Klassendiagramm

CanvasRenderingContext

Vector

x : number
y : number

---

constructor (_x : number, y : number)
set (_x : number, y : number) : void
scale (_factor : number) : void
add (_addend : vector) : void
copy () : vector
random (_minLength, _maxLength) : vis

Moveable

position : vector
velocity : vector
size : number

---

constructor (position : velcity)
move (_timeline : number)
draw () : void

Leaf

size : number
type : number
speed : vector

---

constructor ()
draw () : void

squirrel

size : vector
food : nut

---

constructor ()
draw () : void

cloud

size : number
speed : vector

---

constructor ()
draw () : void

nut

---

constructor ()
draw () : void

Aktivitätendiagramm

Moveable

● → install load listeners → ◉

▷ load → handload rh → ◉

▷ click → nut rh → ◉

draw ● ↓ ◉

handload ●
- Let moveables: Moveable[] :[]
- window add eventListener
- get cre 2
- draw Background rh
- create Sun()
  create mountains()
  create trees()
  create bushes()
- call draw Cloud
  draw leaf
  draw squirrel
  draw nut
- ◉

constructor
- position: vector
- velocity: vector
- size: number
- Set position to _position
- Set velocity to _velocity
- ◉

move _timeslice: number
- add velocity * timeslice to position
- ◉ ◇
  - add canvas dimension to component
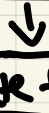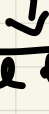 - subtract canvas dimension from component
- ◇ → ◉

Update
● → background → ◇ → drawable.move() → drawable.draw()
◉
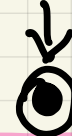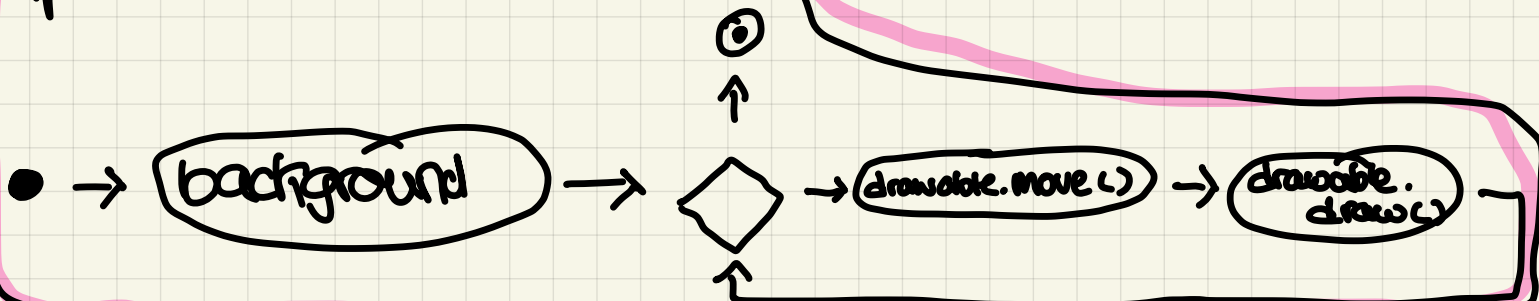
draw

●
↓
(save transform)
↓
(translate to position)
↓
(scale to size)
↓
(draw path representing type)
↓
(restore transform)
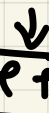
Cloud
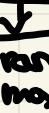
constructor ●
↓
(super (_position, _velocity))
↓
(set velocity to random direction and random >min <max)
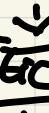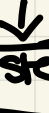↓
◉

draw

●
↓
(begin Path)
↓
(radiusParticle: number / Particle: Path2D gradient: radial)
↓
(particle.arc)
↓
(add color steps to gradient)
↓
(save transform)
↓
(fill with gradient)
↓
(arc x,y 2* Math.PI)
↓
(close Path)
↓
(fill) → (restore) → ◉
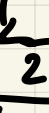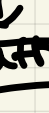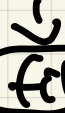
Squirrel

draw   _ size: vector
↓
(draw body)
↓
(begin Path & fillStyle = „brown")
↓
(draw arcs with 2* Math.PI)
↓
(close Path and fill)
↓
(draw Eye)
↓
(begin Path & fillStyle)
↓
(draw arcs with 2* Math.PI)
↓
(close Path & fill)
↓
(draw Ear & Tail with arc 2* Math.PI)
↓
close Path   → ◉
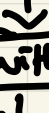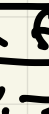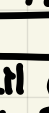
constructor ●
↓
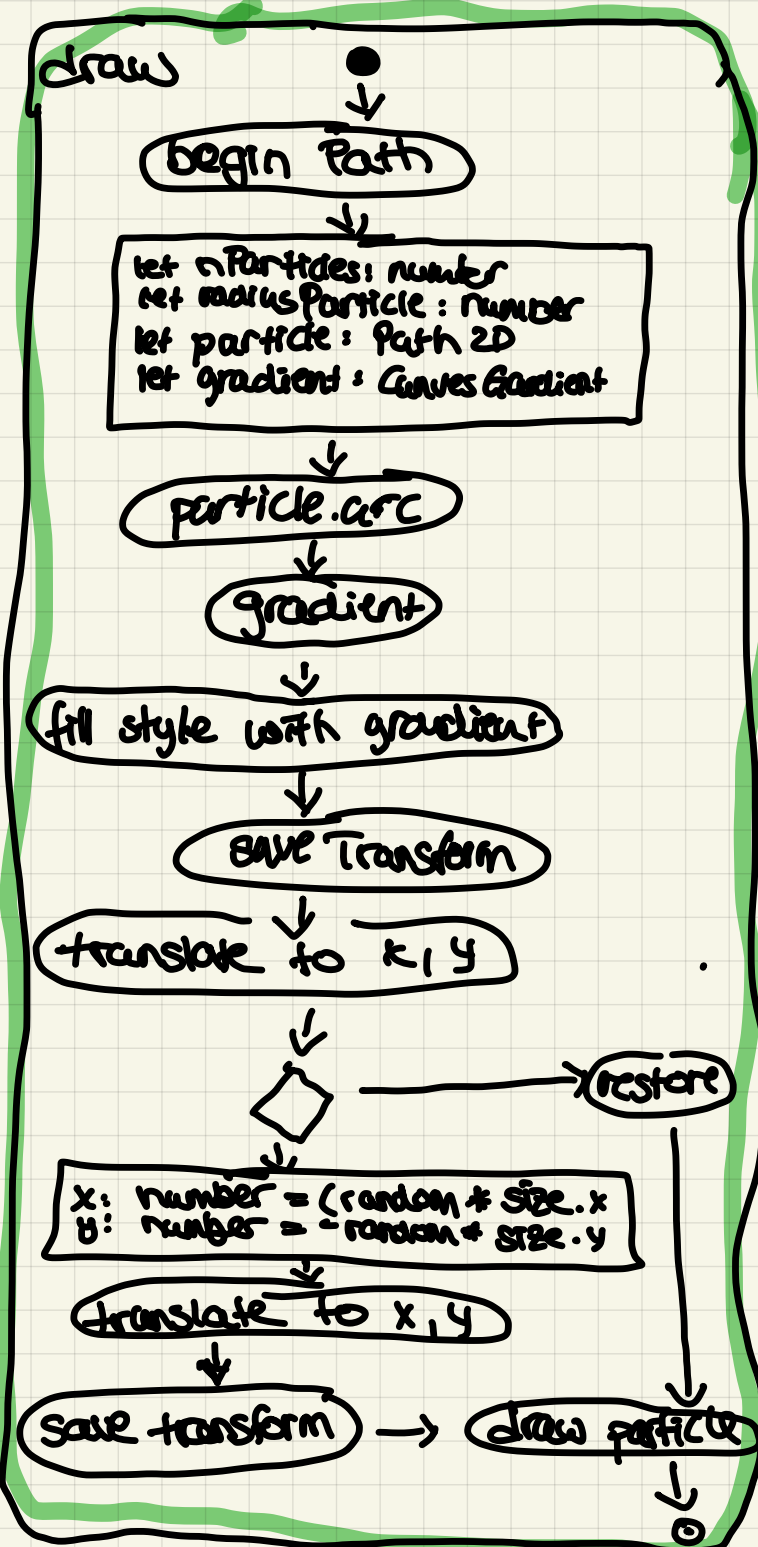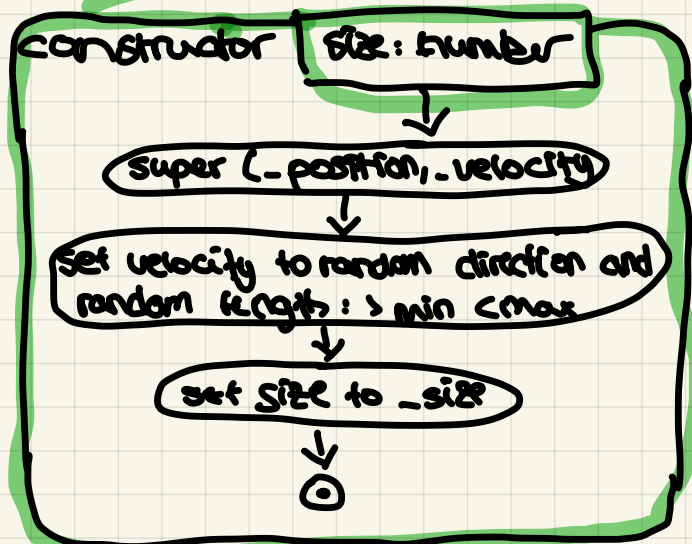(super(_position,_velocity))
↓
(set velocity to random direction & random length >min <max)
↓
◉

leafs

constructor (size: number)

- super (_position, _velocity)
- set velocity to random direction and random length : > min < max
- set size to _size
- ⊙

draw ●

- begin Path
- let nParticles: number
  let radiusParticle : number
  let particle : Path 2D
  let gradient : CanvasGradient
- particle.arc
- gradient
- fill style with gradient
- save Transform
- translate to x, y
- ◇ ──────→ restore
- x: number = (random * size . x
  y: number = - random * size . y
- translate to x, y
- save transform ──→ draw particle
- ⊙

nut

constructor (-x : number
                   -y : number)

- super (-x, -y)
- ⊙

draw ●

- save transform
- translate to position
- draw nut
- restore
- ⊙