Chapter 3

Transformations

Transformations

- Transformation = *one-to-one* and *onto* mapping of R^n to itself
- *Affine* transformation – T(v) = Av+b
  - A – matrix
  - v, b – vectors
  - In 2D:

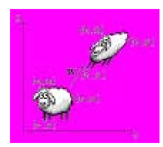
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Assignment 1 Marking

- Slots of 7 min each student:
  - Use piazza linked doodle poll to select time blocks
  - Signup sheet in **next** class + labs
- Code must compile and run on lab machines
- Timestamp: DON'T touch any files after deadline
- Be there & be ready (arrive 10 min early)
- Showcase your code & answer questions
  - Explain what works, what doesn't, what extras you added, etc...

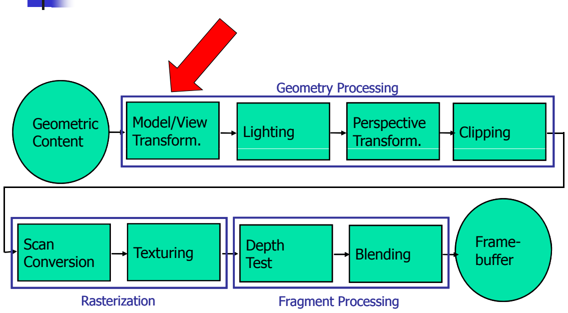
Geometric Transformations

- *Geometric Transformation* = affine transformation with geometric meaning



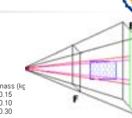
- Mathematically transformations are defined on vectors $\Rightarrow$ for point P, use vector P-Origin

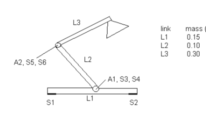
The Rendering Pipeline

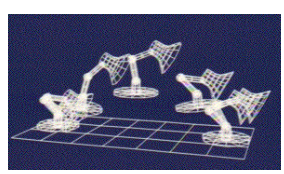


Applications

- Viewing (more later)
- Modeling
- Articulation







Modeling Transformations: sylabus

- 2D Transformations
- Homogeneous Coordinates
- 3D Transformations
- Composing Transformations
- Transformation Hierarchies
- Transforming Normals
- Assignment 2 – Robots
  - Use transformations to create and animate robots made from (scaled) boxes

Scaling

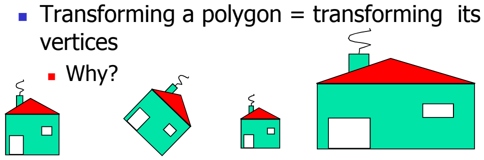
- Matrix form:

$$S^{(s_x,s_y)}(V) = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} s_x v_x \\ s_y v_y \end{pmatrix}$$

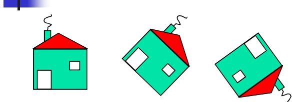
- Independent in x and y

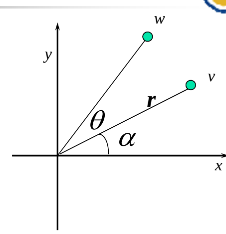
Transformations

- Transforming an object = transforming all its points
- Transforming a polygon = transforming its vertices
  - Why?



Rotation (using high school trigo…)





- Polar form:

$$v = \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} r\cos\alpha \\ r\sin\alpha \end{pmatrix}$$

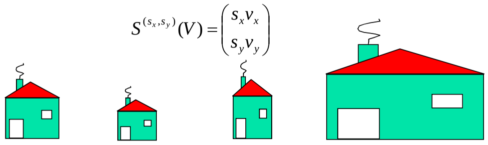
- Rotating v counterclockwise by θ to w:

$$w = \begin{pmatrix} w_x \\ w_y \end{pmatrix} = \begin{pmatrix} r\cos(\alpha+\theta) \\ r\sin(\alpha+\theta) \end{pmatrix} = \begin{pmatrix} r\cos\alpha\cos\theta - r\sin\alpha\sin\theta \\ r\cos\alpha\sin\theta + r\sin\alpha\cos\theta \end{pmatrix}$$

Scaling

- $V = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$ – vector in XY plane
- *Scaling* operator S with parameters (s_x, s_y):

$$S^{(s_x,s_y)}(V) = \begin{pmatrix} s_x v_x \\ s_y v_y \end{pmatrix}$$



Rotation

- Matrix form:

$$w = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} r\cos\alpha \\ r\sin\alpha \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} v$$

- *Rotation* operator R (at the origin) with parameter θ:

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

Rotation Properties

- R^{θ} is orthonormal

$$(R^{\theta})^{-1} = (R^{\theta})^{T}$$

- $R^{-\theta}$ - rotation by $-\theta$ is

$$R^{-\theta} = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = (R^{\theta})^{-1}$$

Translation: Homogeneous Coordinates

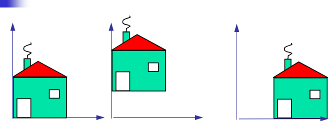
- Conversion (projection) from homogeneous space to Euclidean:

$$v = \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x^h / v_w^h \\ v_y^h / v_w^h \end{pmatrix}$$

- Projections is not 1:1

$$\begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 4 \\ 4 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0.5 \end{pmatrix} \text{ all project to } \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Translation



- Translation operator T with parameters (t_x, t_y):

$$T^{(t_x,t_y)}(v) = \begin{pmatrix} v_x + t_x \\ v_y + t_y \end{pmatrix}$$

- How can we write this in matrix form?

Translation

- Using homogeneous coordinates, translation operator may be expressed as:

$$T^{(t_x,t_y)}(v^h) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix} = \begin{pmatrix} v_x + t_x \\ v_y + t_y \\ 1 \end{pmatrix}$$

Translation: Homogeneous Coordinates

- To represent T in matrix form – introduce homogeneous coordinates:

$$v = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \rightarrow v^h = \begin{pmatrix} v_x^h \\ v_y^h \\ v_w^h \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

Homogeneous Coordinates

$$\textbf{R}otation = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\textbf{S}cale = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Other ideas for uniform scale?

3D Transformations

- All 2D transformations extend to 3D
- In homogeneous coordinates:

Scaling

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

glScalef(a,b,c);

Translatio n

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

glTranslatef(a,b,c);

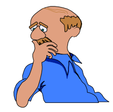
Rotation around the z axis

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

glRotatef(angle,0,0,1);

Transformations Quiz

- And these 2D homogeneous ones?



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

3D Rotation in X, Y

around x axis:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

glRotatef(angle,1,0,0);

around y axis:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

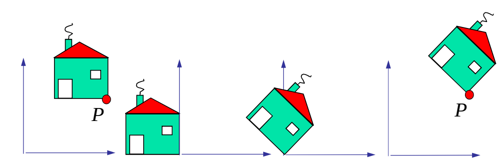
glRotatef(angle,0,1,0);

- general OpenGL command

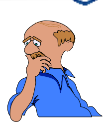
glRotatef(angle,x,y,z);

Transformation Composition

- What operation rotates XY by θ around $P = \begin{pmatrix} p_x \\ p_y \end{pmatrix}$?
- Answer:
  - Translate P to origin
  - Rotate around origin by θ
  - Translate back



Transformations Quiz



- What do these 2D transformations do?

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$$

Transformation Composition

$$T^{(p_x,p_y)} R^\theta T^{(-p_x,-p_y)} (V)$$

$$= \begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

Compositing of Affine Transformations

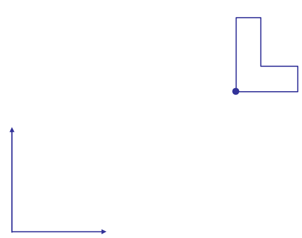
- In general:
  - Transform geometry into coordinate system where operation becomes simpler
  - Perform operation
  - Transform geometry back to original coordinate system
- Note: composition of affine transformations is an affine transformation

Compositing of Affine Transformations

- Two different interpretations of composite:
  - 1) read from inside-out as transformation of object
    - 1a) Translate object by –t
    - 1b) Rotate object by Φ
    - 1c) Translate object by t
  - 2) read from outside-in as transformation of the coordinate frame
    - 2c) Translate frame by t
    - 2b) Rotate frame by –Φ (i.e. rotate object by Φ)
    - 2a) Translate frame by –t

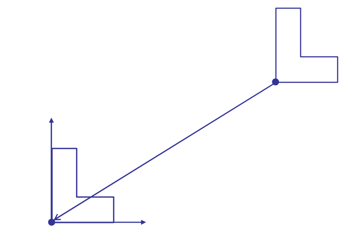
Compositing of Affine Transformations

- Example scene:



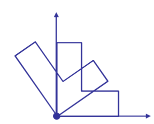
Compositing of Affine Transformations

- First Interpretation:
  - Step 1: translate object by –t (move to origin)



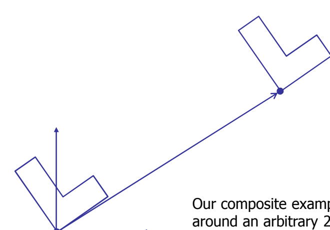
Compositing of Affine Transformations

- First Interpretation:
  - Step 2: rotate object by Φ



Compositing of Affine Transformations

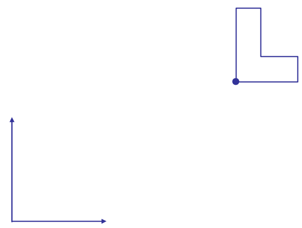
- First Interpretation:
  - Step 3: translate object by t (move back)



Our composite example is a rotation around an arbitrary 2D point with position t!

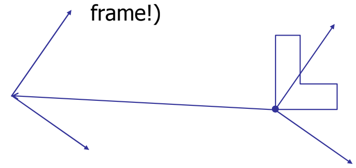
Compositing of Affine Transformations

- Example scene, second interpretation:



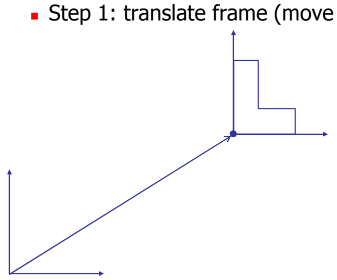
Compositing of Affine Transformations

- Second interpretation:
  - Step 3: translate frame back (vector -t in new frame!)



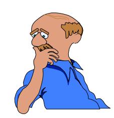
Compositing of Affine Transformations

- Second interpretation:
  - Step 1: translate frame (move origin to object)



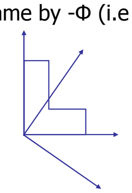
Transformations Composition

- How to mirror through arbitrary line in XY?



Compositing of Affine Transformations

- Second interpretation:
  - Step 2: rotate frame by -Φ (i.e rotate obj. by Φ)



Rotation About an Arbitrary Axis

- Axis defined by two points P_1 P_2
- Translate point P_1 to the origin
- Rotate to align P_1 P_2 axis with z-axis (or x or y)
  - How?
- Perform rotation
- Undo aligning rotations
- Undo translation

3D Transformations - Composition

- Does order matter?
  - Is $S_1S_2 = S_2S_1$?
  - Is $T_1T_2 = T_2T_1$?
  - Is $R_1R_2 = R_2R_1$?
  - Is $S_1R_2 = R_2S_1$?
  - .....

Undoing Transformations: Inverses

$$\mathbf{T}(x,y,z)^{-1} = \mathbf{T}(-x,-y,-z)$$
$$\mathbf{T}(x,y,z)\,\mathbf{T}(-x,-y,-z) = \mathbf{I}$$

$$\mathbf{R}(z,\theta)^{-1} = \mathbf{R}(z,-\theta) = \mathbf{R}^{T}(z,\theta)$$ **(R is orthogonal)**
$$\mathbf{R}(z,\theta)\,\mathbf{R}(z,-\theta) = \mathbf{I}$$

$$\mathbf{S}(sx,sy,sz)^{-1} = \mathbf{S}(\frac{1}{sx},\frac{1}{sy},\frac{1}{sz})$$
$$\mathbf{S}(sx,sy,sz)\,\mathbf{S}(\frac{1}{sx},\frac{1}{sy},\frac{1}{sz}) = \mathbf{I}$$

Composing Translations

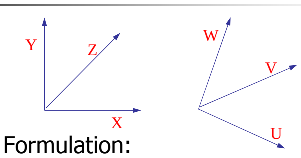
$$T1 = T(dx_1,dy_1) = \begin{bmatrix} 1 & & & dx_1 \\ & 1 & & dy_1 \\ & & 1 & \\ & & & 1 \end{bmatrix} \qquad T2 = T(dx_2,dy_2) = \begin{bmatrix} 1 & & & dx_2 \\ & 1 & & dy_2 \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

$$P'' = T2 \bullet P' = T2 \bullet [T1 \bullet P] = [T2 \bullet T1] \bullet P, \text{where}$$

$$T2 \bullet T1 = \begin{bmatrix} 1 & & & dx_1 + dx_2 \\ & 1 & & dy_1 + dy_2 \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

Translations add

Switching Coordinate Systems



- Problem Formulation:
  - Given two orthonormal coordinate systems XYZ and UVW
  - Find transformation from one to the other
- Answer:
  - Transformation matrix R whose columns are U,V,W (in XYZ system):

$$R = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}$$

Composing Transformations

- scaling

$$S2 \bullet S1 = \begin{bmatrix} sx_1 * dx_2 & & & \\ & sy_1 * sy_2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

scales multiply

- rotation

$$R2 \bullet R1 = \begin{bmatrix} \cos(\theta 1 + \theta 2) & -\sin(\theta 1 + \theta 2) & & \\ \sin(\theta 1 + \theta 2) & \cos(\theta 1 + \theta 2) & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

rotations add

Switching Coordinate Systems

- Proof:

$$R(X) = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = U$$

- Similarly $R(Y) = V$ & $R(Z) = W$

Switching Coordinate Systems

- Inverse (=transpose) transformation R^{-1} provides mapping from UVW to XYZ
- E.g.

$$R^{-1}(U) = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{bmatrix} \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \begin{pmatrix} u_x^2 + u_y^2 + u_z^2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = X$$

- Comment: Used for mapping between XY and arbitrary plane

What each transformation preserves

	Straight lines	parallel lines	distance	angles	normals
scaling					
rotation					
translation					
shear					