University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2005

Tamara Munzner

Transformations

Week 2, Wed Jan 12

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

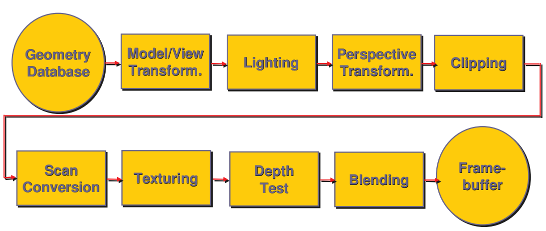
Reading

- FCG Chap 5 (except 5.1.6, 5.3.1),
- FCG pages 224-225
- RB Chap **Viewing**:
  - Sect. Viewing and Modeling Transforms until Viewing Transformations
  - Sect. Examples of Composing Several Transformations through Building an Articulated Robot Arm
- RB Appendix **Homogeneous Coordinates and Transformation Matrices**
  - until Perspective Projection

Textbook Errata

- list at http://www.cs.utah.edu/~shirley/fcg/errata
  - math review: also p 48
    - a x (b x c) != (a x b) x c
  - transforms: p 91
    - should halve x (not y) in Fig 5.10
  - transforms: p 106
    - second line of matrices: [x_p, y_p, 1]

Review: Rendering Pipeline



Review: OpenGL

- pipeline processing, set state as needed

```
void display()
{
  glClearColor(0.0, 0.0, 0.0, 0.0);
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(0.0, 1.0, 0.0);
  glBegin(GL_POLYGON);
    glVertex3f(0.25, 0.25, -0.5);
    glVertex3f(0.75, 0.25, -0.5);
    glVertex3f(0.75, 0.75, -0.5);
    glVertex3f(0.25, 0.75, -0.5);
  glEnd();
  glFlush();
}
```


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

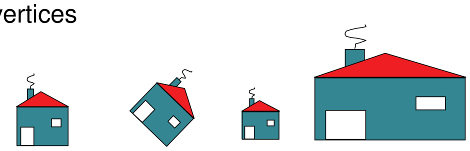
Keyboard/Mouse Callbacks

- do minimal work
- request redraw for display
- example: keypress triggering animation
  - do not create loop in input callback!
    - what if user hits another key during animation?
  - shared/global variables to keep track of state
  - display function acts on current variable value

Transformations

Transformations

- transforming an object = transforming all its points
- transforming a polygon = transforming its vertices



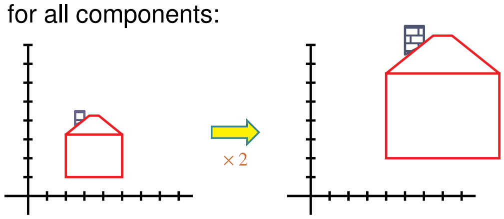
Matrix-Vector Multiplication

$$\mathbf{Mv} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} & \mathbf{M}_{13} \\ \mathbf{M}_{21} & \mathbf{M}_{22} & \mathbf{M}_{23} \\ \mathbf{M}_{31} & \mathbf{M}_{32} & \mathbf{M}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_z \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

- linear transformation of a vector
  - maps one vector to another
  - preserves linear combinations
- any linear transform can be represented by a matrix
  - scaling
  - rotation
  - translation

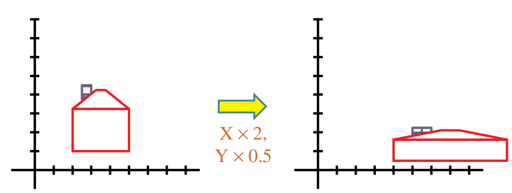
Scaling

- scaling a coordinate means multiplying each of its components by a scalar
- uniform scaling means this scalar is the same for all components:



Scaling

- non-uniform scaling: different scalars per component:



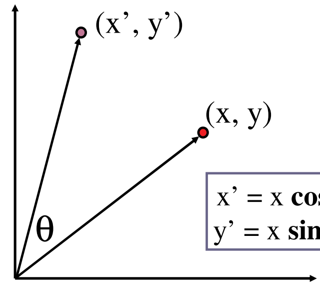
- how can we represent this in matrix form?

Scaling

- scaling operation: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ax \\ by \end{bmatrix}$
- or, in matrix form: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$

13

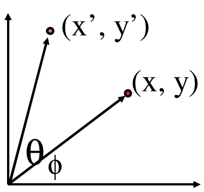
2D Rotation



x' = x **cos**(θ) - y **sin**(θ)
y' = x **sin**(θ) + y **cos**(θ)

14

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)

15

2D Rotation Matrix

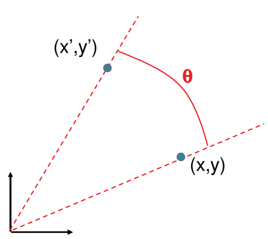
- easy to capture in matrix form:

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

- even though sin(q) and cos(q) are nonlinear functions of q,
  - x' is a linear combination of x and y
  - y' is a linear combination of x and y

16

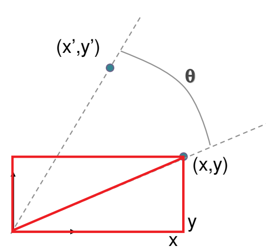
2D Rotation: Another Derivation



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

17

2D Rotation: Another Derivation

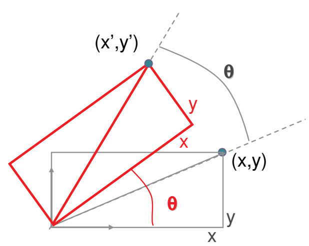


$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

18

Page 3

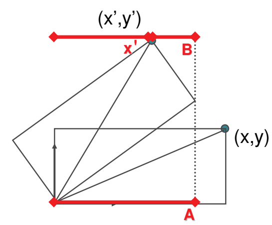
2D Rotation: Another Derivation



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

19

2D Rotation: Another Derivation

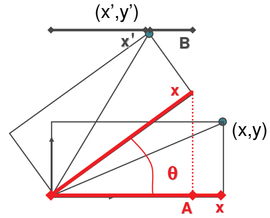


$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

$$x' = A - B$$

20

2D Rotation: Another Derivation

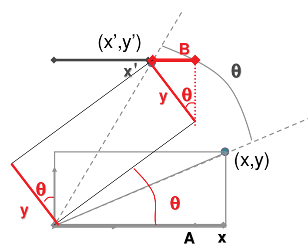


$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

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

21

2D Rotation: Another Derivation

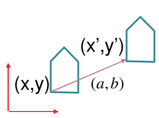


$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

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

22

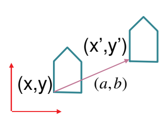
2D Translation



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

23

2D Translation



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

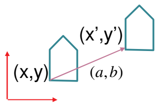
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\text{rotation matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

scaling matrix *rotation matrix*

24

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

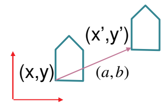
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

matrix multiplication

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\textit{rotation matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

25

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

matrix multiplication

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\textit{rotation matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix}}_{\textit{translation multiplication matrix??}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

translation multiplication matrix??

26

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)

27

Homogeneous Coordinates

- our 2D transformation matrices are now 3x3:

$$\mathbf{R}otation = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{S}cale = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

- use rightmost column

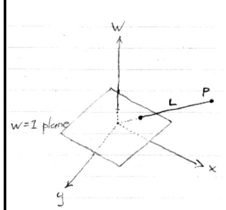
$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x*1+a*1 \\ y*1+b*1 \\ 1 \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \\ 1 \end{bmatrix}$$

28

Homogeneous Coordinates Geometrically

homogeneous **cartesian**

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



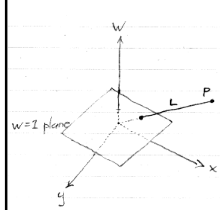
- point in 2D cartesian + weight w = point P in 3D homog. coords
- multiples of (x,y,w)
  - form a line L in 3D
  - all homogeneous points on L represent same 2D cartesian point
  - example: (2,2,1) = (4,4,2) = (1,1,0.5)

29

Homogeneous Coordinates Geometrically

homogeneous **cartesian**

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



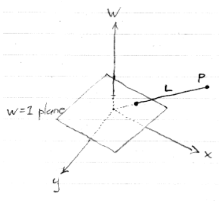
- homogenize to convert homog. 3D point to cartesian 2D point:
  - divide by w to get (x/w, y/w, 1)
  - projects line to point onto w=1 plane
- when w=0, consider it as direction
  - points at infinity
  - these points cannot be homogenized
- (0,0,0) is undefined

30

Homogeneous Coordinates Geometrically

homogeneous → **cartesian**

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

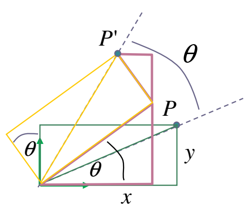


- w=0 denotes points at infinity
  - think of as direction
  - cannot be homogenized
  - lies on x-y plane
- (0,0,0) is not allowed

Homogeneous Coordinates Summary

- may seem unintuitive, but they make graphics operations much easier
- allow all linear transformations to be expressed through matrix multiplication
- 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

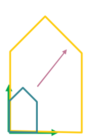
around x axis: **glRotatef(angle,1,0,0);**

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

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

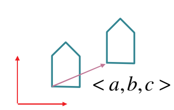


scale(a,b,c)

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

glScalef(a,b,c);

3D Translation



translate(a,b,c)

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

glTranslatef(a,b,c);