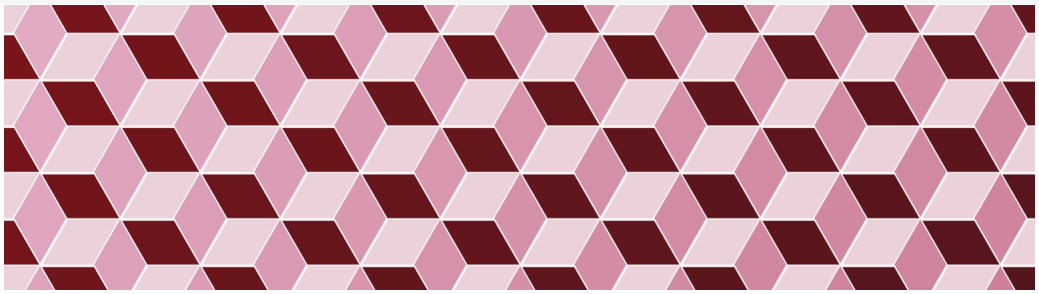


15-104 Introduction to Computing for Creative Practice

Fall 2021

14 More Arrays

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1

Arrays

- An array with n elements (n > 0) is an ordered collection of values of the same type, indexed from 0 to n-1.

```
var temps = [79, 81, 57, 64, 63, 57, 58]

          0    1    2    3    4    5    6
        ---------------------------------
temps | 79 | 81 | 57 | 64 | 63 | 57 | 58 |
        ---------------------------------

temps.length                    7
temps[0]                        79
temps[temps.length – 1]         58
```

2

Array methods: `push` and `shift`

```
                          0    1    2    3    4    5    6
                        ------------------------------------
                temps   | 79 | 81 | 57 | 64 | 63 | 57 | 58 |
                        ------------------------------------

                          0    1    2    3    4    5    6    7
                        -------------------------------------------
temps.push(65);   temps  | 79 | 81 | 57 | 64 | 63 | 57 | 58 | 65 |
                        -------------------------------------------

                          0    1    2    3    4    5    6
                        ------------------------------------
x = temps.shift();  temps | 81 | 57 | 64 | 63 | 57 | 58 | 65 |
                        ------------------------------------
```

3

Arrays operations

Some JavaScript array methods that may be useful:

- push(element) — adds a new element to an array (at the end)
- pop() — deletes the last element, and returns it
- shift() — returns the first element and shifts the rest down
- unshift(element) — adds a new element to an array (at the beginning)
- reverse() — reverses the elements in an array

See the p5.js reference under Data for more methods!

4

Storing Data for Later

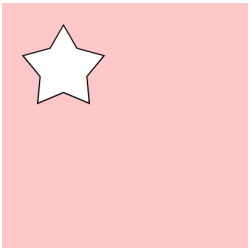
- A good use of arrays is when you need to process the data later for some information.
- Example: Find the number of values in your array that exceed the average.

```
sum = 0;
for (var i = 0; i < temps.length; i++) {
     sum += temps[i];
}
average = sum / temps.length;
count = 0;
for (var i = 0; i < temps.length; i++) {
     if (temps[i] > average) { count++; }
}
```

5

Star

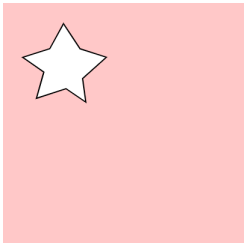
```
var x = [50, 61, 83, 69, 71, 50, 29, 31, 17, 39];
var y = [18, 37, 43, 60, 82, 73, 82, 60, 43, 37];

function setup() {
    createCanvas(200, 200);
}

function draw() {
    background(255, 200, 200);
    var nPoints = x.length;
    beginShape();
    for (var i = 0; i < nPoints; i++) {
        vertex(x[i], y[i]);
    }
    endShape(CLOSE);
    noLoop();
}
```



You can create arbitrary shapes by using the `beginShape` function, then create a series of vertices using the `vertex` function, and then end the shape with the `endShape` function.

6

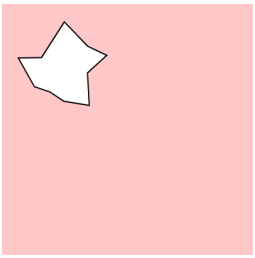
Star Jitter



```
var x = [50, 61, 83, 69, 71, 50, 29, 31, 17, 39];
var y = [18, 37, 43, 60, 82, 73, 82, 60, 43, 37];

function setup() {
    createCanvas(200, 200);
    frameRate(10);
}

function draw() {
    background(255, 200, 200);
    var nPoints = x.length;
    beginShape();
    for (var i = 0; i < nPoints; i++) {
        vertex( x[i] + random(-3,3), y[i] + random(-3,3) );
    }
    endShape(CLOSE);
}
```

7

Star Walk



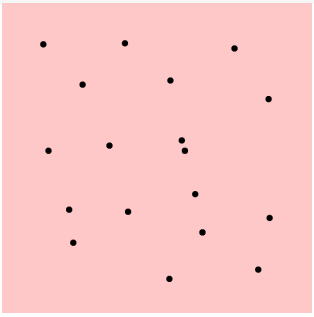
```
var x = [50, 61, 83, 69, 71, 50, 29, 31, 17, 39];
var y = [18, 37, 43, 60, 82, 73, 82, 60, 43, 37];

function setup() {
    createCanvas(200, 200);
    frameRate(5);
}

function draw() {
    background(255, 200, 200);
    var nPoints = x.length;
    for (var i = 0; i < nPoints; i++) {
        x[i] += random(-1.5, 1.5);
        y[i] += random(-1.5, 1.5);
    }
    fill(255);
    stroke(0);
// cont'd
    beginShape();
    for (var i = 0; i < nPoints;
             i++) {
        vertex(x[i], y[i]);
    }
    endShape(CLOSE);
}
```

8

Spots on the Canvas

```
var ex;
var ey;

function setup() {
    createCanvas(300, 300);
    background(255, 200, 200);
    frameRate(5);
}

function draw() {
    fill(0);
    ellipse(ex, ey, 5, 5);
}

function mousePressed() {
    ex = mouseX; ey = mouseY;
}
```

What if this were done in `draw` instead?



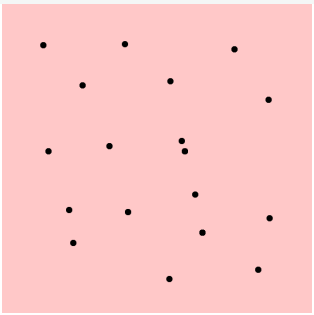
What if we wanted to perform some animation with the points on this canvas? We can't do this since we don't store all of the points for processing later.

Unless we use arrays…

9

Spots on the Canvas

```
var ex = [];  // start with empty arrays
var ey = [];

function setup() {
    createCanvas(300, 300);
    frameRate(5);
}

function draw() {
    background(255, 200, 200);  // moved from setup
    fill(0);
    for (var i = 0; i < ex.length; i++) {
        ellipse(ex[i], ey[i], 5, 5);
    }
}

function mousePressed() { ex.push(mouseX); ey.push(mouseY); }
```



Append (push) each new click position to the arrays:

10

Jitterpoints

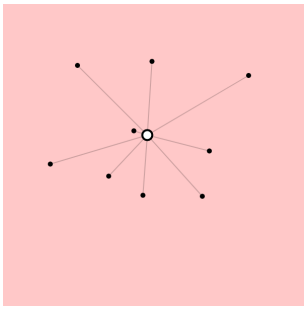
```
var ex = [];
var ey = [];

function setup() {
    createCanvas(300, 300);
    frameRate(5);
}

function draw() {
    background(255, 200, 200);
    fill(0);
    for (var i=0; i < ex.length; i++) {
        ellipse(ex[i]+random(-3,3), ey[i]+random(-3,3), 5, 5);
    }
}

function mousePressed() { ex.push(mouseX); ey.push(mouseY); }
```

Because we save the points in the `ex` and `ey` arrays, we can use them to redraw the canvas and add a random jitter to get shaky points.

11

Centroid

- Let's compute and display the **centroid** or geometric center of a collection of points by calculating the average x-coordinate and the average y-coordinate. (The centroid is also known as the *center of mass*.)
- How would we modify the previous program to compute the centroid and draw it connected to all of the points with lines?



12

Centroid

```
function draw() {
    background(255, 200, 200);
    fill(0);
    noStroke();
    var nPoints = ex.length;

    if (nPoints < 1) {
        text("Click to begin", 5, 15);
    }

    // Draw the dots
    for (var i = 0; i < nPoints; i++) {
        ellipse (ex[i], ey[i], 5, 5);
    }
    // cont'd on next slide
```

Only the `draw` function needs to be updated.

We have this `text` function here when the program starts so the user knows that they need to click on the canvas.

13

Centroid (cont'd)

```
    // Compute their centroid point
    // (exAverage, eyAverage)
    var exSum = 0;
    var eySum = 0;
    for (var i = 0; i < nPoints; i++) {
        exSum += ex[i];
        eySum += ey[i];
    }
    exAverage = exSum / nPoints;
    eyAverage = eySum / nPoints;


    // continued on next slide
```

14

Centroid (cont'd)

```
    // Draw lines from the centroid to every point
    strokeWeight(1);
    stroke(0, 0, 0, 50);
    for (var i = 0; i < nPoints; i++) {
        line (ex[i], ey[i], exAverage, eyAverage);
    }

    // Draw the centroid
    fill(255);
    stroke(0);
    strokeWeight(2);
    ellipse(exAverage, eyAverage, 10, 10);
}
```

15

Using a Function!

```
// Compute their centroid point
// (exAverage, eyAverage)
var exSum = 0;
var eySum = 0;
for (var i = 0; i < nPoints; i++) {
    exSum += ex[i];
    eySum += ey[i];
}
exAverage = exSum / nPoints;
eyAverage = eySum / nPoints;
```

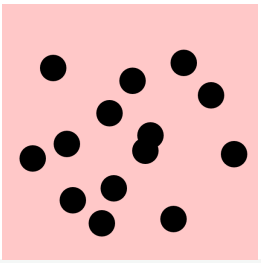
```
exAverage = computeAvg(ex, nPoints);
eyAverage = computeAvg(ey, nPoints);
```

```
function computeAvg(arr, len) {
    var sum = 0;
    for (var i = 0; i < len; i++) {
        sum += arr[i];
    }
    return sum / len;
}
```

16

Try This

- Modify Jitterpoints so that the diameter of each spot is given by the seconds value in the current time of the clock.
  - For example, if the time is 10:09:43, then the diameter of the spots would be 43.
- As the program runs, the spots should get bigger, once per second until the seconds is reset to 00, and then the spots will disappear and begin growing again.



17