

EIA - Endaufgabe:

To-Do:

- aktueller Stand: Vögel fliegen von links nach rechts (oben ohne Beine, unten mit)
- Vögel sollen zufällig durchs Bild fliegen bzw. von oben schräg nach unten (sobald eine bestimmte height - wahrscheinlich die Hälfte - erreicht wird, ändern sich die Vögel zu stehenden Vögeln. Also alles unter der Hälfte der Height sind immer stehende Vögel, alles darüber fliegende)

Futter hinwerfen: durch Doppelklick - Event (Event: "doppelklick", "auxclick")

- Futter an eine bestimmte Stelle hinwerfen (mehrere Partikel verstreut um ein Zielposition in einem bestimmten Radius, braune ausgefüllte Kreise, Art und Weise siehe Asteroids oder Postkarte)
- ein paar zufällig ausgewählte Vögel sollen dort hinfliegen (d.h. sie teleportieren sich nicht gleich zu dem Futter, sondern man sieht wie sie dort hinfliegen, also ihren Weg. Dafür ist ebenfalls eine bestimmte Zeit festgelegt evtl. ca. 3sec)
- Timer für eine bestimmte Zeit z.B. 3-4sec, in der die Vögel sich an der Position des Futters befinden und in der das Futter angezeigt wird
- nach Ablauf dieses "Timers" verschwindet das Futter und die Vögel fliegen zurück (evtl. zu ihrer vorherigen Position oder fliegen/bewegen sich irgendwie zufällig weiter). Sie fliegen nun wieder wie gewohnt

Mit Schneeball Vögel abwerfen:

- Als Schneeball soll ein weißer ausgefüllter Kreis angezeigt werden, also wie eine Schneeflocke nur größer

Möglichkeit zur Lösung:

- Button mit der Benennung: Vogel abwerfen o.ä.
- Drückt man auf den Button so erscheint unten in der Mitte am Canvas Bildschirm der Schneeball (nur einer)
- Durch Tippen auf eine Stelle im Canvas (erst nach Drücken des Button mit dem jz gefolgten Effekt möglich) soll der Schneeball sich zu dieser Position bewegen (Also Weg von unterem Bildschirmrand zu der Position, an der man hingeklickt hat)

→ Dabei soll man den Weg/Bewegung also den Wurf sehen können d.h. er teleportiert sich nicht einfach zur der Zielposition

- Ist der Schneeball an der Zielposition angekommen, so verschwindet er nach ungefähr 1sec verweilen an dieser Position (Schneeball wird gelöscht)

Überprüfung:

Es gibt eine Überprüfung an der Zielposition, ob sich dort derzeit ein Vogel aufhält, Überprüfung findet aber erst statt, wenn Schneeball sich an Zielposition befindet und dort eine Sekunde verweilte; oder Überprüfung findet vor dem Verweilen statt und falls sich dann an dieser Position ein Vogel befindet verweilt dieser genau so lang (also auch 1sec)

Möglichkeit der Umsetzung der Überprüfung:

- Es wird ein Array durchgegangen am besten extra Array für die Vögel, damit nur das durchgegangen werden muss und sich somit auch nicht der Schneeball drin befindet ODER normales Moveables-Array wird durchgegangen (beinhaltet Vögel, Schneeflocken, Schneeball), alle Elemente, die sich an dieser Position befinden, sollen dann nach Verweilzeit gelöscht werden → evtl. Problem, da auch normale Schneeflocken gelöscht werden können, wenn sie sich auch an der Position gerade zufällig befinden
- Wenn sich kein Vogel an der Zielposition befinden:
 - Schneeball verschwindet einfach nur (wird einfach nur gelöscht)
 - Ansonsten passiert nichts
- Wenn sich an der Zielposition ein Vogel befindet:
 - So verschwindet der Schneeball wie zuvor auch nach 1sec verweilen an der Position (wird gelöscht)
 - Vogel wird auch gelöscht → erst dann wenn Schneeball sich an der Zielposition befindet (also hier gäbe es ein Problem wenn die Überprüfung bei Klick auf die Stelle auf dem Bildschirm stattfindet würde) Problem wird aber durch die Überprüfung, die erst stattfindet wenn Schneeball an der Zielposition ist, behoben (FRAWE: Wie merkt der Vogel, dass er gerade getroffen wurde und gelöscht werden muss?)

Möglichkeit: Überprüfung ob der Schneeball sich an der gleichen Position wie ein Vogel befindet, wenn ja wird Vogel gelöscht. Timer/Zeit wurde bereits vorher ja schon festgelegt (1sec verweilen an Zielposition). Ablauf dieser Zeit wird nicht nur der Schneeball gelöscht, sondern auch der Vogel, der sich an der gleichen Position befindet)

Punkte vergeben für getroffene Vögel:

- wird ein Vogel getroffen, also muss bei der Überprüfung erkenntlich sein, dass ein Vogel getroffen wurde, so wird eine bestimmte Punktzahl gutgeschrieben (evtl. 10 Punkte)
- Zum Beispiel erkenntlich, dass ein Vogel getroffen wurde, durch das Durchgehen eines Birds-Array, wenn diese Überprüfung erfolgreich war, wird nicht nur der Vogel gelöscht, sondern auch die Punktzahl um 10 erhöht.
- Bei der anderen Überprüfungs-Variante, also das Durchgehen des Movables-Arrays muss zusätzlich überprüft werden ob das gefundene Objekt der Klasse Bilds abgehört, wenn ja wird die Punktzahl um 10 erhöht
 - (erzielte Punkte werden parallel in einer Datenbank gespeichert und werden mit zusätzlich hinzukommenden addiert --> ist wahrscheinlich nicht mal nötig, reicht denke ich, wenn die Daten erst am Ende des Spiels durch die Eingabe des Spielers (+Name) in der Datenbank gespeichert wird)
 - erzielte Punkte werden auch auf dem Bildschirm neben dem Canvas angezeigt, Beispiel: "Punktestand: ... " Diese Anzeige wird andauernd aktualisiert, sodass neu hinzukommende Punkte der dafür vorgesehenen Variabel hinzugerechnet wird und diese aktualisierte Variabel dann letztendlich angezeigt wird.

Ende des Spiels:

Möglichkeiten:

1. Zeit läuft ab/Timer (evtl. 2min), dieser wird auf dem Bildschirm angezeigt. Ist der Timer abgelaufen also null, so hört das Spiel auf.
2. Wenn eine bestimmte Punktzahl erreicht wurde, also die Punktzahl-Variable einen bestimmten Wert erreicht (-> ständiges Abgleichen ob, die Variable diesen einen festgelegten Wert erreicht hat, also den gleichen Wert angenommen hat), so ist das Spiel vorbei.

Danach (unabhängig von der Möglichkeit):

Es erscheint ein Alert, in dem die erreichte Punktzahl angezeigt wird (also der Wert der Variablen, die für die Punktzahl genutzt wird), außerdem kann man seinen Namen eintragen. Danach bestätigt man seine Abgabe mit einem Button. Die Eingabe wird, anschließend in die Liste (Highscore-Liste) der Datenbank aufgenommen.

Datenbank:

- Es wird eine Liste (Highscore-Liste) erstellt mit allen Einträgen, also alle Einträge werden dauerhaft gespeichert
- Ein Eintrag wird in die Liste erst übernommen, wenn das Spiel vorbei ist, dabei wird die Punktzahl verwendet, die als letztes abgespeichert wurde/aktualisiert wurde
- Diese Liste wird sortiert nach der Punktzahl, die letztendlich erreicht wurde. Höchste Punktzahl als erstes anzeigen. Möglichkeit: Hier soll mit einer Scheife schrittweise und chronologisch das Array der Liste durchgegangen werden und dabei sollen diese Werte nach der Höhe der Punktzahl sortiert werden. (siehe Array, das die Reihenfolge mehrerer Zahlen nach ihrer Größe ordnet --> im Tutorium EIA1 bei Maxim wurde das mal durchgenommen, in EIA-Unterlagen nachschauen!)

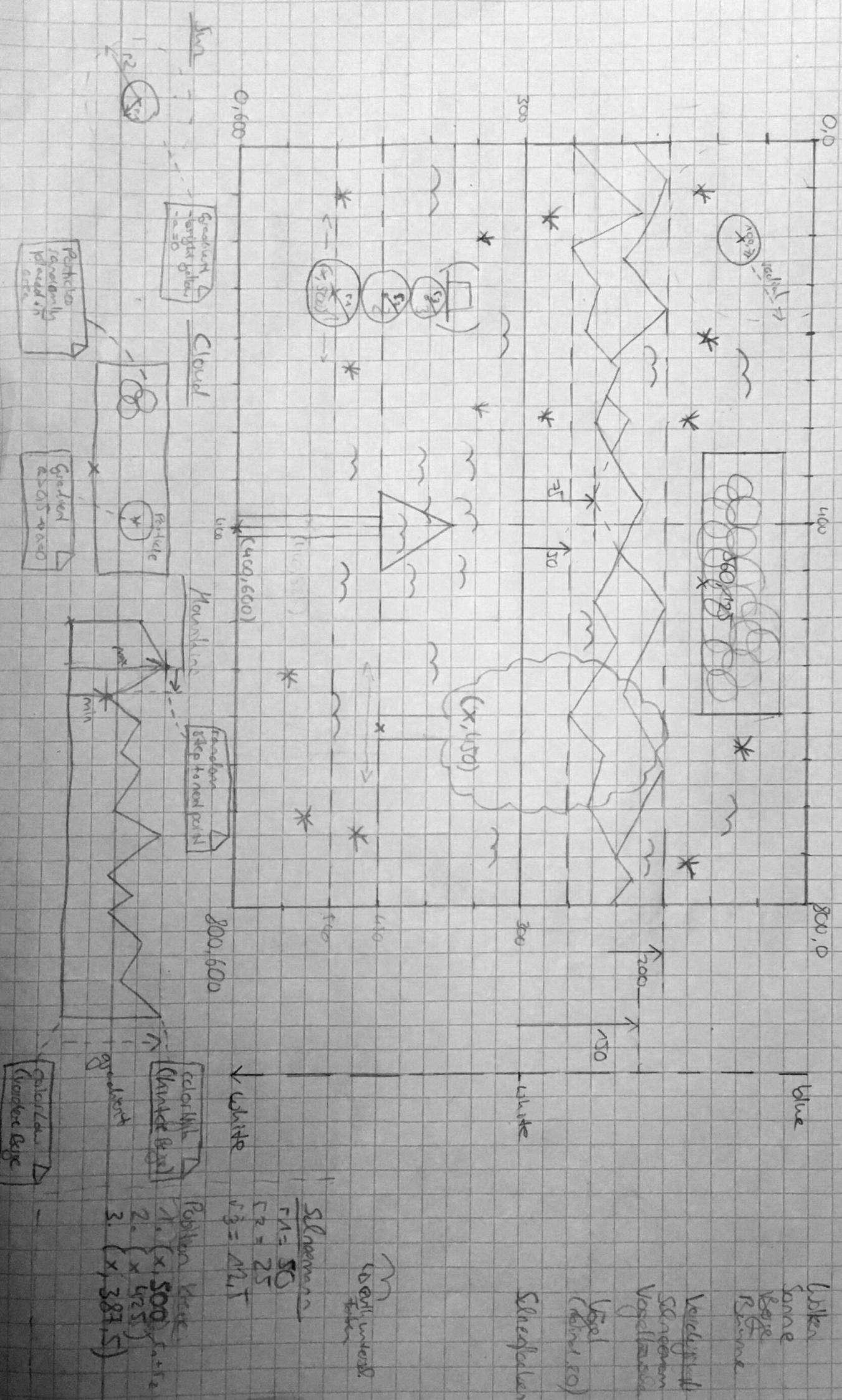
→ Hier (in der Datenbank) gibt es demnach nur die Highscore-Liste!

canvas / 2 ?

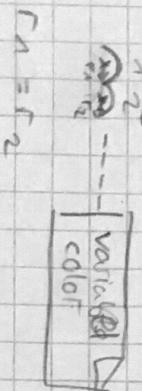
- ① \hookrightarrow bei mit $y=300$ $s=300$
- Horizont \hookrightarrow Vögel sollen sich ändern (zu steuern)
 \hookrightarrow ggf irgendwie nicht bei $y=300$, irgendwie durch ausweichen
verhindern, dass es bei $y=150$ geht
 - ② - Mouse-Klick-Event: normales Zugreifen durch click x, y hat nicht funktioniert, dann aufgrund von ① mit der Maus gesteuert
 \Rightarrow hat geklappt

(Ergänzung - canvas) Vogelhaus: Skizze

(Ergänzung - canvas) Vogelhaus: Skizze



Vogel

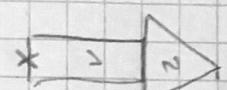
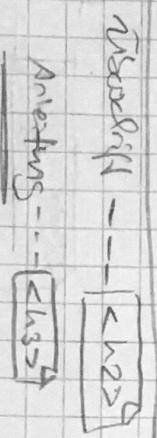


1. Kreis
(x_1, y_1)

2. Kreis
($x_1 + r_1 + 2, y_1$)

arc \rightarrow Skew: 0
End: ANGLE_PI

\rightarrow Rauten-Kreis: y1 - Weite
M-Scribble Skizze:

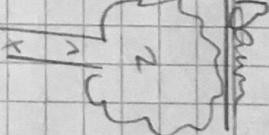


1. Position (400, 600)
fillRect(x, y, 20, 20)

2. draw Path
Dreieck

Vogelzwerfen

Baum

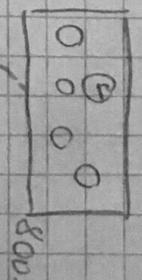


1. Position (x, y50)

fillRect(x, y, 20, 20)
color = "brown"

2. Skizzieren

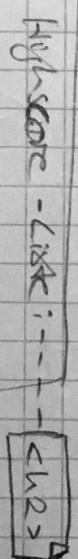
Schneeflocken



Skizzierter Rahmen
placed in area

color = white

M-Scribble Skizze Seite:



``
`<div>`
`id = "inhaltn"`

`<div>`
`id = "linien"`

`<div>`
`color =`
`id = "inhaltn"`

`<div>`
`color =`
`id = "inhaltn"`

`<div>`
`color =`
`id = "inhaltn"`

`<div>`

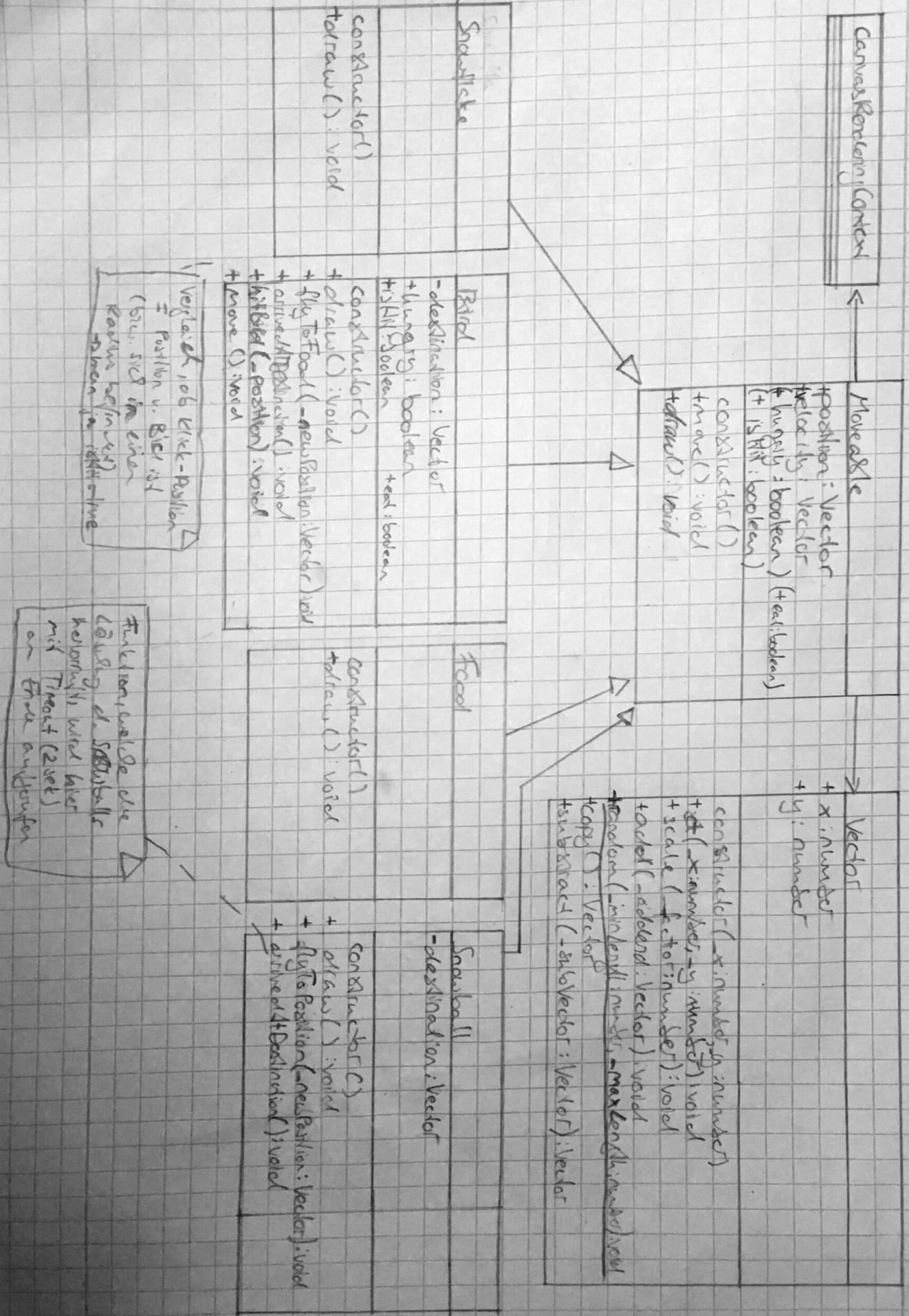
`<div>`

Ergebnisse
ca href = "http://..."
"Abgabe.html"

`<div>`
`color =`
`id = "inhaltn"`
`body id = "inhaltn"`

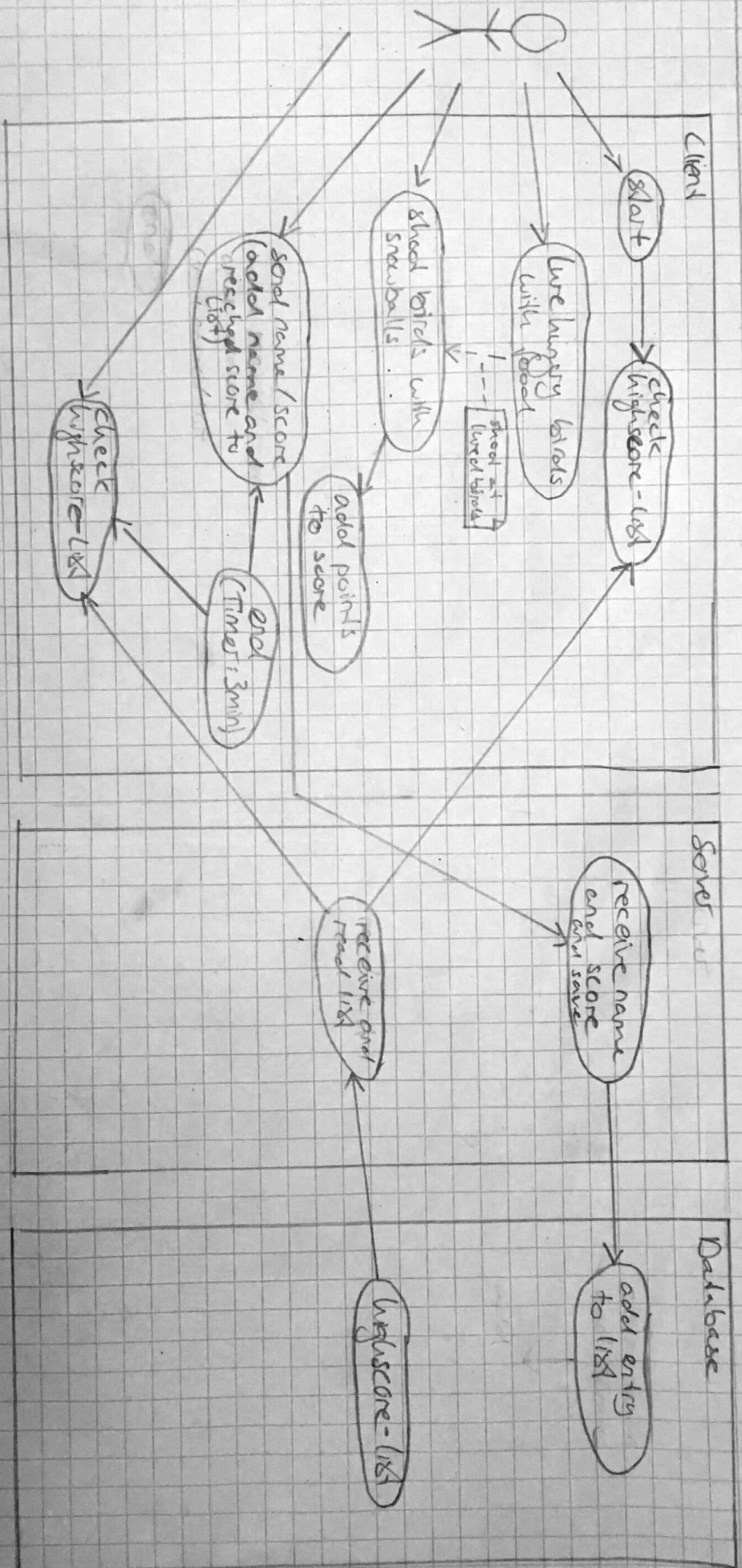
L13 - Asselwesendage

Vogelländchen: Class Diagram



L13 - Abschlussaufgabe

User-Case-Diagramm / Anwendungsfalldiagramm



L13 - Auseinandersetzung

let snowballFly = boolean = false;

> click → snowballHandler →

snowballHandler

let snowballHandler boolean = false;

```
let newPosition =  
Vector(event.x, event.y)
```

drawSnowball([snowball])

Aktivitätsdiagramm: Schneeball werfen

flyToPosition
(method of
Snowball)

newPosition Vector
destination = newPosition
velocity = (destination - actualPosition)
snowballFly = true;

Schneeball
start run
over

arriveAt Destination
(method of Snowball)

newPosition Vector
(this.destination)

newPosition Vector
(this.destination)

newPosition Vector
(this.destination)

newPosition Vector
(this.destination)

drawSnowball([snowball])

drawSnowball([snowball])

drawSnowball([snowball])

drawSnowball([snowball])

drawSnowball([snowball])

drawSnowball([snowball])

drawSnowball([snowball])

startTargetBox

newPosition Vector

moveSnowballToPosition(newPosition)

drawTargetBox([snowballFly = false])

moveSnowballToPosition(newPosition)

drawTargetBox([snowballFly = false])

moveSnowballToPosition(newPosition)

targetCrossWithAreaShell:
true

drawTargetBox([snowballFly = true])

moveSnowballToPosition(newPosition)

drawTargetBox([snowballFly = true])

moveSnowballToPosition(newPosition)

drawTargetBox([snowballFly = true])

moveSnowballToPosition(newPosition)

deleteSnowball
→ Position bei -newPosition

moveSnowballToPosition(newPosition)

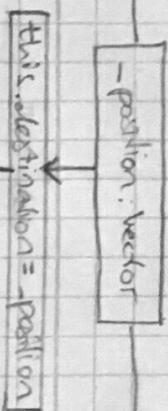
drawTargetBox([snowballFly = false])

moveSnowballToPosition(newPosition)

drawTargetBox([snowballFly = false])

moveSnowballToPosition(newPosition)

L1Bird
Meshell of Bird



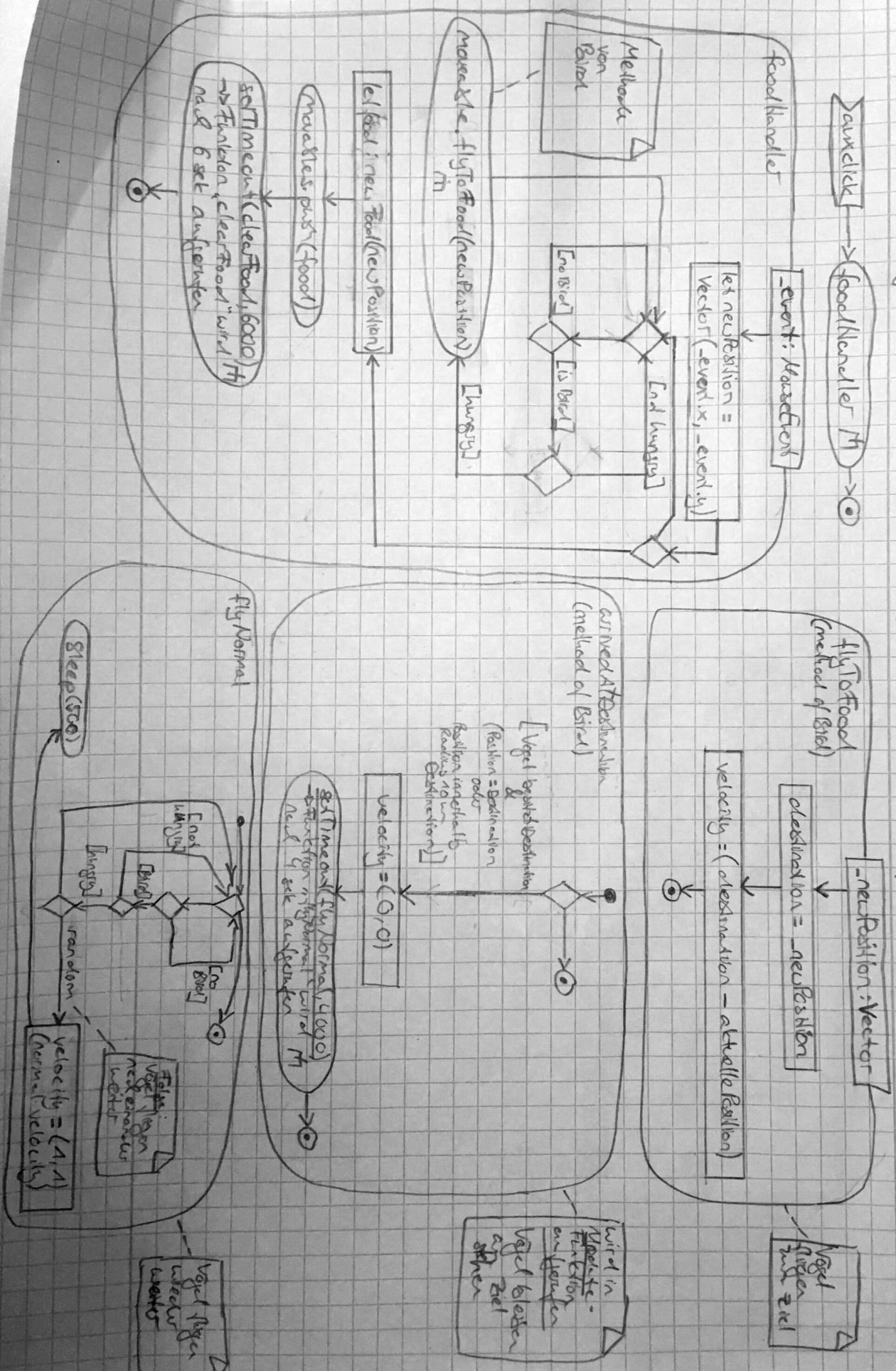
Vgl mit Destination

(Position = Destination
oder
Position innerhalb des
Raumung um
Destination)

`this.ISHT = true`

L13 - Abschlussaufgabe

Aktivitätsprogramm: Futter hinweisen



clearFood

[$i \geq \text{mavables.length}$] \rightarrow

[$i < \text{mavables.length}$]

if
[mavables[i] is Food]

mavables.pop(i)

fuller
wider petard
(and ready
to roll back)

L13 - Abschlussaufgabe

Aktivitätsdiagramm : Hauptprogramm (Main)

```

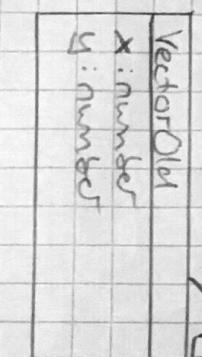
let score: number = 0
let crc2: CanvasRenderingContext2D
let snowballify: boolean = false
let moveables: Moveable[] = []

```

```

VectorOld
x: number
y: number

```



```

handleLoad
- event: Event
    let canvas: HTMLCanvasElement
    (...) = document.querySelector("canvas")
    crc2 = canvas.getContext("2d") as canvas
    (...) setContext("2d")
    let background: ImageData = crc2.
    (...) getImageData(0,0,800,600)
    let posMountain: VectorOld = {x:0,y:300}

```

```

Background
drawBackground()
drawSun()
drawCloud()
drawMountainShell()
drawMountains(dark)
drawTree()
drawSnowman()
drawHouse()
drawSnowball()
drawBirds()
drawSnowball()

```

install listeners

```

drawBackground
drawSun()
drawCloud()
drawMountainShell()
drawMountains(dark)
drawTree()
drawSnowman()
drawHouse()
drawSnowball()
drawBirds()
drawSnowball()

```

```

drawSnowflake
let nSnowflake: number = 100
[< nSnowflake]
    < isSnowflake
        < isSnowflake
            < isSnowflake
                < isSnowflake
                    < isSnowflake
                        < isSnowflake
                            < isSnowflake
                                < isSnowflake
                                    < isSnowflake
                                        < isSnowflake
                                            < isSnowflake
                                                < isSnowflake
                                                    < isSnowflake
                                                        < isSnowflake
                                                            < isSnowflake
                                                                < isSnowflake
                                                                    < isSnowflake
                                                                        < isSnowflake
                                                                            < isSnowflake
                                                                                < isSnowflake
                                                                                    < isSnowflake
                                                                                        < isSnowflake
                                                                                            < isSnowflake
                                                                                                < isSnowflake
                                                                                                    < isSnowflake
                                                                                                        < isSnowflake
                                                                                                            < isSnowflake
                                                                                                                < isSnowflake
                                                                                                                    < isSnowflake
                                                                                                                        < isSnowflake
                                                                                                                            < isSnowflake
                                                                                                                                < isSnowflake
                                                                                                                                    < isSnowflake
................................................................

```

```

drawSnowball
let snowball: Snowball
= new Snowball
[< nBird]
    < isBird
        < isBird
            < isBird
                < isBird
                    < isBird
                        < isBird
                            < isBird
                                < isBird
                                    < isBird
                                        < isBird
                                            < isBird
                                                < isBird
                                                    < isBird
................................................................

```

moveables.push(snowball)

update

backgroundData:imageData

```
cnv2.drawImageData(backgroundData, 0, 0)
```

```
(moveables[7] → move(), draw())
```

```
(all birds.hungry → arrivedAtDestination() →
```

```
all birds.isHit → render  
> bird.eat → score += 10  
> bird.fly → score += 20
```

```
snowball → arrivedAtDestination() →
```

```
let paragraph = document.querySelector("#stone")
```

```
paragraph.innerHTML =  
"Dein Score beträgt: " + score
```

endOfGame

```
let name: any = prompt  
(`"Dein Score: " + score, "Gebe deinen Namen ein")
```

sendEntry(name, score) →

sendEntry

```
-name: string, -score: number
```

```
let query: string = "score=" + -score +  
"&name=" + -name  
let response: Response = await  
fetch(url + "?2" + query)
```

await response.text()

window.open([Link startete]) →

Highscore +=

```
let url: string = [Höhepunkte-Link]
```

```
install load-listener
```

load

handleLoad

event: Event

HandleRedirects

event: Event

let query: string = "command=retrieve"

```
let response: Response = await  
fetch(url + "?2" + query)
```

```
let responseText: string = await response.text()  
let highscore: number = JSON.parse(responseText)
```

OK → highscore.innerHTML = responseText