Lecture 03: Geometric Transformations (Chapter 7,9)

I. Operations applied to the geometric description of an object to change its size, position, or orientation are called geometric transformations

A. Goal: Learn geometric transformations as they are the backbone to converting model coordinates to ~~viewing~~ world coordinates in the viewing pipeline

II. 2D Geometric Transformations (Chapter 7)

A. Basic Geometric Transformations

i. Translation - add offsets to the coordinates of a point to generate a new point.

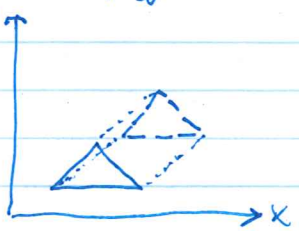
$(x', y') = (x + t_x, y + t_y)$ (t_x, t_y) is a translation vector

a. as matrices:

$P = \begin{bmatrix} x \\ y \end{bmatrix}$, $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$, $T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$ we will use P, P' throughout lecture.

$P' = P + T$

b. To translate ~~a~~ polygon - apply translation to each vertex and then regenerate the polygon. (same for circle, ellipse, curves) *Ask how for these*



△ old
△ new
/ translation vector

c. A rigid-body transformation moves an object w/out deformation

ii. Rotation - specified by rotation angle and rotation axis.

"rotate point by angle about axis"

a. in 2D the axis is perpendicular to the plane. Thus the rotation axis is considered a rotation, or pivot, point.

b. Angle θ is applied in a counterclockwise direction.

c. Assuming pivot at origin, point is distance from Origin at an angle ϕ from the x axis:

$x = r\cos\phi$
$y = r\sin\phi$

$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$
$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$

$x' = x\cos\theta - y\sin\theta$
$y' = x\sin\theta + y\cos\theta$ or as a matrix

$P' = RP$ where $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

about a general pivot (x_r, y_r)

$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$
$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$

will revisit later.

Ask how to apply to polygon, circle, ellipse, curve

iii. Scaling by scale factors (s_x, s_y)

a. $(x', y') = (s_x x, s_y y)$ or as matrices

$P' = SP$ where $S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$

b. if $s_x = s_y$ it is considered uniform otherwise differential

c. Negative values cause reflection about axes

d. To scale about a fixed point (x_s, y_s)

$x' = x_p + (x - x_p) s_x = x s_x + x_p (1 - s_x)$

$y' = y_s + (y - y_s) s_y = y s_y + y_s (1 - s_y)$

revisited later

Ask how for polygon, circle, ellipse, curves

B. Matrix representations and homogeneous coordinates

i. considering sequences of transformations, we want to compute these efficiently.

We could compute intermediate points after each transform but is very inefficient.

Instead combine transforms into a single transformation.

ii. Homogeneous coordinates - motivated by converting translation to a matrix multiplication. They are a D+1 coordinate representation where the last parameter is constant

$(x, y) \to (x_h, y_h, h) = (hx, hy, h)$. h must be non zero. For now we will always use $h = 1$. $P = \begin{bmatrix} x \\ y \end{bmatrix}$ $P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$

a. Translation: $T(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ $P' = T(t_x, t_y) P$

b. Rotation about origin: $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $P' = R(\theta) P$

c. Scaling about origin: $S(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $P' = S(s_x, s_y) P$

iii. Inverting transforms

a. Translation $T(t_x, t_y)^{-1} = T(-t_x, -t_y)$

b. rotation $R(\theta)^{-1} = R(-\theta) = R(\theta)^T$ *why R matrix is orthonormal*

c. Scaling $S(s_x, s_y)^{-1} = S(\frac{1}{s_x}, \frac{1}{s_y})$

C. Composite transformations – Apply series of transforms as 1 matrix. (order appears reversed because column major order)

$P' = (M_2(M_1 P))$ is less efficient than $P' = (M_2 M_1)P = MP$ because the same matrix is applied to an entire geometric object.

i. Two translations: $T(t_{2x}, t_{2y}) \cdot T(t_{1x}, t_{1y}) = T(t_{1x}+t_{2x}, t_{1y}+t_{2y})$ *Ask class on 1/2/3*

ii. Two Rotations: $R(\theta_2) \cdot R(\theta_1) = R(\theta_1+\theta_2)$

iii. Two scales: $S(s_{2x}, s_{2y})\, S(s_{1x}, s_{1y}) = S(s_{1x}s_{2x}, s_{1y}s_{2y})$

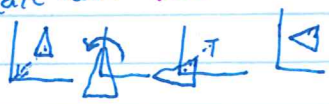
iv. General rotation about pivot: (pivot aligned w/ origin)

a. Algorithm: (1) Translate to origin (2) rotate about origin (3) Translate back

b. Matrices: $R(x_r, y_r, \theta) = T(x_r, y_r)\, R(\theta)\, T(-x_r, -y_r)$ (3) (2) (1)

c. Ex.

aside Approx *Center of mass* $\frac{\sum_{p \in O} p}{|O|}$

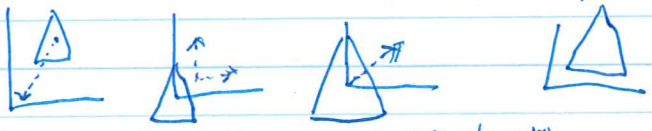


v. General scaling about fixed point

a. Algorithm: (1) translate to origin (align fixed point) (2) Scale (3) Translate back

b. Matrices: $S(x_f, y_f, s_x, s_y) = T(x_f, y_f)\, S(s_x, s_y)\, T(-x_f, -y_f)$ (1) (2) (3)

c. Ex.

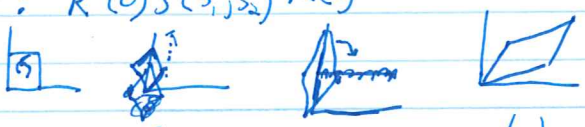


vi. General scaling directions (align directions w/ axis)

a. Algorithm: (1) rotate to origin (2) Scale (3) rotate back

b. Matrices: $R^{-1}(\theta)\, S(s_1, s_2)\, R(\theta)$

c. Ex:



vii. Composite transforms are associative but not always commutative.

viii. 2D Rigid transform is rotation + translation

$$T(t_x, t_y)\, R(x_r, y_r, \theta) = T(t_x, t_y)\, T(x_r, y_r)\, R(\theta)\, T(-x_r, -y_r)$$

D. Other transforms

i. Reflection

about X: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ about Y: $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ arbitrary: Rotate Reflect Rotate.

ii. Shear $\begin{bmatrix} 1 & hx & 0 \\ hy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ arbitrary point: translate shear translate.

iii. Applies to general reference frames translate origin to align. Rotate axis to align.

$$R(-\theta)\, T(-x_0, -y_0)$$

III. 3D Geometric Transformations (Chapter 9)

A. Uses homogeneous coordinates w/ 4 dimensions. Most transforms are analogous to 2D version. Rotation will be different.

B. Basic Geometric Transformations

i. Translation

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T(t_x, t_y, t_z)^{-1} = T(-t_x, -t_y, -t_z)$$

ii. Rotations

a. About coordinate axes

1. z-axis

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_z(\theta)^{-1} = R_z(\theta)^T = R_z(-\theta)$$

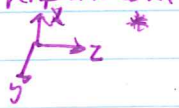
2. x-axis

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_x(\theta)^{-1} = R_x(\theta)^T = R_x(-\theta)$$

3. y-axis

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_y(\theta)^{-1} = R_y(\theta)^T = R_y(-\theta)$$

* Different because of perspective when looking down y-axis



4. Generally can combine. This representation is an "Euler-Angle" or Fixed angle rotation. Arbitrarily decide order of axis: e.g. x, y, z.

$$R(\alpha, \beta, \gamma) = R_z(\gamma) R_y(\beta) R_x(\alpha)$$

5. Remember for general rotations about fixed points:
(1) Translate to origin (2) Rotate object (3) Translate back

$$p' = T^{-1} R T p$$

6. About arbitrary axis: (Ch. 9-2)
(1) Translate axis to origin (2) Rotation axis coincides w/ axis (3) Rotate (4) invert rotation (5) invert translation

$$p' = T^{-1} R^{-1} R(\theta) R T p$$

For single rotation matrix easy to determine from rotation axis

b. Using Quaternions.

1. Issue w/ rotation matrices is that 4x4 matrix multiplication use lots of storage, require many multiplications, and are difficult to interpolate motions

2. Quaternions are a "3D" complex number that solves these issues

$q = (s, \vec{v})$ s - scalar component, v - vector component

* efficient scaling and adding which makes them useful *

$q = s + v_x\hat{i} + v_y\hat{j} + v_z\hat{k}$ in full form

3. Angle-axis rotations are easy to express $(\theta, \vec{u})$

$q = (s, \vec{v}) = \left(\cos\frac{\theta}{2}, \vec{u}\sin\frac{\theta}{2}\right)$

Point $\vec{p}$ to rotate as quaternion: $P = (0, \vec{p})$

So rotations are as follows:

$p' = qPq^{-1}$ where $q^{-1} = (s, -\vec{v})$ and $p' = (0, \vec{p'})$

$\vec{p'} = s^2\vec{p} + \vec{v}(\vec{p}\cdot\vec{v}) + 2s(\vec{v}\times\vec{p}) + \vec{v}\times(\vec{v}\times\vec{p})$

* Based on $q_1 q_2 = (s_1, \vec{v}_1)(s_2, \vec{v}_2) = (s_1 s_2 - \vec{v}_1\cdot\vec{v}_2,\ s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1\times\vec{v}_2)$

with $+(\vec{a}\times\vec{b}) = -(\vec{b}\times\vec{a})$ anticommutative property

and $\vec{a}\times(\vec{b}+\vec{c}) = \vec{a}\times\vec{b} + \vec{b}\times\vec{c}$ distributive property *

4. Easily converted back to a matrix

iii. Scaling

a. About origin

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad S(s_x, s_y, s_z)^{-1} = S\left(\frac{1}{s_x}, \frac{1}{s_y}, \frac{1}{s_z}\right)$$

b. About fixed point

(1) Translate to origin (2) Scale (3) Translate back

$P' = T^{-1}STP$

C. Composite transforms - Multiply matrices first then apply to whole object

D. Other transforms

i. Reflections - negative values in scaling matrix

ii. Shear about origin: $H = \begin{bmatrix} 1 & h_{yx} & h_{zx} & 0 \\ h_{xy} & 1 & h_{zy} & 0 \\ h_{xz} & h_{yz} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

E. Can transform between coordinate systems. Translate + Rotate to align Axis.

F. All transforms discussed are affine: parallel lines stay parallel, points stay as points