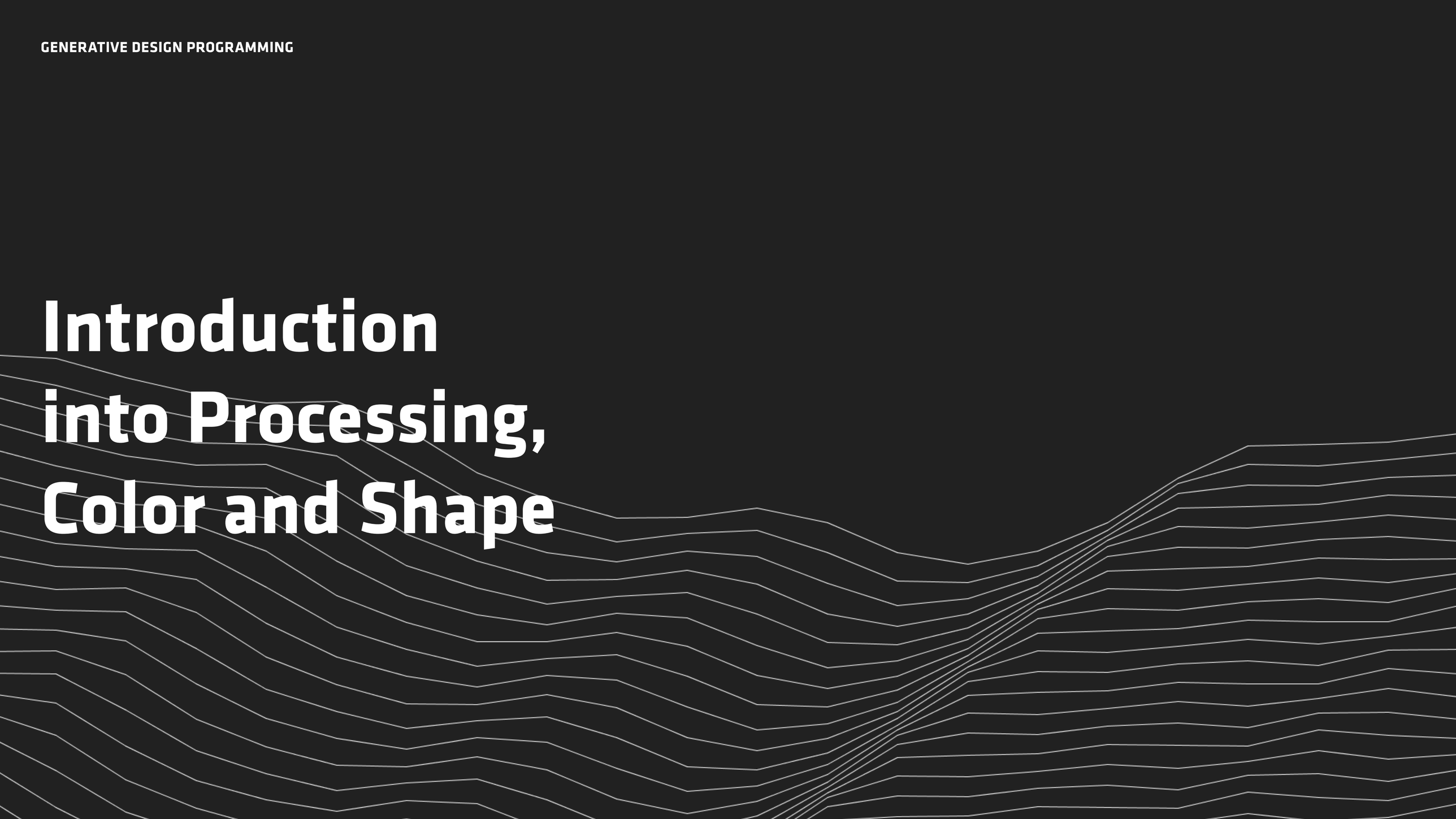
Introduction into Processing, Color and Shape



KRISTÍNA ZÁKOPČANOVÁ

Introduction into Processing

Where to look for Processing

Processing website, the tool
www.processing.org

Daniel Shiffman, the person
get a taste of him

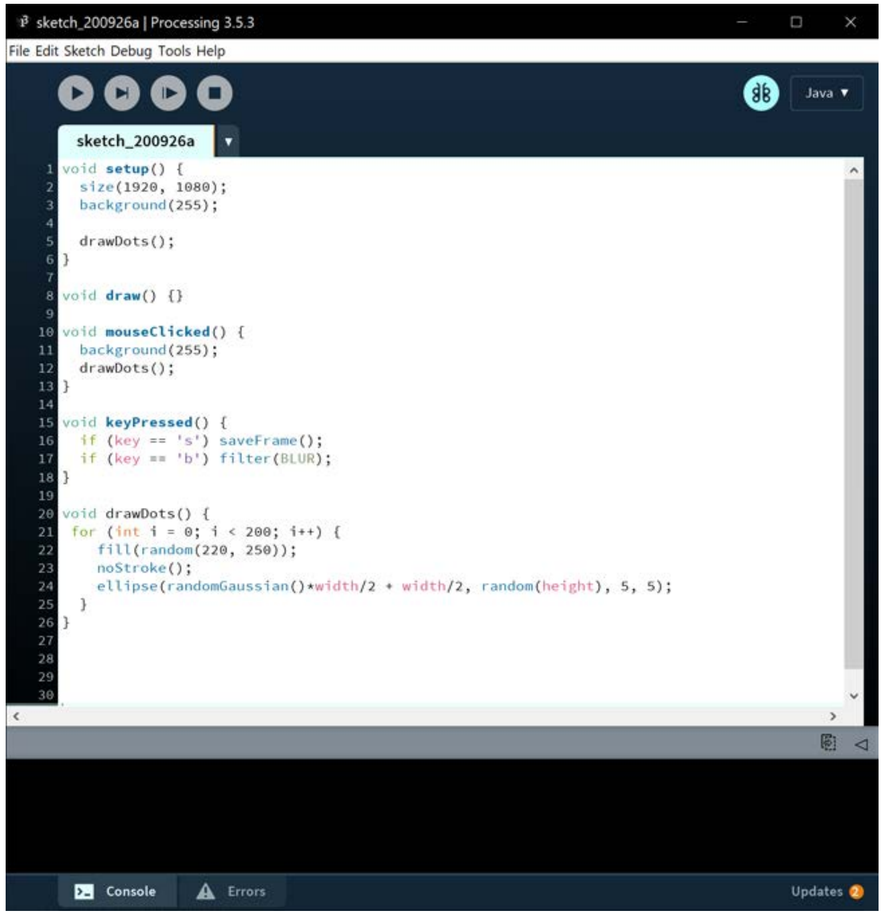
Coding train, the channel
youtube

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

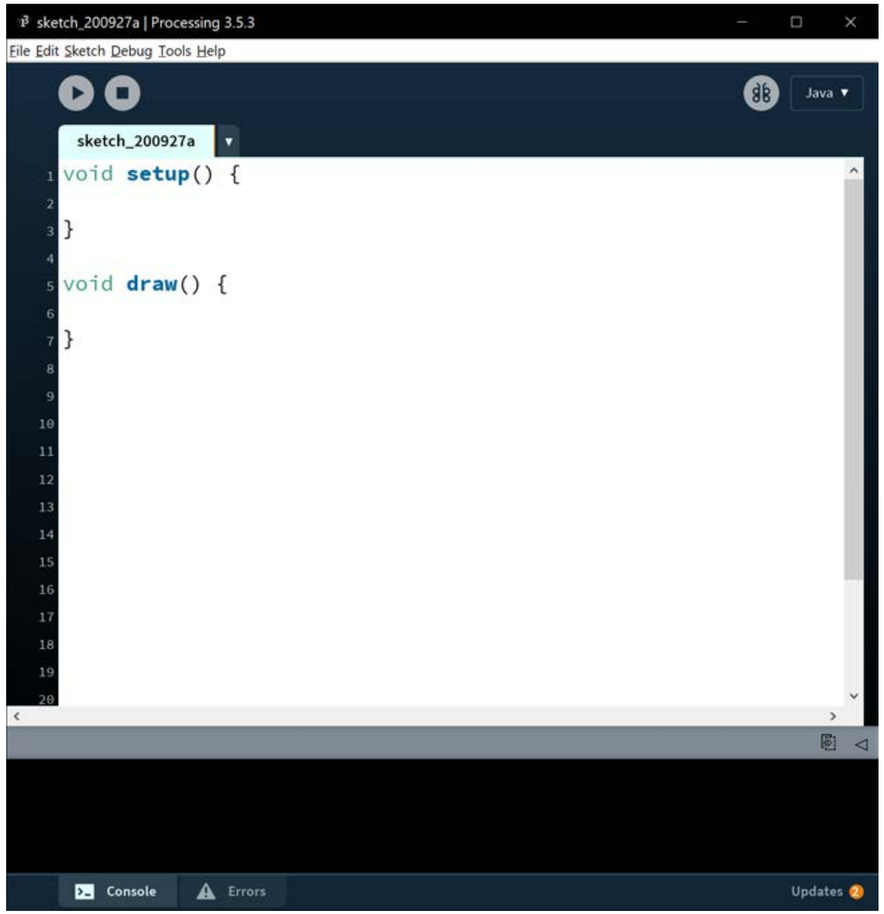


VS code extension for Processing
github, video tutorial for Win, video tutorial for Mac

p5.js in VS code
video tutorial

The inevitable functions

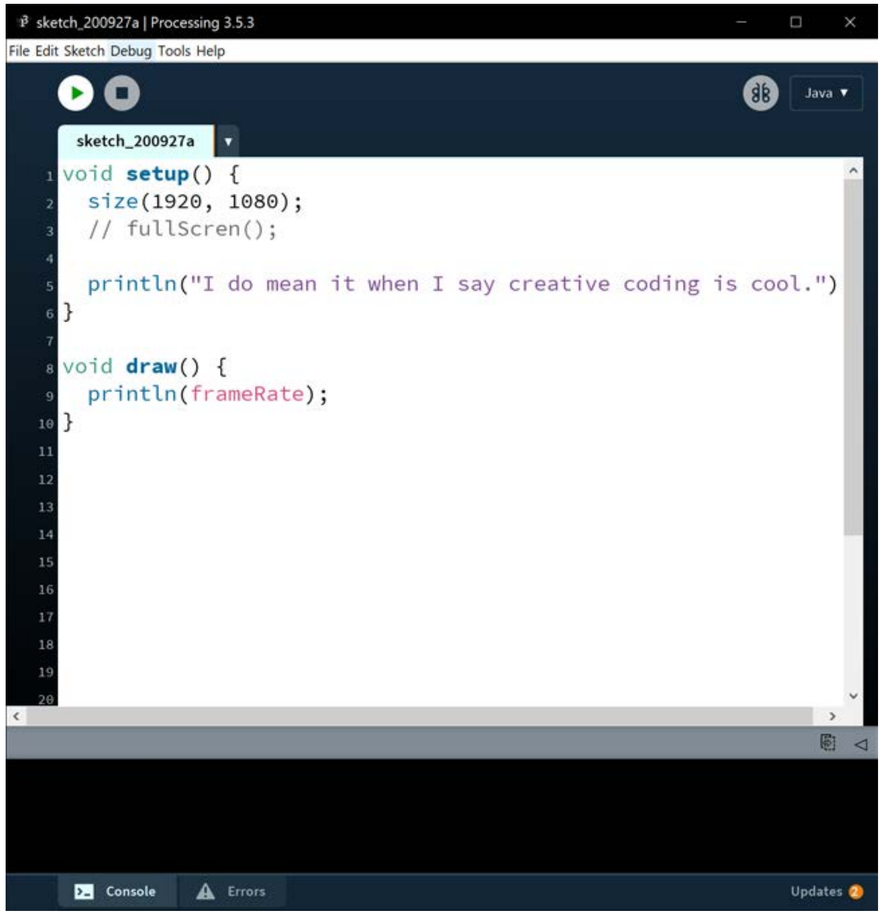
```
void setup() {
  // runs only once, at the beginning of the program
}

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)
```



Useful shortcuts for Processing IDE

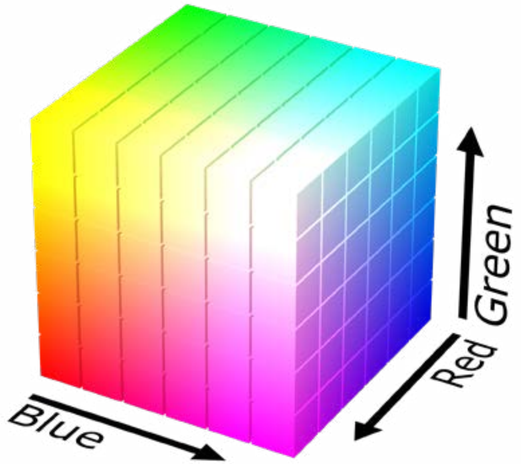
Ctrl+R	Run the sketch
Ctrl+K	Open the sketchfolder
Ctrl+Shift+T	Run in Tweak mode

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

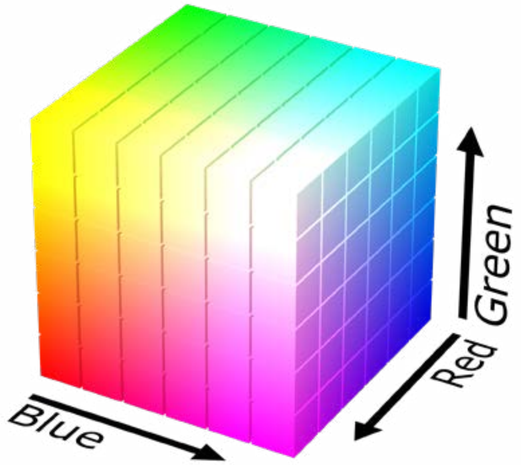
Also, the main .pde file needs to have the same name as the folder it appears in.

Color

RGB (red-green-blue)



RGB (red-green-blue)



255r 200g 87b

245r 138g 90b

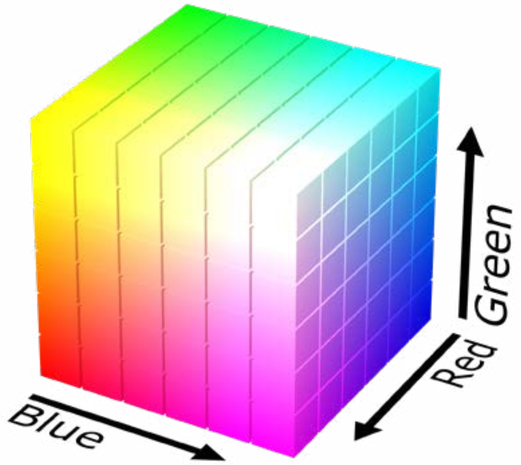
235r 71g 118b

158r 82g 170b

81r 88g 187b

?r ?g ?b

RGB (red-green-blue)



255r 200g 87b

245r 138g 90b

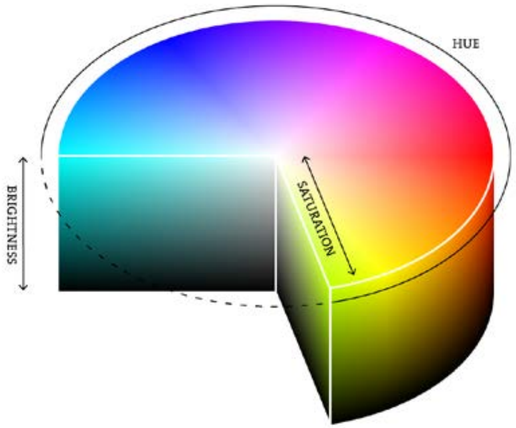
235r 71g 118b

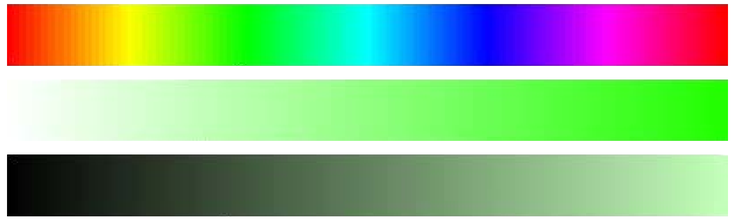
158r 82g 170b

81r 88g 187b

14r 188g 191b

HSB (hue-saturation-brightness)

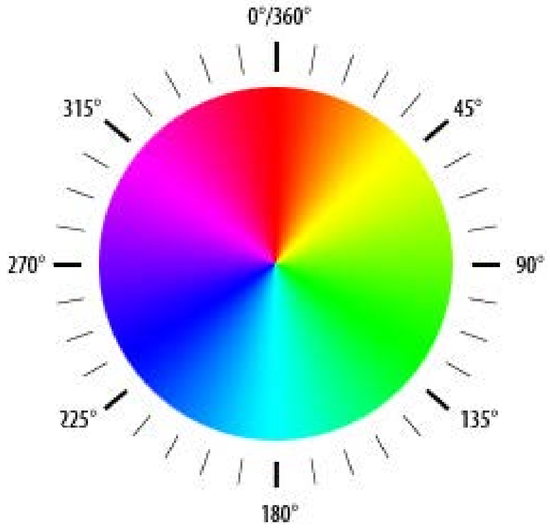




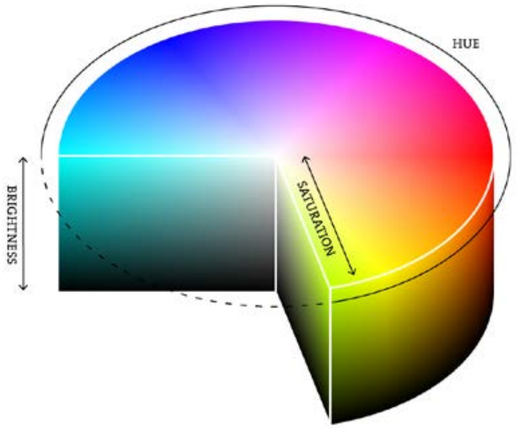
hue

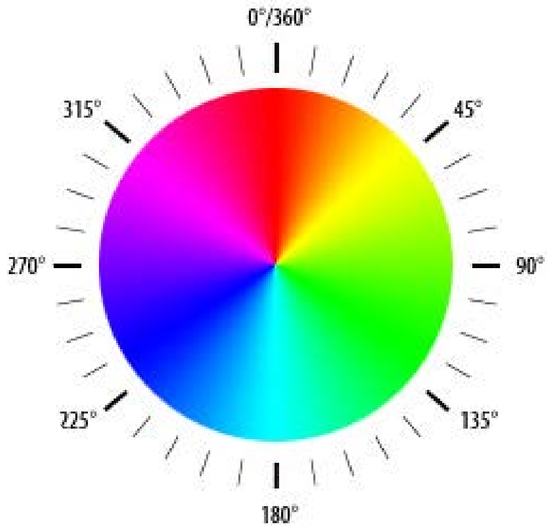
saturation

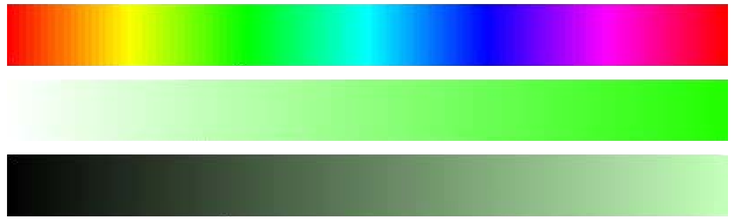
brightness



HSB (hue-saturation-brightness)







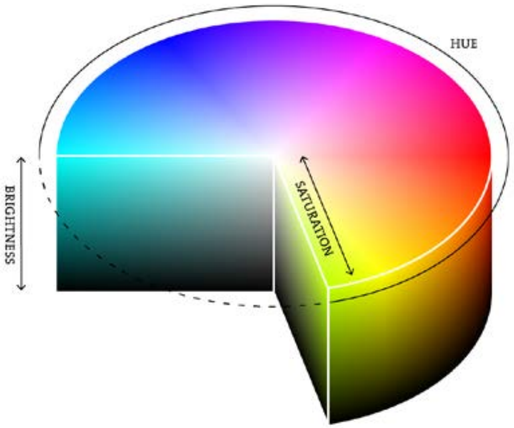
hue

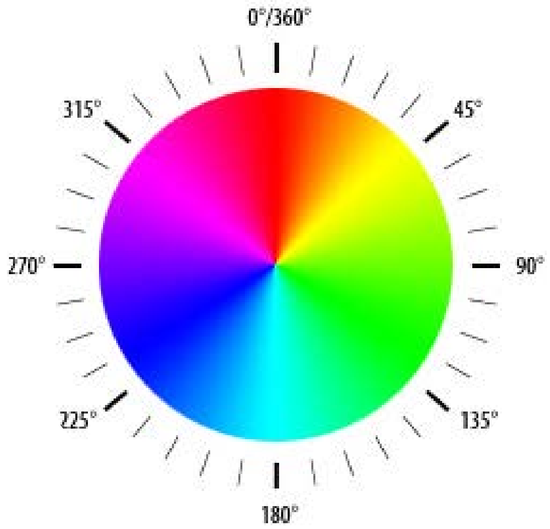
saturation

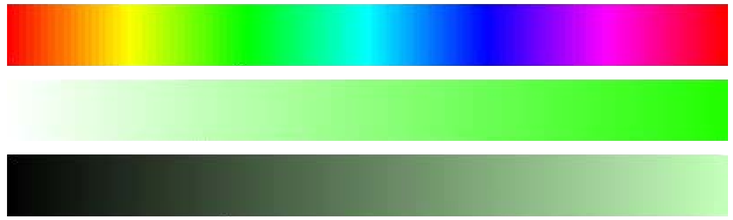
brightness

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

HSB (hue-saturation-brightness)







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 coolors

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**.

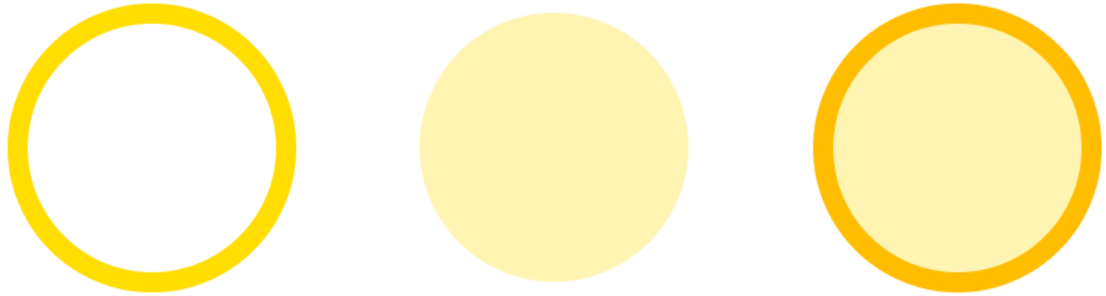
```
color magenta;
magenta = color(float r, float g, float b);
magenta = color(255, 0, 255);

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.

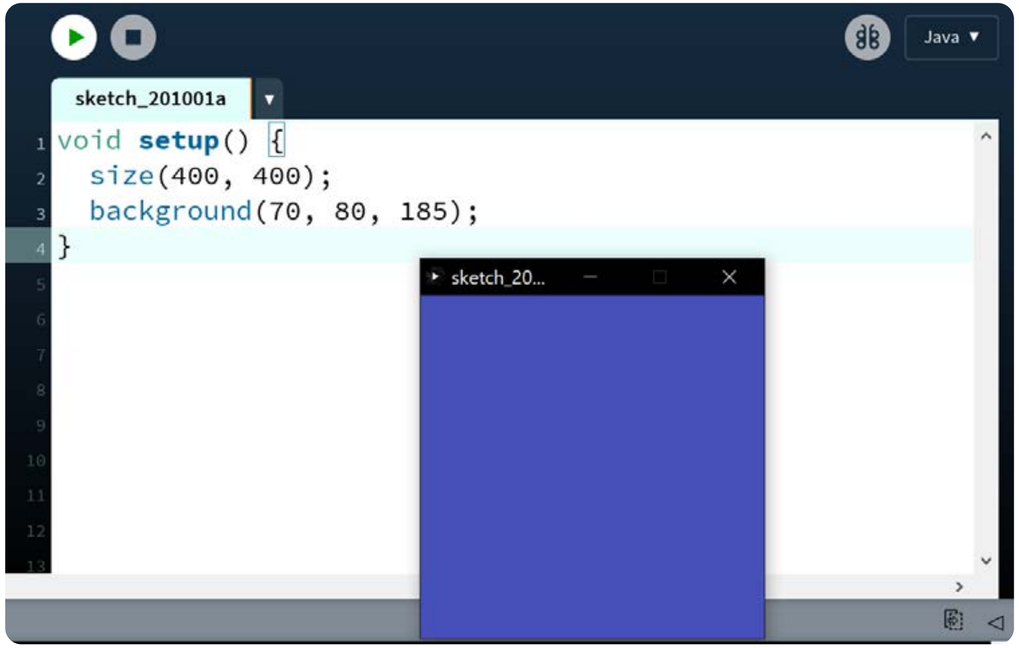
Color functions

Fill `fill(color)`
výplň `noFill()`

Stroke `stroke(color)`
obrys `noStroke()`

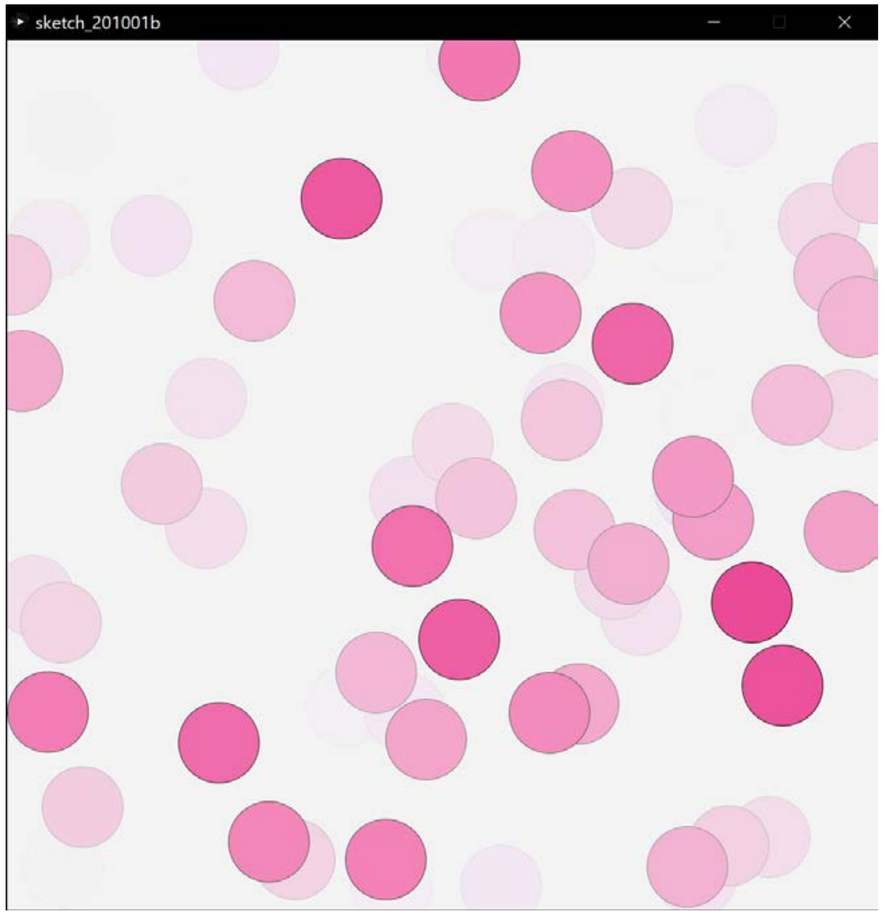


Background `background(color)`
pozadí `clear()`



#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);
}
```

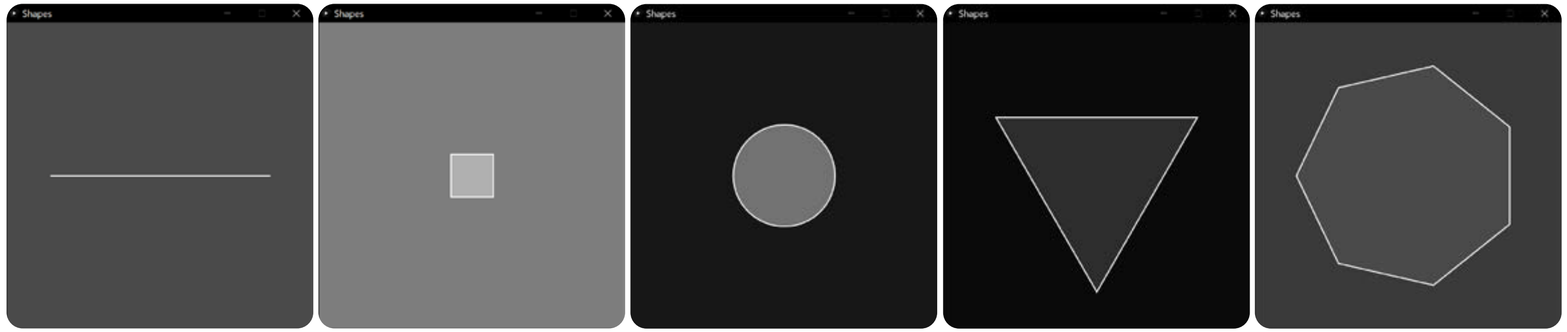


You'll notice it is a bit tough to get the color numbers right so you avoid ghost objects in canvas.

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 a 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”)
}
```