FlowerMeadow_L10_2: ClassDiagram

CanvasRenderingContext

Vector

x: number
y: number

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

Moveable

position: Vector
velocity: Vector

constructor: (_position: Vector)
draw(): void
move(_timeslice:number):void

Bee

position: Vector
velocity: Vector

constructor: (_position: Vector)
draw(): void
move(_timeslice:number):void

Cloud

size: Vector

constructor: (_position: Vector)
draw(): void
move(_timeslice:number):void

Background

position: Vector
size:Vector

constructor: (_position: Vector, _size?:Vector)
drawSun(): void
drawMountain(_min:number, _max: number,
_colorLow:string, _colorHigh:string):void
drawtrees():void
drawHills():void

Flower

position: Vector

constructor: (_position: Vector)
drawRoundFlower(): void

L10_2: FlowerMeadow: main - AD

= new stuff

install loadListener

export let canvas:HTMLCAnvasElement
export let crc2: CanvasRenderingContext

let moveables : Moveable[] = []
let background: ImageData

load on window

handleLoad

20sec

animate

handleClouds

let k: number = 3
let i: number = 0

i++

i<k

let xPos: number = (Math.random() * canvas.width)
let yPos: number = ((Math.random() * 170) + 35

let xSize: number = ((Math.random() * 150) + 90)
let ySize: number = ((Math.random() * 30) + 20)

let posCloud: Vector = new Vector(xPos,yPos)
let sizeCloud: Vecor = new Vector(xSize, ySize)

let cloud: Clourd = new Cloud(posCloud, sizeCloud)

cloud.draw()
moveables.push(cloud)

handleLoad

get canvas and rendering context

drawBackground

handleBees

handleClouds

window.setInterval(animate,20)

L10_2: FlowerMeadow: main - AD

drawBackground

save transform

create linear Background
Gradient (sunset colors)

fill rect with gradient

restore transform

create new Vector for sun position,
let sun: new Background(position)

sun.drawSun()

create new Vector for mountain
position,
let mountain1: new
Background(position)

mountain1.drawMountain(70,200, color1, color2)

create new Vector for mountain position,
let mountain2: new
Background(position)

mountain2.drawMountain(50,150, color1, color2)

create new Vector for tree position and tree size
let tree: new Background(position, size)

tree.drawTrees()

create new Vector for hills position
let hills: new Background(position)

hills.drawHills()

handleFlowers()

background = getImageData

L10_2: FlowerMeadow: main - AD

handleFlowers

let nFLowers: number = 10

i++

do same for-loop for
ovalFlowers and
dropFlowers

let i: number = 0 < nFlowers

let x: number = (Math.random()*canvas.width)
let y: number = (320 + Math.random()* canvas.height/4)

let posFlower: Vector = new Vector (x,y)
let flower: Flower = new Flower(posFlower)

flower.drawRoundFlower()

L10_2: FlowerMeadow: main - AD

handleBees

let nBees: number = 10

let i:number = 0 < nBees

let x: number = (Math.random()*canvas.width)
let y: number = (Math.random()* canvas.height)

i++

let position: Vector = new Vector (x,y)
let bee: Bee = new Bee(position)

bee.draw()
moveables.push(bee)

animate

put imgData

move all moveables

draw all moveables

constructor

_position: Vector

position to _position.copy
oder (0,0)

set velocity to (0,0)

move

_timeslice: number

```
let offset: Vector = this.velocity.copy()
        offset.scale(_timeslice)
        this.position.add(offset)
```

draw

L10_2: FlowerMeadow: background - AD

constructor

_position:Vector,
_size?: Vector

↓

position to
_position.copy()
oder (0,0)

↓

size to _size.copy()
oder (0,0)

→ ●

drawHills

_pos:Vector

↓

save transform

↓

translate to
this.position

↓

draw Hill with bezier
curve

↓

create green gradient
--> fill, stroke

↓

restore transform

→ ●

drawSun

● ↓

r1: number = 300
r2: number = 150
gradient = radialGr.

↓

set color step for
gradient and
transparency

↓

save transform

↓

translate to
this.position

↓

draw full circle with r2

↓

restore transform

→ ●

drawMountains

_pos:Vector
_min:number
_max:number
_colorLow: string
_colorHigh: string

↓

stepMin: number = 90
stepMax: number =100
x: number = 0

↓

save transform

↓

translate to _pos

↓

move to 0,0

line to 0, -_max

↓

◇

↓

x += random between stepMin and stepMax

↓

y: number = -_min - random *(_max-_min)

↓

line to x,y

↓

◇ [If x < canvas.width]

↓

line to x,0

↓

close Path

↓

create gradient with
given colors

↓

draw Path

↓

restore transform

→ ●

L10_2: FlowerMeadow: background - AD

drawTrees

let ntrees number = 300

restore transform

save transform

translate to
this.position

let tree: Path2D = new Path2D()

draw tree with -->
tree.moveTo()
tree.lineTo()
tree.closePath()

set shadow-style for tree

let drawn: number = 0

restore transform

drawn< trees

let gradient: create CanvasGradient =
crc2.createLinearGradient(...)

save transform

let x: number = (Math.random() - 0.5) * this.size.x
let y: number = -(Math.ranodm() * this.size.y

gradient.addColorStop(0, "brown")
gradient.addColorStop(0.1, "random green color")

crc2.fillStyle = gradient

translate to(x,y)

crc2.fill(tree)

restore transform

drawn ++

constructor

_position: Vector

super(_position)

this.velocity.add(new Vector(20,0)

move

_timeslice: number

super.move(_timeslice)

let offset: Vector = this.velocity.copy()
offset.x *= _timeslice * 3
offset.y *= _timeslice * 0.2
this.position.add(offset)

[positioncomponent > canvas dimension]

[position component < 0]

add canvas dimension to component

subtract canvas dimension from component

draw

restore transformation

save transform

translate to this.position

nParticle: number = 40
radiusParticle: number = 20

particle = path with full circle with radiusParticle
gradient: Radial with a=0,8 --> a=0.1

save transform

let drawn: number = 0

restore transformation

[drawn particles < nParticles]

save transform

x: number = (random-0,5*_size.x)
y: number = -random*_size.y

translate to x,y

draw particle

restore transformation

drawn ++

constructor

_position:Vector

position to
_position.copy()
oder (0,0)

drawRoundFlower

_x:number,
y:number

translate to _x,_y

stem: create new
Path2D

draw rect

save transform

add fillStyle =green
add shadow

fill rect

restore transform

radius Flower: number = 10;
let roundFlower: newPAth2D

create colorful gradient
add shadows

save transform

translate to _x, _y

fill circle

restore transform

drawOvalFlower

_x:number,
y:number

do same es in
method
drawRoundFlower,
but create an oval
petal for the flower

drawDropFlower

_x:number,
y:number

do same es in
method
drawRoundFlower,
but create an drop-
shaped petal for the
flower

L10_2: FlowerMeadow: Bee - AD

constructor

_position: Vector

↓

super(_position)

↓

set velocity to random
length >min <max

→ ●

move

_timeslice: number

↓

super.move(_timeslice)

↓

let offset: Vector = this.velocity.copy()
offset.x *= _timeslice * 3
offset.y *= _timeslice * 0.2
this.position.add(offset)

↓

◇ [positioncomponent >
canvas dimension]

→ ●

[position
component < 0]

add canvas
dimension to
component

subtract canvas
dimension from
component

◇

↓

●

draw

●

↓

restore transformation

↓

save transform

↓

translate to
this.position

↓

create black and yellow
stripes with gradient

↓

draw an ellipse as
bees body

↓

save transform

↓

restore transformation

→ ●