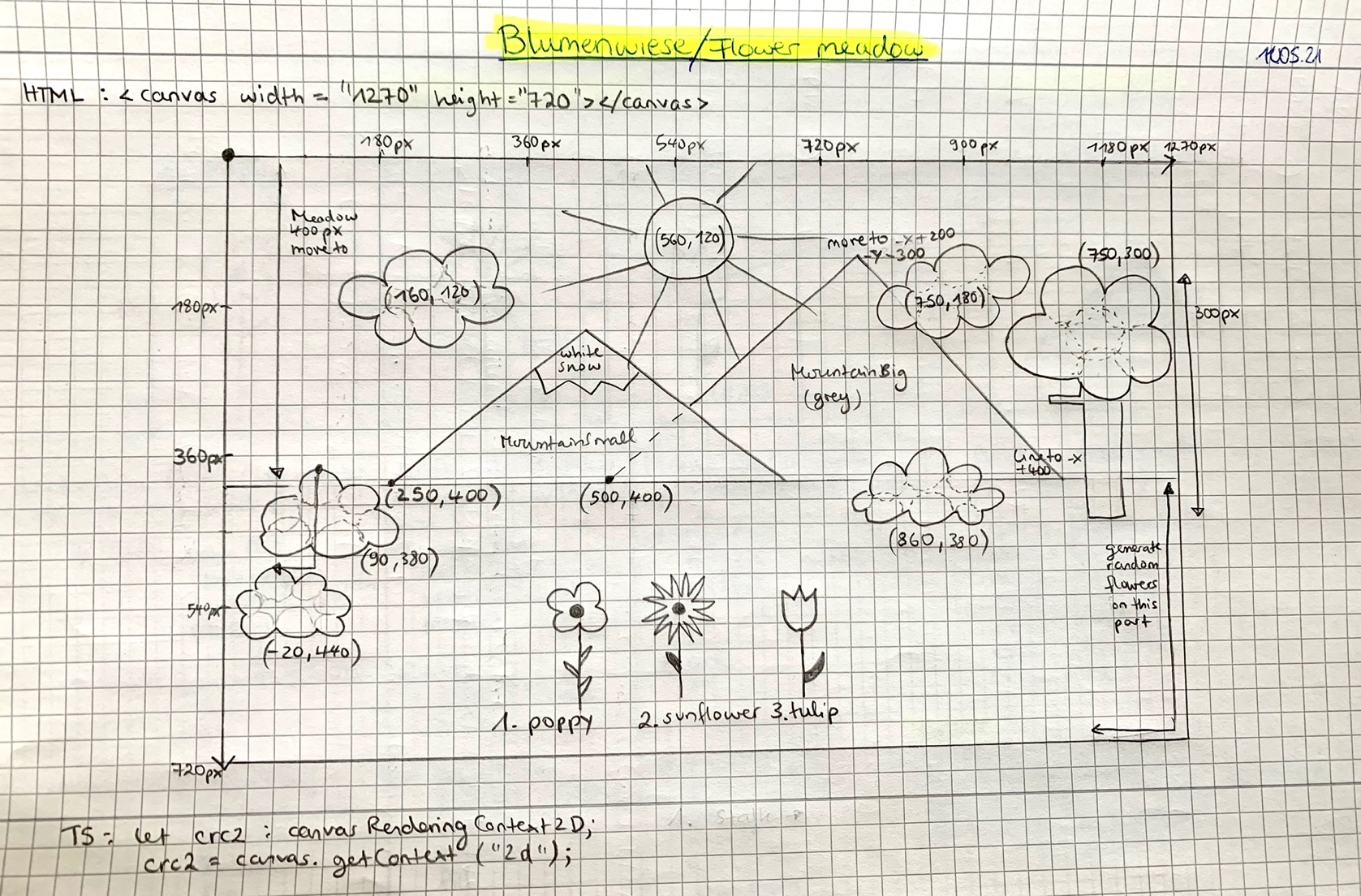
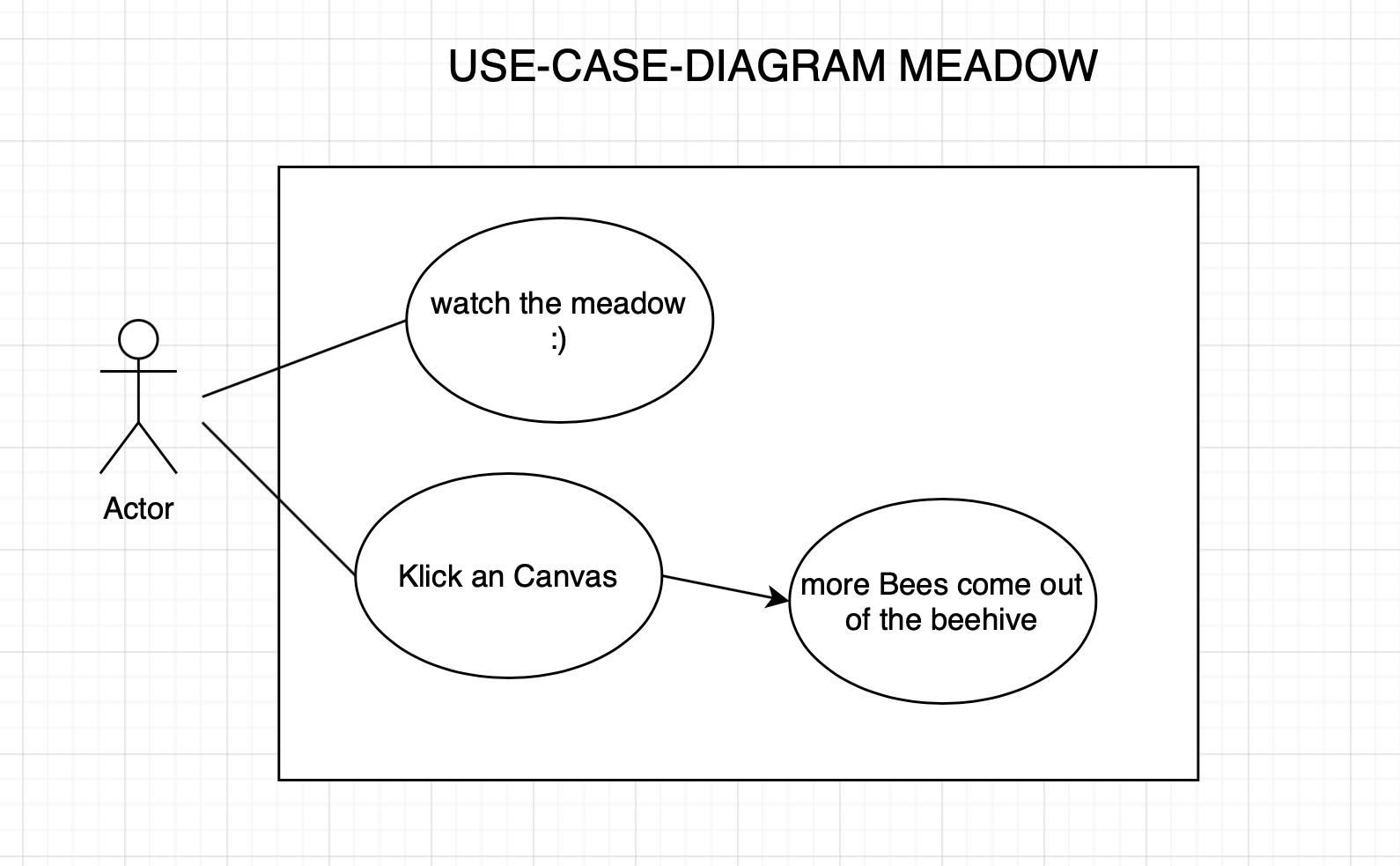
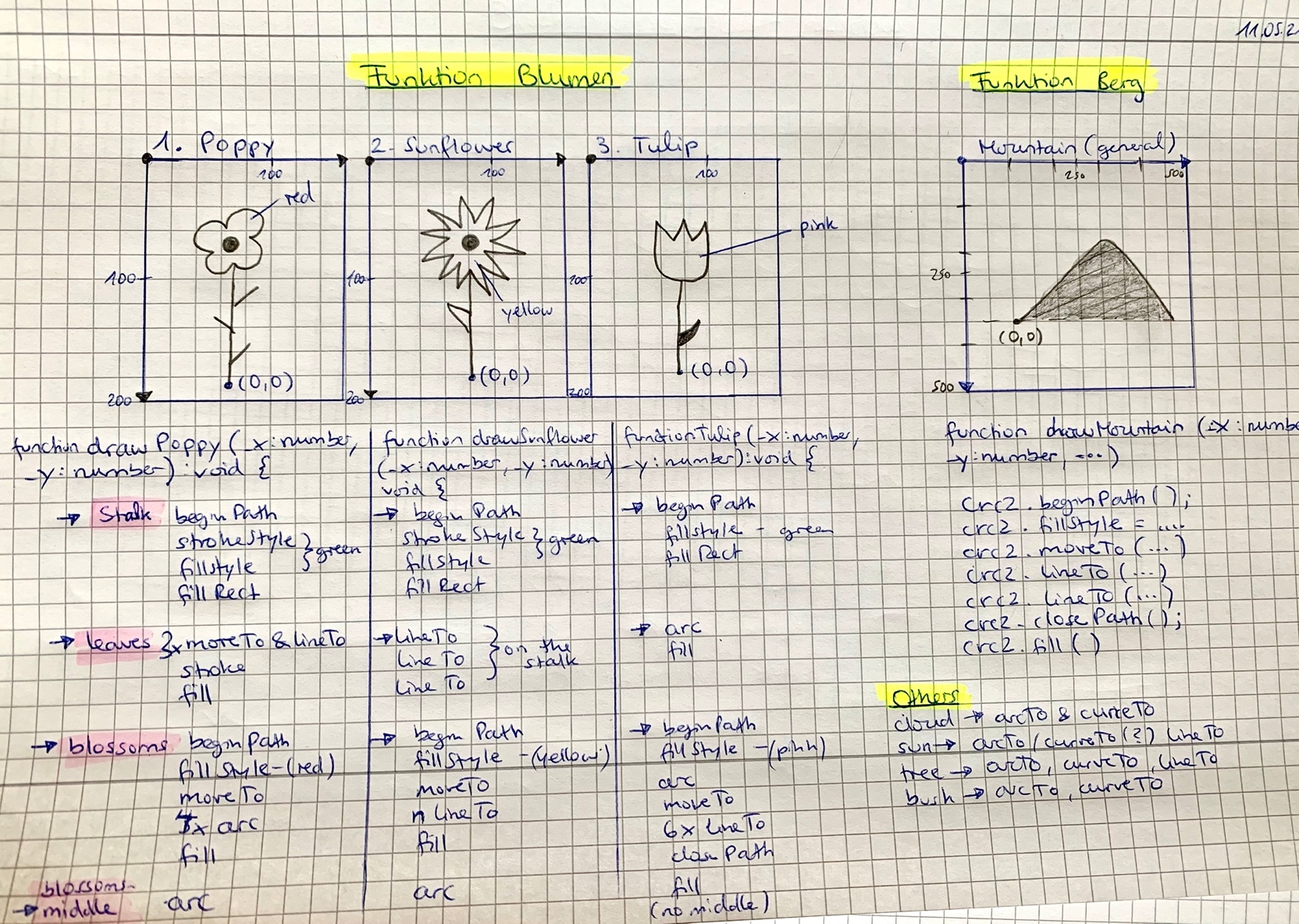
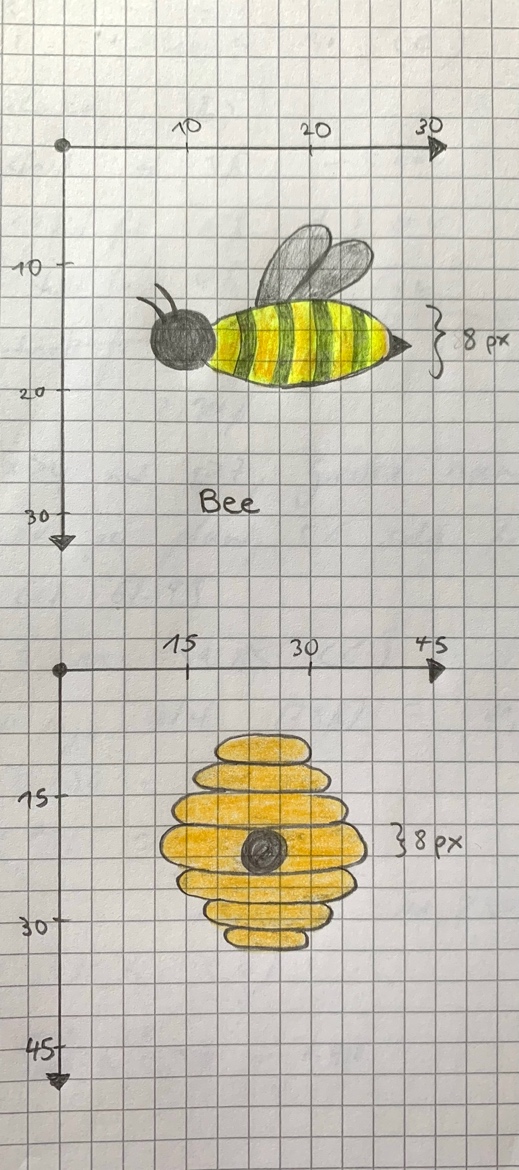
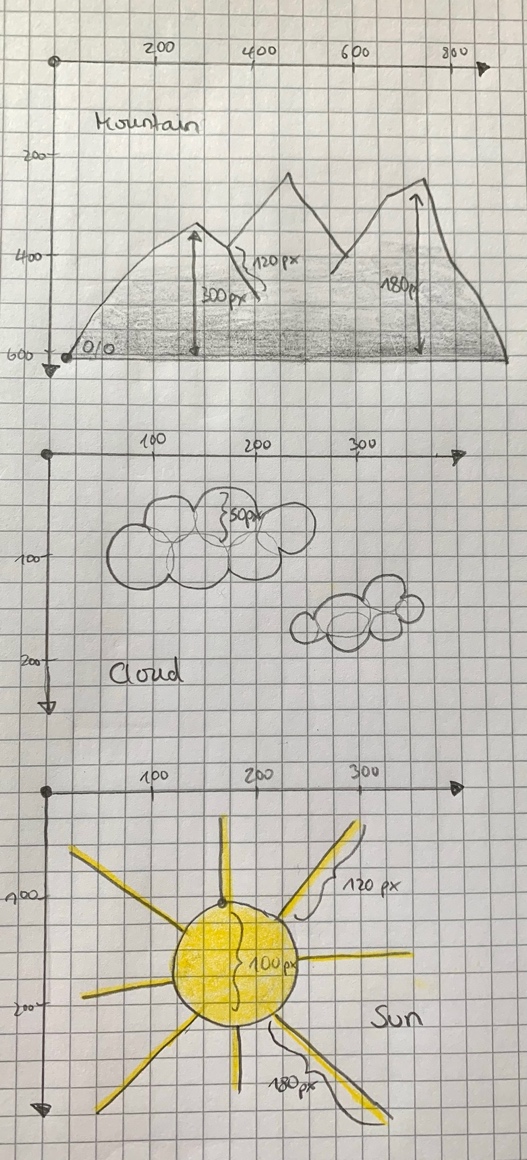
**KONZEPT MEADOW MONA STINGL**



Ein Bild, das Metall, Haufen enthält.

Automatisch generierte Beschreibung

CLASS-DIAGARM MEADOW

load

handleLoad

click or touch

moreBees

export canvas : HTML CanvasElement

export crc2 : CanavsRendering Context2D

let imgData: ImageData

install load listener

let bigClouds: Cloud[] = []

let smallClouds: Cloud[] = []

let z: number = 10

let allBees: Bee[] = []

let nectarFlowers: Flowers[] = []

x: number

y: number

type: string

**FLOWERS**

constructor()

draw()

setRandomPosition()

x: number

y: number

color: string

sting: boolean

**BEE**

constructor()

drawBee()

move()

update()

setRandomStyle()

animate()

x: number

y: number

speed: number

**CLOUD**

constructor()

drawCloud()

move()

animate()

CanvasRenderingContext

**BACKGROUND**

constructor()

drawMeadow()

drawSky()

drawMountain()

drawSun()

drawTree()

drawBird()

drawBush()

drawBank()

drawBeehive()

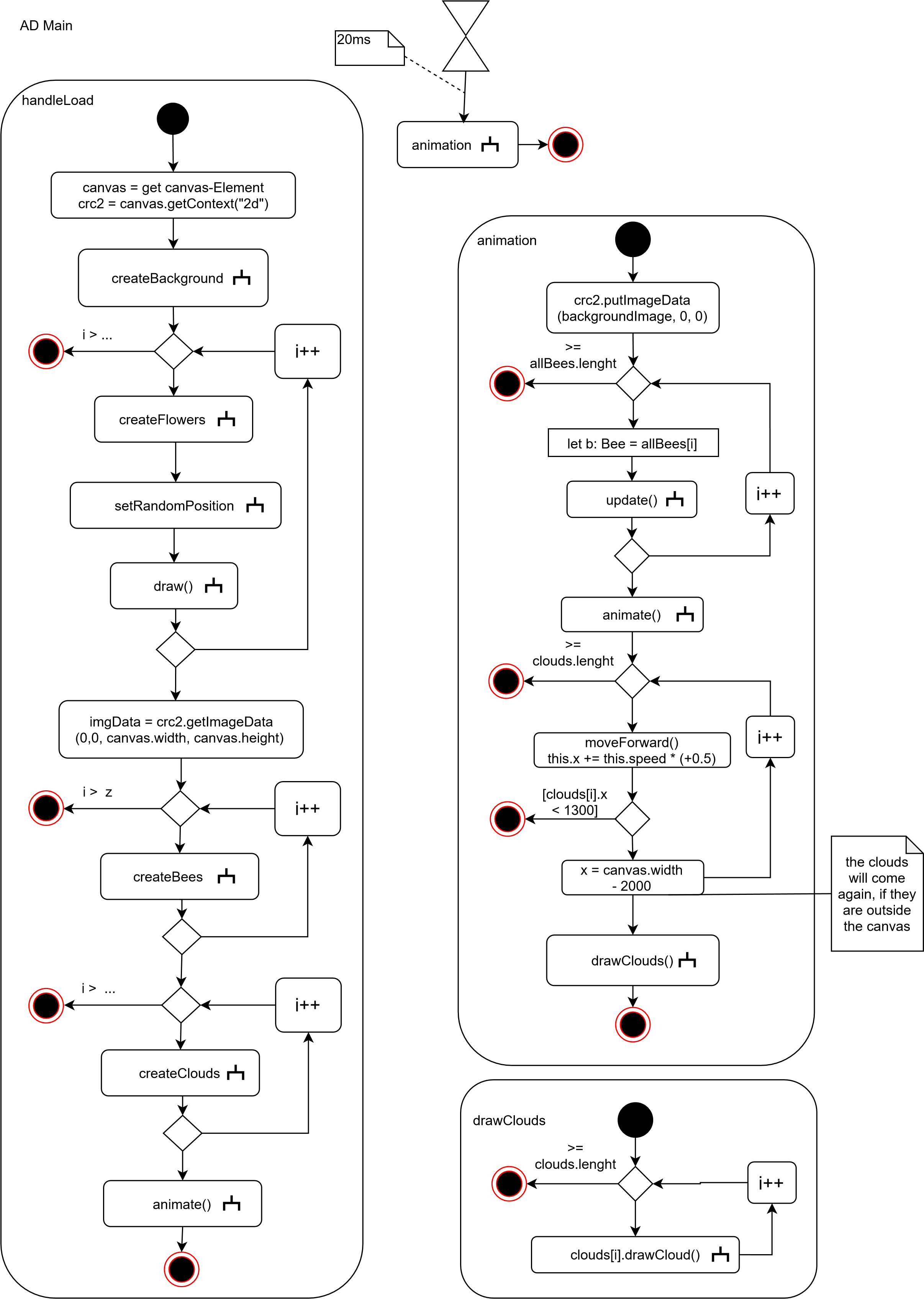
x: number

y: number

strokeColor: string

fillColor: string

ACTIVITY-DIAGRAM MEADOW



nectarFlowers: Flowers[] = []

i: number

x: number

y: number

for-Loop

nectarFlowers

save x from nectarFlowers []

save y from nectarFlowers []

move to x, y

move to beehive

moreBees

b.setRandomStyle()

allBees.push(b)

z++

let b: Bee = new Bee

\_event: Event

drawMeadow

beginPath()

strokeStyle

fillStyle

moveTo

lineTo

closePath()

stroke()

fill()

gradient: CanvasGradient =

crc2.createLinearGradient

drawSky

beginPath()

strokeStyle

fillStyle

moveTo

lineTo

closePath()

stroke()

fill()

gradient: CanvasGradient =

crc2.createLinearGradient

drawMountain

beginPath()

strokeStyle

fillStyle

lineWidth

moveTo

lineTo

fill()

stroke()

closePath()

gradient: CanvasGradient =

crc2.createLinearGradient

\_x: number

\_y: number

\_x: number

\_y: number

\_fillColor: string

\_x: number

\_y: number

\_strokeColor: string

beginPath()

strokeStyle

moveTo

lineTo

stroke()

drawSun

\_x: number

\_y: number

\_strokeColor: string

\_fillColor: string

beginPath()

strokeStyle

fillStyle

arc

moveTo

lineTo

closePath()

stroke()

fill()

AD Background

drawTree

\_x: number

\_y: number

beginPath()

strokeStyle

fillRect

beginPath

fillStyle

arc

closePath()

fill()

drawBird

\_x: number

\_y: number

\_strokeColor: string

\_fillColor: string

beginPath()

strokeStyle

fillStyle

arc

moveTo

lineTo

closePath()

fill()

stroke()

drawBush

\_x: number

\_y: number

\_fillColor: string

beginPath()

fillStyle

arc

closePath()

fill()

drawBank

\_x: number

\_y: number

\_strokeColor: string

\_fillColor: string

beginPath()

strokeStyle

fillStyle

moveTo

lineTo

closePath()

fill()

stroke()

drawBeehive

\_x: number

\_y: number

beginPath()

strokeStyle

arc

rect

lineWidth

stroke()

fill()

AD Background

drawCloud

beginPath()

fillStyle

arc

closePath()

fill()

AD Clouds

moveForward

this.x += this.speed + (+0.5)

AD Bees

update

this.drawBee()

this.move()

drawBee

beginPath

strokeStyle

fillStyle

arc

beginPath

fillStyle

rect

stroke()

fill()

moveTo()

lineTo()

[

]

sting == true

stroke()

fill()

beginPath

fillStyle

arc

moveTo()

arc()

closePath()

fill()

move

this.x += Math.random() \* 5 - 4

this.y += Math.random() \* 6 - 3

i >= z

this.x < 0

this.y = crc2.canvas.width

i++

this.y = 0

this.x =

crc2.canvas.width

this.y < 0

this.y >=

crc2.canvas.height

setRandomStyle

let randomColor: string = random color

between yellow and red

let randomSting: boolean =

Boolean(Math.round(Math.random()))

this.color = randomColor

this.sting = randomSting

