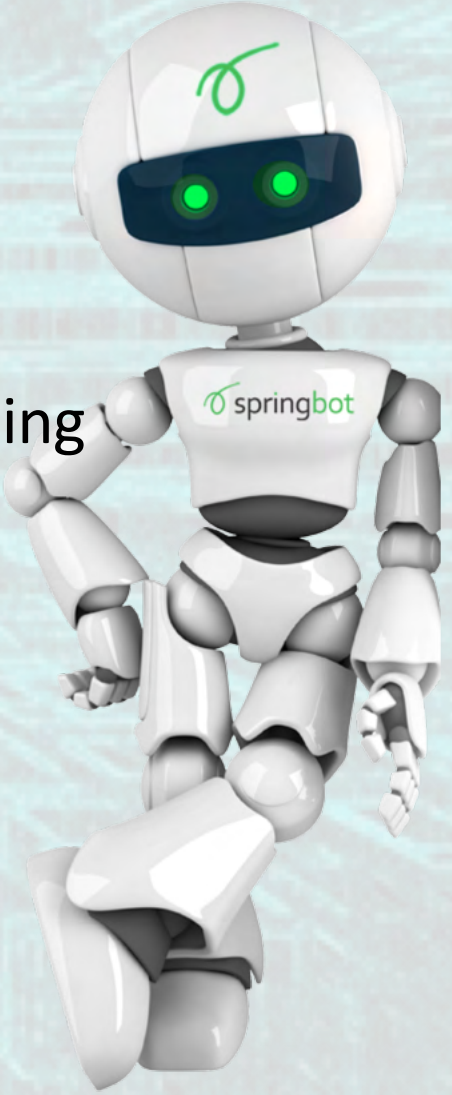
Processing

A hands on visual introduction to programming

Using JavaScript and p5js.org

stuff2oh.com - stuff20h@gmail.com



What is Processing ??

- “Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts.”
- “Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology.”
- “There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.”

- https://processing.org

What is JavaSript?

- An interpreted (not compiled) scripting language that is mainly used in web pages to create dynamic (changing) content.
- However, it can also be used outside the browser as a foundation for doing other things as well. (Processing, Node.js and even inside PDF files to make them interactive.)
- It is very easy to learn and is more forgiving than other, more formal languages such as Java or C.

Lilly, Did you know that Processing's native language is Java, but it also supports JavaScript, Python, and other languages as well?

Yeah Marvin, I heard that. Really, Processing is just a set of 'commands' (or what geeks like to call an A.P.I.) that support drawing, animation, interaction, etc. The core set of commands are the same, but the way you write the code is language specific.

You know, there are many languages in the world, but probably all have verbs like draw, move and nouns like circle, box, triangle. Am I mewing any sense here?

https://editor.p5js.org – The Editor

Connect to WiFi first. Then, in your browser, go to https://editor.p5js.org
Get familiar with the main stuff !!

Stop your program

Menu Bar

Run your program

This is where you type in your 'sketch' (code). Notice the line numbers.

https://editor.p5js.org

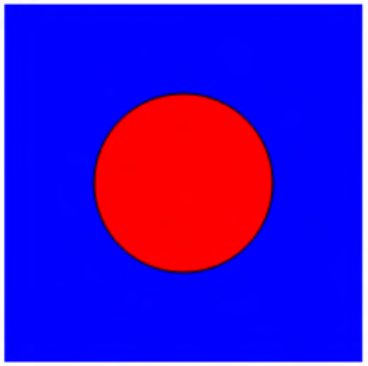
p5* File Edit Sketch Help English Log in or Sign up

Auto-refresh My Sketch

sketch.js

```
1  function setup() {
2    createCanvas(200, 200);
3    console.log("Hello !!");
4  }
5
6  function draw() {
7    background(0,0,255);
8    fill(255,0,0);
9    circle(100,100,100);
10 }
```

Preview



Settings

General settings Accessibility

Theme
Light Dark High Contrast

Text Size
Decrease 18 Increase

Autosave
On Off

Autoclose Brackets and Quotes
On Off

Word Wrap
On Off

Console Clear

Hello !!

This is the 'console'. Where you see messages from your program and any errors too if there are..

Blah Blah Blah… Text overload? Press here !!

The 'preview' window is where you see the magnificent output of your program. It is where out 'canvas' lives.

Useful for settings to make the text easier to read if you need to. And other things too.

Our first program – "Hello World"

```
function setup() {
  createCanvas(200, 200);
  console.log("Hello World !!");
}

function draw() {
  background(0,0,255);
  fill(255,0,0);
  circle(100,100,100);
}
```

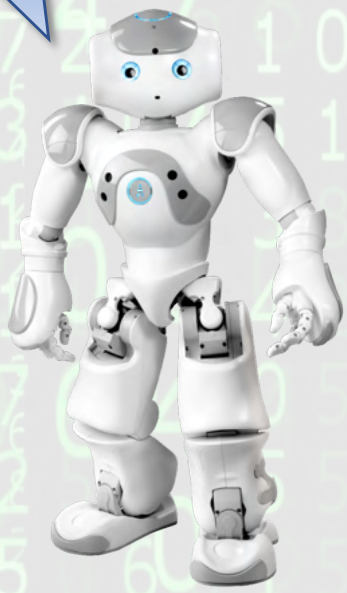
A 'function' is just a container for code. The {} , or 'brackets' are like the sides of a box. They keep stuff inside. The function 'setup' is executed only once. Can you guess what is does?

The draw function is executed over and over again. (for animation) But here, nothing changes, so that is not obvious. Try putting a log statement here. What do you see in the console?

Greetings organic lifeform. My name is Rob. Go ahead and type in this program and see what is does. Discuss each command.

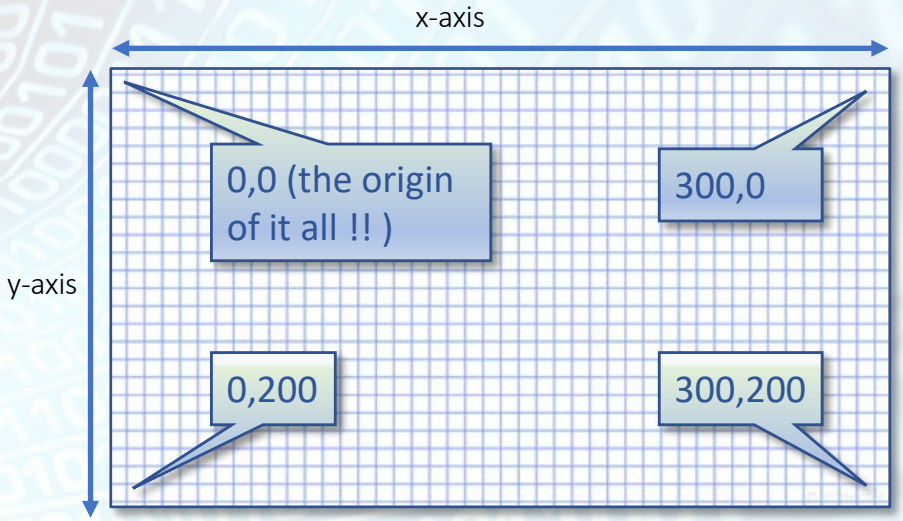
What are these three numbers? They represent the red, green, blue (RGB) parts of a color. The values can be between 0 and 255. 0 means no color at all, while 128 would be medium brightness and 255 would be a full amount of that color. Change each of the values. What happens?

Don't Panic !! If you make any mistakes, we have the backspace key, ya know !!



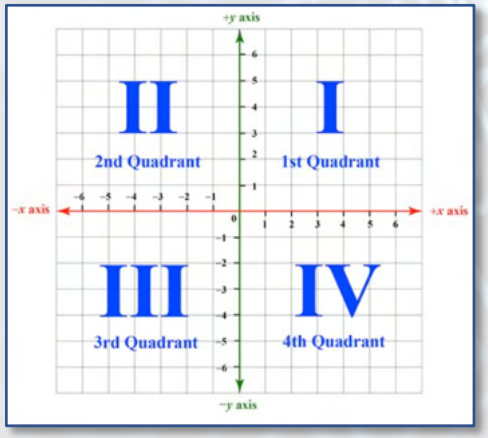
Life on the grid….

Numbers on this grid are presented in the format of (x,y). That is, the x number comes first, then the y number. X numbers define a point's position left to right while the y number is for top to bottom.




```
createCanvas(300, 200);
```

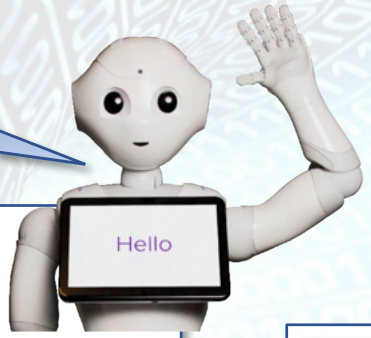
You have probably seen something like this in a math class where you work with graphs and such. This is known as the 'Cartesian Coordinate System'. It's like this 'cause it has negative numbers. Processing does not use this type of system to refer to point on the screen.



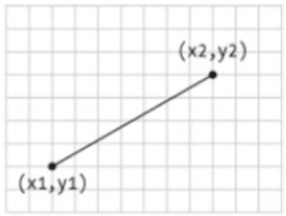
Open your mind to the possibility that your canvas is like a piece of graph paper. It's 'origin' is in the upper left hand corner has the coordinates of 0,0. That is x = 0 and y = 0. In this example, we are creating a canvas that is 300 'pixels' (little dots) wide and 200 pixels tall.

Basic shapes and their commands

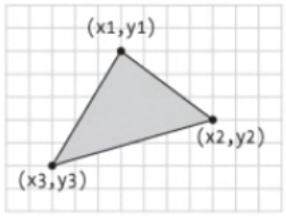
Uh, don't forget about little ol' point. Even though he is just dot and is hard to see.
- point(x,y);



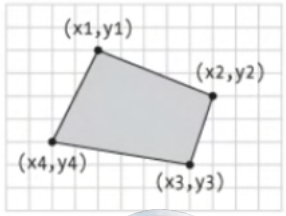
Notice that the 'origin' of the rectangle is in the upper left hand corner. PRO TIP: using the command 'rectMode(CENTER);' moves the origin to the center instead. If you want it to be that way.



line(x1, y1, x2, y2)

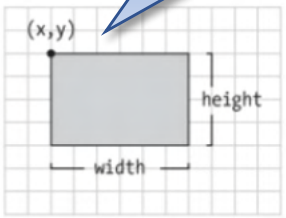


triangle(x1, y1, x2, y2, x3, y3)



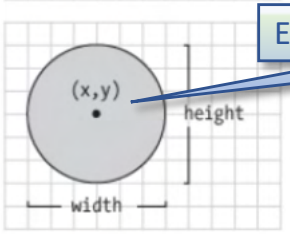
quad(x1, y1, x2, y2, x3, y3, x4, y4)

rect is short for rectangle. You get a square if width and height are equal.

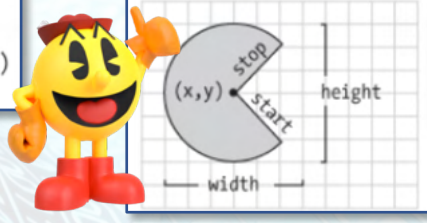


rect(x, y, width, height)

Ellipse 'origin' is in the middle.



ellipse(x, y, width, height)



arc(x, y, width, height, start, stop)

An 'ellipse' (oval) is a circle whose width and height are not equal. How can you make a circle?

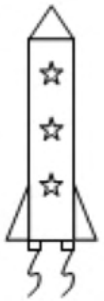
LOL, this suspiciously looks like page 20 from the 'Getting Started with p5.js' book.
- by Lauren McCarthy, Casey Reas, and Ben Fry

PRO TIP: use angleMode(); in your setup() method so that angles can be expressed in degrees. By default, they take 'radians'

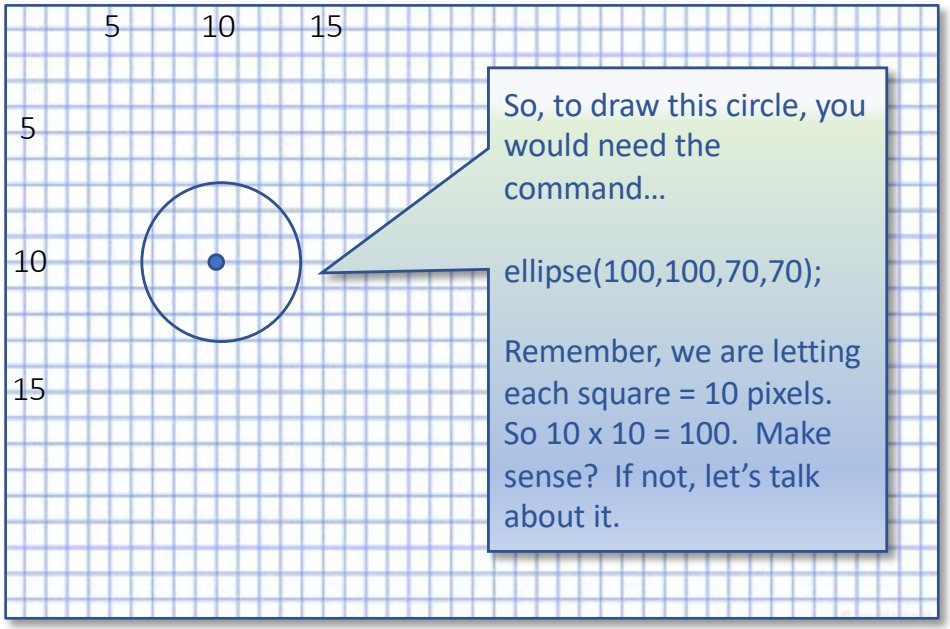
Basic shapes in everyday things…

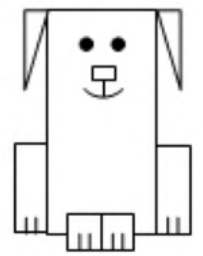
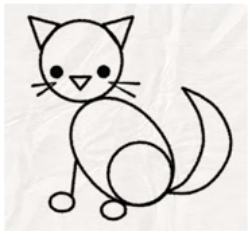
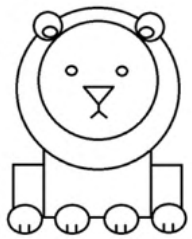


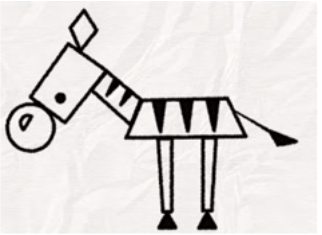
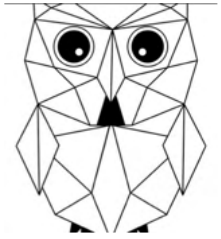
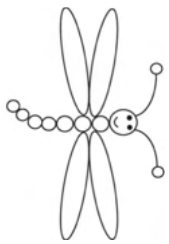
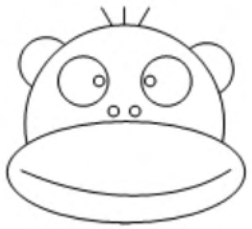
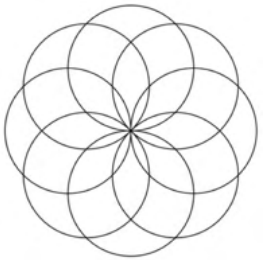


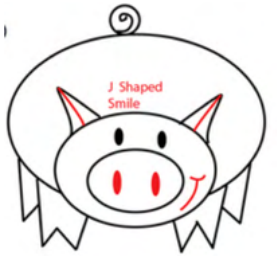


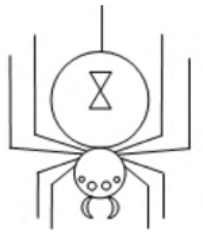

Get some graph paper and start planning out your drawing. That is, what point will define your image. Pick something very simple at first. Let each square be 10 pixels.







Making shapes not so plane (lol) !!

p5* File Edit Sketch Help

Auto-refresh My Sketch

sketch.js

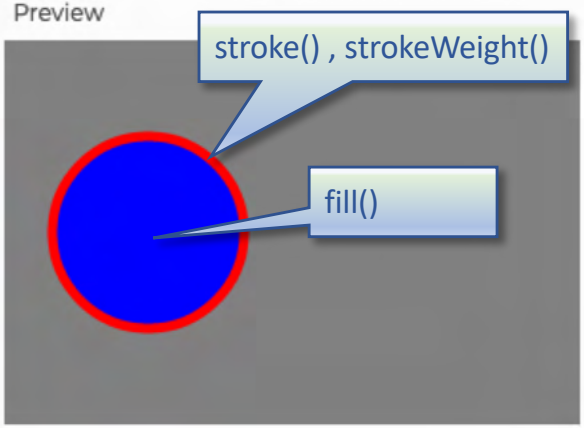
```
1  function setup() {
2    createCanvas(300, 200);
3  }
4
5  function draw() {
6    background(128);
7
8    fill(0,0,255);
9    stroke(255,0,0);
10   strokeWeight(5);
11   circle(75,100,100);
12 }
```

Console

Commands such as background(), fill(), and stroke() can take a single number to represent a level of grey, or 3 numbers (RBG) for color.

To control the thickness or 'stroke' of this circle, we use strokeWeight() to make it 5 pixels wide. The stroke() command sets it to red.

Preview



By default, shapes you draw will be filled with the color white and have a stroke of 1 pixel in a black color. BTW, the 'stroke' means the line that defines the shape. The 'fill' is the color inside the shape.

It’s hip not to be square !!

p5* File Edit Sketch Help

Auto-refresh My Sketch

sketch.js

```
function setup() {
  createCanvas(300, 200);
  strokeWeight(10);
}

function draw() {
  background(128);

  strokeJoin(ROUND);
  rect(25, 25, 50, 50);
  strokeJoin(BEVEL);
  rect(125, 25, 70, 50);

  strokeCap(SQUARE);
  line(25, 100, 100, 125);
  strokeCap(ROUND);
  line(125, 100, 250, 125);

}
```

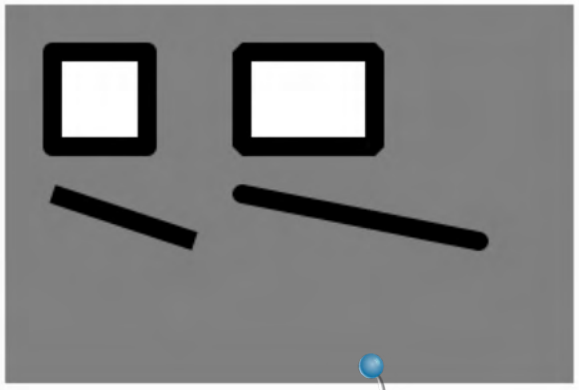
Console

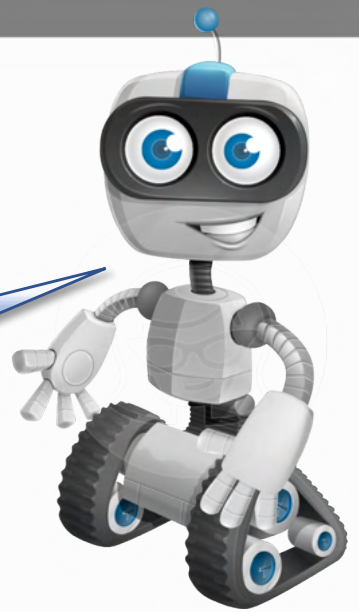
strokeJoin() lets you say how the edges of a shape look. You can think of this command as ‘how should my shape look at the places where the lines that define is meet”. In the case of rectangles, that means the corners.

strokeCap() lets you say how the ends of a line look. ‘To cap’ means putting something on the end of it. Think about it, what do you put on your head on a sunny day? A ‘cap’, right?.

To make the edge of a rectangle not square we can either ‘bevel’ it or round it. Bevel means to kind of chop off the sharp edge of the corners. This is done with a straight line versus a curve. In other drawing programs, this is known as a ‘chamfer’.

Preview





Getting things in order !!

p5* File Edit Sketch Help

Auto-refresh My Sketch

sketch.js

```
1  function setup() {
2    createCanvas(300, 200);
3  }
4
5  function draw() {
6    background(128);
7
8    fill(255,255,0);
9    ellipse(50, 50, 75, 75);
10   fill(255,0,255);
11   ellipse(125, 50, 100, 75);
12
13   noStroke();
14   ellipse(50, 150, 75, 75);
15   fill(255,0,255);
16   stroke(1);
17   noFill();
18   ellipse(125, 150, 100, 75);
19 }
```

Console

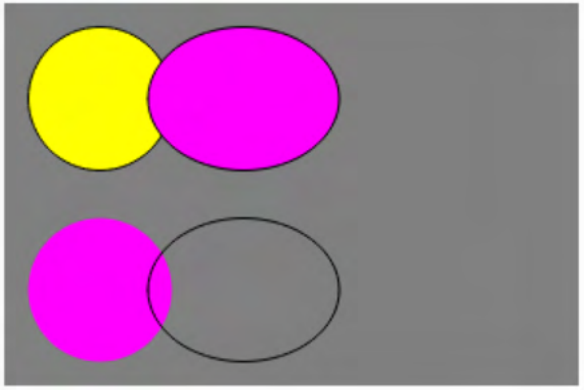
This oval will be on top of the circle because it was second to be coded and is also overlapping.

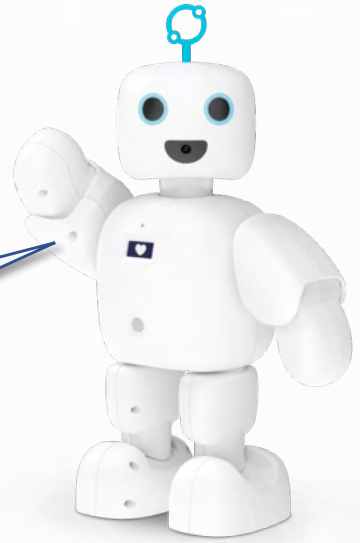
We have to turn stroke back 'on' since we turned it off for the last shape.

Using noFill() allows anything underneath a shape to show through.

The order in which you draw your shapes is important. They basically stack on top of one another. Also, it is possible to turn off fill and stroke for a shape. Just use noFIll() and noStroke().

Preview





Always be transparent (and honest)

p5* File Edit Sketch Help

Auto-refresh My Sketch

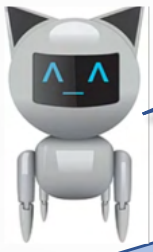
sketch.js

```
1  function setup() {
2    createCanvas(300, 200);
3  }
4
5  function draw() {
6
7    // Light blue color
8    background(204, 226, 225);
9
10   // Notice there is a 4th number on fill now
11   // This controls how transparent the shape is.
12   // 0 is invisible, 255 is completely solid.
13   fill(255, 0, 0, 128);
14   ellipse(100, 100, 100, 100); // It's Red
15
16   fill(0, 255, 0, 128);
17   ellipse(150, 60, 100, 100); // It's Green
18
19   fill(0, 0, 255, 128);
20   ellipse(150, 120, 100, 100); // It's Blue
21 }
```

Console

This is a special line in our code called a 'comment'. It starts with a '//'. The computer just ignores anything after this. Read all the comments in the programs from now on. Good programmers use tons of comments to clarify and explain what they are trying to accomplish with their code.

Notice it can be on a line by itself or even after some code. Comments cannot be before code on the same line, just in case you were wondering.



Did you read all the comments?

Transparency can also be referred to as 'opacity'. The word 'opaque' means NOT to be transparent. So if a shape is totally opaque (255), it is solid and will not allow anything to show from under neath.

Build your own shape - Polygons

p5* File Edit Sketch Help

Auto-refresh My Sketch

sketch.js

```
1  function setup() {
2    createCanvas(250, 250);
3  }
4
5  function draw() {
6
7    // Light blue color
8    background(204, 226, 225);
9
10   // Draw a star...
11   beginShape();
12     vertex(100,100); // Top
13     vertex(125,25);
14     vertex(150, 100);
15
16     vertex(225, 100); // Right
17     vertex(150, 150);
18
19     vertex(200, 225); // Bottom Right
20     vertex(125, 175);
21
22     vertex(75, 225); // Bottom Left
23     vertex(100, 150);
24
25     vertex(25,100); // Left
26     // Back to our starting point !!
27     vertex(100, 100);
28   endShape();
29 }
```

Console Clear

Preview

A 'polygon' is any closed shape with three or more sides. So, a triangle and a quad are polygons of 3 and 4 sides respectfully

This bug has infected our program and messed up our perfect star!!

A 'vertex' is a point within a 'polygon'. Our star has many vertices or points that define it. Each has a unique coordinate. (x,y)

See how these comments identify which part of the star we are trying to draw. This info might be helpful in 'debugging' this star.

In Processing, we use beginShape() and endShape() to define custom shapes.

Interact with the mouse

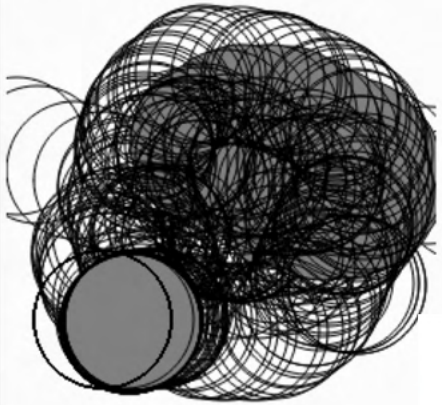
p5* File Edit Sketch Help

Auto-refresh My Sketch

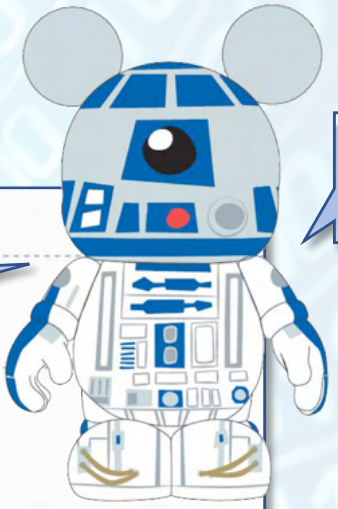
sketch.js

```
1  function setup() {
2    createCanvas(250, 250);
3  }
4
5  function draw() {
6
7    if (mouseIsPressed) {
8      fill(128);
9    } else {
10     noFill();
11   }
12
13     ellipse(mouseX, mouseY, 80, 80);
14 }
```

Preview



Console



R2 DMouse

Beep, beep, squeak, beep.

In programming, 'if' & 'else' define what is known as a 'conditional'. That is, make a decision and do one thing or another. Here, the {} hold the code based on if you press the mouse button.

It's behind me, isn't it?

The 'global variables' mouseX & mouseY hold the current position of the mouse pointer. This, of course, corresponds exactly where the pointer is within your defined canvas.

Remember how we talked about the draw() function being useful for animation? This is our first insight into what's going on. The draw() function is called many times per second and each time we move the mouse, its location changes and also the location of the circle being drawn.