Previous Lecture:
- Probability and random numbers
- 1-d array—vector

Today’s Lecture:
- More examples on vectors and simulation
- Color computation
- Linear interpolation

Announcement:
- Project 3 due on Friday 3/1
- Prelim 1 on Thurs 3/7 at 7:30pm
Simulation

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

% Given a vector v

for k = 1:length(v)
    % Work with v(k)
    % E.g., disp(v(k))
end

% Given a vector v
k = 1;
while k <= length(v)
    % Work with v(k)
    % E.g., disp(v(k))
    k = k+1;
end
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 $1 \times 1$.

Walk until you reach the boundary.
RandomWalk2D.m
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
RandomWalk2D_v2.m
Example: polygon smoothing

\[(x_2, y_2) \quad (x_1, y_1) \quad (x_3, y_3) \quad (x_4, y_4) \quad (x_5, y_5)\]
Example: polygon smoothing

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

(x_1, y_1)  (x_5, y_5)
(x_2, y_2)
(x_3, y_3)
(x_4, y_4)
First operation: centralize

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

Before

After
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
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;
end
Second operation: normalize

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

Before

After
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;

Applied to a vector, \( \text{max} \) 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
\[ x_{\text{New}(1)} = \frac{x(1) + x(2)}{2} \]
\[ y_{\text{New}(1)} = \frac{y(1) + y(2)}{2} \]
\[ x_{\text{New}(2)} = \frac{x(2) + x(3)}{2} \]
\[ y_{\text{New}(2)} = \frac{y(2) + y(3)}{2} \]
$x_{\text{New}(3)} = \frac{x(3) + x(4)}{2}$

$y_{\text{New}(3)} = \frac{y(3) + y(4)}{2}$
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
\[
x_{\text{New}(4)} = (x(4) + x(5))/2 \\
y_{\text{New}(4)} = (y(4) + y(5))/2
\]
\[ x_{\text{New}}(5) = \frac{x(5) + x(1)}{2} \]
\[ y_{\text{New}}(5) = \frac{y(5) + y(1)}{2} \]
Smooth

\begin{verbatim}
for i=1:n
    xNew(i) = (x(i) + x(i+1))/2;
    yNew(i) = (y(i) + y(i+1))/2;
end
\end{verbatim}

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
for $i=1:n-1$

$$\begin{align*}
\text{xNew}(i) &= (\text{x}(i) + \text{x}(i+1))/2; \\
\text{yNew}(i) &= (\text{y}(i) + \text{y}(i+1))/2;
\end{align*}$$

end

$$\begin{align*}
\text{xNew}(n) &= (\text{x}(n) + \text{x}(1))/2; \\
\text{yNew}(n) &= (\text{y}(n) + \text{y}(1))/2;
\end{align*}$$
Show a simulation of polygon smoothing

Create a polygon with randomly located vertices.

Repeat:

- Centralize
- Normalize
- Smooth
ShowSmooth.m
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
Making an x-y plot

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

- **x-values** (a vector)
- **y-values** (a vector)
- Line/marker format
Making an x-y plot with multiple graphs (lines)

\[
a = [0 \ 4 \ 3 \ 8] ; \\
b = [1 \ 2 \ 5 \ 3] ; \\
f = [0 \ 4 \ 6 \ 8 \ 10] ; \\
g = [2 \ 2 \ 6 \ 4 \ 3] ; \\
plot(a,b,\'-*\',f,g,\'c\')
\]
function paintChips(c1,c2,n)
% n tiles from color c1 to c2.

Use linear interpolation to obtain the colors. Each chip has a color v that is a linear combination of colors c1 and c2. Let f be a fraction in (0,1) ...

<table>
<thead>
<tr>
<th>f</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>[ 1.00, 0.00, 1.00 ]</td>
</tr>
<tr>
<td>0.10</td>
<td>[ 0.90, 0.10, 1.00 ]</td>
</tr>
<tr>
<td>0.20</td>
<td>[ 0.80, 0.20, 1.00 ]</td>
</tr>
<tr>
<td>0.30</td>
<td>[ 0.70, 0.30, 1.00 ]</td>
</tr>
<tr>
<td>0.40</td>
<td>[ 0.60, 0.40, 1.00 ]</td>
</tr>
<tr>
<td>0.50</td>
<td>[ 0.50, 0.50, 1.00 ]</td>
</tr>
<tr>
<td>0.60</td>
<td>[ 0.40, 0.60, 1.00 ]</td>
</tr>
<tr>
<td>0.70</td>
<td>[ 0.30, 0.70, 1.00 ]</td>
</tr>
<tr>
<td>0.80</td>
<td>[ 0.20, 0.80, 1.00 ]</td>
</tr>
<tr>
<td>0.90</td>
<td>[ 0.10, 0.90, 1.00 ]</td>
</tr>
<tr>
<td>1.00</td>
<td>[ 0.00, 1.00, 1.00 ]</td>
</tr>
</tbody>
</table>

\[
f = ??? \\
v = (1-f) \times c1 + f \times c2;
\]
Linear interpolation

<table>
<thead>
<tr>
<th>x</th>
<th>g(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>118</td>
</tr>
<tr>
<td>11</td>
<td>126</td>
</tr>
<tr>
<td>12</td>
<td>134</td>
</tr>
</tbody>
</table>
### Linear interpolation

<table>
<thead>
<tr>
<th>x</th>
<th>g(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>118</td>
</tr>
<tr>
<td>10.25</td>
<td>?</td>
</tr>
<tr>
<td>10.50</td>
<td>?</td>
</tr>
<tr>
<td>10.75</td>
<td>?</td>
</tr>
<tr>
<td>11</td>
<td>126</td>
</tr>
<tr>
<td>12</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
g(10.5) = \left\{ \frac{g(11) + g(10)}{2} \right\}
\]
### Linear interpolation

<table>
<thead>
<tr>
<th>x</th>
<th>g(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>9</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>118</td>
</tr>
<tr>
<td>10.25</td>
<td>?</td>
</tr>
<tr>
<td>10.50</td>
<td>?</td>
</tr>
<tr>
<td>10.75</td>
<td>?</td>
</tr>
<tr>
<td>11</td>
<td>126</td>
</tr>
<tr>
<td>12</td>
<td>134</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

\[
g(10.5) = \frac{1}{2} g(11) + \frac{1}{2} g(10)
\]

\[
g(10.25) = \frac{1}{4} g(11) + \frac{3}{4} g(10)
\]

\[
g(10.50) = \frac{2}{4} g(11) + \frac{2}{4} g(10)
\]

\[
g(10.75) = \frac{3}{4} g(11) + \frac{1}{4} g(10)
\]
### Linear interpolation

<table>
<thead>
<tr>
<th>$x$</th>
<th>$g(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>118</td>
</tr>
<tr>
<td>10.25</td>
<td>?</td>
</tr>
<tr>
<td>10.50</td>
<td>?</td>
</tr>
<tr>
<td>10.75</td>
<td>?</td>
</tr>
<tr>
<td>11</td>
<td>126</td>
</tr>
<tr>
<td>12</td>
<td>134</td>
</tr>
</tbody>
</table>

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

\[
g(10) = \frac{0}{4} \cdot g(11) + \frac{4}{4} \cdot g(10)
\]

\[
g(10.25) = \frac{1}{4} \cdot g(11) + \frac{3}{4} \cdot g(10)
\]

\[
g(10.50) = \frac{2}{4} \cdot g(11) + \frac{2}{4} \cdot g(10)
\]

\[
g(10.75) = \frac{3}{4} \cdot g(11) + \frac{1}{4} \cdot g(10)
\]

\[
g(11) = \frac{4}{4} \cdot g(11) + \frac{0}{4} \cdot g(10)
\]

\[
f \cdot g(11) + (1-f) \cdot g(10)
\]
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
function paintChips(c1,c2,n)
% n tiles from color c1 to c2

x= [0 3 3 0];
y= [0 0 1 1];
for k= 0:n-1
    % Compute color of kth tile
    f= k/(n-1);
    v= (1-f)*c1 + f*c2;
    % Draw kth tile
    fill(______, ______, v)
end
function paintChips(c1,c2,n)
% n tiles from color c1 to c2

x = [0 3 3 0];
y = [0 0 1 1];
for k = 0:n-1
% Compute color of kth tile
f = k/(n-1);
v = (1-f)*c1 + f*c2;
% Draw kth tile
fill(x, y+k, v)
text(3.5, k+.5, ...
    sprintf(’[%.1f %.1f %.1f]’,
    v(1),v(2),v(3) )
end