Bee
+ posX: number;
+ posY: number;
+ velocityX: number;
+ velocityY: number;
+ randomScaleX: number;
+ randomScaleY: number;
+ randomNumber: number;
+ counter: number = 0;
+ constructor(_position: Vector, _velocity: Vector, _randomScale: number
+ draw(): void
+ update(): void

Flower
+ xPos: number;
+ flowerType: number;
+ yRandomMin: number;
+ yRandomMax: number;
+ constructor(_flowerType: number, _xPos: number, _yRandomMin: number, _yRandomMax: number)
+ draw(): void

Cloud
+ posX: number;
+ posY: number;
+ velocityX: number = 0.17;
+ velocityY: number = 0.1;
+ constructor(_position: Vector)
+ draw(): void
+ update(): void

<< interface >> Vector
x: number;
y: number;

Activitydiagram **MAIN**

load

handleLoad

let crc2: CanvasRenderingContext2D;
let goldenCut: number = 0.62;
let bee: Bee[] = [];
let flowers: Flower [] = [];
let clouds: Cloud [] = [];
let imageData: ImageData;
interface Vector for x & y coordinates

install load-listener

createBackground

let horizon: number = crc2.canvas.height * goldenCut;
let posMountains: Vector = {x: 0, y: horizon};

drawBackground();
drawSun({x: crc2.canvas.width * .15, y: crc2.canvas.height * .2})
drawMountains(posMountains, 75, 200, "black", "white")
drawMountains(posMountains, 50, 150, "grey", "white")
drawPine(-30, -50, .6, .9);
drawPine(-70, -90, .5, .7);

handleLoad

let canvas: HTMLCanvasElement =
<HTMLCanvasElement>document.querySelector;

crc2 = canvas.getContext("2d");
canvas.width = window.innerWidth; (smaller)
canvas.height = window.innerHeight; (smaller)
createBackground();
createFlowers();
createBees(12);
createCloud();
imageData = crc2.getImageData(0,0, width, height)
animate()

animate

requestAnimationFrame(animate);
crc2.clearRect(0, 0, crc2.canvas.width, crc2.canvas.height);
crc2.putImageData(imageData, 0, 0);

index ++   [index > bee.length]

[index < bee.length]

bee[index].update();

let index: number = 0;

index ++   [index > clouds.length]

[index < clouds.length]

clouds[index].update();

createCloud

clouds.push(new Cloud({ x: crc2.canvas.width * .10, y:
crc2.canvas.height * .10 }));
clouds.push(new Cloud({ x: crc2.canvas.width * .5, y:
crc2.canvas.height * .05 }));

createFlowers

```
let xPos: number = 0;
```

```
flowers.push(new Flowers ( new Flower(Math.floor(Math.random() * 3) +
1, xPos, 120 + (crc2.canvas.height * goldenCut), crc2.canvas.height));
xPos += 5 + Math.random() * (20 - 5);
```

[xPos < canvas.width]   [xPos >= canvas.width]

_nBee: number

createBees

```
index ++
```

[index > _nBee]

[index < _nBee]

```
let randomScaleY: number = 0.15 + Math.random() * (0.15 - 0.05);
let randomScaleX: number = randomScaleY;
```

```
randomScaleX *= Math.round(Math.random()) ? 1 : -1;
```

```
let randomVelocityX: number = (Math.random() - 0.5) * 5;
let randomVelocityY: number = (Math.random() - 0.5) * 5;
```

```
bee.push(new Bees({x: crc2.canvas.width / 2, y:
crc2.canvas.height * 0.8}, { x: randomVelocityX, y:
randomVelocityY }, {x: randomScaleX, y: randomScaleY}));
```

Activitydiagram **BACKGROUND**

drawBackground

(start)

```
let gradient: CanvasGradient = crc2.createLinearGradient
```

gradient properties

```
crc2.fillStyle = gradient;
crc2.fillRect(0, 0, crc2.canvas.width, crc2.canvas.height);
```

(end)

_position: Vector

drawSun

(start)

```
let r1: number = 20;
let r2: number = 70;
let gradient: CanvasGradient = crc2.createRadialGradient(0, 0, r1,
0, 0, r2);
```

create gradient

```
save transform
translate to _position.x, _position.y
fillStyle with gradient
arc with r2
fill
restore transform
```

(end)

```
_position: Vector
_min: number
_max: number
_colorLow: string
_colorHigh: string
```

drawMountains

(start)

```
let stepMin: number = 50;
let stepMax: number = 150;
let x: number = 0;
```

```
save transform
translate to _position.x, _position.y
beginPath at 0, 0
lineTo 0, _max
```

◇

x += random number between stepMin and stepMax

let y: number = random number between _min and _max

line to x, y

◇

[x < canvas.width]

```
lineTo x, 0
closePath
create gradiant
fillStyle
restore transform
```

(end)

```
                          ┌──────────────────────┐
                          │  _min: number        │
                          │  _max: number        │
                          │  _minSize: number    │
                          │  _maxSize: number    │
                          └──────────────────────┘

drawPine

                                    ●

                    ┌────────────────────────────────────────┐
                    │  let stepMin: number = 30               │
                    │  let stepMax: number = 70               │
                    │  let x: number = 0                      │
                    │  let horizon: number = crc2.canvas.height * goldenCut │
                    └────────────────────────────────────────┘

  ┌──────────────────────────┐
  │ x += random number between│              ◇──────────────────────────►  ◉
  │   stepMin and stepMax     │───────────►                [x >= canvas.width]
  │   restore transform       │
  └──────────────────────────┘
                                       [x < canvas.width]

                    ┌────────────────────────────────────────────────────┐
                    │ let y: number = random number between _min and _max │
                    │ let yTree1: number = -40                            │
                    │ let yTree2: number = -120                           │
                    │ let treeColor: string[] = [colors for the leaves]   │
                    │ let treeSize: number = random number between _minSize and _maxSize │
                    └────────────────────────────────────────────────────┘

                          ┌──────────────────────────┐
                          │ save transform           │
                          │ translate to x, y + horizon + 15 │
                          │ scale Tree with treeSize │
                          │ fillStyle                │
                          │ fillRect (tree trunk)    │
                          └──────────────────────────┘

                                    ◇──────────────────────
                                                          │
                              [z < 3]                    ┌────┐
                                                         │z++ │
                                                         └────┘
              ┌──────────────────────────────┐
              │ beginPath                    │
  [z >= 3]    │ moveTo -50, yTree1           │
──────────────│ lineTo 72, yTree1            │
              │ lineTo 11,yTree2             │───────────
              │ closePath                    │
              │ fillStyle with colors from treeColor [] │
              │ yTree1 += -40                │
              │ yTree2 += -40                │
              └──────────────────────────────┘
```

Activitydiagram **FLOWER**

```
public xPos: number;
public flowerType: number;
public yRandomMax: number;
public yRandomMin: number;
```

constructor

```
_flowerType: number;
_xPos: number;
_yRandomMin: number
_yRandomMax: number
```

```
this.flowerType = _flowerType;
this.xPos = _xPos;
this.yRandomMax = _yRandomMax;
this.yRandomMin = _yRandomMin;
this.draw();
```

draw

```
let y: number = Random number between
this.randomMin and this.randomMax
```

```
crc2.save();
crc2.translate(this.xPos, y);
```

[flowerType == 1]   [flowerType == 2]

draw Lavender   [flowerType == 3]   draw Dandelion

draw Starflower

Activitydiagram **BEE**

```
public posX: number;
public posY: number;
public velocityX: number;
public velocityY: number;
public randomScaleX: number;
public randomScaleY: number;
public randomNumber: number =
(Math.floor(Math.random() * 800) +100);
public counter: number = 0;
```

constructor

```
_position: Vector,
_velocity: Vector,
_randomScale: Vector
```

```
this.posY = _position.y;
this.posX = _position.x;
this.randomScaleX = _randomScale.x;
this.randomScaleX = _randomScale.y;
this.velocityX = _velocity.x;
this.velocityY = _velocity.y;
```

draw

```
crc2.save();
crc2.translate(this.posX, this.posY);
crc2.scale(this.randomScaleX, this.randomScaleY);
Drawing the Bee
crc2.restore();
```

update

[this.posX > crc2.canvas.width || this.posX < 0]

this.velocityX = -this.velocityX

[this.posY > crc2.canvas.height || this.posY
< crc2.canvas.height * 0.65]

this.velocityY = -this.velocityY

[this.counter == this.randomNumber]

```
this.velocityX = -this.velocityX
this.velocityY = -this.velocityY
this.counter = 0;
randomNumber: number =
(Math.floor(Math.random() * 800) +
100);
```

```
this.posX += this.velocityX
this.posY += this.velocityY
counter++
this.draw();
```

Activitydiagram **CLOUDS**

●

```
public posX: number;
public posY: number;
public velocityX: number = 0.17;
public velocityY: number = 0.1;
```

◉

constructor

_position: Vector

```
this.posY = _position.y;
this.posX = _position.x;
```

◉

draw

●

let gradient: CanvasGradient;

```
crc2.save();
crc2.translate(this.xPos, y);
```

```
cloud draws
crc2.restore();
```

◉

update

●

[this.posX > crc2.canvas.width || this.posX < 0]

this.velocityX = -this.velocityX

[this.posY > crc2.canvas.height * 0.20 || this.posY < 10]

this.velocityY = -this.velocityY

```
this.posX += this.velocityX
this.posY += this.velocityY
this.draw();
```

◉