

Klassendiagramm

überarbeitet

Bee

size X: number
size Y: number

Constructor (-position: Vector)

Cloud

x: number
y: number

Constructor (-position: Vector)

Vector

X: number

Y: number

Constructor (-x: number,
-y: number)

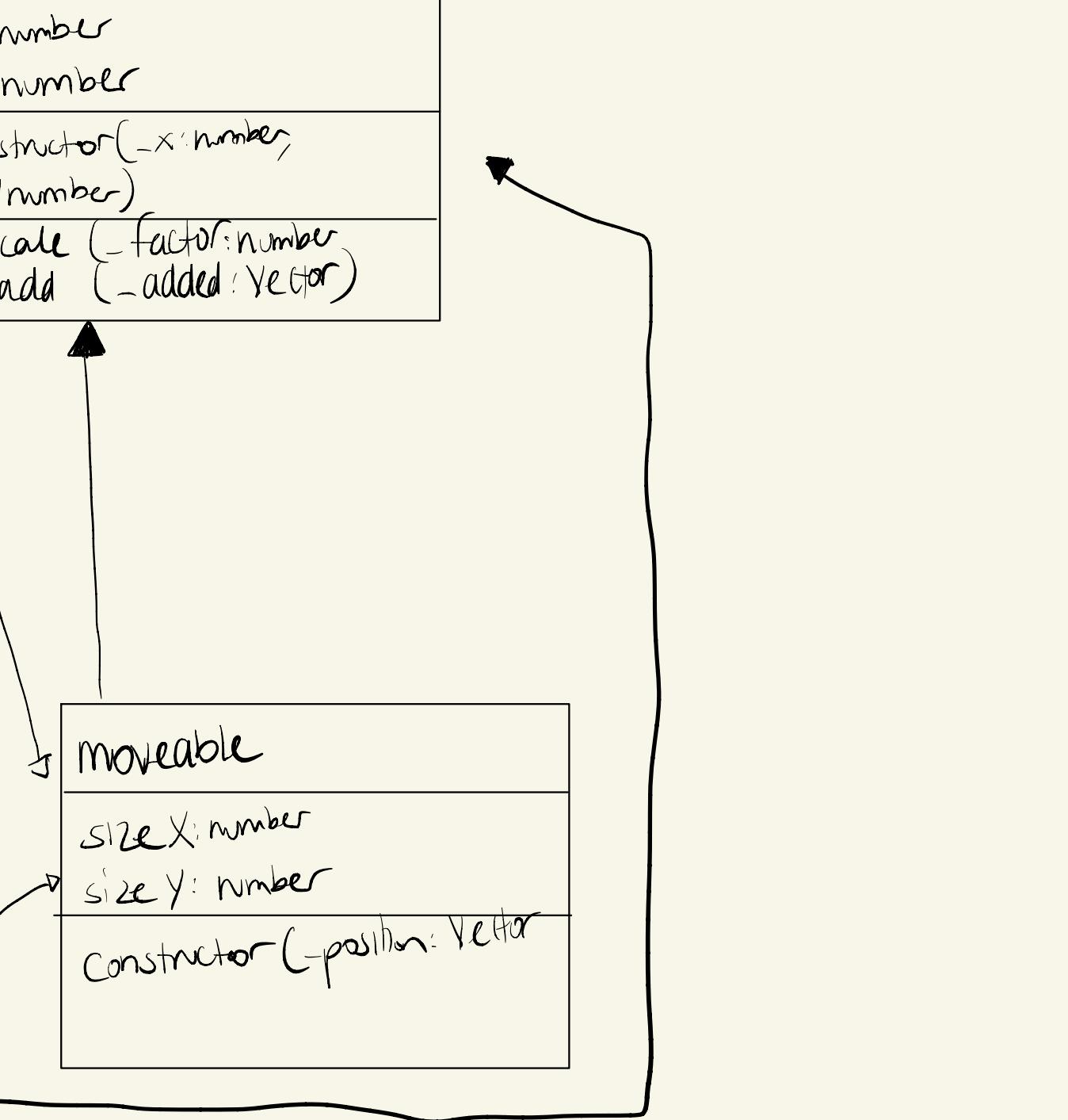
+ scale (-factor: number)
+ add (-added: Vector)

moveable

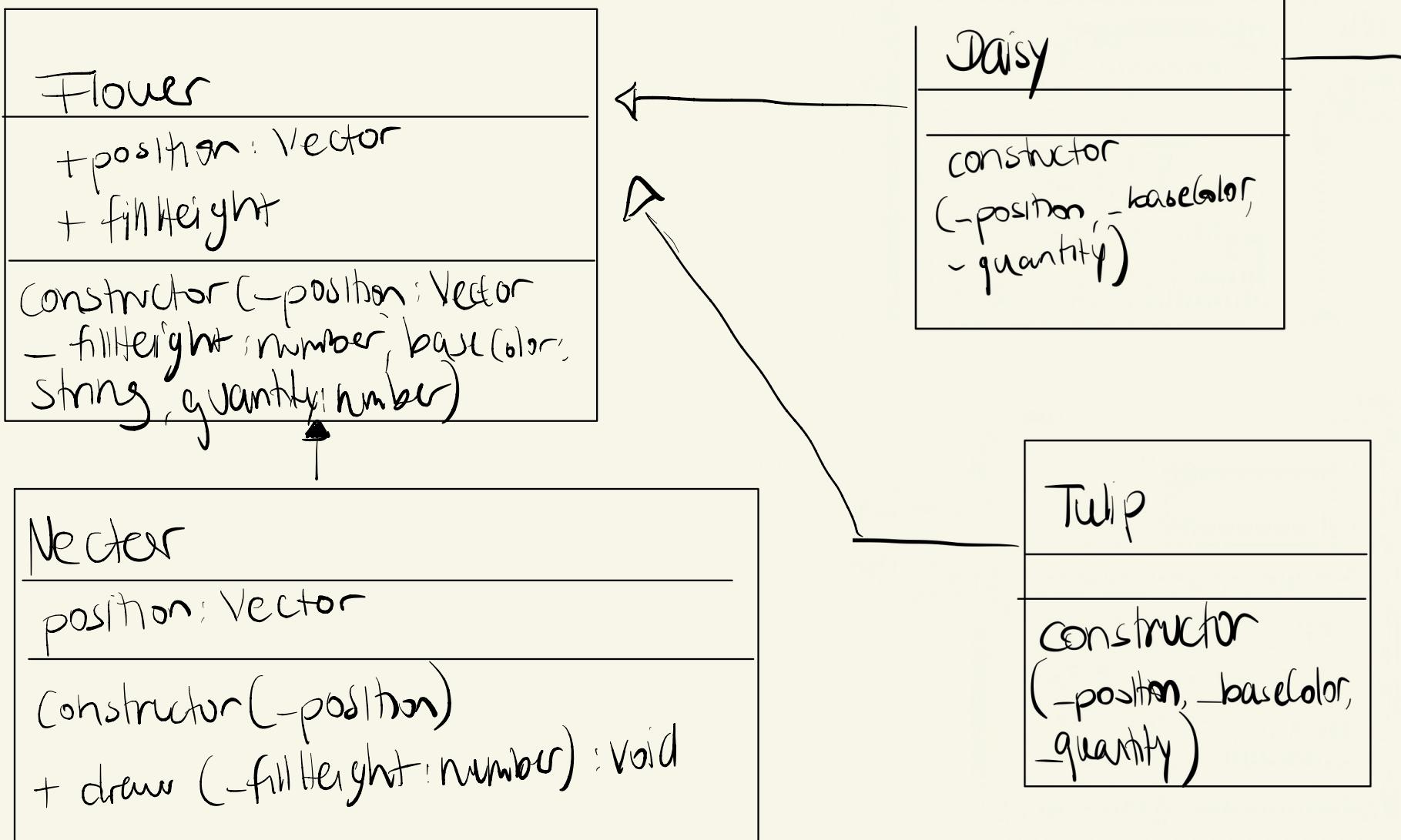
size X: number

size Y: number

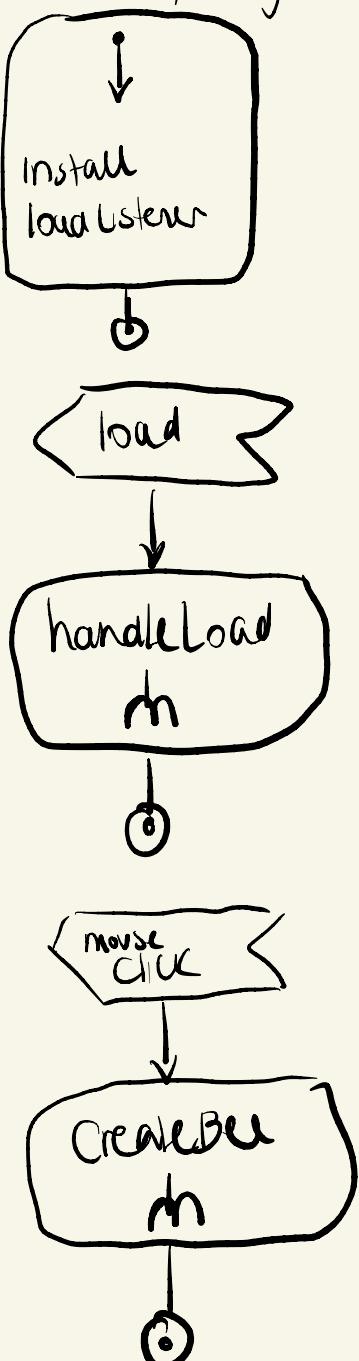
Constructor (-position: Vector)



Klassendiagramm II - Überarbeitet

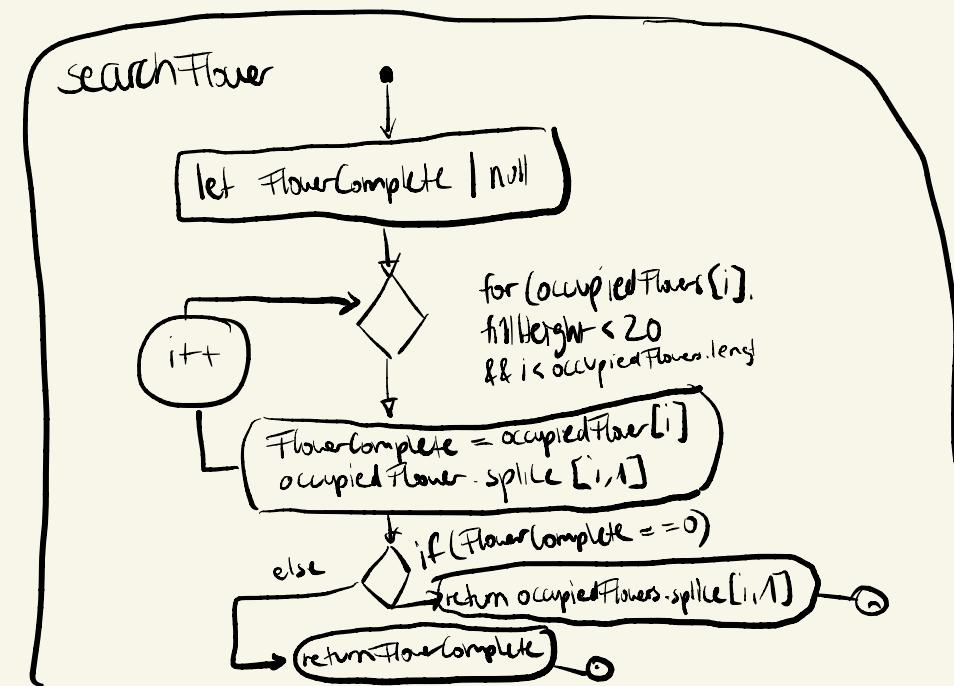
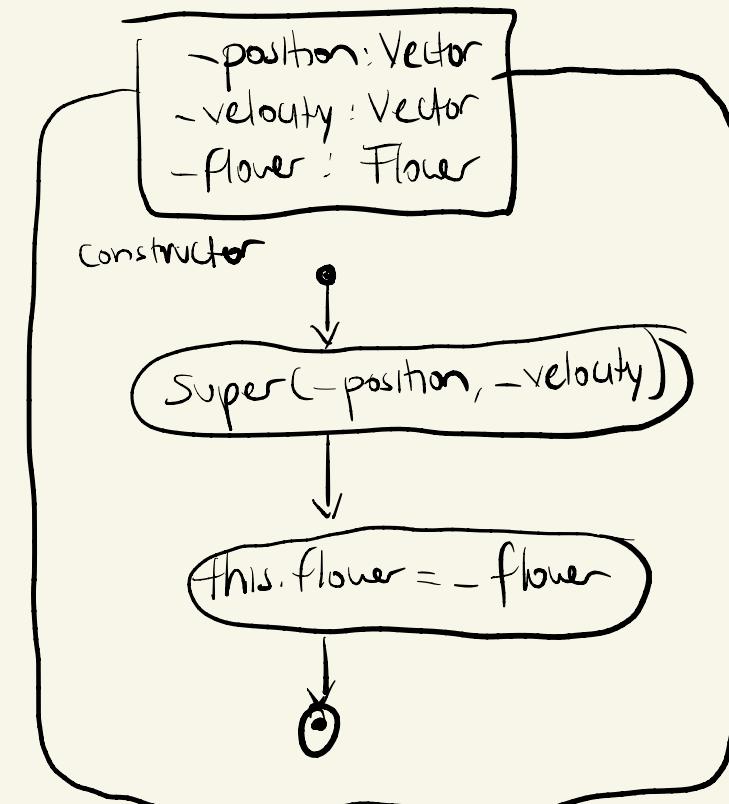
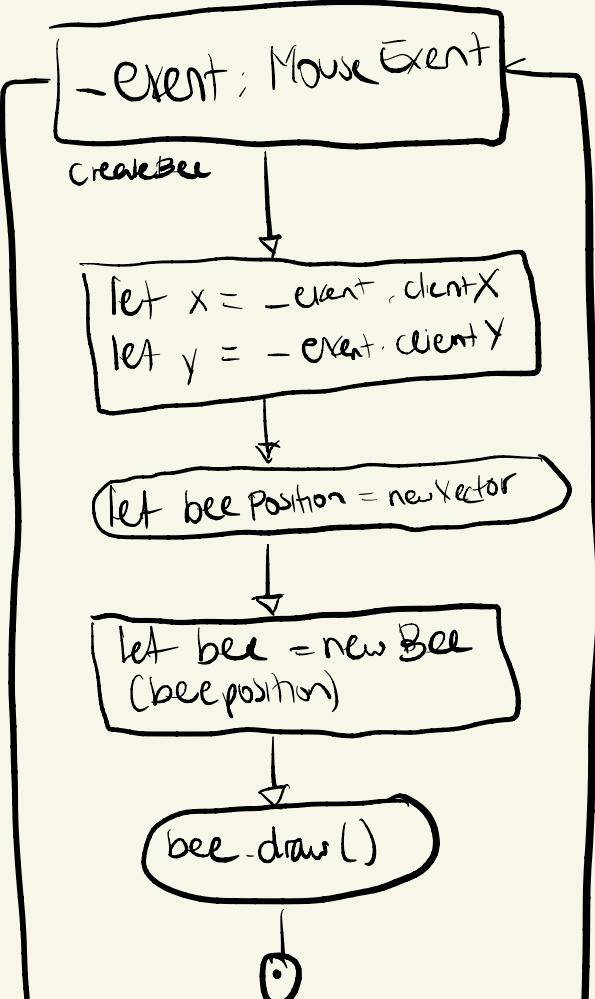


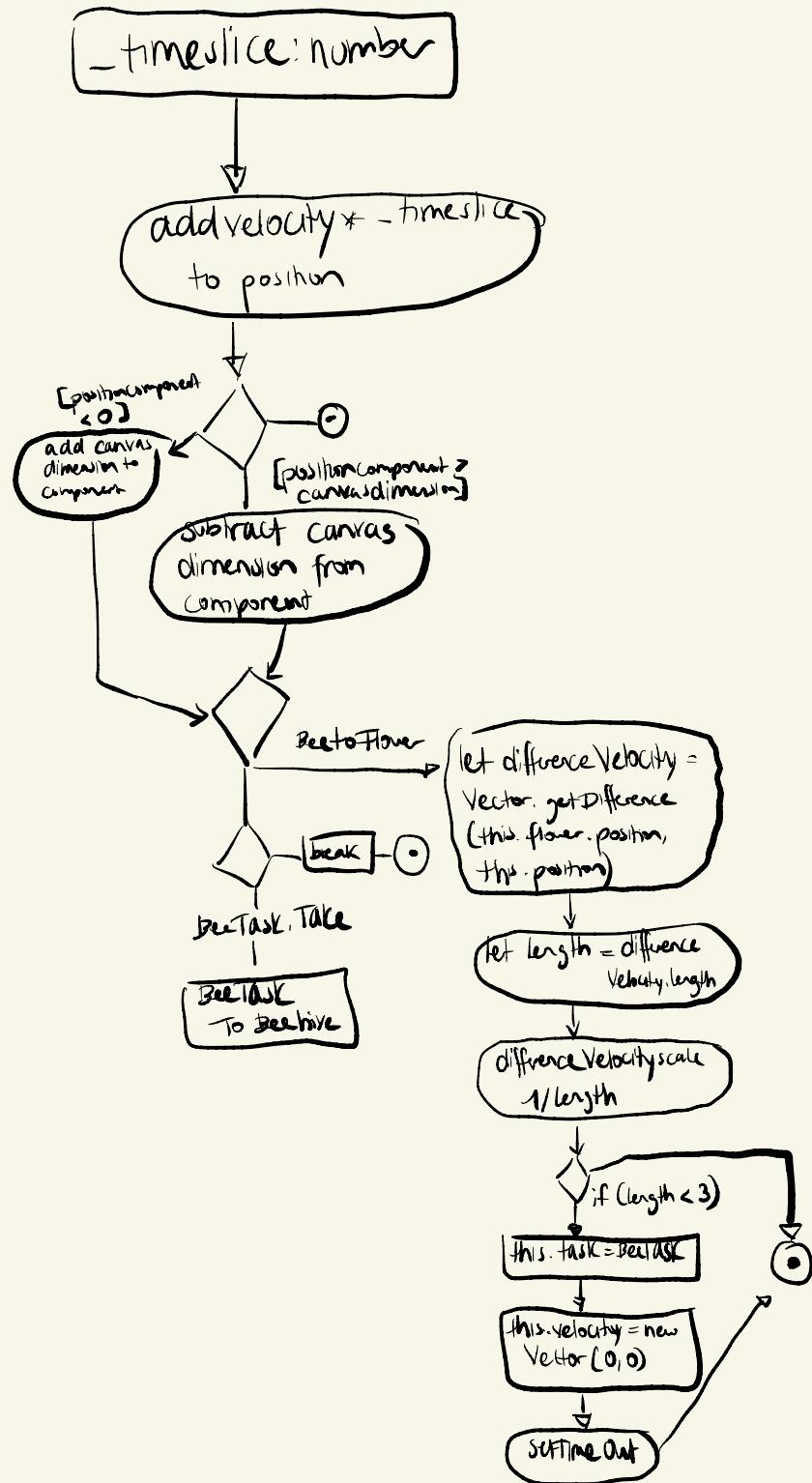
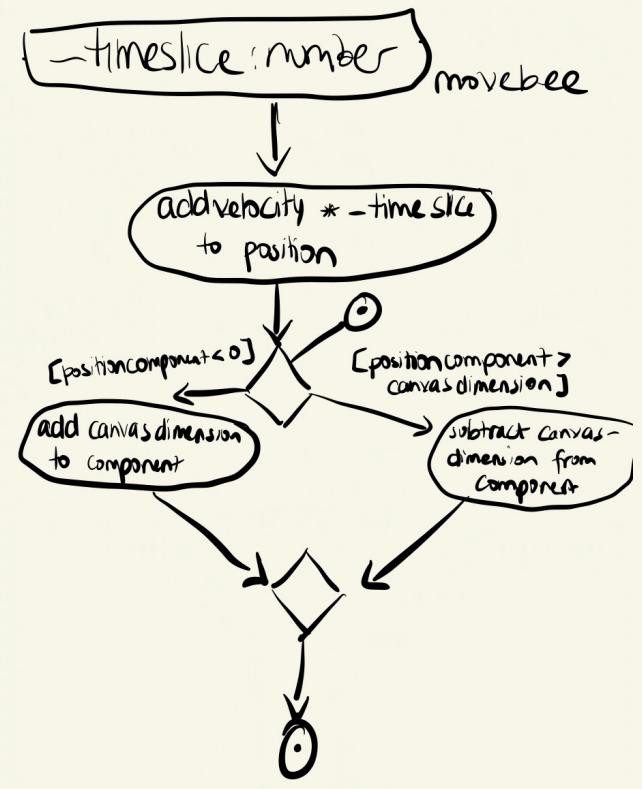
Activity Diagram



```

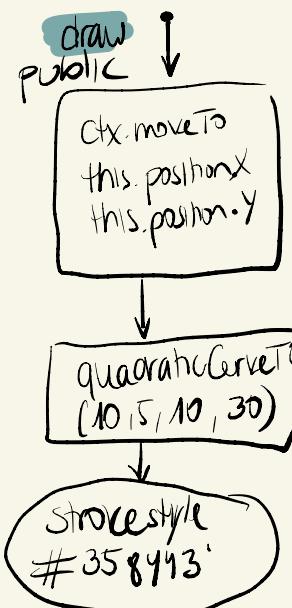
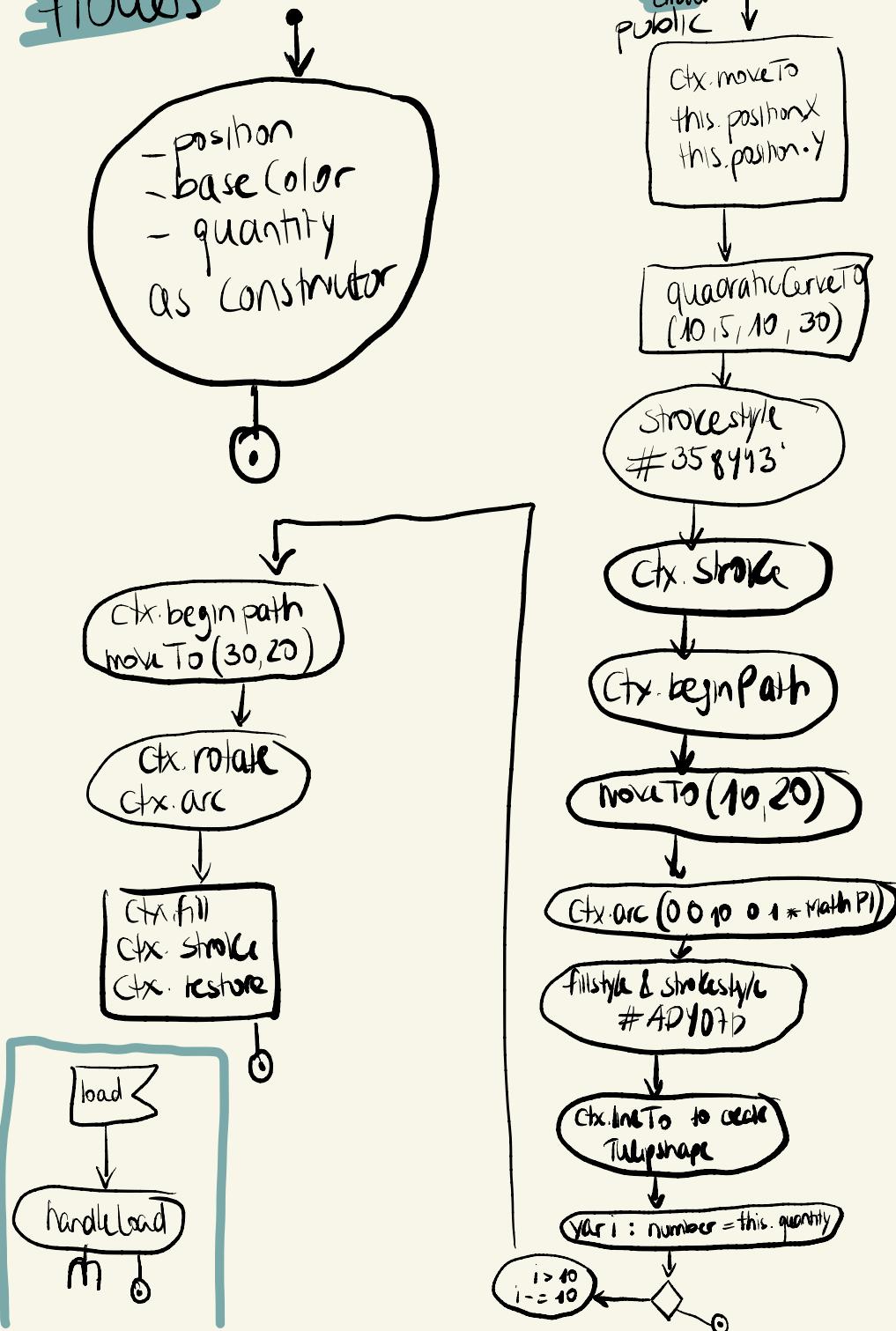
let canvas: HTMLCanvasElement
let ctx: CanvasRenderingContext2D
flowerArray1[] = []
flowerArray2[] = []
  
```





Der Rest bleibt
genauso wie L11.1

flowers



```

main = let horizon: number
let flowerArray: number []
let xCloudArray: number []
let yCloudArray: number []
let moveableArray: number []

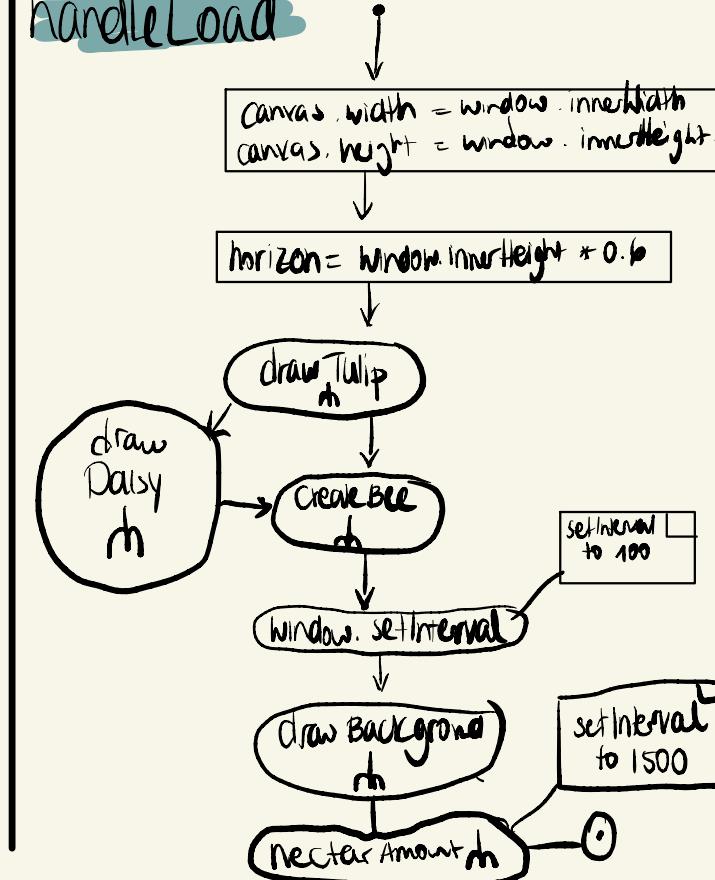
```

```

export let canvas: HTMLCanvasElement = document.querySelector("canvas")!
export let VectorMountains
x: number
y: number
export let ctx: CanvasRenderingContext2D

```

handleLoad



drawBackground / update
Animation

Create linear gradient
(0,0,0, ctx.canvas.height)

Add 4 gradient
Colorstops

ctx.fillStyle = gradient
ctx.fillRect(0,0, canvas.width
 , canvas.height)

for let i: number = 0
 ↓
 i < flowerArray.length
 i++
 draw()
 ↓

for let moveable of moveableArray

moveable.draw

Draw Moon

Draw Moon

```
let r1: number = 30  
let r2: number = 180
```

```
let gradientSun: CanvasGradient  
= ctx.createRadialGradient
```

Add 3 Colorstops
with gradientSun

```
let X: number = ctx.canvas.width  
let Y: number = ctx.canvas.height / 2 - 100
```

ctx.SAVE

ctx.translate
ctx.fillStyle = gradientSun
ctx.arc

ctx.fill
ctx.restore

moveable

abstract

Export class
Moveable

position: vector
velocity: vector

protected
both

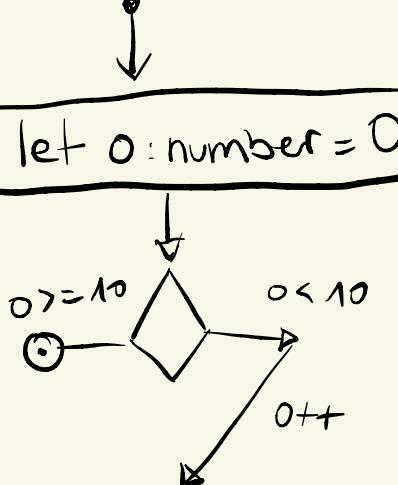
Create different move patterns
based on:

this.position.X
this.positionY

window.innerWidth
window.innerHeight

Create 2 situations:
true & false
(if loop)

drawDaisy



drawBeehive

ctx.save

ctx.strokeStyle "black"

ctx.fillStyle "orange"

ctx.translate (120, 500)

ctx.scale (0.3, 0.3)

ctx.lineWidth = 18

ctx.beginPath

Continue to drawBeehive

ctx.fillStyle = "black"

ctx.fill

ctx.restore

```
let randomX: number = math.random() * window.innerWidth
let randomY: number = 
```

```
let flowerPosition: vector = new Vector(randomX, randomY)
let flowerLeft2: Flower = new Flower(flowerPosition, 10)
```

```
flowerLeft2.draw2()
flowerArray2.push(flowerLeft2)
```