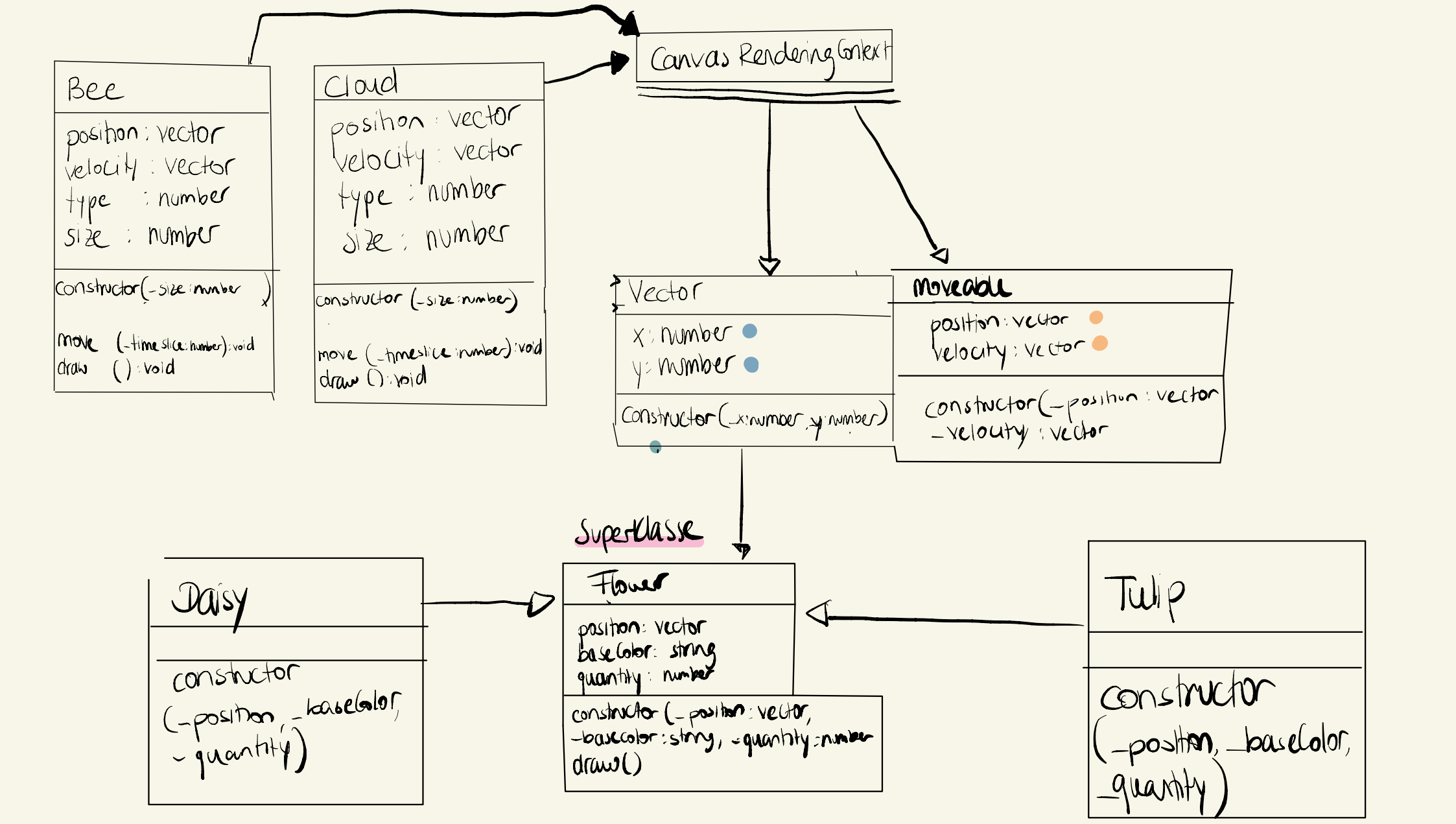
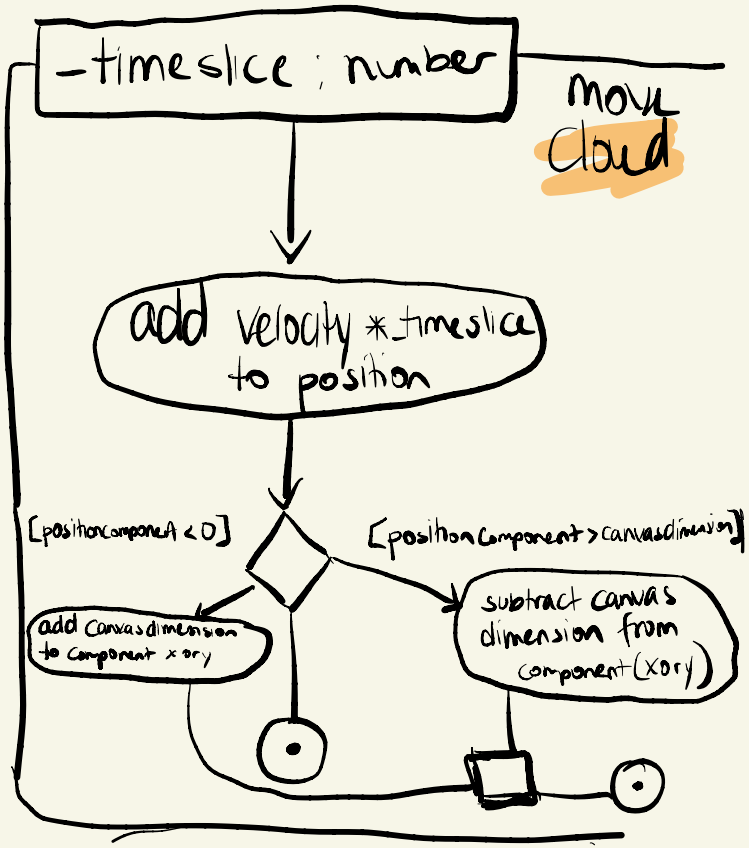
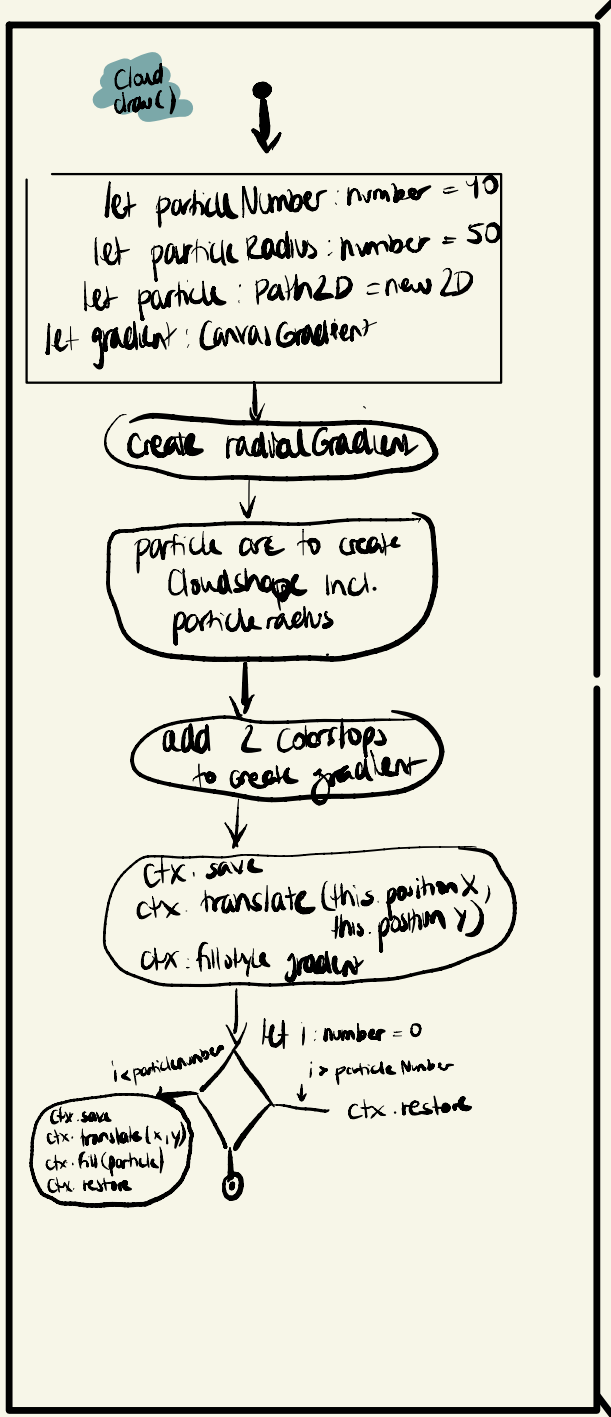
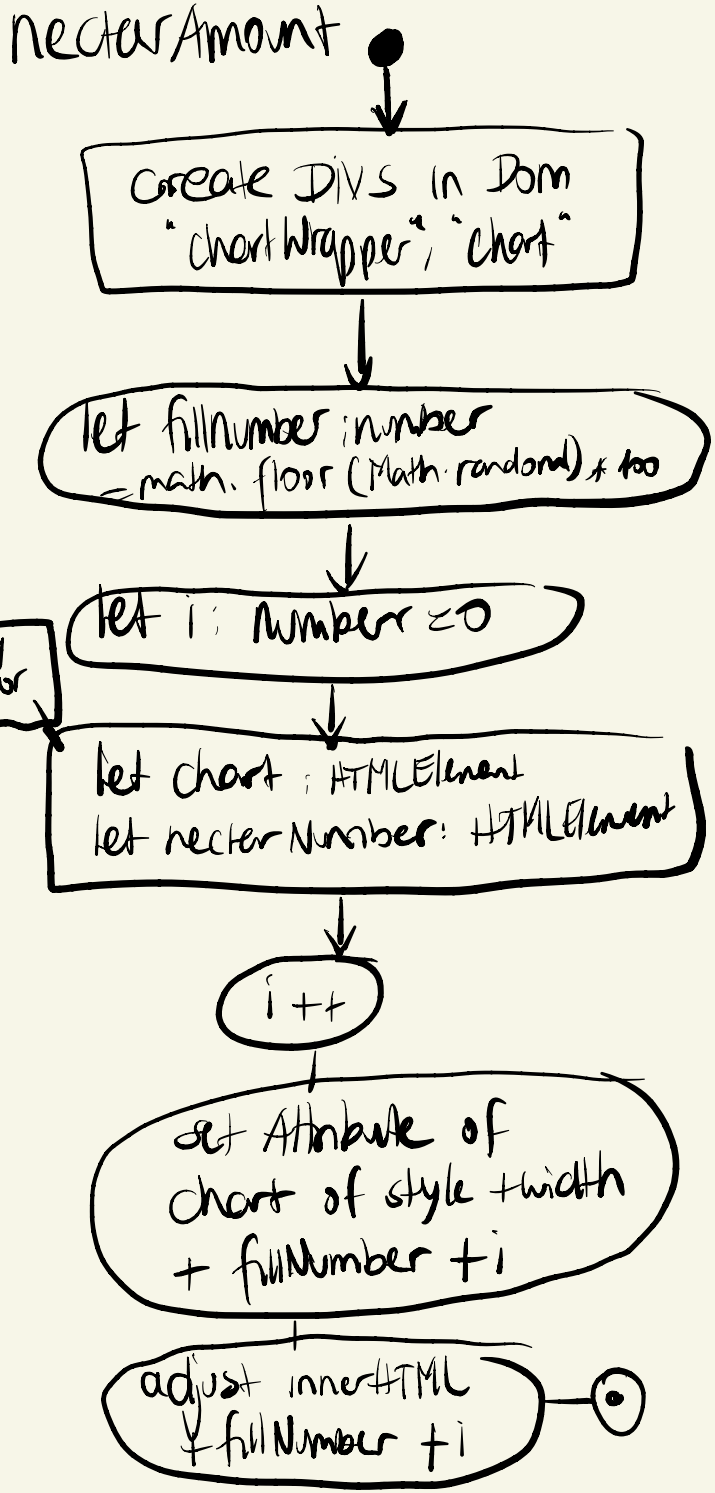
Polymorphie - Blumenwiese

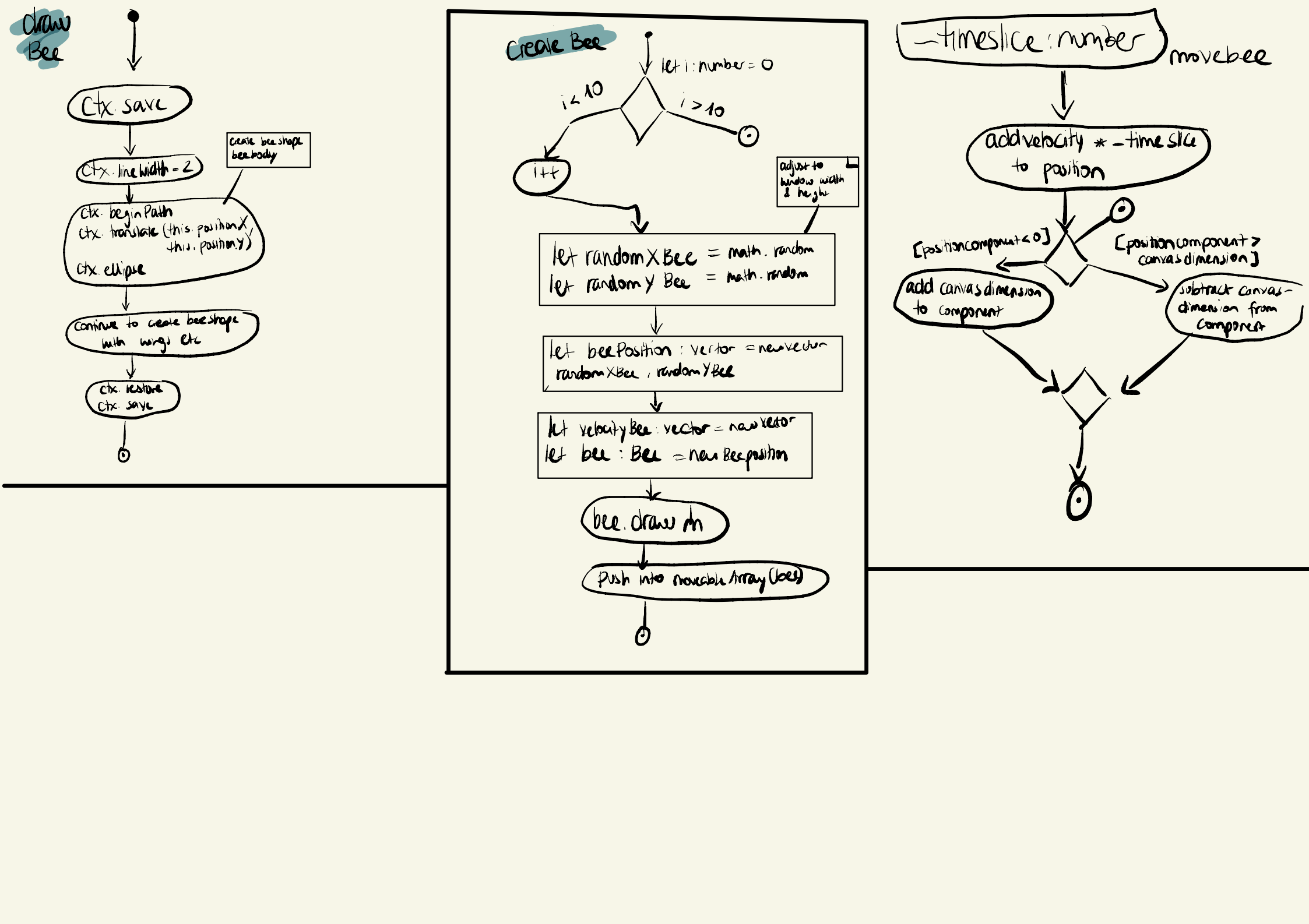
● = public
● = protected

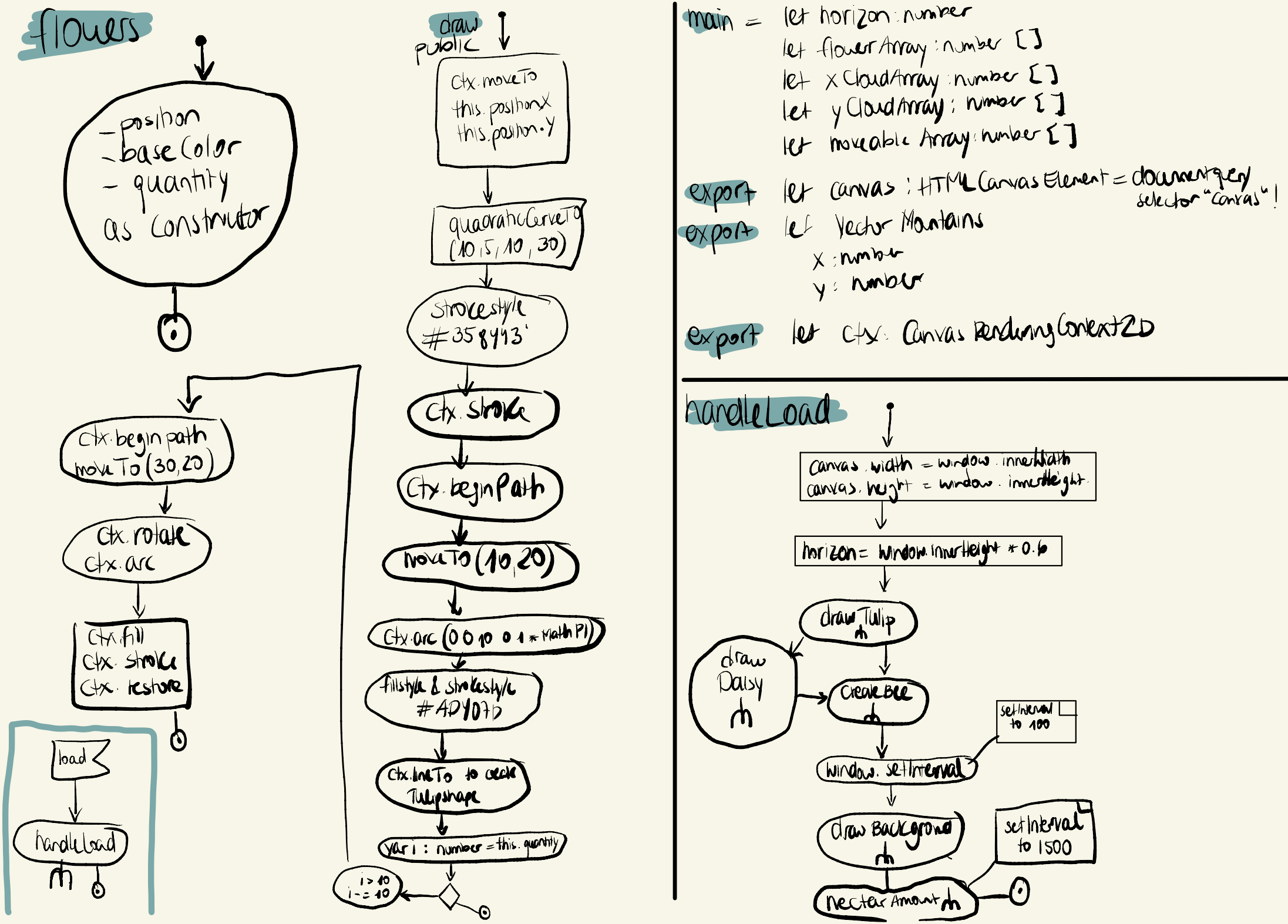


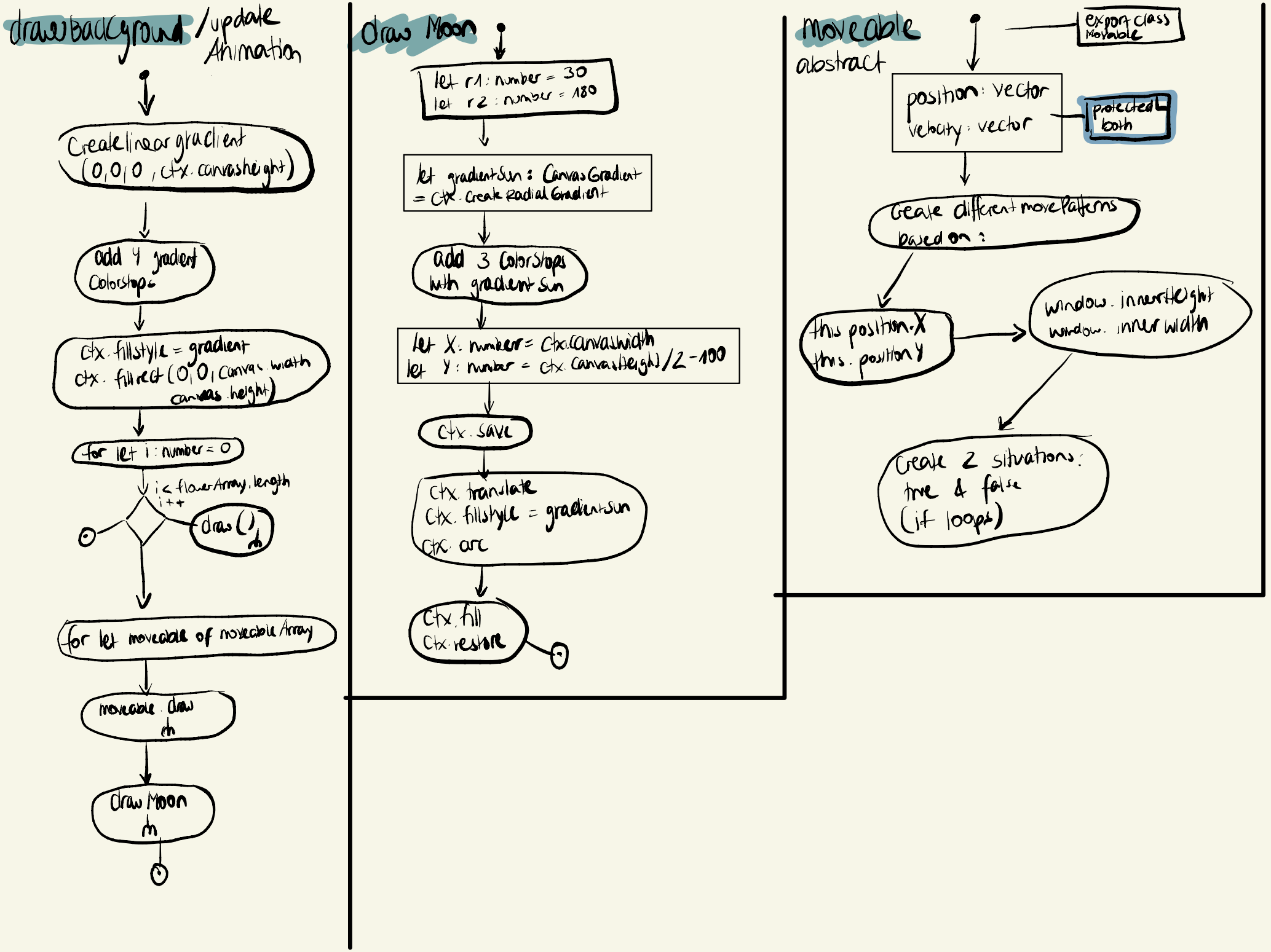

-timeslice : number
move Cloud
add velocity * timeslice to position
[positionComponent < 0]
[positionComponent > canvasdimension]
add canvas dimension to component x or y
subtract canvas dimension from component (x or y)

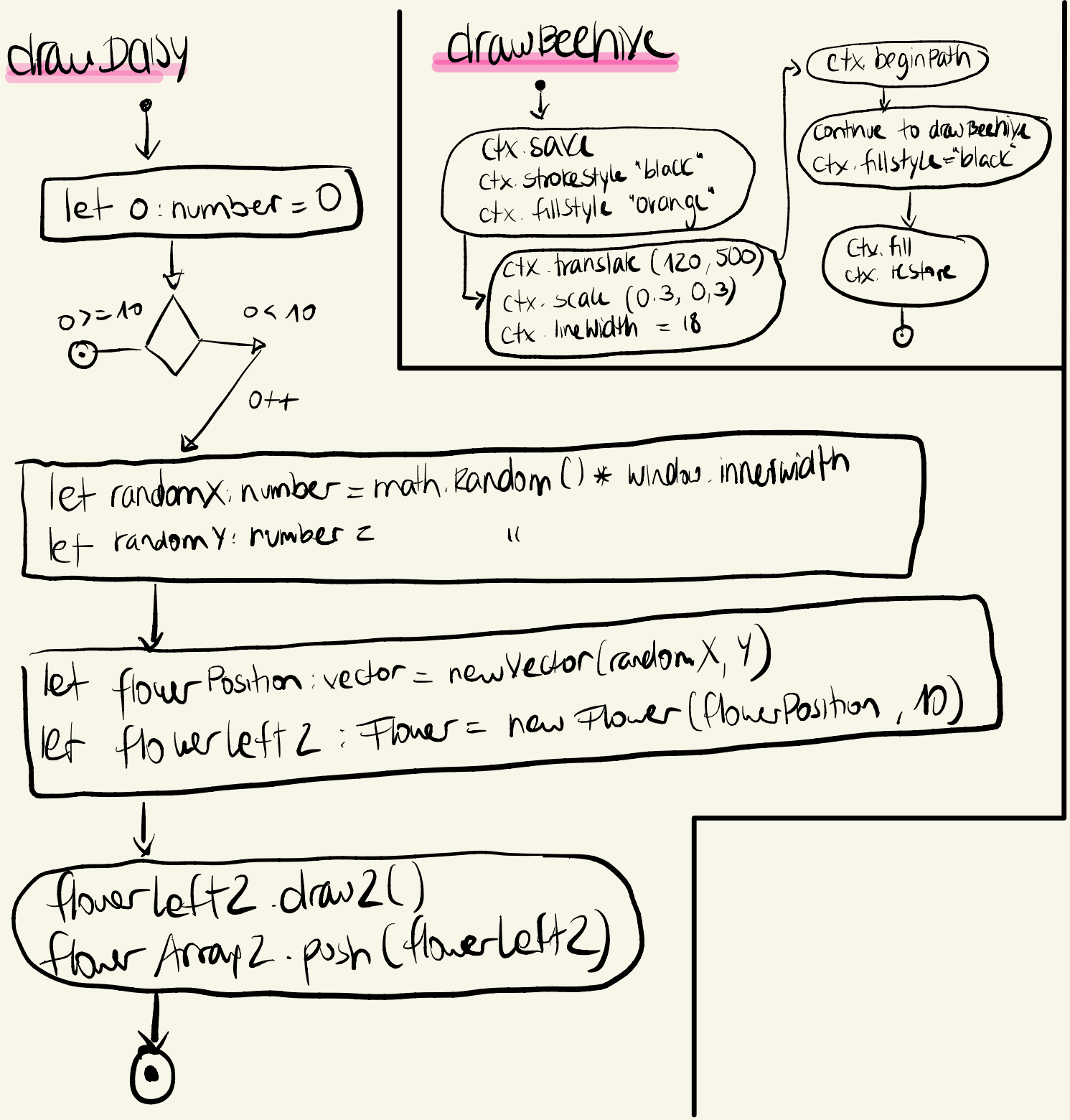

Cloud draw()
let particleNumber : number = 40
let particleRadius : number = 50
let particle : Path2D = new 2D
let gradient : CanvasGradient
create radialGradient
particle are to create Cloudshape incl. particle radius
add 2 colorstops to create gradient
ctx.save
ctx.translate (this.positionX, this.positionY)
ctx.fillstyle gradient
let i : number = 0
i < particleNumber
i > particleNumber
ctx.save
ctx.translate (x, y)
ctx.fill (particle)
ctx.restore
ctx.restore


necterAmount
create DIVs in Dom "chartWrapper", "chart"
let fillnumber : number = math.floor (Math.random) * 100
let i : number = 0
query selector
let chart : HTMLElement
let necterNumber : HTMLElement
i ++
set Attribute of chart of style + width + fillNumber + i
adjust innerHTML + fillNumber + i


drawBee
ctx.save
ctx.lineWidth = 2
create bee shape bee body
ctx.beginPath
ctx.translate (this.positionX, this.positionY)
ctx.ellipse
continue to create bee shape with wings etc
ctx.restore
ctx.save
createBee
let i : number = 0
i < 10
i > 10
i++
adjust to window width & height
let randomXBee = math.random
let randomYBee = math.random
let beePosition : vector = new vector randomXBee, randomYBee
let velocityBee : vector = new vector
let bee : Bee = new Bee position
bee.drawMe
push into moveable Array (bee)
_timeSlice : number
moveBee
add velocity * _timeSlice to position
[position component < 0]
[position component > canvas dimension]
add canvas dimension to component
subtract canvas-dimension from component


flowers
- position
- baseColor
- quantity
as constructor
ctx.beginPath
moveTo(30,20)
ctx.rotate
ctx.arc
ctx.fill
ctx.stroke
ctx.restore
load
handleLoad
draw
public
ctx.moveTo
this.positionX
this.position.y
quadraticCurveTo
(10,5,10,30)
strokestyle
#358493
ctx.stroke
ctx.beginPath
moveTo(10,20)
ctx.arc(0 0 10 0 1 * Math.PI)
fillstyle & strokestyle
#AD9078
ctx.lineTo to create
Tulipshape
var i : number = this.quantity
i > 10
i -= 10
main =
let horizon: number
let flowerArray: number []
let xCloudArray: number []
let yCloudArray: number []
let moveableArray: number []
export
let canvas : HTMLCanvasElement = document.querySelector("canvas")!
export
let VectorMountains
x: number
y: number
export
let ctx: CanvasRenderingContext2D
handleLoad
canvas.width = window.innerWidth
canvas.height = window.innerHeight
horizon = window.innerHeight * 0.6
drawTulip
drawDaisy
createBee
setInterval
to 100
window.setInterval
drawBackground
setInterval
to 1500
nectarAmount


drawBackground / updateAnimation
Create linear gradient (0,0,0, ctx.canvasHeight)
add 4 gradient Colorstops
ctx.fillStyle = gradient
ctx.fillRect(0,0, canvas.width canvas.height)
for let i: number = 0
i < flowerArray.length
i++
draw()
for let moveable of moveableArray
moveable.draw
drawMoon
drawMoon
let r1: number = 30
let r2: number = 180
let gradientSun: CanvasGradient = ctx.createRadialGradient
add 3 Colorstops with gradientSun
let X: number = ctx.canvasWidth
let y: number = ctx.canvasHeight/2 -100
ctx.save
ctx.translate
ctx.fillStyle = gradientSun
ctx.arc
ctx.fill
ctx.restore
moveable
abstract
export class Movable
position: vector
velocity: vector
protected both
Create different movePatterns based on:
this.position.X
this.positionY
window.innerHeight
window.innerWidth
Create 2 situations: true & false (if loops)


drawDaisy
let 0 : number = 0
0 >= 10
0 < 10
0++
drawBeehive
ctx.save
ctx.strokeStyle "black"
ctx.fillStyle "orange"
ctx.translate (120, 500)
ctx.scale (0.3, 0.3)
ctx.lineWidth = 18
ctx.beginPath
Continue to drawBeehive
ctx.fillStyle="black"
Ctx.fill
ctx.restore
let randomX: number = math.Random() * window.innerWidth
let randomY: number =
"
let flowerPosition: vector = newVector(randomX, Y)
let flowerLeft2 : Flower = new Flower (flowerPosition, 10)
flowerLeft2.draw2()
flowerArray2.push(flowerLeft2)