FILE:
index.html
===

```html
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Boids</title>

    <style type="text/css">

        canvas {

            border: 1px solid rgb(0, 0, 0);

            background-color: rgb(35, 35, 38);

        }


        body {

            margin:0px;

            overflow: hidden;

        }

    </style>

</head>

<body>

    <canvas></canvas>

    <script type="module" src="canvas.js"></script>

</body>

</html>
```
===

FILE:
canvas.js

```
===============================================================================
= import { PreyBoid } from './PreyBoid.js';


var canvas = document.querySelector("canvas");

canvas.width = window.innerWidth;

canvas.height = window.innerHeight;

var c = canvas.getContext("2d");


var preyArray = [];

var numPrey = 2;


function load() {

  for (var i = 0; i < numPrey; i++) {

    var obj = new PreyBoid();

    preyArray.push(obj);

  }

}


function update() {

//  document.onkeypress = function (e) {

//    e = e || window.event;

    requestAnimationFrame(update);

//  };

  c.clearRect(0, 0, canvas.width, canvas.height);


  for (var i = 0; i < preyArray.length; i++) {

    preyArray[i].preyTakeStep();

  }

}
```

```
load();

export {c, canvas, preyArray};

update();
=================================================================================
```

FILE:
PreyBoid.js

```
=================================================================================
import {c, canvas, preyArray} from './canvas.js';


class PreyBoid{

    constructor() {

      this.size = 9;

      this.pos = {

        x: Math.random() * (canvas.width - this.size * 2) + this.size,

        y: Math.random() * (canvas.height - this.size * 2) + this.size

      };

      this.velocity = {

        dx: (Math.random() - 0.5) * 20,

        dy: (Math.random() - 0.5) * 20

      };


      this.angle = Math.atan2(this.velocity.dy,this.velocity.dx);

      this.visRadius = 250;

      this.visAngle = Math.PI * 2;

      this.boidSeen = []

    }


    preyDrawBoid() {

      if (this == preyArray[0]) {
        this.preyDrawVisionCone();
      }

      c.fillStyle = "green";
      c.beginPath();
      c.moveTo(this.pos.x + Math.cos(this.angle) * this.size, this.pos.y +
Math.sin(this.angle) * this.size);
      c.lineTo(this.pos.x + Math.cos(this.angle - (Math.PI * 2 / 3)) * this.size,
this.pos.y + Math.sin(this.angle - (Math.PI * 2 / 3)) * this.size);
      c.lineTo(this.pos.x + Math.cos(this.angle + (Math.PI * 2 / 3)) * this.size,
this.pos.y + Math.sin(this.angle + (Math.PI * 2 / 3)) * this.size);
      c.closePath();
      c.fill();
    }

    preyDrawVisionCone() {
        c.fillStyle = "rgba(255, 0, 0, 0.2)";
        c.beginPath();
        c.moveTo(this.pos.x,this.pos.y);
        c.arc(this.pos.x, this.pos.y, this.visRadius, this.angle - (this.visAngle
/ 2), this.angle + (this.visAngle / 2));
        c.lineTo(this.pos.x,this.pos.y);
        c.closePath();
        c.fill();
    }

    preyEdgeHandling() {
      this.pos.x = (this.pos.x + canvas.width) % canvas.width;
      this.pos.y = (this.pos.y + canvas.height) % canvas.height;
    }
```

```
preyAngleNormalize(){
  this.angle = Math.atan2(this.velocity.dy, this.velocity.dx);
  this.angle = (this.angle % (2 * Math.PI) + 2 * Math.PI) % (2 * Math.PI);
}

preyResolve() {
//TODO NEXT STEP, IMPLEMENT LOGIC WHEN BOIDS SEE EACH OTHER
// switch () {
//   case (/*too close*/) {
//     preySeperate();
//   }

//   case () {
//     preyAlign(/*facing the wrong way*/);
//   }

//   case () {
//     preyCohere(/*too far away*/);
//   }
// }
}

preySight() {
  for (var i = 0; i < preyArray.length; i++) {
    var distX = preyArray[i].pos.x - this.pos.x;
    var distY = preyArray[i].pos.y - this.pos.y;
    var distVector = Math.sqrt(distX ** 2 + distY ** 2);
```

```
        /*
        cos(a) = (a0*b1 + b0*b1) / (|a0|*|b1| + |b0|*|b1|)

        a = acos((a0*b1 + b0*b1) / (|a0|*|b1| + |b0|*|b1|))

        a = [this.vecocity.dx, this.vecocity.dy]
        a = [PreyArray[i].vecocity.dx, PreyArray[i].vecocity.dy]

        if a = this, b = that:
          a = acos((this.velocity.dx * preyArray[i].velocity.dy + 
preyArray[i].velocity.dx * this.velocity.dy) / |this.velocity.dx| * 
|preyArray[i].velocity.dy| + |preyArray[i].velocity.dx|)

        */
        if (this == preyArray[0]) {
          if (distVector <= this.visRadius && 
!this.boidSeen.includes(preyArray[i]) && this !== preyArray[i]) {
            this.boidSeen.push(preyArray[i])
          }
        }
      }
    }

    preyTakeStep() {
      this.preyEdgeHandling();
      this.preyAngleNormalize();
      this.preySight();
      this.preyResolve();
      this.pos.x += this.velocity.dx;
      this.pos.y += this.velocity.dy;
      this.preyDrawBoid();
    }
  }

export { PreyBoid };
================================================================================
```