University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2016

Tamara Munzner

Transformations 3

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2016

Readings for Transformations 1-5

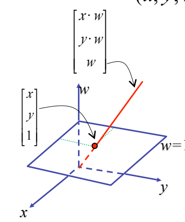
- Shirley/Marschner
  - Ch 6: Transformation Matrices
    - *except* 6.1.6, 6.3.1
  - Sect 12.2 Scene Graphs
- Gortler
  - Ch 2: Linear, Sec 2.5-2.6
  - Ch 3: Affine
  - Ch 4: Respect
  - Ch 5: Frames in Graphics, 5.3-5.4

Homogeneous Coordinates Review

Homogeneous Coordinates Geometrically

homogeneous $\xrightarrow{/w}$ **cartesian**

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



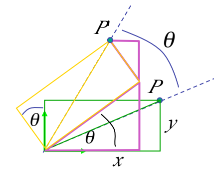
- point in 2D cartesian + weight w = point P in 3D homog. coords
- multiples of (x,y,w)
  - form a line L in 3D
  - all homogeneous points on L represent same 2D cartesian point
  - example: (2,2,1) = (4,4,2) = (1,1,0.5)

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

3D Transformations

3D Rotation About Z Axis



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

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

3D Rotation in X, Y

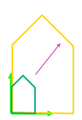
around x axis:

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

around y axis:

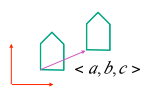
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D Scaling



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

3D Translation



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

3D Shear

- general shear $shear(hxy, hxz, hyx, hyz, hzx, hzy) = \begin{bmatrix} 1 & hyx & hzx & 0 \\ hxy & 1 & hzy & 0 \\ hxz & hyz & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- to avoid ambiguity, always say "shear along <axis> in direction of <axis>"

$$shearAlongXinDirectionOfY(h) = \begin{bmatrix} 1 & h & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad shearAlongXinDirectionOfZ(h) = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$shearAlongYinDirectionOfX(h) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ h & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad shearAlongYinDirectionOfZ(h) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & h & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$shearAlongZinDirectionOfX(h) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ h & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad shearAlongZinDirectionOfY(h) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & h & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Summary: 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)

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

$\text{Rotate}(x, \theta)$

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

$\text{Rotate}(y, \theta)$

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

$\text{Rotate}(z, \theta)$

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

Undoing Transformations: Inverses

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

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

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

Composing Transformations

Composing Transformations

- translation

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

$$P'' = T2 \bullet P' = T2 \bullet [T1 \bullet P] = [T2 \bullet T1] \bullet P, \textit{where}$$

$$T2 \bullet T1 = \begin{bmatrix} 1 & & & dx_1 + dx_2 \\ & 1 & & dy_1 + dy_2 \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

so translations add

Composing Transformations

- scaling

$$S2 \bullet S1 = \begin{bmatrix} sx_1 * dx_2 & & & \\ & sy_1 * sy_2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

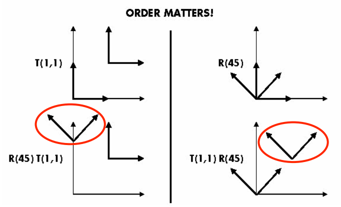
so scales multiply

- rotation

$$R2 \bullet R1 = \begin{bmatrix} \cos(\theta 1 + \theta 2) & -\sin(\theta 1 + \theta 2) & & \\ \sin(\theta 1 + \theta 2) & \cos(\theta 1 + \theta 2) & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

so rotations add

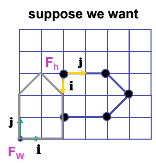
Composing Transformations



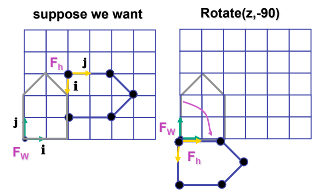
Ta Tb = Tb Ta, but Ra Rb != Rb Ra and Ta Rb != Rb Ta

- translations commute
- rotations around same axis commute
- rotations around different axes do not commute
- rotations and translations do not commute

Composing Transformations

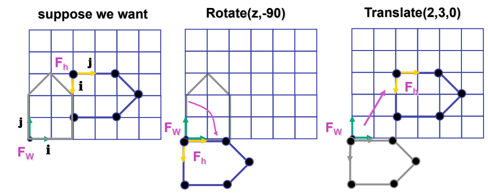


Composing Transformations



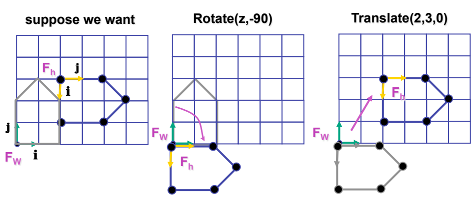
$\mathbf{p'} = \mathbf{R}(z,-90)\,\mathbf{p}$

Composing Transformations



$\mathbf{p'} = \mathbf{R}(z,-90)\,\mathbf{p}$ $\quad$ $\mathbf{p''} = \mathbf{T}(2,3,0)\,\mathbf{p'}$

Composing Transformations



$\mathbf{p'} = \mathbf{R}(z,-90)\,\mathbf{p}$ $\quad$ $\mathbf{p''} = \mathbf{T}(2,3,0)\,\mathbf{p'}$

$\mathbf{p''} = \mathbf{T}(2,3,0)\mathbf{R}(z,-90)\mathbf{p} = \mathbf{TRp}$

Composing Transformations

$\mathbf{p'} = \mathbf{TRp}$

- which direction to read?

Composing Transformations

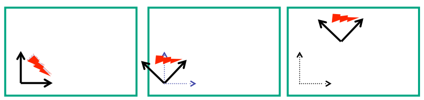
$\mathbf{p'} = \mathbf{TRp}$

- 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
    - in GL, cannot move object once it is drawn!!
      - object specified as set of coordinates wrt specific coord sys

Correction: Composing Transformations

$\mathbf{p'} = \mathbf{TRp}$

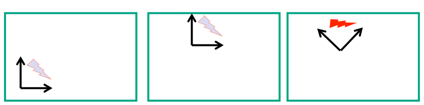
- which direction to read?
  - left to right
    - interpret operations wrt local coordinates
    - moving object
      - draw thing
      - rotate thing by -45 degrees wrt fixed global coords
      - translate it (2, 3) over wrt fixed global coordinates



Correction: Composing Transformations

$\mathbf{p'} = \mathbf{TRp}$

- which direction to read?
  - left to right
    - interpret operations wrt local coordinates
    - changing coordinate system
      - translate coordinate system (2, 3) over
      - rotate coordinate system -45 degrees wrt LOCAL origin
      - draw object in current coordinate system



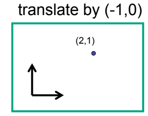
Composing Transformations

$\mathbf{p'} = \mathbf{TRp}$

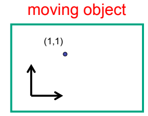
- which direction to read?
  - right to left
    - interpret operations wrt fixed coordinates
    - moving object
  - left to right **GL pipeline ordering!**
    - interpret operations wrt local coordinates
    - changing coordinate system
    - GL updates current matrix with postmultiply
      - translate(2,3,0);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

Interpreting Transformations

translate by (-1,0)

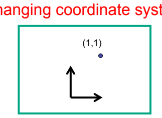


moving object



intuitive?

changing coordinate system



GL

- same relative position between object and basis vectors

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