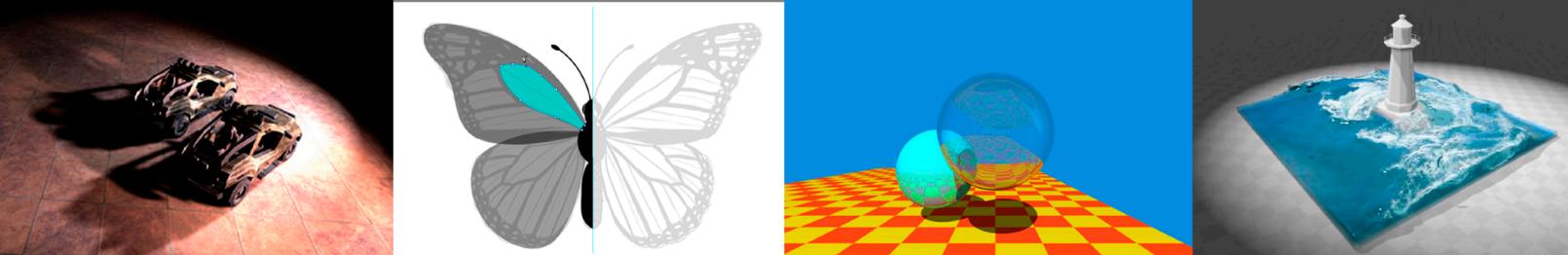


Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 4: Transformation Cont.



Announcement

- Homework 0 will be released TODAY
- This lecture will be difficult :)

Last Lecture

- Transformation
 - Why study transformation
 - 2D transformations: rotation, scale, shear
 - Homogeneous coordinates
 - Composite transform
 - 3D transformations

Today

- 3D transformations
- Viewing (观测) transformation
 - View (视图) / Camera transformation
 - Projection (投影) transformation
 - Orthographic (正交) projection
 - Perspective (透视) projection

3D Transformations

Use homogeneous coordinates again:

- 3D point = $(x, y, z, 1)^T$
- 3D vector = $(x, y, z, 0)^T$

In general, (x, y, z, w) ($w \neq 0$) is the 3D point:

$$(x/w, y/w, z/w)$$

3D Transformations

Use 4×4 matrices for affine transformations

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

线性变换
平移.

What's the order?

Linear Transform first or Translation first?

3D Transformations

Scale

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

Translation

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

3D Transformations

Rotation around x-, y-, or z-axis

x (y 又乘 z)

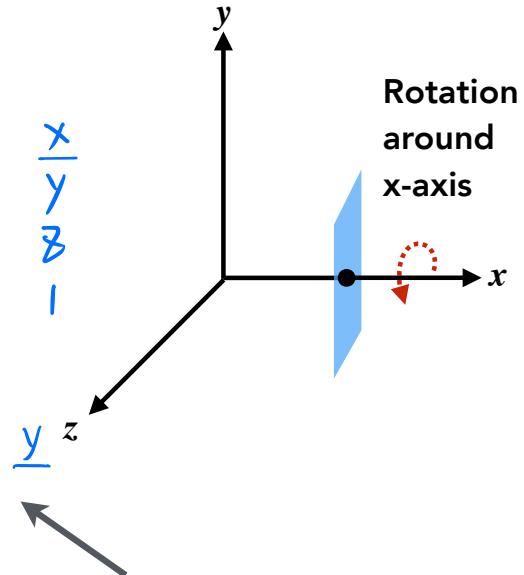
y (z 又乘 x)
相反

z (x 又乘 y)

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

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

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



Anything strange about R_y ?

z

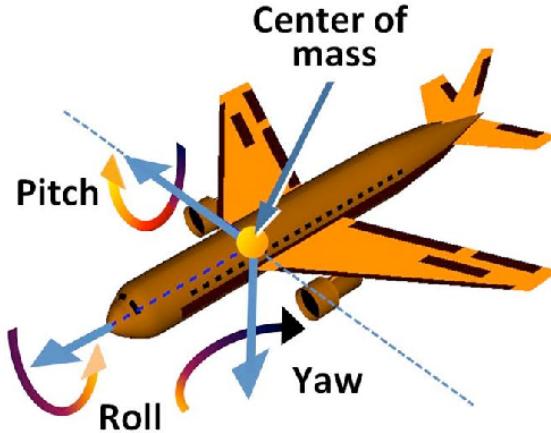
3D Rotations

任意三维旋转

Compose any 3D rotation from $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$?

$$\mathbf{R}_{xyz}(\alpha, \beta, \gamma) = \mathbf{R}_x(\alpha) \mathbf{R}_y(\beta) \mathbf{R}_z(\gamma)$$

- So-called *Euler angles*
- Often used in flight simulators: roll, pitch, yaw



Rodrigues' Rotation Formula

绕轴的旋转角度

Rotation by angle α around axis n

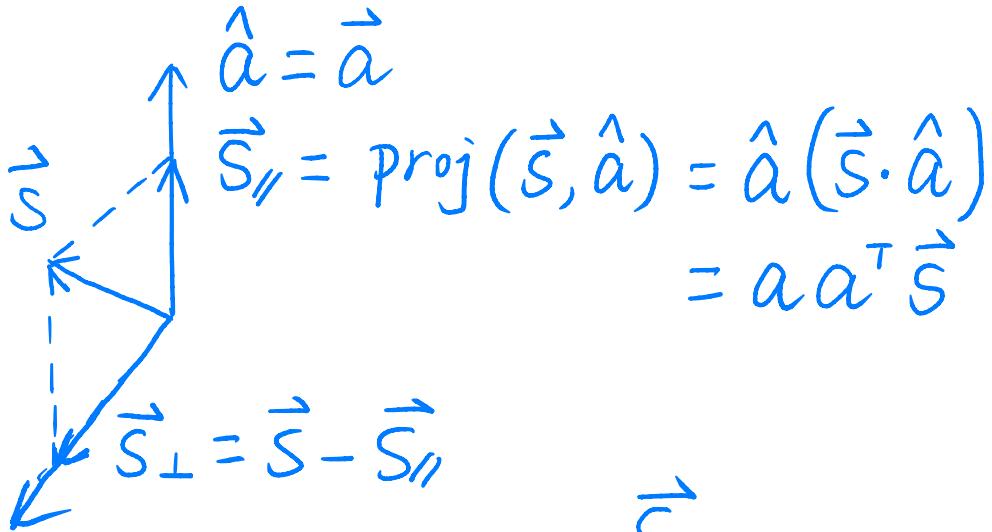
$$\mathbf{R}(\mathbf{n}, \alpha) = \cos(\alpha) \mathbf{I} + (1 - \cos(\alpha)) \mathbf{n}\mathbf{n}^T + \sin(\alpha) \underbrace{\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}}_{\mathbf{N}}$$

How to prove this magic formula?

Check out the supplementary material on the course website!

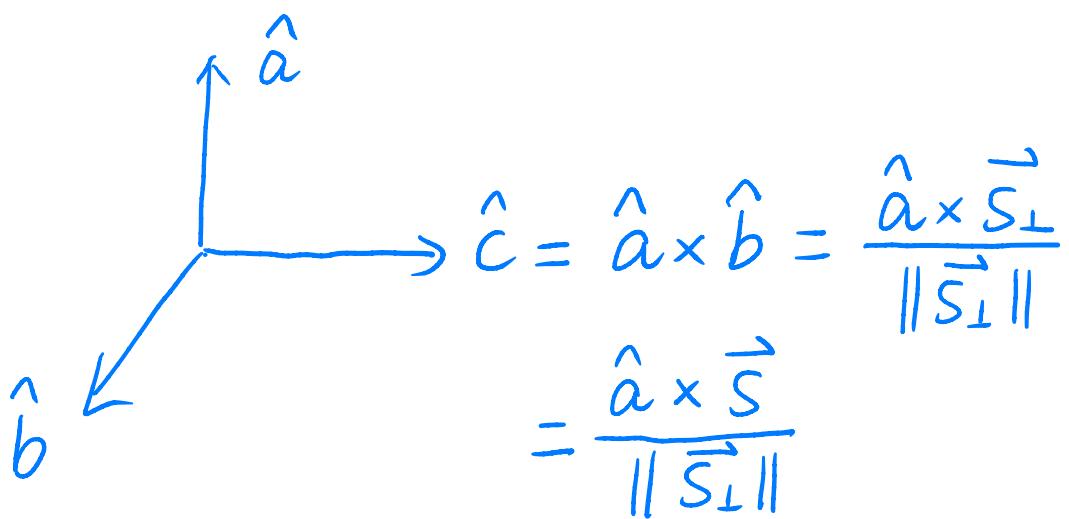
必四元数 旋转与旋转之间的插值 .

Step 1:



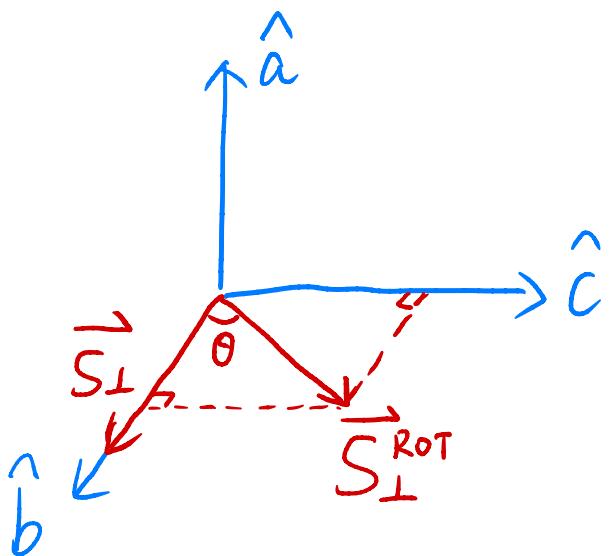
$$\hat{b} = \text{normalize}(\vec{s}_{\perp}) = \frac{\vec{s}_{\perp}}{\|\vec{s}_{\perp}\|}$$

Step 2:



$$= \frac{\hat{a} \times \vec{s}}{\|\vec{s}_{\perp}\|}$$

Step 3:



$$\begin{aligned}\vec{s}_{\perp}^{\text{ROT}} &= \|\vec{s}_{\perp}\| \cos \theta \hat{b} \\ &\quad + \|\vec{s}_{\perp}\| \sin \theta \hat{c}\end{aligned}$$

Today

- 3D transformations
- Viewing transformation
 - View / Camera transformation
 - Projection transformation
 - Orthographic projection
 - Perspective projection

View / Camera Transformation

- What is view 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** transformation)
 - Cheese! (**projection** transformation)

模型变换

视图变换

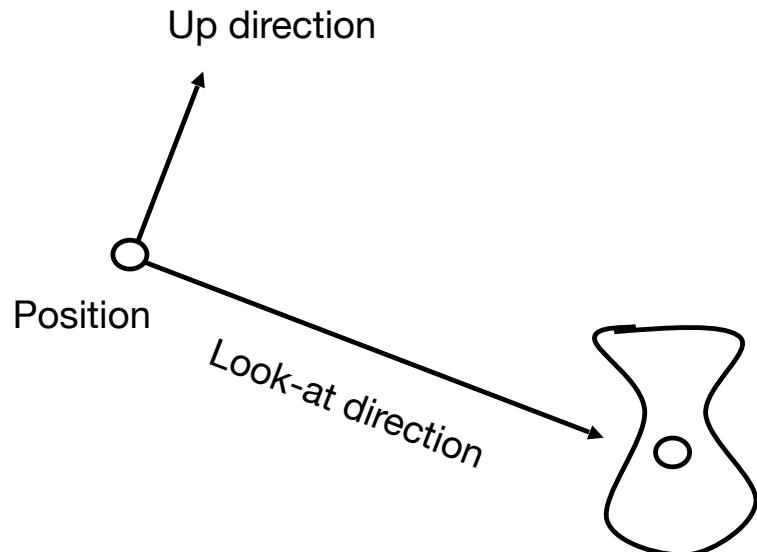
投影变换

“MVP”

View / Camera Transformation

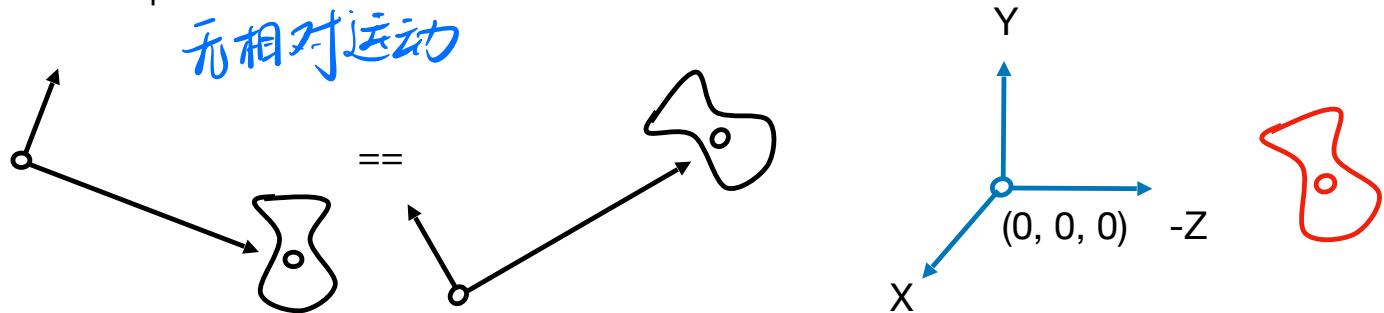
- How to perform view transformation?
- Define the camera first

位置 - Position \vec{e}
视线方向 - Look-at / gaze direction \hat{g}
向上方向 - 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
摄像机是在原点，往+Y方向看，向上方向为y.
 - The origin, up at Y, look at -Z
 - And transform the objects along with the camera
和相机一起变换物体

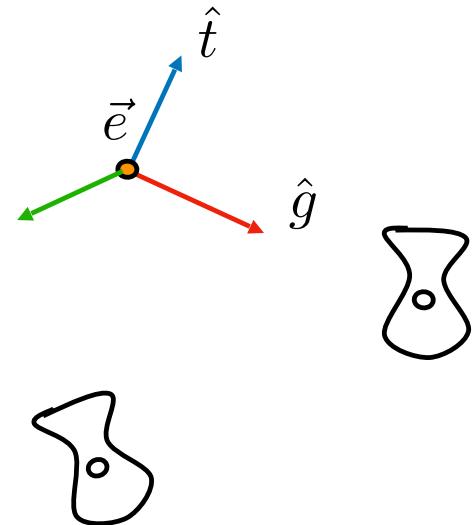
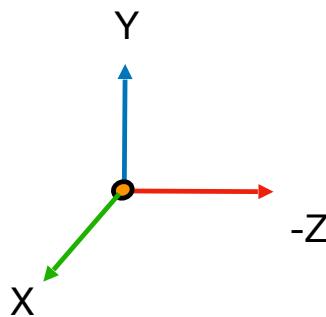
View / Camera Transformation

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

- M_{view} in math?

平行

- Translates e to origin
- Rotates g to -Z
- Rotates t to Y
- Rotates $(g \times t)$ To X
- Difficult to write!



视图变换

View / Camera Transformation

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

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

- Rotate g to -Z, t to Y, $(g \times t)$ To X
无尾逆旋转
 $(1,0,0,0)^T$ $(0,1,0,0)^T$ $(0,0,1,0)^T$
- Consider its **inverse** rotation: X to $(g \times t)$, Y to t, Z to -g

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



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

旋转变换(正矩阵) 置置=逆

View / Camera Transformation

- Summary

与相机一起转换对象

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

相机位于原点，向上为Y，看-Z方向

- Also known as ModelView Transformation

- But why do we need this?

- For projection transformation!

Today

- 3D transformations
- Viewing transformation
 - View / Camera transformation
 - **Projection transformation**
 - Orthographic projection
 - Perspective projection

Projection Transformation

- Projection in Computer Graphics

- 3D to 2D

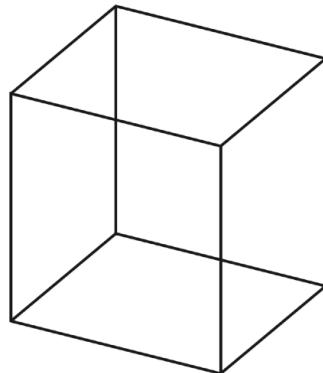
- Orthographic projection

正交投影：不会带来近大远小的观感

- Perspective projection

透视投影：“近大远小”

Orthographic
projection



Perspective
projection

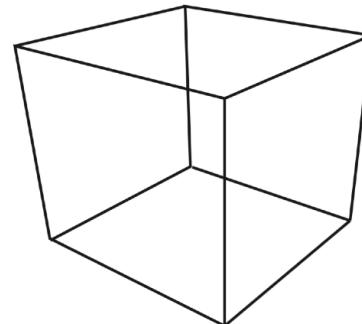
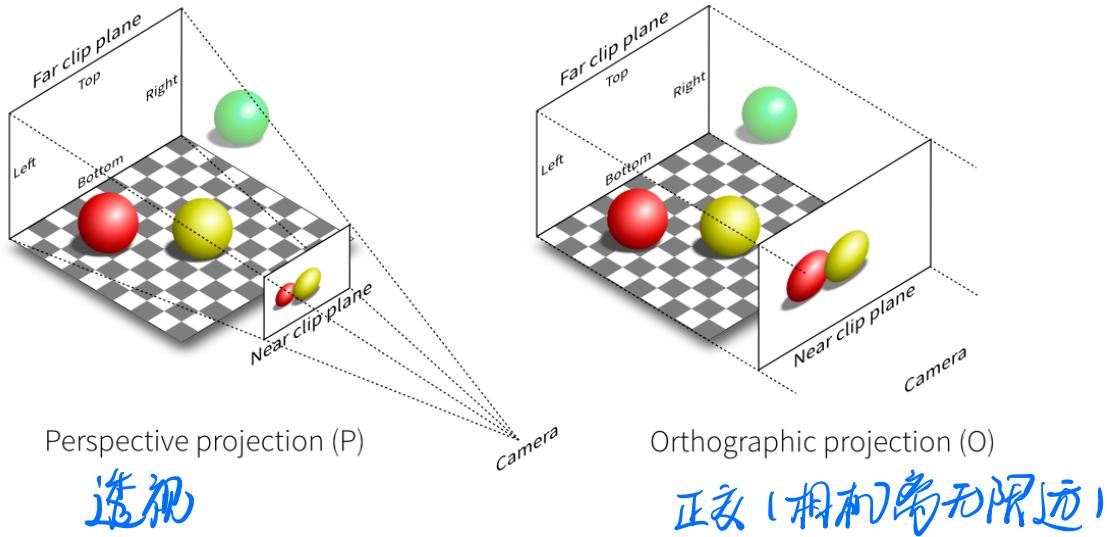


Fig. 7.1 from *Fundamentals of Computer Graphics, 4th Edition*

正交

Projection Transformation

- Perspective projection vs. orthographic projection



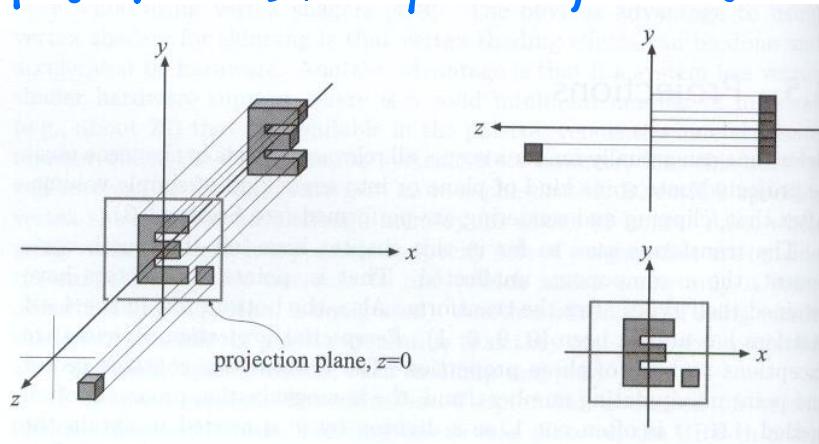
<https://stackoverflow.com/questions/36573283/from-perspective-picture-to-orthographic-picture>

正交

Orthographic Projection

- A simple way of understanding
 - Camera located at origin, looking at -Z, up at Y (looks familiar?)
 - Drop Z coordinate
 - Translate and scale the resulting rectangle to $[-1, 1]^2$

将字母E投影到平面并缩放到 [-1, 1]²



正交

Orthographic Projection

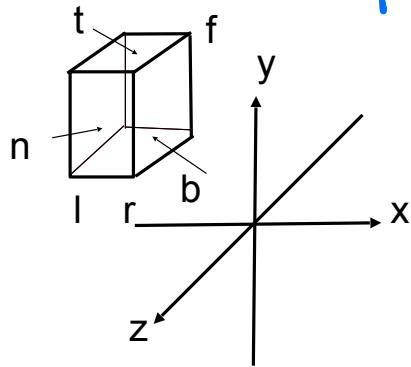
$n > f$

沿-z方向 { 近 z值大
远 z值小

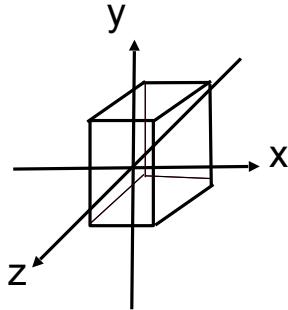
- In general

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

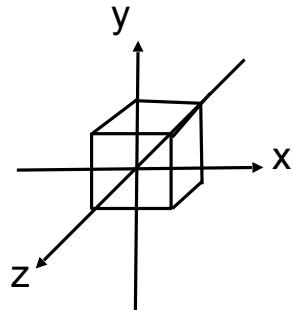
将任意一个长方体变成标准立方体



Translate



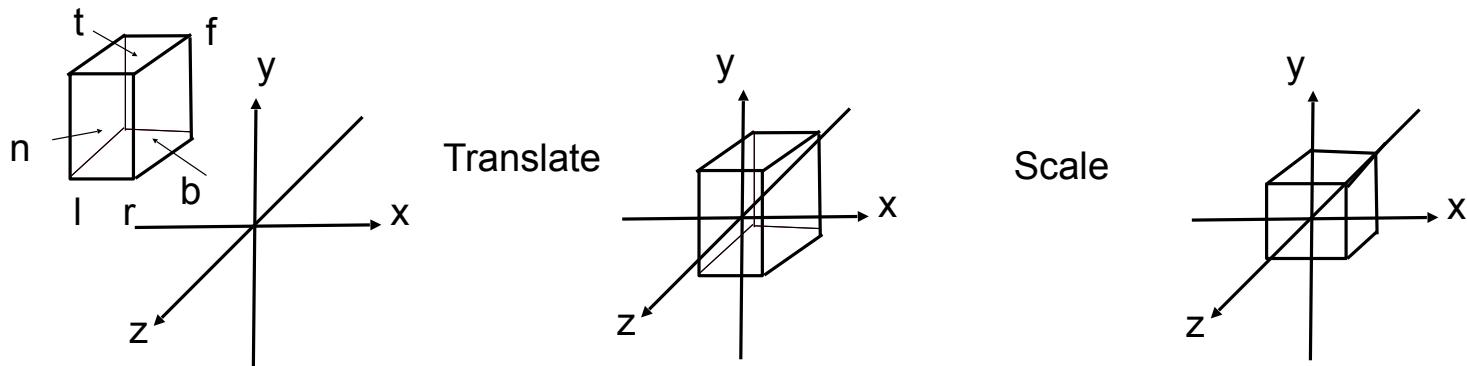
Scale



正

Orthographic Projection

- Slightly different orders (to the “simple way”)
 - Center cuboid by translating 平移到中心.
 - Scale into “canonical” cube 缩放到“规范”立方体

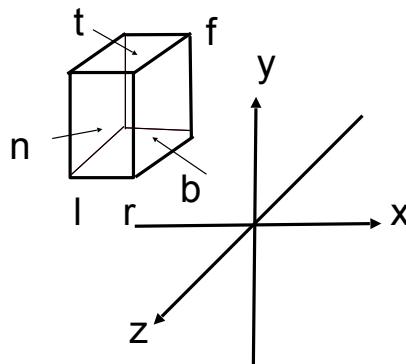


正交

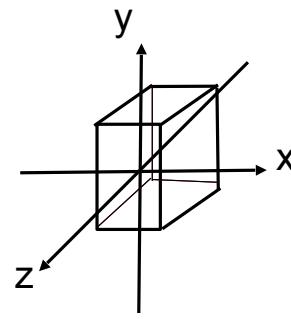
Orthographic Projection

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

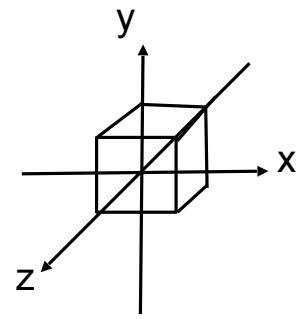
$$M_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Translate

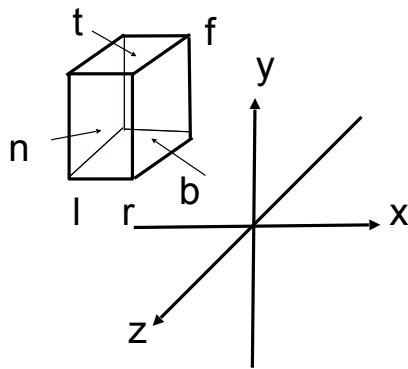


Scale

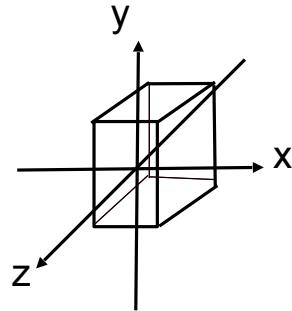


Orthographic Projection

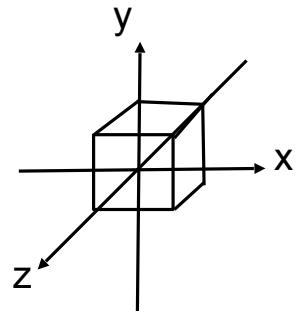
- Caveat
 - Looking at / along -Z is making near and far not intuitive ($n > f$)
 - FYI: that's why OpenGL (a Graphics API) uses left hand coords.



Translate



Scale

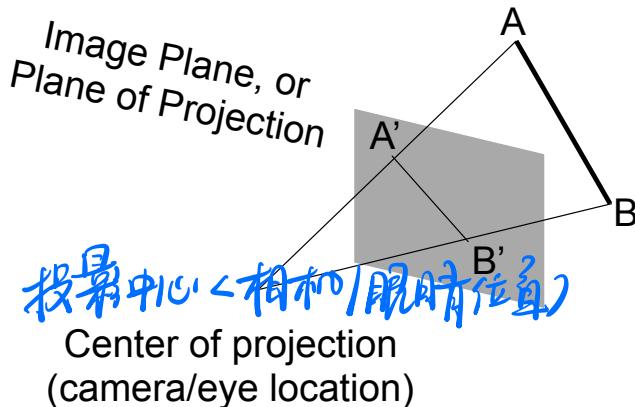


透視投影

Perspective Projection

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

圖像平面(透影平面)



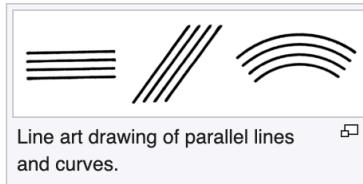
Perspective Projection

平行线在同一平面内平行。

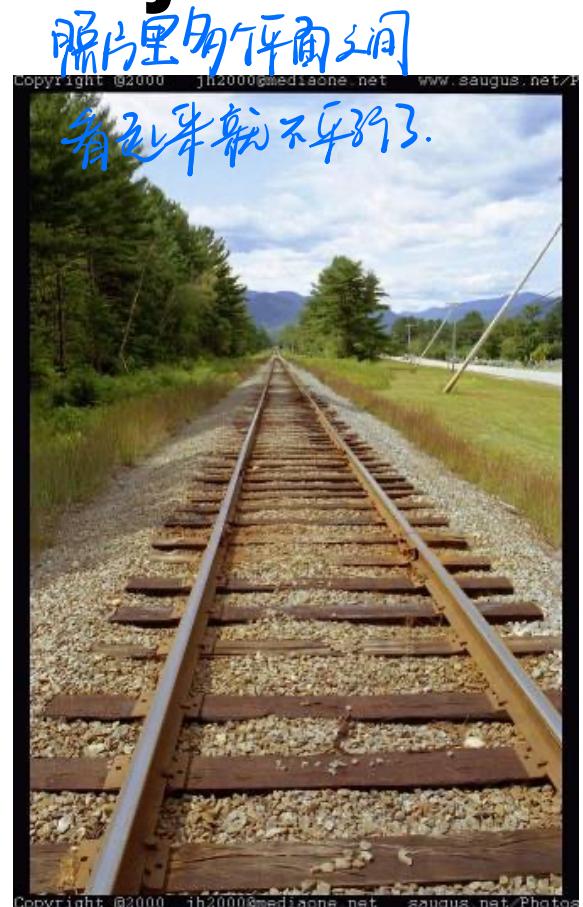
- Euclid was wrong??!!

In **geometry**, **parallel** lines are **lines in a plane** which do not meet; that is, two lines in a plane that do not **intersect** or **touch** each other at any point are said to be parallel. By extension, a line and a plane, or two planes, in **three-dimensional Euclidean space** that do not share a point are said to be parallel. However, two lines in three-dimensional space which do not meet must be in a common plane to be considered parallel; otherwise they are called **skew lines**. Parallel planes are planes in the same three-dimensional space that never meet.

Parallel lines are the subject of **Euclid's parallel postulate**.^[1] Parallelism is primarily a property of **affine geometries** and **Euclidean geometry** is a special instance of this type of geometry. In some other geometries, such as **hyperbolic geometry**, lines can have analogous properties that are referred to as parallelism.



Line art drawing of parallel lines and curves.



[https://en.wikipedia.org/wiki/Parallel_\(geometry\)](https://en.wikipedia.org/wiki/Parallel_(geometry))

Perspective Projection

- Before we move on
- Recall: property of homogeneous coordinates
 - $(x, y, z, 1)$, $(kx, ky, kz, k \neq 0)$, **$(xz, yz, z^2, z \neq 0)$** 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

透視投影

Perspective Projection

远平面中心点挤压后仍是中心点

(f)

- How to do perspective projection

① 将裁剪体“挤压”成长方体 1. 近平面不变 2. 远平面不变

- First “squish” the frustum into a cuboid ($n \rightarrow n$, $f \rightarrow f$) ($M_{\text{persp} \rightarrow \text{ortho}}$)

② 做正交投影

- Do orthographic projection (M_{ortho} , already known!)

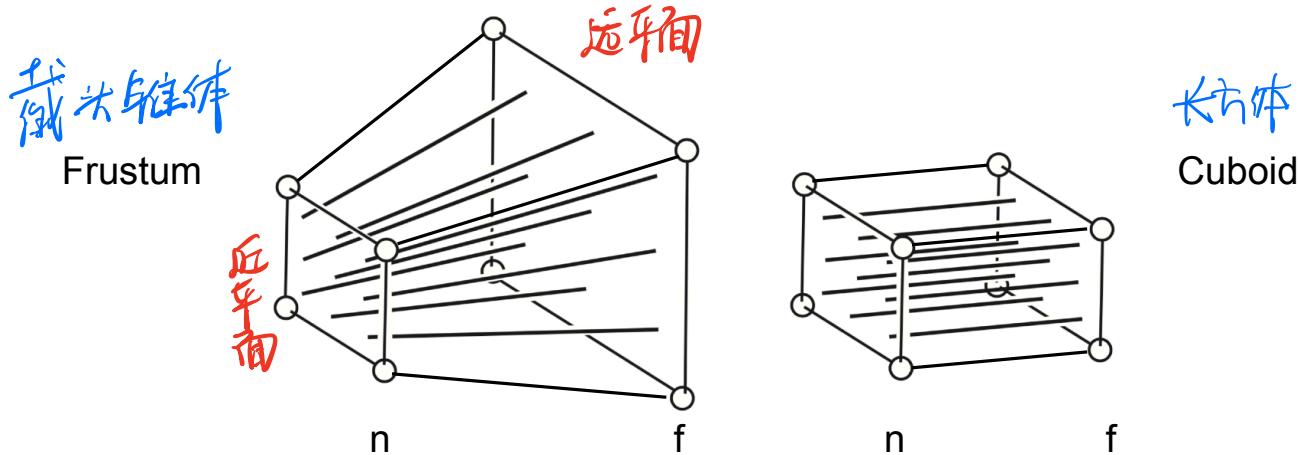
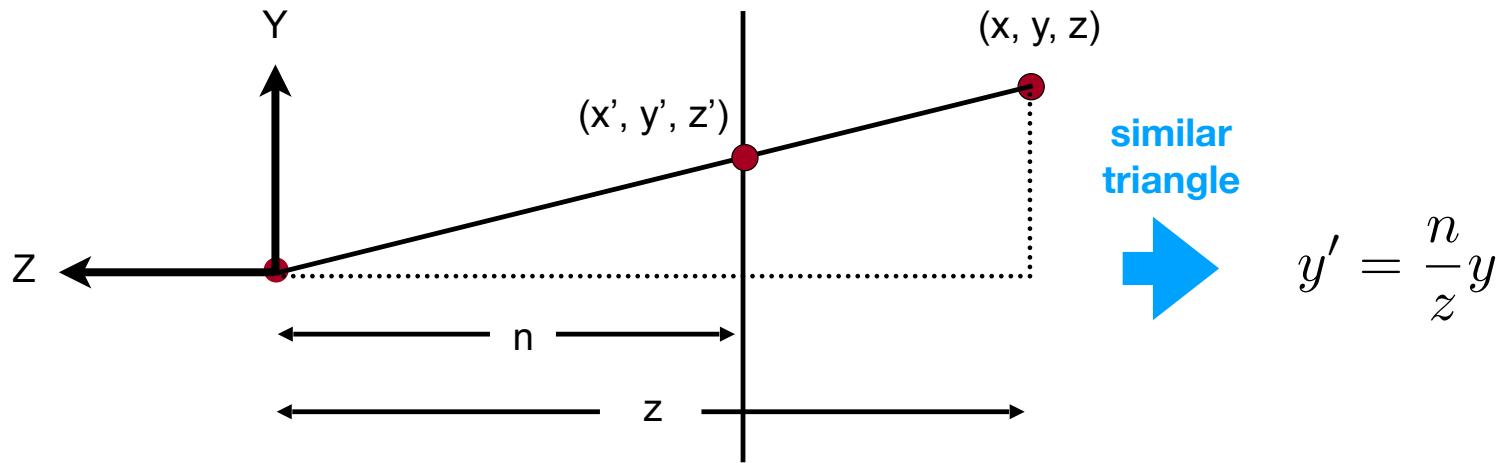


Fig. 7.13 from *Fundamentals of Computer Graphics, 4th Edition*

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)



Perspective Projection

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

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

- In homogeneous coordinates,

$$\left(\begin{array}{c|c} \cdot & \cdot \\ \hline & 1 \end{array} \right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} nx/z \\ ny/z \\ \text{unknown} \\ 1 \end{pmatrix} \begin{matrix} \text{mult.} \\ \text{by } z \end{matrix} \begin{pmatrix} nx \\ ny \\ \text{still unknown} \\ z \end{pmatrix}$$

Perspective Projection

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

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

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

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

WHY?

Perspective Projection

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

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

- Observation: the third row is responsible for z'

- Any point on the near plane will not change

$$\begin{pmatrix} x \\ y \\ n \end{pmatrix} \xrightarrow{\text{不变}} \begin{pmatrix} x \\ y \\ n \end{pmatrix} \text{ 近平面}$$

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

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \xrightarrow{\text{不变}} \begin{pmatrix} x' \\ y' \\ z \end{pmatrix} \text{ 远平面}$$

Perspective Projection

- Any point on the near plane will not change

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

replace
z with n

近平面

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

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

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

n² has nothing
to do with x and y

Perspective Projection

- What do we have now?

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

- Any point's z on the far plane will not change
远平面的中心点 $(0, 0, f)$

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

Perspective Projection

- Solve for A and B

$$An + B = n^2$$

$$Af + B = f^2$$

$$(x, y, \frac{ntf}{2})$$

这T点?

$$\frac{(n+f)^2 - nf}{\frac{ntf}{2}} = \frac{n^2 + f^2}{n+f}$$

新坐标
更近

$$A = n + f$$

$$B = -nf$$

$$\frac{n^2 + f^2}{n+f} - \frac{ntf}{2}$$

$$= \frac{2n^2 + 2f^2 - cn + f^2}{2}$$

$$= \frac{2n + f}{2}$$

$$= \frac{n^2 + f^2 - 2nf}{2n + f}$$

$$= \frac{(n-f)^2}{2n+f}$$

$$> 0$$

<<挤压>>

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

- What's next?

正交投影

- Do orthographic projection (M_{ortho}) to finish

$$M_{persp} = M_{ortho} M_{persp \rightarrow ortho}$$

透视投影

正

<<挤>>

Thank you!