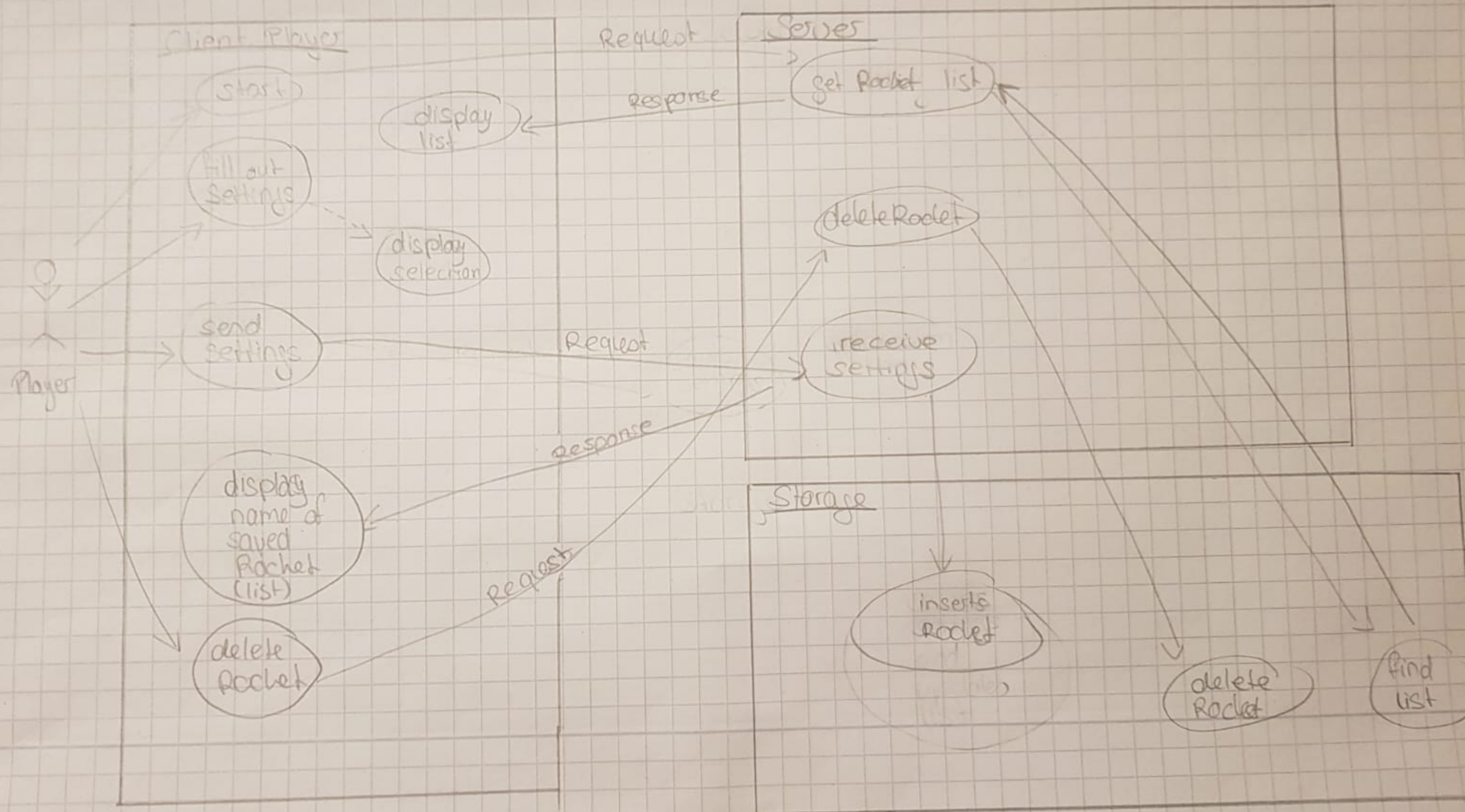


Firework: Use Case Diagramm



Firework: Ul-Scribble

```

end>Settings

```

```
<input> placeholder = "Name"  
type = "text" id = "name"
```

```
<div id="list">
```

```
<div id="canvas"
```

- as HTML ButtonElement
- innerHTML = "delete"

Name: _____

Particle:

Color 1:

Color 2:

```
<select>
  name="formArticle"
  id="formArticle"
  <option>value="dots"
  <option>
    value="triangle"
  <option>
    value="line"
```

Form Package: Dots v

Amount Particle: 0

Lifetime Particle: ∞

Explosion:

Radius:

Small ☒ Medium ☐ Large ☐

Save

```
<input>  
type = "radio"  
name = "drone"  
id = "small"  
id = "medium" id = "large"
```

```
<button  
id = "save"  
type = "button"
```

```
inputs
type = "range"
id = "lifetimeParticle"
"min" = "1" max = "5"
name = LifetimeParticle
```

```
<input>  
type="range"  
id="amountParticle"  
min="100" max="500"  
name="amountParticle"
```

```



```

```

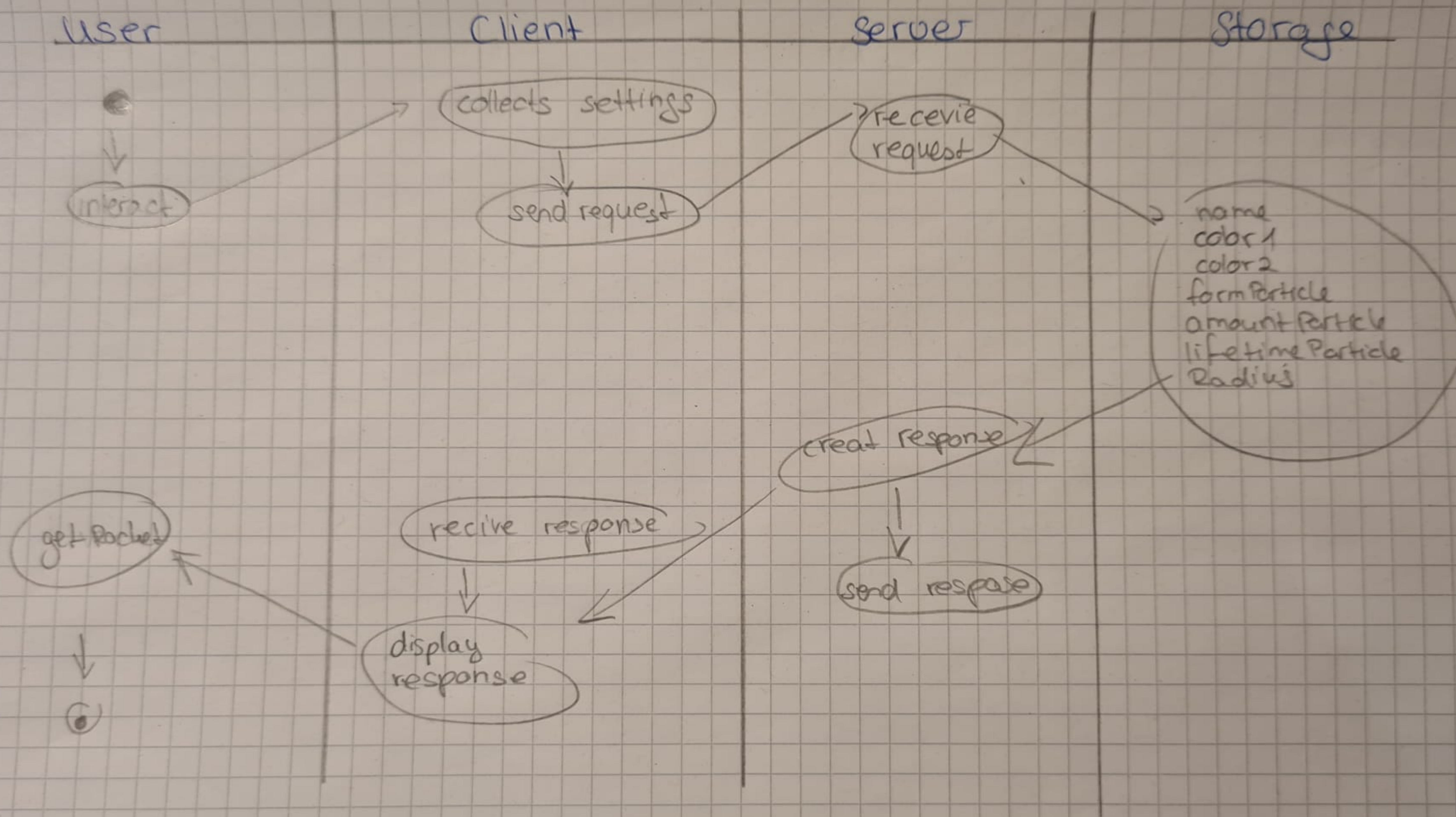


```

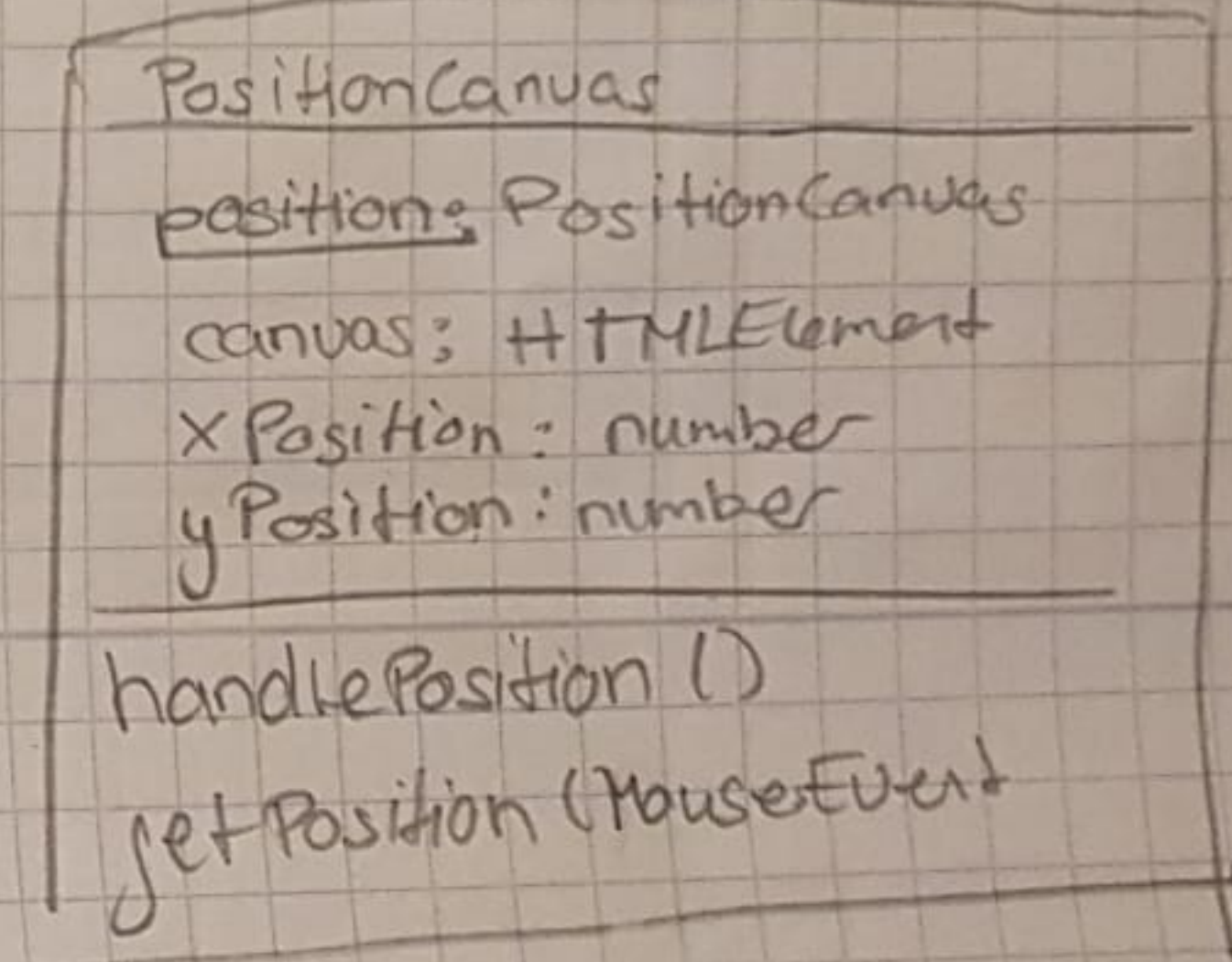
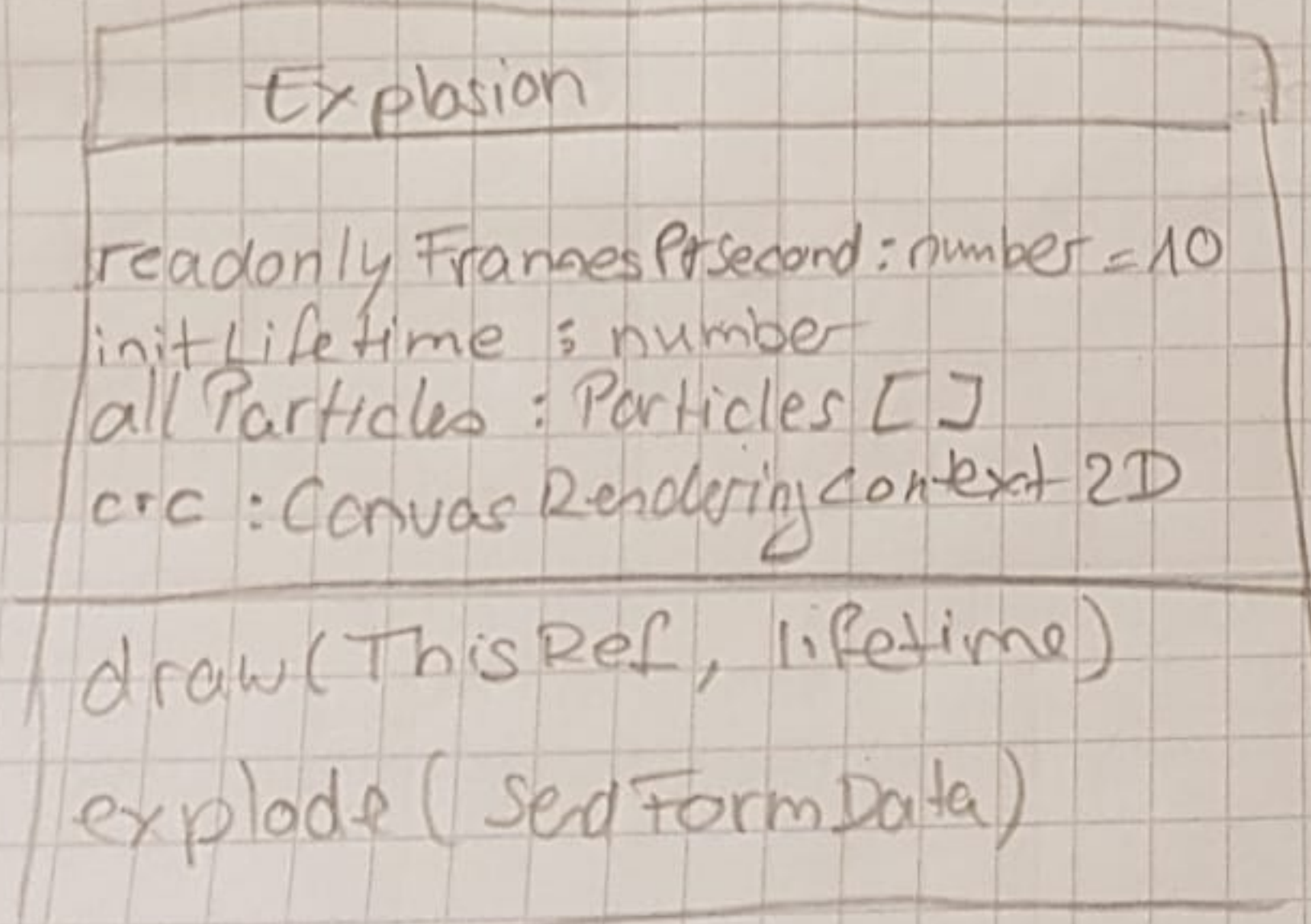
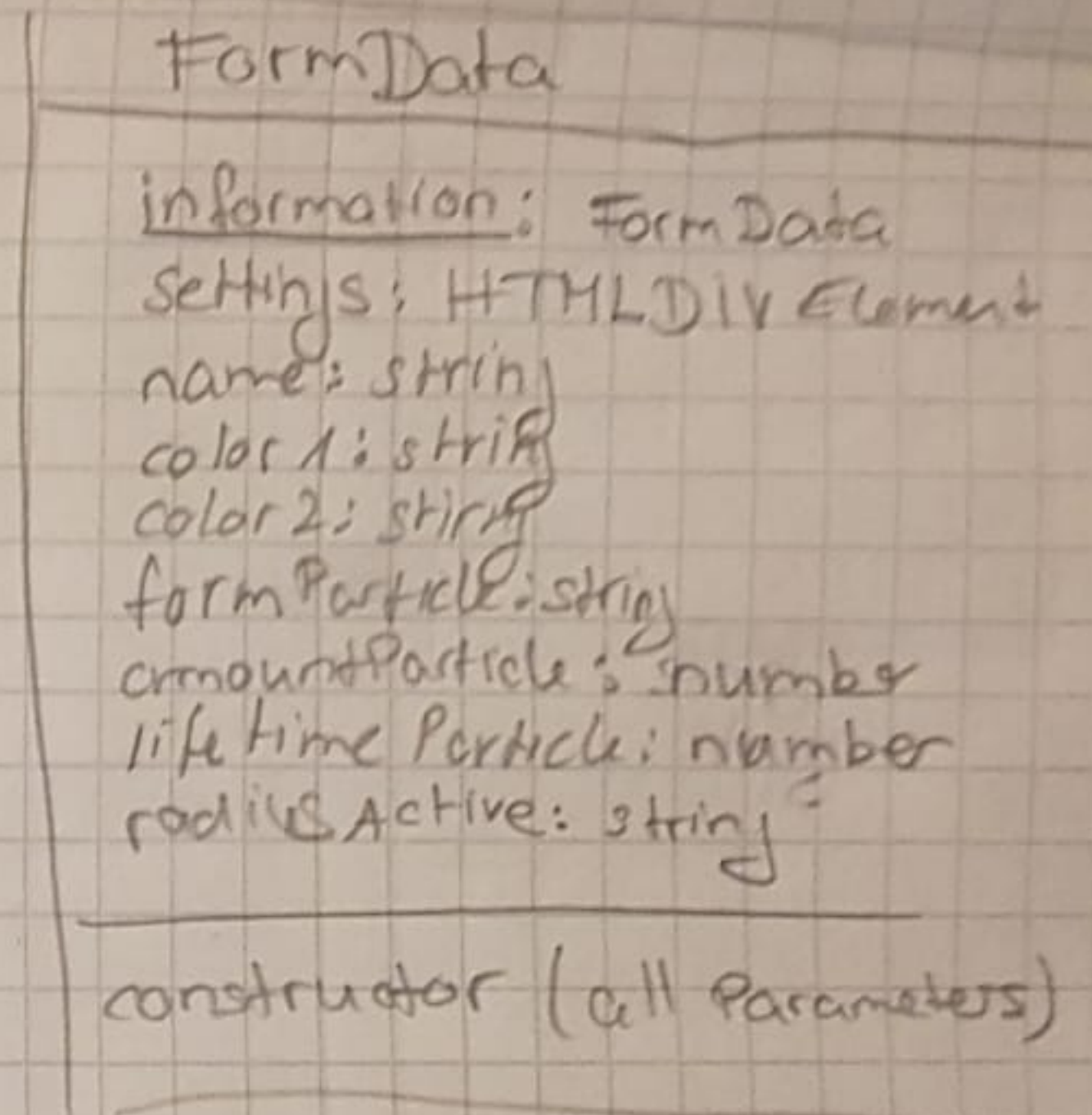
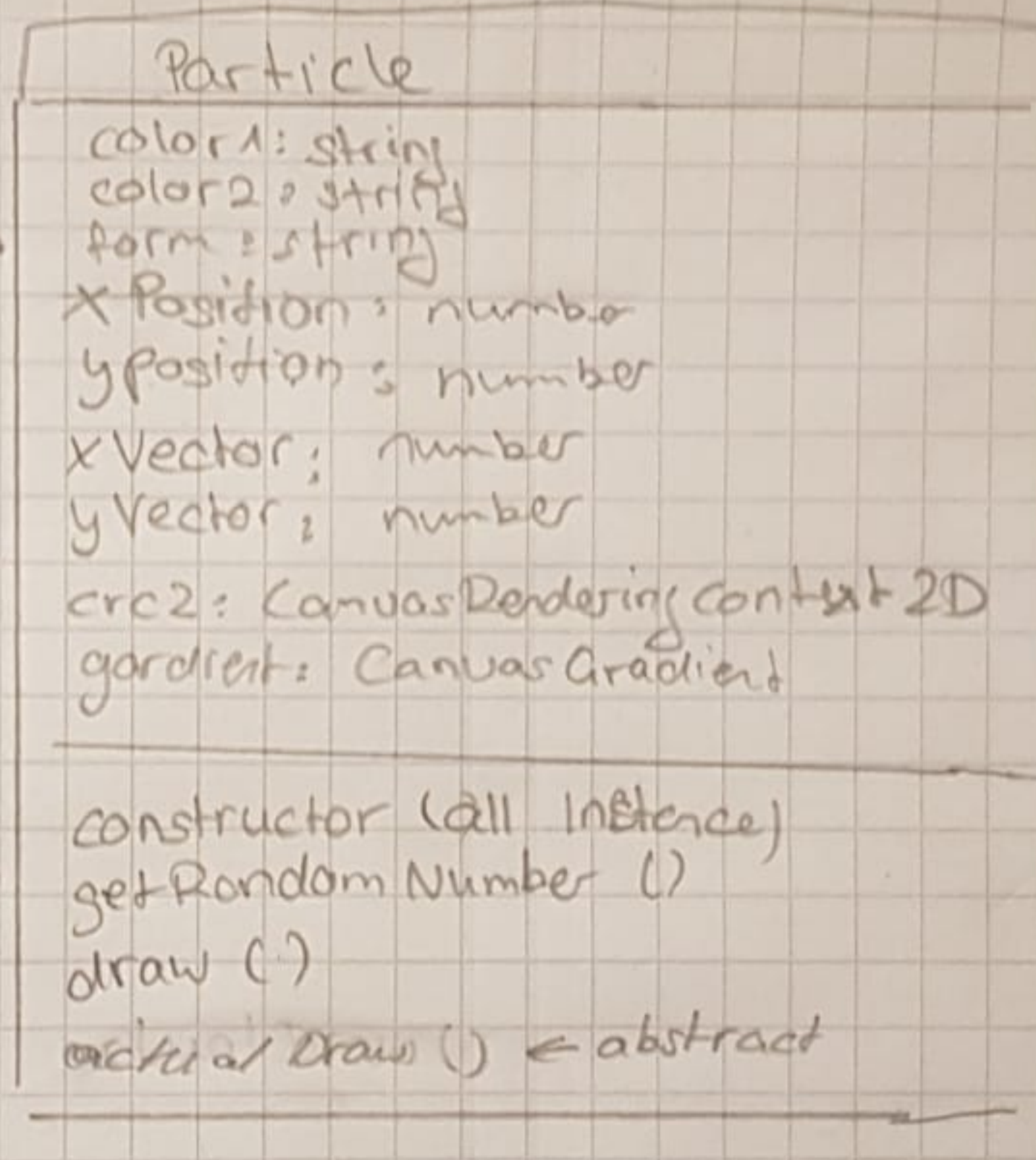
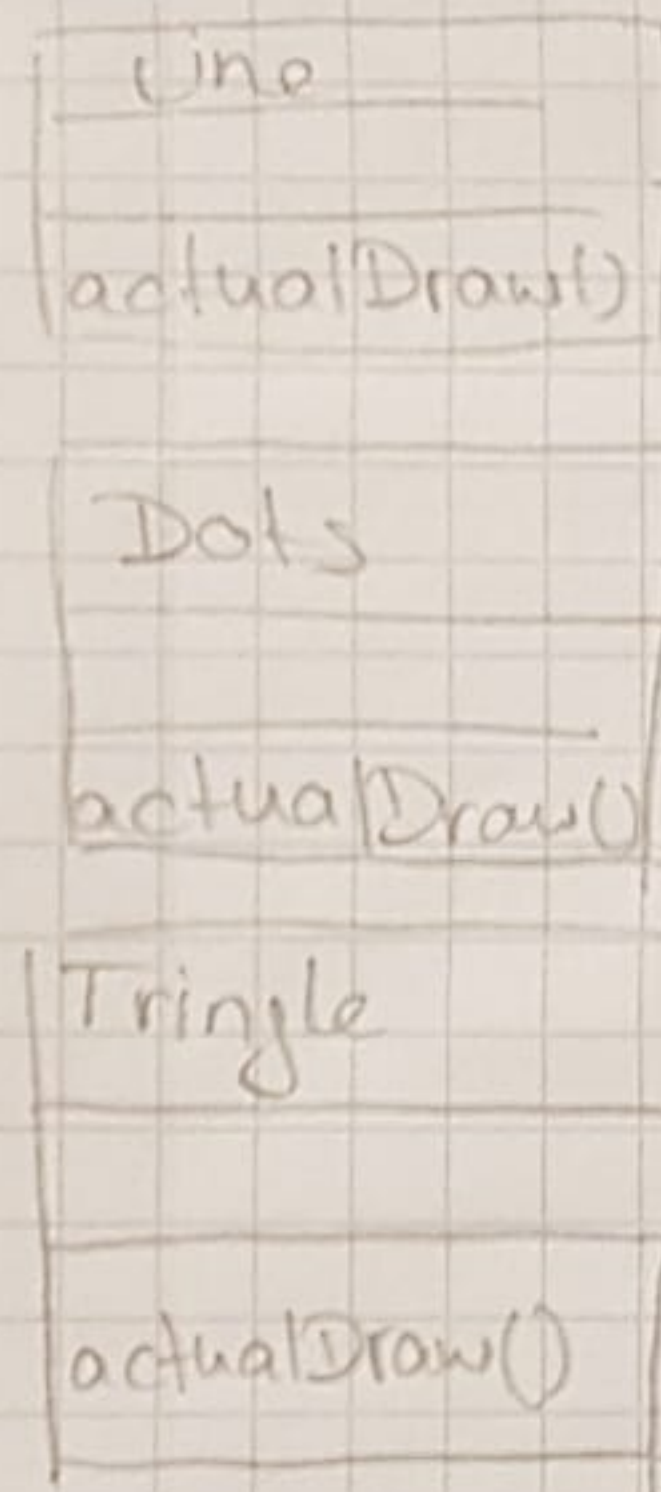

Firework: Datastructure

category	name: string/number	value	data
Settings	name: string	Red Yellow	category
For-sale	color 1: string	#FF0000	value
	color 2: string	#FFFF00	
	formParticle: string	Dots	
	amountParticle: number	300	
	lifetimeParticle: number	3sec	
Explosion	radiusActive	Medium	

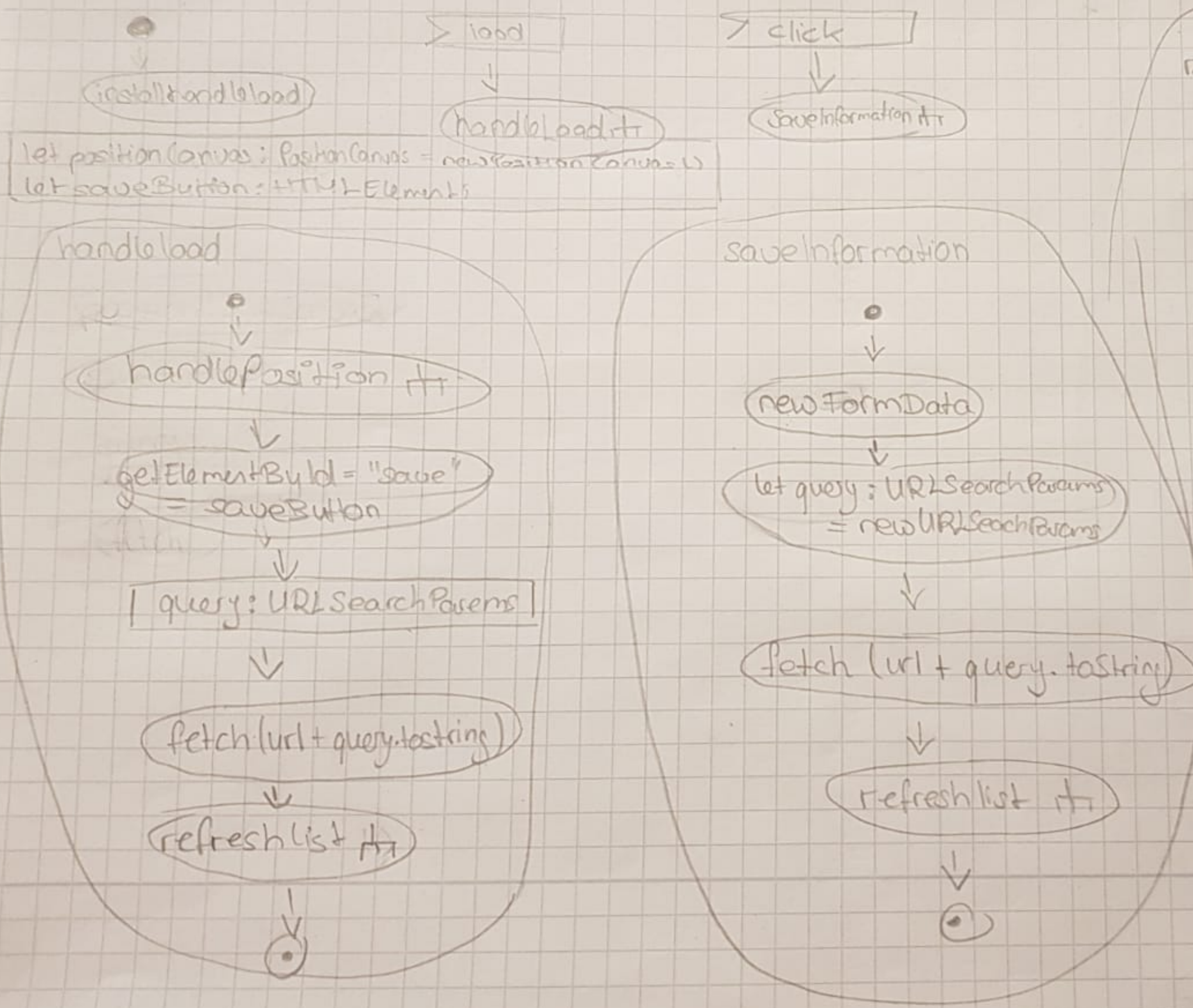
Firework: Swimlane



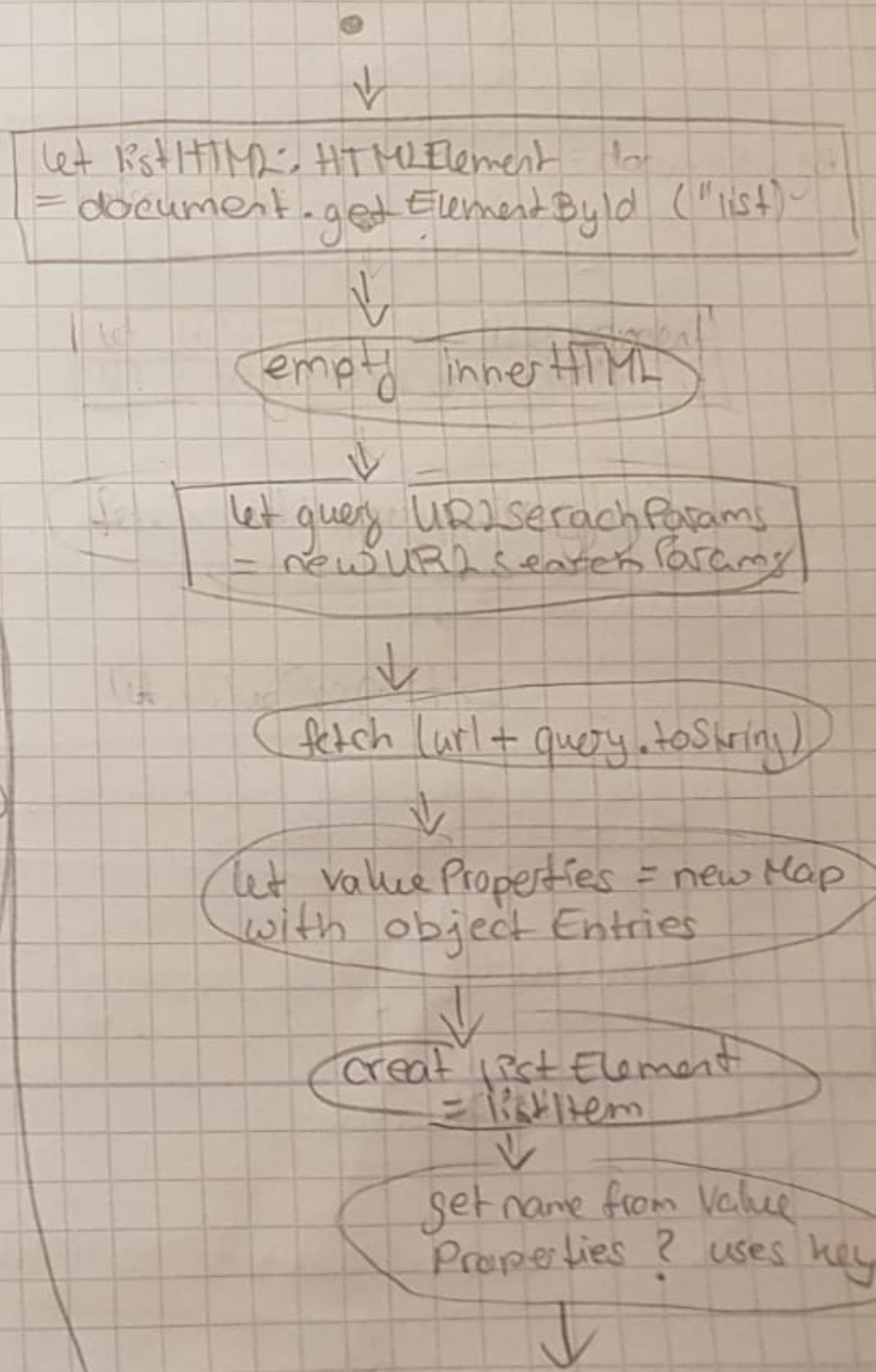
Firework: Class Diagramm



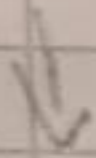
Firework: Activity Diagramm : Main



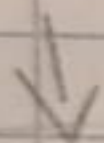
refreshList



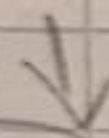
refreshList



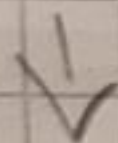
addEventListener("click") on listItem



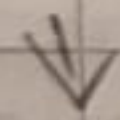
new FormData
with all entries from FormData



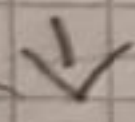
listHTML.appendChild
(listItem)



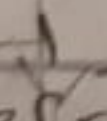
create delete Button with delete
and innerHTML and addEventListener
click on Button



let queryDelete : URLSearchParams
= new URLSearchParams



fetch(url + queryDelete.toString())

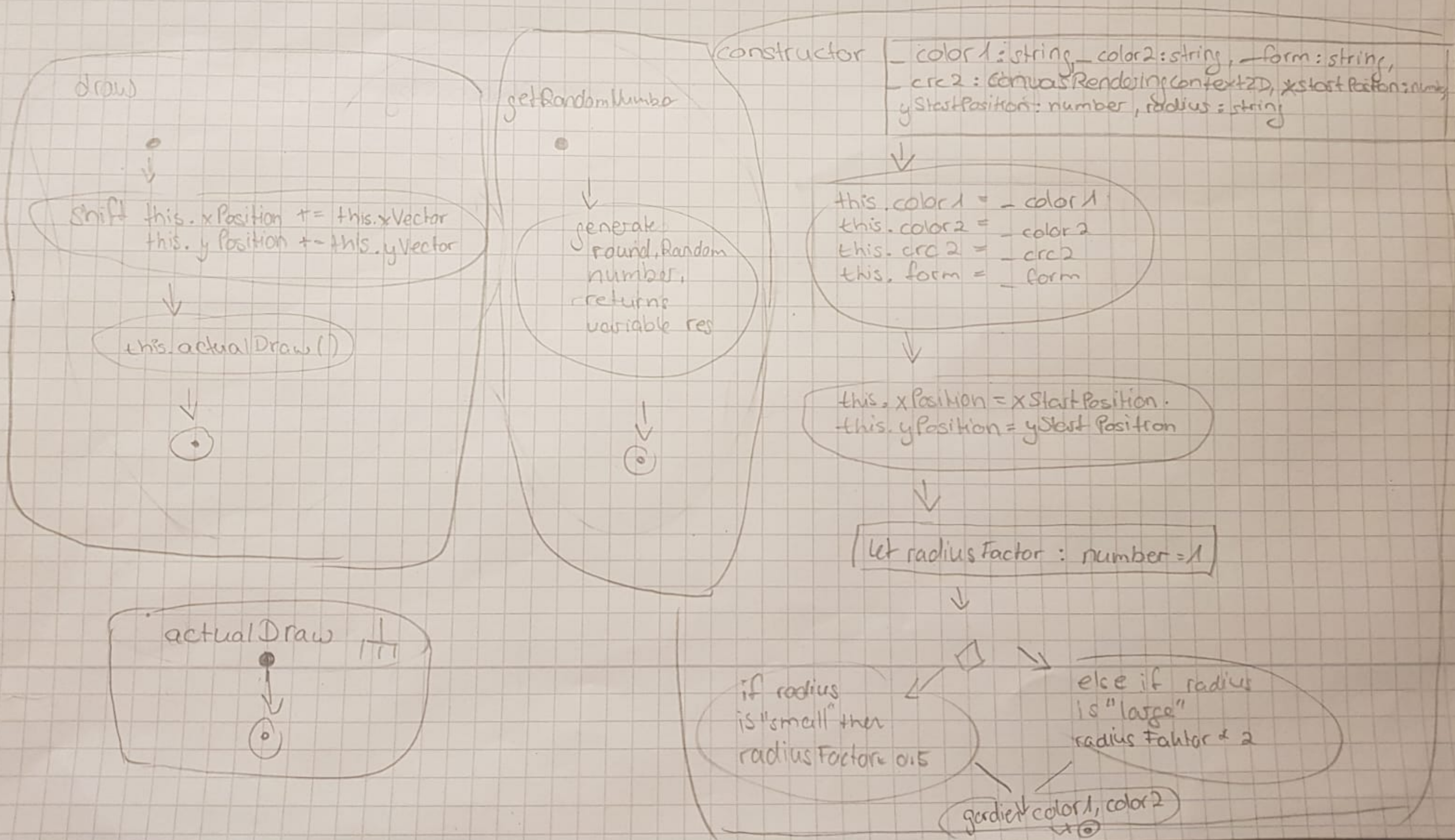


refreshList



append Button to listHTML

Firework: Activity Diagramm : Particle



Firework: Activity Diagram: Dots

actualDraw

let radius: number = 2

this.ctx.beginPath();



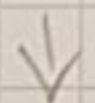
this.ctx.arc(this.xPosition, radius, 0, Math.PI * 2)



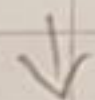
this.ctx.fillStyle = this.gradient



this.ctx.fill()



this.ctx.closePath()



Firework: Activity Diagram: Line

Actual Draw

`this.xPosition += this.xVector`
`this.yPosition += this.yVector`

↓
`this.crc2.beginPath`

↓
`this.crc2.moveTo` `this.xPosition + 10`
`this.yPosition + 10`

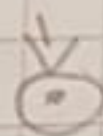
↓
`this.crc2.lineTo` `this.xPosition + 3`
`this.yPosition + 3`

↓
`crc2.closePath()`

↓
`this.crc2.strokeStyle = this.gradient`

↓
`this.crc2.lineWidth = 3`

↓
`this.crc2.stroke()`



Firework: Activity Diagramm : Triangle

actual Draw

this.crc2.beginPath();



this.crc2.moveTo(this.xPosition, this.yPosition)



this.crc2.lineTo(this.xPosition, this.yPosition + 10)



this.crc2.lineTo(this.xPosition + 20, this.yPosition + 2)



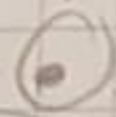
this.crc2.closePath()



this.crc2.fillStyle = this.gradient



this.crc2.fill()



Firework: Activity Diagramm : Explosion

readonly FramesPerSecond: number = 10

initLifetime: number

allParticles: Particle []

crc: CanvasRenderingContext2D

draw ThisRef: Explosion, lifetime: number

if firstElementChild
from canvas -
== ThisRef.crc canvas

else

draw Particle
black

clear canvas

count lifetime
down

if lifetime
bigger than
0

else

clear canvas

draw all Particles

draw () +

setTimeout (1/ExplosionFramesPerSecond)

explode

if we we
don't have
FormData

SendFormData: FormData, x: number, y: number

return

let newCanvas = creatCanvas

style newCanvas
with css elements

append newCanvas
to id = canvas

this. allParticles = []

this. crc = new Canvas. get context (2d)

index < sendFormData

let newElement: Particle

explode

SendFormData, from Particle

case "dots"

new Element = new Dots with DataValues

case "triangle"

new Element = new Triangle with DataValues

case "line"

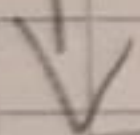
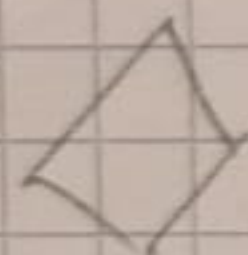
new Element = new Line with DataValues

default

console.log

back to for-loop

for-loop

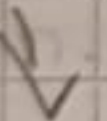


allParticles[Index] = new Element



let lifetime; number = SendFormData.lifetimeParticle * Explosion.FramesPerSecond
this.initLifetime = lifetime

Explosion.draw



Explosion.draw(this, lifetime) return



Firework: Activity Diagramm: PositionCanvas

position: PositionCanvas
canvas: HTML-Element
xPosition: number
yPosition: number

handlePosition

getElementById("canvas")

install mousedown
Event on this.canvas

>mousedown

get Position

get Position

this.xPosition = event.offsetX
this.yPosition = event.offsetY

new Explosion()

Firework : Activity Diagram : FormData

Information : FormData.

Settings: HTMLDivElement = document.querySelector("#settings")

name: string

color 1: string

color 2: string

formParticle: number

amountParticle: number

lifetimeParticle: number

radiusActive: string

- name?: string, - color1?: string, - color2?: string,
- formParticle?: string, - amountParticle?: number,
- lifetimeParticle?: number, - radiusActive?: string

Constructor

If parameter
is passed
in this instance

