

Tamara Munzner

Transformations II

Week 2, Wed Jan 17

<http://www.ugrad.cs.ubc.ca/~cs314/V/jan2007>

Readings for Jan 15-22

- FCG Chap 6 Transformation Matrices
 - except 6.1.6, 6.3.1
- FCG Sect 13.3 Scene Graphs
- RB Chap Viewing
 - Viewing and Modeling Transforms *until* Viewing Transformations
 - Examples of Composing Several Transformations *through* Building an Articulated Robot Arm
- RB Appendix Homogeneous Coordinates and Transformation Matrices
 - *until* Perspective Projection
- RB Chap Display Lists

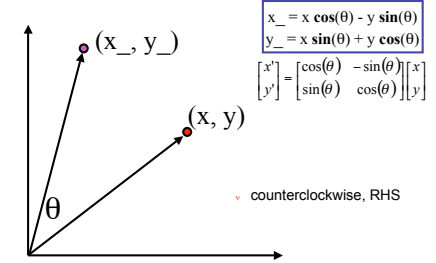
2

Review: Event-Driven Programming

- main loop not under your control
 - vs. procedural
- control flow through event **callbacks**
 - redraw the window now
 - key was pressed
 - mouse moved
- callback functions called from main loop when events occur
 - mouse/keyboard state setting vs. redrawing

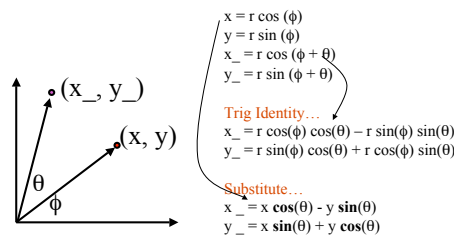
3

Review: 2D Rotation



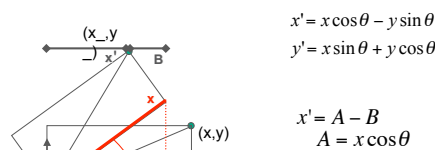
4

Review: 2D Rotation From Trig Identities



5

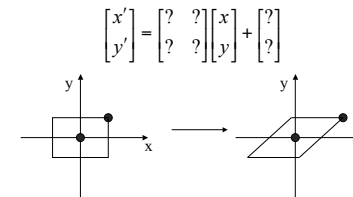
Review: 2D Rotation: Another Derivation



6

Shear

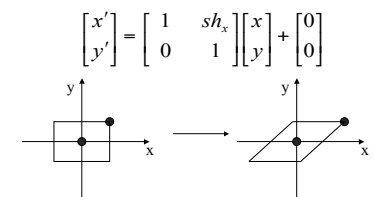
- shear along x axis
 - push points to right in proportion to height



7

Shear

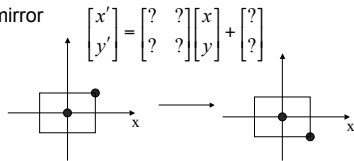
- shear along x axis
 - push points to right in proportion to height



8

Reflection

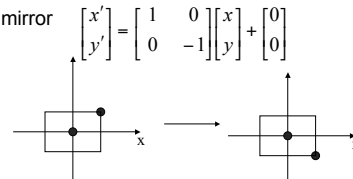
- reflect across x axis
 - mirror



9

Reflection

- reflect across x axis
 - mirror



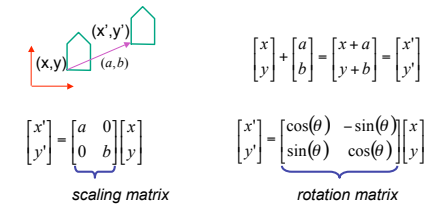
10

2D Translation



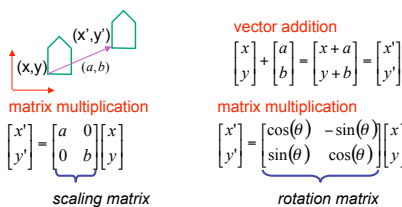
11

2D Translation



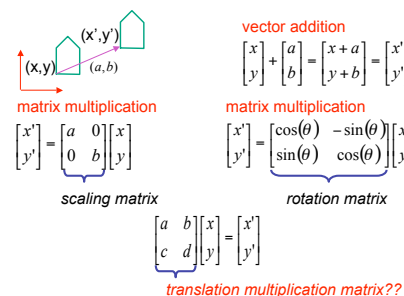
12

2D Translation



13

2D Translation



14

Linear Transformations

- linear transformations are combinations of
 - shear
 - scale $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ $x' = ax + by$
 - rotate $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ $y' = cx + dy$
 - reflect
- properties of linear transformations
 - satisfies $T(sx+ty) = sT(x) + tT(y)$
 - origin maps to origin
 - lines map to lines
 - parallel lines remain parallel
 - ratios are preserved
 - closed under composition

15

Challenge

- matrix multiplication
 - for everything except translation
 - how to do everything with multiplication?
 - then just do composition, no special cases
- homogeneous coordinates trick
 - represent 2D coordinates (x,y) with 3-vector (x,y,1)

16

Homogeneous Coordinates

- our 2D transformation matrices are now 3x3:

$$\text{Rotation} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Scale} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

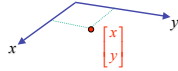
$$\text{Translation} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \quad \cdot \text{ use rightmost column}$$

$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+1+a \\ y+1+b \\ 1 \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \\ 1 \end{bmatrix}$$

17

Homogeneous Coordinates Geometrically

- point in 2D cartesian

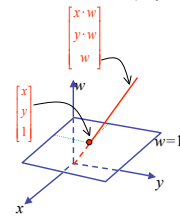


18

Homogeneous Coordinates Geometrically

homogeneous cartesian

$$(x, y, w) \xrightarrow{/w} \left(\frac{x}{w}, \frac{y}{w} \right)$$



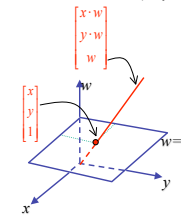
- point in 2D cartesian + weight w = point P in 3D homog. coords
- multiples of (x,y,w)
- form a line L in 3D
- all homogeneous points on L represent same 2D cartesian point
- example: (2,2,1) = (4,4,2) = (1,1,0.5)

19

Homogeneous Coordinates Geometrically

homogeneous cartesian

$$(x, y, w) \xrightarrow{/w} \left(\frac{x}{w}, \frac{y}{w} \right)$$



- homogenize** to convert homog. 3D point to cartesian 2D point:
 - divide by w to get (x/w, y/w, 1)
 - projects line to point onto w=1 plane
 - like normalizing, one dimension up
- when w=0, consider it as direction
 - points at infinity
 - these points cannot be homogenized
 - lies on x-y plane
- (0,0,0) is undefined

20

Affine Transformations

- affine transforms are combinations of

- linear transformations
- translations

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- properties of affine transformations

- origin does not necessarily map to origin
- lines map to lines
- parallel lines remain parallel
- ratios are preserved
- closed under composition

21

Homogeneous Coordinates Summary

- may seem unintuitive, but they make graphics operations much easier
- allow all affine transformations to be expressed through matrix multiplication
 - we'll see even more later...
- use 3x3 matrices for 2D transformations
 - use 4x4 matrices for 3D transformations

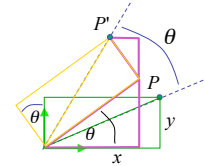
22

3D Rotation About Z Axis

$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta + y \cos \theta$$

$$z' = z$$



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

general OpenGL command
`glRotatef(angle,x,y,z);`
 rotate in z
`glRotatef(angle,0,0,1);`

23

3D Rotation in X, Y

around x axis: `glRotatef(angle,1,0,0);`

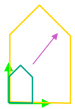
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

around y axis: `glRotatef(angle,0,1,0);`

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

24

3D Scaling

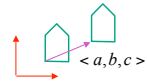


$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

`glScalef(a,b,c);`

25

3D Translation



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

`glTranslatef(a,b,c);`

26

3D Shear

- shear in x

$$xshear(sy,sz) = \begin{bmatrix} 1 & sy & sz & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- shear in y

$$yshear(sx,sz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ sx & 1 & sz & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- shear in z

$$zshear(sx,sy) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sx & sy & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

27

Summary: Transformations

`translate(a,b,c)`

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

`scale(a,b,c)`

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

`Rotate(x,theta)`

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

`Rotate(y,theta)`

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

`Rotate(z,theta)`

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

28

Undoing Transformations: Inverses

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

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

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

$$\mathbf{R}(z,\theta) \mathbf{R}(z,-\theta) = \mathbf{I}$$

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

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

29

Composing Transformations

Composing Transformations

- translation

$$T1 = T(dx1,dy1) = \begin{bmatrix} 1 & dx1 \\ 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad T2 = T(dx2,dy2) = \begin{bmatrix} 1 & dx2 \\ 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$T2 \cdot T1 = \begin{bmatrix} 1 & dx1+dx2 \\ 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{so translations add}$$

30

Composing Transformations

- scaling

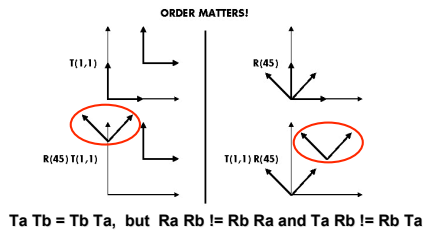
$$S2 \cdot S1 = \begin{bmatrix} sx1 \cdot dx2 & & & \\ & sy1 \cdot dy2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \quad \text{so scales multiply}$$

- rotation

$$R2 \cdot R1 = \begin{bmatrix} \cos(\theta1+\theta2) & -\sin(\theta1+\theta2) & & \\ \sin(\theta1+\theta2) & \cos(\theta1+\theta2) & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \quad \text{so rotations add}$$

31

Composing Transformations



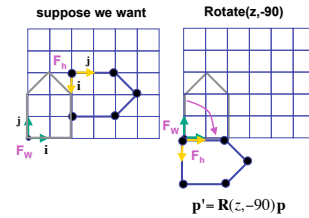
33

Composing Transformations



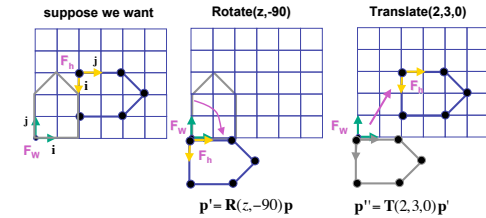
34

Composing Transformations



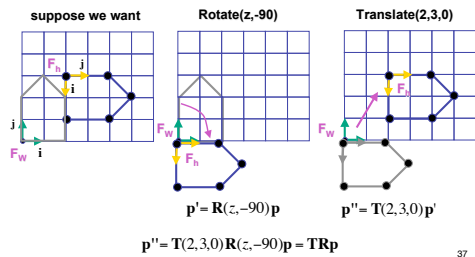
35

Composing Transformations



36

Composing Transformations



37

Composing Transformations

$$p' = TRp$$

- which direction to read?
 - right to left
 - interpret operations wrt fixed coordinates
 - moving object
 - left to right
 - interpret operations wrt local coordinates
 - changing coordinate system

38

Composing Transformations

$$p' = TRp$$

- which direction to read?
 - right to left
 - interpret operations wrt fixed coordinates
 - moving object
 - left to right
 - OpenGL pipeline ordering!
 - interpret operations wrt local coordinates
 - changing coordinate system

39

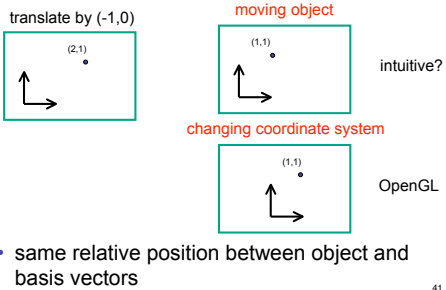
Composing Transformations

$$p' = TRp$$

- which direction to read?
 - right to left
 - interpret operations wrt fixed coordinates
 - moving object
 - left to right
 - OpenGL pipeline ordering!
 - interpret operations wrt local coordinates
 - changing coordinate system
 - OpenGL updates current matrix with postmultiply
 - `glTranslatef(2,3,0);`
 - `glRotatef(-90,0,0,1);`
 - `glVertex(1,1,1);`
 - specify vector last, in final coordinate system
 - first matrix to affect it is specified second-to-last

40

Interpreting Transformations



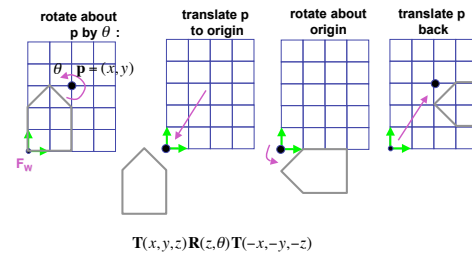
41

Matrix Composition

- matrices are convenient, efficient way to represent series of transformations
 - general purpose representation
 - hardware matrix multiply
 - matrix multiplication is associative
 - $p_- = (T^*(R^*(S^*p)))$
 - $p_- = (T^*R^*S^*)p$
- procedure
 - correctly order your matrices!
 - multiply matrices together
 - result is one matrix, multiply vertices by this matrix
 - all vertices easily transformed with one matrix multiply

42

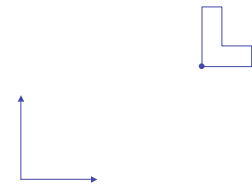
Rotation About a Point: Moving Object



43

Rotation: Changing Coordinate Systems

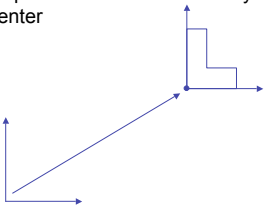
- same example: rotation around arbitrary center



44

Rotation: Changing Coordinate Systems

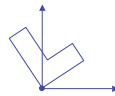
- rotation around arbitrary center
 - step 1: translate coordinate system to rotation center



45

Rotation: Changing Coordinate Systems

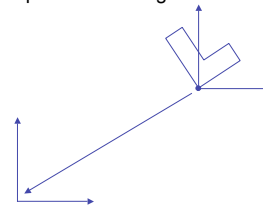
- rotation around arbitrary center
 - step 2: perform rotation



46

Rotation: Changing Coordinate Systems

- rotation around arbitrary center
 - step 3: back to original coordinate system



47

General Transform Composition

- transformation of geometry into coordinate system where operation becomes simpler
 - typically translate to origin
- perform operation
- transform geometry back to original coordinate system

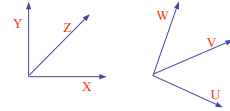
48

Rotation About an Arbitrary Axis

- axis defined by two points
- translate point to the origin
- rotate to align axis with z-axis (or x or y)
- perform rotation
- undo aligning rotations
- undo translation

49

Arbitrary Rotation



- problem:
 - 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 :

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

Arbitrary Rotation

- why?

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

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

51