PV259

Generative Design Programing

Week 3

Introduction to p5.js, continued
Geometric patterns

Marko Řeháček & Megi Kejstová
rehacek@mail.muni.cz

Feedback from last class

- use Typescript
- have walk breaks
- what to do next instructions
- include more code in slides

Make sure you have joined

https://discord.gg/CcxnBVPtcR

Recap

program structure

```
// runs once when program starts
function setup(){
  createCanvas(800, 600); // width, height in pixels
  // can use windowWidth, windowHeight

  // switch to full screen anytime (also in draw())
  fullscreen(boolean)
}

// run continuously after setup
function draw(){
  // rendering loop
}
```

globals

```
windowWidth / windowHeight // of browser window
width / height // of canvas

mouseX / mouseY
// current horizontal / vertical mouse position
```

non-visual feedback

```
print();
console.log(msg, …);
```

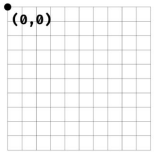
JS

```
let empty_arr = [];
const empty_object = {};
```

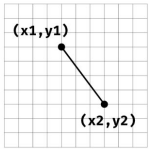
geometry

```
point(x, y)
line(x1, y1, x2, y2)
circle(x, y, radius)
ellipse(x, y, width, height)
square(x, y, side_length)
rect(x, y, width, height)
rectMode(MODE) // CENTER, CORNERS
arc(x, y, width, height, start, stop)
triangle(x1, y1, x2, y2, x3, y3)
```

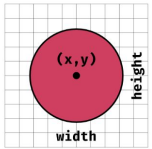
grid system



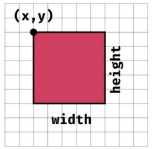
line()



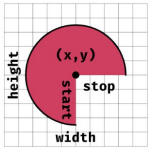
ellipse()



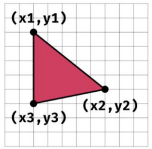
rect()



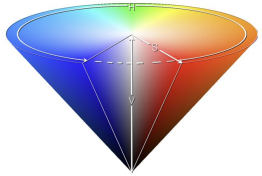
arc()



vertex()



colors



```
background(color)
clear()

fill(color)
noFill()

fill(120); //gray: 0-255
fill(100,125,255); //r, g, b: 0-255
fill(255, 0, 0, 50); //r, g, b, alpha
fill('red'); //color string
fill('#ccc'); //3-digit hex
fill('#222222'); //6-digit hex fill

const mycolor = color(0, 0, 255);
//p5.Color object

stroke(color)
strokeWeight(weight: number)
noStroke()

colorMode(MODE) // HSB, RGB
colorMode(MODE, maxValue: number)
// default for HSB
colorMode(HSB, 360, 100, 100, 1.0)
// change default for RGB
colorMode(RGB, 100)
// HSB is preferred
// RGB used for specific calculations
```

p5.js docs

Variable initialization

Cannot define variables with p5 functions outside of setup() or draw(). The correct way is:

```
let myColor; // declare without initialization

function setup() {
  createCanvas(800, 800);
  myColor = color(255, 0, 0); // initialize here
}

function draw() {
  background(220);
  fill(myColor);
  ellipse(400, 400, 100, 100);
}
```

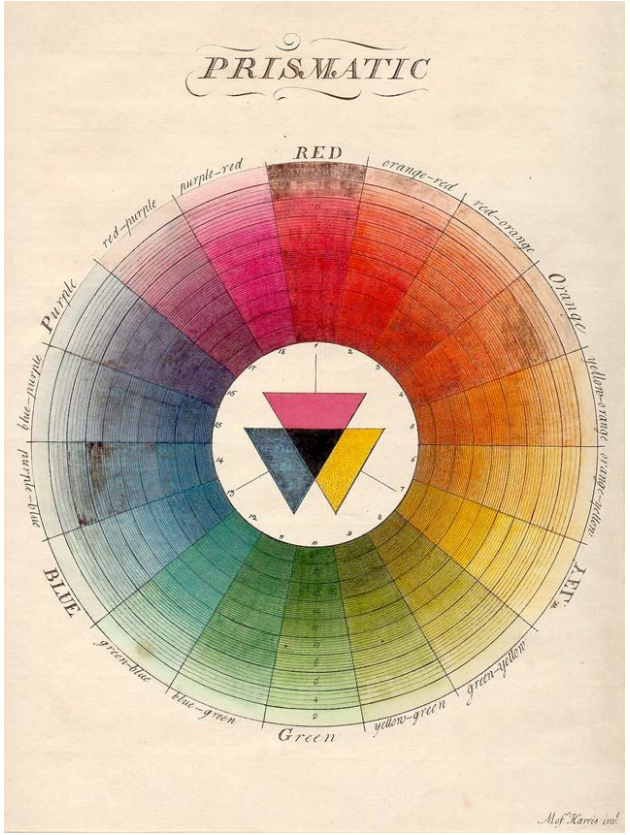
Choosing good colors

Color theory, color wheel, harmonic colors
https://www.canva.com/colors/color-wheel/

Generate them!

https://coolors.co/

https://editor.p5js.org/mrehacek/sketches/DQs9Z9Q3s



Random()

Return pseudo-random float from specified range.

```
random()          // from 0 to 1 (not including 1)
random(max)       // from 0 to max (not including max)
random(min,max)   // from min to max (not including max)
random(array)     // pick random element
```

Pssst, hidden gem:

```
let shuffled_arr = shuffle(array)
// randomize order of elements
```

p5 reference

Map()

Take any number and scale it to a new number that is more useful.

```
map(value, start1, stop1, start2, stop2, withinBounds?: boolean)
```

value: the number to be converted
start1: lower bound of the number's current range
stop1: upper bound of the number's current range
start2: lower bound of the number's target range
stop2: upper bound of the number's target range
withinBounds: constrain the number to the newly mapped range (Optional)

Example
Use mouse position to control the size or color of a shape.
Set HSB color mode, and map mouseX values in interval 0-width to 0-360 (hue).

fill(map(mouseX, 0, **width**, 0, 360));

p5 reference

Sketch #3: Loops and grids

Try **for loop** for drawing (js docs: MDN: Loops and iterations).

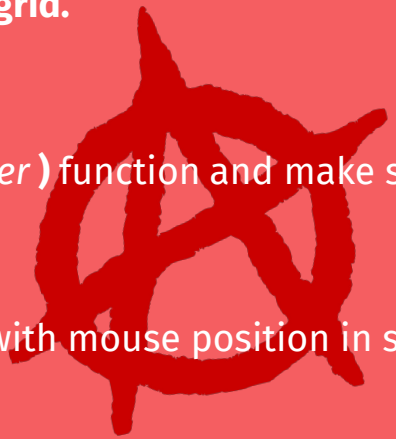
```
for (let i= 0; i< 5; i++) { ... }
```

Draw any **shape** and **color.**

Make inner **for loop** to generate an ordered **grid.**

Next, use **random (min**: *number*, **max**: *number* **)** function and make some chaos.

Make it interactive! For example, use **map()** with mouse position in some variable.



Finished example

Coding train tutorial

Geometric patterns

Inspiration in art / Helena Lukášová

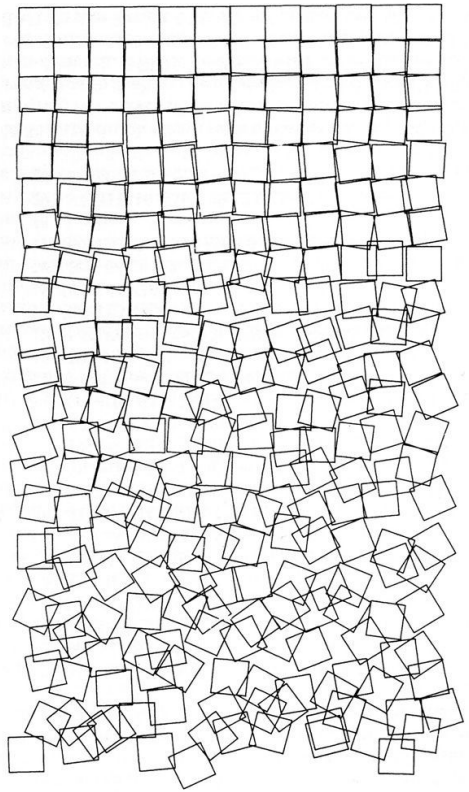
« PAINTINGS »

Schotter (Gravel) / Cubic Disarray, 1968

Georg Nees

One of the first computer-art pieces.

« Image 38 [«Schotter»] is produced by invoking the SERIE procedure [...]. The non-parametric procedure QUAD serves to generate the elementary figure which is reproduced multiple times in the composition process controlled by SERIE. QUAD is located in lines 4 through 15 of the generator. This procedure draws squares with sides of constant length but at random locations and different angles. From lines 9 and 10, it can be seen that the position of a single square is influenced by random generator J1, and the angle placement by J2. The successively increasing variation between the relative coordinates P and Q, and the angle position PSI of a given square, is controlled by the counter index I, which is invoked by each call from QUAD (see line 14). » Nees



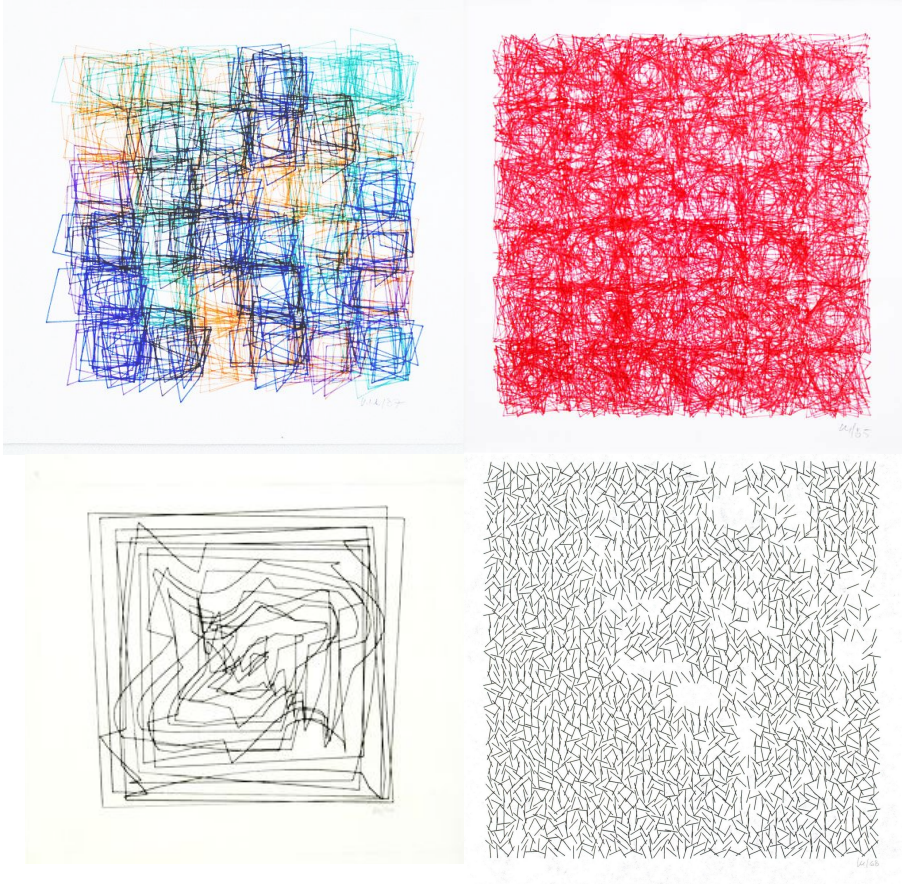
Source

« ARTIST »

Vera Molnár

Profile

Sotheby's: The Life and Work of Vera Molnár

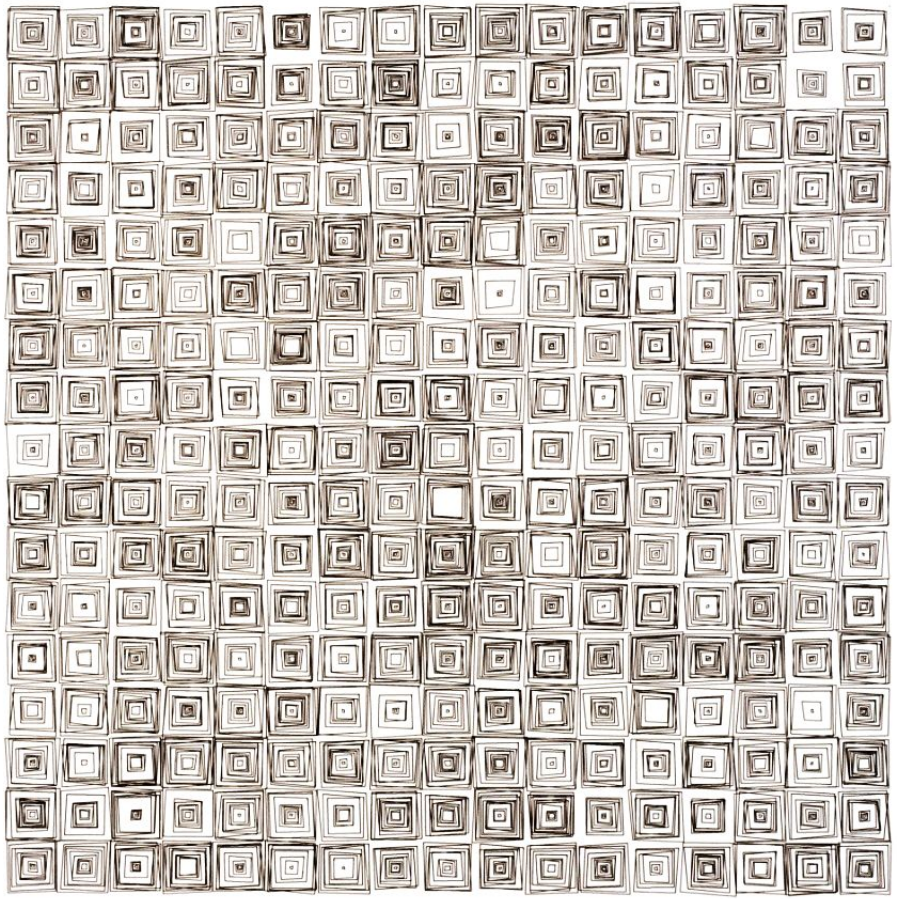


« PAINTINGS »

(Des)Ordres, 1974

Vera Molnár

Wordplay on “désordres” (disorders) and “des ordres” (some orders), which implies that within the apparent disarray one can find an underlying logic.

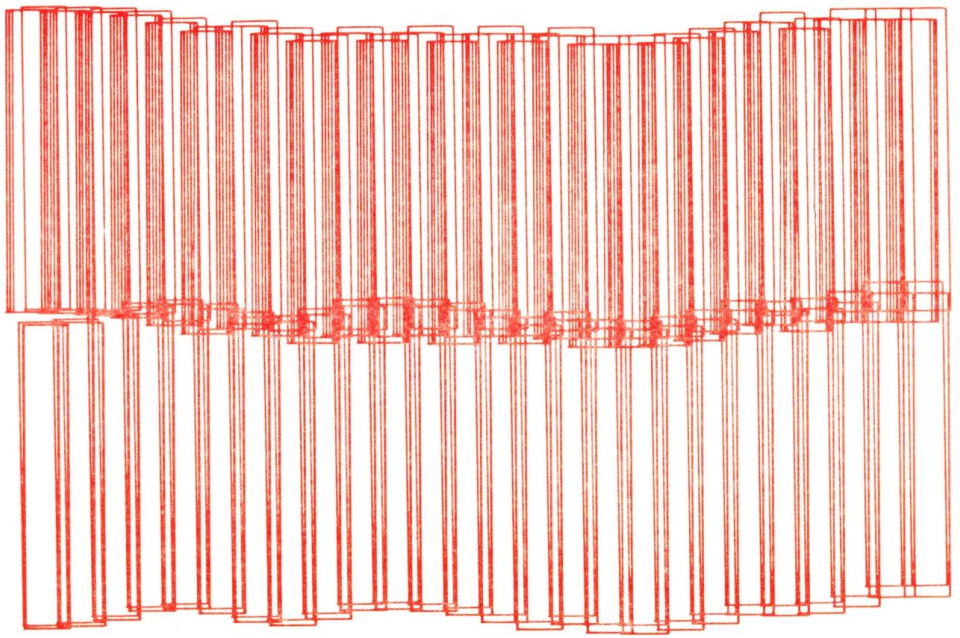


« PAINTINGS »

1975

Vera Molnár

Invitation card for an exhibition.



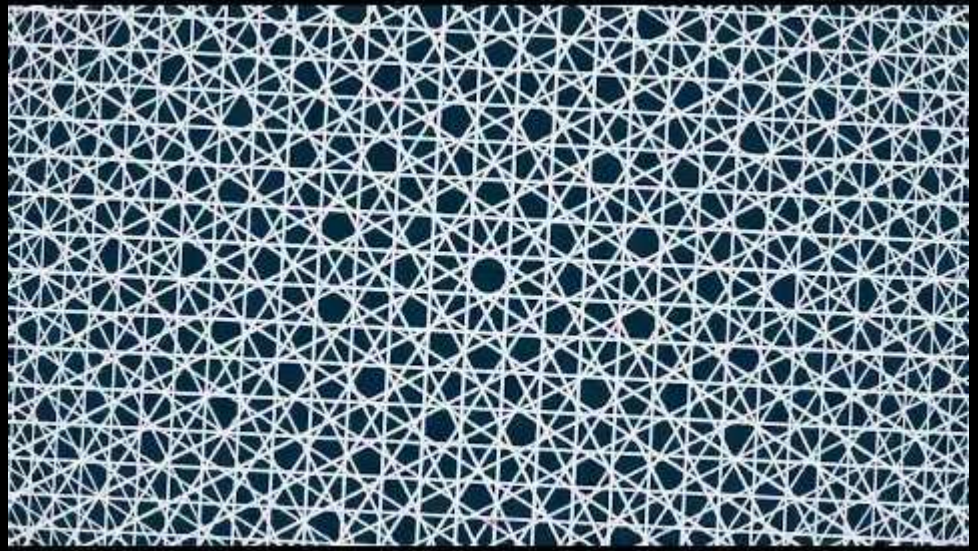
Source

« VIDEO »

Line of Symmetry

Thedotisblack / David Mdrugala

Islamic patterns are considered the first generative artworks.

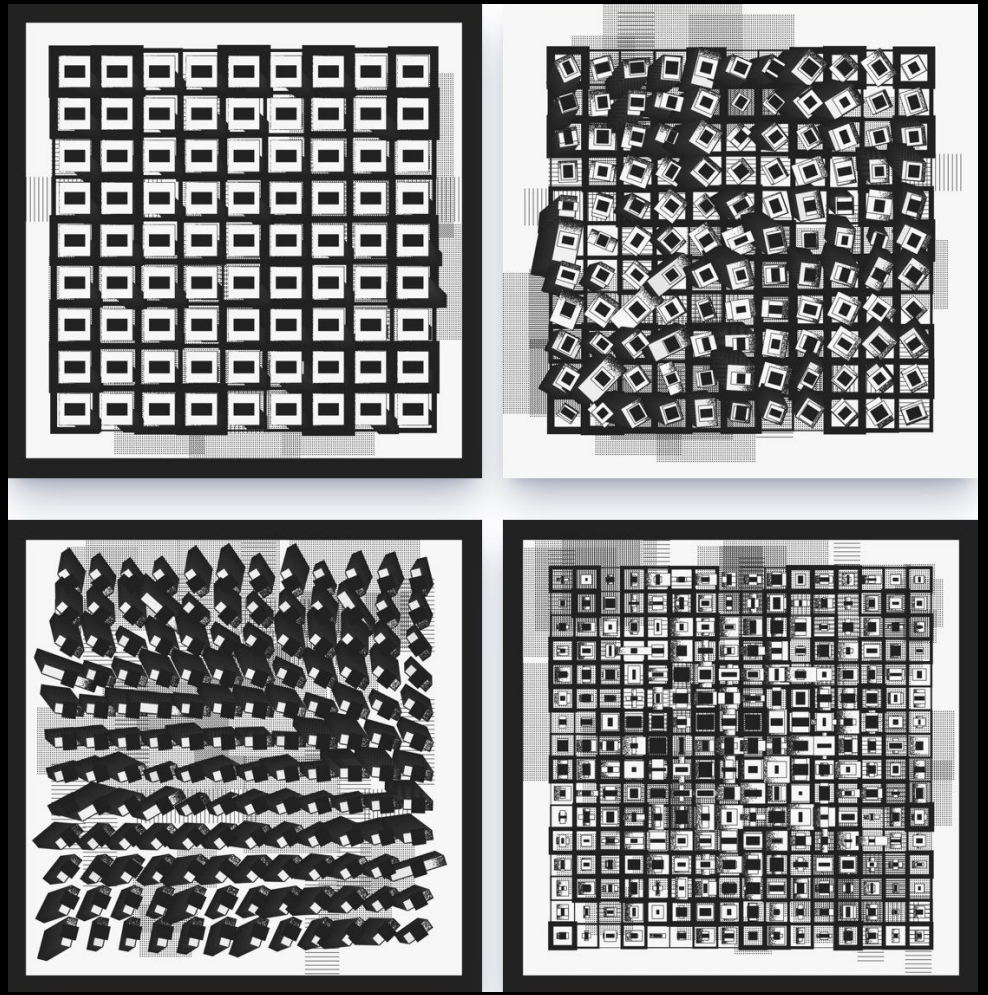


« VIDEO »

Decadence B&W

Thedotisblack / David Mdrugala

https://twitter.com/davidmrugala

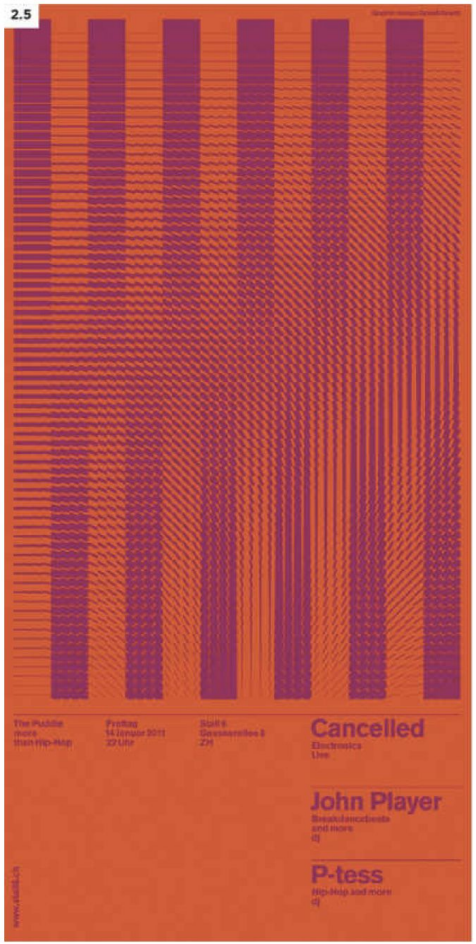


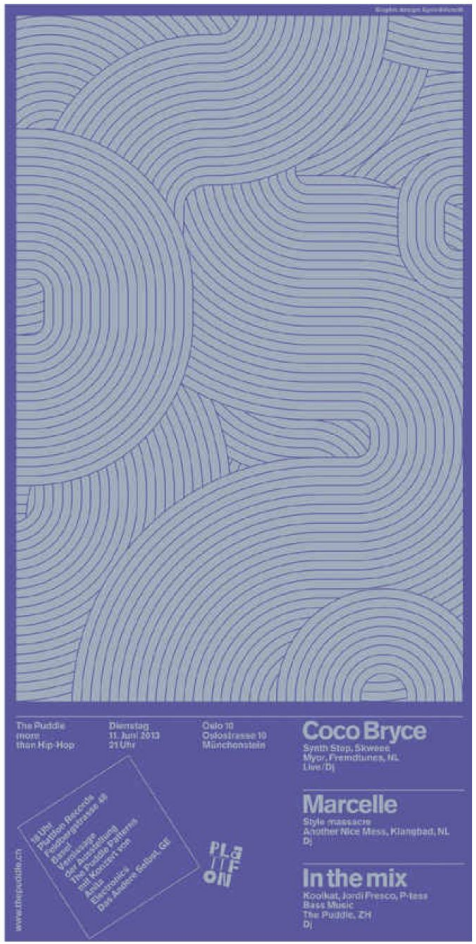
« POSTERS »

Puddle Builder

Andreas Gysin and Sidi Vanetti

Drawing tool that generates a range of graphics from numeric sequences that were ultimately used as promotional posters
and flyers for the Puddle, a live electronic music event around Zürich.





https://www.creativeapplications.net/processing/puddle-builder-03-processing/

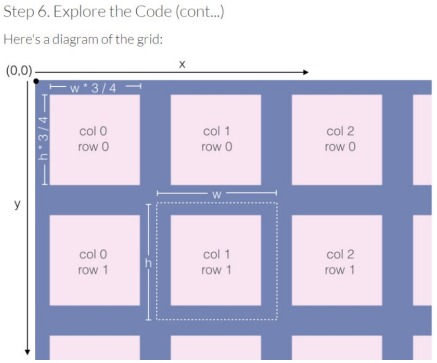
« CLOTHES »

WOVNS

Computational textiles

Set of tutorials for Processing for generating patterns on clothes.

https://www.wovns.com/tutorials/designing-computational-textiles-with-processing/



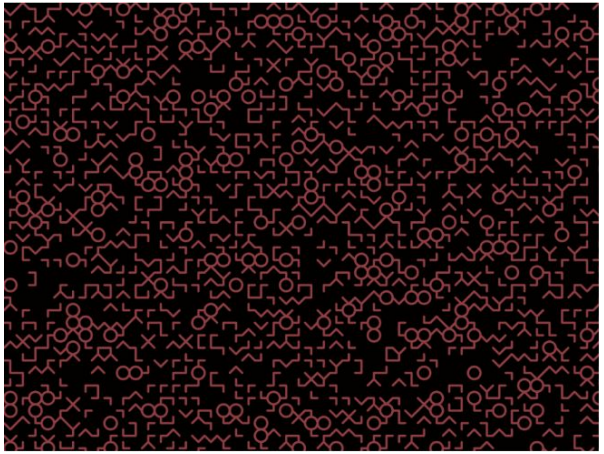

```
rect(w * col + w / 8, h * row + h / 8,
     w * 3 / 4, h * 3 / 4);
```

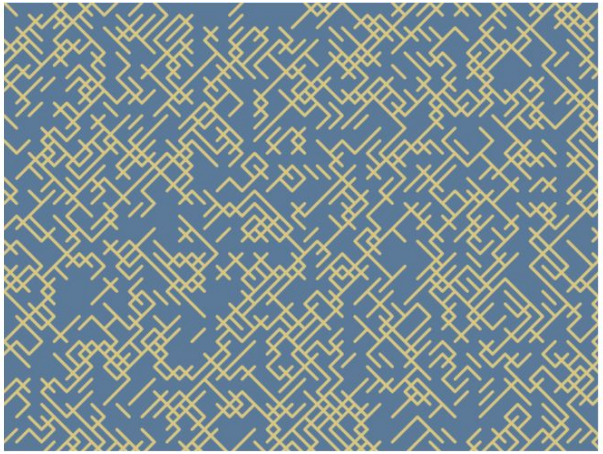
Here's the inside of the two loops, the **rect()** function. It runs 18 * 48 (864) times, drawing a square each time. The first two arguments to rect(), **w * col + w / 8** and **h * row + h / 8** specify the x and y coordinates of the upper left corner of the rectangle. Because the x coordinate includes **w * col**, each time col increases by 1, the x coordinate will increase by w – that is, the rectangle will start w pixels to the right of the previous one. Similarly, each time row increases by 1, the rectangle will move down by h pixels. (The origin, or (0, 0) point of the screen is in the upper left corner.) We add **w / 8** to the x-coordinate and **h / 8** to the y-coordinate to center the square within the cell of the grid. (The square is 3/4 the size of the cell, so we want it to have a margin of 1/8 of the cell on every side.)

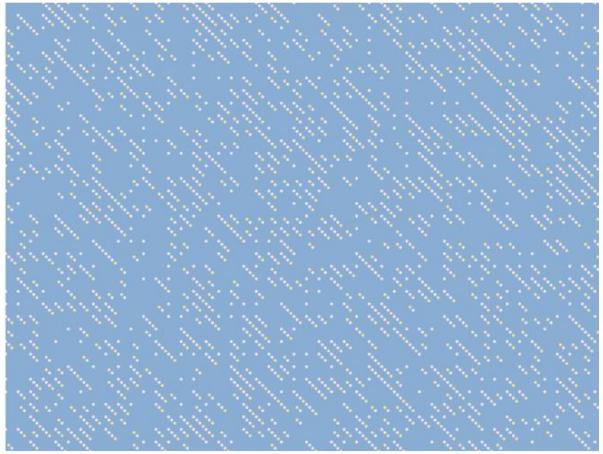
The next two arguments to rect(), **w * 3 / 4** and **h * 3 / 4** specify the width and height of the rectangle (in pixels). Here, we make the width of our

```
save("grid.png");
```

The last line of the progr
PNG file containing the
resulting file in the folde
the **Sketch > Show Sket**







Make-a-pattern co-player edition

Find a partner and make a pattern together.

Patterns on OpenProcessing

CreativeApplications.Net

Database of digital art projects

Login

user: agdm_fimu

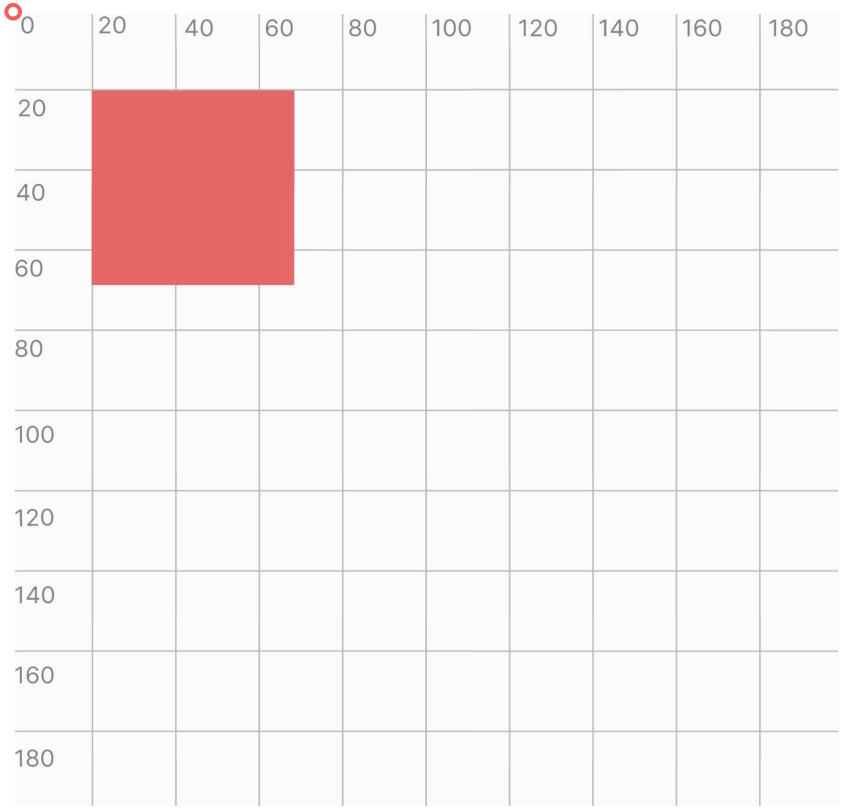
pass: iloveagdm

Creative constraints generator

Transformations

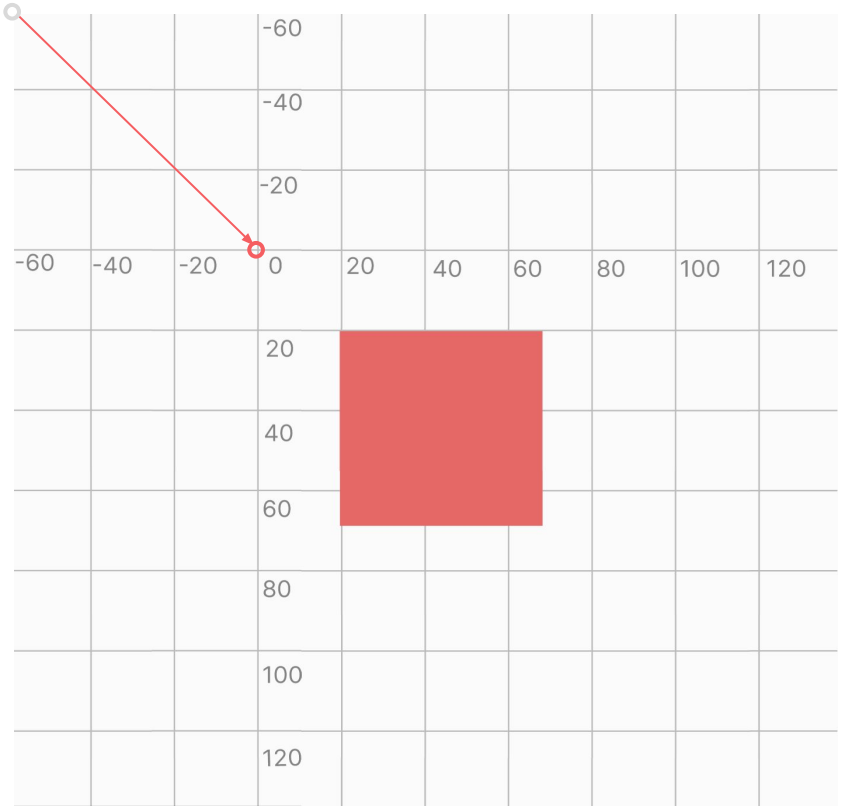
Translate

Move square 60 pixels down and right?



Translate

Move square 60 pixels down and right?



Translate

Built in function that shifts origin (point 0,0) of coordinate system

translate(number, number**)**

It is cumulative:

Resets each draw cycle.

```
translate(10, -10);
translate(0, 50);
```

is the same as

```
translate(10, 40);
```

Rotate

Rotate coordinate system by an angle
- always rotates around origin
- expects angle in radians
- positive numbers rotate objects in a clockwise direction
- accumulates as translate()

rotate(angle **)**
rotate(PI) // HALF_PI

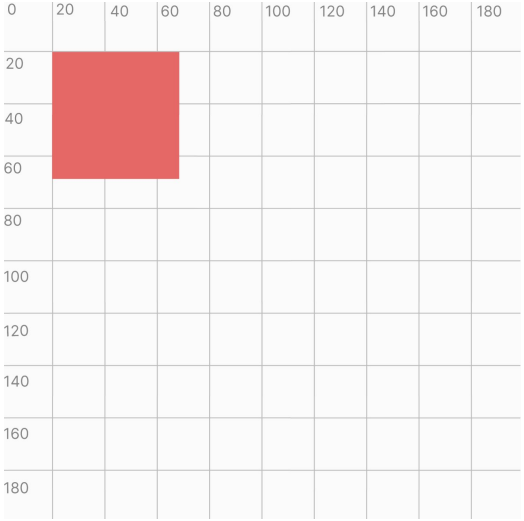
the angle of rotation, specified in radians or degrees, depending on current angleMode

angleMode(MODE **)** // RADIANS, DEGREES

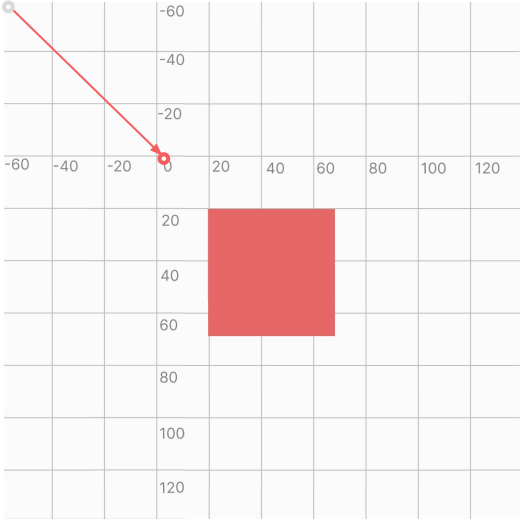
rotate(180 **)**

rotate(radians(degrees **))** // also possibility

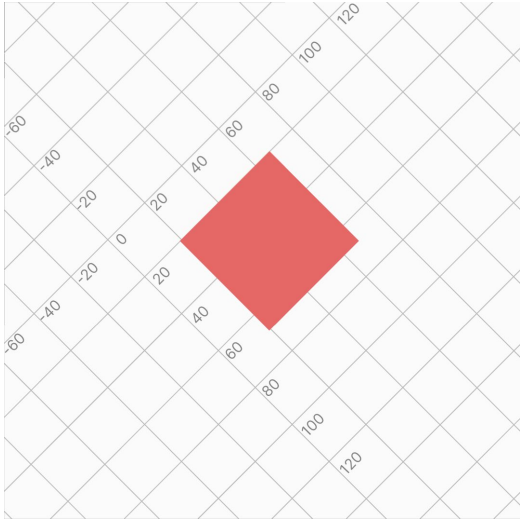
Order matters



Translation (60, 60)



Rotation????



Move it

1. Draw a square in the middle of canvas, by drawing it at origin (point 0,0) and moving it towards center using transformation.

2. Make it rotate. Try function **millis()** which returns number of milliseconds from start of sketch.

3. Make it slower, make it faster, make it stronger, make it better.

4. Play with **framerate()**.

5. Try **sin(x)** together with **map()**.

Finished example

Scale

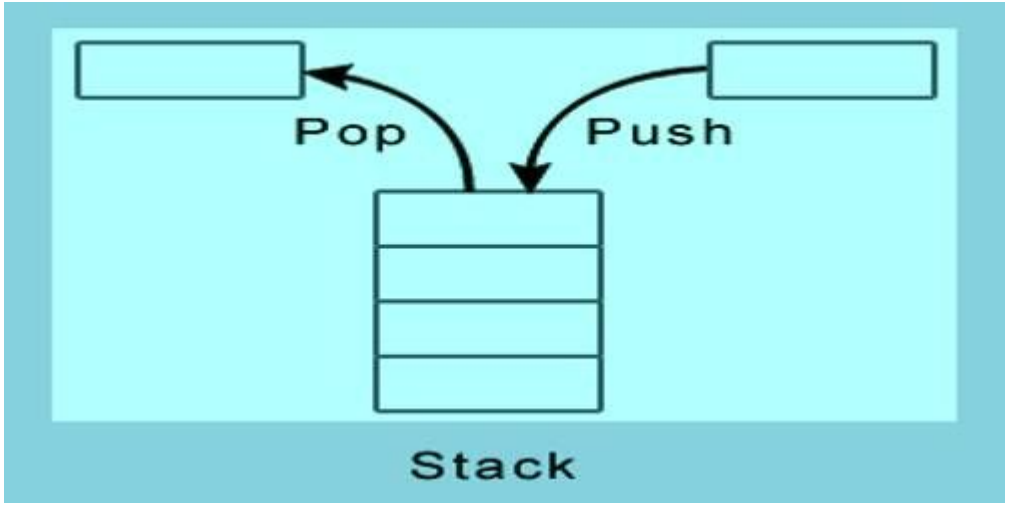
Stretches or shrinks coordinate system

scale(float **)**

Push & pop

The **push**() function saves the current drawing style settings and transformations, while **pop**() restores these settings.

When a new state is started with **push**(), it builds on the current style and transform information.



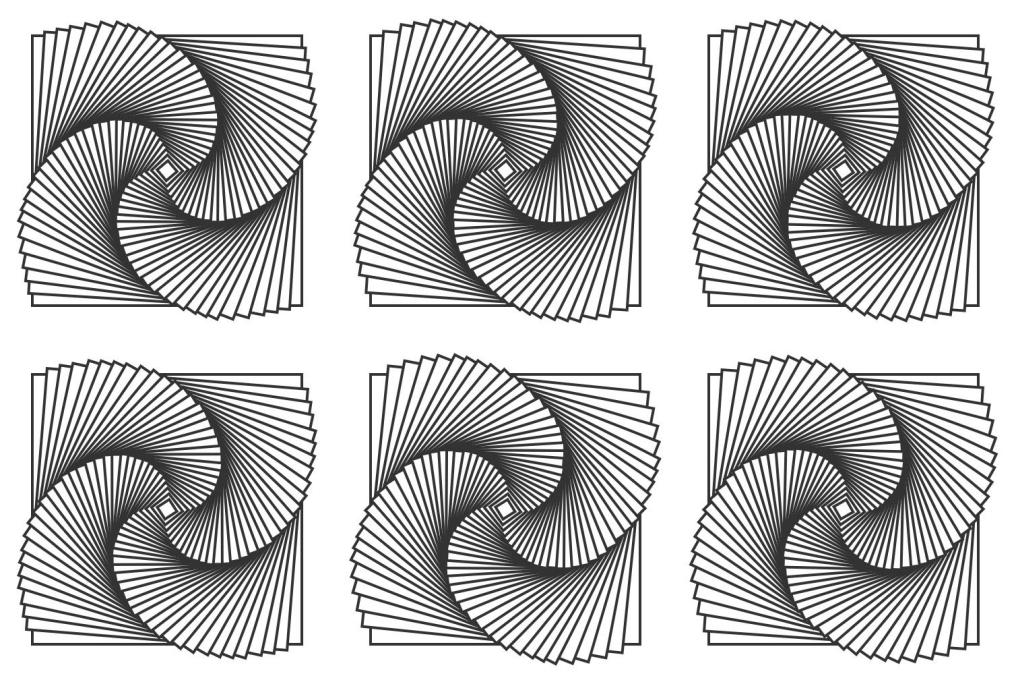
Grid using transformations

Duplicate your sketch with grid or use this starting code. Instead of drawing at the points calculated in loops, draw at origin (0,0) and use transformations to move the drawing.

Use **push()** and **pop()** in the right places, as the transformations inside **draw()** are cumulative.

Complex grid

Create a painting using recursion and transformations like this:



Hints

1. draw a single element from the grid: create a recursive function; don't draw it inside draw()
2. use *for* loops to create a grid, then use transformations to draw the element; do not use push(), pop() or translate() more than once

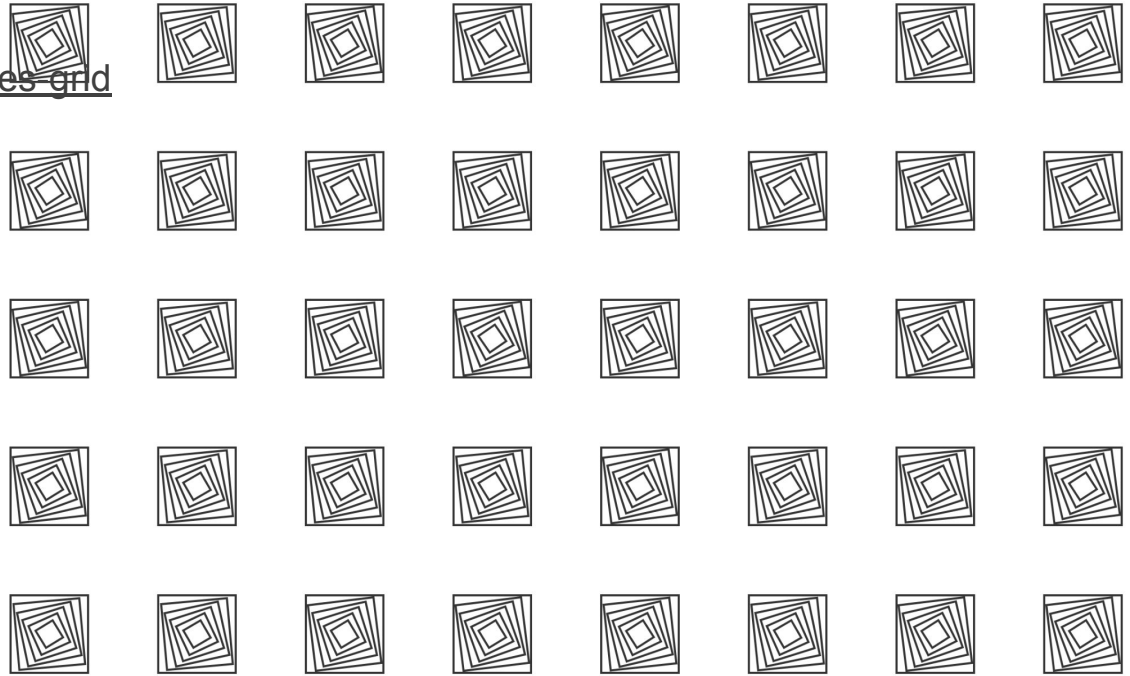
When done

- try different shapes
- experiment with color – handpick a nice color scheme or generate it
- use random() for shape positions, rotations, colors, and more
- don't stop here omg!

Finished examples: 1 (circles), 2 (squares+code)

Interactive example

https://github.com/mrehacek/p5-examples-grid

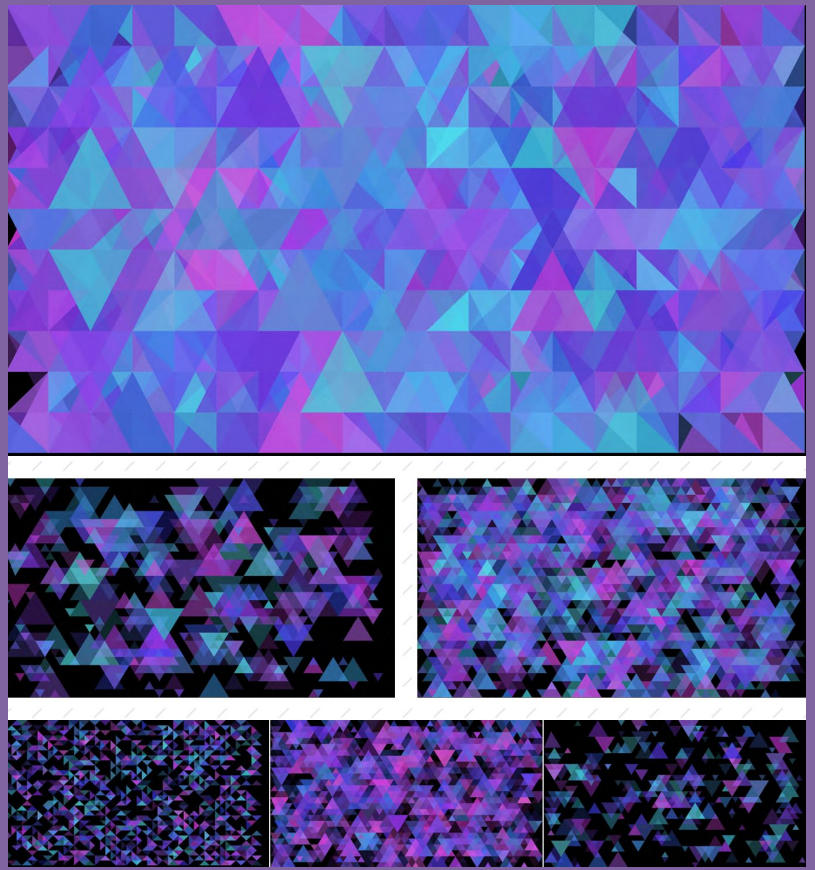


Homework 1: Geometric patterns

Your task is to create a sketch that generates a set of artworks of geometric pattern (for example, for a new desktop background). The emphasis of this project lies on the generative creation, i.e. your sketch is expected to create multiple artworks. This can be achieved by using randomness, input data, mathematical functions or user interaction such as mouse movement. There is no limitation to your ideas.

To export your artwork, use function **save()**. In case you are interested in artwork postprocessing, you can try vector format (SVG).

Technical information regarding the submission and deadline are in interactive syllabus in IS.

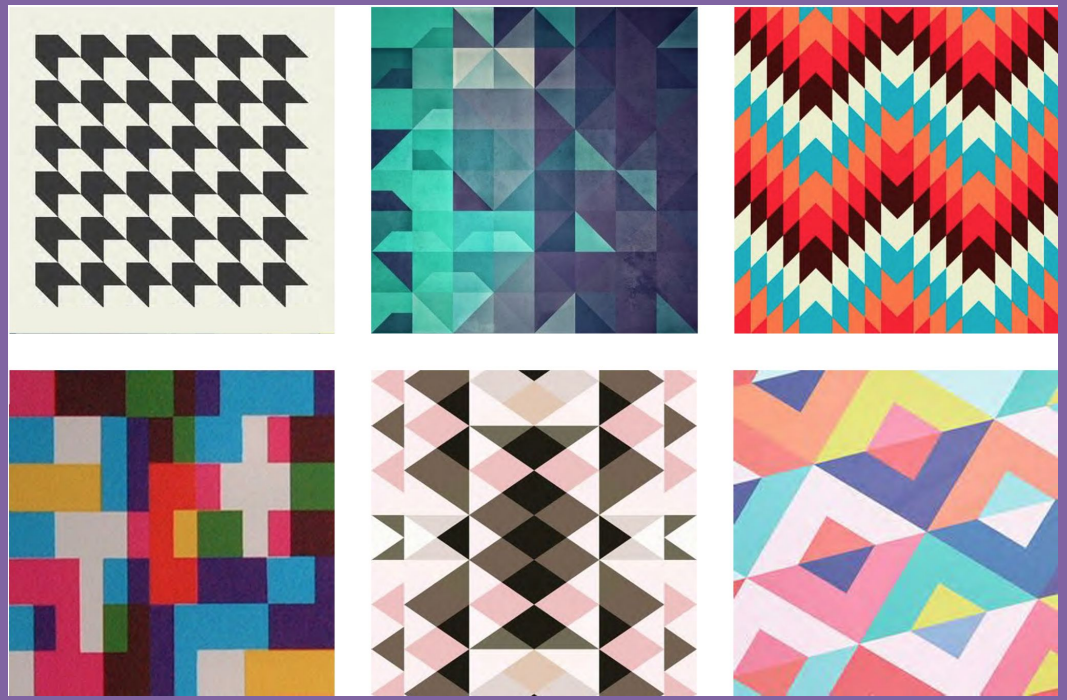


Examples

Tell me, what to do.

Examples

Give me nice colors.



Examples

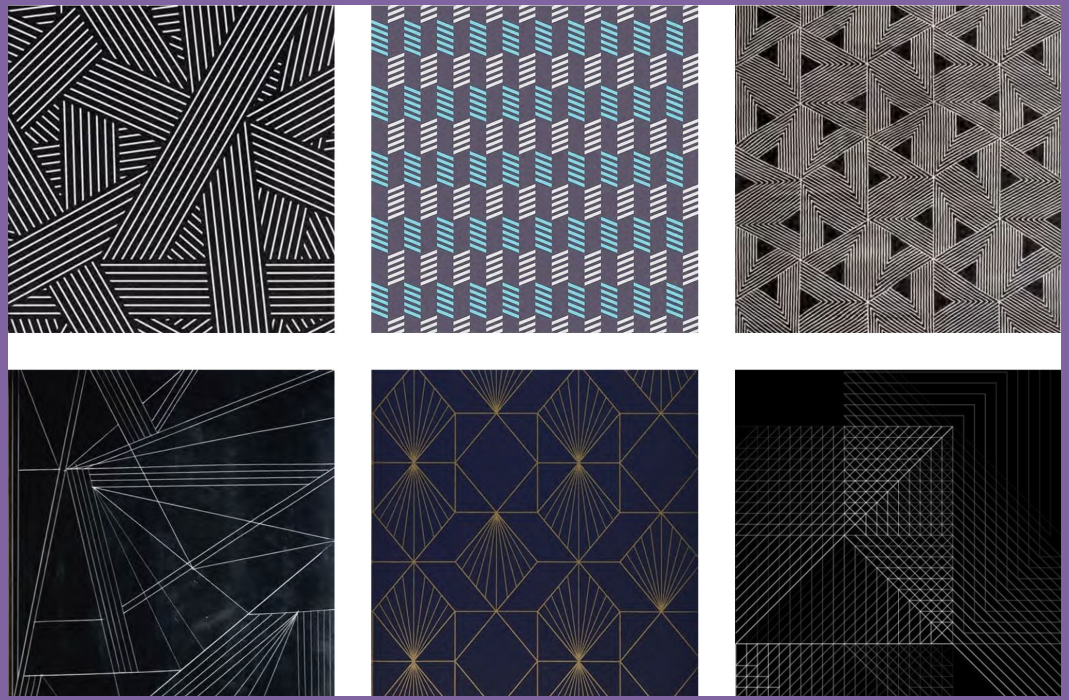
You can be a little bit fuzzy about it. I do like randomness.

Examples

Or you can try just lines.

Examples

You can do a lot with me. A lot.



Examples

Draw a shape.
Maybe try it multiple times.
In a grid.
Or randomly?
Assign it color, or multiple of them.
Again randomly?
Or parametrically?
You say based on position?
Sure. Also the size sounds cool!
Sure, they can overlap.
Yeah, go along and tweak it.
Maybe redraw it with a new shape?
Or just copy from your classmate.
But you can also just ask anyone to give you a tip.

And now it’s you
and geometric patterns

PS

We will have a competition for the best homework!!!

The winner gets a large print of his artwork.

Voting will be anonymous and held at the beginning of the following class through discord. You will have 2 votes, and each teacher will have 4.

Have fun creating your sketches!