559 Course Notes – 2010
Geometric Graphics

Mike Gleicher

October 2007

Notes for lectures, not shown in class

Lead-in

- Surveys, pictures, handin directories – wait
- Practice assignment due Weds 9/15
  - Make a picture
  - Gallery
  - Mechanics
- Pictures from world vs. Pictures on screen
  - Light picture, computer screen/window
  - Toolkit vs. class
- Geometric vs. Image-based

Geometric graphics

- How do we draw shapes (What Shapes?)
- Primitives (simple shapes – build up bigger ones)
  - Points
  - Lines
  - Polygons
  - Curves (later)
  - Surfaces vs. Volumes
- 0d vs. 1d vs 2d vs. space embedded into
- Primitives in world vs. primitives on screen

Triangles as “The” primitive

- Vs. lines/points
- Vs. solids
- Vs. curves/surfaces

- In world vs. screen

- What to know?
  - Position, 3D geometry (normal), Color/draw style

A little practical details - OpenGL

- Need to get a window, etc.
- Drawing context

- State oriented system
- Set “state” (color (RGB aside), style, …)
- Draw in current state
  - Some ways around this

- Many ways to send triangles

Coordinate Systems

- What do positions mean?
  - Need coordinate systems
- Tells us how to interpret positions (coordinates)
- In graphics we deal with many coordinate systems and move between them
  - Use what is convenient for what we're doing

- Examples
  - Chalkboard as coordinate system
  - One panel of chalkboard as coordinate system
  - Monitor as coordinate system

What is a coordinate system

- Position of the zero point
- Directions for each axis
  - Represent points as a linear combination of vectors
  - Vectors (basis) are axes
  - Scale of vectors matter (what is “1 unit”)
  - Directions matter (which way is up)
  - Doesn’t need to be perpindicular (just can’t be parallel)

Describing Coordinate systems

- Need to have some “reference”
  - Where we will measure from
- Give origin, vectors
- Once we have 1 system, can define others

- Can move points by changing their coordinate system
  - Piece of paper is a coordinate system
  - Move piece of paper around
  - If it were a rubber sheet could stretch it as well

Aside on OpenGL

- Normalized Device Coordinates
- Local coordinates
- World Coordinates?

- Detail: projected coordinates, multiple stacks
  - later

Changing Coordinate Systems

- Changing coordinate systems allows us to change large numbers of points all at once

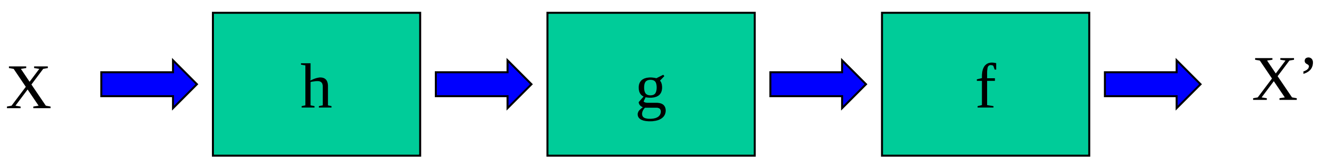
- Need to move points between coordinate systems
  - A coordinate system *transforms* points to a more canonical coordinate system
  - Can define coordinate systems by transformations between coordinate systems

Transformations

- Something that changes points
  - y',y' = f(x,y) $f \in R^2 \rightarrow R^2$
- Coordinate systems are a special case
- Other examples
  - F(x,y) = x+2, y+3
  - F(x,y) = -y, x
  - F(x,y) = x^2, y
- Easy way to effect large numbers of points

Interpreting Transformations

- Can be viewed as a change of coordinates
  - What happens to a piece of graph paper?
  - Just sometimes to a stretchy piece of paper
- View as a function applied to points
- Function composition
  - F(g(h(x))) (note order)



Linear Transformations

- Important special case – linear functions
- Can be written as a matrix $x' = M x$ (x is a vector)

- Good points
  - Many useful transformations are of this form
  - Composition by matrix multiply
  - Easy analysis
  - Straight lines stay straight lines
  - Inverses by inverting the matrix
- Note: linear operators preserve zero!

Example Linear Operators

- Uniform Scale

$$scale(s) = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

- Non-Uniform Scale

$$nuscale(s,t) = \begin{bmatrix} s & 0 \\ 0 & t \end{bmatrix}$$

- Reflect

$$reflect(s,t) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Skew

$$skew(a) = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$$

More linear operators

- Rotate

$$rotate(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

- Linear transformation (non-linear to determine what Linear

- All of this keeps zero
- All linear operations are around the origin (?)

Understanding linear operators

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

- This is POST-Multiply (vector on the right)
  - Pre-multiply convention works too
  - All the matrices get transposed
- What does each element do?
  - Left column – where does X axis go (put in unit X vector)
  - Right column – where does Y axis go
- Can’t do anything about origin!

Post-Multiply vs. Pre-Multiply

- Post multiply – column vector on the left

  $\mathbf{F\ G\ H\ x}$

- Pre-multiply – row vector on the right
  - Older convention, not used as often

  $\mathbf{x}^T\ \mathbf{H}^T\ \mathbf{G}^T\ \mathbf{F}^T$

- I will (almost always) use the post-multiply convention

Affine Transformations

- Translation = move all points the same (vector +)
- Affine = Linear operations plus translation
- Cannot be encoded in a 2x2 matrix (for 2d)
  - Need six numbers for 2d
  - Could be a 3x2 matrix – but then no more multiplies

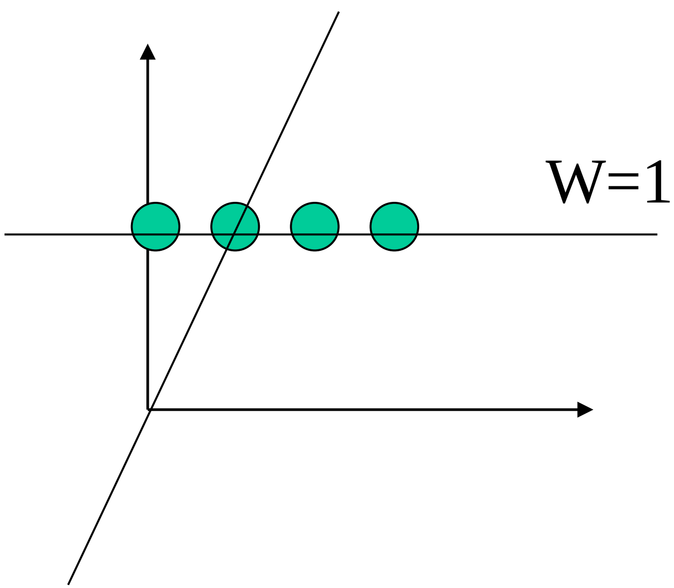
- Rather than treat as a special case, improve our coordinates a bit

Homogeneous Coordinates

- Big idea for graphics – really important
  - Will be used for several things – translation is just 1
- Basic idea: add an extra coordinate
  - 2D becomes 3D (3x3 matrices)
  - 3D becomes 4D (4x4 matrices)
- Convert “back” from homogeneous coordinates by division
  - (x,y) -> (x,y,1)
  - (x,y,w) -> (x/w, y/w)
- Projection
  - Many points in higher dim space = 1 point in lower dim space
- For now, just make w=1

Homogeneous Coordinates

- “Normal” space is a subspace
  - W = 1
- Think about 1D case (so embed into 2D x,w)
- Many equivalent points (projection)

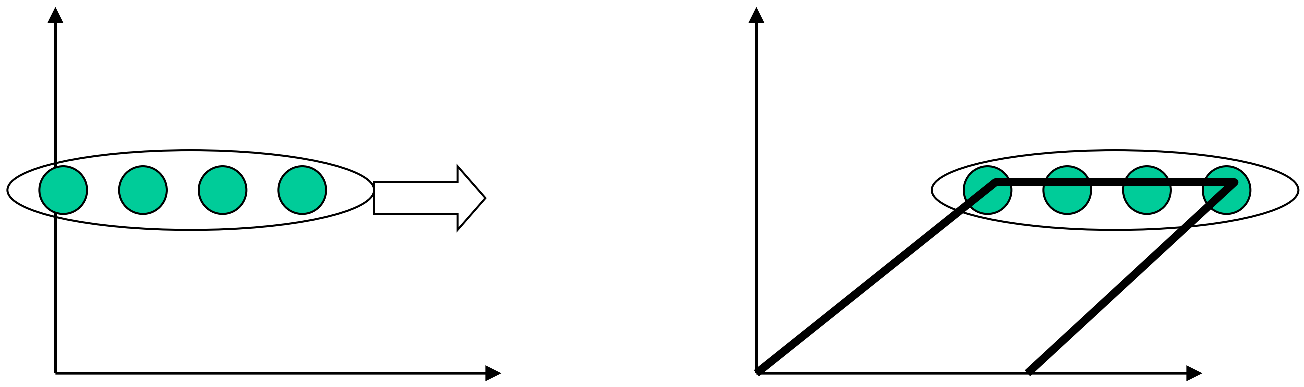


Only 1D Linear operation is scale

(about origin)

Translation in Homogeneous Coords

- Translate in 2D = Skew in 3D
  - Deck of cards



$$trans(x, y) = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

What about other linear ops

- Just add an extra coordinate
- Don’t change w (unless you know what you’re doing)

$$scale(s) = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$rotate(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrices as Coordinate Systems

- Where does X axis go?
- Where does Y axis go?
- Where does origin go?

- Assumes that bottom row is [0 0 1]

- Can you scale by changing w?
  - Yes, but often we prefer to renormalize so bottom right number is 1

Homogeneous Coordinates

- Makes translation (affine transforms) linear
- Need to work in higher dimensional space

- Useful for lots of other things
  - Viewing (perspective)

Matrices as Coordinate Systems

- Where does X axis go?
- Where does Y axis go?
- Where does origin go?

- Assumes that bottom row is [0 0 1]

- Can you scale by changing w?
  - Yes, but often we prefer to renormalize so bottom right number is 1

Composing Transformations

- Order matters!
  - Scale / rotate vs. rotate/scale

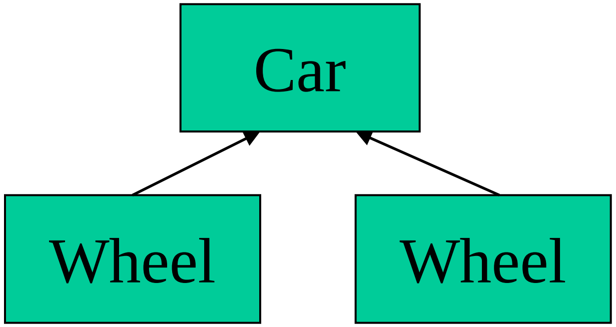
- Can implement by multiplying matrices
  - $T_1\ T_2\ T_3\ \mathbf{x} = (T_1\ T_2\ T_3)\ \mathbf{x}$

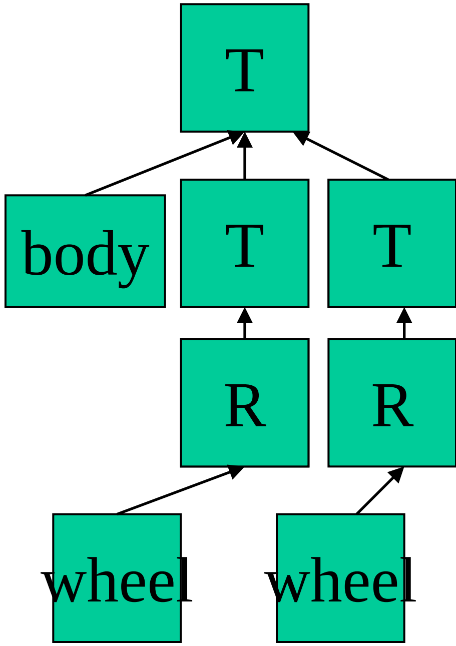
Why Compose?

- Rotate about a point
  - $T_c \; R \; T_{-c} \; x$
- Scale along an axis
  - Move point to origin
  - Align axis w/major axis
  - Scale
  - Put things back
  - $T_c \; R_\theta \; S \; R_{-\theta} \; T_{-c} \; x$

Hierarchical coordinate Systems

- Car
  - Wheel
  - Wheel





- Person
  - Head / Neck
  - Arm / forearm / hand

Matrix Stack

- Multiply things onto the top
- Top is “current” coordinate system
- Push (copy the top) if you’ll come back
- Pop to go back

- Think about it as moving the coordinate system
- Top of stack is “current coordinate system”
  - Where we will draw
- Transformations change current coord system
  - Or change the objects that we are going to draw

Matrix Stack Example

- Draw Car = …. Push trans wheel pop …
- Push trans – draw car – pop push trans – draw car