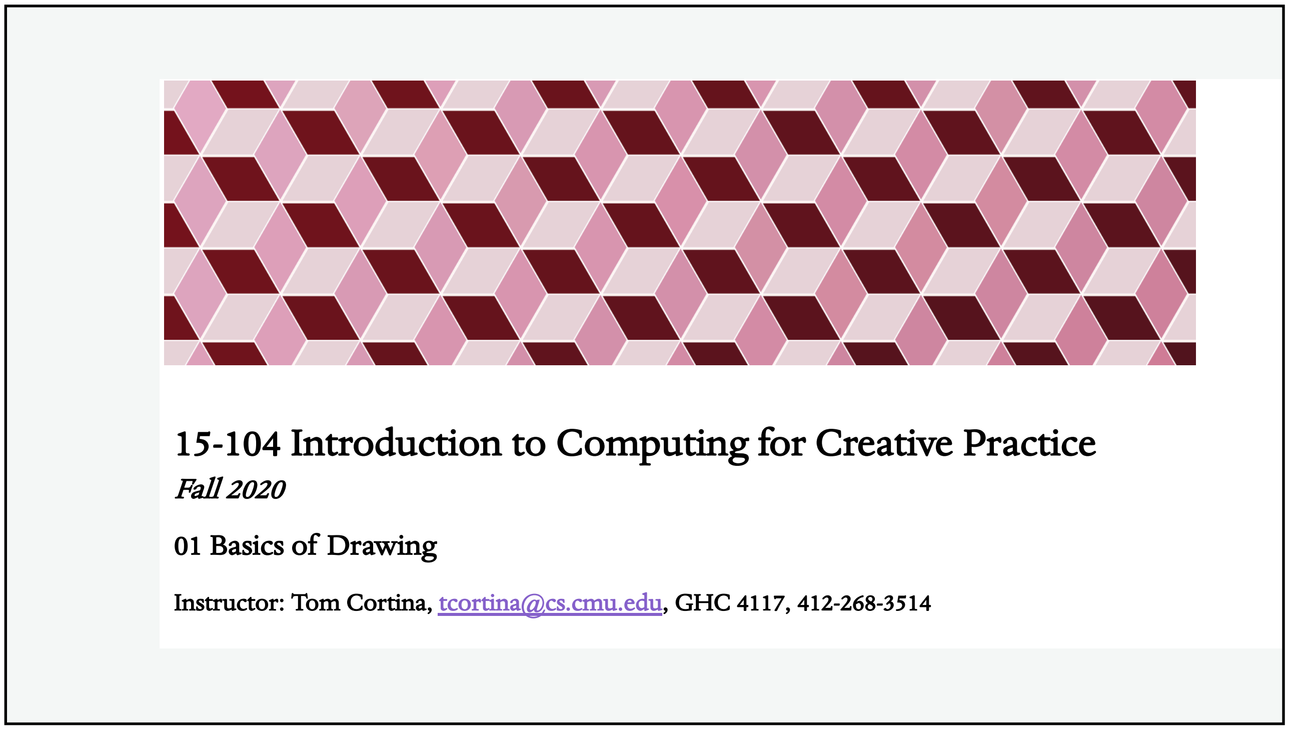


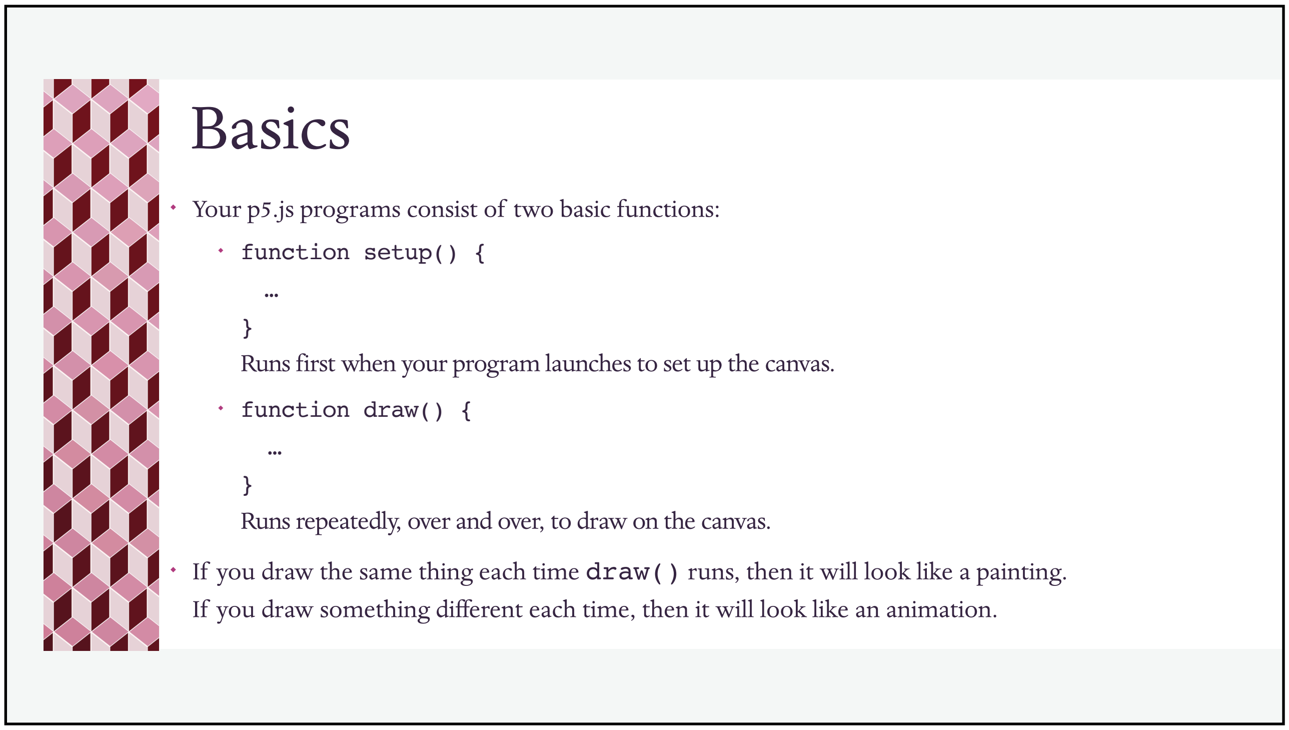
15-104 Introduction to Computing for Creative Practice

Fall 2020

01 Basics of Drawing

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

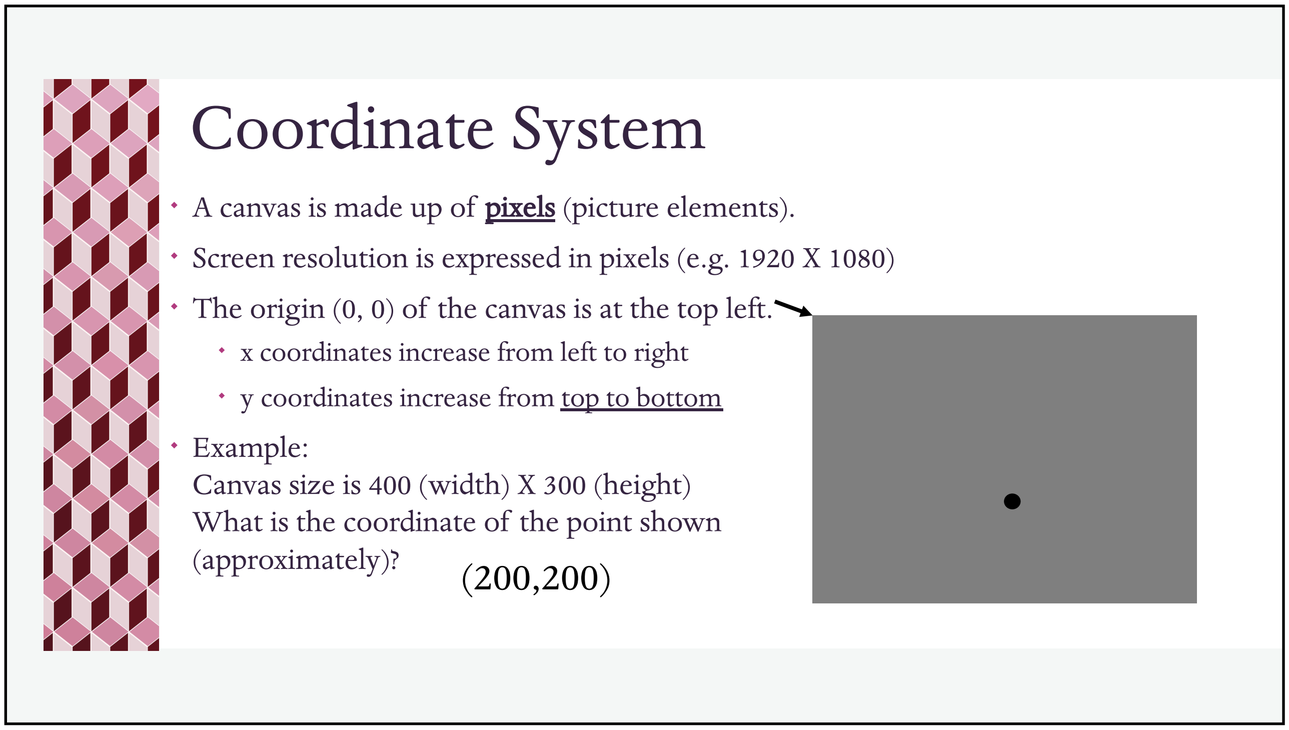
1



Basics

- Your p5.js programs consist of two basic functions:
  - `function setup() {`
    `…`
    `}`
    Runs first when your program launches to set up the canvas.
  - `function draw() {`
    `…`
    `}`
    Runs repeatedly, over and over, to draw on the canvas.
- If you draw the same thing each time `draw()` runs, then it will look like a painting.
  If you draw something different each time, then it will look like an animation.

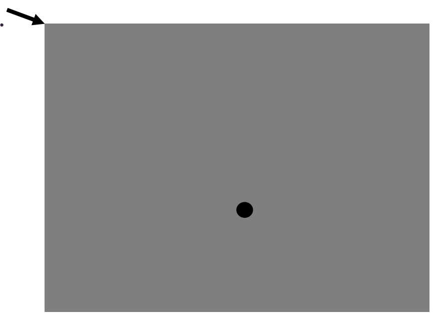
2



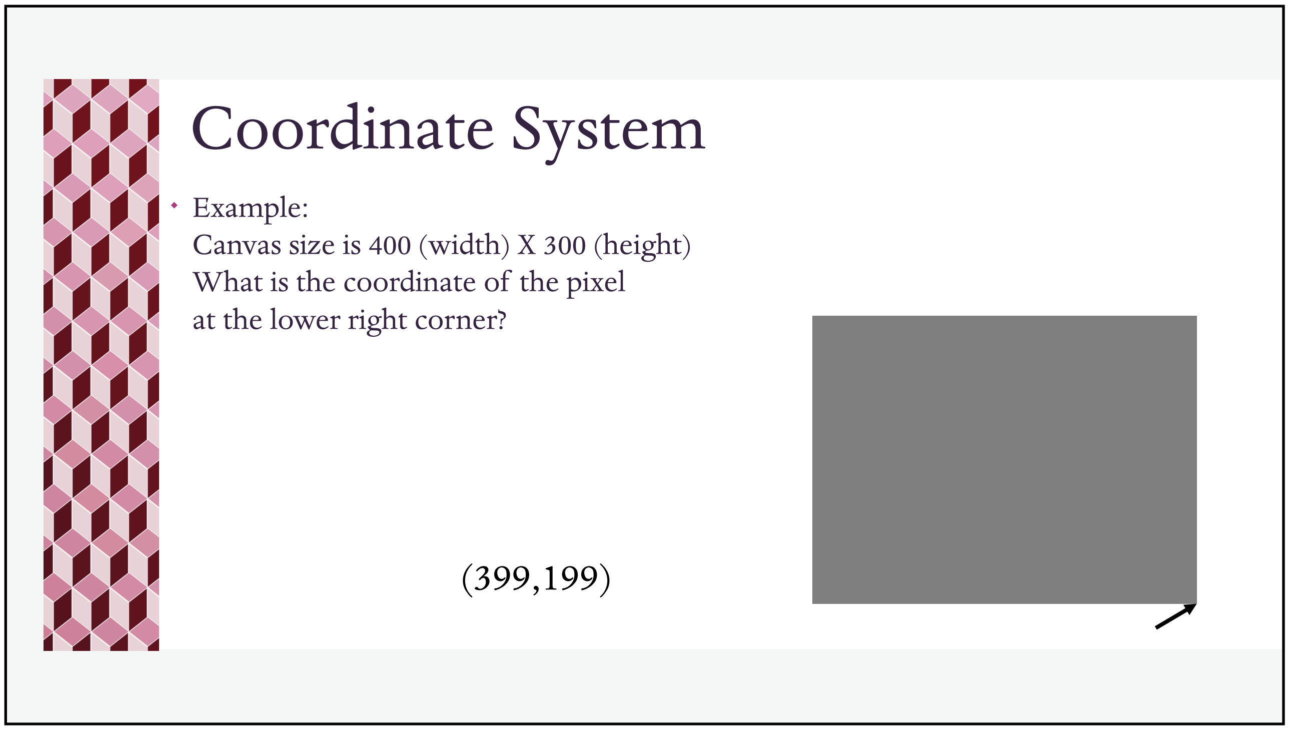
Coordinate System

- A canvas is made up of **pixels** (picture elements).
- Screen resolution is expressed in pixels (e.g. 1920 X 1080)
- The origin (0, 0) of the canvas is at the top left.
  - x coordinates increase from left to right
  - y coordinates increase from top to bottom
- Example:
  Canvas size is 400 (width) X 300 (height)
  What is the coordinate of the point shown (approximately)?

(200,200)



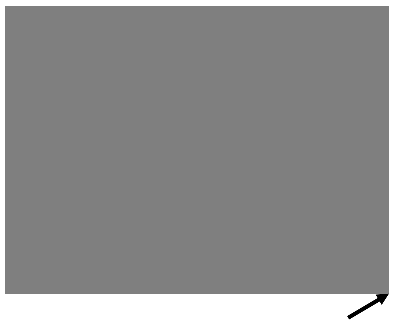
3



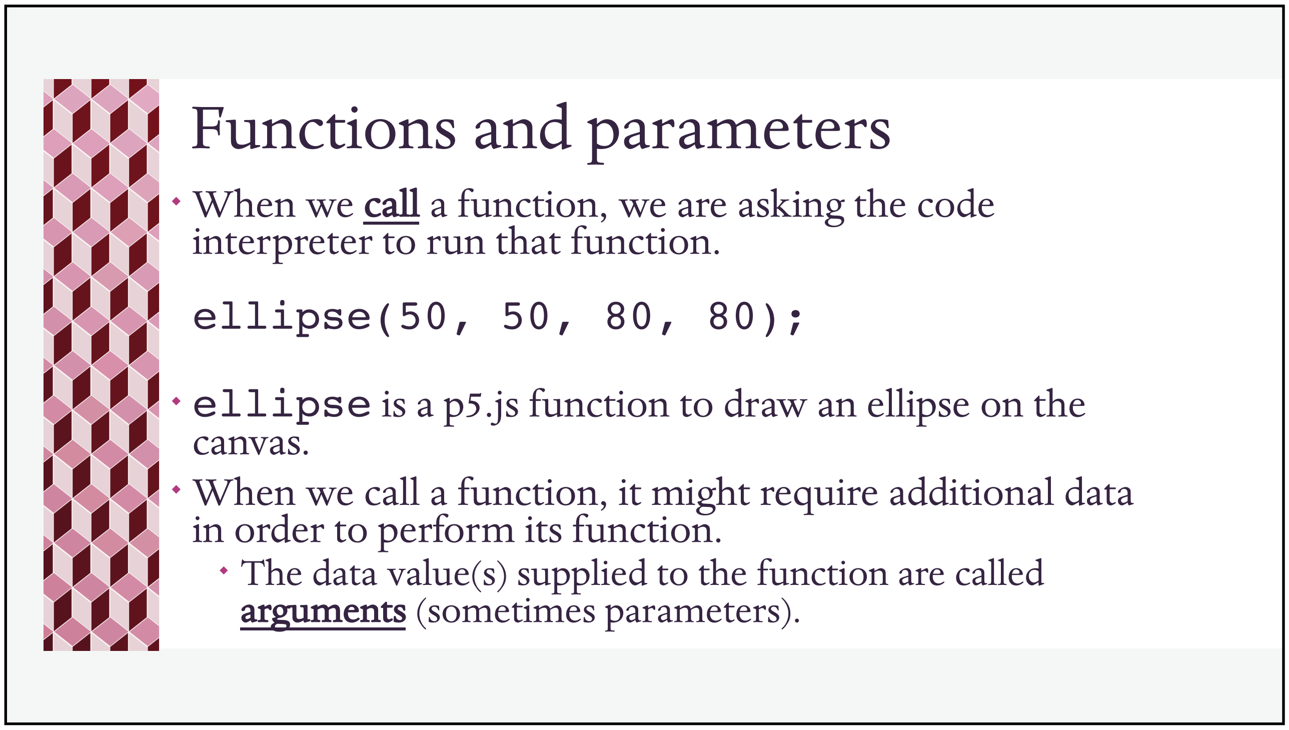
Coordinate System

- Example:
  Canvas size is 400 (width) X 300 (height)
  What is the coordinate of the pixel at the lower right corner?

(399,199)



4



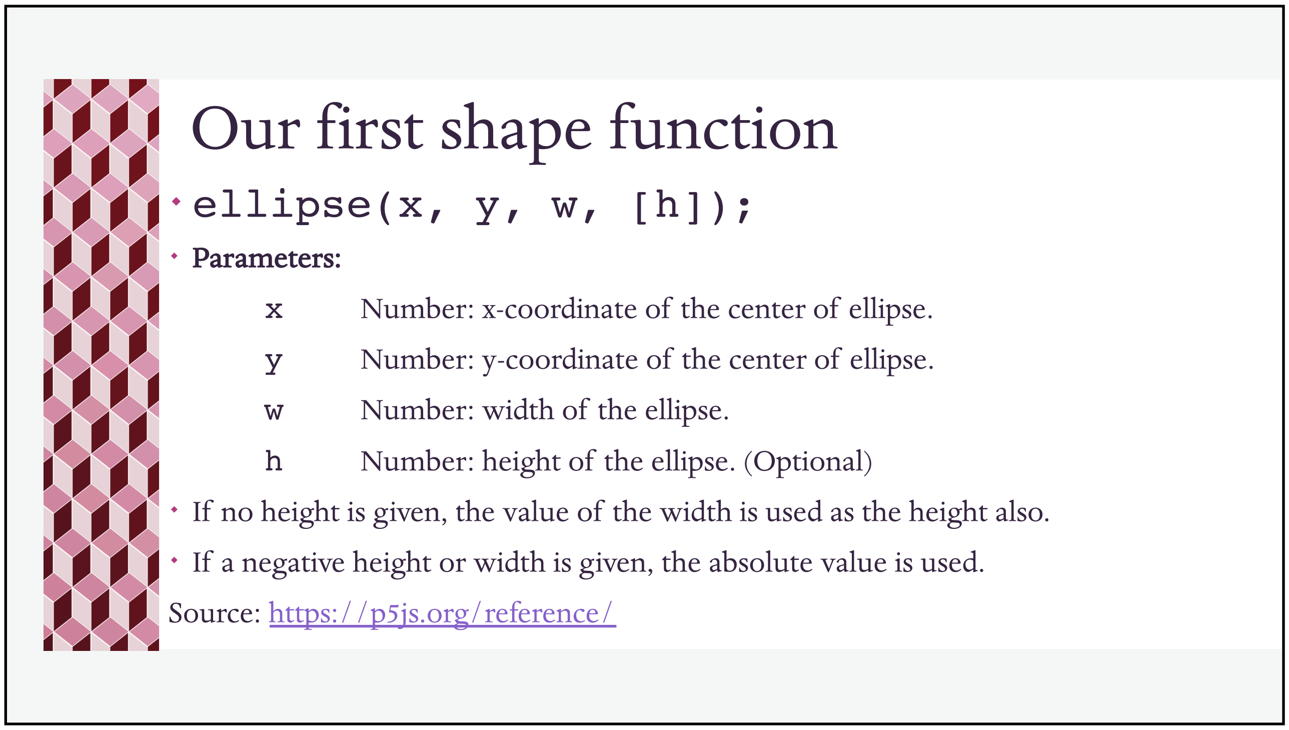
Functions and parameters

- When we **call** a function, we are asking the code interpreter to run that function.

```
ellipse(50, 50, 80, 80);
```

- `ellipse` is a p5.js function to draw an ellipse on the canvas.
- When we call a function, it might require additional data in order to perform its function.
  - The data value(s) supplied to the function are called **arguments** (sometimes parameters).

5



Our first shape function

- `ellipse(x, y, w, [h]);`
- **Parameters:**

`x`	Number: x-coordinate of the center of ellipse.
`y`	Number: y-coordinate of the center of ellipse.
`w`	Number: width of the ellipse.
`h`	Number: height of the ellipse. (Optional)

- If no height is given, the value of the width is used as the height also.
- If a negative height or width is given, the absolute value is used.

Source: https://p5js.org/reference/

6

Parameters vs. Arguments

- In p5.js, there is a function already defined for you to draw an ellipse with four parameters:

```
function ellipse(x, y, w, h) {
   …
}
```

- When we call this function in our program, we supply arguments which will be assigned to the variables specified in the function.

```
ellipse(50, 50, 80, 80);
```

7

Point

- `point(x, y);`
- **Parameters:**

`x`	Number: x-coordinate.
`y`	Number: y-coordinate.

Source: https://p5js.org/reference/

8

Line

- `line(x1, y1, x2, y2);`
- **Parameters:**

`x1`	Number: x-coordinate of the starting point.
`y1`	Number: y-coordinate of the starting point.
`x2`	Number: x-coordinate of the ending point.
`y2`	Number: y-coordinate of the ending point.

Source: https://p5js.org/reference/

9

Stroke and stroke weight

- `stroke(grayvalue);`
- **Parameter:**

  `grayvalue` Number: a value to represent the gray level
  from 0 for maximum gray (black) to 255 for minimum gray (white).

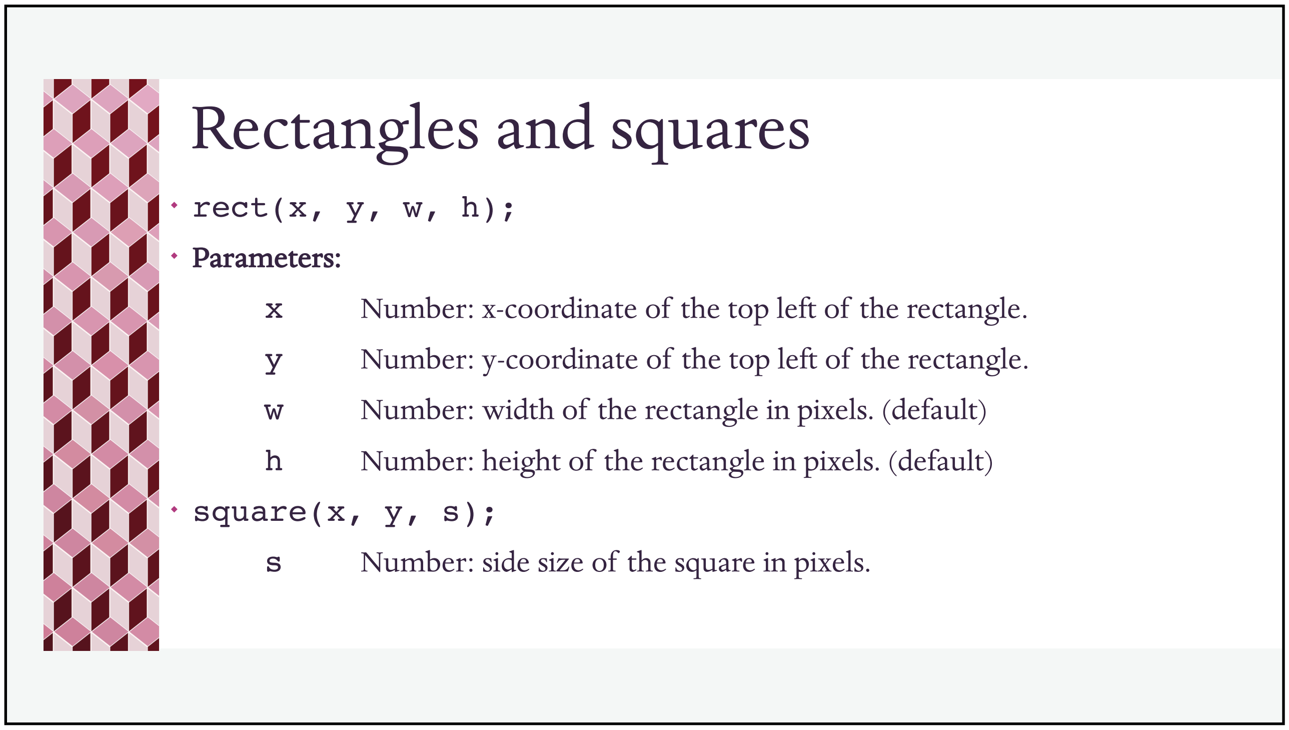
- `strokeWeight(weight);`
- **Parameter:**

  `weight` Number: a value to represent weight of a stroke or line
  in pixels (minimum 1)

Make sure you type `strokeWeight` not `strokeweight` or `StrokeWeight`.

Everything drawn after one of these functions is run will have that stroke or stroke weight until one of these function is called again. Defaults?

10



Rectangles and squares

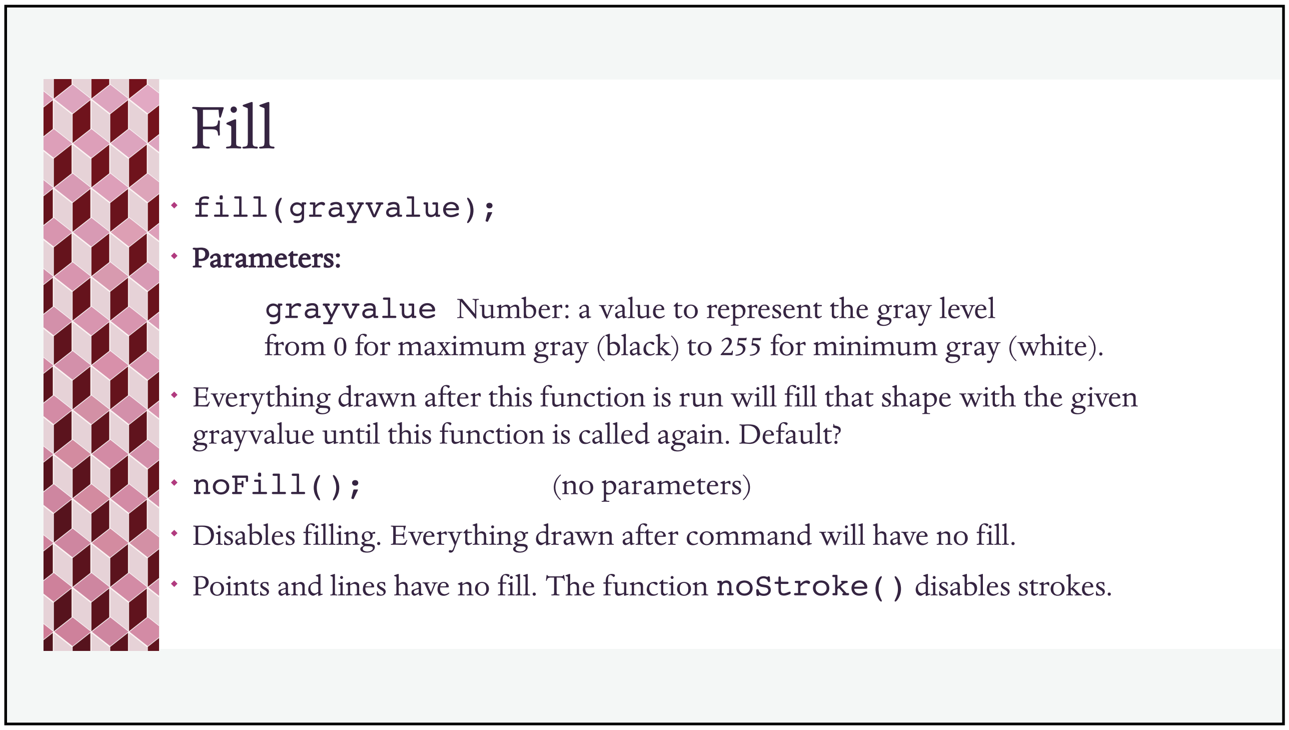
- `rect(x, y, w, h);`
- **Parameters:**

`x`	Number: x-coordinate of the top left of the rectangle.
`y`	Number: y-coordinate of the top left of the rectangle.
`w`	Number: width of the rectangle in pixels. (default)
`h`	Number: height of the rectangle in pixels. (default)

- `square(x, y, s);`

`s`	Number: side size of the square in pixels.

11

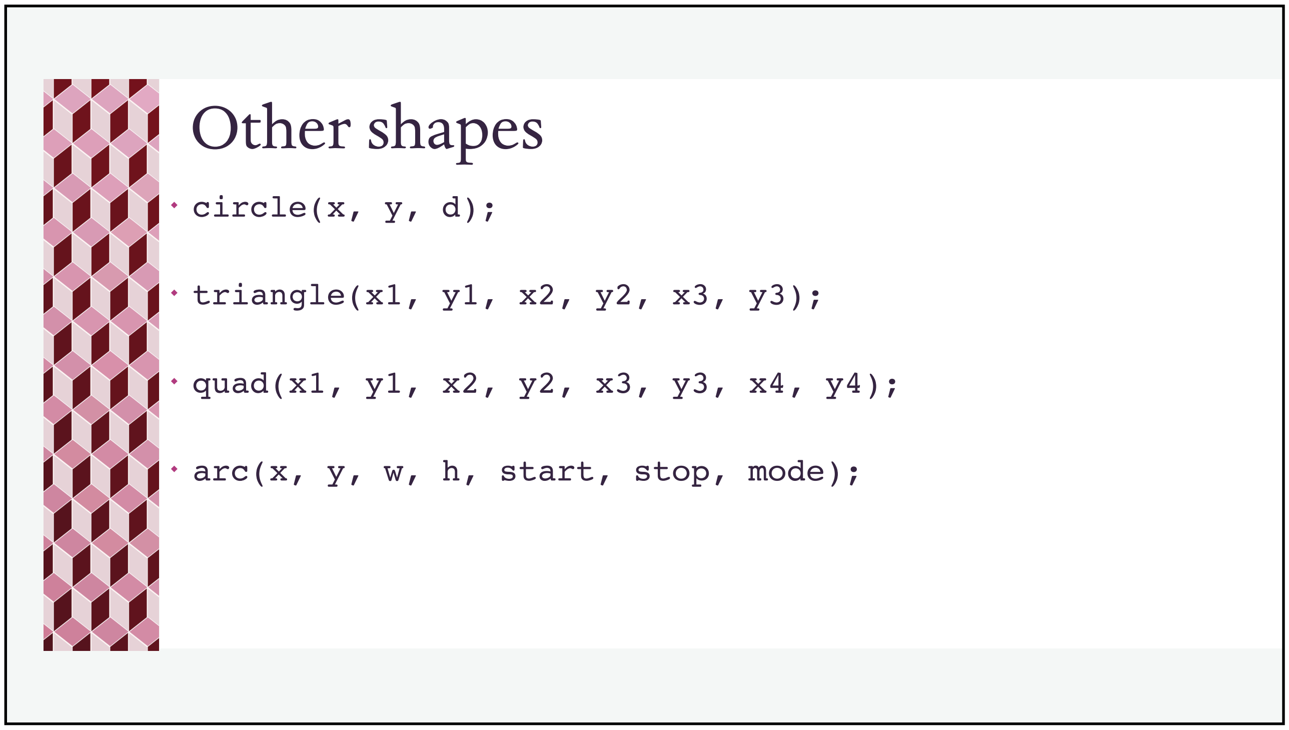


Fill

- `fill(grayvalue);`
- **Parameters:**

  `grayvalue` Number: a value to represent the gray level from 0 for maximum gray (black) to 255 for minimum gray (white).
- Everything drawn after this function is run will fill that shape with the given grayvalue until this function is called again. Default?
- `noFill();` (no parameters)
- Disables filling. Everything drawn after command will have no fill.
- Points and lines have no fill. The function `noStroke()` disables strokes.

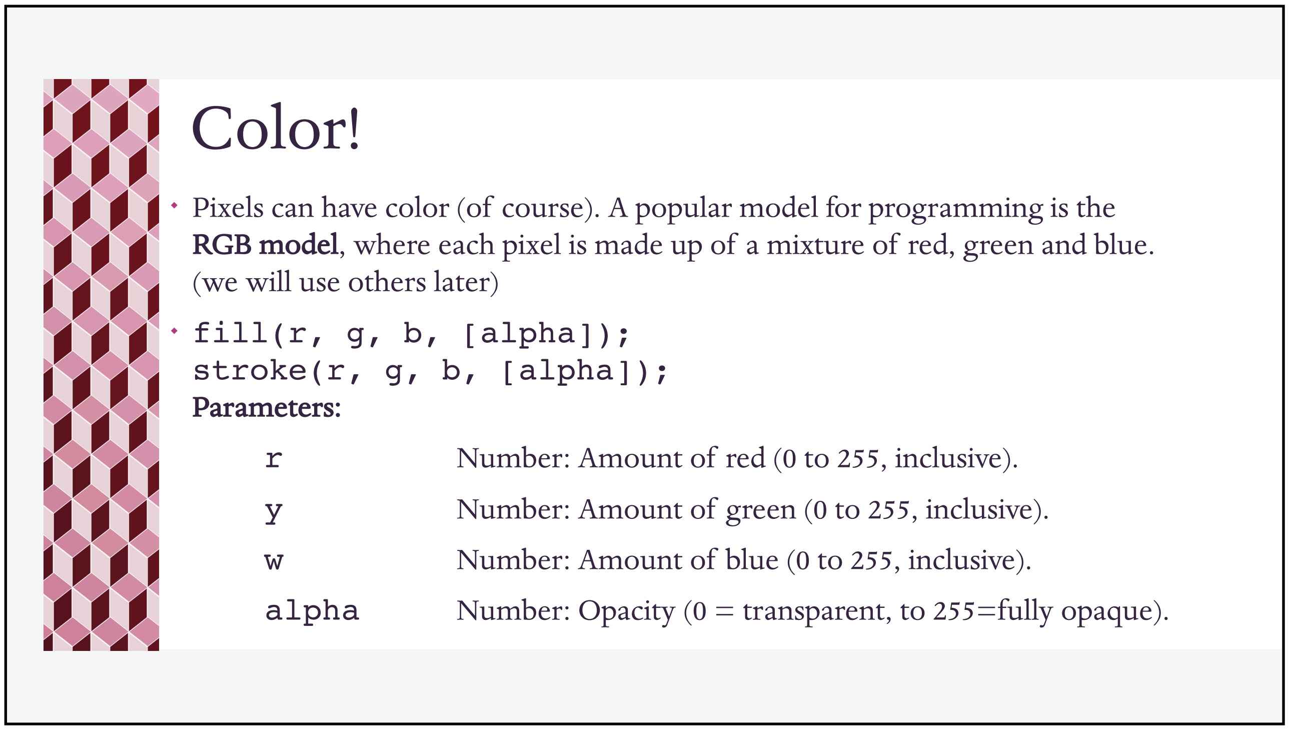
12



Other shapes

- `circle(x, y, d);`
- `triangle(x1, y1, x2, y2, x3, y3);`
- `quad(x1, y1, x2, y2, x3, y3, x4, y4);`
- `arc(x, y, w, h, start, stop, mode);`

13



Color!

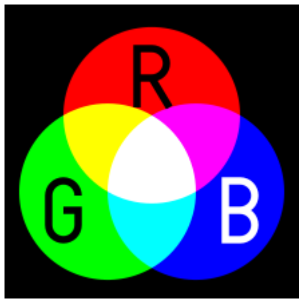
- Pixels can have color (of course). A popular model for programming is the **RGB model**, where each pixel is made up of a mixture of red, green and blue. (we will use others later)
- `fill(r, g, b, [alpha]);`
  `stroke(r, g, b, [alpha]);`
  Parameters:

`r`	Number: Amount of red (0 to 255, inclusive).
`y`	Number: Amount of green (0 to 255, inclusive).
`w`	Number: Amount of blue (0 to 255, inclusive).
`alpha`	Number: Opacity (0 = transparent, to 255=fully opaque).

14

RGB

- The RGB color model is *additive* in the sense that the three light beams are added together, and their light spectra add, wavelength for wavelength, to make the final color's spectrum. (Wikipedia)



- https://www.rapidtables.com/web/color/RGB_Color.html

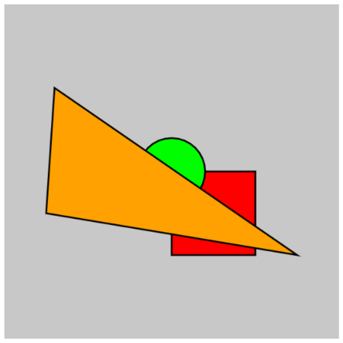
15

Drawing is like painting…

- It's sequential. New paint goes on top of old paint.
  - The order you write the instructions is the order that your painting will be constructed.

Comment for ourselves

```
function draw() {
  background(200);
  fill(255, 0, 0);        // red
  rect(100, 100, 50, 50);
  fill(0, 255, 0);        // green
  circle(100, 100, 40);
  fill(255, 165, 0);      // orange
  triangle(30, 50, 25, 125, 175, 150);
}
```

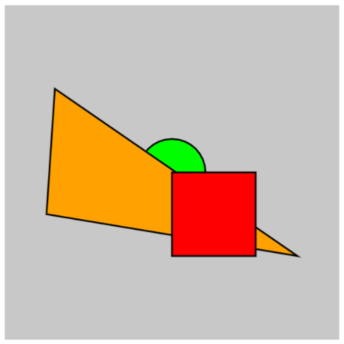


16

Drawing is like painting…

- Same set of commands, why does this painting look different?

```
function draw() {
  background(200);
  fill(0, 255, 0);         // green
  circle(100, 100, 40);
  fill(255, 165, 0);       // orange
  triangle(30, 50, 25, 125, 175, 150);
  fill(255, 0, 0);         // red
  rect(100, 100, 50, 50);
}
```

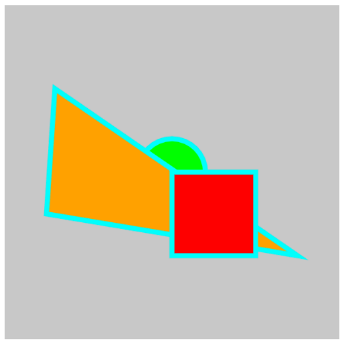


17

One last thing…

In this example, we want to draw only the red rectangle with a cyan border, but all the shapes get a cyan border! Why?

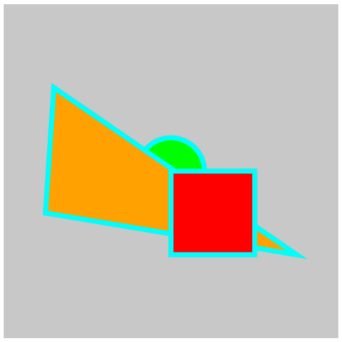
```
function draw() {
  background(200);
  fill(0, 255, 0);         // green
  circle(100, 100, 40);
  fill(255, 165, 0);       // orange
  triangle(30, 50, 25, 125, 175, 150);
  stroke(0, 255, 255);     // cyan
  strokeWeight(3);
  fill(255, 0, 0);         // red
  rect(100, 100, 50, 50);
}
```



18

The `draw` function loops!

- Remember that the `draw` function automatically loops/repeats itself.
- So once we change the stroke color and weight, that color and weight remain in effect for the next repetition of `draw`.
- If you are drawing a painting (static image) and don't want the draw function to repeat itself, you can call the `noLoop()` function at the end of the `draw` function so it doesn't loop back on itself.

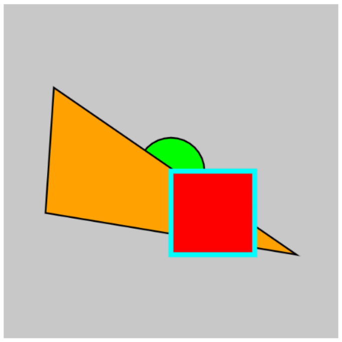


19

noLoop()

```
function draw() {
  background(200);
  fill(0, 255, 0);        // green
  circle(100, 100, 40);
  fill(255, 165, 0);      // orange
  triangle(30, 50, 25, 125, 175, 150);
  stroke(0, 255, 255);    // cyan
  strokeWeight(3);
  fill(255, 0, 0);        // red
  rect(100, 100, 50, 50);
  noLoop();
}
```

That's better!



20