Processing and Drawing

CSE 120 Winter 2018

Instructor:

Justin Hsia

Teaching Assistants:

Anupam Gupta, Cheng Ni, Eugene Oh,
Sam Wolfson, Sophie Tian, Teagan Horkan

The World Health Organization Identifies Gaming Disorder as a Mental Health Condition

"The WHO's impending beta draft… classifies gaming disorder as a pattern of behavior with 'impaired control over gaming,' in terms of its frequency, intensity, duration, and the capacity to quit. The disorder… is characterized by giving increased priority to gaming over other daily activities.

"The WHO's decision highlights a schism among psychologists: some think the new designation is a welcome one, but others don't see enough evidence to justify it.

"As our video game experience expands with virtual reality (VR) and augmented reality (AR), the argument gets even murkier."

- https://futurism.com/world-health-organization-identifies-gaming-disorder-mental-health-condition/

Administrivia

- Assignments:
  - Lightbot Functions [hw] due today *before* 11:59 pm (1/8)
  - Taijitu [lab] due before lab on Thursday (1/11)

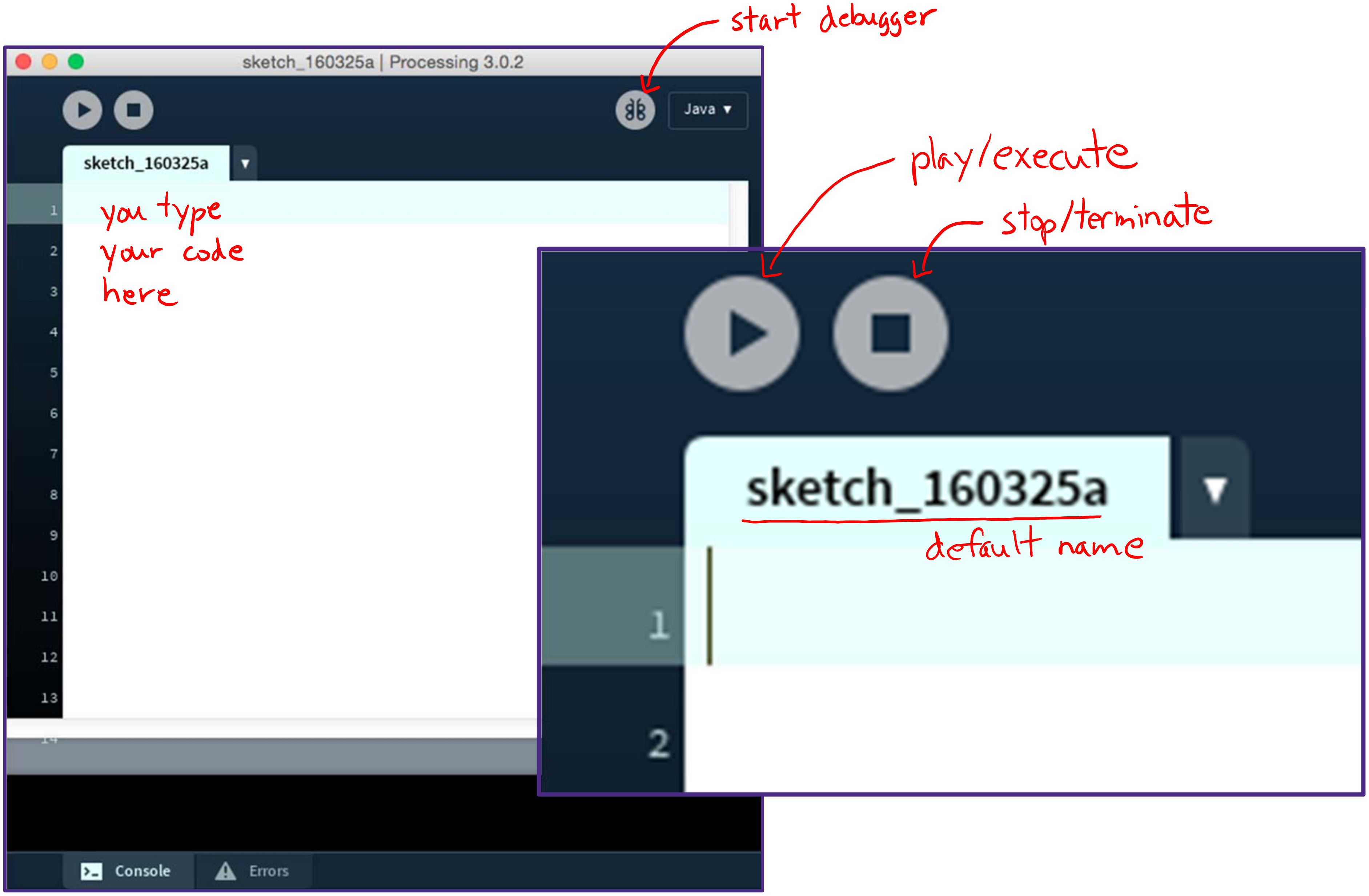
- First "big ideas" lecture this week: Binary
  - Reading due before lab on Thursday (1/11)
  - Brief discussion in lab on Thursday

Processing

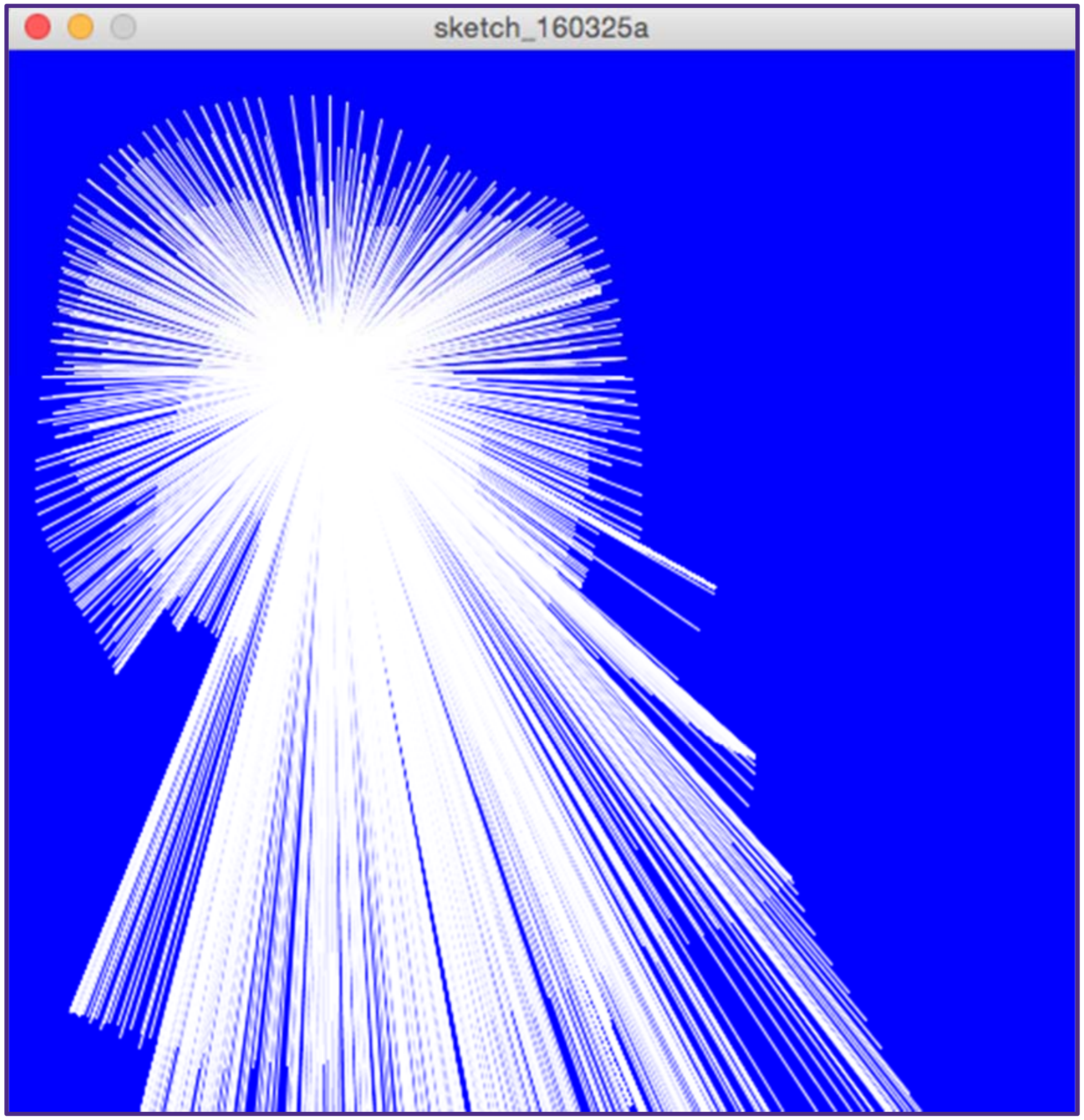
- Our programming language for this course
  - Text-based language that is good for visuals and interaction
  - Try to focus on ideas and techniques, not the specific commands
  - No language is perfect – Processing has its fair share of quirks and deficiencies ☹

- 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

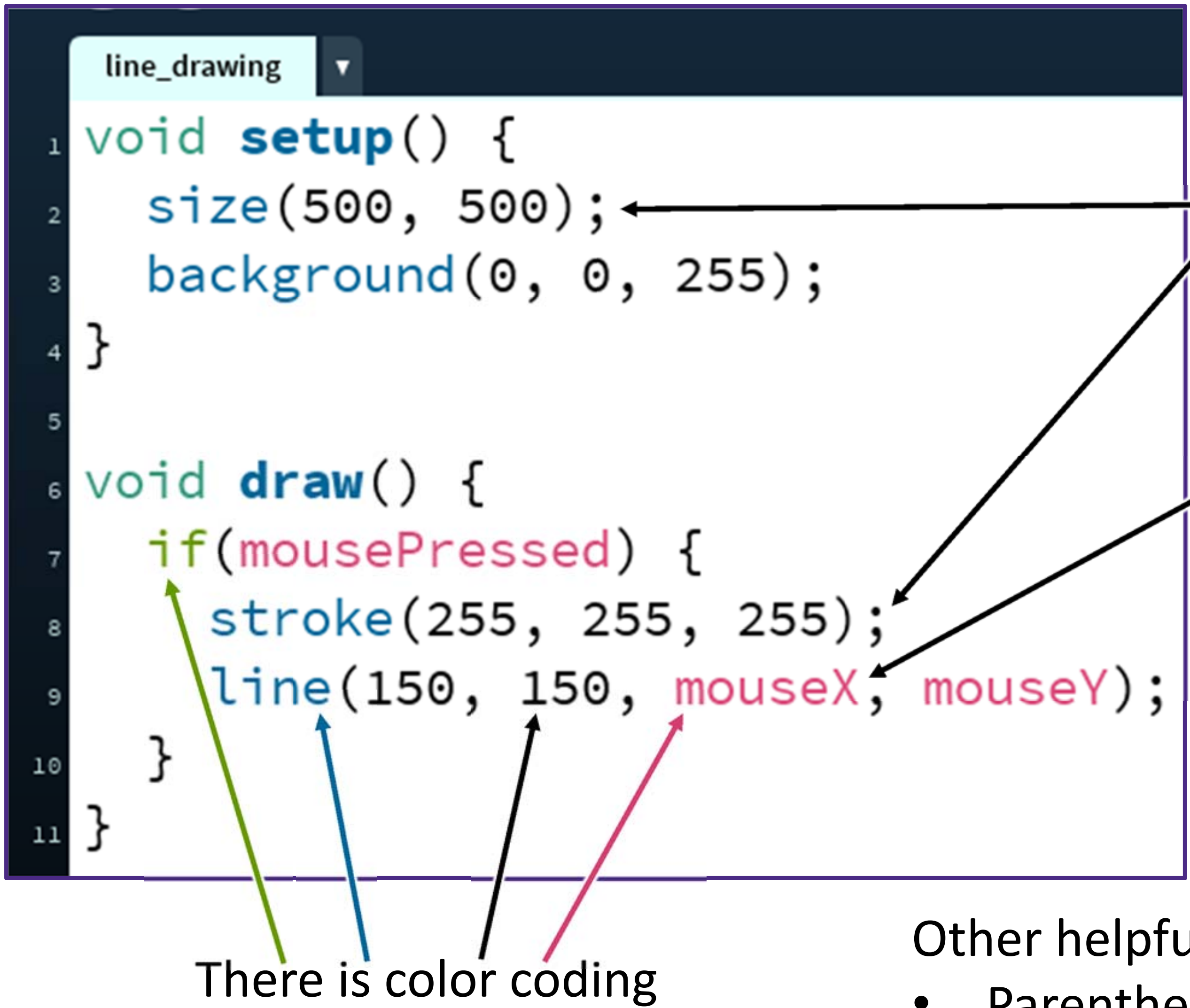
What You See


start debugger
sketch_160325a | Processing 3.0.2
Java
sketch_160325a
you type your code here
play/execute
stop/terminate
sketch_160325a
default name
Console
Errors

Interactive Line Drawing



Line Drawing Code



```
void setup() {
  size(500, 500);
  background(0, 0, 255);
}

void draw() {
  if(mousePressed) {
    stroke(255, 255, 255);
    line(150, 150, mouseX, mouseY);
  }
}
```

semi-colon indicates end of statement

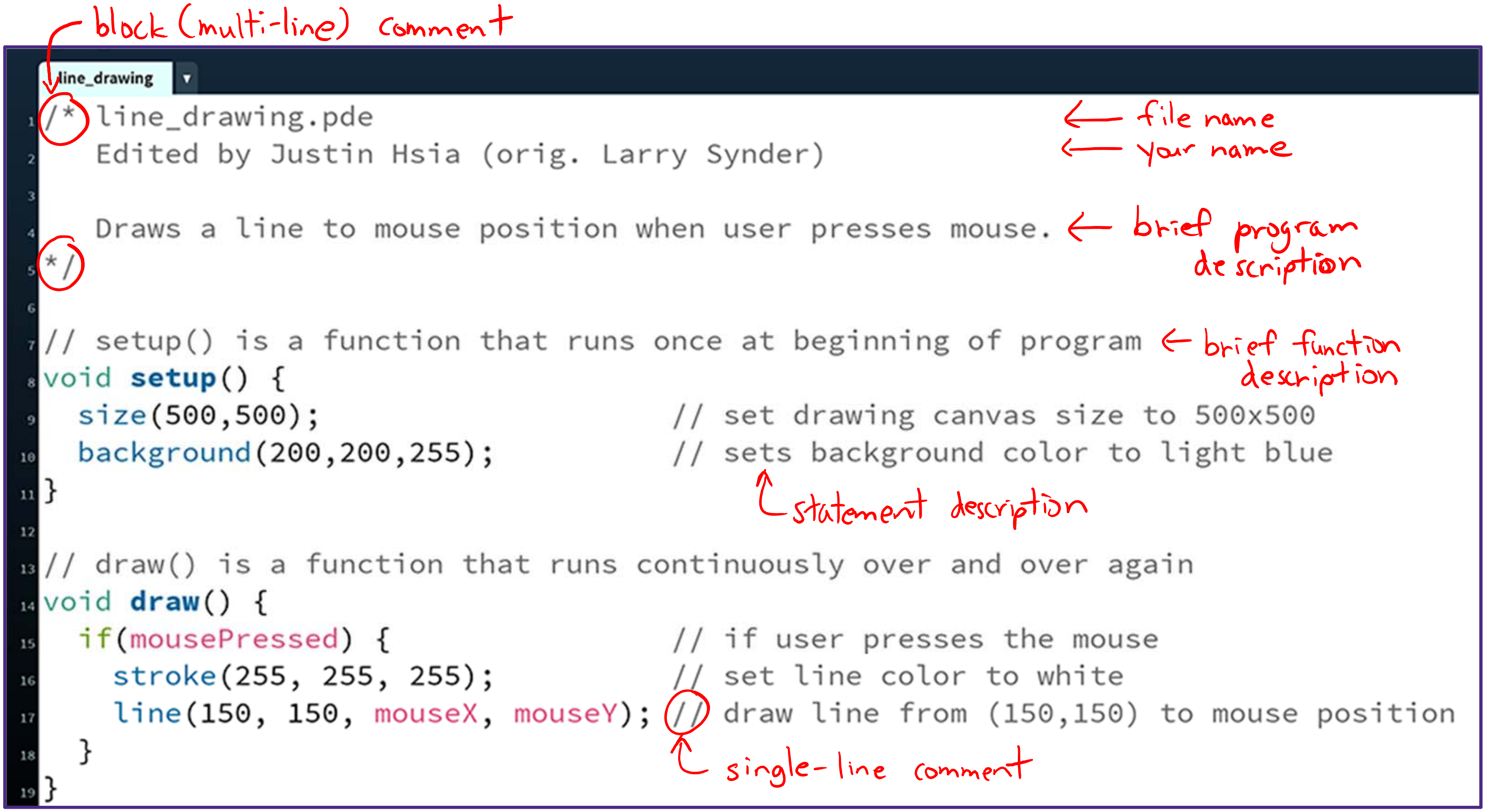
case-sensitive
mouseX ≠ mousex

There is color coding

Other helpful *environment* features:

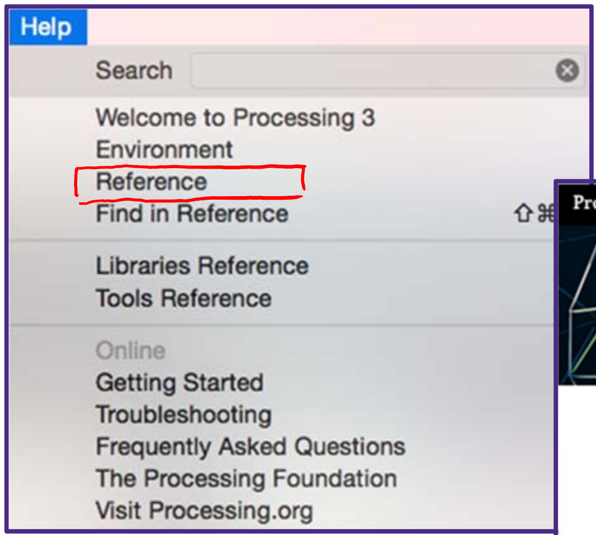
- Parentheses matching
- Error messages

Comments Are Critical!!!


block (multi-line) comment
line_drawing
file name
your name
brief program description
brief function description
statement description
single-line comment

```
/* line_drawing.pde
   Edited by Justin Hsia (orig. Larry Synder)

   Draws a line to mouse position when user presses mouse.
*/

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

// 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
  }
}
```

The Processing Reference



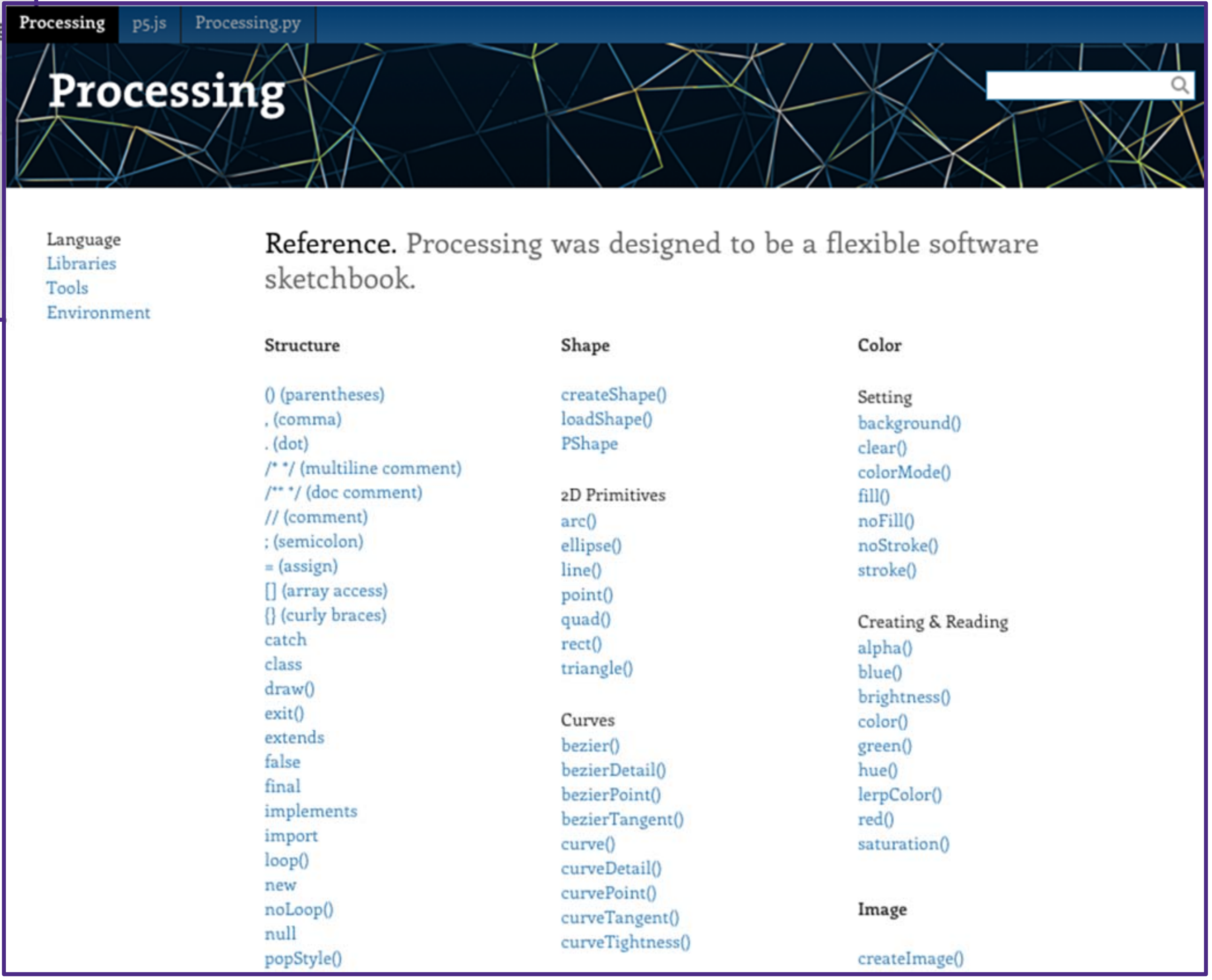
Help

Search

Welcome to Processing 3
Environment
Reference
Find in Reference

Libraries Reference
Tools Reference

Online
Getting Started
Troubleshooting
Frequently Asked Questions
The Processing Foundation
Visit Processing.org



Processing | p5.js | Processing.py

Processing

Language
Libraries
Tools
Environment

Reference. Processing was designed to be a flexible software sketchbook.

Structure

() (parentheses)
, (comma)
. (dot)
/* */ (multiline comment)
/** */ (doc comment)
// (comment)
; (semicolon)
= (assign)
[] (array access)
{} (curly braces)
catch
class
draw()
exit()
extends
false
final
implements
import
loop()
new
noLoop()
null
popStyle()

Shape

createShape()
loadShape()
PShape

2D Primitives
arc()
ellipse()
line()
point()
quad()
rect()
triangle()

Curves
bezier()
bezierDetail()
bezierPoint()
bezierTangent()
curve()
curveDetail()
curvePoint()
curveTangent()
curveTightness()

Color

Setting
background()
clear()
colorMode()
fill()
noFill()
noStroke()
stroke()

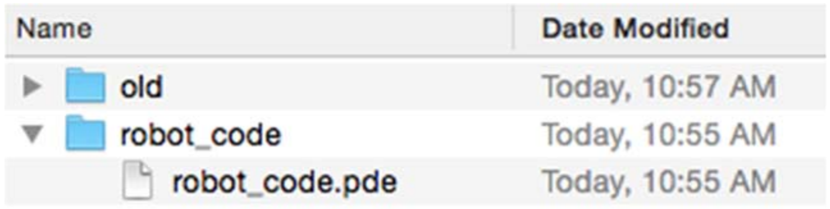
Creating & Reading
alpha()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()

Image

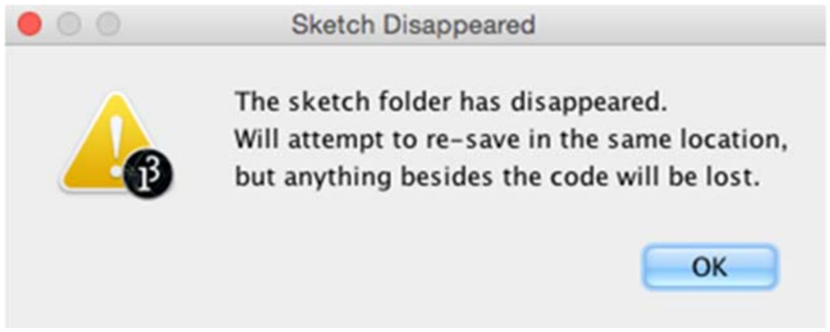
createImage()

Aside: Processing Files

- Processing files have extension `.pde`
  - File names *cannot* contain dashes (`-`) use underscore (_) instead
- 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

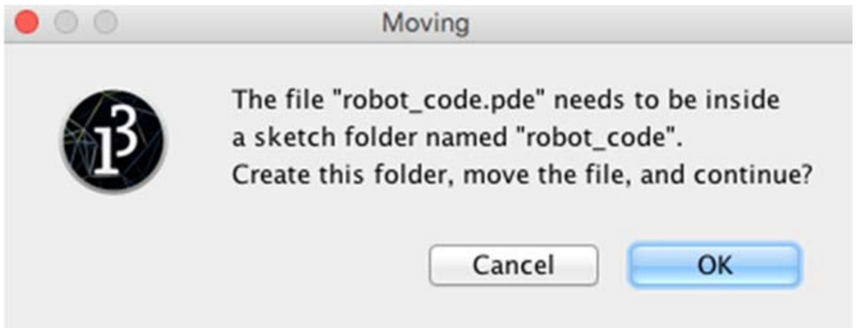


Name	Date Modified
▶ old	Today, 10:57 AM
▼ robot_code	Today, 10:55 AM
robot_code.pde	Today, 10:55 AM



Sketch Disappeared

The sketch folder has disappeared.
Will attempt to re-save in the same location,
but anything besides the code will be lost.

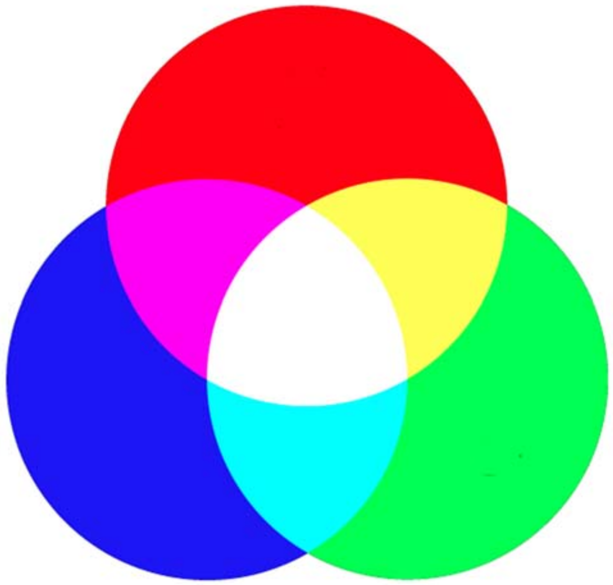


Moving

The file "robot_code.pde" needs to be inside
a sketch folder named "robot_code".
Create this folder, move the file, and continue?

Understanding Color

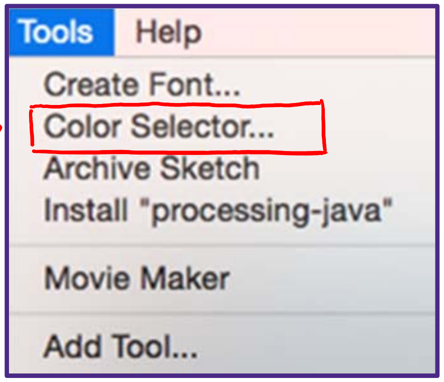
- In electronic systems, color specified using the RGB color model
  - **R**ed, **G**reen, **B**lue



- 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]
    - 0 is completely off
    - 255 is highest intensity

Processing's Color Selector

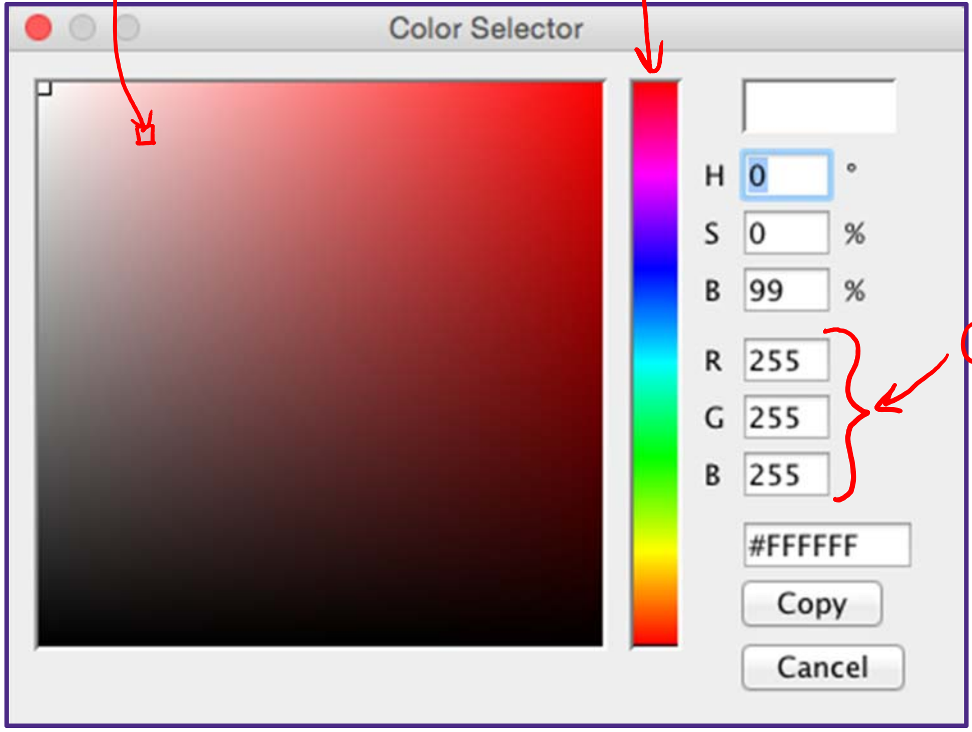
⓪ open color selector



① use color slider to get to different color ranges

② use color field to select color

③ copy RGB values from here



Guess the Color

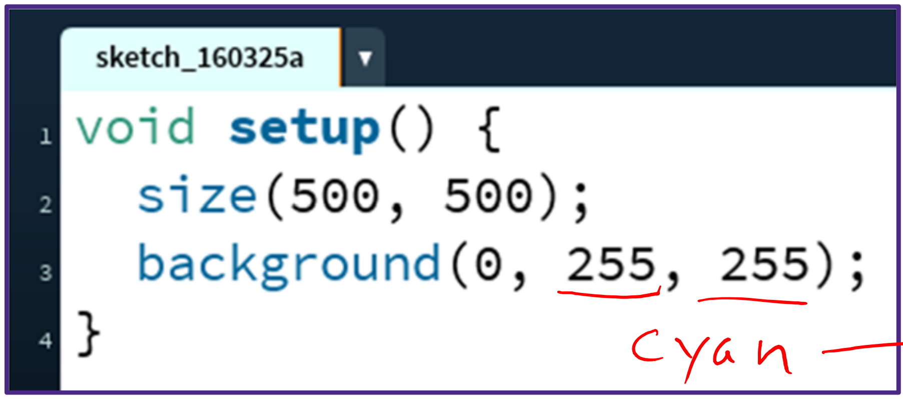
- `color(  R,   G,   B);`
- `color(255,   0,   0);`
- `color(  0, 255,   0);`
- `color(  0,   0, 255);`
- `color(  0,   0,   0);`
- `color(255, 255, 255);`
- `color(255, 255,   0);`
- `color(255,   0, 255);`
- `color(  0, 255, 255);`

Guess the Color

```
❖ color(  R,   G,   B);
❖ color(255,   0,   0); // 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

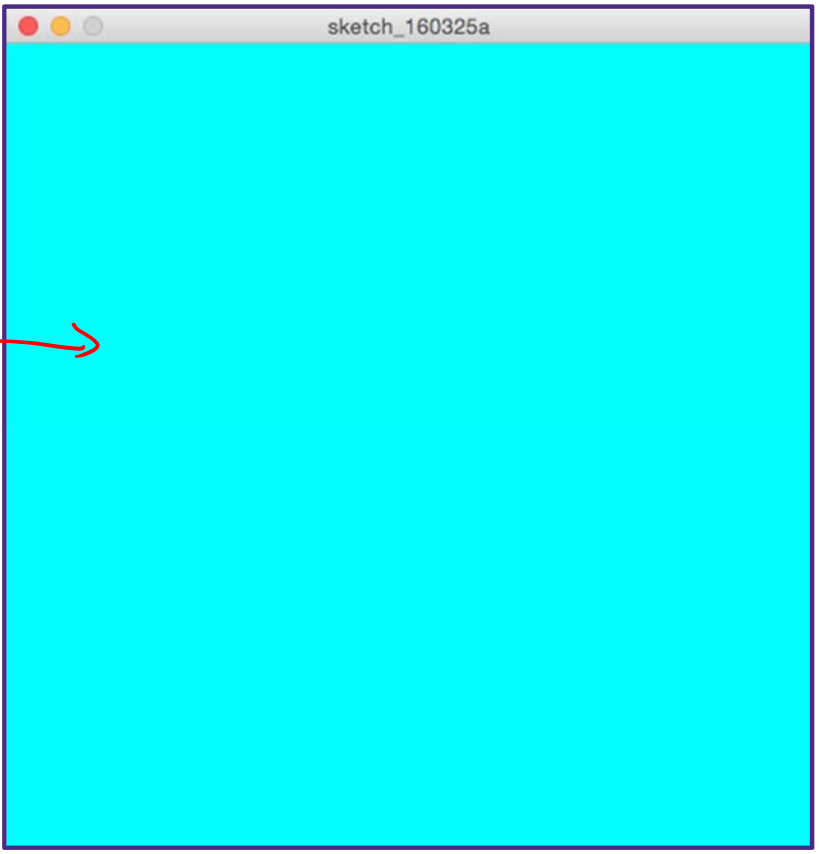
- `background(R, G, B);`
  - Covers the entire drawing canvas with the specified color
  - Will draw over anything that was previously drawn



sketch_160325a

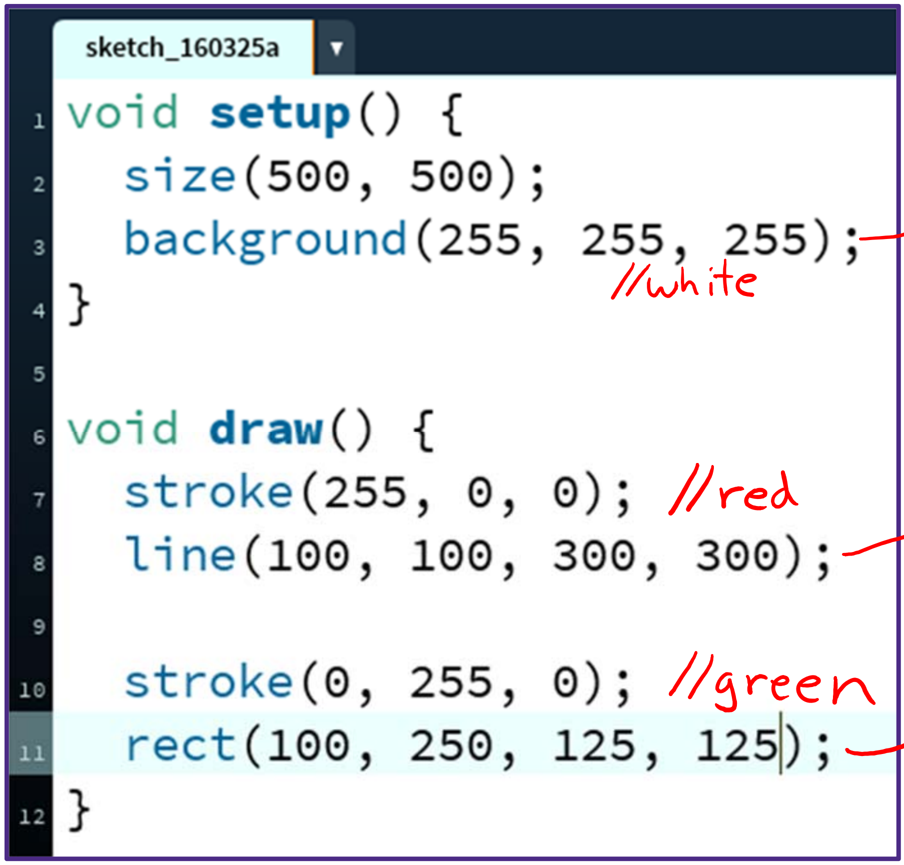
```
void setup() {
  size(500, 500);
  background(0, 255, 255);
}
```

cyan

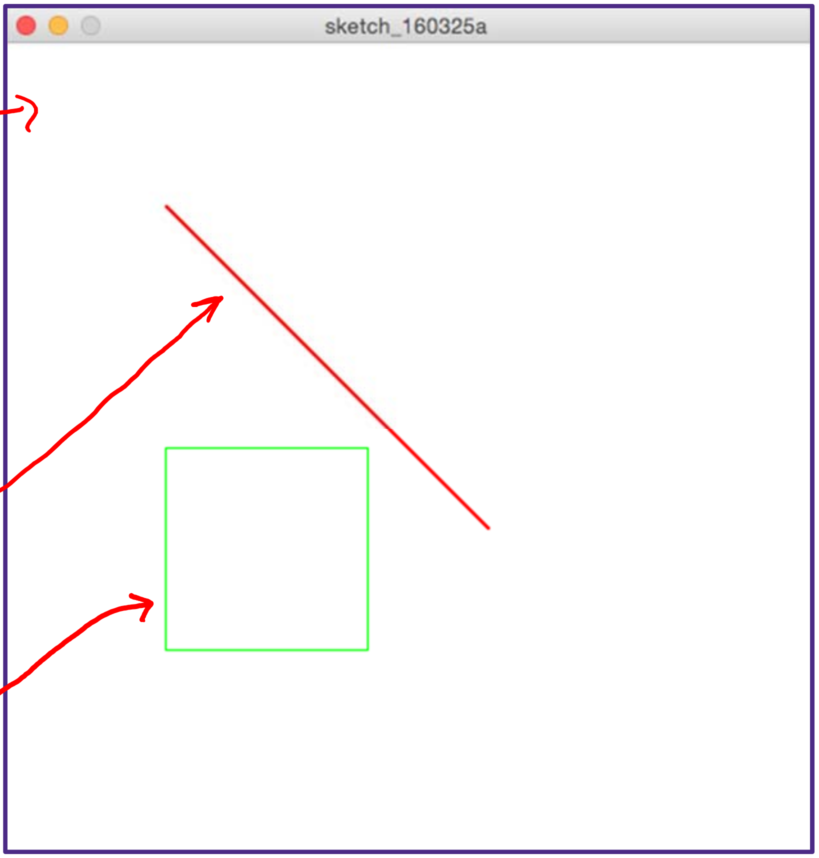


Color Functions

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

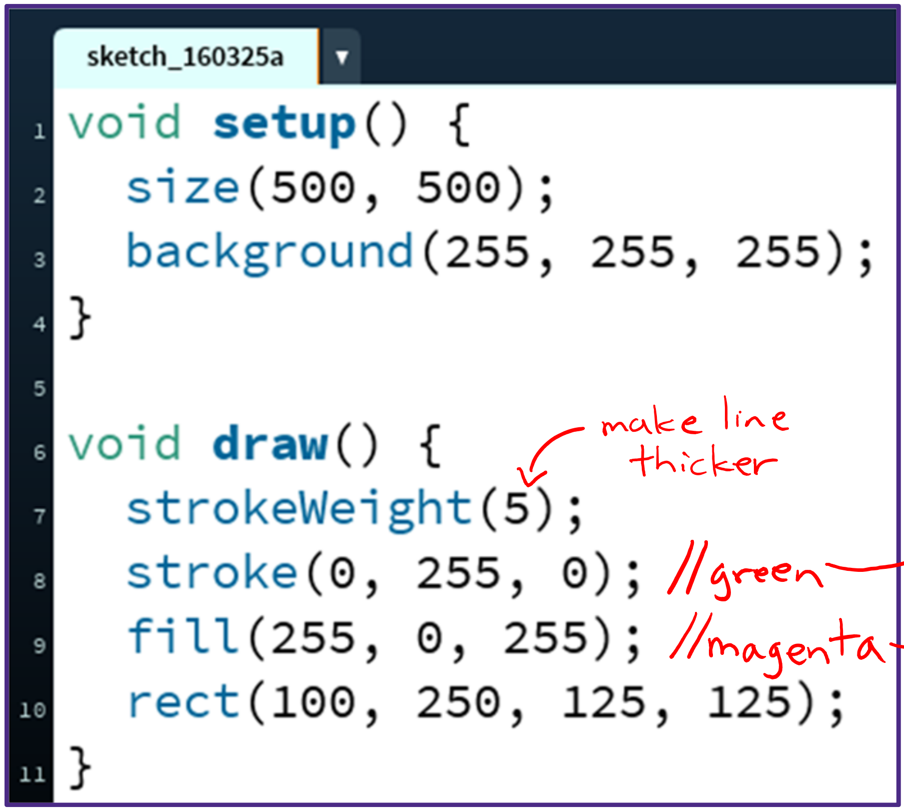


```
void setup() {
  size(500, 500);
  background(255, 255, 255);
}                    //white

void draw() {
  stroke(255, 0, 0); //red
  line(100, 100, 300, 300);

  stroke(0, 255, 0); //green
  rect(100, 250, 125, 125);
}
```



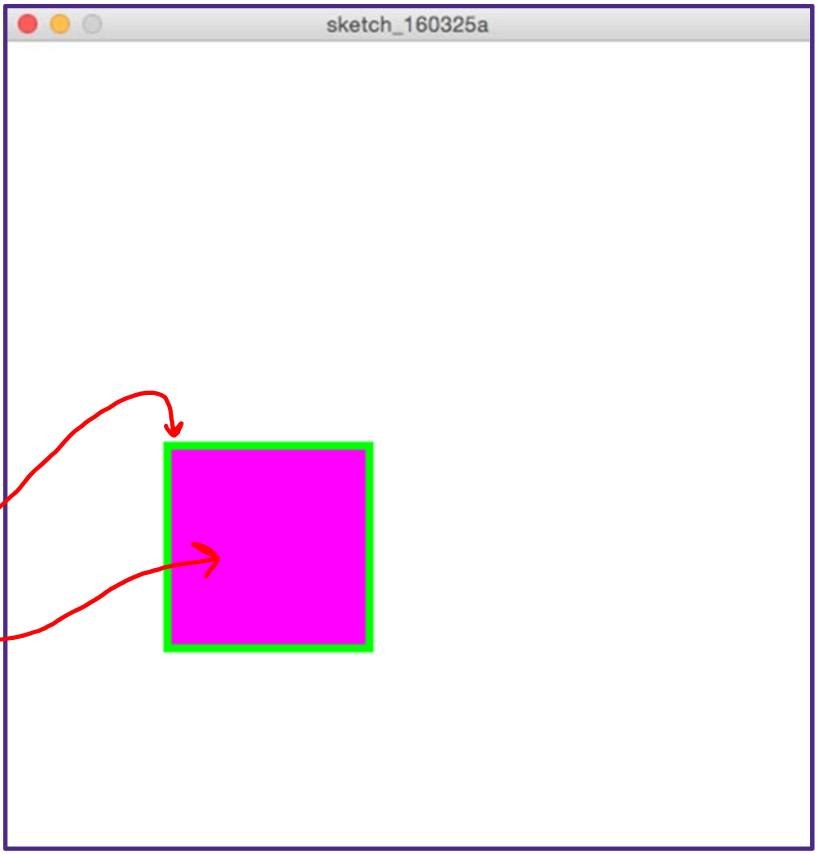
Color Functions

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



```
void setup() {
  size(500, 500);
  background(255, 255, 255);
}

void draw() {
  strokeWeight(5);
  stroke(0, 255, 0); //green
  fill(255, 0, 255); //magenta
  rect(100, 250, 125, 125);
}
```

make line thicker

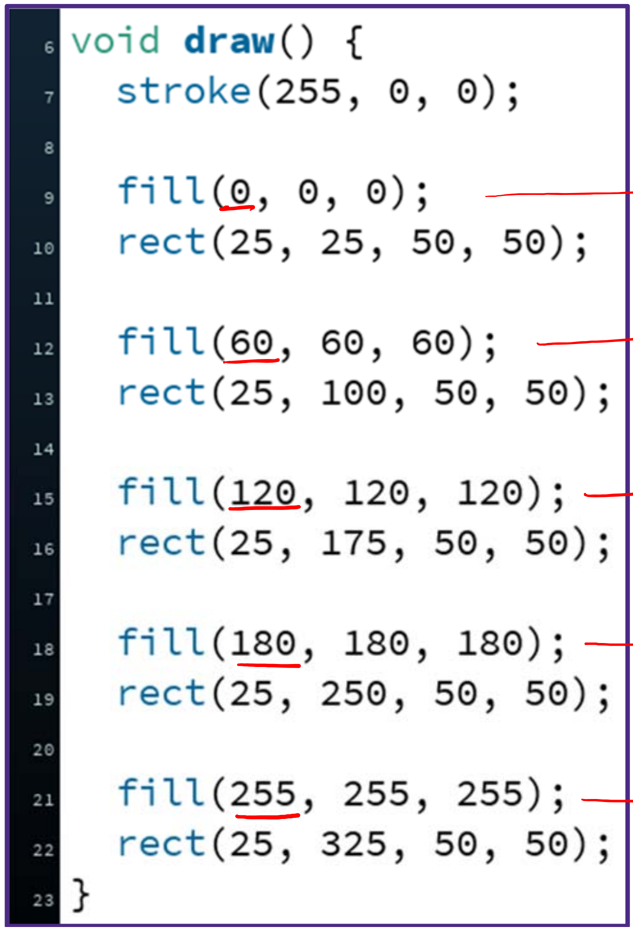


Color: “Grays"

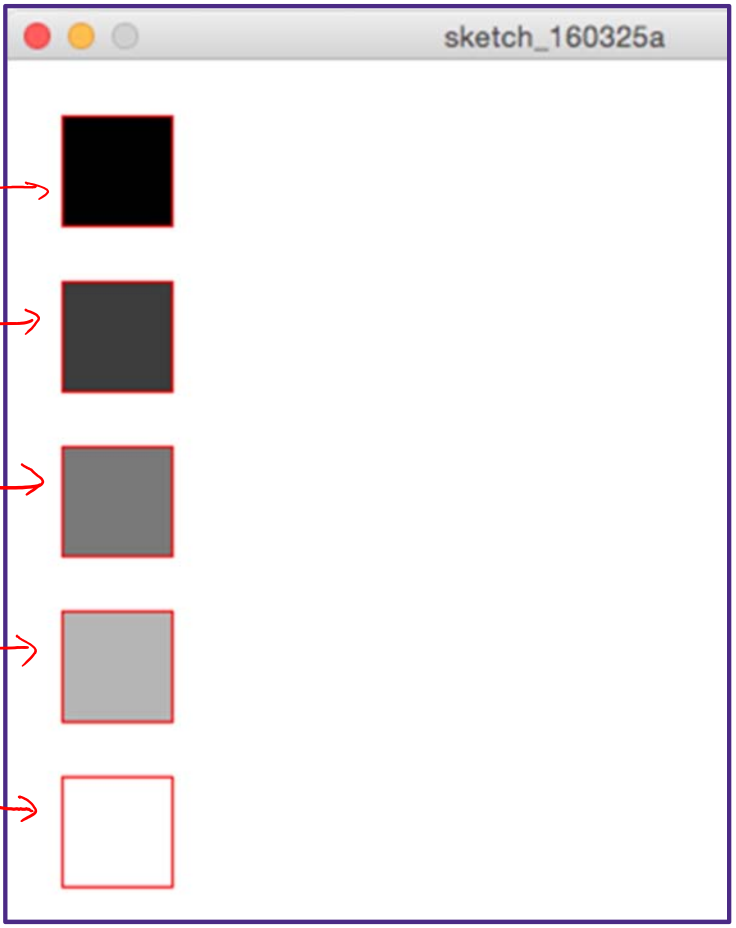
- When the values for RGB are all the same, then the color will be white, black, or some shade of gray

darker (closer to black)

lighter (closer to white)

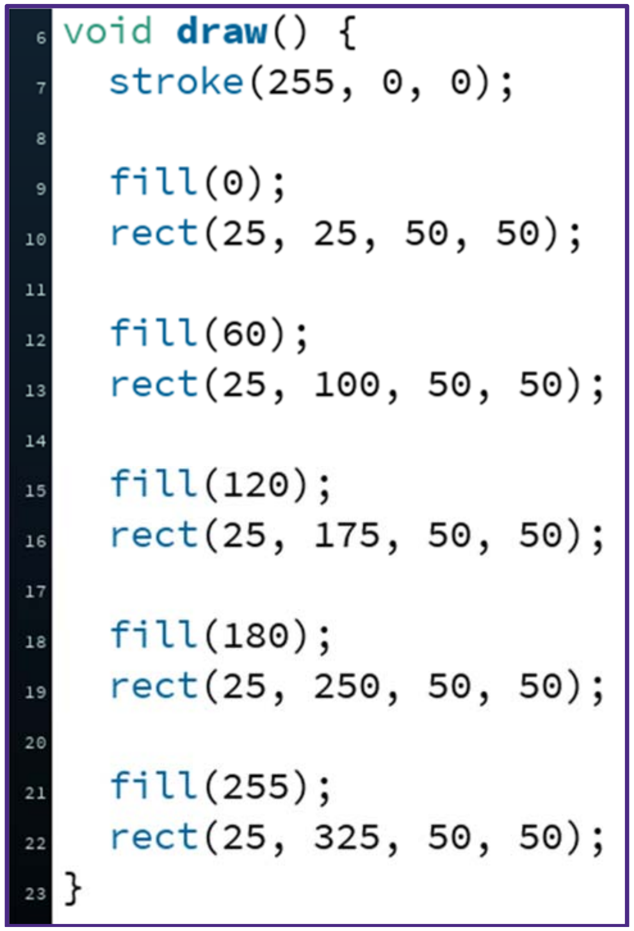


```
void draw() {
  stroke(255, 0, 0);

  fill(0, 0, 0);
  rect(25, 25, 50, 50);

  fill(60, 60, 60);
  rect(25, 100, 50, 50);

  fill(120, 120, 120);
  rect(25, 175, 50, 50);

  fill(180, 180, 180);
  rect(25, 250, 50, 50);

  fill(255, 255, 255);
  rect(25, 325, 50, 50);
}
```

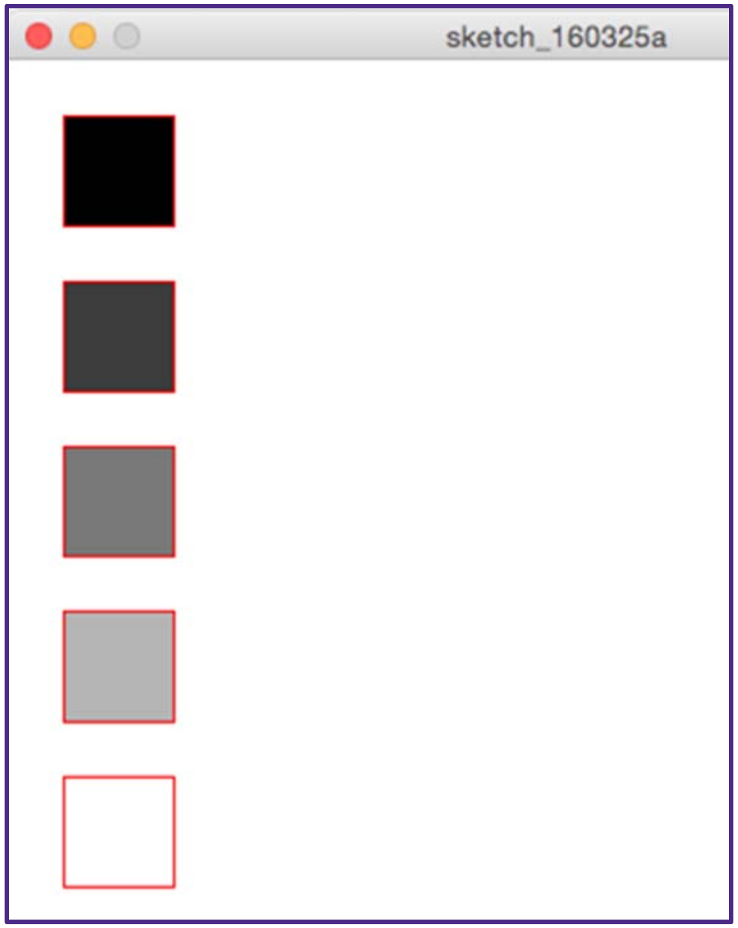


Color: “Grays"

- 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



```
void draw() {
  stroke(255, 0, 0);

  fill(0);
  rect(25, 25, 50, 50);

  fill(60);
  rect(25, 100, 50, 50);

  fill(120);
  rect(25, 175, 50, 50);

  fill(180);
  rect(25, 250, 50, 50);

  fill(255);
  rect(25, 325, 50, 50);
}
```



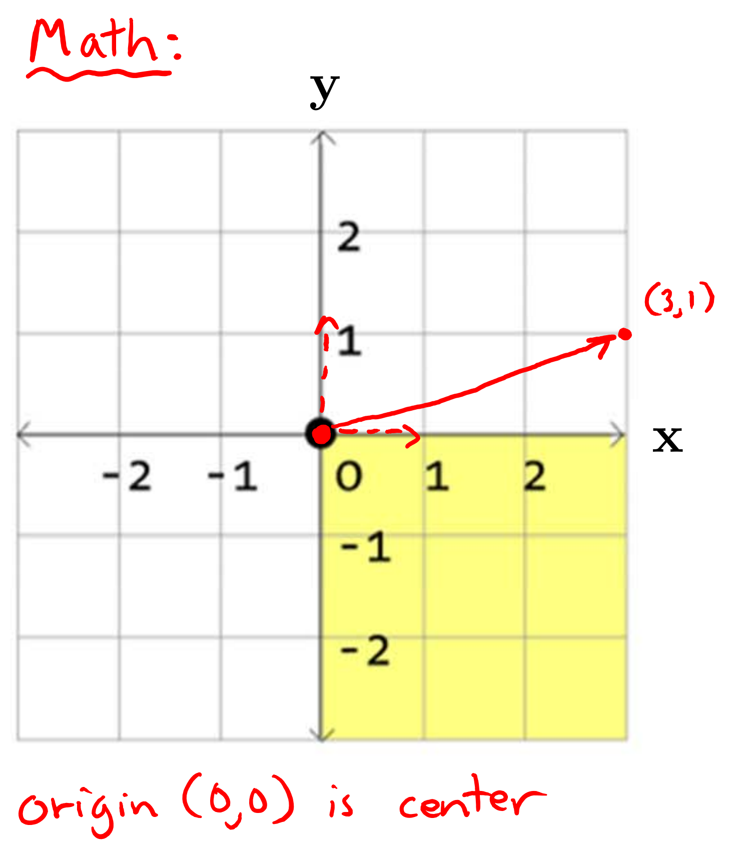
The Color "State" of Your Program

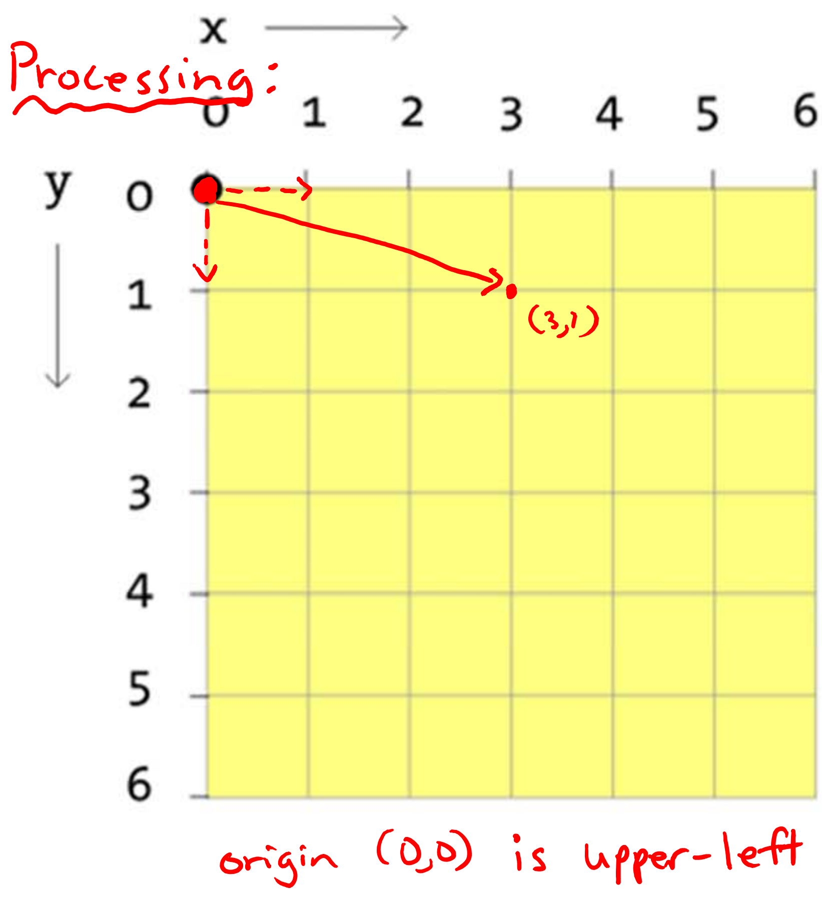
- Recall that programs are executed sequentially (*i.e.* instruction-by-instruction)

- `stroke()` and `fill()` apply to *all* subsequent drawing statements
  - Until a later call overrides

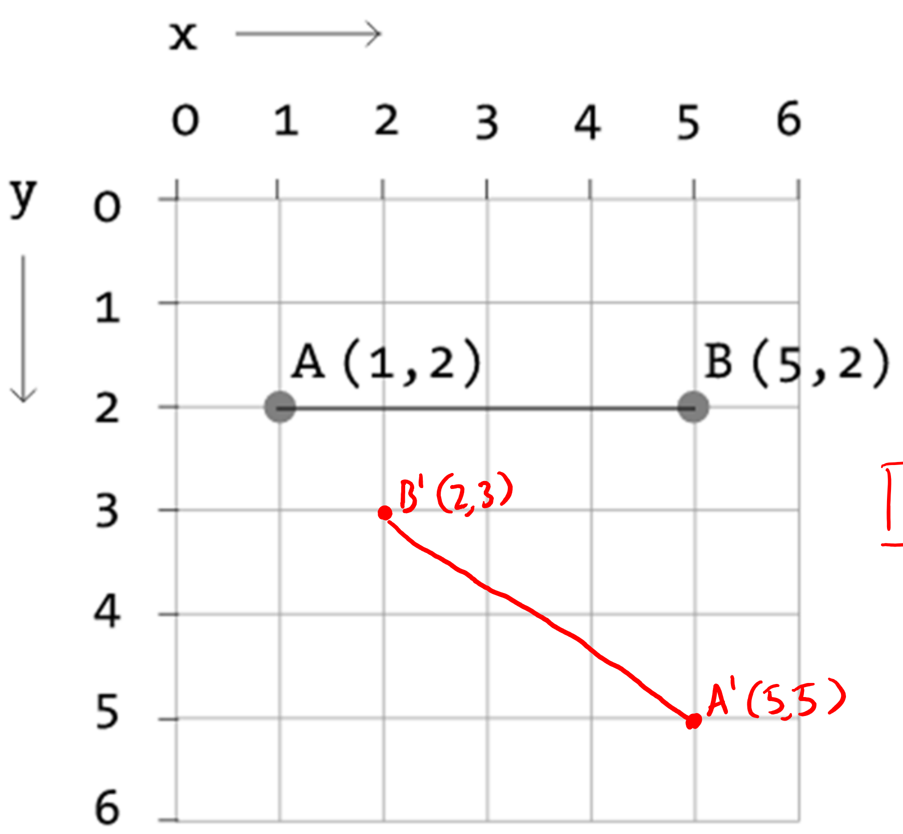
- Hidden color "state" that knows the current values of `stroke()`, `strokeWeight()`, and `fill()`
  - In complex programs, can be difficult to keep track of
  - Early rule of thumb: always explicitly set colors before each drawing element

Coordinate System





Drawing: Line



$\boxed{\texttt{line(x1,y1,x2,y2);}}$

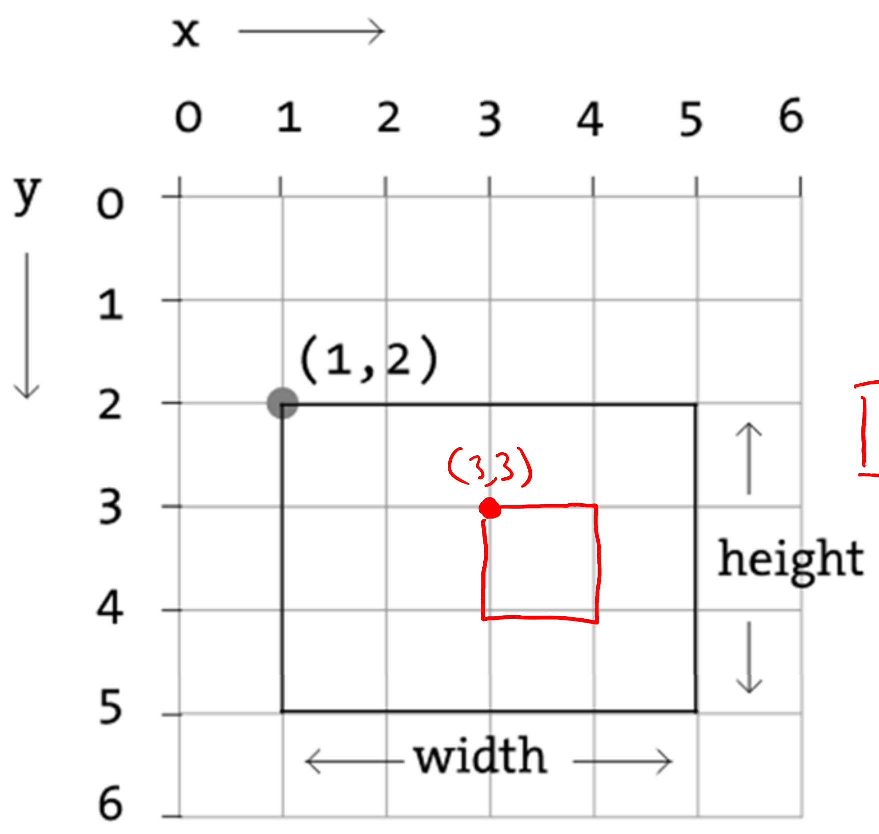
Point A Point B

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

line(5, 5, 2, 3);

Drawing: Rectangle

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

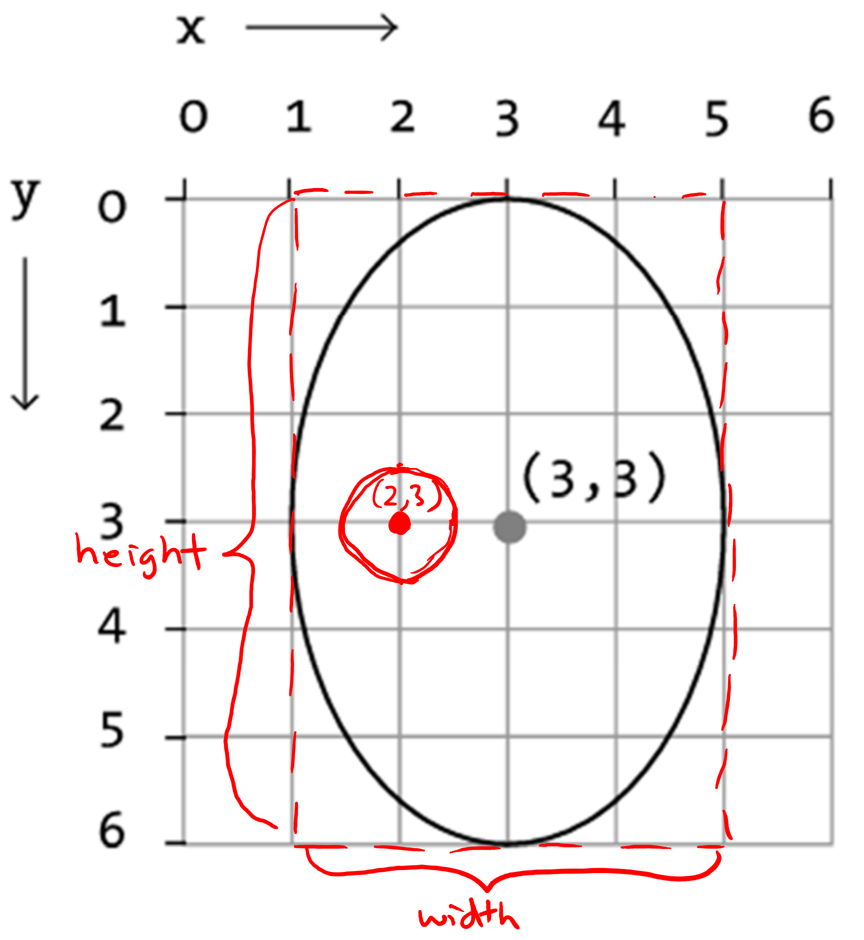


rect (x, y, width, height);

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

Drawing: Ellipse/Circle

- Default *mode* is CENTER



ellipse(x, y, width, height);

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

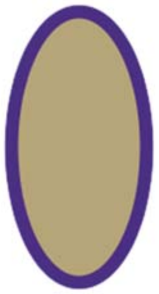
ellipse(2, 3, 1, 1);

Peer Instruction Question

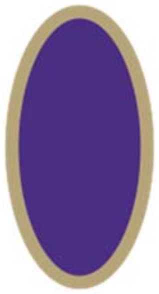
- Which of the following drawings corresponds to the Processing code below?
  - Vote at http://PollEv.com/justinh

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

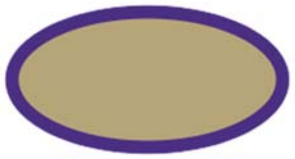
[✓ A.]



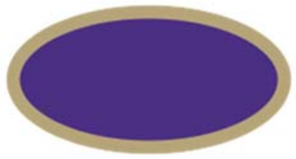
~~B.~~



C.

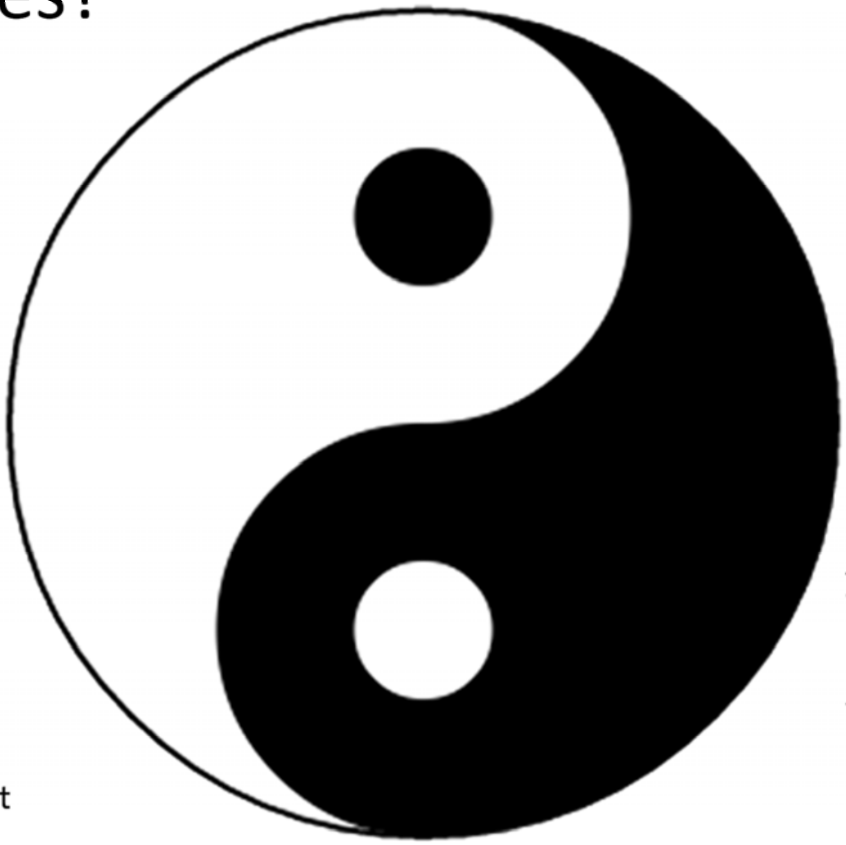


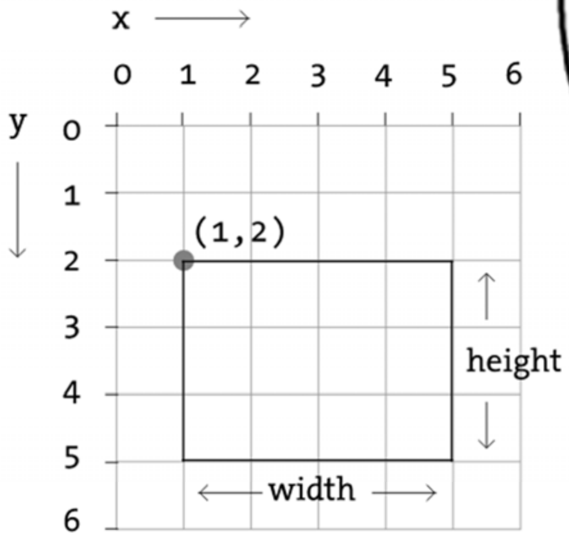
~~D.~~



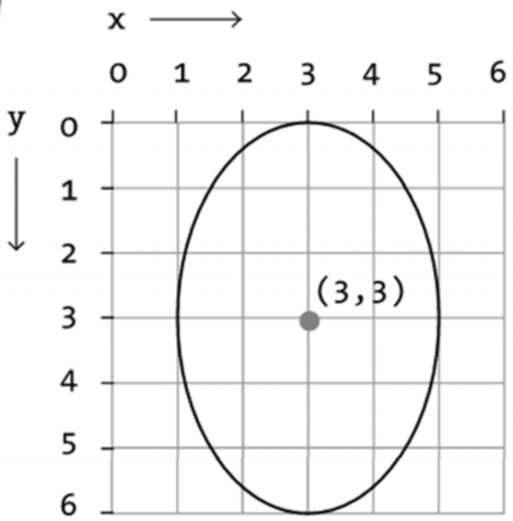
Activity: Taijitu

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





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



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