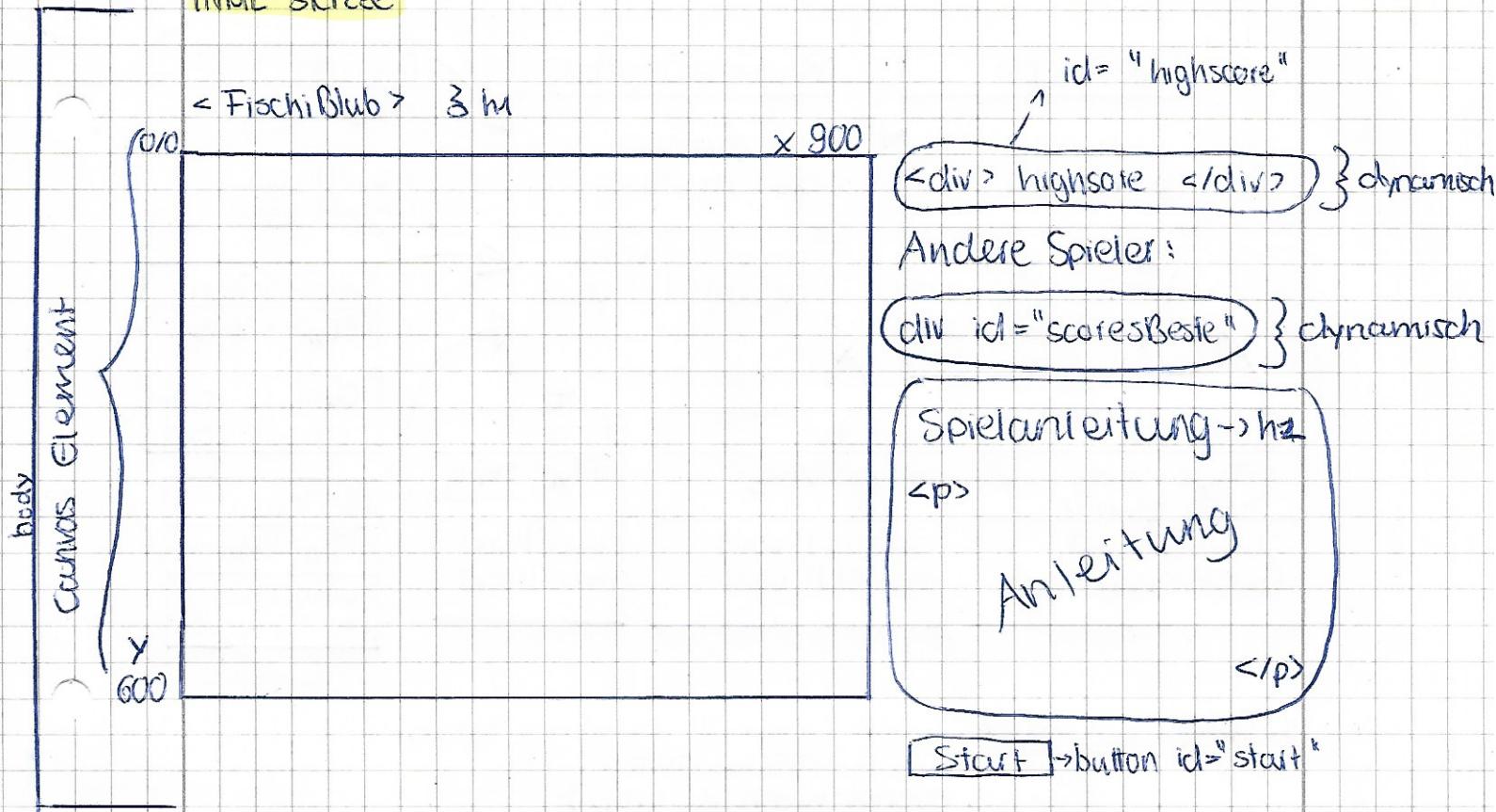
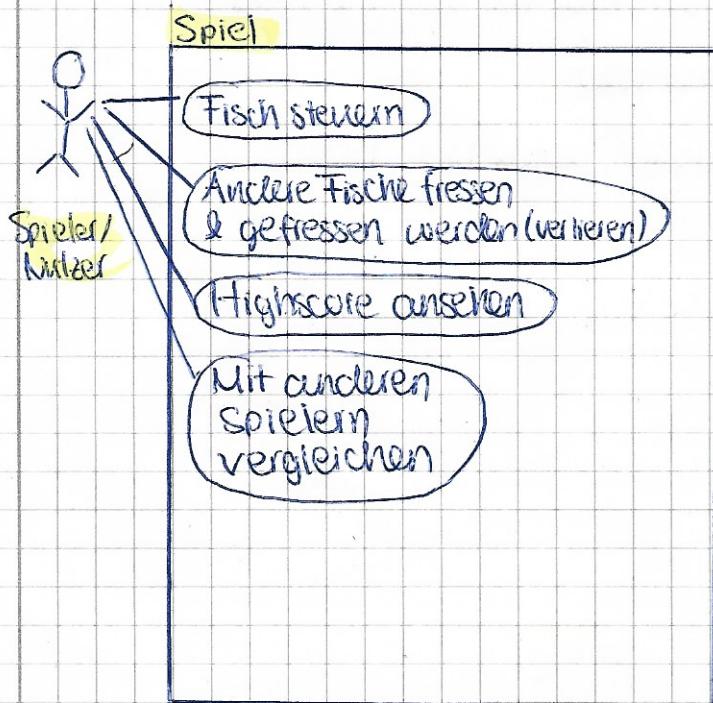


Konzepte Endabgabe: Funktionaler Teil

HTML Skizze



→ Eine PC Anwendung, da mit Pfeiltasten steuerbar und zu groß für eine mobile Anwendung!

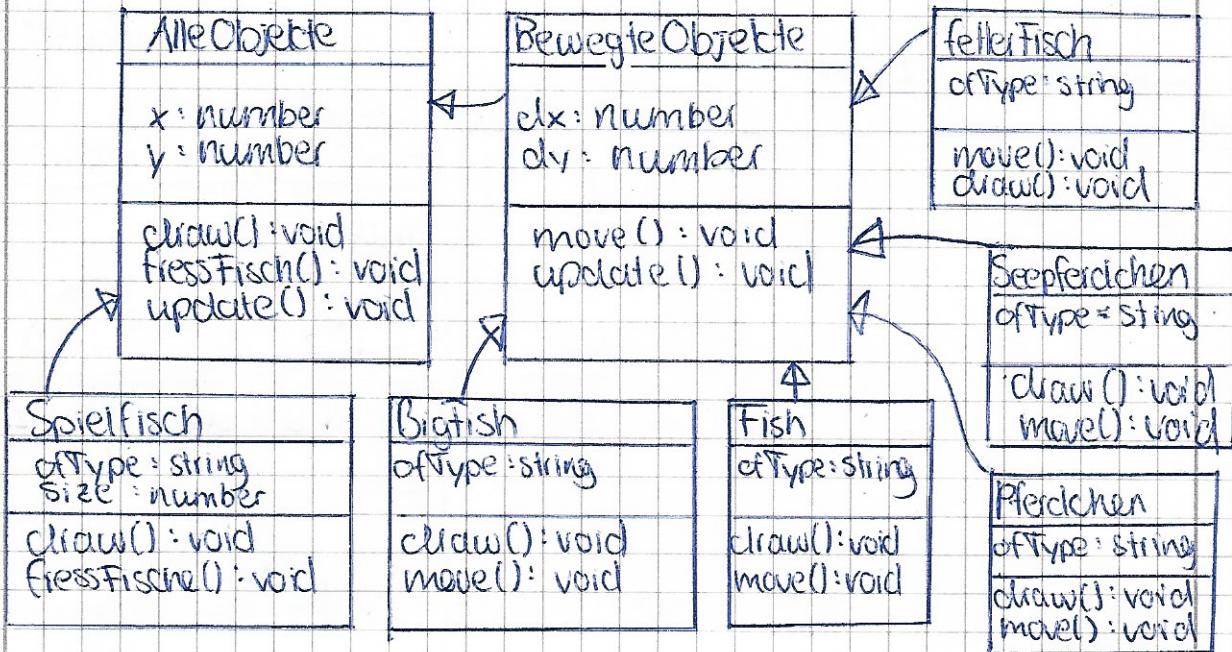


Anwendungsfalldiagramm

- ① Eventlistener Keydown registriert die Anwendung als Nutzer und löst Funktion aus welche reagiert.
- ② Mit if Bedingungen werden bestimmte Szenarien festgelegt, welche der Nutzer dann Spielt
- ③ Highscore als Variable welche bei bestimmten Bedingungen hoch oder runter gerechnet wird und an Server & Datenbank gerichtet wird.
- ④ Eintragen des Namens & das Abspeichern in der Datenbank.

Was passiert im System

Klassendiagramm



1. Spieler soll sterben, wenn kleiner als berührter Fisch
, wenn falsches Futter
(, eventl. wenn zu groß)
2. Spieler soll kleiner werden, wenn falsches Futter
zu groß
, berühren von Blasen & Futter?
3. Gegner welcher einen immer tötet
↳ Einfacher Stil des Aquariums, dafür viele Fische.
Was soll Spieler spielen können

Interaktionsmöglichkeiten

Start → Spiel startet durch Eventlistener click
 ↳ (Funktion ruft init auf)

Spielfisch → Wenn ich größer dann soll der andere Fisch aus dem Array verschwinden (wird heraus gespliced) und mein Fisch soll wachsen (increase size) und der Highscore soll erhöht werden.

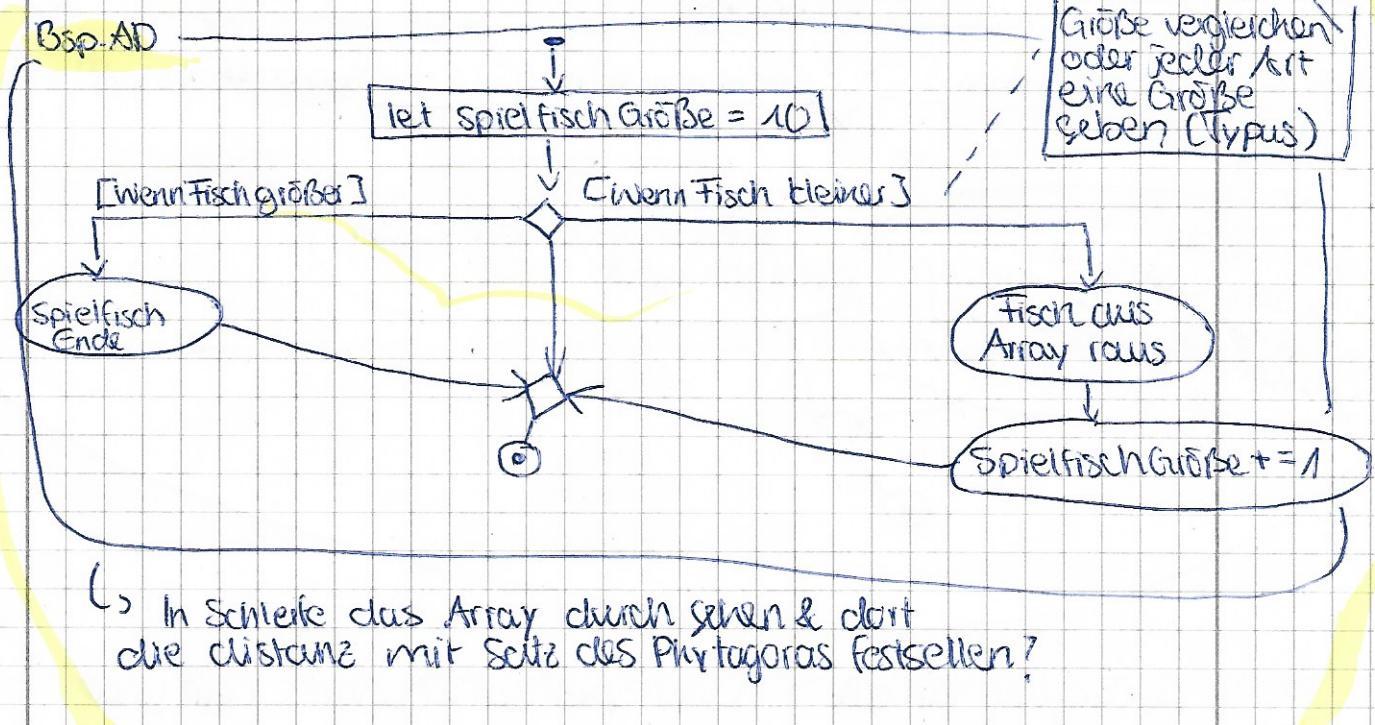
Spielfisch → Bin ich kleiner so soll das Spiel beendet werden.
↳ Ich muss meinen Namen eintragen & mein endgültiger Highscore wird angezeigt.
↳ Diese Infos werden an den Server und von dort in die Datenbank gespeist.

Gegner → Einige Fische bedeuten den sicheren Tod

→ Es soll andere Gefahren im Wasser geben, welche einen Schrumpfen lassen und/oder Punkte abziehen.

Futter → Die vorherige Futter fallen lassen Funktion soll den Fisch schrumpfen lassen.

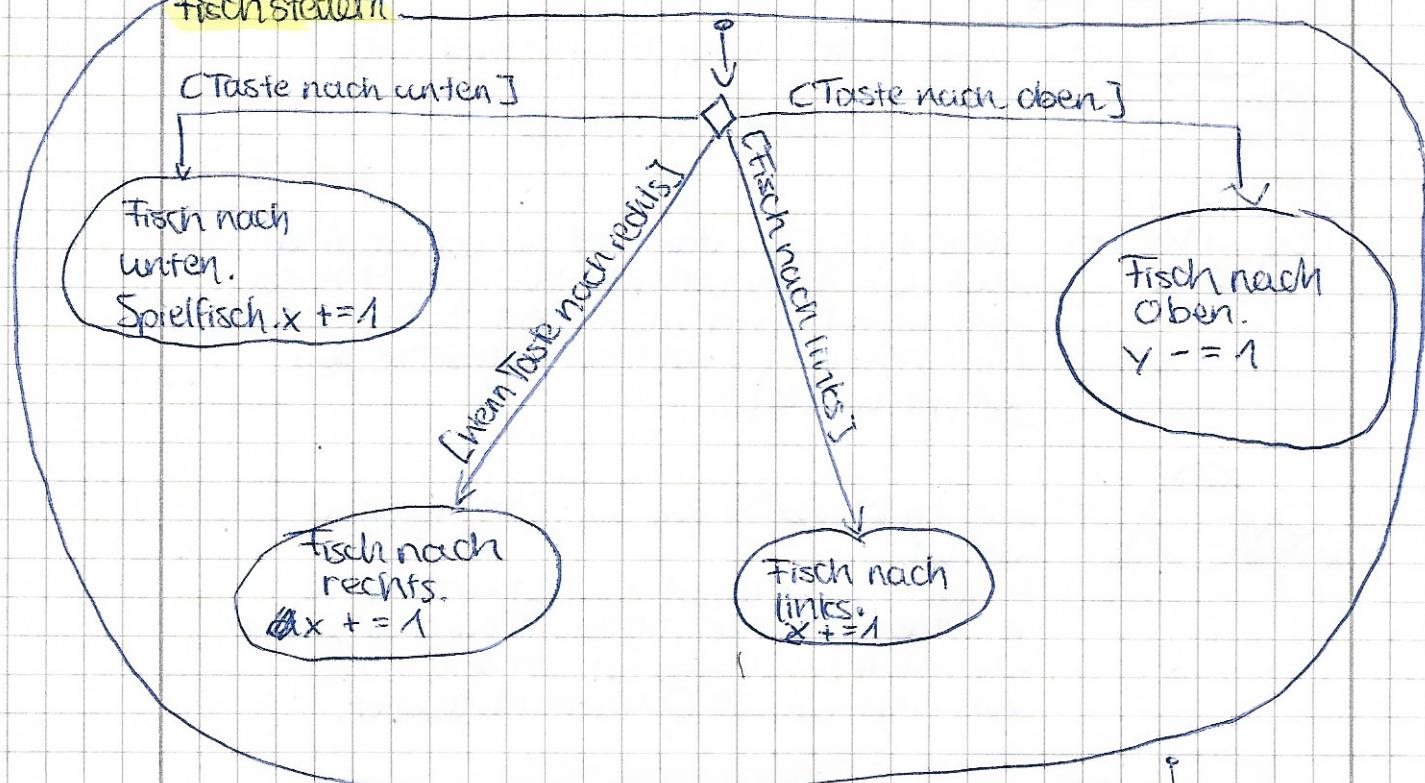
Bsp. AD



frühes AD für FressenFunktion

frühes AD für Steuern

Eventlistener (keydown, Fischsteuern) global



Fische fressen

let i = number = 0

④ C i < AllesArray.length ⑥

let distance : number = Abstand zwischen this.x und
AllesArray[i].x
↳ Rechnung?

[distance zu klein & Spielfisch klein]

[nicht mein Fisch & Größe vergleichen & Abstand klein genug]

avert (Fad)
→ Daten eintragen
mit prompt und dies in
Variable speichern und
ein Server geben

AllesArray[i].splice(i, 1)
& Fisch wachsen lassen
Spielfisch.größe += 1

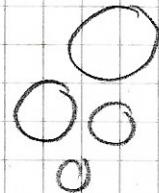
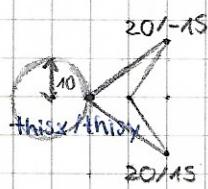
④

Frühes AD für Fressen und
Abstand

- if Bedingung für alle = Nicht Spielfisch & distance klein genug
 - ↳ Darin dann noch mehr ifs
- Fische Typen geben, dann kann ich Größe vergleichen.

0/0

Kleiner Fisch

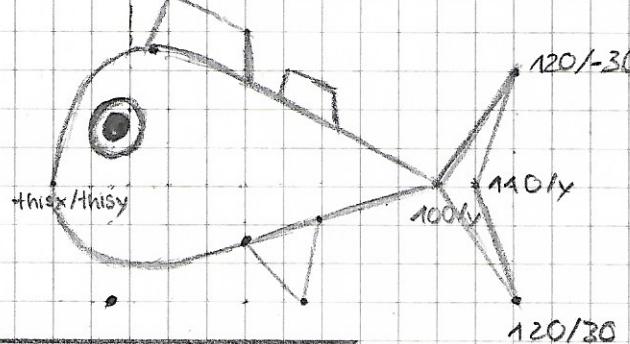


→ Für großen Spielfisch gefährlich

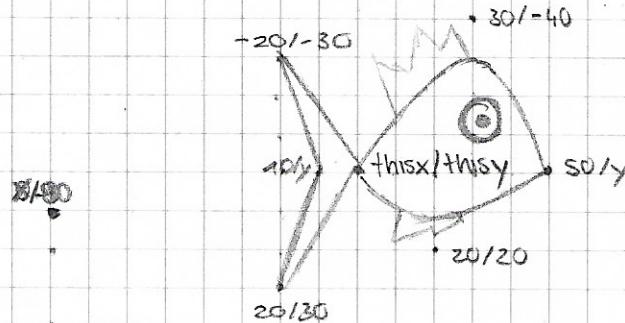
Skizzen der verschiedenen Fische.

y 600

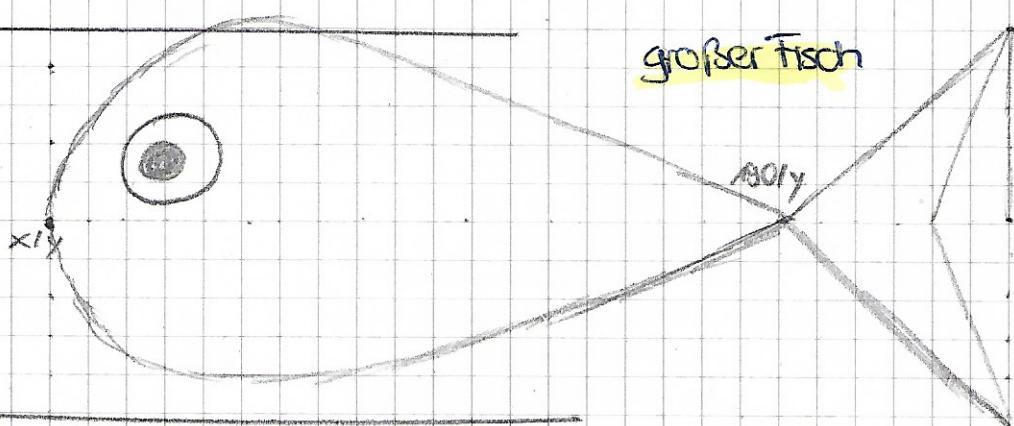
900x



mittelgroßer Fisch

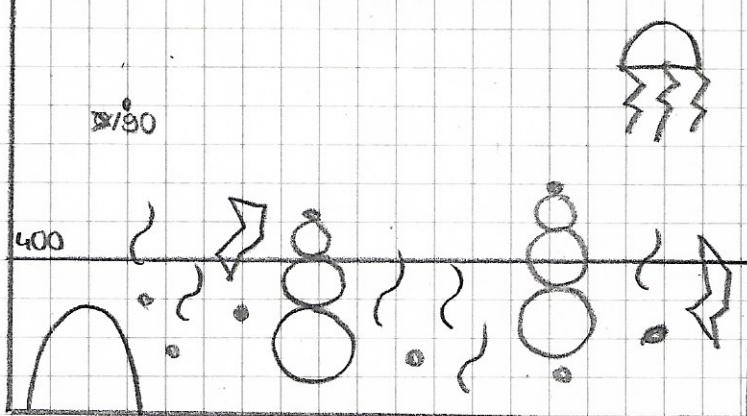


mittel kleiner Fisch



0/0

900



→ Statisch & 1x zufällig generiert

Classendiagramm Endabgabe



1. EIA - Endabgabe AD's Technischer Teil

main.ts

Anfang Global

`document.addEventListener("DOMContentLoaded", starten);`

`document.addEventListener("keydown", bewegungSpielfisch);`
→ Für die Steuerung des Spielfisches

```
export let crc: CanvasRenderingContext2D;
export let canvas: HTMLCanvasElement;
export let AllesArray: AlleObjekte[] = [];
let fps: number = 30;
let imageData: ImageData;
export let spielfisch: Spielfisch;
export let highscore: number = 0;
let timeout: number; → Für das Anhalten des Canvases
export let SpielerName: String;
```

starten

`document.getElementById("start").addEventListener("click", init);`

Damit man Anleitung lesen kann und Spiel erst dann beginnt

bewegungSpielfisch

`-event: KeyboardEvent`

[`keyCode == 39`]

`spielfisch.x += 8`

`if [spielfisch.x > 500]`

`spielfisch.x = 0`

`spielfisch.x = 8;`

`if [spielfisch.x < 0]`

`spielfisch.x = 500`

[`event.keyCode == 37`]

`spielfisch.x -= 8`

`if [event.keyCode == 103]`

`spielfisch.y -= 8`

`if [event.keyCode == 102]`

`spielfisch.y += 8`

`if [event.keyCode == 104]`

`spielfisch.y = 0`

`spielfisch.y = 8`

`if [spielfisch.y < 0]`

`spielfisch.y = 600`

`if [spielfisch.y > 600]`

`spielfisch.y = 0`

2 main.js

init

canvas = document.getElementsByTagName("canvas")[0]
crc = canvas.getContext("2d")

Hintergrundfunktion() ↴

refresh ↴

canvas.addEventListener("click", h, {capture: false})
imageData = crc.getImageData(0, 0, canvas.width / height)
↳ Hintergrundbild in imageData Variable speichern, vor den bewegten Objekten

let i: number = 0

for (i < Anzahl gewollter Elemente)

let blub: Blub = new Blub(RandomNumber, RandomNumber);

Blub in AllesArray pushen

Alle bewegten Objekte haben so eine Schleife

Spielfisch = new Spielfisch(canvas.width / 2, canvas.height / 2)

Spielfisch ins Array AllesArray pushen

update ↴

setInterval()

crc.clearRect(0, 0, canvas.width, canvas.height)
crc.putImageData(imageData, 0, 0)
↳ Hintergrundbild platzieren

let i: number = 0

(i < AllesArray.length)

AllesArray[i].update() ↴

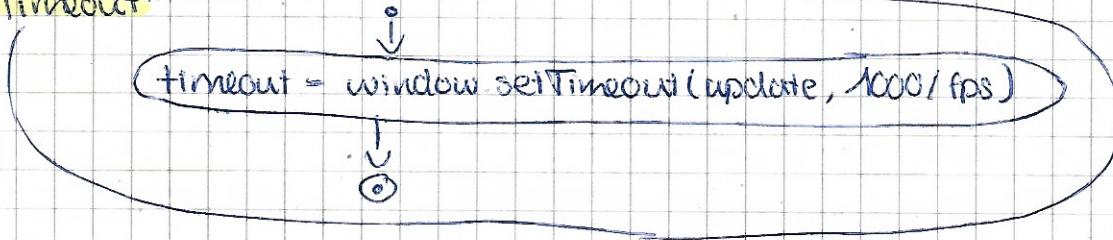
ruft für alle Elemente im Array die update auf, welche dann wiederum die anderen Funktionen wie draw & move ausführt

update

3 EA - Konzepte Endabgabe

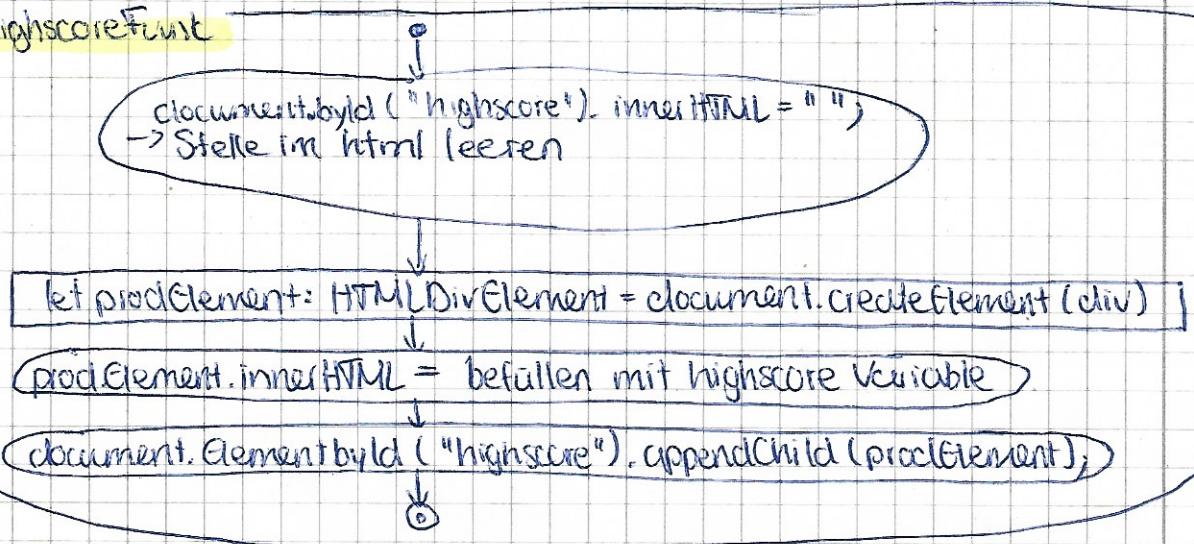
für das spätere Anhalten des Canvas

setTimeOut



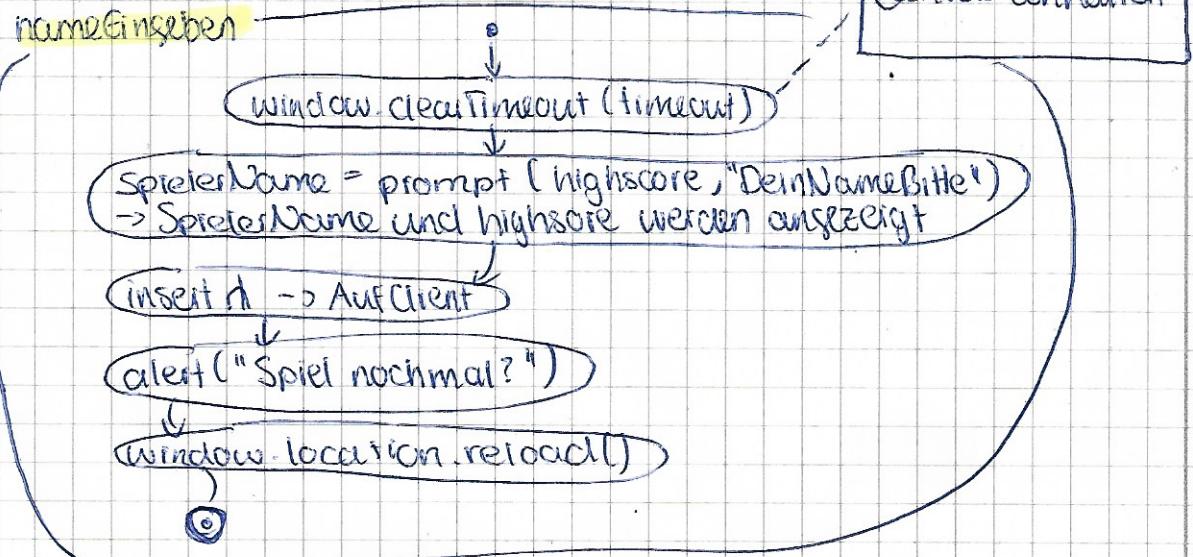
export

highscoreFunkt



export

nameEingeben



Funktion wird aufgerufen, wenn der Spieler stirbt

EIA-Endabgabe

API's

Client

Variable query beinhaltet globale Variable des Spielernamens & des highscores. Das wird auf den Server übermittelt

insert

```
let query: string = "command = insert";  
query += `&name = ${PlayerName} &point = ${highscore}`  
sendRequest(query, handleInsertResponse)
```

handleFindResource - event: ProgressEvent

Vor refresh ↑

```
let xhr: XMLHttpRequest = (<XMLHttpRequest>event.target)
```

xhr.readyState == XMLHttpRequest.DONE
↳ Antwort des Servers eingetroffen?

```
let AlleSpieler: Spieler[] = JSON.parse(xhr.response);
```

JSON in einen Array parsen
um diesen sortieren
zu können

```
let i: number = 0
```

i < AlleSpieler.length

```
AlleSpieler.sort(compareNumbers)
```

```
let i: number = 0
```

i < 6

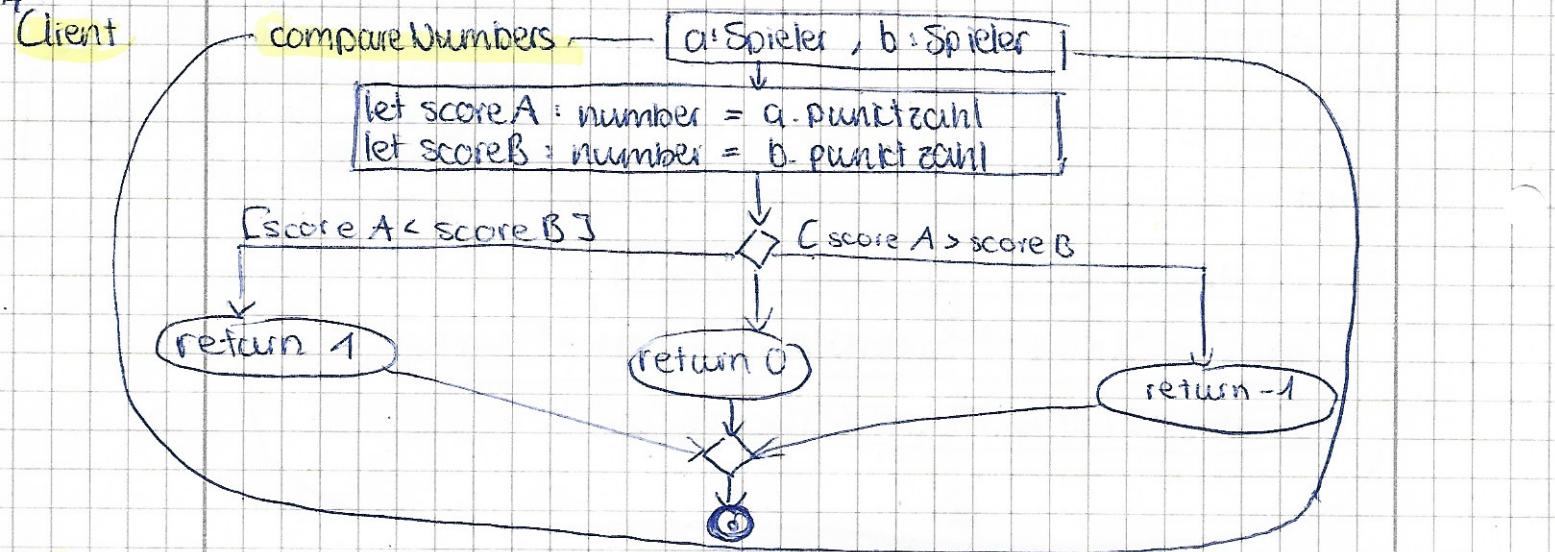
Anzahl der am Ende angesetzten Highscores

```
let prodElement: HTMLDivElement = document.createElement(div)
```

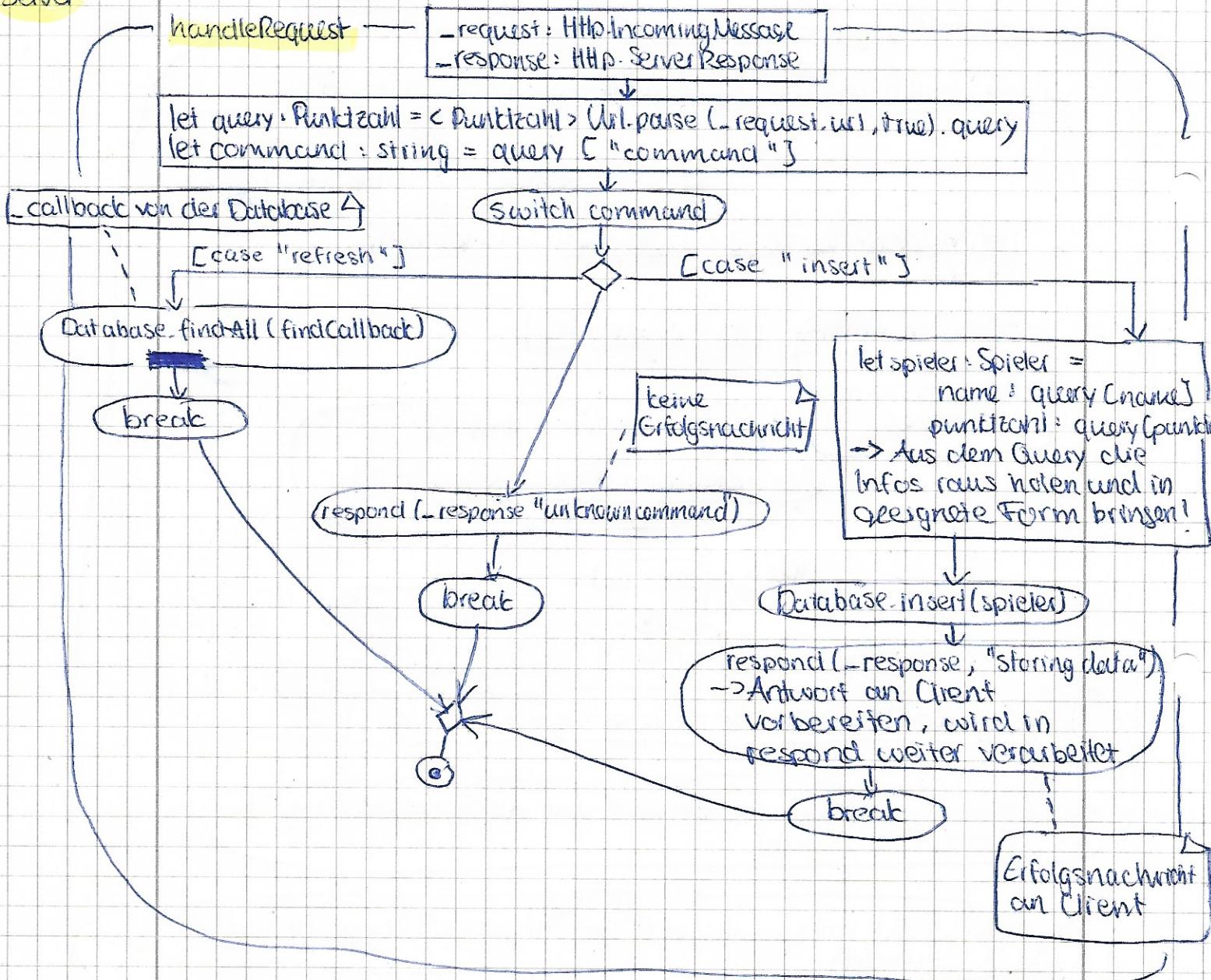
prodElement.innerHTML und befüllen mit AlleSpieler[i].name & AlleSpieler[i].punktzahl

```
document.getElementById("bestehighscore").appendChild(prodElement)
```

↳ 3000 String vom Server in ein Array parsen.
Dieses sortieren und dann die gewollte Anzahl an Highscores im HTML darstellen.

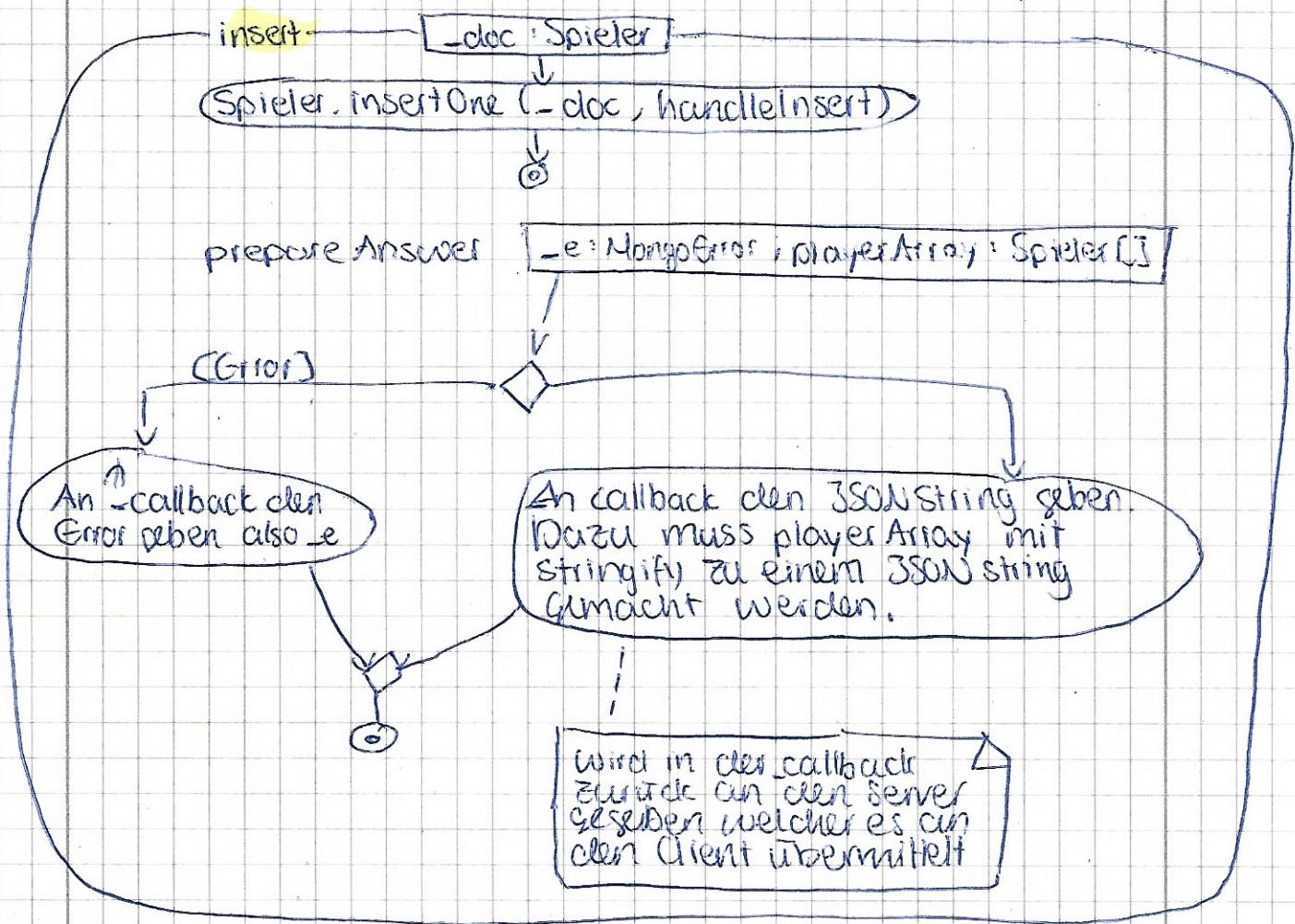


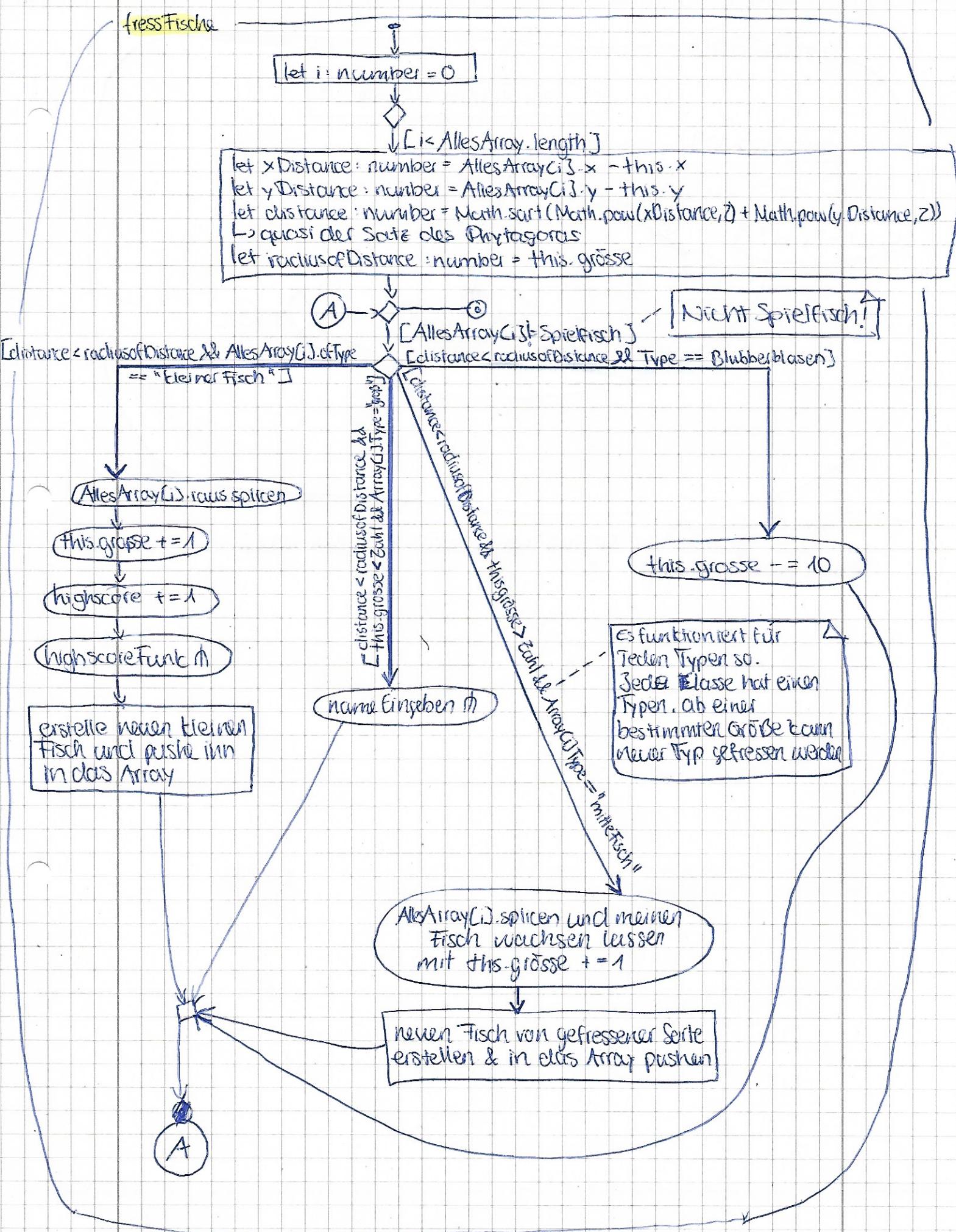
Server



Einführung Endabgabe

Database





Beispielhafte if Bedingungen, es funktioniert für alle nach dem selben Prinzip

ändern farbe

let color = String = "yellow"

[this.größe > 30]

color = "red"

color = "yellow"

color = "blue"

[this.größe < 30]

[this.größe > 10]

→ Farbveränderung
um dem Spieler zu signalisieren, dass er einen neuen
Fisch fressen kann.