

Nutzererlebnis

Dieses Spiel ist eines für Jung und Alt. Man braucht keine bestimmten Skills, es ist einfach zu verstehen und es macht Spaß sich mit anderen Spielern zu messen. Erzielt man einen hohen Score bzw. findet sich in der Highscore-Liste weit oben, hat der Spieler ein positives Erlebnis. Allgemein sich einfach Mal ein bisschen ablenken und hat dabei Spaß ein lustiges Game zu spielen.

Festlegung der Plattform

Für das Spiel wurde ein Grundkonzept entwickelt. Bei der Überlegung wie man dieses umsetzen möchte, war es die einfachste Lösung den PC als Plattform zu wählen. Dieser hat mehr Eingabemöglichkeiten, die während dem Spiel benutzt werden können. Deswegen mussten keine extra Buttons oder Ähnliches erstellt werden, sondern es können einfach die Maustasten für das Spielen der Anwendung verwendet werden.

Informationen für den Nutzer

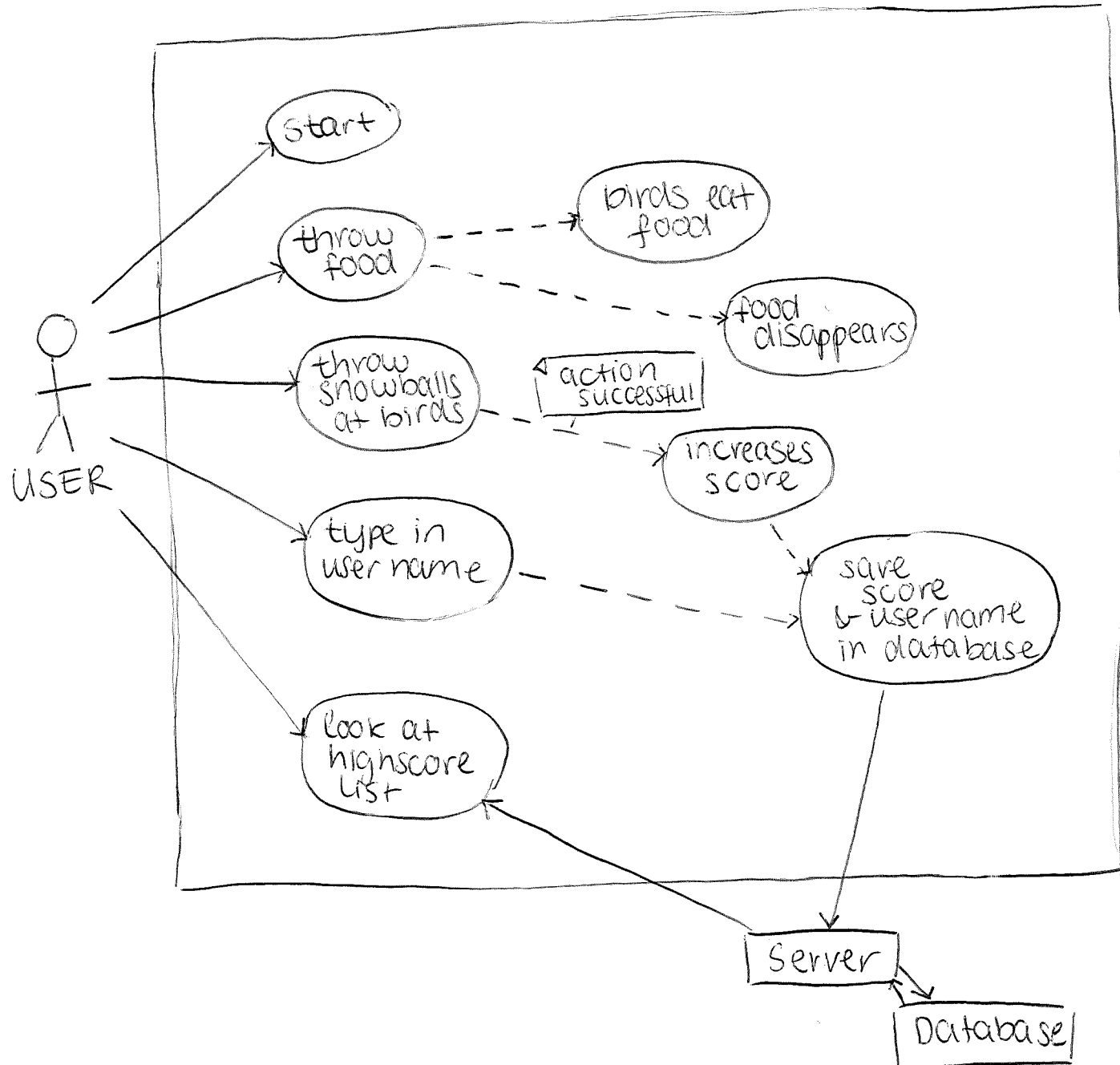
Vor und nach Ende des Spiels kann der Nutzer die Highscore-Liste einsehen. Man kann sich mit anderen Spielern messen und es ist ersichtlich welchen Platz man in der Rangliste einnimmt. An dieser Stelle kann er das Spiel auch (neu) starten. Während des Spiels kann er seinen Score, die Anzahl von zur Verfügung gestellten Futteranzahl und die Anzahl noch verfügbarer Schneebälle.

Allgemeine Nutzung des Spiels

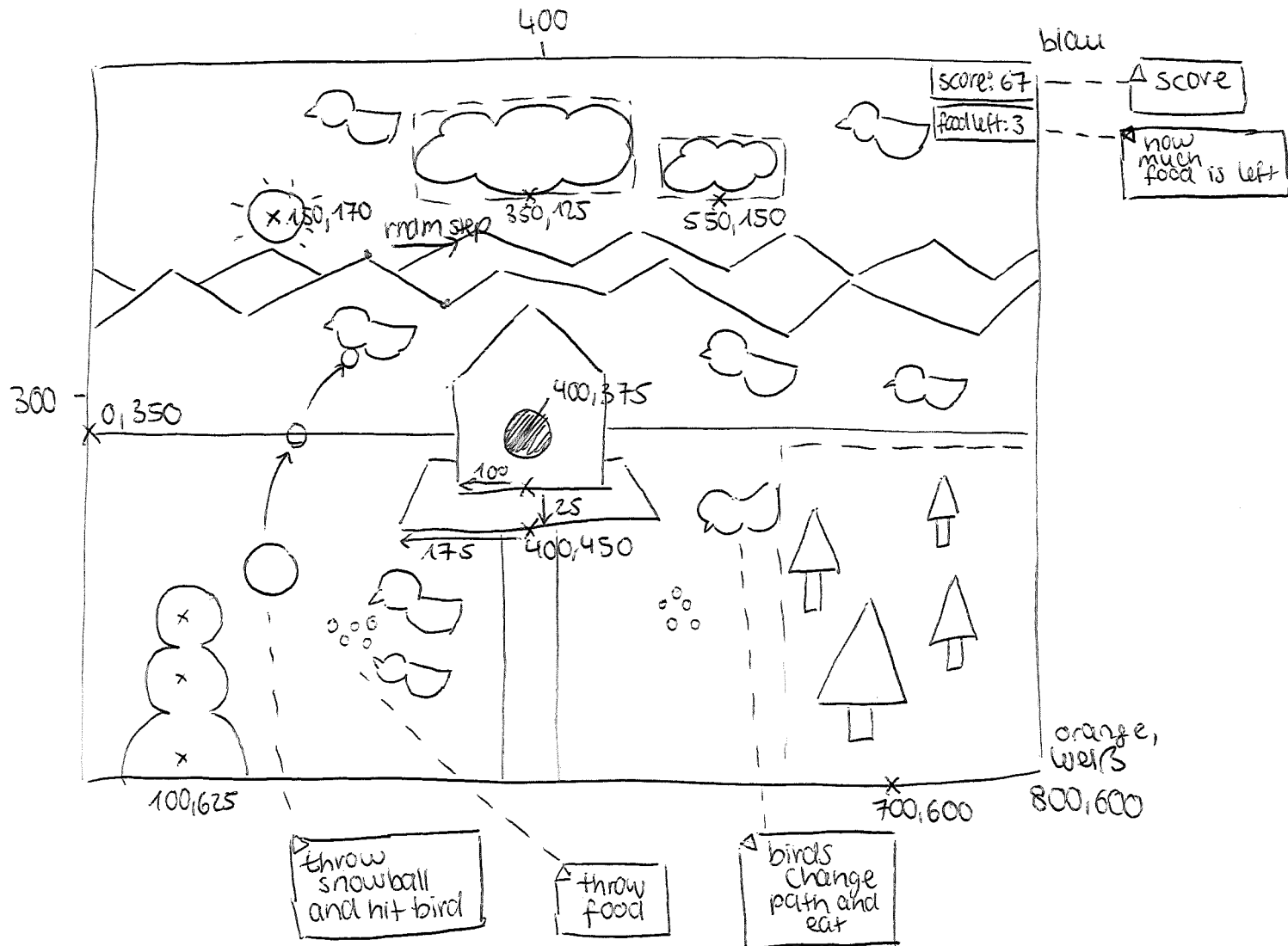
Der Spieler hat die Möglichkeit durch das Werfen eines Schneeballs Vögel zu treffen. Dies ist möglich mit einem Klick der linken Maustaste. Bei erfolgreicher Aktion verschwinden diese und 10 Punkte wandern auf den Score des Users. Bei Misserfolg wird ein Punkt abgezogen. Ist der Score nicht negativ wandert dieser auf eine Highscore-Liste auf dem alle erfolgreichen Spieler eingetragen sind.

Um Vögel anzulocken und damit einen höheren Score zu erlangen, kann man Futter werfen. Dies passiert durch Klick auf die mittlere oder rechte Maustaste. Insgesamt hat der Spieler drei Futterwürfe zur Verfügung. Das Futterwerfen ist auch nur im unteren Bereich der Canvas möglich.

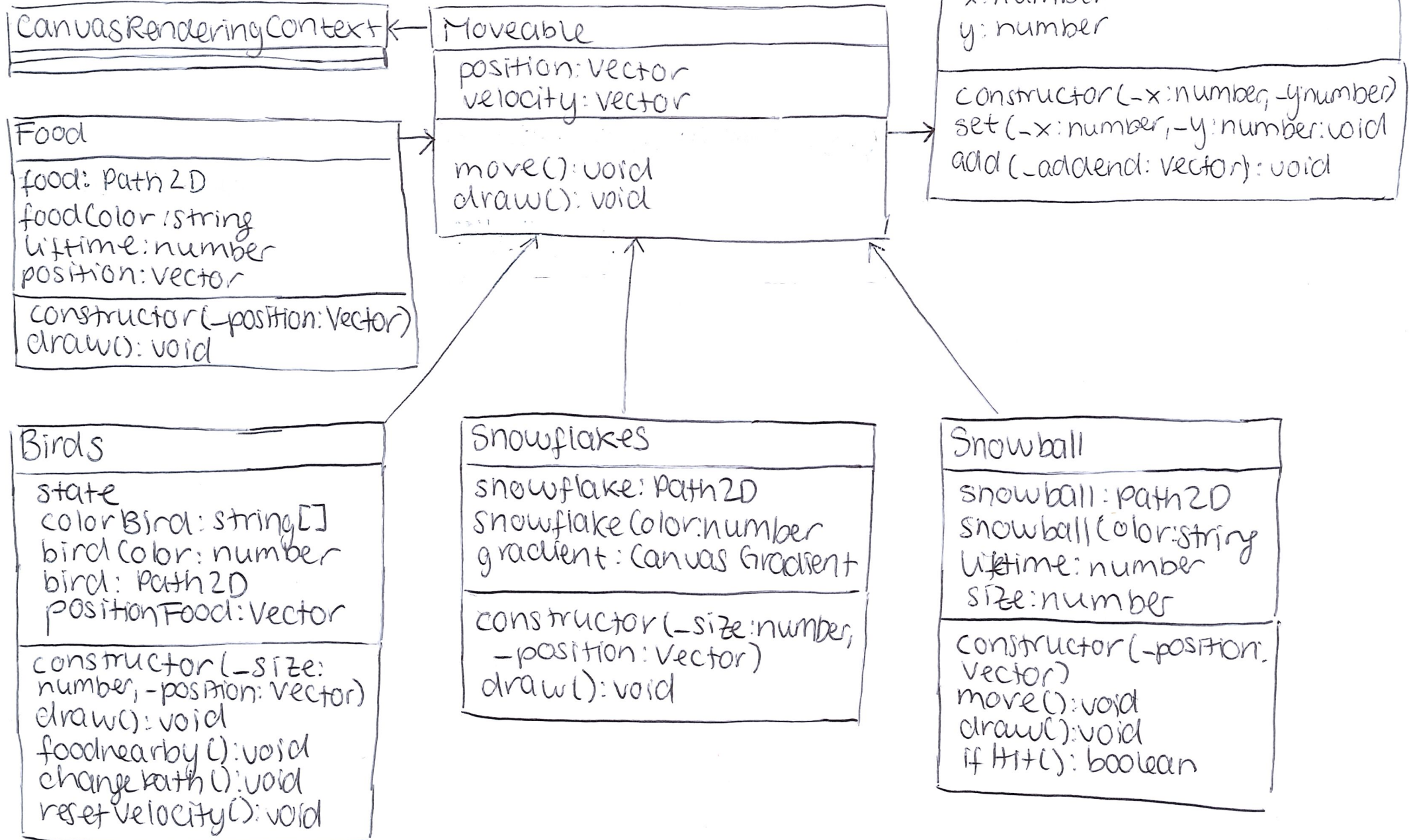
USE CASE DIAGRAM Vogelhaus



USER INTERFACE Vogelhaus

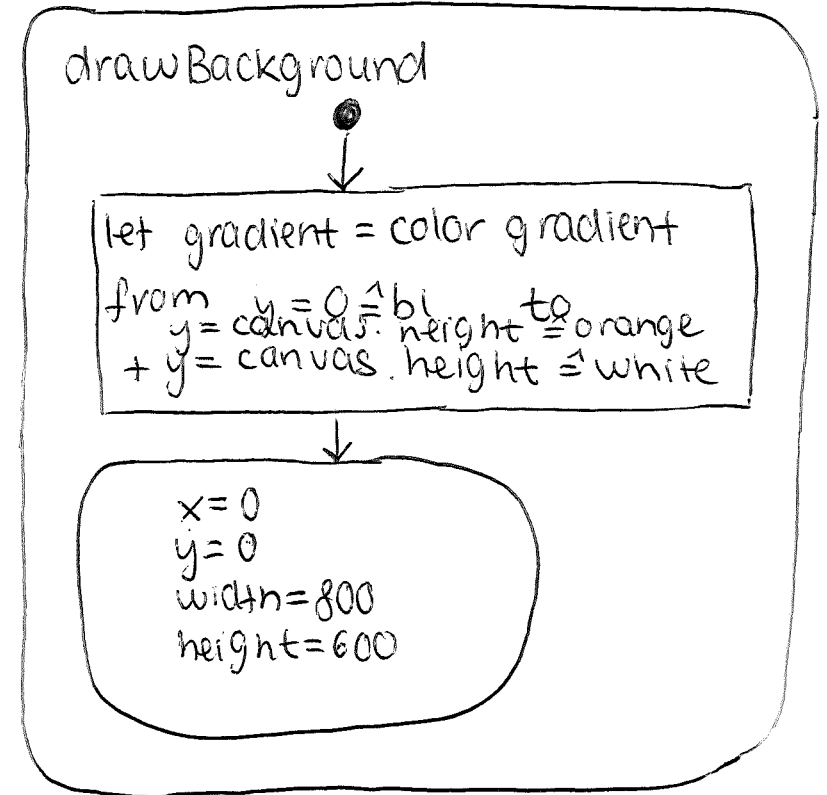
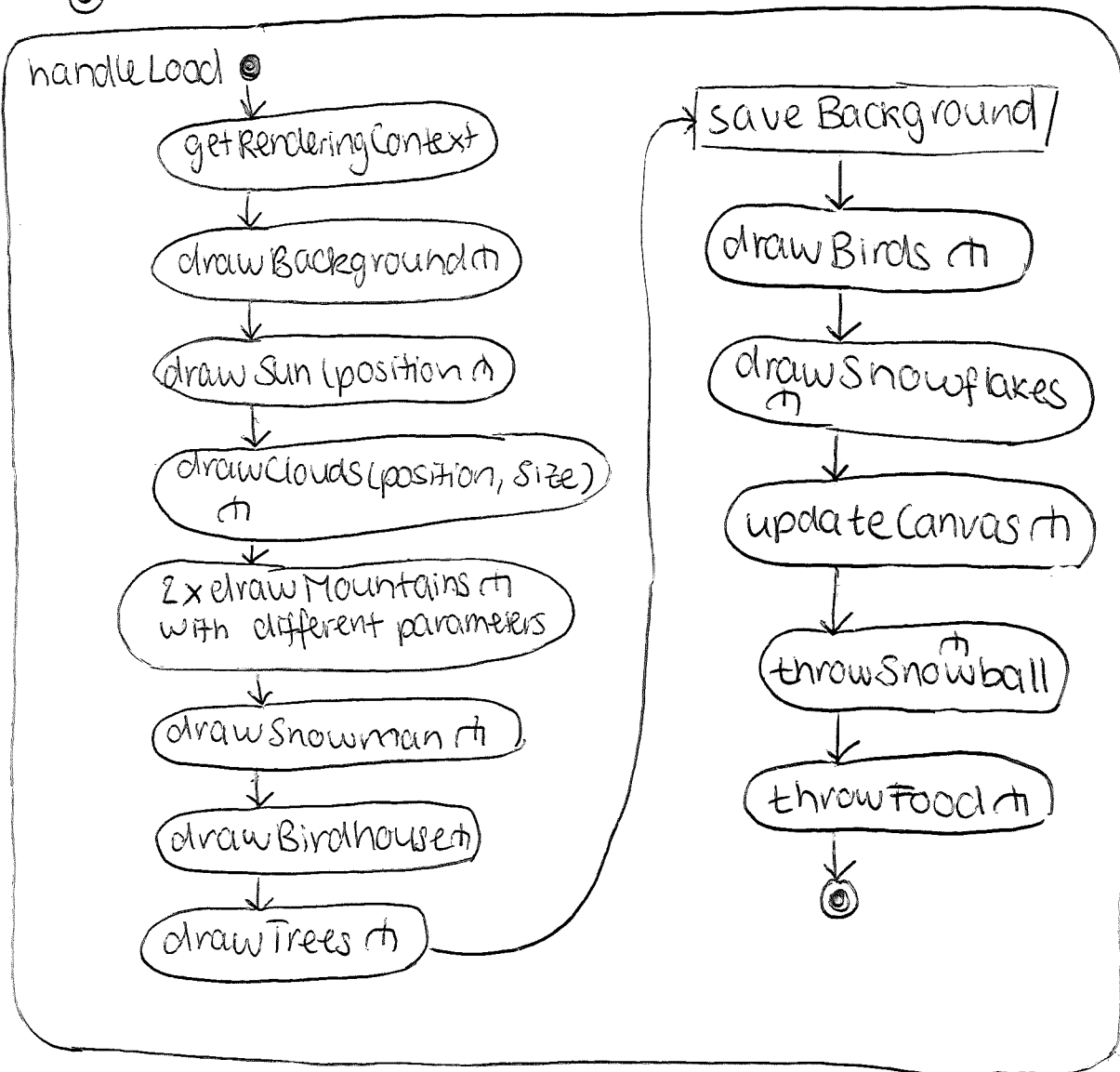
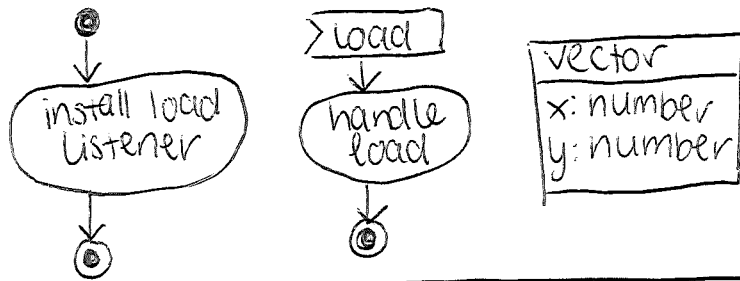


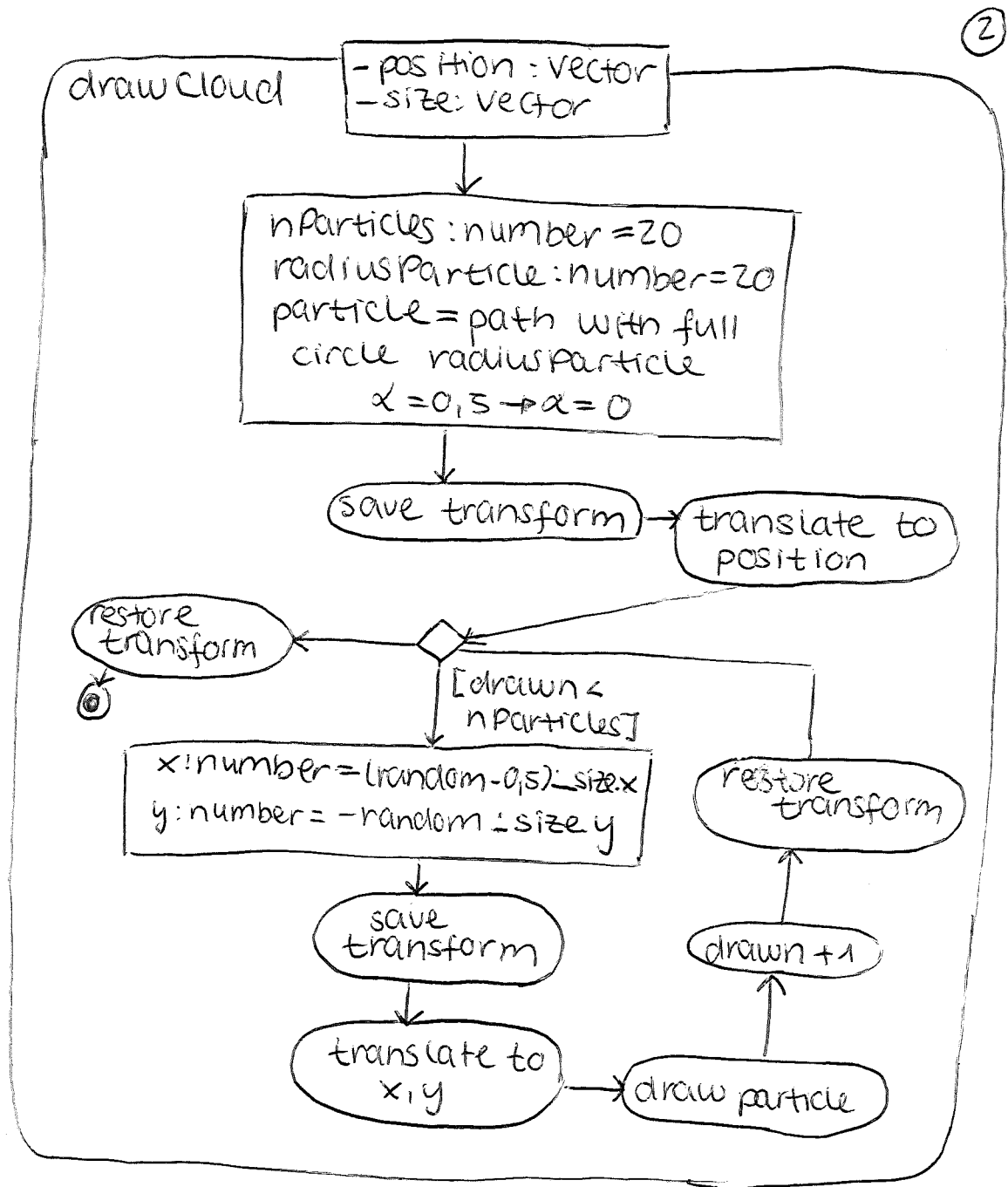
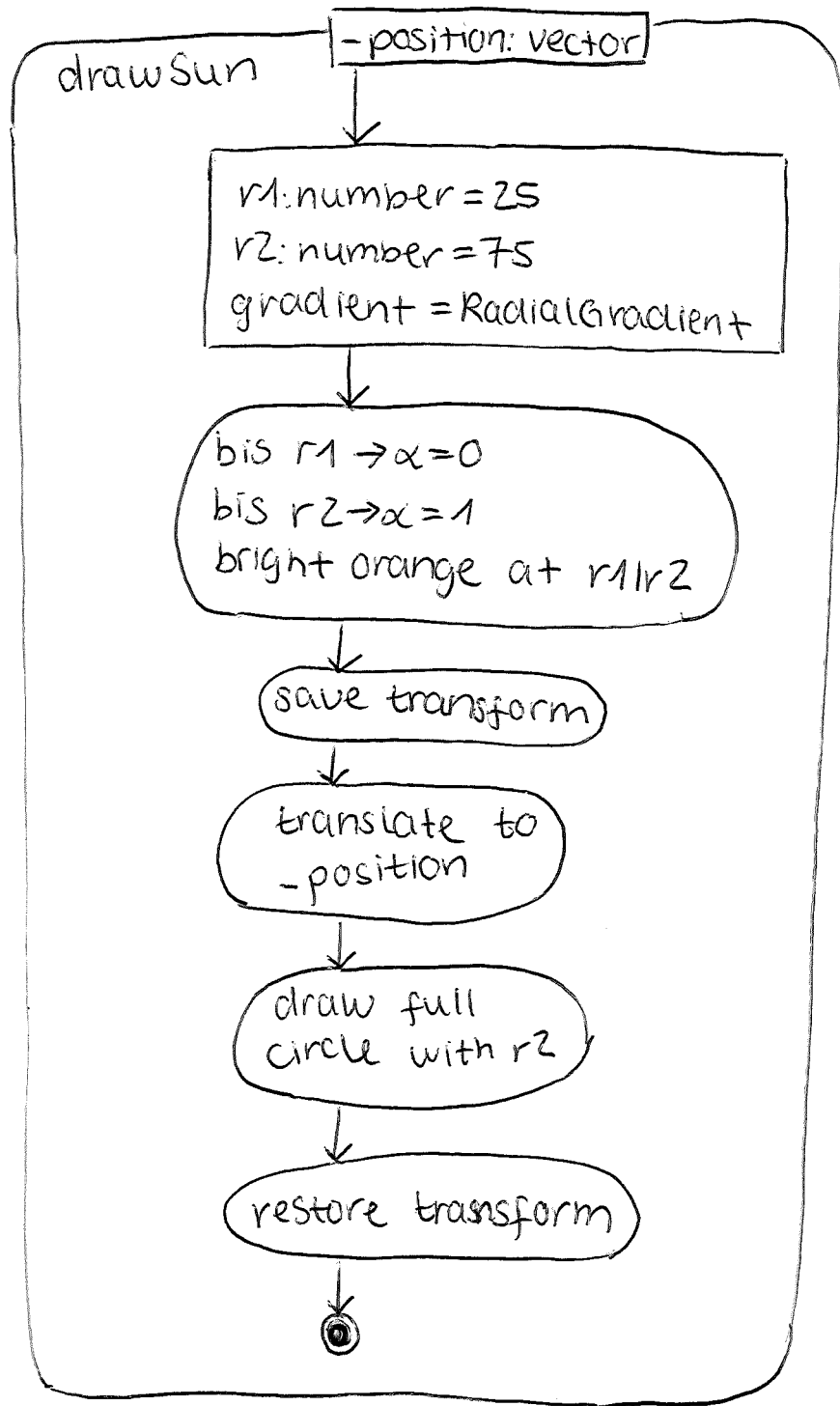
CLASS DIAGRAM Vogelhaus



ACTIVITY DIAGRAMS Vogelhaus

①

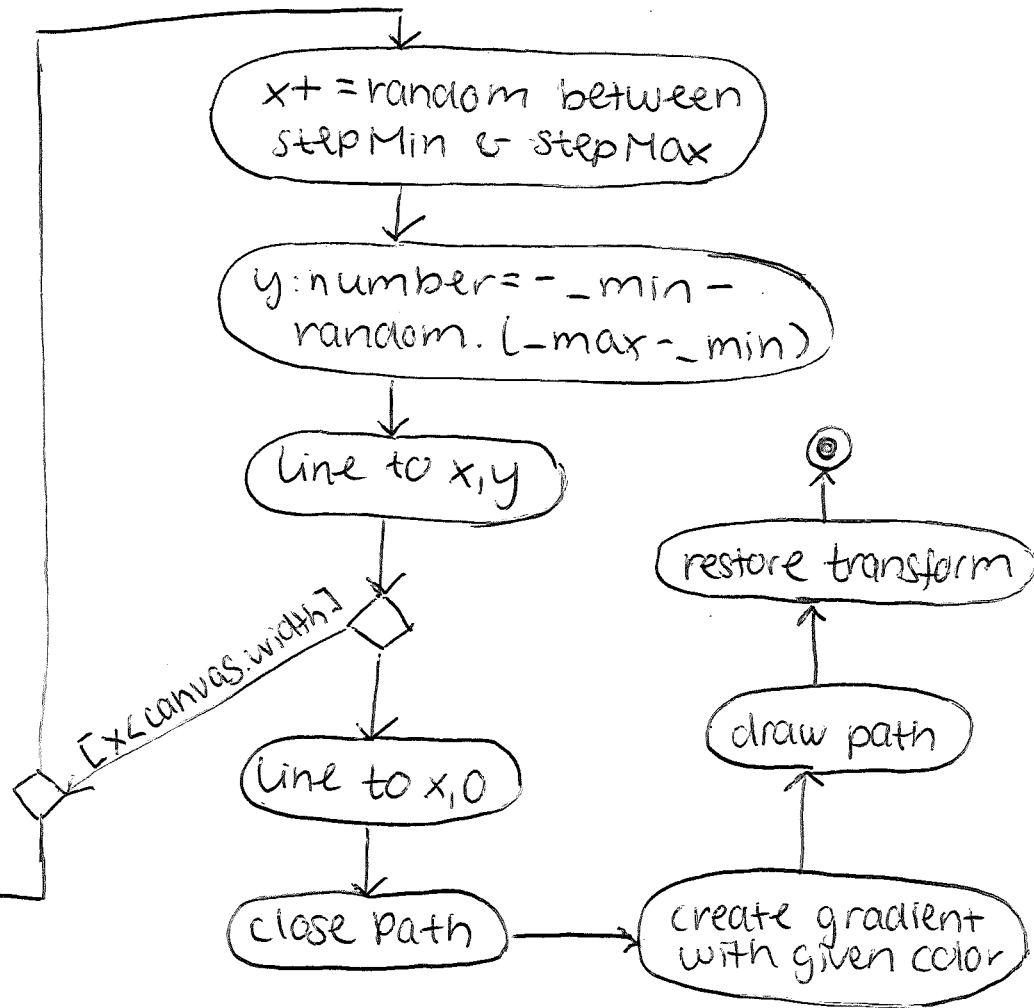
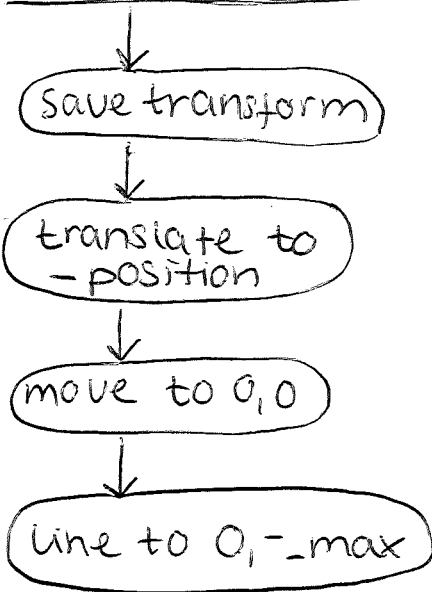


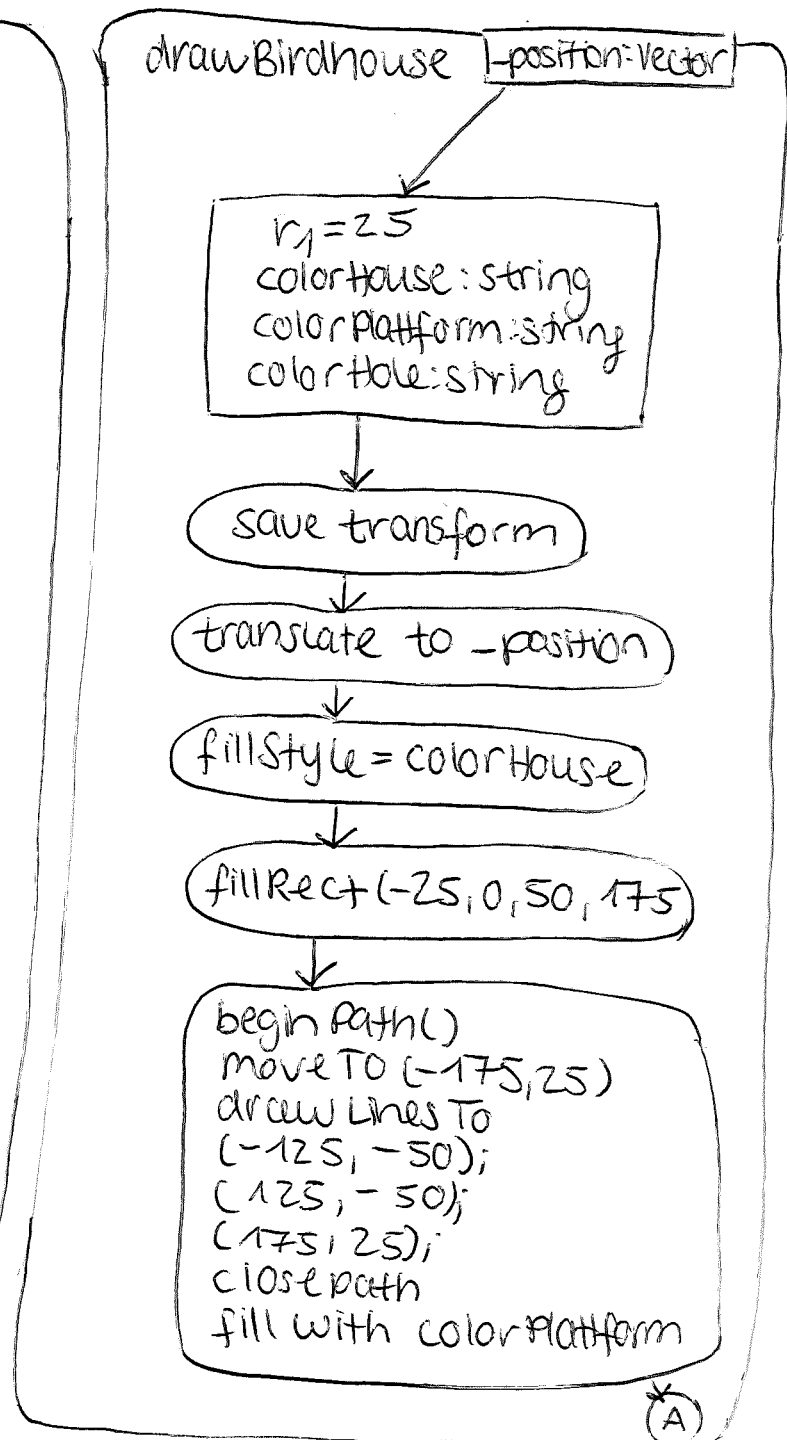
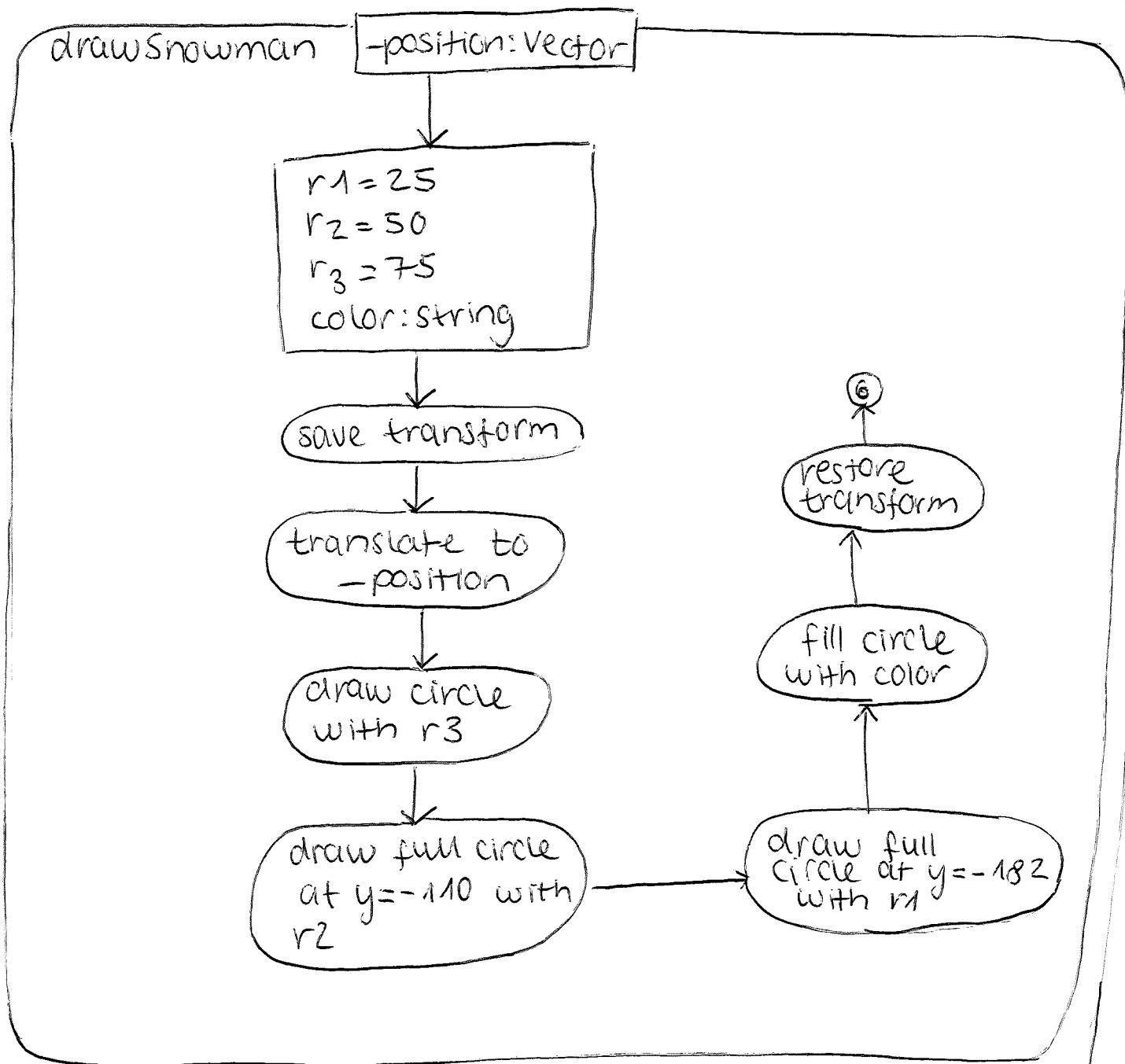


draw Mountains

- position: Vector
- min: number
- max: number
- colorLow: string
- colorHigh: string

stepMin: number = 10
stepMax: number = 50
x: number = 0





(A) → beginPath()
 moveTo(-100, 0)
 drawLinesTo
 (-100, -150)
 (0, -225)
 (100, -150)
 (100, 0)
 closePath()
 fill with colorHouse

draw full
 circle with
 r1 at y = -75

fill circle with
 colorHole

restore
 transform



drawTrees

- position: Vector
 - size: Vector

nTrees: number = 4
 colorTrunk: string
 colorCrown: string

save transform

translate to
 -position

restore
 transform



[drawn < nTrees]

scale

scale: number
 calculate scale
 depending on y

fill Rec

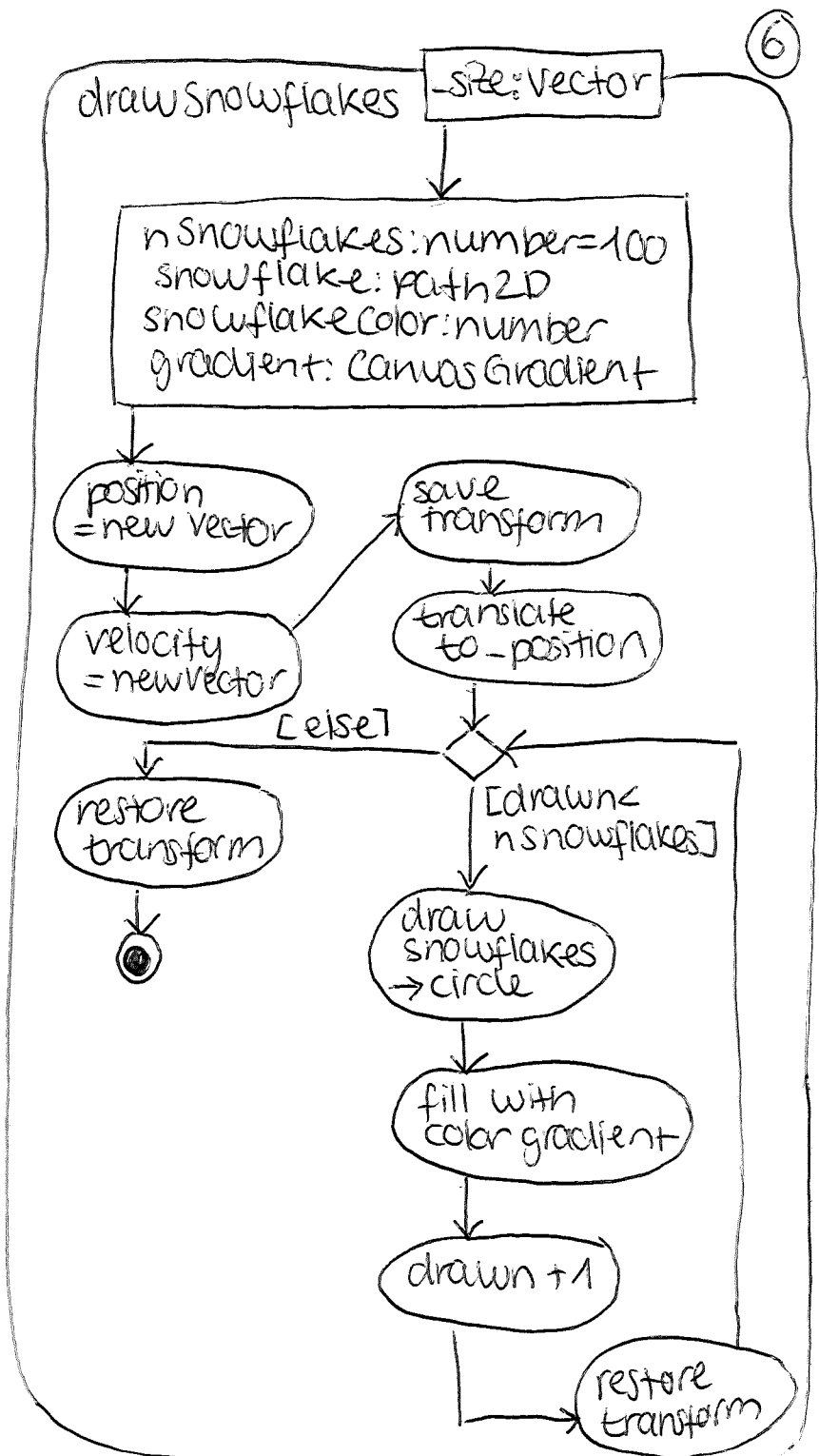
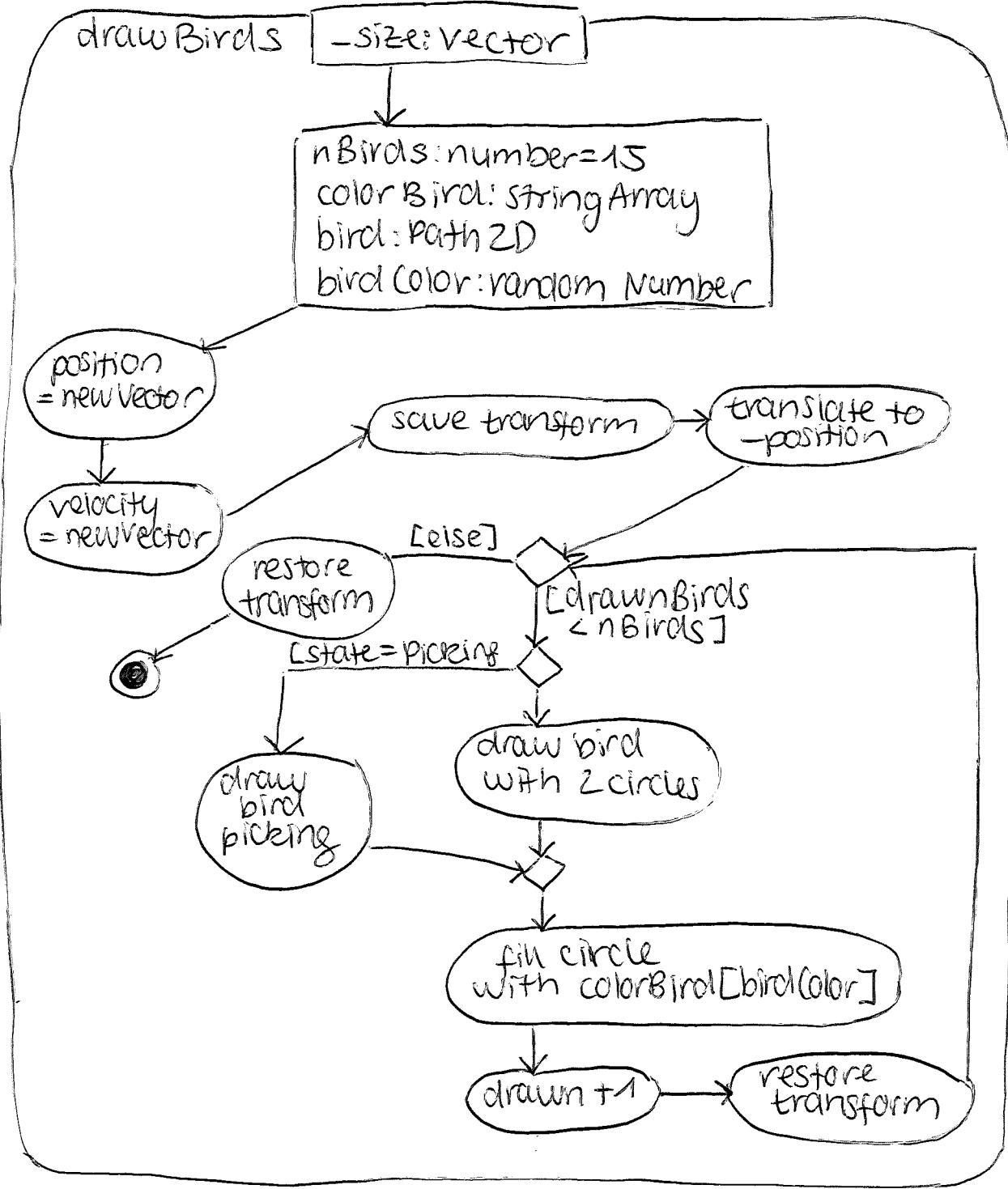
draw Crown

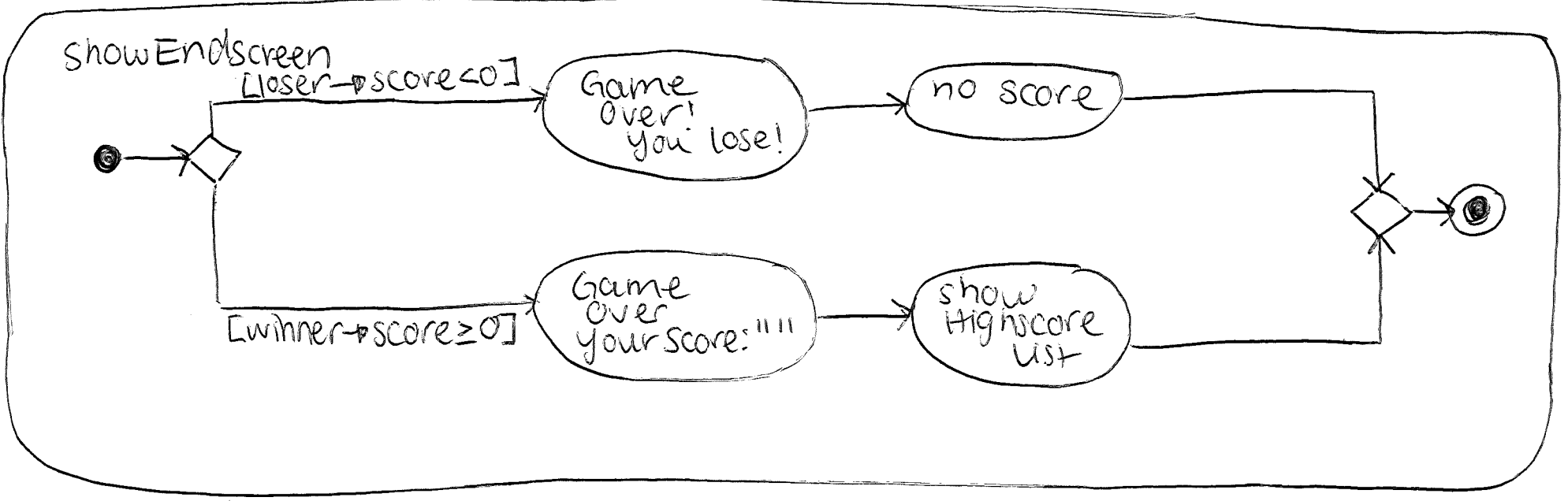
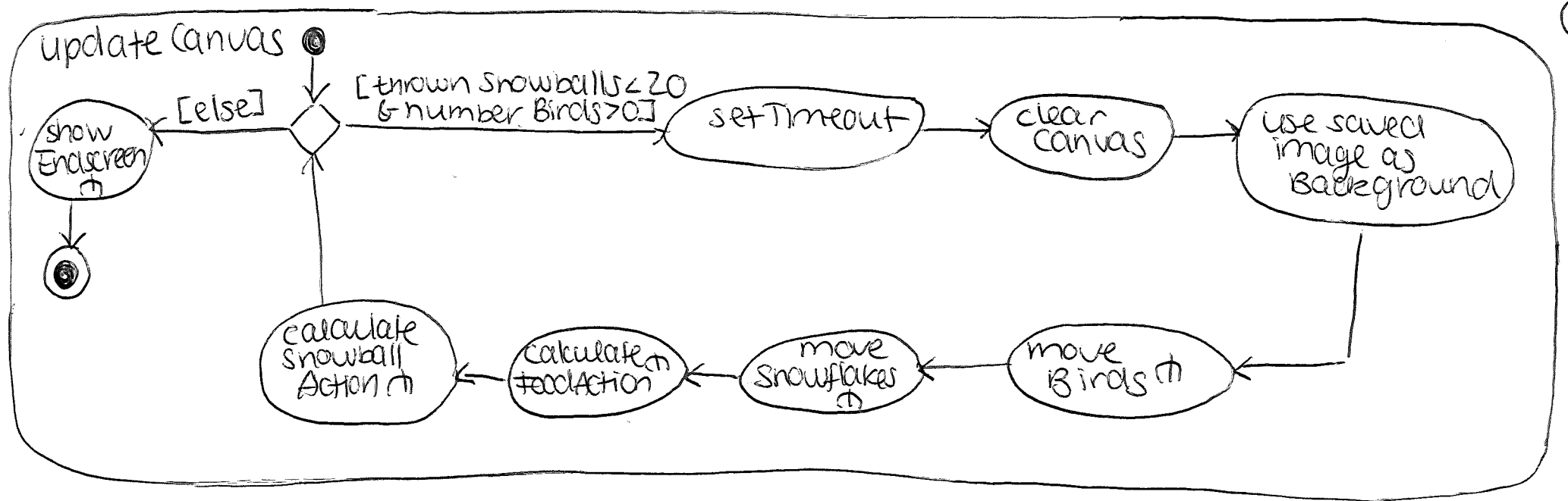
fill with
 color

restore

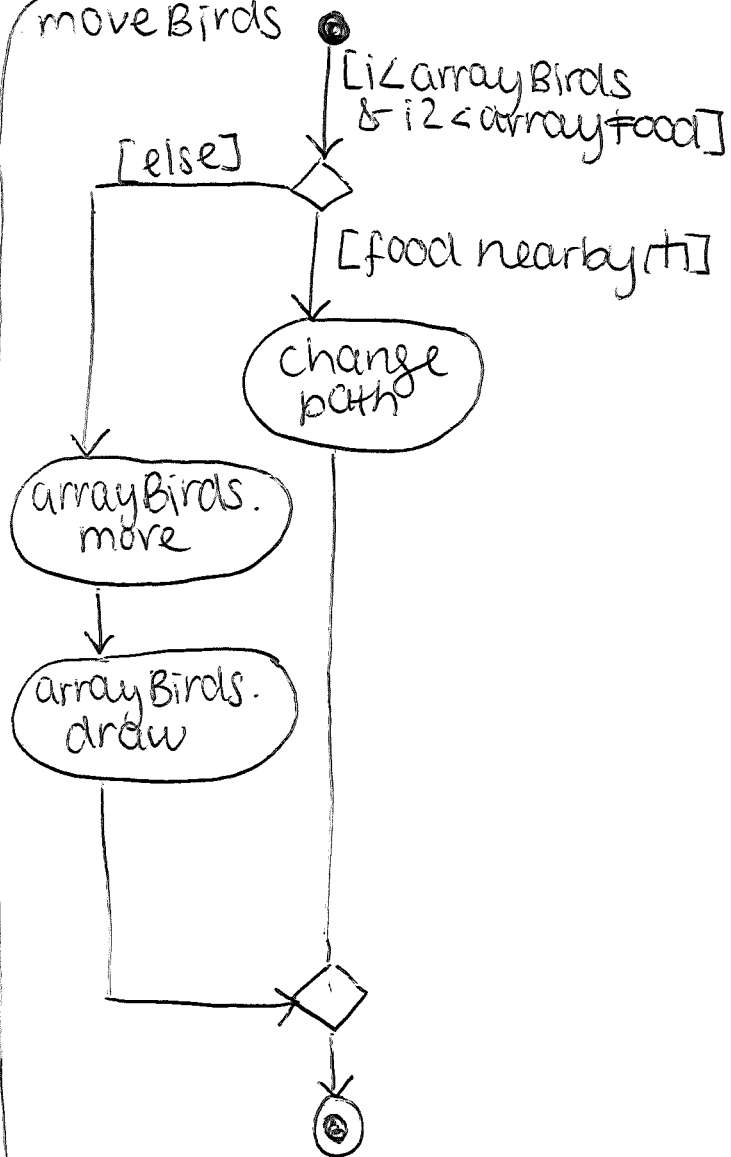
restore
 transform

drawn + 1

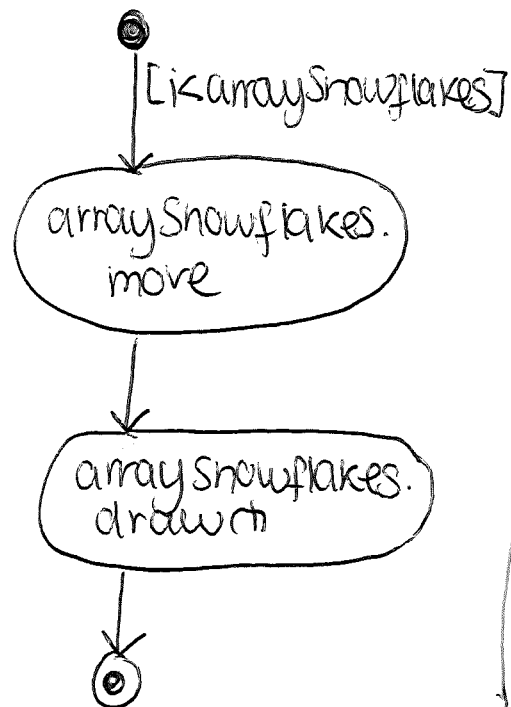




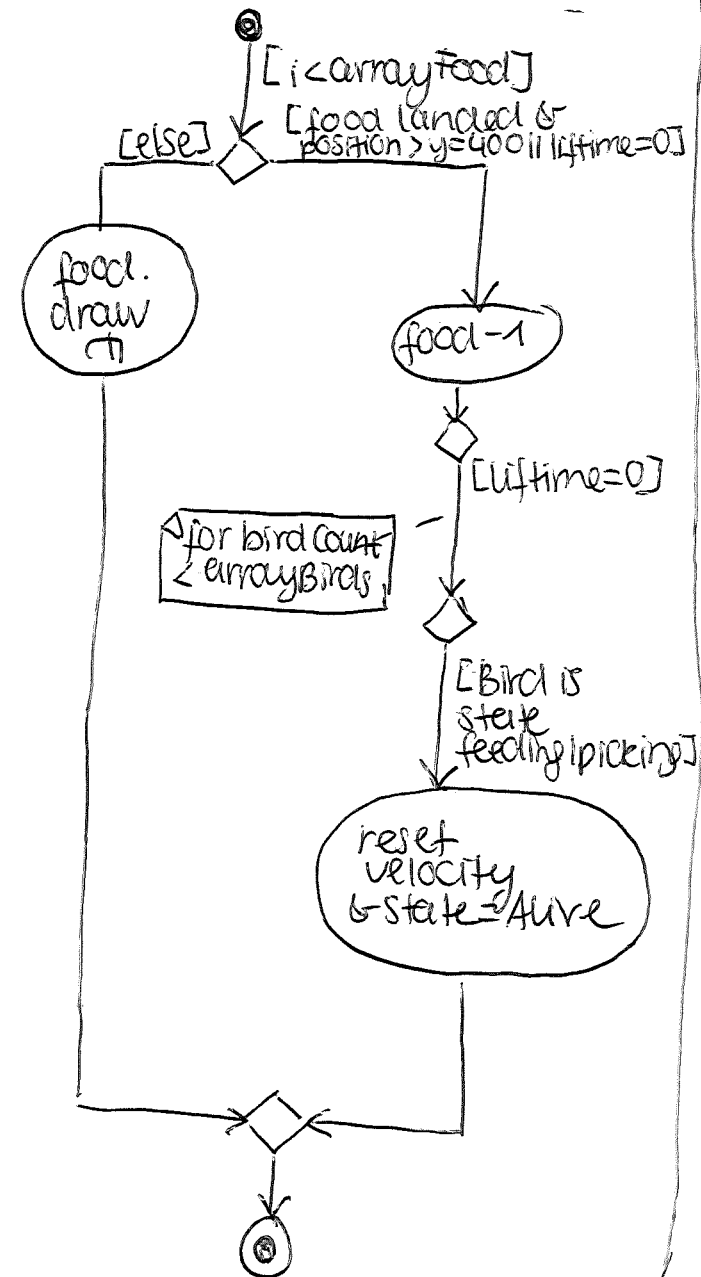
move Birds



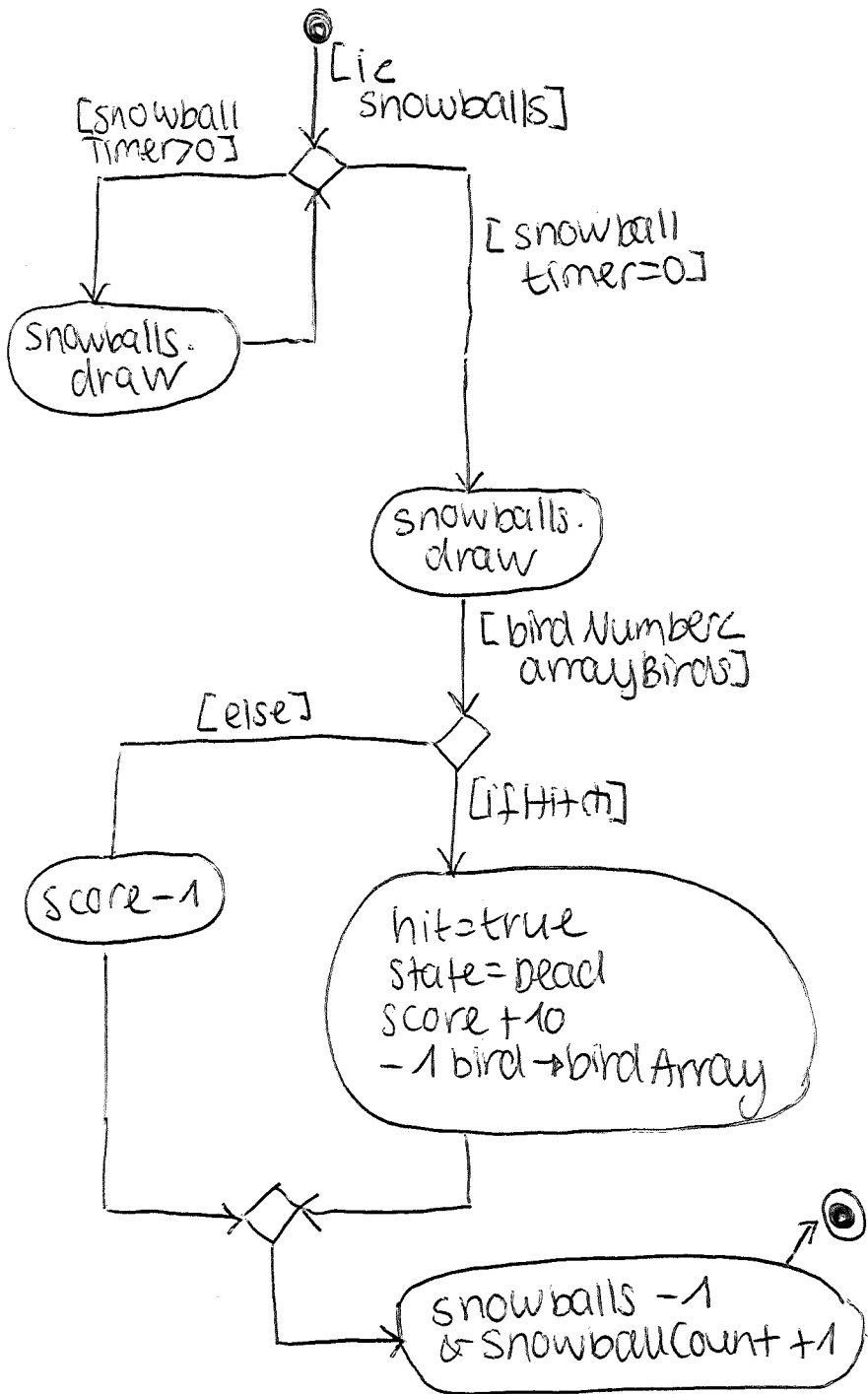
move Snowflakes



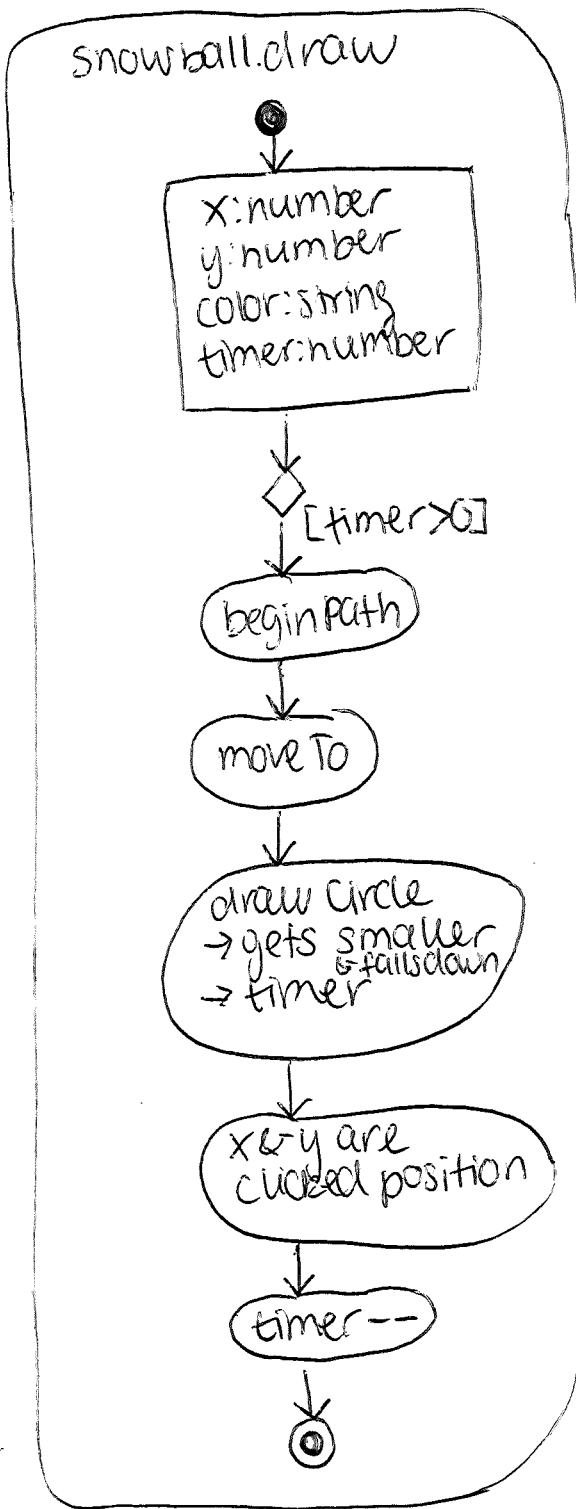
calculate Food Action



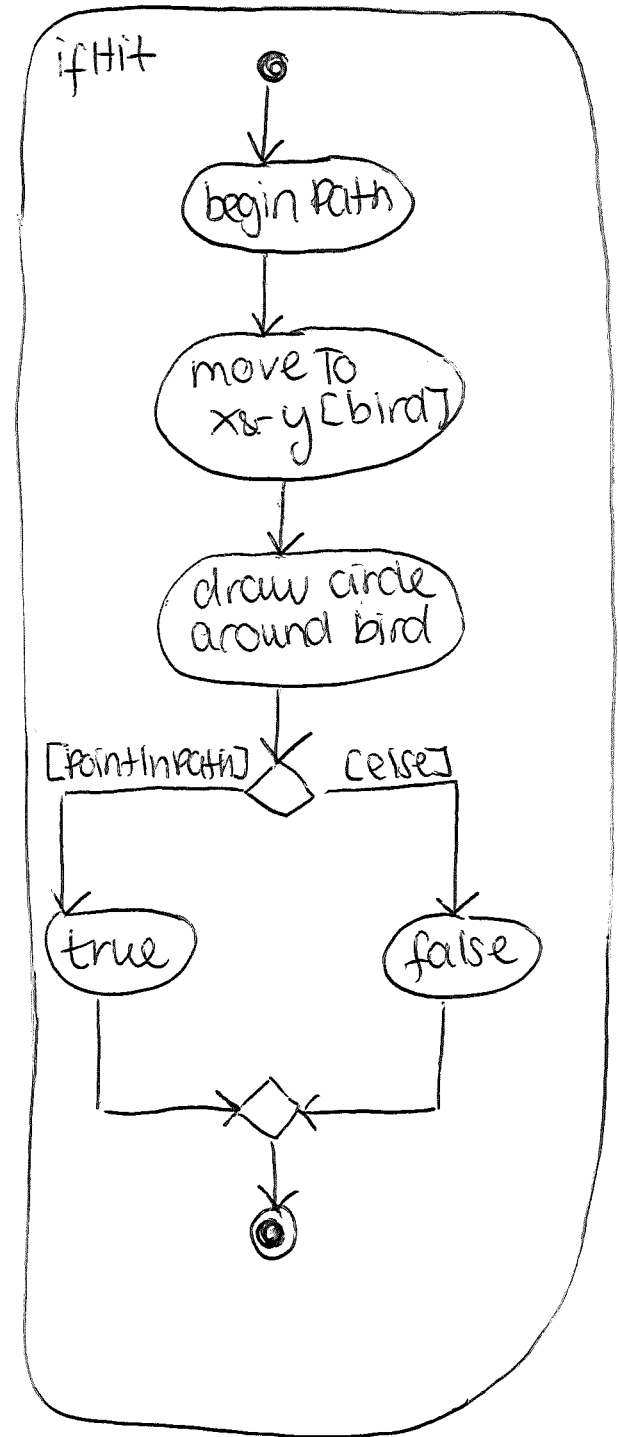
calculate Snowball Action



snowball.draw



if Hit



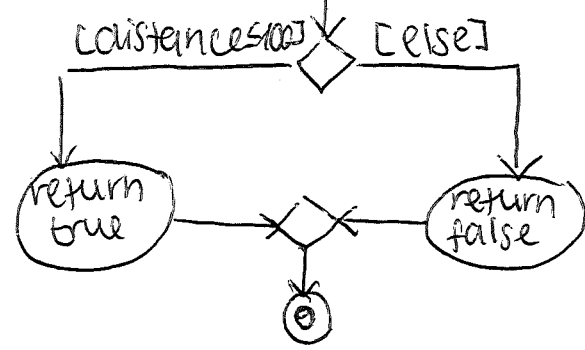
food nearby

a: number = position Bird (x)
- position Food (x)
b: number = position Bird (y)
- position Food (y)

calculate relative
distance coordinates

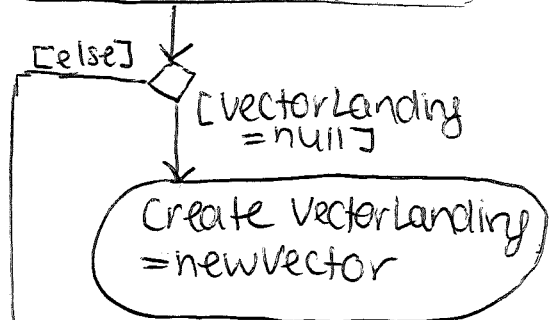
calculate with
Pythagorean
theorem

calculate
square root
→ distance
≡ hypotenuse

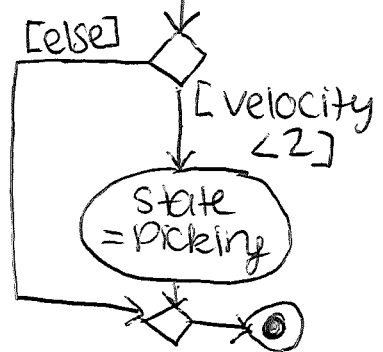


change path

vectorLanding: vector
a: number: x - food.x
b: number: y - food.y



velocity =
new vector
→ depending
on vectorLanding
and a & b



draw.food

radiusFood: number=10
food: Path2D

[else] [timer > 0]

draw food
& save
coordinates

move to
x & y

let nFood = 5

draw food
circle →
smaller when
timer--

save

translate
to position

size: Vector
x: number[]
y: number[]

lifetime--

save
food coordinates

[timer = 0]

timer--

restore

generate
random
position

for
each
food
circle

draw 5
food circles

click

throwSnowball

throwSnowball

x: number = _event.clientX
y: number = _event.clientY
ball: Snowball = new Snowball(x, y)

ball.x = x
ball.y = y
ball.timer = 25

push(ball)

aux click

throwFood

x: number = _event.clientX
y: number = _event.clientY
food: Food = new Food(x, y)

food.x = x
food.y = y
food.timer = 25
food.lifetime = 250

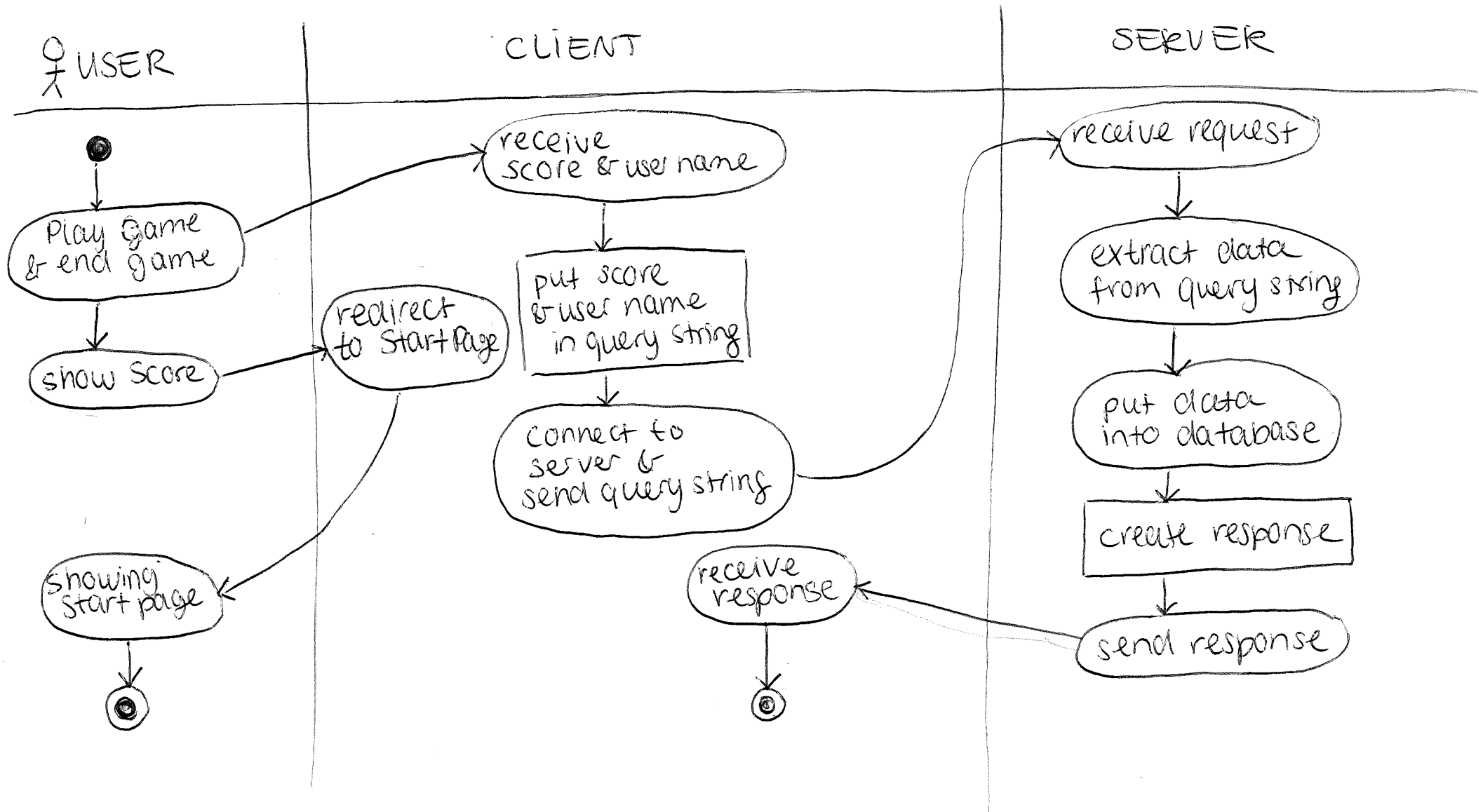
push(food)

foodCount + 1

DOMÄNENÜBERGREIFENDES AKTIVITÄTSDIAGRAMM Vogelhaus

→ save score in database

①



→ request highscores

