Introduction to Processing

Kids & Technology Meetup

April 5, 2014

Washington, DC

Download and Install Processing

- Go to https://processing.org/download/
- Download
- Extract the files
- Processing can be run from any location, but on Windows it is conventional to place the folder into C:\Program Files\processing-2.1.1 (or whatever version)
- Instructions for Mac and Linux can be found here: http://www.processing.org/tutorials/gettingstarted/
- Create a shortcut for the Processing.exe file and place on the Desktop

Processing Development Environment (PDE)

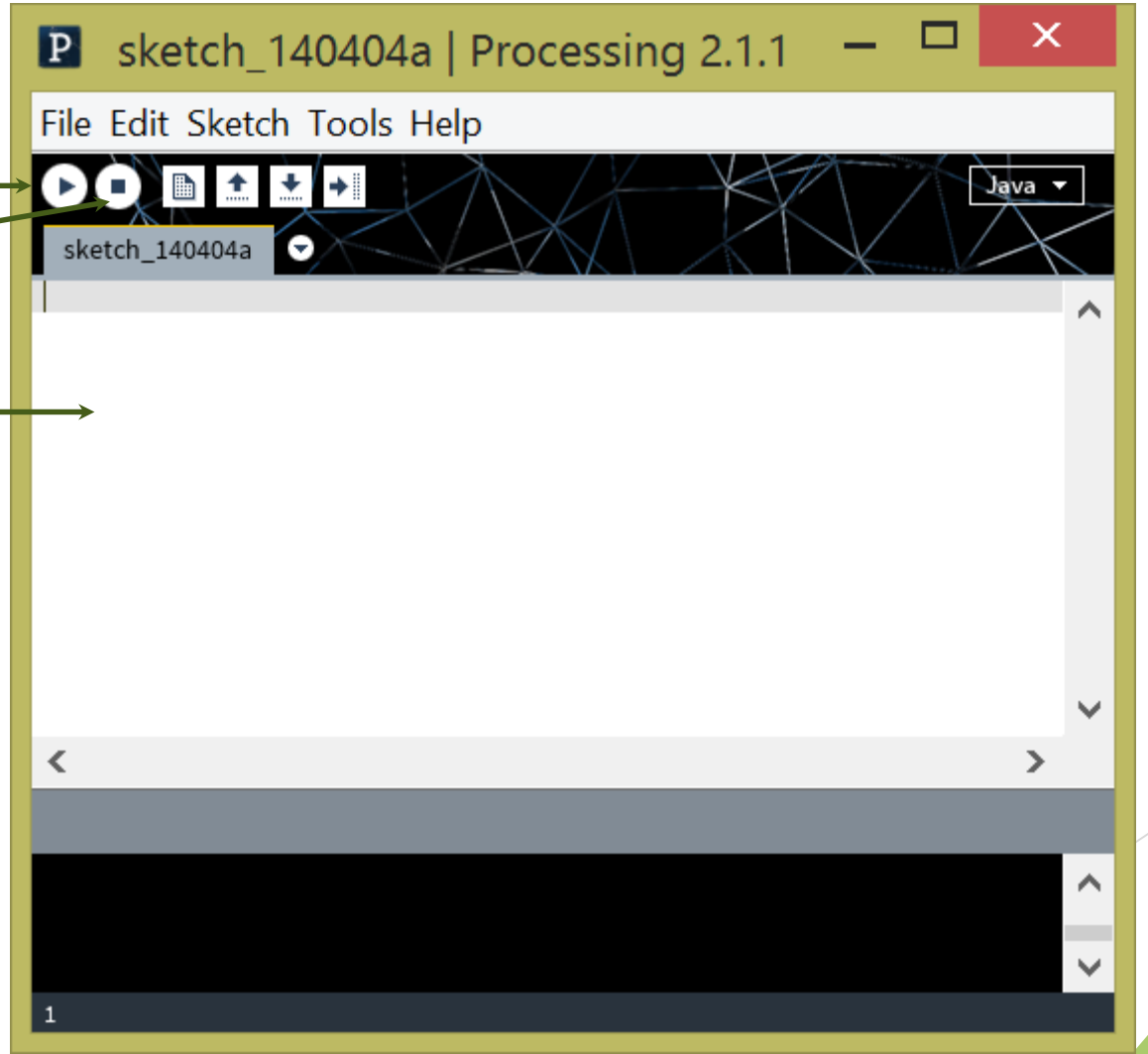
Run program

Stop

Type code here

Remember to save! (click on "File")

Code files in Processing are called "sketches"

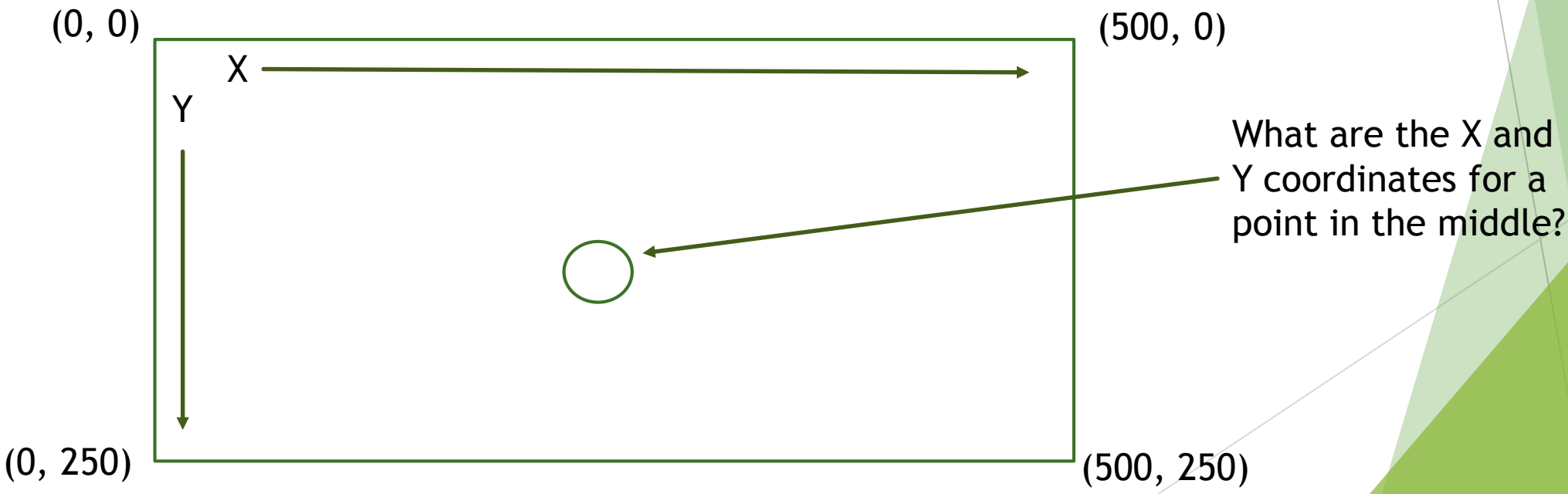


The Canvas

Must be lower case

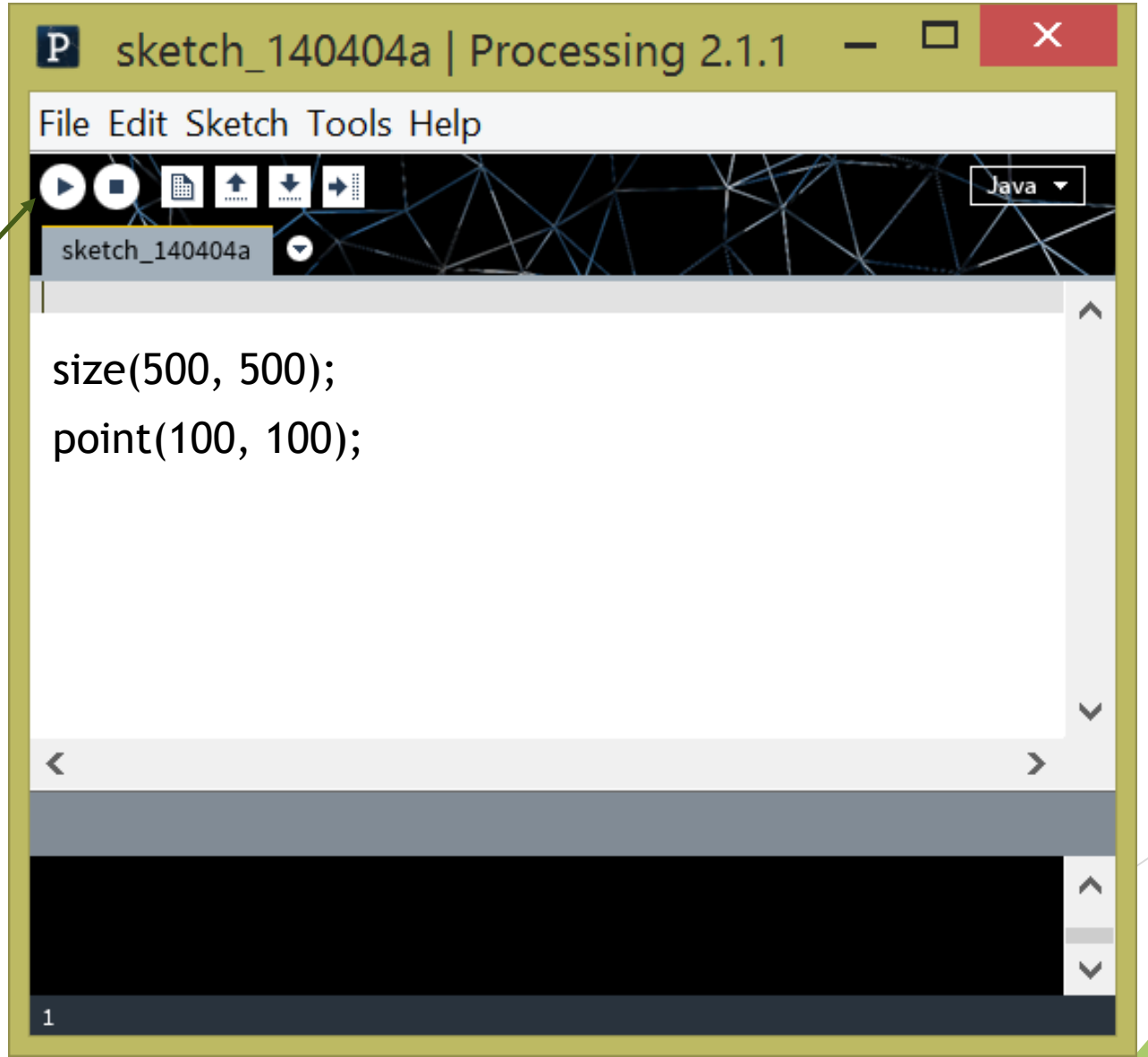
Must end with ;

- To control the size of the canvas, type: size(X, Y);
- X = width
- Y = height
- size(500, 250); creates the following:



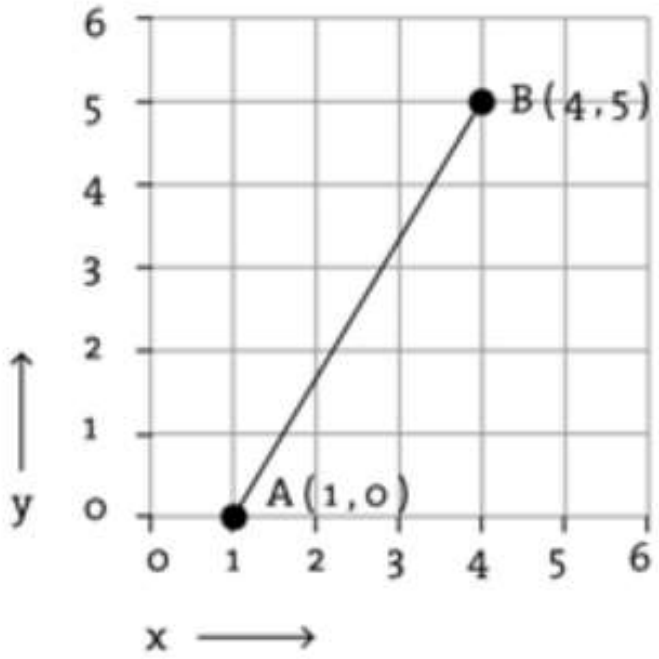
Let’s Code!

Press Run!

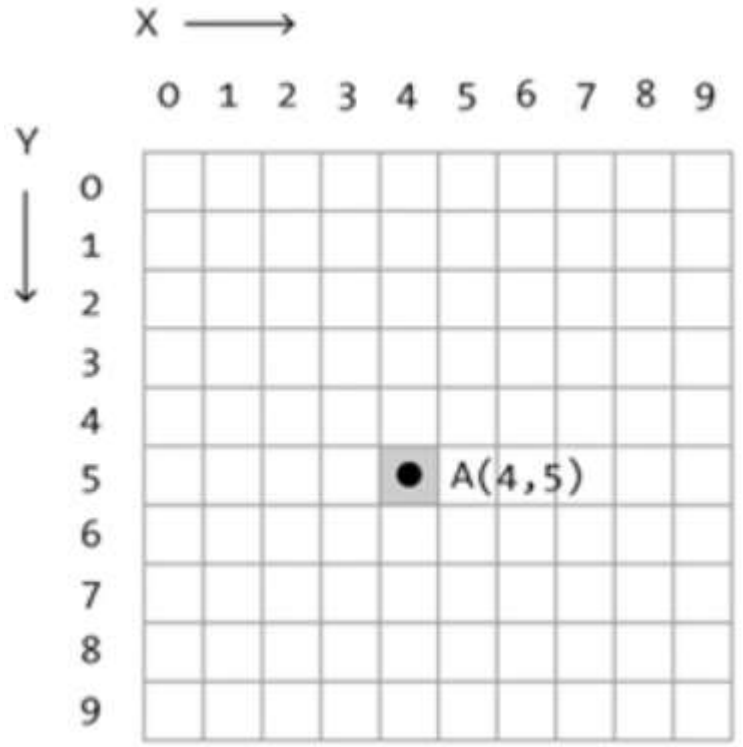


Canvas Layout

This is the X,Y coordinate system we learn in school:



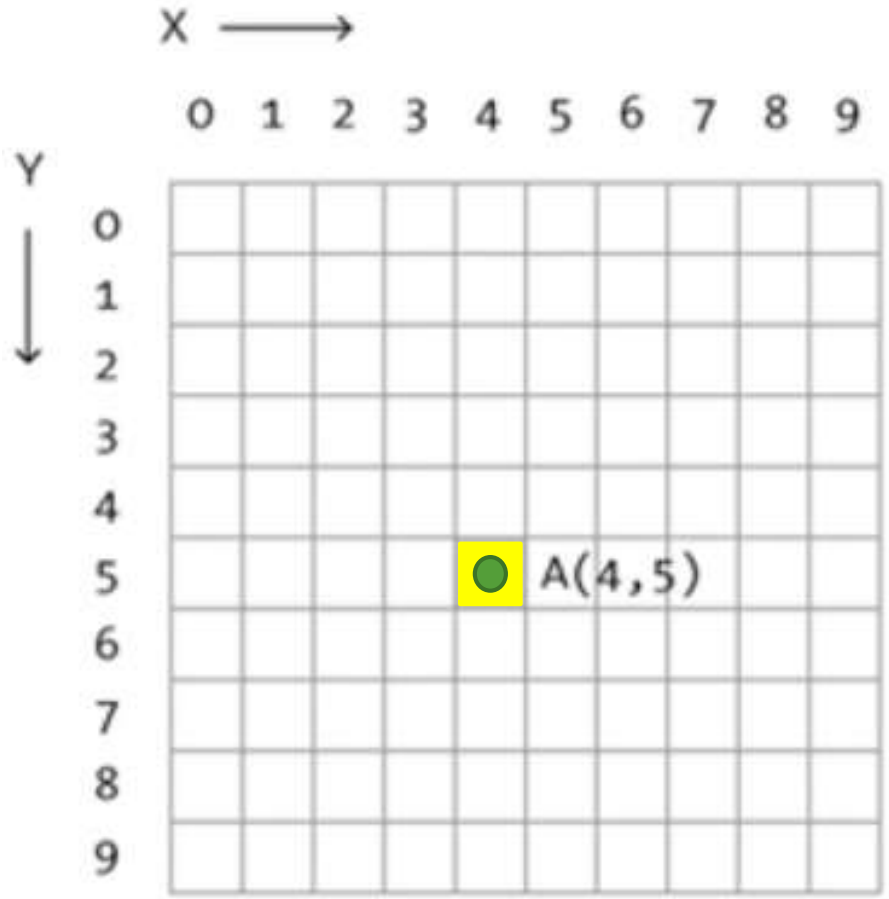
This is the coordinate system for the Canvas. Each point is a pixel:



Basic Shapes - Point

Point

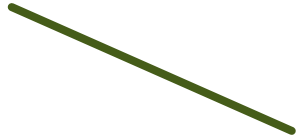
point(X, Y);



point(4, 5);

Basic Shapes - Line

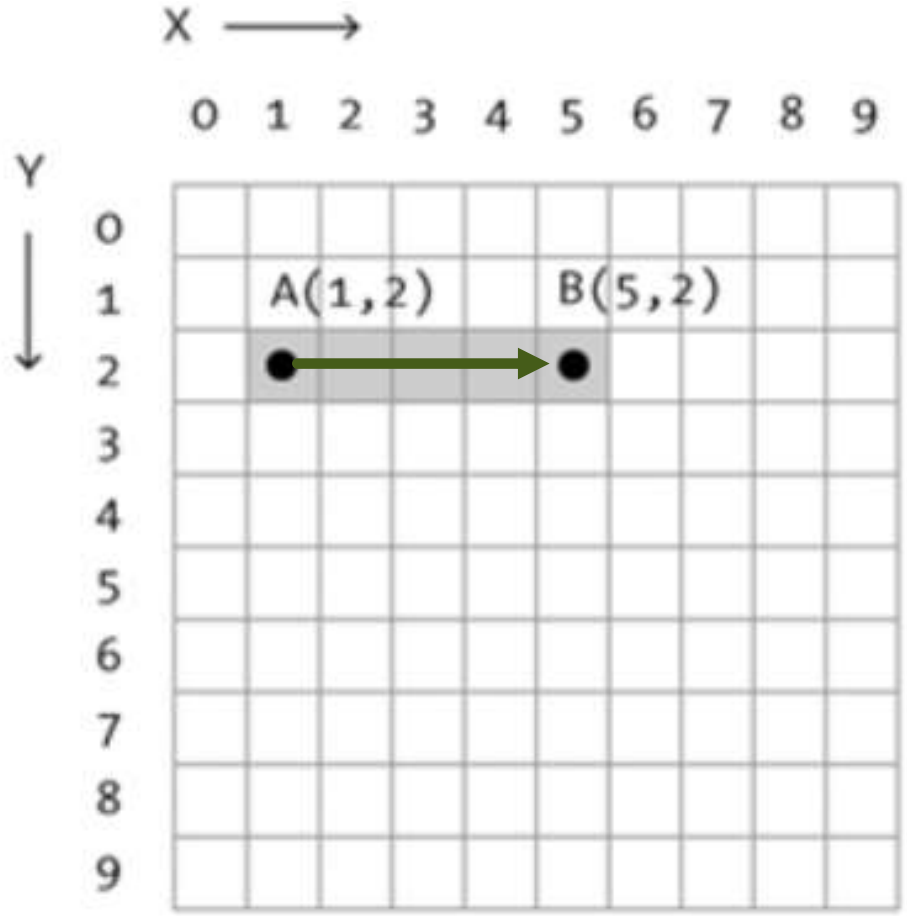
Line



line(X1, Y1, X2, Y2);

Starting point

Ending point



line(1, 2, 5, 2);

Basic Shapes - Re

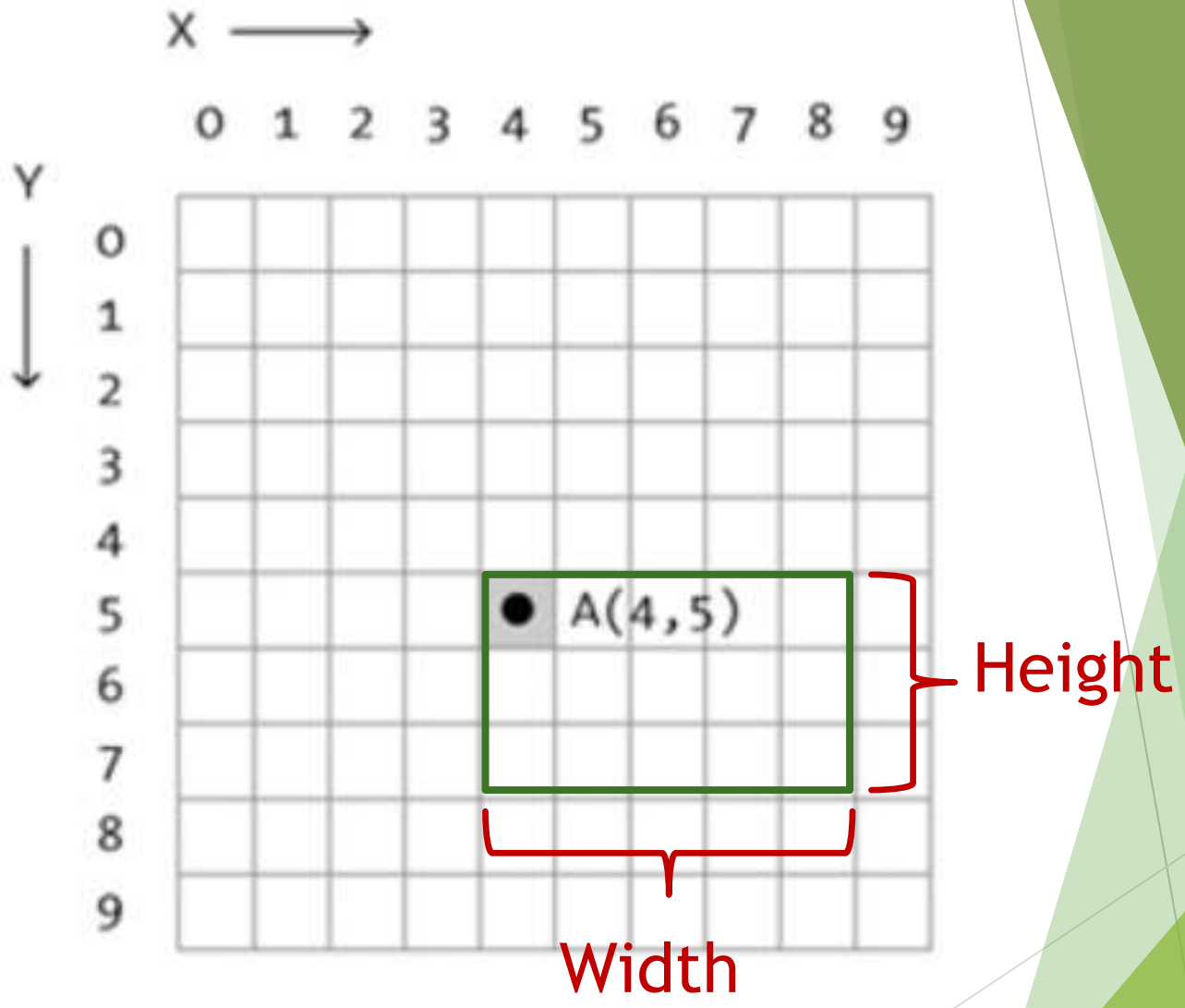
Rectangle

rect(X, Y, W, H);

Starting point

Width

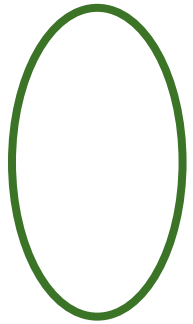
Height



rect(4, 5, 5, 3);

Basic Shapes - Ellip

Ellipse

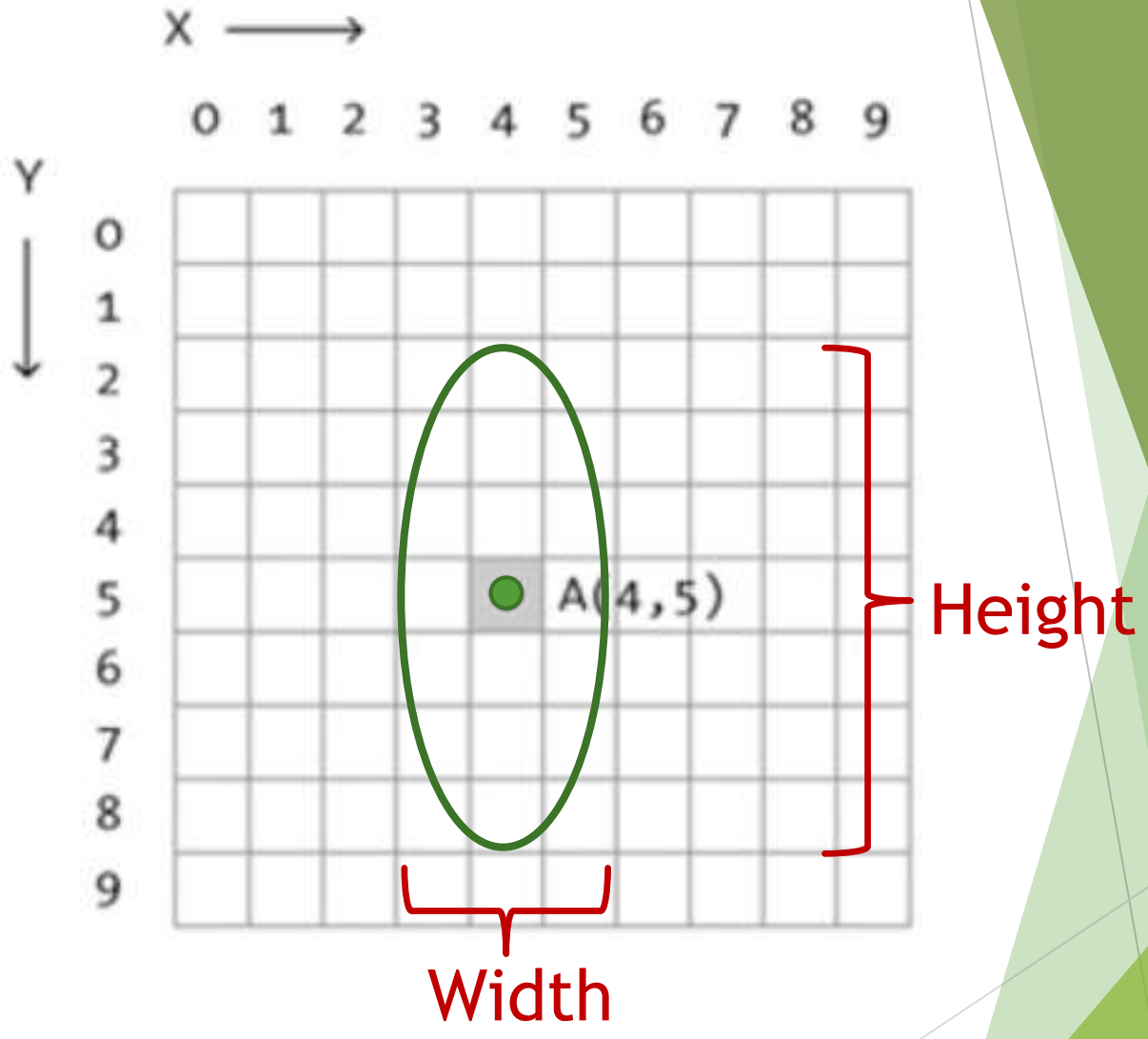


ellipse(X, Y, W, H);

Center

Width

Height



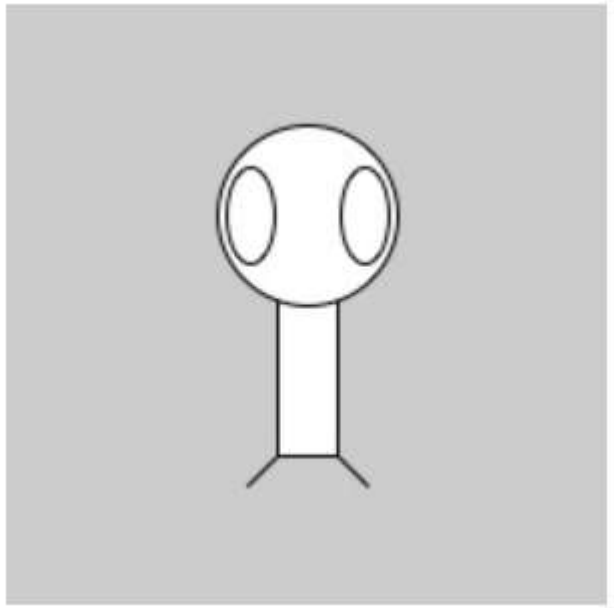
ellipse(4, 5, 3, 7);

An ellipse where W = H is a circle !

Draw a Figure

- Use what we have learned to combine shapes and draw something
- Or you can use the code below – make sure you understand what each line is doing

```
size(200,200);
rectMode(CENTER);
rect(100,100,20,100);
ellipse(100,70,60,60);
ellipse(81,70,16,32);
ellipse(119,70,16,32);
line(90,150,80,160);
line(110,150,120,160);
```

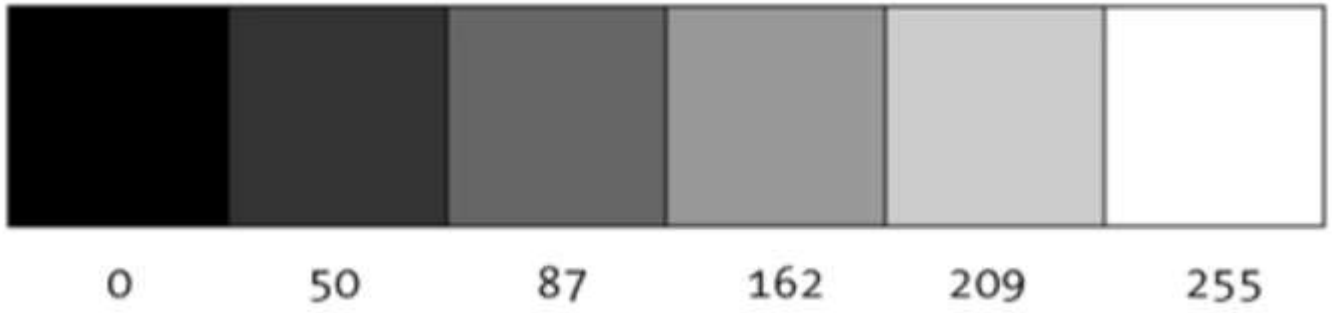


Source: http://processing.org/tutorials/drawing/

Add Some “Color”

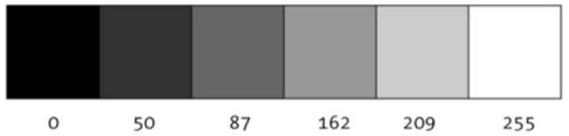
Set the background color to white:
background(255);

Why do we use 255 for white?



Grayscale

stroke() and fill()



The **line color** (or outline color) is set with stroke();
Set the line color to black:

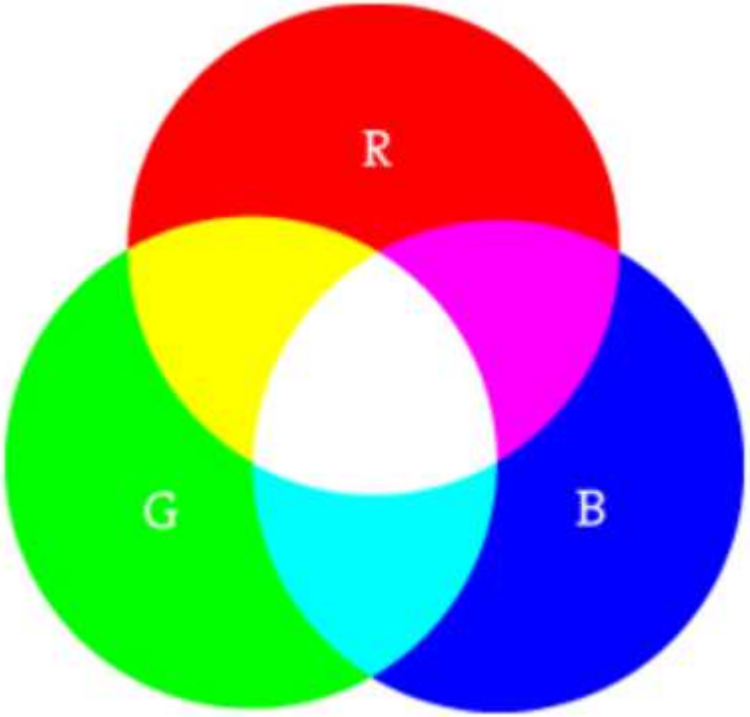
```
stroke(0);
```

Set the shape's **fill** color to gray:

```
fill(150);
```

Numbers closer to 0 will be darker, closer to 255 will be lighter.

Now Let's Really Add Some Color



RGB – Red Green Blue

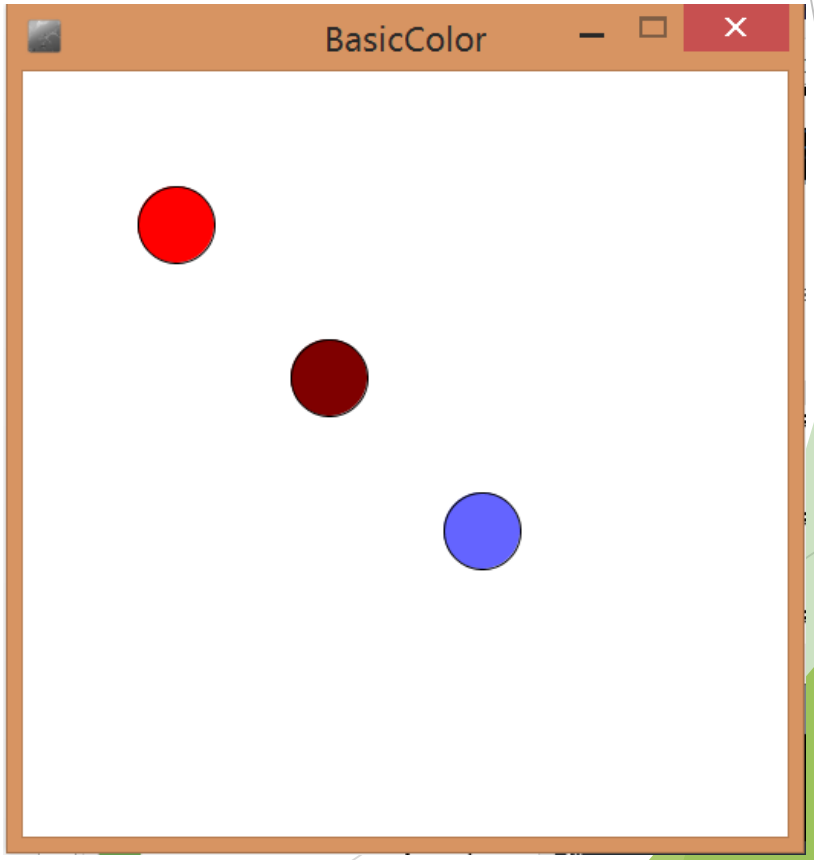
Instead of typing fill(150), we can use the three RGB values:
fill(R, G, B);

```
fill(255, 0, 0);    // red
fill(0, 255, 0);    // green
fill(0, 0, 255);    // blue

fill(100, 50, 50);    // to mix colors
```

Using Multiple Colors in One Sketch

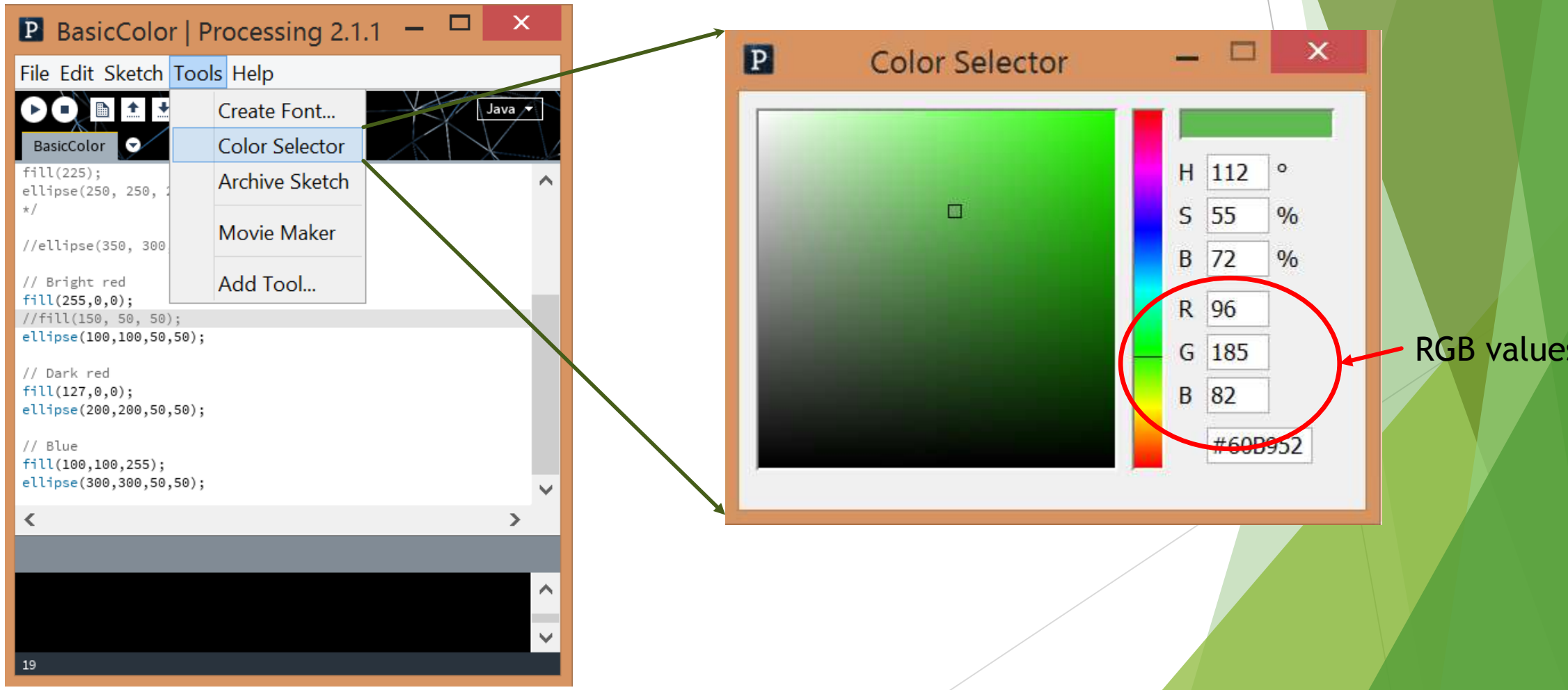
- Every time you add the stroke() or fill() statements, the shapes that follow will get those colors
- You can keep changing colors within a sketch, as shown here:

```
background(255);
size(500, 500);

// Bright red
fill(255,0,0);
ellipse(100,100,50,50);

// Dark red
fill(127,0,0);
ellipse(200,200,50,50);

// Blue
fill(100,100,255);
ellipse(300,300,50,50);
```



Find the Perfect Color

- Processing has a color selector tool

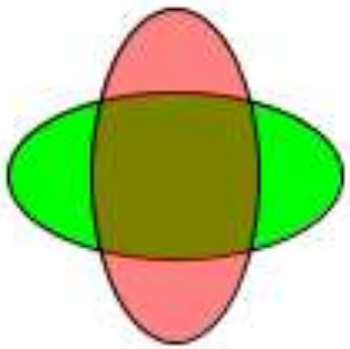


Color Transparency

- To let portions of objects “beneath” other objects show through, we an add transparency

```
fill(0, 255, 0, 255);    // green, opaque
fill(255, 0, 0, 125);    // red, about 50% transparent
```

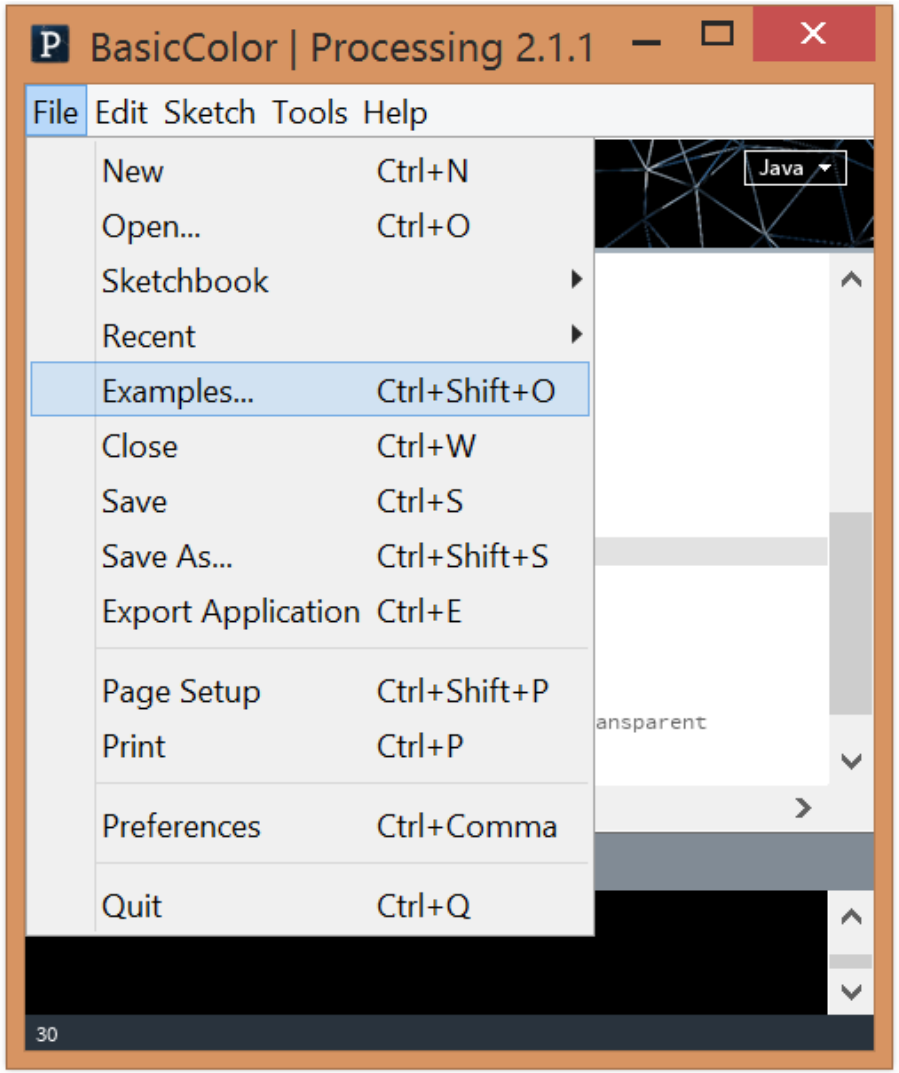
0 = completely transparent
255 = completely opaque

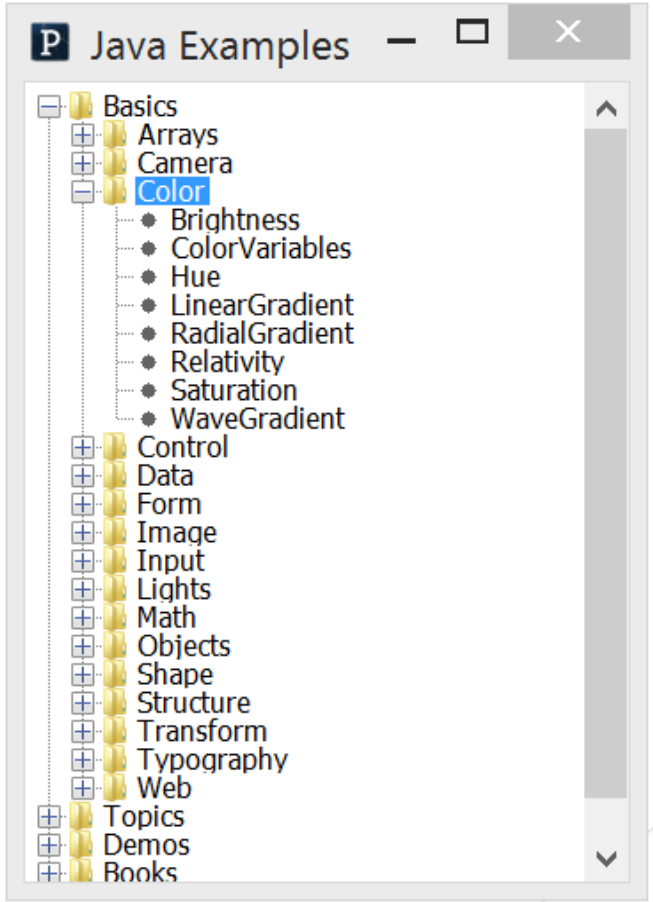


The first object in the sketch will be on the bottom.
The next will go “above” that one. Each object is added to a new layer.

Examples

- File / Examples





How About Animation?

- Yes, you can do that in Processing
- But, we need to learn a few more things first
  - Variables
  - Loops
  - Conditionals

Variables

- Variables are used for storing values
- We can change those values if we want to
- For drawing, we are mostly going to use integers and decimal numbers (floating point numbers)

```
int boxWidth = 75;      // the box width is an integer and its value is 75 pixels
int boxHeight = 50;

float y = 2.5;              // y is a decimal and its value is 2.5
```

We can change the values of variables in our code.

Other variable types are described here:
http://www.openobject.org/physicalprogramming/Using_Variables_in_Processing

Loops

- To make an object move, we will have to “loop” or repeat some code many times
- Now we will use the Processing program structure

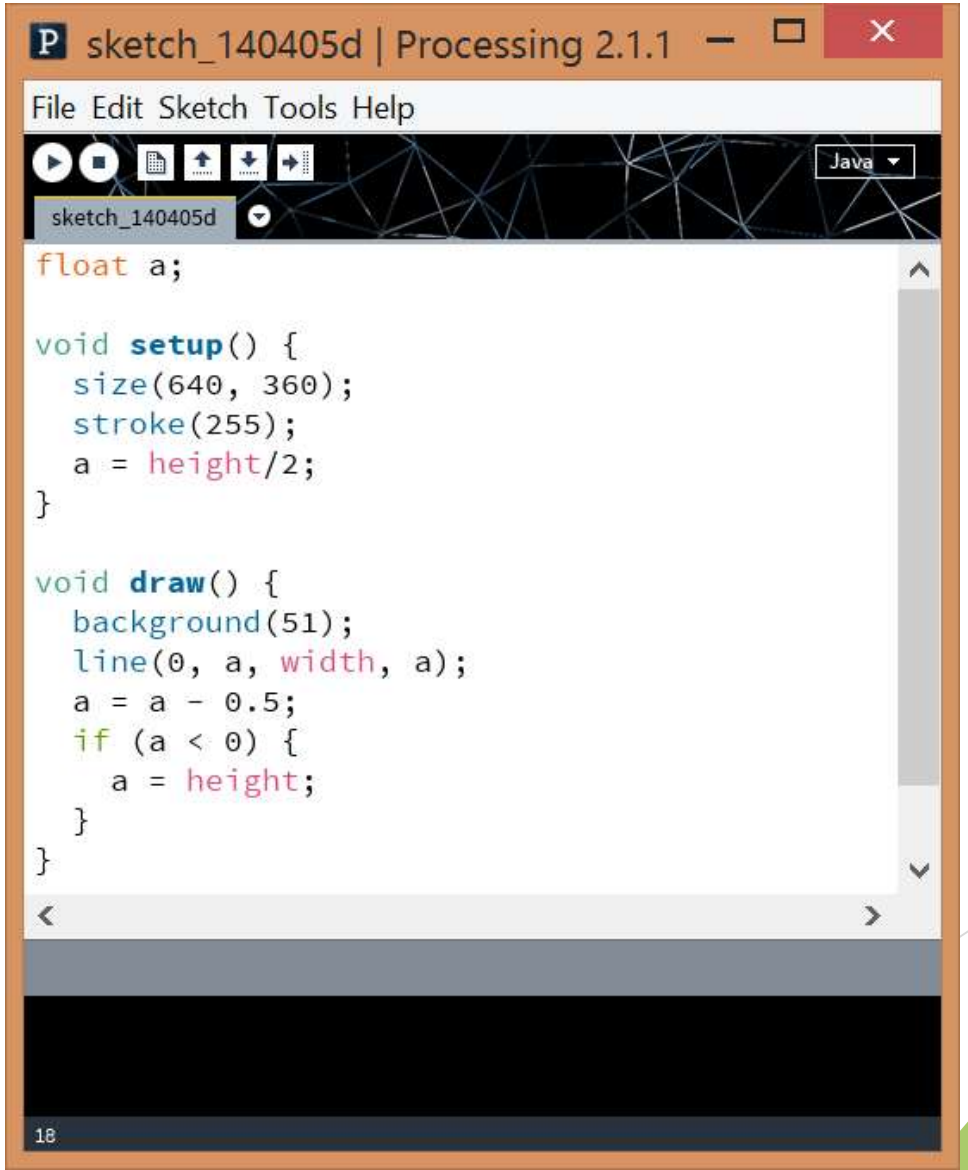
Loops – Processing Structure

Declare variables

setup() – these commands are only done once

Loop – the draw() loop repeats over and over

Notice that setup() and draw() enclose their contents inside curly braces.



Conditionals

- If this happens, then do that

```
if (test) {
  then do something;
}
```

```
if (test) {
  then do something;
}
else
{
  do something else;
}
```

Within each block (between the curly braces), there can be multiple lines of code.

Conditionals - continued

```
x = 0;

draw()
{
    x = x + 1;

    if (x > 100) {
    x = 0;
    }

}
```

Add one to x (increment x).

Check if x is greater than 100.

If so, set x back to 0.

Repeat that over and over.

Animation Example

- Let's draw a box that moves across the screen
- First draw the box on the left side of the canvas
- Do this in setup()

```
int x;        // declare x and y for our starting point
int y;

void setup() {

  size(500, 500);
  background(255);
  stroke(255);
  fill(100,100,255);
  x = 0;
  y = 150;

}
```

Animation Example - continued

- Now let's think about our draw() loop
- Let's start with a box on the left side of the screen
- How do we move the box toward the right hand side?

```
void draw() {

  rect(x, y, 100, 75);    // draw the box, use x & y for start location
  x = x + 5;              // increase x by 5 (you can try other values)

  if (x > 500) {          // if x is more the 500, it has passed the right side
    x = 0;                // set x back to zero so it goes back to the left
  }

}
```

Animation Example - continued

- But there's a problem!
- The box seems to be writing over itself, leaving a trail
- We need to erase the old box before we draw the new box

```
void draw() {

  background(255);    // redraw the background each time
  rect(x, y, 100, 75);
  x = x + 5;

  if (x > 500) {
    x = 0;
  }
}
```

You can find the full code listing here http://green.mn/1hszuUZ

Animation Example 2 – add a twist

- Instead of the box moving to the right, then appearing back at the left hand side after “dropping off” the right-hand side…
- Let’s make it “bounce” off of the left side of the canvas, then travel back to the right and “bounce” off of the right side of the canvas, etc.
- Think about how you would do that…

Animation Example 2 – continued

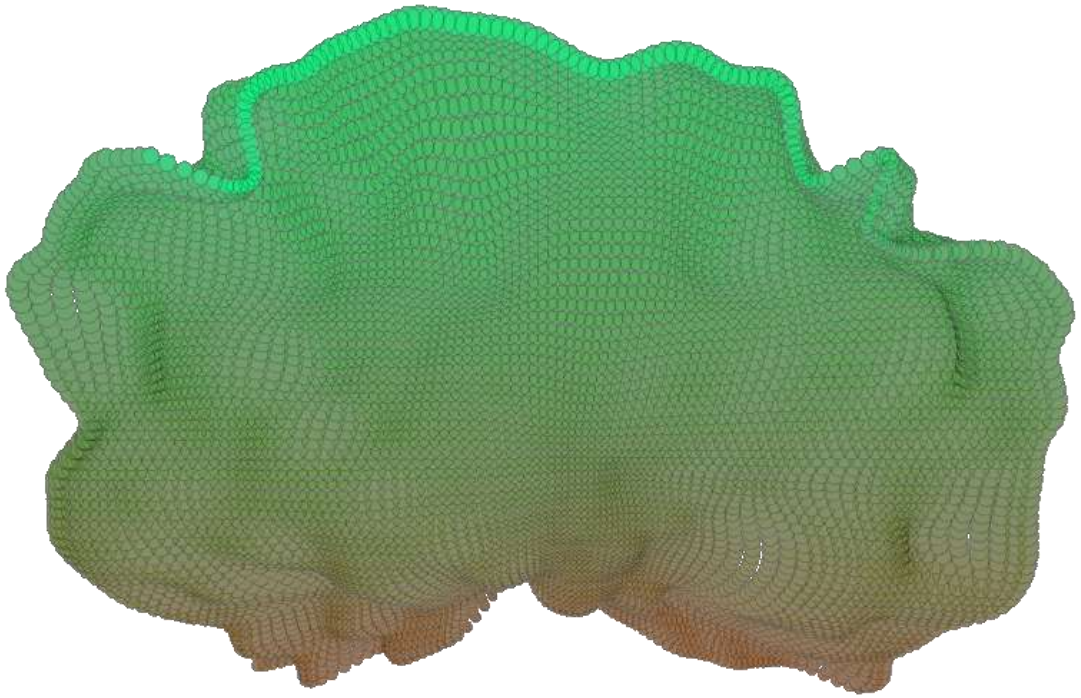
- When the box touches the right-hand side of the canvas, instead of x = x + 5 we need to have x = x – 5
- One way to do this is to make the 5 value a variable
- Then we can change its sign

```
void draw() {
  background(255);
  rect(x, y, 100, 75);
  x = x + step;          // "step" variable

  if (x == 400 || x == 0)   // if the rectangle touches either side
  {
    step = step * (-1);     // reverse the sign of "step"
  }
}
```

You can find the full code listing here http://green.mn/1gBdBCE

A Little More Advanced: Generative Art

- A program that draws something differently each time you run it
- Or it keeps drawing until you stop it



Source: *Generative Art: A Practical Guide Using Processing*, by Matt Pearson

Tutorials and Resources

- https://www.processing.org/reference/
- http://processing.org/tutorials/
- http://hello.processing.org/
- http://funprogramming.org/
- http://carrot.whitman.edu/IntroProcessing/syllabus.html
- http://sketchpad.cc

Questions?

- These slides will be posted on SlideShare and the link will be provided
- Contact me on Twitter: @JohnDukovich or @GreenMoonArt
- Follow this MeetUp: http://www.meetup.com/Kids-and-Technology/