INTRODUCTION TO PROCESSING

Alark Joshi, Amit Jain, Jyh-haw Yeh and Tim Andersen

What is Processing?

- Processing is a programming language designed to make programming easier
- Developers were frustrated with how difficult it was to write small programs for new programmers
  - Need for compilers
  - Software that wasn't always free
  - Didn't always work together
  - Took the joy out of process of learning programming

Goal

- Goal was to make a programming language that would allow fast prototyping of ideas
- Easy to use and teach at a middle-school and high-school level
  - With minimal setup
  - Immediate visual feedback
  - Potential to undertake big projects using the language

Processing

- Prototyping is done in the form of a *sketch*
- Programmers keep adding a few lines and adding to the sketch
- Visual feedback feeds the curiosity that is inherent in new programmers
- The vision behind processing is to enable the process of learning programming through creating interactive graphics

Let us begin!

- Easy to download and install processing from http://processing.org

Let us write our First Program

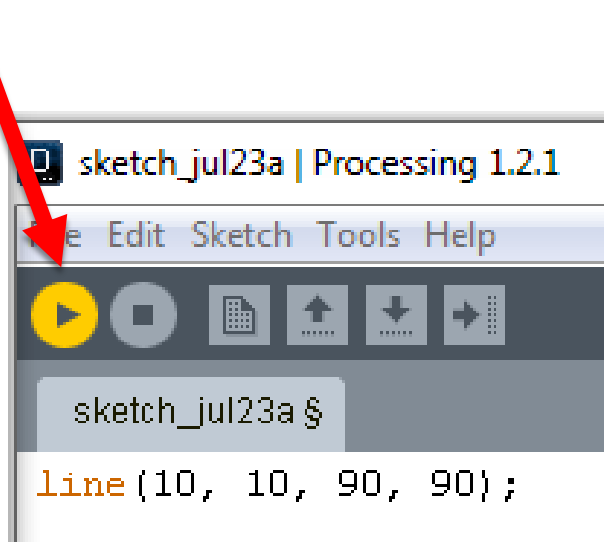
- Think about graph paper
- How would you draw a line between (0, 0) and (5,5)?
- Pick the x,y location of the starting point and the x,y location of the destination point and connect the dots in between
- Please work with another student and draw a line on the graph between
  1. (2,0) and (2,6)
  2. (2,4) and (6,4)

First Program

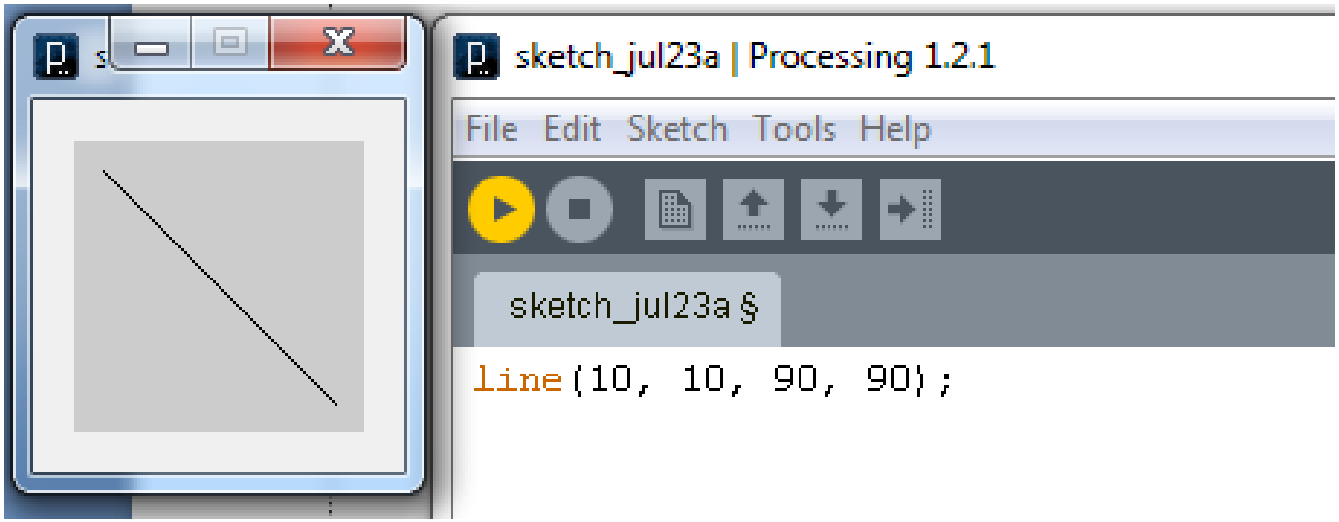
X, Y-coords of 1st point

X, Y-coords of 2nd point

- line(10, 10, 90, 90);
  - This is your first program
  - Type it in and click on the **Triangle (Play button)** in the top menu or select **Sketch→Run**



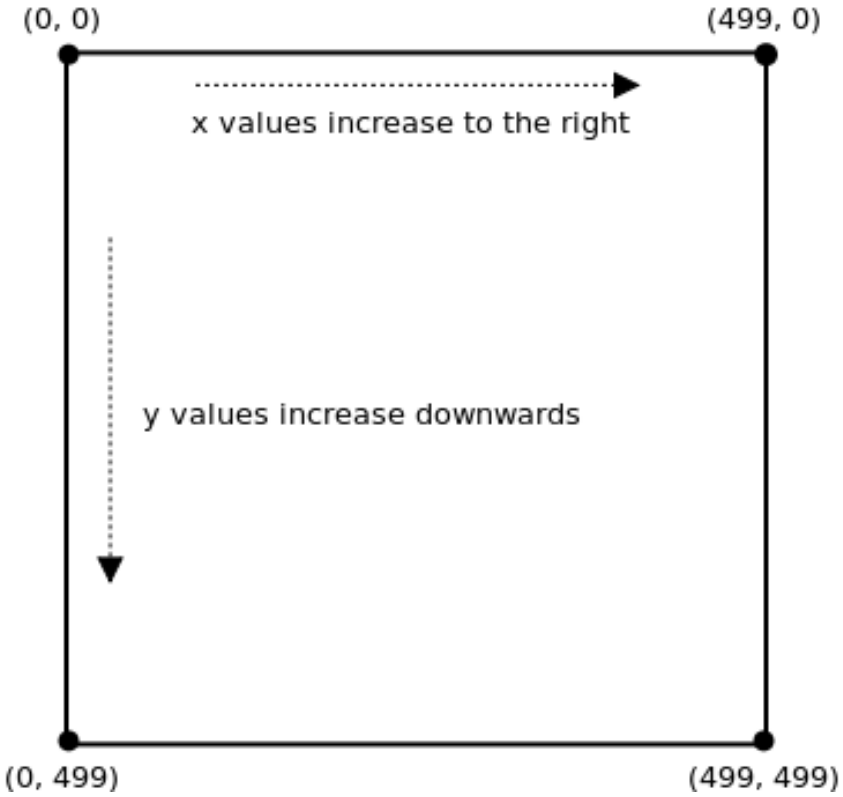
First Program



What is *different* about the direction of the line?

Coordinates on the Screen

- On a computer screen, the origin (0, 0) is **always** at the top left location of the screen



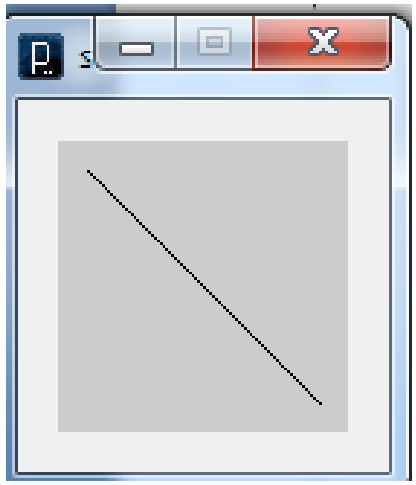
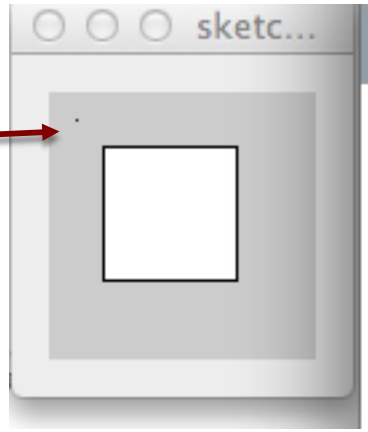


Image credits: http://www.cs.sfu.ca/CourseCentral/166/tjd/translation_rotation.html

Basic Shapes

- Points, Rectangles, Triangles, Quadrilaterals etc.



point at (10,10)

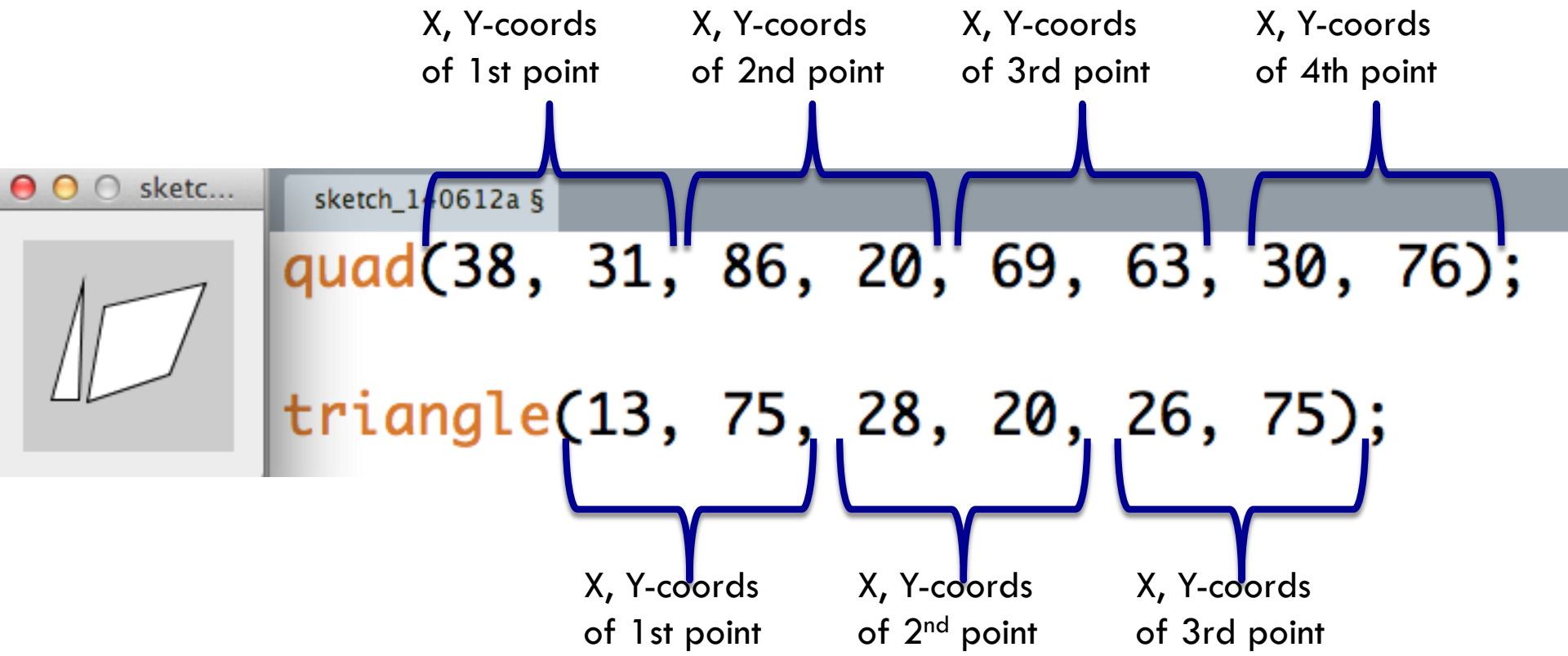
```
sketch_140612a §
point(10, 10);

rect(20, 20, 50, 50);
```

X, Y-coords of 1st point

Length and Width of the rectangle

Basic Shapes

- Points, Rectangles, Triangles, Quadrilaterals etc.



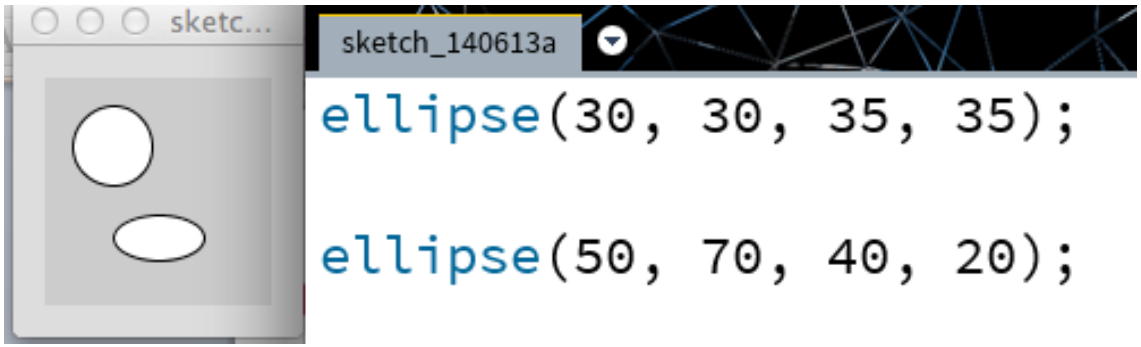
Draw basic shapes

- Type out the code in Processing and Click on Run to see a rectangle, a quadrilateral, and a triangle


```
rect(15, 15, 20, 20);

quad(38, 31, 86, 20, 69, 63, 30, 76);

triangle(13, 75, 28, 20, 26, 75);
```

Ellipses and Circles

- Ellipse can be represented by specifying the
  1. The coordinates of the center of the ellipse
  2. The diameter of the ellipse in the X-direction
  3. The diameter of the ellipse in the Y-direction
- `ellipse(xc, yc, xdia, ydia);`



- A circle is an ellipse with the **same** value for the x-diameter and y-diameter

Arcs

- Partial ellipses/circles can be drawn by the arc function.
  arc (x, y, width, height, startAngle, stopAngle);
- The outer edge of the ellipse defined by top-left corner *x*, *y*, *width* and *height* (where *x*, *y* is the center if we use ellipseMode(CENTER) before calling arc)
- The angle is in radians or use radians(*degrees*) to convert degrees to radians
- Try the following:
  - `arc(50, 50, 100, 100, 0, radians(180));`
  - `arc(50, 50, 100, 100, radians(270), radians(360));`
  - `arc(50, 50, 100, 100, radians(270), radians(90));`
  - `arc(50, 50, 100, 100, radians(270), radians(45`

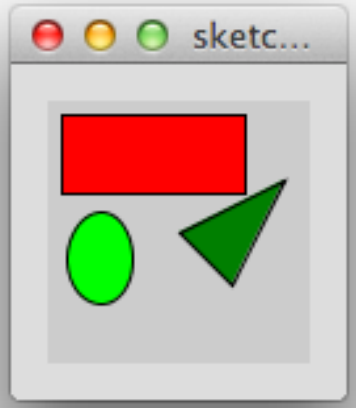
Representing Colors

- Colors in Processing are represented as a combination of (Red, Green, Blue) values
- 0 = no contribution of a particular color
- 255 = maximum contribution of a particular color
- Pure Red = 255, 0, 0
- Pure Green = 0, 255, 0

Filling in shapes

- Shapes can be filled in with different colors

```
// Set fill color to Red
fill(255, 0, 0);
rect(5, 5, 70, 30);

// Set fill color to Green
fill(0, 255, 0);
ellipse(20, 60, 25, 35);

// Set fill color to a shade of Green
fill(0, 127, 0);
triangle(50, 50, 70, 70, 90, 30);
```



Experiment with Color

- Type out the following lines and see if you can change the shapes and their fill colors

```
// Set fill color to Red
fill(255, 0, 0);
rect(5, 5, 70, 30);

// Set fill color to Green
fill(0, 255, 0);
ellipse(20, 60, 25, 35);

// Set fill color to a shade of Green
fill(0, 127, 0);
triangle(50, 50, 70, 70, 90, 30);
```

Red, Green, Blue combinations

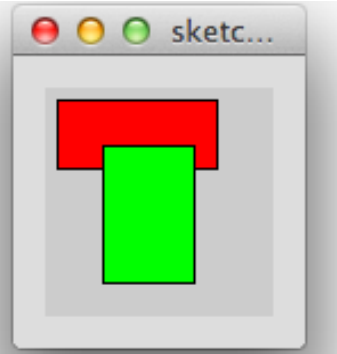
Black = 0, 0, 0
Yellow = 255, 255, 0
Orange = 255, 102, 0
Brown = 165, 42, 42
Fuchsia = 255, 0, 255
Olive = 128, 128, 0
White = 255, 255, 255

More combinations can be found using the Color Selector tool under the Tools menu

Opacity/Transparency

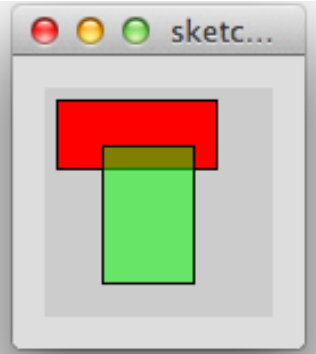
- Opacity/Transparency also defined separately
- 0 = completely transparent
- 255 = completely opaque

```
fill(255, 0, 0);
rect(5, 5, 70, 30);

fill(0, 255, 0);
rect(25, 25, 40, 60);
```



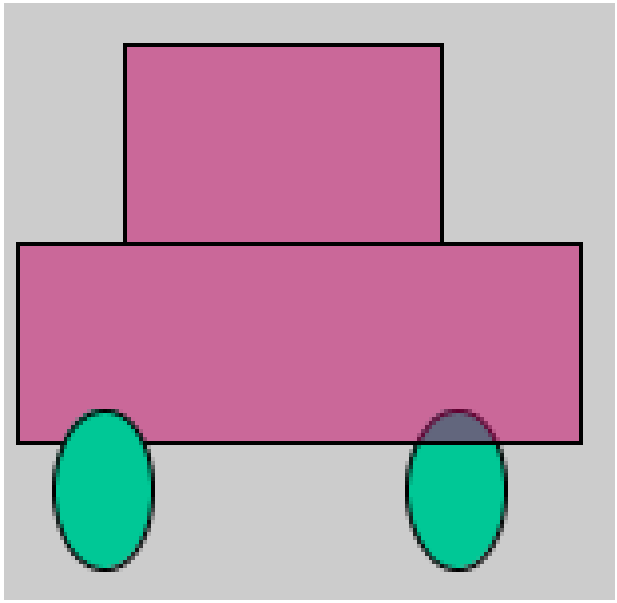
Overlapping rectangles without transparency

```
fill(255, 0, 0);
rect(5, 5, 70, 30);

fill(0, 255, 0, 127);
rect(25, 25, 40, 60);
```



Overlapping rectangles with transparency

Using color & opacity

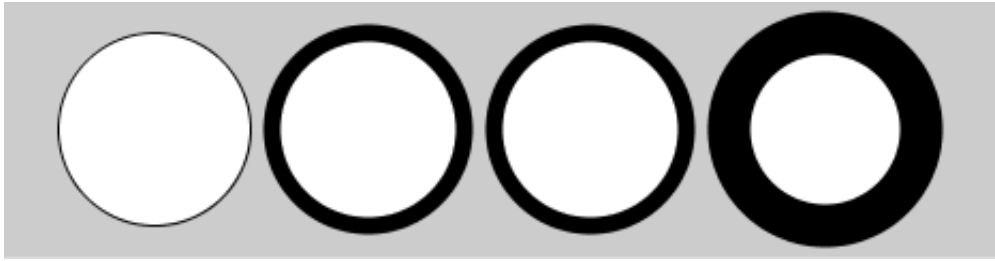


Resize the drawing canvas

```
size(200, 200);
fill(0, 200, 150);
ellipse(117, 130, 25, 40);

fill(200, 0, 100, 125);
rect(6, 68, 142, 50);
rect(33, 18, 80, 50);

fill(0, 200, 150);
ellipse(28, 130, 25, 40);
```

Stroke Weight

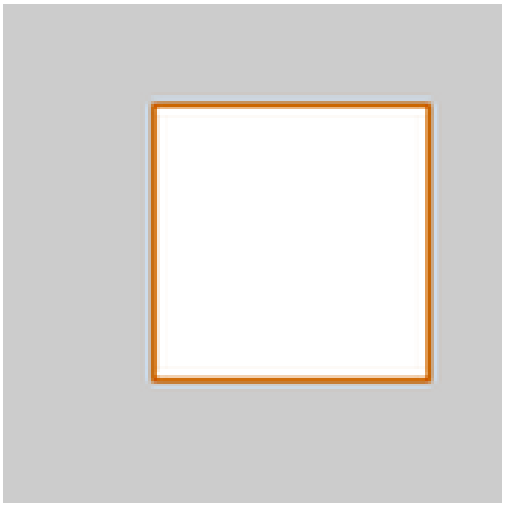
- strokeWeight(value);



```
size(480, 120);
smooth();
ellipse(75, 60, 90, 90);
strokeWeight(8);  // weight = 8 pix
ellipse(175, 60, 90, 90);
ellipse(279, 60, 90, 90);
strokeWeight(20); // weight = 20 pix
ellipse(389, 60, 90, 90);
```

Stroke Color

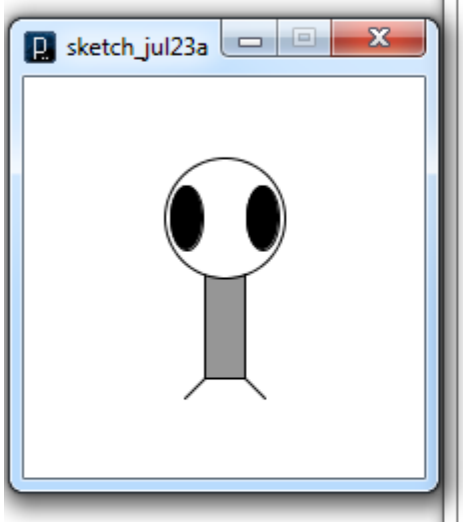
- `stroke(color);`
  - Could be a single value (to specify gray scale)
  - Could be 3 values for RGB



```
stroke(204, 102, 0);
rect(30, 20, 55, 55);
```

Composition of Shapes

- Drawing an alien – Lets call it Rooba



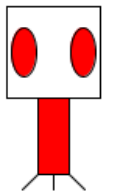

```
size(200,200);
background(255);  // Specifies the background color
smooth();       // Specifies extra smoothing of drawn primitives

ellipseMode(CENTER);
rectMode(CENTER);

// Body
stroke(0);
fill(150);
rect(100,100,20,100);

// Head
fill(255);
ellipse(100,70,60,60);

// Eyes
fill(0);
ellipse(81,70,16,32);
ellipse(119,70,16,32);

// Legs
stroke(0);
line(90,150,80,160);
line(110,150,120,160);
```

Activity – Modify Rooba

Modify the Rooba program to give the Alien a Square head. Change its eyes and body to red. Then make it have three legs!

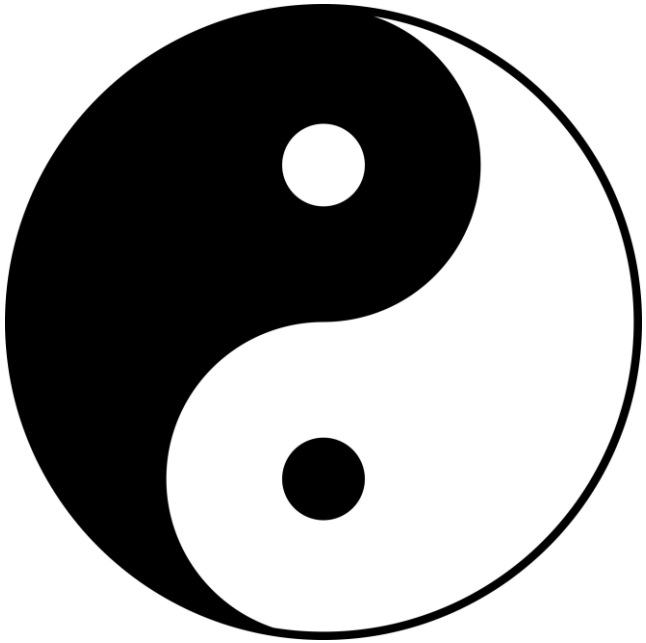
```
size(200,200);  // Set the size of the window
background(255);  // White background
smooth();
// Set ellipses and rects to CENTER mode
ellipseMode(CENTER);
rectMode(CENTER);

// Draw Rooba's body
stroke(0);
fill(150);
rect(100,100,20,100);
```

```
// Draw Rooba's head
fill(255);
ellipse(100,70,60,60);

// Draw Rooba's eyes
fill(0);
ellipse(81,70,16,32);
ellipse(119,70,16,32);

// Draw Rooba's legs
stroke(0);
line(90,150,80,160);
line(110,150,120,160);
```



Processing Sketch for Yin and Yang

- Write a sketch to draw Yin and Yang. Make the sketch size be 400 x400 and the Yin and Yang be 200 x 200
- Need to use ellipses and arcs
- Write Yin Yang on the screen using
  - `text("Yin Yang", 10, 30);`



Variables

- If you have a number stored on your computer and would like to add 1 to it
  - An easy way to access to that location
  - Find the value of the number stored at that location
  - Increment the value
  - Store the new updated value back to that location

- Definition: A variable contains some known or unknown quantity or information, a value.

Variable

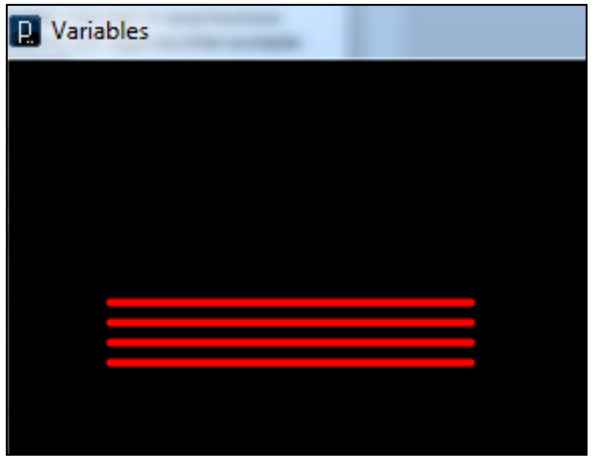
- Examples:
  - char value = ‘a’;
  - int i = 100;
  - float x = 0.33;
  - String str1 = “USA”; //note that S is uppercase

Variables Program – 3 circles

```
size(480, 120);
smooth();
int y = 60;
int d = 80;
ellipse(75, y, d, d);
ellipse(175, y, d, d);
ellipse(275, y, d, d);
```

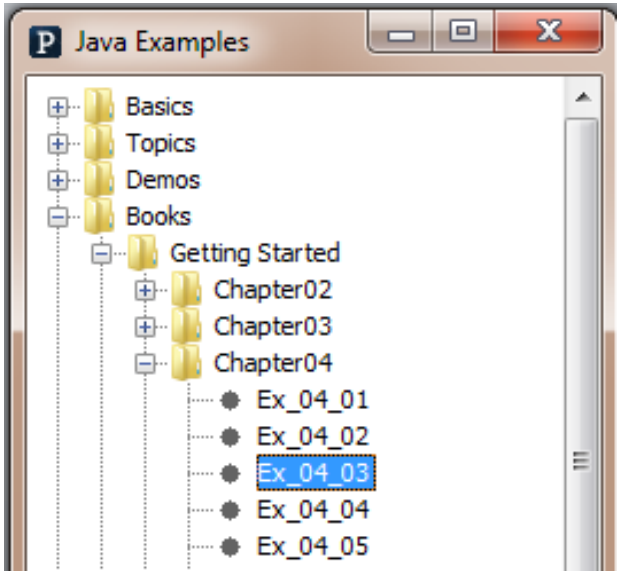
Variables Program

```
size(300, 300); background(0);
stroke(255, 0, 0);// Set the Line Color to Red
strokeWeight(4);  // Set Line Thickness to 4

int a = 50;
int b = 120;
int c = 180;
int distance = 10;

line(a, b, a + c, b);
line(a, b + distance, a + c, b + distance);
line(a, b + 2*distance, a + c, b + 2*distance);
line(a, b + 3*distance, a + c, b + 3*distance);
```



Variables Program

- Running built-in examples
- File->Examples->Contributed Examples->Chapter 04->
- Run Example 4-3 from the Getting Started book. Code on next slide.



Variables Program

```
size(480, 120);
// Line from (0,0) to (480, 120)
line(0, 0, width, height);
// Line from (480, 0) to (0, 120)
line(width, 0, 0, height);
ellipse(width/2, height/2, 60, 60);
```

Refactor the Yin & Yang

- Use variables to refactor your previous sketch such that the Yin and Yang symbol is drawn relative to one fixed point (such as the center of the Yin and Yang)
- Refactoring refers to restructuring a program without changing its basic behavior.

Printing text on console

- To display text use the print line function (println)
  - `println("Hello");`

    -prints a line on the screen with the words `Hello` in it
- To print the value of a variable (e.g. celsius=32), use
  - `println("The temperature is " + celsius);`

    –will print The temperature is 32 on the screen

Conditional statements

- Conditional statements allow conditional execution

```
int temperature = 90;
if (temperature > 85)
{
     println("It is hot outside");
}
else
{
     println("It is pleasant outside");
}
```

Conditional statements

- What will be printed on the screen if temperature=70?

```
int temperature = 70;
if (temperature > 85)
{
     println("It is hot outside");
}
else
{
     println("It is pleasant outside");
}
```

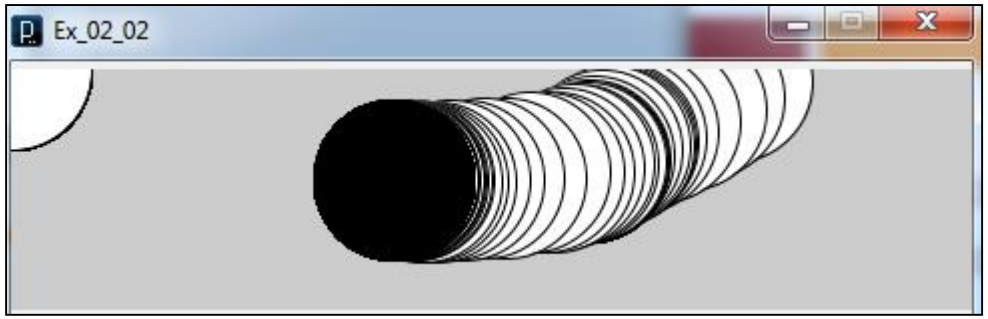
Conditional statements

- Can we add more conditions? Sure!

```
int temperature = 70;
if (temperature > 85)
{
   println("It is hot outside");
}
else if (temperature < 50)
{
   println("It is cold outside");
}
else  // Only when the variable temperature is >50 and <85
{
   println("It is pleasant outside");
}
```

Conditional Statement example

```
void setup() {
  size(480, 120);
  smooth();
}

void draw() {
  if (mousePressed) {
    fill(0, 0, 0);
  }
  else {
    fill(255, 255, 255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```



Example from “Getting Started with Processing Book”

Loops

- To print all the numbers from 1-10 on the screen, you could have 10 consecutive print statements
- What would you do to print 1-1000 numbers?
- Use loops to iterate over a number and call a specific command

```
for (int i=0; i < 10; i = i + 1)
{
  println(i);
}
```

Loops

1. Initialize variable `i` to `0`
2. Increment `i` by `1` until `i<10`
3. For every distinct value of i execute the statements within the curly braces

```
for (int i = 0; i < 10; i = i + 1)
{
  println("The value of i is " + i);
}
```

Loops

- Run Example 4-7 from the Getting started book

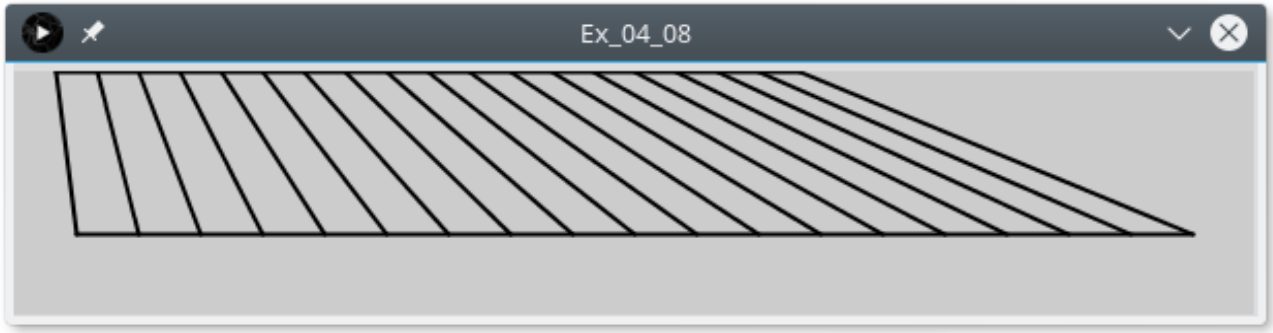
```
size(480, 120);
strokeWeight(2);
for (int i = 20; i < 400; i = i + 8) {
    line(i, 40, i + 60, 80);
}
```

- Run Example 4-8 from the Getting started book (modify the size to 600 x 120 though)

```
size(600, 120);
strokeWeight(2);
for (int i = 20; i < 400; i = i + 20) {
    line(i, 0, i + i/2, 80);
}
```

Loops Exercise 1

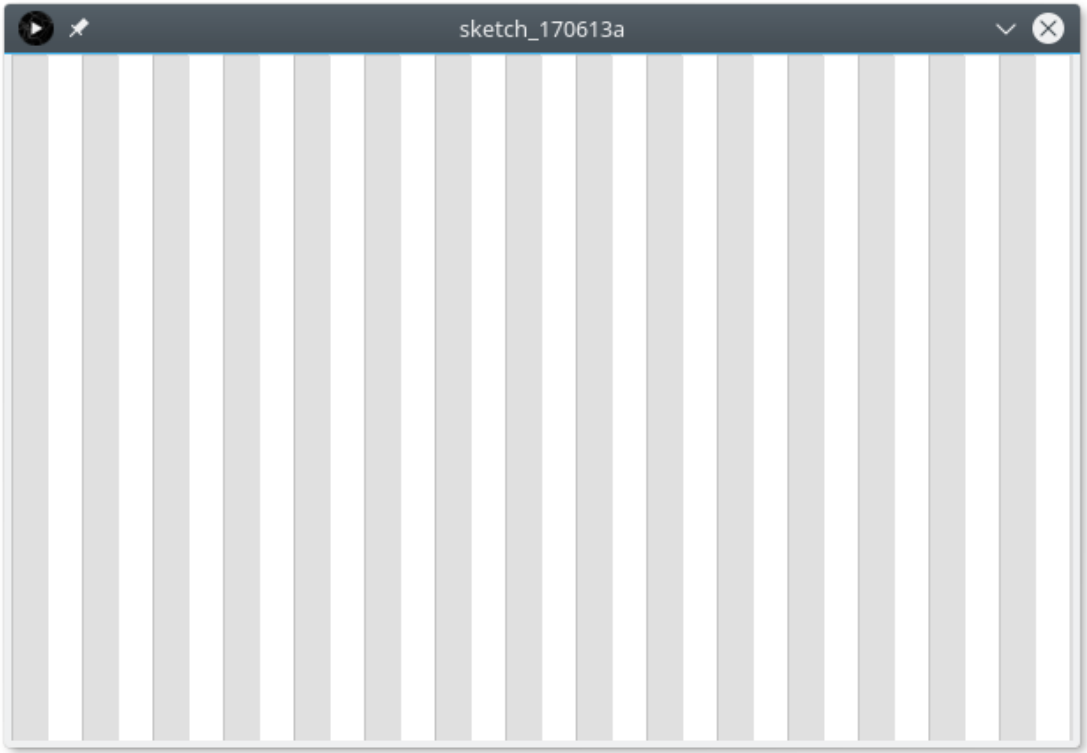
- Starting with code for Example 4-8, draw two horizontal lines that connect the vertical lines on top and bottom to make a *boardwalk*!



- *Hint*: Thin about starting and ending values for the loop counter *i*

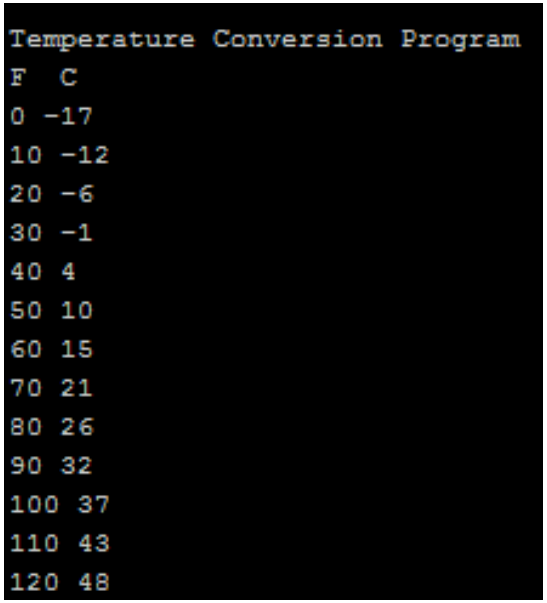
Loops Exercise 2

- Write code using a loop and conditional statement that displays the following pattern.



Fahrenheit to Celsius conversion

```
println("Temperature Conversion Program");
println("F " + " C");
int celsius;
for (int i=0;i < 130; i=i+10)
{
  celsius = (i - 32) * 5/9;
  println(i + " " + celsius);
}
```



Nested Loops

- Run Example 4-10 from the Getting started book
- Compare Example 4-11 with 4-10
  - What is the difference?

Functions/Procedures

- Used to perform generalizable and repeatable tasks

```
void printTemp(int temp)
{
  println("The temperature is a balmy " + temp + " F");
}

printTemp(35);
printTemp(85);
printTemp(120);
```

Output:

```
The temperature is a balmy 35 F
The temperature is a balmy 85 F
The temperature is a balmy 120 F
```

Functions

- Invoking a function with a variable

```
int temperature = 35;
printTemp(temperature);
temperature = 85;
printTemp(temperature);
```

Output:

```
The temperature is a balmy 35 F
The temperature is a balmy 85 F
```

Functions

- Major benefit of defining functions is the flexibility
- Suppose you decide to display the temperature in Celsius instead of Fahrenheit
- Without functions, you would have to change every line that displays the temperature in Fahernheit

```
void printTemp(int temp)
{
  int celsius = (temp - 32) * 5/9;
  println("The temperature in celsius is " + celsius);
}
```

Functions

- Now when you invoke

```
printTemp(35);
printTemp(85);
printTemp(120);
```

- The output is

```
The temperature in celsius is 1
The temperature in celsius is 29
The temperature in celsius is 48
```

Functions example - Calculator

- Writing a program for a simple calculator (+-*/)
- You need functions for addition, subtraction, division and multiplication
- You can then provide two numbers to the appropriate function and it will compute and display the result

```
int add(int a, int b)
{
     int c = a + b;
     println("The result of addition is " + c);
}
```

Functions Exercise 1 (2 minutes)

- Write a function that accepts two integers and displays the value of their product
- Use the add function below for reference

```
int add(int a, int b)
{
    int c = a + b;
    println("The result of addition is " + c);
}
```

- Discuss your solution with your neighbor

Functions Exercise 2

- Start with the *RoobaVariables* example.
- Write a function named *drawRooba* that accepts two integers *centerX* and *centerY* as parameters and displays a *Rooba* at that location! (*Hint*: leave the other variables at the top so they are visible to the whole program)
- Call the function from the draw() method to draw the Rooba.
- Call the *drawRooba*() function twice to display two Roobas at two different locations.
- Write a loop to draw 10 *Roobas*? 100 Roobas?
- Discuss your solution with your neighbor

Built-in Functions - Processing

- line(x1, y1, x2, y2) – draws a line
- println(message) – prints a message to the screen
- ellipse(centerX, centerY, xradius,yradius) - draws an ellipse on the screen with specified parameters
- stroke(255, 0, 0) // Set the line color to Red
- and many more…

RESPONSE IN PROCESSING

Mouse responses

- Mouse actions:
  - mouseX and mouseY are the coordinates for where the mouse is.
  - pmouseX, pmouseY are the coordinates for where the mouse was in the previous frame
  - mousePressed: boolean variable that is true if the mouse is pressed, false otherwise
  - mouseButton: LEFT, CENTER or RIGHT
- Examples: 5-4, 5-5, 5-6, 5-7, 5-12
- Examples: MouseDemo

Key responses

- Key actions:
  - keyPressed: boolean variable that is true if a key is pressed, false otherwise
  - key: the char value of key pressed
  - keyCoded: true if key is coded and not tied to a specific key. For example: ALT, CONTROL, SHIFT, UP, DOWN, LEFT, RIGHT
  - mouseButton: LEFT, CENTER or RIGHT
- Examples: 5-18, 5-19, 5-21

ANIMATION

Animation

- Requires redrawing
- setup() function is invoked only once – at the beginning of the program
- draw() function is invoked repeatedly – to simulate animation

Animation

```
void setup() {
  size(640, 480);
  smooth();
  noStroke();
}

void draw() {
  fill(mouseX, mouseY, 0);
  ellipse(mouseX, mouseY, 30, 30);
}
```

Animation – Speed and Direction

- Run Example 8-3 from the Getting started book
- Run Example 8-4 from the Getting started book
- Modify Example 8-4 to make the *pacman* bounce back and forth between left and right walls
- Modify pacman to flip its mouth when it bounces back from a wall

Animated Rooba

- **Version 1:** Change the example *RoobaVariables* so that Rooba moves left to right and bounces back along the X axis
- **Version 2:** Change the example *RoobaVariables* so that Rooba moves left to right and bounces back along the Y axis
- **Version 3:** Now have Rooba move in both X and Y directions

Animated Yin Yang

- **Version 1**: Change the sketch size to be 800 x 400 and animate the Yin Yang to move left to right until it touches the right edge. Then it should reverse and move right to left until it touches the left edge. Then it continues to move back and forth.
- **Version 2**: Make the Yin Yang roll along its center! You will need to use trigonometric functions sine and cosine to calculate the coordinates of the center of the small circles. The Processing functions are named sin() and cos() and they take angles in radians. To use degrees, use the radians() function. For example: sin(radians(180));
- **Version 3**: Make it bounce! (more difficult)

Animated Sine Wave

- Examples->Basics->Math->SineWave
- Examples->Basics->Math->PolarToCartesian

Loading Images

- For loading images, use the image() function
- The image must be stored in a folder named data under the sketch folder. See example *LoadPicture*

```
PImage img;

void setup() {
  size(300, 300);
  img = loadImage("beach.jpg");
}

void draw() {
  image(img, 0, 0);
  image(img, 0, 0, width/2, height/2);
}
```

Loading and Playing Sound

- Add sound library. *Examples* -> *Add Examples* → *Libraries* and look for Sound library from the Processing Foundation
- Reference for the sound library is found at this website: https://processing.org/reference/libraries/sound/index.html

```
//See this example under SoundDemo1 folder in class examples
import processing.sound.*;
SoundFile file;
void setup() {
  size(640, 360);
  background(255);
  // Load a sound file from the data folder of the sketch and play it back
  file = new SoundFile(this, "sample.mp3");
  file.play();
}
void draw() {  }
```

A FOR ARRAYS!
A FOR AWESOME!

Store the price of an item

- To store the price of an item you would use a variable
- To store the price of two items you would use two variables
- To store the price of three items you would use three variables

- To store the price of 17 items, ….

15% Discount on all items

- Let us assume that you want to give a 15% discount on every item in the store
- If A = 100, after discount
  - A = A – (15 / 100 * A) = 85
- Find every variable (x) and perform
  - x = x – (15 / 100 * x)
  - y = y – (15 / 100 * y)
  - z = z – (15 / 100 * z) and so on…

Find the average item price

- Find every variable
- Find their price
- Compute the sum
- Keep a track of the total number of variables/items
- Compute the average = sum/num of vars

- What if you forgot to include one variable??

Arrays to the rescue

- `int [] price = {60, 20, 40, 10};`

- To sum all the prices, use a loop

```
int sum = 0;
for (int i = 0; i < price.length; i++){
    sum = sum + price[i];
}
println(sum);
```

15% discount

```
for (int i = 0; i < price.length; i++){
   price[i] = price[i] - (15 / 100 * price[i]);
}
```

Average of prices

- Print the average of all the prices in the prices array

- Refer to the **sum** code snippet

```
int [] price = {60, 20, 40, 10};
int sum = 0;
for (int i = 0; i < price.length; i++){
     sum = sum + price[i];
}
float avg = sum / price.length;
// Compute the average
println("The avg is " + avg);
```

Arrays Exercise

- Write code to calculate the minimum price?
- How would you calculate the max?
- How would you calculate min and max at the same time?

Arrays and Animation

- See example *CircleArray*
- See example *RandomCircleArray*
- Discuss Rainfall Simulation project