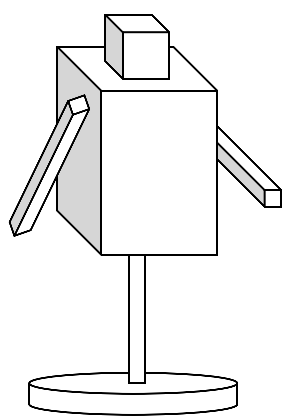
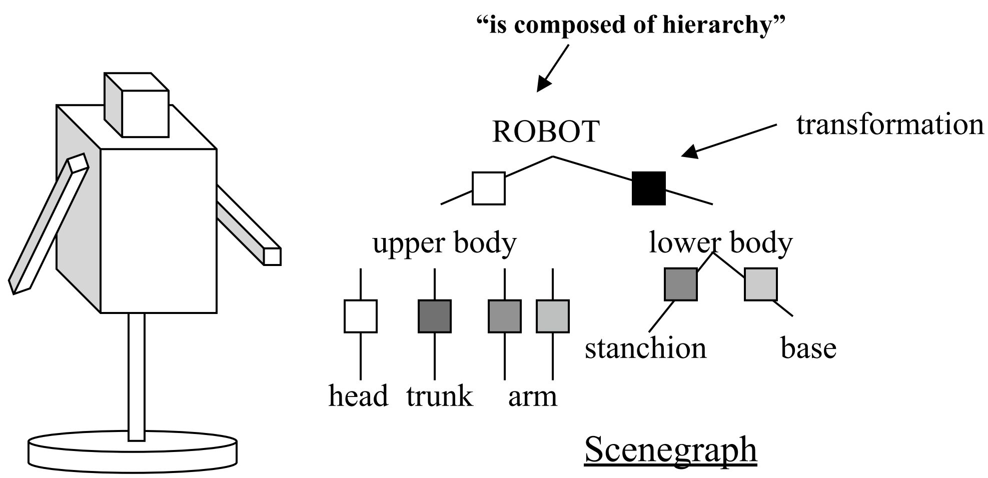
Geometric Transformations



Liz Marai

How Are Geometric Transformations (T,R,S) Used in Computer Graphics?

- Object construction using assemblies/hierarchy of parts à la Sketchpad's masters and instances; leaves of scenegraph contain primitives



- Aid to realism
  - objects, camera use realistic motion

- Synthetic camera/viewing

- Note: Helpful applets
  - Experiment with these concepts on the cs1566 webpage: *Applets->Linear Algebra* and *Applets->Scenegraphs*

Using Matrix Notation

- Can express sums of products more compactly (see non-geometric example from last time):

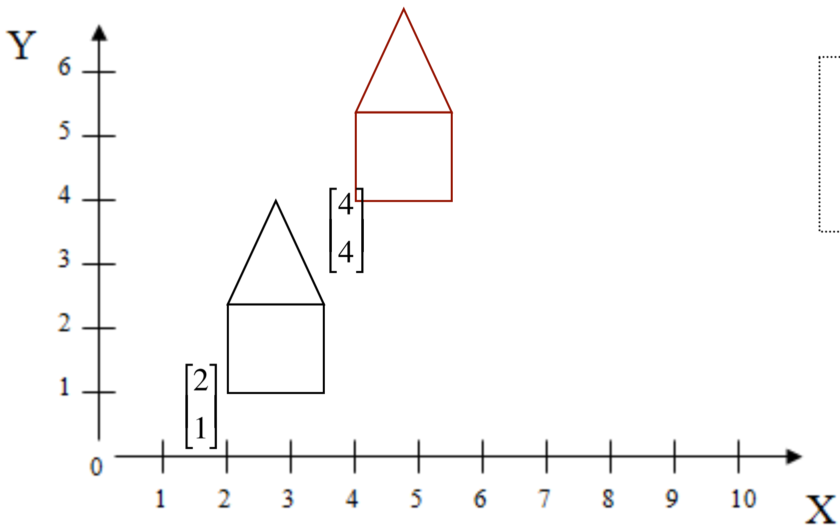
$$P(All) = \begin{bmatrix} totalCost_A \\ totalCost_B \\ totalCost_C \end{bmatrix} = \begin{bmatrix} 0.20 & 0.93 & 0.64 & 1.20 \\ 0.65 & 0.95 & 0.75 & 1.40 \\ 0.95 & 1.10 & 0.90 & 3.50 \end{bmatrix} \begin{bmatrix} 6 \\ 5 \\ 1 \\ 2 \end{bmatrix}$$

- Determine totalCost vector by row-column multiplication
  - dot product is the sum of the pairwise multiplications
    - Apply this operation to rows of prices and column of quantities

$$\begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = ax + by + cz + dw$$

2D Translation

dx = 2
dy = 3



Note: House shifts position relative to origin

- Component-wise addition of vectors

$\boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{t}$ 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 $x' = x + dx$

$y' = y + dy$

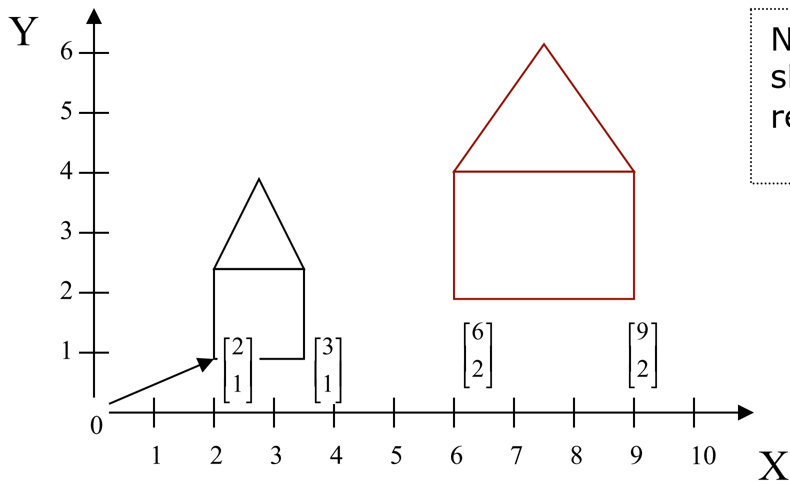
To move polygons: translate vertices (vectors) and redraw lines between them

- Preserves lengths (isometric)
- Preserves angles (conformal)

2D Scaling

$s_x = 3$

$s_y = 2$



Note: House shifts position relative to origin

- Component-wise scalar multiplication of vectors

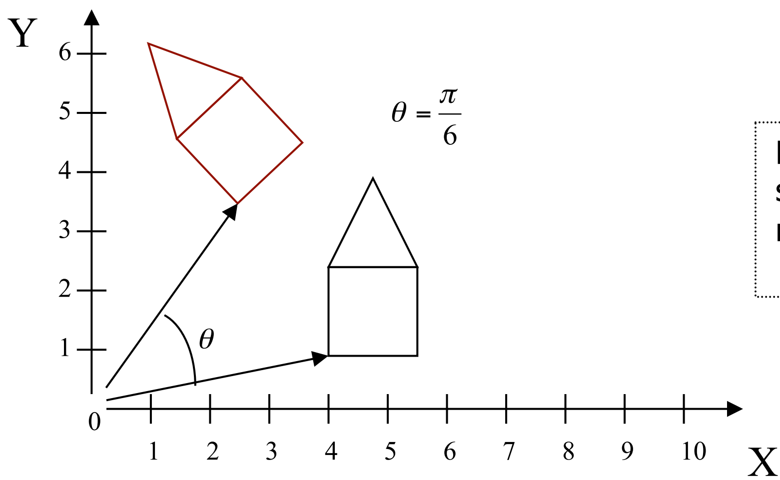
$\boldsymbol{v'} = \boldsymbol{Sv}$ where $v = \begin{bmatrix} x \\ y \end{bmatrix}, \quad v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

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

$y' = s_y y$

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

2D Rotation



Note: House shifts position relative to origin

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

- Rotation of vectors through an angle θ

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

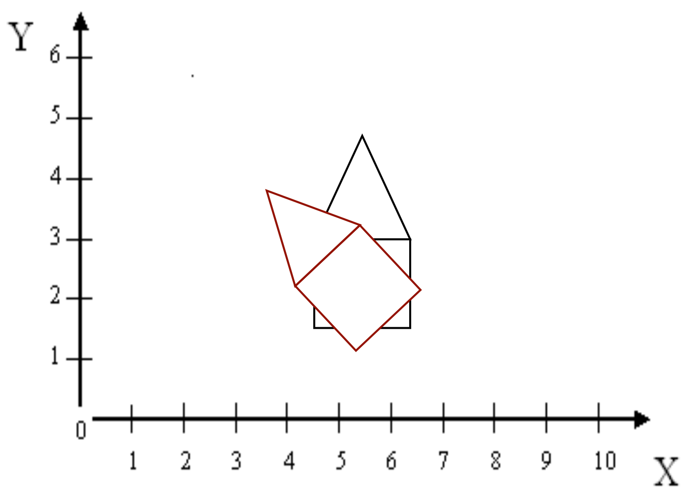
and $x' = x \cos \Theta - y \sin \Theta$
$y' = x \sin \Theta + y \cos \Theta$

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

- Preserves lengths and angles

2D Rotation and Scale are Relative to Origin

- Suppose object is not centered at origin
- Solution: move to the origin, scale and/or rotate, then move it back.



- Would like to compose successive transformations…

Homogeneous Coordinates

- Translation, scaling and rotation are expressed (non-homogeneously) 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 as 3x3 matrices for easy composition

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

$$P_h(x', y', w), \quad w \neq 0$$

$$P_{2d}(x, y) = P_{2d}\left(\frac{x'}{w}, \frac{y'}{w}\right)$$

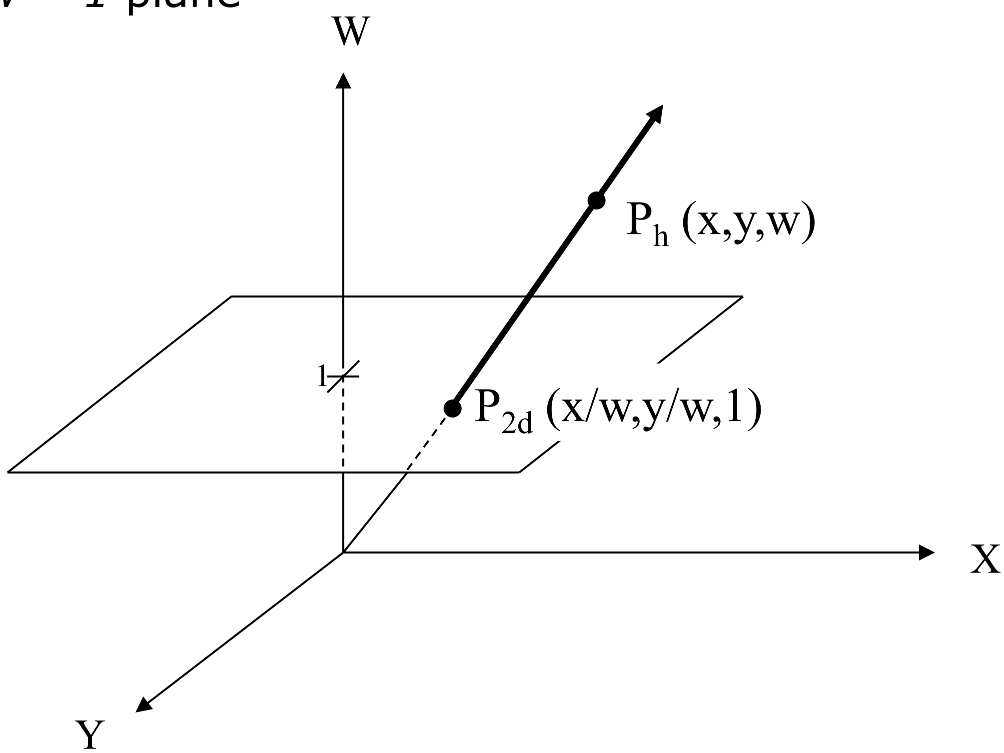
- w is 1 for affine transformations in graphics

(affine transformation = linear transformation followed by translation)

What is $\begin{bmatrix} x \\ y \\ w \end{bmatrix}$?

- P_{2d} is intersection of line determined by P_h with the

  w = 1 plane



- Infinite number of points correspond to $(x, y, 1)$: they constitute the whole line (tx, ty, tw)

2D Homogeneous Coordinate Transformations (1/2)

- 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:

$$T(dx, dy) = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \qquad v' = T(dx, dy)v$$

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

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

2D Homogeneous Coordinate Transformations (2/2)

- Consider the rotation matrix:

$$R(\phi) = \left[\begin{array}{cc|c} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ \hline 0 & 0 & 1 \end{array}\right]$$

- The 2 x 2 submatrix columns are:
  - unit vectors (length=1)
  - perpendicular (dot product=0)

- The 2 x 2 submatrix rows are:
  - unit vectors
  - perpendicular
- Preserves lengths and angles of original geometry. Therefore, the R matrix is a "rigid body" transformation.

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

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

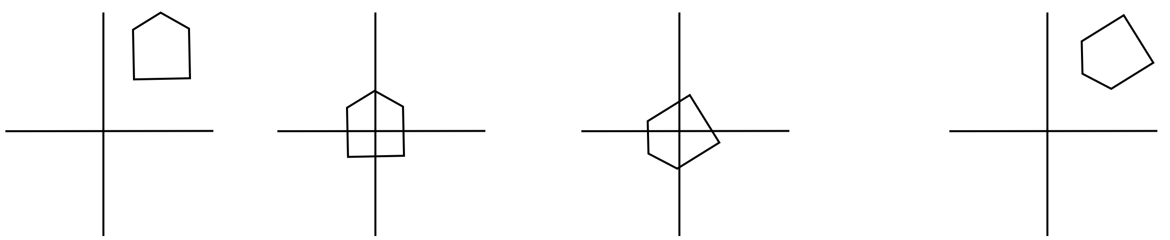
- Rotate [2,2] by 90° (π/2)

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

Using Matrix Compositions

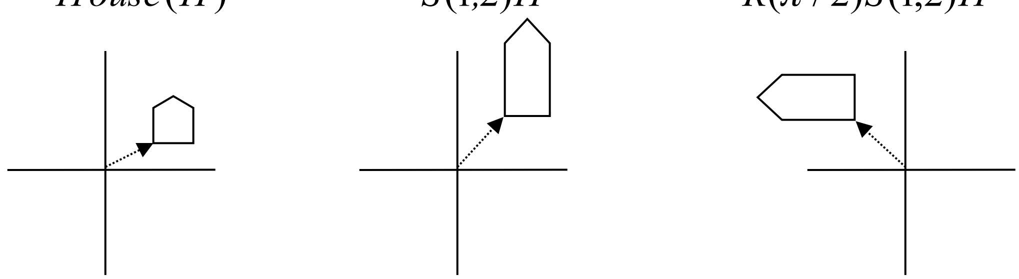
- Avoiding unwanted translation when scaling or rotating an object not centered at origin:
  - translate object to origin, perform scale or rotate, translate back.

$House(H)$ $\quad T(dx,dy)H \quad R(\theta)T(dx,dy)H \quad T(-dx,-dy)R(\theta)T(dx,dy)H$



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

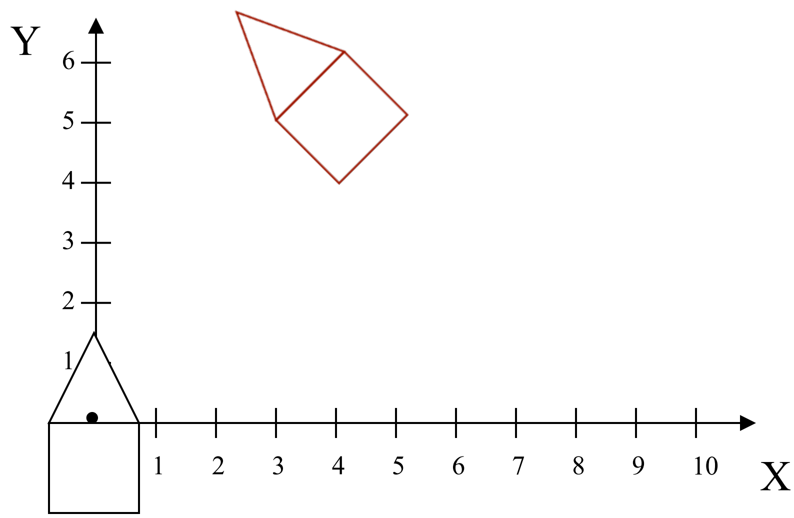
$House(H) \quad S(1,2)H \quad R(\pi/2)S(1,2)H$



- Remember: matrix multiplication is <u>not</u> commutative! Hence order matters! (refer to the Transformation Game at Applets->*Scenegraphs*)

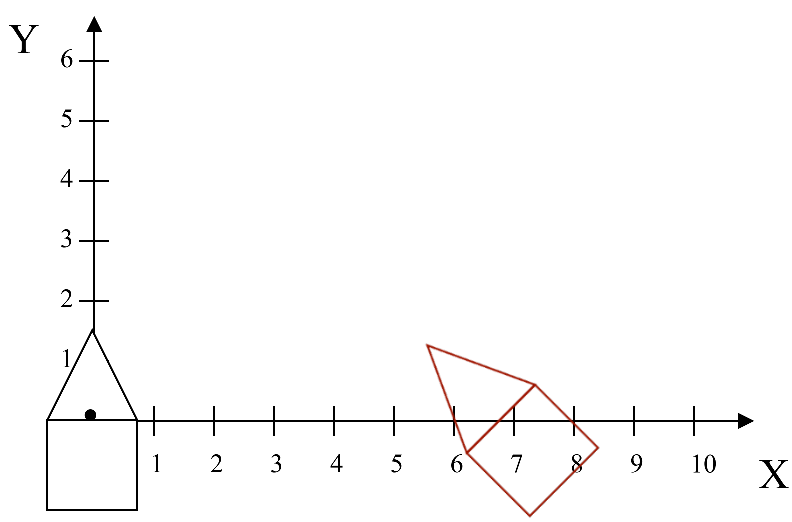
Transformations are NOT Commutative

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



Translation → Rotation

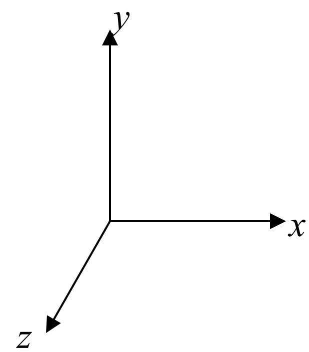
Rotate by 45º then translate by x=6, y=0



Rotation → Translation

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

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

Rodrigues's Formula…

- Rotation by angle θ around vector $\boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$

  Note: This is an arbitrary **unit** vector $\boldsymbol{u}$ in xyz space

- Here's a not so friendly rotation matrix:

$$R = \begin{bmatrix} \cos\theta + u_x^2\left(1-\cos\theta\right) & u_x u_y\left(1-\cos\theta\right) - u_z\sin\theta & u_x u_z\left(1-\cos\theta\right) + u_y\sin\theta \\ u_y u_x\left(1-\cos\theta\right) + u_z\sin\theta & \cos\theta + u_y^2\left(1-\cos\theta\right) & u_y u_z\left(1-\cos\theta\right) - u_x\sin\theta \\ u_z u_x\left(1-\cos\theta\right) - u_y\sin\theta & u_z u_y\left(1-\cos\theta\right) + u_x\sin\theta & \cos\theta + u_z^2\left(1-\cos\theta\right) \end{bmatrix}.$$

Homogeneous Coordinates

Some uses we'll be seeing later

- Placing sub-objects in parent's coordinate system to construct hierarchical scene graph
  - transforming primitives in own coordinate system
- View volume normalization
  - mapping arbitrary view volume into canonical view volume along z-axis
- Parallel (orthographic, oblique) and perspective projection
- Perspective transformation

Skew/Shear/Translate (1/2)

- "Skew" a scene to the side:

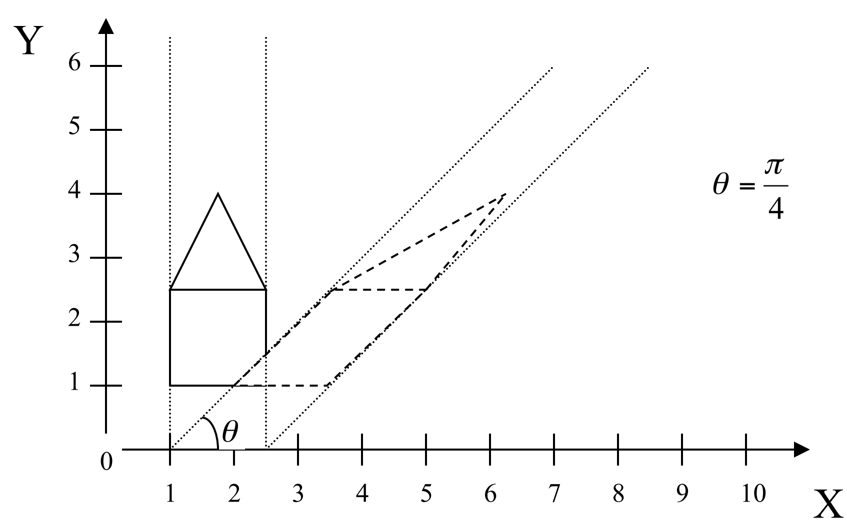
$$Skew_\theta = \begin{bmatrix} 1 & \frac{1}{\tan\theta} \\ 0 & 1 \end{bmatrix}$$

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
- 90° between axes becomes θ
- Like pushing top of deck of cards to the side – each card shifts relative to the one below
- Hmmm… 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

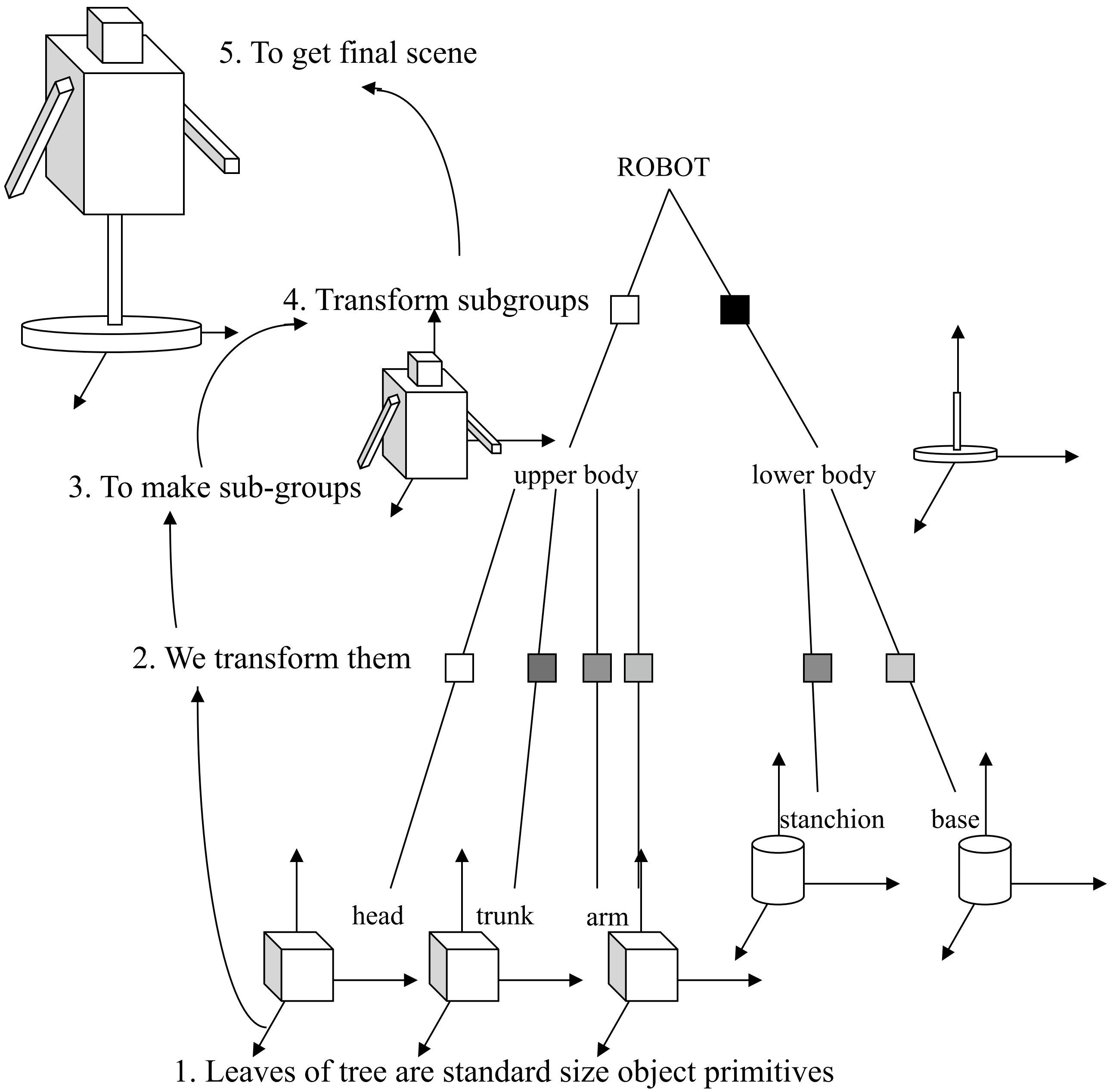
Transforms in Scene Graphs (1/3)

- 3D scenes are often stored in a directed acyclic graph (DAG) called a *scene graph*

- 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)

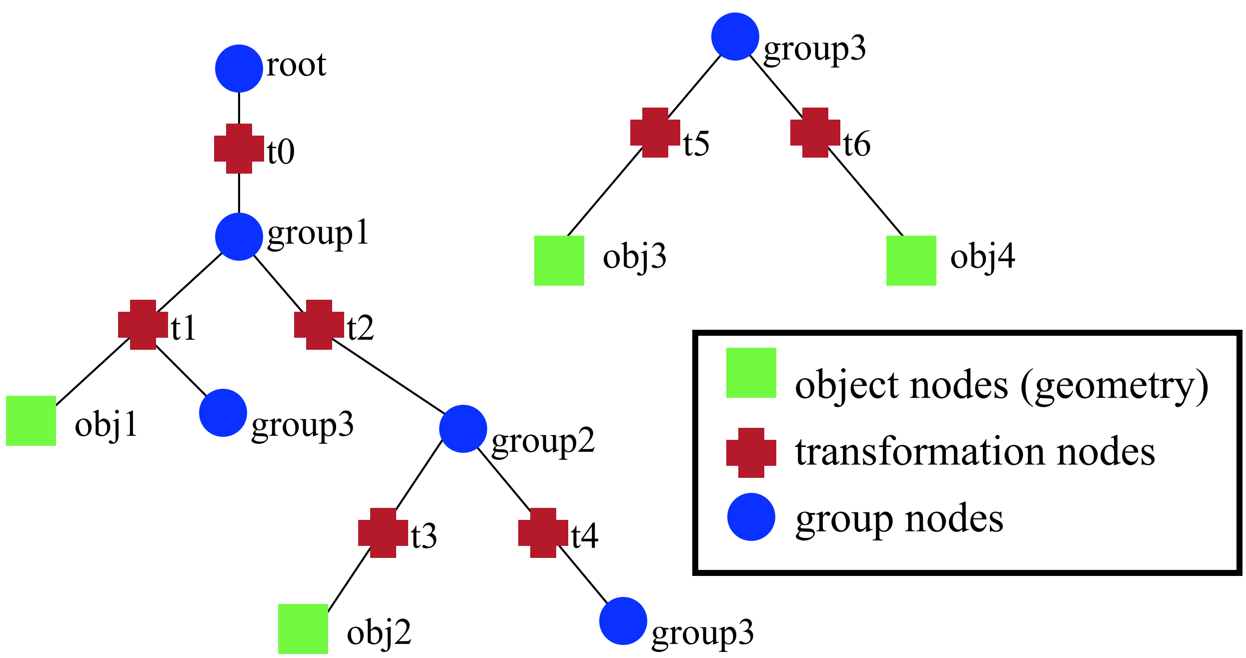
Transforms in Scene Graphs (2/3)

Closer look at Scenegraph from slide 2 …



Transforms in Scene Graphs (3/3)

- Below, transformation t0 affects all objects
- t2 affects only obj2 and one instance of group3 (includes instance of obj3 and obj4)
  - t2 doesn't affect obj1, other instance of group3



- Note: to use multiple instances of a sub-tree (i.e. group3), must define it before use
  - easier to implement

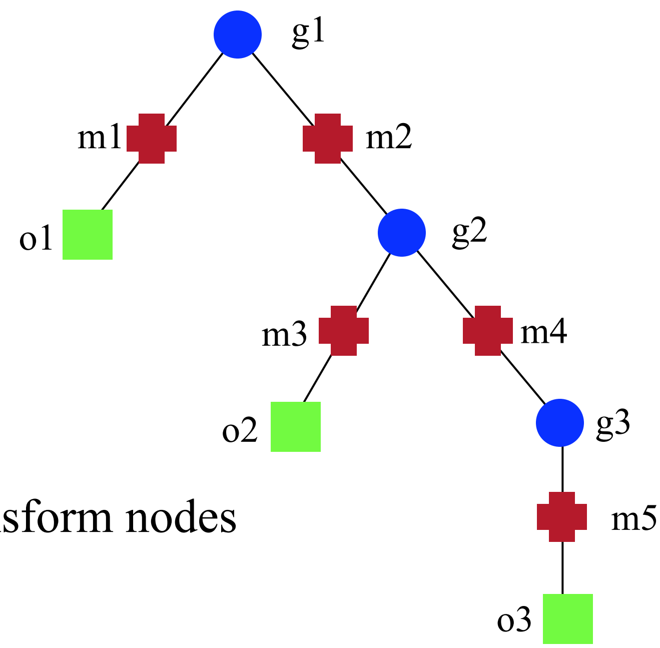
Composing Transformations in a Scene Graph (1/2)

- Transformation nodes contain at least a matrix that handles the transformation;
  - may also contain individual transformation parameters
  - refer to scene graph hierarchy applet by Dave Karelitz (URL on slide 2)

- 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:



g: group nodes

m: matrices of transform nodes

o: object nodes

- 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

Summary

- Geometric Transformations: essential in CG
- Using Matrix Notation
- 2D Translation, 2D Scaling, 2D Rotation
- Isometry and conformity
- Homogeneous coordinates (rationale and use)
- 3D Translation, 3D Scaling, 3D Rotation
- Rotation around an arbitrary axis
- Transformation Composition (order matters)
- Transformations in Scenegraphs
- Computing the final CTM