PV259
Generative Design Programing

Week 4

Custom shapes & curves

MUNI FI

Marko Řeháček
rehacek@mail.muni.cz

New JS syntax

Object

A collection of related data.

```
let object = {
    number : 100,
    string : “Hello”,
    array :  [
        “some”,
        “text”,
        “for”,
        “example”
    ]
}
```

MDN Docs

Example ->

JS array - basics I.

Declaring & Initializing:

```
const arr = ['a', 'b', 'c'];
```

Accessing array elements (indexing):

```
let first = arr[0];       // indices start with 0
```

Changing an Array Element:

```
arr[0] = 'x';             // ['a', 'b', 'c'] → ['x', 'b', 'c']
```

Useful methods:

```
arr.push('elem');         // ['a', 'b', 'c'] → ['a', 'b', 'c', 'elem']
let last = arr.pop();     // last = 'elem' && arr = ['a', 'b', 'c']
let arrLen = arr.length;  // arrLen = 3
```

JS array - basics II.

Arrays are special kinds of objects.

- Objects use names to access its "members".
- Arrays use numbers to access its “elements".

Because of this, you can have variables of different types in the same Array.

```
const shapes = [
                 "Star",
                 3,
                 { size: 100, type: “Square” }
               ];
```

Stars

Draw many stars with random size and position. Define parameters of star as an object, and store multiple generated objects in array.

Starter code

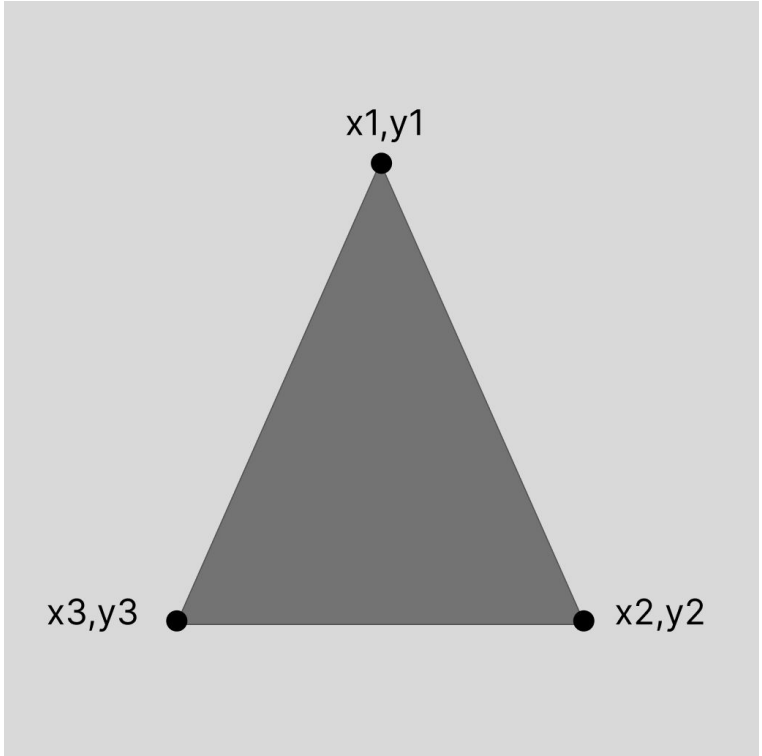
Custom shapes and curves

The commands

```
beginShape()

vertex(x1,y1)
vertex(x2,y2)
vertex(x3,y3)
vertex(x1,y1)
…

endShape()
// or endShape(CLOSE)
```



Example ->

Shape types

`beginShape( TYPE )`

- **POINTS**
- **LINES (makes line between pairs of points, e.g. 1—2 3—4... it's dashed)**
- **TRIANGLES**
- **TRIANGLE_STRIP**

... more in reference of beginShape

Spline curves

“Cables attached to points”. To create them, you define each point using **`curveVertex()`**:

```
beginShape() // no parameter!

curveVertex(x,y) // or curveVertex(x,y,z)
…

endShape()
```

You need at least 4 vertices to draw curve between 2. and 3. point

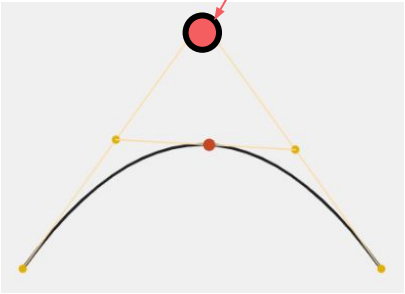
- adding fifth point will draw curve also between 3. and 4. point …

Example ->

Beziér curve

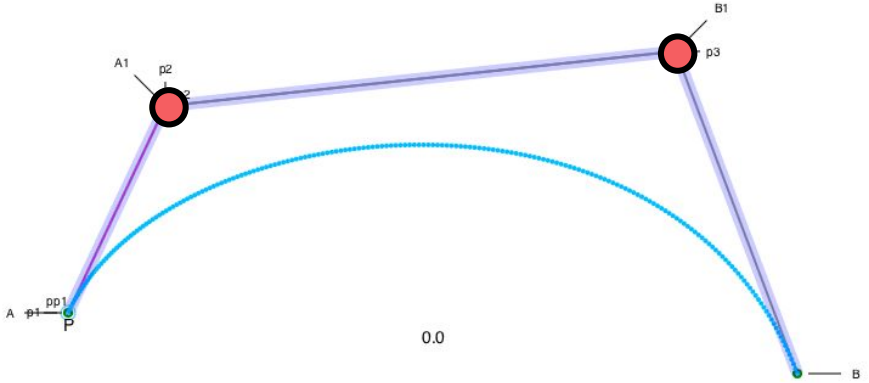
Quadratic: 1 control point

`quadraticBezier()`

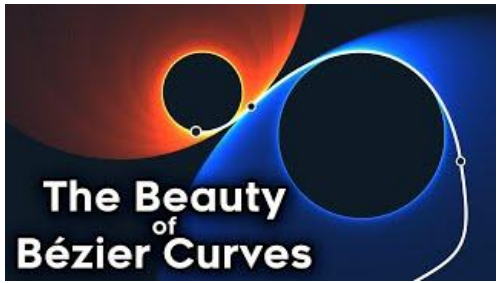


Cubic: 2 control points

`bezierVertex()`



https://sebastiandedeyne.com/tags/bezier



Contour

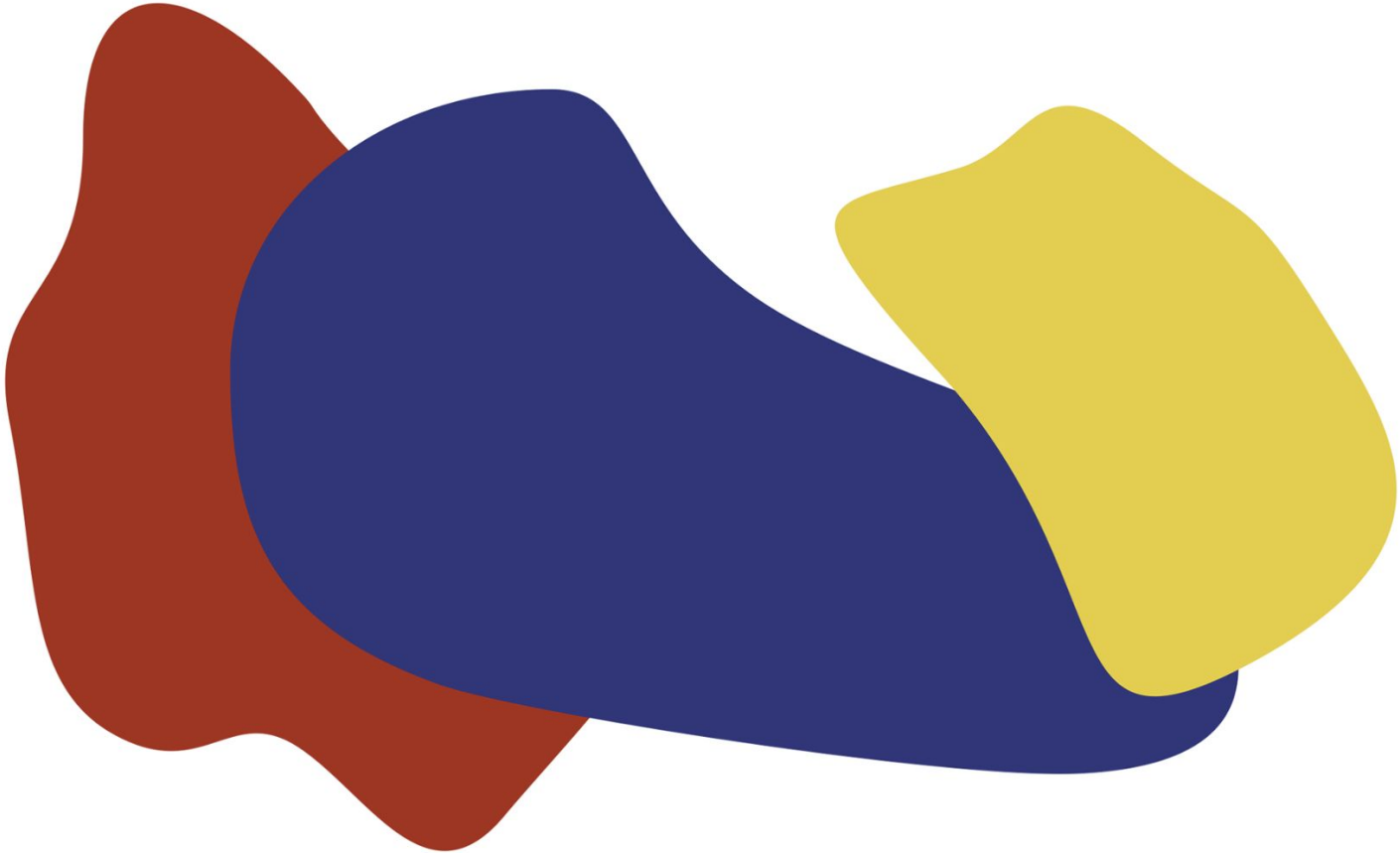
Create negative shape within shape. Only possible inside **beginShape**() / **endShape**() sequence.

```
beginShape()

// create shape in clockwise order
beginContour()
// inner shape in counter-clockwise order
endContour()

endShape()
```

Example ->



Blob maker

Create a shape using mouse clicking.

- ❏ create an array, which will store vertices (points) of the shape
- ❏ on click, save current mouse position (**mouseX**, **mouseY,** **mousePressed()**) into the array as object

  `{x: _, y: _ }`
- ❏ draw spline curves using **beginShape()**-**curveVertex(**x,y**)**-**endShape()** sequence from the points in the array

Classes in JS

Not always needed.

```
class Person {
  constructor() {
    this.age = 10;
    this.name = “David”;
  }
  makeOlderBy(n){
    this.age += n;
  }
}

const person = new Person();
obj.makeOlderBy(23);
```

MDN Docs

Example ->

Example 2 ->

Classes in JS

Another example.

```
class Rectangle {
  // define private member variables
  #height = 0;
  #width;

  constructor(height, width) {
    this.#height = height;
    this.#width = width;
  }
}

const r = new Rectangle(100, 200);
```

MDN Docs

Blobs as classes

- turn blob from previous sketch to `BlobbityBlob` class (cannot name it Blob, it clashes with builtin JS class)

```
class BlobbityBlob {
    points = [];

    addPoint(x, y) {
        …
    }

    draw() {
        …
    }
}
```

Bonus: wet and sharp

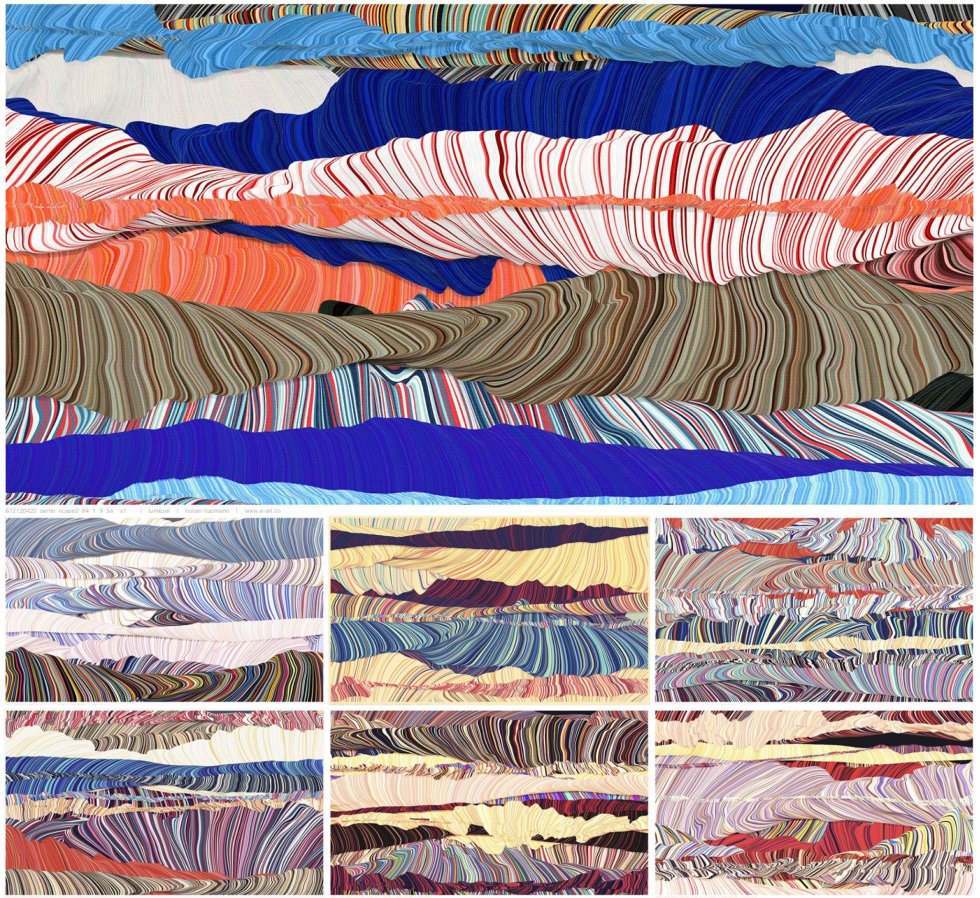
Create a drawing with two shapes in black and white that **represent** (no text, feeling) the words ***wet*** and ***sharp***.
The shapes have to be created with the **beginShape**() and **endShape**() functions.

→ IMAGE

perlinScape 2021

Holger Lippmann, 2021

The series "perlinScape 2021" emerged from the initial linear noise structures, from sinuous and organically warping waves, an increasingly refined compositional structure, a pictorial motif, the apparent motif "landscape". a gently undulating landscape, which can be found in the German Erzgebirge area of my childhood as well as around my current place of residence in Brandenburg.

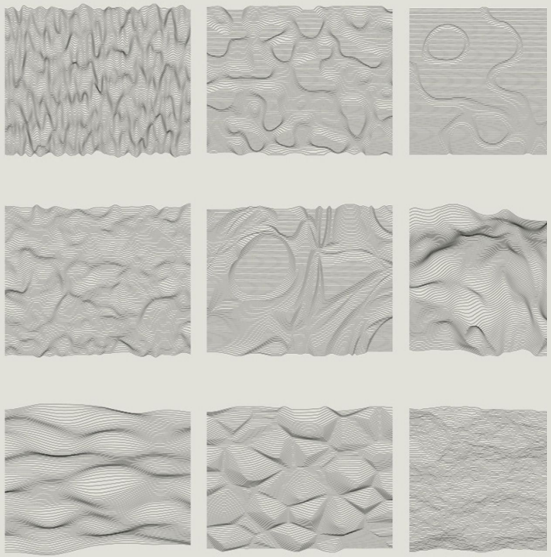


Behance

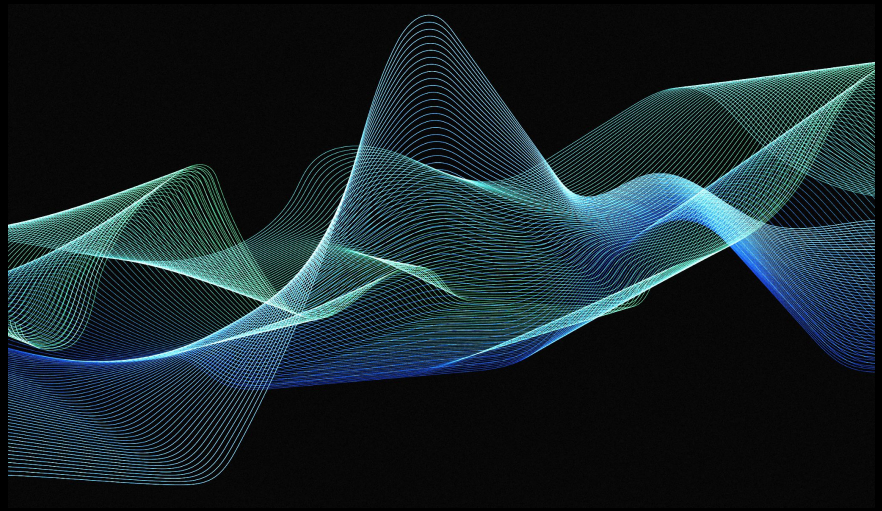
→ IMAGE

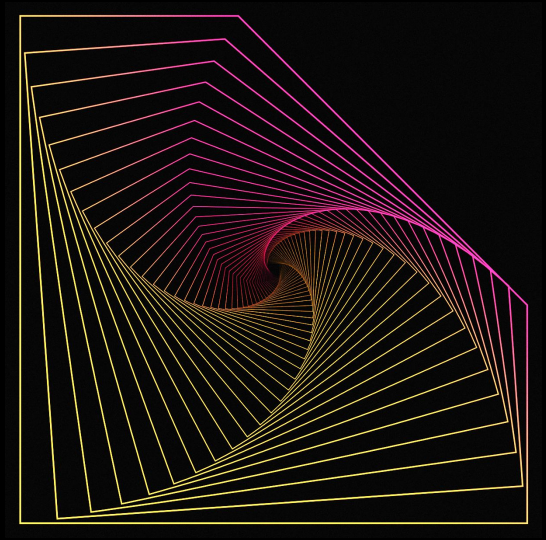
60 Abstract Waves

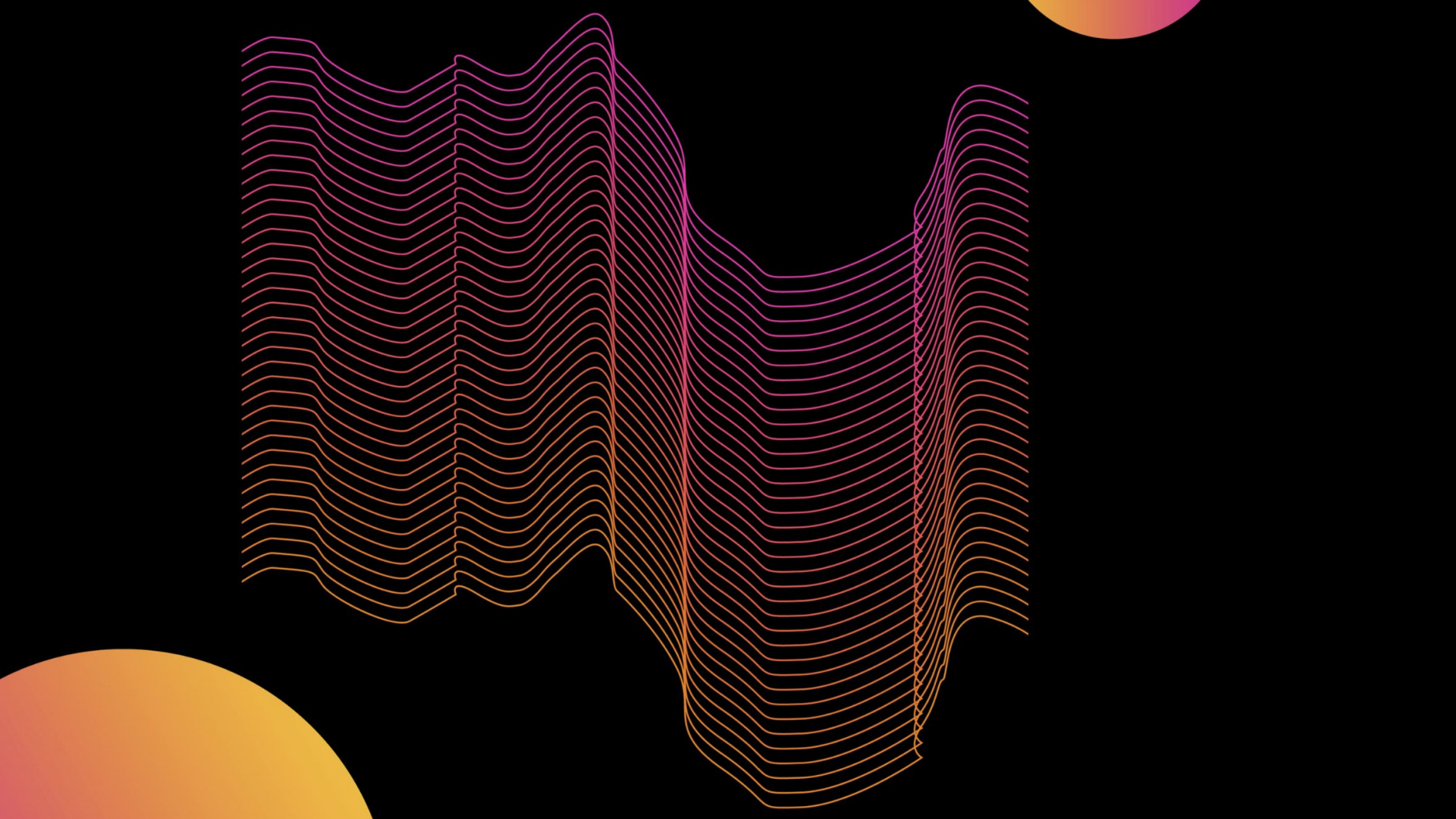
Artem Ottoson, 2022



Behance



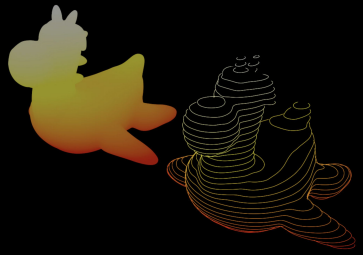


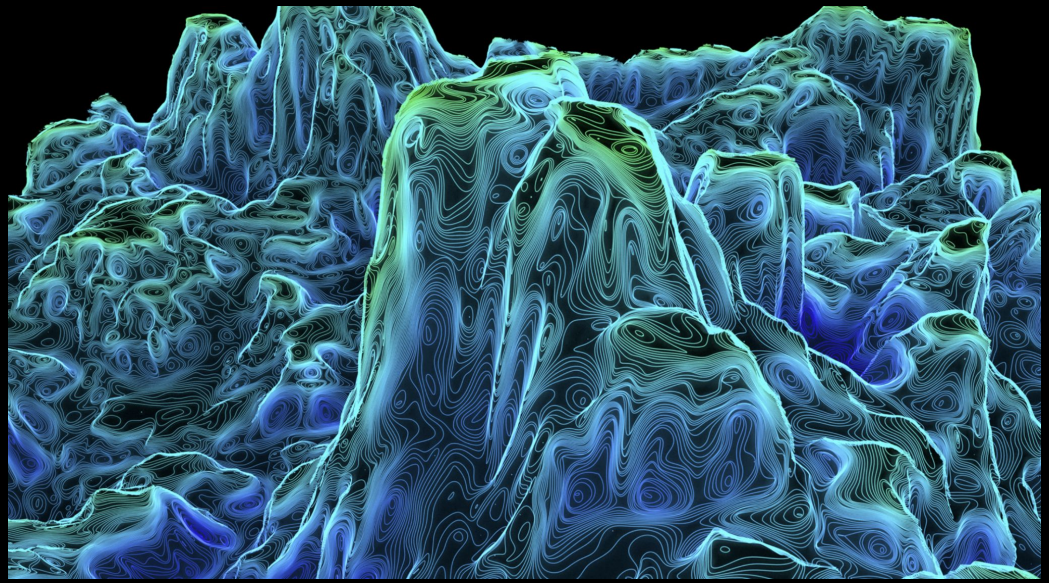


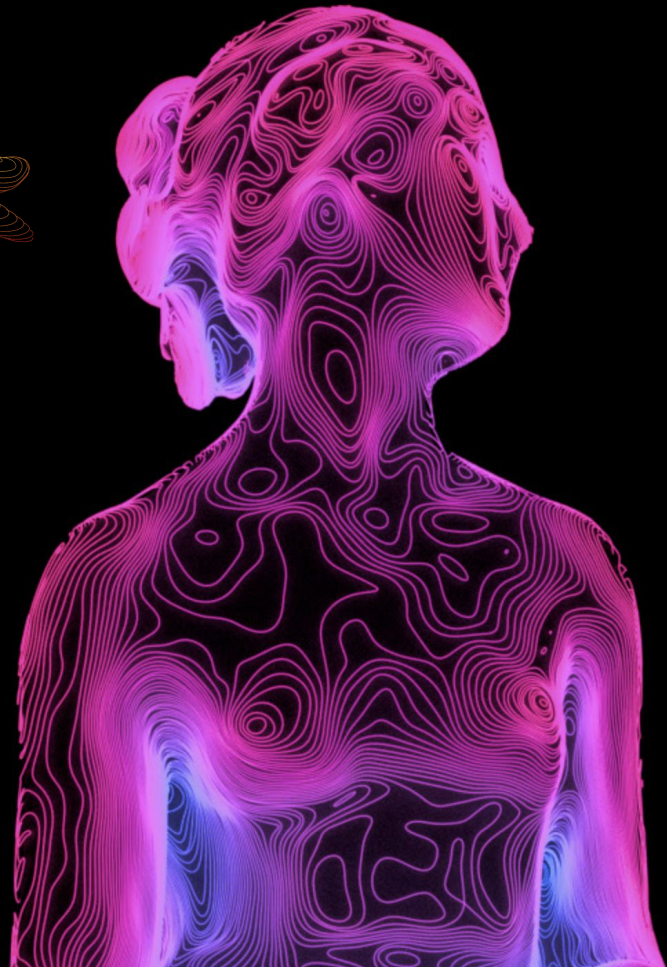
→ IMAGE

Occlusion Based Isolines Procedure

Pascal Beeckmans



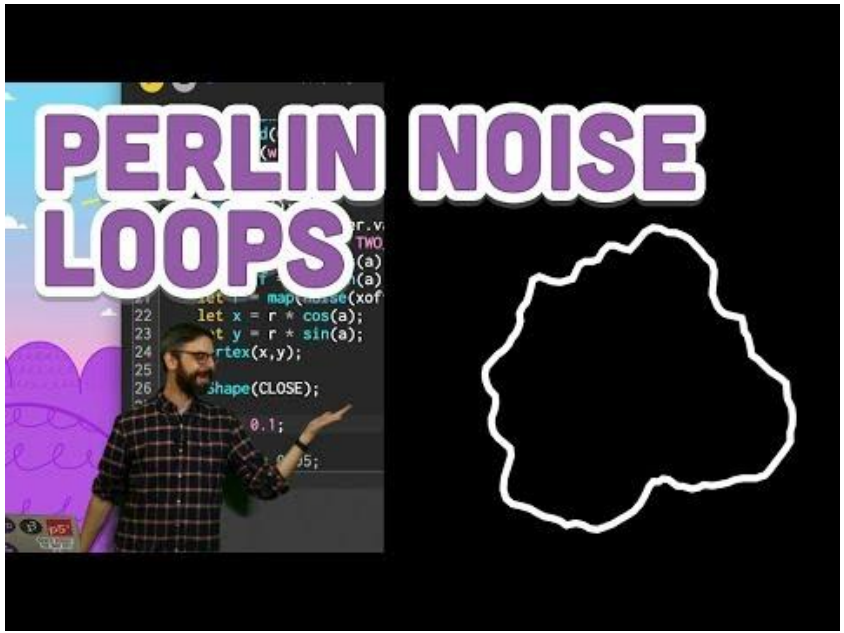


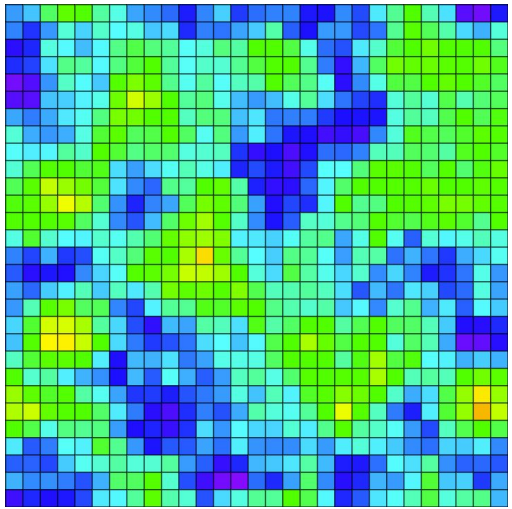


Behance

Polar coordinates

If you travel through noise map in a circle, you end up in the same position.



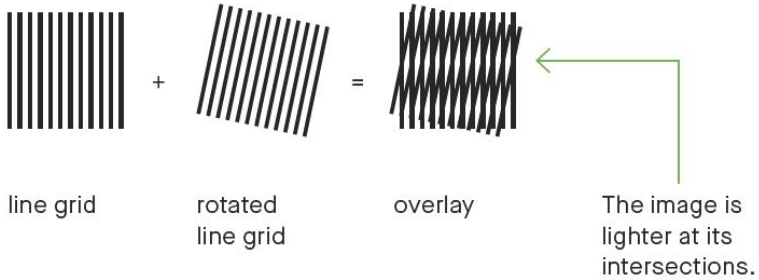


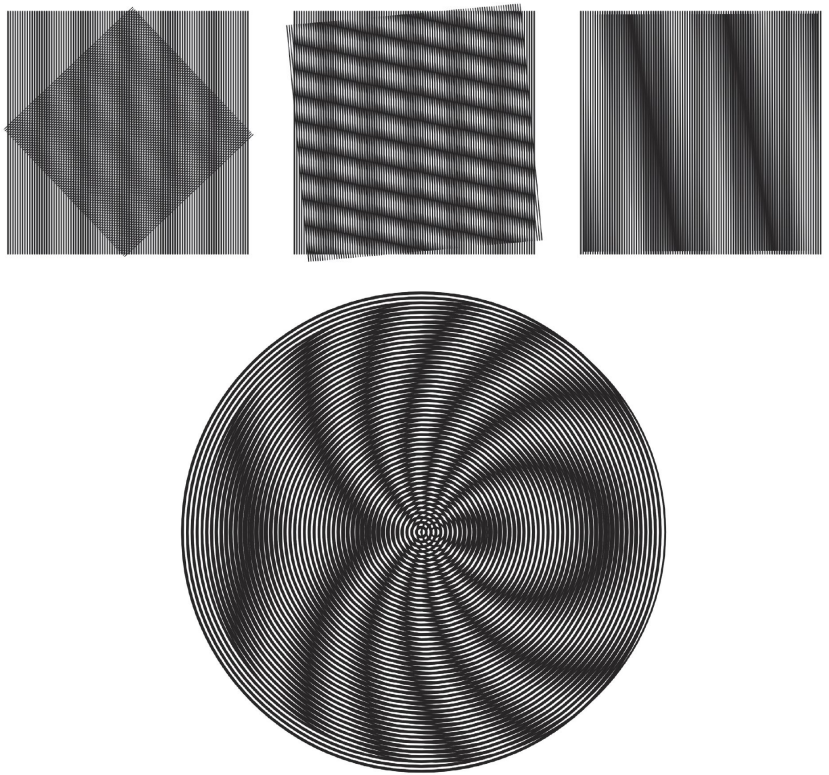
W04 lecture -> 2D noise

Example ->

Moiré

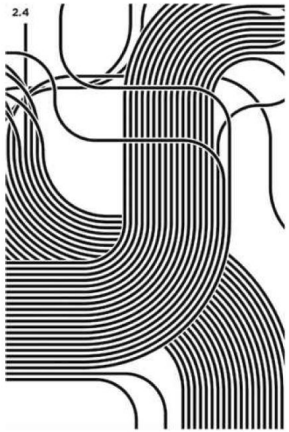
“Usually considered a mistake in printing technology, but desirable for us. By laying one graphic grid over another identical grid and moving it, you can generate unexpected optical illusions that you can change in real time.”

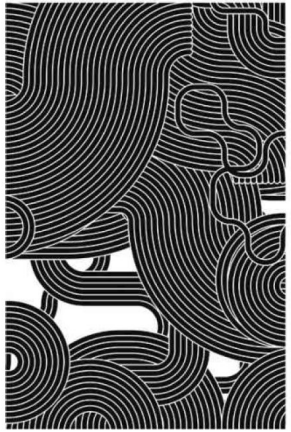




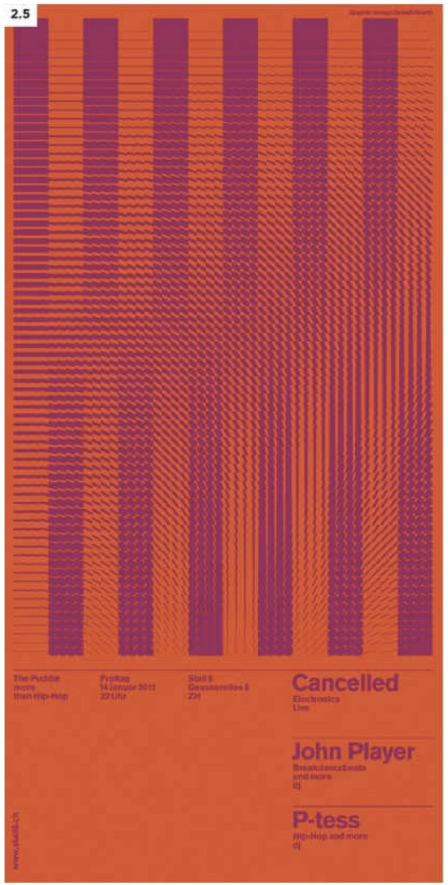
Images and text from the book Generative Design: Visualize, Program, and Create with JavaScript in p5.js, Benedikt Groß Hartmut Bohnacker Julia Laub Claudius Lazzeroni

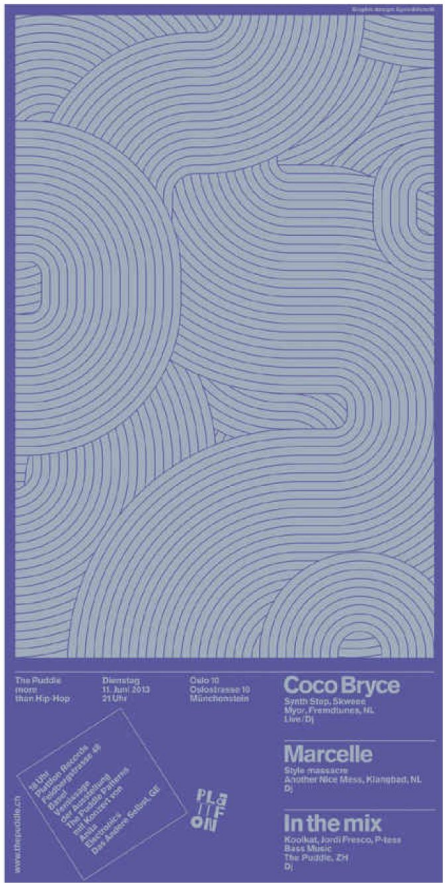
2.4





2.5



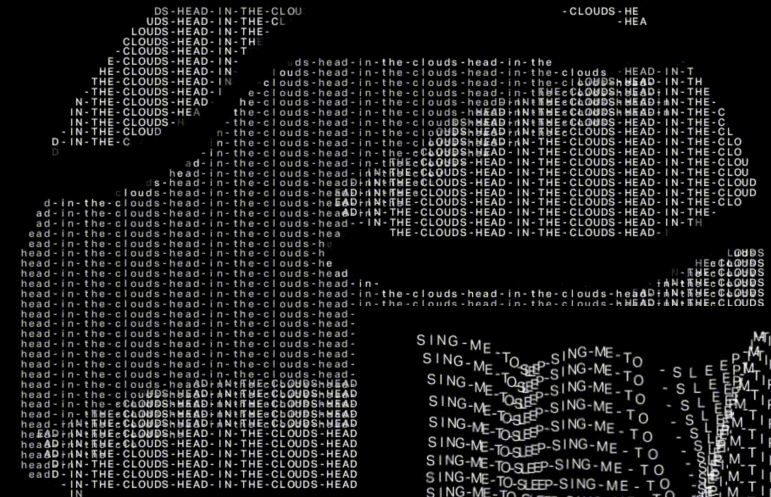


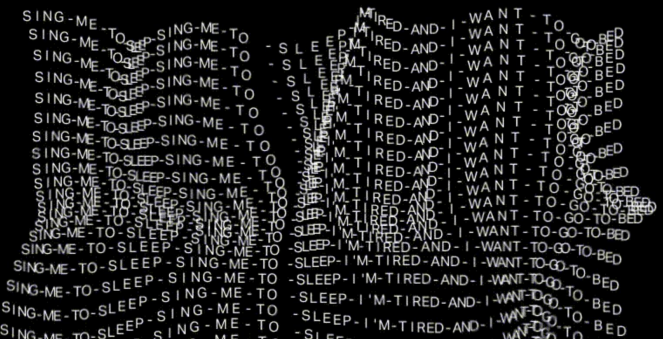
Puddle Builder by Andreas Gysin and Sidi Vanetti

→ DYNAMIC TYPE

CREATIVE CODING EXPLORATIONS II

Andreion de Castro





Behance

JS cheat sheets

JS array lambdas

```
const arr = ['a', 'b', 'c'];

arr.forEach(element ⇒
console.log(element));
// "a"
// "b"
// "c"
```

```
arr.filter(e ⇒ e ≠= "b");
// ["a", "c”]
```

example:
Remove unwanted elements from array.

Find object by its property:

```
const fruits = [
  { name: "apples", quantity: 2 },
  { name: "bananas", quantity: 0 },
  { name: "cherries", quantity: 5 }
];

fruits.find(f ⇒ {
    return f.name == "cherries"
});
// OR this one-liner
fruits.find(f ⇒ f.name == "cherries");
```

Arrays in JS

```
['a', 'b'].concat(['c']) //['a','b','c']
['a', 'b'].join('~') //'a~b'
['a','b','c'].slice(1) //['b', 'c' ]
['a','b','b'].indexOf('b') // 1
['a','b','b'].lastIndexOf('b') //2
```

```
['a','b','c'].forEach(x => console.log(x))

[1,2,3].every(x => x < 10 )//true
[1,2,3].some(x => x < 2 )//true
[1,2,3].filter(x => x < 2 )//[1]
```

```
[1, 2, 3].map(x => x * 2 )//[2, 4, 6]
[1, 2, 3].reduce((x,y) => x * y)//6
[2, 15,3].sort()//[ 15, 2, 3 ] 😅
[1, 2, 3].reverse()//[ 3, 2, 1 ]
[1, 2, 3].length//3
```

```
const arr = [1, 2, 3]
const x=arr.shift()//arr=[ 2, 3 ],x=1
const x=arr.unshift(9)//arr=[ 9,1,2,3],x=4
const x=arr.pop()//arr=[ 1, 2 ],x=3
const x=arr.push(5)//arr=[1,2,3,5],x=4
```

Objects in JS

Object Declaration

```
const twit = {
  name: "Proful",
  follower: 7815,
  1 : "hi"
}
```

any type

converted to string

- storing key-value pairs.
- data unordered
- keys are unique

Dot Notation

```
twit.name          // "Proful"
twit.follower      // 7815
twit.follower.count
```

Accessing nested props

Square Notation

```
twit['name']       // "Proful"
```

Can be dynamic/variable

```
const {name, followers} = twit
name               // Proful
followers          // 7815

const linkedin = { name }

                   { name: 'Proful' }
```

Empty Object creations

```
const person = {}
const person = new Object()
```

```
const twit = {
  name: "Proful"
}

function change(insta){
  insta.name = "Steve"
}

change(twit)
twit.name // "Steve"
```

- Pass by reference

```
delete twit.name // both key & value

twit.randomKey   // undefined

twit.follower = 5000
```

declared as const but mutable

```
for(const key in twit) {
  console.log(key)       // name
                         // followers
}
```

```
const twit = {
  name: "Proful",
  get profile() {
    return `Hi ${this.name.toLowerCase()}`
  },
  set profile(prof) {
    this.name = "Mr " + prof
  }
}
twit.profile                // 'Hi proful'
twit.profile = 'Steve'
twit.name                   // 'Mr Steve'
```

getter

seter

```
const twit = {
  name: "Proful",
  hi() {
    console.log(`Hi ${this.name}`)
  },
  hello: () => {
    console.log(`Hello ${twit.name}`)
  },
}
```

this refer to twit object

You cannot use this here

Sketches from classes available at
https://editor.p5js.org/mrehacek/collections/LLLIfXiAP