Geometric Transformations

Readings: Chapters 5-6

□□□□□□□□□□

Announcement

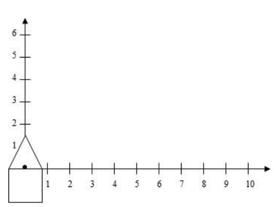
- Proj 1 posted on 2/2, due two weeks from today.

Exercises

- Translate [1,3] by [7,9]
- Scale [2,3] by 5 in the X direction and 10 in the Y direction
- Rotate [2,2] by 90˚ (π/2)
- What is the result for each of the following two cases applied to the house in the figure.

Case 1: Translate by x=6, y=0 then rotate by 45º

Case 2: Rotate by 45 and then translate by x=6, y=0



Examples

- Translate [1,3] by [7,9]

$$\begin{bmatrix}1 & 0 & 7\\0 & 1 & 9\\0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}1\\3\\1\end{bmatrix}=\begin{bmatrix}8\\12\\1\end{bmatrix}$$

- Scale [2,3] by 5 in the X direction and 10 in the Y direction

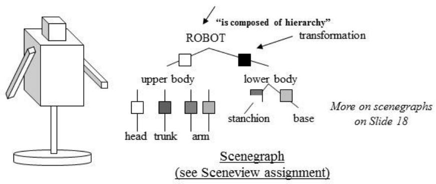
$$\begin{bmatrix}5 & 0 & 0\\0 & 10 & 0\\0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}2\\3\\1\end{bmatrix}=\begin{bmatrix}10\\30\\1\end{bmatrix}$$

- Rotate [2,2] by 90˚ (π/2)

$$\begin{bmatrix}\cos(\pi/2) & -\sin(\pi/2) & 0\\\sin(\pi/2) & \cos(\pi/2) & 0\\0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}2\\2\\1\end{bmatrix}=\begin{bmatrix}0 & -1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}2\\2\\1\end{bmatrix}=\begin{bmatrix}-2\\2\\1\end{bmatrix}$$

Why do we need geometric Transformations (T,R,S) in Graphics?

- Scene graph (DAG) construction from primitives

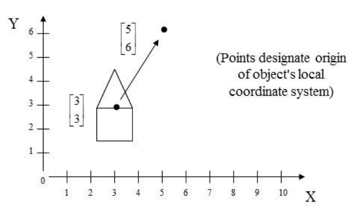


- Object motion (this lecture)
- Camera motion (next lecture on viewing)

2D Transformations

2D Translation

dx = 2
dy = 3



- Component-wise addition of vectors

$v' = v + t$ where $v = \begin{bmatrix} x \\ y \end{bmatrix}, \quad v' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad t = \begin{bmatrix} dx \\ dy \end{bmatrix}$

and $x' = x + dx$

$y' = y + dy$

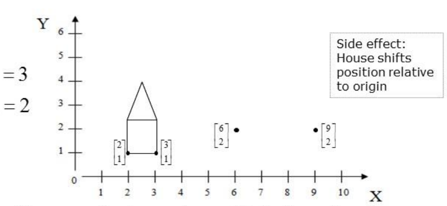
To move polygons: translate vertices (vectors) and redraw lines between them

- Preserves lengths (isometric)
- Preserves angles (conformal)
- Translation is thus "rigid-body"

2D Scaling

$s_x = 3$

$s_y = 2$



- Component-wise scalar multiplication of vectors

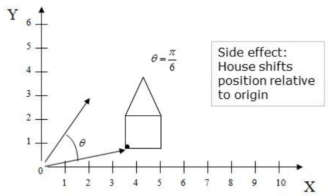
$v' = Sv$ where $v = \begin{bmatrix} x \\ y \end{bmatrix}, \quad v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

and $S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$ $\quad x' = s_x x$ $\quad y' = s_y y$

- Does not preserve lengths
- Does not preserve angles (except when scaling is uniform)

2D Rotation

Rotate by θ about the origin



$v' = R_\theta v$ where $v = \begin{bmatrix} x \\ y \end{bmatrix}$, $v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

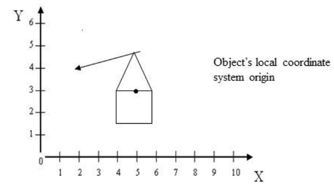
and $x' = x \cos \Theta - y \sin \Theta$
$y' = x \sin \Theta + y \cos \Theta$
$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

A rotation by 0 angle, i.e. no rotation at all, gives us the identity matrix

- Preserves lengths in objects, and angles between parts of objects
- Rotation is rigid-body

2D Rotation and Scale are Relative to Origin

- Suppose object is not centered at origin and we want to scale and rotate it.
- Solution: move to the origin, scale and/or rotate *in its local coordinate system*, then move it back.



- This sequence suggests the need to compose successive transformations…

Homogenous Coordinates

- Translation, scaling and rotation are expressed as:

  translation: $v' = v + t$

  scale: $v' = Sv$

  rotation: $v' = Rv$

- Composition is difficult to express
  - translation is not expressed as a matrix multiplication
- Homogeneous coordinates allows expression of all three transformations as 3x3 matrices for easy composition

$$P_{2d}(x, y) \rightarrow P_h(wx, wy, w), \quad w \neq 0$$
$$P_h(x', y', w), \quad w \neq 0$$
$$P_{2d}(x, y) = P_{2d}\left(\frac{x'}{w}, \frac{y'}{w}\right)$$

- *w* is 1 for affine transformations in graphics
- Note: $\mathrm{p} = (\mathrm{x}, \mathrm{y})$ becomes $\mathrm{p} = (\mathrm{x}, \mathrm{y}, 1)$

  This conversion does not transform *p*. It is only changing notation to show it can be viewed as a point on *w = 1* hyperplane

2D Homogeneous Coordinate Transformations

- For points written in homogeneous coordinates,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

translation, scaling and rotation relative to the origin are expressed homogeneously as:

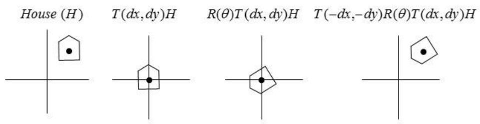
$$T_{(d_x, d_y)} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \quad v' = T_{(d_x, d_y)} v$$

$$S_{(s_x, s_y)} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad v' = S_{(s_x, s_y)} v$$

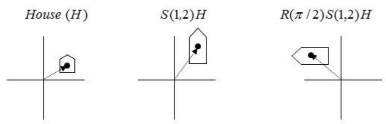
$$R_{(\phi)} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad v' = R_{(\phi)} v$$

Matrix Compositions: Using Translation

- Avoiding unwanted translation when scaling or rotating an object not centered at origin:
  - translate object to origin, perform scale or rotate, translate back.



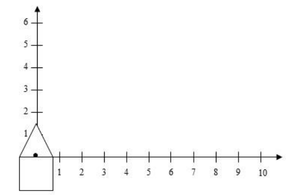
- How would you scale the house by 2 in “its” y and rotate it through 90° ?



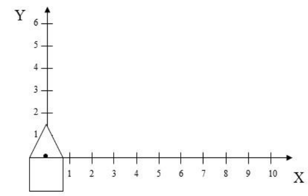
- Remember: matrix multiplication is <u>not</u> commutative! Hence order matters! (Refer to class website: resources/SIGGRAPH2001_courseNotes_source/transformation for camera transformation)

Matrix Multiplication is NOT Commutative

Translate by x=6, y=0 then rotate by 45º



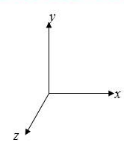
Rotate by 45º then translate by x=6, y=0



3D Transformations

3D Basic Transformations (1/2)

(right-handed coordinate system)



- Translation $\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- Scaling $\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

3D Basic Transformations (2/2)

(right-handed coordinate system)

- Rotation about X-axis

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

- Rotation about Y-axis

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

- Rotation about Z-axis

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

More transformation matrices

Skew/Shear/Translate (1/2)

- "Skew" a scene to the side:

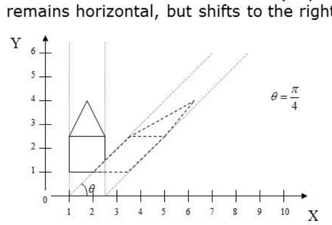
$$Skew_\theta = \begin{bmatrix} 1 & \frac{1}{\tan\theta} \\ 0 & 1 \end{bmatrix}$$

2D non-homogeneous

$$Skew_\theta = \begin{bmatrix} 1 & \frac{1}{\tan\theta} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2D homogeneous

- Squares become parallelograms - x coordinates skew to right, y coordinates stay same
- Degree between axes becomes θ
- Like pushing top of deck of cards to the side – each card shifts relative to the one below
- Notice that the base of the house (at y=1) remains horizontal, but shifts to the right.



NB: A skew of 0 angle, i.e. no skew at all, gives us the identity matrix, as it should

Scene graph manipulation

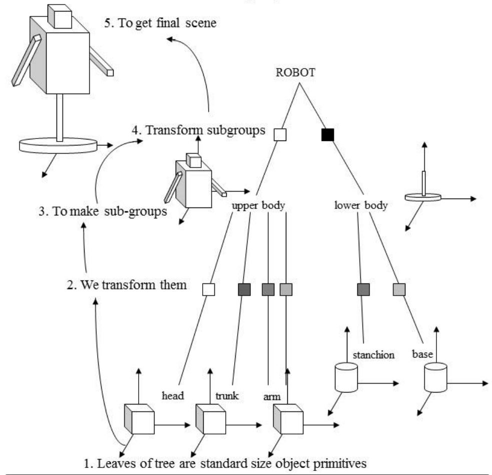
Transformations in Scene Graphs
(1/3)

- 3D scenes are often stored in a *scene graph:*
  - Open Scene Graph
  - Sun's Java3D™
  - X3D ™ (VRML ™ was a precursor to X3D)

- Typical scene graph format:
  - **objects** (cubes, sphere, cone, polyhedra etc.)
    - stored as nodes (default: unit size at origin)
  - **attributes** (color, texture map, etc.) and **transformations** are also nodes in scene graph (labeled edges on slide 2 are an abstraction)

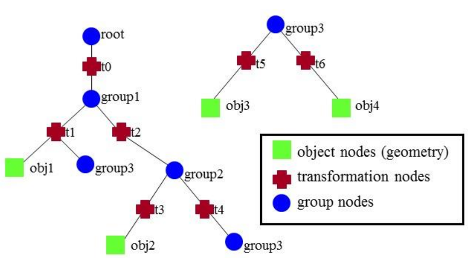
Transformations in Scene Graphs
(2/3)

Closer look at Scenegraph from slide 2 …



Transformations in Scene Graphs (3/3)

- Transformations affect all child nodes
- Sub-trees can be reused, called group nodes
  - instances of a group can have different transformations applied to them (e.g. group3 is used twice– once under t1 and once under t4)
  - must be defined before use



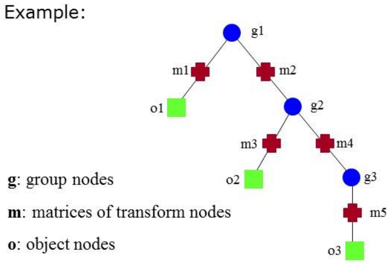
Composing Transformations in a Scene Graph (1/2)

- Transformation nodes contain at least a matrix that handles the transformation;
  - may also contain individual transformation parameters

- To determine final composite transformation matrix (CTM) for object node:
  - compose all parent transformations during prefix graph traversal
  - exact detail of how this is done varies from package to package, so be careful

Composing Transformations in a Scene Graph (2/2)

- Example:



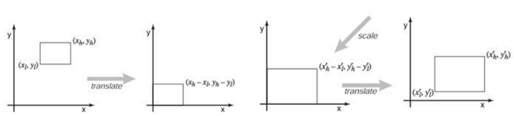
- for o1, CTM = m1
- for o2, CTM = m2* m3
- for o3, CTM = m2* m4* m5

- for a vertex v in o3, position in the world (root) coordinate system is:
CTM v = (m2*m4*m5)v

Exercises

E1: Windowing transforms

- Create a transform matrix that takes points in the rectangle $[x_l, x_h] \times [y_l, y_h]$ to the rectangle $[x_l', x_h'], [y_l', y_h']$ (Shirley Figure 6.18)



$$\begin{bmatrix} 1 & 0 & x_l' \\ 0 & 1 & y_l' \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x_h'-x_l'}{x_h-x_l} & 0 & 0 \\ 0 & \frac{y_h'-y_l'}{y_h-y_l} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_l \\ 0 & 1 & -y_l \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{x_h'-x_l'}{x_h-x_l} & 0 & \frac{x_l' x_h - x_h' x_l}{x_h-x_l} \\ 0 & \frac{y_h'-y_l'}{y_h-y_l} & \frac{y_l' y_h - y_h' y_l}{y_h-y_l} \\ 0 & 0 & 1 \end{bmatrix}$$

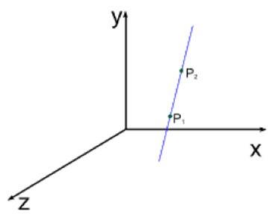
E2: Transformation between two coordinate systems

- Create a transform matrix that takes a point in the (X,Y) coordinate system to the point in the (U, V) coordinate system; and vice versa (Shirley Figure 6.20)

Take-home exercise

E3: Rotation about arbitrary axis

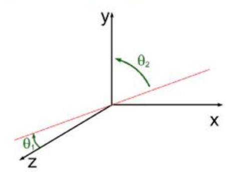
- To rotate about axis through P_1P_2



Rotation axis specification

(right-handed coordinate system)

1. Point and two rotation angles



2. Two points

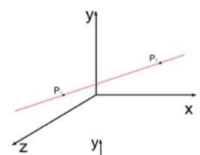
$V = P_2 - P_1$

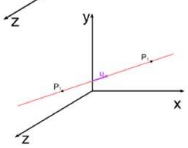
$U = V / |V| = (a,b,c)$

$a = (x_2 - x_1) / |V|$

$b = (y_2 - y_1) / |V|$

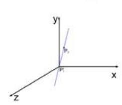
$c = (z_2 - z_1) / |V|$



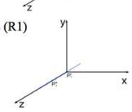


Rotation about arbitrary axis

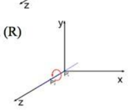
- Translate P_1 to origin (T)



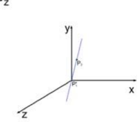
- Rotate until P_2 lies on z axis (R1)



- Perform desired rotation (R)



- Rotate axis back (R_1^{-1})



- Translate axis back (T^{-1})