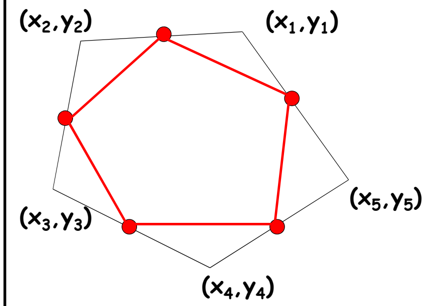
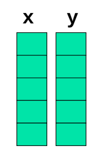
- Previous Lecture:
  - Vectors
  - Simulation
- Today's Lecture:
  - Finite vs. Infinite; Discrete vs. Continuous
  - Linear Interpolation
  - Vectorized code
- Announcements:
  - Discussion this week in UP B7 computer lab
  - Prelim 1: 10/4 (R) 7:30-9pm
  - Review sessions: T5-6:30pm Hollister B14; W7-8:30pm Upson B17. They're *optional.*

Example: polygon smoothing

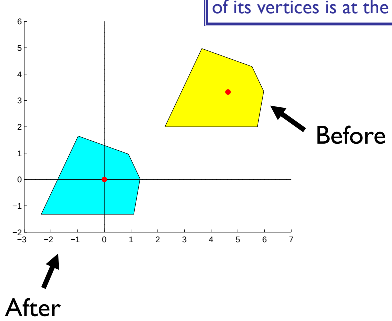


Can store the x-y coordinates in vectors x and y



First operation: centralize

Move a polygon so that the centroid of its vertices is at the origin

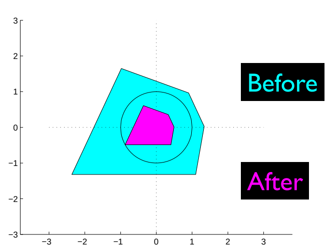



```
function [xNew,yNew] = Centralize(x,y)
% Translate polygon defined by vectors
% x,y such that the centroid is on the
% origin. New polygon defined by vectors
% xNew,yNew.

n = length(x);
xBar = sum(x)/n;    yBar = sum(y)/n;
xNew = zeros(n,1); yNew = zeros(n,1);
for k = 1:n
    xNew(k) = x(k)-xBar;
    yNew(k) = y(k)-yBar;
end
```

sum returns the sum of all values in the vector

Second operation: normalize

Shrink (enlarge) the polygon so that the vertex furthest from the (0,0) is on the unit circle

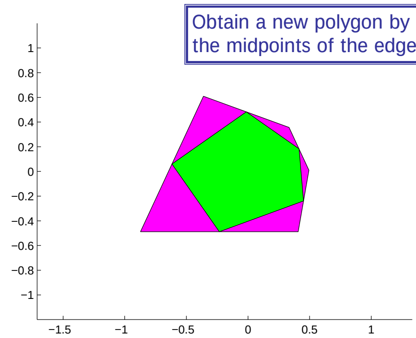



```
function [xNew,yNew] = Normalize(x,y)
% Resize polygon defined by vectors x,y
% such that distance of the vertex
% furthest from origin is 1

n = length(x);
for k = 1:n
   d(k) = sqrt(x(k)^2 + y(k)^2);
end
maxD = max(d);
xNew = zeros(n,1);  yNew = zeros(n,1);
for k = 1:n
   xNew(k)=x(k)/maxD; yNew(k)=y(k)/maxD;
end
```

Applied to a vector, max returns the largest value in the vector

Third operation: smooth

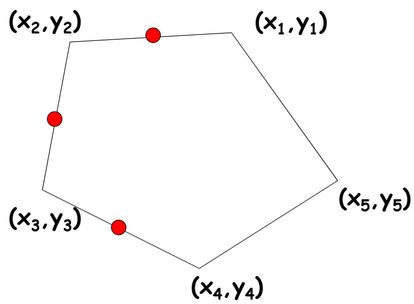
Obtain a new polygon by connecting the midpoints of the edges




```
function [xNew,yNew] = Smooth(x,y)
% Smooth polygon defined by vectors x,y
% by connecting the midpoints of
% adjacent edges

n = length(x);
xNew = zeros(n,1);
yNew = zeros(n,1);

for i=1:n
   Compute the midpt of ith edge.
   Store in xNew(i) and yNew(i)
end
```



```
xNew(3) = (x(3)+x(4))/2
yNew(3) = (y(3)+y(4))/2
```



Smooth

```
for i=1:n
   xNew(i) = (x(i) + x(i+1))/2;
   yNew(i) = (y(i) + y(i+1))/2;
end
```

Will result in a subscript out of bounds error when i is n.

Smooth

```
for i=1:n
   if i<n
     xNew(i) = (x(i) + x(i+1))/2;
     yNew(i) = (y(i) + y(i+1))/2;
   else
     xNew(n) = (x(n) + x(1))/2;
     yNew(n) = (y(n) + y(1))/2;
   end
end
```


Smooth

```
for i=1:n-1
     xNew(i) = (x(i) + x(i+1))/2;
     yNew(i) = (y(i) + y(i+1))/2;
end
xNew(n) = (x(n) + x(1))/2;
yNew(n) = (y(n) + y(1))/2;
```

Show a simulation of polygon smoothing

Create a polygon with randomly located vertices.

Repeat:
- Centralize
- Normalize
- Smooth

See `ShowSmooth.m`

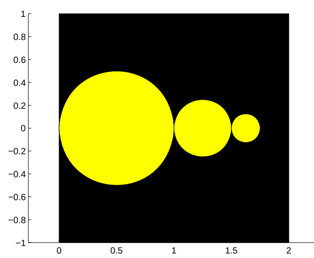
End of Material for Prelim 1

Xeno's Paradox

- A wall is two feet away
- Take steps that repeatedly halve the remaining distance
- You never reach the wall because the distance traveled after n steps =

$$1 + \frac{1}{2} + \frac{1}{4} + \ldots + 1/2^n = 2 - 1/2^n$$

Example: "Xeno" disks

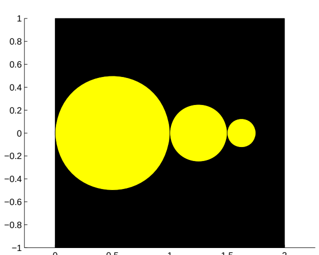


Draw a sequence of 20 disks where the (k+1)th disk has a diameter that is half that of the kth disk.

The disks are tangent to each other and have centers on the x-axis.

First disk has diameter 1 and center (1/2, 0).

Example: "Xeno" disks



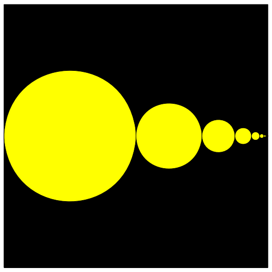
What do you need to keep track of?

- Diameter (d)
- Position
  Left tangent point (x)

Disk	x	d
1	0	1
2	0+1	1/2
3	0+1+1/2	1/4


```
% Xeno Disks

DrawRect(0,-1,2,2,'k')
% Draw 20 Xeno disks
d= 1;
x= 0;  % Left tangent point
for k= 1:20
   % Draw kth disk

   % Update x, d for next disk



end
```

Here's the output… Shouldn't there be 20 disks?

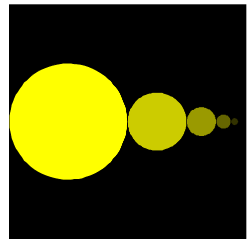


The "screen" is an array of dots called pixels.

Disks smaller than the dots don't show up.

The 20th disk has radius<.000001

Fading Xeno disks

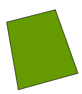


- First disk is yellow
- Last disk is black (invisible)
- *Interpolate* the color in between

Color is a 3-vector, sometimes called the RGB values

- Any color is a mix of red, green, and blue
- Example:

```
colr= [0.4  0.6  0]
```



- Each component is a real value in [0,1]
- [0 0 0] is black
- [1 1 1] is white

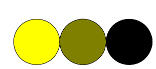

```
% Draw n Xeno disks
d= 1;
x= 0;  % Left tangent point


for k= 1:n



   % Draw kth disk
   DrawDisk(x+d/2, 0, d/2, [1 1 0])
   x= x+d;
   d= d/2;
end
```

A vector of length 3


```
% Draw n fading Xeno disks
d= 1;
x= 0;  % Left tangent point
yellow= [1 1 0];
black=  [0 0 0];
for k= 1:n
   % Compute color of kth disk


   % Draw kth disk
   DrawDisk(x+d/2, 0, d/2, _______)
   x= x+d;
   d= d/2;
end
```


Example: 3 disks fading from yellow to black

```
r= 1;  % radius of  disk
yellow= [1 1 0];
black = [0 0 0];

% Left disk yellow, at x=1
DrawDisk(1,0,r,yellow)
% Right disk black, at x=5
DrawDisk(5,0,r,black)

% Middle disk with average color, at x=3
colr= 0.5*yellow + 0.5*black;
DrawDisk(3,0,r,colr)
```



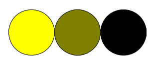
Example: 3 disks fading from yellow to black

```
r= 1;  % radius of  disk
yellow= [1 1 0];
black = [0 0 0];

% Left disk yellow, at x=1
DrawDisk(1,0,r,yellow)
% Right disk black, at x=5
DrawDisk(5,0,r,black)

% Middle disk with average color, at x=3
colr= 0.5*yellow + 0.5*black;
DrawDisk(3,0,r,colr)
```

.5 * | 1 | 1 | 0 | → | .5 | .5 | 0 |

.5 * | 0 | 0 | 0 | → | 0 | 0 | 0 |

Vectorized multiplication



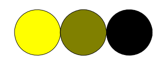
Lecture 12 50

Example: 3 disks fading from yellow to black

```
r= 1;  % radius of  disk
yellow= [1 1 0];
black = [0 0 0];

% Left disk yellow, at x=1
DrawDisk(1,0,r,yellow)
% Right disk black, at x=5
DrawDisk(5,0,r,black)

% Middle disk with average color, at x=3
colr= 0.5*yellow + 0.5*black;
DrawDisk(3,0,r,colr)
```

Vectorized addition

| .5 | .5 | 0 |
+ | 0 | 0 | 0 |
= | .5 | .5 | 0 |



Lecture 12 51

Vectorized code allows an operation on multiple values at the same time

Vectorized addition

| .5 | .5 | 0 |
+ | 0 | 0 | 0 |
= | .5 | .5 | 0 |

```
yellow= [1 1 0];
black = [0 0 0];

% Average color via vectorized op
colr= 0.5*yellow + 0.5*black;
```

Operation performed on vectors

```
% Average color via scalar op
for k = 1:length(black)
   colr(k)= 0.5*yellow(k) + 0.5*black(k);
end
```

Operation performed on scalars

Lecture 12 52

```
% Draw n fading Xeno disks
d= 1;
x= 0;  % Left tangent point
yellow= [1 1 0];
black=  [0 0 0];
for k= 1:n
   % Compute color of kth disk
   f= ???
   colr= f*black + (1-f)*yellow;
   % Draw kth disk
   DrawDisk(x+d/2, 0, d/2, colr)
   x= x+d;
   d= d/2;
end
```

Lecture 12 54

Linear interpolation

x	g(x)
:	:
9	110
10	118
10.25	?
10.50	?
10.75	?
11	126
12	134
:	:

$g(10.5) = \frac{1}{2}\,g(11) + \frac{1}{2}\,g(10)$

$g(10\quad) = 0/4\cdot g(11) + 4/4\cdot g(10)$
$g(10.25) = 1/4\cdot g(11) + 3/4\cdot g(10)$
$g(10.50) = 2/4\cdot g(11) + 2/4\cdot g(10)$
$g(10.75) = 3/4\cdot g(11) + 1/4\cdot g(10)$
$g(11\quad) = 4/4\cdot g(11) + 0/4\cdot g(10)$

$f \cdot g(11) + (1-f) \cdot g(10)$

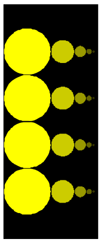
Lecture 12 58

```
% Draw n fading Xeno disks
d= 1;
x= 0;  % Left tangent point
yellow= [1 1 0];
black=  [0 0 0];
for k= 1:n
   % Compute color of kth disk
   f= ???
   colr= f*black + (1-f)*yellow;
   % Draw kth disk
   DrawDisk(x+d/2, 0, d/2, colr)
   x= x+d;
   d= d/2;
end
```

- A k/n
- B k/(n-1)
- C (k-1)/n
- D (k-1)/(n-1)
- E (k-1)/(n+1)

Lecture 12 59

Rows of Xeno disks



```
for y = __:__:__

    Code to draw one
    row of Xeno disks
    at some y-coordinate

end
```

Be careful with "initializations"

Where to put the loop header `for y=__:__:__`

```
A ⇨
     yellow=[1 1 0];  black=[0 0 0];
B ⇨
     d= 1;
C ⇨
     x= 0;
D ⇨
     for k= 1:n
E ⇨     % Compute color of kth disk
        f= (k-1)/(n-1);
        colr= f*black + (1-f)*yellow;
        % Draw kth disk
        DrawDisk(x+d/2, 0, d/2, colr)
        x=x+d;  d=d/2;
     end
end
```


Does this script print anything?

```
k = 0;
while 1 + 1/2^k > 1
   k = k+1;
end
disp(k)
```


Computer Arithmetic—floating point arithmetic

Suppose you have a calculator with a window like this:

+	2	4	1	-	3

representing 2.41×10^{-3}

Floating point addition

+	2	4	1	-	3
+	1	0	0	-	6

Result:

+	2	4	1	-	3

Not enough room to represent .002411

The loop DOES terminate given the limitations of floating point arithmetic!

```
k = 0;
while 1 + 1/2^k > 1
   k = k+1;
end
disp(k)
```

1 + 1/2^53 is calculated to be just 1, so "53" is printed.

Patriot missile failure

www.namsa.nato.int/gallery/systems

In 1991, a Patriot Missile failed, resulting in 28 deaths and about 100 injured. The cause?

0.1

Inexact representation of time/number

- System clock represented time in tenths of a second: a clock tick every 1/10 of a second
- Time = number of clock ticks x 0.1

"exact" value

.00011001100110011001100110011…

.00011001100110011001100110011 value in Patriot system

Error of .000000095 every clock tick

Resulting error

… after 100 hours

.000000095 x (100x60x60)

0.34 second

At a velocity of 1700 m/s, missed target by more than 500 meters!

Computer arithmetic is *inexact*

- There is error in computer arithmetic—floating point arithmetic—due to limitation in "hardware." Computer memory is finite.
- What is $1 + 10^{-16}$?
  - 1.0000000000000001 in real arithmetic
  - 1 in floating point arithmetic (IEEE)
- Read Sec 4.3