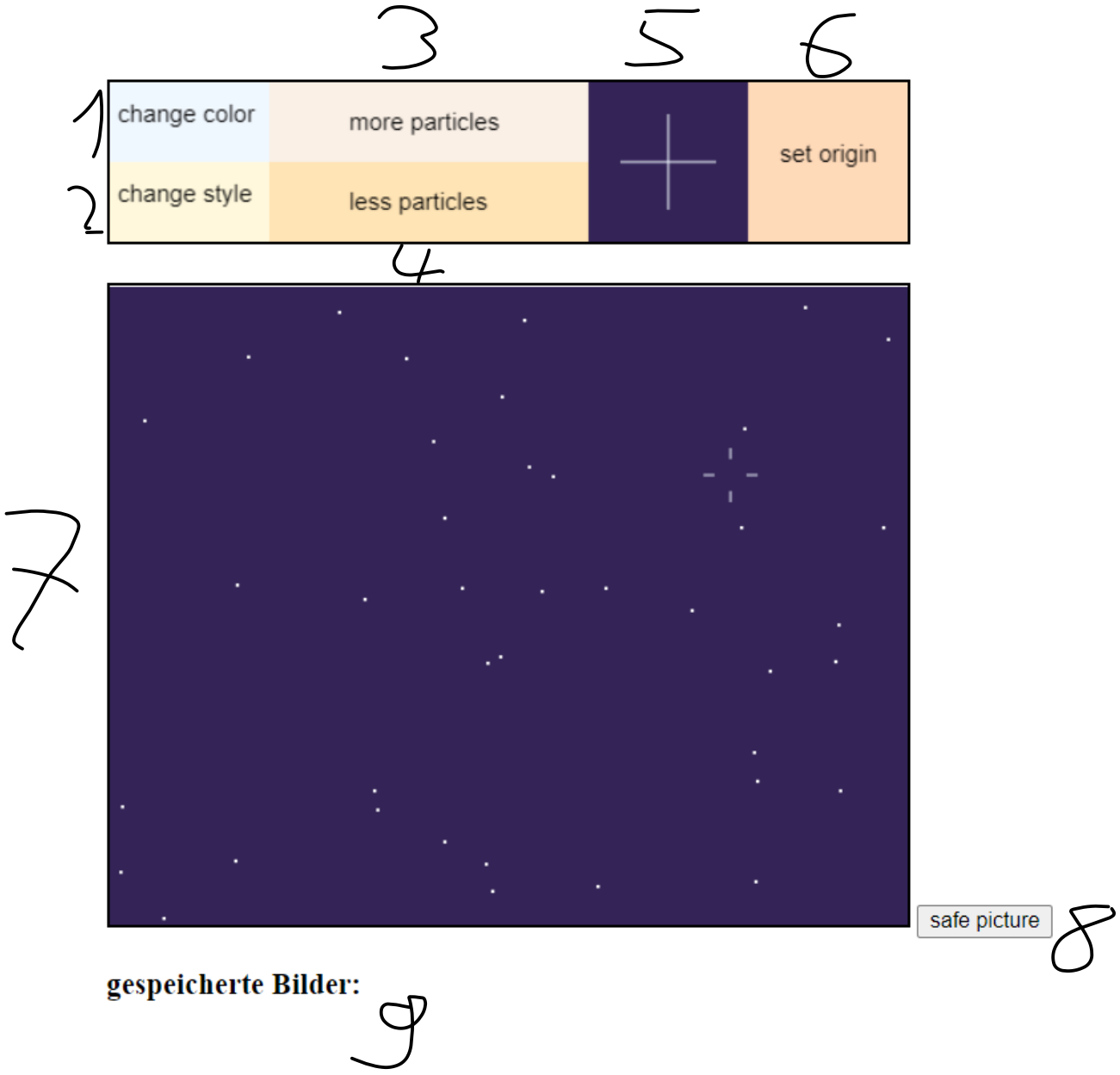
Anleitung:



Stelle dir mit den oben verfügbaren „Buttons“ dein Feuerwerk zusammen.

Bei Punkt 1 kannst du zwischen 6 verfügbaren Farben auswählen.

Bei Punkt 2 kannst du die Art der Explosion auswählen.

Bei Punkt 3 kannst du dem Feuerwerk mehr Partikel hinzufügen.

Bei Punkt 4 kannst du dem Feuerwerk weniger Partikel hinzufügen.

Bei Punkt 5 kannst du dir eine Preview deines Feuerwerks anschauen.

(eine andere Interaktion lässt dieses Feld nicht zu)

Nach drücken von Punkt 6 kannst du das erstellte Feuerwerk auf dem Canvas platzieren

Bei Punkt 7 ist das Canvas auf dem man das Feuerwerk platzieren kann.

Bei Punkt 8 kannst du dir dein Bild in der Datenbank abspeichern.

Bei Punkt 9 werden die Gespeicherten Bilder als Buttons angezeigt und durch ein Click darauf können sie wieder hergestellt und weiterbearbeitet werden.

Instalation Heroku und MongoDB:

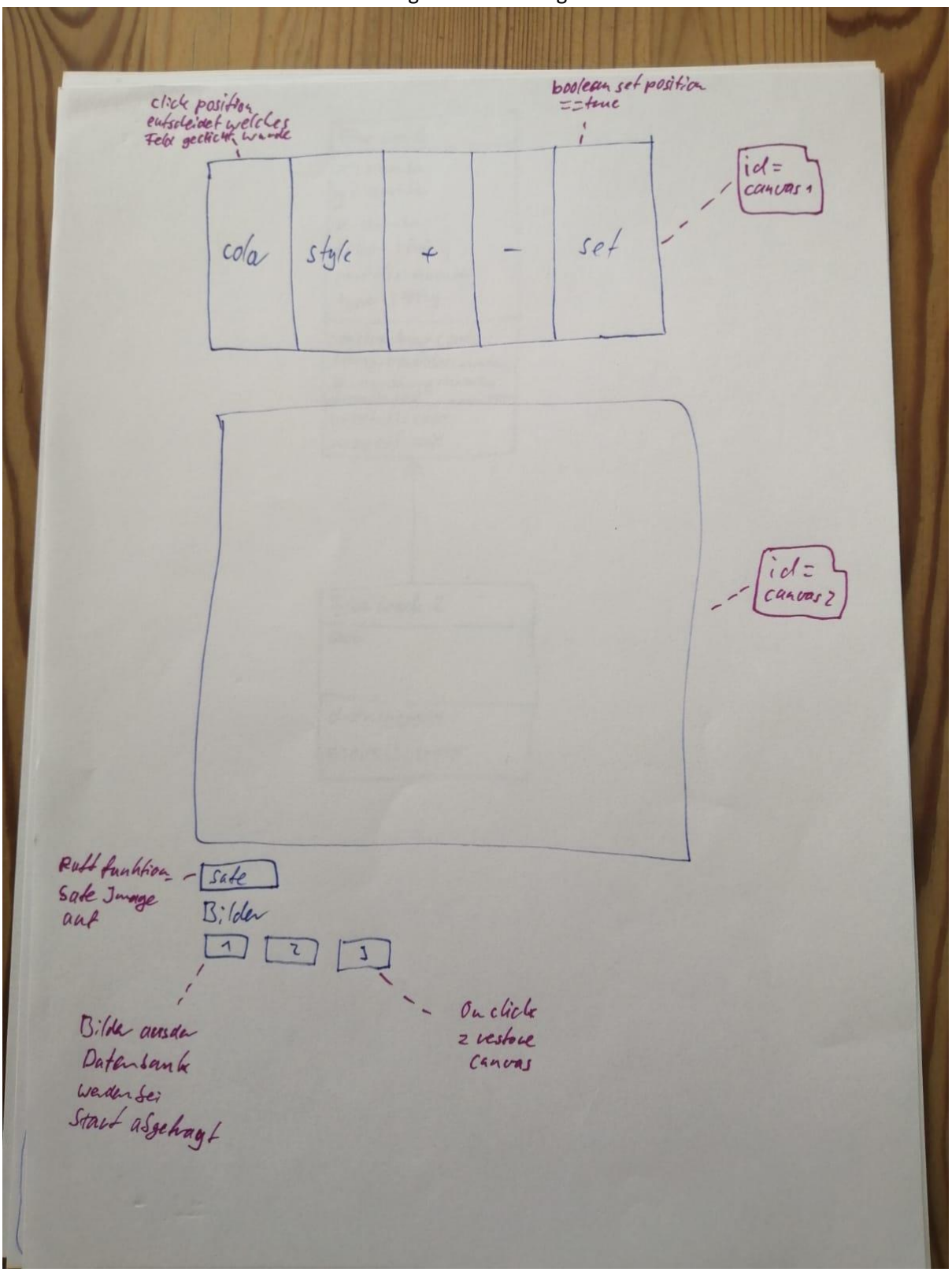
Heroku:

1. Bevor Heroku installiert wird, sollte Node mit dem Befehl npm install @types node auf der obersten Ebene des Projektes installiert werden.
2. Lege ein Nutzerkonto auf Heroku an oder logge dich in einem bereits existierenden Account ein. Wichtig: Die primary Language muss node.js sein.
3. Erstelle eine App und verbinde diese mit dem zutreffenden Github-Repository.
4. In der package.json Datei, die bei der Installation von node kreiert wurde, unter „start“ den relativen Pfad zur server.js datei angeben.
5. Ob alles geklappt hat testest du mit dem Befehl npm start, ein „build succeeded“ sollte im Heroku-Logs auftauchen.
6. Im Client muss die URL-Variable angepasst werden. Hierzu muss der Pfad zur App an die Stelle der deklarierten Variable URL in der main.ts eingesetzt werden. Zu diesem Pfad kommst du unter „Open App“.

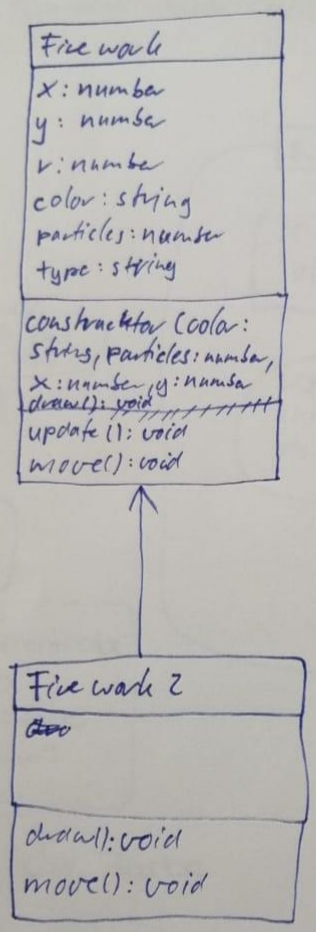
Mongo DB Initialisierung:

7. Installiere zuallererst MongoDB in Node. Gebe hierzu im Terminal auf der obersten Ebene den Befehl npm install @types MongoDB sowie npm install MongoDB ein.
8. Füge in den server.ts datei nun das Mongo-Modul mit dem Befehl import * as Mongo from „MongoDB“;
9. Erstelle einen neuen Account oder nimm ein bereits erstelltes Benutzerkonto
10. Lege ein neues Cluster an, mache dies mit der IP-Adresse 0.0.0.0/0 von überall aus sichtbar
11. Lege eine neue Collection an, in die du eine Datenbank (in meinem Fall „eia“) und eine Collection (bei mir „score“) erstellst.
12. Um Mongo mit deinem Server verbinden zu können musst du den Link, den du bei Connect -> Connect your Application finden kannst, in die server.ts Datei einbinden (in meinem Fall: let serverPage: string = "https://eia-endabgabe.herokuapp.com/";).

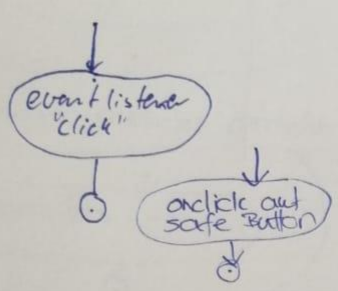
Diagramme Planung:

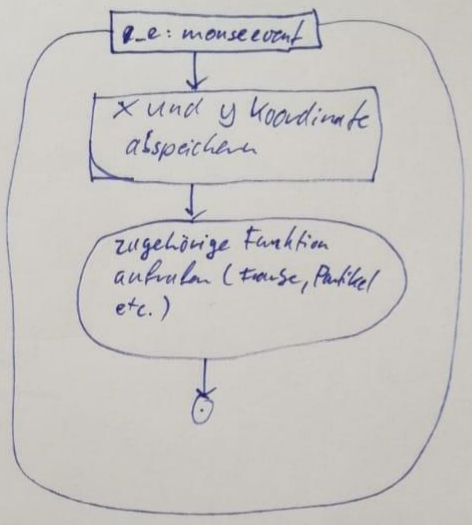


Klassendiagram



1. Funktion use click canvas

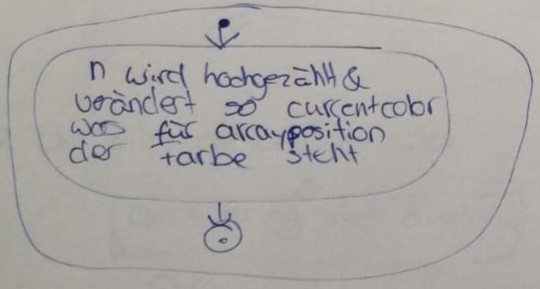




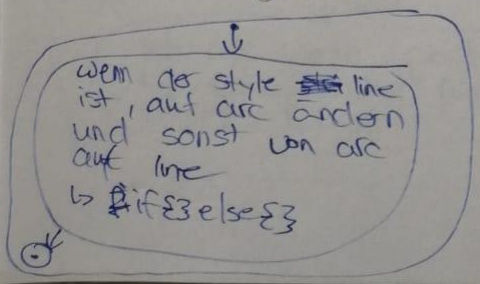

```
let colors: string[]
= alle Farben
```

```
let pickedColor: string = currentcolor
let particles: number = 10
let setPoint: boolean = false
let style: string = "line"
```

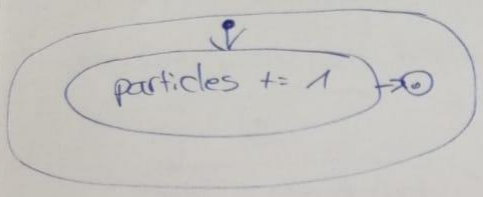
2. Funktion ändern der Farbe



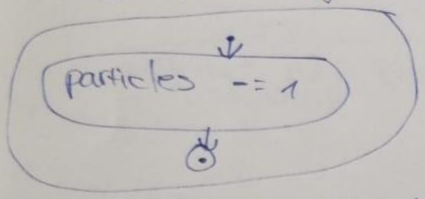
3. Style wird geändert



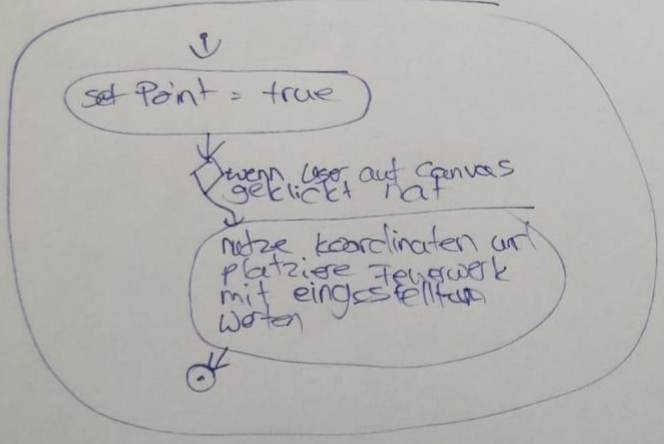
4. mehr particles geklickt



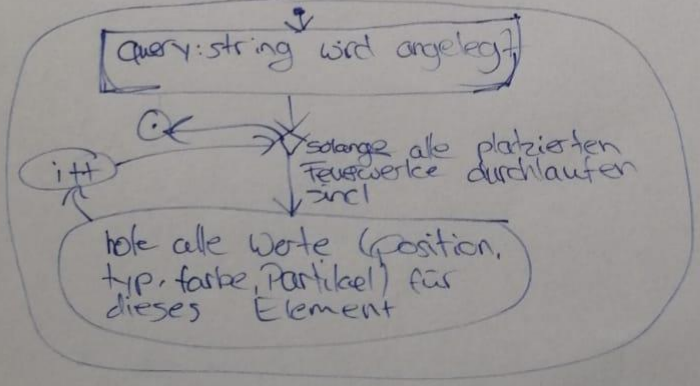
5. weniger particles geklickt



6. Wie Feuerwerk platziert wird

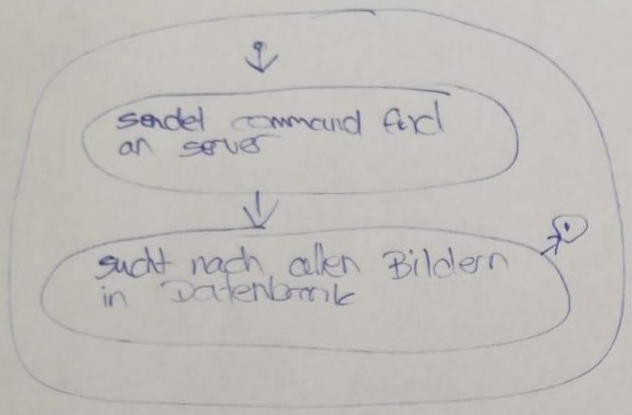


7. Wie Bild gespeichert wird

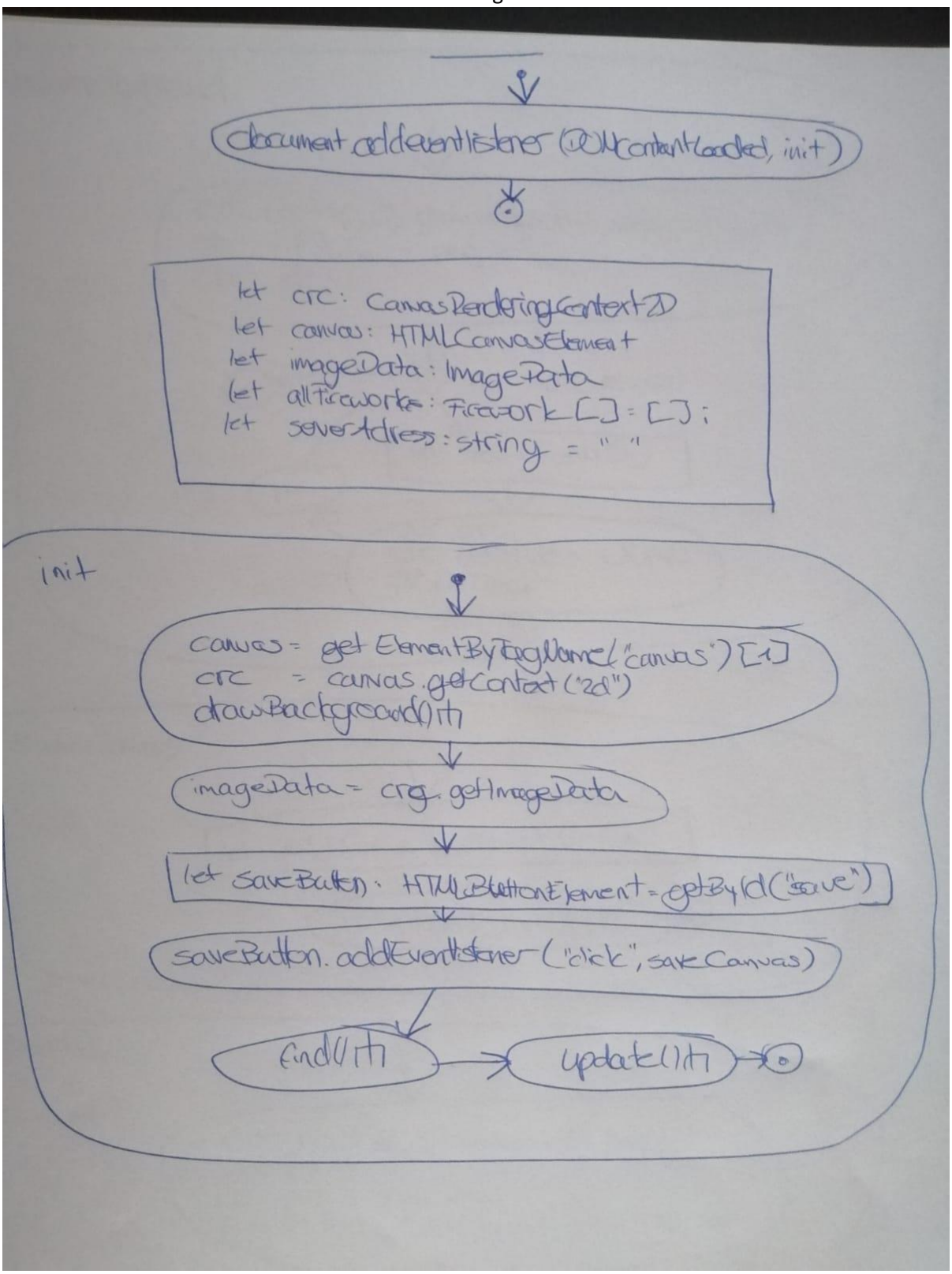


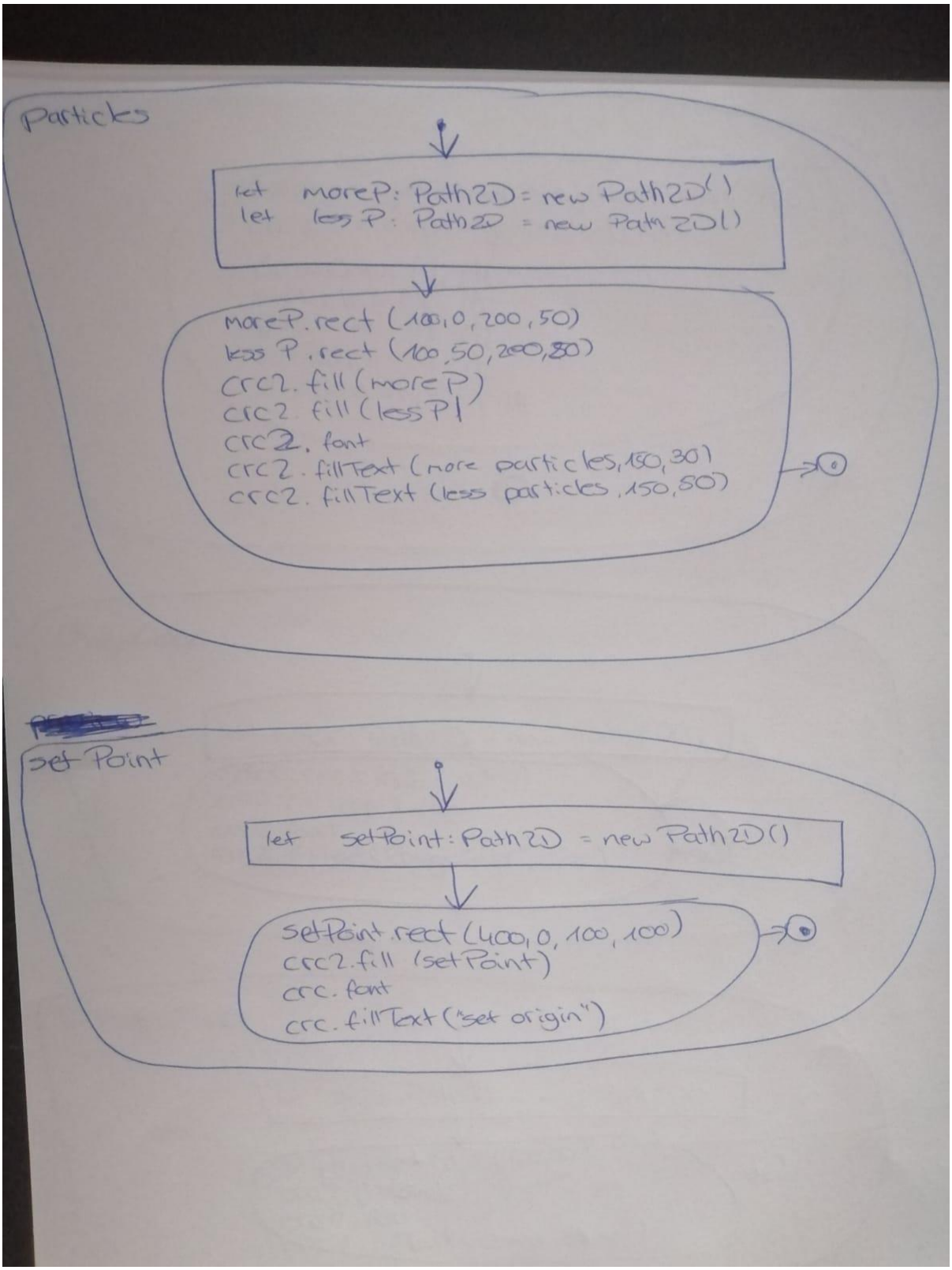
8. Wie vorhandene Bilder aus Datenbank abgefragt werden

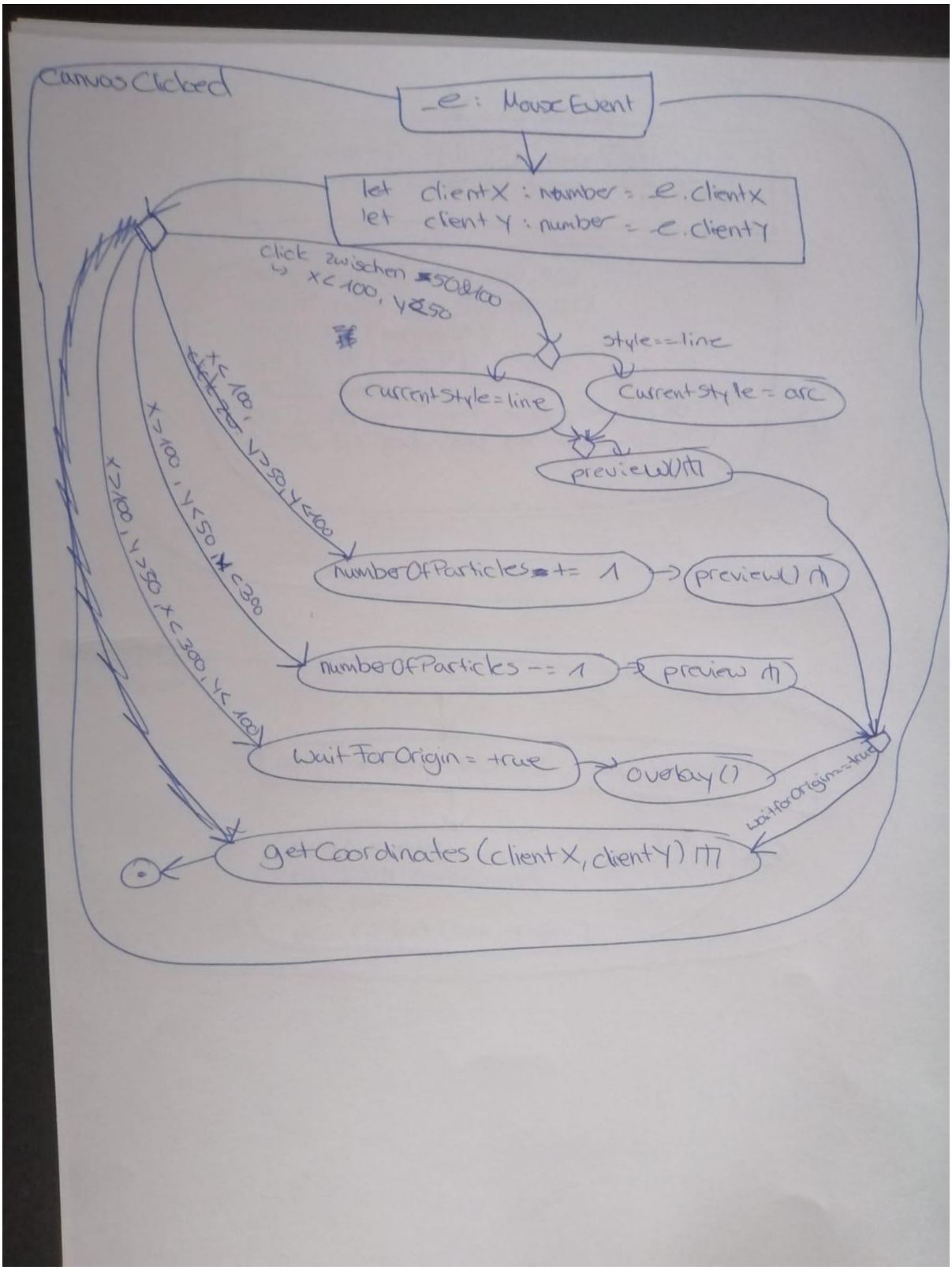
- Funktion wird nach Programmstart in init aufgerufen

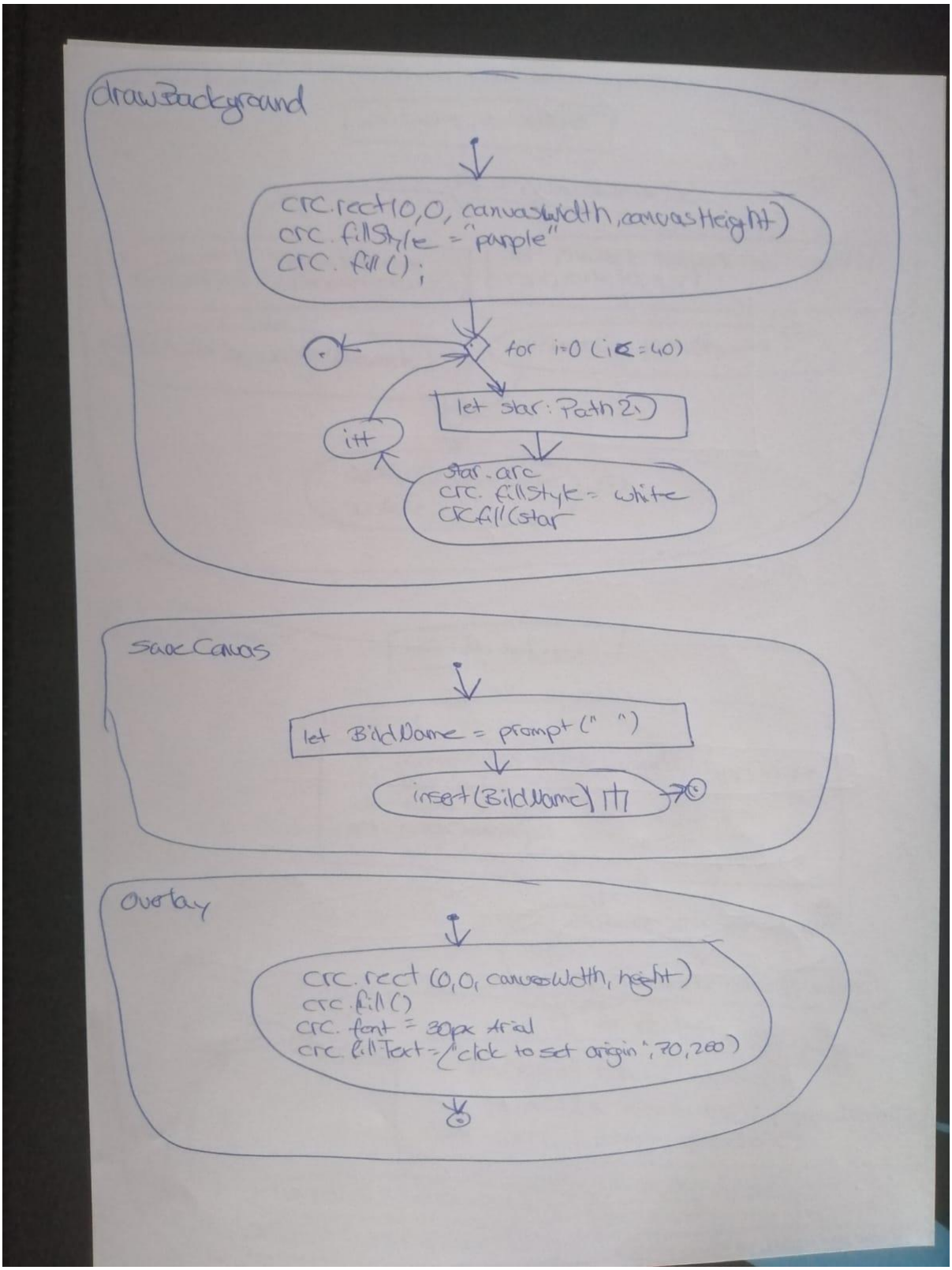


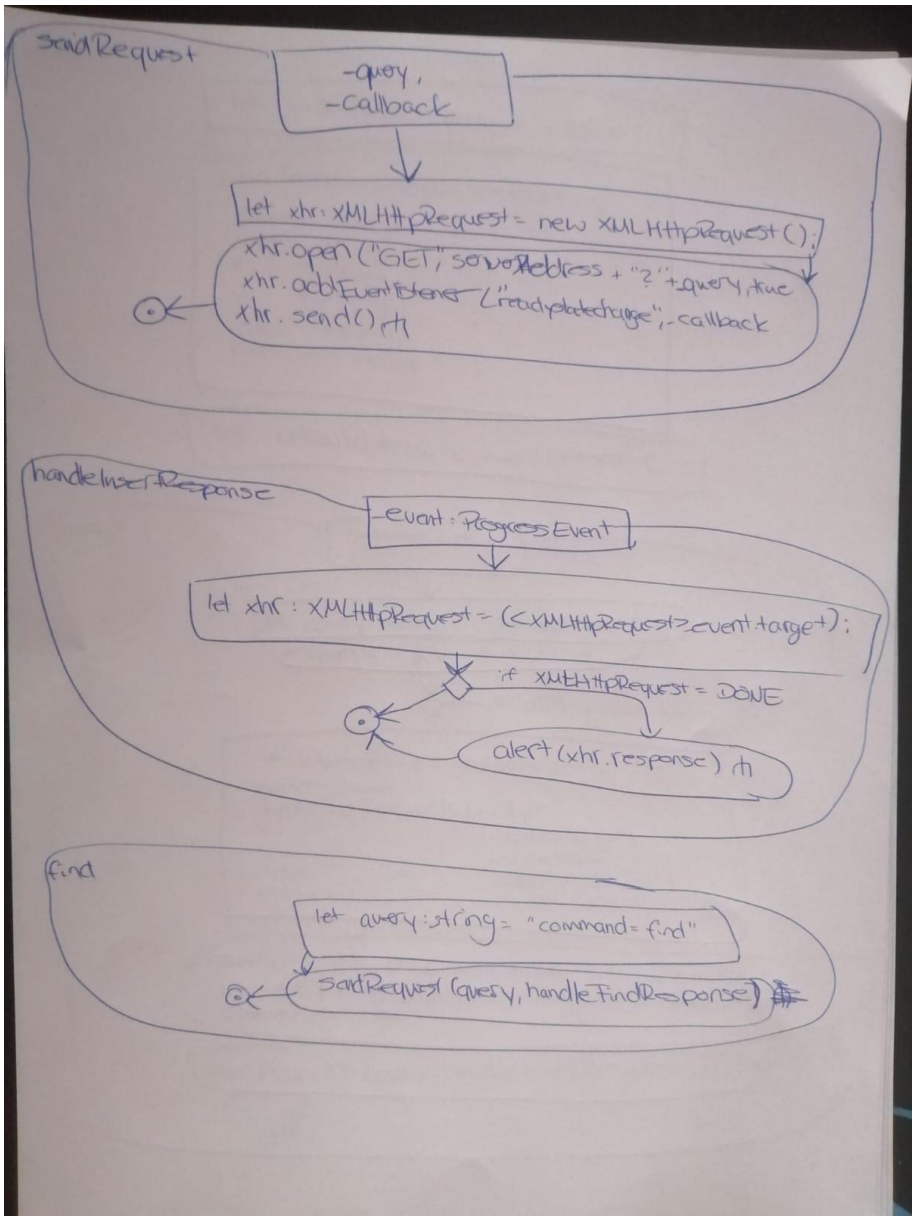
Aktivitätsdiagramme:

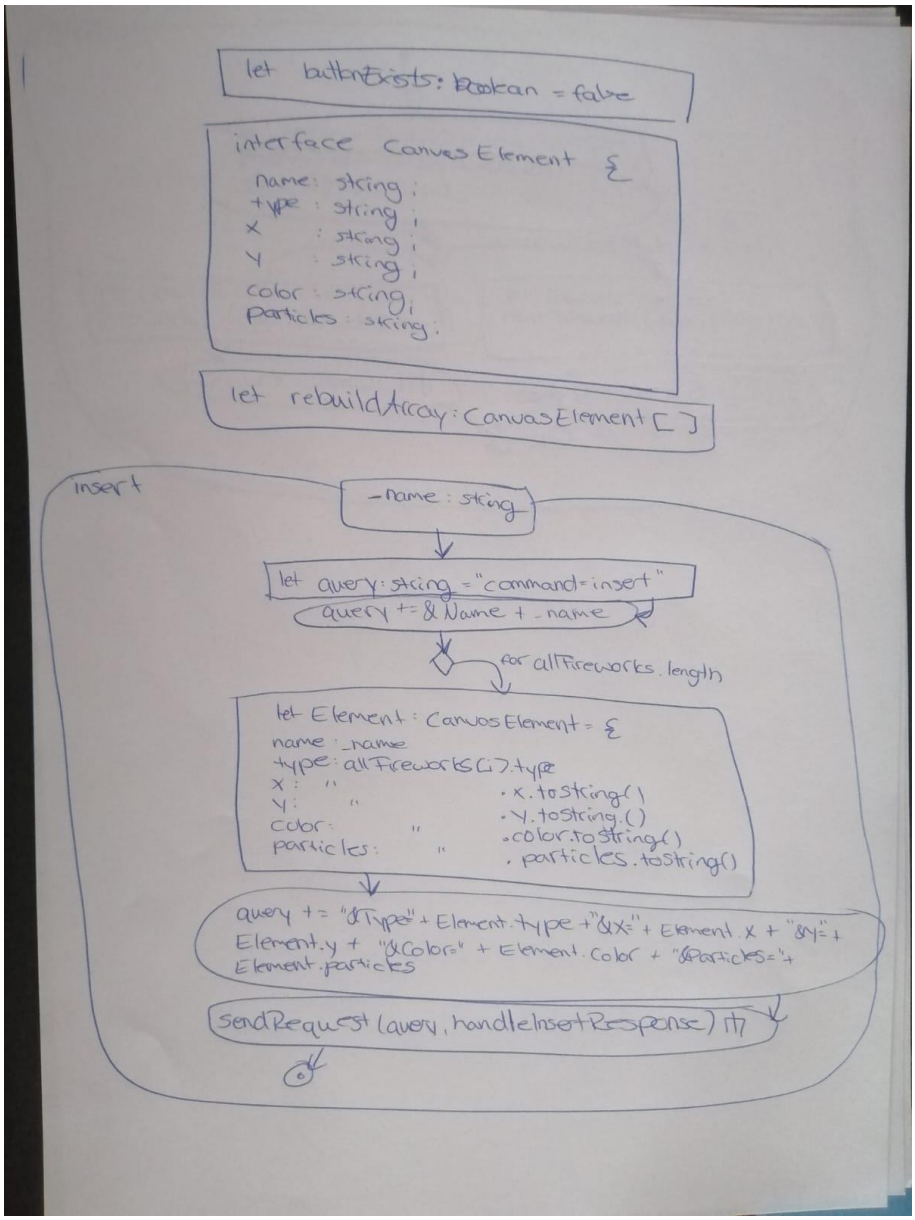


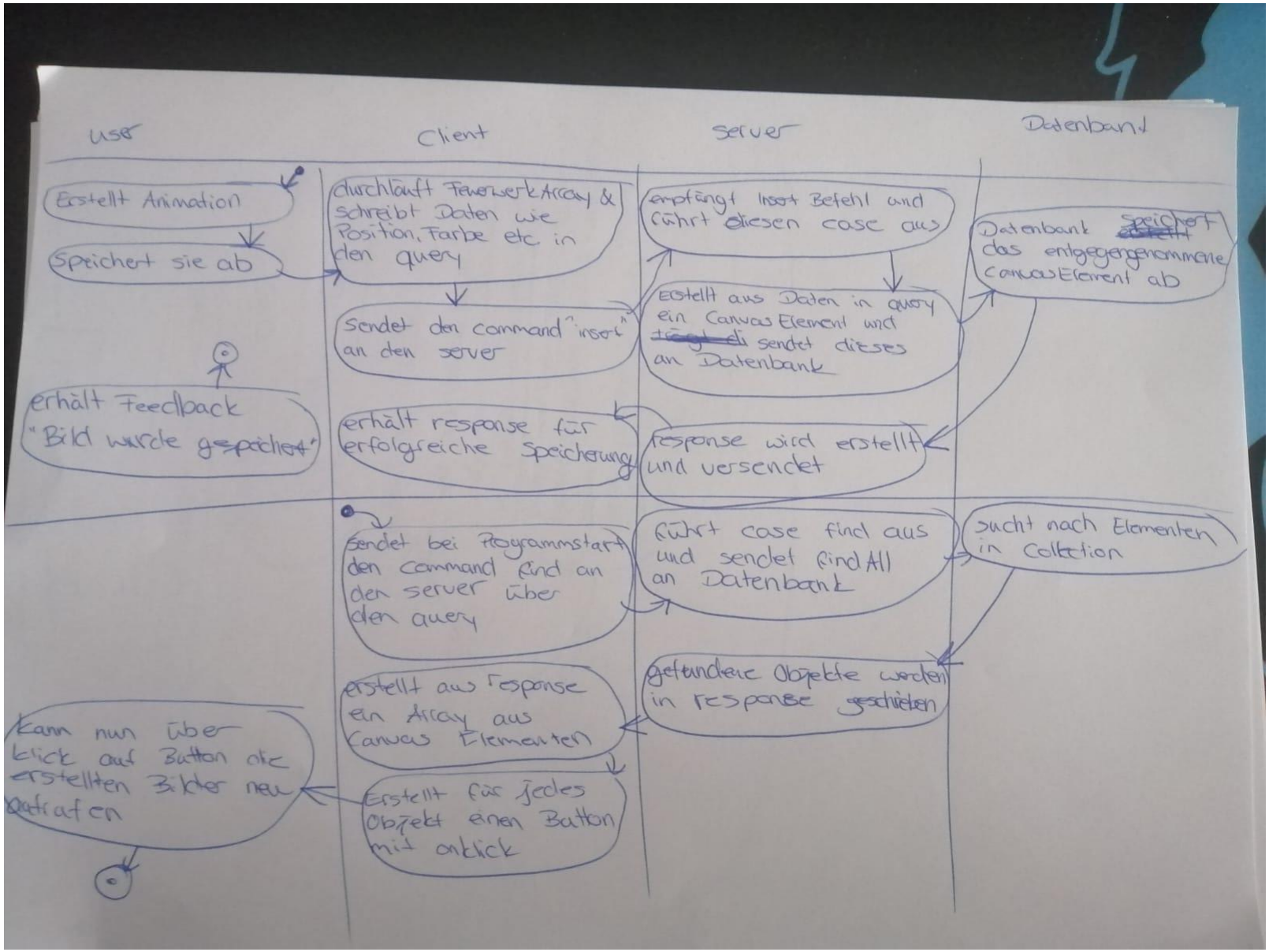

Particles
let moreP: Path2D = new Path2D()
let lessP: Path2D = new Path2D()
moreP.rect(100,0,200,50)
lessP.rect(100,50,200,80)
crc2.fill(moreP)
crc2.fill(lessP)
crc2.font
crc2.fillText(more particles,150,30)
crc2.fillText(less particles,150,50)
setPoint
let setPoint: Path2D = new Path2D()
setPoint.rect(400,0,100,100)
crc2.fill(setPoint)
crc.font
crc.fillText("set origin")

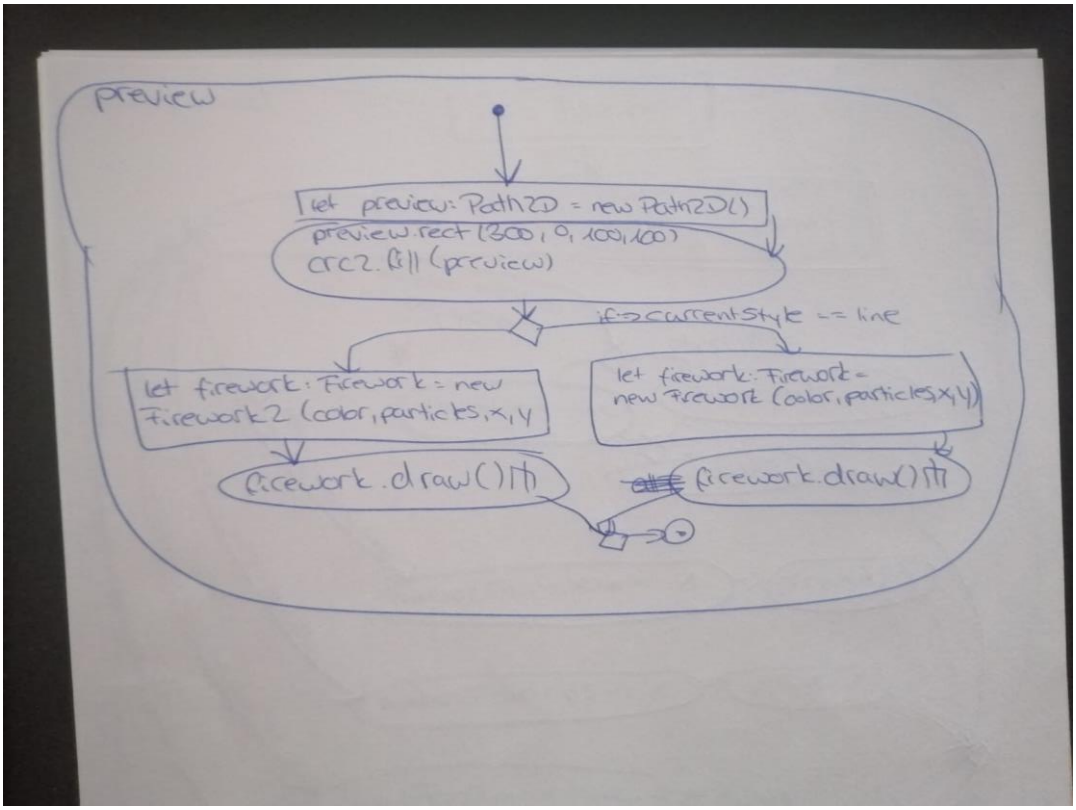

canvasClicked
_e: MouseEvent
let clientX: number = e.clientX
let clientY: number = e.clientY
Click zwischen 50&100
↳ x<100, y≥50
style==line
currentStyle=line
currentStyle=arc
previewAll()
x<100, y≥50, y<100
x>100, y<50, x<300
x>100, y>50, x<300, y<100
numberOfParticles += 1
preview()
numberOfParticles -= 1
preview()
waitForOrigin = true
overlay()
waitForOrigin==true
getCoordinates(clientX, clientY)

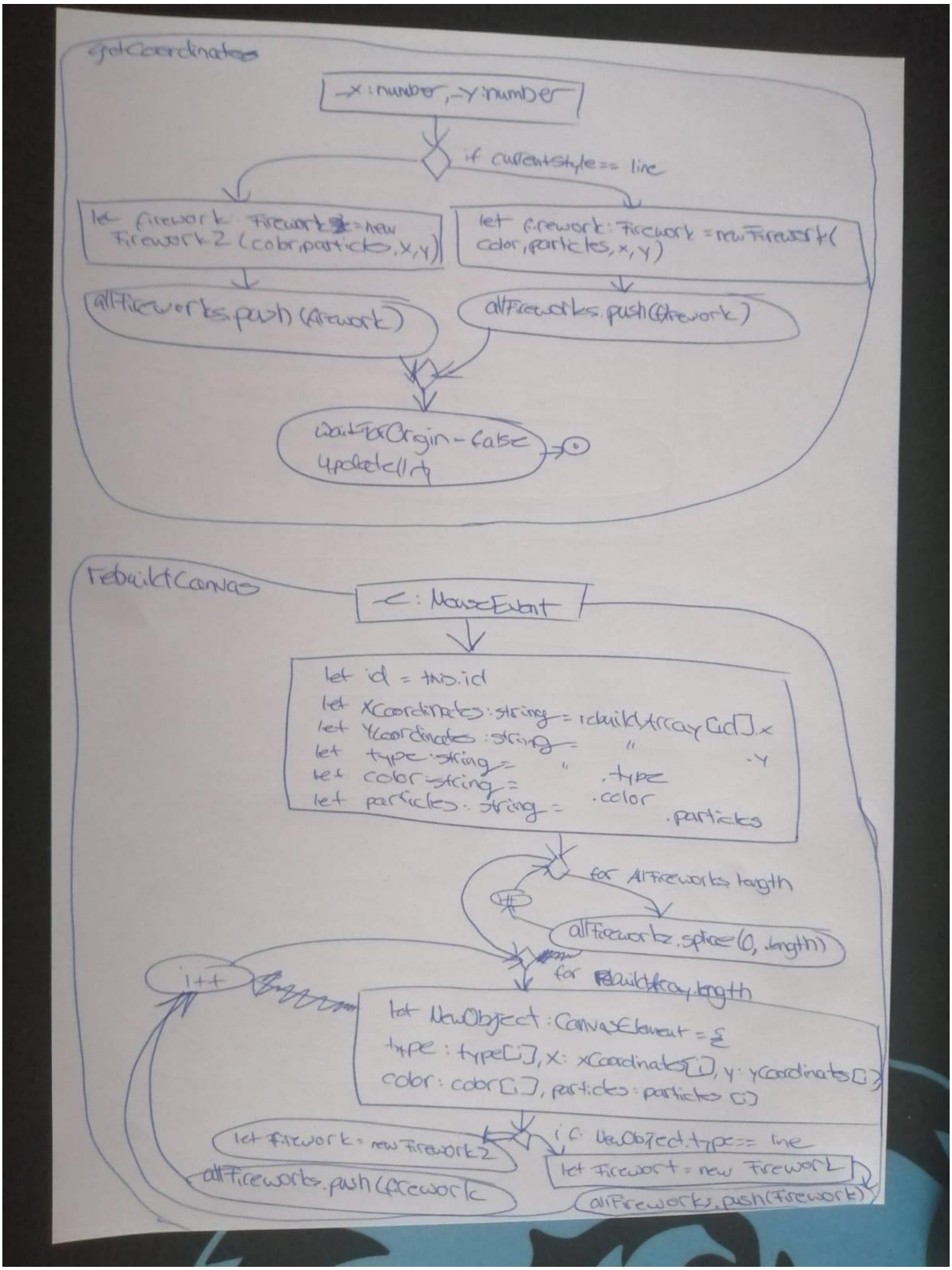

drawBackground
ctx.rect(0,0, canvasWidth, canvasHeight)
ctx.fillStyle = "purple"
ctx.fill();
for i=0 (i<=40)
let star: Path2D
i++
star.arc
ctx.fillStyle = white
ctx.fill(star
saveCanvas
let BildName = prompt(" ")
insert(BildName)
Overlay
ctx.rect(0,0, canvasWidth, height)
ctx.fill()
ctx.font = 30px Arial
ctx.fillText("click to set origin", 70,200)

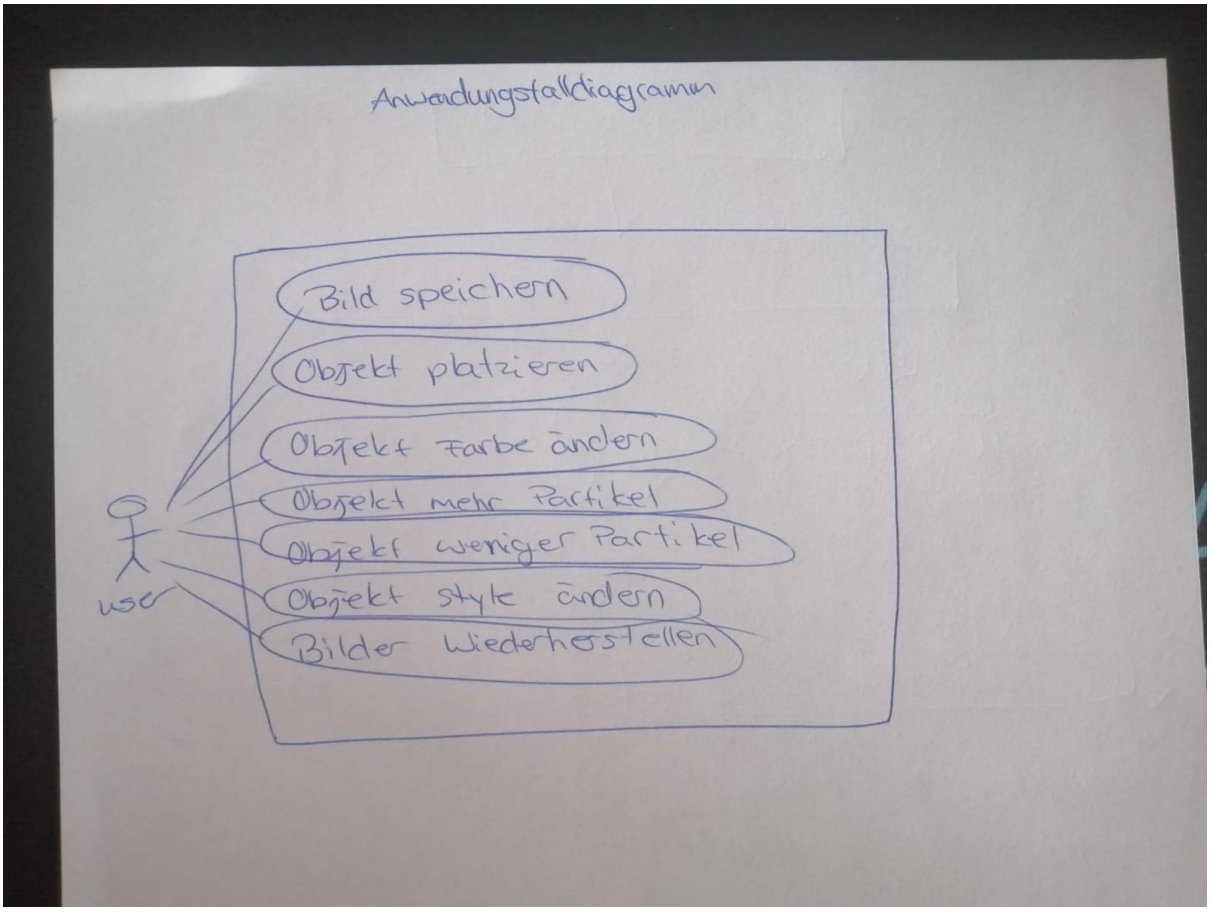

sendRequest
-query,
-callback
let xhr: XMLHttpRequest = new XMLHttpRequest();
xhr.open("GET", serverAddress + "?" + query, true
xhr.addEventListener("readystatechange", -callback
xhr.send();
handleUserResponse
-event: ProgressEvent
let xhr: XMLHttpRequest = (<XMLHttpRequest>event.target);
if XMLHttpRequest = DONE
alert(xhr.response)
find
let query: string = "command=find"
sendRequest(query, handleFindResponse)

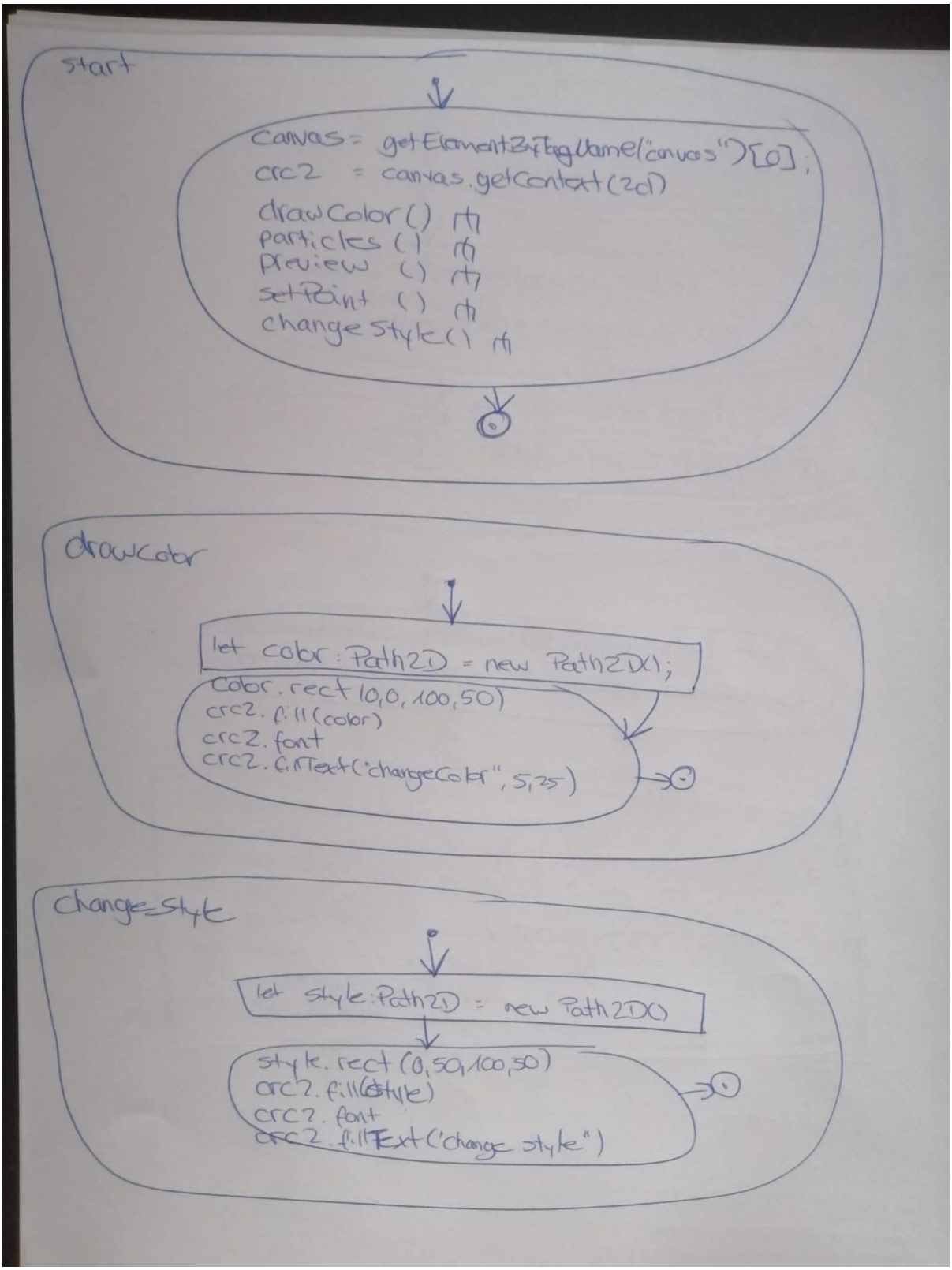

let buttonExists: boolean = false
interface CanvasElement {
name: string;
type: string;
x: string;
y: string;
color: string;
particles: string;
let rebuildArray: CanvasElement[]
insert
-name: string
let query: string = "command=insert"
query += &Name + _name
for allFireworks.length
let Element: CanvasElement = {
name: _name
type: allFireworks[i].type
x: ".x.toString()
y: ".y.toString()
color: ".color.toString()
particles: ".particles.toString()
query += "&Type=" + Element.type + "&X=" + Element.x + "&Y=" + Element.y + "&Color=" + Element.color + "&Particles=" + Element.particles
sendRequest(query, handleInsertResponse)

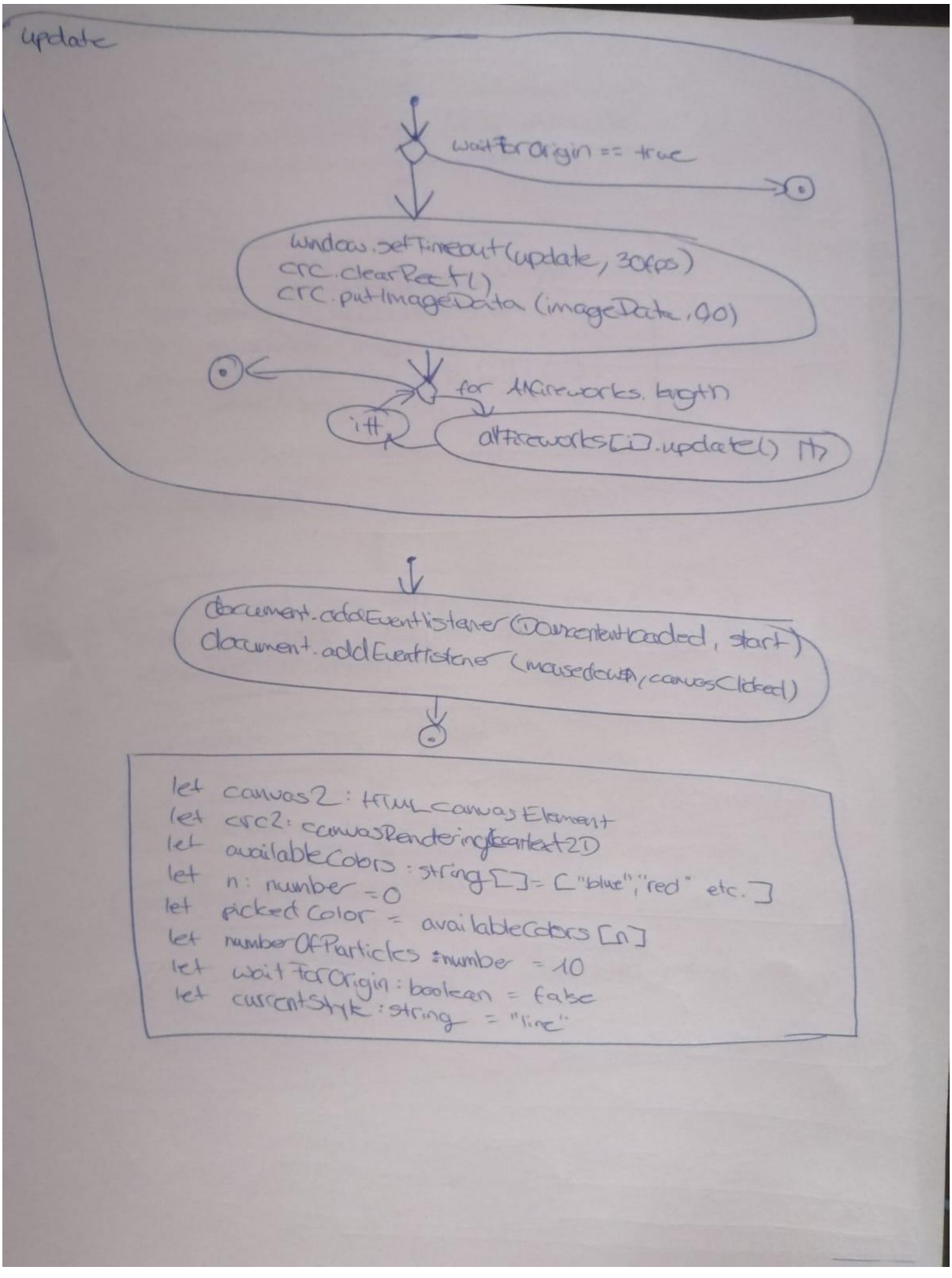

User
Client
Server
Datenbank
Erstellt Animation
Speichert sie ab
durchläuft Feuerwerk Array & schreibt Daten wie Position, Farbe etc. in den query
empfängt Insert Befehl und führt diesen case aus
Datenbank speichert das entgegengenommene CanvasElement ab
Sendet den command "insert" an den server
Erstellt aus Daten in query ein CanvasElement und sendet dieses an Datenbank
erhält Feedback "Bild wurde gespeichert"
erhält response für erfolgreiche Speicherung
response wird erstellt und versendet
Sendet bei Programmstart den command find an den server über den query
führt case find aus und sendet findAll an Datenbank
sucht nach Elementen in Collection
gefundene Objekte werden in response geschrieben
erstellt aus response ein Array aus CanvasElementen
Kann nun über Klick auf Button die erstellten Bilder neu laden
Erstellt für jedes Objekt einen Button mit onClick


preview
let preview: Path2D = new Path2D()
preview.rect(300, 0, 100, 100)
ctx2.fill(preview)
if->currentStyle == line
let firework: Firework = new Firework2 (color, particles, x, y
let firework: Firework = new Firework (color, particles, x, y)
firework.draw()
firework.draw()


getCoordinates
_x:number, _y:number
if currentStyle == line
let firework: Firework = new Firework2(color,particles,x,y)
let firework: Firework = newFirework(color,particles,x,y)
allFireworks.push(firework)
allFireworks.push(firework)
waitForOrigin = false
updateAll()
rebuildCanvas
e: MouseEvent
let id = this.id
let xCoordinates: string = rebuildArray[id].x
let yCoordinates: string = "" .y
let type: string = "" .type
let color: string = .color
let particles: string = .particles
for AllFireworks.length
allFireworks.splice(0, length)
i++
for rebuildArray.length
let NewObject: CanvasElement = {
type: type[i], x: xCoordinates[i], y: yCoordinates[i]
color: color[i], particles: particles[i]
let firework = newFirework2
allFireworks.push(firework)
if NewObject.type == line
let firework = new Firework
allFireworks.push(firework)


Anwendungsfalldiagramm
Bild speichern
Objekt platzieren
Objekt Farbe ändern
Objekt mehr Partikel
Objekt weniger Partikel
Objekt style ändern
Bilder wiederherstellen
User


start
canvas = getElementsByTagName("canvas")[0];
crc2 = canvas.getContext(2d)
drawColor() rh
particles() rh
Preview () rh
setPoint () rh
changeStyle() rh
drawcolor
let color: Path2D = new Path2D();
Color.rect(10,0,100,50)
crc2.fill(color)
crc2.font
crc2.fillText("changeColor", 5,25)
changeStyle
let style:Path2D = new Path2D()
style.rect(0,50,100,50)
crc2.fill(style)
crc2.font
crc2.fillText("change style")


update
waitForOrigin == true
window.setTimeout(update, 30fps)
crc.clearRect()
crc.putImageData(imageData, 0,0)
for allFireworks.length
if
allFireworks[i].update()
document.addEventListener(DOMContentLoaded, start)
document.addEventListener(mousedown, canvasClicked)
let canvas2: HTMLCanvasElement
let crc2: canvasRenderingContext2D
let availableColors: string[] = ["blue", "red" etc.]
let n: number = 0
let pickedColor = availableColors[n]
let numberOfParticles: number = 10
let waitForOrigin: boolean = false
let currentStyle: string = "line"