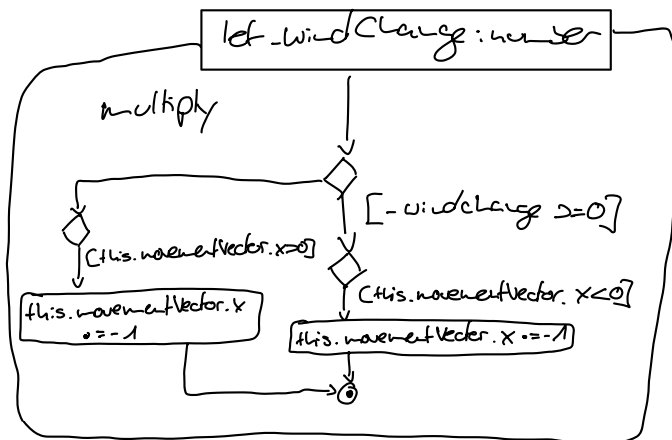**Moveable**

name: String
startingX: number
startingY: number
movementVector: Vector
color: string

---

constructor (_name:String, _startingX: number
    _startingY: number, _movementVector: Vector,
    _color: string)
createVector(): Vector
movement(): void
reset(): void

---

let _windChange: number

multiply

[_windChange >=0]

[this.movementVector.x>0]
this.movementVector.x *= -1

[this.movementVector.x<0]
this.movementVector.x *= -1

---

createVector

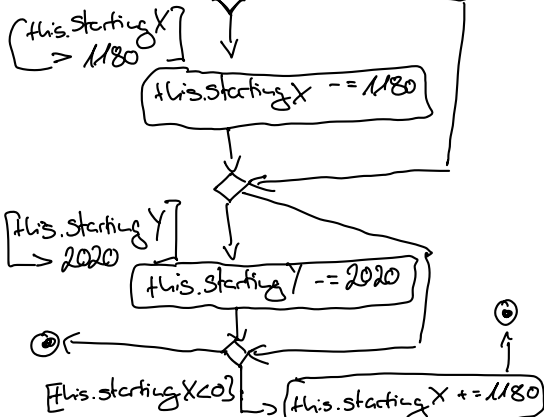let velocity: number = 5

let random Vector: Vector = new Vector
    (velocity · random , velocity · random)

return random Vector

---

movement

this.startingX += this.movementVector.x
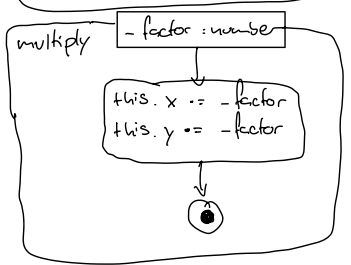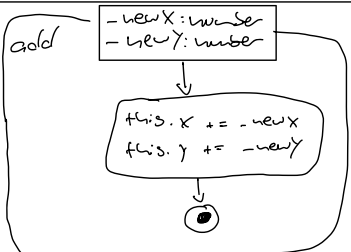this.starting Y += this.movementVector.y

---

reset

[this.startingX]
> 1180
this.startingX -= 1180

[this.starting Y]
> 2020
this.starting Y -= 2020

[this.startingX<0]
this.startingX += 1180

Vector

X: number
Y: number

---

constructor (_X: number, _Y: number)
add (_newX: number, _newY: number)
multiply (_factor: number)

add

_newX: number
_newY: number

this.X += _newX
this.Y += _newY

multiply

_factor: number

this.X *= _factor
this.Y *= _factor

Bug

super(_name, _startingX, _startingY, _color)

---

constructor(_name: string, _startingX: number,
_startingY: number, _color: string)
movement(): void

movement

super.movement()

draw Bug
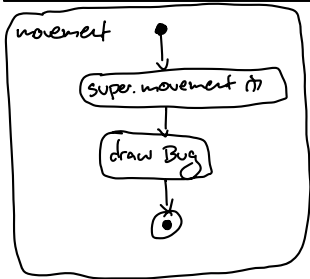
Balloon

super(_name, _startingX, _startingY, _color)

---

constructor(_name: string, _startingX: number,
_startingY: number, _color: string)
movement(): void

movement

super.movement()

draw Balloon
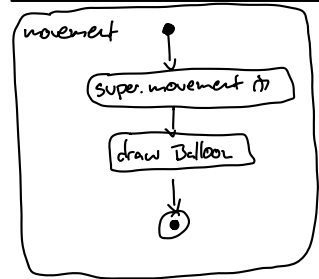
Paraglider

super(_name, _startingX, _startingY, _color)

---

constructor(_name: string, _startingX: number,
_startingY: number, _color: string)
movement(): void

movement

super.movement()

draw Paraglider ()

changeToPerson

return false

[this.startingY > 1600]

return true

changeToParaglider

return false

[this.startingY < 300]

return true

```
●
↓
┌─────────────────────────┐
│ add Event Listener a    │
│ "load"                  │
└─────────────────────────┘
        ↓

┌──────────────────────────────────────┐
│ let crc2 : Canvas Rendering Context 2D│
│ let moveables : Moveable[] = []       │
│ let persa Vector : Vector = new Vector(0, -3) │
│ let imgData : Image Data              │
│ let wind : number = 0                 │
│ let time : string = "day"             │
└──────────────────────────────────────┘
        ↓
       (a)
```
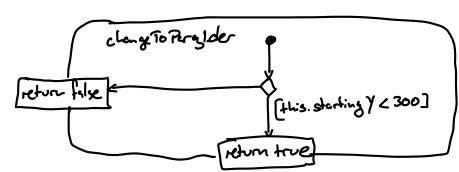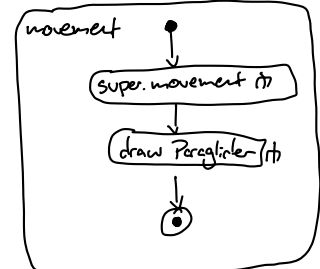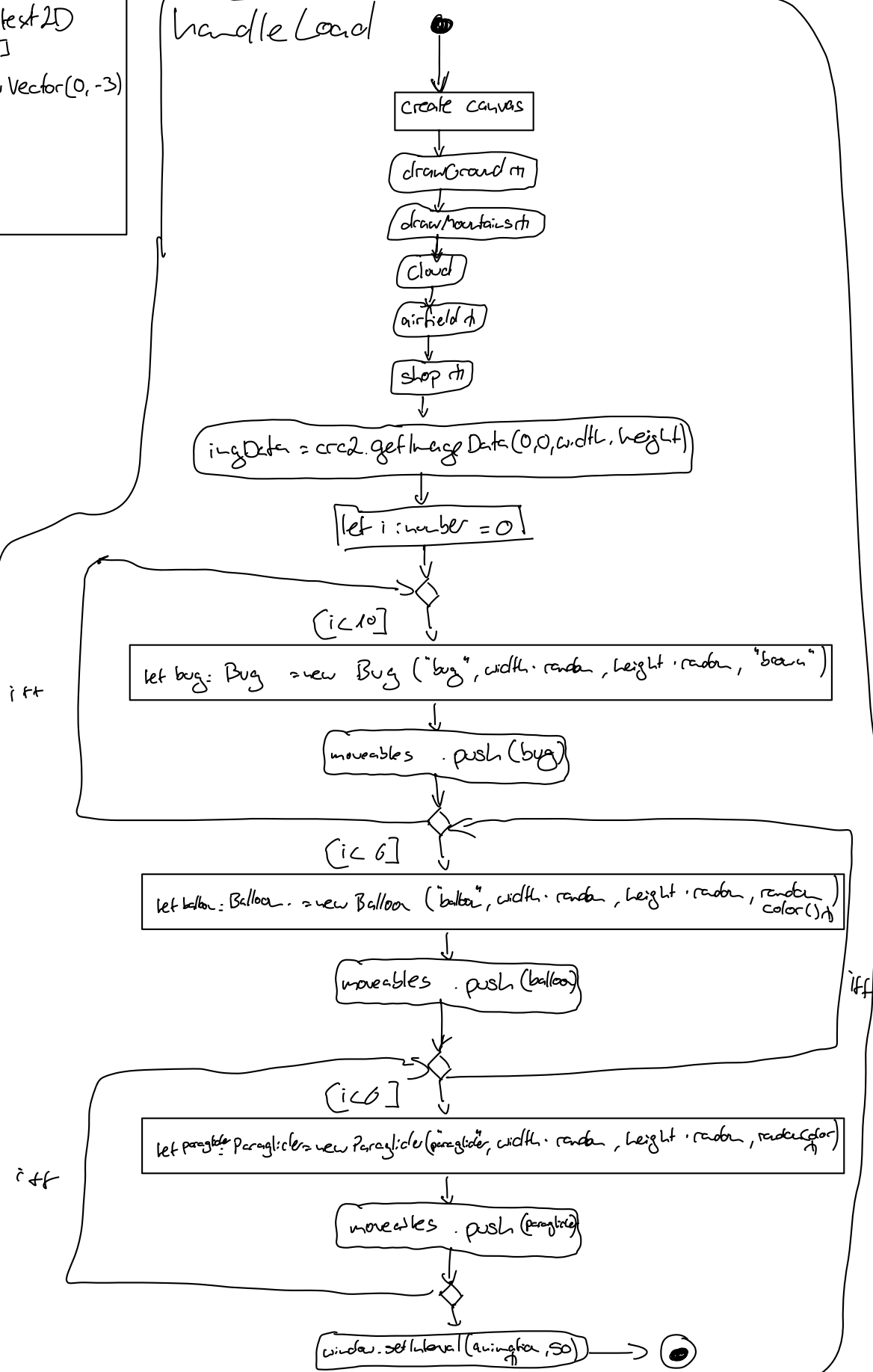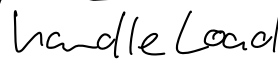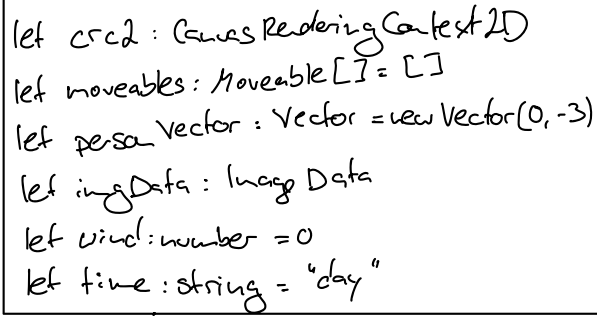
```
> load                  > keydown              > click
  ↓                        ↓                      ↓
┌──────────────┐    ┌──────────────┐    ┌──────────────┐
│ handle Load ↱│    │ checkArrowkey│    │ changeTime   │
└──────────────┘    │      ↱       │    │     ↱        │
                    └──────────────┘    └──────────────┘
```

```
handle Load                    ●
                               ↓
                        ┌──────────────┐
                        │ create canvas│
                        └──────────────┘
                               ↓
                        ( drawGround ↱ )
                               ↓
                        ( drawMountains ↱ )
                               ↓
                        ( cloud )
                               ↓
                        ( airfield ↱ )
                               ↓
                        ( shop ↱ )
                               ↓
          ( imgData = crc2.getImage Data(0,0,width,height) )
                               ↓
                  │ let i : number = 0 │
                               ↓
                             ◇ ←───────────────────
                    [i < 10]  ↓
   ┌──────────────────────────────────────────────────────────┐
   │ let bug : Bug  = new Bug("bug", width·random, height·random, "brown") │
   └──────────────────────────────────────────────────────────┘
                               ↓
              i++      ( moveables .push(bug) )
                               ↓
                             ◇
                    [i < 6]   ↓
   ┌──────────────────────────────────────────────────────────┐
   │ let ballon : Balloon = new Balloon("ballon", width·random, height·random, random  │
   │                                                            color() ↱)│
   └──────────────────────────────────────────────────────────┘
                               ↓
                    ( moveables .push(ballon) )        i++
                               ↓
                             ◇
                    [i < 6]   ↓
   ┌──────────────────────────────────────────────────────────┐
   │ let paraglider : Paraglider = new Paraglider(paraglider, width·random, height·random, randacolor) │
   │                                                            ↱│
   └──────────────────────────────────────────────────────────┘
            i++                ↓
                    ( moveables .push(paraglider) )
                               ↓
                             ◇
                               ↓
          ( window.setInterval(animation ↱, 50) ) ──→  (b)
```

Animation

ctx2.putImageData(imgData, 0,0)

[time === "night"] ◇ [time === day]

noon() ← ◇ → sun()

◇ → ♡

[inc moveables length]

moveables.movement()

moveables.reset()

drawGround

create gradient

add color stops
0 , blue
0.2 , lightblue
0.6 , gray
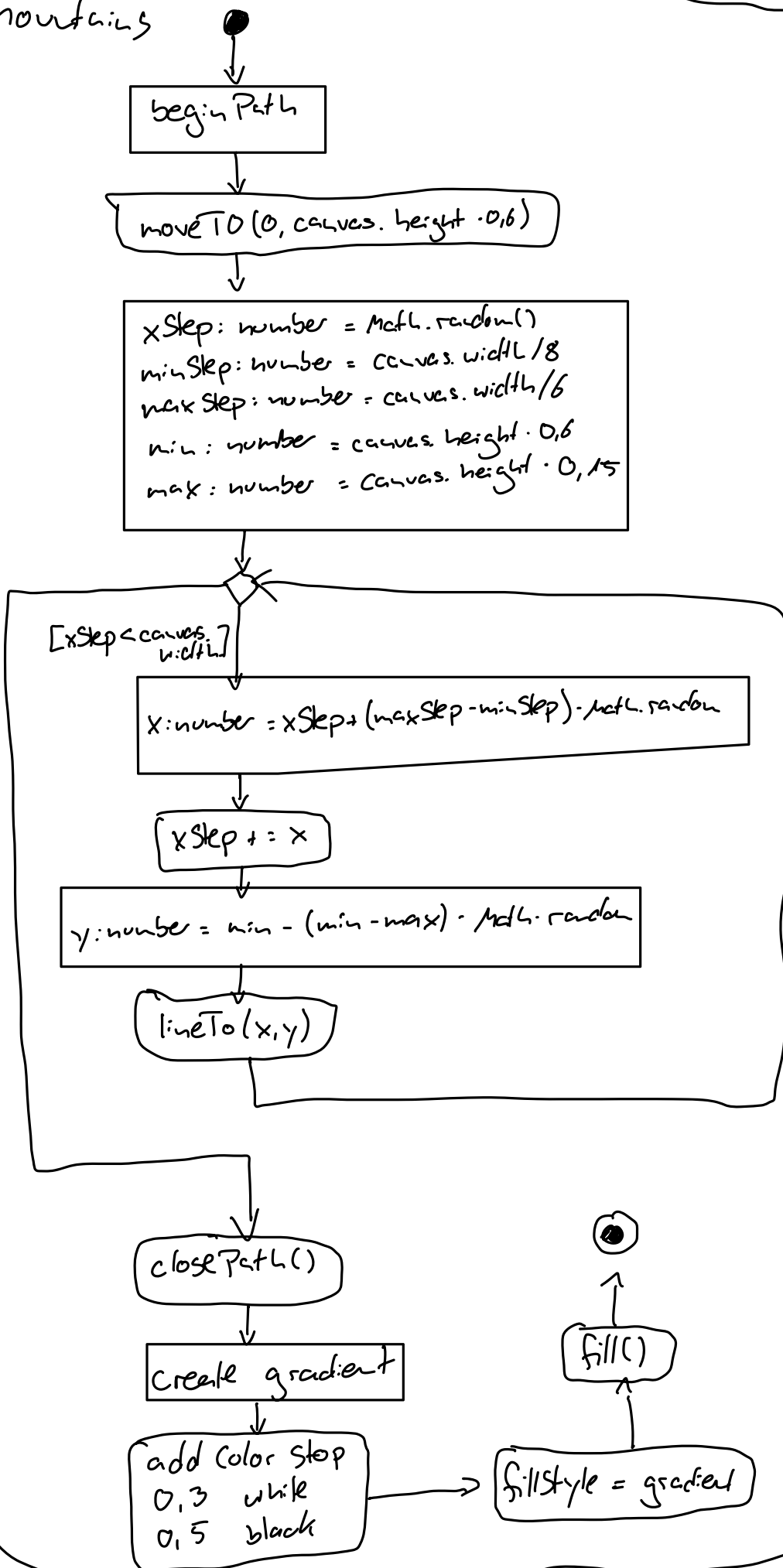1 , green

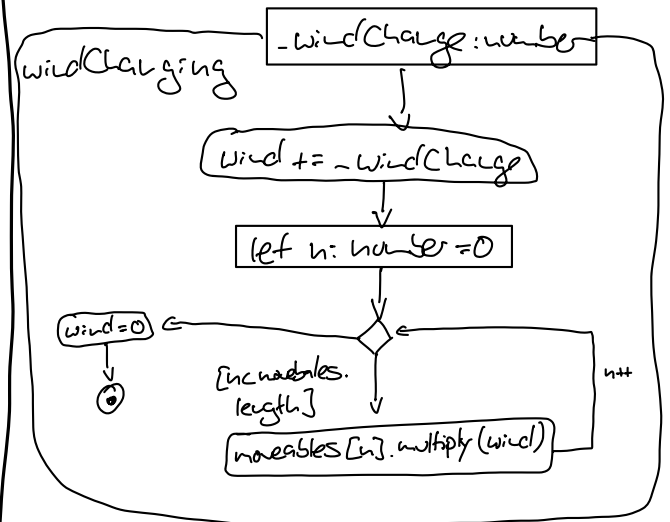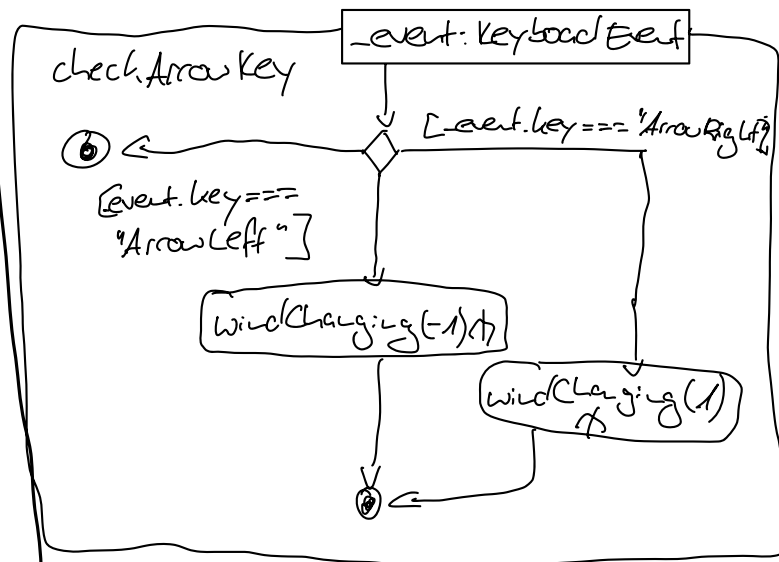fillStyle = gradient

fillRect(0,0, canvas.width, canvas.height)

randomColor

let r : number = Math.random() · 256
let g : number = Math.random() · 256
let b : number = Math.random() · 256

"rgb(" + r + ", " + g + ", " + b + ")"

drawMountains

```
beginPath
```

moveTO (0, canvas.height · 0,6)

```
xStep: number = Math.random()
minStep: number = canvas.width /8
maxStep: number = canvas.width /6
min: number = canvas.height · 0,6
max: number = canvas.height · 0,15
```

[xStep < canvas.
width]

```
X: number = xStep+(maxStep - minStep) · Math.random
```

xStep += x

```
y: number = min - (min - max) · Math.random
```

lineTo(x,y)

closePath()

```
create gradient
```

add color Stop
0,3 while
0,5 black

fillStyle = gradient

fill()

Sun

```
● (start)
  ↓
[Create gradient]
  ↓
[add color stop
 0    white
 1    yellow]
  ↓
[min : number = 0
 max: number = height · 0,15]
  ↓
(save())
  ↓
(translate (width · random, (max - min) · random))
  ↓
[sun = new Path2D]
  ↓
[sun.arc (0,0,75,0,2·PI)]
  ↓
(fillStyle = gradient)
  ↓
(fill (sun))
  ↓
(restore)
  ↓
● (end)
```

checkArrowKey

```
_event: KeyboardEvent
  ↓
◇ ──[event.key === "ArrowRight"]──→ (windChanging(1))
  │                                        ↓
[event.key ===            ● (end)
 "ArrowLeft"]
  ↓
(windChanging(-1))
  ↓
● (end)
```

windChanging

```
_windChange: number
  ↓
(wind += _windChange)
  ↓
[let n: number = 0]
  ↓
◇ ──[n++]
  │
  [n < moveables.length]
  ↓
(moveables[n].multiply (wind))

(wind = 0) ←─ ◇
  ↓
● (end)
```

paraglider

⬤

```
let n: number = 0
```

⬤

◇

[n < paragliderArray.
length]

```
let x: number = paragliderArray [n]. startingX
let y: number = paragliderArray [n]. startingY
let size: number = 60
```

◇

paragliderArray [n].
closeToPerson
== true &&
paragliderArray [n].
name = " paraglider

```
. movementVector. x = person Vector.x
. movementVector. y = person Vector. y
. name = "climber
```

(draw Person)

◇

[name == "climber"]

```
. movementVector. x = person Vector.x
. movementVector.y = person Vector.y
```

(draw Person)

◇

[. closeToParaglider == true &&
. name == "climber"]

```
. name = " paraglider'
. movementVector.x = . createVector().x ↑
. movementVector.y = . createVector().y ↑
```
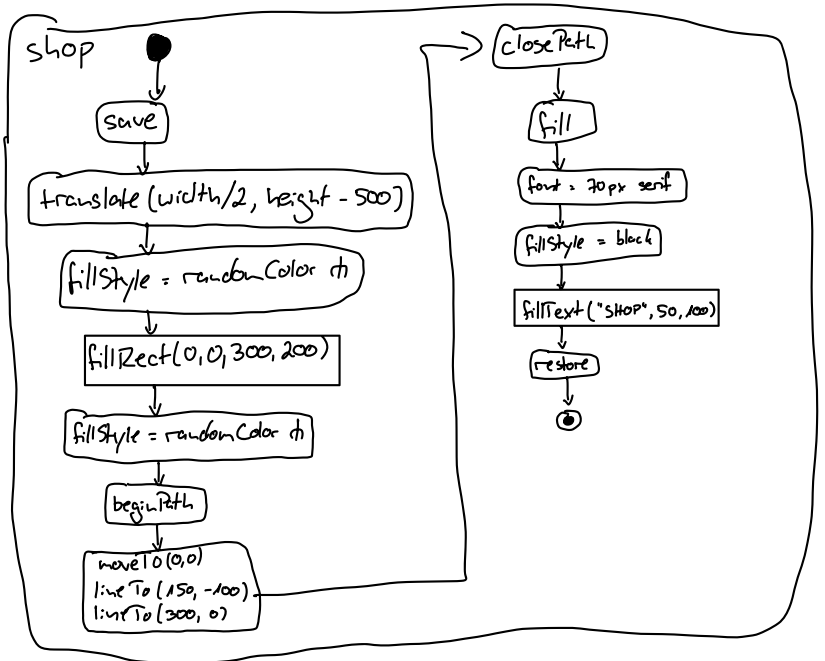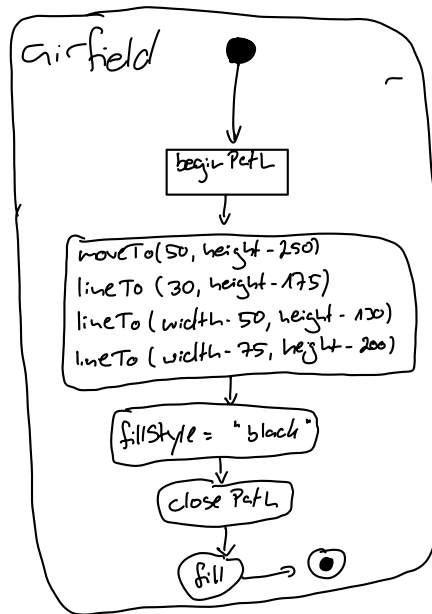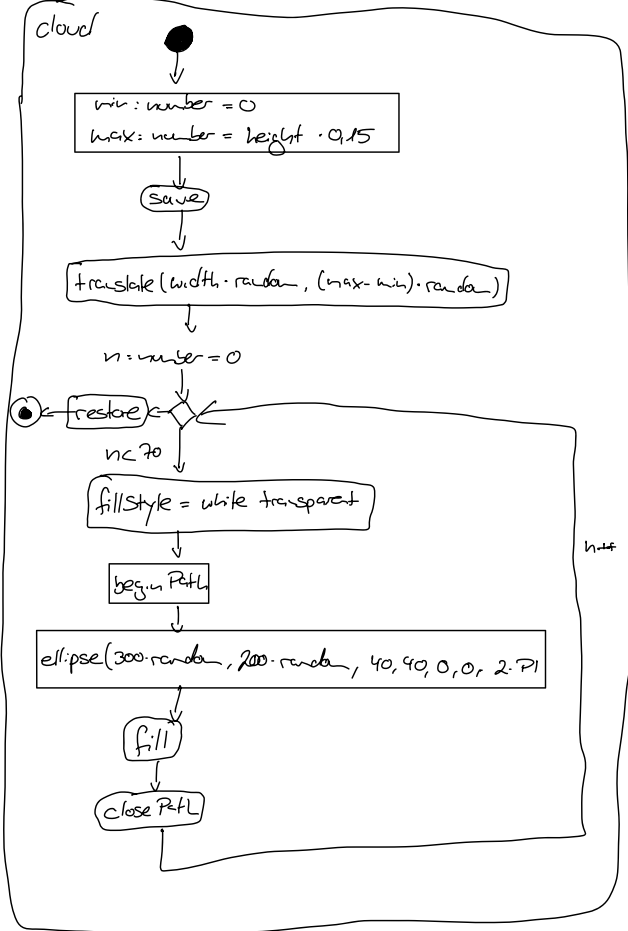
(draw paraglider)

◇

name == "paraglider"

n++

(.reset() ↑)

(. movement() ↑)

(draw paraglider)

cloud

```
min : number = 0
max : number = height · 0.15
```

save

translate(width · random, (max - min) · random)

n : number = 0

restore ◉

n < 70

fillStyle = white transparent

begin Path

ellipse(300 · random, 200 · random, 40, 40, 0, 0, 2 · PI)

fill

close Path

airfield

begin Path

```
moveTo(50, height - 250)
lineTo (30, height - 175)
lineTo (width · 50, height - 130)
lineTo (width - 75, height - 200)
```

fillStyle = "black"

close Path

fill → ◉

shop

save

translate (width/2, height - 500)

fillStyle = randomColor ()

fillRect(0, 0, 300, 200)

fillStyle = randomColor ()

begin Path

```
moveTo (0, 0)
lineTo (150, -100)
lineTo (300, 0)
```

close Path

fill

font = 70px serif

fillStyle = black

fillText ("SHOP", 50, 100)

restore

◉

noon

```
Create gradient
        ↓
add color stop
0   white
1   light blue
        ↓
min : number = 0
max : number = height · 0,15
        ↓
      save()
        ↓
translate (width · random, (max·min)·random)
        ↓
sun = new Path2D
        ↓
sun.arc (0,0,75,0,2·PI)
        ↓
fillStyle = gradient
        ↓
fill (sun)
        ↓
     restore
        → ⬤
```