

function drawSnow

Canvas inner width

function snow

Canvas inner height

function draw  
(x:)

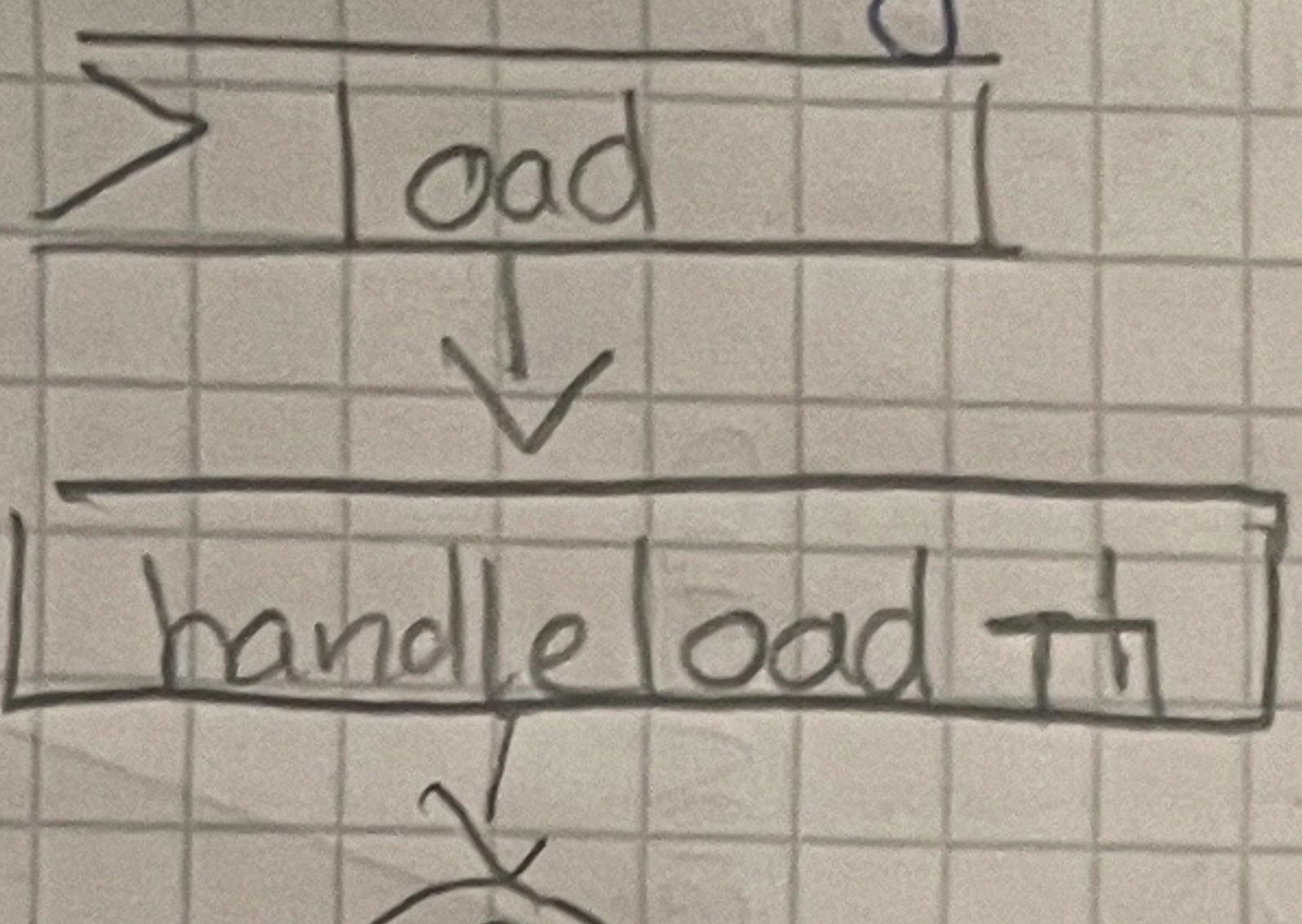
let horizon =  
ctx.canvas.height \* 0.6

max  
+  
min  
= 10

andré Schäemann

## A08.2 - Aktivitätsdiagramm

install load  
listener



Vector

x: number  
y: number

handle load  
get Rendering Context  
set horizont = canvas  
drawbackground();  
drawSun (position th)

drawMountains (pos., size) th

drawhouse (pos)

loch (pos)

Schnemann (pos)

birds

drawBirdat house(pos)

sunPos

snowPos()

```

    i: number = 0;
    increase i by 1
    if [i < 250] then
        snowPos
        snowPosY
        th ← drawSnow(x, y)
    end
  
```

draw Snow [snowPosX, snowPosY]

let p = Math.random() \* 30

ctx.save()

ctx.translate(-position)

ctx.arc(0, 0, p, 2 \* Math.PI)

ctx.fill

ctx.closePath

ctx.restore -> eye

drawSun [positionVector]

x = 100, y = 25

r1 = 25

r2 = 175

gradient.createRadialGradient

set colorstop at 1  
yellow, > hsl(353, 42%, 76%)

save()

translate(-position.x, -position.y)

ctx(0, 0, r2, 0, 2 \* Math.PI)

ctx.fill()

ctx.restore();

birdhouse

= positiv Vektor  
 $x: 150 \quad y: 2000$

ctx. save

↓  
translate( $-postx - posy$ )

↓  
beginPath()

↓  
moveTo

↓  
lineTo

↓  
closePath()

↓  
fillStyle = "brown"

↓  
restore

loch { $x: 150 \quad y: 82000$ }

ctx. save()

↓  
translate( $postx : posty$ )

↓  
ctx.beginPath

↓  
ctx.arc(0, -250, 150, 2 \* Math.PI)

↓  
ctx.fillStyle = "black";

↓  
fill()

↓  
restore

drawmountains

```
_position: Vector  
_min: number  
_max: number  
_colorLow: string  
_colorHigh: string
```

```
stepMin: 60;  
stepMax: 150;  
let x: number = 0
```

```
save transform  
translate(_position.x, _position.y)  
begin path()  
ctx.moveTo(0, 0)  
ctx.lineTo(0, -max)
```

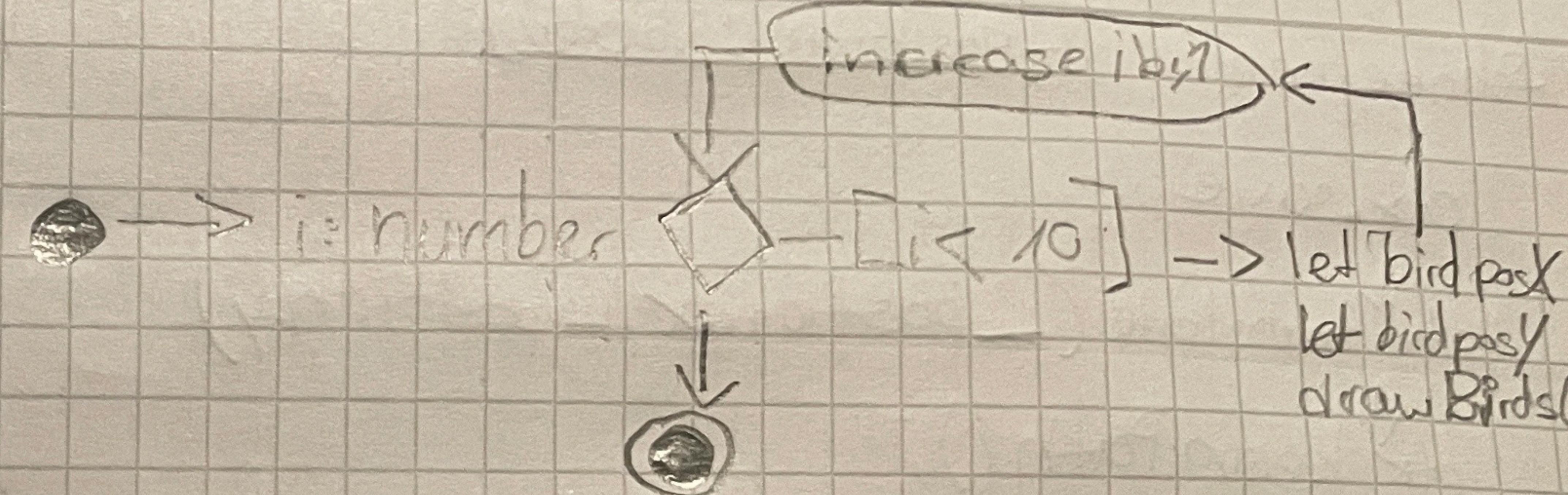
```
x += stepMin + Math.random() * (stepMax - stepMin);  
let y: number = _min - Math.random() * (_max - _min)
```

lineTo(x0)

line to  $x_{10}$  → close path → execute gradient with give color

restore transform  $kt$  drawPath

birds()



drawBirds()

ctx.save

ctx.translate(\_position.x, \_position.y)

ctx.moveTo(0, 0); ctx.lineTo(0, 0)

ctx.fillStyle = "hsl(" + Math.random() \* 180

ctx.fill()

ctx.beginPath()

ctx.lineTo(-5, 0)

ctx.fillStyle = "white"

ctx.stroke → ctx.restore

Schneemann `[Position : Vector]`

ctx.save

ctx.translate(-position.x, -position.y)

ctx.beginPath();

↓

ctx.arc(0, -(r1 = 80; r2 = 100; r = 150))

↓

ctx.closePath();

↓

ctx

ctx.fillStyle = "#000000";

↓

ctx.beginPath();

↓

ctx.arc(25, 450, 20, 2 \* Math.PI)

↓

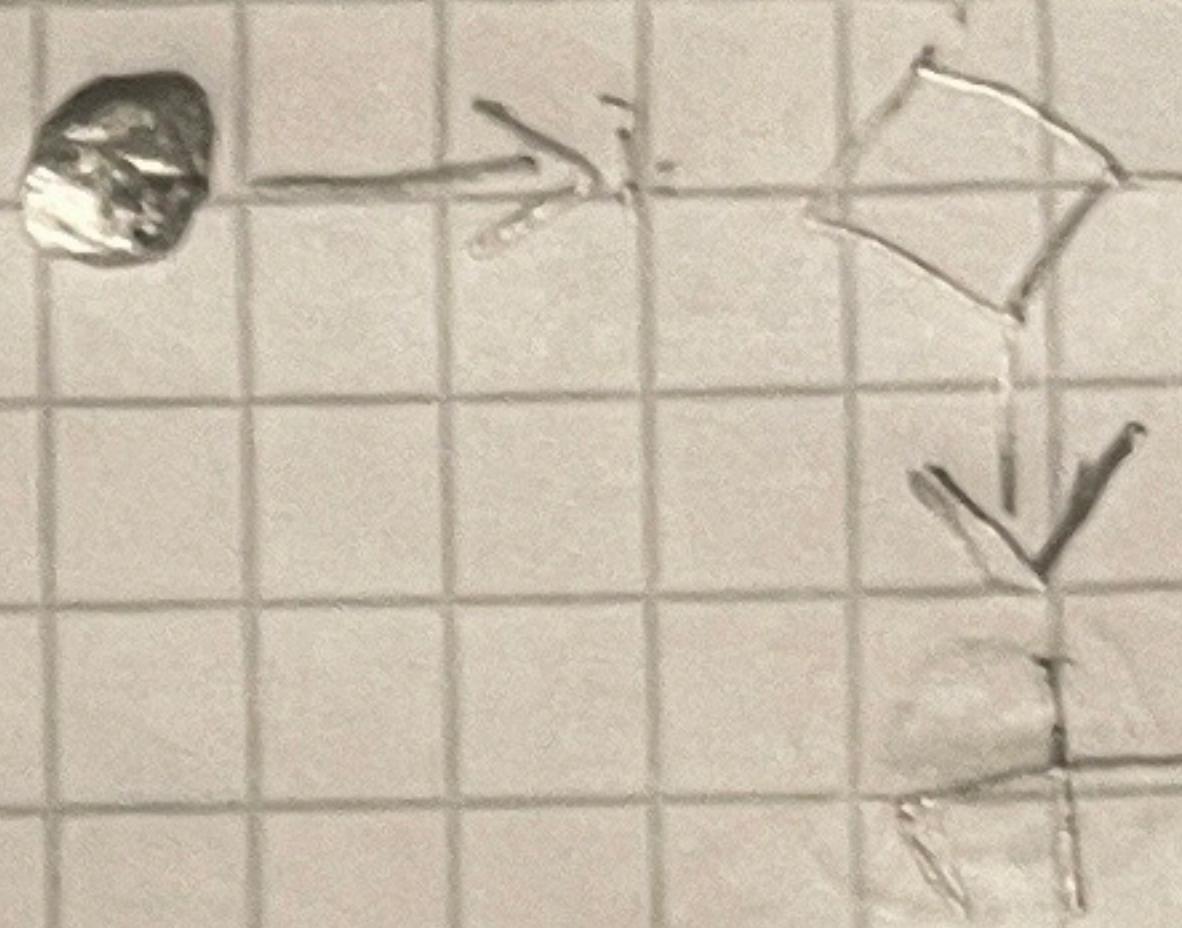
ctx.fill();

↓

ctx.stroke();

drawTrees

{



[x,y]

IN

draws (x: position  
y: stem)

drawTree (x: positionbaum, y: stammhohe)

↓

ctx.save();

ctx.beginPath();

shadow

↓

ctx.moveTo

↓

ctx.lineTo

↓

ctx.stroke

↓

ctx.restore

↓

ctx.beginPath();