part1

2021年7月8日 12:43

Games 101

§ Transformation.

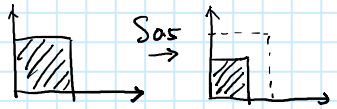
1. Why transformation?

modeling { ① rotation ② translation ③ scaling

Viewing — ① 3D to 2D projection.

2. 2D transformation

① Scale



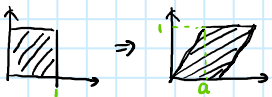
Scale Matrix

$$\begin{cases} x' = sx \\ y' = sy \end{cases} \Leftrightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

② Reflection Matrix

$$\begin{cases} x' = -x \\ y' = y \end{cases} \Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

③ Shear Matrix 切变

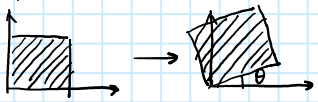


Horizontal shift exists
vertical shift is always 0.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

④ Rotate (about the origin)

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$



Linear Transforms = Matrix

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

$$x' = Mx$$

3. homogenous coordinates 齐次坐标

① why homogenous coordinates?

shift = $\begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases}$ 但无法写成矩阵形式.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$ 平移量.

② Solution:

Add a third coordinate (w-coordinate)

2D point = $(x, y, 1)^T$

2D vector = $(x, y, 0)^T$ 向量有平移不变性.

Matrix representation of translations

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

Why w-coordinate of point and vec differ?

vec + vec = vec
point − point = vec.
point + vec = point
} so w of vec is 0

In homogenous coordinates

$\begin{pmatrix} x \\ y \\ w \end{pmatrix}$ is 2D point $\begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix}$ 扩充定义

Thus "point + point" is given a meaning.

Affine Transformation 仿射变换

Thus point + point is given a meaning.

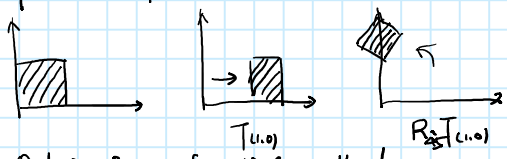
Affine Transformation 仿射变换

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

Inverse Transform

M^{-1}

Composite transformation



Ordering of transformations matter!

$R_{45}T_{(1,0)} \neq T_{(1,0)}R_{45}$

Composing Transforms

sequence of affine transforms $A_1, A_2, A_3 \cdots$

Compose by matrix multiplication

⇒ Very important for performance!

$A_n(\cdots A_2(A_1(x))) = A_n \cdots A_2 \cdot A_1 \begin{pmatrix} x \\ y \end{pmatrix}$

Pre-multiply n matrices to obtain a single matrix

3x3矩阵相乘结果恒为3x3. 意味着任何transform都可用3x3 matrix表示

Decomposing Complex transforms

rotate around point c:

① move center to origin
② rotate
③ move center back.

} decomposition.

$T(c)R(\alpha)T(-c)$

4. 3D Transformation.

3D point $= (x,y,z,1)^T \Rightarrow (x,y,z,1)^T$

3D vector $= (x,y,z,1)^T \Rightarrow (x,y,z,0)^T$

Use 4x4 matrices

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

5. Linear first or Translation first?

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

Thus, linear transformation comes first

补充:

$R_{-\theta} = R_\theta^T$

an important conclusion

§ Transformation Cont.

1. 3D transformation. continue.

Rotation around x-, y- or z- axis

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

$$R_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ 注意 R_y 中 $\pm\sin\alpha$ 是 R_x, R_z 不同

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

Compose 3D rotation

$R_{xyz}(\alpha,\beta,\gamma) = R_x(\alpha)R_y(\beta)R_z(\gamma)$

Rodrigues' Rotation Formula

Rotation by angle α around axis n

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

Rotation by angle α around axis n

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

注意 n 过 origin (by default)

2. Viewing Transformation
   - View / Camera tranformation
   - Projection transformation.
     - Othographic projection
     - Perspective projection.

3. View / Camera Transformation (3D → 2D)

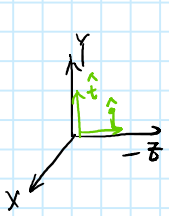
① how to take photos
   - build a model (Model)
   - find an angel to place camera (View)
   - take the picture (Projection)

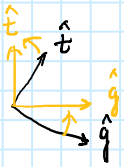
② Define camera pos.
   - Position $\vec{e}$
   - Look-at direction $\hat{g}$
   - Up direction $\hat{t}$ (camera本身的旋转)

③ Let camera be still and positioned at origin.
   - The origin, up at Y, look at $-Z$.

$\hat{t} = \hat{Y}$

④ Mview in math.
   - Translates e to origin.
   - Rotates g to $-Z$
   - Rotates t to Y





$M_{view} = R_{view} T_{view}$

Rotate g to $-Z$, t to Y is difficult to write out its Trans Matrix

Consider inverse rotation: X to $(\vec{g}\times\vec{t})$, Y to t, Z to $-g$

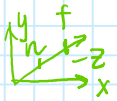
$$R_{view}^{-1} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & x_t & x_{-g} & 0 \\ y_{\cdots} & y_t & y_{-g} & 0 \\ z_{\cdots} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow R_{view} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t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

4. Projection Transformation.
   - Orthographic 正交
   - Perspective 透视

① Orthographic projection
   - Place camera correctly
   - Drop z coordinate
   - Translate and scale result to $[-1,1]\times[-1,1]$

In general

map a cuboid $[l,r]\times[b,t]\times[f,n]$ to canonical cube $[-1,1]^3$ 正则. 规范

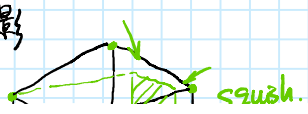


when looking at $-z$, farther side has a smaller z value

5. Perspective Projection.

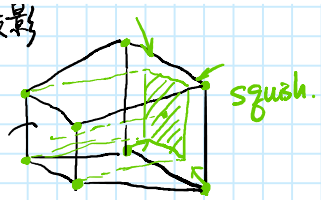
将 Perspective Projection 看作对远平面的"挤压"，再进行一次正交投影

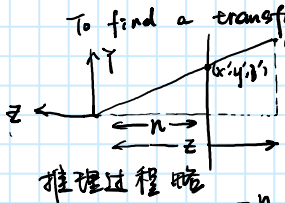
squish.



将 Perspective Projection 看作对远平面的挤压，再进行一次正交投影

squish.





推理过程略

$$y' = \frac{n}{z}y \qquad x' = \frac{n}{z}x \qquad z' = ?$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{n}{z}y \\ \frac{n}{z}x \\ ? \\ 1 \end{pmatrix} \Leftrightarrow \text{等价于} \begin{pmatrix} ny \\ nx \\ ? \\ n \end{pmatrix}$$

$$M_{persp-ortho} = \begin{bmatrix} -n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

其中 n 为近平面，f 为远平面 z 坐标

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

Question：对于 n,f 中间点，经过变换后，z 会如何变化？（z=n,f 时，squish 变换后 z 不变）

$$(0,0,n+f,-fn)\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow z' = \frac{(n+f)z - nf}{z} = (n+f) - \frac{nf}{z}$$

z 坐标为 z

当 $z \in (f,n)$ 时，$z' - z = z + \frac{nf}{z} - (n+f) = \frac{(z-n)(z-f)}{z} > 0$ （$\because z<0$）

$\therefore z'$ 会变大.

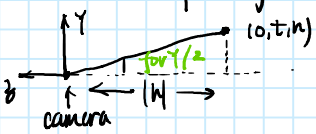
§ Rasterization (triangle)

1. perspective projection

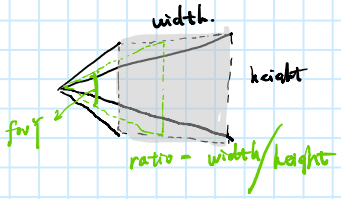
what's near plane's l,r,b,t ?

— sometimes people prefer to use field-of-view (fovY) and aspect ratio to specify a plane.

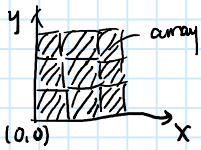
(assume symmetry)



$$\begin{cases} \tan\frac{fovY}{2} = \frac{t}{|n|} \\ aspect = \frac{r}{t} \end{cases}$$



2. screen space.



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

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

Screen covers range (0,0) → (width, height)

3. Canonical cube to Screen

- Transform in xy plane without regard to z

$[-1,1]^2$ to [0, width] × [0, height]

例 Viewport transform matrix

$$M_{viewport} = \begin{pmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

视口变换.

4. Raster displays

① Oscilloscope 示波器

② CRT

③ Frame Buffer: Memory for a raster display

④ Flat panel displays = LCD

⑤ Electronic ink.

5. triangle meshes is important

Why ?:

① basic

② planar

③ interior : well-defined (无凸凹性、洞等)

③ planar
⑤ interior: well-defined (无凸凹性、洞、等)
④ 便于插值

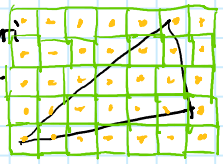
6. How to approximate a Triangle?

Input = position of 3 vertex. ⇒ out = rasterized Image

Sample:

Define inside(tri, x, y)

$$= \begin{cases} 1 & (x,y) \text{ inside tri} \\ 0 & \text{otherwise} \end{cases}$$

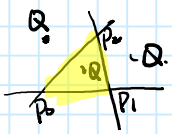


Consider center of pixels

If = center is inside the triangle

then = the pixel is sampled!

How to decide whether inside or not? ⇒ Cross Products

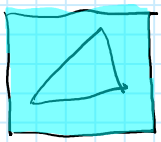


$\overrightarrow{P_2Q} \propto \overrightarrow{P_2P_0}$
$\overrightarrow{P_0Q} \times \overrightarrow{P_0P_1}$
$\overrightarrow{P_1Q} \times \overrightarrow{P_1P_2}$

} 全正或全负. Q才在△内.

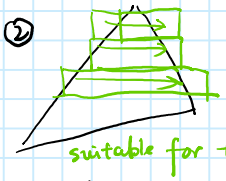
7. Boost efficiency:

① Use a Bounding box to increase efficiency



AABB = axis-aligned bounding box.

②



每一行考虑最左与最右.

suitable for thin and rotated triangles.

8. Aliasing (走样)

§ Rasterization 2. (Anti-aliasing & Z-buffering)

1. Sampling Artifacts (errors, mistakes, ...)

① Moiré Patterns
② Jaggies
③ Wagon wheel effect = 高速汽车车轮倒转.

Behind artifacts:

Signals are changing too fast

2. Antialiasing Idea:

Blurring (Pre-blurring) before sampling

Notice: Sample then filter ≠ Filter then Sample

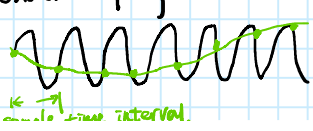
Blurred Aliasing

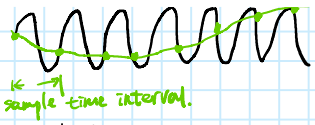
3. Frequency Domain 频域

Fourier transform = $F(w) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i w x} dx$

Inverse fourier transform: $f(x) = \int F(w) e^{2\pi i w x} dw$

Undersampling:





High-frequency signal is insufficiently sampled.

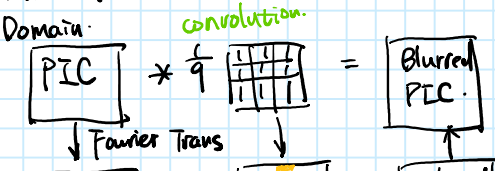
4. Filtering: Getting rid of certain frequency content.

低频：模糊的图像

高频：边界

5. Convolution Theorem

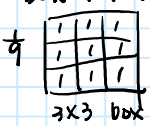
Spatial Domain



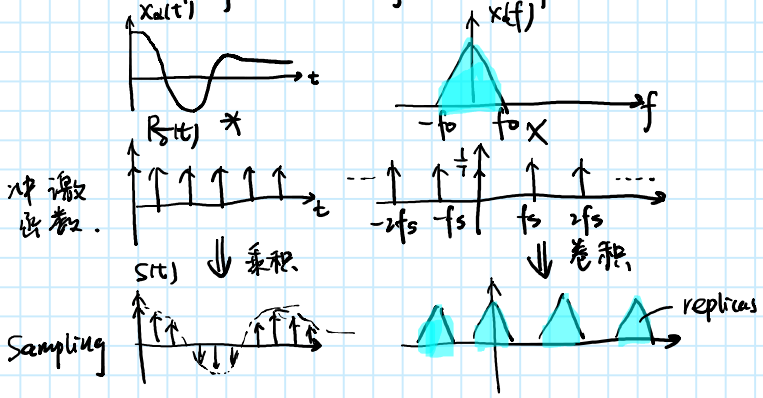
Fre. Domain

时域乘积 = 频域卷积

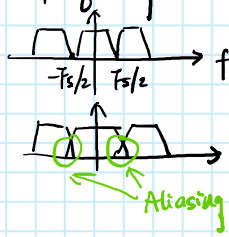
6. Box Filter



7. Sampling = Repeating Frequency Content



8. Aliasing = Mixed Frequency Contents



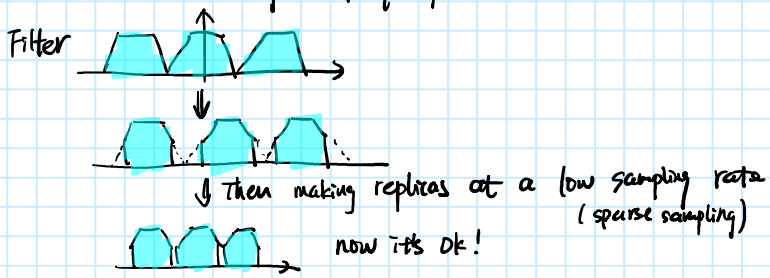
9. Anti-Aliasing

Option 1: sampling rate ↑

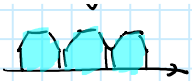
- increase distance between replicas in Fourier domain.
- Higher resolution displays

Option 2:

- make Fourier contents "narrower" before repeating
- take out higher freq. part



(sparse sampling)



now it's ok!

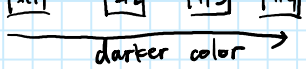
Practical Pre-filter

one-pixel box filter (low pass, blurring)

Solution:

- Convolve f(x,y) by 1-pixel box-blur
- Sample at center

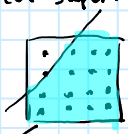
Original 

Filtered 

darker color

} #define inside(triangle, x, y)
as area of the pixel covered by the triangle

10. Supersampling (MSAA)



根据在三角形内点数量 n
n/16 × color 得到一个渐变色 模拟 inside(...) 中的 "area"

1 pixel 4×4 supersampling

Notice: MSAA is a pre-filtering method
(not increasing resolution or sampling)

11. Other Solutions

Milestones:

- FXAA (Fast Approximate AA)
- TAA (Temporal AA)

12. Super resolution / Super sampling

- From low to high resolution
- DLSS (Deep Learning Super Sampling)