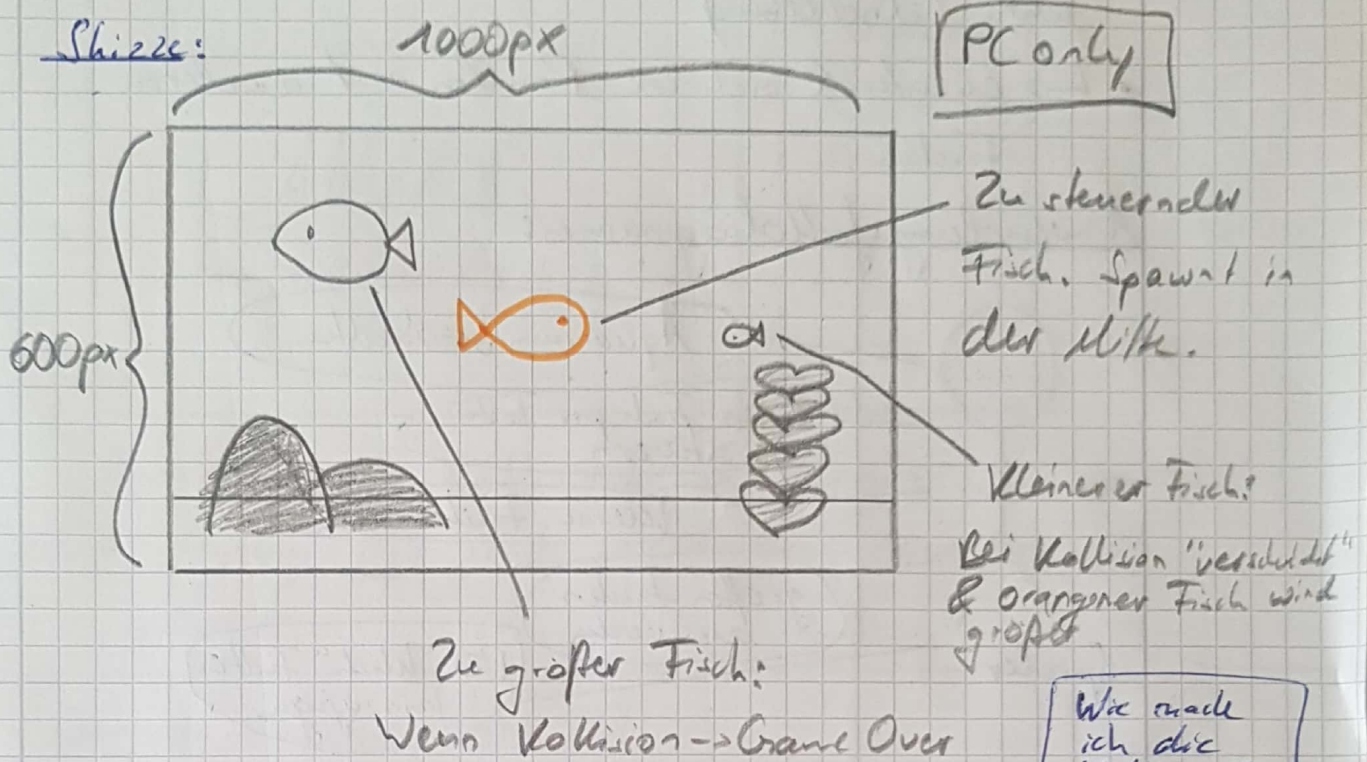


Endabgabe - ELA2

Manuel Matern



Basis: Code von Aufgabe 12.

Nutzer:

Startet das Programm und kann den Fisch mit WASD oder den Pfeiltasten steuern. Ziel ist einen höheren Score als der der in der Scoreliste stehenden Spieler zu haben. Durch kleine Futter hinzufügen. (Vielleicht später einbauen, dass essbar).

-> Canvas muss groß genug sein. & Steuerung anpassen, damit man besser navigieren kann (Schnelligkeit/Kuchelfrei).

• Pfeiltaste Links $\hat{=}$ Fisch bewegt sich nach links

• " Rechts $\hat{=}$ " rechts

• " Oben $\hat{=}$ " oben

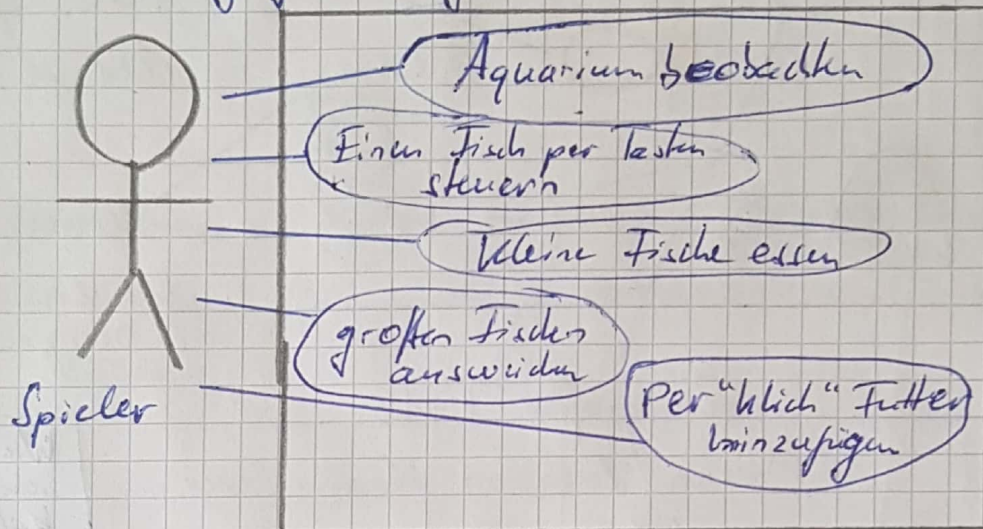
• " unten $\hat{=}$ " unten

• Nutzer klickt $\hat{=}$ Futter spawnt an Stelle des Cursors

• Nutzer/Spieler navigiert zwischen dem zu großen Fischen hindurch zu kleineren Fischen, um sie "zu essen".

- Nutzer versucht so weit zu kommen wie er geht. (überleben)
- ↳ eventuell mehrere Runden mit anderen Fischen.

Anwendungsfalldiagramm:



System/ Aquarium Spiel

Anwendung zum Zeitvertreib / Spass.

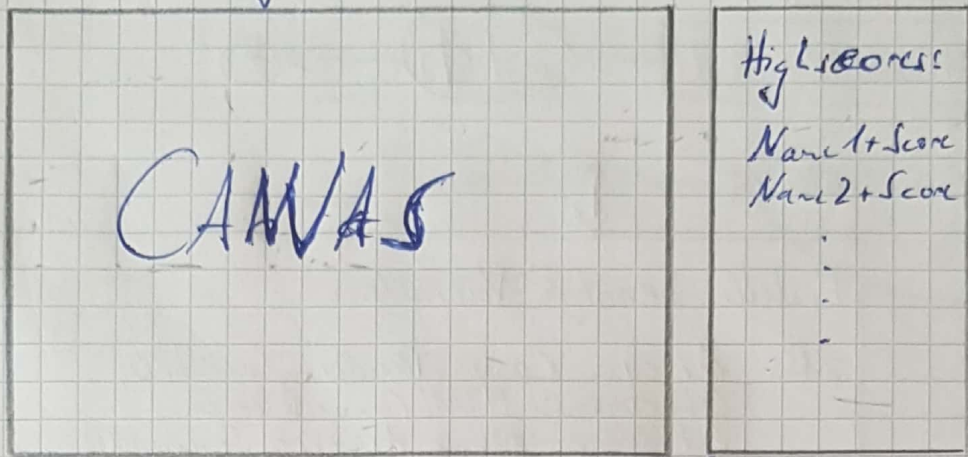
Weitere Nutzer Interaktionen und Reaktionen des Spiels:

- Schwimmt man aus dem Canvas heraus (egal ob oben, unten, rechts oder links), kommt man auf der gegenüberliegenden Seite wieder heraus.
- Bei Fall Crane Over, erscheint ein Textfeld mit der Möglichkeit seinen Namen einzutragen und dem erreichten Punktes.
- ↳ Neustart $\hat{=}$ Reload der Seite

Deine Punktzahl: 12

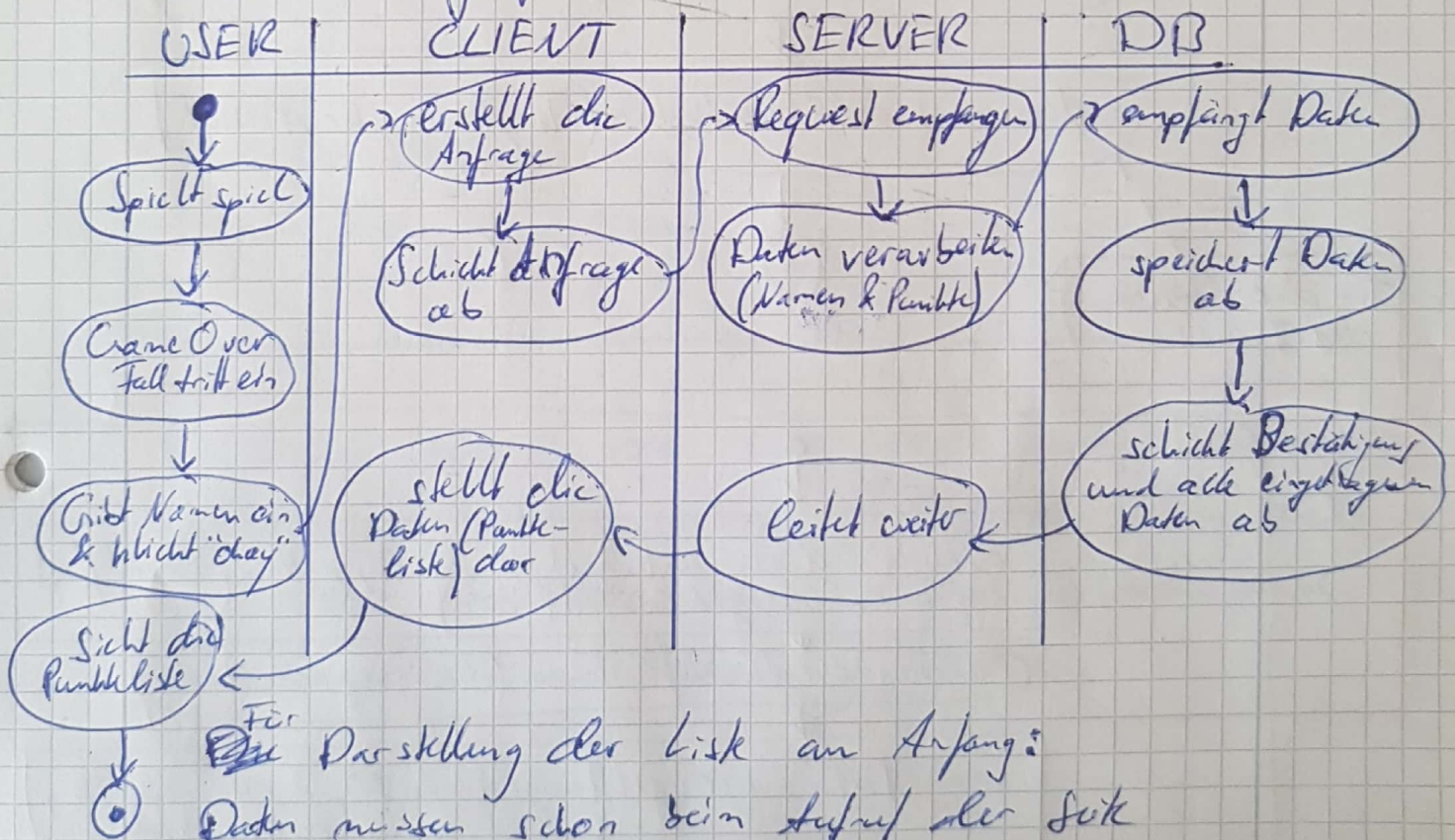
Name: Hier eingeben

Skizze ~~der~~ Highscore Liste:



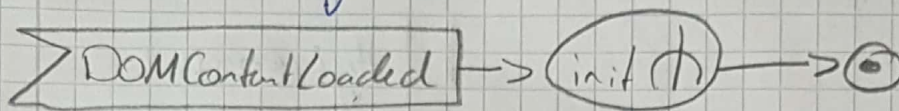
Anordnung mit CSS margin & Flexbox
→ display: flex und flex-direction: row im
Body:

Domänen-übergreifendes AD:



Für die Darstellung der Liste am Anfang:
Daten müssen schon beim Aufruf der Seite
geladen werden. → Command ~~show~~ "Anzeigen" muss
aufgerufen werden.

Aktivitätsdiagramme:



Beziehungsweise:

Erstelle benötigte Variablen:

z.B. let crc: CanvasRenderingContext2D;
let canvas: HTMLCanvasElement;
let seaworldthingsArray: SeaWorldThings[]; [?]
let punktzahl: number; [?]
let fps: number = 30;
let imageData: ImageData;

Sollte vor der
"Erstellung der
Variablen"

document.addEventListener("DOMContentLoaded", init)

document.addEventListener("keydown", keyPresser)

Für die Steuerung
des Fisches

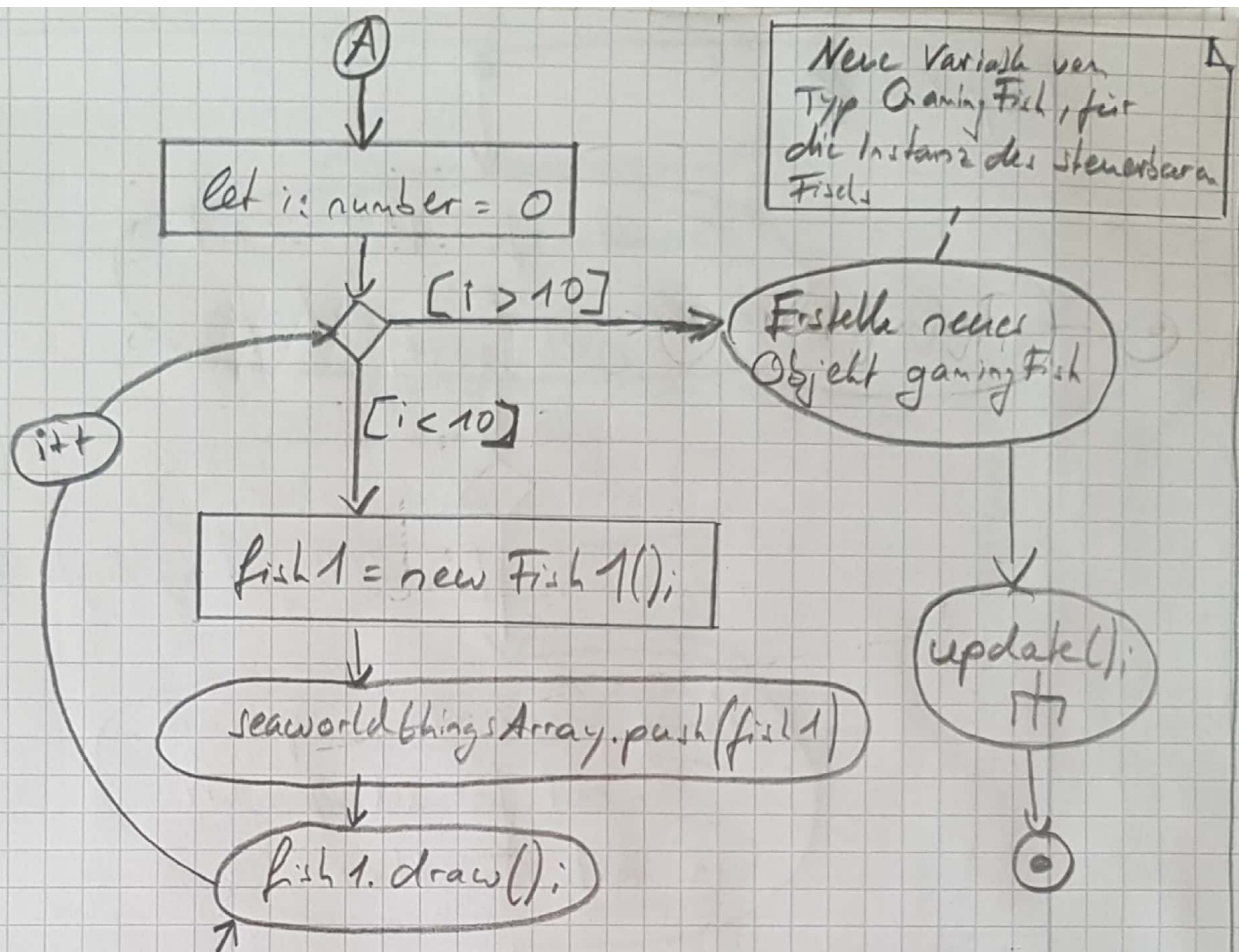
init

canvas = document.getElementById("canvas");
ctx = canvas.getContext("2d");

canvas.addEventListener("click", spawnSnacks)

imageData = ctx.getImageData(0, 0, canvas.width, canvas.height)

A



Als Basis wurde der Code aus Task 12 verwendet. Falls manche AD's unvollständig sind oder fehlen, dann sind diese Funktionen schon im Code vorhanden und wurden bereits in Task 12 geplant.

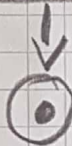
update

Funktion für Kollision mit
anderem Fisch aufrufen
collisionstr



[seaworldthings Array.length == 0]

Lasst neue Fische spawnen
mit for Schleifen wie
in init-Funktion



collision

let i: number = 0

[i > seaworldthings Array.length]



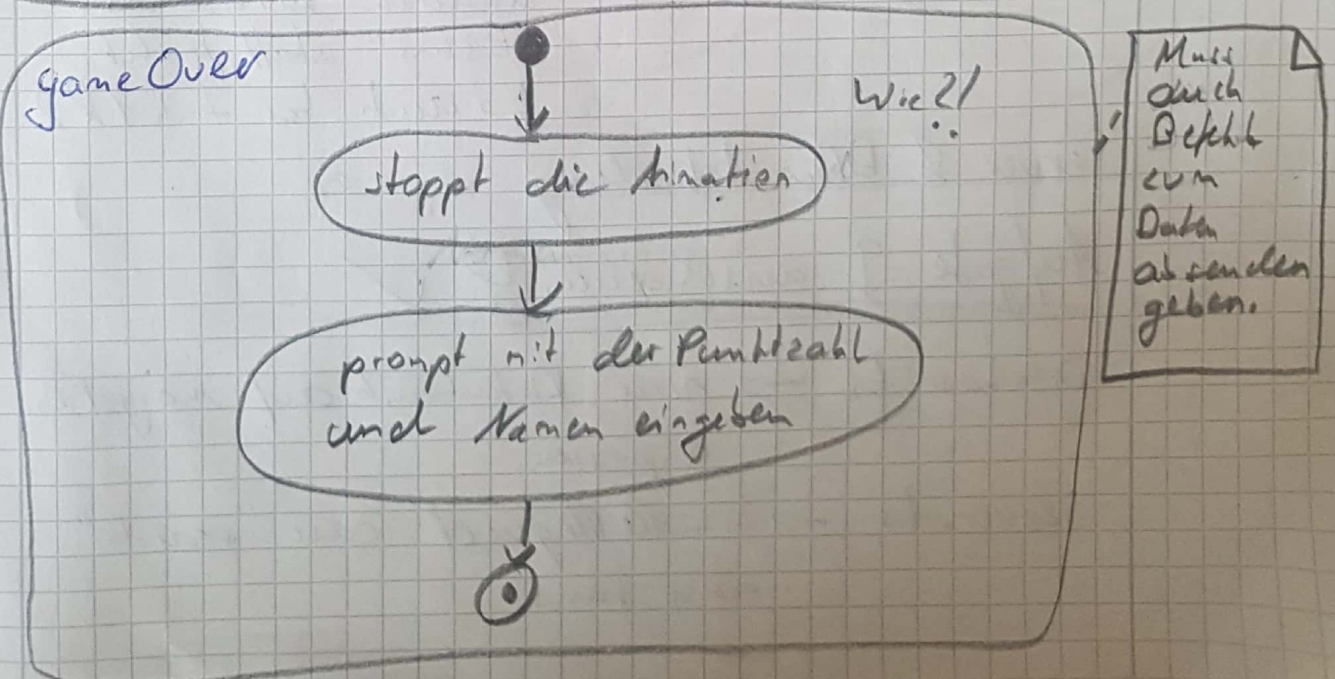
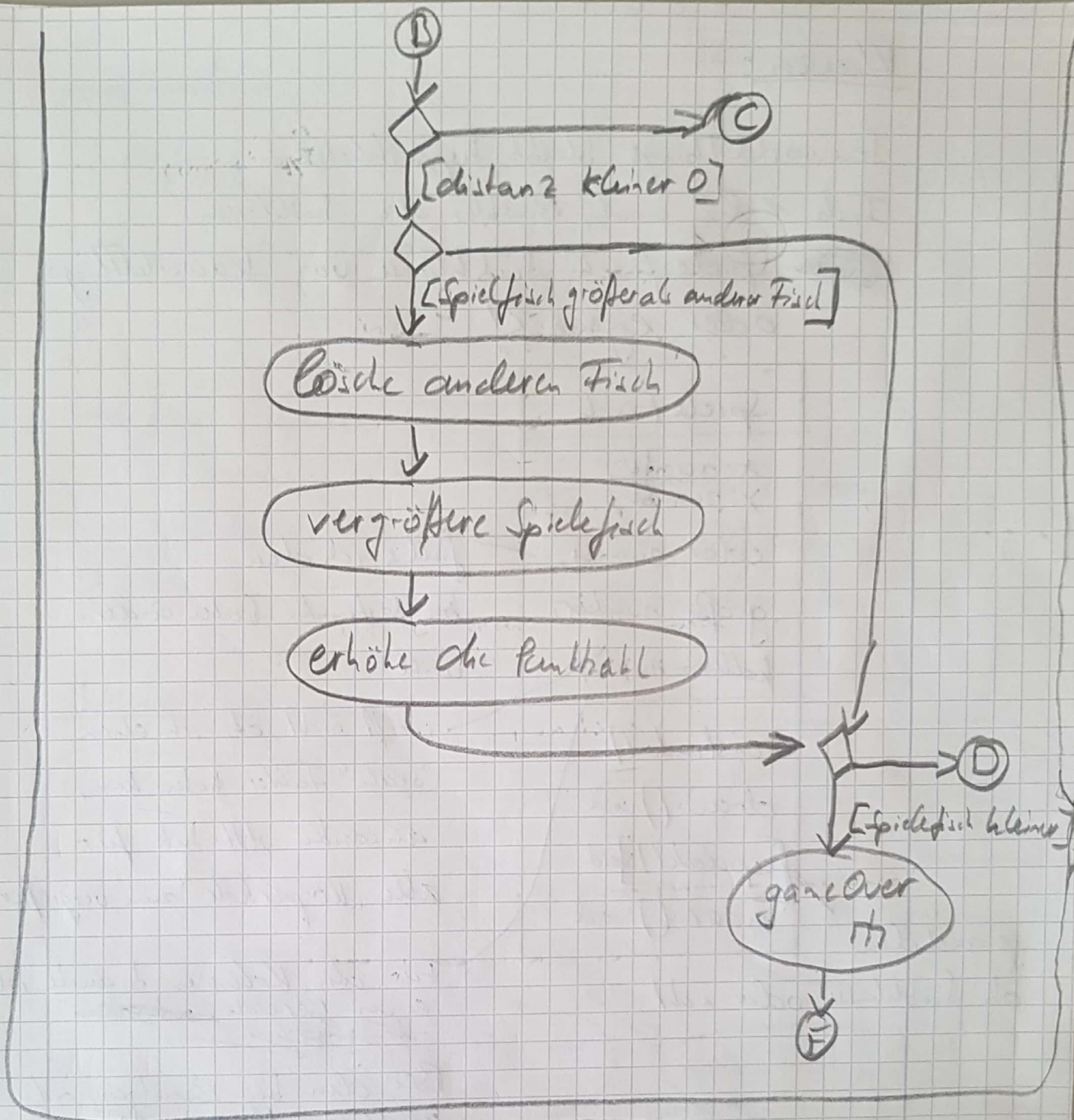
C

F

[i > seaworldArray.length]

Variablen für Distanzberechnung



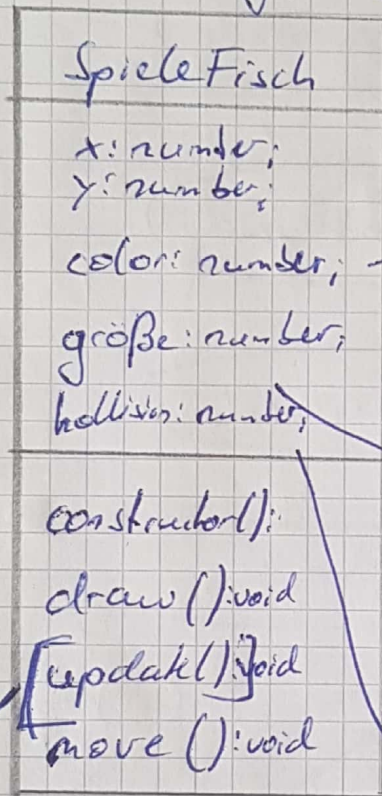


Klassen:

SeaworldThings bleibt Superklasse.

Fish 1 & 2 und Bubbles die Subklassen.

~~SeaworldThings~~ Spielfisch Subklasse von SeaworldThings oder eigene?!



falls ich diese
bei bestimmten Sachen ändern
will

weiß nicht ob ich einen
"scale" Faktor haben kann,
ansonsten Attribute für
alle "Körperhite" zum vergrößern

je nachdem
ob Subklasse, oder nicht

Für die Kollision brauche ich
einen Bereich ~~mit dem~~
~~ich arbeite~~

Bei dem bei Berührung mit
anderem Bereich das
"Essen" statt findet.

↳ auch bei Fish 1 & 2

Server & DB - Anbindung:

Aufgabe 9 anschauen ✓

Database.ts → neue Datenbank auf mongoDB.
anpassen

Server.ts → handleRequest case "search"
muss raus.

Types.ts

```
interface Punktzahl {
```

```
  [key: string]: string;
```

```
}
```

```
interface Spieler {
```

```
  name: string;
```

```
  punktzahl: number;
```

```
}
```

brauche ich ~~es~~
für DB und
zum Anzeigen.

Steuerung:

