



Tamara Munzner

Transformations II

Week 2, Fri Jan 14

<http://www.ugrad.cs.ubc.ca/~cs314/Vjan2005>

Reading

- same as last time, through Monday

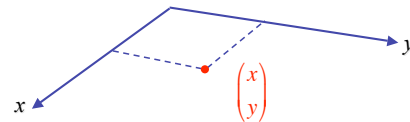
2

News

- corrected slides posted for
 - Week 1 Fri
 - Week 2 Wed
- reminder: extra handouts in CICSR 011
- reminder: my office hours are Wed 3:45-4:45 in CICSR 011

Review: Homogeneous Coordinates

- Cartesian coordinates

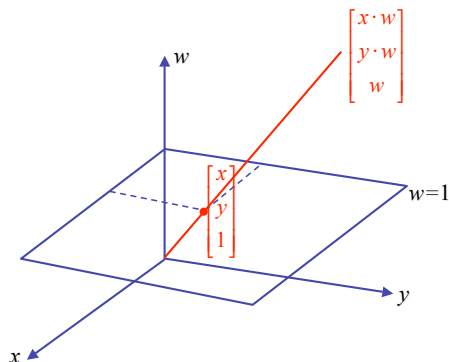


3

4

Review: Homogeneous Coordinates

- Homogeneous coordinates



5

Review: Transformations

translate(a,b,c)

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

scale(a,b,c)

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

Rotate(x,θ)

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

Rotate(y,θ)

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

Rotate(z,θ)

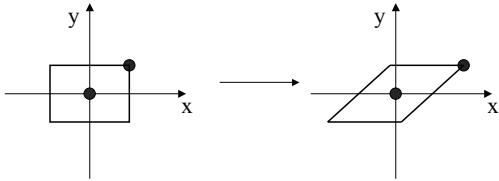
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & & \\ \sin\theta & \cos\theta & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

6

Shear

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} ? \\ ? \end{bmatrix}$$

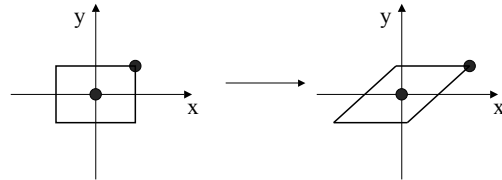


7

Shear

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



- shear in y? 3D shear in x?

8

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

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

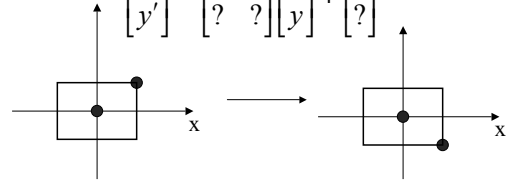
9

Reflection

- reflect across x axis

- mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} ? \\ ? \end{bmatrix}$$



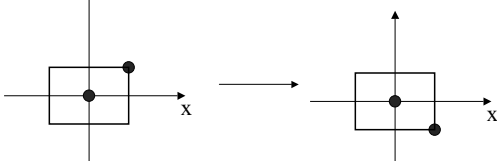
10

Reflection

- reflect across x axis

- mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



- reflect about Y axis?

11

Undoing Transformations: Inverses

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

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

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

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

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

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

12

Composing Transformations

- translation

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

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

$$T2 \cdot T1 = \begin{bmatrix} 1 & & dx_1 + dx_2 \\ & 1 & dy_1 + dy_2 \\ & & 1 \end{bmatrix} \quad \text{so translations add}$$

13

Composing Transformations

- scaling

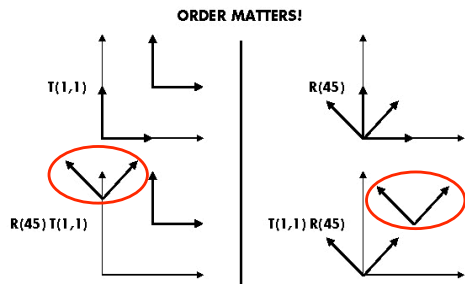
$$S2 \cdot S1 = \begin{bmatrix} sx_1 \cdot dx_2 & & \\ & sy_1 \cdot sy_2 & \\ & & 1 \end{bmatrix} \quad \text{so scales multiply}$$

- rotation

$$R2 \cdot R1 = \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & \\ & & 1 \end{bmatrix} \quad \text{so rotations add}$$

14

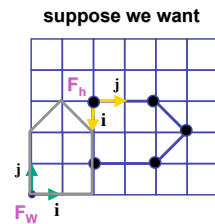
Composing Transformations



$Ta Tb = Tb Ta$, but $Ra Rb \neq Rb Ra$ and $Ta Rb \neq Rb Ta$

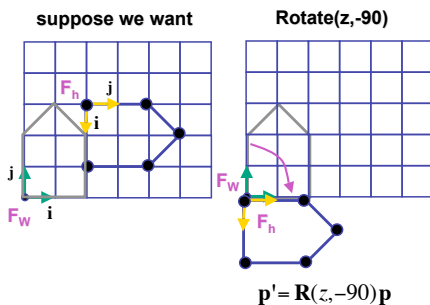
15

Composing Transformations



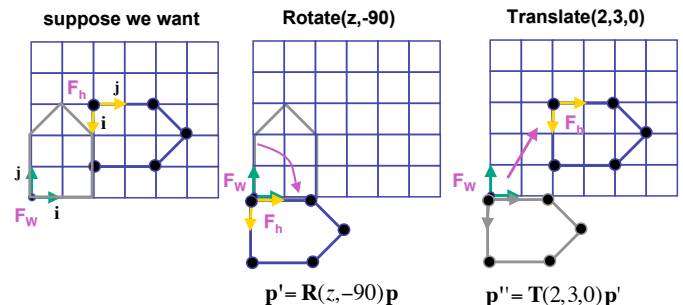
16

Composing Transformations



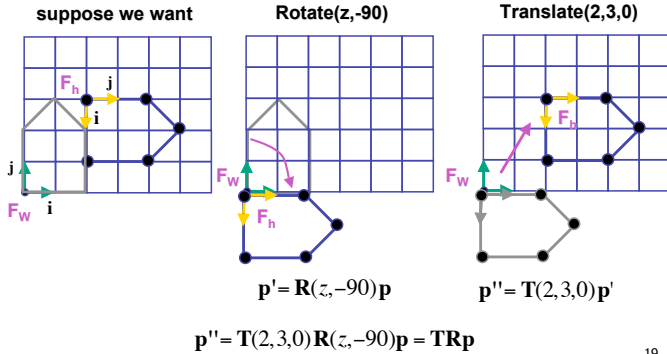
17

Composing Transformations



18

Composing Transformations



19

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

20

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

21

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);`
 - `glVertexf(1,1,1);`
 - specify vector last, in final coordinate system
 - first matrix to affect it is specified second-to-last

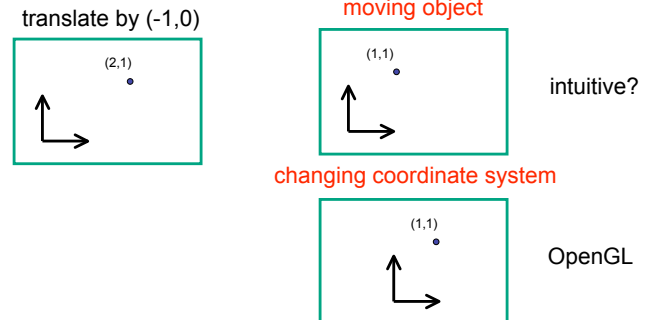
22

Composing Transformations

- correctly ordering your matrices
- multiply matrices together
- result is one matrix
- multiply vertices by this matrix
 - all vertices easily transformed with one matrix multiply

23

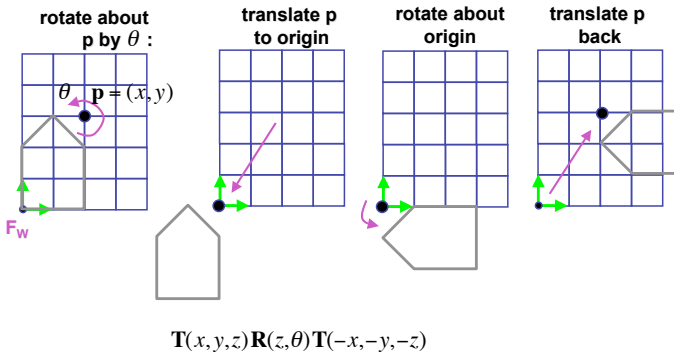
Interpreting Transformations



- same relative position between object and basis vectors

24

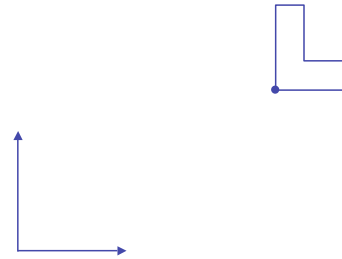
Rotation About a Point: Moving Object



25

Rotation: Changing Coordinate Systems

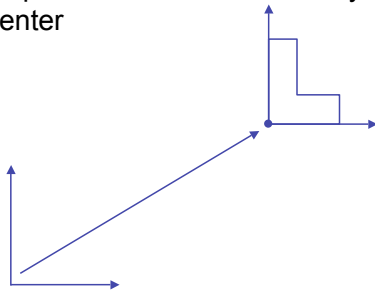
- same example: rotation around arbitrary center



26

Rotation: Changing Coordinate Systems

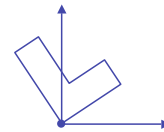
- rotation around arbitrary center
 - step 1: translate coordinate system to rotation center



27

Rotation: Changing Coordinate Systems

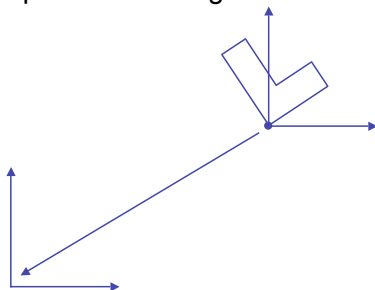
- rotation around arbitrary center
 - step 2: perform rotation



28

Rotation: Changing Coordinate Systems

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



29

General Transform Composition

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

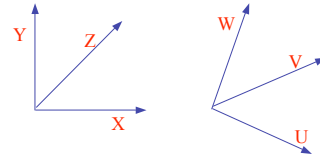
30

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

31

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?

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

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

33

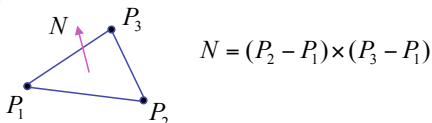
Transforming Geometric Objects

- lines, polygons made up of vertices
- just transform the vertices, interpolate between
- does this work for everything? no!

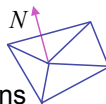
34

Computing Normals

- polygon:



- assume vertices ordered CCW when viewed from visible side of polygon
- normal for a vertex
 - used for lighting
 - supplied by model (i.e., sphere), or computed from neighboring polygons



35

Transforming Normals

- what is a normal?
 - a **direction**
 - homogeneous coordinates: $w=0$ means direction
 - often normalized to unit length
 - vs. points/vectors that are object vertex locations
- what are normals for?
 - specify orientation of polygonal face
 - used when computing lighting
- so if points transformed by matrix M , can we just transform normal vector by M too?

36

Transforming Normals

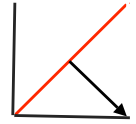
$$\begin{bmatrix} x' \\ y' \\ z' \\ 0 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

- translations OK: $w=0$ means unaffected
- rotations OK
- uniform scaling OK
- these all maintain direction

37

Transforming Normals

- nonuniform scaling does not work
- $x-y=0$ plane
 - line $x=y$
 - normal: $[1,-1,0]$
 - direction of line $x=-y$
 - (ignore normalization for now)

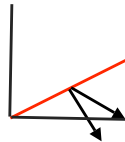


38

Transforming Normals

- apply nonuniform scale: stretch along x by 2
 - new plane $x = 2y$
- transformed normal: $[2,-1,0]$

$$\begin{bmatrix} 2 \\ -1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}$$



- normal is direction of line $x = -2y$ or $x+2y=0$
- not perpendicular to plane!
- should be direction of $2x = -y$

39

Planes and Normals

- plane is all points perpendicular to normal
 - $N \cdot P = 0$ (with dot product)
 - $N^T P = 0$ (matrix multiply requires transpose)

$$N = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, P = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- explicit form: plane = $ax + by + cz + d$

40

Finding Correct Normal Transform

- transform a plane

$$\begin{matrix} P \\ N \end{matrix} \longrightarrow \begin{matrix} P' = MP \\ N' = QN \end{matrix} \quad \begin{matrix} \text{given M,} \\ \text{what should Q be?} \end{matrix}$$

$$N'^T P' = 0 \quad \text{stay perpendicular}$$

$$(QN)^T (MP) = 0 \quad \text{substitute from above}$$

$$N^T Q^T M P = 0 \quad (AB)^T = B^T A^T$$

$$Q^T M = I \quad N^T P = 0 \text{ if } Q^T M = I$$

$$Q = (M^{-1})^T$$

thus the normal to any surface can be transformed by the inverse transpose of the modelling transformation

41