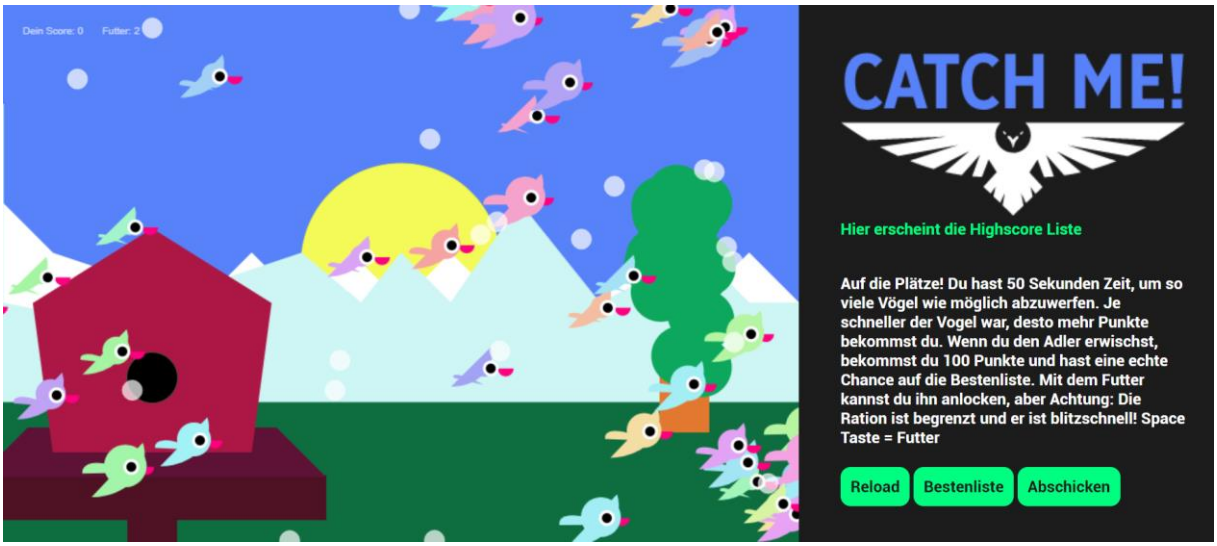
Entwicklung einer interaktiven Unterhaltungsanwendung

unter Verwendung von Typescript, Nodejs, MongoDB, HTML, und CSS



Technische und konzeptionelle Analyse der Anwendung „Catch me!"

Inhaltsangabe

1. Funktionale Analyse

1.1. Auswahl der Plattform

Die Unterhaltungsanwendung soll im Browser am PC dargestellt und gespielt werden können. Die Plattform wurde dabei anhand der folgenden Kriterien ausgewählt:

1. **Große Anwendungsfläche:** Die Anwendung soll farbenfroh und visuell ansprechend gestaltet werden. Aufgrund seiner großen Anzeigefläche bietet sich daher der PC an.
2. **Leichte Bedienung:** Es müssen für die Steuerung keine Extra-Buttons erstellt werden, da das Eingabemedium die Maus und das Keyboard schon vorhanden sind.
3. **Mehr Konzentration:** Wenn der Nutzer vor dem PC spielt, wird er nicht so leicht abgelenkt. Da es bei der Anwendung auf Reaktion und Genauigkeit ankommt, eignet sich ein mobiles Endgerät weniger, da Smartphones/Tablets häufig in Situationen genutzt werden, in denen man abgelenkt wird (Beim Warten, in der Vorlesung).

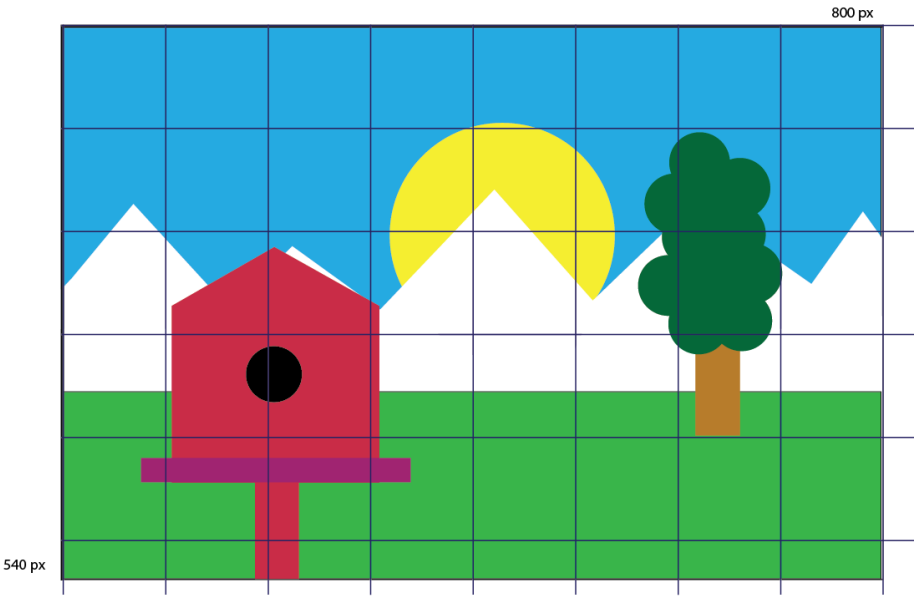
1.2. Gestaltung: Skizzen der Canvas-Elemente

1.2.1. Beschreibung des gewünschten visuellen Eindrucks

Die Anwendung soll visuell ansprechend sein. Mithilfe von bunten/pastelligen Farben soll ein fröhliches Bild entstehen. Die Gestaltung soll mithilfe von einfachen geometrischen Formen umgesetzt werden, da diese simpel und ohne viel Code mit Canvas zu malen sind. Die Gestaltung sollte ein relativ hohes Abstraktionslevel haben und real existierende Dinge als einfache Elemente darstellen.

1.2.2. Skizze des kompletten Canvas

Der Canvas soll eine winterliche Szenerie darstellen.
Es wird ein Hintergrundbild mit Bergen (a), einer Wiese (b), einem Baum (c), einer Sonne (d) und einem Vogelhaus (e) kreiert.
Bei der Gestaltung des Hintergrunds sind daher die Berge schneebedeckt.
Außerdem werden Schneeflocken (f) fallen (siehe Punkt 1.2.3.1.).



Der Canvas streckt sich über eine Größe von 540px (x-Achse) und 800px (y-Achse), da er so auf den meisten PCs vollständig im Browser (bei Zoom = 100%) erscheint und nicht skaliert werden muss.

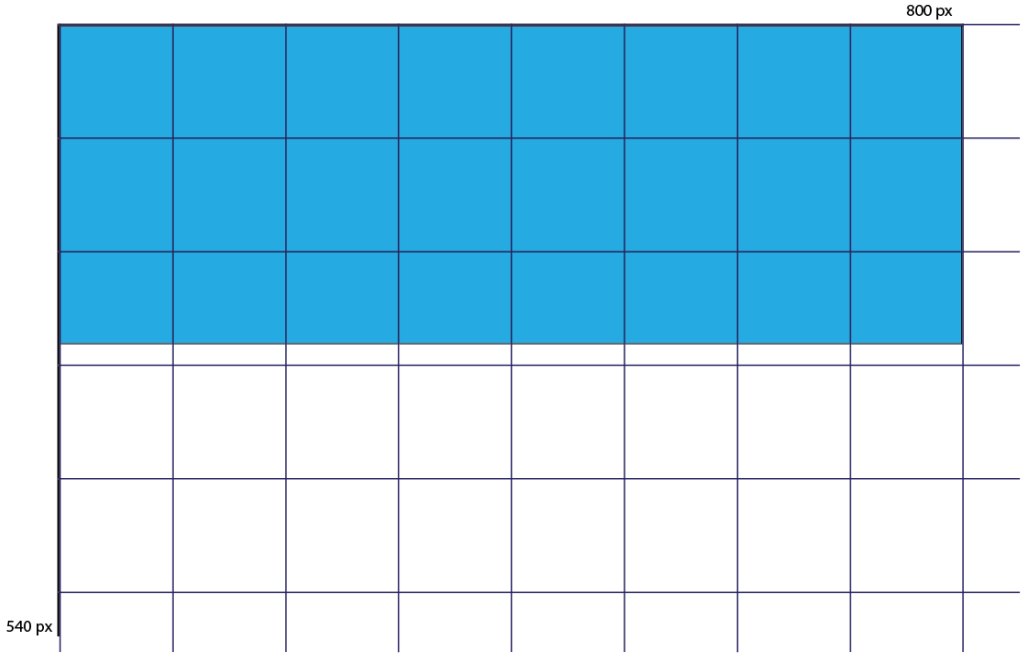
Neben dem Canvas soll sich eine Fläche mit einem Logo, einer kurzen Spielanleitung und zwei Buttons befinden. Die Buttons sind grün eingefärbt und haben eine weiße Schrift. Sie ermöglichen es dem Nutzer

Text der Spielanleitung:
Auf die Plätze! Du hast 50 Sekunden Zeit, um so viele Vögel wie möglich abzu werfen. Je schneller der Vogel war, desto mehr Punkte bekommst du. Wenn du den Adler erwischst, bekommst du 100 Punkte und hast eine echte Chanc e auf die Bestenliste. Mit dem Futter kannst du ihn anlocken, aber Achtung: D ie Ration ist begrenzt und er ist blitzschnell! Space Taste = Futter.

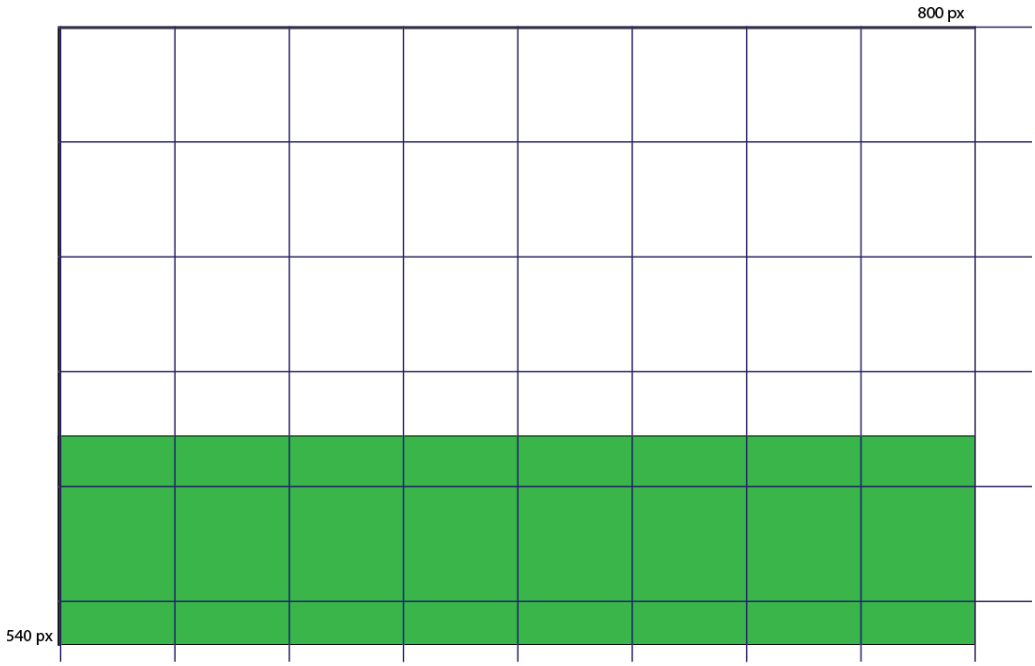
1.2.3. Skizzen der einzelnen Elemente

1.2.3.1. Statische Objekte (Static Objects)

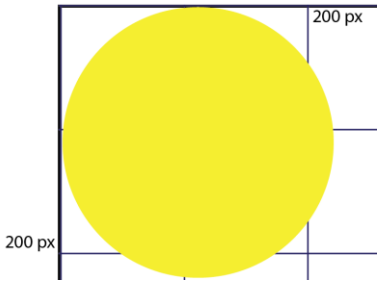
a) Himmel



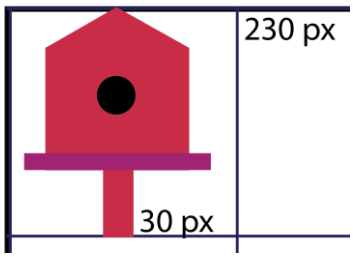
b) Wiese



c) Sonne

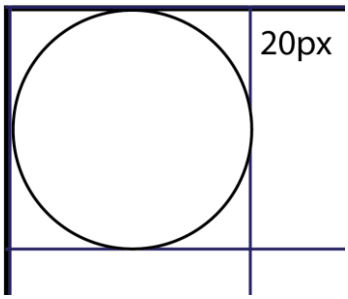


d) Vogelhaus

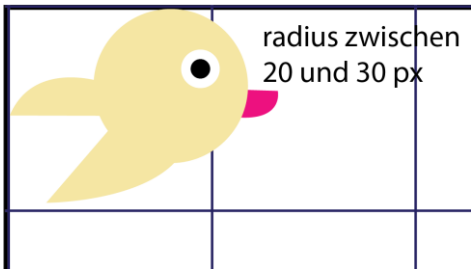


1.2.3.2. Bewegte Objekte (Moving Objects)

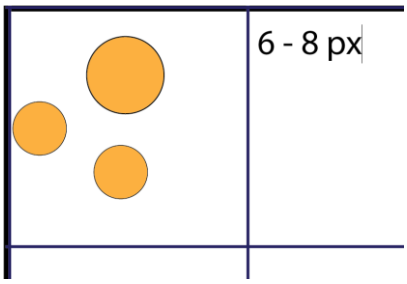
e) Schneeball



f) Vogel



g) Futter



1.3. Ablauf des Spiels

1.3.1. Aktionen des Users

Beim Laden des Spiels kann der Nutzer sofort beginnen.
Verschiedene Vögel fliegen von links nach rechts über die Fläche.
Der User kann per Klick einen Schneeball abwerfen.
Trifft er die Vögel, bekommt er in Abhängigkeit von der Geschwindigkeit des Vogels Punkte.
Er kann die Vögel jederzeit anlocken, in dem er Futter wirft. Nach drei Futterrationen ist der Vorrat aufgebraucht.
Beim Werfen des Futters erscheint ein Adler im Sturzflug, der beim Abwerfen 100 Punkte extra gibt, aber er ist schneller als die anderen Vögel.
Nach 50 Sekunden ist das Spiel vorbei und der Nutzer wird aufgefordert, seinen Namen einzugeben. Dieser Name wird dann die High Score-Liste eingetragen.

1.4. Ziel des Spiels

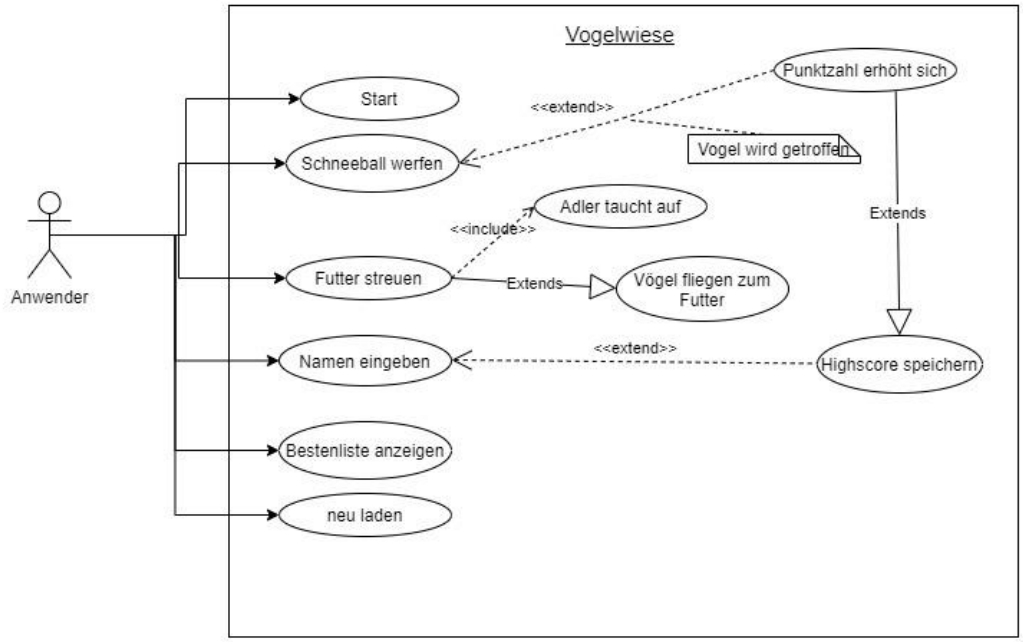
Das Ziel des Spiels ist es innerhalb der ablaufenden Zeit soviel Punkte wie möglich zu erreichen. (Möglich sind ca. 20 der Vögel und 2 Adler) Der Canvas sollte nie ganz leer sein, damit man immer eine Motivation hat weiter zu üben.

2. Technische Analyse

2.1. Anwendungsfalldiagramm

2.1.1. Use-Case-Diagramm

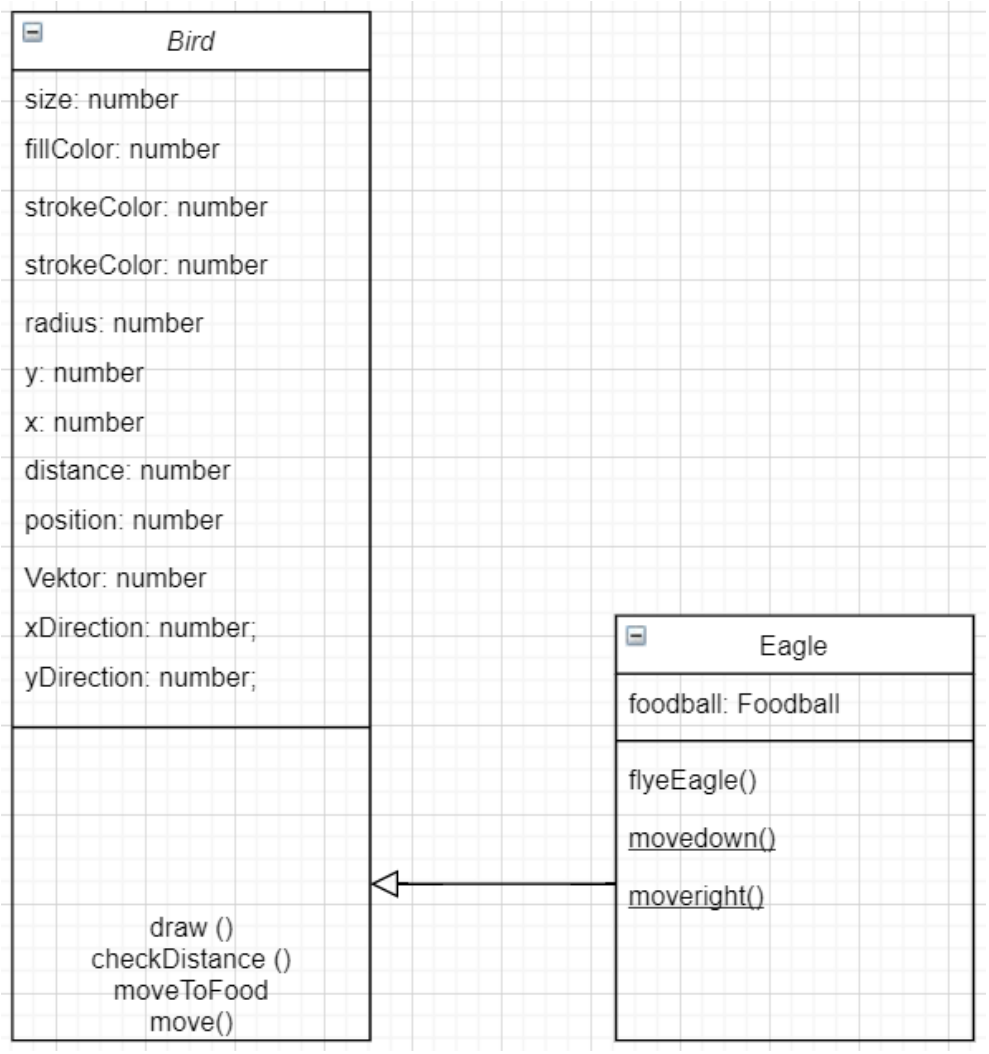
Der Nutzer kann Vögel abwerfen, Adler abwerfen, Futter fallen lassen, die Bestenliste einsehen und das Fenster neu laden.



2.2. Klassendiagramme und Vererbung

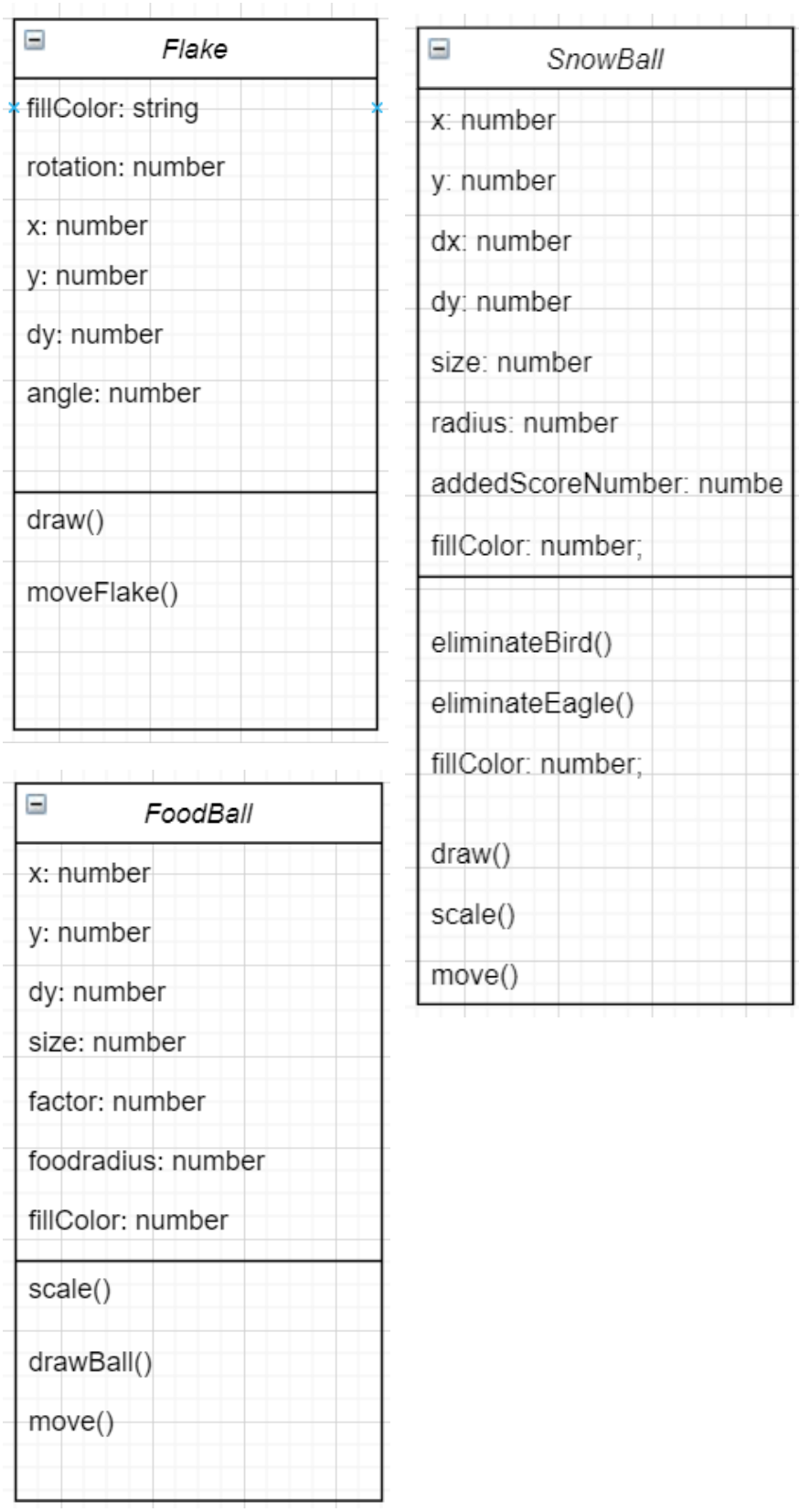
2.2.1. Hinweis zur Vererbung von Bird und Eagle

Der Eagle (Adler) ist eine Unterart des Vogels, er bewegt sich zwar anders und schneller und sieht auch anders aus (wird anders gezeichnet), aber er hat alle Eigenschaften, die ein Vogel auch hat (u.a. das gleiche Ziel: Futter)
Klassendiagramm:



2.2.2. Klassendiagramme

Die restlichen Klassen werden lediglich erstellt, aber keine weitere Vererbung ist nötig.



2.3. Aktivitätsdiagramme

Globale Variablen werden definiert /
DOM Content Loaded ruft Initialisierung auf:

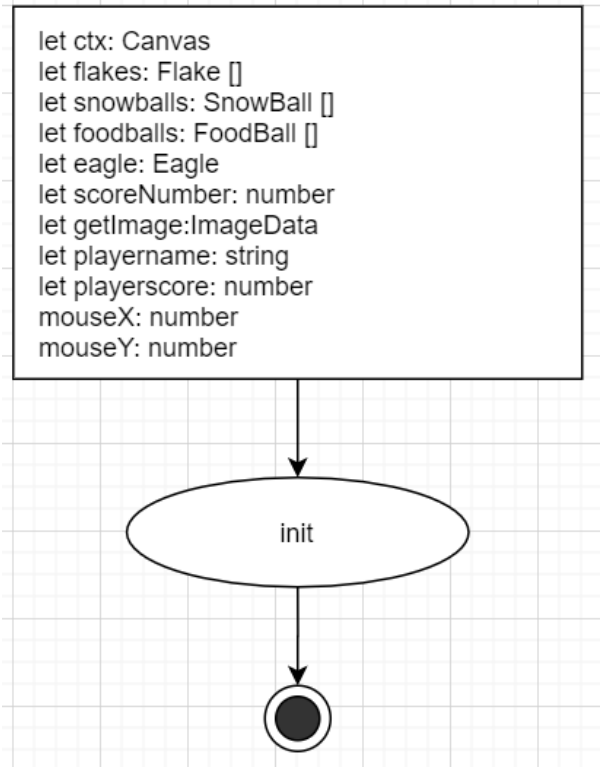
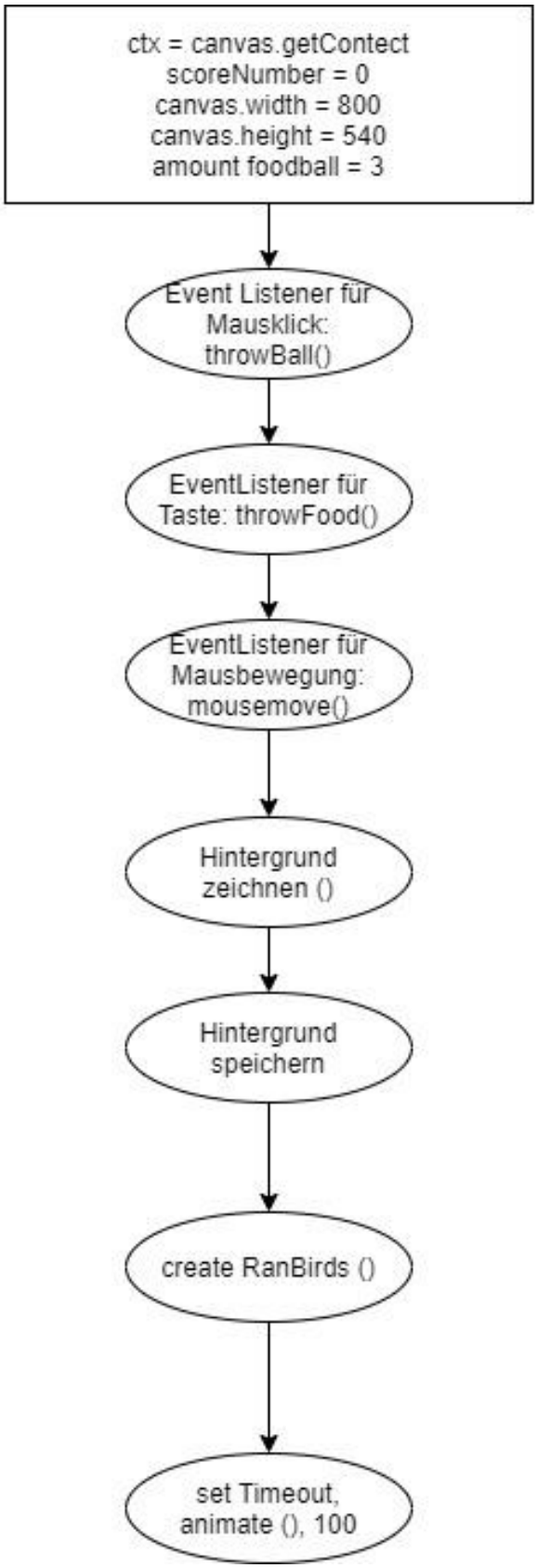
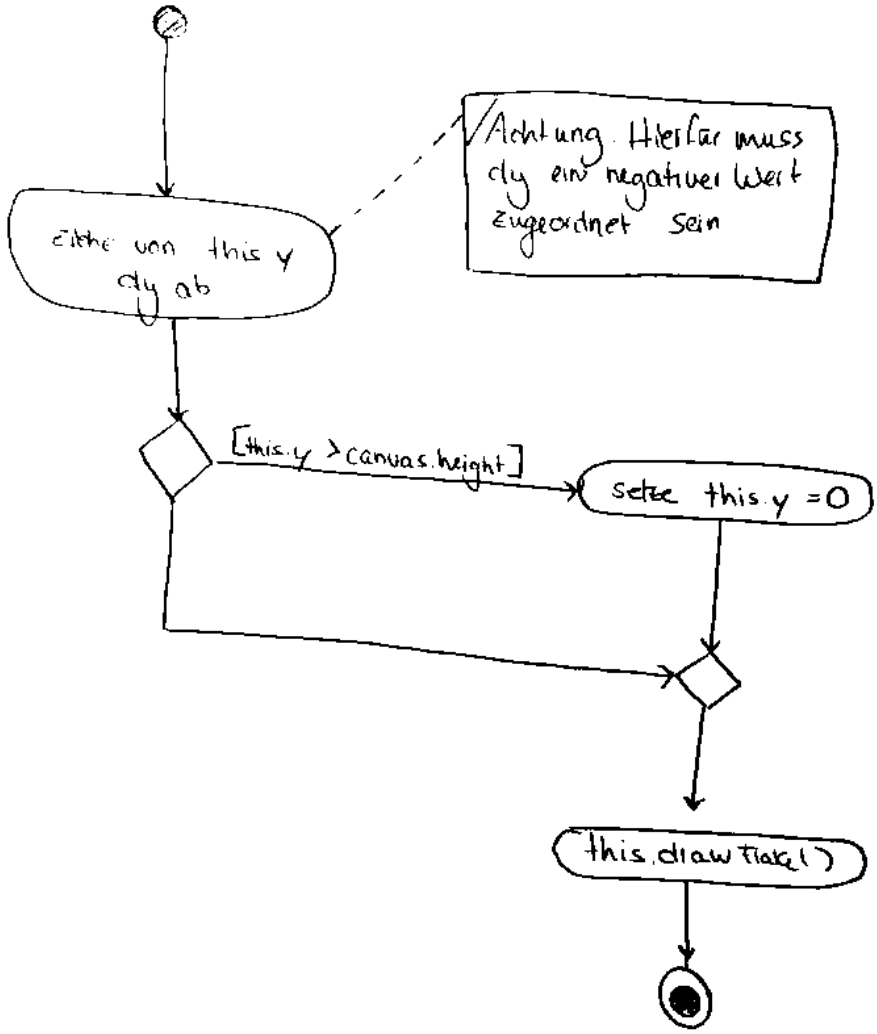


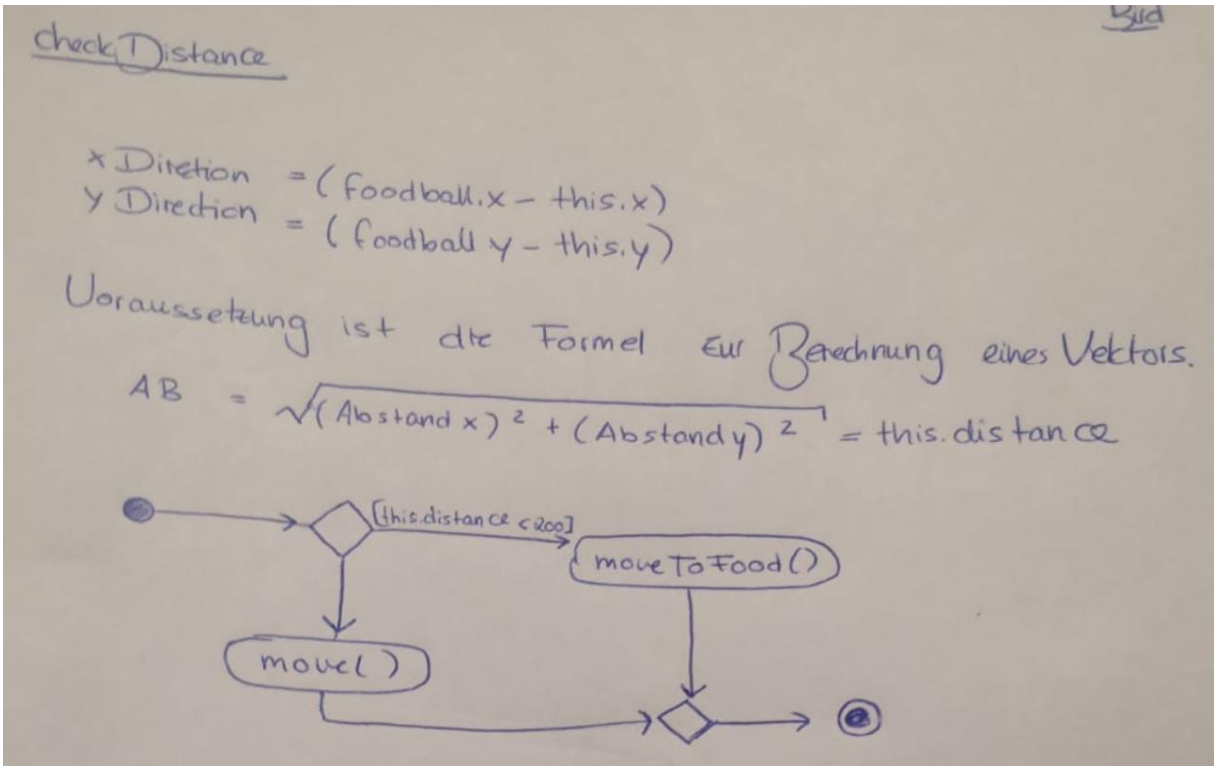
Diagramm der Initialisierung:

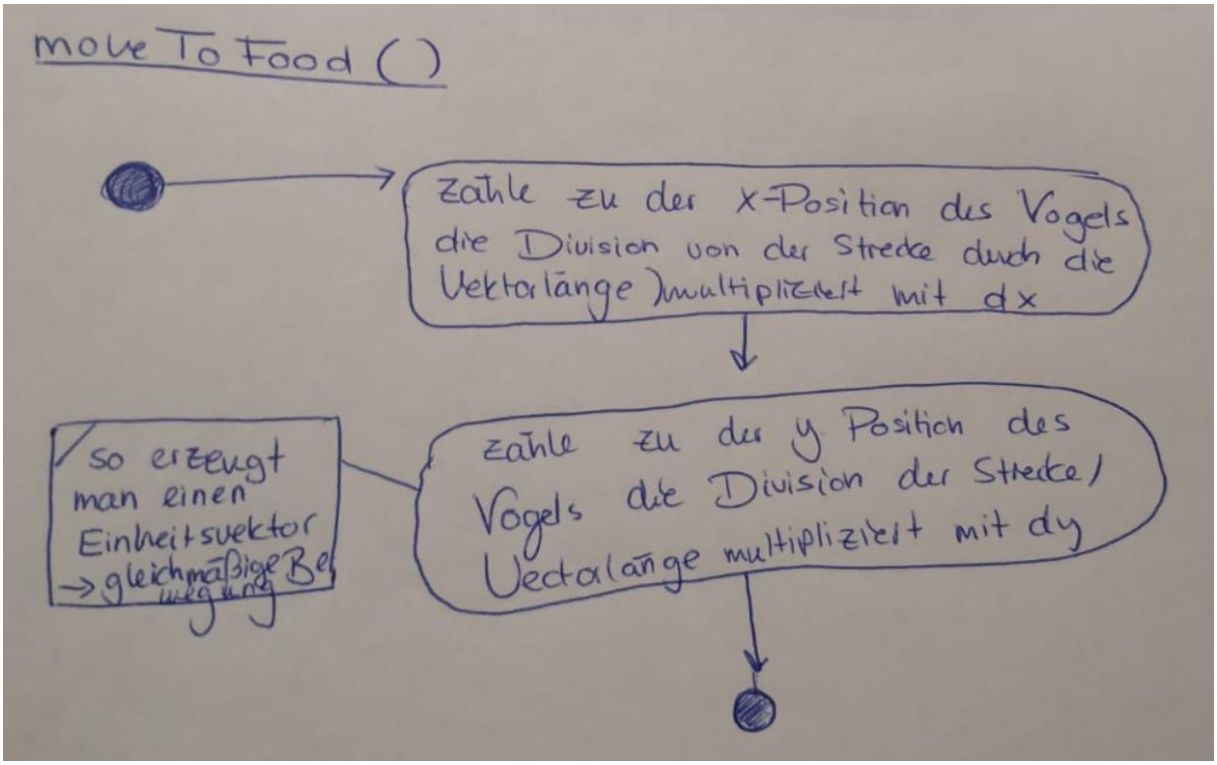

ctx = canvas.getContect
scoreNumber = 0
canvas.width = 800
canvas.height = 540
amount foodball = 3
Event Listener für Mausklick: throwBall()
EventListener für Taste: throwFood()
EventListener für Mausbewegung: mousemove()
Hintergrund zeichnen ()
Hintergrund speichern
create RanBirds ()
set Timeout, animate (), 100

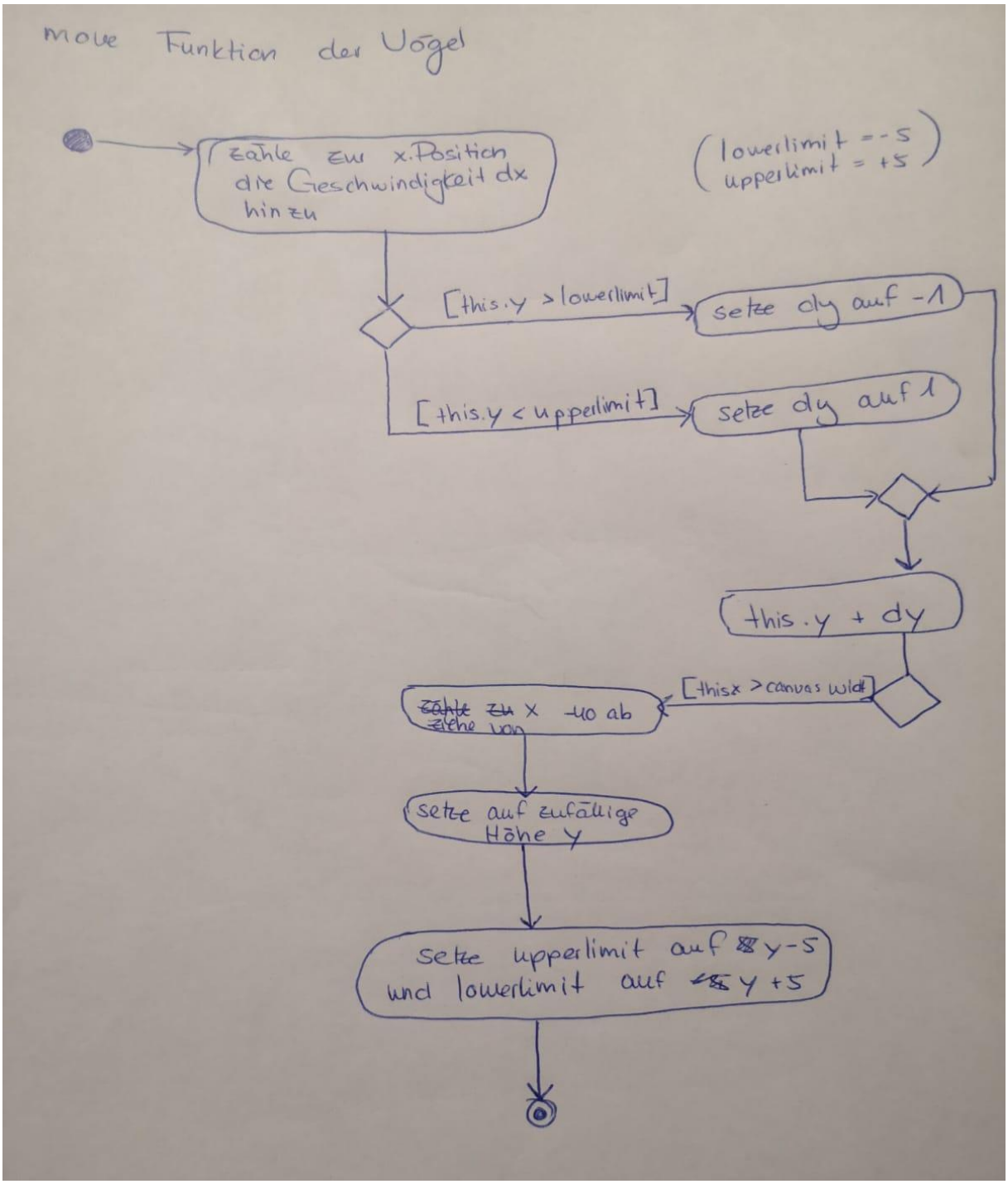
Aktivitätsdiagramm für die move-Methode der Flake-Klasse:



Aktivitätsdiagramme der Bird-Klasse:

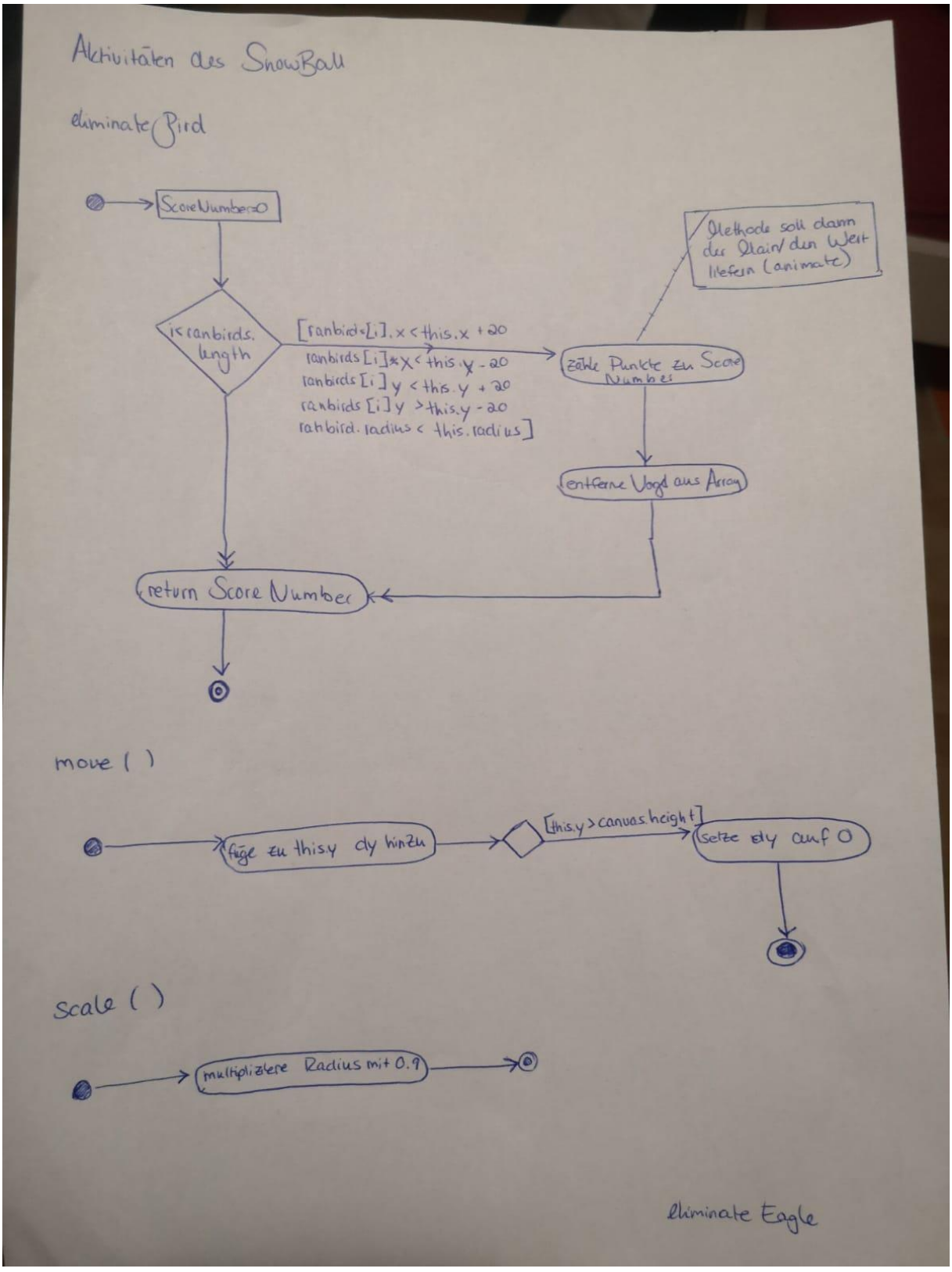

checkDistance
xDirection = (foodball.x - this.x)
yDirection = (foodball.y - this.y)
Voraussetzung ist die Formel zur Berechnung eines Vektors.
AB = √((Abstand x)² + (Abstand y)²) = this.distance
[this.distance < 200]
moveToFood()
move()


moveToFood()
Zähle zu der x-Position des Vogels die Division von der Strecke durch die Vektorlänge multipliziert mit dx
Zähle zu der y Position des Vogels die Division der Strecke/ Vectorlänge multipliziert mit dy
so erzeugt man einen Einheitsvektor → gleichmäßige Bewegung

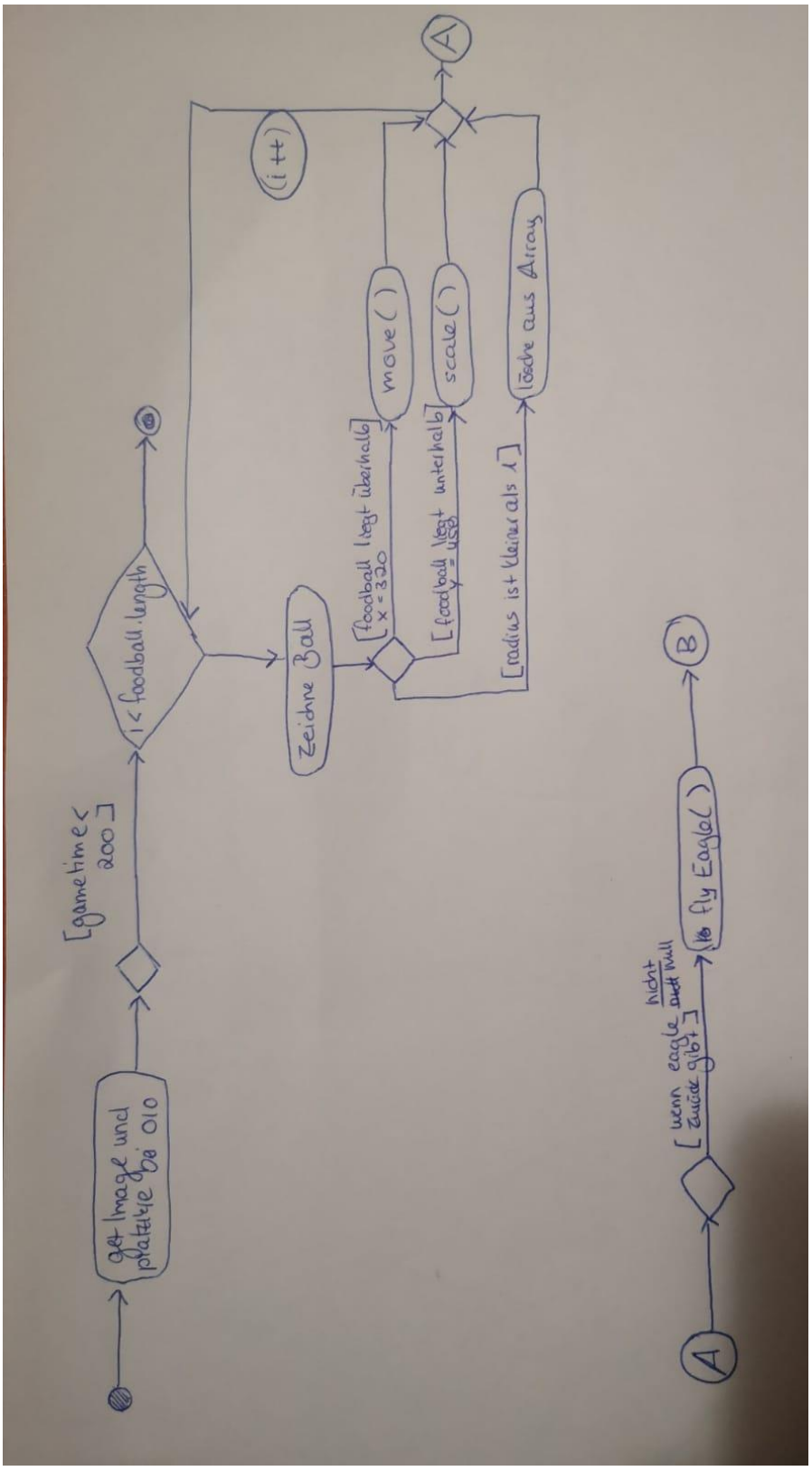

move Funktion der Vögel
zähle zu x-Position die Geschwindigkeit dx hinzu
(lowerlimit = -5
upperlimit = +5)
[this.y > lowerlimit]
setze dy auf -1
[this.y < upperlimit]
setze dy auf 1
this.y + dy
[this.x > canvas.width]
zähle zu x -40 ab
setze auf zufällige Höhe y
setze upperlimit auf y-5 und lowerlimit auf y+5

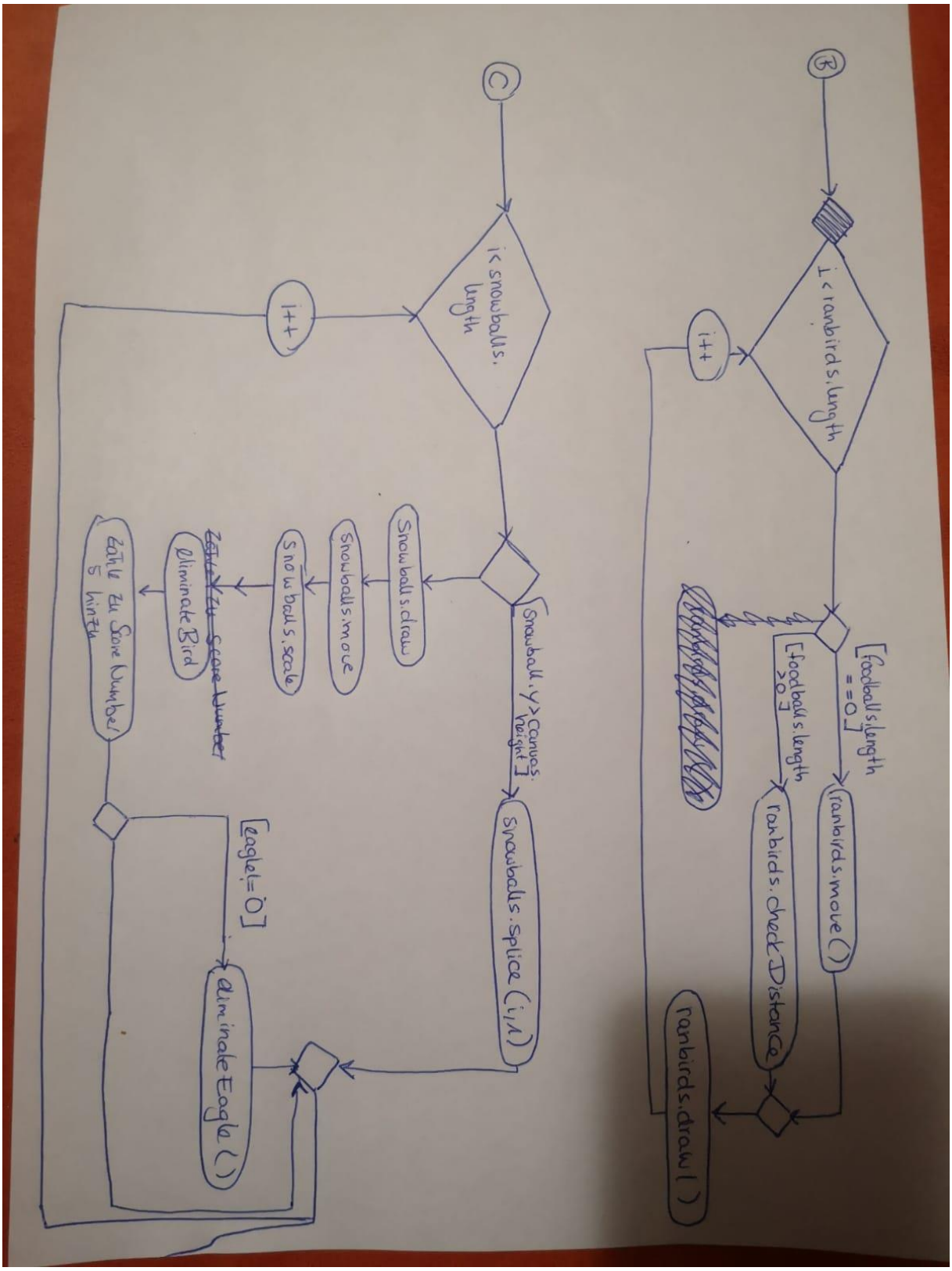
Aktivitäten der SnowBall-Klasse

Die Eliminate Methode gilt genauso für den Eagle mit dem Unterschied, dass vom Eagle nur ein Objekt erzeugt und kein Array erstellt wird.



Aktivitätsdiagramme der Animate-Funktion:




B
i < ranbirds.length
i++
[footballs.length == 0]
ranbirds.move()
[footballs.length > 0]
ranbirds.checkDistance
ranbirds.draw()
C
i < snowballs.length
i++
Snowball, y > canvas.height
Snowballs.splice(i,1)
Snowballs.draw
Snowballs.move
Snowballs.scale
eliminateBird
Eagle zu Score hinzu
[eagle!=0]
eliminateEagle()

2.4. Serveranbindung

2.4.1. Domänenübergreifendes Aktivitätsdiagramm