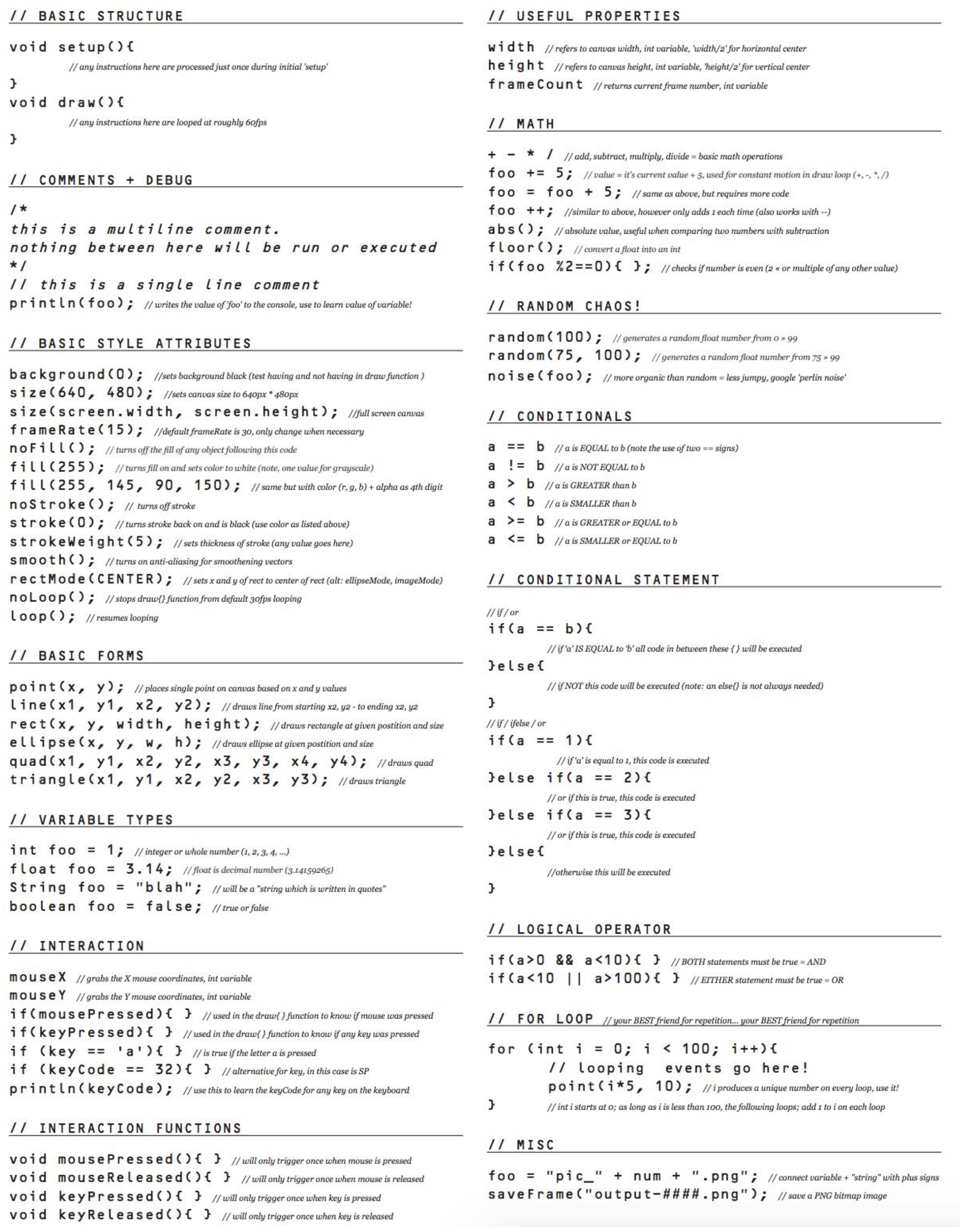
# Processing Formula Sheet



## // BASIC STRUCTURE

```
void setup(){
        // any instructions here are processed just once during initial 'setup'
}
void draw(){
        // any instructions here are looped at roughly 60fps
}
```

## // COMMENTS + DEBUG

```
/*
this is a multiline comment.
nothing between here will be run or executed
*/
// this is a single line comment
println(foo); // writes the value of 'foo' to the console, use to learn value of variable!
```

## // BASIC STYLE ATTRIBUTES

```
background(0); //sets background black (test having and not having in draw function )
size(640, 480); //sets canvas size to 640px * 480px
size(screen.width, screen.height); //full screen canvas
frameRate(15); //default frameRate is 30, only change when necessary
noFill(); // turns off the fill of any object following this code
fill(255); // turns fill on and sets color to white (note, one value for grayscale)
fill(255, 145, 90, 150); // same but with color (r, g, b) + alpha as 4th digit
noStroke(); // turns off stroke
stroke(0); // turns stroke back on and is black (use color as listed above)
strokeWeight(5); // sets thickness of stroke (any value goes here)
smooth(); // turns on anti-aliasing for smoothening vectors
rectMode(CENTER); // sets x and y of rect to center of rect (alt: ellipseMode, imageMode)
noLoop(); // stops draw{} function from default 30fps looping
loop(); // resumes looping
```

## // BASIC FORMS

```
point(x, y); // places single point on canvas based on x and y values
line(x1, y1, x2, y2); // draws line from starting x2, y2 - to ending x2, y2
rect(x, y, width, height); // draws rectangle at given postition and size
ellipse(x, y, w, h); // draws ellipse at given postition and size
quad(x1, y1, x2, y2, x3, y3, x4, y4); // draws quad
triangle(x1, y1, x2, y2, x3, y3); // draws triangle
```

## // VARIABLE TYPES

```
int foo = 1; // integer or whole number (1, 2, 3, 4, ...)
float foo = 3.14; // float is decimal number (3.14159265)
String foo = "blah"; // will be a "string which is written in quotes"
boolean foo = false; // true or false
```

## // INTERACTION

```
mouseX // grabs the X mouse coordinates, int variable
mouseY // grabs the Y mouse coordinates, int variable
if(mousePressed){ } // used in the draw{ } function to know if mouse was pressed
if(keyPressed){ } // used in the draw{ } function to know if any key was pressed
if (key == 'a'){ } // is true if the letter a is pressed
if (keyCode == 32){ } // alternative for key, in this case is SP
println(keyCode); // use this to learn the keyCode for any key on the keyboard
```

## // INTERACTION FUNCTIONS

```
void mousePressed(){ } // will only trigger once when mouse is pressed
void mouseReleased(){ } // will only trigger once when mouse is released
void keyPressed(){ } // will only trigger once when key is pressed
void keyReleased(){ } // will only trigger once when key is released
```

## // USEFUL PROPERTIES

```
width // refers to canvas width, int variable, 'width/2' for horizontal center
height // refers to canvas height, int variable, 'height/2' for vertical center
frameCount // returns current frame number, int variable
```

## // MATH

```
+ - * / // add, subtract, multiply, divide = basic math operations
foo += 5; // value = it's current value + 5, used for constant motion in draw loop (+, -, *, /)
foo = foo + 5; // same as above, but requires more code
foo ++; //similar to above, however only adds 1 each time (also works with --)
abs(); // absolute value, useful when comparing two numbers with subtraction
floor(); // convert a float into an int
if(foo %2==0){ }; // checks if number is even (2 « or multiple of any other value)
```

## // RANDOM CHAOS!

```
random(100); // generates a random float number from 0 » 99
random(75, 100); // generates a random float number from 75 » 99
noise(foo); // more organic than random = less jumpy, google 'perlin noise'
```

## // CONDITIONALS

```
a == b // a is EQUAL to b (note the use of two == signs)
a != b // a is NOT EQUAL to b
a > b // a is GREATER than b
a < b // a is SMALLER than b
a >= b // a is GREATER or EQUAL to b
a <= b // a is SMALLER or EQUAL to b
```

## // CONDITIONAL STATEMENT

```
// if / or
if(a == b){
        // if 'a' IS EQUAL to 'b' all code in between these { } will be executed
}else{
        // if NOT this code will be executed (note: an else{} is not always needed)
}
// if / ifelse / or
if(a == 1){
        // if 'a' is equal to 1, this code is executed
}else if(a == 2){
        // or if this is true, this code is executed
}else if(a == 3){
        // or if this is true, this code is executed
}else{
        //otherwise this will be executed
}
```

## // LOGICAL OPERATOR

```
if(a>0 && a<10){ } // BOTH statements must be true = AND
if(a<10 || a>100){ } // EITHER statement must be true = OR
```

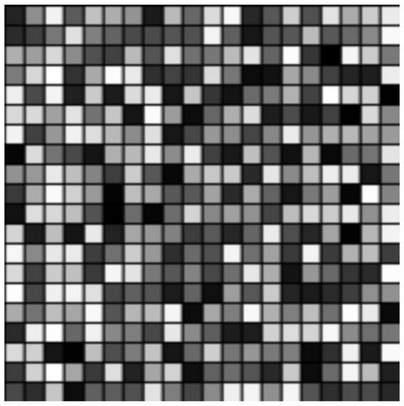
## // FOR LOOP // your BEST friend for repetition... your BEST friend for repetition

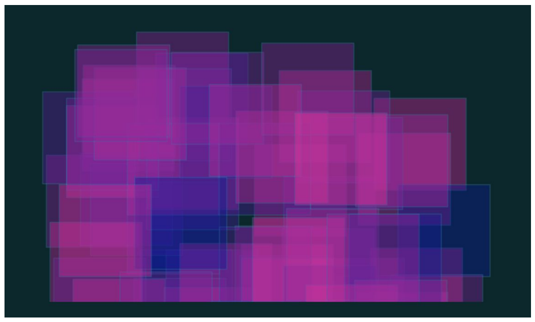
```
for (int i = 0; i < 100; i++){
        // looping  events go here!
        point(i*5, 10); // i produces a unique number on every loop, use it!
}       // int i starts at 0; as long as i is less than 100, the following loops; add 1 to i on each loop
```

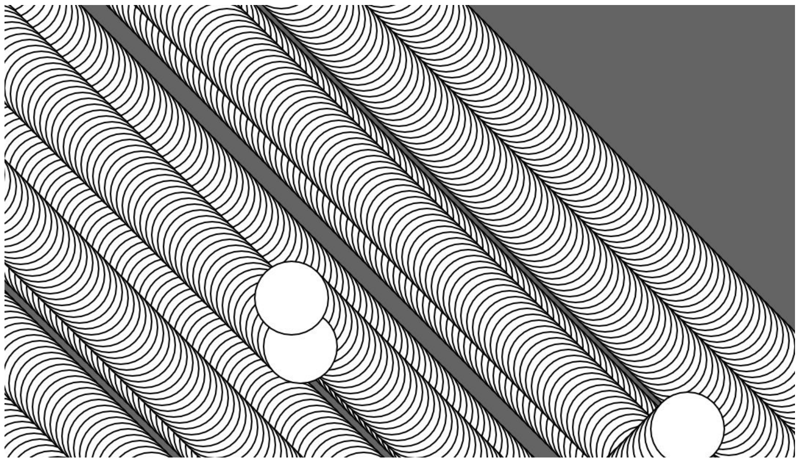
## // MISC

```
foo = "pic_" + num + ".png"; // connect variable + "string" with plus signs
saveFrame("output-####.png"); // save a PNG bitmap image
```

## Think about it…







## Tutorials

### Demo 1: Bubble mouse path

Concepts: function calls, mouse input, color mode, speed.

### Demo 2: Line manipulation

Concepts: loops, mouse input, bezier, redrawing, strobing lines, strokes, random values.

## Demo 3: Recursive trees

Concepts: frame rates, translation, rotation, 2D transformations, transformation matrices, scaling.

## Demo 4: Moving objects

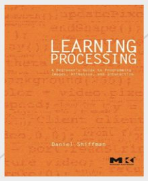
Concepts: classes, methods, damping,

## Libraries:

https://processing.org/reference/libraries/

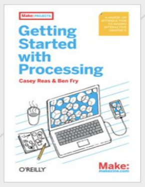
## Reference materials:

**Learning Processing: A Beginner's Guide**



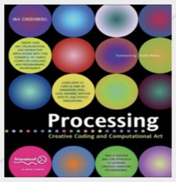
An excellent book for beginners. Covers a lot of topics. Perfectly explained.

**Getting Started with Processing.**



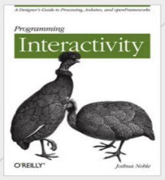
A good book as a complement to "Learning Processing", both of them make a good introduction.

**Processing: Creative coding and Computational Art.**



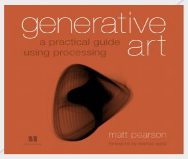
Another good alternative to start with Processing. Contains a diverse amount of examples.

**Programming Interactivity.**



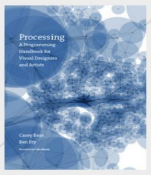
It's an introduction to Processing, Openframeworks and Arduino. Covers many aspects of of the three.

**Generative Art: a practical guide using processing.**



It's a Generative art oriented book. Covers some Processing based projects and comes with a lot of examples to download.

**Processing: A programming handbook.**



This book reviews some important aspects to go further in the task of learning Processing.