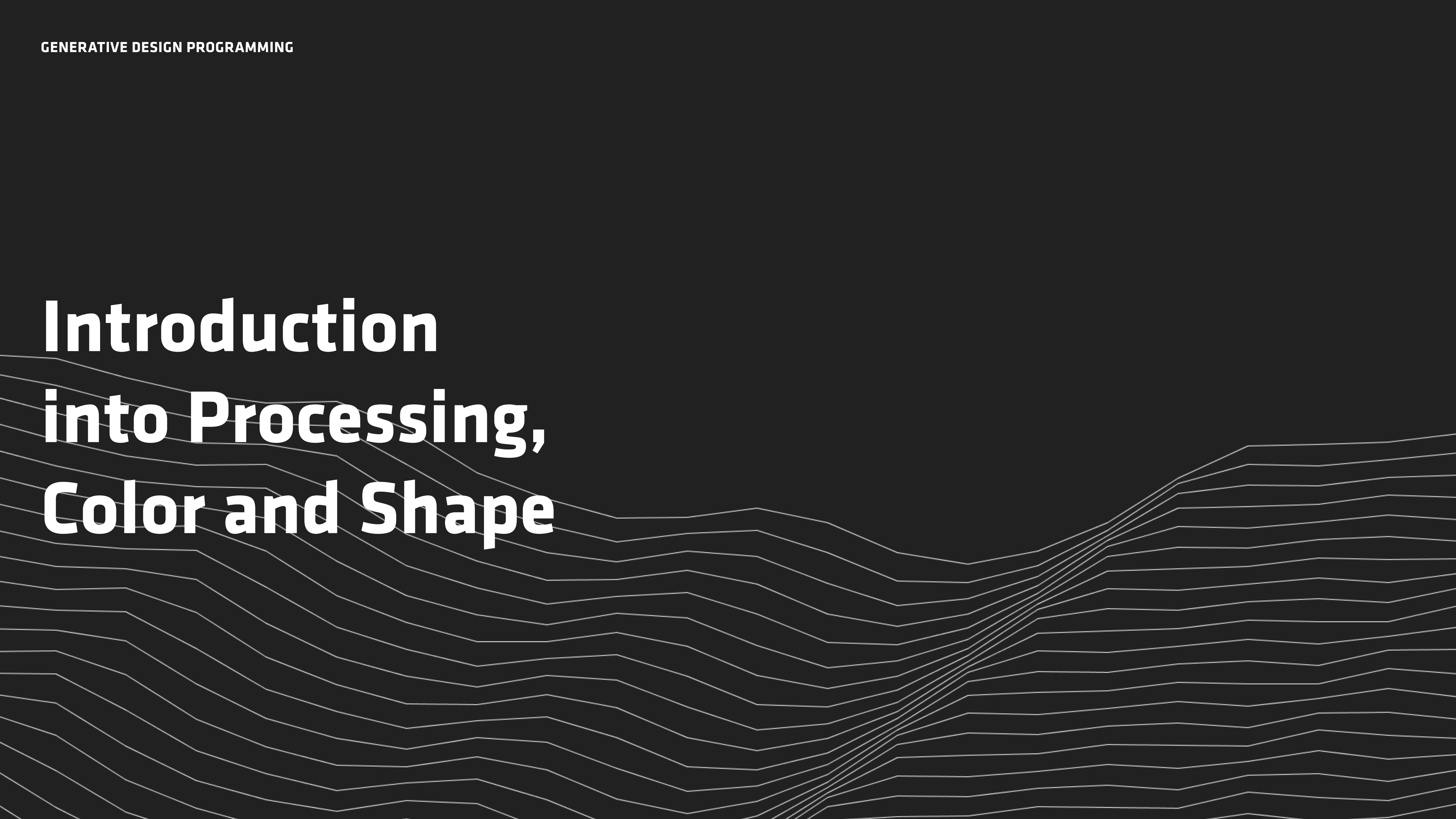


Introduction into Processing, Color and Shape

The background features a series of white, wavy, horizontal lines that create a sense of depth and movement, resembling a stylized landscape or a digital wave pattern. The lines are more densely packed on the right side and become more sparse towards the left.

Introduction into Processing

Where to look for Processing

Processing website, the tool

www.processing.org

Daniel Shiffman, the person

[get a taste of him](#)

Coding train, the channel

[youtube](#)

Reference page, the documentation

processing.org/reference

Generative Gestaltung, the book and the examples

generative-gestaltung.de

Welcome, Processing IDE

Processing IDE

that's what we use during classes



VS code extension for Processing

[github](#), [video tutorial for Win](#), [video tutorial for Mac](#)

p5.js in VS code

[video tutorial](#)

```
sketch_200926a | Processing 3.5.3
File Edit Sketch Debug Tools Help

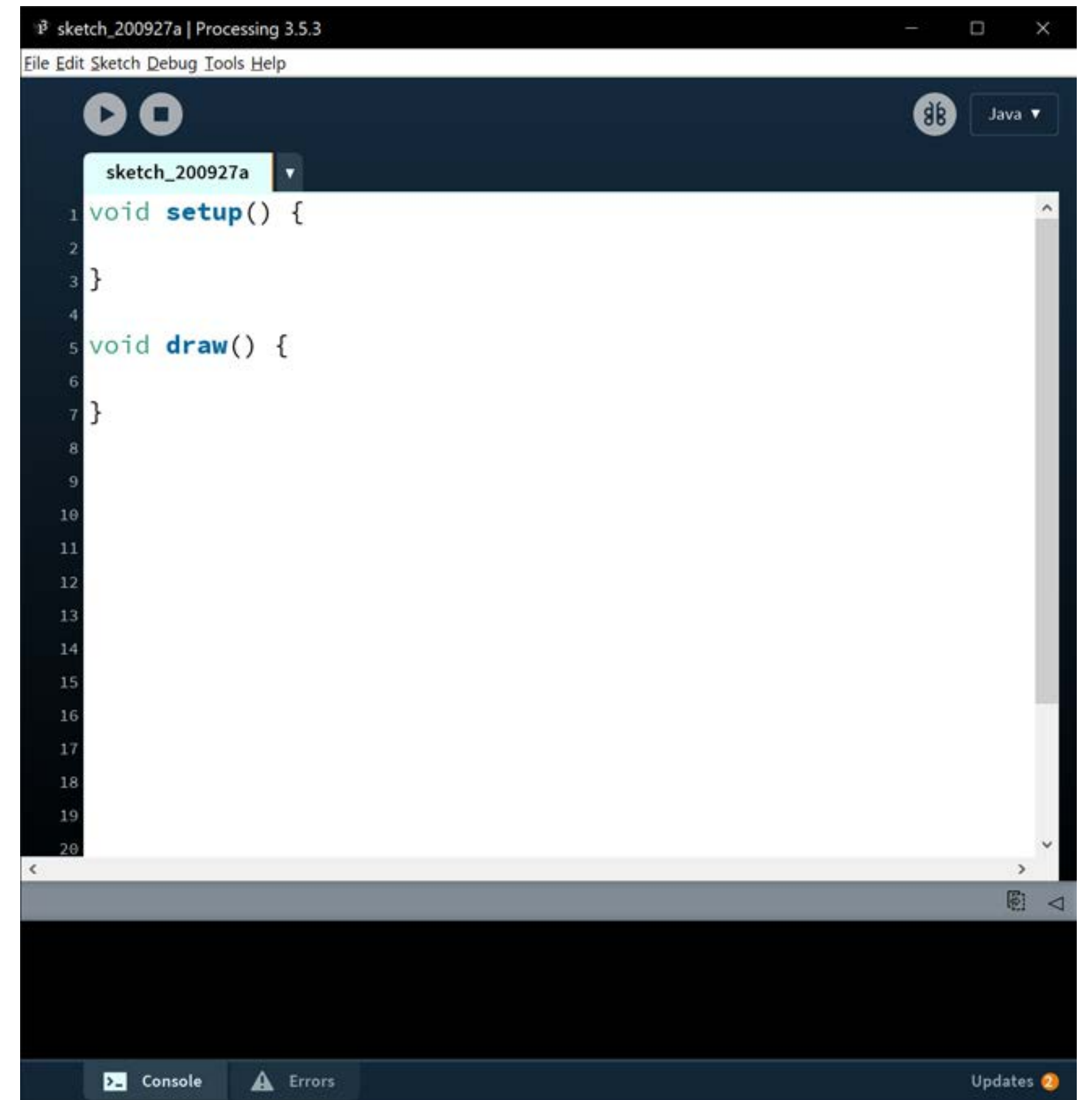
sketch_200926a
1 void setup() {
2   size(1920, 1080);
3   background(255);
4
5   drawDots();
6 }
7
8 void draw() {}
9
10 void mouseClicked() {
11   background(255);
12   drawDots();
13 }
14
15 void keyPressed() {
16   if (key == 's') saveFrame();
17   if (key == 'b') filter(BLUR);
18 }
19
20 void drawDots() {
21   for (int i = 0; i < 200; i++) {
22     fill(random(220, 250));
23     noStroke();
24     ellipse(randomGaussian()*width/2 + width/2, random(height), 5, 5);
25   }
26 }
27
28
29
30
```

The inevitable functions

```
void setup() {  
  // runs only once, at the beginning of the program  
}
```

```
void draw() {  
  // runs in loop  
}
```

You can use `loop()` and `noLoop()` functions



The essential functions

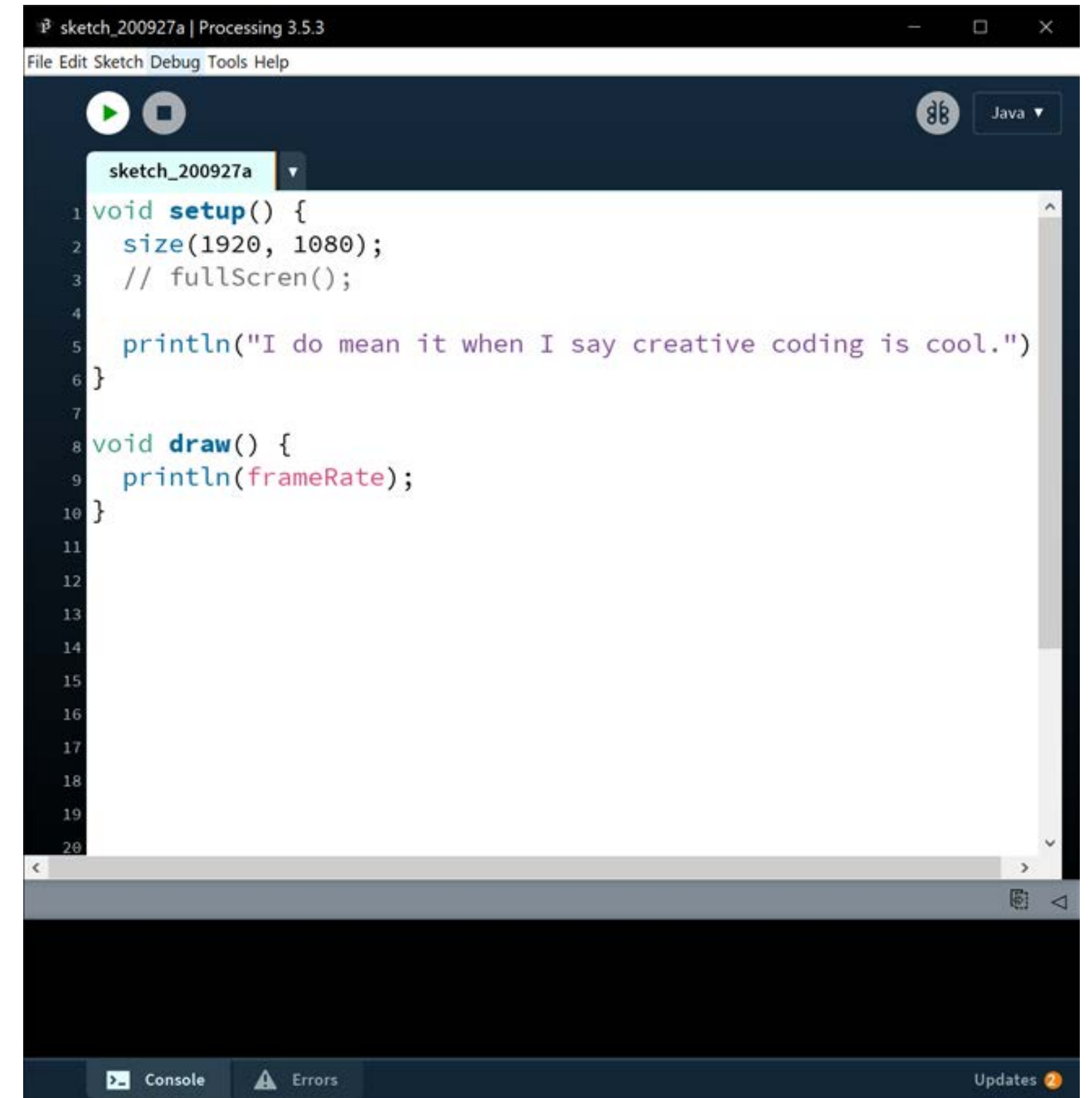
size(float width, float height)

fullScreen()

fullScreen(int screenId)

frameRate(int fps)

println(String s)



The screenshot shows the Processing IDE interface. The title bar reads "sketch_200927a | Processing 3.5.3". The menu bar includes "File Edit Sketch Debug Tools Help". The code editor displays the following code:

```
1 void setup() {  
2   size(1920, 1080);  
3   // fullScreen();  
4  
5   println("I do mean it when I say creative coding is cool.");  
6 }  
7  
8 void draw() {  
9   println(frameRate);  
10 }  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20
```

The IDE interface includes a toolbar with a play button, a stop button, and a Java dropdown menu. At the bottom, there are tabs for "Console", "Errors", and "Updates".

Useful shortcuts for Processing IDE

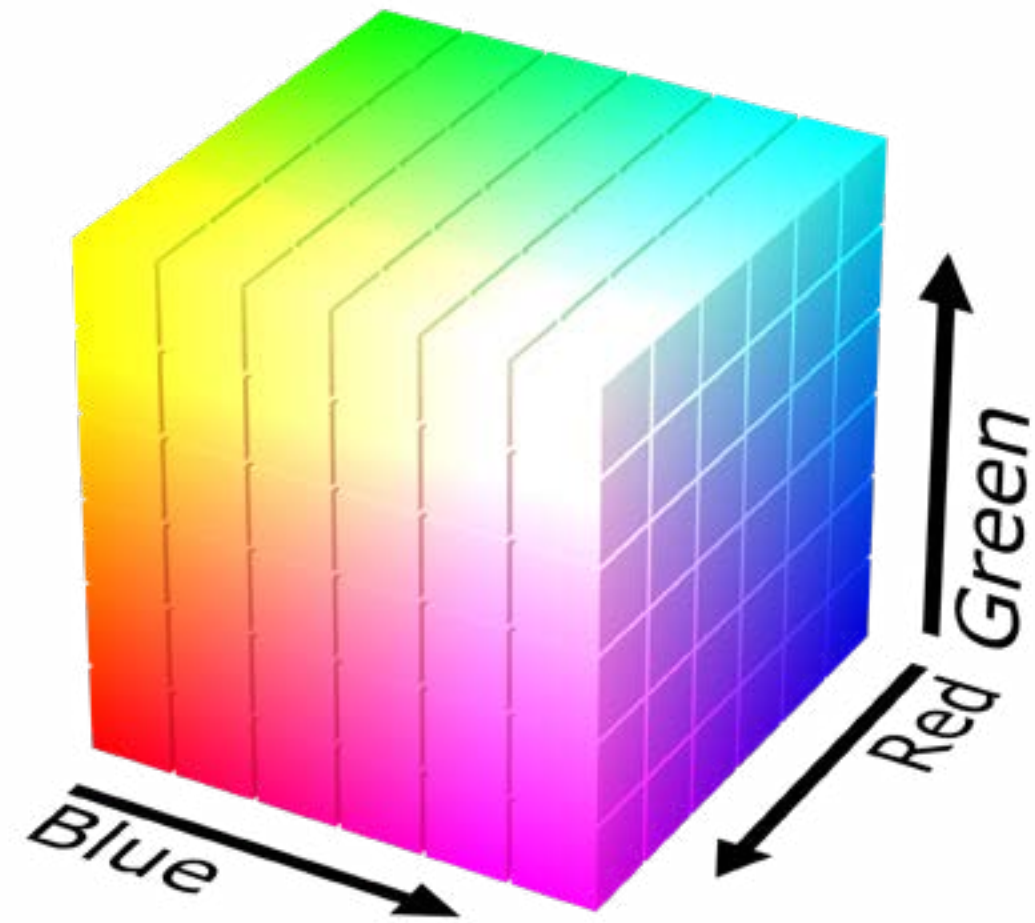
Ctrl+R	Run the sketch
Ctrl+K	Open the sketchfolder
Ctrl+Shift+T	Run in Tweak mode

Yeah, I thought there would be more of these.
But these are good to know, hh.

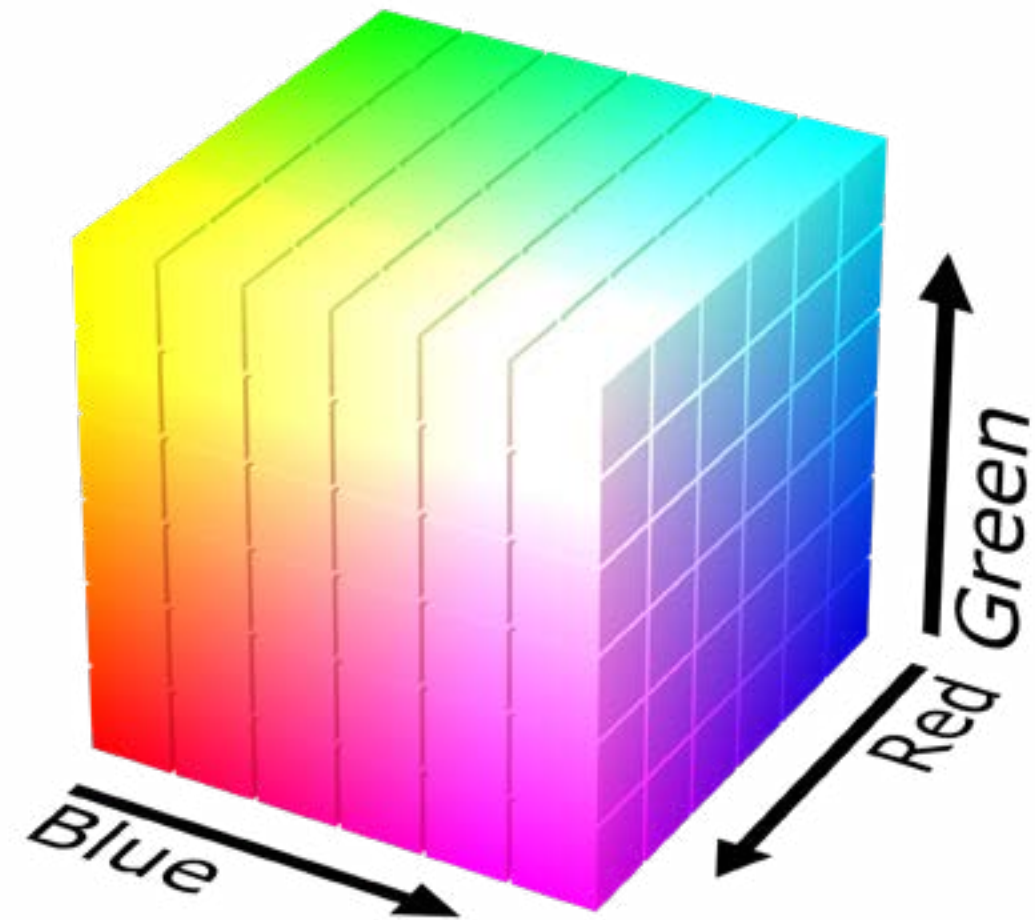
Also, the main .pde file needs to
have the same name as the folder
it appears in.

Color

RGB (red-green-blue)

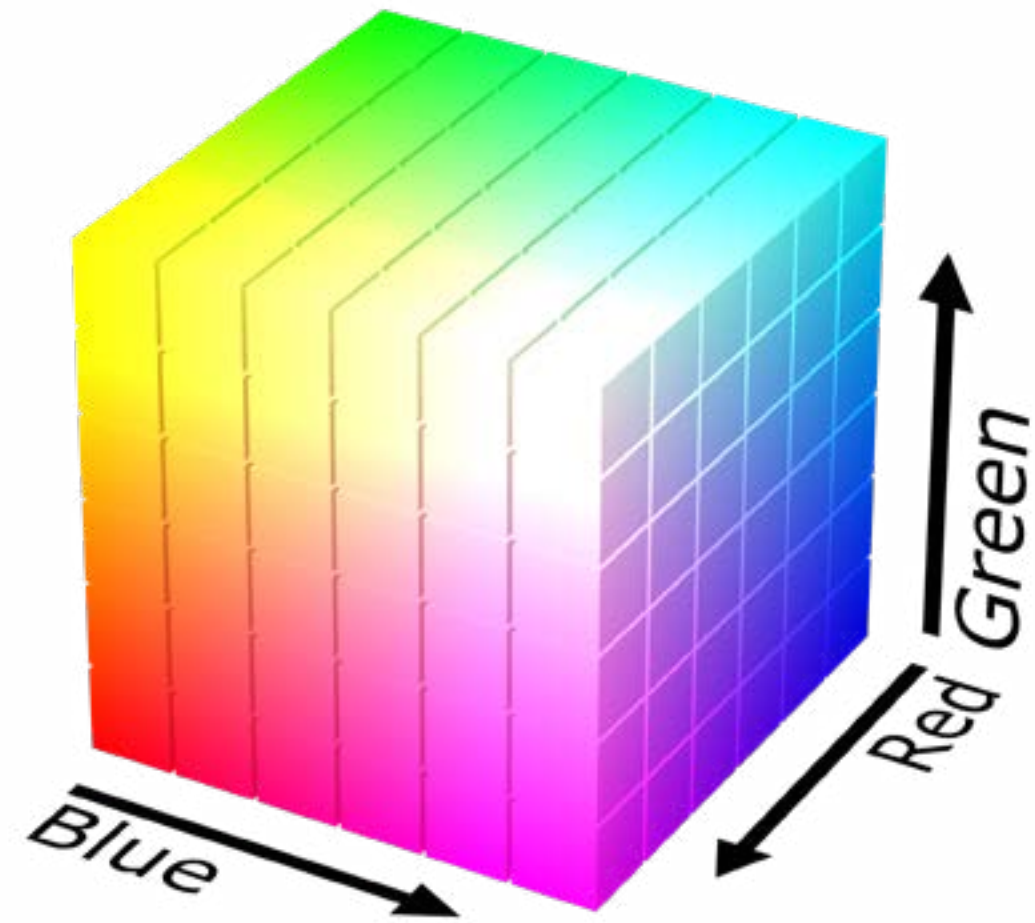





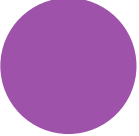


RGB (red-green-blue)



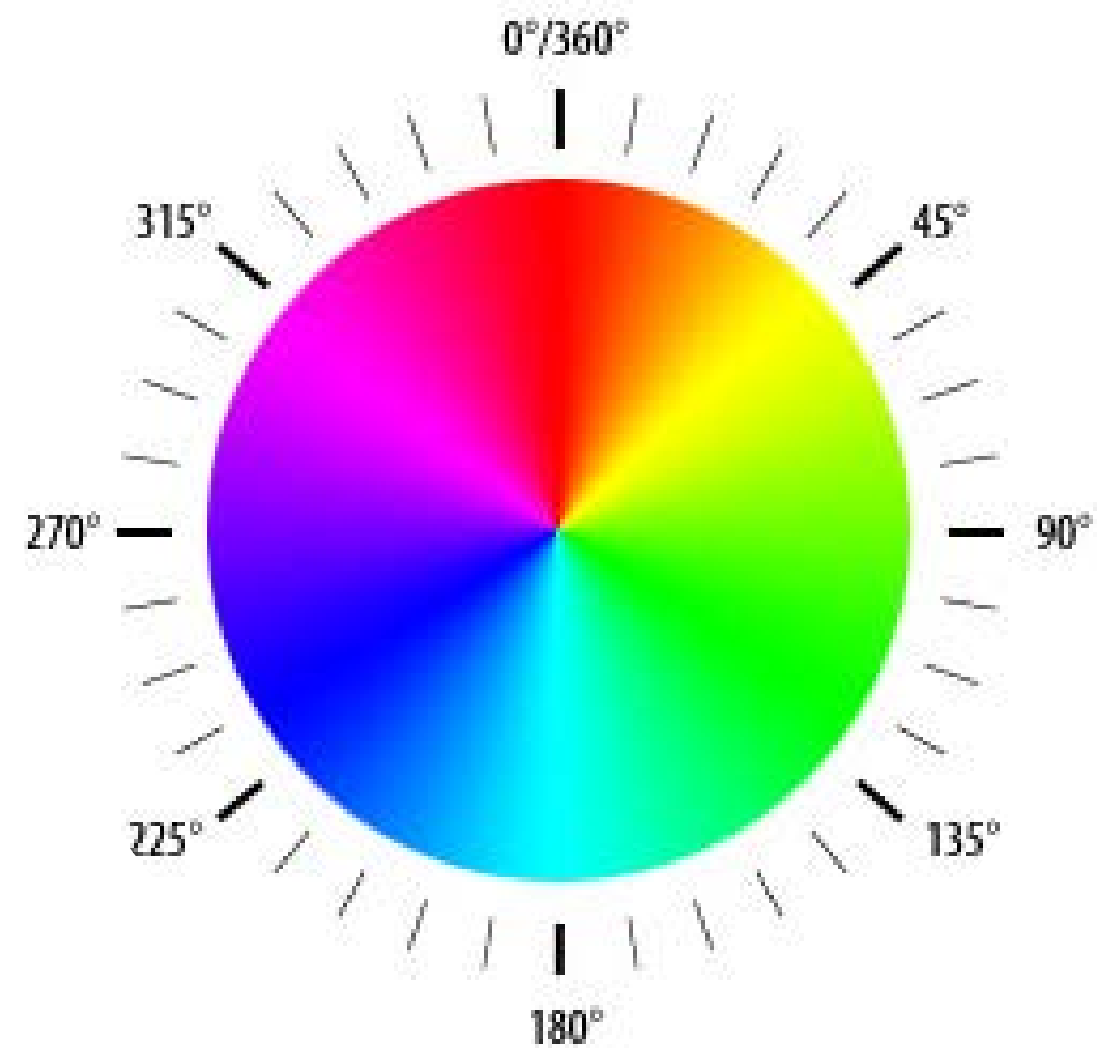
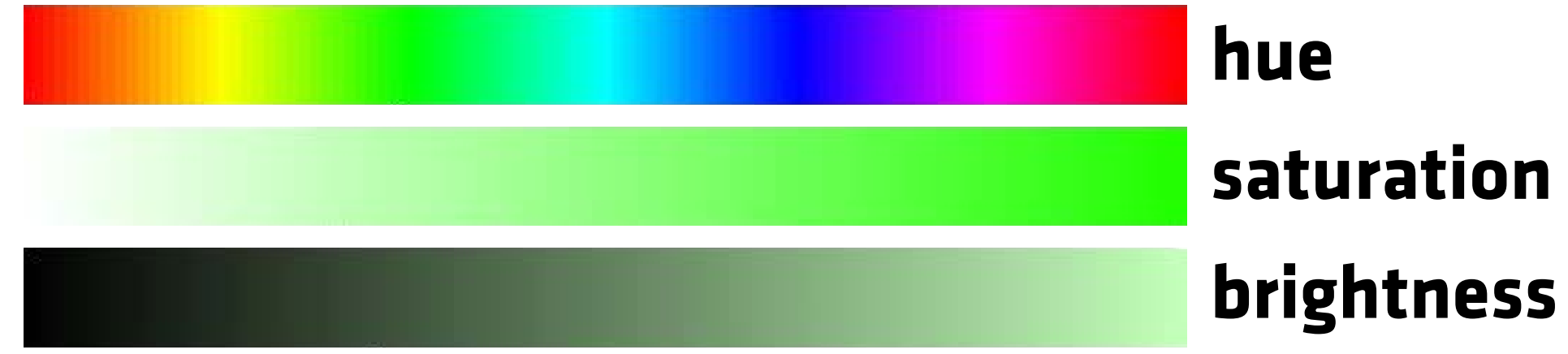
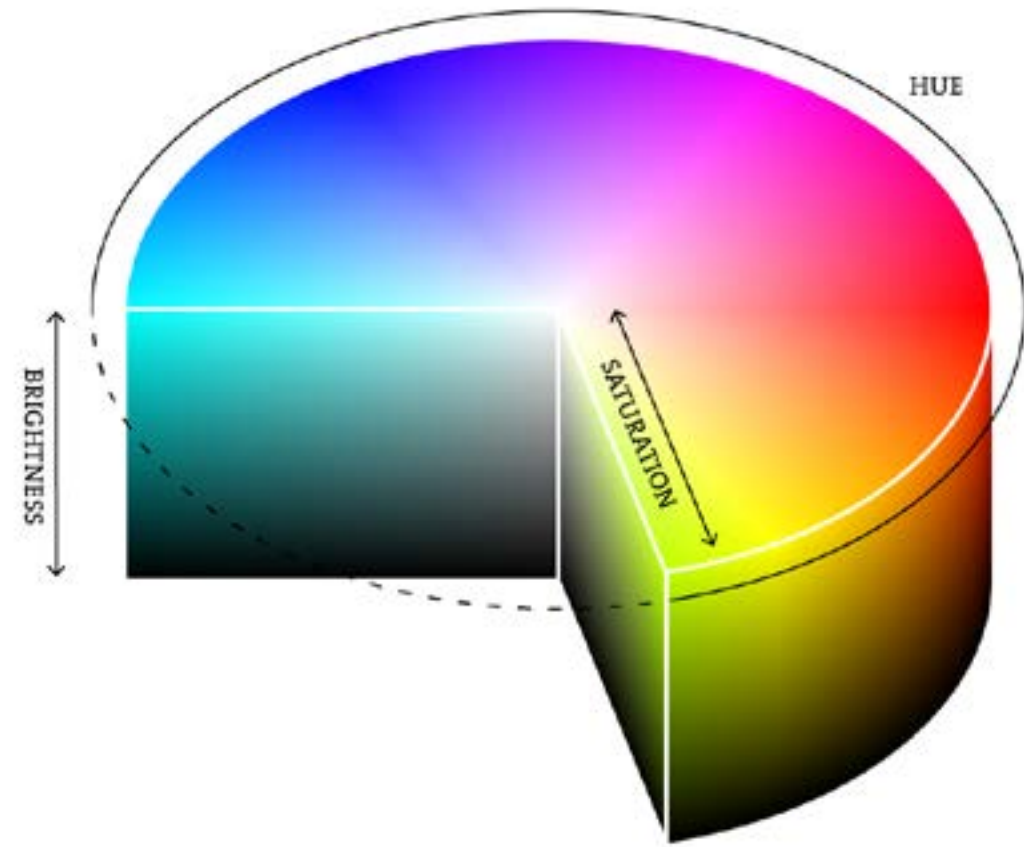
●	255r	200g	87b
●	245r	138g	90b
●	235r	71g	118b
●	158r	82g	170b
●	81r	88g	187b
●	?r	?g	?b

RGB (red-green-blue)

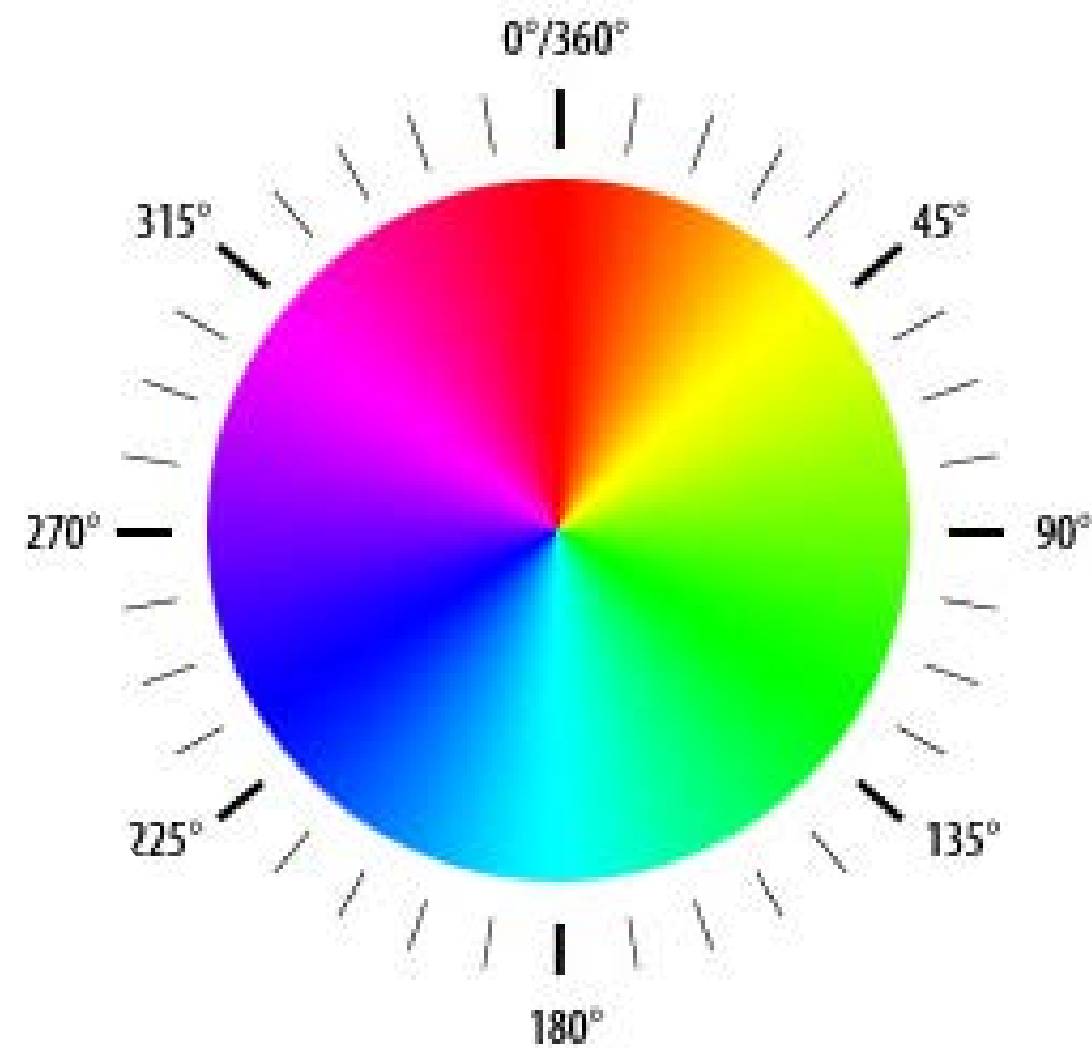
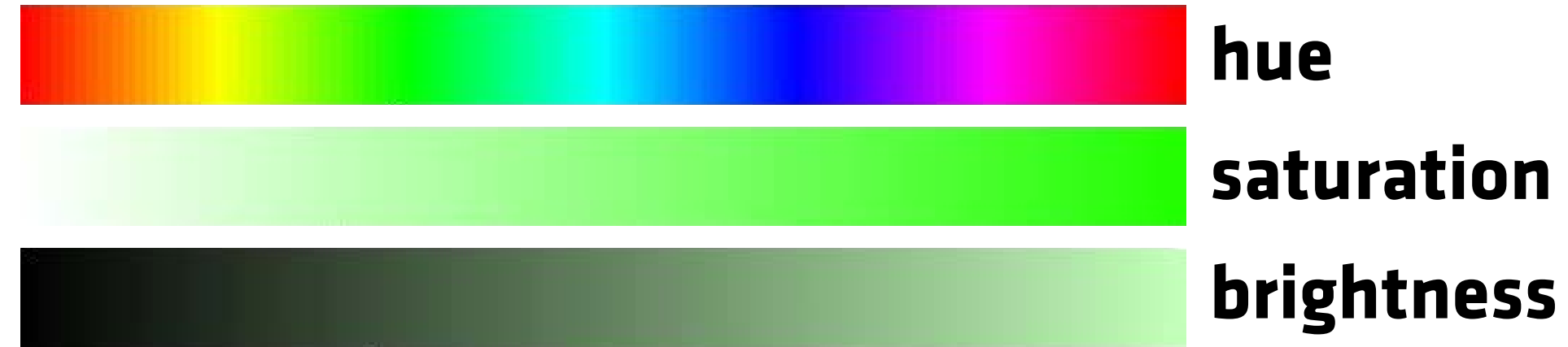
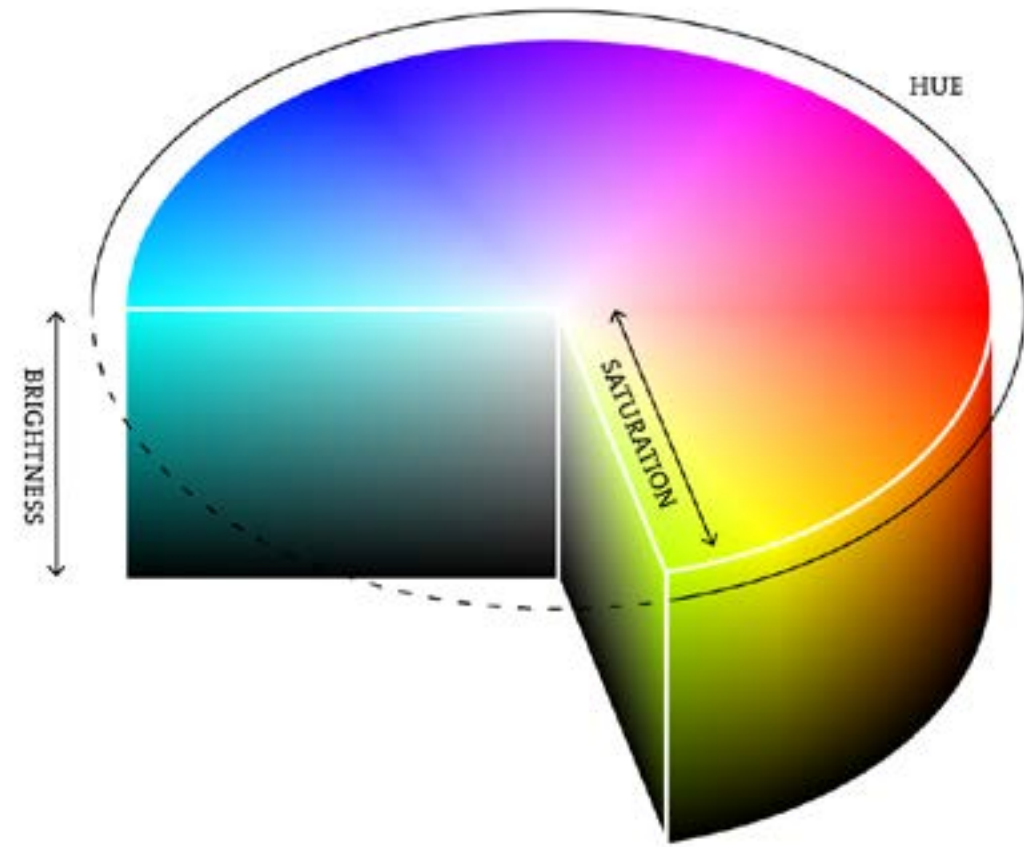


	255r	200g	87b
	245r	138g	90b
	235r	71g	118b
	158r	82g	170b
	81r	88g	187b
	14r	188g	191b

HSB (hue-saturation-brightness)

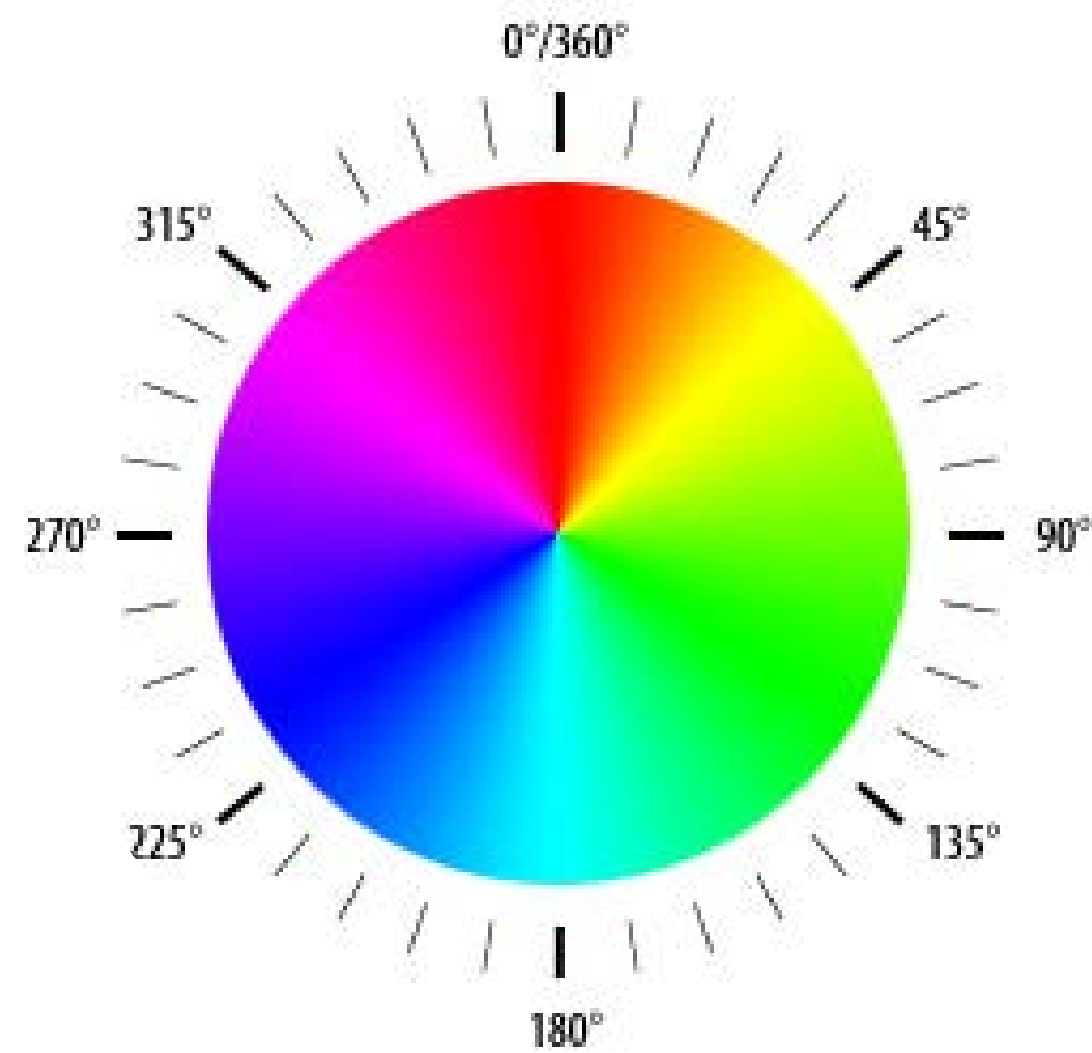
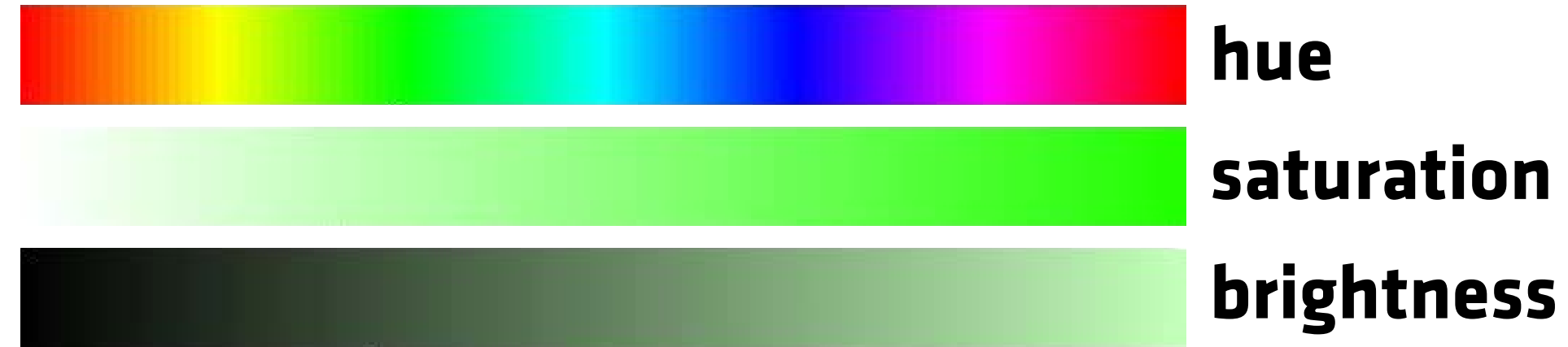
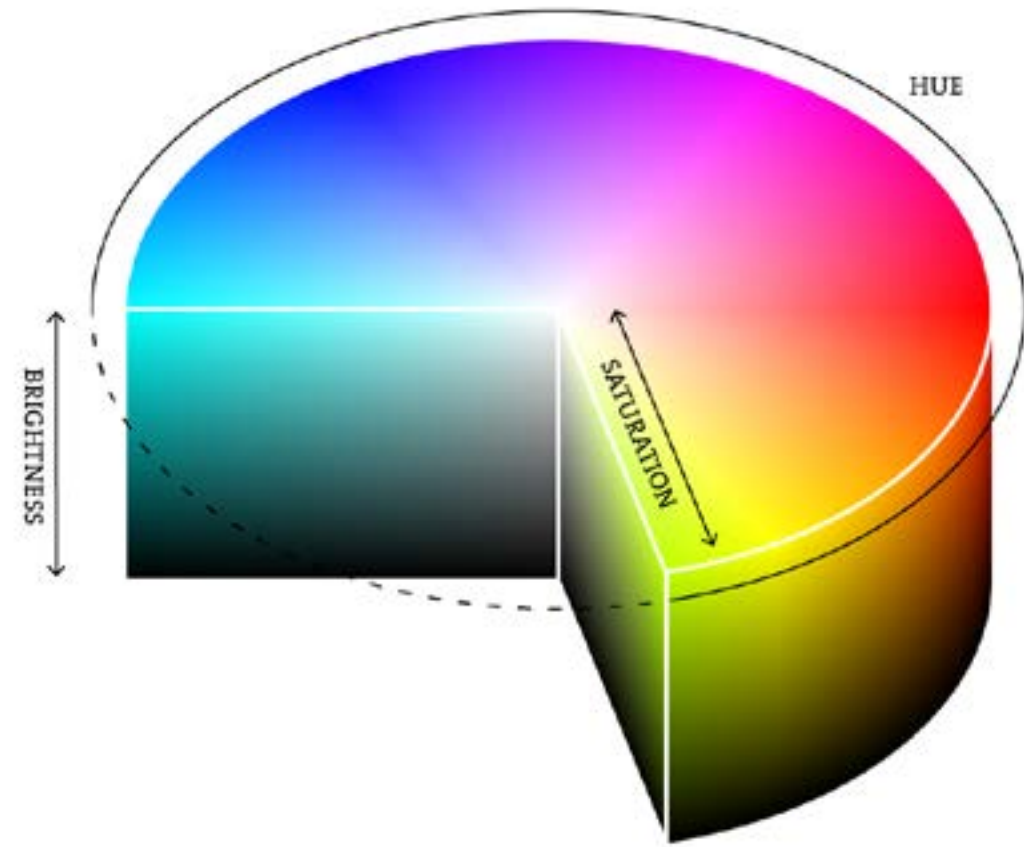


HSB (hue-saturation-brightness)



●	40h	66s	100b
●	19h	63s	96b
●	343h	70s	92b
●	292h	52s	67b
●	236h	57s	73b
●	?h	?s	?b

HSB (hue-saturation-brightness)



●	40h	66s	100b
●	19h	63s	96b
●	343h	70s	92b
●	292h	52s	67b
●	236h	57s	73b
●	181h	93s	75b

Explore it
yourselves
at [colors](#)

Color modes

```
colorMode(mode, int max, ...)
```

```
// colorMode(RGB, r, g, b, alpha)  
colorMode(RGB, 256, 256, 256, 256) // rgba -- default  
colorMode(RGB, 100, 100, 100, 100)
```

```
// colorMode(HSB, h, s, b, alpha)  
colorMode(HSB, 360, 100, 100, 100) // hsba
```

Color definition

Processing offers a **color datatype**.

```
color magenta;  
magenta = color(float r, float g, float b);  
magenta = color(255, 0, 255);
```

```
color copper = #e26d5c;
```

```
fill(color anycolor)
```

```
fill(int grayscale)
```

```
fill(int grayscale, float alpha)
```

```
fill(float r, float g, float b)
```

```
fill(float r, float g, float b, float alpha)
```

Don't forget that color is technically just a number. Check-out [details](#) how you can work with it.

Color functions

Fill fill(color)
výplň noFill()

Stroke stroke(color)
obrys noStroke()

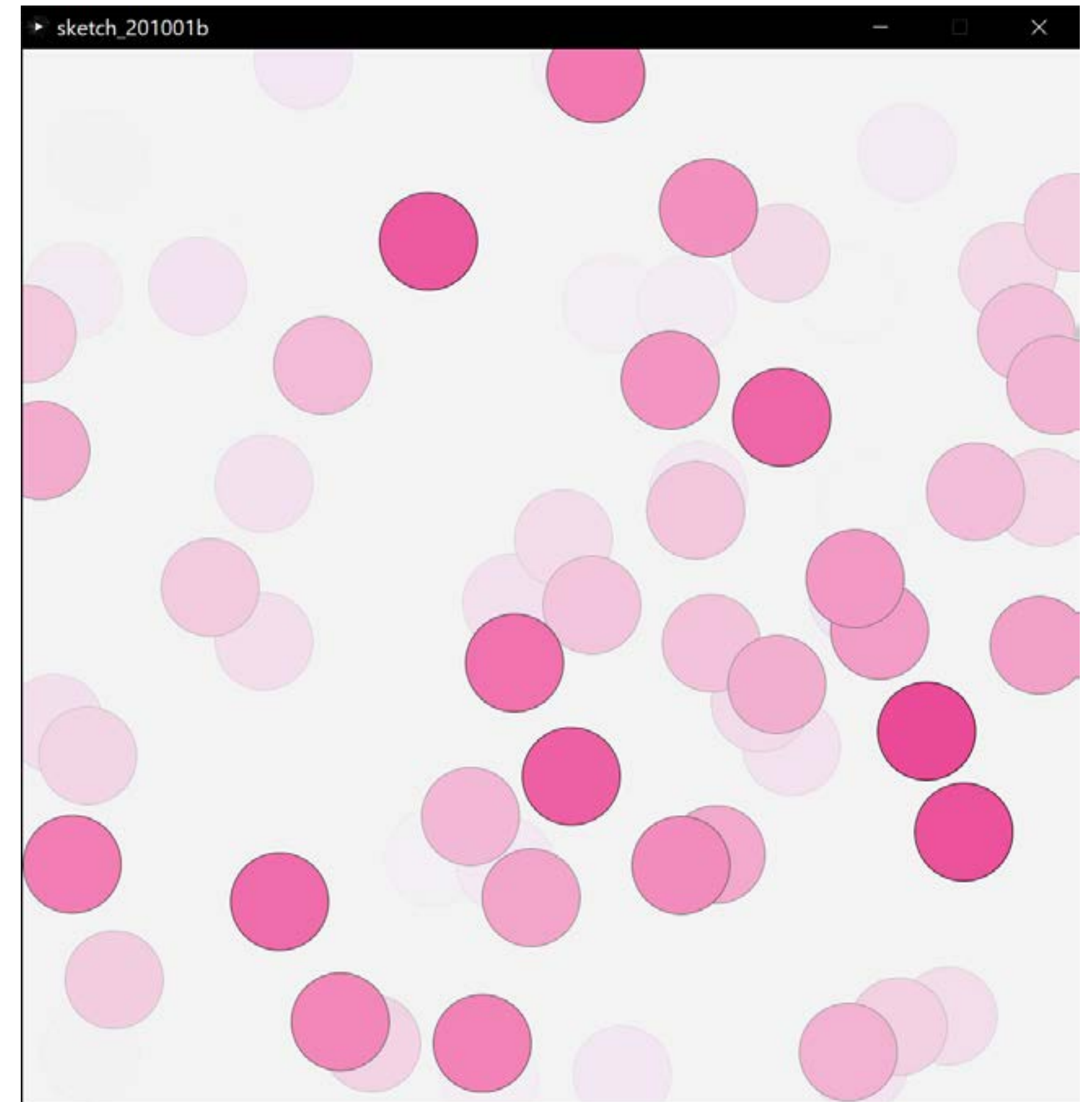


Background background(color)
pozadí clear()

```
sketch_201001a
1 void setup() {
2   size(400, 400);
3   background(70, 80, 185);
4 }
5
6
7
8
9
10
11
12
13
```

#dummy fading effect

```
void setup() {  
  size(1080, 1080);  
  background(240);  
}  
  
void draw() {  
  fill(250, 10); // semi-transparent white  
  rect(0, 0, width, height);  
  
  fill(#EB4B98);  
  ellipse(random(width), random(height), 100, 100);  
}
```



You'll notice it is a bit tough to get the color numbers right so you avoid ghost objects in canvas.

Shape

2D shapes

`point(x, y)`

`line(x1, y1, x2, y2)`

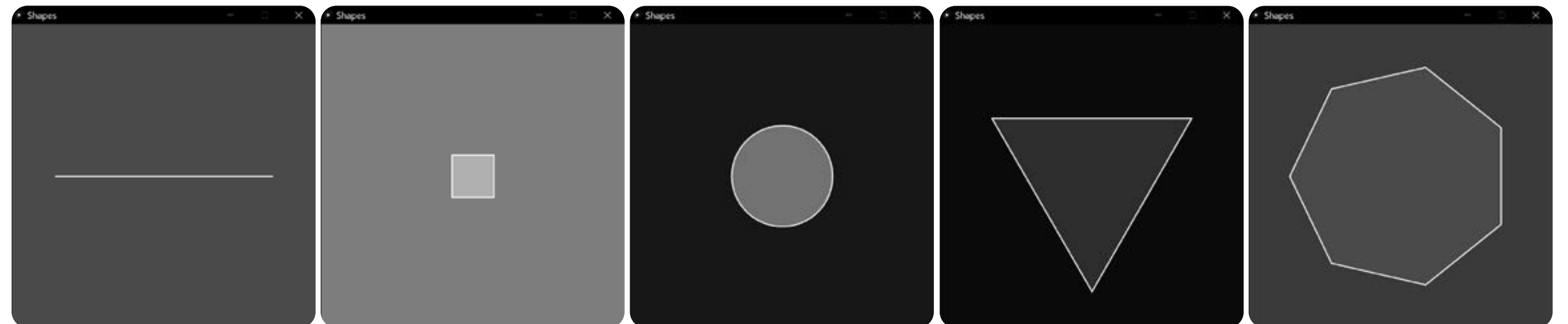
`rect(x, y, width, height)`

`ellipse(x, y, width, height)`

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

`quad(...)`

`arc(...)`



Let's have fun

Built-in variables

```
mouseX, mouseY // x and y mouse position  
pmouseX, pmouseY // previous x and y mouse position  
  
width, height // width and height of the artboard in pixels  
  
frameCount // number of the current frame
```

Taste of randomness

```
random(max)  
random(min, max)
```

Processing interaction functions

```
void keyPressed() { // here goes your art }
```

```
void mouseClicked() { // here goes your art }
```

```
void mouseDragged() { // here goes your art }
```

```
void mousePressed() { // here goes your art }
```

```
void mouseReleased() { // here goes your art }
```

And there are others, check out the reference

Export raster image

Saving frames into the sktechbook folder

```
saveFrame(); // saves frames: 'screen-0001.png', 'screen-0045.png', etc.
```

```
saveFrame("img_name-####.png")
```

are replaced by the current **frameCount** value

Supported raster formats: .png .jpg .tif .tga

Export vector image

Processing offers a PDF library that allows to **draw directly into PDF or SVG files**.

This can be useful to store graphics in high quality (ehm, vector).

Morover, this allows you to finish or post-process works created in Processing using vector editors such as Adobe Illustrator or Sketch.

You can store either **one frame**, or a **sequence of frames** as an animated PDF or as multiple page file.

```
import processing.pdf.*;
import processing.svg.*;
```

```
beginRecord(PDF, "my_vector_art.pdf")
beginRecord(SVG, "my_vector_art.svg")
endRecord()
```

For details, check out the Processing's [PDF Export](#) and [SVG Export](#) pages, or our template codes.

#timestamp

```
String timestamp;
```

```
void setup() {
```

```
    timestamp = year() + nf(month(),2) + nf(day(),2) + “-” +  
               nf(hour(),2) + nf(minute(),2) + nf(second(),2);
```

```
    println(timestamp);
```

```
    ...
```

```
}
```

```
void keyPressed() {
```

```
    if (key == ‘s’ || key == ‘S’) saveFrame(“img_name-“ + timestamp + “-####.png”)
```

```
}
```

