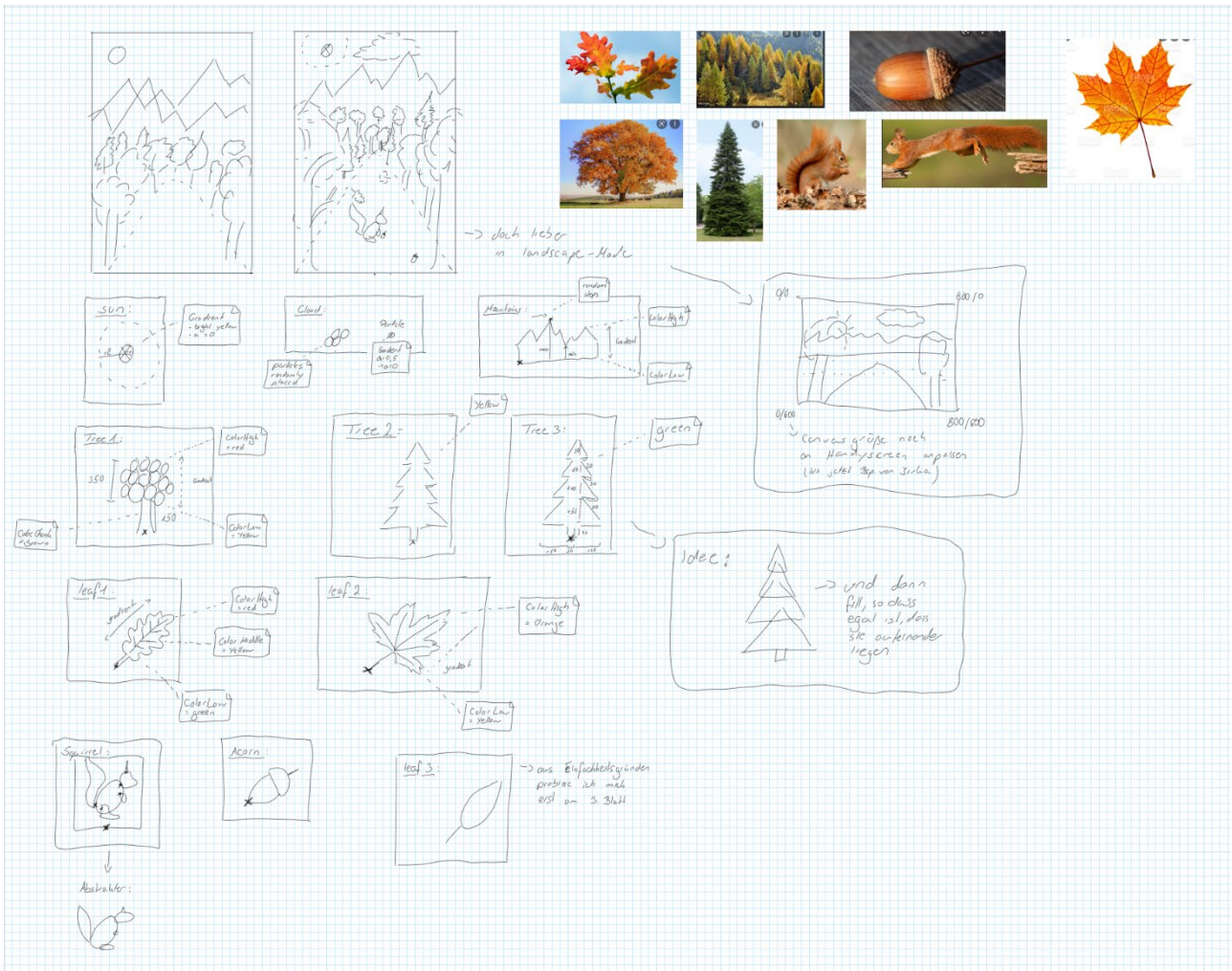
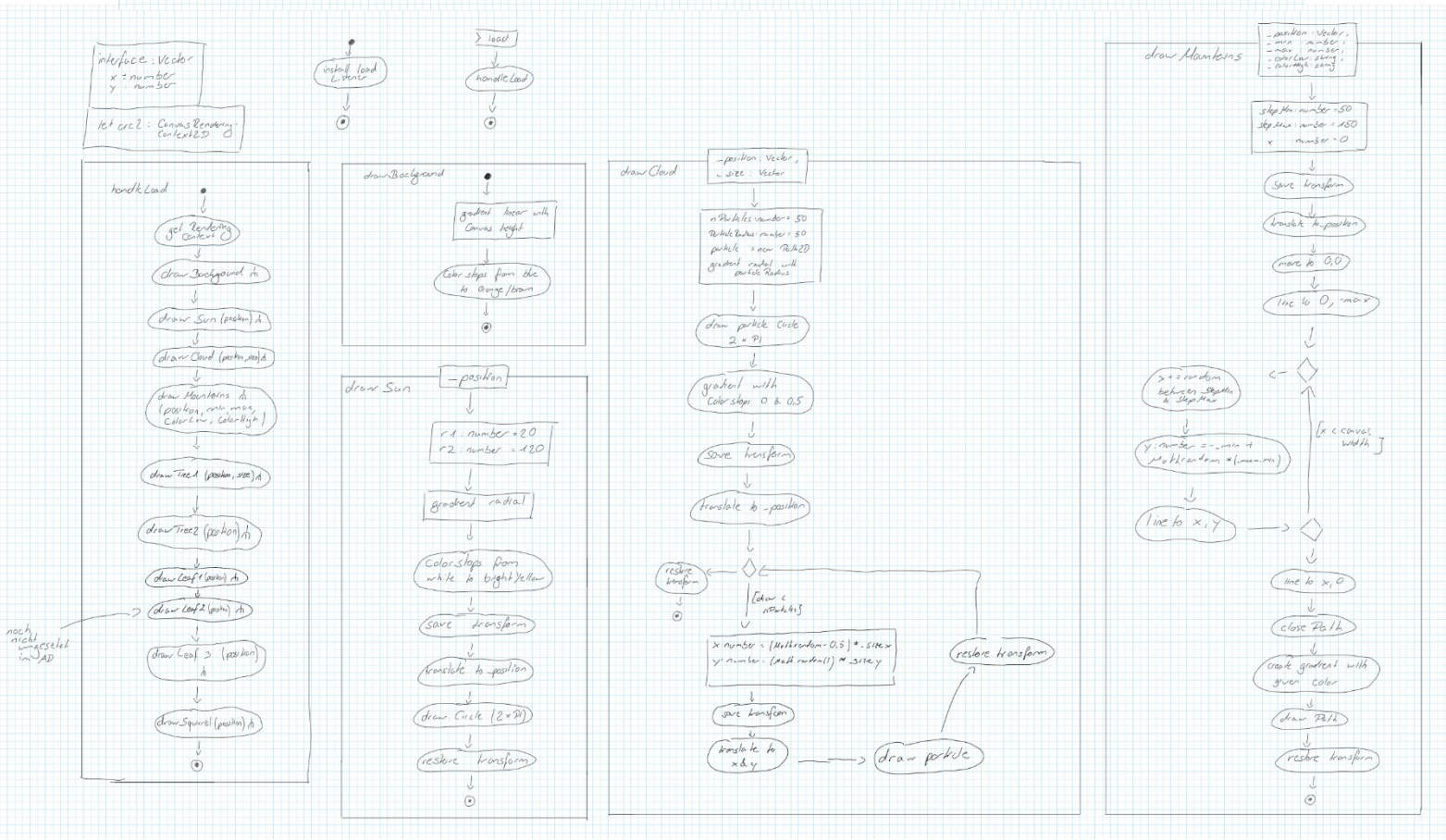
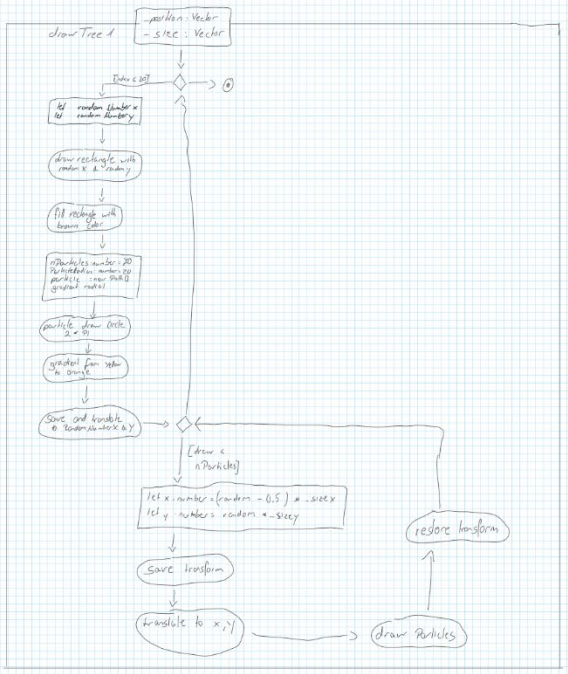
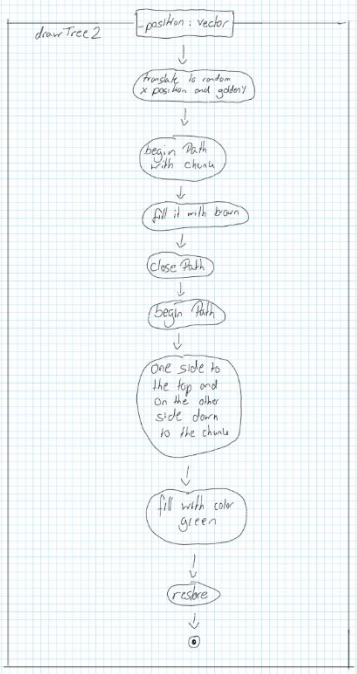
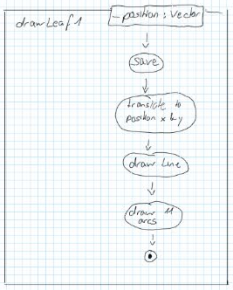
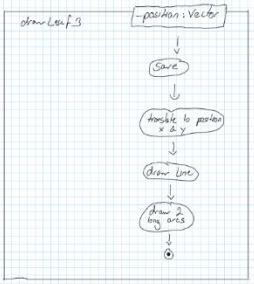

→ doch lieber in landscape-Mode
Sun:
Gradient = bright yellow
Cloud
particles randomly placed
Mountains:
random steps
ColorHigh
ColorLow
Tree 1:
ColorHigh = red
ColorLow = Yellow
Tree 2:
Yellow
Tree 3:
green
0/0
800/0
0/600
800/600
Canvas größe noch an Handyscreen anpassen (als jetzt 16:9 von Sirka)
Idee:
→ und dann fill, so dass egal ist, dass sie aufeinander liegen
leaf 1:
ColorHigh = red
ColorMiddle = Yellow
ColorLow = green
leaf 2:
ColorHigh = Orange
ColorLow = Yellow
Squirrel:
Acorn:
leaf 3:
→ aus Einfachheitsgründen probiere ich mich erst am 3. Blatt
Acorn:

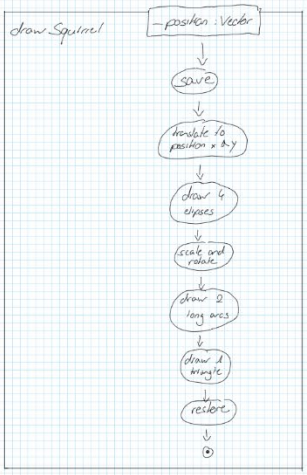

interface: Vector
x: number
y: number
let ctx2: CanvasRenderingContext2D
install load listener
load
handleLoad
handleLoad
get Rendering Context 2D
draw Background (ctx)
draw Sun (position) (ctx)
draw Cloud (position, size) (ctx)
draw Mountains (ctx, position, min, max, ColorLow, ColorHigh)
draw Tree1 (position, size) (ctx)
draw Tree2 (position) (ctx)
draw Leaf1 (position) (ctx)
draw Leaf2 (position) (ctx)
draw Leaf 3 (position) (ctx)
draw Squirrel (position) (ctx)
noch nicht umgesetzt in JAD
drawBackground
gradient linear with Canvas height
Color stops from blue to orange/brown
draw Sun
_position
r1: number = 20
r2: number = 120
gradient radial
Color stops from white to bright yellow
save transform
translate to position
draw Circle (2×PI)
restore transform
draw Cloud
_position: Vector,
_size: Vector
nParticles: number = 50
Particle Radius: number = 50
particle = new Path2D
gradient radial with particle Radius
draw particle Circle 2×PI
gradient with Color stops 0 & 0.5
save transform
translate to _position
restore transform
[draw x nParticles]
x: number = (Math.random - 0.5) * _size.x
y: number = (math random()) * _size.y
save transform
translate to x & y
draw particle
restore transform
draw Mountains
_position: Vector,
_min: number,
_max: number,
_colorLow: string,
_colorHigh: string
stepMin: number = 50
stepMax: number = 150
x: number = 0
save transform
translate to position
move to 0,0
line to 0, -max
x += random between stepMin & stepMax
y: number = -_min + Math.random * (_max - _min)
line to x, y
[x < canvas width]
line to x, 0
close Path
create gradient with given color
draw Path
restore transform


drawTree1
_position : Vector
_size : Vector
Index < 30?
let random Number x
let random Number y
draw rectangle with random x & random y
fill rectangle with brown color
nParticles : number = 20
for index index < nParticles index++
particle = new Shell(g, graphics context)
particle.draw Circle
g.strokeStyle from yellow to orange
save and translate to RandomNumberX & Y
[draw < nParticles]
let x number = (random - 0.5) * _sizeX
let y number = random * _sizeY
save transform
translate to x,y
draw Particles
restore transform


drawTree2
_position : Vector
translate to random x position and golden y
begin Path with chunk
fill it with brown
close Path
begin Path
one side to the top and on the other side down to the chunk
fill with color green
restore


drawLeaf1
_position : Vector
save
translate to position x & y
draw Line
draw 4 arcs


drawLeaf3
_position : Vector
save
translate to position x & y
draw Line
draw 2 long arcs


drawSquirrel
_position : Vector
save
translate to position x & y
draw 4 elypses
scale and rotate
draw 2 long arcs
draw 1 triangle
restore