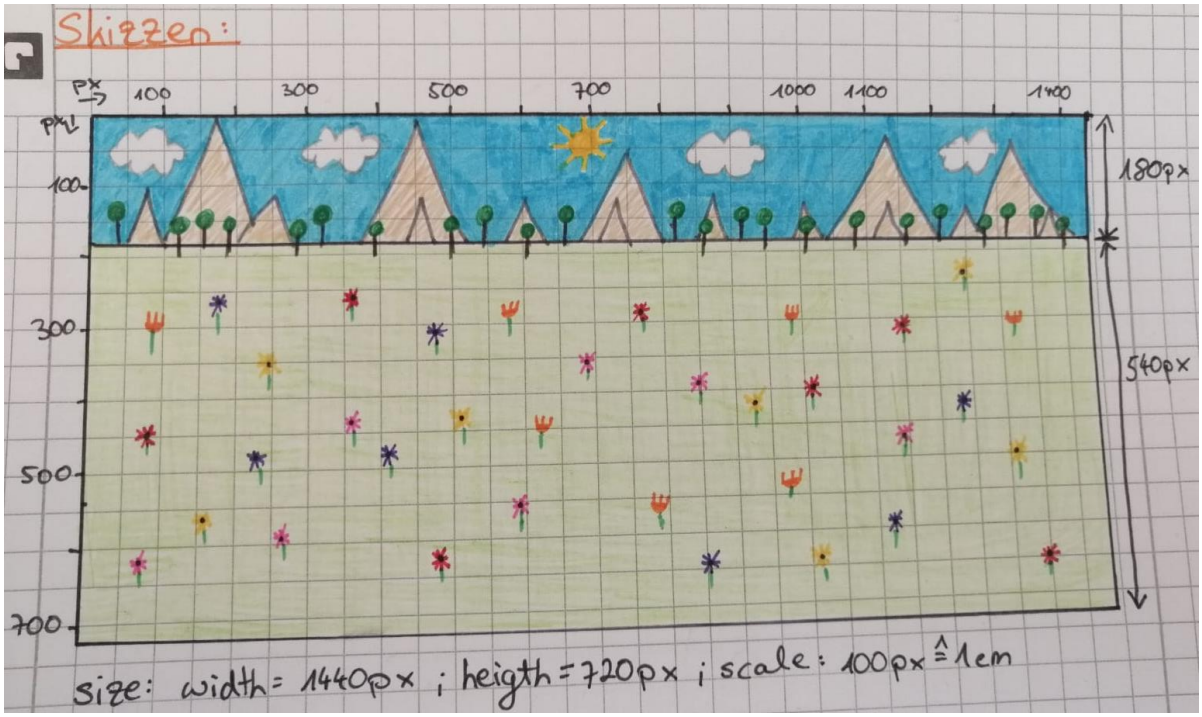
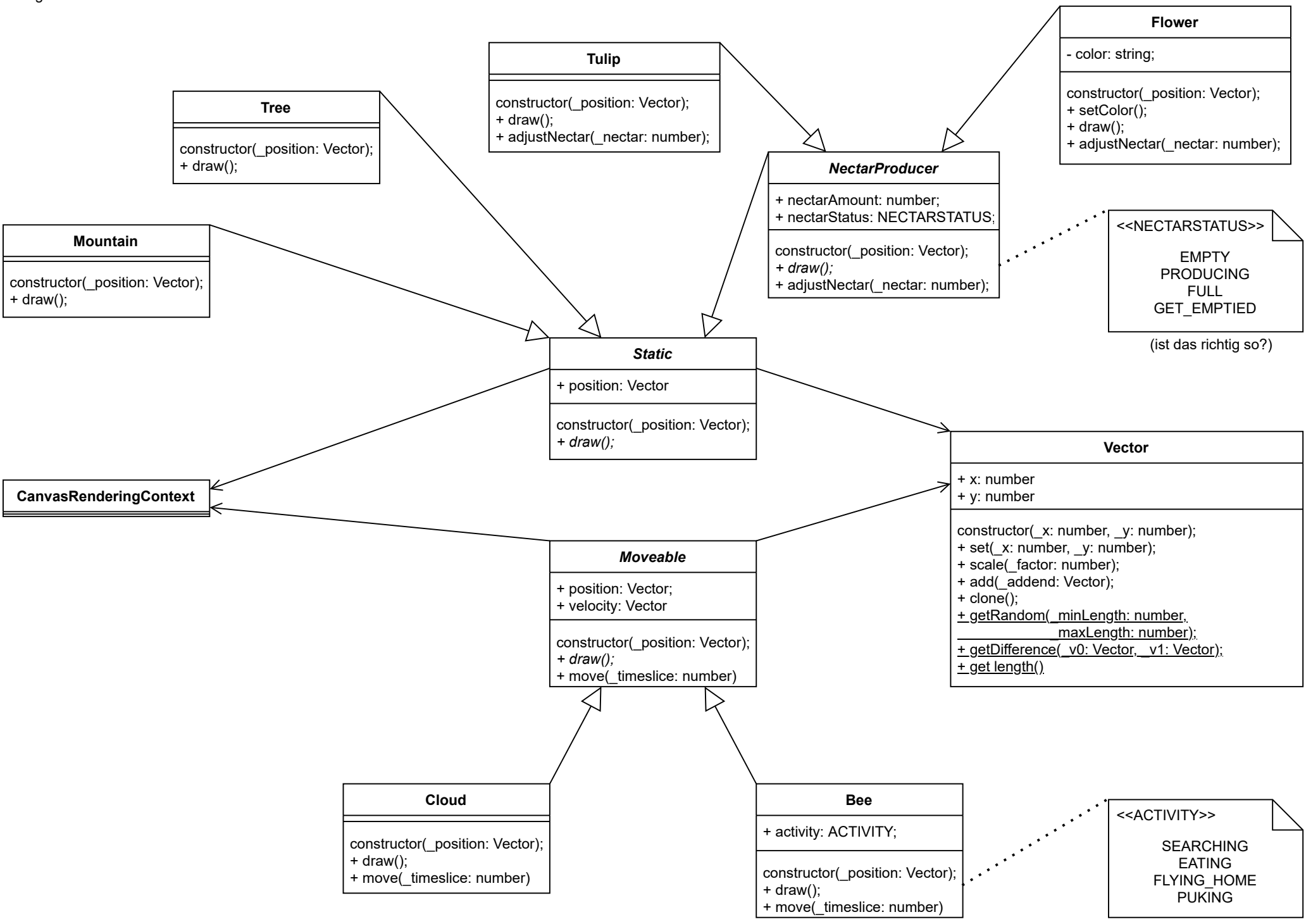
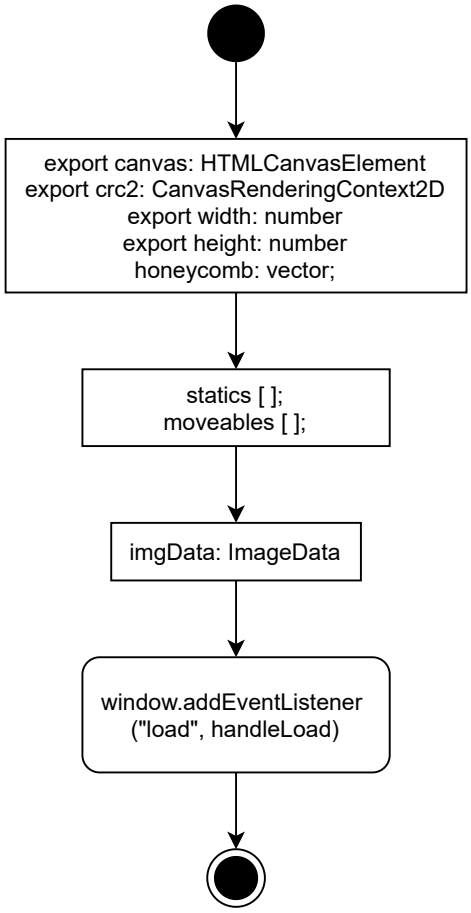
USER INTERFACE SKIZZE:

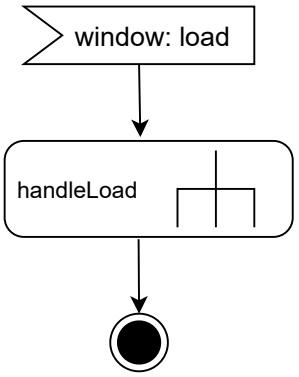


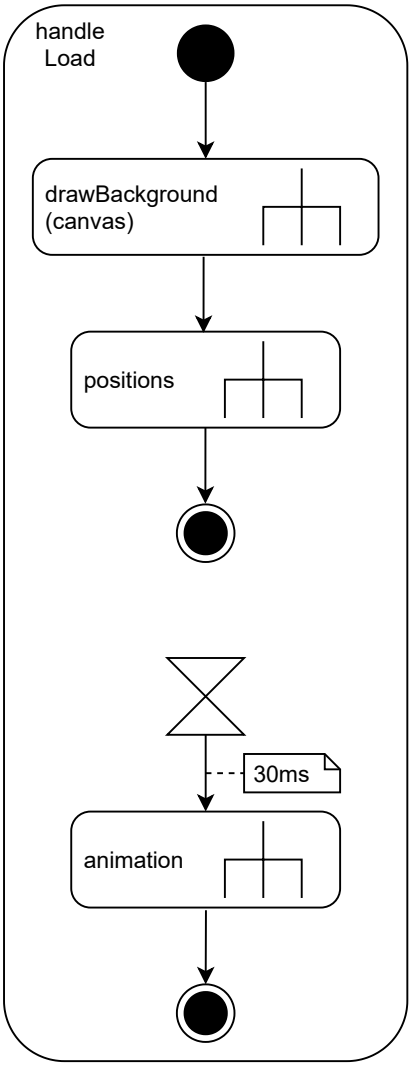
Class Diagram

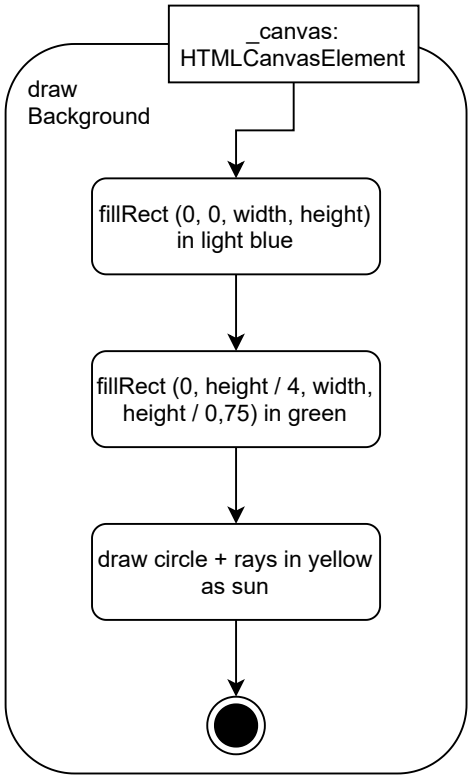


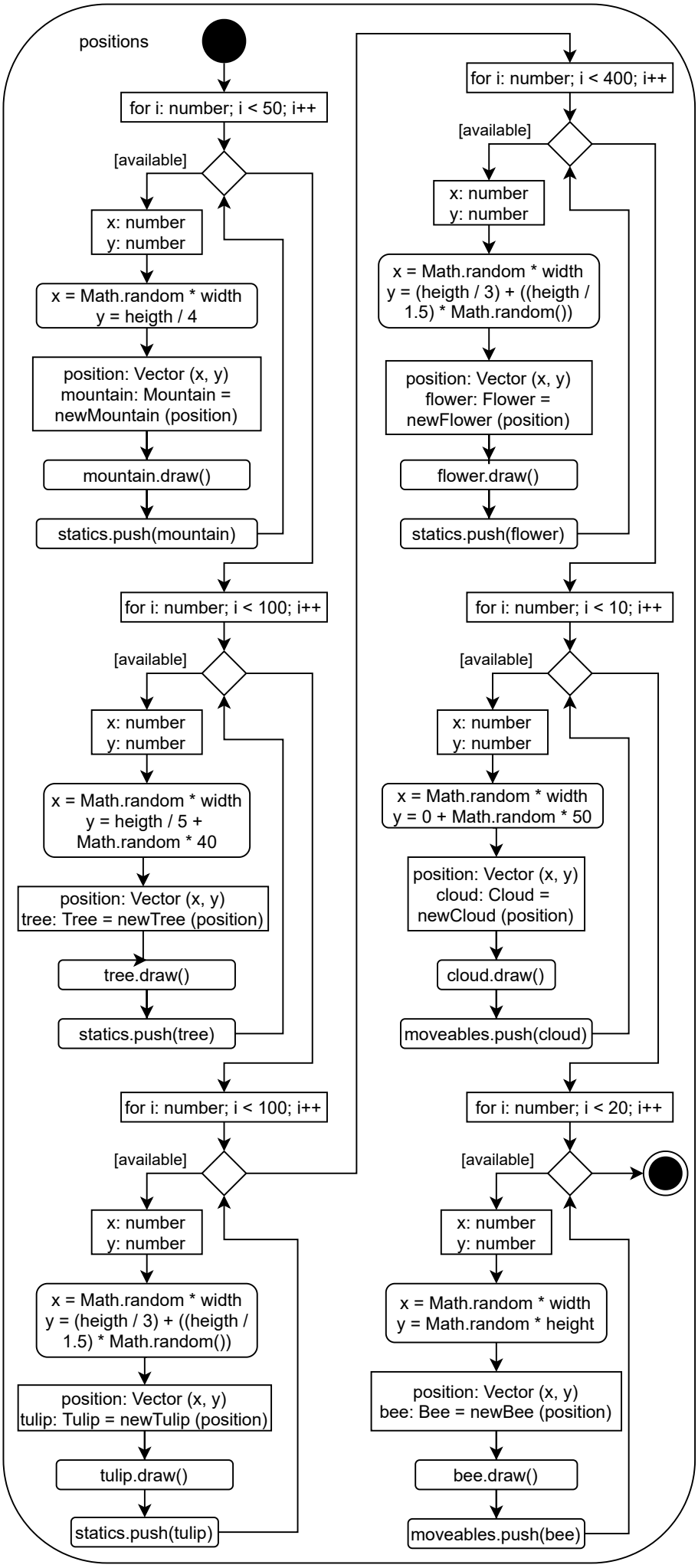
AD: main

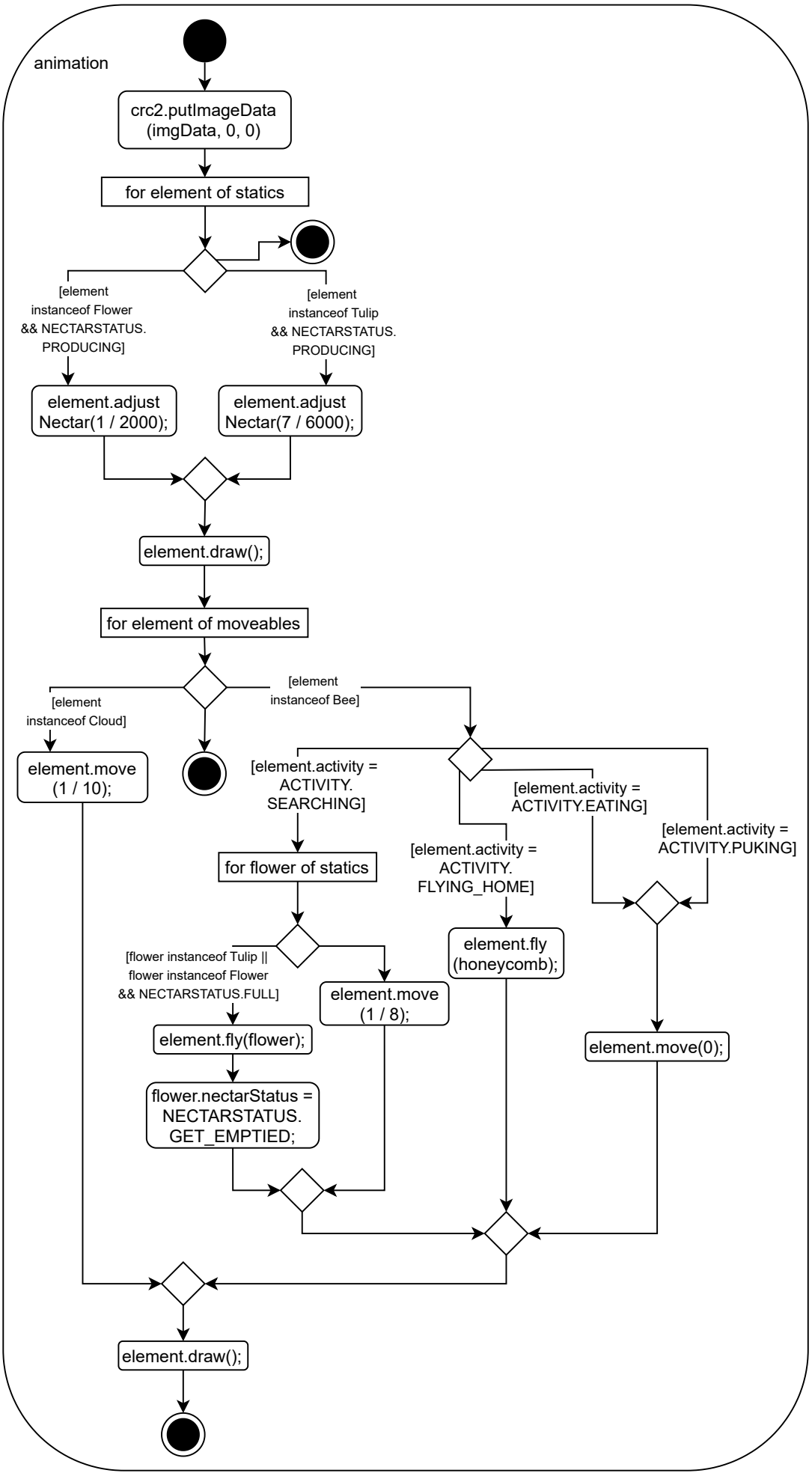

export canvas: HTMLCanvasElement
export crc2: CanvasRenderingContext2D
export width: number
export height: number
honeycomb: vector;
statics [ ];
moveables [ ];
imgData: ImageData
window.addEventListener
("load", handleLoad)

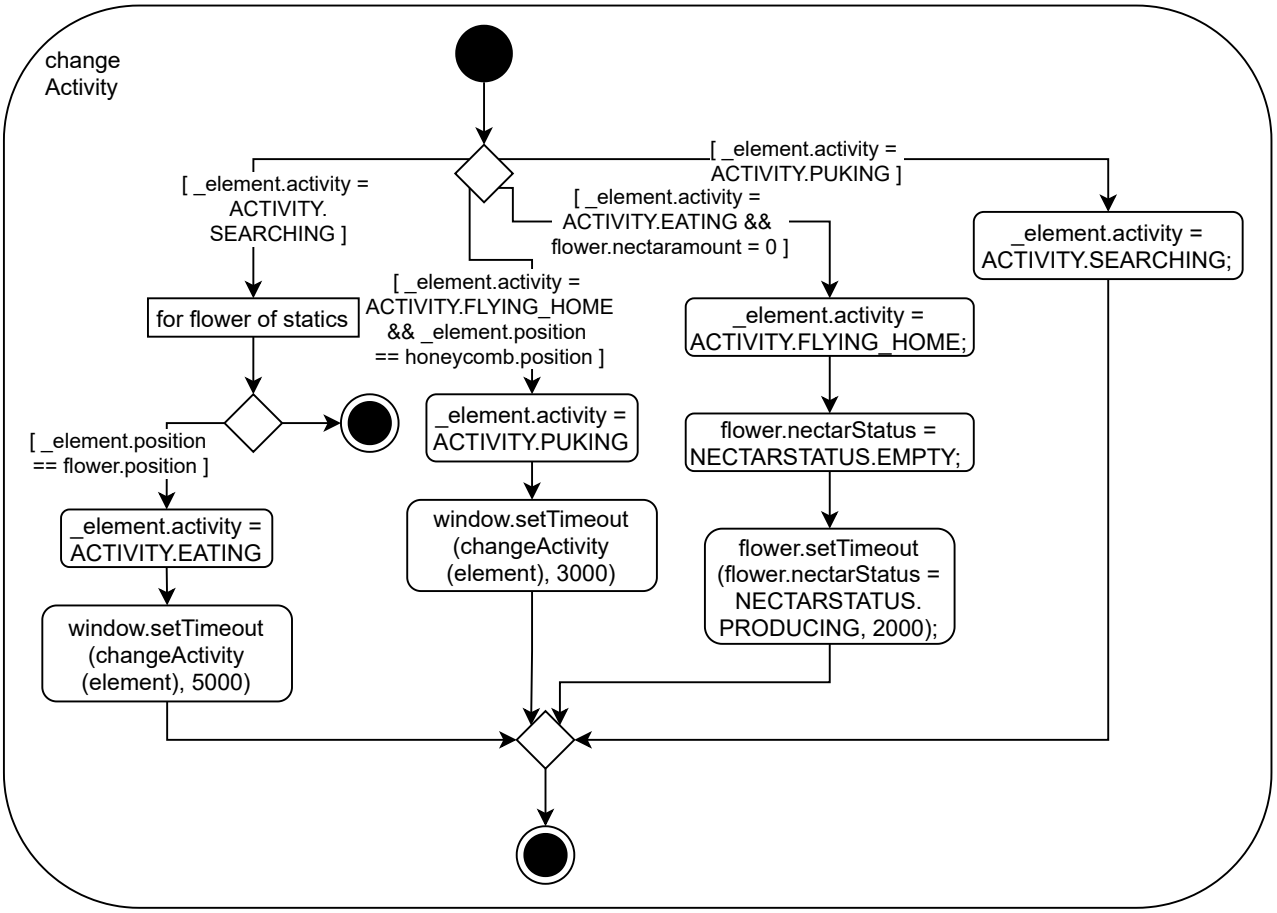

window: load
handleLoad


handle
Load
drawBackground
(canvas)
positions
30ms
animation

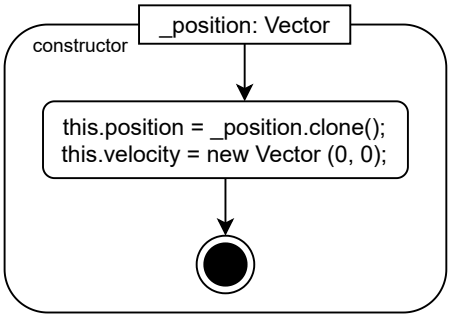

_canvas:
HTMLCanvasElement
draw
Background
fillRect (0, 0, width, height)
in light blue
fillRect (0, height / 4, width,
height / 0,75) in green
draw circle + rays in yellow
as sun

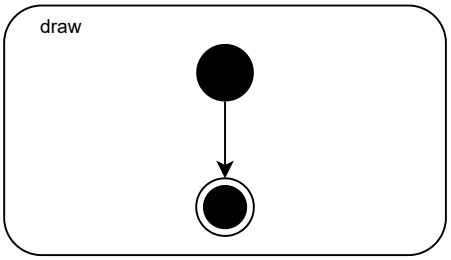

positions
for i: number; i < 50; i++
[available]
x: number
y: number
x = Math.random * width
y = heigth / 4
position: Vector (x, y)
mountain: Mountain =
newMountain (position)
mountain.draw()
statics.push(mountain)
for i: number; i < 100; i++
[available]
x: number
y: number
x = Math.random * width
y = heigth / 5 +
Math.random * 40
position: Vector (x, y)
tree: Tree = newTree (position)
tree.draw()
statics.push(tree)
for i: number; i < 100; i++
[available]
x: number
y: number
x = Math.random * width
y = (heigth / 3) + ((heigth /
1.5) * Math.random())
position: Vector (x, y)
tulip: Tulip = newTulip (position)
tulip.draw()
statics.push(tulip)
for i: number; i < 400; i++
[available]
x: number
y: number
x = Math.random * width
y = (heigth / 3) + ((heigth /
1.5) * Math.random())
position: Vector (x, y)
flower: Flower =
newFlower (position)
flower.draw()
statics.push(flower)
for i: number; i < 10; i++
[available]
x: number
y: number
x = Math.random * width
y = 0 + Math.random * 50
position: Vector (x, y)
cloud: Cloud =
newCloud (position)
cloud.draw()
moveables.push(cloud)
for i: number; i < 20; i++
[available]
x: number
y: number
x = Math.random * width
y = Math.random * height
position: Vector (x, y)
bee: Bee = newBee (position)
bee.draw()
moveables.push(bee)

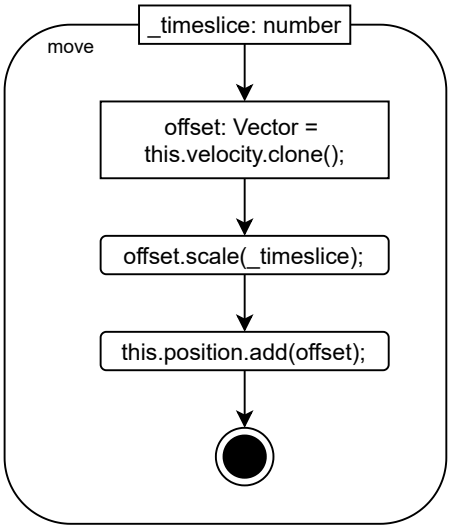

animation
crc2.putImageData (imgData, 0, 0)
for element of statics
[element instanceof Flower && NECTARSTATUS. PRODUCING]
[element instanceof Tulip && NECTARSTATUS. PRODUCING]
element.adjust Nectar(1 / 2000);
element.adjust Nectar(7 / 6000);
element.draw();
for element of moveables
[element instanceof Cloud]
[element instanceof Bee]
element.move (1 / 10);
[element.activity = ACTIVITY. SEARCHING]
[element.activity = ACTIVITY.EATING]
[element.activity = ACTIVITY.PUKING]
[element.activity = ACTIVITY. FLYING_HOME]
for flower of statics
element.fly (honeycomb);
[flower instanceof Tulip || flower instanceof Flower && NECTARSTATUS.FULL]
element.move (1 / 8);
element.fly(flower);
element.move(0);
flower.nectarStatus = NECTARSTATUS. GET_EMPTIED;
element.draw();


change
Activity
[_element.activity =
ACTIVITY.
SEARCHING]
[_element.activity =
ACTIVITY.EATING &&
flower.nectaramount = 0]
[_element.activity =
ACTIVITY.PUKING]
_element.activity =
ACTIVITY.SEARCHING;
for flower of statics
[_element.activity =
ACTIVITY.FLYING_HOME
&& _element.position
== honeycomb.position]
_element.activity =
ACTIVITY.FLYING_HOME;
[_element.position
== flower.position]
_element.activity =
ACTIVITY.PUKING
flower.nectarStatus =
NECTARSTATUS.EMPTY;
_element.activity =
ACTIVITY.EATING
window.setTimeout
(changeActivity
(element), 3000)
flower.setTimeout
(flower.nectarStatus =
NECTARSTATUS.
PRODUCING, 2000);
window.setTimeout
(changeActivity
(element), 5000)

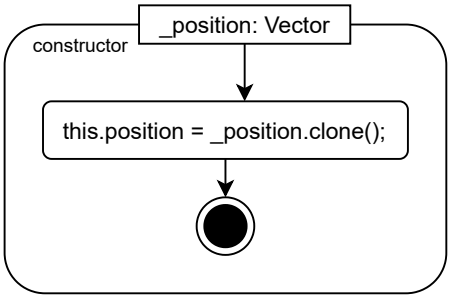
Moveable

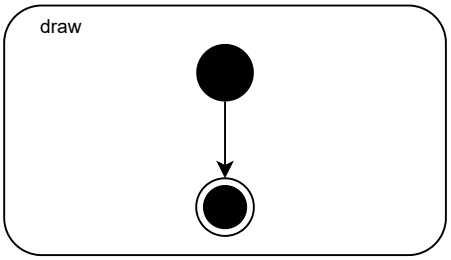

_position: Vector
constructor
this.position = _position.clone();
this.velocity = new Vector (0, 0);


draw

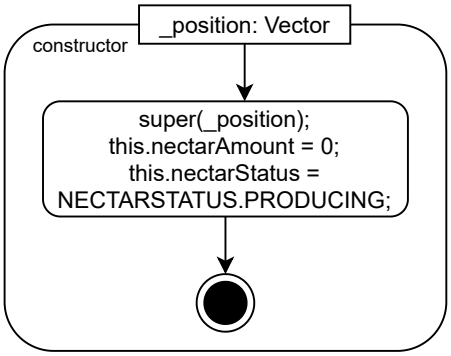

_timeslice: number
move
offset: Vector =
this.velocity.clone();
offset.scale(_timeslice);
this.position.add(offset);

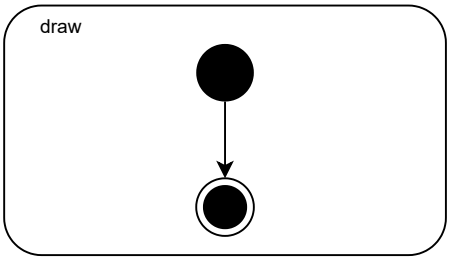
Static

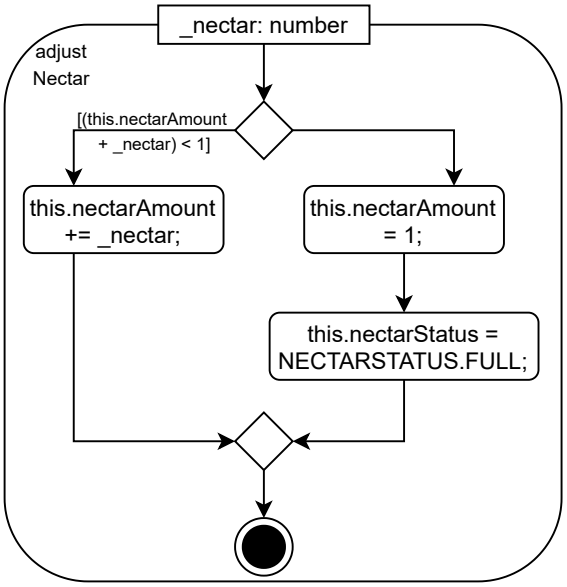

_position: Vector
constructor
this.position = _position.clone();


draw

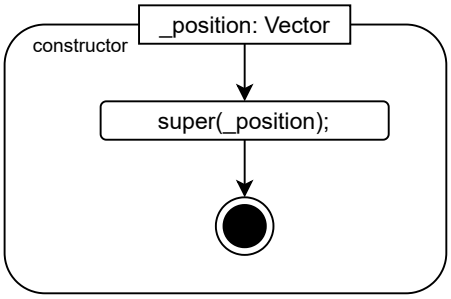
NectarProducer

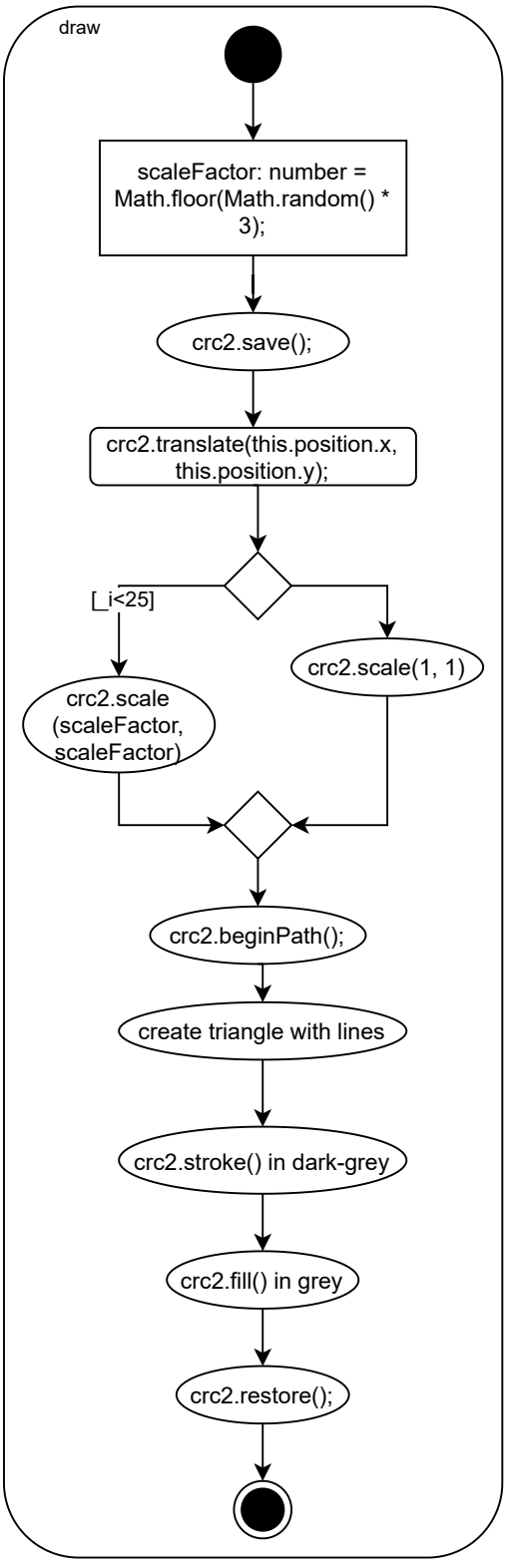




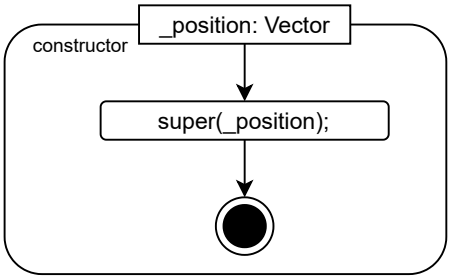


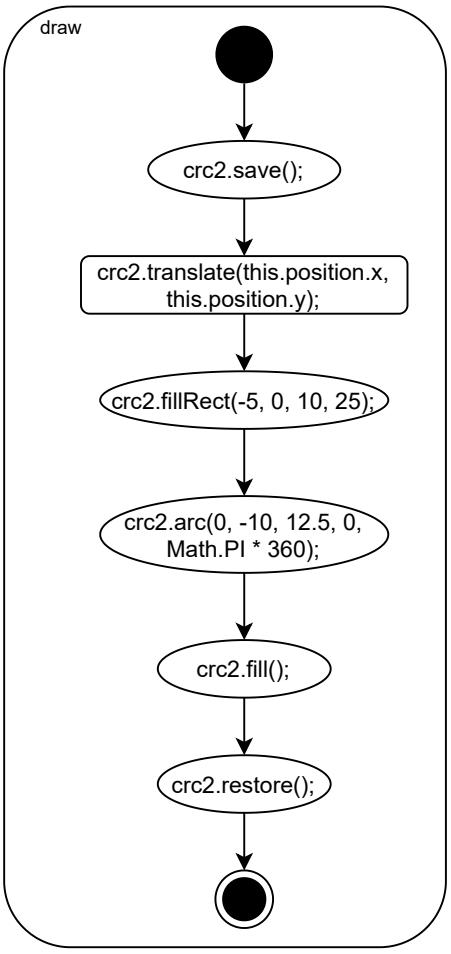
Mountain


constructor
_position: Vector
super(_position);

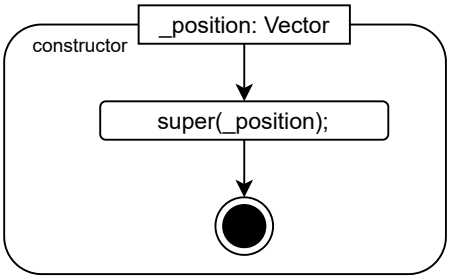

draw
scaleFactor: number = Math.floor(Math.random() * 3);
crc2.save();
crc2.translate(this.position.x, this.position.y);
[_i<25]
crc2.scale (scaleFactor, scaleFactor)
crc2.scale(1, 1)
crc2.beginPath();
create triangle with lines
crc2.stroke() in dark-grey
crc2.fill() in grey
crc2.restore();

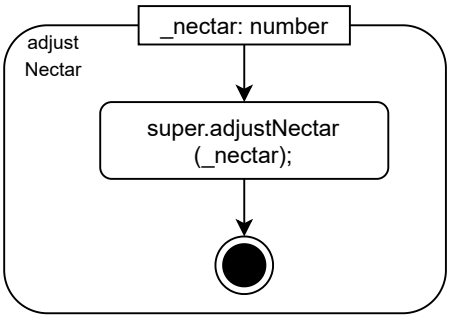
Tree

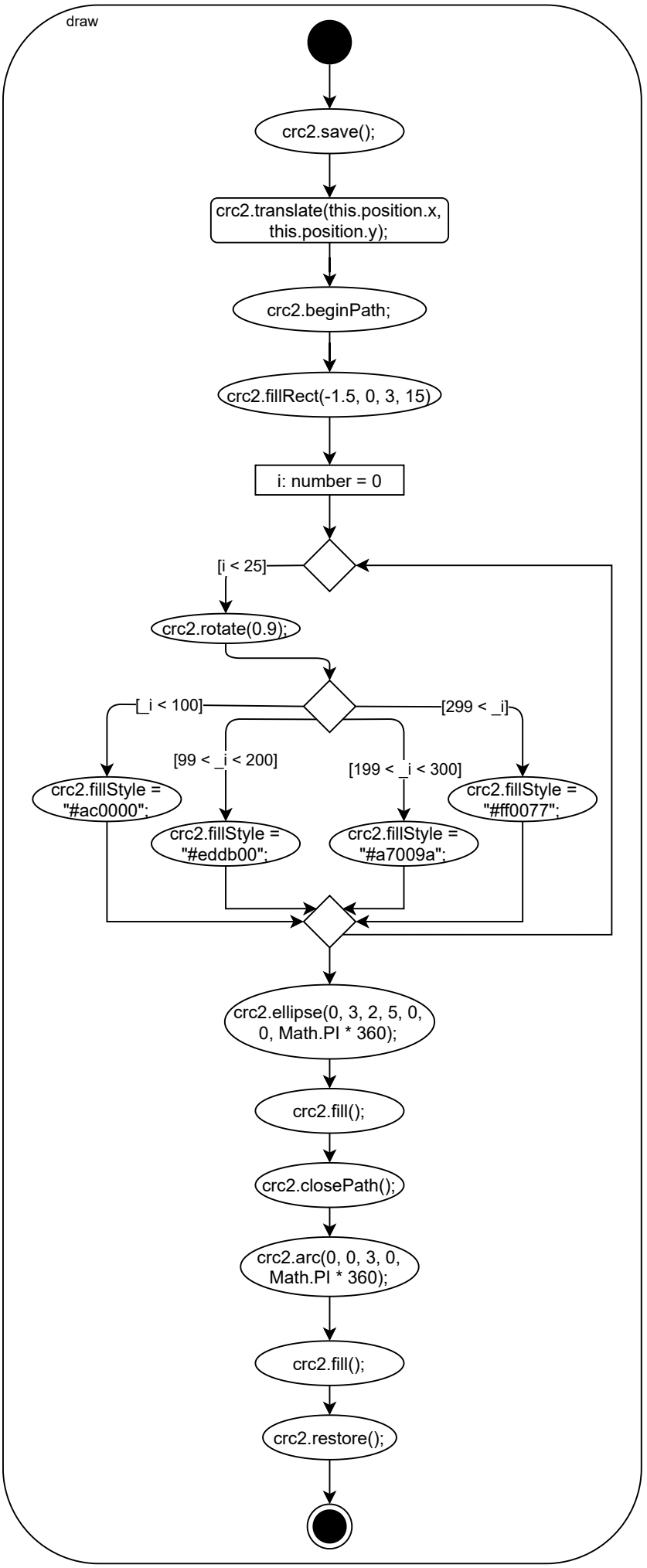

constructor
_position: Vector
super(_position);


draw
crc2.save();
crc2.translate(this.position.x, this.position.y);
crc2.fillRect(-5, 0, 10, 25);
crc2.arc(0, -10, 12.5, 0, Math.PI * 360);
crc2.fill();
crc2.restore();

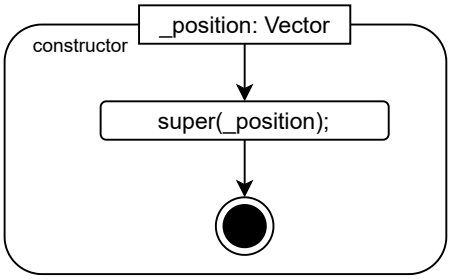
Flower

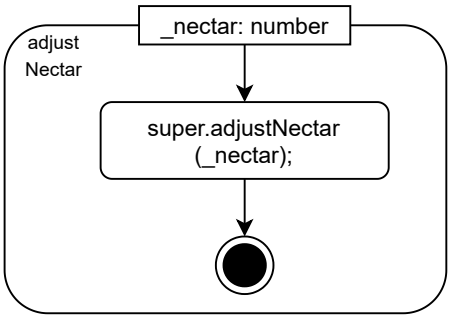

constructor
_position: Vector
super(_position);

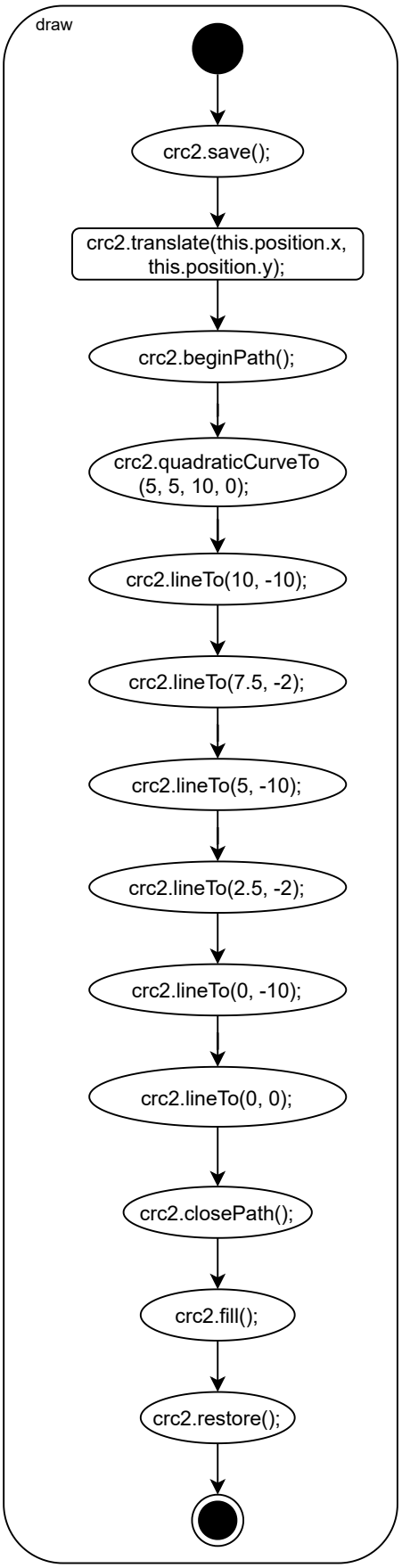

adjust
Nectar
_nectar: number
super.adjustNectar
(_nectar);


draw
crc2.save();
crc2.translate(this.position.x,
this.position.y);
crc2.beginPath;
crc2.fillRect(-1.5, 0, 3, 15)
i: number = 0
[i < 25]
crc2.rotate(0.9);
[_i < 100]
[99 < _i < 200]
[199 < _i < 300]
[299 < _i]
crc2.fillStyle =
"#ac0000";
crc2.fillStyle =
"#eddb00";
crc2.fillStyle =
"#a7009a";
crc2.fillStyle =
"#ff0077";
crc2.ellipse(0, 3, 2, 5, 0,
0, Math.PI * 360);
crc2.fill();
crc2.closePath();
crc2.arc(0, 0, 3, 0,
Math.PI * 360);
crc2.fill();
crc2.restore();

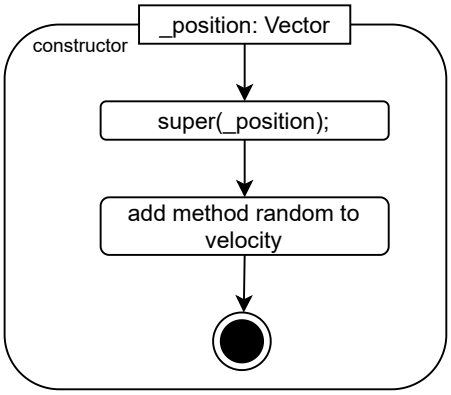
Tulip

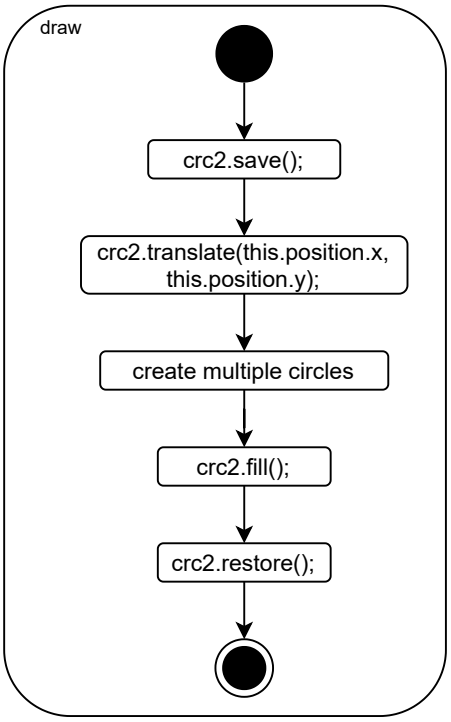

constructor
_position: Vector
super(_position);

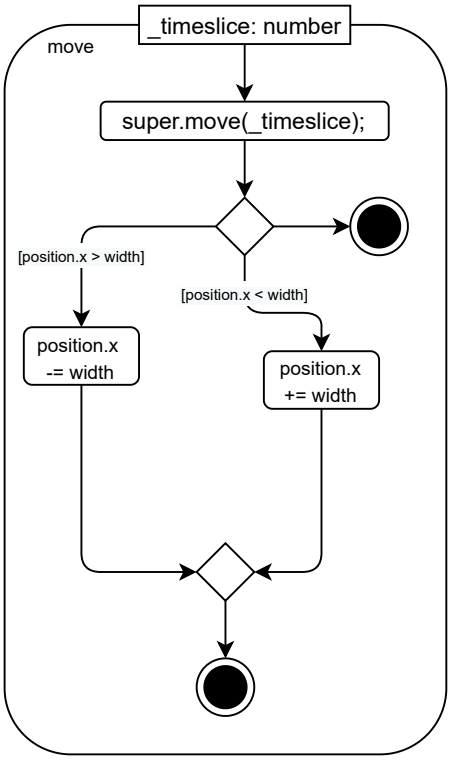

adjust
Nectar
_nectar: number
super.adjustNectar
(_nectar);


draw
crc2.save();
crc2.translate(this.position.x,
this.position.y);
crc2.beginPath();
crc2.quadraticCurveTo
(5, 5, 10, 0);
crc2.lineTo(10, -10);
crc2.lineTo(7.5, -2);
crc2.lineTo(5, -10);
crc2.lineTo(2.5, -2);
crc2.lineTo(0, -10);
crc2.lineTo(0, 0);
crc2.closePath();
crc2.fill();
crc2.restore();

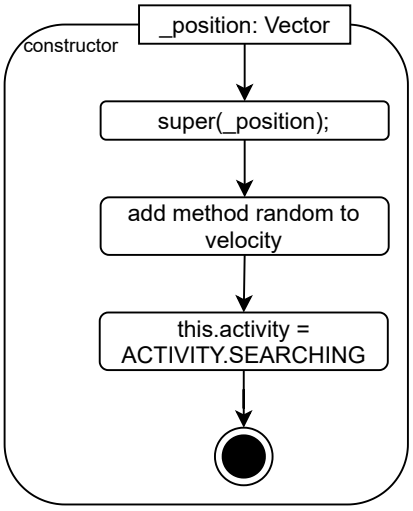
Cloud

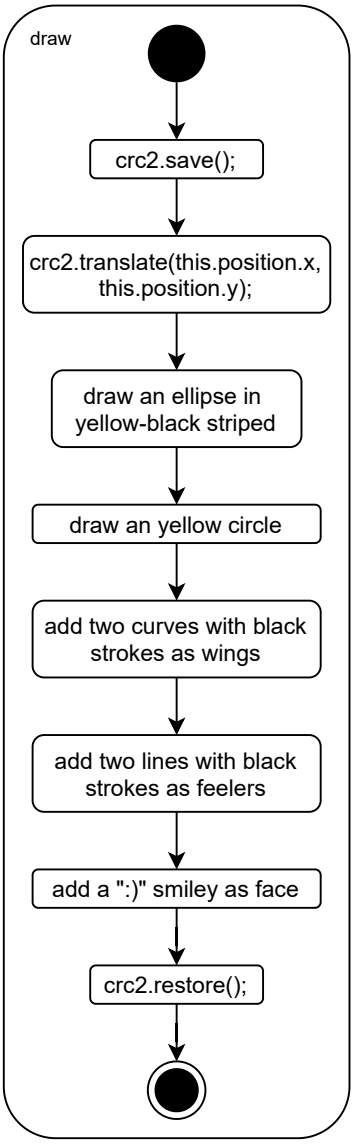

constructor
_position: Vector
super(_position);
add method random to velocity

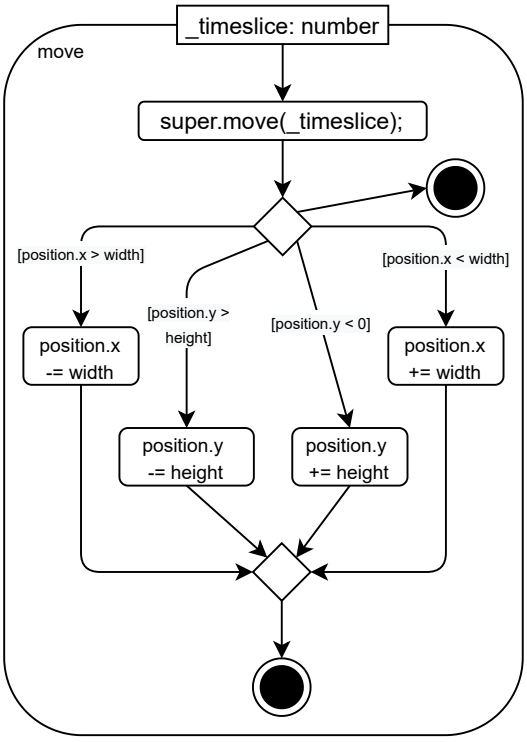

draw
crc2.save();
crc2.translate(this.position.x, this.position.y);
create multiple circles
crc2.fill();
crc2.restore();

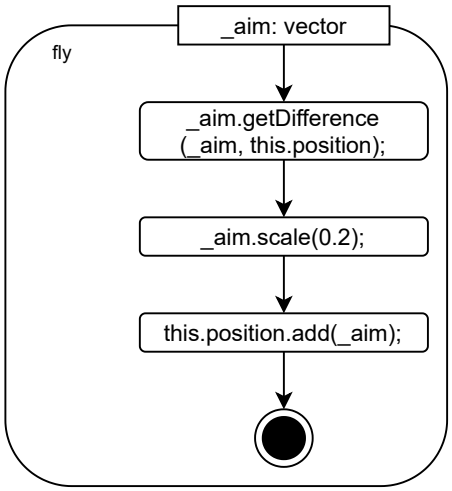

move
_timeslice: number
super.move(_timeslice);
[position.x > width]
[position.x < width]
position.x -= width
position.x += width

Bee

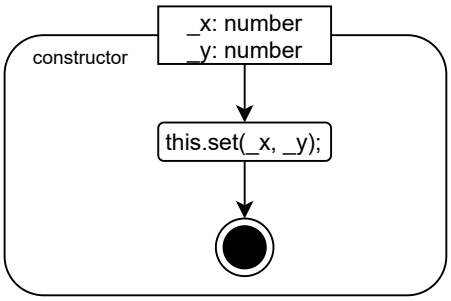

constructor
_position: Vector
super(_position);
add method random to velocity
this.activity = ACTIVITY.SEARCHING

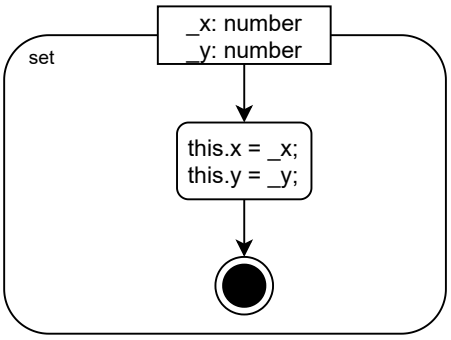

draw
crc2.save();
crc2.translate(this.position.x, this.position.y);
draw an ellipse in yellow-black striped
draw an yellow circle
add two curves with black strokes as wings
add two lines with black strokes as feelers
add a ":)" smiley as face
crc2.restore();

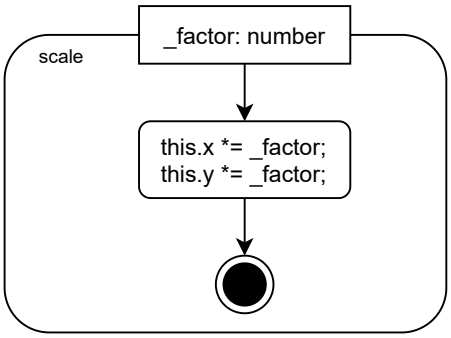

move
_timeslice: number
super.move(_timeslice);
[position.x > width]
[position.y > height]
[position.y < 0]
[position.x < width]
position.x -= width
position.y -= height
position.y += height
position.x += width

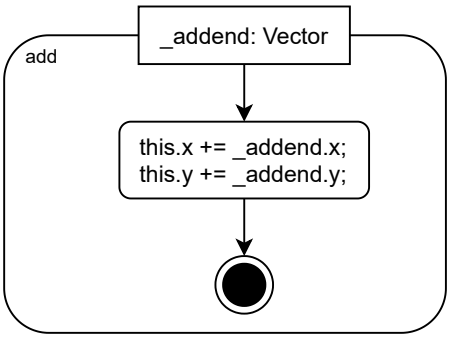

fly
_aim: vector
_aim.getDifference (_aim, this.position);
_aim.scale(0.2);
this.position.add(_aim);

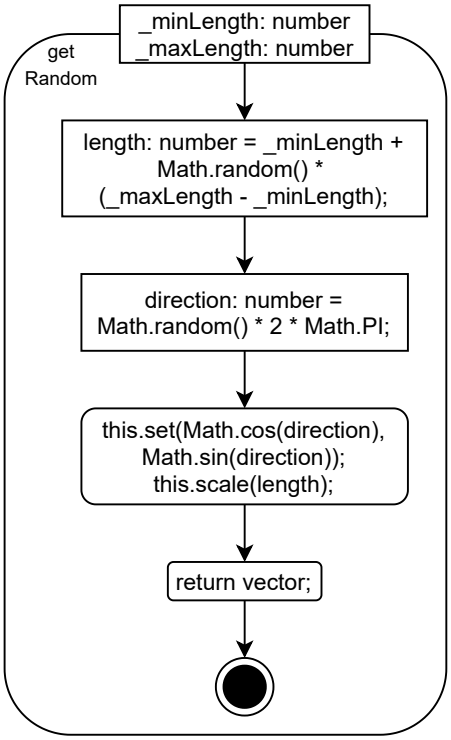
Vector

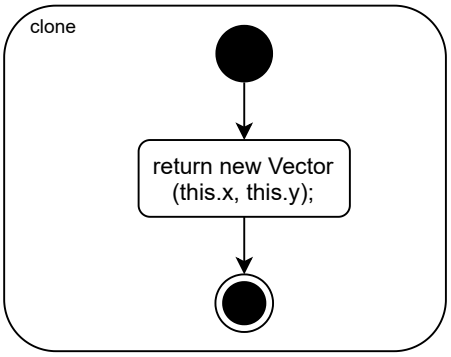

constructor
_x: number
_y: number
this.set(_x, _y);

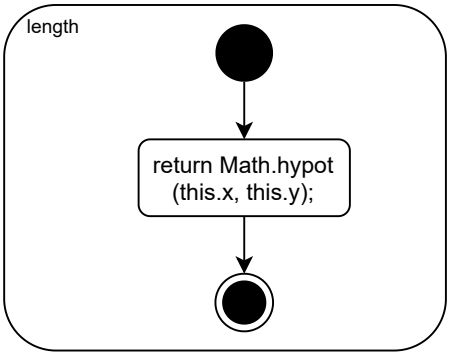

set
_x: number
_y: number
this.x = _x;
this.y = _y;

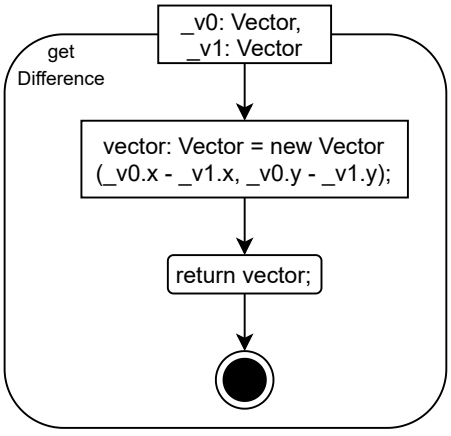

scale
_factor: number
this.x *= _factor;
this.y *= _factor;


add
_addend: Vector
this.x += _addend.x;
this.y += _addend.y;


get
Random
_minLength: number
_maxLength: number
length: number = _minLength +
Math.random() *
(_maxLength - _minLength);
direction: number =
Math.random() * 2 * Math.PI;
this.set(Math.cos(direction),
Math.sin(direction));
this.scale(length);
return vector;


clone
return new Vector
(this.x, this.y);


length
return Math.hypot
(this.x, this.y);


get
Difference
_v0: Vector,
_v1: Vector
vector: Vector = new Vector
(_v0.x - _v1.x, _v0.y - _v1.y);
return vector;