

# Nested Loops & Arrays

CSE 120 Spring 2017

**Instructor:**

Justin Hsia

**Teaching Assistants:**

Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

# Administrivia

- ❖ Assignments:
  - Creativity Planning due Tuesday (4/18)
    - Find a partner, come up with *two* proposed programs
  - Portfolio Update 1 due Tuesday (4/18)
  - Binary Practice (4/21)
  - Creativity Assignment (4/24)
  
- ❖ Midterm in class on Wednesday, 4/26
  - 1 sheet of notes (2-sided, letter, handwritten)
  - Fill-in-the-blank(s), short answer questions, maybe simple drawing

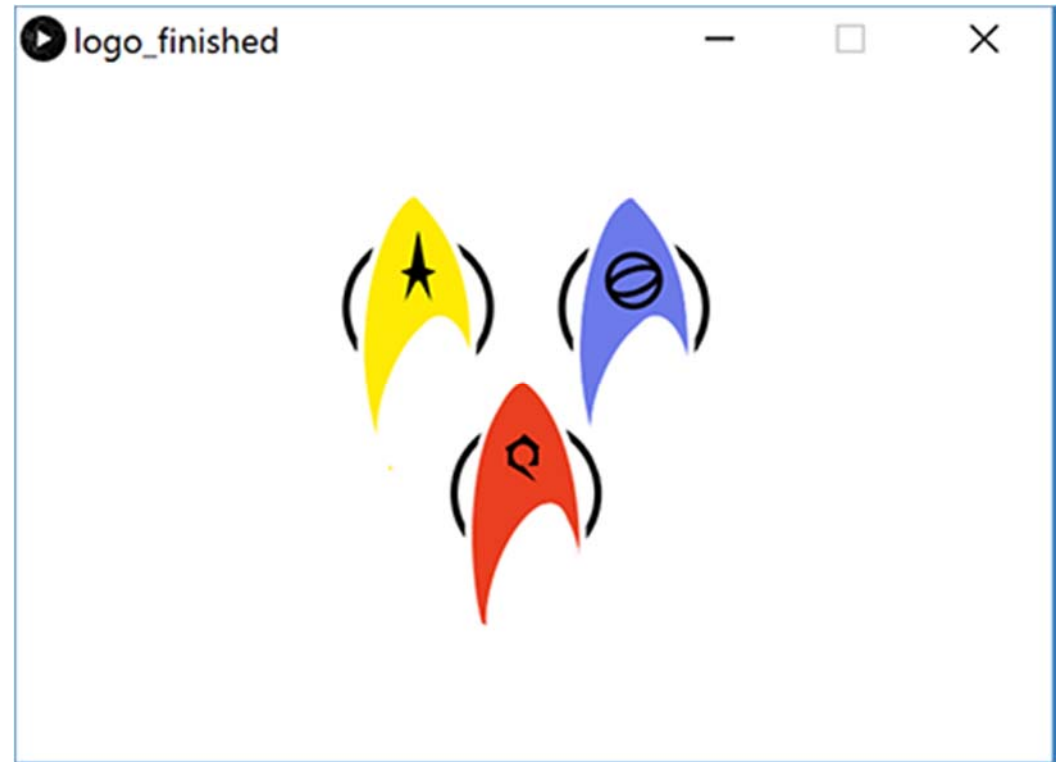
# Outline

- ❖ **Student Work Showcase**
- ❖ For-Loop Review
- ❖ Nested Loops
- ❖ Arrays
  - Arrays and Loops

# Custom Logo



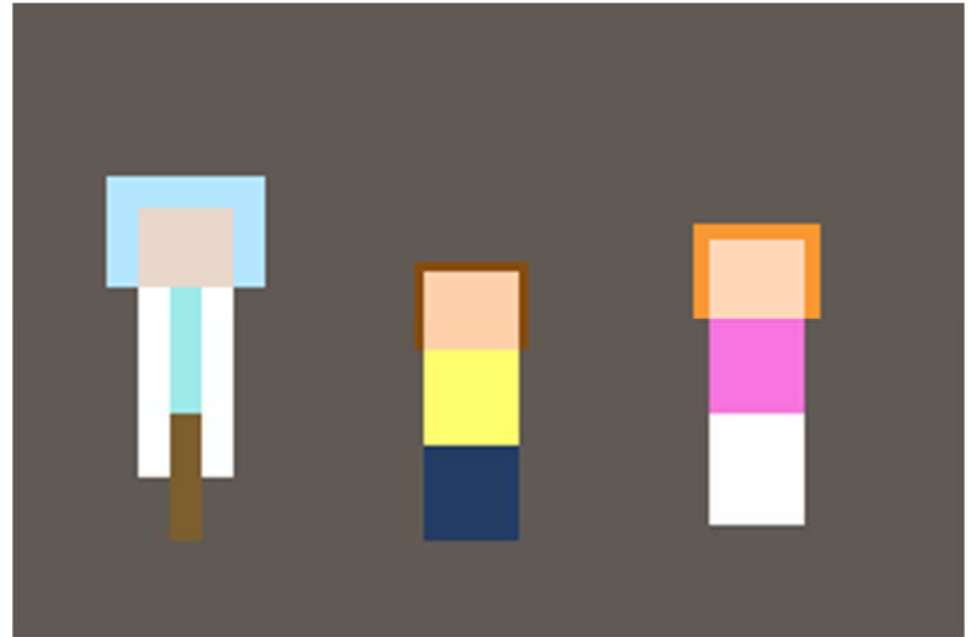
Cadey Kwon



# Lego Family



Sarah Liu

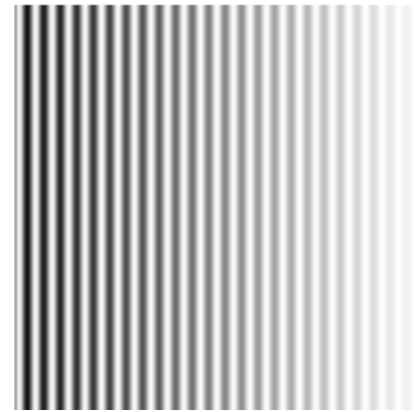
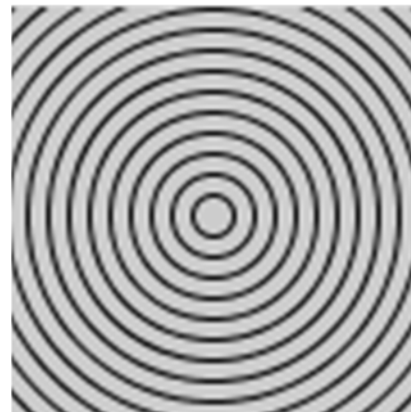
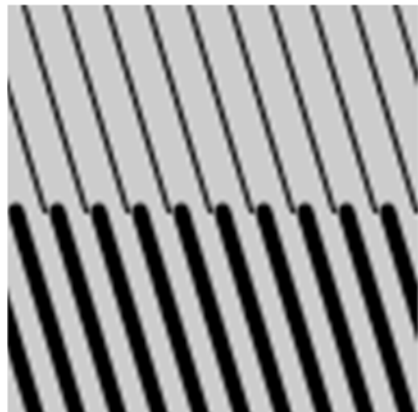


# Outline

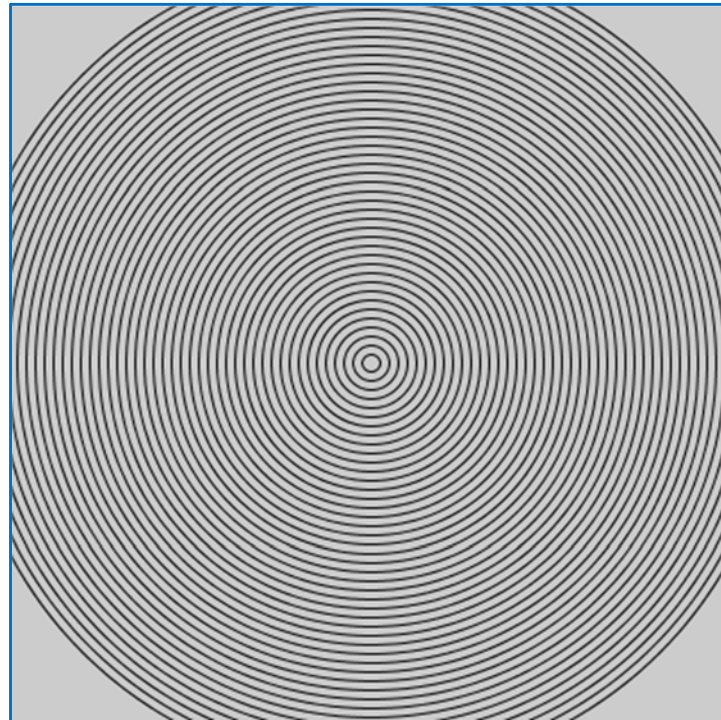
- ❖ Student Work Showcase
- ❖ **For-Loop Review**
- ❖ Nested Loops
- ❖ Arrays
  - Arrays and Loops

# For-Loop Review

- ❖ Loops control a sequence of *repetitions*
  - Do the same thing (or similar things) over and over again
- ❖ Examples: What is changing?



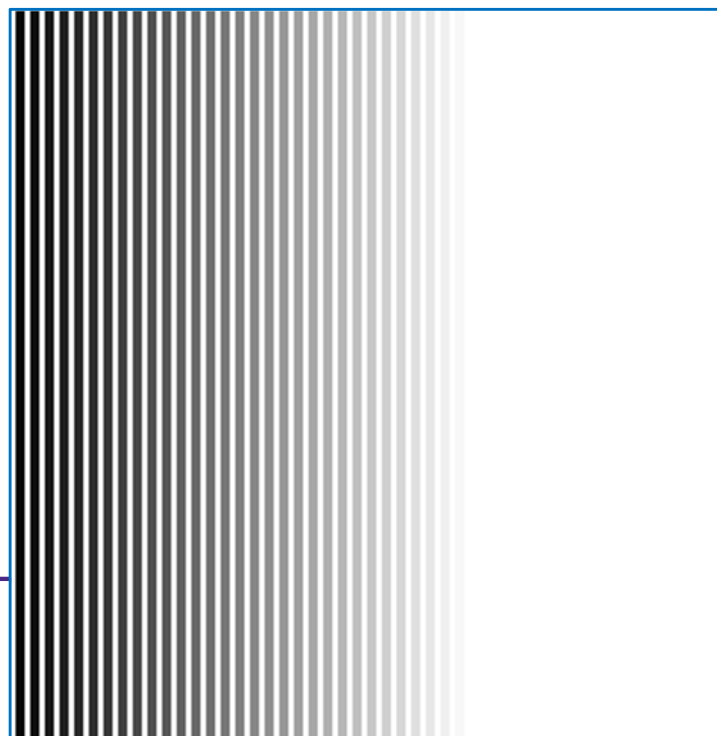
# Example: Circle Loop



```
size(400, 400);  
  
noFill();  
for(int d = 450; d > 0; d = d - 10) {  
    ellipse(width/2, height/2, d, d);  
}
```



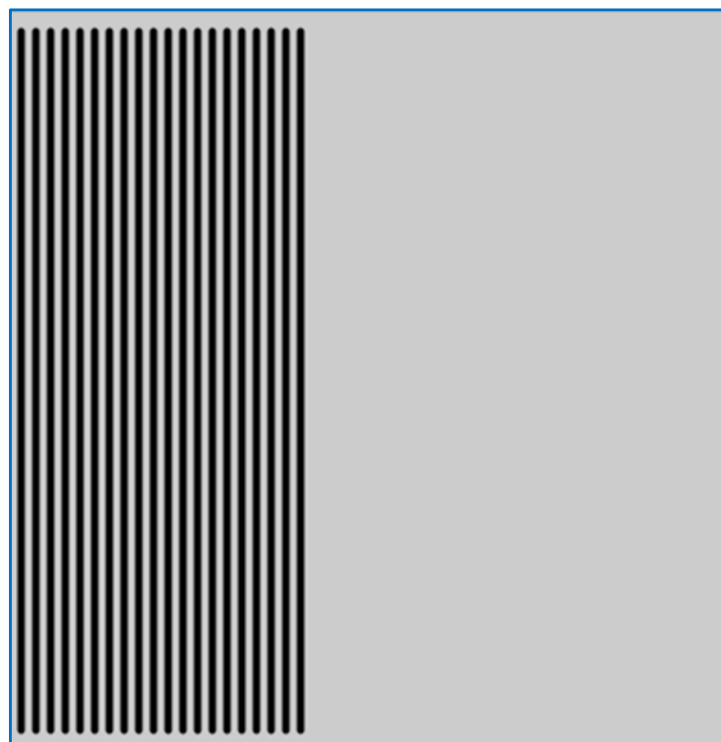
# Example: Line Gradient



```
size(400, 400);  
  
background(255);  
strokeWeight(5);  
  
for(int i = 0; i < 400; i = i + 8){  
    stroke(i);  
    line(i, 0, i, 400);  
}
```

# Example: Looping with User Interaction?

- ❖ Draw lines from left side of screen to the horizontal position of the mouse



# Example: Draw Lines to mouseX

```
void setup() {  
  size(400, 400);  
  strokeWeight(4);  
}  
  
void draw() {  
  background(204);  
  
  for(int i = 10; i < mouseX; i = i + 8){  
    line(i, 10, i, 390);  
  }  
}
```

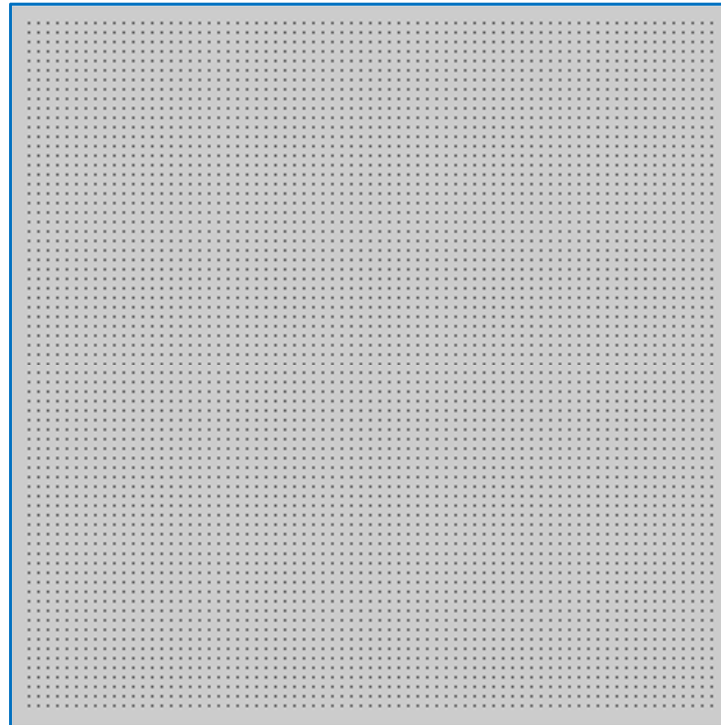
# Outline

- ❖ Student Work Showcase
- ❖ For-Loop Review
- ❖ **Nested Loops**
- ❖ Arrays
  - Arrays and Loops

# Nested Loops

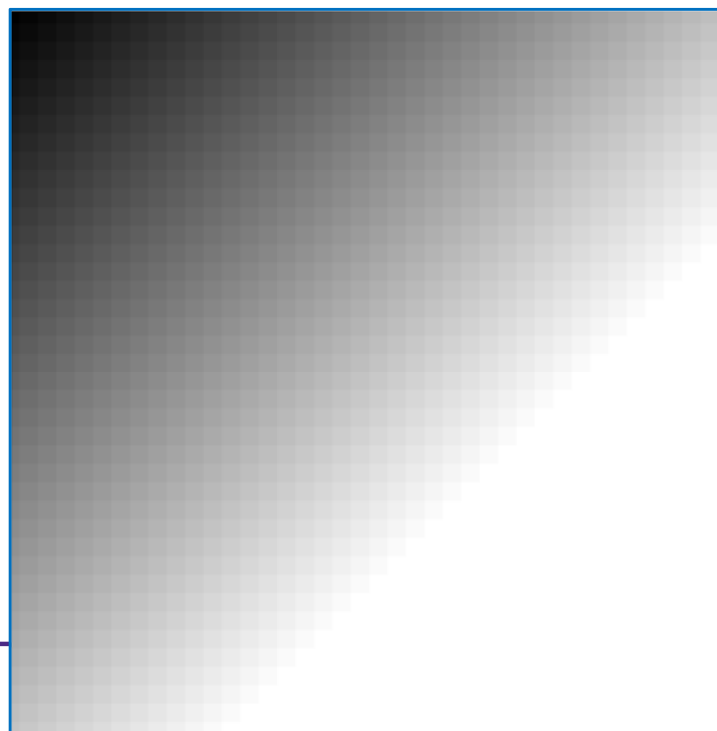
- ❖ Generally a for-loop has a single loop variable that changes with each iteration
- ❖ What if you need/want more things to change?
  - Can **nest** loops – *i.e.* put a loop inside of another loop

# Example: Dot Grid



```
size(400, 400);  
  
for(int y = 20; y <= height-20; y = y + 5){  
    for(int x = 20; x <= width-20; x = x + 5){  
        point(x, y);  
    }  
}
```

# Example: 2D Gradient



```
size(400, 400);  
noStroke();  
  
for(int y = 0; y < width; y = y + 10){  
  for(int x = 0; x < height; x = x + 10){  
    fill((x+y)*0.5);  
    rect(x, y, 10, 10);  
  }  
}
```

# Outline

- ❖ Student Work Showcase
- ❖ For-Loop Review
- ❖ Nested Loops
- ❖ **Arrays**
  - **Arrays and Loops**



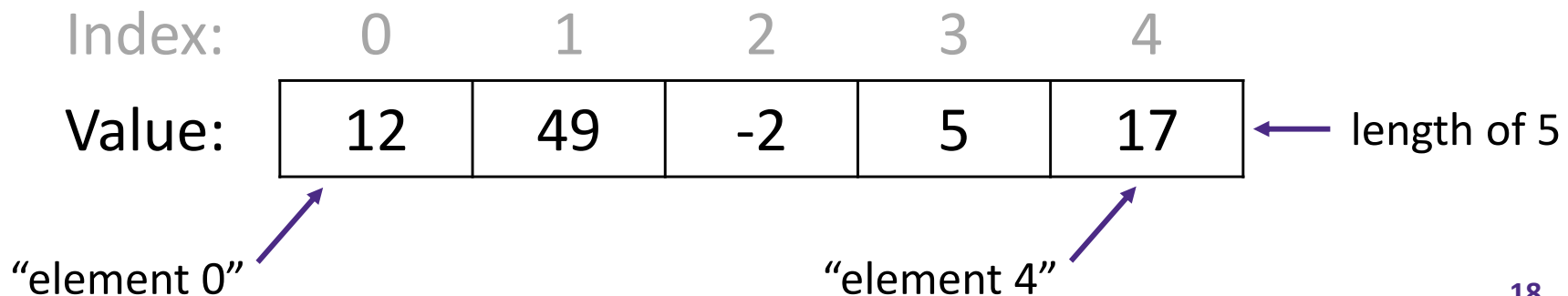
# Arrays

- ❖ “Structures” that store many values *of the same datatype*
  - Help us group related data
- ❖ Arrays store large amounts of data that you can access using a single variable name
  - Accessing arrays with loops is very convenient

# Arrays

- ❖ “Structures” that store many values *of the same datatype*
  - **Element**: a single value in the array
  - **Index**: a number that specifies the location of a particular element of the array
    - Start from 0
  - **Length**: total number of elements in the array

- ❖ Example:



# Arrays in Processing

- ❖ Declaration: `type[] name`
  - e.g. `int[]` is array of integers, `color[]` is array of colors
- ❖ Creation: `new type[num]`
  - e.g. `int[] intArr = new int[5];`
  - Default value for *all* elements is “zero-equivalent” (0, 0.0, `false`, black)
  - Remember that actual indices are from 0 to `num-1`
- ❖ Initialization: `{elem0, elem1, ..., elemN};`
  - e.g. `int[] intArr = {12, 49, -2, 5, 17};`

# Arrays in Processing

- ❖ Use element: `name[index]`
  - In *expression*, uses value of that index of the array
  - In *assignment*, modifies value of that index of the array

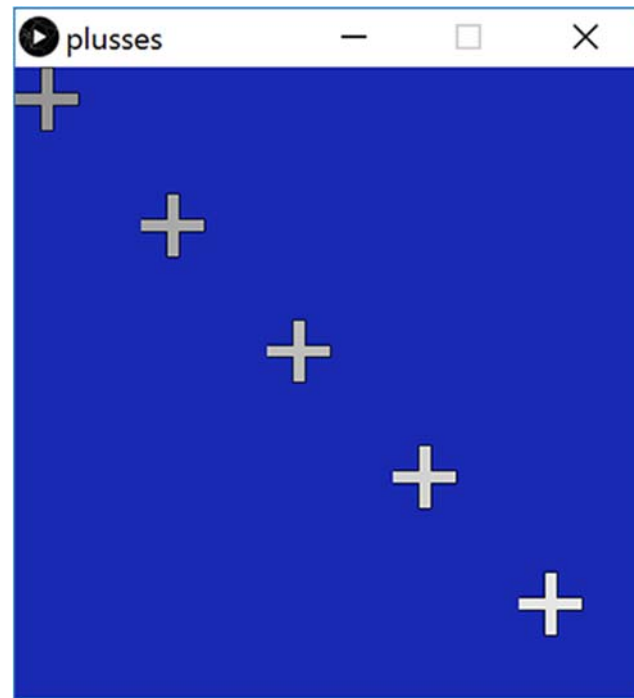
- ❖ Get length: `name.length`

- ❖ Example:

```
int[] intArr = {12, 49, -2, 5, 17};  
println(intArr[0]);           // prints 12 to console  
intArr[2] = intArr.length; // changes -2 to 5
```

Index:	0	1	2	3	4
Value:	12	49	-2	5	17

# Example: Lots of Pluses



# Example: Index of Smallest Number

## ❖ Algorithm:

- Keep track of the *index* of the smallest number seen so far
  - Start with index 0
- Check each *element* 1-by-1; if number is smaller, then update the smallest index

```
9 // returns the index of the smallest number in a list
10 int find_smallest(float[] list) {
11     int smallest = 0;
12     for(int i = 1; i < list.length; i=i+1) {
13         if(list[i] < list[smallest]) {
14             smallest = i;
15         }
16     }
17     return smallest;
18 }
```