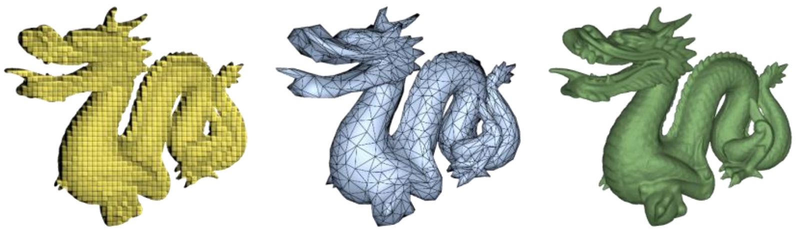
Computer Graphics

Viewing Transformation

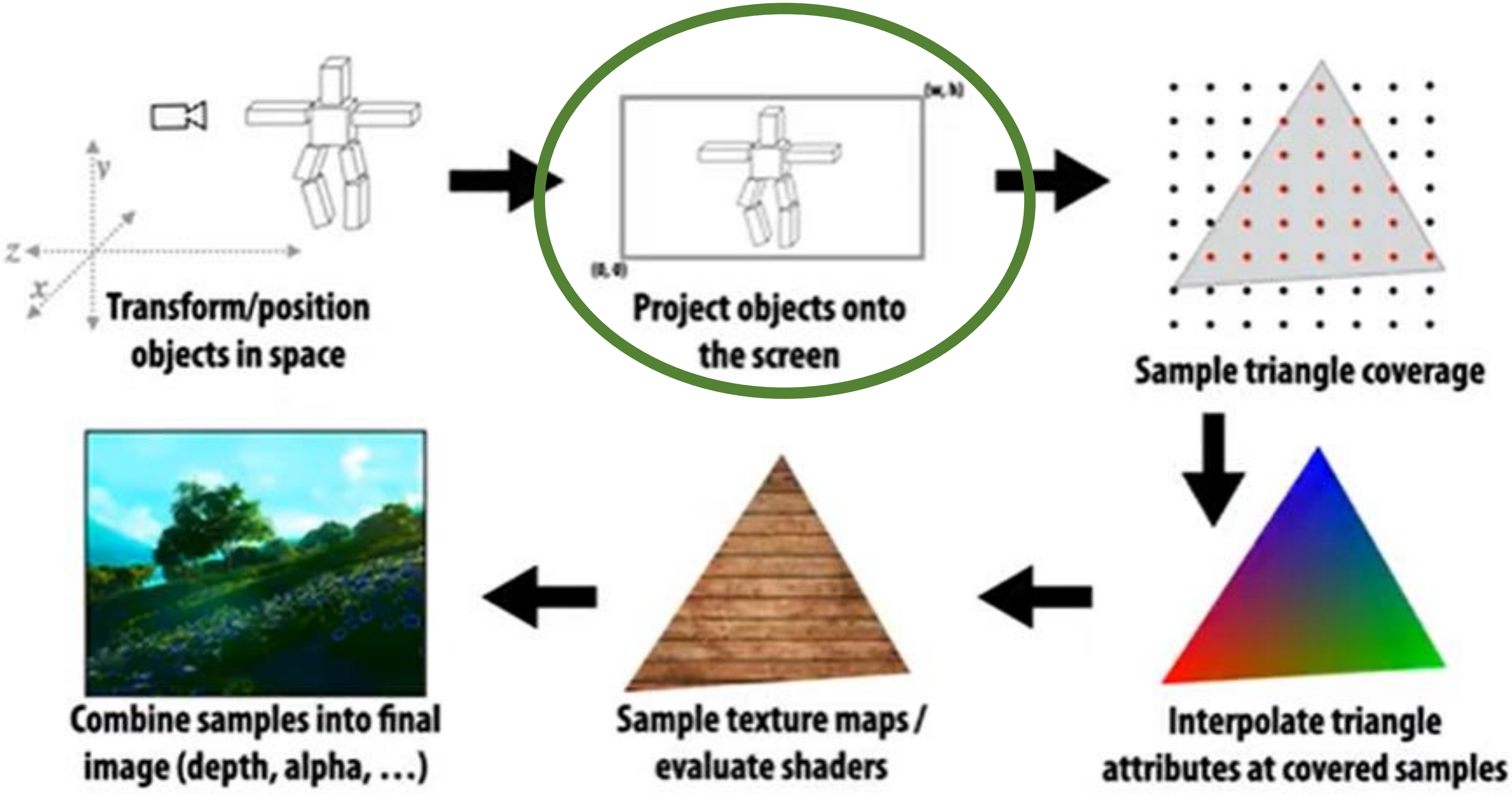
A. Prof. Chengying Gao(高成英)

School of Computer Science and Engineering

Sun Yat-Sen University

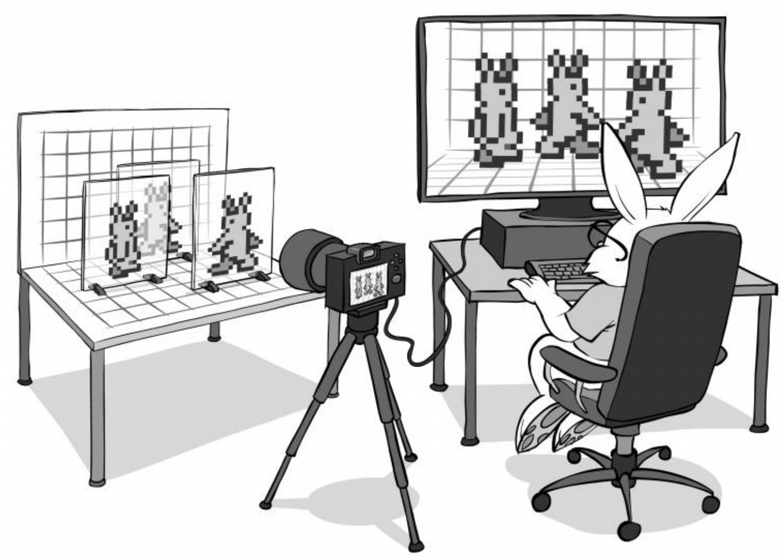


The Rasterization Pipeline



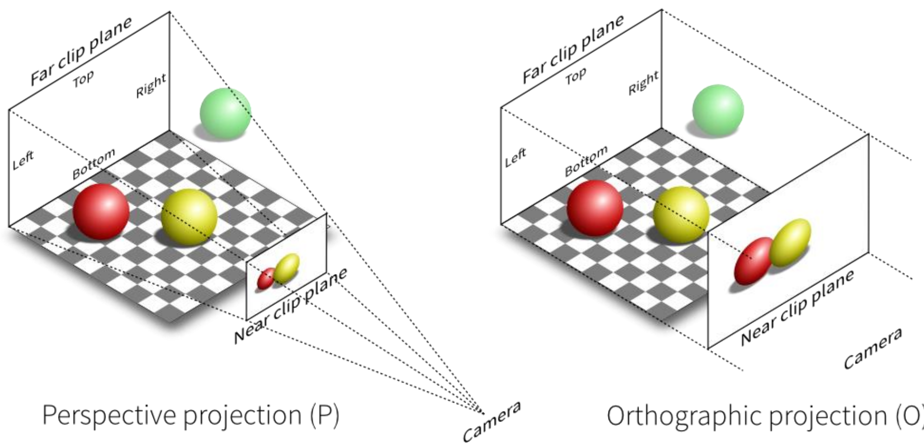
Outline

- Viewing Transformation
  - View / Camera transformation
  - Projection transformation
    - Orthographic projection
    - Perspective projection
- Viewport Transformation



What Is Viewing Transformation（观测变换）

- To display a 3D world onto a 2D screen
  - Additional task of reducing dimensions from 3D to 2D (Projection)
  - 3D viewing transformation is analogous to taking a picture with a camera



Perspective projection (P)

Orthographic projection (O)

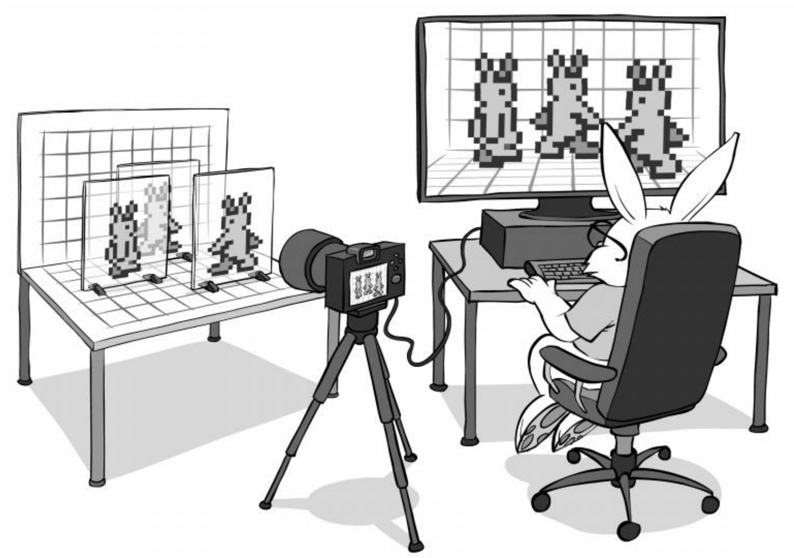
Viewing / Camera Transformation

- Think about how to take a photo
  - Find a good place and arrange people (**model** transformation)
  - Find a good "angle" to put the camera (**view/camera** transformation)
  - Shoot! (**projection** transformation)

MVP Transformation

Outline

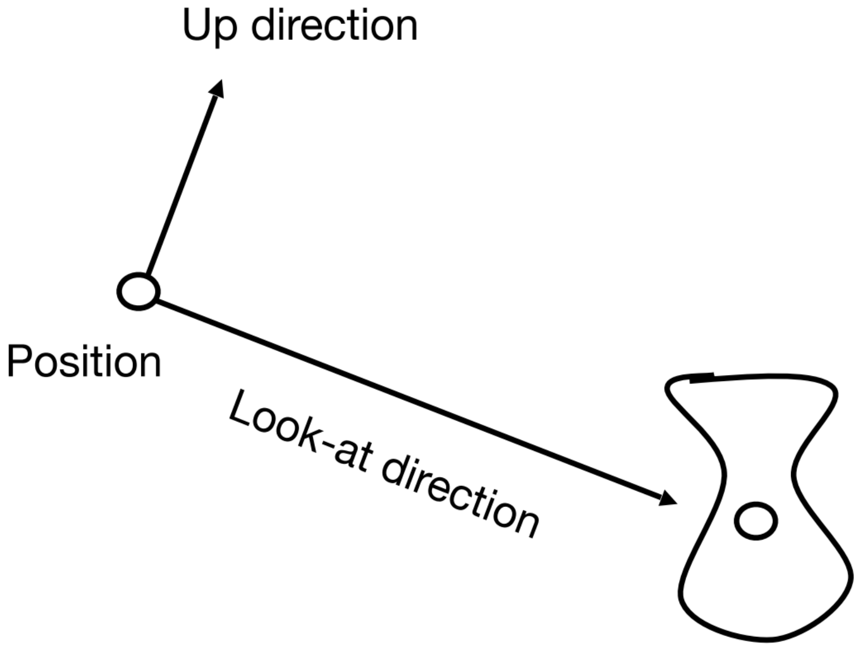
- Viewing Transformation
  - View / Camera transformation
  - Projection transformation
    - Orthographic projection
    - Perspective projection
- Viewport Transformation



View / Camera Transformation（视图变换）

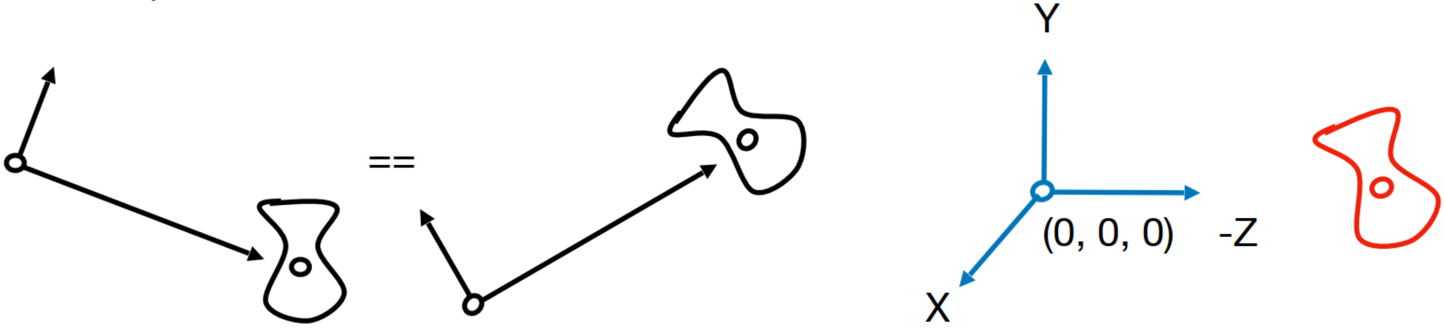
- How to perform view transformation?

- Define the camera first
  - Position $\vec{e}$
  - Look-at / gaze direction $\hat{g}$
  - Up direction $\hat{t}$
    (assuming perp. to look-at)



View / Camera Transformation

- Key observation
  - If the camera and all objects move together, the "photo" will be the same

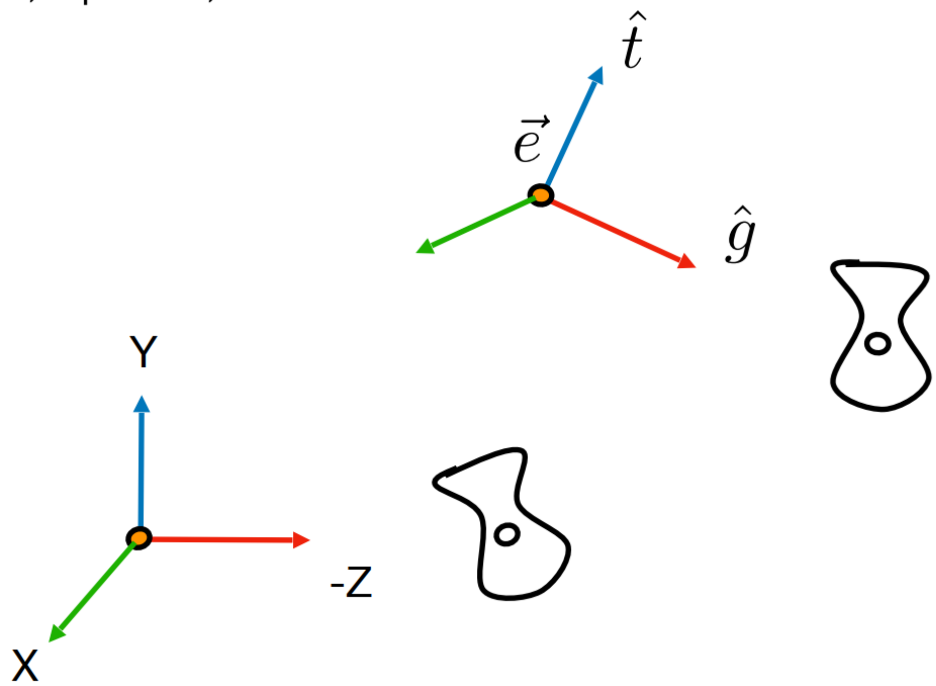


- How about that we always transform the camera to
  - The origin, up at Y, look at -Z
  - And transform the objects along with the camera

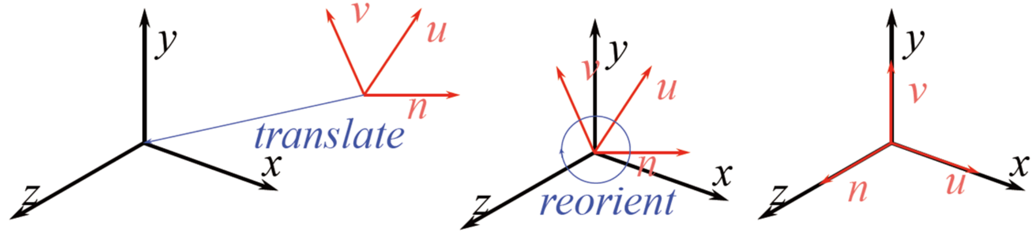
View / Camera Transformation

- Transform the camera by M_{view}
  - So it's located at the origin, up at Y, look at -Z

- M_{view} in math?
  - Translates e to origin
  - Rotates g to -Z
  - Rotates t to Y
  - Rotates (g x t) To X
  - Difficult to write!



View / Camera Transformation



- M_{view} in math?
  - Let's write $M_{view} = R_{view}T_{view}$
  - Translate e to origin

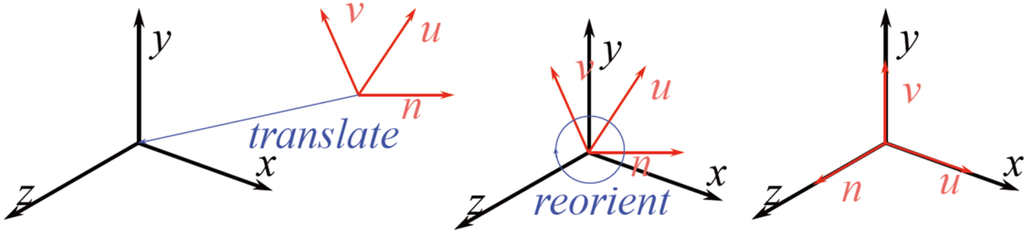
$$\begin{bmatrix} x_{\hat{g}\times\hat{t}} & x_t & x_{-g} & 0 \\ y_{\hat{g}\times\hat{t}} & y_t & y_{-g} & 0 \\ z_{\hat{g}\times\hat{t}} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} \\ y_{\hat{g}\times\hat{t}} \\ z_{\hat{g}\times\hat{t}} \\ 0 \end{bmatrix}$$

$$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Rotate g to -Z, t to Y, (g x t) To X
  - Consider its inverse rotation: X to (g x t), Y to t, Z to -g

$$R_{view}^{-1} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & x_t & x_{-g} & 0 \\ y_{\hat{g}\times\hat{t}} & y_t & y_{-g} & 0 \\ z_{\hat{g}\times\hat{t}} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xRightarrow{\text{WHY?}} R_{view} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t & y_t & z_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

View / Camera Transformation



$$
\begin{aligned}
M_{view} &= R_{view} T_{view} \\
&= \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t & y_t & z_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & -(\vec{X}\cdot E) \\ x_t & y_t & z_t & -(\vec{Y}\cdot E) \\ x_{-g} & y_{-g} & z_{-g} & -(\vec{Z}\cdot E) \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

where

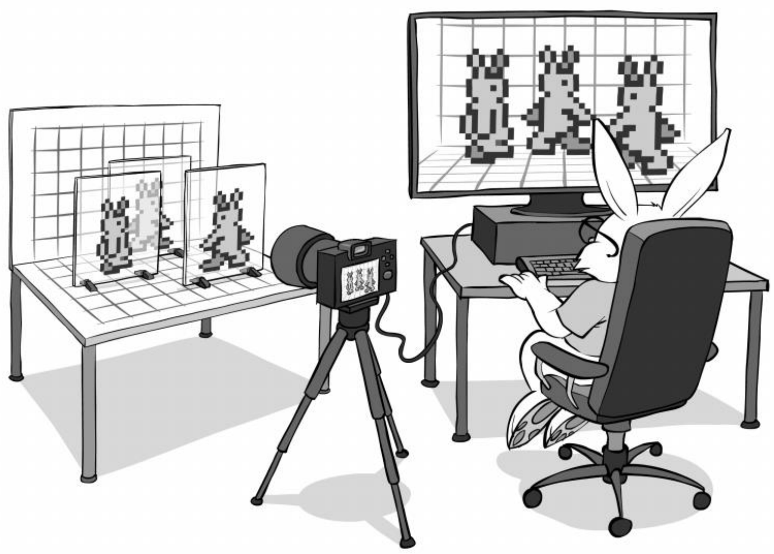
$$
\begin{aligned}
E &= (x_e, y_e, z_e) \\
\vec{X} &= (x_{\hat{g}\times\hat{t}}, y_{\hat{g}\times\hat{t}}, z_{\hat{g}\times\hat{t}}) \\
\vec{Y} &= (x_t, y_t, z_t) \\
\vec{Z} &= (x_{-g}, y_{-g}, z_{-g})
\end{aligned}
$$

View / Camera Transformation

- Summary
  - Transform objects together with the camera
  - Until camera's at the origin, up at Y, look at -Z

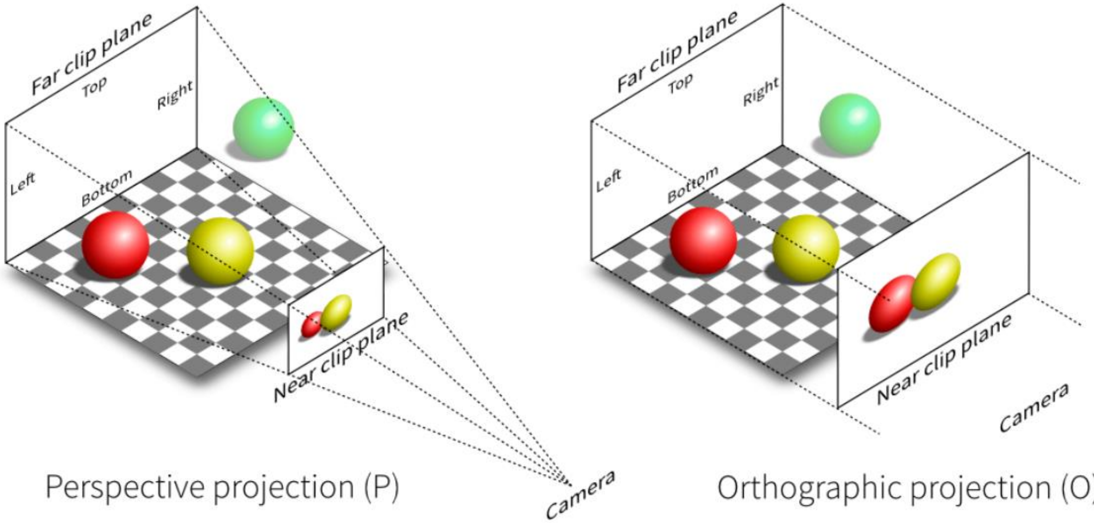
Outline

- Viewing Transformation
  - View / Camera transformation
  - Projection transformation
    - Orthographic projection
    - Perspective projection
- Viewport Transformation



Projection Transformation

- Projection in Computer Graphics
  - 3D to 2D
  - Orthographic projection
  - Perspective projection



Perspective projection (P) Orthographic projection (O)

View Frustum

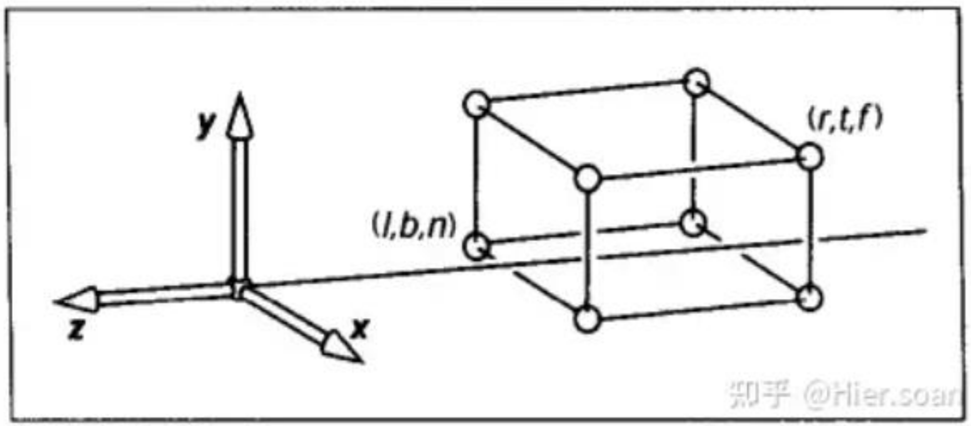
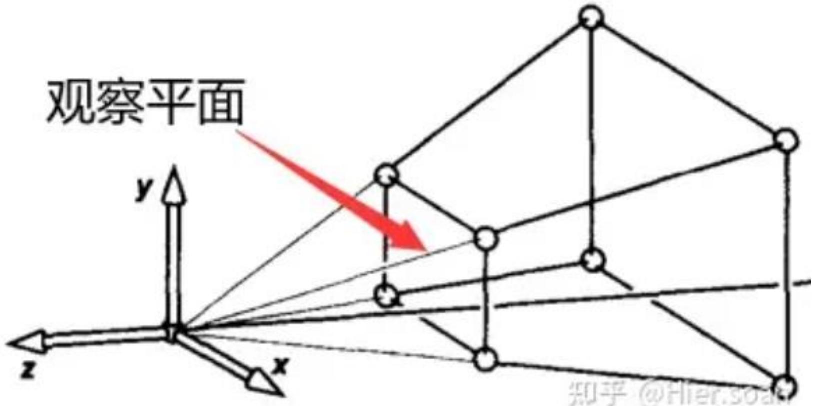


图2：观察空间中标定的正射投影视体



观察空间内标定的透视投影体

Standard View Frustum

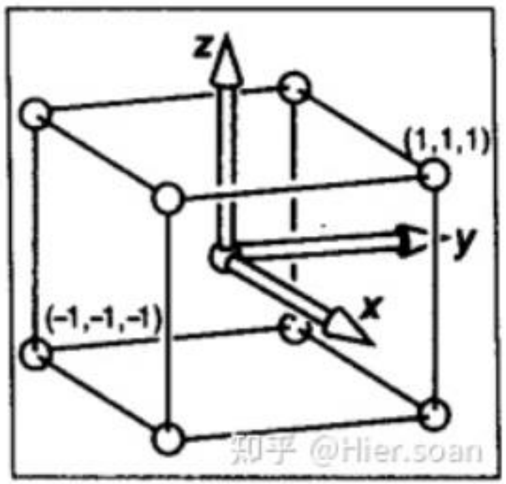
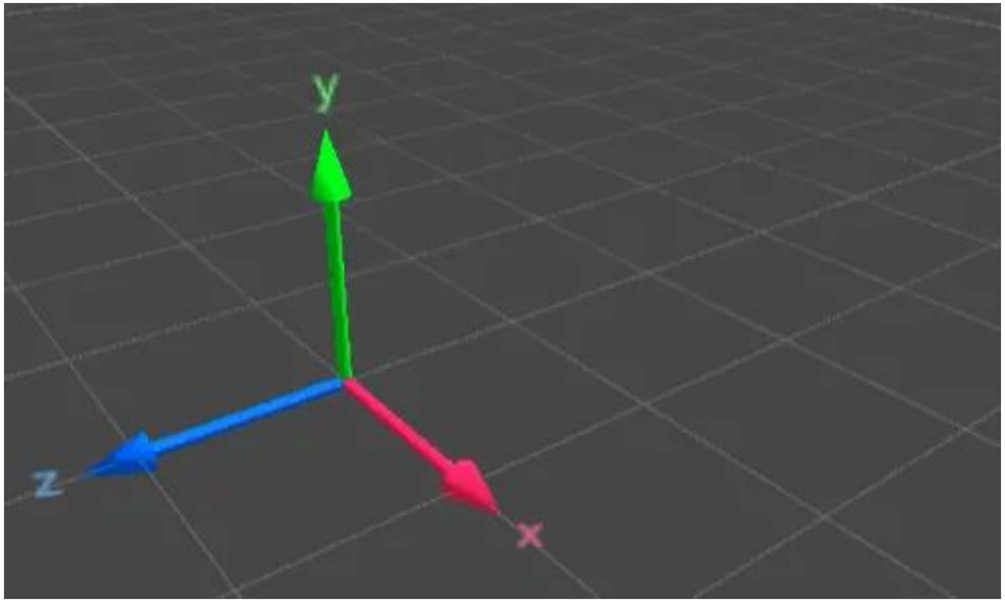
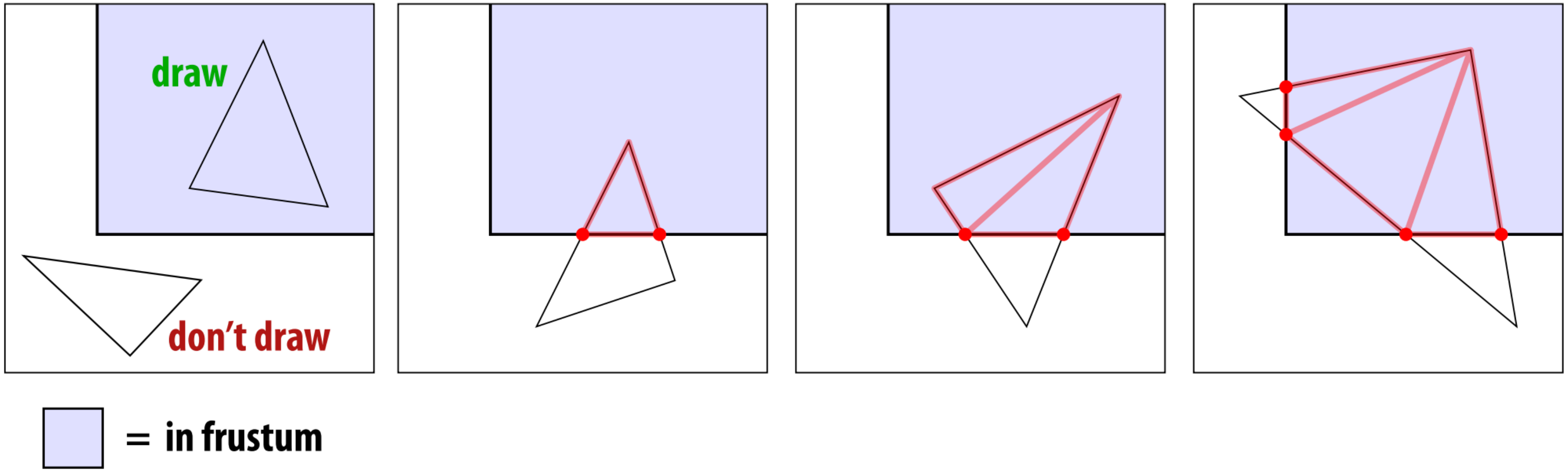


图1：标准视体



Clipping

- "Clipping" eliminates triangles not visible to the camera / in view frustum
  - Don't waste time rasterizing primitives (e.g., triangles) you can't see!
  - Discarding individual fragments is expensive ("fine granularity")
  - Makes more sense to toss out whole primitives ("coarse granularity")
  - Still need to deal with primitives that are partially clipped…



[计算机图形学补充2：齐次空间裁剪(Homogeneous Space Clipping) - 知乎 (zhihu.com)]

Clipping

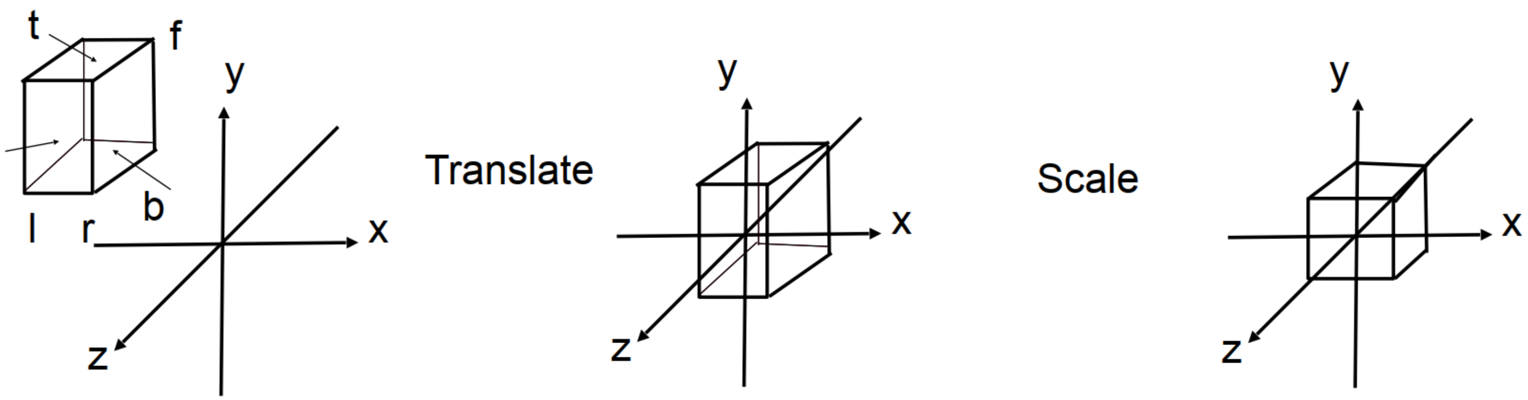
整个三角形删除

部分删除，重新组合

三角形顶点落在视椎体外不同处理方法的结果

Orthographic Projection

- In general
  - We want to map a cuboid [l, r] x [b, t] x [**f, n**] to the “canonical (正则、规范、标准)” cube $[-1, 1]^3$



  - Center cuboid by translating
  - Scale into “canonical” cube

Orthographic Projection

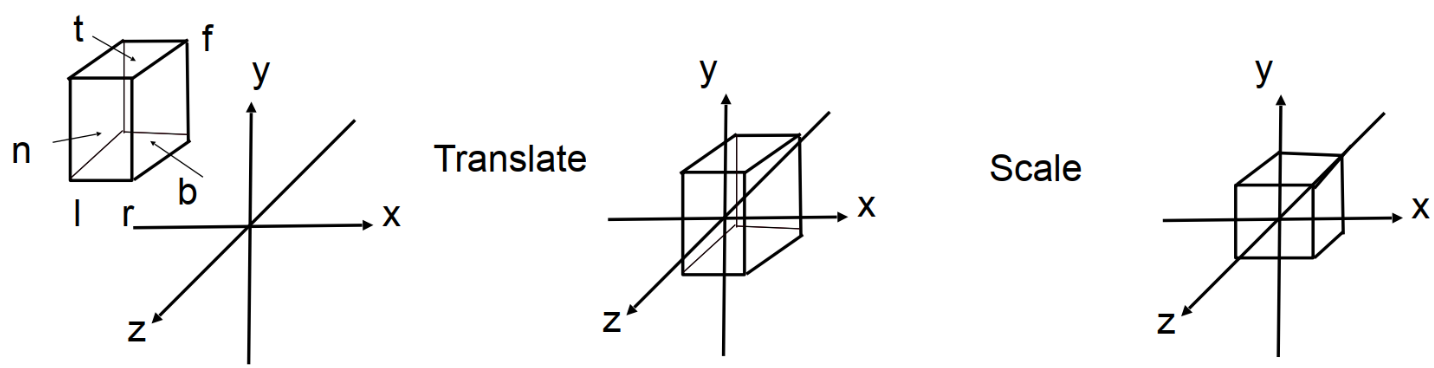
- Transformation matrix?
  - Translate (**center** to origin) first, then scale (length/width/height to **2**)

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

scale to size 2

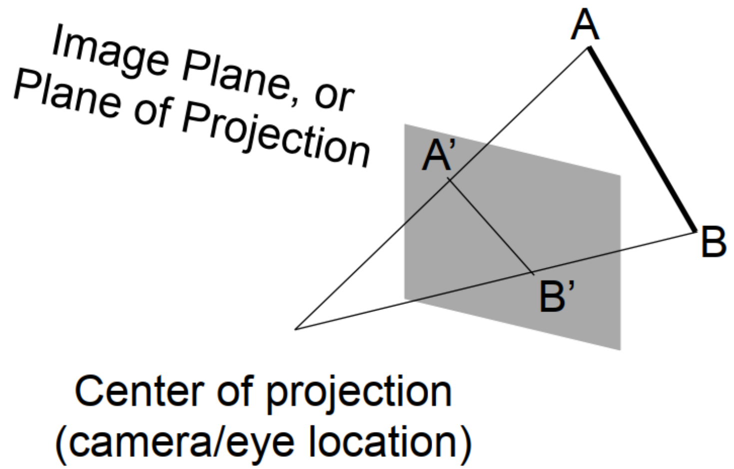
$$A = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & \frac{l+r}{l-r} \\ 0 & \frac{2}{t-b} & 0 & \frac{b+t}{b-t} \\ 0 & 0 & \frac{2}{n-f} & \frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

translate to origin



Perspective Projection

- Most common in Computer Graphics, art, visual system
- Further objects are smaller
- Parallel lines not parallel; converge to single point



Perspective Projection

- How to do perspective projection
  - First "squish" the frustum into a cuboid (n -> n, f -> f) ($M_{persp->ortho}$)
  - Do orthographic projection (M_{ortho}, already known!)

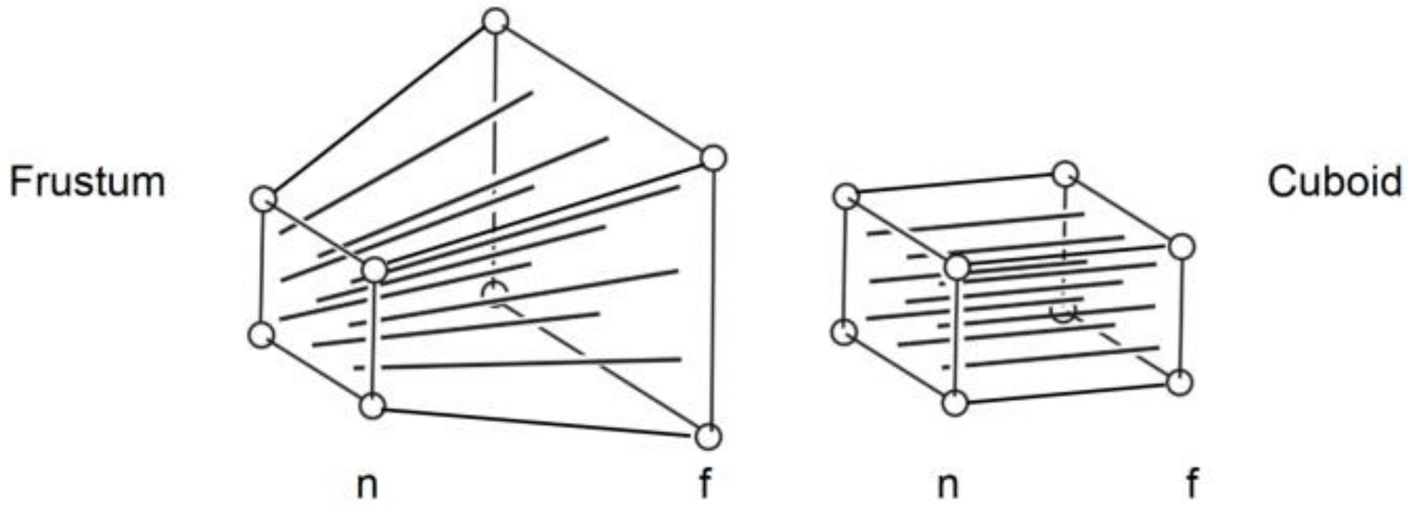
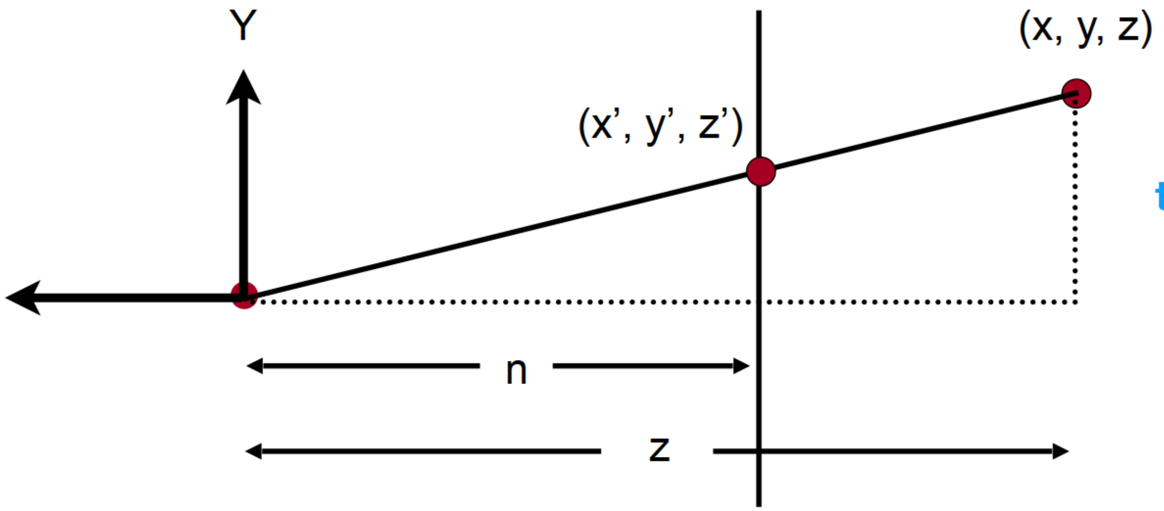


Fig. 7.13 from *Fundamentals of Computer Graphics, 4th Edition*

Perspective Projection

- In order to find a transformation
  - Recall the key idea: Find the relationship between transformed points (x', y', z') and the original points (x, y, z)



similar triangle

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

$$\frac{y'}{n} = \frac{y}{z} \rightarrow y' = \frac{n}{z} y$$

Perspective Projection

- In order to find a transformation
  - Find the relationship between transformed points (x', y', z') and the original points (x, y, z)

$$y' = \frac{n}{z} y \qquad x' = \frac{n}{z} x \text{ (similar to y')}$$

- This is clearly a non-linear transformation
- BUT: we can split it a linear transformation followed by a division

Perspective Projection

- Recall: property of homogeneous coordinates
  - (x, y, z, 1), (kx, ky, kz, k != 0), **(xz, yz, z^2, z != 0)** all represent the same point (x, y, z) in 3D
  - e.g. (1, 0, 0, 1) and (2, 0, 0, 2) both represent (1, 0, 0)

- Simple, but useful

- In homogeneous coordinates,

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} nx/z \\ ny/z \\ \text{unknown} \\ 1 \end{pmatrix} \overset{\text{mult. by z}}{==} \begin{pmatrix} nx \\ ny \\ \text{still unknown} \\ z \end{pmatrix}$$

Perspective Projection

- So the "squish" (persp to ortho) projection does this

$$M_{persp\to ortho}^{(4\times 4)}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}nx\\ny\\\text{unknown}\\z\end{pmatrix}$$

- Already good enough to figure out part of $M_{persp\text{->}ortho}$

$$M_{persp\to ortho}=\begin{pmatrix}n&0&0&0\\0&n&0&0\\?&?&?&?\\0&0&1&0\end{pmatrix}$$

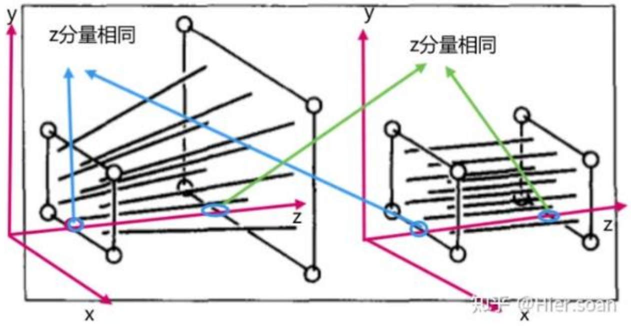
WHY?

Perspective Projection

- How to figure out the third row of $M_{persp\text{->}ortho}$
  - Any information that we can use?

$$M_{persp\to ortho} = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- Observation: the third row is responsible for z'
  - Any point on the near plane will not change
  - Any point's z on the far plane will not change



Perspective Projection

- Any point on the near plane will not change

$$M_{persp\to ortho}^{(4\times4)}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}nx\\ny\\\text{unknown}\\z\end{pmatrix}$$

replace z with n

$$\begin{pmatrix}x\\y\\n\\1\end{pmatrix}\Rightarrow\begin{pmatrix}x\\y\\n\\1\end{pmatrix}==\begin{pmatrix}nx\\ny\\n^2\\n\end{pmatrix}$$

- So the third row must be of the form (0 0 A B)

$$\begin{pmatrix}0 & 0 & A & B\end{pmatrix}\begin{pmatrix}x\\y\\n\\1\end{pmatrix}=n^2$$

n^2 has nothing to do with x and y

Perspective Projection

- What do we have now?

$$\begin{pmatrix} 0 & 0 & A & B \end{pmatrix} \begin{pmatrix} x \\ y \\ n \\ 1 \end{pmatrix} = n^2 \quad \Rightarrow \quad An + B = n^2$$

- Any point's z on the far plane will not change

$$\begin{pmatrix} 0 \\ 0 \\ f \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 \\ 0 \\ f \\ 1 \end{pmatrix} == \begin{pmatrix} 0 \\ 0 \\ f^2 \\ f \end{pmatrix} \quad \Rightarrow \quad Af + B = f^2$$

Perspective Projection

- Solve for A and B

$$An + B = n^2 \qquad\qquad A = n + f$$
$$Af + B = f^2 \qquad\qquad B = -nf$$

- Finally, every entry in $M_{persp\text{->}ortho}$ is known!

- What's next?
  - Do orthographic projection (M_{ortho}) to finish
  - $M_{persp} = M_{ortho} M_{persp \to ortho}$

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

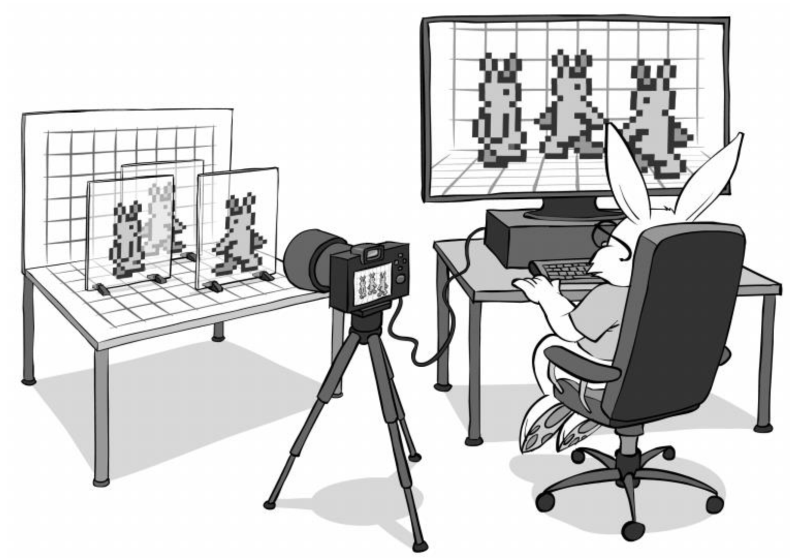
l = left $\quad b$ = bottom $\quad n$ = near
r = right $\quad t$ = top $\quad f$ = far

Perspective Projection

- Expressible with 4x4 homogeneous matrix
- Perspective division is nonlinear and results in non-uniform shortening
  - Objects far from the COP are projected to a much smaller size
- Perspective transformations are linear preserving, but not affine transformations
- Perspective projection is irreversible
  - many 3D points can be mapped to same (x, y, d) on the projection plane
  - no way to retrieve the unique z values

Outline

- Viewing Transformation
  - View / Camera transformation
  - Projection transformation
    - Orthographic projection
    - Perspective projection



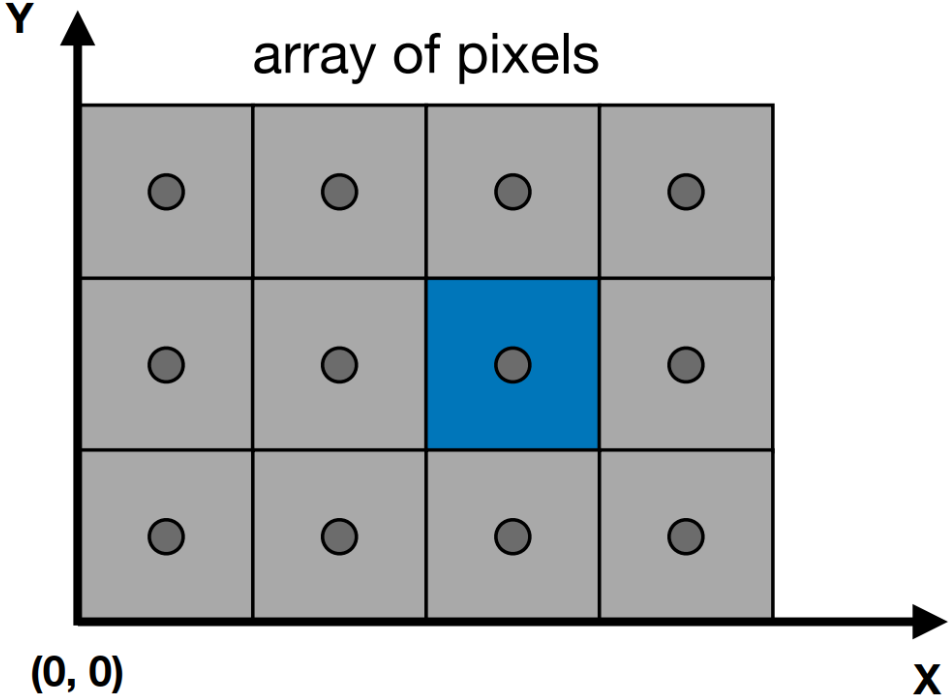
- Window to Viewport Transformation

What's after MVP?

- **M**odel transformation (placing objects)
- **V**iew transformation (placing camera)
- **P**rojection transformation
  - Orthographic projection (cuboid to "canonical" cube $[-1, 1]^3$)
  - Perspective projection (frustum to "canonical" cube)
- Canonical cube to ?

Canonical Cube to Screen

- Defining the screen space



Pixels' indices are in the form of (x, y), where both x and y are integers

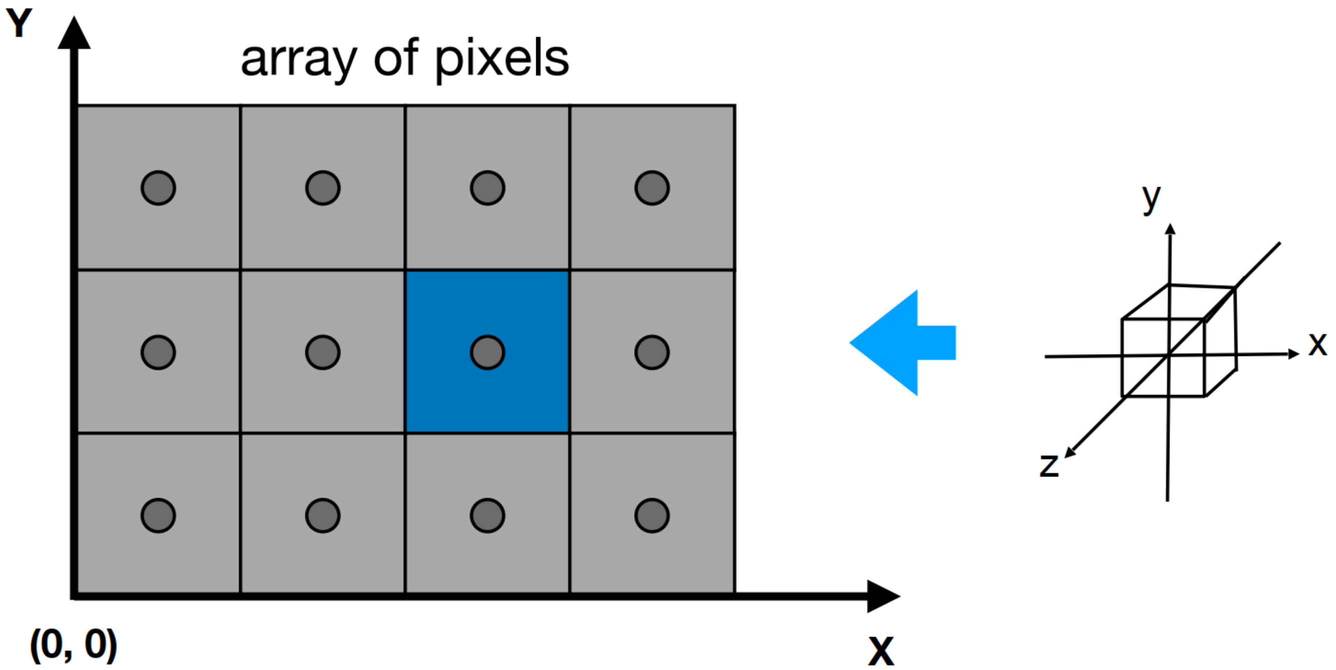
Pixels' indices are from (0, 0) to (width - 1, height - 1)

Pixel (x, y) is centered at (x + 0.5, y + 0.5)

The screen covers range (0, 0) to (width, height)

Canonical Cube to Screen

- Irrelevant to z
- Transform in xy plane: $[-1, 1]^2$ to [0, width] x [0, height]



Window to Viewport Transformation

- Irrelevant to z

- **Projection will take points to [-1,1] x [-1,1] on the z = 1 plane; transform into a W x H pixel image**

“normalized device coordinates”

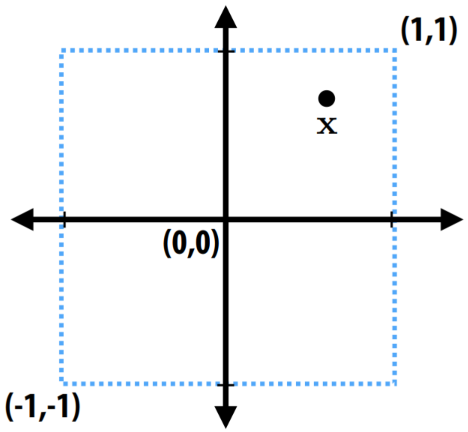
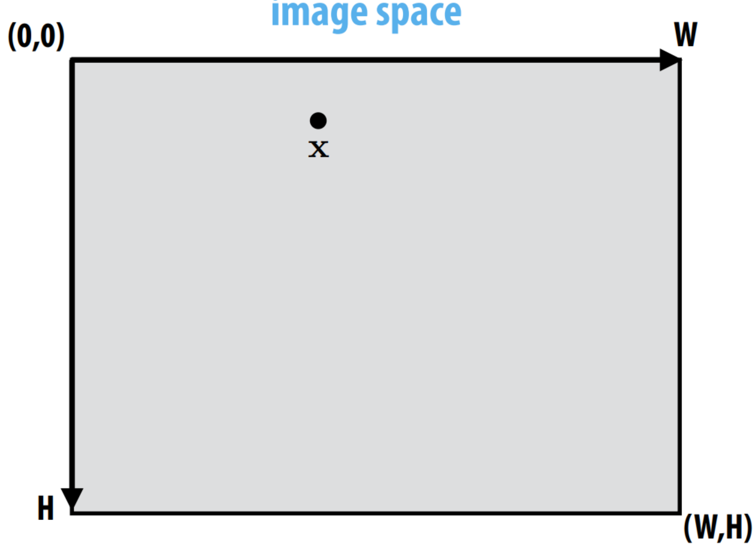


image space



Step 1: reflect about x-axis
Step 2: translate by (1,1)
Step 3: scale by (W/2,H/2)

CMU 15-462/662

Transformations from 3D Space to 2D Screen Space

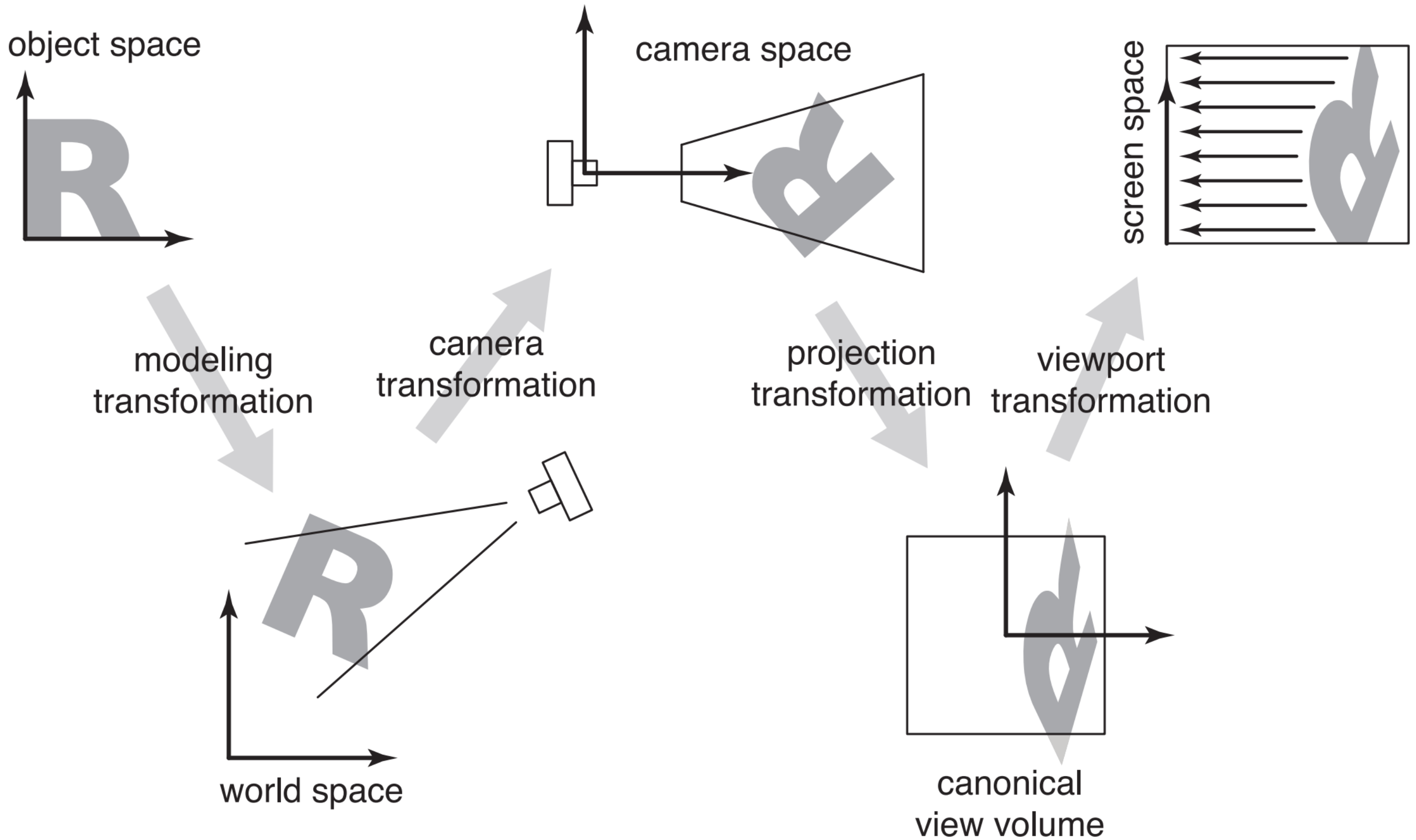


Figure 7.2. The sequence of spaces and transformations that gets objects from their original coordinates into screen space.

Homework 1