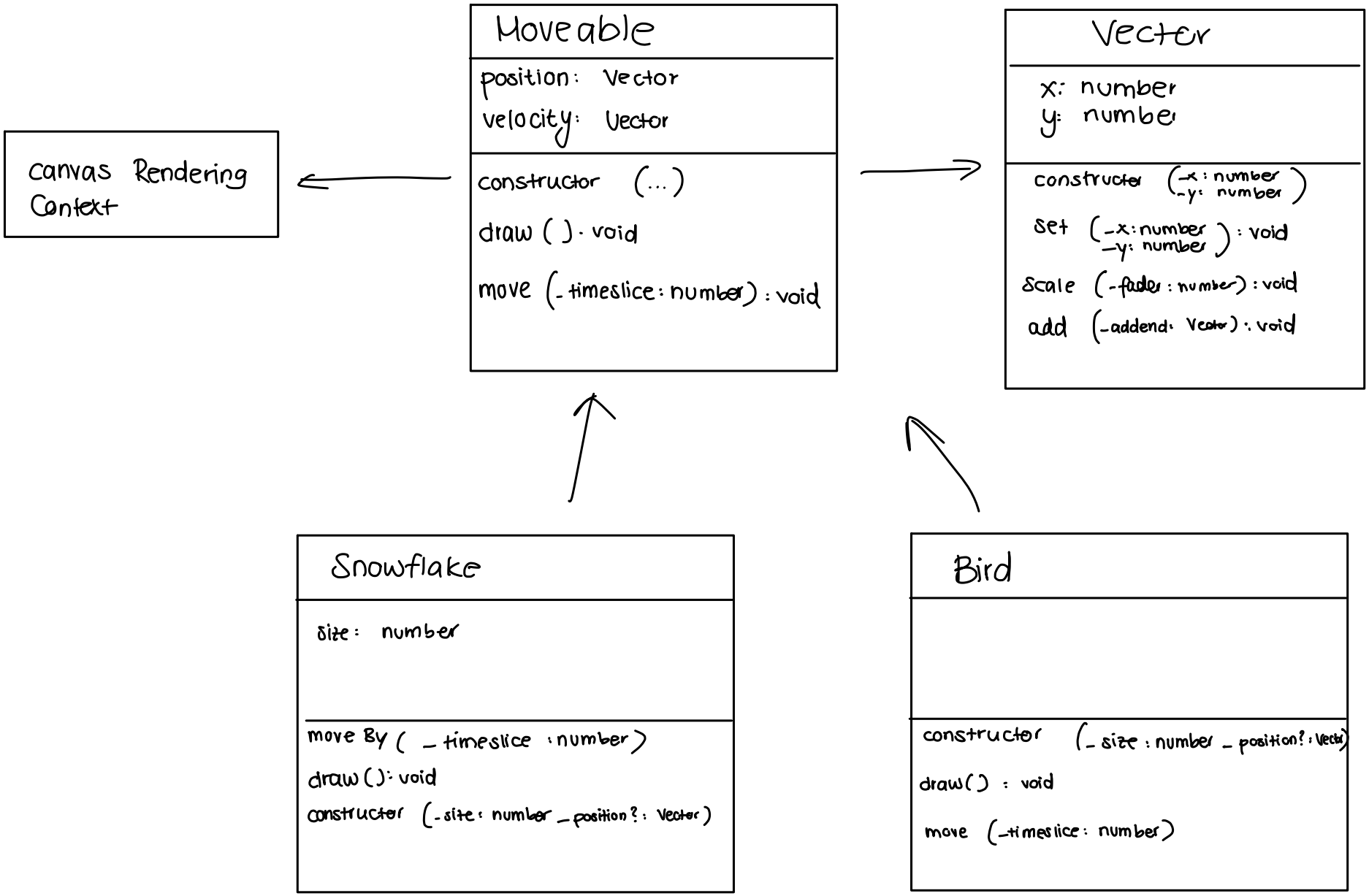
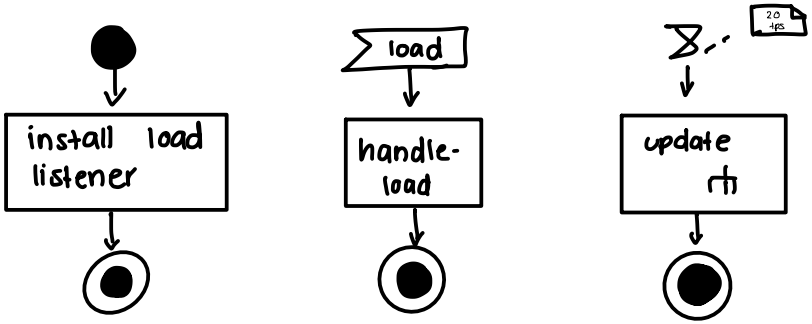
Klassendiagramm

14.01.23



Aktivitätsdiagramm



crc 2

golden

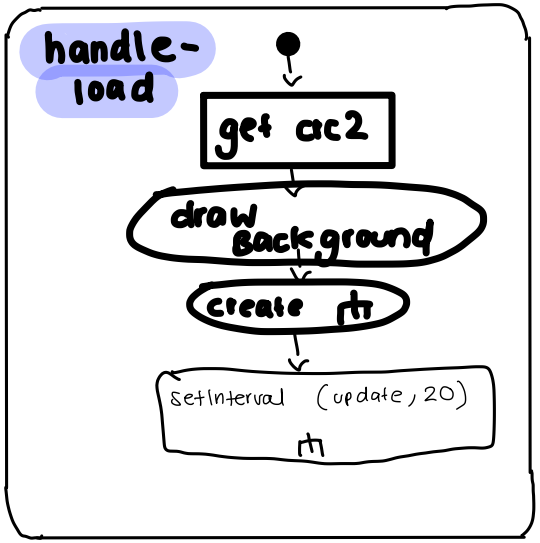
x: number

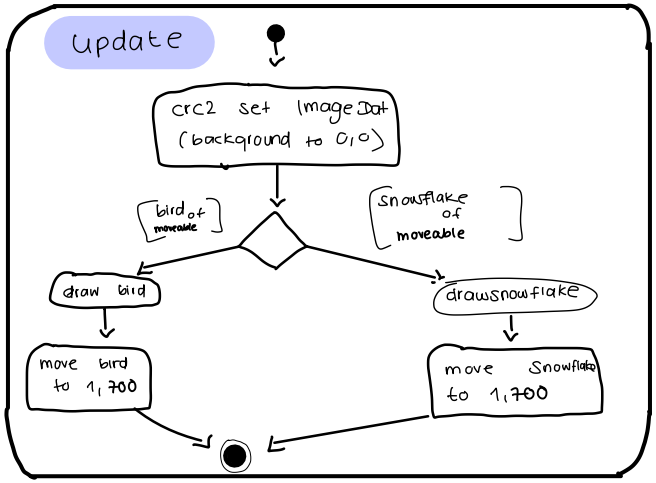
y: number

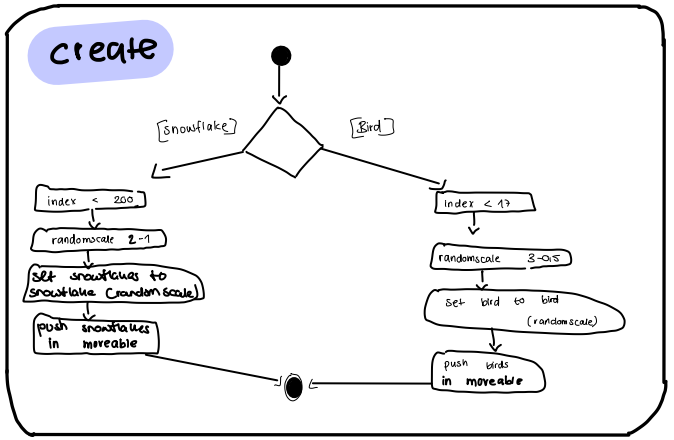
background: ImageData

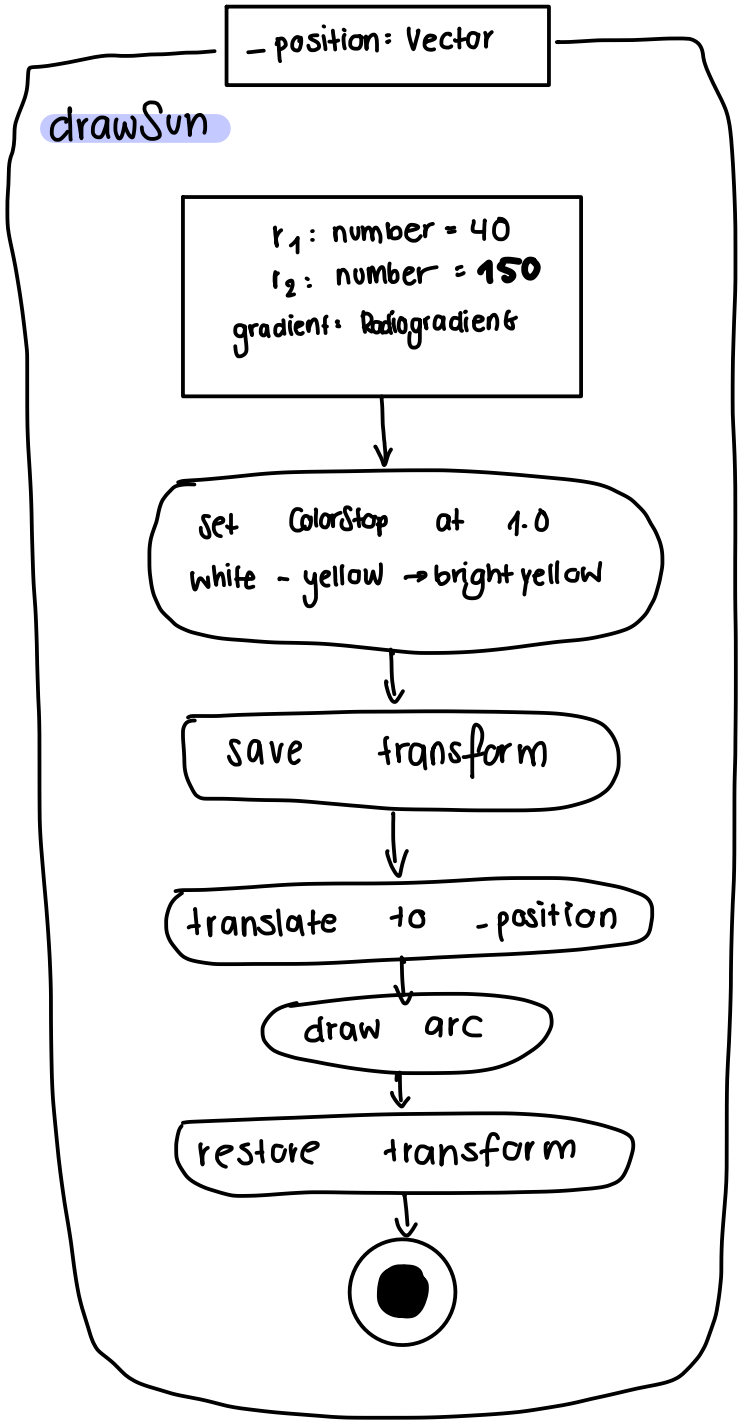
snowflakes: Snowflake []

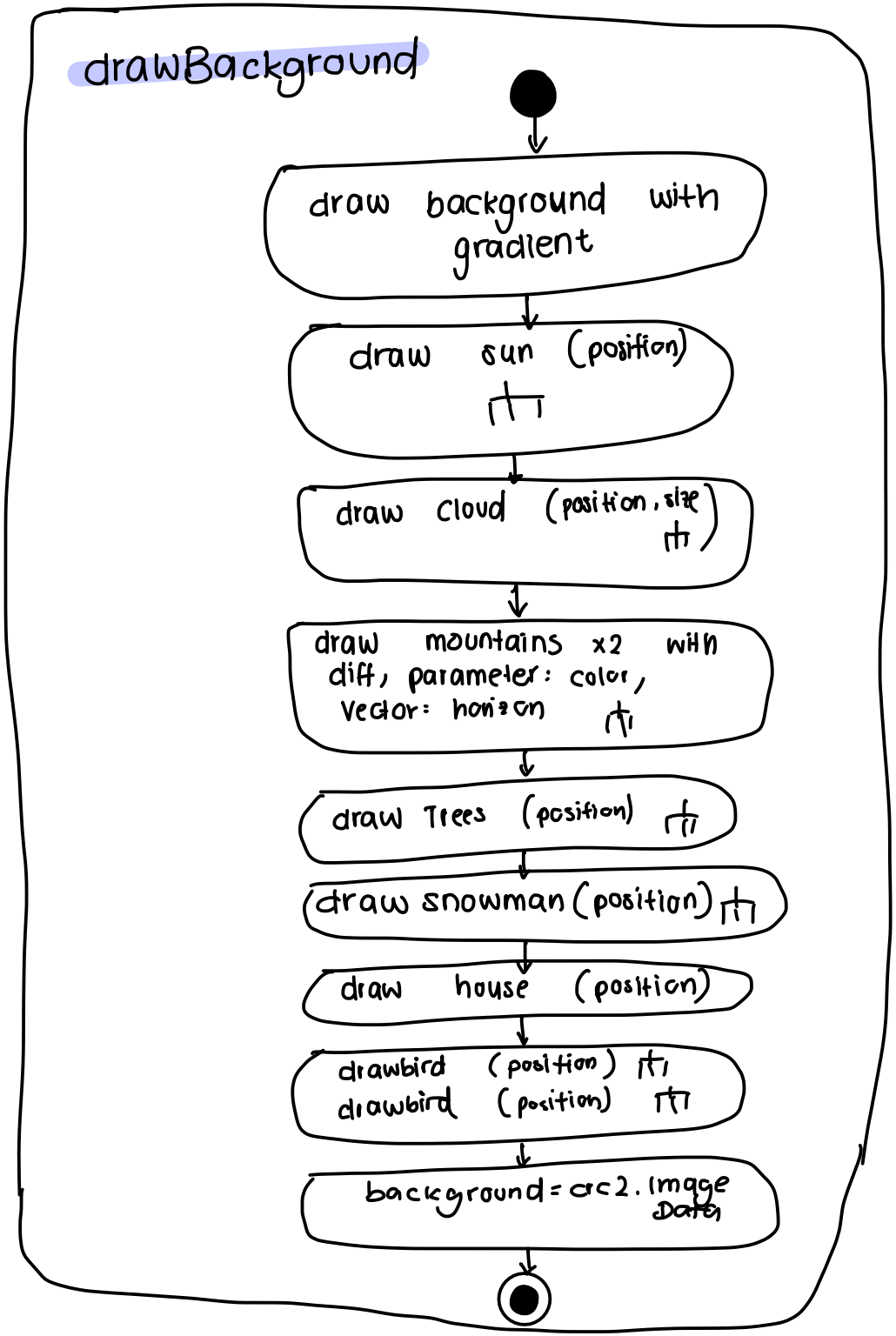
birds: Bird []

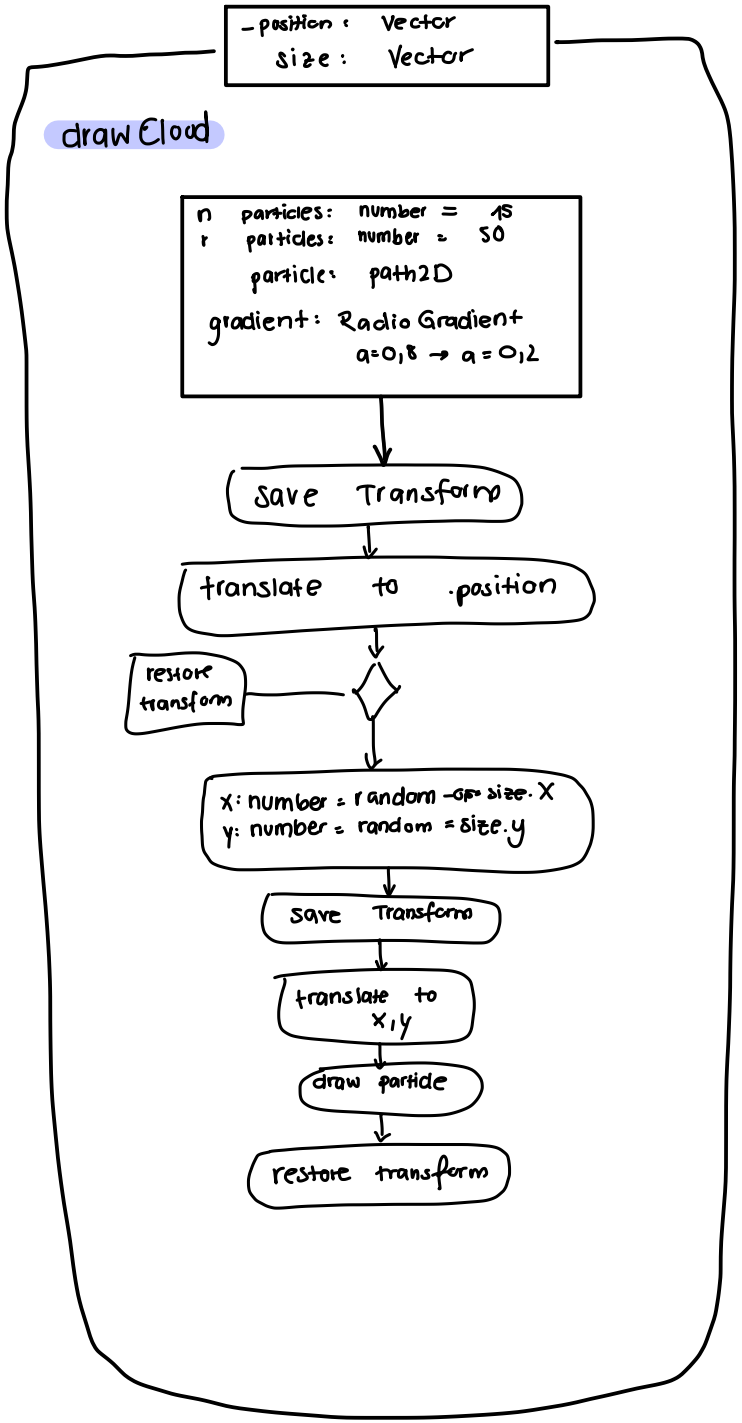


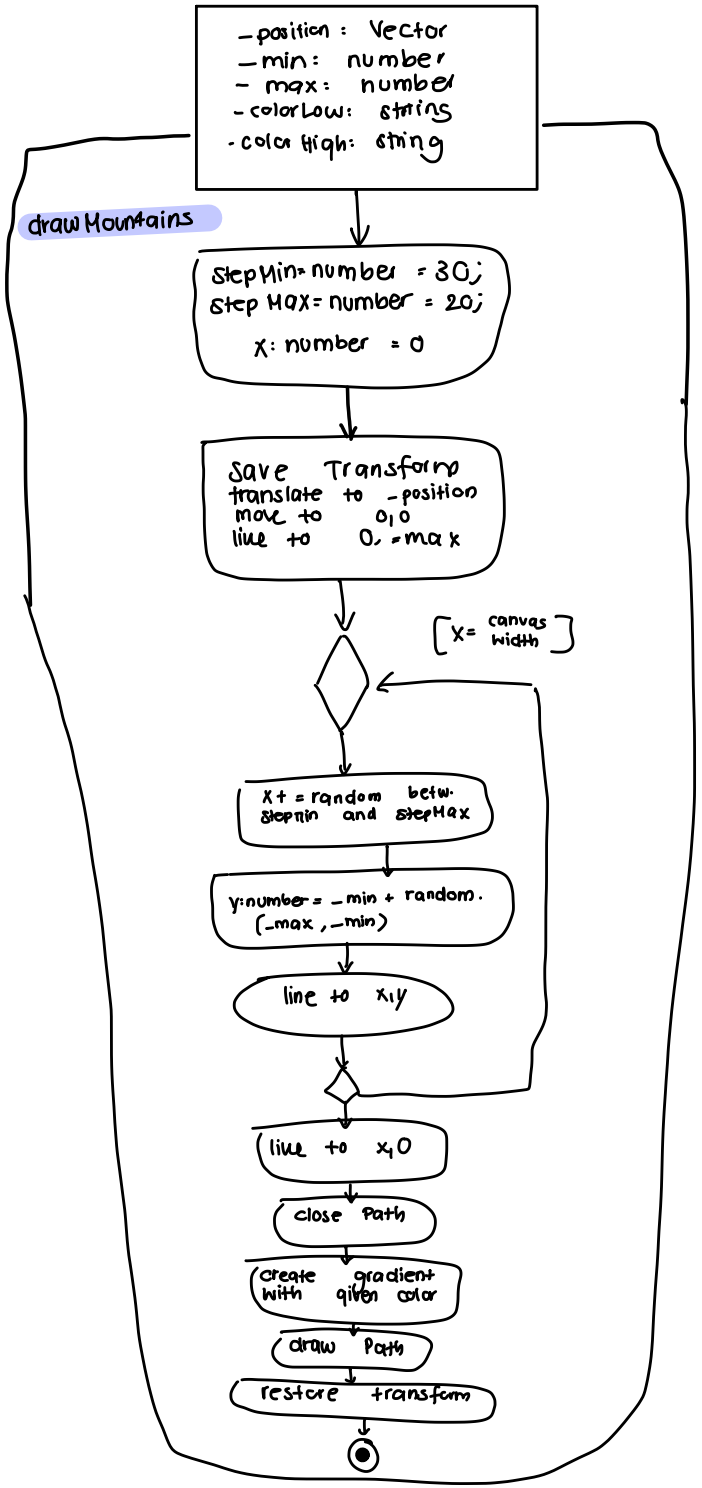


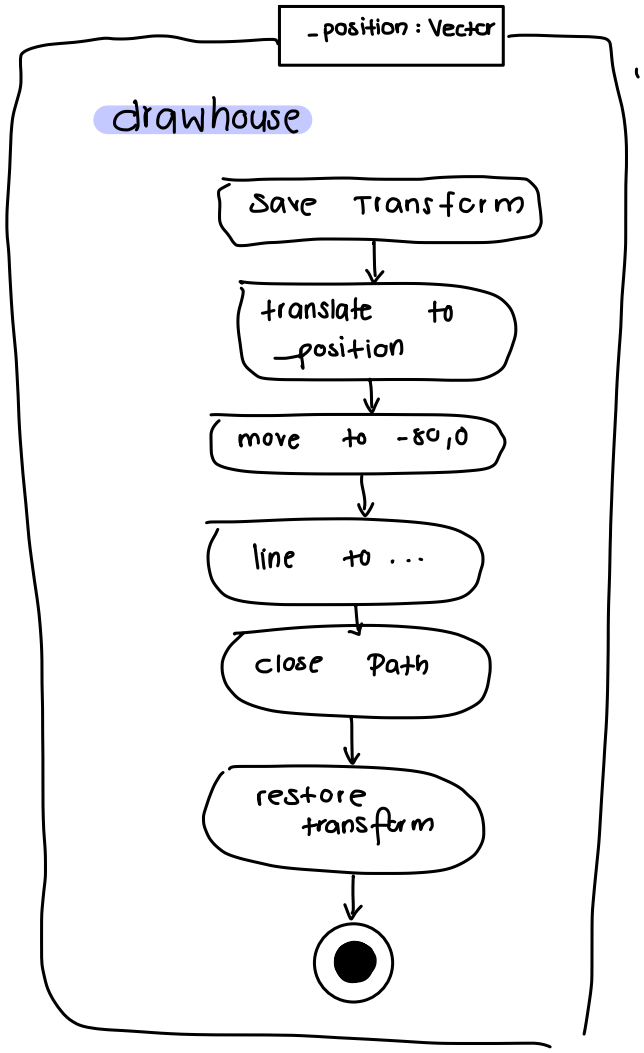


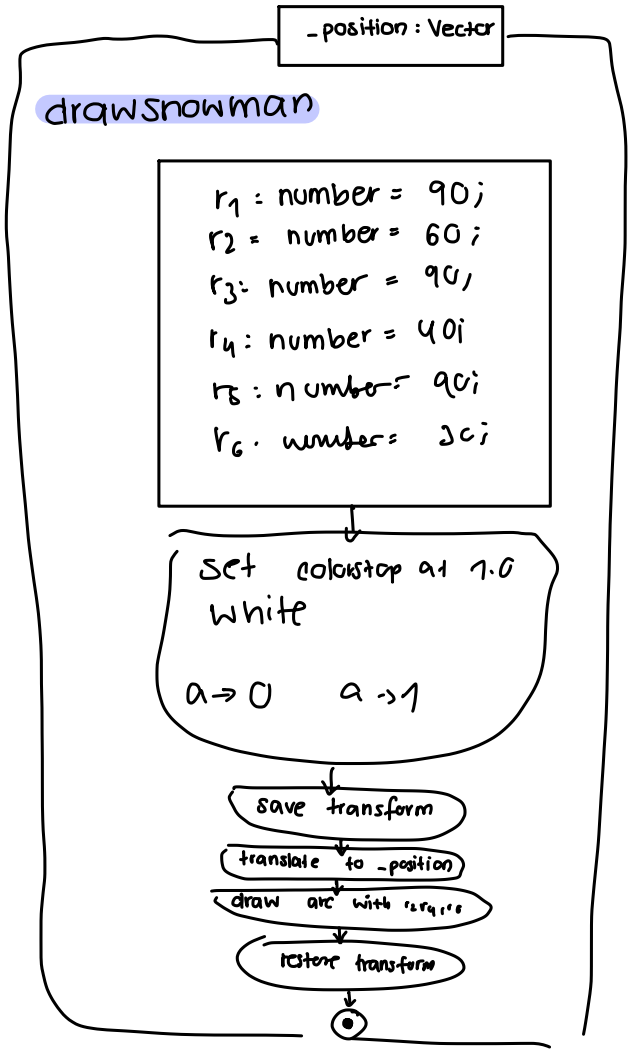

_position: Vector
drawSun
r1: number = 40
r2: number = 150
gradient: Radiogradient
set ColorStop at 1.0
white - yellow → bright yellow
save transform
translate to _position
draw arc
restore transform

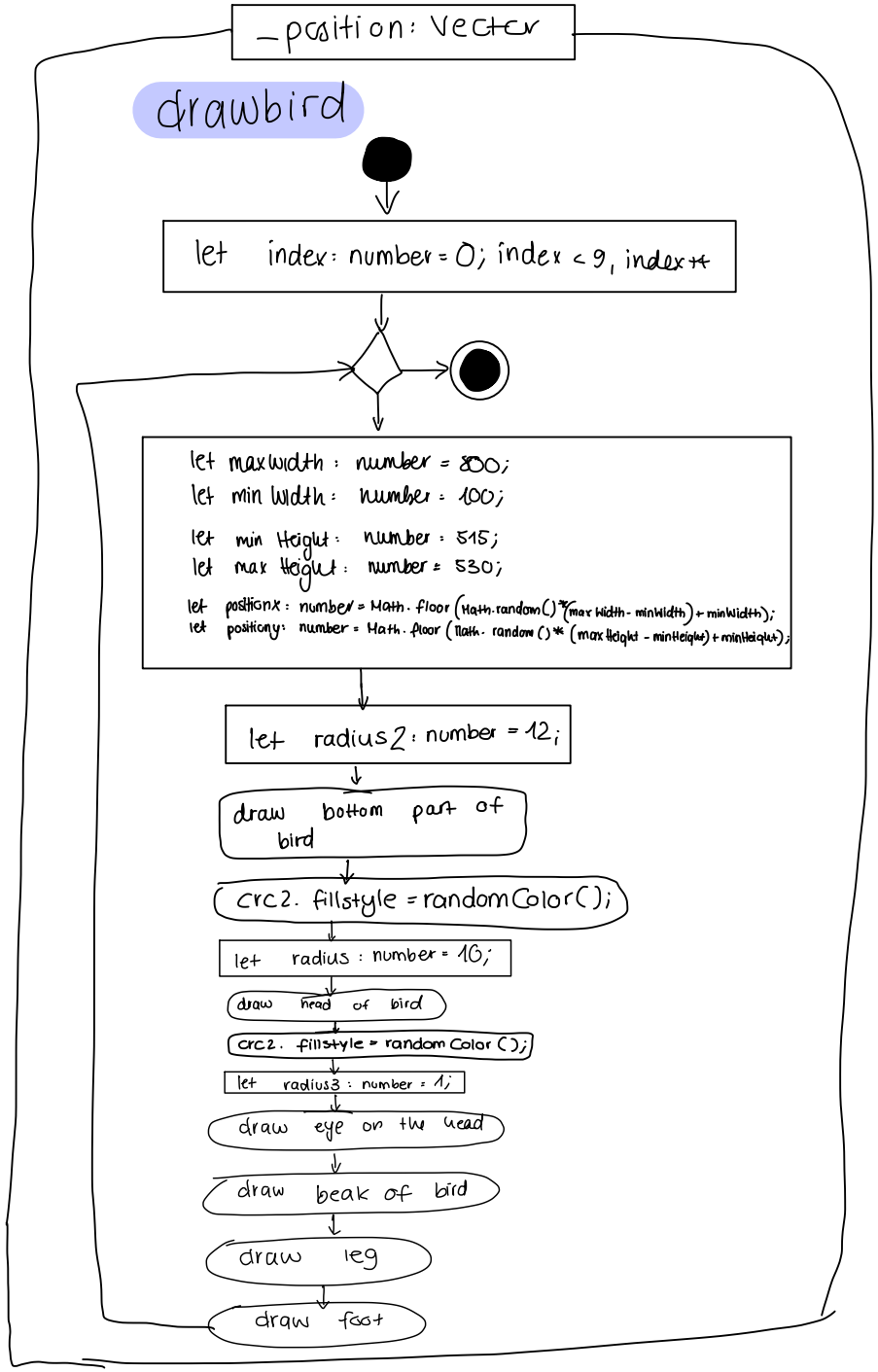

drawBackground
draw background with gradient
draw sun (position)
draw cloud (position, size)
draw mountains x2 with diff, parameter: color, vector: horizon
draw Trees (position)
draw snowman (position)
draw house (position)
drawbird (position)
drawbird (position)
background = arc2.imageData

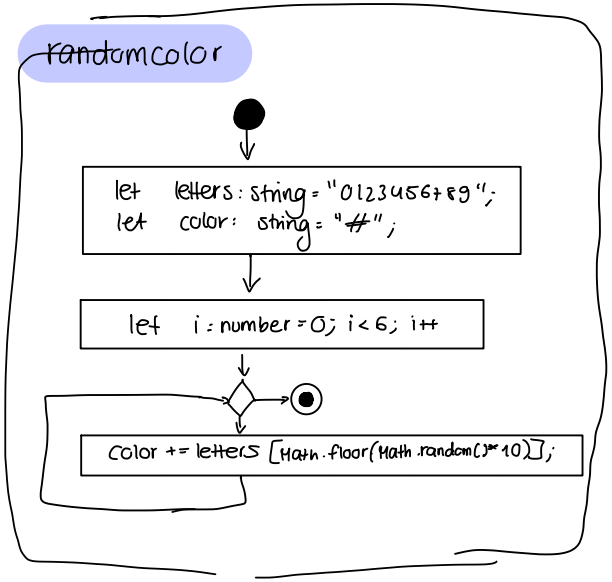

_position : Vector
size : Vector
drawCloud
n particles: number = 15
r particles: number = 50
particle: path2D
gradient: Radio Gradient
a=0,8 → a=0,2
Save Transform
translate to .position
restore transform
x: number = random - size.x
y: number = random = size.y
Save Transform
translate to x,y
draw particle
restore transform

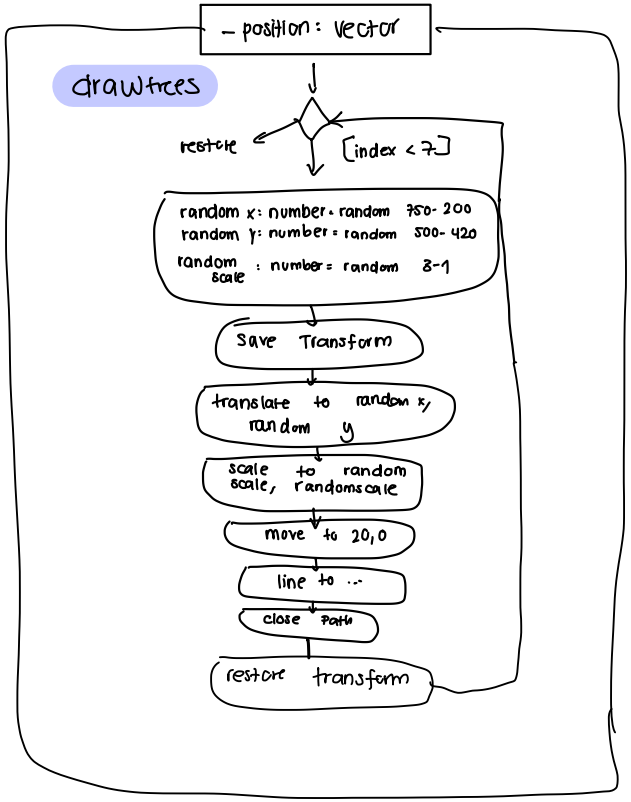

_position : Vector
_min : number
_max : number
_colorLow: string
_color High: string
drawMountains
stepMin= number = 30;
step MAX= number = 20;
x: number = 0
Save Transform
translate to _position
move to 0,0
line to 0, -max
[x= canvas width]
x+ = random betw. stepMin and stepMax
y:number = _min + random. (_max, _min)
line to x,y
line to x,0
close Path
create gradient with given color
draw Path
restore transform


_position : Vector
drawhouse
Save Transform
translate to _position
move to -80,0
line to ...
close Path
restore transform

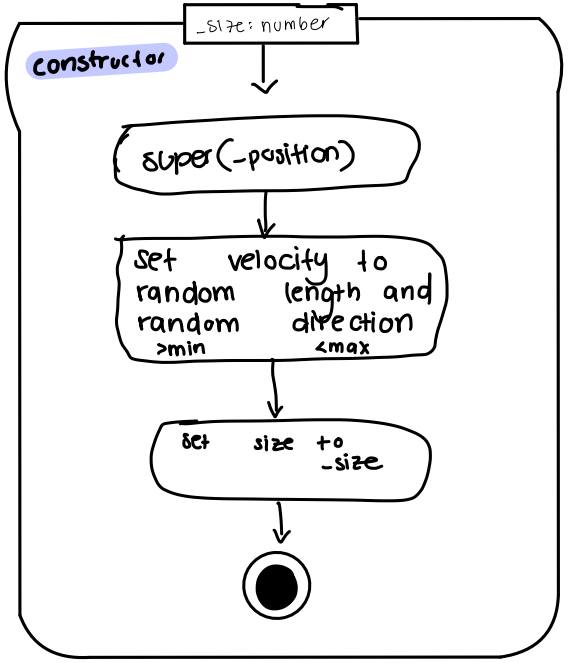

_position : Vector
drawsnowman
r1 : number = 90;
r2 = number = 60;
r3 : number = 90;
r4 : number = 40;
r5 : number = 90;
r6 : number = 30;
set colorstop a1 1.0 white
a -> 0 a -> 1
save transform
translate to _position
draw arc with radius
restore transform

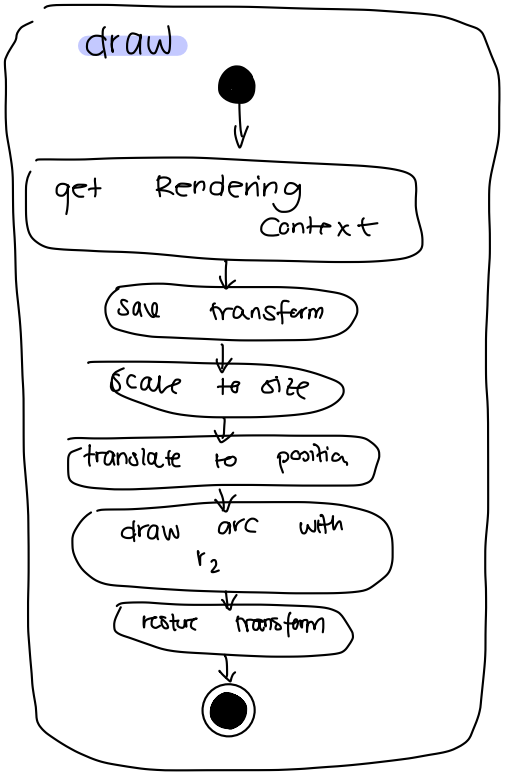

_position: Vector
drawBird
let index: number = 0; index < 9, index++
let maxWidth: number = 800;
let minWidth: number = 100;
let minHeight: number = 515;
let maxHeight: number = 530;
let positionX: number = Math.floor(Math.random()*(maxWidth - minWidth) + minWidth);
let positionY: number = Math.floor(Math.random()*(maxHeight - 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

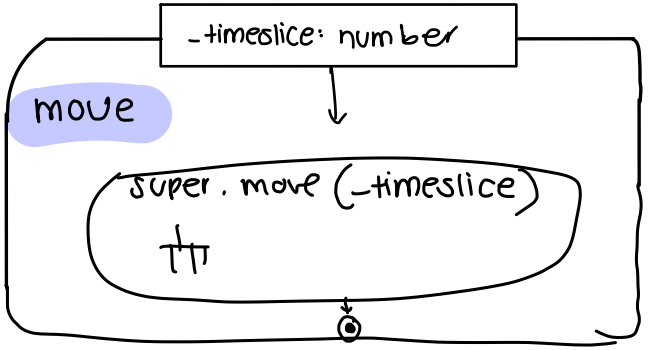

randomColor
let letters: string = "0123456789";
let color: string = "#";
let i: number = 0; i<6; i++
color += letters[Math.floor(Math.random()*10)];


_position: Vector
drawTrees
restore
[index < 7]
random x: number = random 750-200
random y: number = random 500-420
random scale: number = random 3-1
save Transform
translate to random x, random y
scale to random scale, randomscale
move to 20,0
line to ...
close Path
restore transform

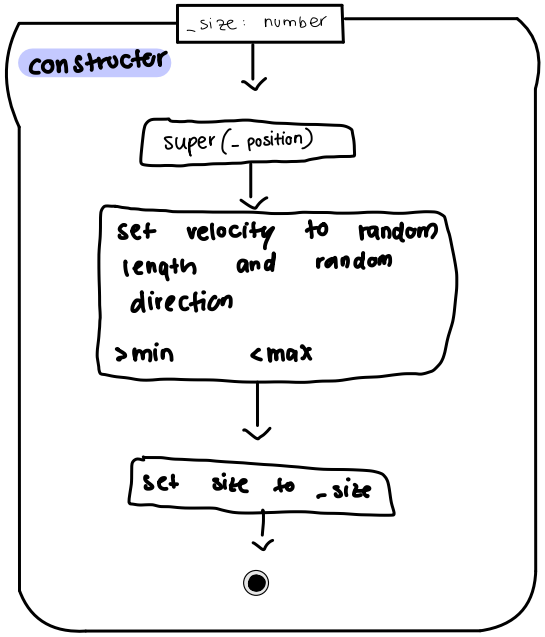
Snowflake

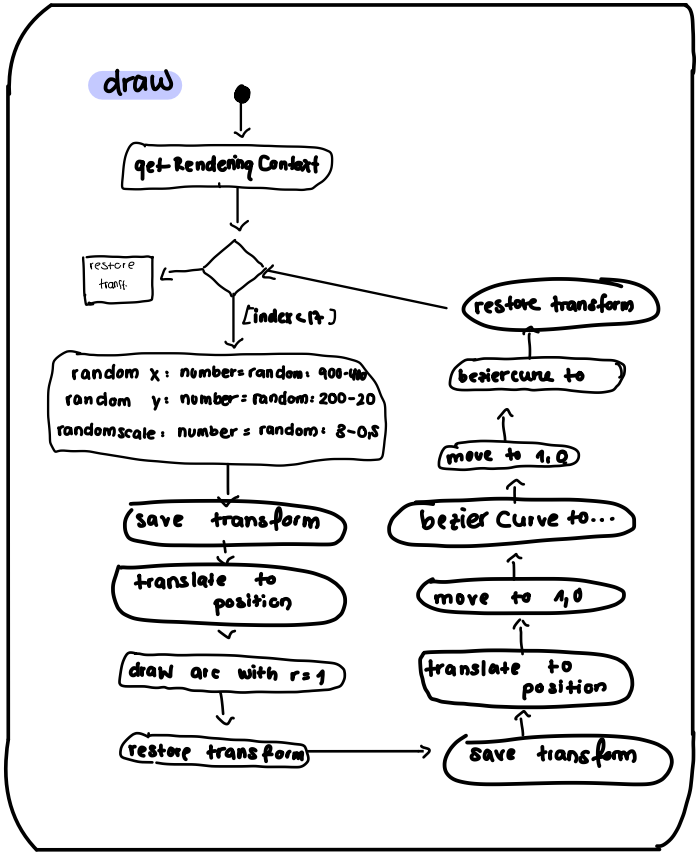

_size: number
constructor
super(_position)
Set velocity to random length and random direction
>min
<max
Set size to _size

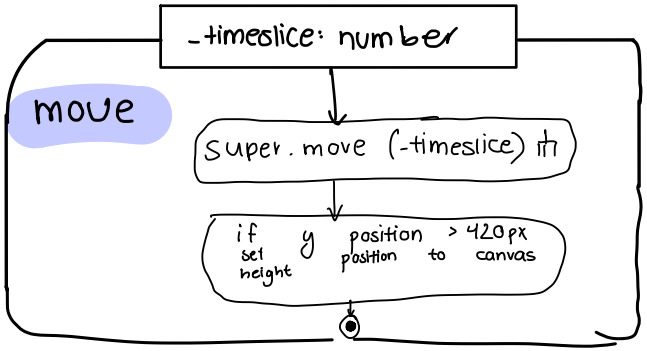

draw
get Rendering Context
save transform
Scale to size
translate to positio
draw arc with
r2
restor transform


_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


draw
get Rendering Context
restore transf.
[index<17]
restore transform
random x: number= random: 900-40
random y: number= random: 200-20
randomscale: number = random: 8-0,5
bezierCurve to
move to 1,0
save transform
bezier Curve to...
translate to position
move to 1,0
draw arc with r=1
translate to position
restore transform
save transform


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