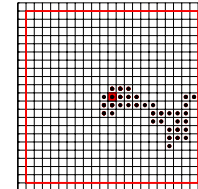


- Previous Lecture:
 - Probability and random numbers
 - 1-d array—vector
- Today's Lecture:
 - More examples on vectors
 - Simulation
- Announcement:
 - Discussion this week in computer lab UP B7
 - Project 3 due Oct 14. Use the lab computers if the simulator doesn't work with your computer.

Simulation

- Imitates real system
- Requires judicious use of random numbers
- Requires many trials
- → opportunity to practice working with vectors!

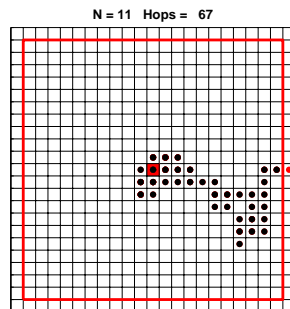


2-dimensional random walk

Start in the middle tile, (0,0).

For each step, randomly choose between N,E,S,W and then walk one tile. Each tile is 1×1.

Walk until you reach the boundary.



Lecture 12


5


Another representation for the random step

- Observe that each update has the form

$$xc = xc + \Delta x$$

$$yc = yc + \Delta y$$
 no matter which direction is taken.
- So let's get rid of the if statement!
- Need to create two “change vectors” deltaX and deltaY

deltaX 

deltaY 

Lecture 12

12

RandomWalk2D_v2.m

Lecture 12

13

Simulate twinkling stars

- Get 10 user mouse clicks as locations of 10 stars—our constellation
- Simulate twinkling
 - Loop through all the stars; each has equal likelihood of being bright or dark
 - Repeat many times
- Can use DrawStar, DrawRect

Lecture 12

14

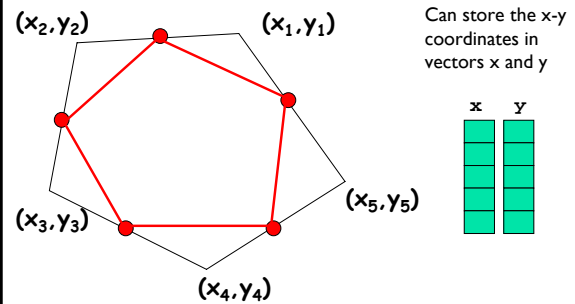
```

% No. of stars and star radius
N=10; r=.5;
% Get mouse clicks, store coords in vectors x,y
[x,y] = ginput(N);
% Twinkle!
for k= 1:20 % 20 rounds of twinkling

end

```

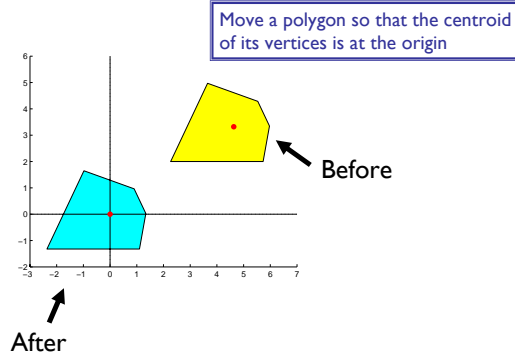
Example: polygon smoothing



Lecture 12

19

First operation: centralize



Lecture 12

20

```

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 = x-xBar;
yNew = y-yBar;

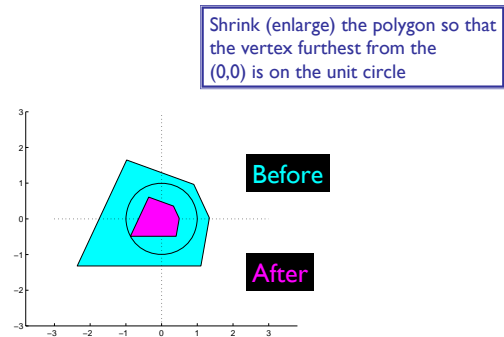
```

Vectorized code

Lecture 12

21

Second operation: normalize



Lecture 12

23

```

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

```

```

d = max(sqrt(x.^2 + y.^2));
xNew = x/d;
yNew = y/d;

```

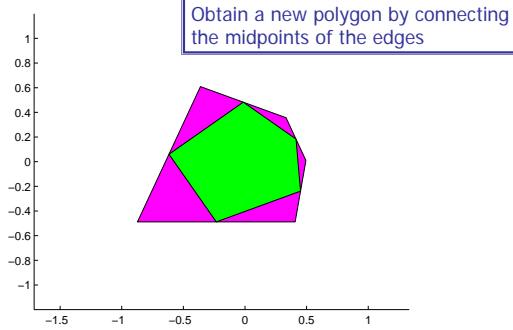
Vectorized ops

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

Lecture 12

24

Third operation: smooth



Lecture 12

25

```
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
```

Lecture 12

26

Polygon Smoothing

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

Does above fragment compute the new n-gon?

A: Yes

B: No

Lecture 12

30

Show a simulation of polygon smoothing

Create a polygon with randomly located vertices.

Repeat:

Centralize

Normalize

Smooth

Lecture 12

36

ShowSmooth.m

Lecture 12

37

Color computation

- Color is a 3-vector, sometimes called the RGB values
- Any color is a mix of red, green, and blue
- Example:

c = [0.4 0.6 0]



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

Lecture 12

38

Start with drawing a single line segment

```
a= 0; % x-coord of pt 1
b= 1; % y-coord of pt 1
c= 5; % x-coord of pt 2
d= 3; % y-coord of pt 2
plot([a c], [b d], '-*')
```

x-values
(a vector)

y-values
(a vector)

Line/marker
format

Lecture 12

39

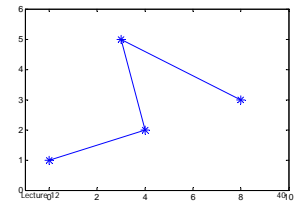
Making an x-y plot

```
a= [0 4 3 8]; % x-coords
b= [1 2 5 3]; % y-coords
plot(a, b, '-*')
```

x-values
(a vector)

y-values
(a vector)

Line/marker
format



Drawing a polygon (multiple line segments)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h.
x= [          ]; % x data
y= [          ]; % y data
plot(x, y)
```

Fill in the missing vector values!

Lecture 12

41

Coloring a polygon (fill)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h,
% and fill it with a color named by c.
x= [          ]; % x data
y= [          ]; % y data
fill(x, y, c)
```

A built-in function

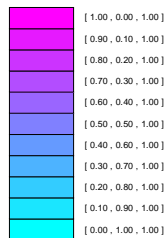
Lecture 12

44

```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2
```

```
for k= 0:n-1
    % Compute color of kth tile
    f= ???
    v= (1-f)*c1 + f*c2;
    % Draw kth tile
```

end



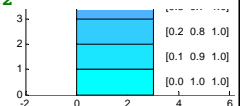
Lecture 12

47

```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2
```

```
for k= 0:n-1
    % Compute color of kth tile
    f= ???
    v= (1-f)*c1 + f*c2;
    % Draw kth tile
```

end



Lecture 12

49