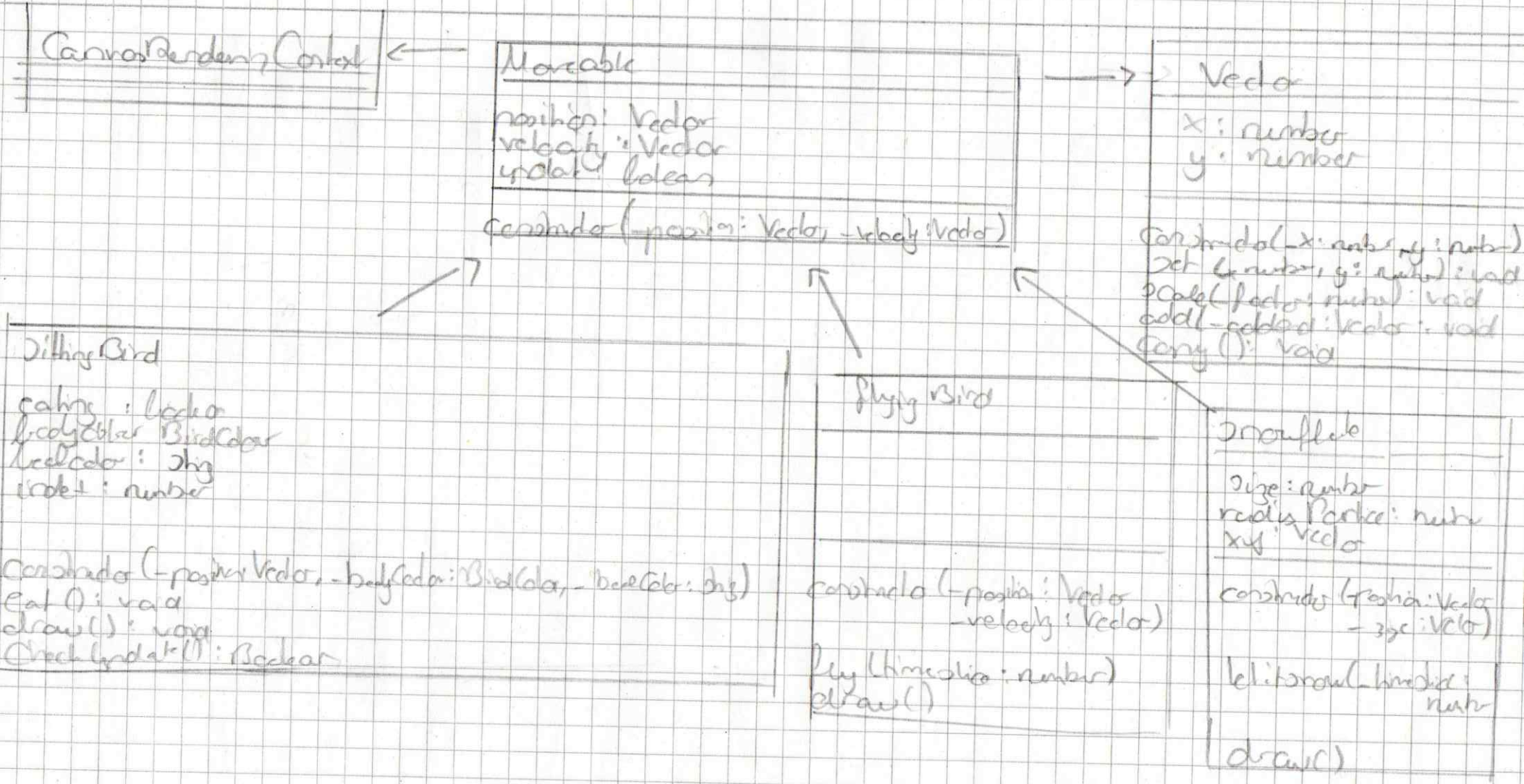
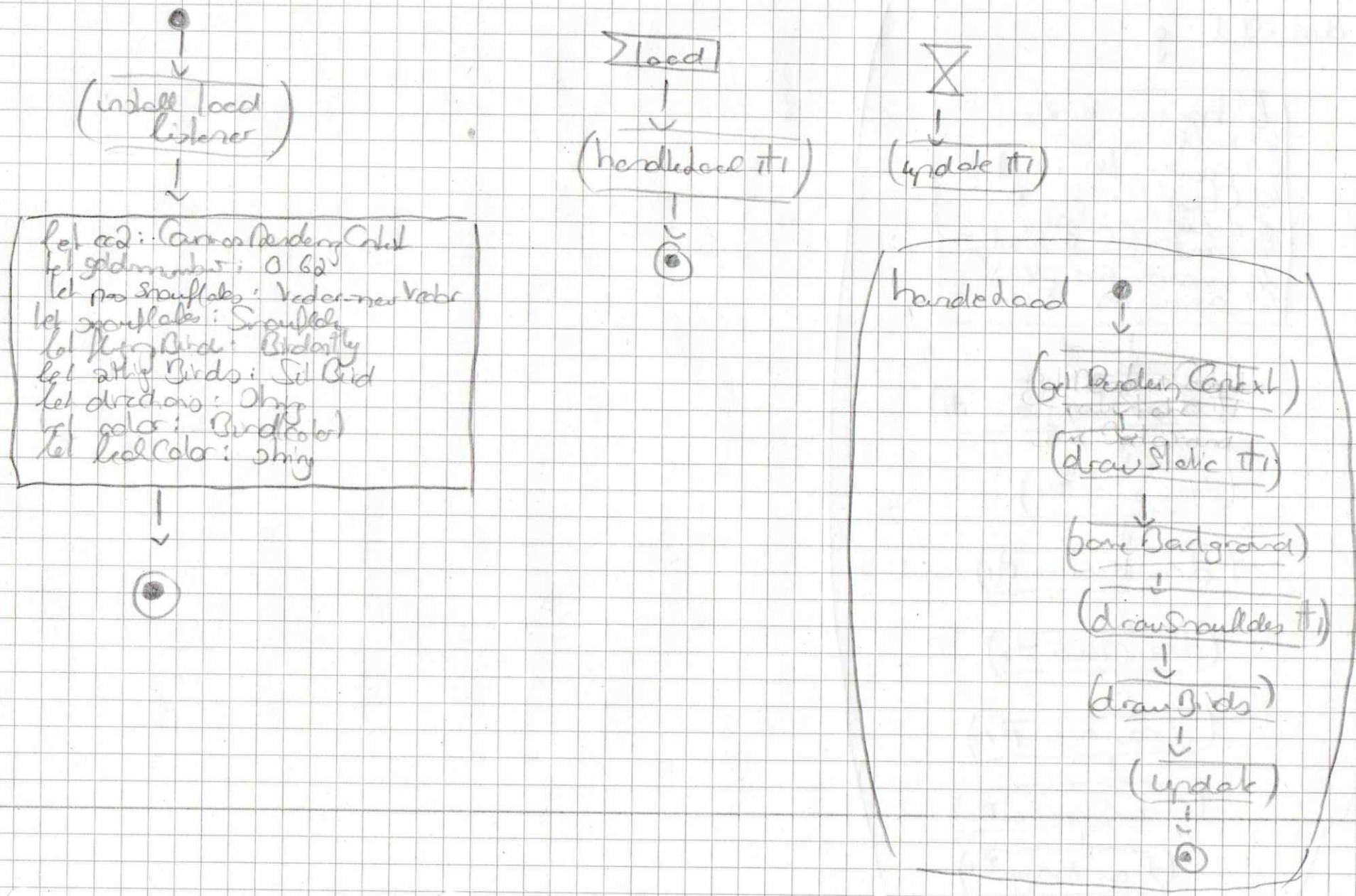
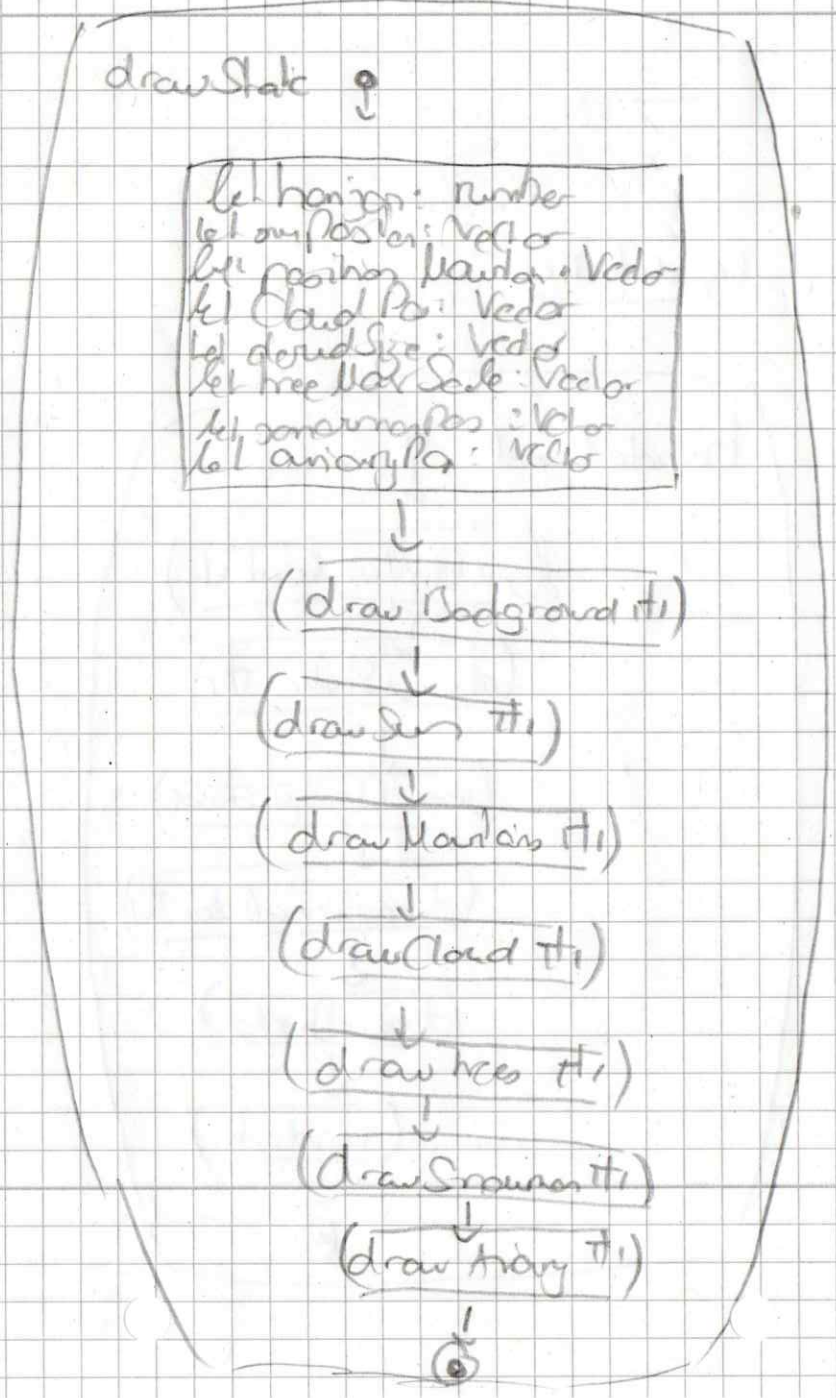
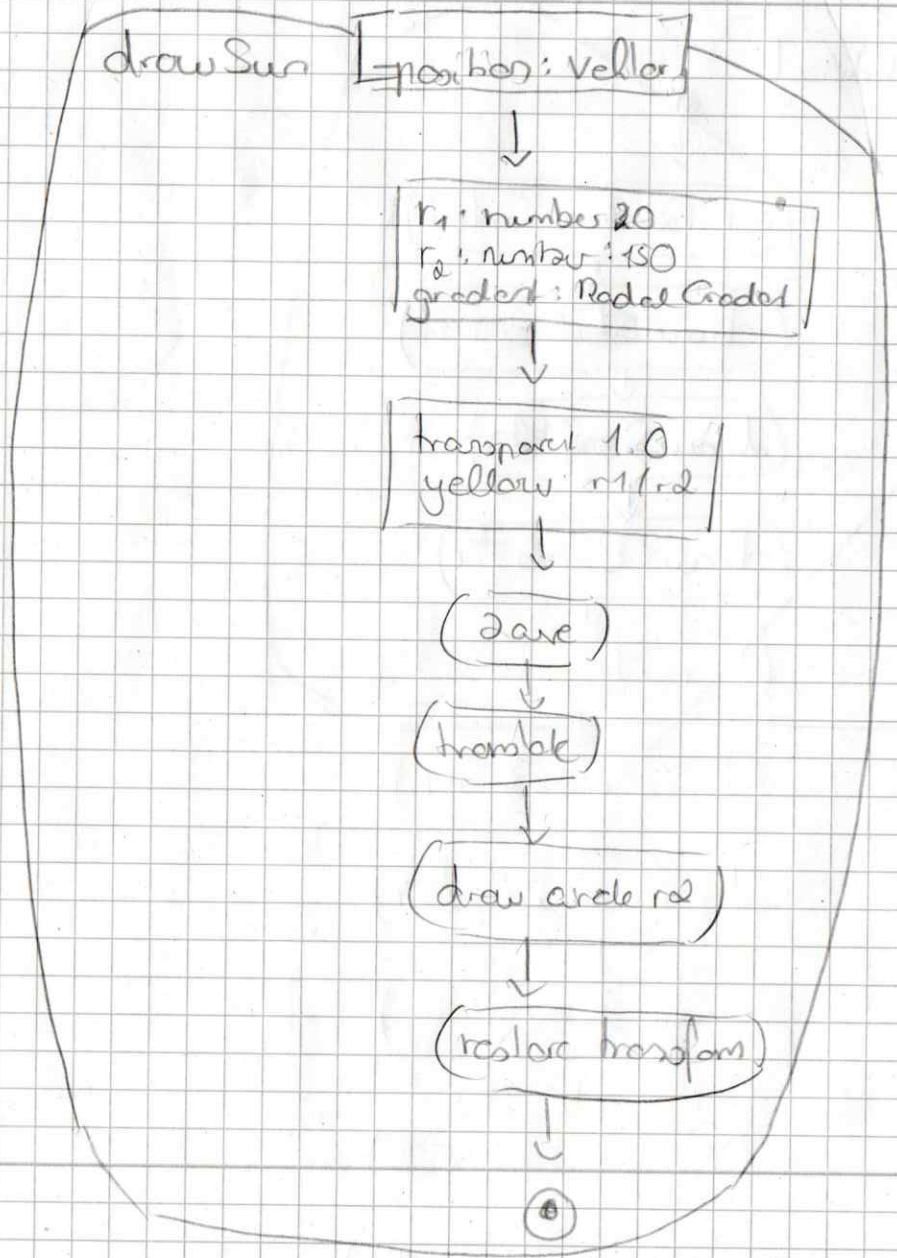

CanvasRenderingContext
Movable
position: Vector
velocity: Vector
update: boolean
constructor(_position: Vector, _velocity: Vector)
Vector
x: number
y: number
constructor(_x: number, _y: number)
set(x: number, y: number): void
scale(factor: number): void
addTo(addend: Vector): void
copy(): Vector
FlyingBird
constructor(_position: Vector
_velocity: Vector)
fly(timeslice: number)
draw()
SittingBird
eating: boolean
birdColor: BirdColor
beakColor: string
count: number
constructor(_position: Vector, _bodyColor: BirdColor, _beakColor: string)
eat(): void
draw(): void
checkUpdate(): Boolean
Snowflake
size: number
velocity: Vector
constructor(_position: Vector
_size: Vector)
letItSnow(_timeslice:
number
draw()

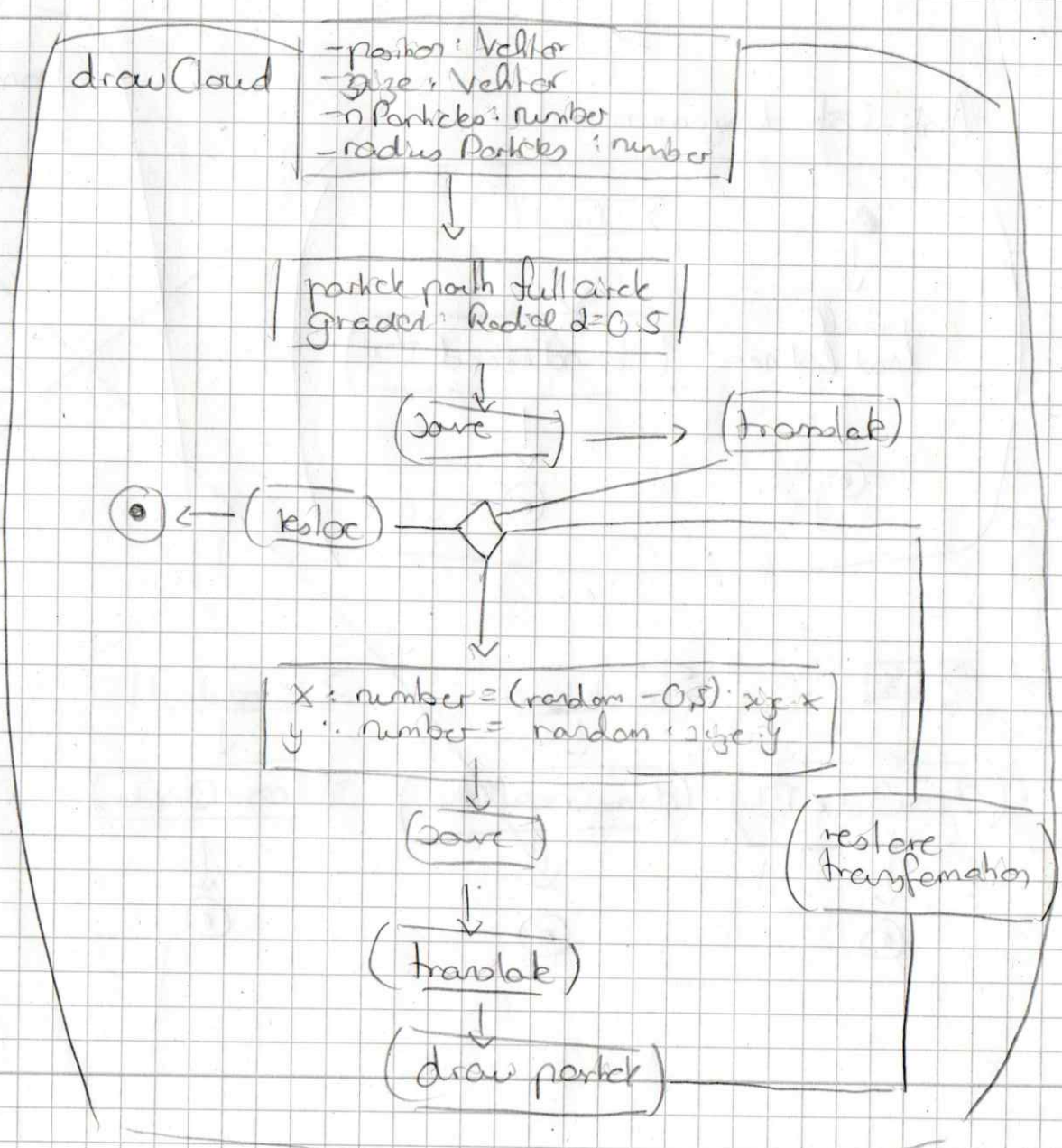
Aktivitätsdiagramm

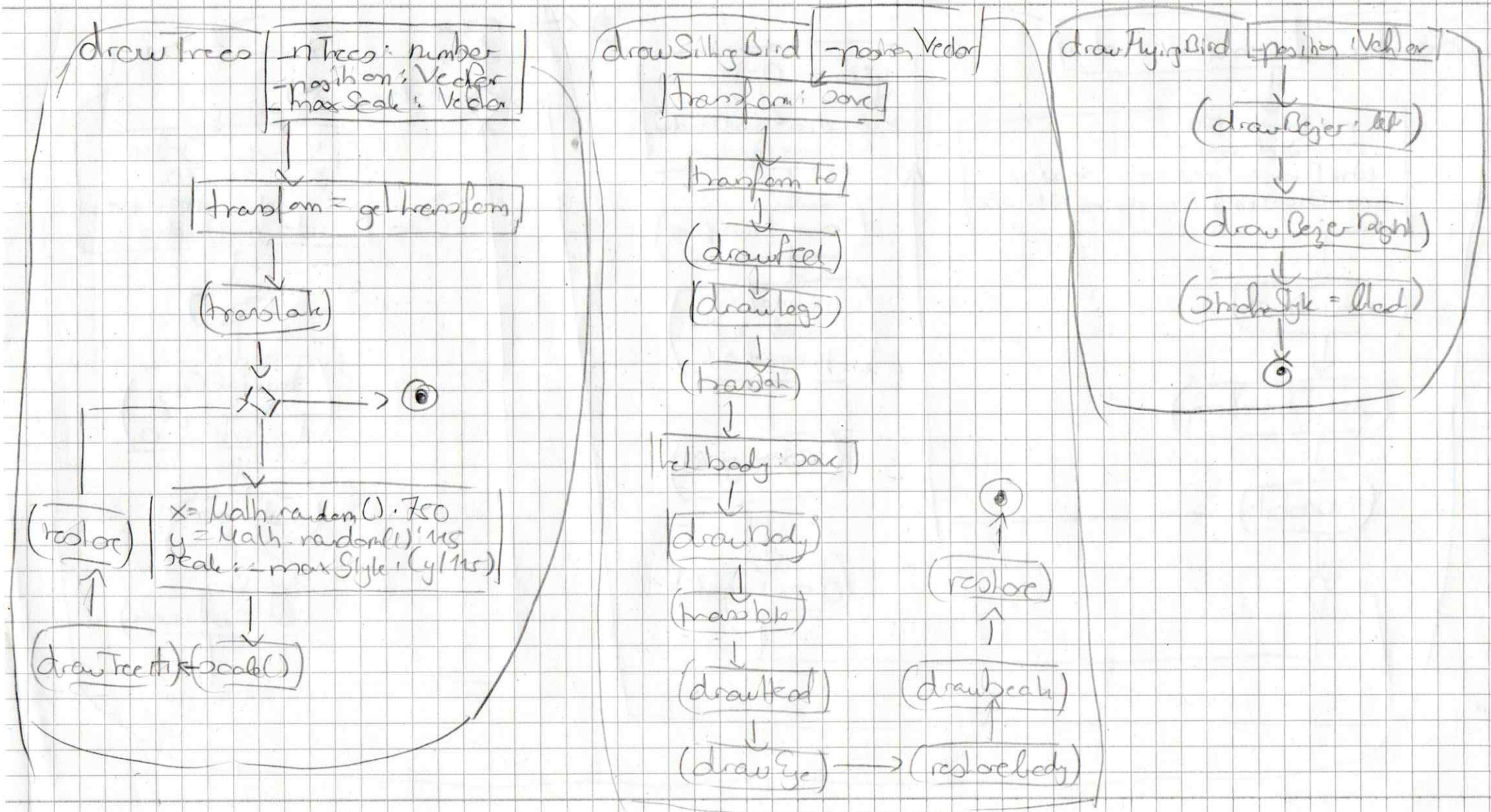


2

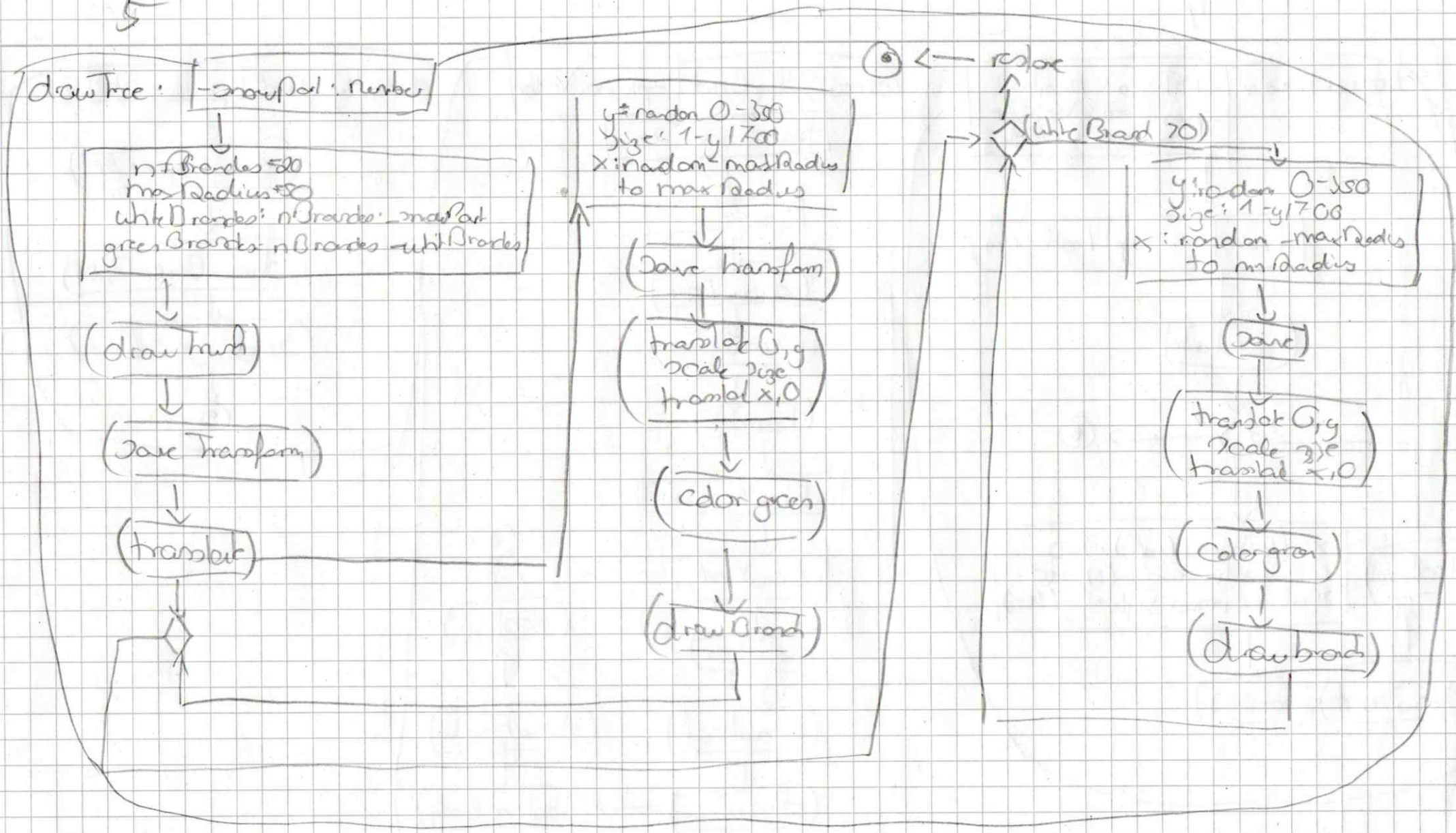

drawStatic
let horizon: number
let sunPosVar: Vector
let mountainPosVar: Vector
let cloudPos: Vector
let cloudSize: Vector
let treeNoiseScale: Vector
let snowmanPos: Vector
let snowPos: Vector
(drawBackground())
(drawSun())
(drawMountains())
(drawCloud())
(drawTrees())
(drawSnowman())
(drawSnow())

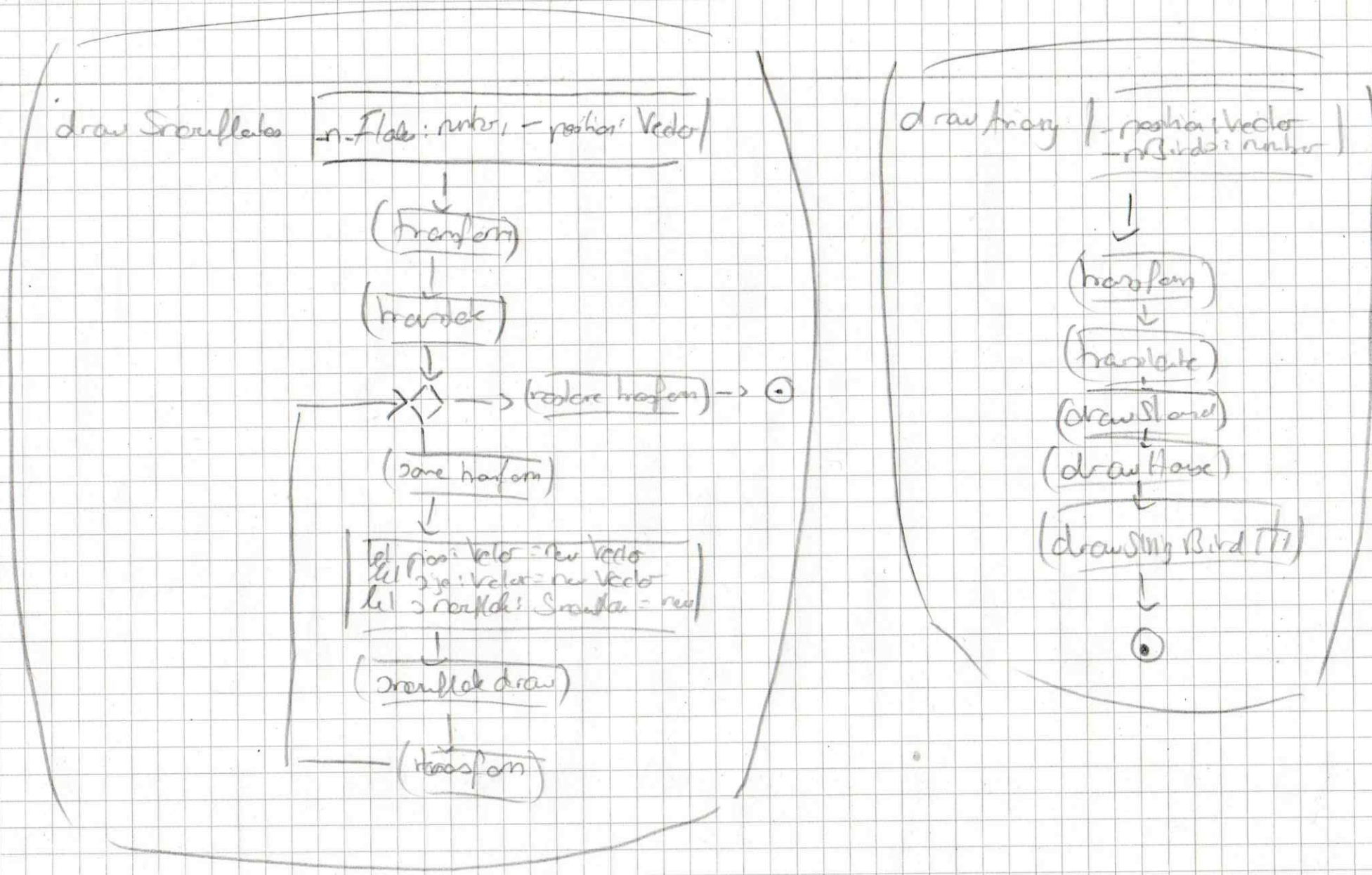

drawSun
-position: Vector
r1: number 20
r2: number 150
gradient: Radial Gradient
transparent 1.0
yellow rgb
save
translate
draw circle r2
restore transform


drawCloud
-position: Vector
-size: Vector
-nParticles: number
-radius Particles: number
particle path full circle
gradient: Radial d=0.5
save
translate
restore
x: number = (random -0.5) * size.x
y: number = random * size.y
save
translate
draw particle
restore transformation


drawTree()
-nTrees : number
-position : Vector
-maxScale : Vector
transform = getTransform
translate
x = Math.random() * 750
y = Math.random() * 145
scale : maxStyle * (y/145)
recolor
drawTree(n-1)
drawSilkyBird
-position Vector
transform : bool
transform to
drawTail
drawLegs
translate
translate
tail body : bool
drawBody
translate
drawHead
drawEye
recolor
drawBeak
restoreBody
drawFlyingBird
-position : Vector
drawBezier : left
drawBezier Right
strokeStyle = Red

5

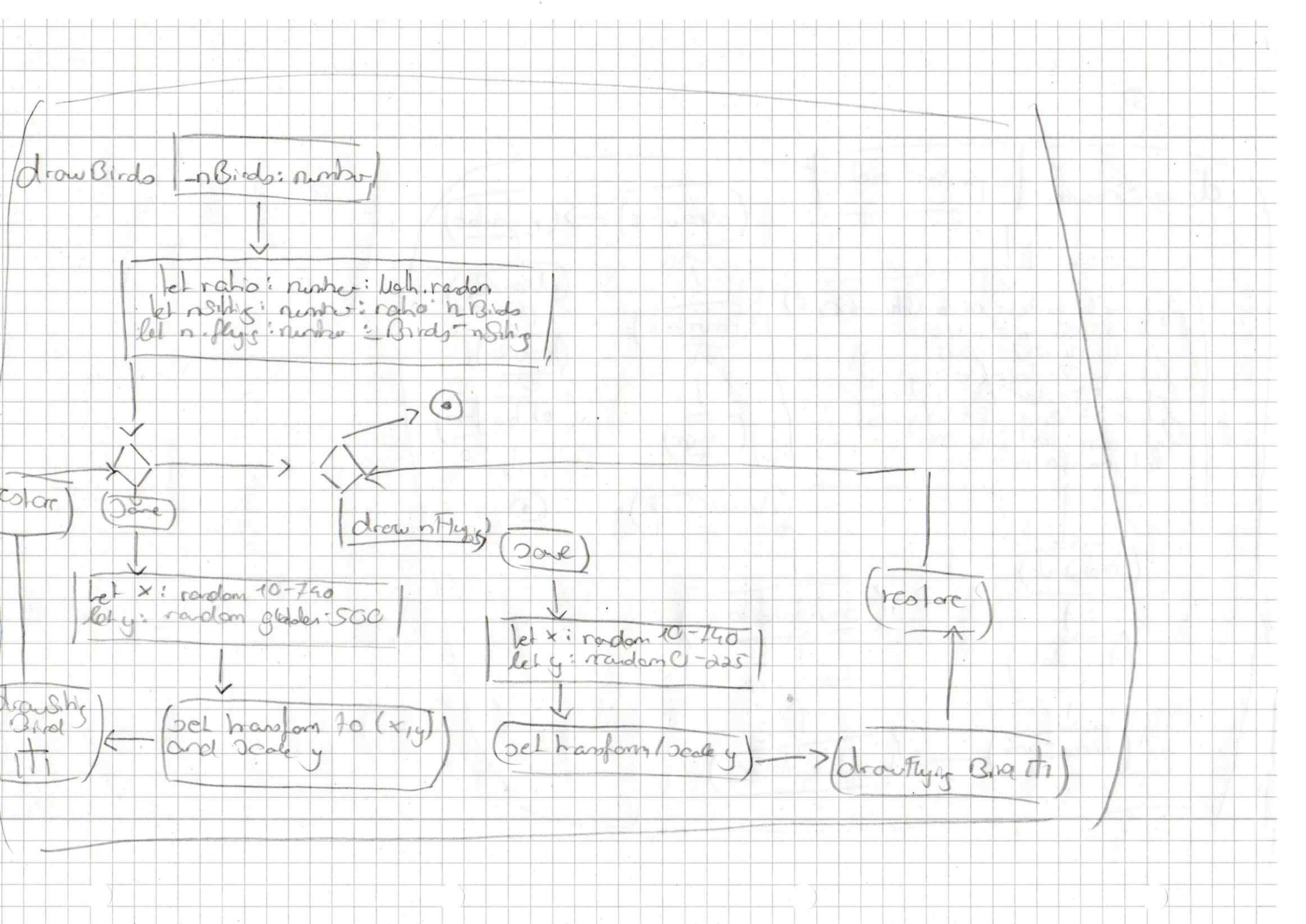

drawTree: nrowPart: Number
nBranches=20
maxRadius=50
whiteBranches: nBranches * rowPart
greenBranches: nBranches - whiteBranches
(drawTrunk)
(save transform)
(translate)
y= random 0-350
size: 1-y/700
x: random maxRadius to maxRadius
(save transform)
(translate 0,y scale size translate x,0)
(color green)
(drawBranch)
(color)
(whiteBranch >0)
y: random 0-350
size: 1-y/700
x: random -maxRadius to maxRadius
(save)
(translate 0,y scale size translate x,0)
(color green)
(drawBranch)

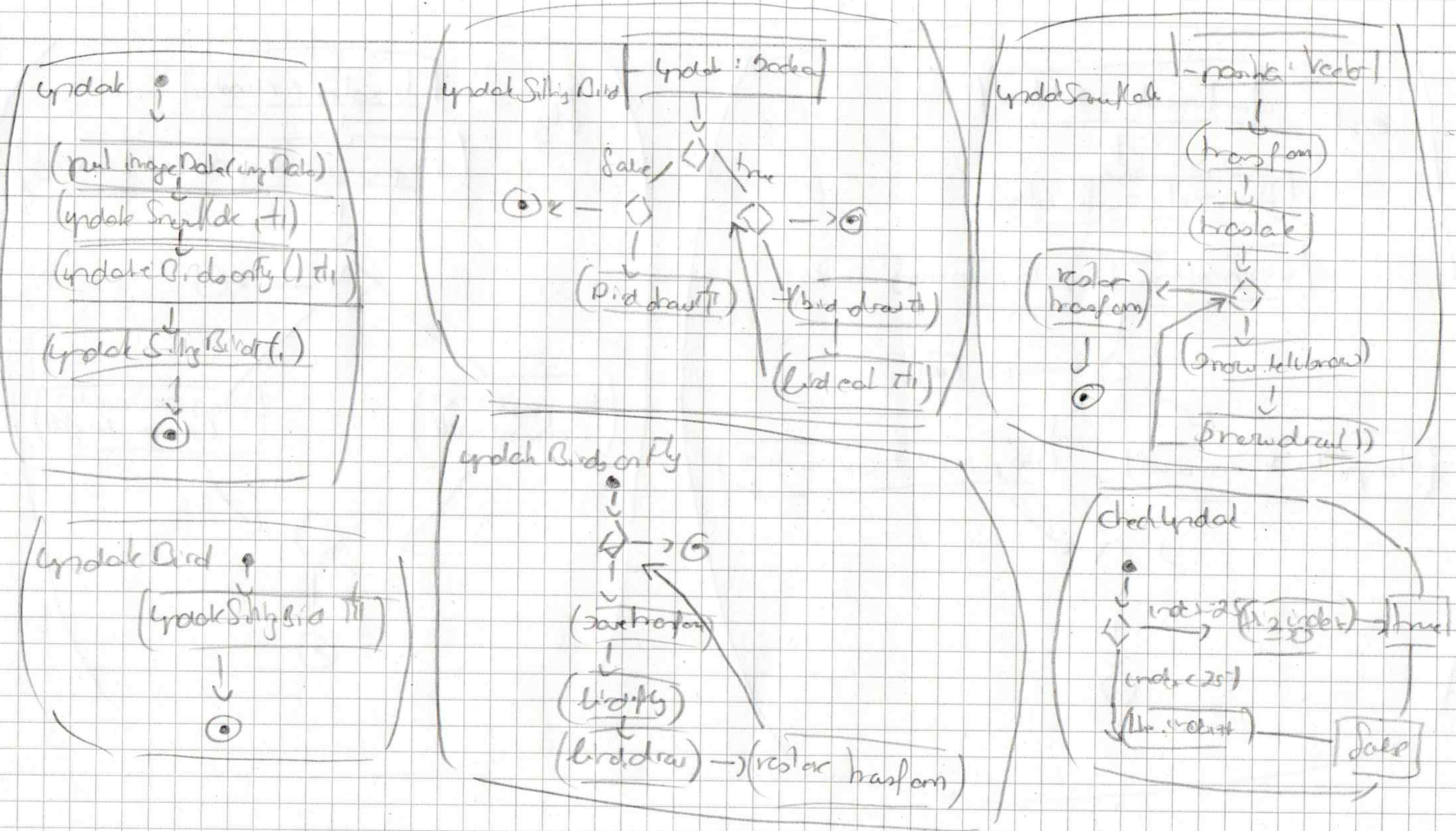

drawSnowflakes
-n_Flakes: number
-position: Vector
transform
translate
<>
restore transform
save transform
for i=0; i<n_Flakes; i++
snowflakes[i].draw()
transform

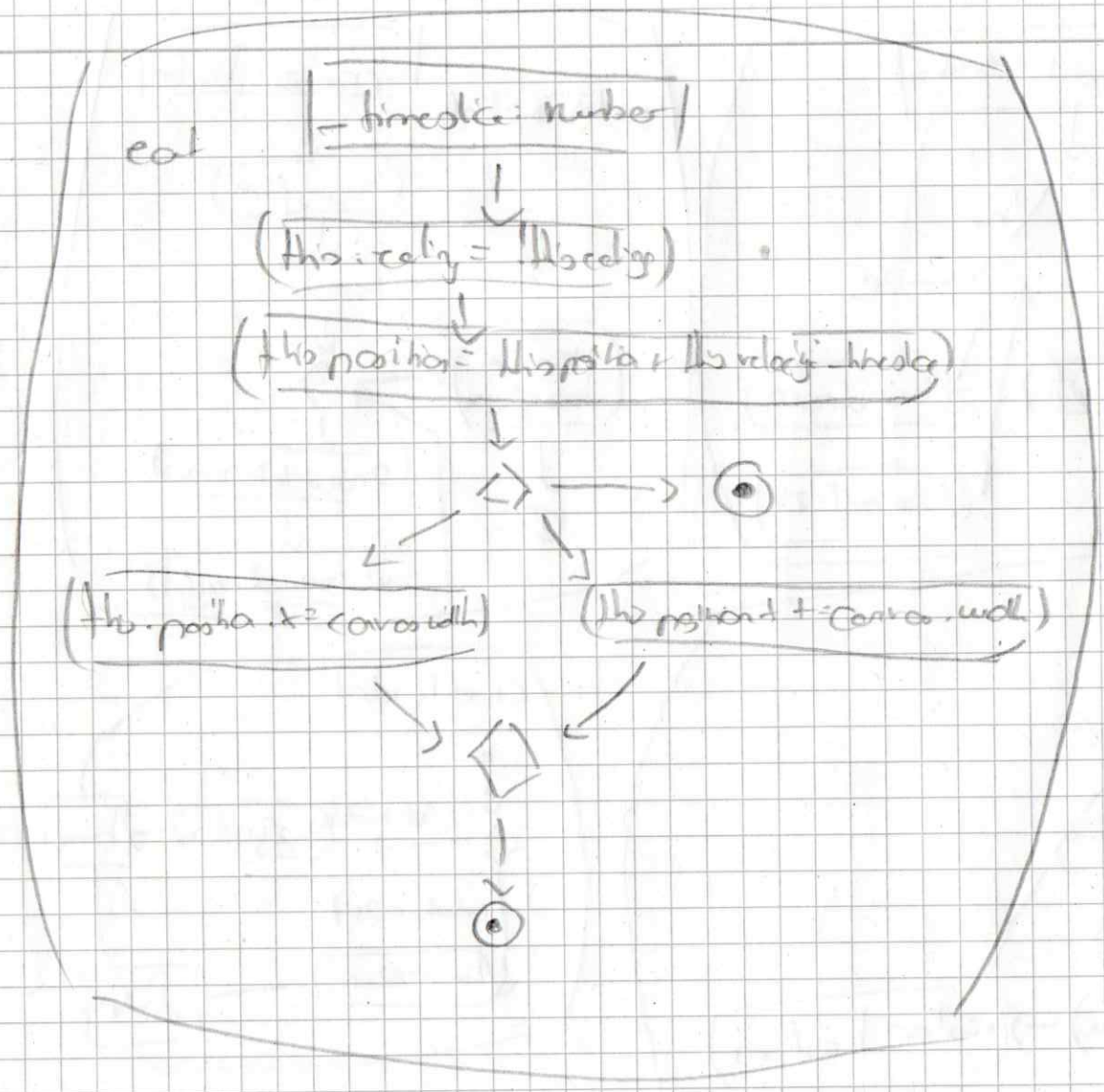
drawTrain
-position: Vector
-nBirds: number
transform
translate
drawStand
drawHorse
drawWingBird(1)

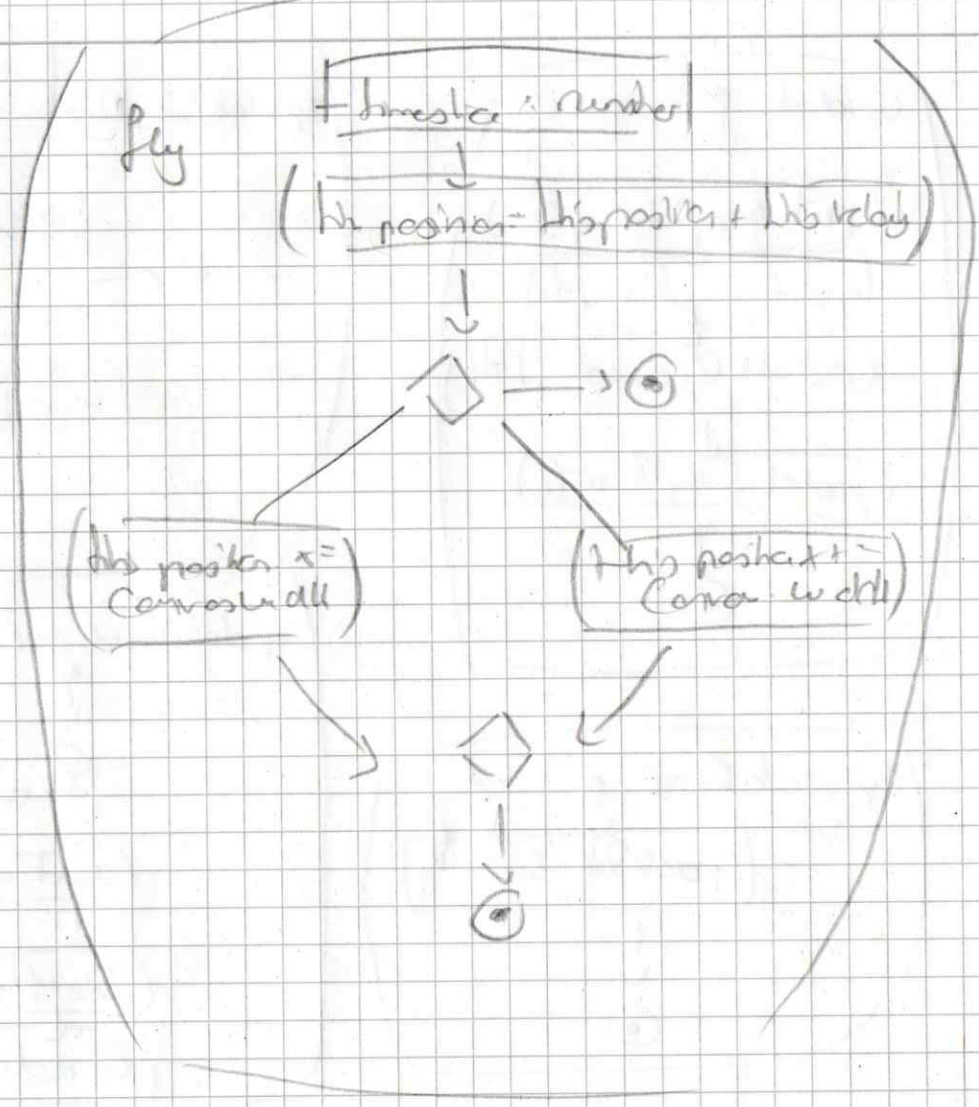
5

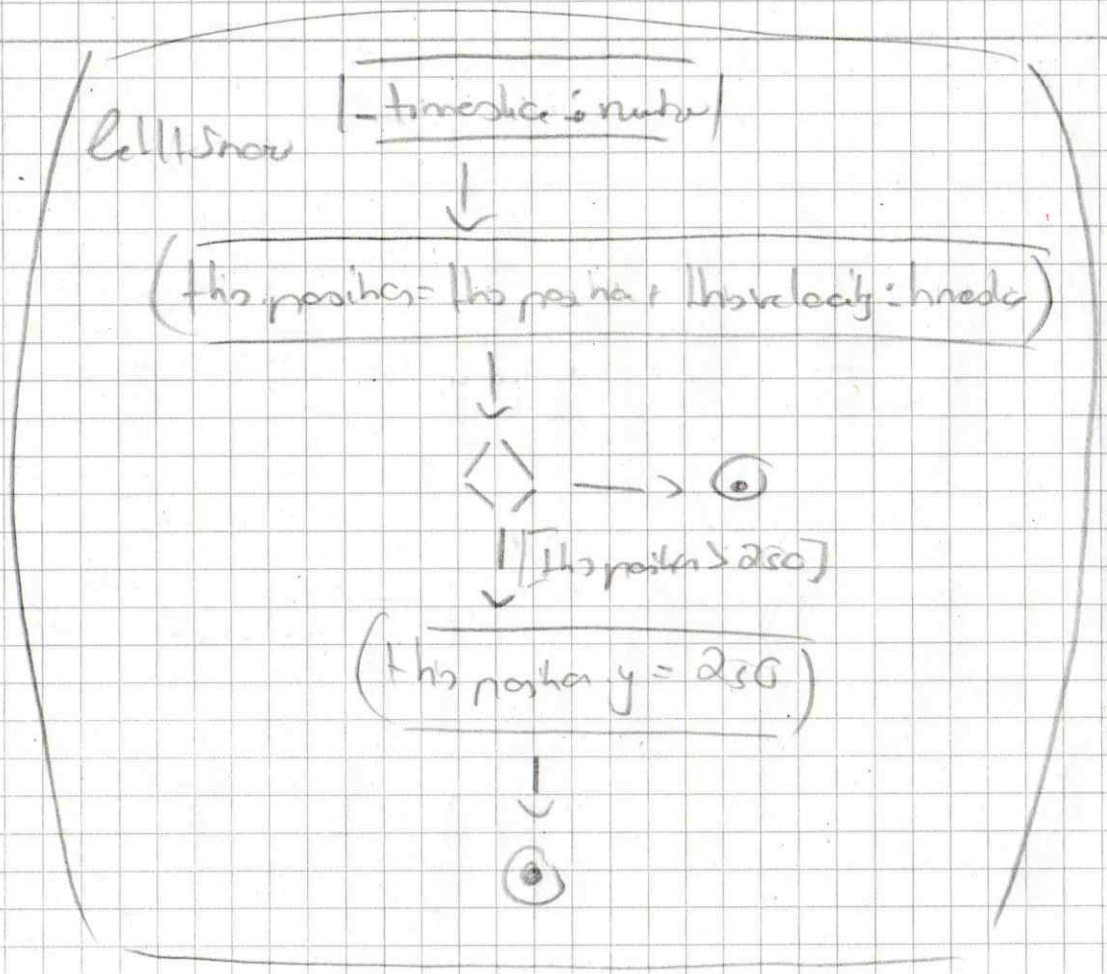

drawSnowman
position: Vector
let r1: number = randomBetween(20, 70)
let r2: number = r1/3
let r3: number = r2/3
let x1: Vector = {x: 0, y: r1}
let x2: Vector = {x: 0, y: r1}
let x3: Vector = {x: 0, y: r2}
let color: default thing color
let transform
translate
translate x1
drawCircle
translate to x2
drawCircle
translate to x3
drawCircle
drawEye
translate
drawEye
reset
set transform


drawBirds
nBirds: number
let ratio: number = Math.random
let nSitting: number = ratio * nBirds
let nFlying: number = Birds - nSitting
done
drawSitting Bird
drawFlying Bird
restore
let x: random 10-740
let y: random gladder-500
let x: random 10-740
let y: random 0-225
set transform to (x,y) and scale y
set transform / scale y
drawFlying Bird


update
updateSillyBird
updateBirdsOnFly
checkUpdate
false
true
transform
restore transform


eat
+=timeslice : number
this.velx = this.velx
this.position = this.position + this.velx * timeslice
this.position.x = canvas.width
this.position.x = canvas.width


fly
+timeslice : number
this.position = this.position + this.velocity
this.position.x = canvas.width
this.position.x = canvas.width


fallSnow
|-timeslice : number|
(this.position = this.position + this.velocity : timeslice)
[this.position > 250]
(this.position.y = 250)