Processing Introduction

CSE 120 Spring 2017

Instructor: Justin Hsia

Teaching Assistants: Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

69 New Emojis Announced, Including Breastfeeding Woman, Woman with Headscarf, and Steak

The newest emoji update may be the most inclusive yet, featuring a woman wearing a head covering, a breastfeeding woman, and also both steak and broccoli. These are just a few of the 69 individual symbols with varying skin tone and gender offered in shadowy technology organization the Unicode Consortium's 10th update, which includes Emoji 5.0.

There will also be dinosaurs, dumplings, cussing, and hedgehogs. Oh, and this barfing emoji:

- https://creators.vice.com/en_us/article/69-new-emojis-breastfeeding-woman-headscarf-steak

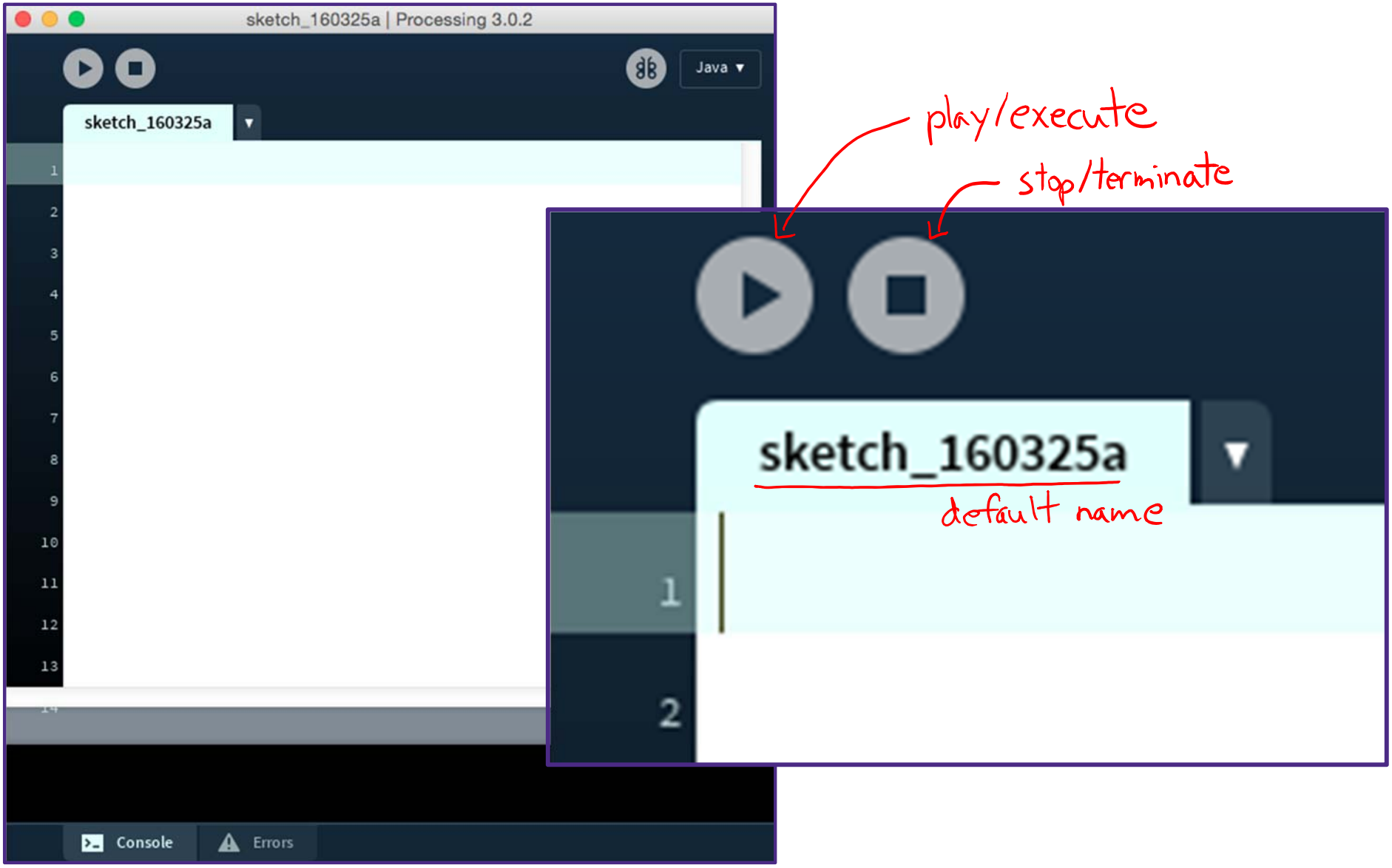
Administrivia

- Assignments:
  - Lightbot Functions due today (4/3)
  - Building a Robot due tomorrow (4/4)
  - Taijitu due Wednesday (4/5)
- No “big ideas” lecture this week
  - More time on programming

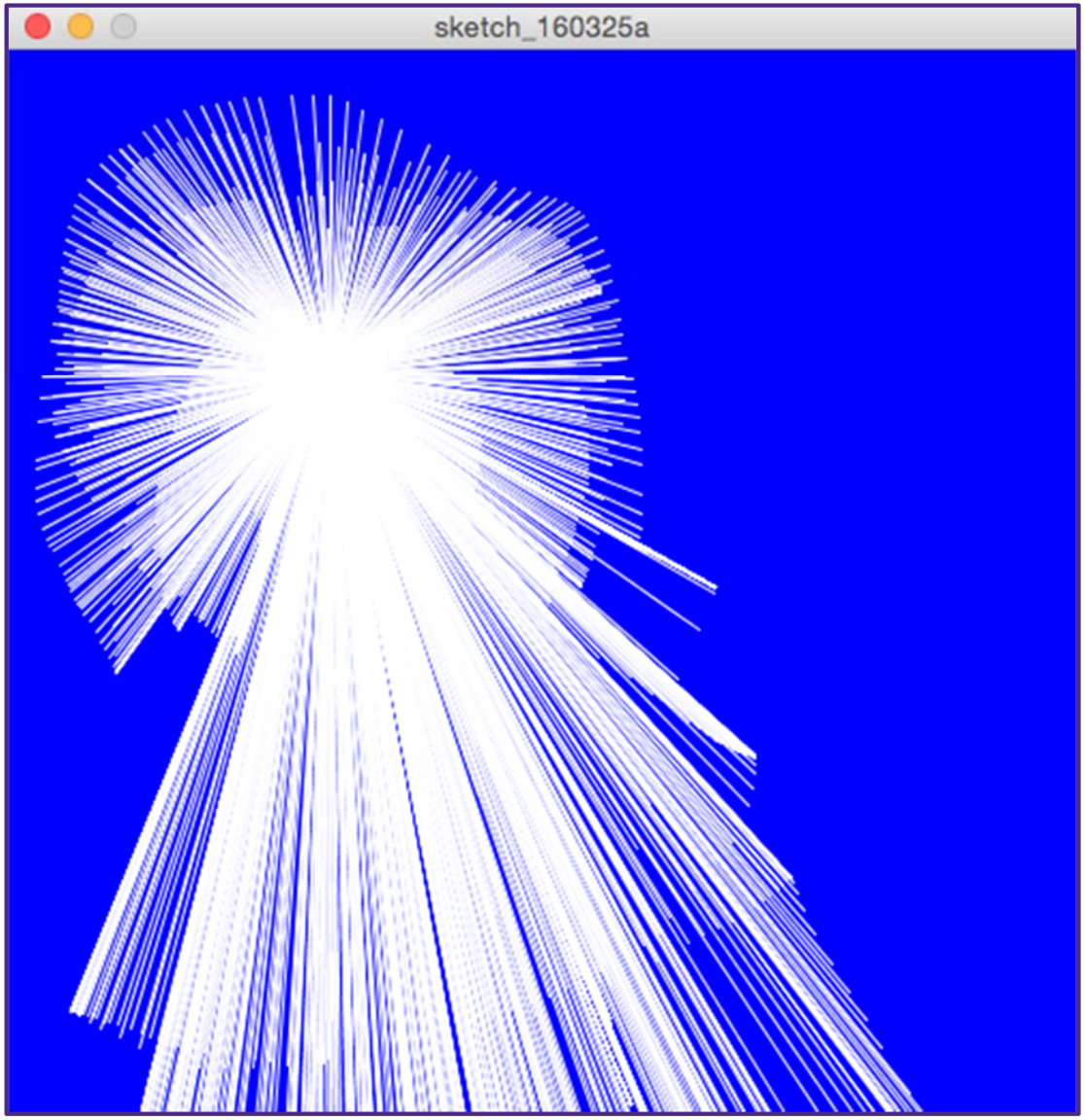
Processing

- Our programming language for this course
  - Text-based language that is good for visuals and interaction
  - Try to focus on ideas and techniques, not the specific commands
  - No language is perfect – Processing has its fair share of quirks and deficiencies ☹
- It is both a programming *environment* (where you type) and a programming *language*
  - You are writing Java code, but they have made a lot of things easier

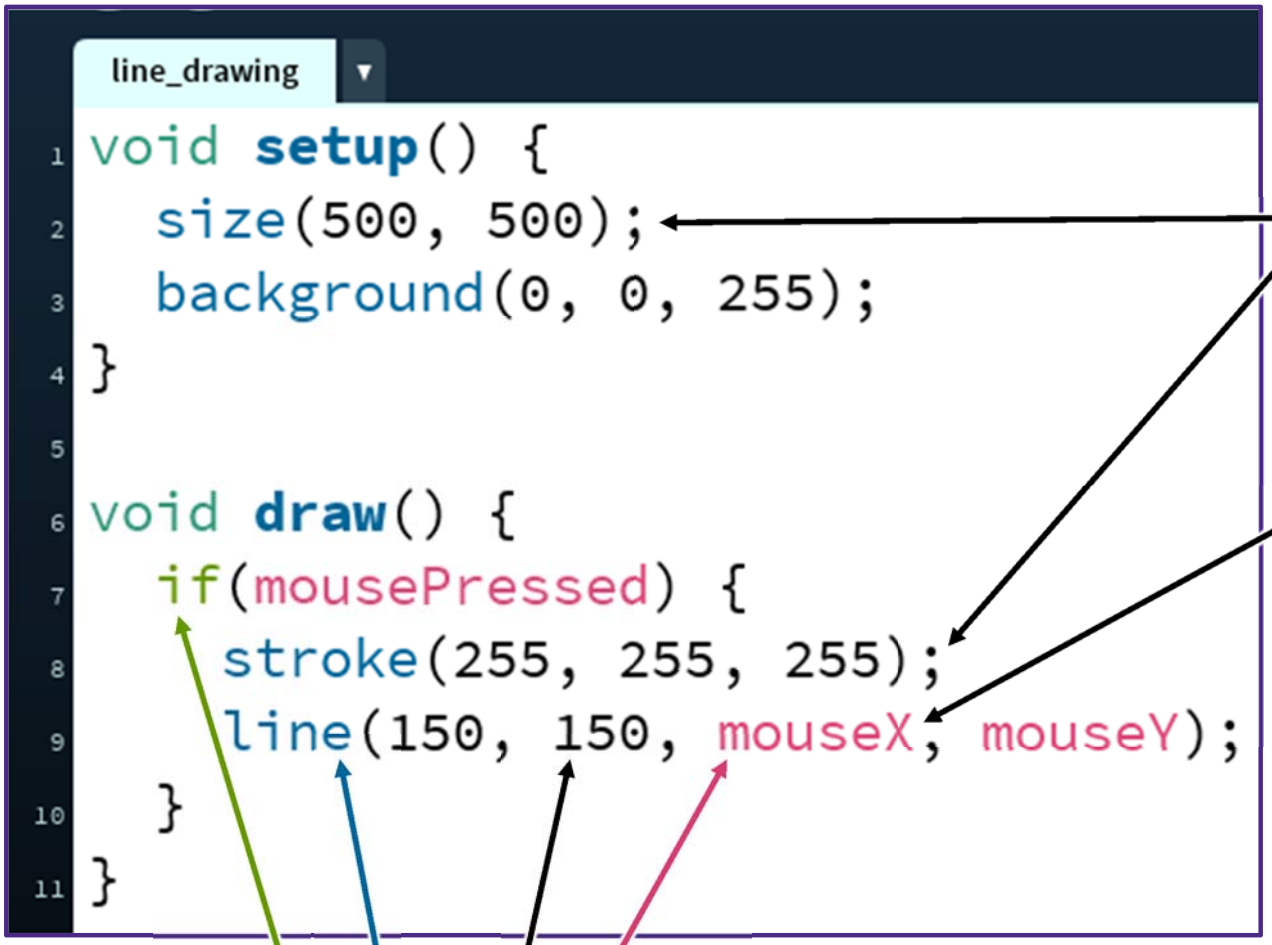
What You See


sketch_160325a | Processing 3.0.2
Java
sketch_160325a
Console
Errors
play/execute
stop/terminate
sketch_160325a
default name
1
2

Interactive Line Drawing



Line Drawing Code



semi-colon indicates end of statement

case-sensitive
mouseX ≠ mousex

There is color coding

Other helpful *environment* features:

- Parentheses matching
- Error messages

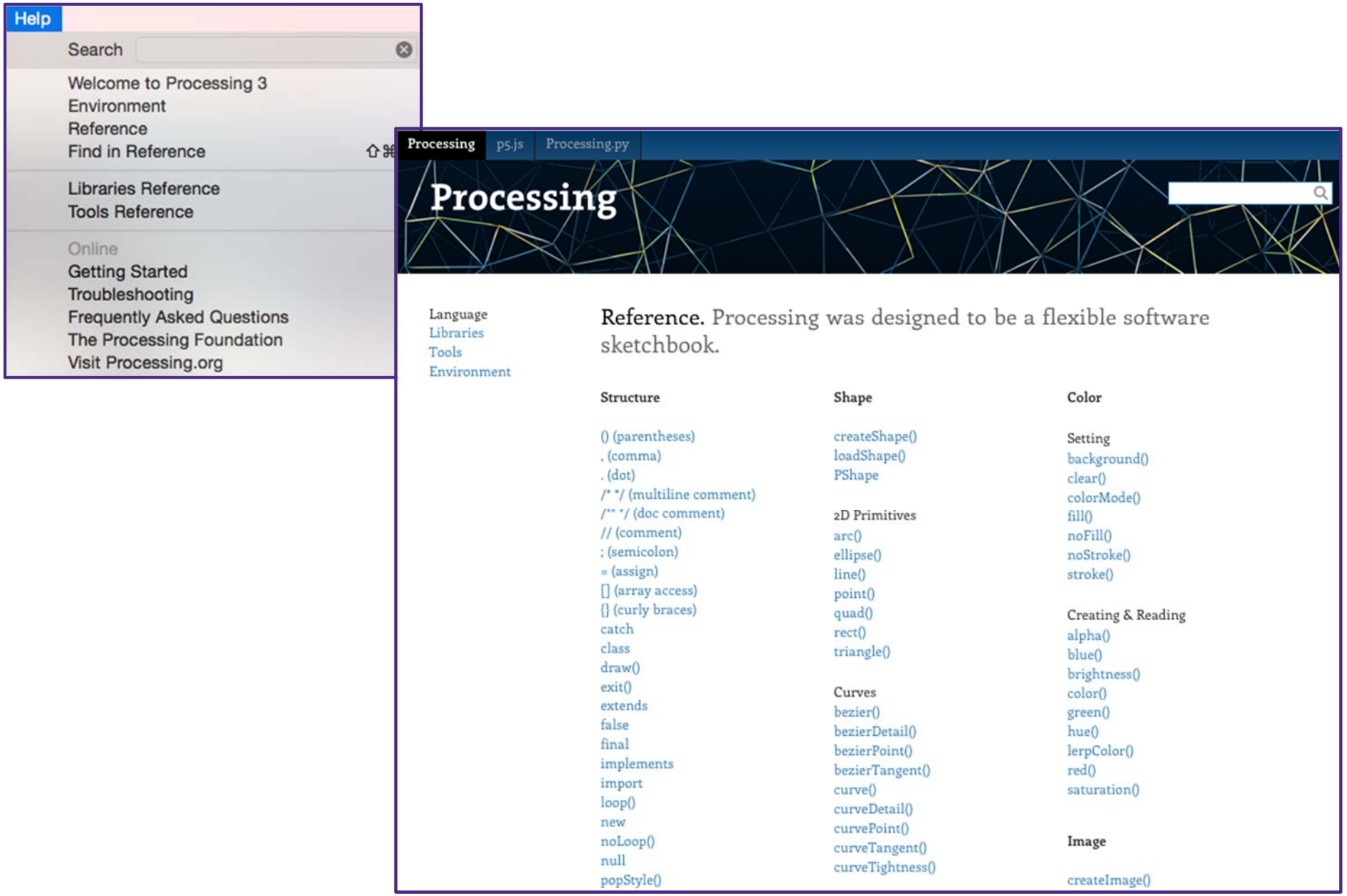
Comments Are Critical!!!

block (multi-line) comment

line_drawing

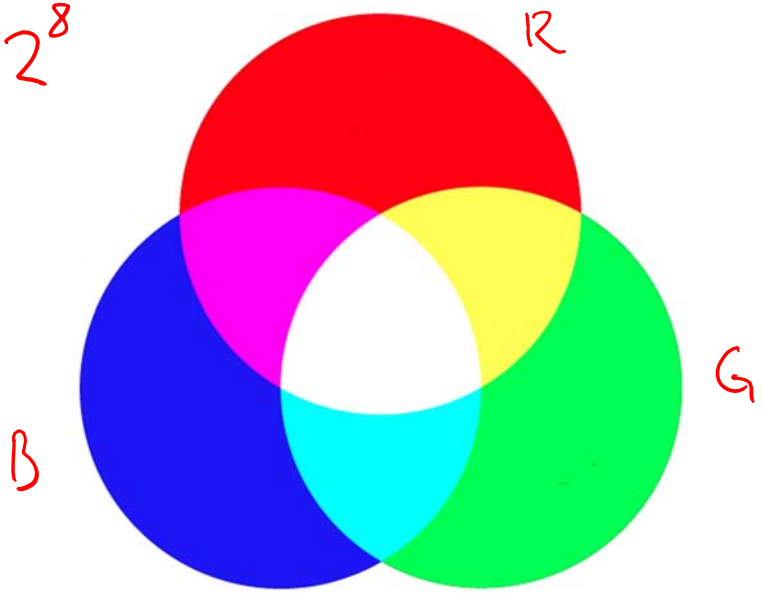
```
/* line_drawing.pde                                          ← file name
   Edited by Justin Hsia (orig. Larry Synder)                ← your name

   Draws a line to mouse position when user presses mouse.   ← brief program description
*/

// setup() is a function that runs once at beginning of program   ← brief function description
void setup() {
  size(500,500);                    // set drawing canvas size to 500x500
  background(200,200,255);          // sets background color to light blue
}                                   ↑ statement description

// draw() is a function that runs continuously over and over again
void draw() {
  if(mousePressed) {                // if user presses the mouse
    stroke(255, 255, 255);          // set line color to white
    line(150, 150, mouseX, mouseY); // draw line from (150,150) to mouse position
  }                                 ↑ single-line comment
}
```

The Processing Reference

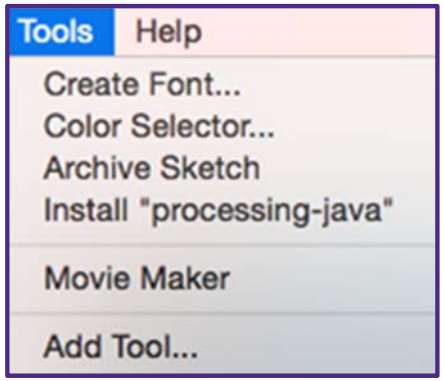


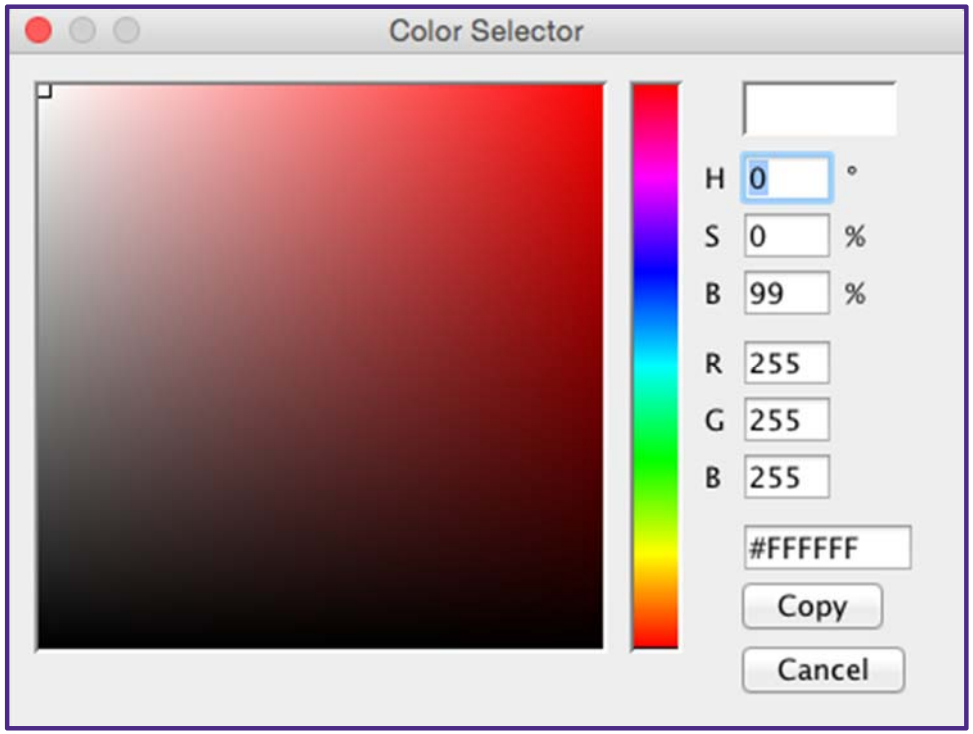
Understanding Color

- In electronic systems, color specified using the RGB color model
  - **R**ed, **G**reen, **B**lue
- Each pixel on your screen is made up of 3 tiny lights, one red, one green, one blue
- Specify the intensity of each light using an integer between 0 and 255 → $256 = 2^8$ (8 bits)
  - 0 is completely off
  - 255 is highest intensity



Processing's Color Selector





Guess the Color

- color(R, G, B);
- color(255, 0, 0);
- color(0, 255, 0);
- color(0, 0, 255);
- color(0, 0, 0);
- color(255, 255, 255);
- color(255, 255, 0);
- color(255, 0, 255);
- color(0, 255, 255);

Guess the Color

```
color(  R,   G,   B);
color(255,   0,   0); // R fully on
color(  0, 255,   0); // G fully on
color(  0,   0, 255); // B fully on
color(  0,   0,   0); // all off
color(255, 255, 255); // all fully on
color(255, 255,   0); // R,G fully on
color(255,   0, 255); // R,B fully on
color(  0, 255, 255); // G,B fully on
```

Guess the Color

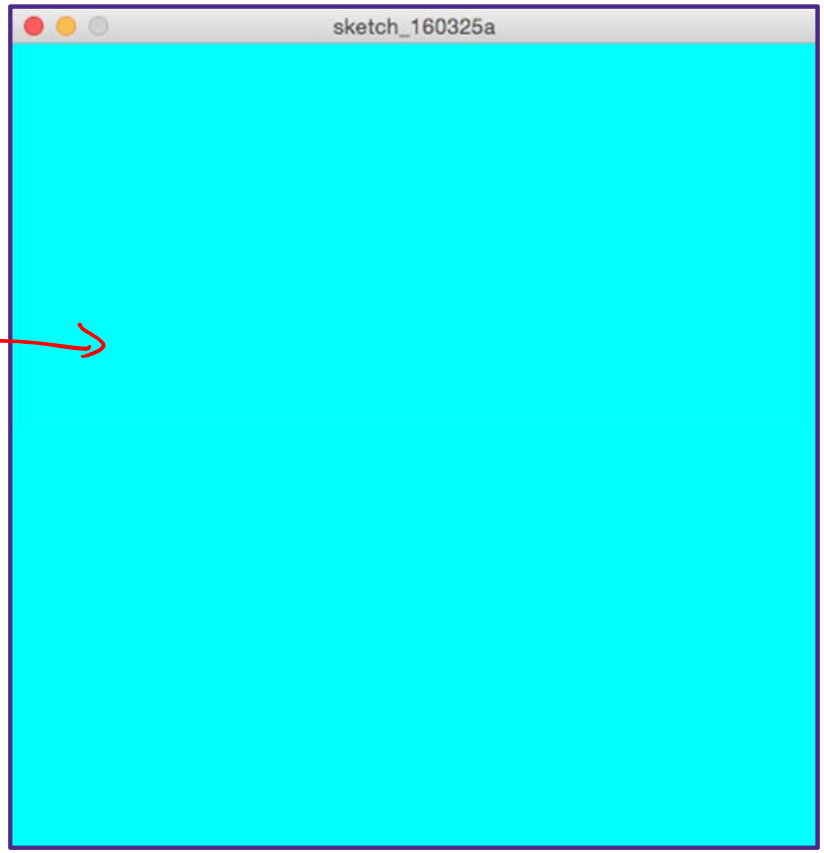
- `color(  R,   G,   B);`
- `color(255,   0,   0); // red`
- `color(  0, 255,   0); // green`
- `color(  0,   0, 255); // blue`
- `color(  0,   0,   0); // black`
- `color(255, 255, 255); // white`
- `color(255, 255,   0); // yellow`
- `color(255,   0, 255); // magenta`
- `color(  0, 255, 255); // cyan`

Color Functions

- `background(R, G, B);`
  - Sets the background color of the drawing canvas

```
sketch_160325a
1 void setup() {
2   size(500, 500);
3   background(0, 255, 255);
4 }
```

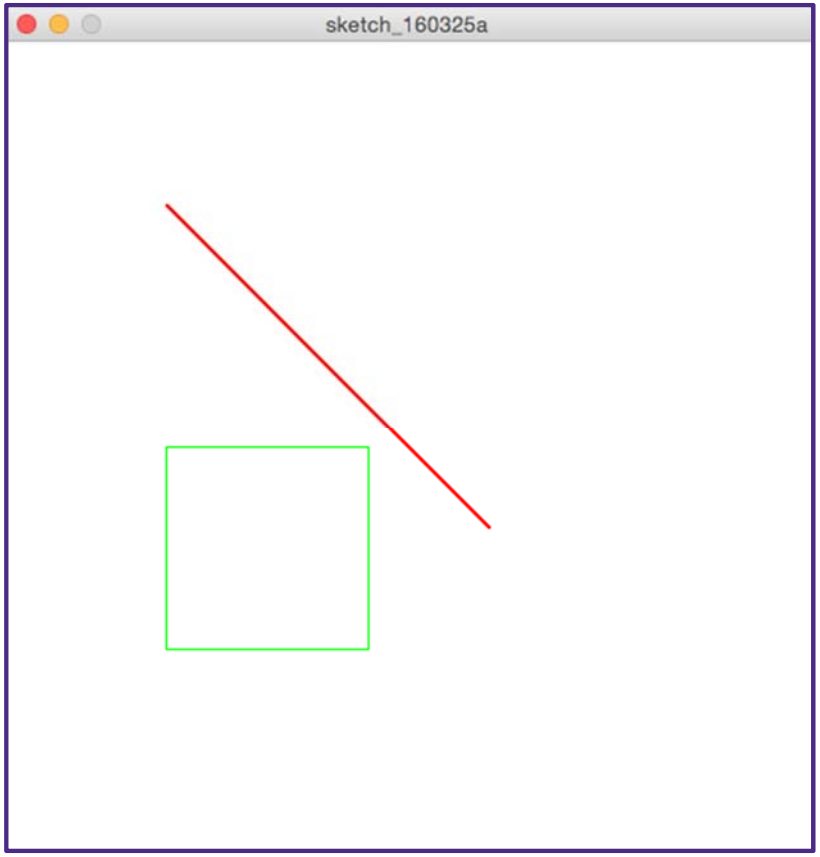
cyan



Color Functions

- `stroke(R, G, B);`
  - Sets the color of the stroke of a *line* or *line around a shape*
  - Can change line size using `strokeWeight(#);`

```
sketch_160325a
1  void setup() {
2    size(500, 500);
3    background(255, 255, 255);
4  }
5
6  void draw() {
7    stroke(255, 0, 0); //red
8    line(100, 100, 300, 300);
9
10   stroke(0, 255, 0); //green
11   rect(100, 250, 125, 125);
12 }
```



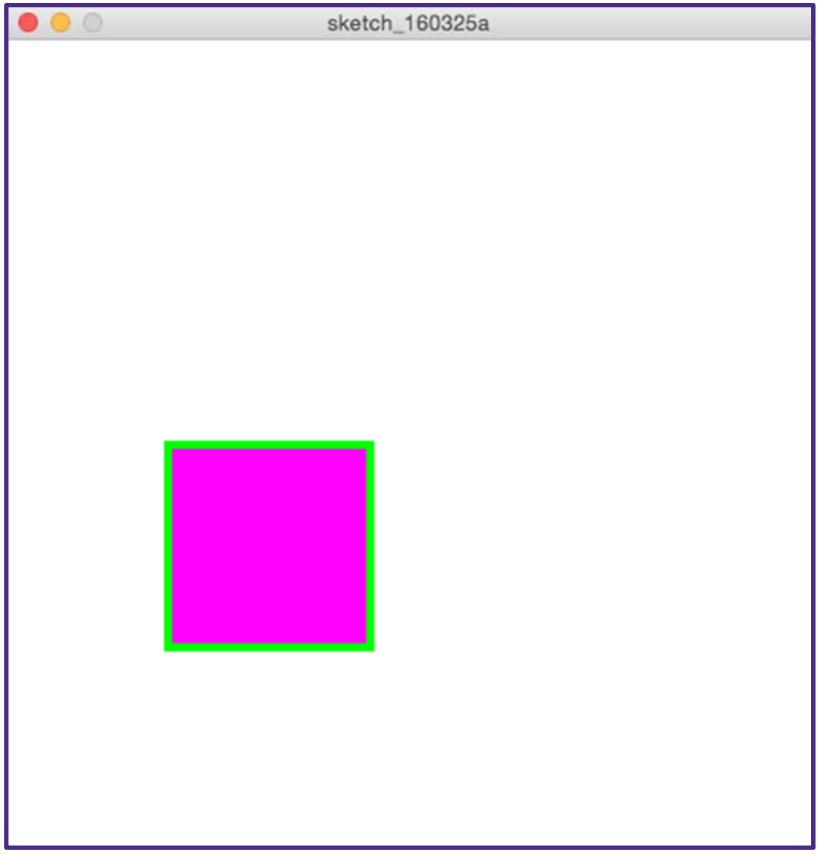
Color Functions

- `fill(R, G, B);`
  - Sets the *inside* color of a shape (**note:** you cannot fill a line)

sketch_160325a

```
1  void setup() {
2    size(500, 500);
3    background(255, 255, 255);
4  }
5
6  void draw() {
7    strokeWeight(5);
8    stroke(0, 255, 0);
9    fill(255, 0, 255); //magenta
10   rect(100, 250, 125, 125);
11 }
```

make line thicker



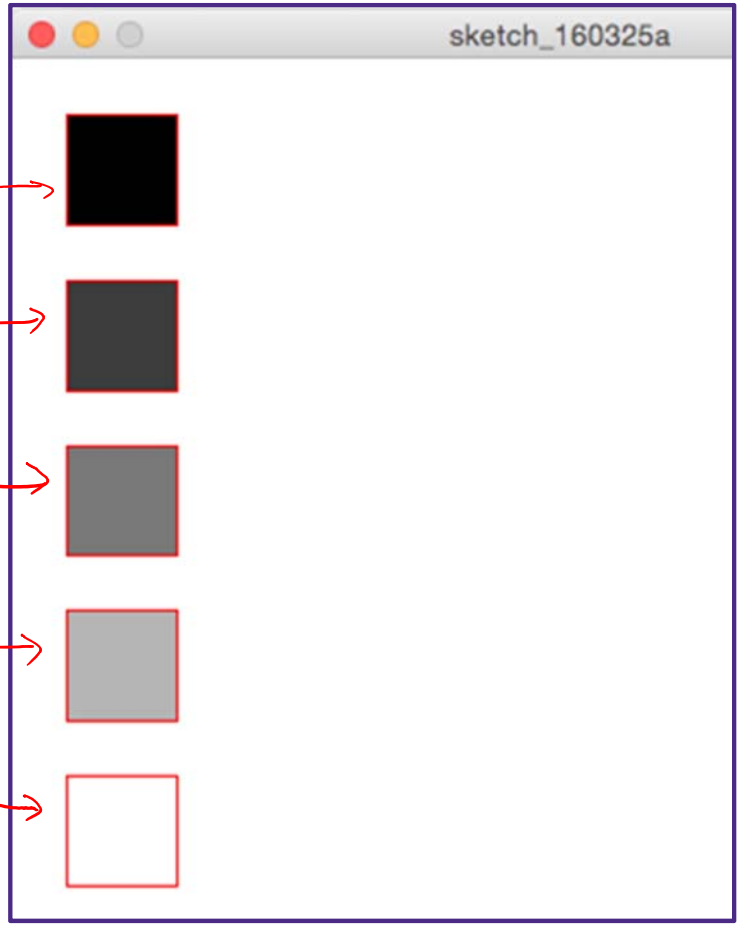
Color: "Grays"

- When the values for RGB are all the same, then the color will be white, black, or some shade of gray

darker (closer to black)

lighter (closer to white)

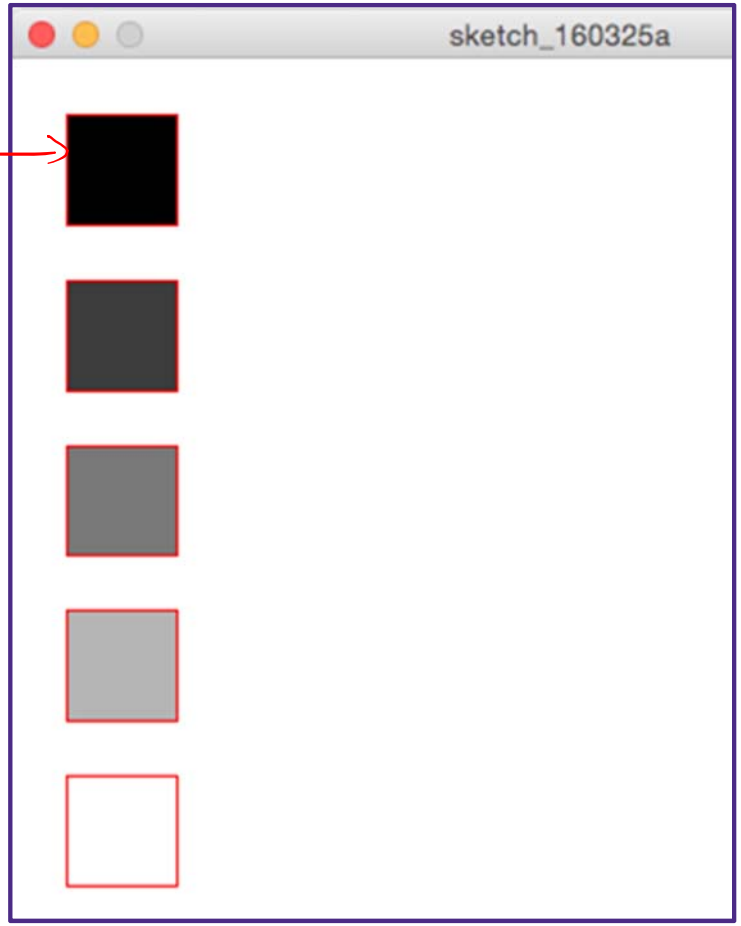
```
6  void draw() {
7    stroke(255, 0, 0);
8
9    fill(0, 0, 0);
10   rect(25, 25, 50, 50);
11
12   fill(60, 60, 60);
13   rect(25, 100, 50, 50);
14
15   fill(120, 120, 120);
16   rect(25, 175, 50, 50);
17
18   fill(180, 180, 180);
19   rect(25, 250, 50, 50);
20
21   fill(255, 255, 255);
22   rect(25, 325, 50, 50);
23 }
```



Color: "Grays"

- When the values for RGB are all the same, then the color will be white, black, or some shade of gray
  - For brevity, can specify just a single number instead

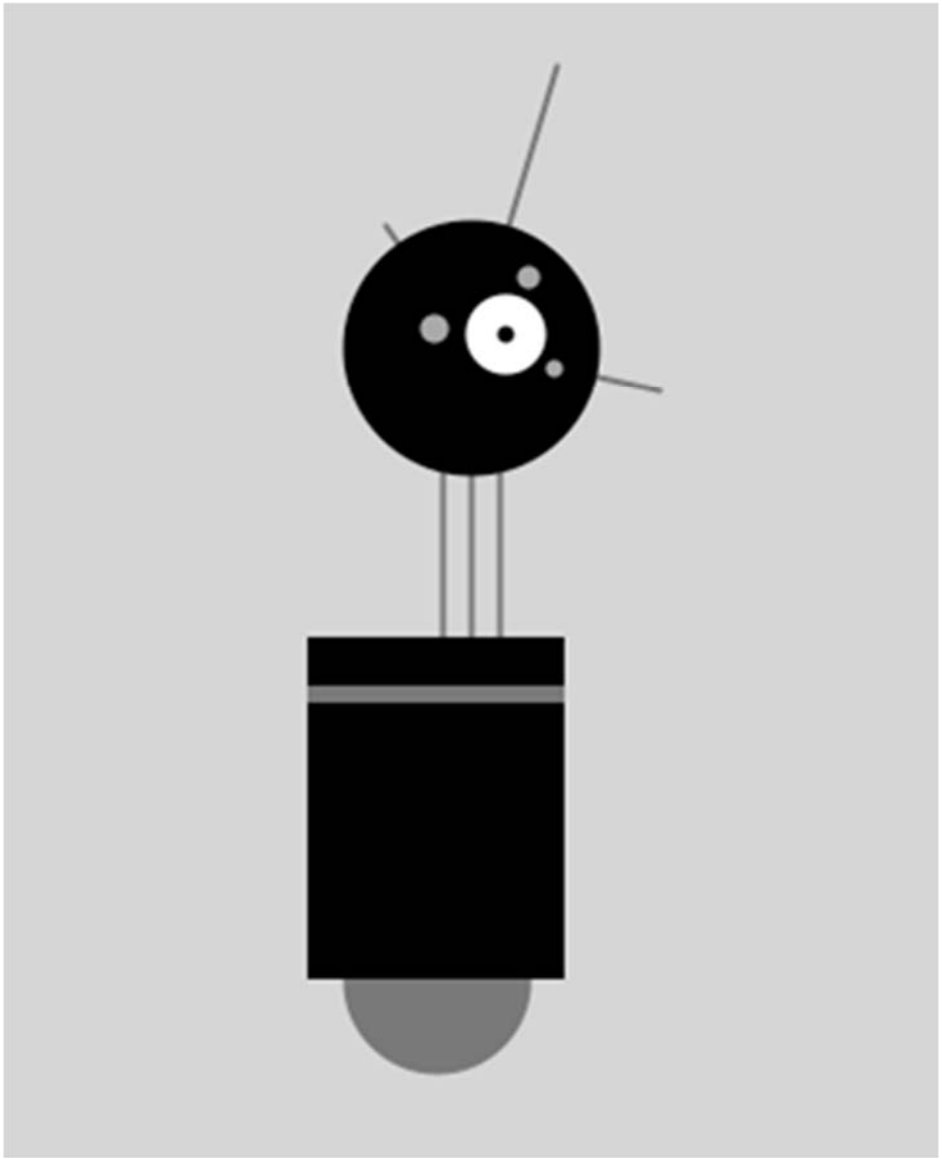
```
6  void draw() {
7    stroke(255, 0, 0);
8
9    fill(0);
10   rect(25, 25, 50, 50);
11
12   fill(60);
13   rect(25, 100, 50, 50);
14
15   fill(120);
16   rect(25, 175, 50, 50);
17
18   fill(180);
19   rect(25, 250, 50, 50);
20
21   fill(255);
22   rect(25, 325, 50, 50);
23 }
```



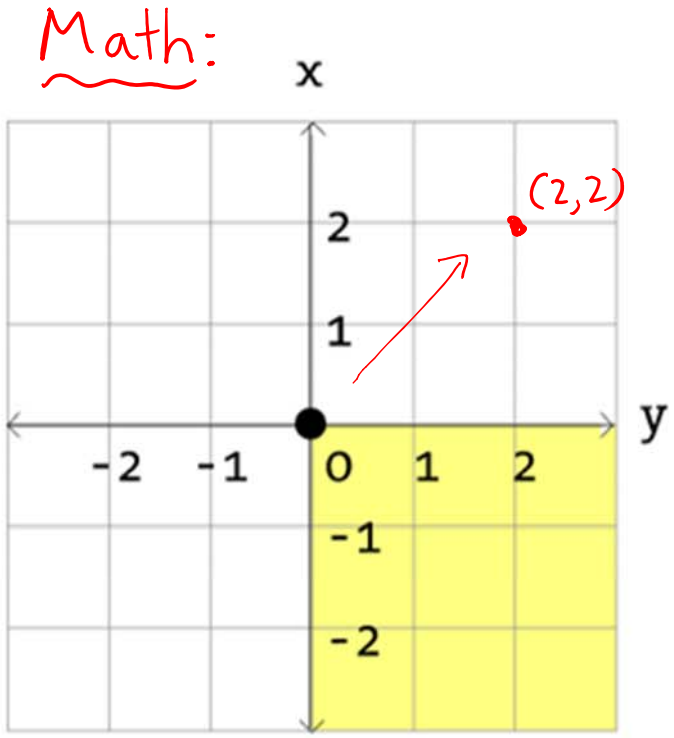
The Color "State" of Your Program

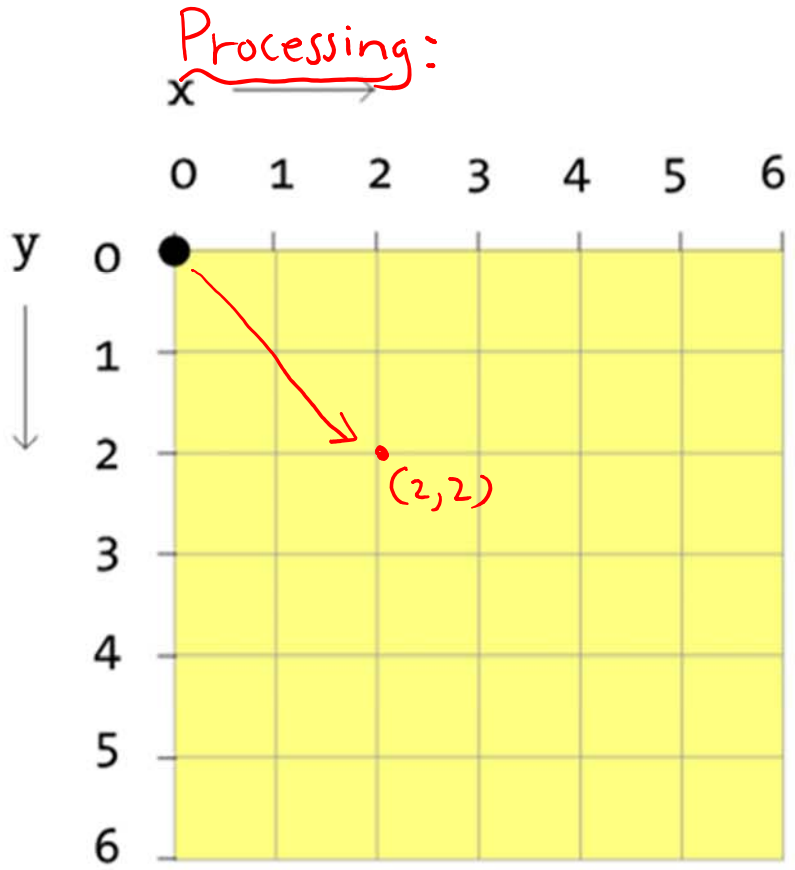
- Recall that programs are executed sequentially (*i.e.* instruction-by-instruction)
- `background()`, `stroke()`, and `fill()` apply to *all* subsequent drawing statements
  - Until a later call overrides
- Hidden color "state" that knows the current values of `background()`, `stroke()`, and `fill()`
  - In complex programs, can be difficult to keep track of
  - Early rule of thumb: always explicitly set colors before each drawing element

Assignment: Coloring a Robot

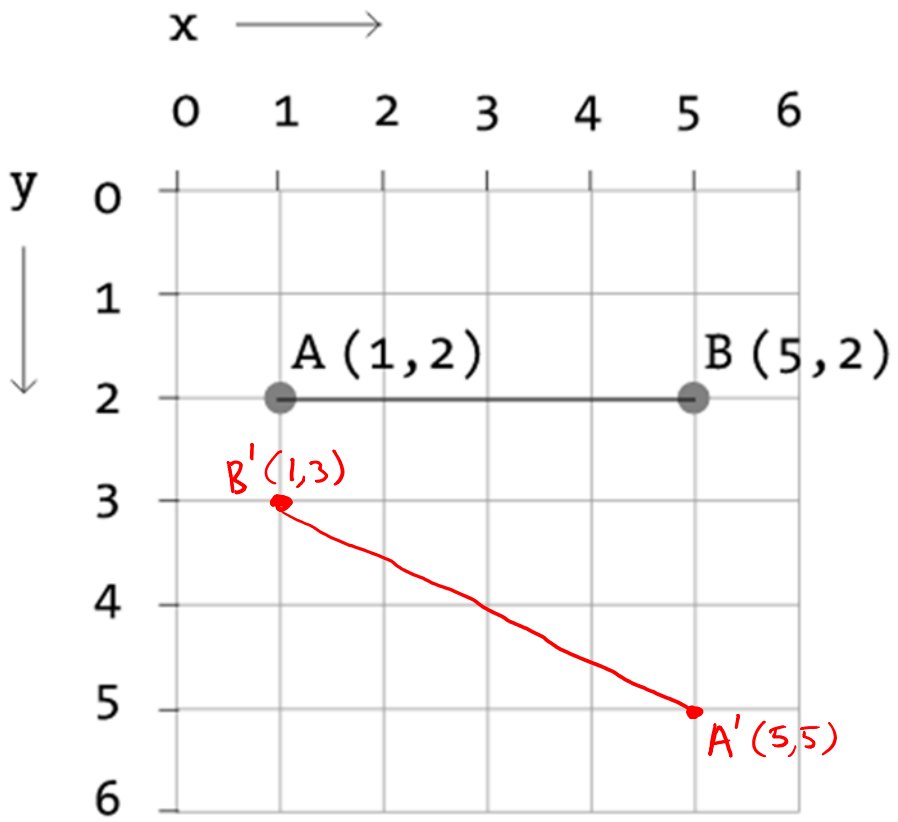


Coordinate System





Drawing: Line



line (x1 , y1 , x2 , y2) ;

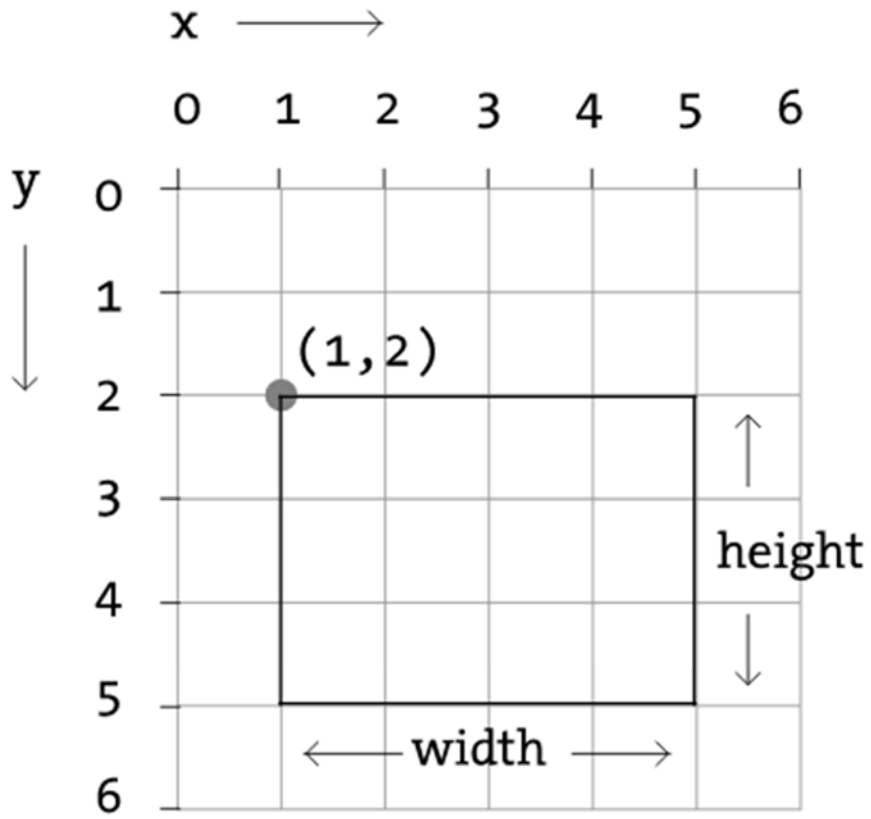
Point A Point B

Example: line (1,2,5,2) ;

line(5,5,1,3);

Drawing: Rectangle

- Default *mode* is CORNER (upper-left always)

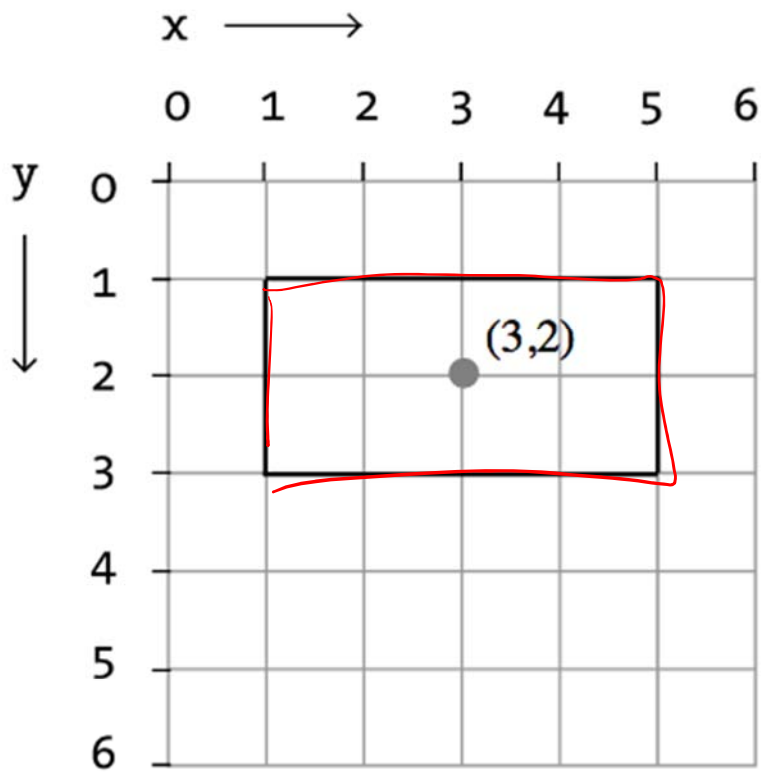


Example: `rect(1,2,4,3);`

(x, y, width, height)

Drawing: Additional Rect Modes

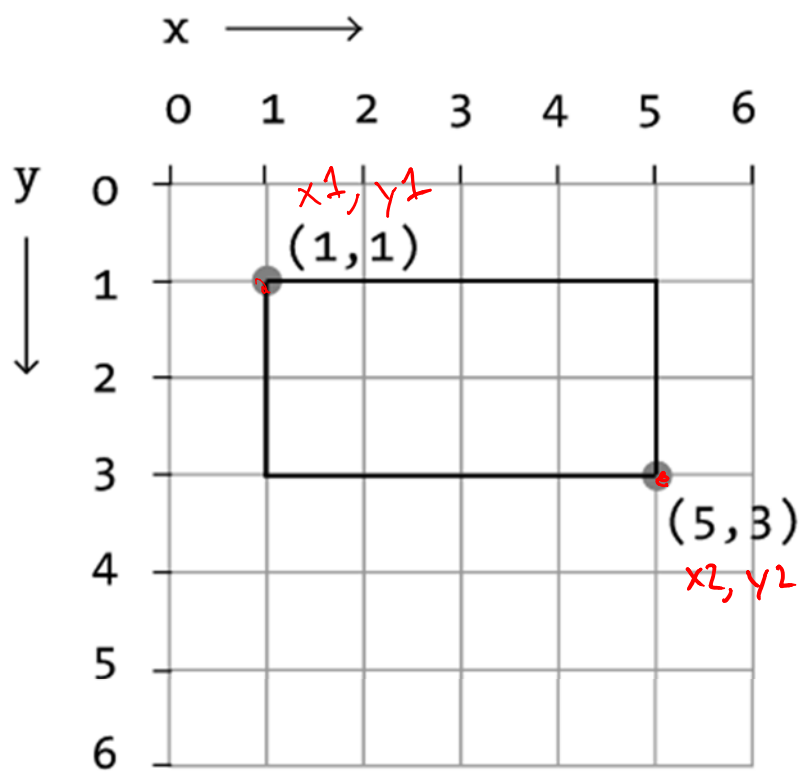
- CENTER




```
Example: rectMode(CENTER);
         rect(3,2,4,2);
```

(x, y, width, height)

- CORNERS (upper-left, lower-right)

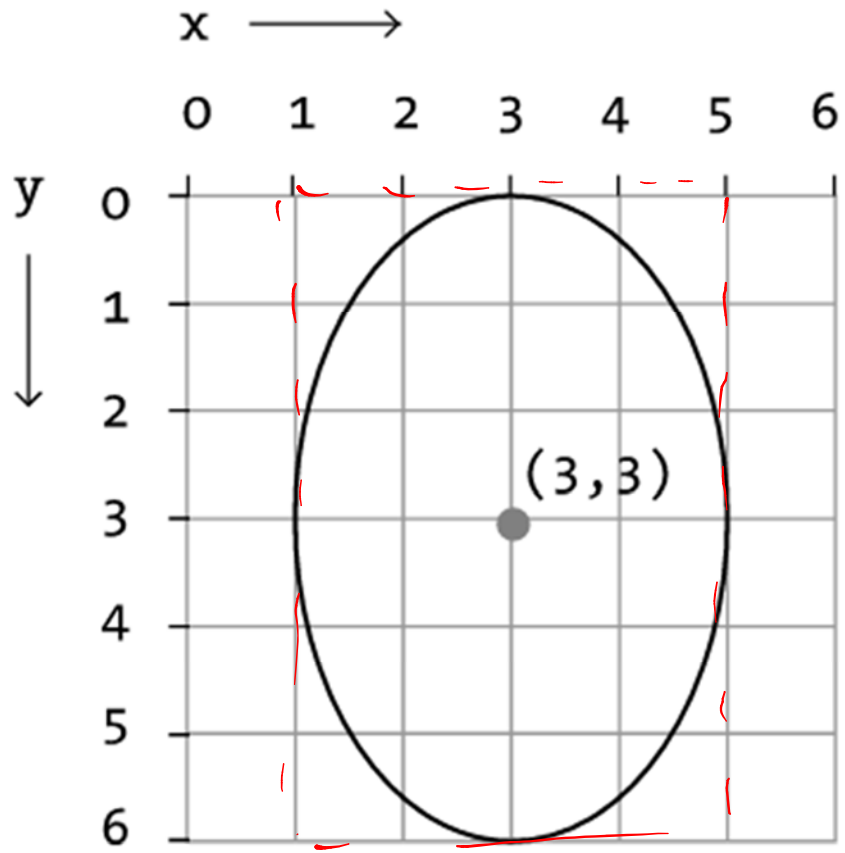



```
Example: rectMode(CORNERS);
         rect(1,1,5,3);
```

(x1, y1, x2, y2)

Drawing: Ellipse/Circle

- Default *mode* is CENTER

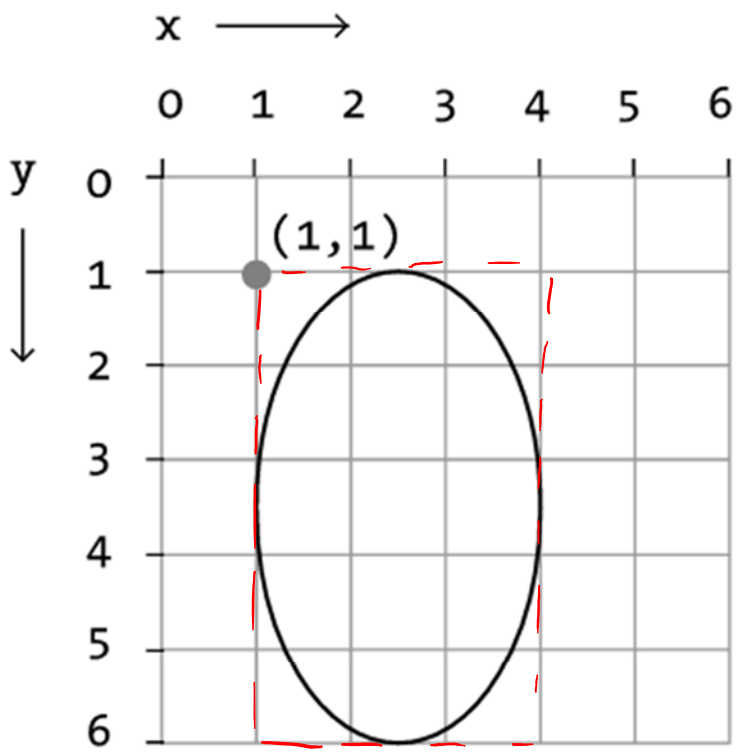


Example: `ellipse(3,3,4,6);`

(x, y, width, height)

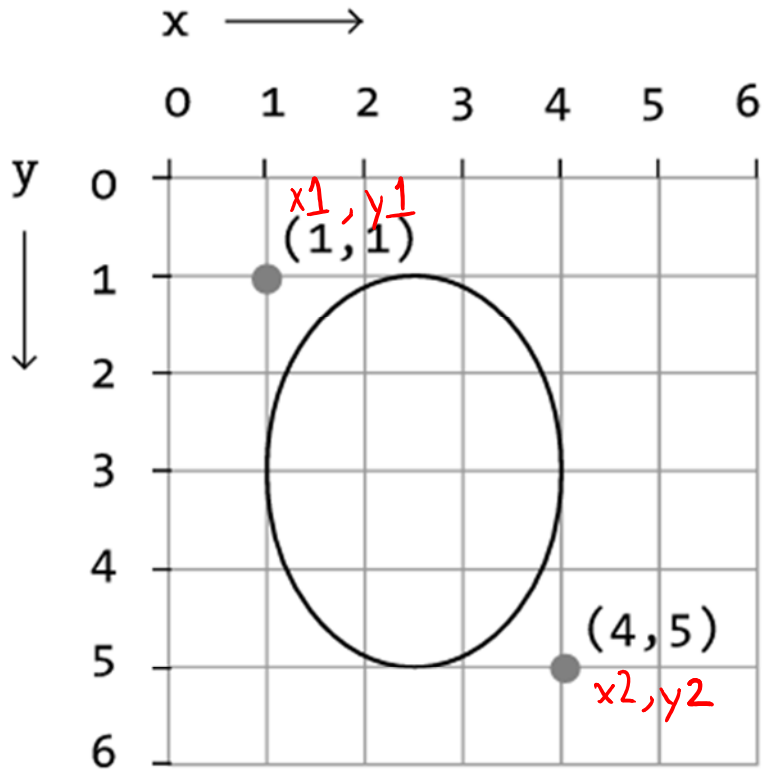
Drawing: Additional Ellipse Modes

- CORNER



Example: ellipseMode (CORNER) ;
ellipse (1 , 1 , 3 , 5) ;
(x, y, width, height)

- CORNERS



Example: ellipseMode (CORNERS) ;
ellipse (1 , 1 , 4 , 5) ;
(x1, y1, x2, y2)

Peer Instruction Question

- Which of the following drawings corresponds to the Processing code below?
  - Vote at http://PollEv.com/justinh

```
strokeWeight(10);
stroke(75, 47, 131);              // UW purple (line)
fill(183, 165, 122);              // UW gold  (inside)
ellipse(100, 100, 100, 200); // CENTER mode
                                  taller
```

A. 

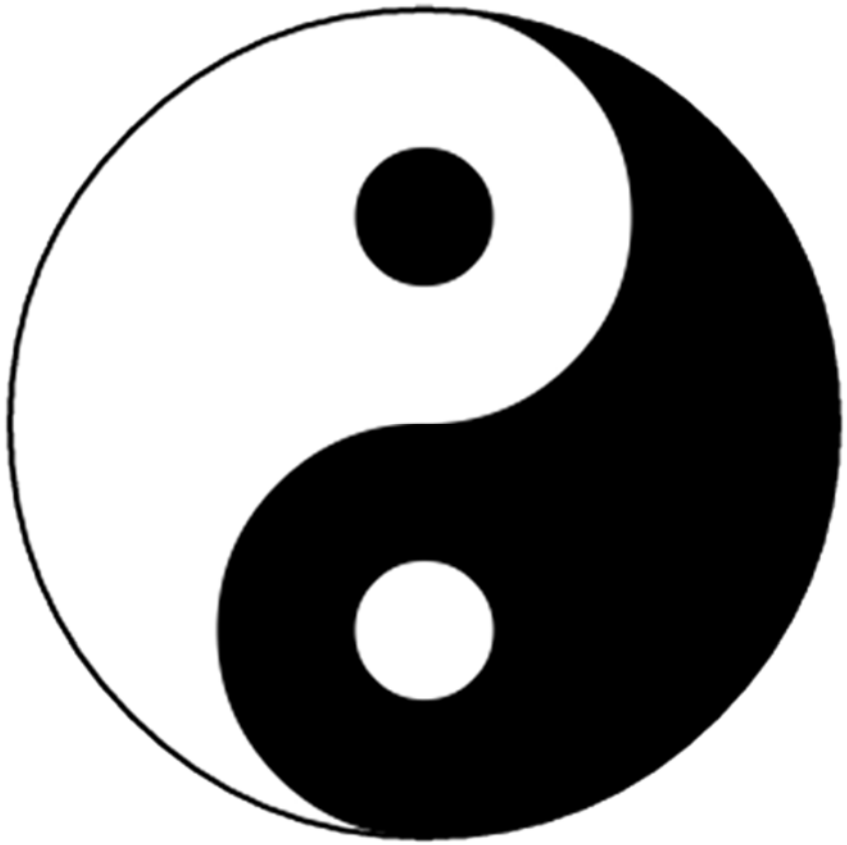
B. 

C. 

D. 

Lab: Taijitu

- How do you build a complex drawing out of these simple shapes?



Aside: Processing Files

- Processing files have extension .pde
  - File names *cannot* contain dashes (-)
- To run a Processing file, it *must* be in a folder of the same name
  - If it's not, then Processing will create the folder for you

— #
robot.pde
robot-2.pde
↑ get rid of