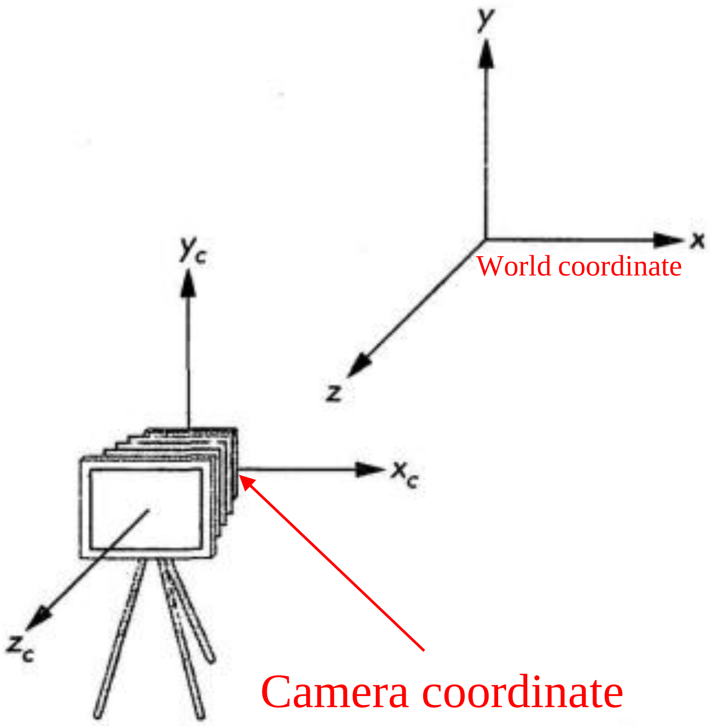


Fig. 5.26 A stylized tricycle with three coordinate systems.

Local coordinate
(Eular coordinate)



Camera coordinate

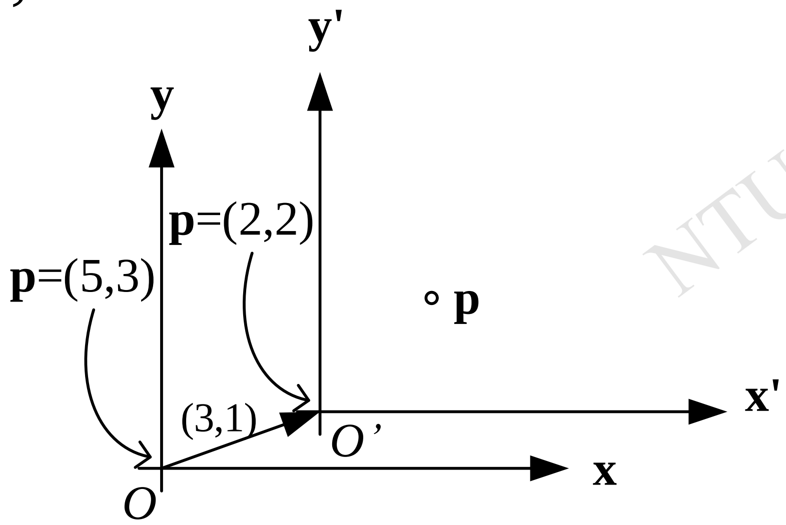
Picture from : Computer Graphics (Foley)
Interactive Computer Graphics, 5th edition

2D coordinate translation

- A 2D point p, which is (5, 3) in (x-y) coordinate, will be (2, 2) in (x'-y') coordinate if the (x'-y') coordinate has a (3, 1) translation relative to the (x-y) coordinate.
- The point p is NOT ever changed in above statement. The key point is to put another coordinate O' in the world coordinate O.
- A translation operation (3, 1) occurs, which means the O' is the coordinate after translating O.

$$\mathbf{p} = 5\mathbf{u} + 3\mathbf{v}$$

$$\mathbf{p} = 2\mathbf{u}' + 2\mathbf{v}'$$

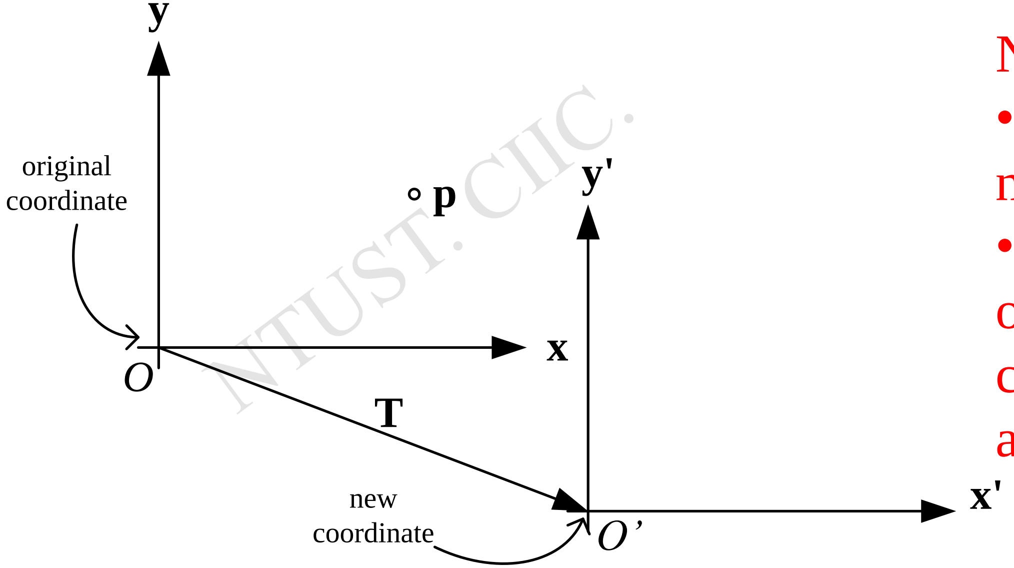


2D coordinate translation

$$\boxed{\mathbf{p}|_{O'}} = \begin{bmatrix}2\\2\\1\end{bmatrix}_{O'} = \mathbf{T}^{-1}\boxed{(\mathbf{p}|_{O})} = \begin{bmatrix}1&0&-3\\0&1&-1\\0&0&1\end{bmatrix}\begin{bmatrix}5\\3\\1\end{bmatrix}_{O}$$

Point in *O*' (new) coordinate

Point in *O* (world/original) coordinate



Note！
- The coordinate of this point is never changed in both coordinates.
- The only difference is that you observe **p** from different coordinates, thus, you describe **p** for another coordinate.

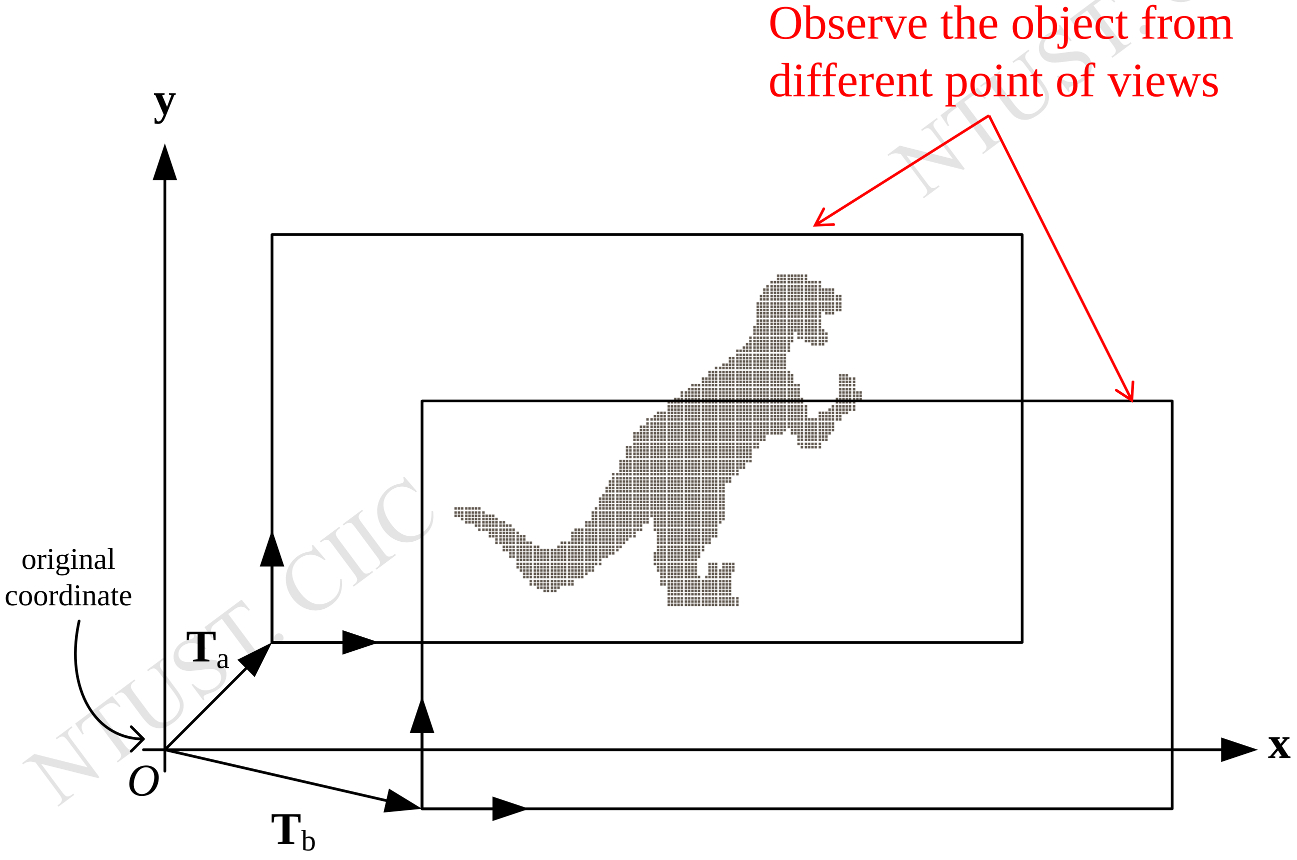
2D coordinate translation

- In summary, 2D coordinate translation

$$\mathbf{p}\,|_{O'} = \boxed{\mathbf{T}^{-1}}(\mathbf{p}\,|_{O}) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} (\mathbf{p}\,|_{O})$$

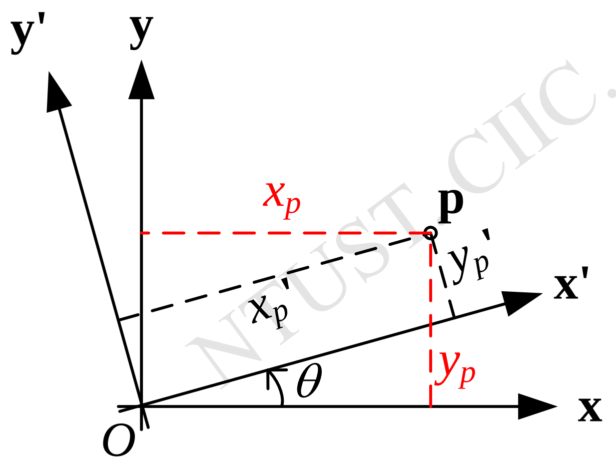
- Note: That means the new coordinate has been translated by a vector T. The value of p in new coordinate will be applied by a matrix (translation form) T^{-1}.

2D coordinate translation



2D coordinate rotation

- Similarly, in 2D coordinate rotation, the (x', y') coordinate is the (x, y) coordinate after rotating q degree.
- Thus, in (x, y) coordinate system $\mathbf{p} = x_p\mathbf{u} + y_p\mathbf{v}$
- And, in (x', y') coordinate system $\mathbf{p} = x_p'\mathbf{u}' + y_p'\mathbf{v}'$
- Here,



$$\mathbf{u}' = (\cos\theta)\mathbf{u} + (\sin\theta)\mathbf{v}$$

$$\mathbf{v}' = \cos(\theta + \frac{\pi}{2})\mathbf{u} + \sin(\theta + \frac{\pi}{2})\mathbf{v} = (-\sin\theta)\mathbf{u} + (\cos\theta)\mathbf{v}$$

2D coordinate rotation—cont.

- By substituting u' and v', we have

$$\mathbf{p} = x_p'\mathbf{u}' + y_p'\mathbf{v}' = x_p'[(\cos\theta)\mathbf{u} + (\sin\theta)\mathbf{v}] + y_p'[(-\sin\theta)\mathbf{u} + (\cos\theta)\mathbf{v}]$$
$$= (x_p'\cos\theta - y_p'\sin\theta)\mathbf{u} + (x_p'\sin\theta + y_p'\cos\theta)\mathbf{v} = x_p\mathbf{u} + y_p\mathbf{v}$$

- Finally, we have

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p' \\ y_p' \\ 1 \end{bmatrix}$$

- In other words,

$$\begin{bmatrix} x_p' \\ y_p' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \mathbf{R}_{-\theta}\cdot\mathbf{p} = \mathbf{R}_{\theta}^{-1}\cdot\mathbf{p}$$

2D coordinate rotation—cont.

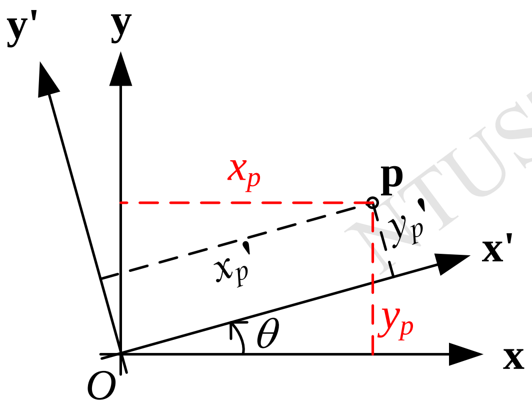
- Example: To observe a point (5, 3) in a new coordinate which has 15.8 degree rotation relative to world coordinate. The value in the new coordinate will be

$$\begin{bmatrix} 5.62 \\ 1.52 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 15.8^\circ & -\sin 15.8^\circ & 0 \\ \sin 15.8^\circ & \cos 15.8^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}^{\boxed{-1}} \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix}$$

(note the inverse operator)

- We may find the column vector in this matrix form is the basis of the new coordinate. (says u' and v' are unit vector)

$$\begin{bmatrix} 5.62 \\ 1.52 \\ 1 \end{bmatrix} = \begin{array}{c} \begin{array}{ccc} \mathbf{u'} & \mathbf{v'} & \end{array} \\ \begin{bmatrix} \boxed{\begin{matrix} \cos 15.8^\circ \\ \sin 15.8^\circ \end{matrix}} & \boxed{\begin{matrix} -\sin 15.8^\circ \\ \cos 15.8^\circ \end{matrix}} & \begin{matrix} 0 \\ 0 \end{matrix} \\ 0 & 0 & 1 \end{bmatrix}^{\boxed{-1}} \end{array} \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix}$$



2D coordinate rotation

- In summary, 2D coordinate rotation is

$$\mathbf{p}|_{O'} = \mathbf{R}^{-1}(\mathbf{p}|_{O})$$

- or

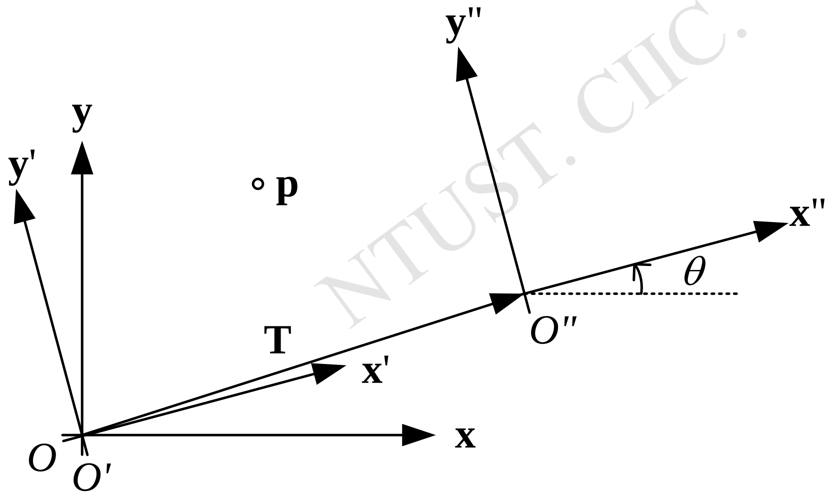
$$\mathbf{p}|_{O'} = \begin{bmatrix} \mathbf{u}' & \mathbf{v}' & \begin{matrix} 0 \\ 0 \end{matrix} \\ 0 & 0 & 1 \end{bmatrix}^{-1} (\mathbf{p}|_{O})$$

2D coordinate transformation

- Method-1: Consider the transformation in two steps
- Step-1: rotation part

$$\begin{bmatrix} x_p' \\ y_p' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

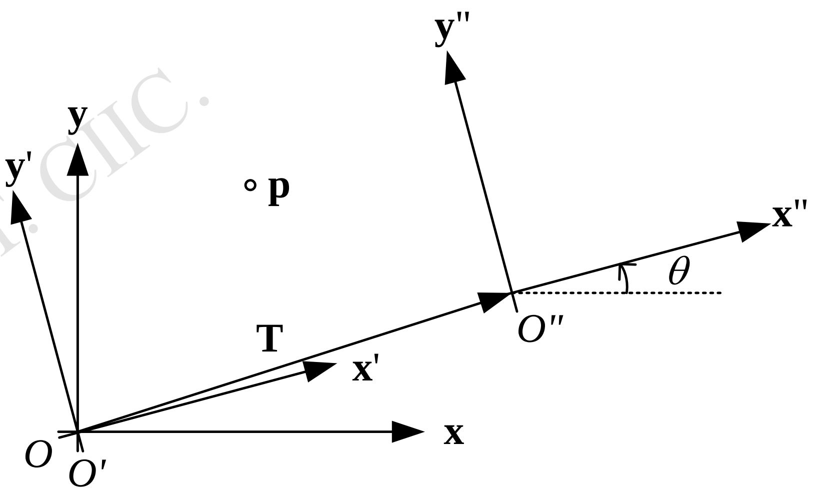
$$\mathbf{T}' = \begin{bmatrix} t_x' \\ t_y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} t_x \\ t_y \\ 1 \end{bmatrix} = \begin{bmatrix} t_x \cos\theta + t_y \sin\theta \\ -t_x \sin\theta + t_y \cos\theta \\ 1 \end{bmatrix}$$



- Step-2: translation part (in next page)

2D coordinate transformation—cont.

- Step-2: translation part—cont.



$$\mathbf{p}''=\begin{bmatrix} x_p'' \\ y_p'' \\ 1 \end{bmatrix}=\begin{bmatrix} 1 & 0 & -(t_x\cos\theta+t_y\sin\theta) \\ 0 & 1 & -(-t_x\sin\theta+t_y\cos\theta) \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_p' \\ y_p' \\ 1 \end{bmatrix}$$

$$=\begin{bmatrix} 1 & 0 & -(t_x\cos\theta+t_y\sin\theta) \\ 0 & 1 & -(-t_x\sin\theta+t_y\cos\theta) \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

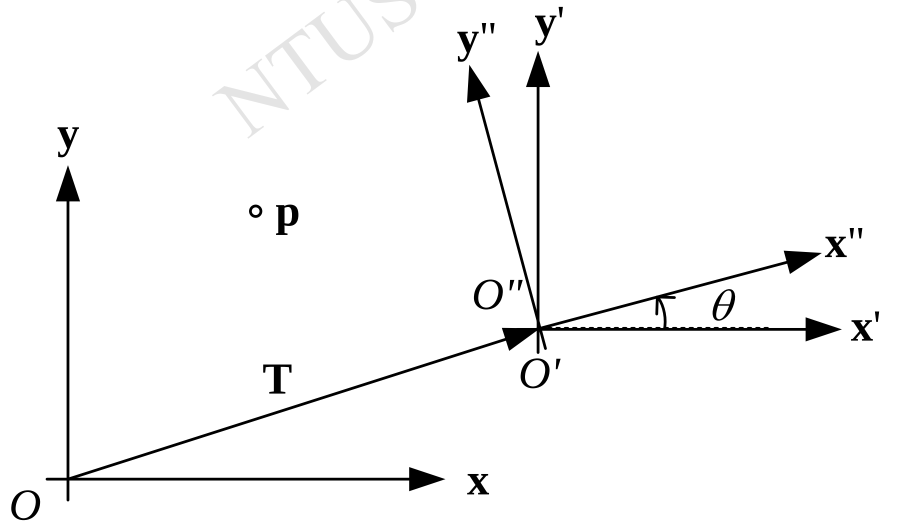
$$=\begin{bmatrix} \cos\theta & \sin\theta & -(t_x\cos\theta+t_y\sin\theta) \\ -\sin\theta & \cos\theta & -(-t_x\sin\theta+t_y\cos\theta) \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

$$=\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}=\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

2D coordinate transformation—cont.

- Method-2: Consider the transformation in two steps
- Step-1: translation part

$$\mathbf{p}' = \mathbf{T}^{-1}\mathbf{p} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$



- Step-2: rotation part

$$\mathbf{p}'' = \mathbf{R}^{-1}\mathbf{p}' = \mathbf{R}^{-1}\mathbf{T}^{-1}\mathbf{p} == \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

2D coordinate transformation

- In short:

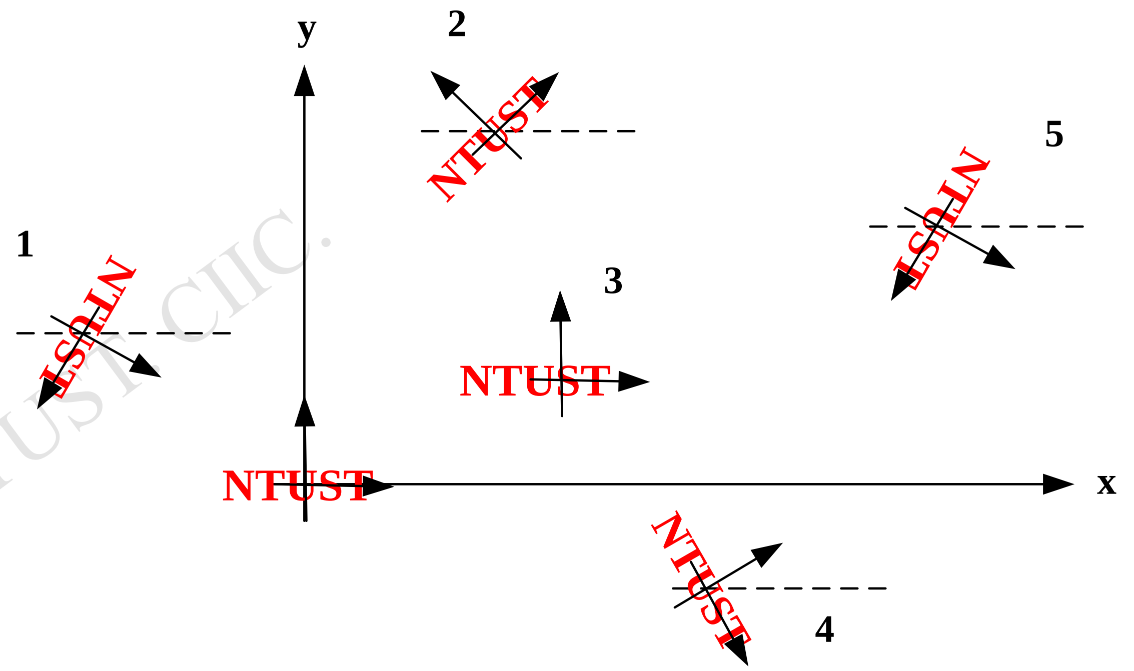
$$\mathbf{p}'' = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}$$

$$\mathbf{p}'' = \left(\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)^{-1} \mathbf{p} = \boxed{\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}}$$

- Conclude that: $\mathbf{p}'' = (\mathbf{TR})^{-1}\mathbf{p}$

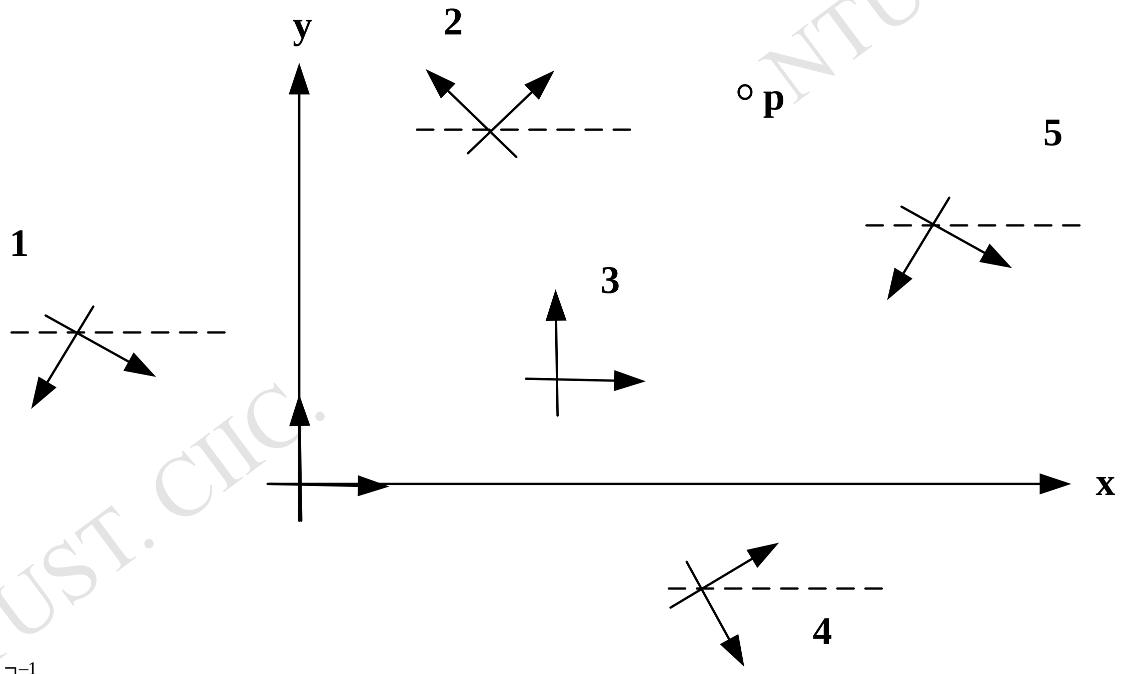
2D coordinate transformation—example

- Recall previous section, remember how to estimate a pose of a transformed object?



2D coordinate transformation—example

- The new point of view from other coordinate, what it should be.



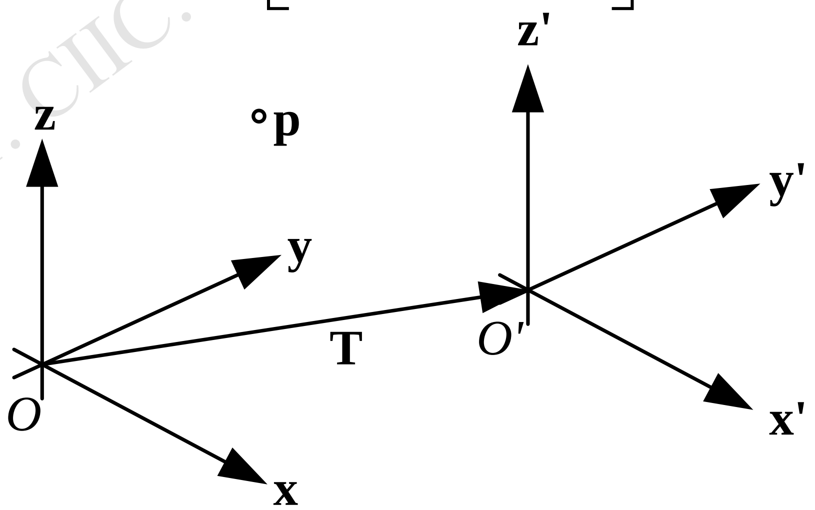
$$ans:(\mathbf{TR})^{-1}\mathbf{p}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1}$$

3D coordinate translation

- 3D coordinate translation is exactly the same with 2D version.

$$\mathbf{p}|_{O'} = \mathbf{T}^{-1}(\mathbf{p}|_{O}) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} (\mathbf{p}|_{O})$$



3D coordinate rotation

- Similar to 2D, the transformed value is formed by three new bases:
- In original coordinate $\mathbf{p} = x_p'\mathbf{u}' + y_p'\mathbf{v}' + z_p'\mathbf{w}'$
- Suppose that the bases are rotated by R as: $\mathbf{p} = x_p\mathbf{u} + y_p\mathbf{v} + z_p\mathbf{w}$

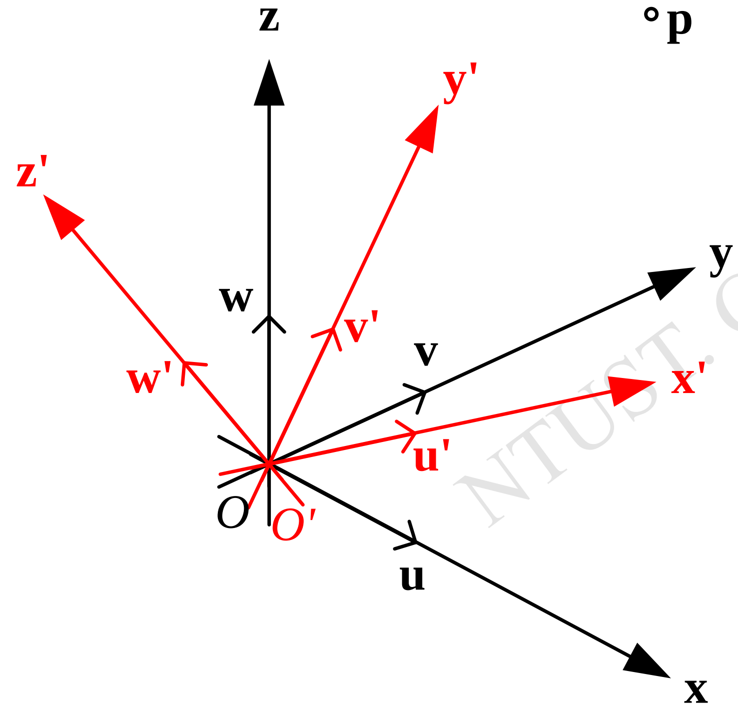
$$\mathbf{u}' = \mathbf{R}\mathbf{u}$$

$$\mathbf{v}' = \mathbf{R}\mathbf{v}$$

$$\mathbf{w}' = \mathbf{R}\mathbf{w}$$

- Then, we have

$$\mathbf{p} = x_p'\mathbf{R}\mathbf{u} + y_p'\mathbf{R}\mathbf{v} + z_p'\mathbf{R}\mathbf{w}$$



3D coordinate rotation—cont.

$$\mathbf{u}' = \begin{bmatrix} u_x' \\ u_y' \\ u_z' \end{bmatrix} = \mathbf{R}_{3\times3}\mathbf{u} = \begin{bmatrix} u_x' & v_x' & w_x' \\ u_y' & v_y' & w_y' \\ u_z' & v_z' & w_z' \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{v}' = \begin{bmatrix} v_x' \\ v_y' \\ v_z' \end{bmatrix} = \mathbf{R}_{3\times3}\mathbf{v} = \begin{bmatrix} u_x' & v_x' & w_x' \\ u_y' & v_y' & w_y' \\ u_z' & v_z' & w_z' \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{w}' = \begin{bmatrix} w_x' \\ w_y' \\ w_z' \end{bmatrix} = \mathbf{R}_{3\times3}\mathbf{w} = \begin{bmatrix} u_x' & v_x' & w_x' \\ u_y' & v_y' & w_y' \\ u_z' & v_z' & w_z' \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

3D coordinate rotation—cont.

- Extend to homogenous coordinate for convenience.

$$\mathbf{R}\begin{bmatrix} x_p' \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{R}\begin{bmatrix} 0 \\ y_p' \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ y_p \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{R}\begin{bmatrix} 0 \\ 0 \\ z_p' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ z_p \\ 1 \end{bmatrix}$$

$$\Rightarrow \mathbf{R}\begin{bmatrix} x_p' \\ y_p' \\ z_p' \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \Rightarrow \mathbf{p}\big|_{O'} = \mathbf{R}^{-1} \cdot \mathbf{p}\big|_{O}$$

Coordinate value from a new coordinate

Coordinate value from world coordinate (original)

3D coordinate rotation—cont.

- R will be obtained as

$$\mathbf{R} = \begin{bmatrix} \boxed{\begin{matrix} u_x' \\ u_y' \\ u_z' \end{matrix}} & \boxed{\begin{matrix} v_x' \\ v_y' \\ v_z' \end{matrix}} & \boxed{\begin{matrix} w_x' \\ w_y' \\ w_z' \end{matrix}} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

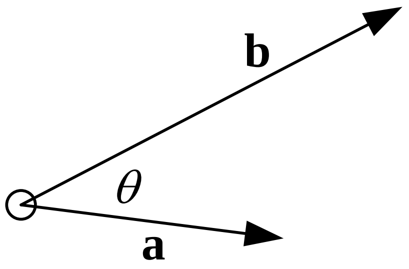
- Here, three bases are defined as (mutually orthogonal)

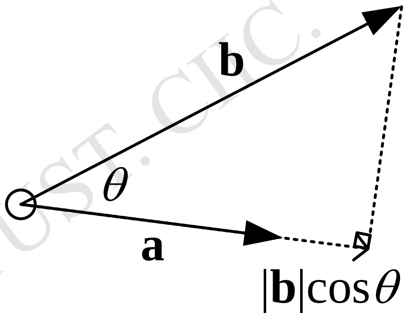
$$\mathbf{u}' = \begin{bmatrix} u_x' \\ u_y' \\ u_z' \end{bmatrix} \quad \mathbf{v}' = \begin{bmatrix} v_x' \\ v_y' \\ v_z' \end{bmatrix} \quad \mathbf{w}' = \begin{bmatrix} w_x' \\ w_y' \\ w_z' \end{bmatrix}$$

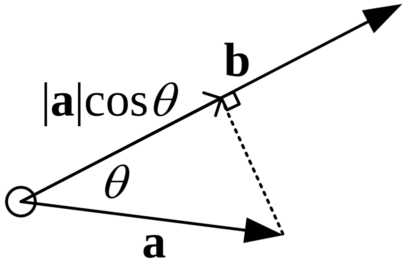
x' axis unit vector　　*y*' axis　　*z*' axis

3D coordinate rotation—cont.

- Recall the concept, inner product of two vectors:







- “Geometrically, it is the product of the Euclidean magnitudes of the two vectors and the cosine of the angle between them.” from Wiki.

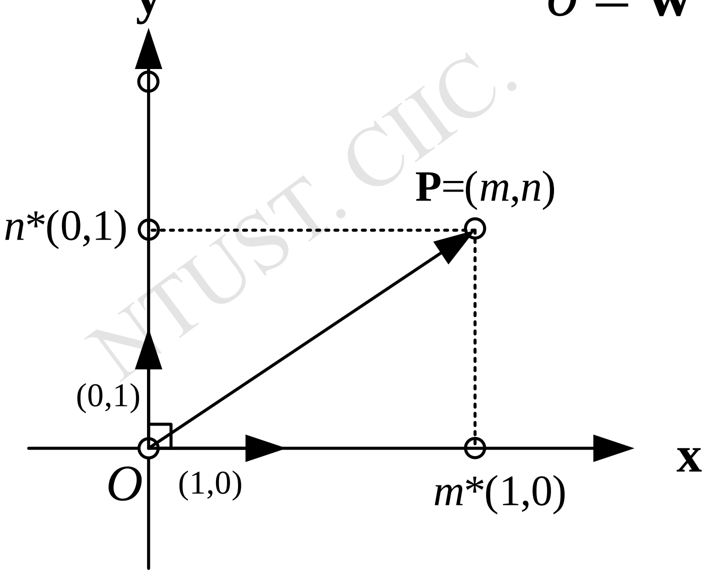
3D coordinate rotation—cont.

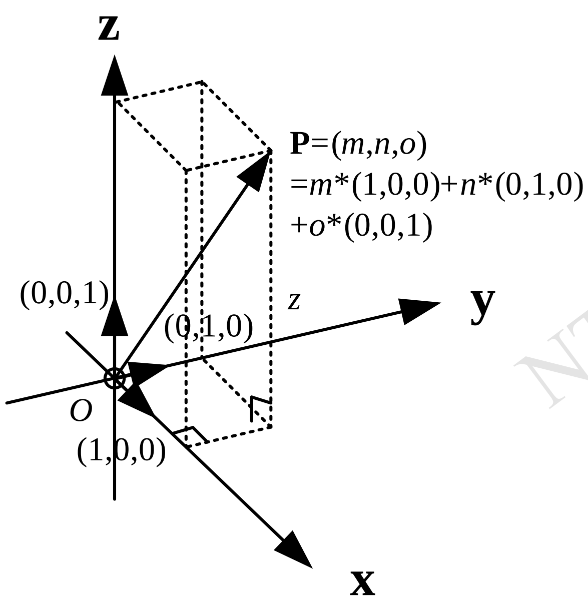
- In 2D/3D space, the independent component at each axis of a vector can be considered as the inner product of itself and the basis.
- For example, in 3D,

$$m = \mathbf{u} \cdot \mathbf{P}$$

$$n = \mathbf{v} \cdot \mathbf{P}$$

$$o = \mathbf{w} \cdot \mathbf{P}$$

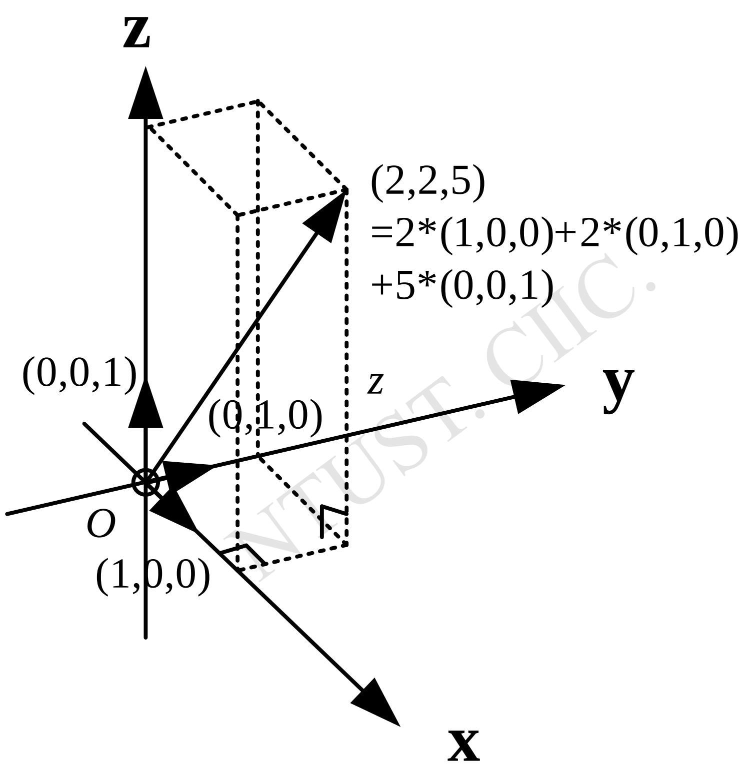




3D coordinate rotation—cont.

- Example:

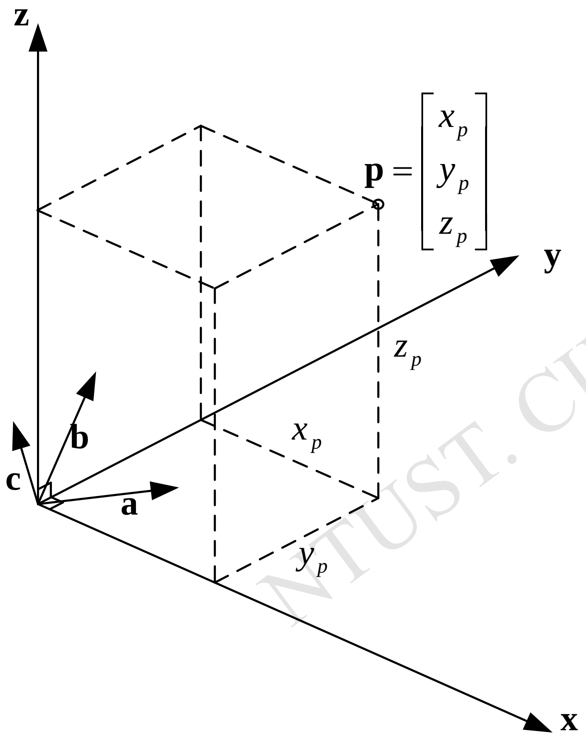
$$m = \mathbf{u} \cdot \mathbf{P}$$
$$n = \mathbf{v} \cdot \mathbf{P}$$
$$o = \mathbf{w} \cdot \mathbf{P}$$

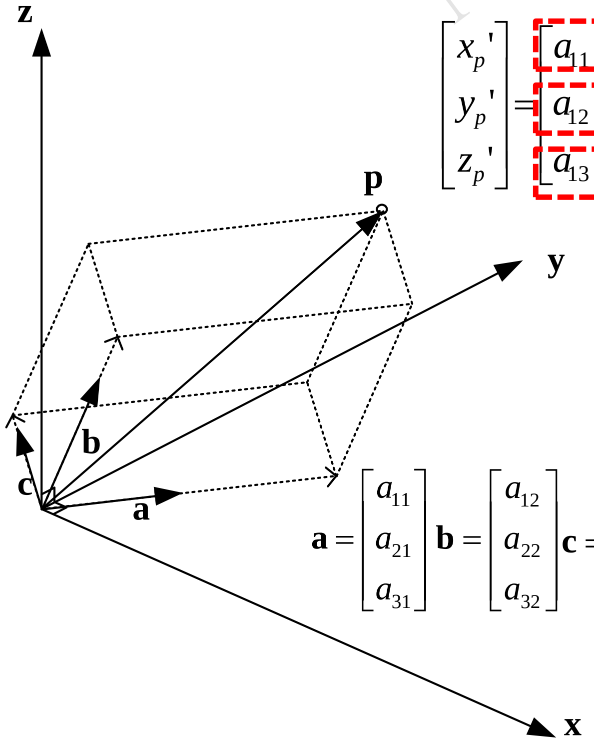


$$\begin{bmatrix} 2 \\ 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 5 \end{bmatrix}$$

x axis (unit vector)
y axis (unit vector)
z axis (unit vector)

3D coordinate rotation—cont.





$$\begin{bmatrix} x_p' \\ y_p' \\ z_p' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$$

a vector

b vector

c vector

$$\mathbf{a} = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix} \mathbf{b} = \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix} \mathbf{c} = \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \end{bmatrix}$$

3D coordinate rotation—cont.

$$\mathbf{R}^{\mathrm{T}} = \mathbf{R}^{-1}$$

Note: *R* is orthogonal matrix

$$\begin{bmatrix} x_p' \\ y_p' \\ z_p' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$$

a vector(**x'**)
b vector(**y'**)
c vector(**z'**)

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^T \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^{\boxed{-1}} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$$

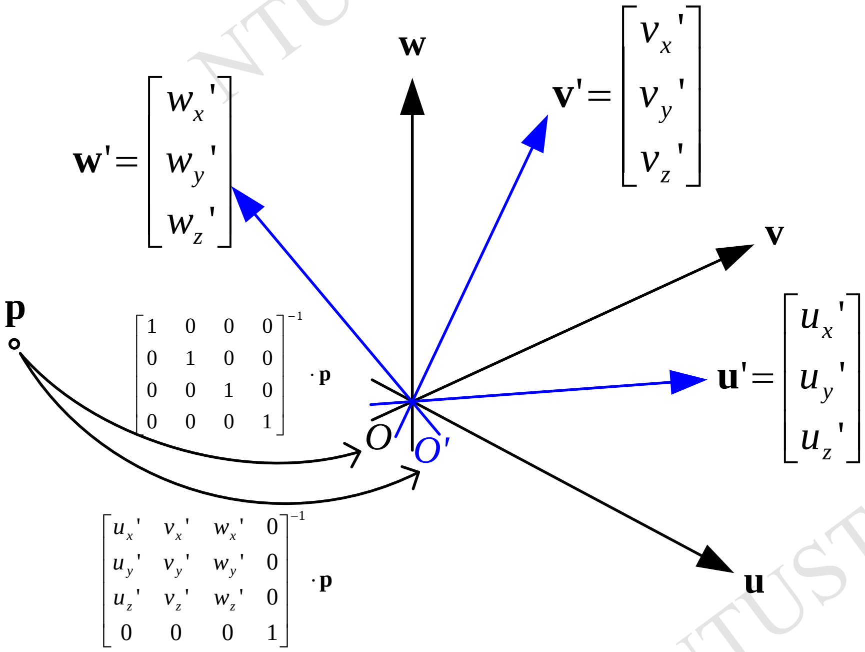
a vector(**x'**) **b** vector(**y'**) **c** vector(**z'**)

3D coordinate rotation—cont.

- Summary

$$\mathbf{R} = \begin{bmatrix} \boxed{\begin{matrix} u_x' \\ u_y' \\ u_z' \end{matrix}} & \boxed{\begin{matrix} v_x' \\ v_y' \\ v_z' \end{matrix}} & \boxed{\begin{matrix} w_x' \\ w_y' \\ w_z' \end{matrix}} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

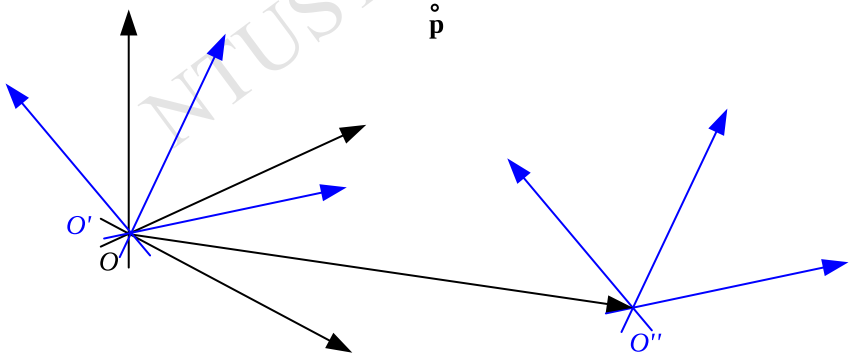
$$\mathbf{p}\big|_{O'} = \mathbf{R}^{-1} \cdot \mathbf{p}\big|_{O}$$



3D coordinate transformation

- Method-1: Similar to 2D, two steps are needed.
- Step-1: coordinate rotation

$$\mathbf{p}' = \mathbf{R}^{-1} \cdot \mathbf{p}$$



- including that the rotated T vector becomes:

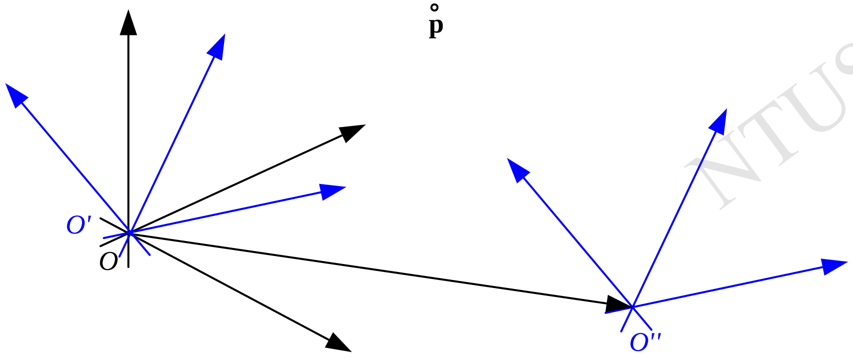
$$\mathbf{T}' = \begin{bmatrix} t_x' \\ t_y' \\ t_z' \\ 1 \end{bmatrix} = \mathbf{R}^{-1}\mathbf{T} = \mathbf{R}^{-1} \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{u}' \cdot \mathbf{T} \\ \mathbf{v}' \cdot \mathbf{T} \\ \mathbf{w}' \cdot \mathbf{T} \\ 1 \end{bmatrix}$$

3D coordinate transformation—cont.

- Method-1: Similar to 2D, two steps are needed.
- Step-2: coordinate translation (with the rotated T, says T')

$$\mathbf{p}''=\begin{bmatrix}1&0&0&-\mathbf{u}'\cdot\mathbf{T}\\0&1&0&-\mathbf{v}'\cdot\mathbf{T}\\0&0&1&-\mathbf{w}'\cdot\mathbf{T}\\0&0&0&1\end{bmatrix}\mathbf{p}'=\begin{bmatrix}1&0&0&\mathbf{u}'\cdot\mathbf{T}\\0&1&0&\mathbf{v}'\cdot\mathbf{T}\\0&0&1&\mathbf{w}'\cdot\mathbf{T}\\0&0&0&1\end{bmatrix}^{-1}\mathbf{R}^{-1}\mathbf{p}$$

$$=(\mathbf{RTR})^{-1}\mathbf{R}^{-1}\mathbf{p}=(\mathbf{TR})^{-1}\mathbf{p}$$



3D coordinate transformation—cont.

- Method-2: two steps are needed.
- Step-1: coordinate translation

$$\mathbf{p}' = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}$$

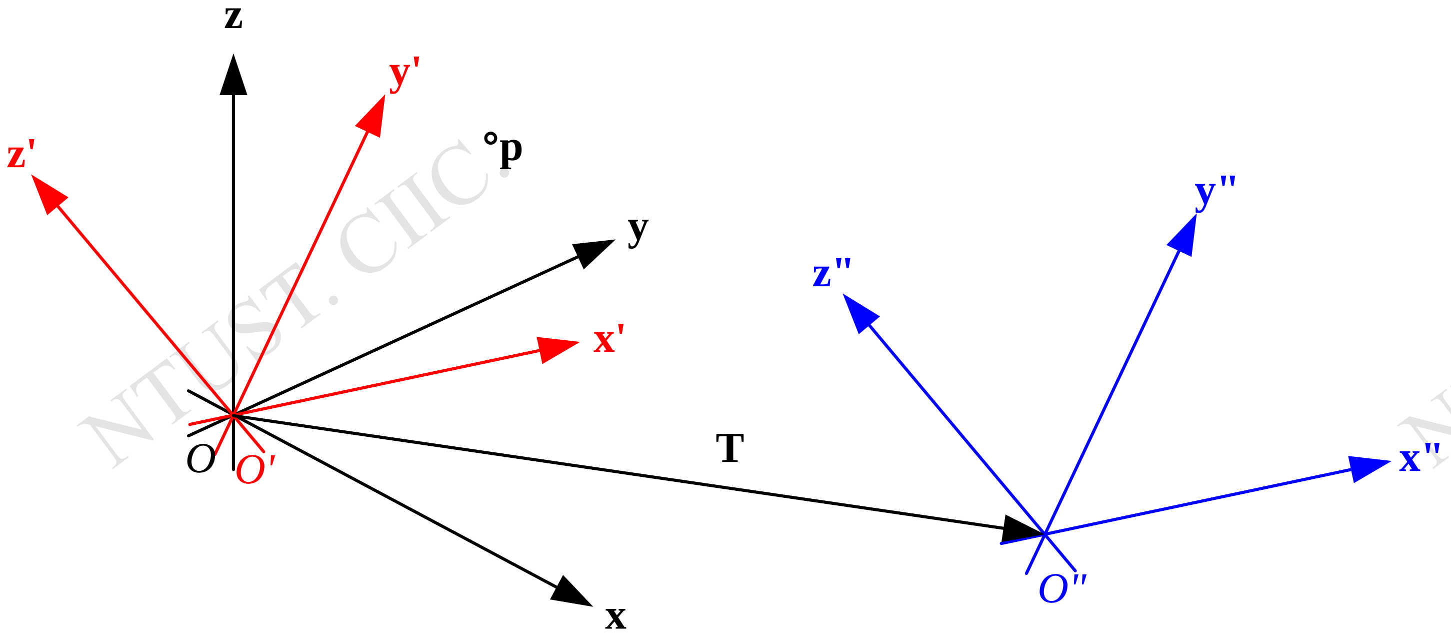
- Step-2: coordinate rotation

$$\mathbf{p}'' = \begin{bmatrix} u_x' & v_x' & w_x' & 0 \\ u_y' & v_y' & w_y' & 0 \\ u_z' & v_z' & w_z' & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}' = \begin{bmatrix} u_x' & v_x' & w_x' & 0 \\ u_y' & v_y' & w_y' & 0 \\ u_z' & v_z' & w_z' & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}$$

3D coordinate transformation—cont.

- Conclude:

$$\mathbf{p''} = (\mathbf{TR})^{-1} \cdot \mathbf{p}$$

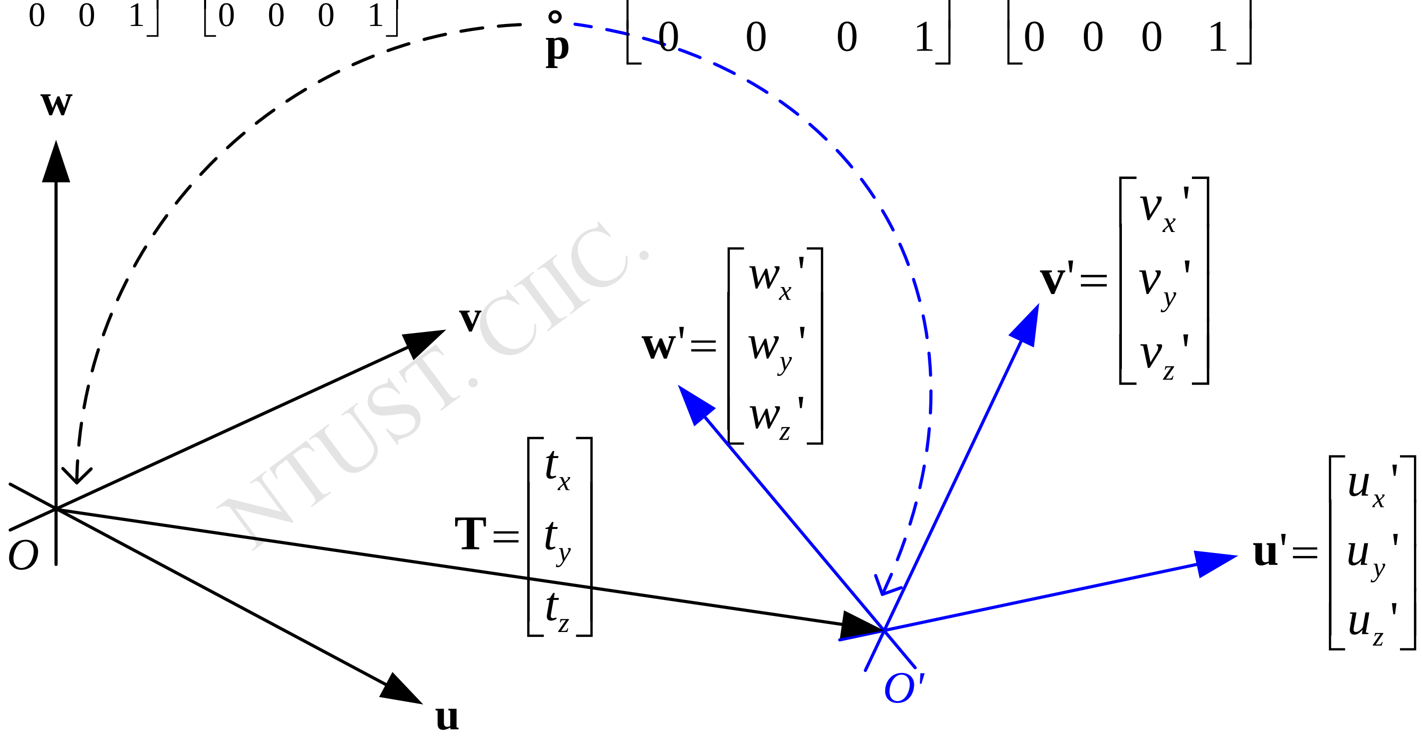


3D coordinate transformation—cont.

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}^{-1}\cdot\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}^{-1}\cdot\mathbf{p}$$

$$\begin{bmatrix}u_x'&v_x'&w_x'&0\\u_y'&v_y'&w_y'&0\\u_z'&v_z'&w_z'&0\\0&0&0&1\end{bmatrix}^{-1}\cdot\begin{bmatrix}1&0&0&t_x\\0&1&0&t_y\\0&0&1&t_z\\0&0&0&1\end{bmatrix}^{-1}\cdot\mathbf{p}$$

$$\begin{bmatrix}u_x'&v_x'&w_x'&t_x\\u_y'&v_y'&w_y'&t_y\\u_z'&v_z'&w_z'&t_z\\0&0&0&1\end{bmatrix}^{-1}\cdot\mathbf{p}$$



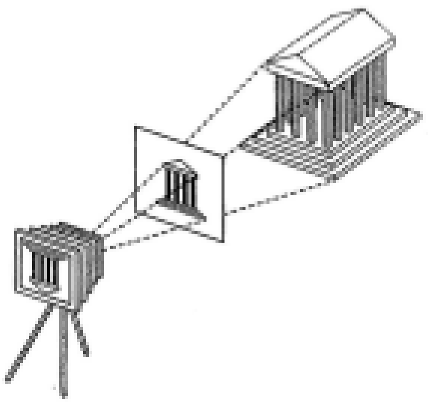
3D coordinate transformation—cont.

- Summary

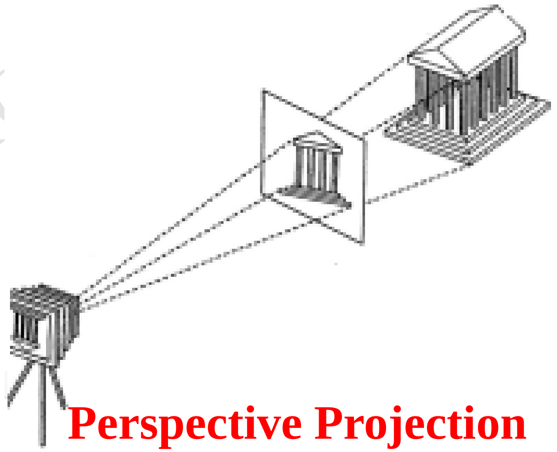
$$\begin{bmatrix} u_x' & v_x' & w_x' & 0 \\ u_y' & v_y' & w_y' & 0 \\ u_z' & v_z' & w_z' & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \mathbf{p} = \boxed{\begin{bmatrix} u_x' & v_x' & w_x' & t_x \\ u_y' & v_y' & w_y' & t_y \\ u_z' & v_z' & w_z' & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}} \cdot \mathbf{p}$$

Viewing from a virtual camera

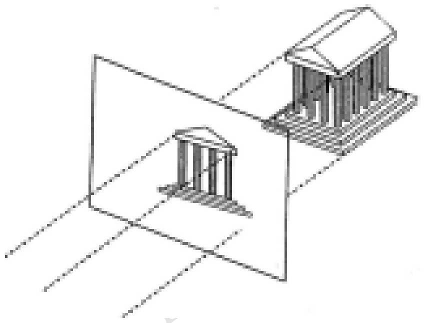
- How to describe a “camera”, and how cameras work
  - Camera parameter
    - Extrinsic parameter: where camera is (coordinate transformation issue)
    - Intrinsic parameter: what lens it is (projection issue)



Perspective Projection (strong)



Perspective Projection (weak)

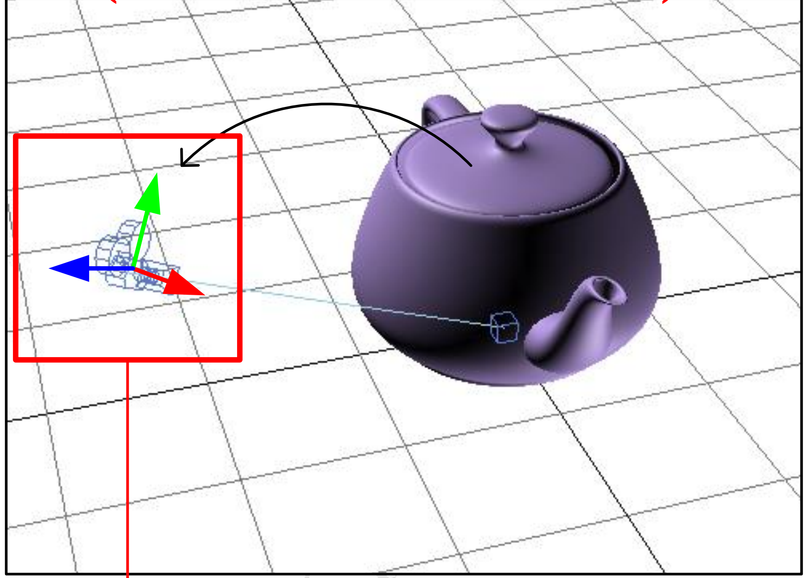


Parallel Projection

Picture from book: Interactive Computer Graphics

Viewing from a virtual camera

camera position
(coordinate transformation)



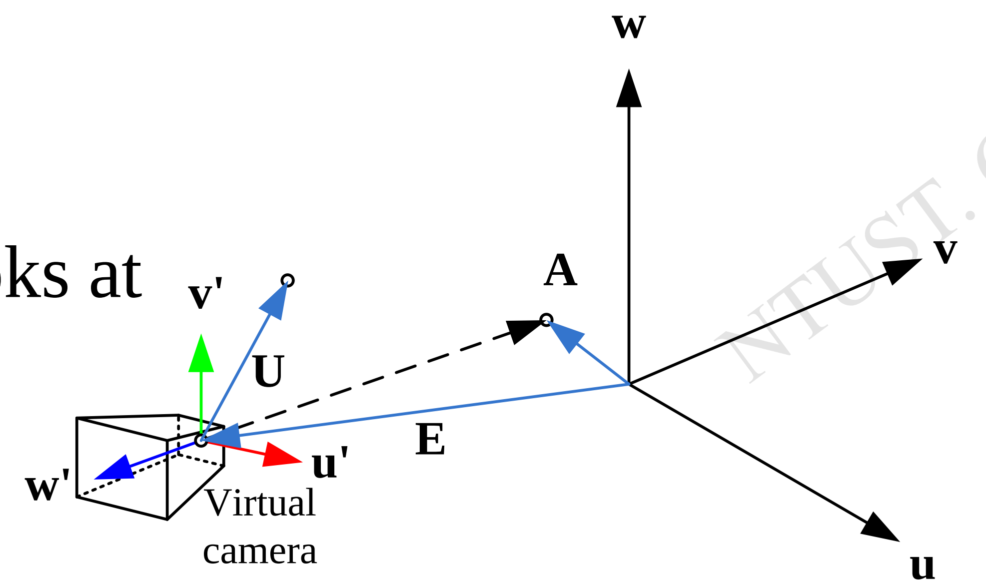
focus length
(projection onto image)

Where the camera is (matrix form)
=extrinsic parameter
(observe the world at the camera position)
Only involve the relation (no image yet)

How camera looks at (matrix form as well)
=intrinsic parameter
Involve "watching", says, an image is formed

3D coordinate and camera transformations

- Camera transformation is one kind of coordinate transformation. It defines a specific coordinate called camera coordinate.
- To describe where camera is needs at least three vectors. Formally, in openGL, there are “eye/camera position”, “reference point” and “up vector of eye/camera”.
- Here,
- E: eye (or camera) position
- A: reference point, where eye looks at
- U: direction of up vector



3D coordinate and camera transformations

- The definition of "gluLookAt" in openGL for camera transformation.

gluLookAt

The **gluLookAt** function defines a viewing transformation.

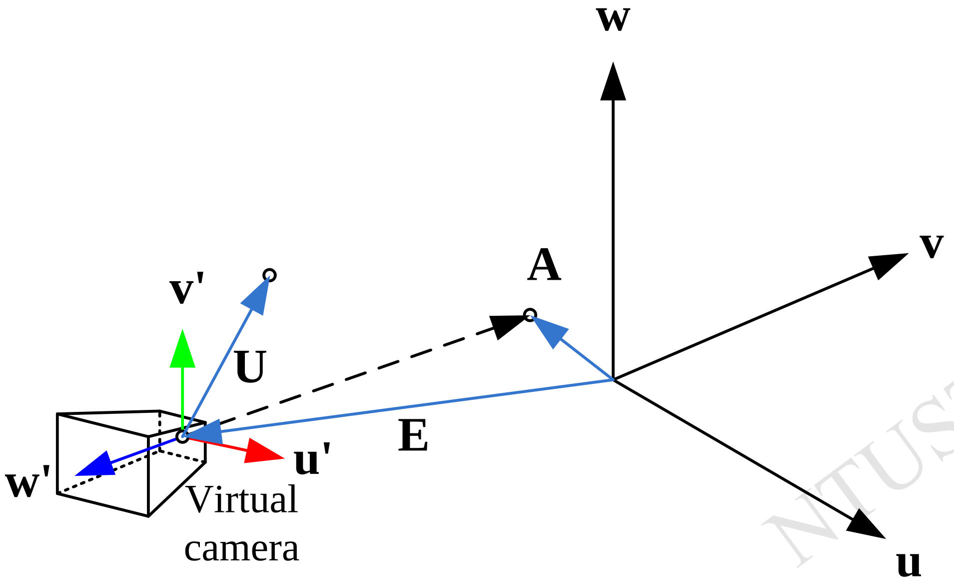
```
void gluLookAt(
  GLdouble eyex,
  GLdouble eyey,
  GLdouble eyez,
  GLdouble centerx,
  GLdouble centery,
  GLdouble centerz,
  GLdouble upx,
  GLdouble upy,
  GLdouble upz
);
```

Parameters

eyex, eyey, eyez
    The position of the eye point.

centerx, centery, centerz
    The position of the reference point.

upx, upy, upz
    The direction of the up vector.



3D camera transformation

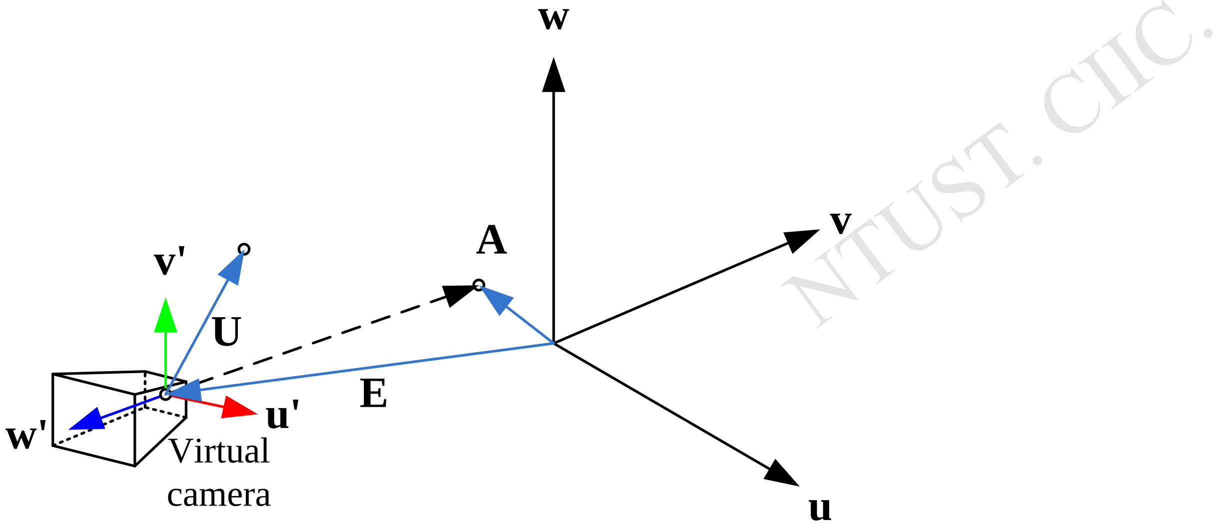
- To determine the camera transformation matrix
- Step-1: find w' from (E-A).
- Step-2: u' from cross product of U and w' (and convert into unit vector)
- Step-3: v' from cross product of w' and u'.

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

$$\mathbf{w}' = Normalize(\mathbf{E} - \mathbf{A})$$

$$\mathbf{u}' = Normalize(\mathbf{U} \times \mathbf{w}')$$

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



3D camera transformation

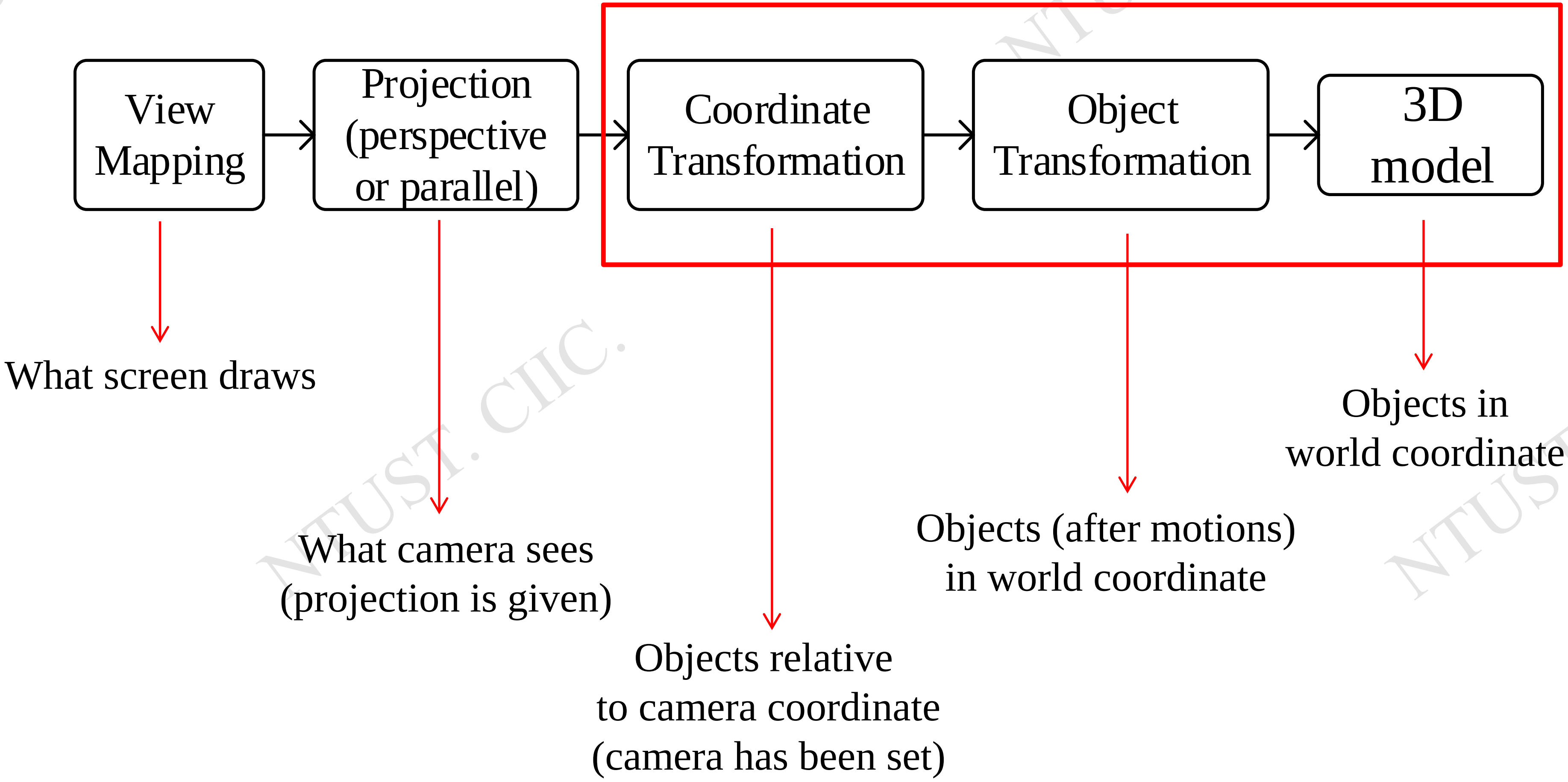
- What we have, when a camera is set.
- In this stage, the values of all points are relative to camera coordinate.

$$\underbrace{\mathbf{p}'}_{\text{After multiplication}}=\underbrace{\begin{bmatrix} u_x' & v_x' & w_x' & E_x \\ u_y' & v_y' & w_y' & E_y \\ u_z' & v_z' & w_z' & E_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \underbrace{\mathbf{p}}_{\text{Object in world coordinate}}}$$

After multiplication:
The value is observed from camera's point of view.
Once again, the object is never changed.

3D camera transformation

- Graphics pipeline



Viewing and projection

- Parallel projection (orthographic projection)
- Perspective projection

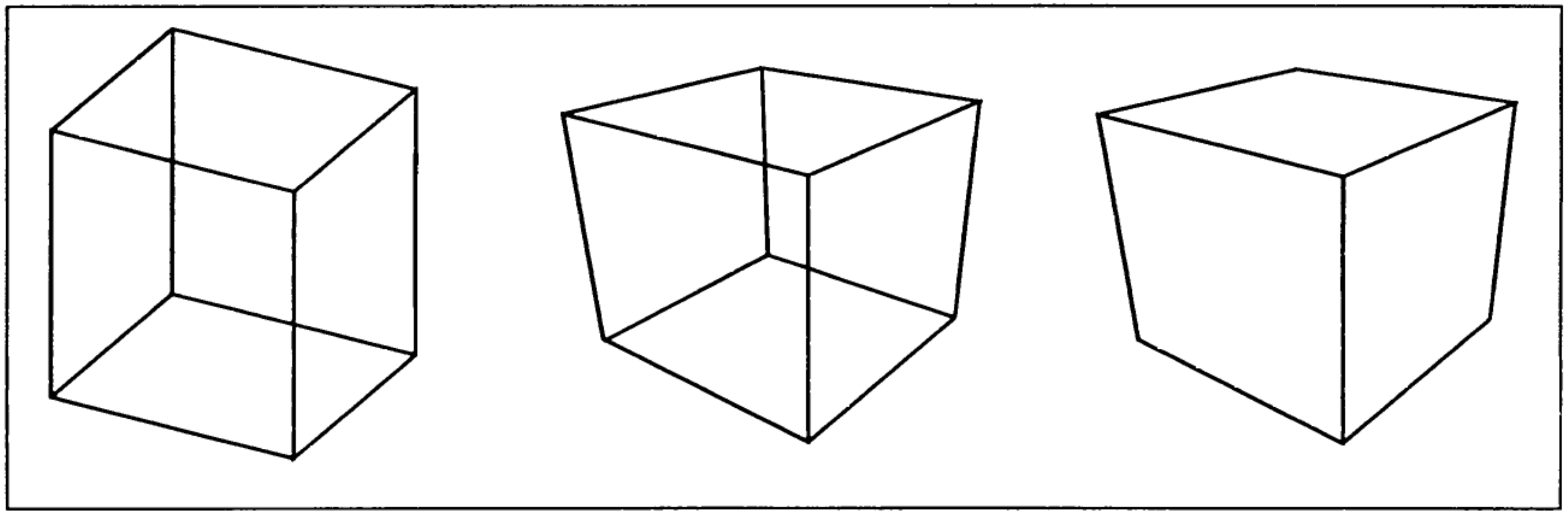
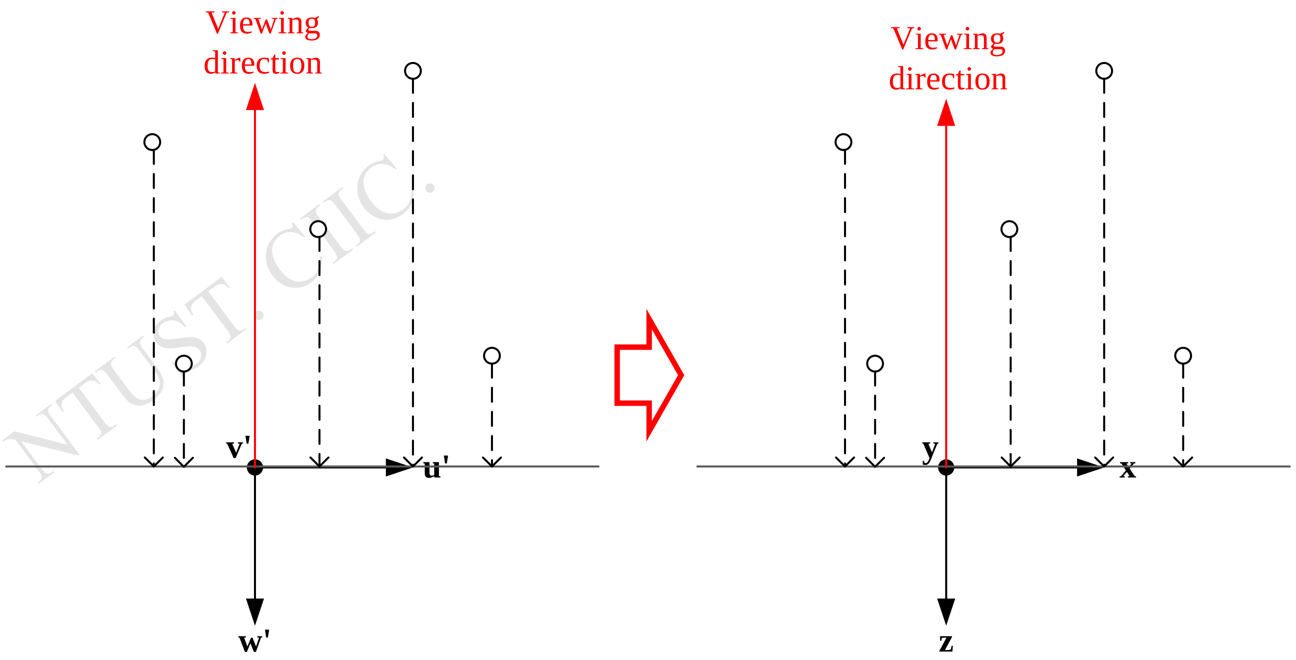


Figure 7.1. Left: orthographic projection. Middle: perspective projection. Right: perspective projection with hidden lines removed.

Picture from book : Fundamentals of Computer Graphics

Viewing and projection: parallel projection

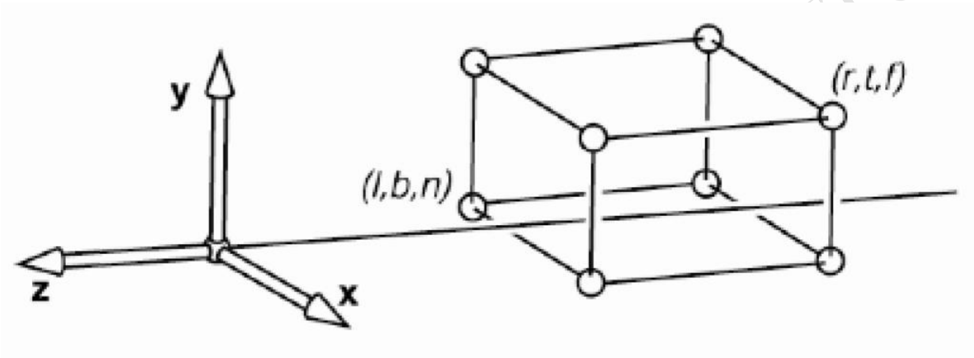
- Parallel (orthographic) projection: usually used in engineering draw.
  - Two steps: 1. compress the view volume into a cube, 2. stretch the cube into the canvas (format: an image with depth).

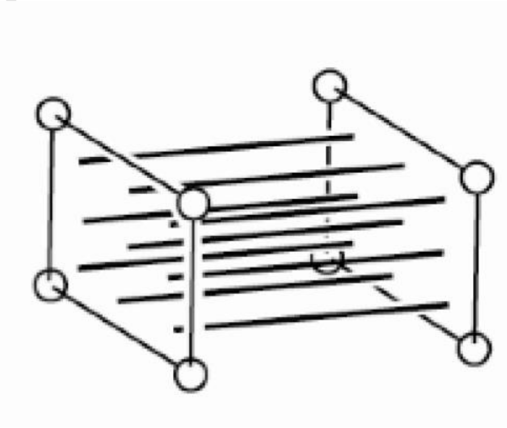


NOTE: all points here are defined in camera coordinate (not world coordinate)

Viewing and parallel projection

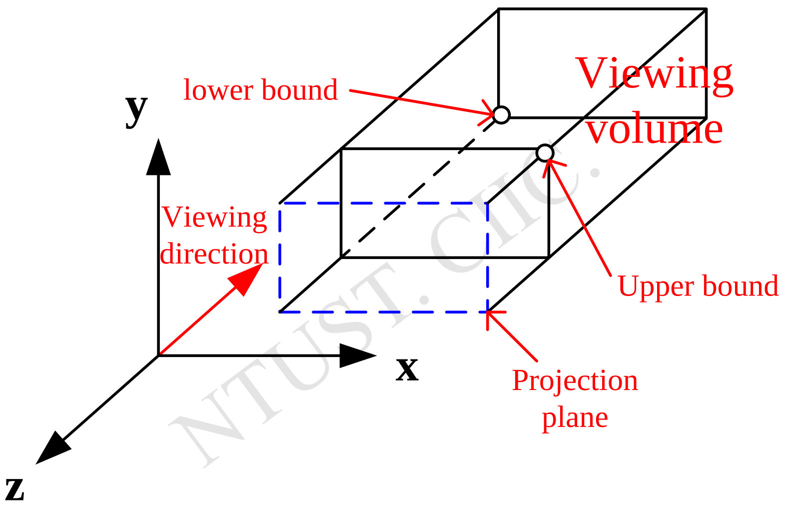
- Overview

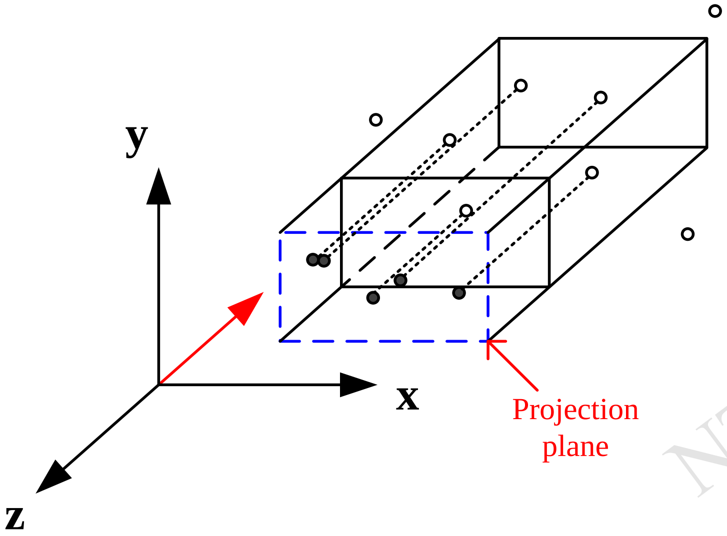




Viewing and projection: parallel projection

- Overview: a viewing volume is required for visualization.
- The goal is to “compress” this volume into a cube (says normalized viewing volume).

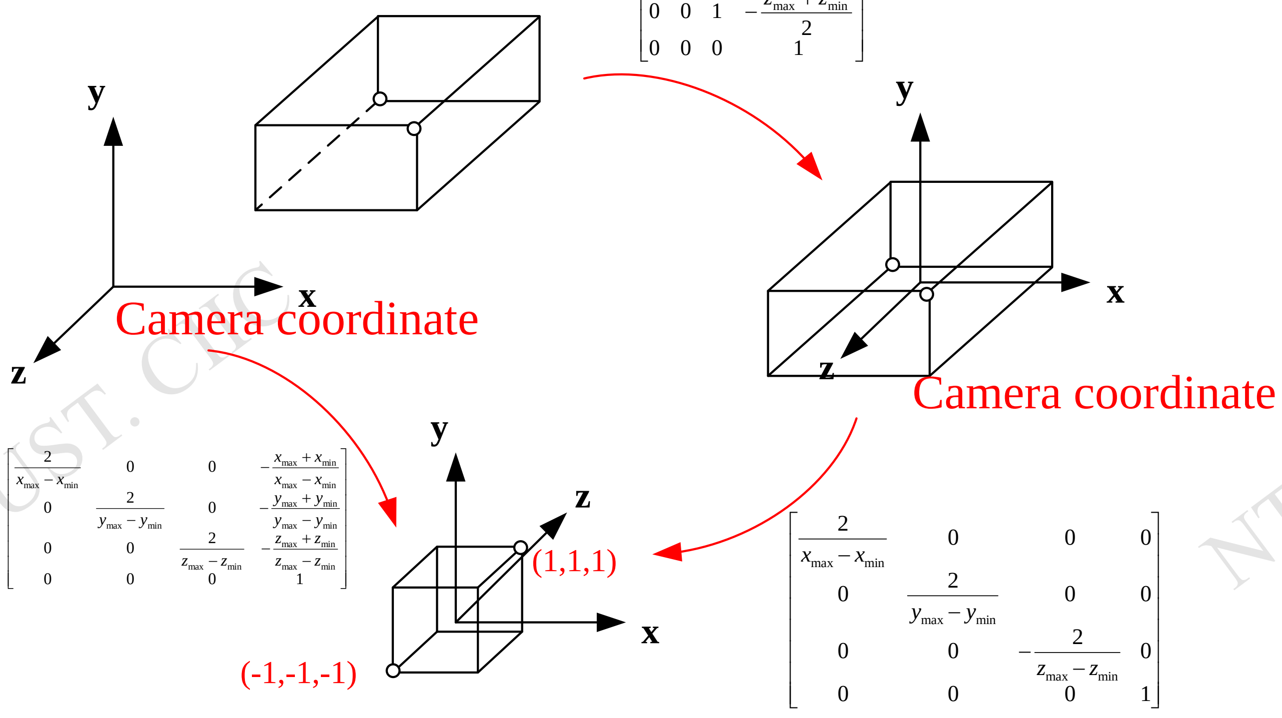




NOTE:
1. All points are converted into camera coordinate.
2. In general, camera center in not necessary to pass through projection-plane center.

Viewing and parallel projection

- Step-1: compress into a normalized viewing volume



$$\begin{bmatrix} 1 & 0 & 0 & -\frac{x_{max}+x_{min}}{2} \\ 0 & 1 & 0 & -\frac{y_{max}+y_{min}}{2} \\ 0 & 0 & 1 & -\frac{z_{max}+z_{min}}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

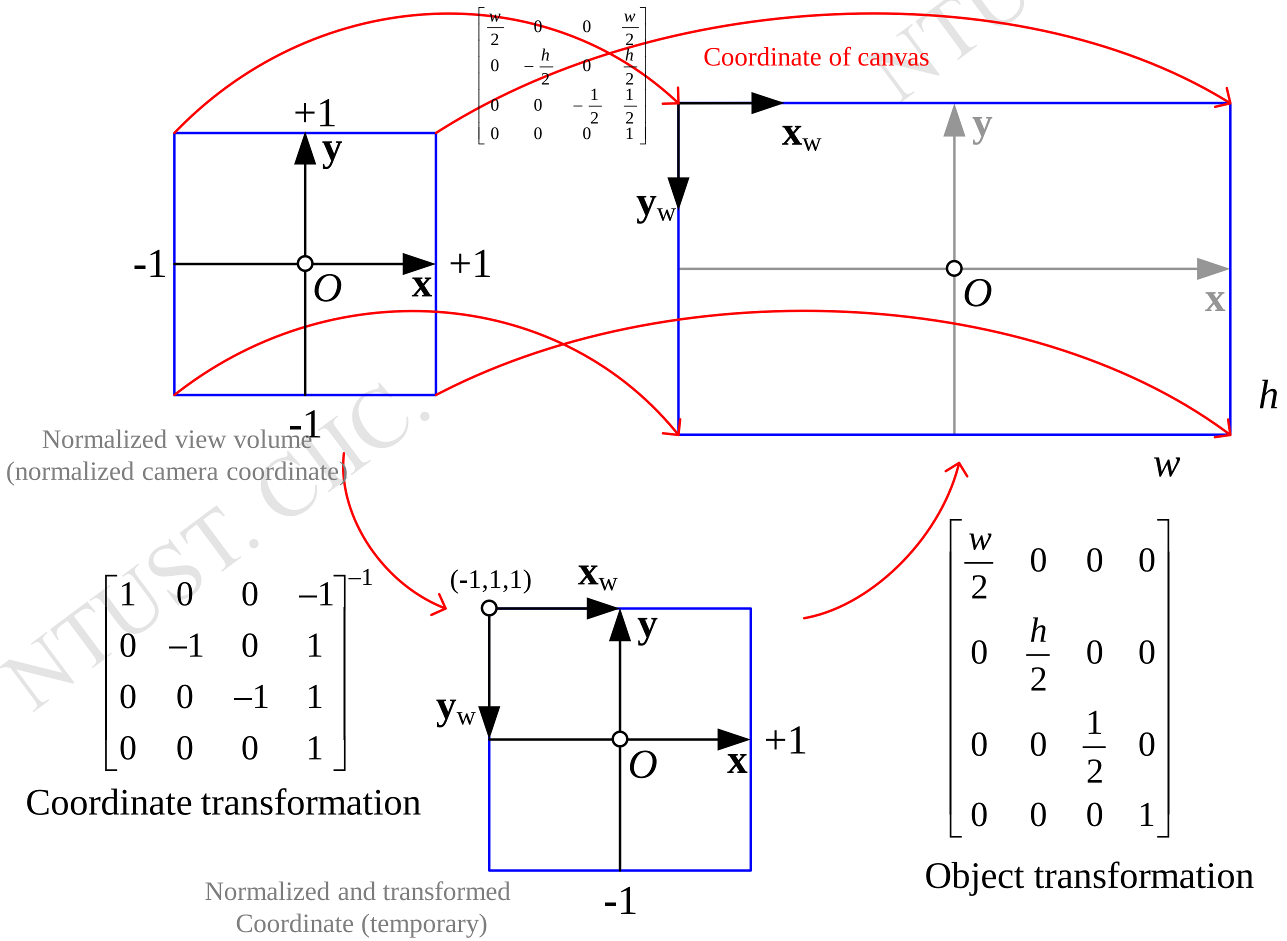
$$\begin{bmatrix} \frac{2}{x_{max}-x_{min}} & 0 & 0 & 0 \\ 0 & \frac{2}{y_{max}-y_{min}} & 0 & 0 \\ 0 & 0 & -\frac{2}{z_{max}-z_{min}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{2}{x_{max}-x_{min}} & 0 & 0 & -\frac{x_{max}+x_{min}}{x_{max}-x_{min}} \\ 0 & \frac{2}{y_{max}-y_{min}} & 0 & -\frac{y_{max}+y_{min}}{y_{max}-y_{min}} \\ 0 & 0 & \frac{2}{z_{max}-z_{min}} & -\frac{z_{max}+z_{min}}{z_{max}-z_{min}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Another coordinate (similar to camera coordinate)
(actually, a normalized viewing volume)

Viewing and parallel projection

- Step-2: stretch the cube into the canvas which is on the screen.



Viewing and parallel projection

- Summarize two steps.
- Finally, a conversion form is obtained for “viewing”.
- Step-1: (glOrtho in openGL)

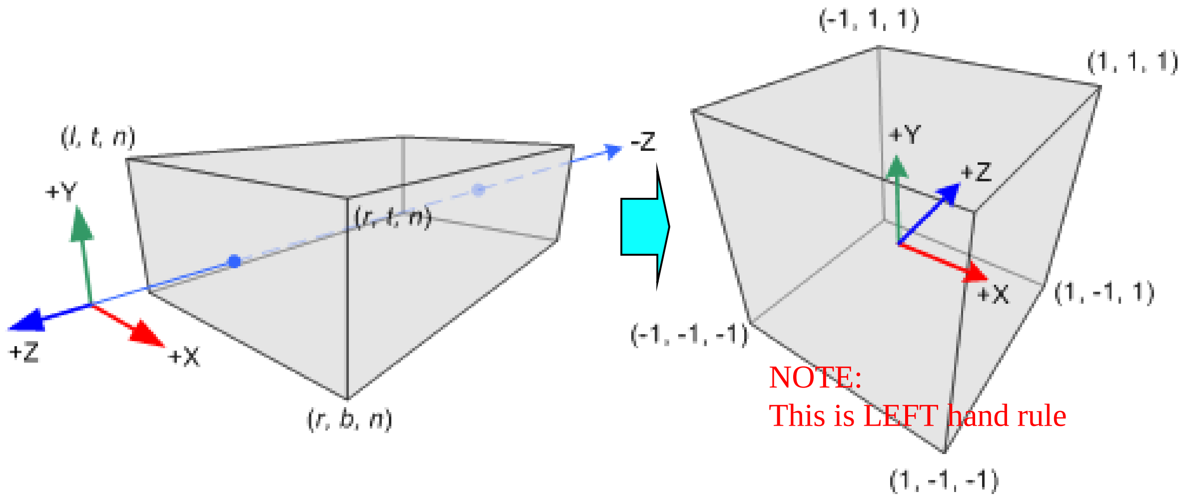
$$\mathbf{p}' = \begin{bmatrix} \frac{2}{x_{\max}-x_{\min}} & 0 & 0 & 0 \\ 0 & \frac{2}{y_{\max}-y_{\min}} & 0 & 0 \\ 0 & 0 & \frac{2}{z_{\max}-z_{\min}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -\frac{x_{\max}+x_{\min}}{2} \\ 0 & 1 & 0 & -\frac{y_{\max}+y_{\min}}{2} \\ 0 & 0 & 1 & -\frac{z_{\max}+z_{\min}}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p} = \begin{bmatrix} \frac{2}{x_{\max}-x_{\min}} & 0 & 0 & -\frac{x_{\max}+x_{\min}}{x_{\max}-x_{\min}} \\ 0 & \frac{2}{y_{\max}-y_{\min}} & 0 & -\frac{y_{\max}+y_{\min}}{y_{\max}-y_{\min}} \\ 0 & 0 & \frac{2}{z_{\max}-z_{\min}} & -\frac{z_{\max}+z_{\min}}{z_{\max}-z_{\min}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p}$$

- Step-2: (glViewport in openGL)

$$\mathbf{p}'' = \begin{bmatrix} \frac{w}{2} & 0 & 0 & 0 \\ 0 & \frac{h}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \mathbf{p}' = \begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & -\frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{p}'$$

Viewing and parallel projection

- Schematic of compressing the viewing volume into to a normalized cube.
- In openGL, the corresponding function is “glOrtho”.

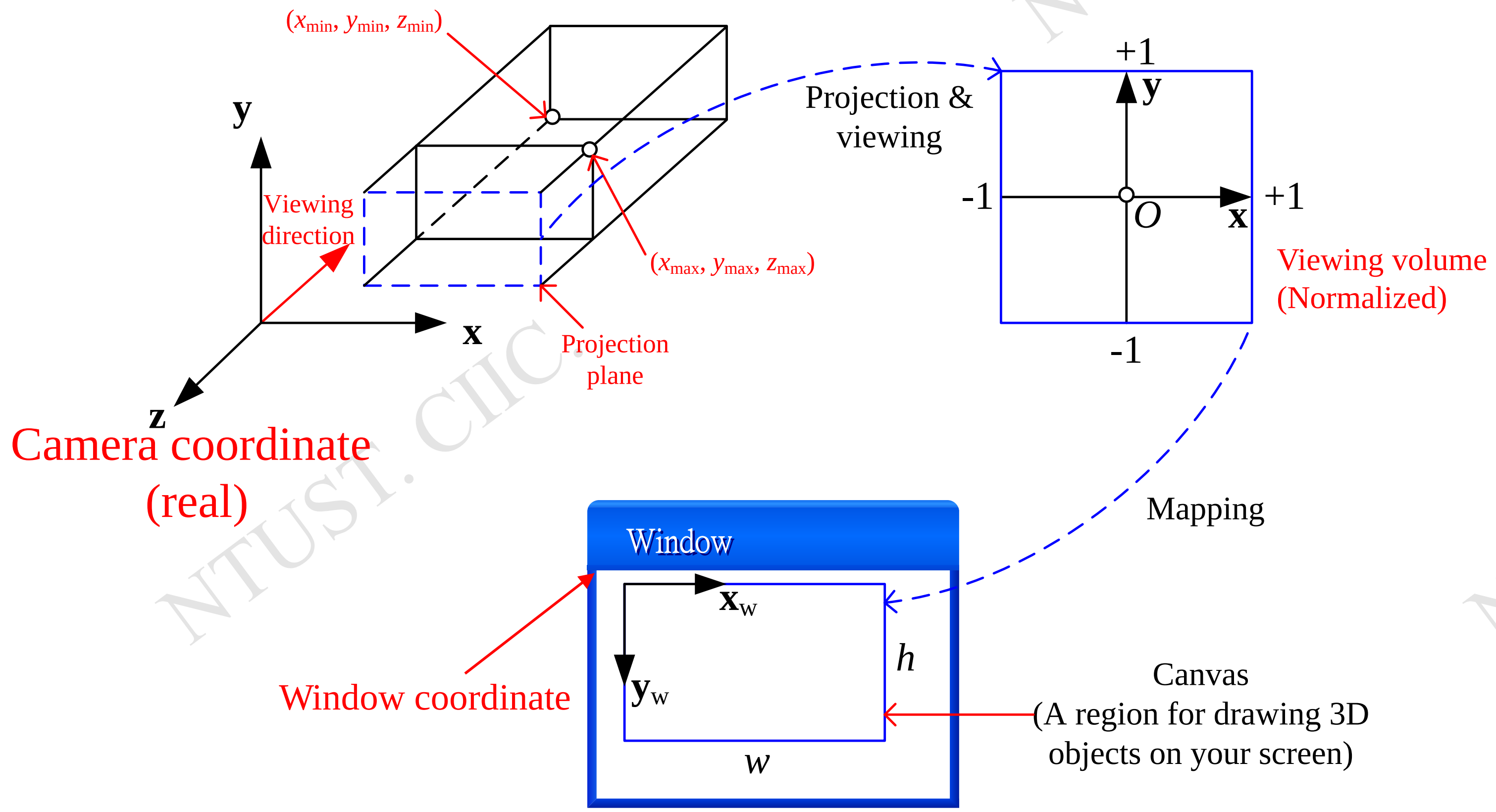


Orthographic Volume and Normalized Device Coordinates (NDC)

http://www.songho.ca/opengl/gl_projectionmatrix.html

Viewing and parallel projection (overall)

- Summary of two steps



Viewing and parallel projection

- Summary: Viewing and parallel projection in graphics pipeline

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \boxed{\begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & -\frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}} \boxed{\begin{bmatrix} \frac{2}{x_{\max}-x_{\min}} & 0 & 0 & 0 \\ 0 & \frac{2}{y_{\max}-y_{\min}} & 0 & 0 \\ 0 & 0 & \frac{2}{z_{\max}-z_{\min}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -\frac{x_{\max}+x_{\min}}{2} \\ 0 & 1 & 0 & -\frac{y_{\max}+y_{\min}}{2} \\ 0 & 0 & 1 & -\frac{z_{\max}+z_{\min}}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x_{ca} \\ y_{ca} \\ z_{ca} \\ 1 \end{bmatrix}$$

View Mapping → Projection (parallel) → Coordinate Transformation → Object Transformation → 3D model

Viewing and parallel projection

- Summary: Viewing and parallel projection in graphics pipeline

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & -\frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{x_{\max}-x_{\min}} & 0 & 0 & -\frac{x_{\max}+x_{\min}}{x_{\max}-x_{\min}} \\ 0 & \frac{2}{y_{\max}-y_{\min}} & 0 & -\frac{y_{\max}+y_{\min}}{y_{\max}-y_{\min}} \\ 0 & 0 & \frac{2}{z_{\max}-z_{\min}} & -\frac{z_{\max}+z_{\min}}{z_{\max}-z_{\min}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ca} \\ y_{ca} \\ z_{ca} \\ 1 \end{bmatrix}$$

Viewing and parallel projection

- Remind the notation

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & -\frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{x_{max} - x_{min}} & 0 & 0 & -\frac{x_{max} + x_{min}}{x_{max} - x_{min}} \\ 0 & \frac{2}{y_{max} - y_{min}} & 0 & -\frac{y_{max} + y_{min}}{y_{max} - y_{min}} \\ 0 & 0 & \frac{2}{z_{max} - z_{min}} & -\frac{z_{max} + z_{min}}{z_{max} - z_{min}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ca} \\ y_{ca} \\ z_{ca} \\ 1 \end{bmatrix}$$

The 3D points defined in camera coordinate system

The viewing volume's upper-bound and lower-bound points in camera coordinate

The canvas image (screen) size including **width** and **height**.

The drawn position of 3D point on the image (screen) including **2D** and **depth**.

Viewing and parallel projection

- Remind: many textbooks do NOT have an identical notation. Please note the definition among them.

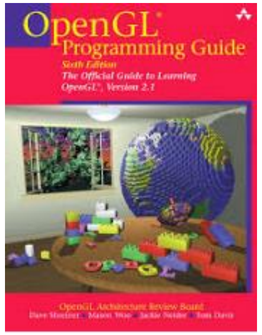
$$\begin{pmatrix} \frac{2}{right-left} & 0 & 0 & t_x \\ 0 & \frac{2}{top-bottom} & 0 & t_y \\ 0 & 0 & \frac{-2}{far-near} & t_z \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

where

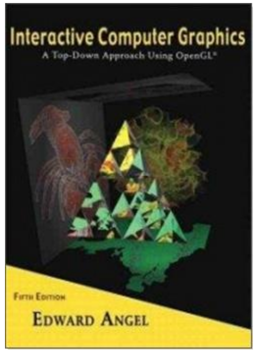
$$t_x = -\frac{right+left}{right-left}$$

$$t_y = -\frac{top+bottom}{top-bottom}$$

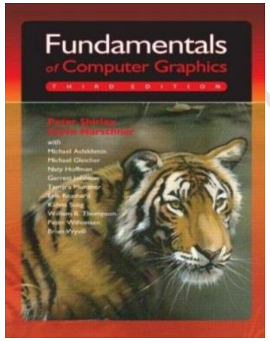
$$t_z = -\frac{far+near}{far-near}$$



$$\mathbf{P} = \mathbf{ST} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{left+right}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\begin{bmatrix} \frac{2}{x_{max}-x_{min}} & 0 & 0 & -\frac{x_{max}+x_{min}}{x_{max}-x_{min}} \\ 0 & \frac{2}{y_{max}-y_{min}} & 0 & -\frac{y_{max}+y_{min}}{y_{max}-y_{min}} \\ 0 & 0 & \frac{2}{z_{max}-z_{min}} & -\frac{z_{max}+z_{min}}{z_{max}-z_{min}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



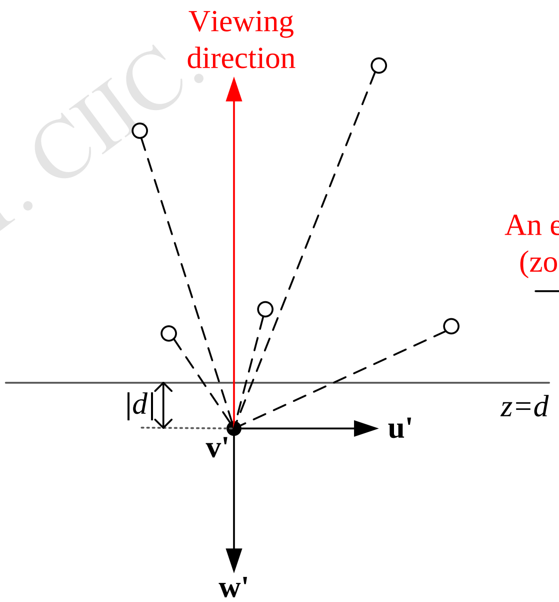
Viewing and perspective projection

- Perspective projection is one kind of popular projection methods similar to pin-hole camera. It is widely used in many fields based on computer visualization.
- Suppose one “projection plane” is in front at distance d of the camera.

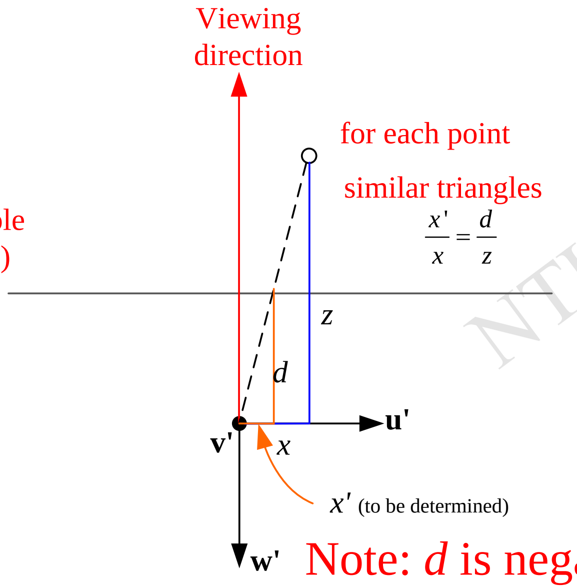
$$x' = \frac{d}{z}x$$

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

$$z' = d$$



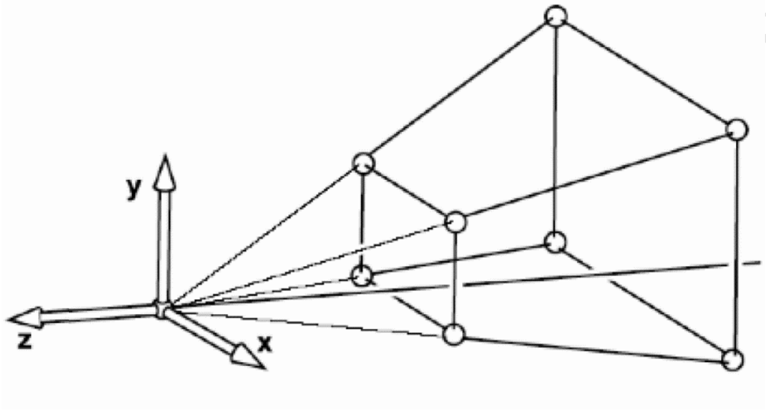
An example
(zoom in)

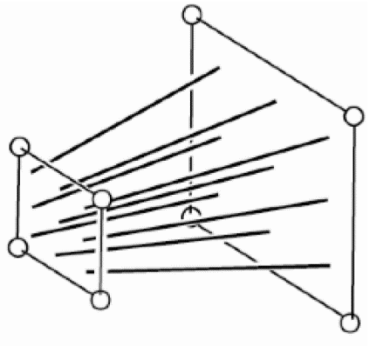


Note: *d* is negative

Viewing and perspective projection

- Overview





Viewing and perspective projection

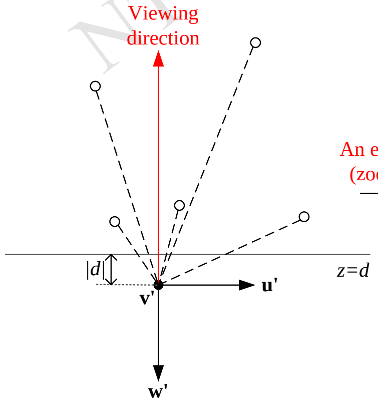
- How the image formed from perspective projection

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} xd/z \\ yd/z \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} x_h/w_h \\ y_h/w_h \\ z_h/w_h \\ 1 \end{bmatrix} \approx \begin{bmatrix} x_h \\ y_h \\ z_h \\ w_h \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}$$

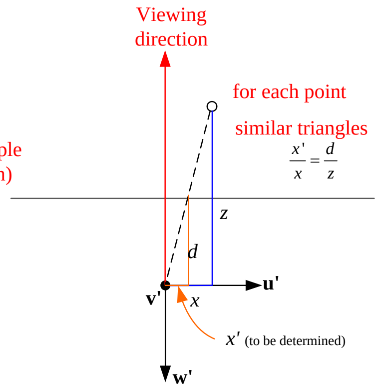
Points at the plane z=-d
(homogenous representation)

Points at the plane z=-d
(normalized, the desired result)

Points in camera coordinate



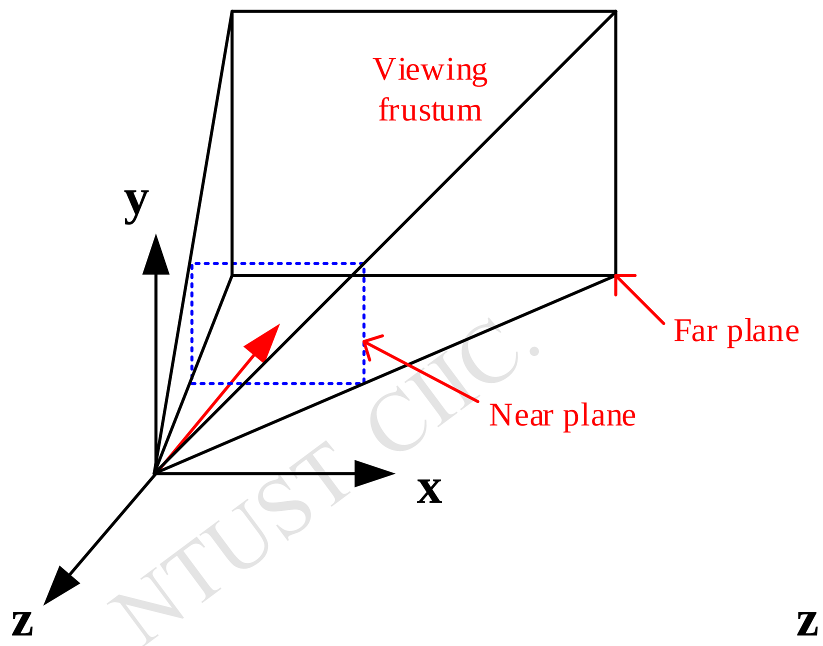
An example (zoom in)

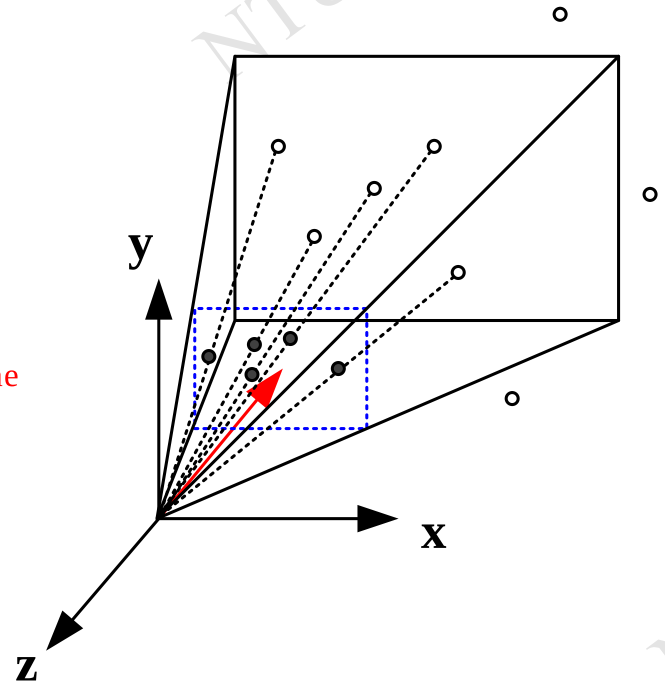


Note: d is negative

Viewing and perspective projection

- The “space” for visualization will be a frustum (similar to pyramid)

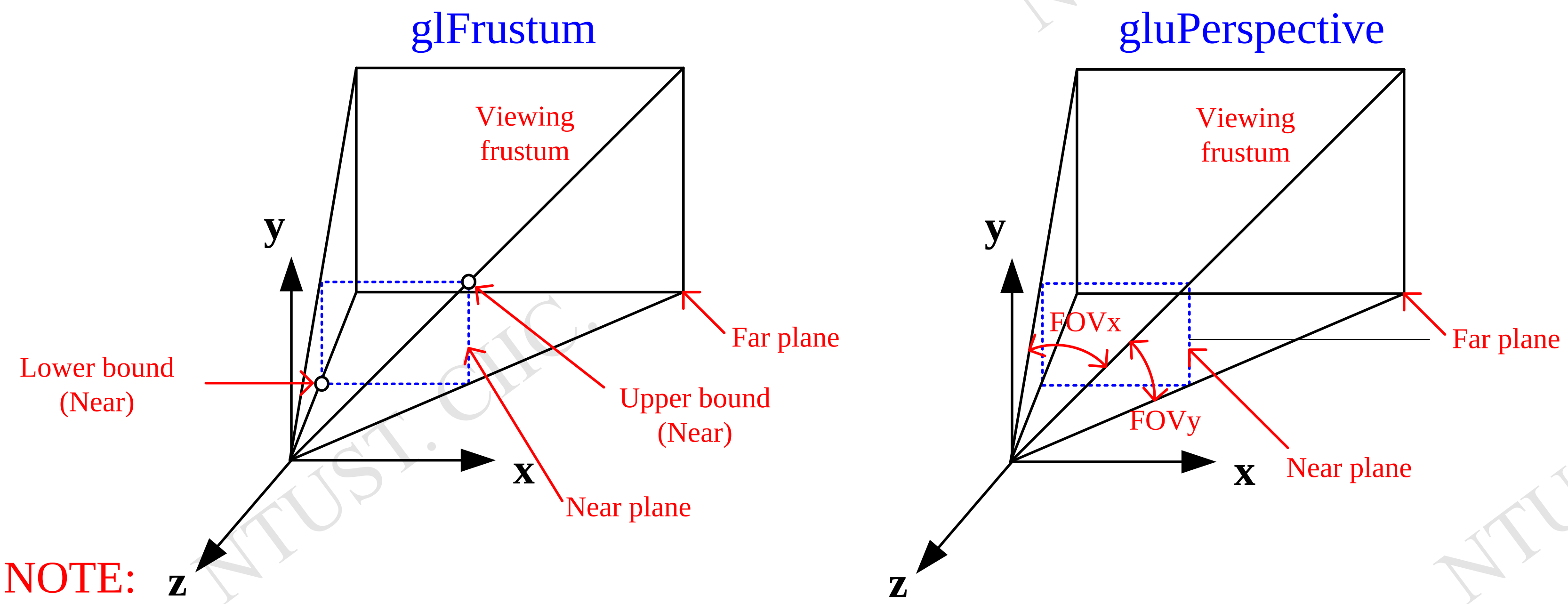




NOTE: all points are converted into camera coordinate.

Viewing and perspective projection

- There are two description types for viewing frustum
  - glFrustum & gluPerspective (in openGL)

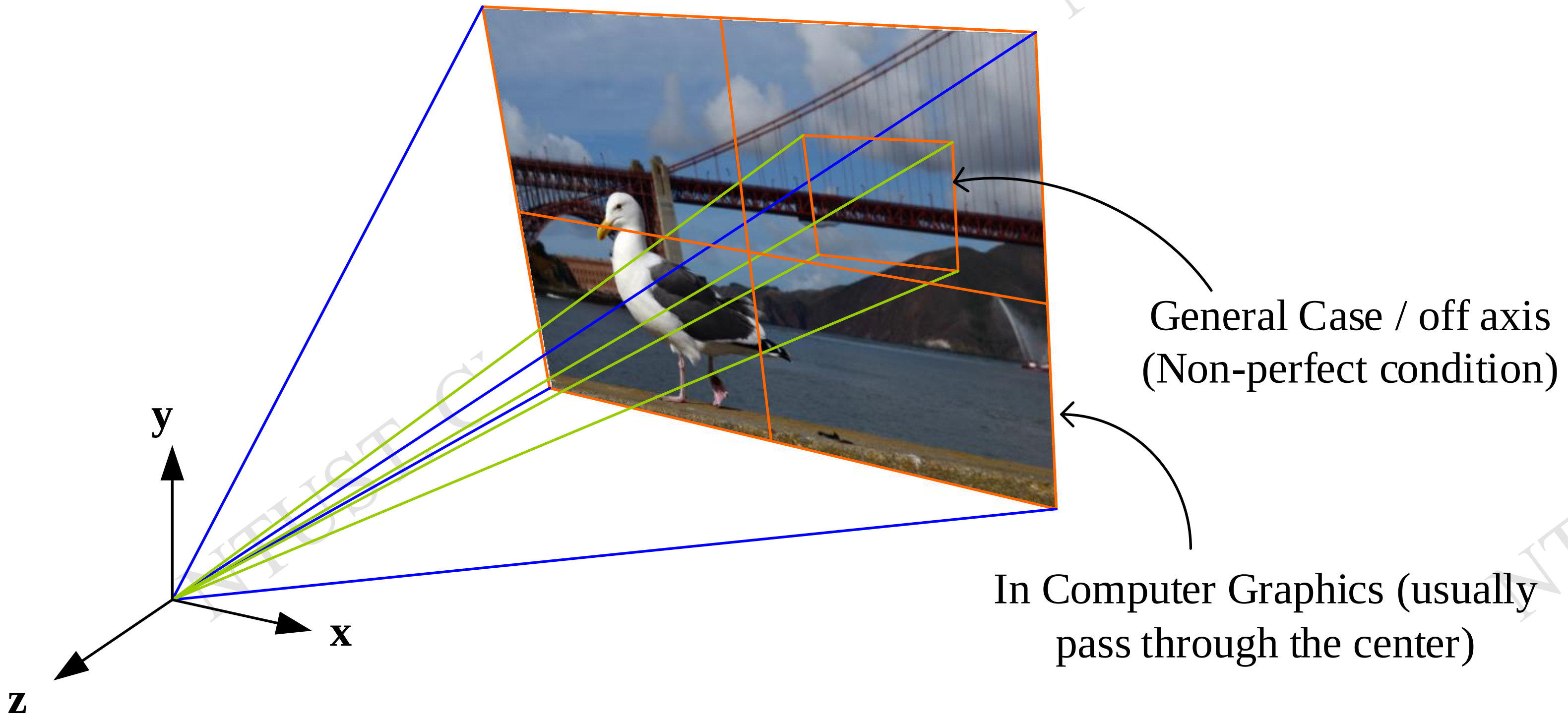


NOTE:

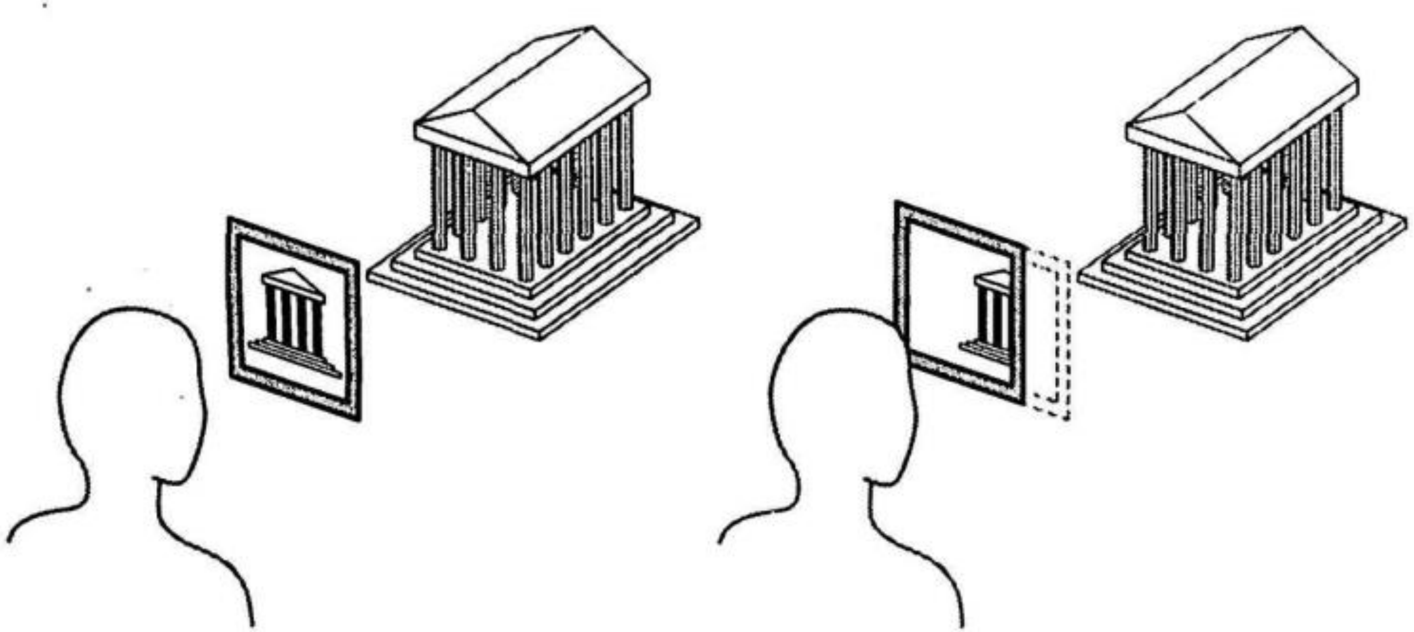
1. All points are converted into camera coordinate.
2. In general, camera center in not necessary to pass through near-plane center.

Viewing and perspective projection

- In general, the centroid of the image is NOT necessary to be the same with optical center.

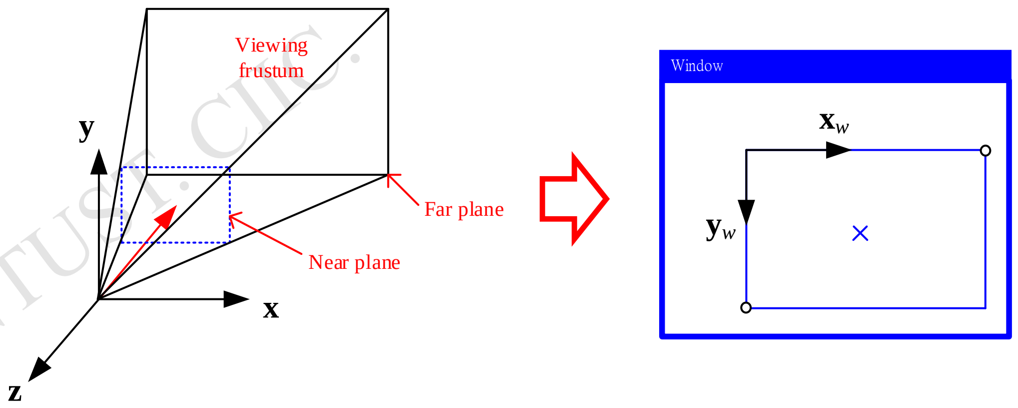


Viewing and perspective projection



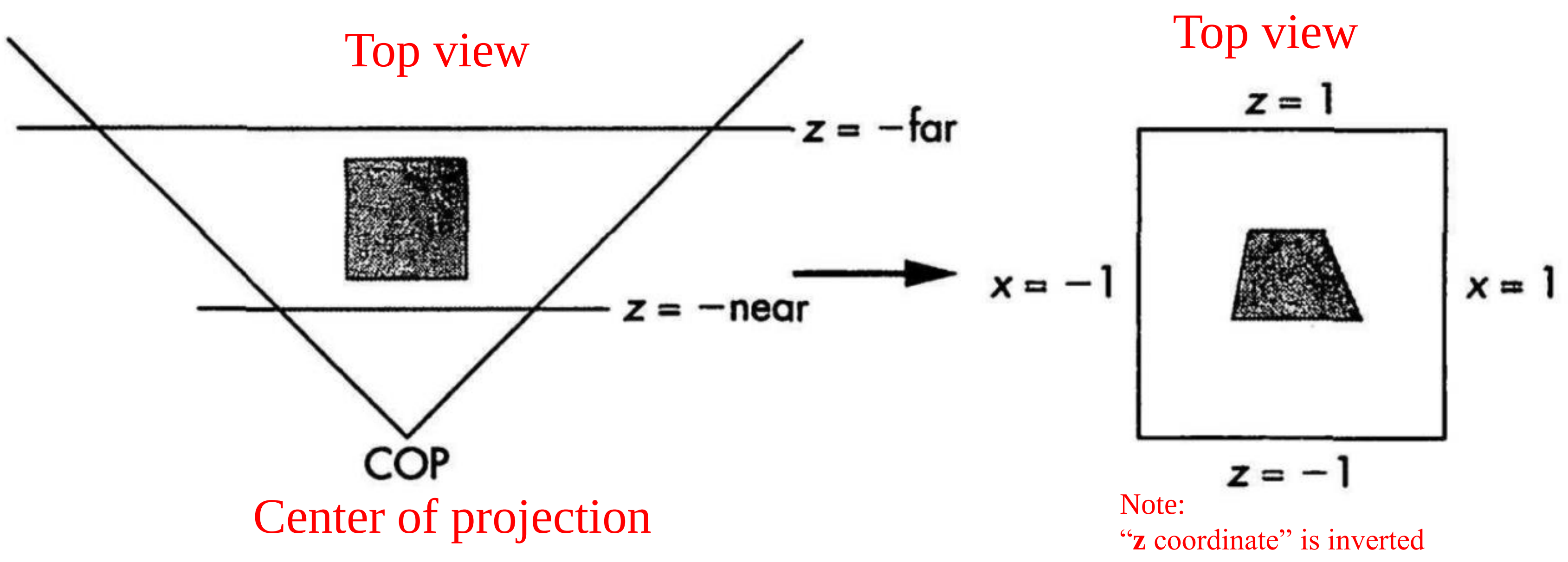
Viewing and perspective projection

- To form an image on screen, two steps are needed
  - Step-1: Compress viewing frustum into a normalized cube.
  - Step-2: Stretch the cube onto the canvas (screen).



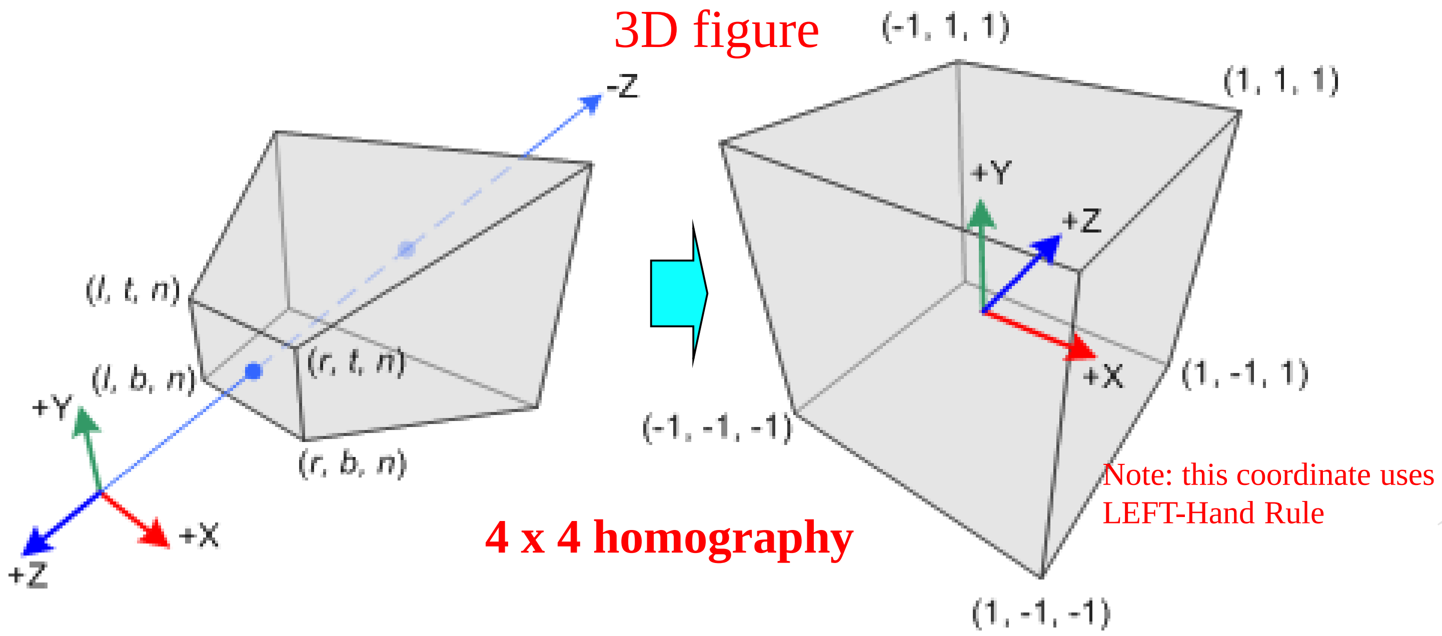
Viewing and perspective projection

- The basic idea is to warp the space into a desired space. However, it is not clear defined in textbook.



Viewing and perspective projection

- The solution to warp is a 4x4 homography mapping operation, which is revealed in computer vision field.



Perspective Frustum and Normalized Device Coordinates (NDC)

http://www.songho.ca/opengl/gl_projectionmatrix.html

Viewing and perspective projection

- Step-1: Compress all into a cube
- (be a formula)

In this lecture

$$\begin{bmatrix} \frac{2z_{\text{near}}}{x_{\max}-x_{\min}} & 0 & \frac{x_{\max}+x_{\min}}{x_{\max}-x_{\min}} & 0 \\ 0 & \frac{2z_{\text{near}}}{y_{\max}-y_{\min}} & \frac{y_{\max}+y_{\min}}{y_{\max}-y_{\min}} & 0 \\ 0 & 0 & \frac{z_{\text{near}}+z_{\text{far}}}{z_{\text{far}}-z_{\text{near}}} & \frac{2z_{\text{near}}z_{\text{far}}}{z_{\text{far}}-z_{\text{near}}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Note: z value is negative

In openGL

glFrustum

The **glFrustum** function multiplies the current matrix by a perspective matrix.

```
void glFrustum(
  GLdouble left,
  GLdouble right,
  GLdouble bottom,
  GLdouble top,
  GLdouble znear,
  GLdouble zfar
);
```

Parameters

left, right
The coordinates for the left and right vertical clipping planes.

bottom, top
The coordinates for the bottom and top horizontal clipping planes.

znear, zfar
The distances to the near and far depth clipping planes. Both distances must be positive.

Remarks

The **glFrustum** function describes a perspective matrix that produces a perspective projection. The (*left*, *bottom*, *znear*) and (*right*, *top*, *znear*) parameters specify the points on the near clipping plane that are mapped to the lower-left and upper-right corners of the window, respectively, assuming that the eye is located at (0, 0, 0). The *zfar* parameter specifies the location of the far clipping plane. Both *znear* and *zfar* must be positive. The corresponding matrix is:

$$\begin{pmatrix} \frac{2\ \text{near}}{\text{right-left}} & 0 & A & 0 \\ 0 & \frac{2\ \text{near}}{\text{top-bottom}} & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$A = \frac{\text{right+left}}{\text{right-left}}$$

$$B = \frac{\text{top+bottom}}{\text{top-bottom}}$$

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

$$D = -\frac{2\ far\ near}{far-near}$$

Note:
values of near and far are both positive

The **glFrustum** function multiplies the current matrix by this matrix, with the result replacing the current matrix. That is, if M is the current matrix and F is the frustum perspective matrix, then **glFrustum** replaces M with M • F.

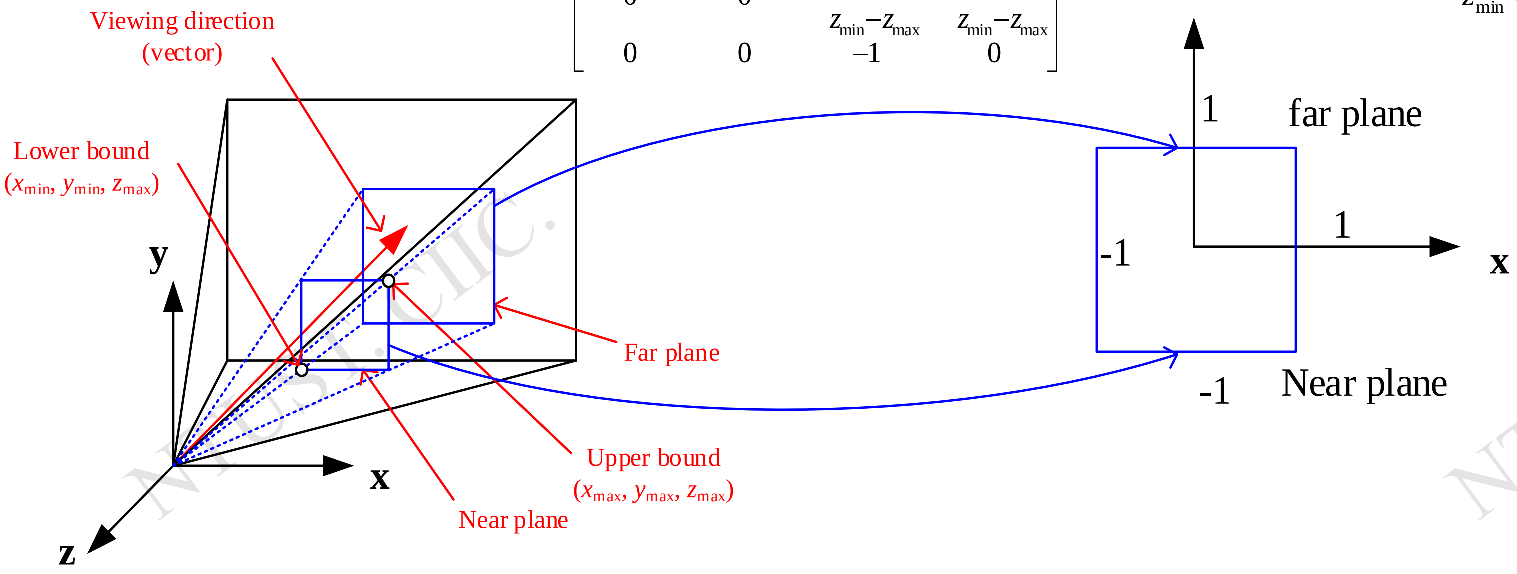
Viewing and perspective projection

- Step-1 cont.:

$$\begin{bmatrix} \frac{-2z_{max}}{x_{max}-x_{min}} & 0 & \frac{x_{max}+x_{min}}{x_{max}-x_{min}} & 0 \\ 0 & \frac{-2z_{max}}{y_{max}-y_{min}} & \frac{y_{max}+y_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & -\frac{z_{min}+z_{max}}{z_{min}-z_{max}} & -\frac{2z_{max}z_{min}}{z_{min}-z_{max}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

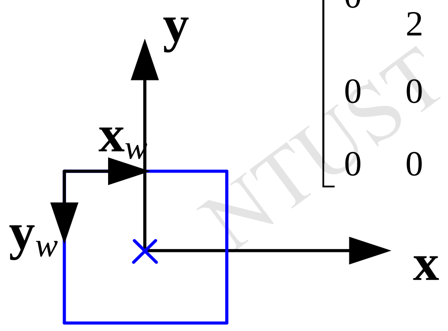
Note: in openGL,
near and far are positive

$$z_{max} = -z_{near}$$
$$z_{min} = -z_{far}$$



Viewing and perspective projection

- Step-2: Stretch the cube to the canvas (screen)



The final canvas width and height

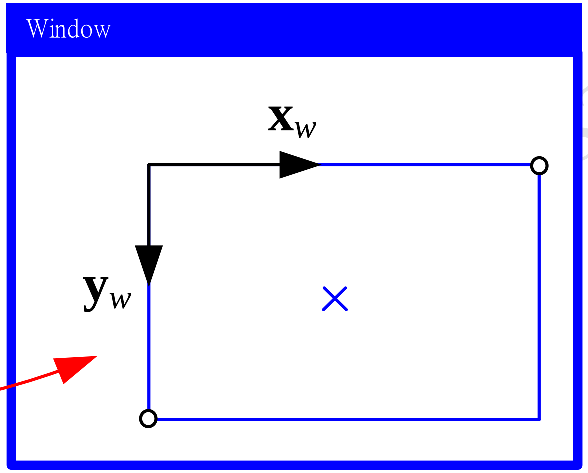
Convert the origin to up-right corner

Convert into RIGHT-hand rule

$$\begin{bmatrix} \frac{w}{2} & 0 & 0 & 0 \\ 0 & \frac{h}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & -\frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composed matrix



Viewing and perspective projection

- Summary
  - Step-1:

$$\begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix} \boxed{\sim=} \begin{bmatrix} \frac{-2z_{\max}}{x_{\max}-x_{\min}} & 0 & \frac{x_{\max}+x_{\min}}{x_{\max}-x_{\min}} & 0 \\ 0 & \frac{-2z_{\max}}{y_{\max}-y_{\min}} & \frac{y_{\max}+y_{\min}}{y_{\max}-y_{\min}} & 0 \\ 0 & 0 & -\frac{z_{\min}+z_{\max}}{z_{\min}-z_{\max}} & \frac{2z_{\max}z_{\min}}{z_{\min}-z_{\max}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \boxed{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}$$

Homogenous conversion

3D points in camera coordinate

  - Step-2:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & -\frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix}$$

Viewing and perspective projection

- Summary: Viewing and perspective projection in graphics pipeline

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & -\frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{-2z_{max}}{x_{max}-x_{min}} & 0 & \frac{x_{max}+x_{min}}{x_{max}-x_{min}} & 0 \\ 0 & \text{free}\ \frac{2z_{max}}{y_{max}-y_{min}} & \frac{y_{max}+y_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & -\frac{z_{min}+z_{max}}{z_{min}-z_{max}} & \frac{2z_{max}z_{min}}{z_{min}-z_{max}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_{ca} \\ y_{ca} \\ z_{ca} \\ 1 \end{bmatrix}$$

View Mapping → Projection (perspective) → Coordinate Transformation → Object Transformation → 3D model

色彩與照明科技研究所
Graduate Institute of
Color and Illumination Technology

Find us on:
facebook