University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2007

Tamara Munzner

Transformations II

Week 2, Wed Jan 17

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2007

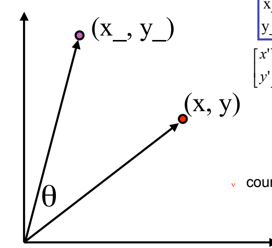
Readings for Jan 15-22

- 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

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

Review: 2D Rotation

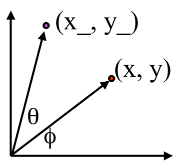


x_ = x **cos**(θ) - y **sin**(θ)
y_ = x **sin**(θ) + y **cos**(θ)

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

- counterclockwise, RHS

Review: 2D Rotation From Trig Identities

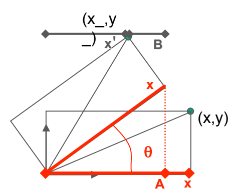


x = r cos (ϕ)
y = r sin (ϕ)
x_ = r cos (ϕ + θ)
y_ = r sin (ϕ + θ)

Trig Identity…
x_ = r cos(ϕ) cos(θ) – r sin(ϕ) sin(θ)
y_ = r sin(ϕ) cos(θ) + r cos(ϕ) sin(θ)

Substitute…
x_ = x **cos**(θ) - y **sin**(θ)
y_ = x **sin**(θ) + y **cos**(θ)

Review: 2D Rotation: Another Derivation



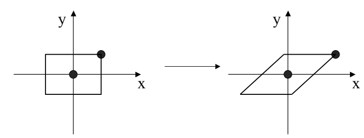
$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

$$x' = A - B$$
$$A = x\cos\theta$$

Shear

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

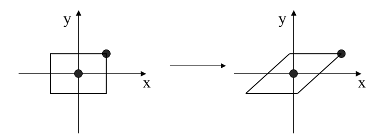
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} ? \\ ? \end{bmatrix}$$



Shear

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

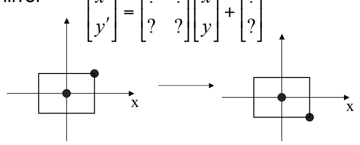
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



Reflection

- reflect across x axis
  - mirror

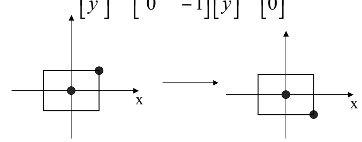
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} ? \\ ? \end{bmatrix}$$



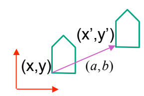
Reflection

- reflect across x axis
  - mirror

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

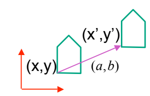


2D Translation



$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

2D Translation



$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

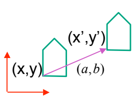
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

scaling matrix

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

rotation matrix

2D Translation



vector addition

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

matrix multiplication

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

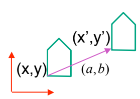
scaling matrix

matrix multiplication

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

rotation matrix

2D Translation



vector addition

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

matrix multiplication

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

scaling matrix

matrix multiplication

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

rotation matrix

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

translation multiplication matrix??

Linear Transformations

- linear transformations are combinations of
  - shear
  - scale
  - rotate
  - reflect

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

$$x' = ax + by$$
$$y' = cx + dy$$

- properties of linear transformations
  - satisifes T(**sx**+**ty**) = *s* T(**x**) + *t* T(**y**)
  - origin maps to origin
  - lines map to lines
  - parallel lines remain parallel
  - ratios are preserved
  - closed under composition

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)

Homogeneous Coordinates

- our 2D transformation matrices are now 3x3:

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

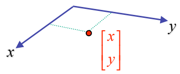
$$\mathbf{Translation} = \begin{bmatrix}1 & 0 & T_x\\ 0 & 1 & T_y\\ 0 & 0 & 1\end{bmatrix}$$

- use rightmost column

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

Homogeneous Coordinates Geometrically

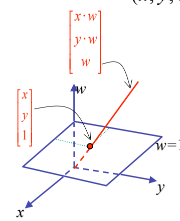
- point in 2D cartesian



Homogeneous Coordinates Geometrically

homogeneous → / w → **cartesian**

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

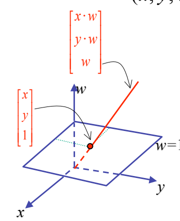


- point in 2D cartesian + weight w = point P in 3D homog. coords
- multiples of (x,y,w)
  - form a line L in 3D
  - all homogeneous points on L represent same 2D cartesian point
  - example: (2,2,1) = (4,4,2) = (1,1,0.5)

Homogeneous Coordinates Geometrically

homogeneous → / w → **cartesian**

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



- 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

Affine Transformations

- affine transforms are combinations of
  - linear transformations
  - translations

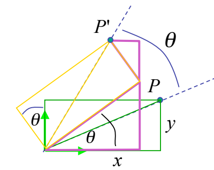
$$\begin{bmatrix}x'\\ y'\\ w\end{bmatrix} = \begin{bmatrix}a & b & c\\ d & e & f\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ w\end{bmatrix}$$

- 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

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 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);

3D Rotation in X, Y

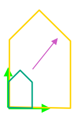
around x axis: **glRotatef(angle,1,0,0);**

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

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

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

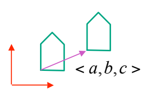
3D Scaling



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

glScalef(a,b,c);

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);

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

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

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)

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

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

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

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

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

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

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

Undoing Transformations: Inverses

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

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

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

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

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

so translations add

Composing Transformations

- scaling

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

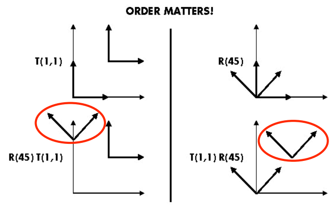
so scales multiply

- rotation

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

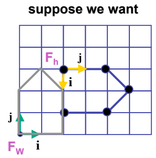
so rotations add

Composing Transformations

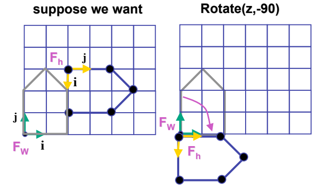


Ta Tb = Tb Ta, but Ra Rb != Rb Ra and Ta Rb != Rb Ta

Composing Transformations

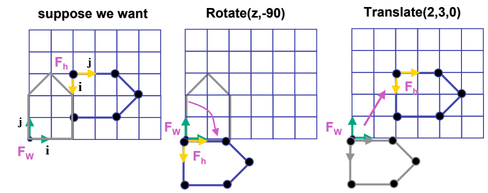


Composing Transformations



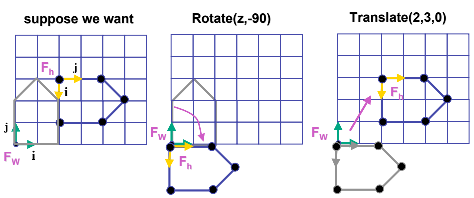
$\mathbf{p'} = \mathbf{R}(z,-90)\,\mathbf{p}$

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

Composing Transformations

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

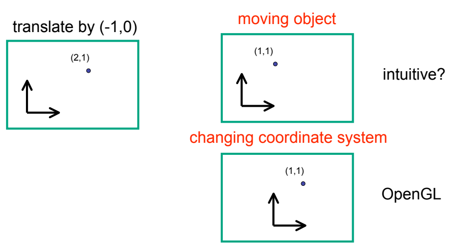
- which direction to read?
  - right to left
    - interpret operations wrt fixed coordinates
    - moving object
  - left to right **OpenGL pipeline ordering!**
    - interpret operations wrt local coordinates
    - changing coordinate system

Composing Transformations

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

- which direction to read?
  - right to left
    - interpret operations wrt fixed coordinates
    - moving object
  - left to right **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

Interpreting Transformations

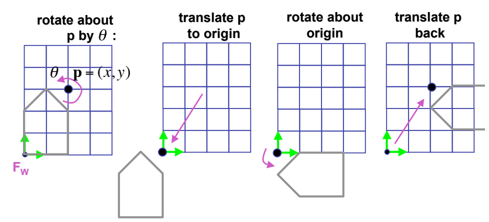


- same relative position between object and basis vectors

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

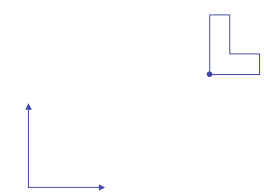
Rotation About a Point: Moving Object



$\mathbf{T}(x,y,z)\mathbf{R}(z,\theta)\mathbf{T}(-x,-y,-z)$

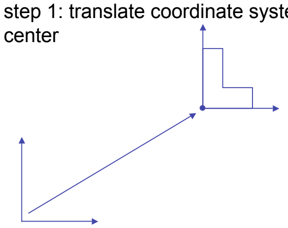
Rotation: Changing Coordinate Systems

- same example: rotation around arbitrary center



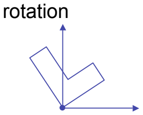
Rotation: Changing Coordinate Systems

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



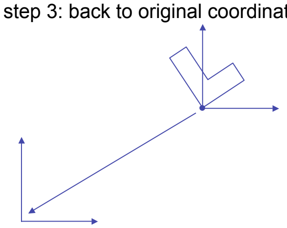
Rotation: Changing Coordinate Systems

- rotation around arbitrary center
  - step 2: perform rotation



Rotation: Changing Coordinate Systems

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



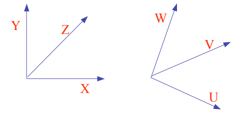
General Transform Composition

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

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

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