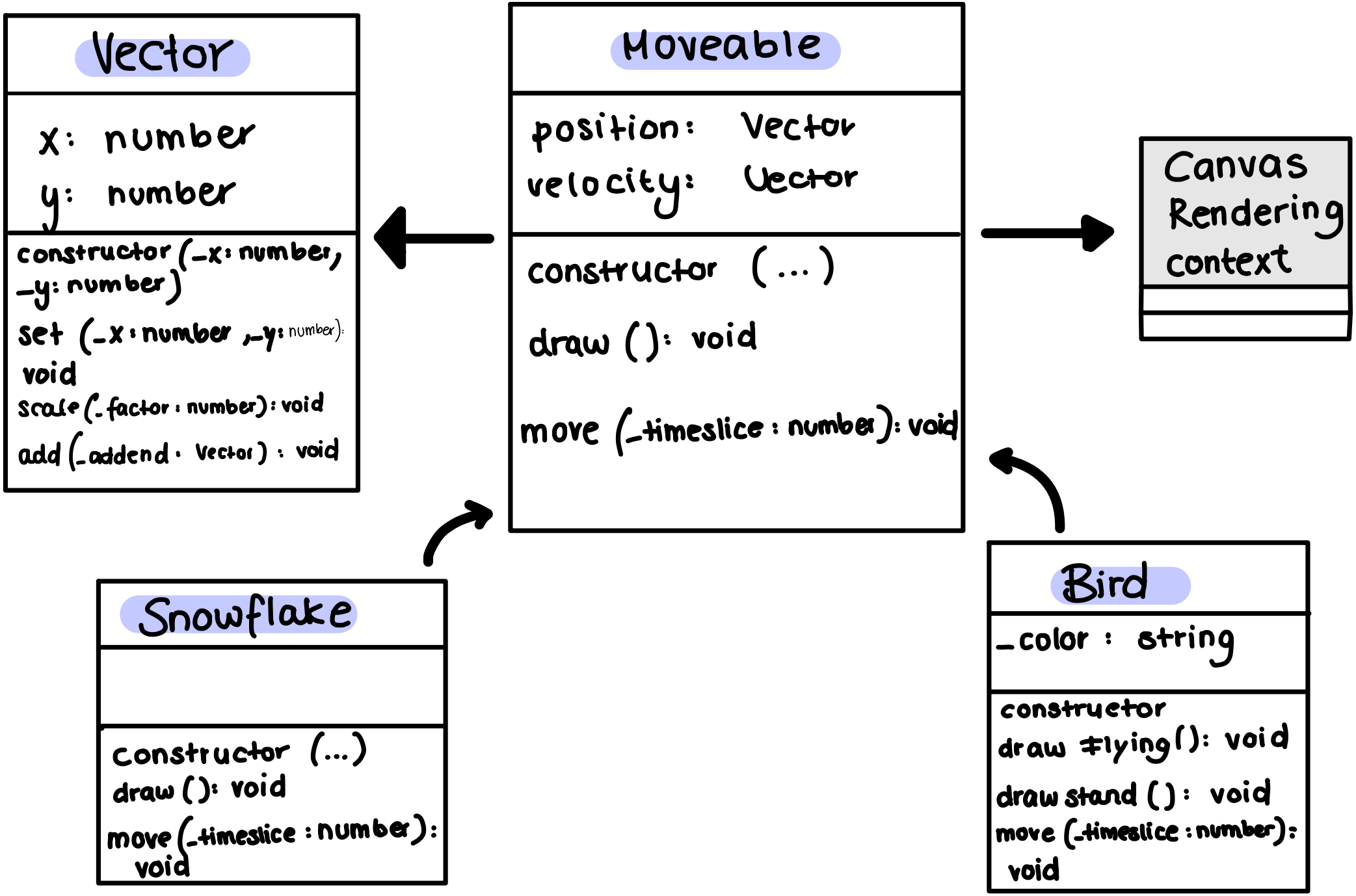
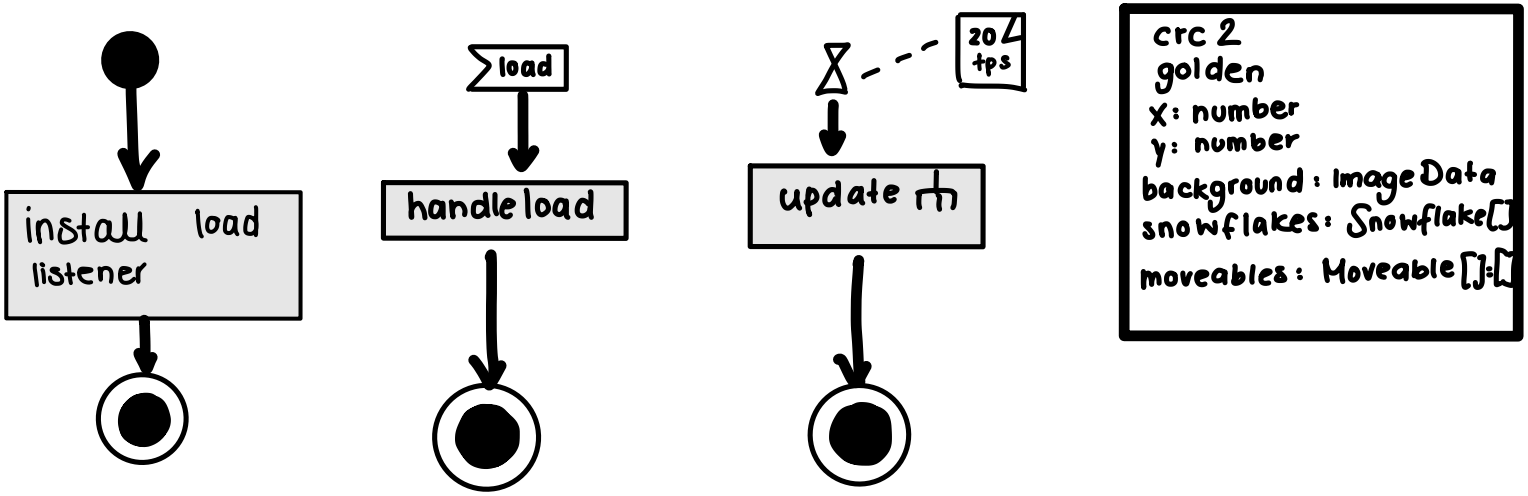
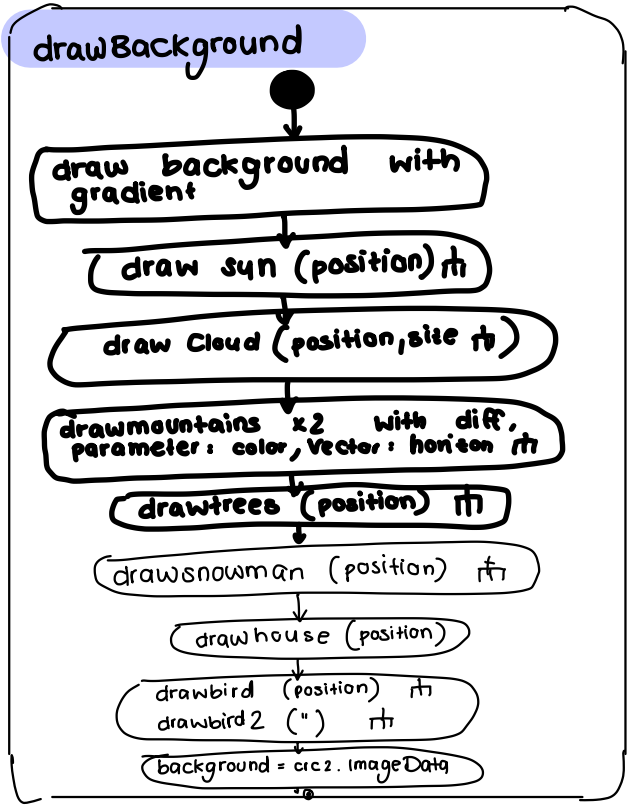
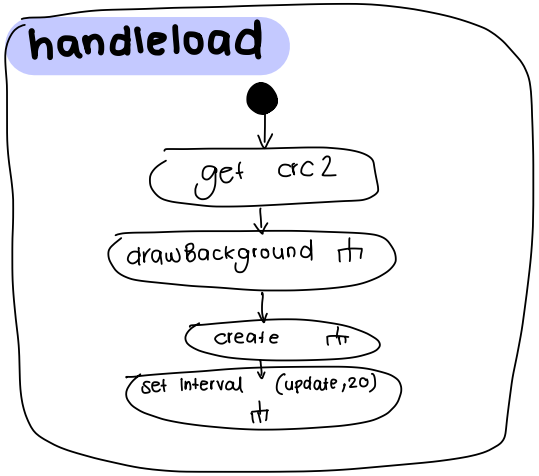
Class Diagramm

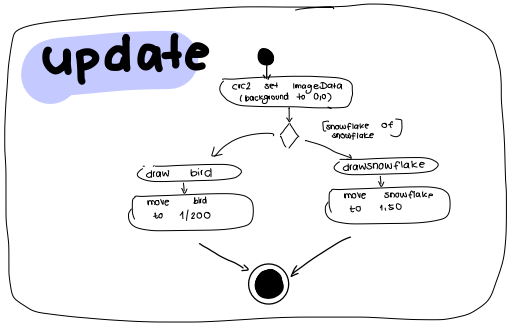


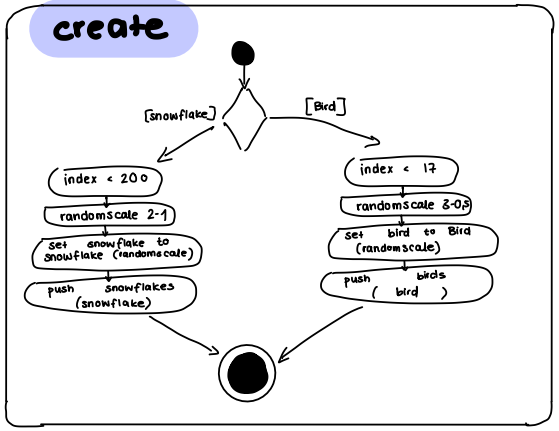
Aktivitätsdiagramm - main

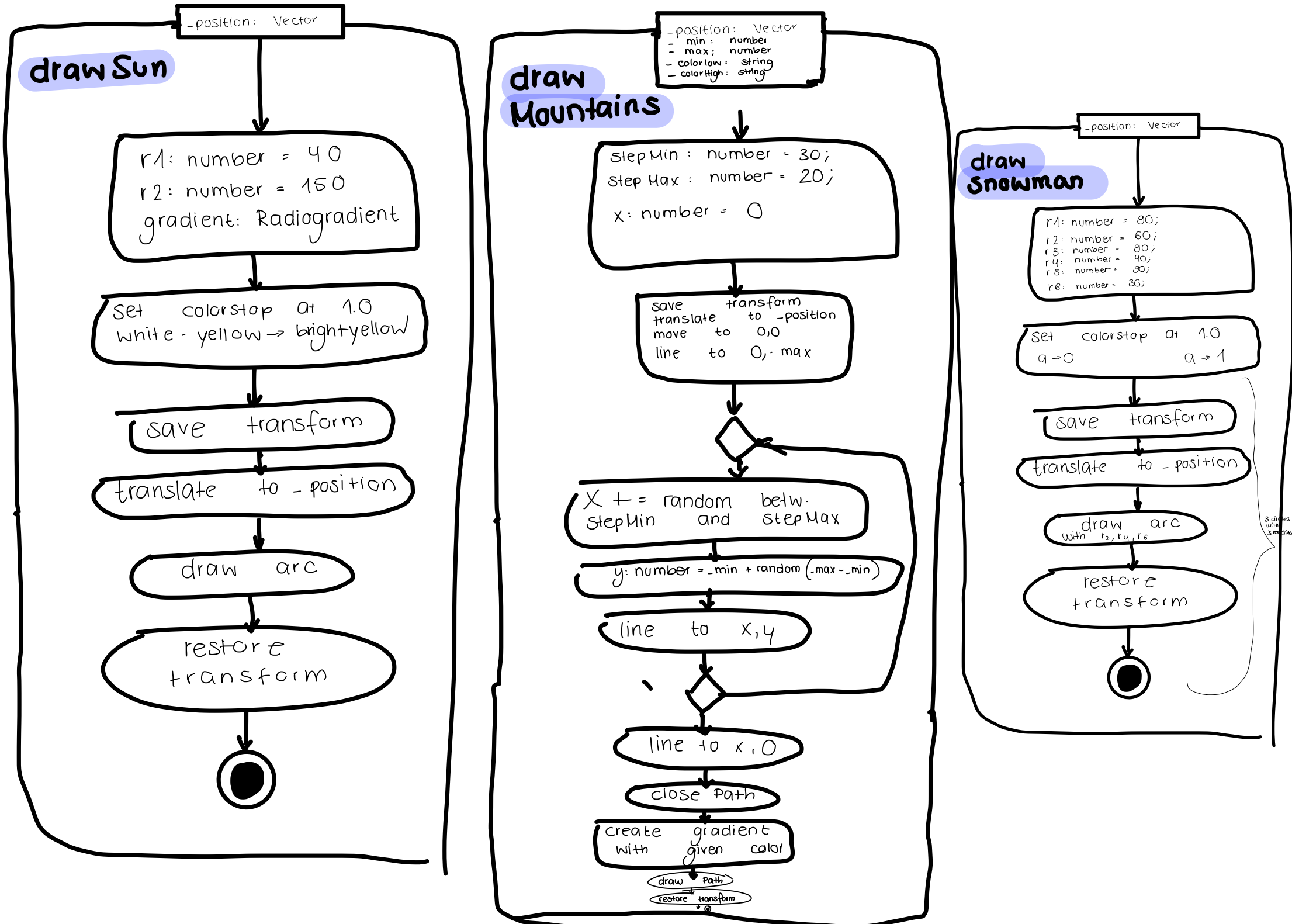


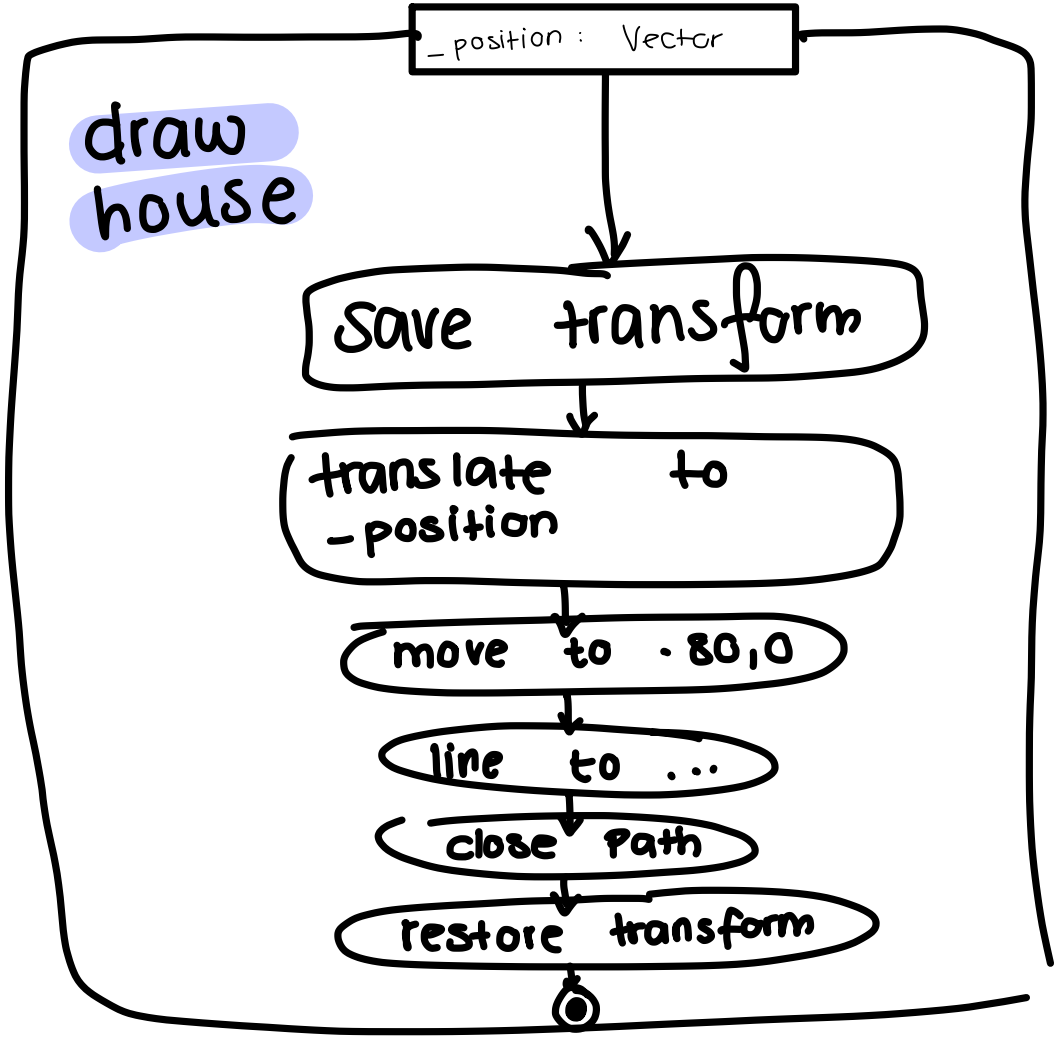


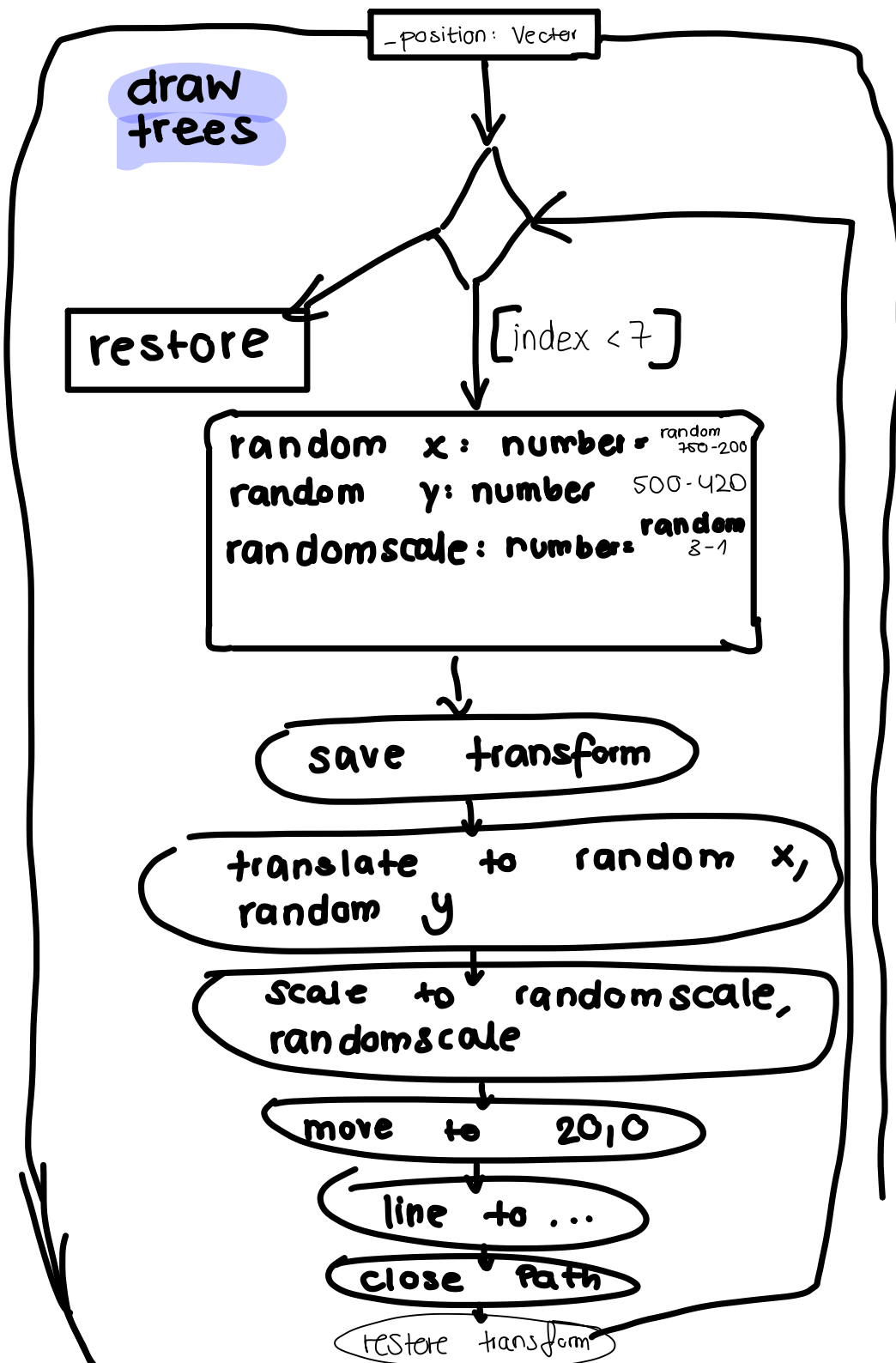


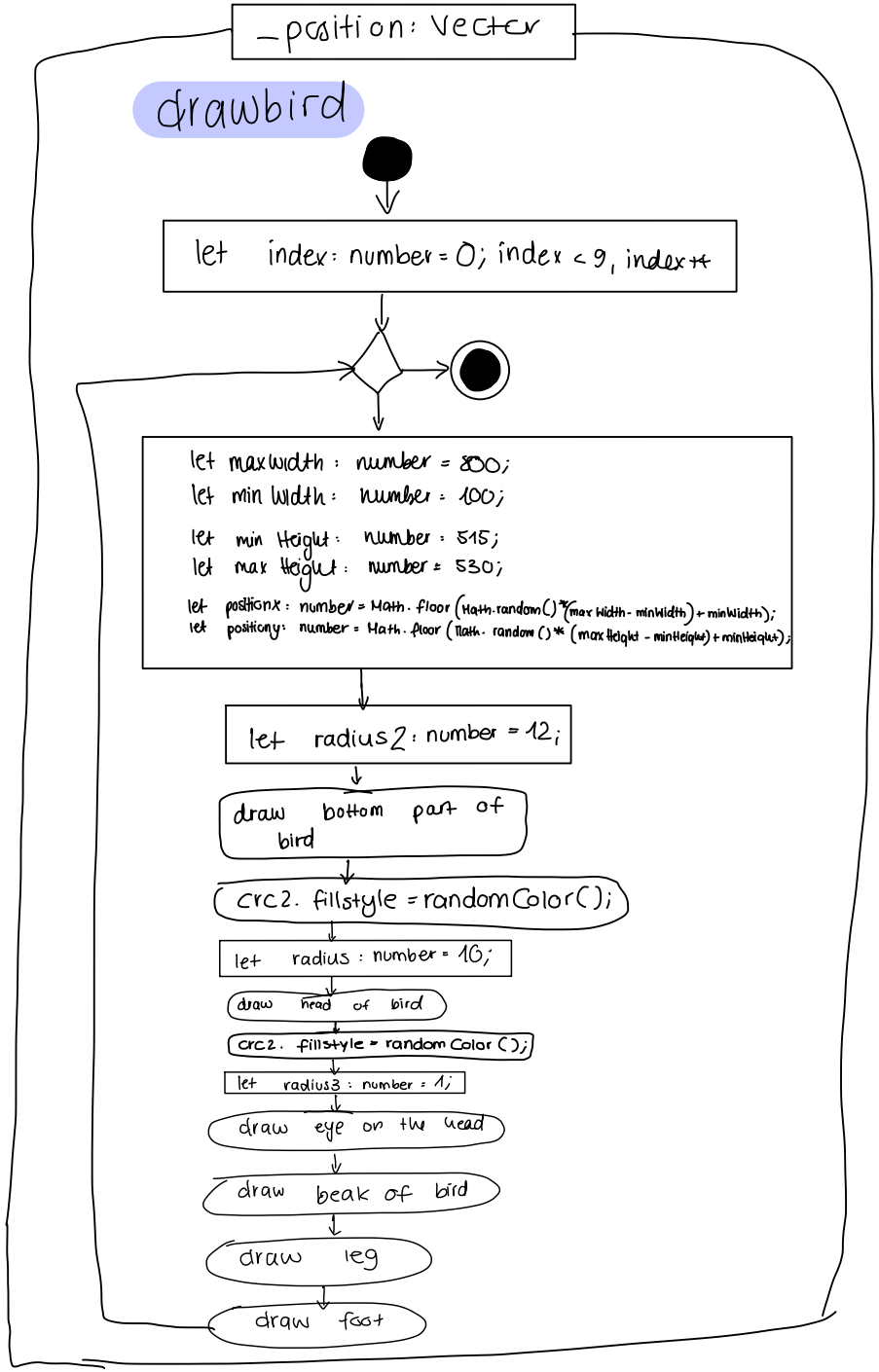


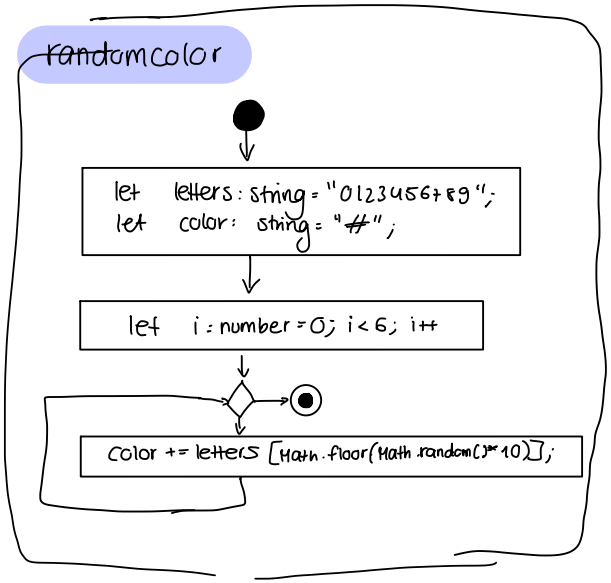



_position: Vector
drawSun
r1: number = 40
r2: number = 150
gradient: Radiogradient
Set colorstop at 1.0
white - yellow -> brightyellow
save transform
translate to -position
draw arc
restore transform
_position: Vector
- min: number
- max: number
- colorLow: string
- colorHigh: string
draw Mountains
stepMin: number = 30;
stepMax: number = 20;
x: number = 0
save transform
translate to -position
move to 0,0
line to 0,-max
X += random betw. stepMin and stepMax
y: number = -min + random(-max - -min)
line to x,y
line to x,0
close Path
create gradient with given color
draw Path
restore transform
_position: Vector
draw Snowman
r1: number = 90;
r2: number = 60;
r3: number = 90;
r4: number = 40;
r5: number = 90;
r6: number = 30;
Set colorstop at 1.0
a -> 0
a -> 1
save transform
translate to -position
draw arc
with r2, ry, r6
restore transform
3 circles with 3 radius

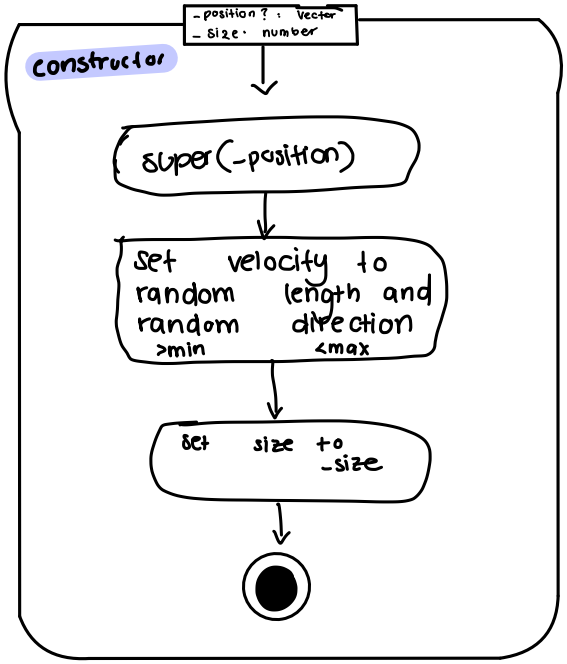

_position : Vector
draw house
save transform
translate to _position
move to -80,0
line to ...
close path
restore transform

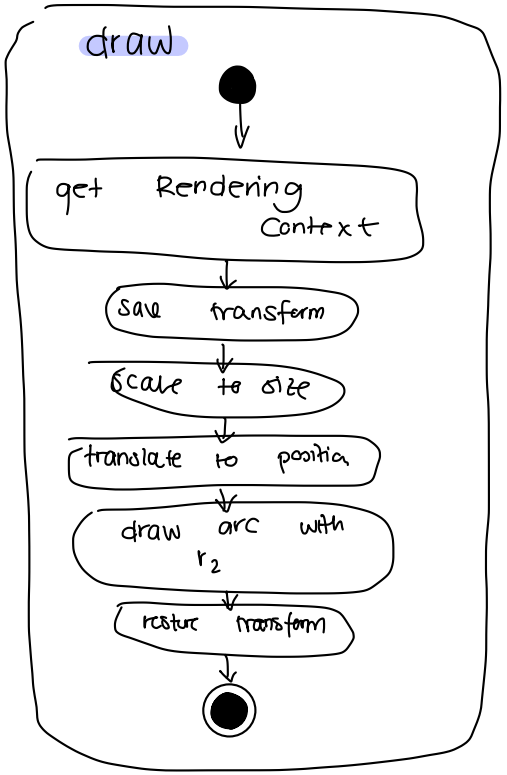

_position : Vector
draw trees
restore
[index < 7]
random x : number = random 150-200
random y : number 500-420
randomscale : number = random 3-1
save transform
translate to random x, random y
scale to randomscale, randomscale
move to 20,0
line to ...
close path
restore transform

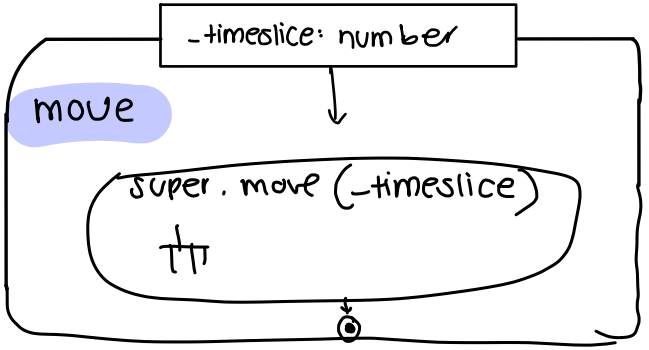

_position: Vector
drawbird
let index: number = 0; index < 9, index++
let maxWidth: number = 800;
let minWidth: number = 100;
let minHeight: number = 515;
let maxHeight: number = 530;
let positionX: number = Math.floor(Math.random()*(maxWidth - minWidth) + minWidth);
let positionY: number = Math.floor(Math.random()* (maxHeight - minHeight) + minHeight);
let radius2: number = 12;
draw bottom part of bird
crc2.fillStyle = randomColor();
let radius: number = 16;
draw head of bird
crc2.fillStyle = randomColor();
let radius3: number = 1;
draw eye on the head
draw beak of bird
draw leg
draw foot


randomcolor
let letters: string = "0123456789";
let color: string = "#";
let i: number = 0; i<6; i++
color += letters[Math.floor(Math.random()*10)];

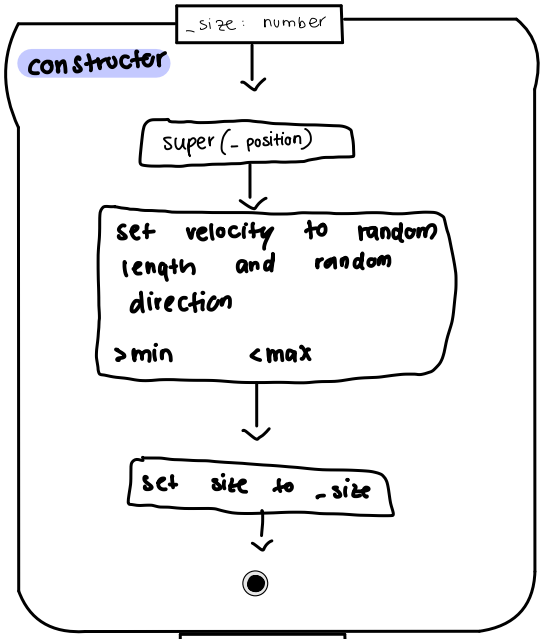
Snowflake

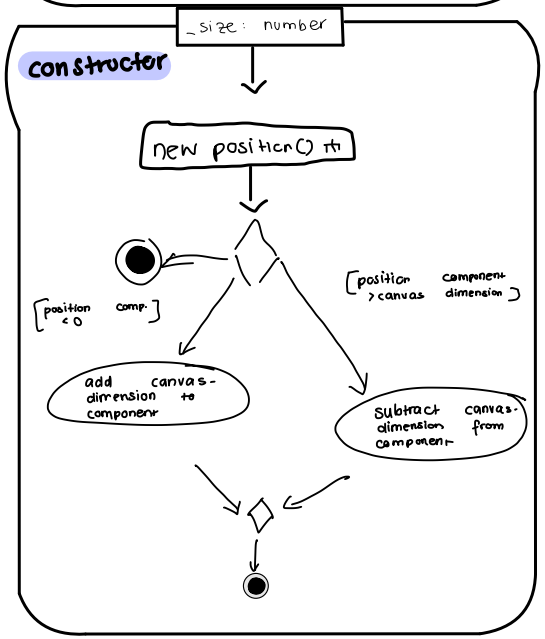

-position ? : Vector
- size: number
constructor
super(-position)
Set velocity to random length and random direction
>min
<max
Set size to -size

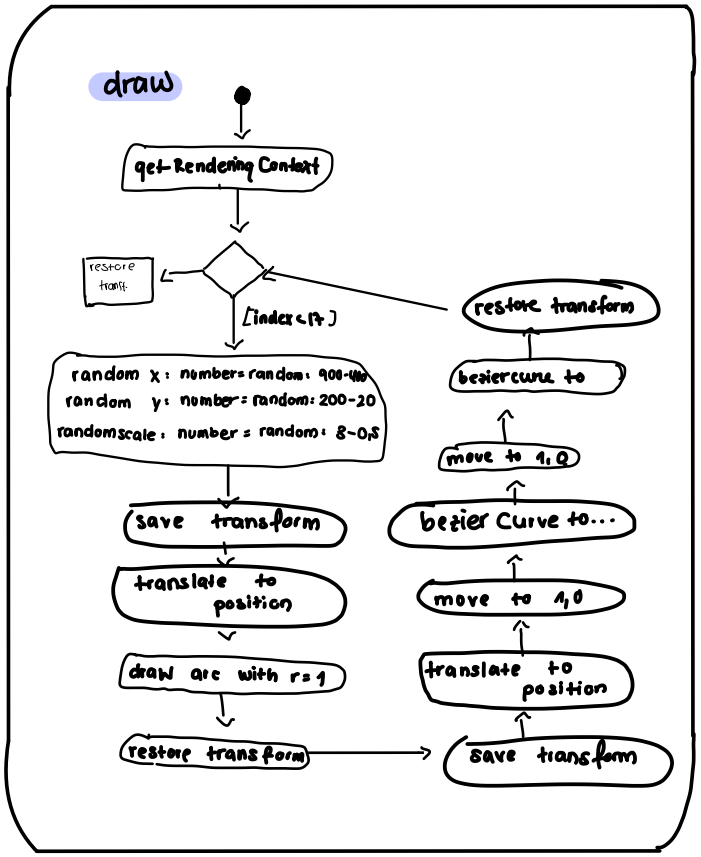

draw
get Rendering Context
save transform
Scale to size
translate to positio
draw arc with
r2
restore transform

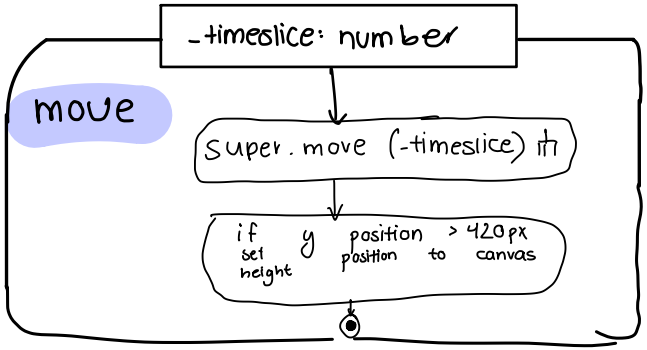

-timeslice: number
move
super.move(-timeslice)

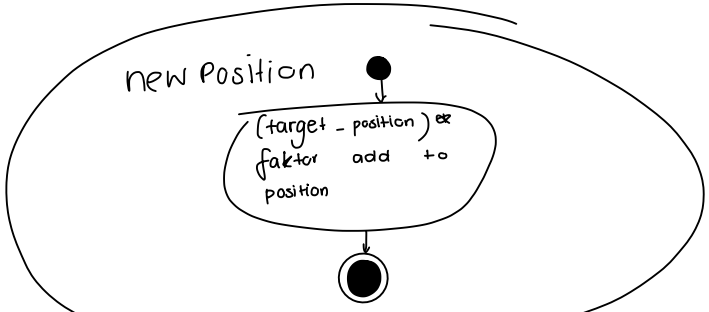
bird

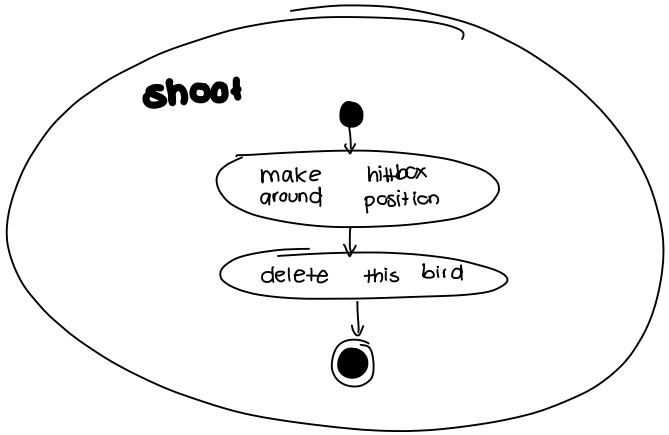

_size: number
constructor
super(_position)
set velocity to random length and random direction
>min <max
set size to _size


_size: number
constructor
new position() th
[position <0 comp.]
[position >canvas component dimension]
add canvas-dimension to component
subtract canvas-dimension from component


draw
getRenderingContext
restore transf.
[index<17]
restore transform
random x: number=random: 900-1100
random y: number=random: 200-20
randomscale: number=random: 8-0,5
bezierCurve to
save transform
move to 1,0
translate to position
bezier Curve to...
draw arc with r=1
move to 1,0
restore transform
translate to position
save transform


_timeslice: number
move
super.move (-timeslice) th
if y position >420px
set height
position to canvas


new Position
(target_position) dr
faktor add to
position


shoot
make hitbox around position
delete this bird

moveable

constructor

-position ? : Vector

Set position to -position or (0,0)

Set velocity to (0,0)

draw

move

_timeslice : number

move

_set offset to this.velocity.y & this.velocity.x

_scale offset (_timeslice)

_add offset to position

[pos. > canvas dimension]

subtract canvas dimension from snowflake

[pos. < 0]

add canvas dimension to snowflake