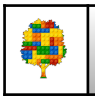
# Section G
# Unit #2
# EvoSteer
# Vehicle

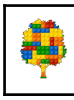


# Content

## Section H Unit #2 Making the vehicles

# Introduction to Unit #2 Making the vehicles

In this unit we make the vehicles move around seeking and devouring both food and poison. We are building the world and all its elements before we start the evolutionary part in unit #4. If you have already completed the physics simulation books 1 and 2 then this is where you would start.



# Sketch H2.1 the seek sketch

The original seek sketch.js starting point

sketch.js

```
let vehicle
let target

function setup()
{
  createCanvas(400, 400)
  vehicle = new Vehicle(100, 100)
}

function draw()
{
  background(220)
  target = createVector(mouseX, mouseY)
  circle(target.x, target.y, 32)
  vehicle.seek(target)
  vehicle.update()
  vehicle.show()
}
```

# Sketch H2.2 and the vehicle class

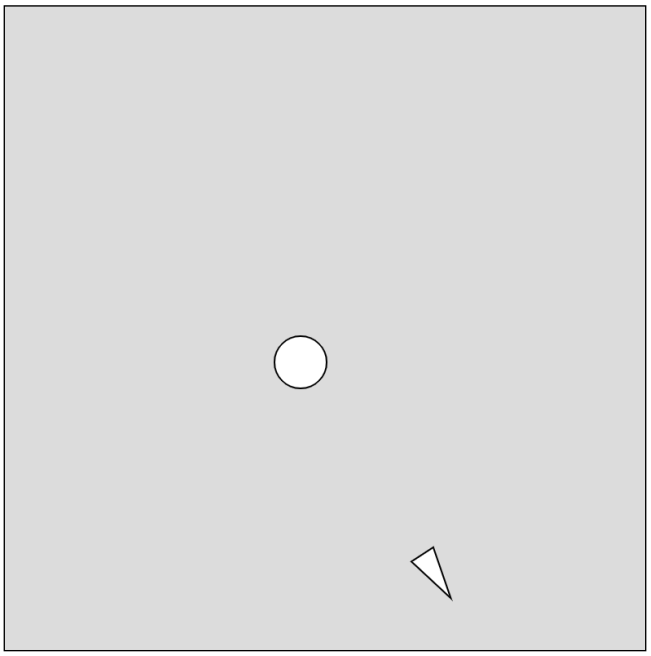
The original seek vehicle.js starting point

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 4
    this.maxForce = 0.01
    this.r = 16
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    this.applyForce(force)
  }

  applyForce(force)
  {
    this.acc.add(force)
```

```
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }
}
```

This is what you should have



## Notes

The vehicle seeks the mouse pointer (circle) still for the moment. We will change this shortly.

# Sketch H2.3 adding food

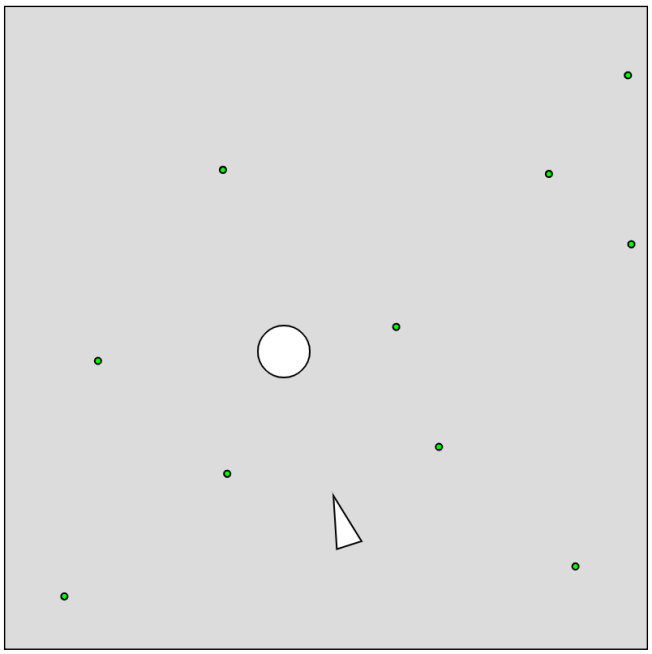
We are going to add some food (as an array) as the target rather than the mouse.

sketch.js

```
let vehicle
let target
let food = []

function setup()
{
  createCanvas(400, 400)
  vehicle = new Vehicle(100, 100)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
}

function draw()
{
  background(220)
  target = createVector(mouseX, mouseY)
  circle(target.x, target.y, 32)
  for (let i = 0; i < food.length; i++)
  {
```

```
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  vehicle.seek(target)
  vehicle.update()
  vehicle.show()
}
```

Green food



## Notes

It will still seek the target which is the mouse pointer but you should have 10 random green circles.

# Sketch H2.4 seek the food

We don't want it to seek the target (mouse) but the food and then eat it. We have replaced it with an eat(food) function.

Delete reference to the target and mouseX, mouseY (I have highlighted them for you)

sketch.js

```
let vehicle
let food = []

function setup()
{
  createCanvas(400, 400)
  vehicle = new Vehicle(100, 100)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
}

function draw()
{
  background(220)
  // target = createVector(mouseX, mouseY)
  // circle(target.x, target.y, 32)
  for (let i = 0; i < food.length; i++)
```

```
  {
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  vehicle.eat(food)
  vehicle.update()
  vehicle.show()
}
```

# Sketch H2.5 eat the food

Now we need to write the eat function in the vehicle class. It is going to eat from a list and eat the piece of food nearest to it. We can use Infinity as our starting longest distance till we find the nearest one.

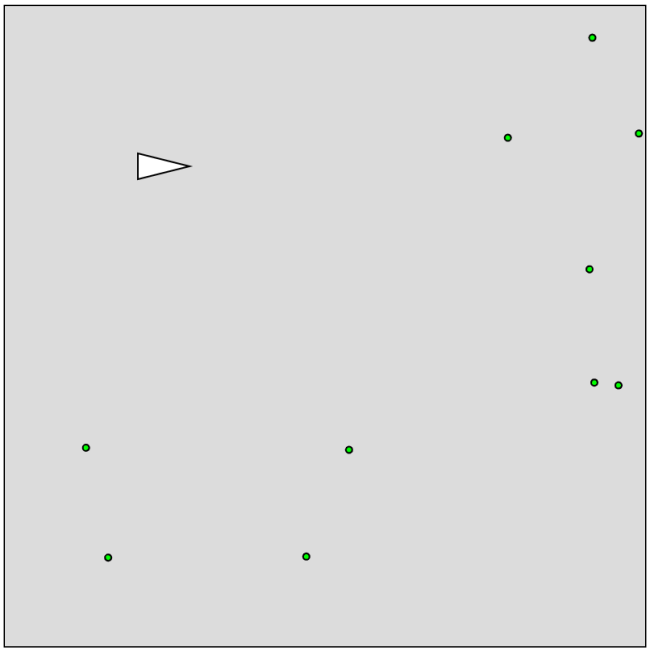
vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.01
    this.r = 16
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    this.applyForce(force)
  }

  applyForce(force)
  {
```

```
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list)
  {
    let record = Infinity
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
```

```
      {
        record = d
      }
    }
  }
}
```

We have our vehicle (static) and food (green dots)



# Sketch H2.6 closest food

We want to find the closest bit of food

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.01
    this.r = 16
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    this.applyForce(force)
  }

  applyForce(force)
  {
    this.acc.add(force)
```

```
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list)
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
```

```
      {
        record = d
        closest = i
      }
    }
  }
}
```

# Sketch H2.7 seek the closest

We want the seek function to find the nearest one

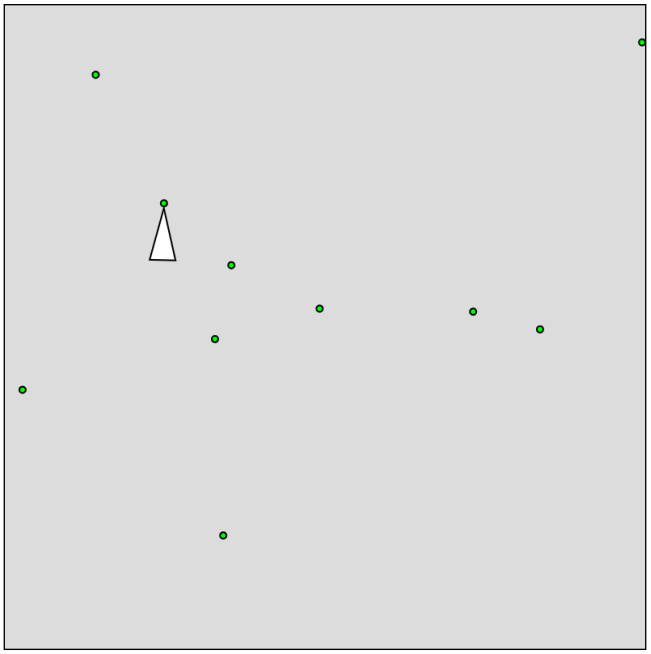
vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.01
    this.r = 16
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    this.applyForce(force)
  }

  applyForce(force)
  {
    this.acc.add(force)
```

```
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list)
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
```

```
      {
        record = d
        closest = i
      }
    }
    this.seek(list[closest])
  }
}
```

Turns and moves to the nearest food



## Notes

You should now see the vehicle move (seek) towards the nearest bit of food (green dot)

## Sketch H2.8 eating the food

Now let us eat that food if it is within 5 pixels of it. The splice function removes an element from the array. Also to speed things up a little I have changed the maxSpeed and maxForce

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 16
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    this.applyForce(force)
  }

  applyForce(force)
  {
```

```
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list)
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
```

```
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
      list.splice(closest, 1)
    }
    this.seek(list[closest])
  }
}
```

## Notes

You will get an error message at some point as it eats one of the food dots

# Sketch H2.9 adding poison

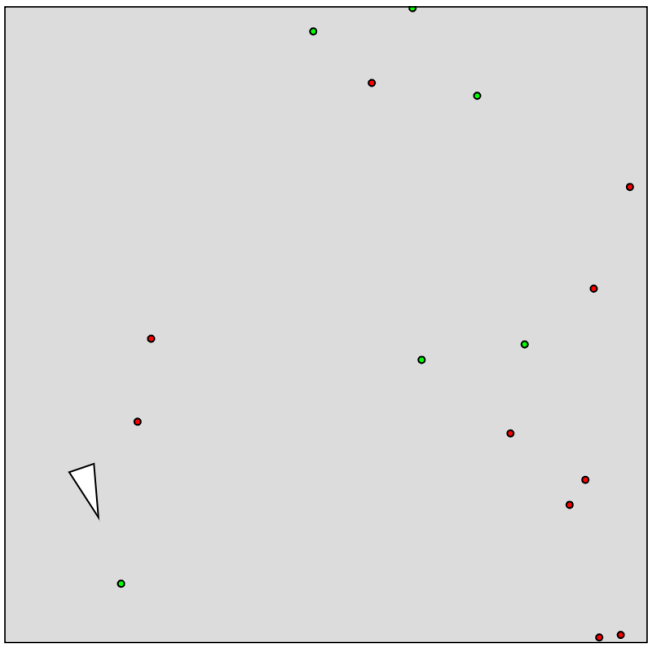
Now let us add the poison as well in sketch.js and give it a red colour dot.

sketch.js

```
let vehicle
let food = []
let poison = []

function setup()
{
  createCanvas(400, 400)
  vehicle = new Vehicle(100, 100)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    poison.push(createVector(x, y))
  }
}

function draw()
{
```

```
  background(220)
  for (let i = 0; i < food.length; i++)
  {
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  for (let i = 0; i < poison.length; i++)
  {
    fill(255, 0, 0)
    circle(poison[i].x, poison[i].y, 4)
  }
  vehicle.eat(food)
  vehicle.eat(poison)
  vehicle.update()
  vehicle.show()
}
```

Food and poison, it is all the same!



## Notes

You will notice that it seems to get stuck, this is because it is attracted to the food and the poison at the same time. It will eat both but is also attracted to both.

## Sketch H2.10 error message

To help with the error message we get this is where it might try to seek a piece of food or poison that it has deleted. We add the following in vehicle.js temporally, it will be improved later but for now this is an error check of sorts. You shouldn't get any error messages now (hopefully)

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 16
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    this.applyForce(force)
  }

  applyForce(force)
```

```
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list)
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
```

```
      let d = this.pos.dist(list[i])
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
      list.splice(closest, 1)
    }
    else if (closest != null)
    {
      this.seek(list[closest])
    }
  }
}
```

## Sketch H2.11 the dna

We are going to give the vehicle some DNA as an array [] so that it has some attributes or behaviour. This is where you can be creative at a later stage and have it behave differently. The key elements of this behaviour/ attributes are its attraction/repulsion with food/poison and its ability to see food/poison. We create a separate function called behaviour to apply these attributes. steerG is steering force for good (food) and steerB is steering force bad (poison)

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 16
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
```

```
    force.limit(this.maxForce)
    this.applyForce(force)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list)
```

```
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
      list.splice(closest, 1)
    }
    else if (closest != null)
    {
      this.seek(list[closest])
    }
  }

  behaviours(good, bad)
  {
    let steerG = this.eat(good)
    let steerB = this.eat(bad)
    steerG.mult(this.dna[0])
    steerB.mult(this.dna[1])
    this.applyForce(steerG)
```

```
    this.applyForce(steerB)
  }
}
```

## Notes

We have created a framework for steering related to the poison and the food, nothing will happen yet as they are not connected with actually eating the food/poison which are the two arguments (good, bad)

## Sketch H2.12 include its behaviour

In sketch.js we add in the vehicle behaviours function (please note that you will instantly get an error message).

sketch.js

```
let vehicle
let food = []
let poison = []

function setup()
{
  createCanvas(400, 400)
  vehicle = new Vehicle(100, 100)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    poison.push(createVector(x, y))
  }
}

function draw()
```

```
{
  background(220)
  for (let i = 0; i < food.length; i++)
  {
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  for (let i = 0; i < poison.length; i++)
  {
    fill(255, 0, 0)
    circle(poison[i].x, poison[i].y, 4)
  }
  vehicle.eat(food)
  vehicle.eat(poison)
  vehicle.update()
  vehicle.show()
  vehicle.behaviours(food, poison)
}
```

# Sketch H2.13 steering force

We go back to the eat function in vehicle.js. first remove this.applyForce(force) in the seek(target) function. We want to extract the steering force from the DNA and apply it to the steering. Each time it runs it will be either be attracted more to the food or the poison randomly. You will get some very different results (behaviour), just keep rerunning the sketch.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 16
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
```

```
    force.limit(this.maxForce)
    return force
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list)
```

```
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
      list.splice(closest, 1)
    }
    else if (closest != null)
    {
      return this.seek(list[closest])
    }
    return createVector(0, 0)
  }

  behaviours(good, bad)
  {
    let steerG = this.eat(good)
    let steerB = this.eat(bad)
    steerG.mult(this.dna[0])
    steerB.mult(this.dna[1])
```

```
    this.applyForce(steerG)
    this.applyForce(steerB)
  }
}
```

## Notes

The vehicle will now have either a strong, weak attraction or a repulsion to either food or poison demanding on the random values of the DNA. You can test this out by changing the this.dna(-2, 2) and giving it fixed values such as

this.dna[0] = 2
this.dna[1] = 0

The vehicle will now only eat the food and ignore the poison.

## Challenges

1. trying different values for the food and the poison e.g.

this.dna[0] = 2
this.dna[1] = -2

2. Why did you think it behaved like it did?
Try this instead

this.dna[0] = 2
this.dna[1] = -1

3. increase the amount of food and poison

# Sketch H2.14 many vehicles

We are going to have many vehicles because we want to evolve them using a genetic algorithm. This is where the dna of one is companioned with the dna of another.

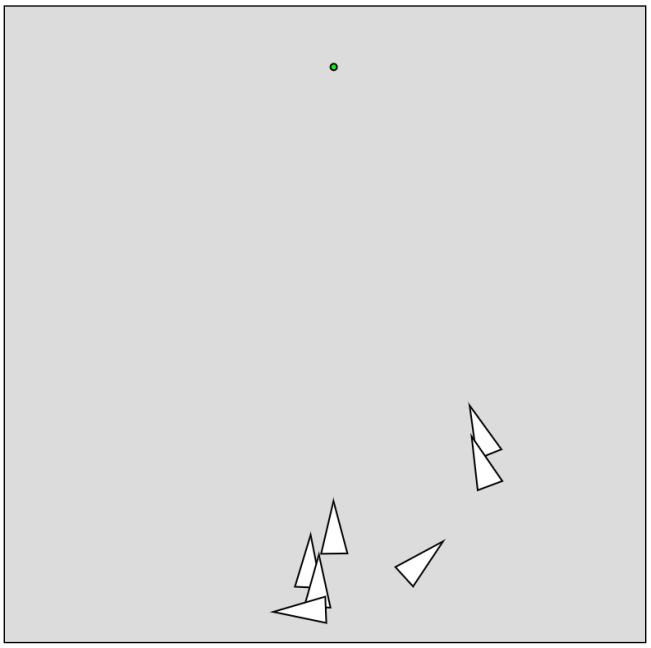
We can also remove the following;
vehicle.eat(food)
vehicle.eat(poison)

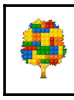
sketch.js

```
let vehicles = []
let food = []
let poison = []


function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    vehicles[i] = new Vehicle(x, y)
  }
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < 10; i++)
```

```
  {
    let x = random(width)
    let y = random(height)
    poison.push(createVector(x, y))
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < food.length; i++)
  {
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  for (let i = 0; i < poison.length; i++)
  {
    fill(255, 0, 0)
    circle(poison[i].x, poison[i].y, 4)
  }
  for (let i = 0; i < vehicles.length; i++)
  {
    vehicles[i].update()
    vehicles[i].show()
    vehicles[i].behaviours(food, poison)
  }
}
```

We have 10 vehicles looking for 10 green or red dots





## Sketch H2.15 health

What we now want to do is give the vehicle some health so that its health increases when it eats the food (0.1) and decreases when it eats the poison (-0.5). Health is between 0 and 1. Their health goes down by 0.01 as it moves around. Also make the vehicle smaller.

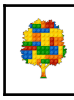
vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 8
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
    this.health = 1
  }


  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
```

```
    return force
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
    this.health -= 0.01
  }

  show()
  {
    fill(255)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list, nutrition)
```

```
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
      list.splice(closest, 1)
      this.health += nutrition
    }
    else if (closest != null)
    {
      return this.seek(list[closest])
    }
    return createVector(0, 0)
  }

  behaviours(good, bad)
  {
    let steerG = this.eat(good, 0.1)
    let steerB = this.eat(bad, -0.5)
    steerG.mult(this.dna[0])
```

```
    steerB.mult(this.dna[1])
    this.applyForce(steerG)
    this.applyForce(steerB)
  }
}
```



## Sketch H2.16 health visualisation

We want to be able to visualise the health of the vehicles by setting the colour of vehicle according to its health, green is healthy, red is ill. we use colour lerp function. This estimates the colour between two colours according to a property.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 8
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
    this.health = 1
  }


  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
```

```
    return force
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
    this.health -= 0.01
  }

  show()
  {
    let gr = color(0, 255, 0)
    let rd = color(255, 0, 0)
    let col = lerpColor(rd, gr, this.health)
    fill(col)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
```

```
  }

  eat(list, nutrition)
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
      list.splice(closest, 1)
      this.health += nutrition
    }
    else if (closest != null)
    {
      return this.seek(list[closest])
    }
    return createVector(0, 0)
  }

  behaviours(good, bad)
  {
```

```
    let steerG = this.eat(good, 0.1)
    let steerB = this.eat(bad, -0.5)
    steerG.mult(this.dna[0])
    steerB.mult(this.dna[1])
    this.applyForce(steerG)
    this.applyForce(steerB)
  }
}
```

## Notes

Notice that they start off green but go red quite quickly. This is because we gave the poison a higher value than the food.

## Challenges

Try different values for the good and bad food in the behaviours function. Try a different value for this.health -= ? In the update function

## Sketch H2.17 it dies

Now we need to delete the vehicle if its health reaches zero. At the moment it just stays very red. We create a boolean expression inside a dead() function, it returns true if the health is less than zero, false if not.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 8
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
    this.health = 1
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    return force
```

```
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
    this.health -= 0.005
  }

  show()
  {
    let gr = color(0, 255, 0)
    let rd = color(255, 0, 0)
    let col = lerpColor(rd, gr, this.health)
    fill(col)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }
```

```
eat(list, nutrition)
{
  let record = Infinity
  let closest = null
  for (let i = 0; i < list.length; i++)
  {
    let d = this.pos.dist(list[i])
    if (d < record)
    {
      record = d
      closest = i
    }
  }
  if (record < 5)
  {
    list.splice(closest, 1)
    this.health += nutrition
  }
  else if (closest != null)
  {
    return this.seek(list[closest])
  }
  return createVector(0, 0)
}

behaviours(good, bad)
{
  let steerG = this.eat(good, 0.1)
```

```
    let steerB = this.eat(bad, -1)
    steerG.mult(this.dna[0])
    steerB.mult(this.dna[1])
    this.applyForce(steerG)
    this.applyForce(steerB)
  }


  dead()
  {
    return (this.health < 0)
  }
}
```

# Sketch H2.18 iterate backwards

In sketch.js we can delete them with splice but we have a problem because at the moment we cycle through the array of vehicles starting from index zero. It is much better to start at the end and iterate backwards.

sketch.js

```
let vehicles = []
let food = []
let poison = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    vehicles[i] = new Vehicle(x, y)
  }
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
```

```
    poison.push(createVector(x, y))
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < food.length; i++)
  {
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  for (let i = 0; i < poison.length; i++)
  {
    fill(255, 0, 0)
    circle(poison[i].x, poison[i].y, 4)
  }
  for (let i = vehicles.length - 1; i > 0; i--)
  {
    vehicles[i].update()
    vehicles[i].show()
    vehicles[i].behaviours(food, poison)
    if (vehicles[i].dead())
    {
      vehicles.splice(i, 1)
    }
  }
}
```

## Notes

You will notice that they die quite quickly as there turn rapidly red.

# Sketch H2.19 adding food

We are going to add more food (5% of the time) using a random function

sketch.js

```
let vehicles = []
let food = []
let poison = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    vehicles[i] = new Vehicle(x, y)
  }
  for (let i = 0; i < 100; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    poison.push(createVector(x, y))
```

```
  }
}

function draw()
{
  background(220)
  if (random(1) < 0.05)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < food.length; i++)
  {
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  for (let i = 0; i < poison.length; i++)
  {
    fill(255, 0, 0)
    circle(poison[i].x, poison[i].y, 4)
  }
  for (let i = vehicles.length - 1; i > 0; i--)
  {
    vehicles[i].update()
    vehicles[i].show()
    vehicles[i].behaviours(food, poison)
    if (vehicles[i].dead())
    {
```

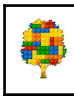
```
      vehicles.splice(i, 1)
    }
  }
}
```

## Notes

I managed to get one to live forever. Keep trying (refreshing if they all die) until you get one that clearing survives.

## Challenge

Change the parameters for how nutritious the food is, also slow down how fast they loose health when on the move etc.



## Sketch H2.20 fixing the edges

At the moment they can wander off the edges of the window but the following function will steer them inwards if they go beyond d pixels from the edge where d will be set at 25 initially. We will do this in another function called boundaries. There is quite a lot of code and it is up to you if you want to add it or not.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 8
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
    this.health = 1
  }


  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
```

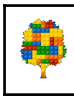
```
    force.limit(this.maxForce)
    return force
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
    this.health -= 0.005
  }

  show()
  {
    let gr = color(0, 255, 0)
    let rd = color(255, 0, 0)
    let col = lerpColor(rd, gr, this.health)
    fill(col)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)
```

```
  pop()
}

eat(list, nutrition)
{
  let record = Infinity
  let closest = null
  for (let i = 0; i < list.length; i++)
  {
    let d = this.pos.dist(list[i])
    if (d < record)
    {
      record = d
      closest = i
    }
  }
  if (record < 5)
  {
    list.splice(closest, 1)
    this.health += nutrition
  }
  else if (closest != null)
  {
    return this.seek(list[closest])
  }
  return createVector(0, 0)
}

behaviours(good, bad)
```

```
  {
    let steerG = this.eat(good, 0.1)
    let steerB = this.eat(bad, -1)
    steerG.mult(this.dna[0])
    steerB.mult(this.dna[1])
    this.applyForce(steerG)
    this.applyForce(steerB)
  }

  dead()
  {
    return (this.health < 0)
  }
```

```
  boundaries()
  {
    let boundary = 25
    let desired = null
    if (this.pos.x < boundary)
    {
      desired = createVector(this.maxSpeed, this.vel.y)
    }
    else if (this.pos.x > width - boundary)
    {
      desired = createVector(-this.maxSpeed, this.vel.y)
    }
    if (this.pos.y < boundary)
    {
      desired = createVector(this.vel.x, this.maxSpeed)
```

```
    }
    else if (this.pos.y > height - boundary)
    {
      desired = createVector(this.vel.x, -this.maxSpeed)
    }
    if (desired !== null)
    {
      desired.normalize()
      desired.mult(this.maxSpeed)
      let steer = p5.Vector.sub(desired, this.vel)
      steer.limit(this.maxForce)
      this.applyForce(steer)
    }
  }
}
```



# Sketch H2.21 adding the boundary

Add in the boundaries() function into sketch.js

sketch.js

```
let vehicles = []
let food = []
let poison = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    vehicles[i] = new Vehicle(x, y)
  }
  for (let i = 0; i < 100; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    poison.push(createVector(x, y))
```

```
  }
}


function draw()
{
  background(220)
  if (random(1) < 0.05)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < food.length; i++)
  {
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  for (let i = 0; i < poison.length; i++)
  {
    fill(255, 0, 0)
    circle(poison[i].x, poison[i].y, 4)
  }
  for (let i = vehicles.length - 1; i > 0; i--)
  {
    vehicles[i].boundaries()
    vehicles[i].update()
    vehicles[i].show()
    vehicles[i].behaviours(food, poison)
    if (vehicles[i].dead())
```

```
    {
      vehicles.splice(i, 1)
    }
  }
}
```

# Sketch H2.22 perception

We add another two properties to the vehicle how far away can it see another pieces of food or poison. Perception radius of random(100). This means that its perception distance (how far it can see the food and the poison) will be different for each vehicle. We will draw this perception radius for each vehicle.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 8
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
    this.dna[2] = random(100)
    this.dna[3] = random(100)
    this.health = 1
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
```

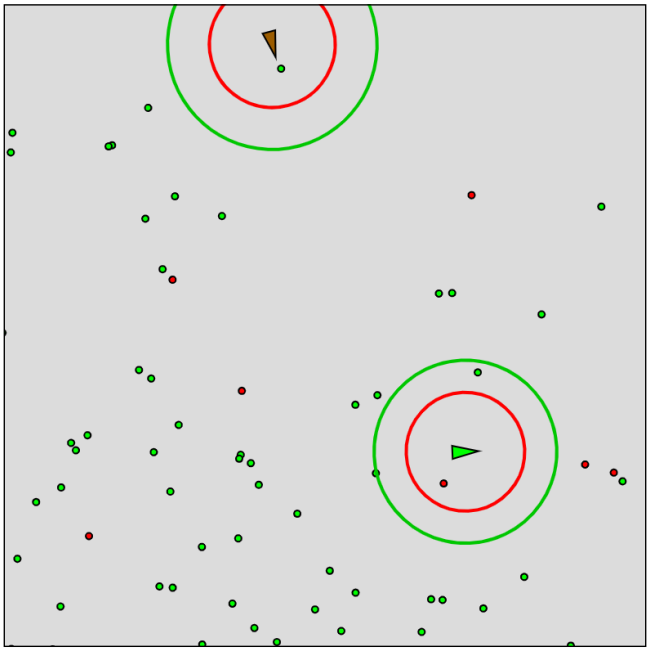
```
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    return force
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
    this.health -= 0.005
  }

  show()
  {
    let gr = color(0, 255, 0)
    let rd = color(255, 0, 0)
    let col = lerpColor(rd, gr, this.health)
    fill(col)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
```

```
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    noFill()
    strokeWeight(2)
    stroke(0, 200, 0)
    circle(0, 0, this.dna[2] * 2)
    stroke(255, 0, 0)
    circle(0, 0, this.dna[3] * 2)
    pop()
  }

  eat(list, nutrition)
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
      list.splice(closest, 1)
      this.health += nutrition
    }
```

```
    else if (closest != null)
    {
      return this.seek(list[closest])
    }
    return createVector(0, 0)
  }

  behaviours(good, bad)
  {
    let steerG = this.eat(good, 0.1)
    let steerB = this.eat(bad, -1)
    steerG.mult(this.dna[0])
    steerB.mult(this.dna[1])
    this.applyForce(steerG)
    this.applyForce(steerB)
  }

  dead()
  {
    return (this.health < 0)
  }

  boundaries()
  {
    let boundary = 25
    let desired = null
    if (this.pos.x < boundary)
    {
      desired = createVector(this.maxSpeed, this.vel.y)
```

```
    }
    else if (this.pos.x > width - boundary)
    {
      desired = createVector(-this.maxSpeed, this.vel.y)
    }
    if (this.pos.y < boundary)
    {
      desired = createVector(this.vel.x, this.maxSpeed)
    }
    else if (this.pos.y > height - boundary)
    {
      desired = createVector(this.vel.x, -this.maxSpeed)
    }
    if (desired !== null)
    {
      desired.normalize()
      desired.mult(this.maxSpeed)
      let steer = p5.Vector.sub(desired, this.vel)
      steer.limit(this.maxForce)
      this.applyForce(steer)
    }
  }
}
```

You get something like this with a red circle and green circle showing the random perception



## Notes

The circles are only illustrative at the moment and we need to add that property.

# Sketch H2.23 a third argument

We add another argument to the eat() function but we need to add it to the steering because if it can see it it will steer towards the food and away from the poison (or not). If it cannot see neither the food nor the poison it doesn't affect the steering.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 8
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
    this.dna[2] = random(100)
    this.dna[3] = random(100)
    this.health = 1
  }

  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
```

```
    force.sub(this.vel)
    force.limit(this.maxForce)
    return force
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
    this.health -= 0.005
  }

  show()
  {
    let gr = color(0, 255, 0)
    let rd = color(255, 0, 0)
    let col = lerpColor(rd, gr, this.health)
    fill(col)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
```

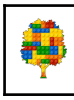
```
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    noFill()
    strokeWeight(2)
    stroke(0, 200, 0)
    circle(0, 0, this.dna[2] * 2)
    stroke(255, 0, 0)
    circle(0, 0, this.dna[3] * 2)
    pop()
  }

  eat(list, nutrition, perception)
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
      list.splice(closest, 1)
      this.health += nutrition
    }
```

```
    else if (closest != null)
    {
      return this.seek(list[closest])
    }
    return createVector(0, 0)
  }

  behaviours(good, bad)
  {
    let steerG = this.eat(good, 0.1, this.dna[2])
    let steerB = this.eat(bad, -1, this.dna[3])
    steerG.mult(this.dna[0])
    steerB.mult(this.dna[1])
    this.applyForce(steerG)
    this.applyForce(steerB)
  }

  dead()
  {
    return (this.health < 0)
  }

  boundaries()
  {
    let boundary = 25
    let desired = null
    if (this.pos.x < boundary)
    {
      desired = createVector(this.maxSpeed, this.vel.y)
```

```
    }
    else if (this.pos.x > width - boundary)
    {
      desired = createVector(-this.maxSpeed, this.vel.y)
    }
    if (this.pos.y < boundary)
    {
      desired = createVector(this.vel.x, this.maxSpeed)
    }
    else if (this.pos.y > height - boundary)
    {
      desired = createVector(this.vel.x, -this.maxSpeed)
    }
    if (desired !== null)
    {
      desired.normalize()
      desired.mult(this.maxSpeed)
      let steer = p5.Vector.sub(desired, this.vel)
      steer.limit(this.maxForce)
      this.applyForce(steer)
    }
  }
}
```

## Notes

Drawing the circles is just a visual to see what is going on. The perception distance is only one property, remember that the other random property is its attraction to either the food or the poison.



# Sketch H2.24 attraction visualisation

We are going to add the visuals for the strength of attraction to food and poison with lines, where the green line is the strength of attraction to the food and the red is an attraction to the poison. After this I will remove both the lines and the circles for the rest of this book.

I suggest you comment them out as you will be adding them back in at the end of the book with a toggle button.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 8
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
    this.dna[2] = random(100)
    this.dna[3] = random(100)
    this.health = 1
  }


  seek(target)
  {
```

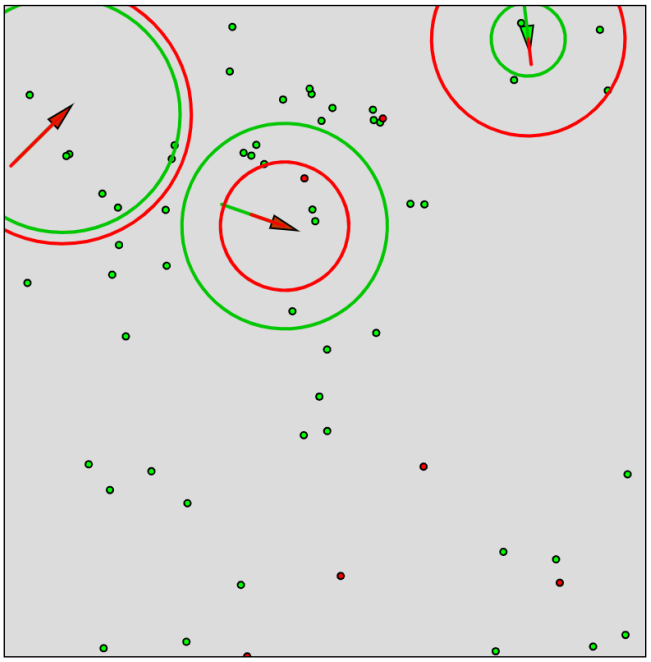
```
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    return force
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
    this.health -= 0.005
  }

  show()
  {
    let gr = color(0, 255, 0)
    let rd = color(255, 0, 0)
    let col = lerpColor(rd, gr, this.health)
    fill(col)
    push()
    translate(this.pos.x, this.pos.y)
```

```
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    noFill()
    strokeWeight(2)
    stroke(0, 200, 0)
    line(0, 0, -this.dna[0] * 25, 0)
    circle(0, 0, this.dna[2] * 2)
    stroke(255, 0, 0)
    line(0, 0, this.dna[1] * 25, 0)
    circle(0, 0, this.dna[3] * 2)
    pop()
  }

  eat(list, nutrition, perception)
  {
    let record = Infinity
    let closest = null
    for (let i = 0; i < list.length; i++)
    {
      let d = this.pos.dist(list[i])
      if (d < record)
      {
        record = d
        closest = i
      }
    }
    if (record < 5)
    {
```

```
    list.splice(closest, 1)
    this.health += nutrition
  }
  else if (closest != null)
  {
    return this.seek(list[closest])
  }
  return createVector(0, 0)
}

behaviours(good, bad)
{
  let steerG = this.eat(good, 0.1, this.dna[2])
  let steerB = this.eat(bad, -1, this.dna[3])
  steerG.mult(this.dna[0])
  steerB.mult(this.dna[1])
  this.applyForce(steerG)
  this.applyForce(steerB)
}

dead()
{
  return (this.health < 0)
}

boundaries()
{
  let boundary = 25
  let desired = null
```

```
    if (this.pos.x < boundary)
    {
      desired = createVector(this.maxSpeed, this.vel.y)
    }
    else if (this.pos.x > width - boundary)
    {
      desired = createVector(-this.maxSpeed, this.vel.y)
    }
    if (this.pos.y < boundary)
    {
      desired = createVector(this.vel.x, this.maxSpeed)
    }
    else if (this.pos.y > height - boundary)
    {
      desired = createVector(this.vel.x, -this.maxSpeed)
    }
    if (desired !== null)
    {
      desired.normalize()
      desired.mult(this.maxSpeed)
      let steer = p5.Vector.sub(desired, this.vel)
      steer.limit(this.maxForce)
      this.applyForce(steer)
    }
  }
}
```

Showing the strength of the attraction as
a line with their perception as circles



# Sketch H2.25 adding poison

We add food randomly every frame (5%) let's add poison every 1% of the time before we move onto the genetic algorithm to evolve an optimum vehicle. We do this in sketch.js

sketch.js

```
let vehicles = []
let food = []
let poison = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    vehicles[i] = new Vehicle(x, y)
  }
  for (let i = 0; i < 100; i++)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
```

```
    poison.push(createVector(x, y))
  }
}

function draw()
{
  background(220)
  if (random(1) < 0.05)
  {
    let x = random(width)
    let y = random(height)
    food.push(createVector(x, y))
  }
  if (random(1) < 0.01)
  {
    let x = random(width)
    let y = random(height)
    poison.push(createVector(x, y))
  }
  for (let i = 0; i < food.length; i++)
  {
    fill(0, 255, 0)
    circle(food[i].x, food[i].y, 4)
  }
  for (let i = 0; i < poison.length; i++)
  {
    fill(255, 0, 0)
    circle(poison[i].x, poison[i].y, 4)
  }
```

```
  for (let i = vehicles.length - 1; i > 0; i--)
  {
    vehicles[i].boundaries()
    vehicles[i].update()
    vehicles[i].show()
    vehicles[i].behaviours(food, poison)
    if (vehicles[i].dead())
    {
      vehicles.splice(i, 1)
    }
  }
}
```

## Sketch H2.26 a tweak

A bit more tweaking because it might happen that it comes across a piece of poison and doesn't eat it if its perception radius is 0 or very tiny. The maxSpeed is 5 that means it moves at 5 pixels per frame and high jump over the poison or food. So what we want is for the vehicle to eat the food/poison if it comes across it regardless.

We will also need to delete food backwards from the array as we did with the dead() function. There is a bit of refactoring or rearranging just to make it work better before we do the evolutionary genetic part.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, -2)
    this.acc = createVector(0, 0)
    this.maxSpeed = 5
    this.maxForce = 0.5
    this.r = 8
    this.dna = []
    this.dna[0] = random(-2, 2)
    this.dna[1] = random(-2, 2)
    this.dna[2] = random(100)
    this.dna[3] = random(100)
    this.health = 1
  }
```

```
  seek(target)
  {
    let force = p5.Vector.sub(target, this.pos)
    force.setMag(this.maxSpeed)
    force.sub(this.vel)
    force.limit(this.maxForce)
    return force
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  update()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
    this.health -= 0.005
  }

  show()
  {
    let gr = color(0, 255, 0)
    let rd = color(255, 0, 0)
    let col = lerpColor(rd, gr, this.health)
    fill(col)
```

```
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    pop()
  }

  eat(list, nutrition, perception)
  {
    let record = Infinity
    let closest = null
    for (let i = list.length - 1; i >= 0; i--)
    {
      let d = this.pos.dist(list[i])
      if (d < this.maxSpeed)
      {
        list.splice(i, 1)
        this.health += nutrition
      }
      else
      {
        if (d < record && d < perception)
        {
          record = d
          closest = list[i]
        }
      }
    }
```

```
    if (closest != null)
    {
      return this.seek(closest)
    }
    return createVector(0, 0)
  }

  behaviours(good, bad)
  {
    let steerG = this.eat(good, 0.1, this.dna[2])
    let steerB = this.eat(bad, -1, this.dna[3])
    steerG.mult(this.dna[0])
    steerB.mult(this.dna[1])
    this.applyForce(steerG)
    this.applyForce(steerB)
  }

  dead()
  {
    return (this.health < 0)
  }

  boundaries()
  {
    let boundary = 25
    let desired = null
    if (this.pos.x < boundary)
    {
      desired = createVector(this.maxSpeed, this.vel.y)
```

```
    }
    else if (this.pos.x > width - boundary)
    {
      desired = createVector(-this.maxSpeed, this.vel.y)
    }
    if (this.pos.y < boundary)
    {
      desired = createVector(this.vel.x, this.maxSpeed)
    }
    else if (this.pos.y > height - boundary)
    {
      desired = createVector(this.vel.x, -this.maxSpeed)
    }
    if (desired !== null)
    {
      desired.normalize()
      desired.mult(this.maxSpeed)
      let steer = p5.Vector.sub(desired, this.vel)
      steer.limit(this.maxForce)
      this.applyForce(steer)
    }
  }
}
```

Removing all the circles and lines from the show function in vehicle.js

vehicle.js

```
  show()
  {
    let gr = color(0, 255, 0)
    let rd = color(255, 0, 0)
    let col = lerpColor(rd, gr, this.health)
    fill(col)
    push()
    translate(this.pos.x, this.pos.y)
    rotate(this.vel.heading())
    triangle(-this.r, -this.r/2, -this.r, this.r/2, this.r, 0)

    // noFill()
    // strokeWeight(2)
    // stroke(0, 200, 0)
    // line(0, 0, -this.dna[0] * 25, 0)
    // circle(0, 0, this.dna[2] * 2)
    // stroke(255, 0, 0)
    // line(0, 0, this.dna[1] * 25, 0)
    // circle(0, 0, this.dna[3] * 2)
    pop()
  }
```