IMGD 1001 - The Game Development Process: 3D Modeling and Transformations

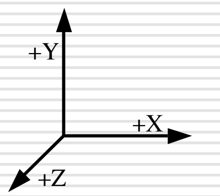
by
Robert W. Lindeman (gogo@wpi.edu)
Kent Quirk (kent_quirk@cognitoy.com)
(with lots of input from Mark Claypool!)

Overview of 3D Modeling

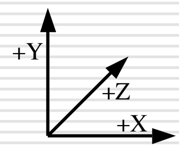
- Modeling
  - Create 3D model of scene/objects
- Coordinate systems (left hand, right hand)
- Basic shapes (cone, cylinder, *etc.*)
- Transformations/Matrices
- Lighting/Materials
- Synthetic camera basics
- View volume
- Projection

Coordinate Systems

- Right-handed and left-handed coordinate systems
  - Make an "L" with index finger and thumb
  - No real "standard," but...
  - Converting from one to the other is a simple transformation



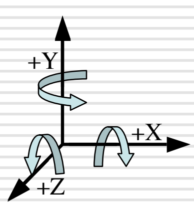
Right-Handed Coordinate System

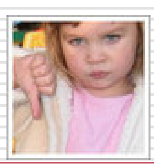


Left-Handed Coordinate System

Right-Handed Coordinates

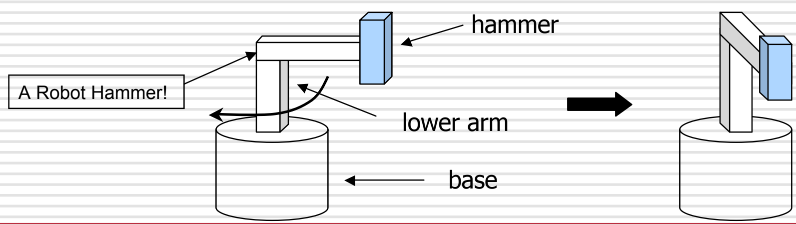
- To determine positive rotations
  - Make a fist with your right hand, and stick thumb up in the air (CCW)





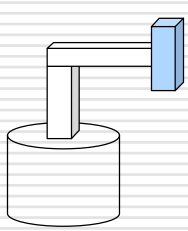
Hierarchical Transformations

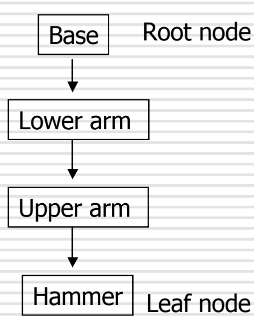
- Graphical scenes have object dependencies
- Many small objects
- Attributes (position, orientation, *etc.*) depend on each other



Hierarchical Transformations (cont.)

- Object dependency description using tree structure





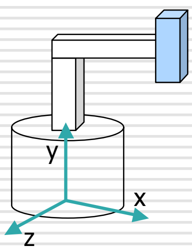
Object position and orientation can be affected by its parent, grand-parent, grand-grand-parent, … nodes

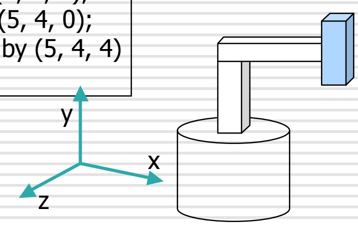
Hierarchical representation is known as **Scene Graph**

Transformations

- Two ways to specify transformations
  1. Absolute transformation: each part of the object is transformed independently relative to the origin

Translate the base by (5, 0, 0);
Translate the lower arm by (5, 2, 0);
Translate the upper arm by (5, 4, 0);
Translate the hammer head by (5, 4, 4)
…

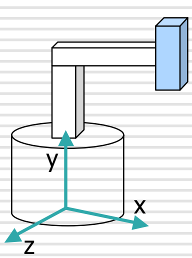


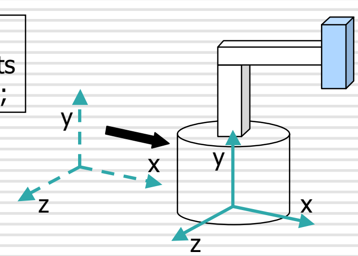


Relative Transformations

- A better (and easier) way
  1. Relative transformation: Specify the transformation for each object relative to its parent

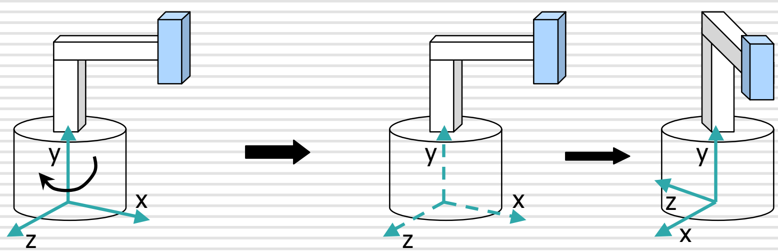
Step 1:
Translate the base (and its descendants) by (5, 0, 0);



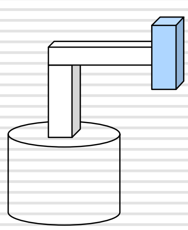


Relative Transformations (cont.)

Step 2:
Rotate the lower arm and (its descendants) relative to the base's local y axis by -90 degrees



Relative Transformations Using a Scene Graph



Base ---- Translate (5, 0, 0)

↓

Lower arm ---- Rotate (-90) about its local y

↓

Upper arm

↓

Hammer

Apply all the way down (Translate (5, 0, 0))

Apply all the way down (Rotate (-90) about its local y)

Introduction to Transformations

- A *transformation* changes an object's
  - Size (scaling)
  - Position (translation)
  - Orientation (rotation)
- We will introduce first in 2D or *(x,y)*, build intuition
- Later, talk about 3D and 4D?
- Transform object by applying sequence of matrix multiplications to object vertices

Why Matrices?

- All transformations can be performed using matrix/vector multiplication
- Allows pre-multiplication of all matrices
- Note: point (x, y) needs to be represented as (x, y, 1), also called ***homogeneous coordinates***

Point Representation

- We use a column matrix (2x1 matrix) to represent a 2D point

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

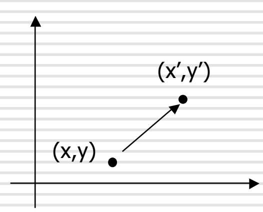
- General form of transformation of a point *(x, y)* to *(x′, y′)* can be written as:

$$x' = ax + by + c \qquad \text{or} \qquad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$
$$y' = dx + ey + f$$

Translation

- To reposition a point along a straight line
- Given point *(x, y)* and translation distance (t_x, t_y)
- The new point: *(x′, y′)*

$$x' = x + t_x$$
$$y' = y + t_y$$



or

$$P' = P + T \quad \text{where} \quad P' = \begin{pmatrix} x' \\ y' \end{pmatrix} \quad P = \begin{pmatrix} x \\ y \end{pmatrix} \quad T = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

3x3 2D Translation Matrix

$x' = x + t_x$
$y' = y + t_y$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

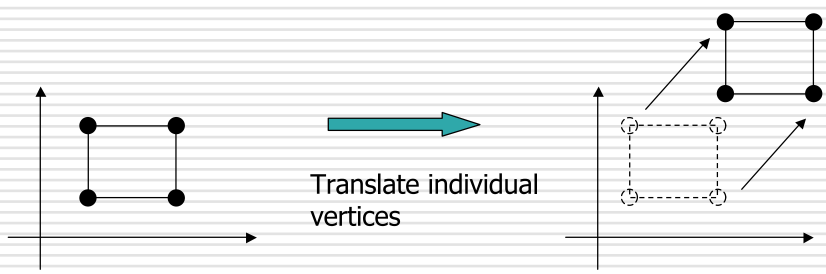
use 3x1 vector

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

Note: it becomes a matrix-vector multiplication

Translation of Objects

- How to translate an object with multiple vertices?

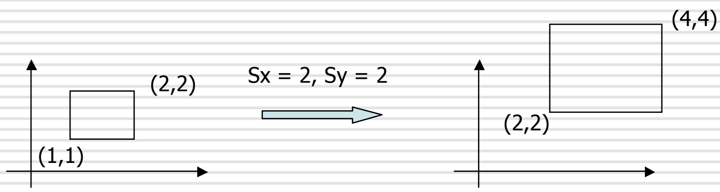


2D Scaling

- Scale: Alter object size by scaling factor (sx, sy). *i.e.*,

$x' = x * Sx$
$y' = y * Sy$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



3x3 2D Scaling Matrix

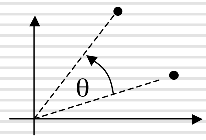
$x' = x * Sx$
$y' = y * Sy$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

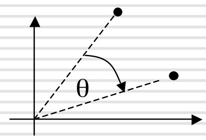
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2D Rotation

- Default rotation center is origin *(0,0)*



θ> 0 : Rotate counter clockwise

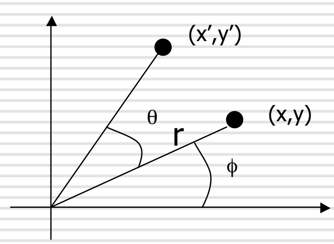


θ< 0 : Rotate clockwise

2D Rotation (cont.)

(x,y) -> Rotate *about the origin* by θ

→ (x', y')



How to compute (x', y') ?

$x = r*cos(\phi)$ $\quad$ $x' = r*cos(\phi + \theta)$

$y = r*sin(\phi)$ $\quad$ $y' = r*sin(\phi + \theta)$

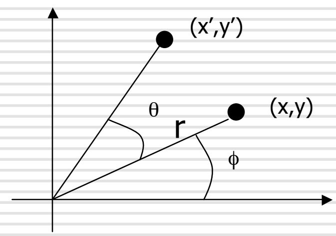
2D Rotation (cont.)

- Using trig. identities

$$\cos(\theta + \phi) = \cos\theta\cos\phi - \sin\theta\sin\phi$$

$$\sin(\theta + \phi) = \sin\theta\cos\phi + \cos\theta\sin\phi$$

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



Matrix form?

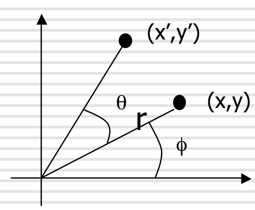
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

3x3 2D Rotation Matrix

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

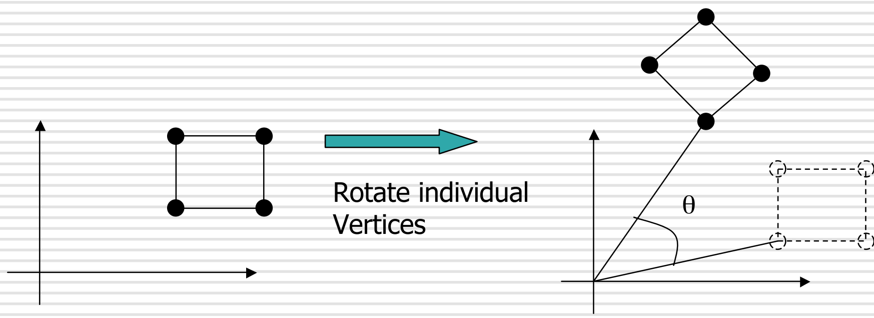
$$\Downarrow$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



2D Rotation

- How to rotate an object with multiple vertices?



Arbitrary Rotation Center

- To rotate about arbitrary point P = (Px, Py) by θ:
  - Translate object by T(-Px, -Py) so that P coincides with origin
  - Rotate the object by R(θ)
  - Translate object back: T(Px, Py)
- In matrix form
  - T(Px,Py) R(θ) T(-Px,-Py) * P

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & Px \\ 0 & 1 & Py \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -Px \\ 0 & 1 & -Py \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Similar for arbitrary scaling anchor

Composing Transformations

- Composing transformations
  - Applying several transforms in succession to form one overall transformation
- Example
  - **M1 X M2 X M3 X P**

  where M1, M2, M3 are transform matrices applied to P
- Be careful with the order!
- For example
  - Translate by (5, 0), then rotate 60 degrees is NOT same as
  - Rotate by 60 degrees, then translate by (5, 0)

3D Transformations

- Affine transformations
  - Mappings of points to new points that retain certain relationships
  - Lines remain lines
  - Several transformations can be combined into a single matrix
- Two ways to think about transformations
  - Object transformations
    - All points of an object are transformed
  - Coordinate transformations
    - The coordinate system is transformed, and models remain defined relative to this