Lecture 3

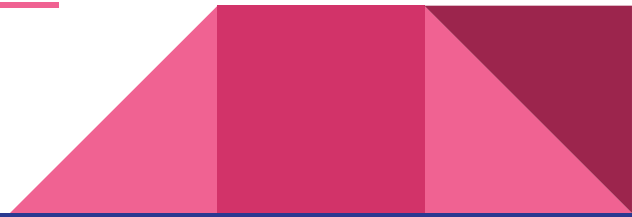
SIGGRAPH trailers from 2016

Going backwards,

https://www.youtube.com/watch?v=I1KO-InHfps

And

https://www.youtube.com/watch?v=dQBJ0r5Pj5s



Today on https://www.dwitter.net/

Today on https://www.shadertoy.com/slideshow

A1 is due - Now Turn On your Debuggers!

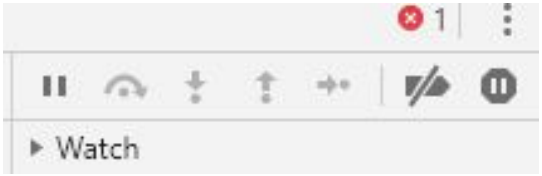
- Let’s activate your debugger for the first time.
  - Breakpoints let you monitor each step of your program and read live variable values.

A1 is due - Now Turn On your Debuggers!

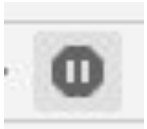
- Let’s activate your debugger for the first time.
  - Each runtime error should stop and bring you to an active breakpoint.
  - Waiting for the console to report the error is too late!!!
    - No longer in the middle of execution

A1 is due - Now Turn On your Debuggers!

- Locate these buttons:



- Press this one until it turns blue:



- Now click the checkbox that appears underneath it:

A1 is due - Now Turn On your Debuggers!

- Expect a big new Piazza post for instructions on using a debugger
  - Once we get some questions about animations breaking
- Let’s take a look at it as applied to A2

Project Proposals and Teams

- Now due at midterm
- You'll turn in another a couple weeks later

Finishing math slides from last week

Next up: Homogeneous Coordinates

Recap

We have reviewed the relevant linear algebra

Matrices

Vectors

Scalars

Next, we will discuss:

Homogeneous representations of points and vectors

Coordinate systems

Transformations

Points vs Vectors

What is the difference?

Points have location, but no size or direction

Vectors have size and direction, but no location

Problem: We represent both as 3-tuples

Homogeneous Representation

Convention: Vectors and Points are represented as 4x1 column matrices, as follows:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$

Switching Representations

Normal to homogeneous:

- Vector: append as fourth coordinate 0

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$$

- Point: append as fourth coordinate 1

$$P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \rightarrow \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$

Switching Representations

Homogeneous to normal:

- Vector: remove fourth coordinate (0)

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

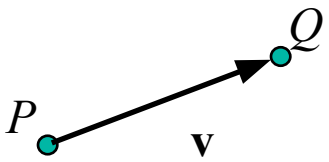
- Point: remove fourth coordinate (1)

$$P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Relationship Between Points and Vectors

A difference between two points is a vector:

$$Q - P = \mathbf{v}$$

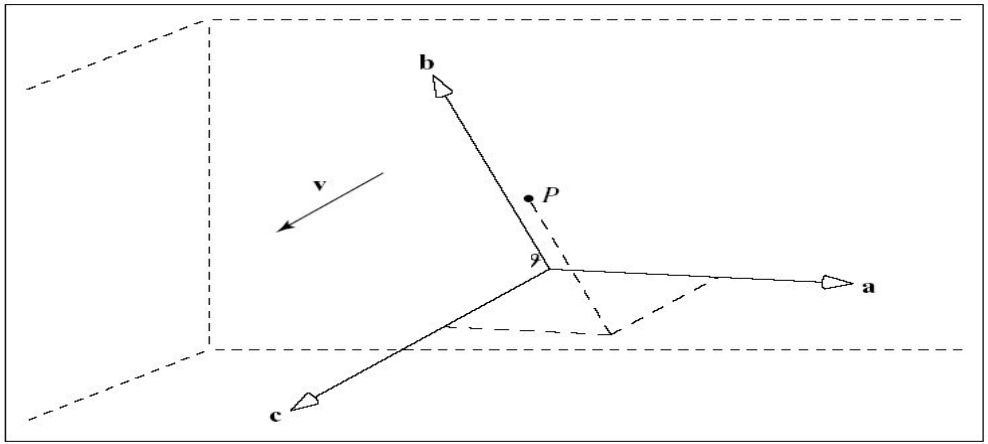


We can consider a point as a base point plus a vector offset:

$$Q = P + \mathbf{v}$$

Coordinate Systems

Defined by: **a**,**b**,**c**,*O*



$$\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c}$$

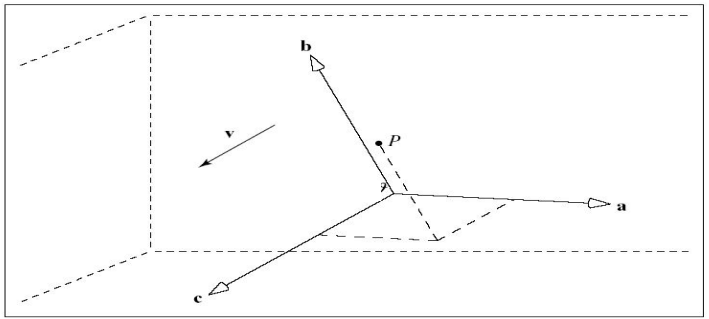
$$P - O = p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$$

$$P = O + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$$

Homogeneous Representation of Points and Vectors

$$\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c} \rightarrow \mathbf{v} = [\mathbf{a}\ \mathbf{b}\ \mathbf{c}\ O] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$$

$$P = O + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c} \rightarrow P = [\mathbf{a}\ \mathbf{b}\ \mathbf{c}\ O] \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$



Does the Homogeneous Representation Support Operations?

Operations :

- $\mathbf{v} + \mathbf{w} = [v_1, v_2, v_3, 0]^T + [w_1, w_2, w_3, 0]^T$
  $= [v_1 + w_1,\ v_2 + w_2,\ v_3 + w_3,\ 0]^T$ Vector
- $a\mathbf{v} = a[v_1, v_2, v_3, 0]^T = [av_1,\ av_2,\ av_3,\ 0]^T$ Vector
- $a\mathbf{v} + b\mathbf{w} = a[v_1, v_2, v_3, 0]^T + b[w_1, w_2, w_3, 0]^T$
  $= [av_1 + bw_1,\ av_2 + bw_2,\ av_3 + bw_3,\ 0]^T$ Vector
- $P + \mathbf{v} = [p_1, p_2, p_3,\ 1]^T + [v_1,\ v_2,\ v_3,\ 0]^T$
  $= [p_1 + v_1,\ p_2 + v_2,\ p_3 + v_3,\ 1]^T$ Point
- $P - Q = [p_1, p_2, p_3, 1]^T - [q_1, q_2, q_3, 1]^T$
  $= [p_1 - q_1,\ p_2 - q_2,\ p_3 - q_3,\ 0]^T$ Vector

Linear Combination of Points

Points P, Q scalars a, b:

$$aP + bQ = a\,[p_1, p_2, p_3, 1]^{\mathrm{T}} + b[q_1, q_2, q_3, 1]^{\mathrm{T}}$$
$$= [ap_1 + bq_1,\ ap_2 + bq_2,\ ap_3 + bq_3,\ a + b]^{\mathrm{T}}$$

What is this?

Linear Combination of Points

Points *P*, *Q* scalars *a*, *b*:

$$aP + bQ = a\,[p_1, p_2, p_3, 1]^{\mathrm{T}} + b[q_1, q_2, q_3, 1]^{\mathrm{T}}$$
$$= [ap_1 + bq_1,\ ap_2 + bq_2,\ ap_3 + bq_3,\ a + b]^{\mathrm{T}}$$

What is it?

- If $(a + b) = 0$ then vector!
- If $(a + b) = 1$ then point!
- Otherwise, ??

Affine Combinations of Points

Definition:

n points P_i: i = 1,…,n

n scalars a_i: i = 1,…,n

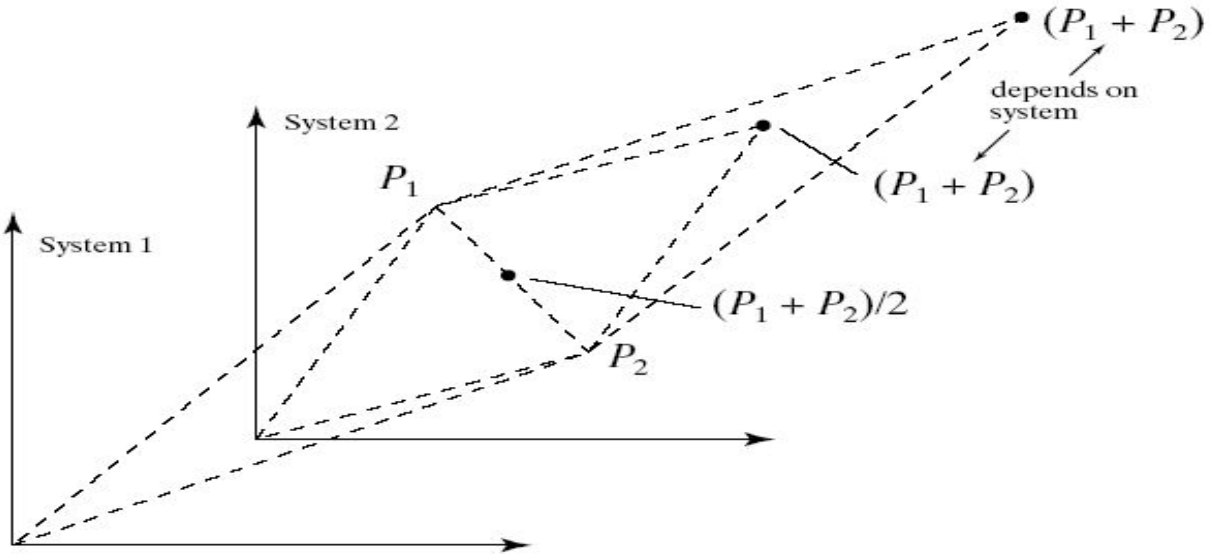
$$a_1P_1 + \ldots + a_nP_n \quad \text{iff} \quad a_1 + \ldots + a_n = 1$$

Example (n = 2): $0.5P_1 + 0.5P_2$

Example (n = 2): $(1\text{-}s)P_1 + sP_2$

Example (n = 3): $(1\text{-}s\text{-}t)P_1 + sP_2 + tP_3$

Geometric Interpretation



Making Shapes in Code

Computer graphics in practice

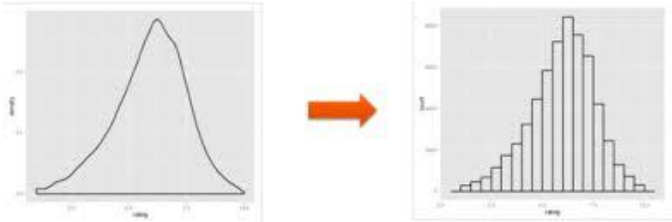
Summary

- Modeling
- Discretizing shapes (Vertices)
- Geometry
  - Data structures
  - Indexing

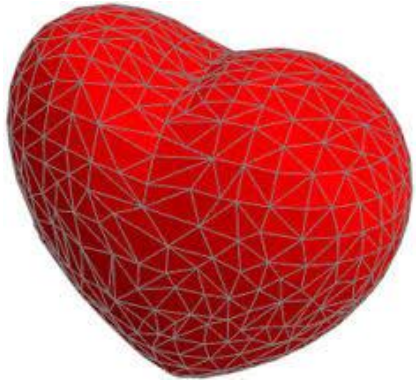
We'll make shapes out of math.

We're mostly trying to draw functions that are not linear or even polynomial.

Discretization



- We don't know how to tell a computer to draw most shapes because of their complicated non-linear formulas.
- Instead, we linearize those shapes: Break them up into a finite number of line segments between N discrete points
- Piecewise planar shapes:

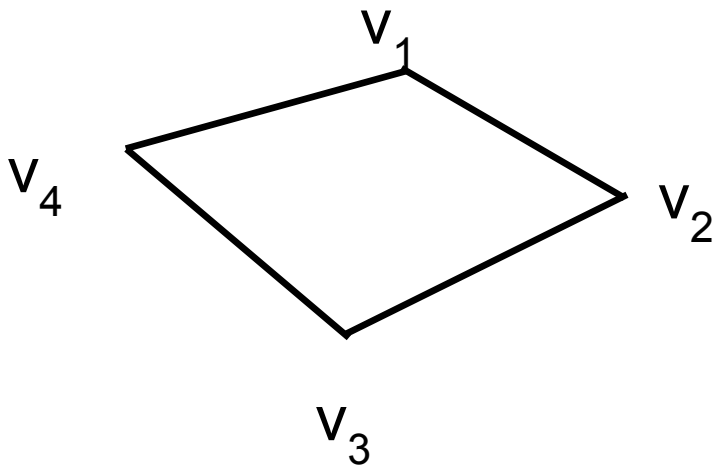


Polygon

Collection of points connected with lines

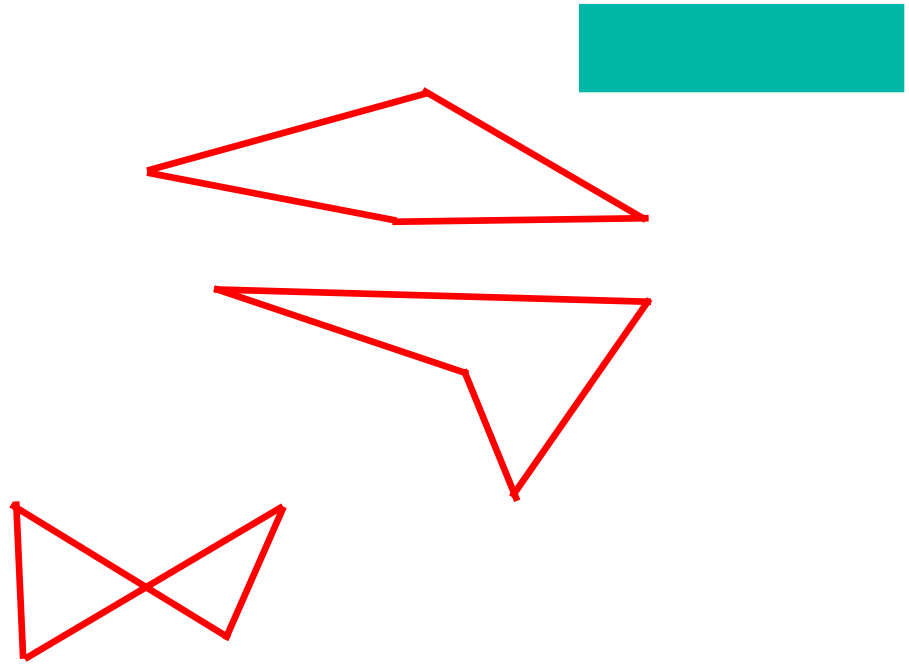
- Vertices: v_1, v_2, v_3, v_4
- Edges:

  $e_1 = v_1v_2$
  $e_2 = v_2v_3$
  $e_3 = v_3v_4$
  $e_4 = v_4v_1$



Polygons

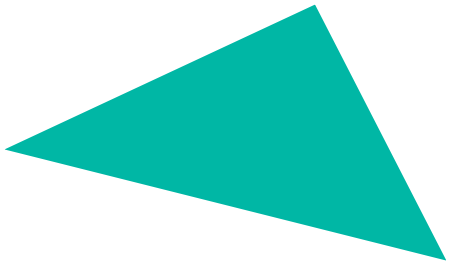
- Open / closed
- Planar / non-planar
- Filled / wireframe
- Convex / concave
- Simple / non-simple



Triangles

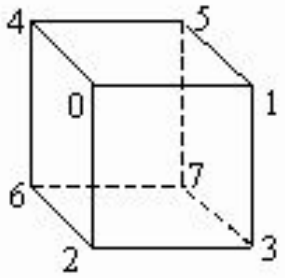
The most common primitive

- Simple
- Convex
- Planar



Polygonal Models / Data Structures

Indexed face set

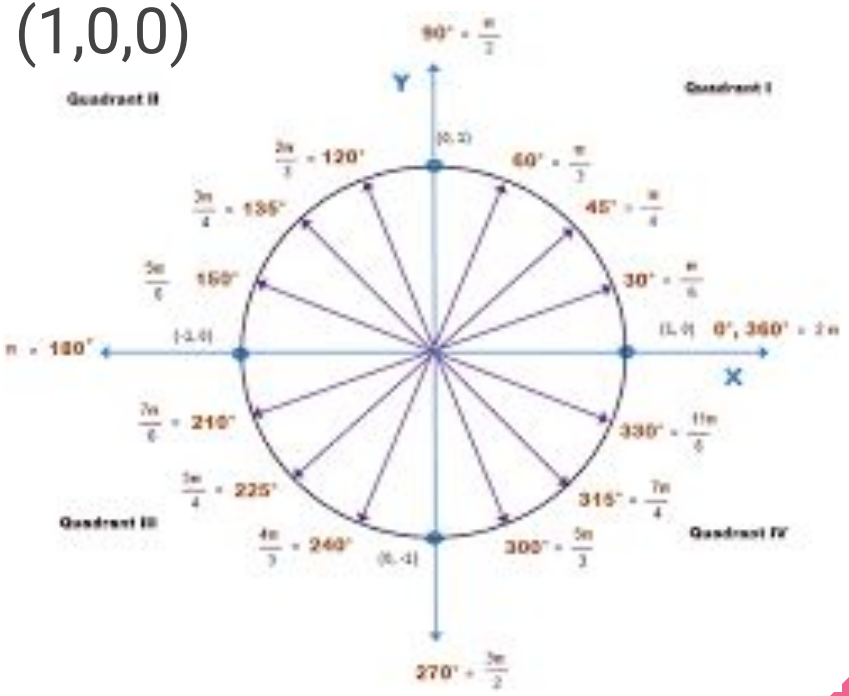


faces		vertex list	
#	vertex list	#	x,y,z
0	0,2,3,1	0	0,1,1
1	1,3,7,5	1	1,1,1
2	5,7,6,4	2	0,0,1
3	4,6,2,0	3	1,0,1
4	4,0,1,5	4	0,1,0
5	2,6,7,3	5	1,1,0
		6	0,0,0
		7	1,0,0

Guerrilla CG Tutorial 01:
The Polygon

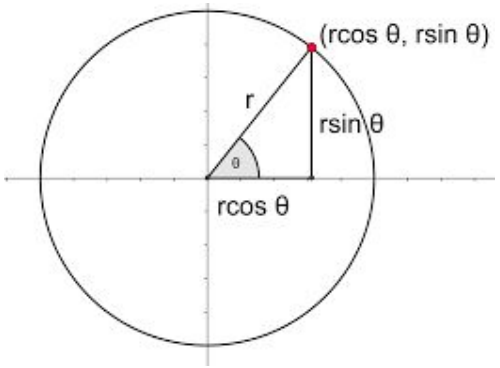
Let's list N points around a circle.

- First point: (1,0,0)



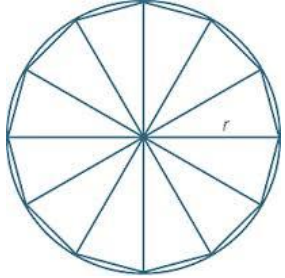
Let's list N points around a circle.

x = r*cos(Θ), y = r*sin(Θ) where theta is as shown below.



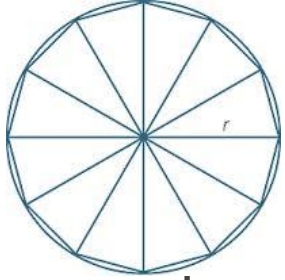
Using Θ as a variable input parameter, take N tiny steps from 0...2*PI.

Triangles



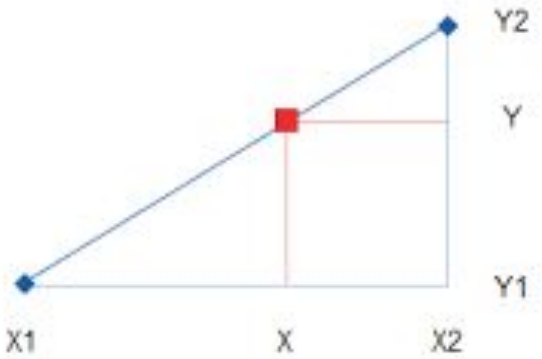
- We want to draw the whole 2D area, not just some points
    - Simplest 2d shape (remove any points and it will make it 1d) - this makes triangles the "2D simplex"

Triangles



- List the points in triangle order - two approaches:
  - Sort list into triples of points
    - (0,0), (1,0),(0.479, 0.878),
      (0,0), (0.479, 0.878), (0.841,0.540)...
    - Repeats are evident here
  - Or, make a separate list of sorted triples of indices
    - Indices are shorter to write, so more triangles can fit in a CPU cache:
    - 0,1,2,0,2,3,0,3,4,0,4,5,0,5,6,0,6,7...

Another exercise:



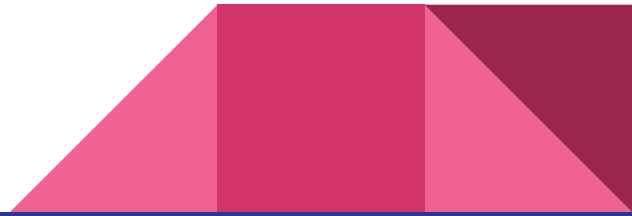
- List some points along a line from one point to another - This process is called convex interpolation

Linear interpolation

- The formula to do that is quite short:

$$p_{interpolated} = (1-a) * p_1 + a * p_2$$

  - It’s only an interpolation (and called “convex”) if 0<=a<=1
  - Otherwise it's an extrapolation
- You’ll be seeing that equation a lot



Linear interpolation

- The formula to do that is quite short:

$$p_{interpolated} = (1-a) * p_1 + a * p_2$$

- Let (a) vary from 0 to 1 in steps - this is a parametric equation.
- Or we could imagine a parameter time (t) rather than (a) -- at each time t between 0 sec and 1 sec we reach a different point on the line segment. Now it’s animated.