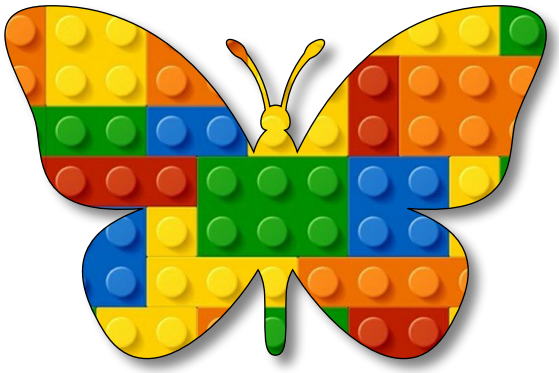
Artificial Intelligence Module C Unit #1 p5.js code snippets 4



Content

Module C Unit #2 code snippets part 4

Sketch C1.27	applying a force
Sketch C1.28	seeking the target
Sketch C1.29	seek, move and show
Sketch C1.30	reset acceleration
Sketch C1.31	steering
Sketch C1.32	rotate to the heading
Sketch C1.33	getting it working
Sketch C1.34	the maximum force
Sketch C1.35	the force is now steering

Introduction to coding snippets part 4

This is a more challenging coding snippets than the first three versions. We are going to cover some concepts which aren't neat little snippets but, rather, large chunks of code. The first section is about perlin noise which is a form of random which is less erratic, it is still random but it is random in relation to the previous value, whereas using the `random()` function just generates a number with no regard to the previous number generated.

The second section looks at a number of things in the context of a moving vehicle. The main concept here is the introduction to classes. Using classes is a way of creating a template that we can use over and over again with variations. I will introduce an example where we will add files and code how a vehicle would seek a moving target.

Introduction to perlin noise

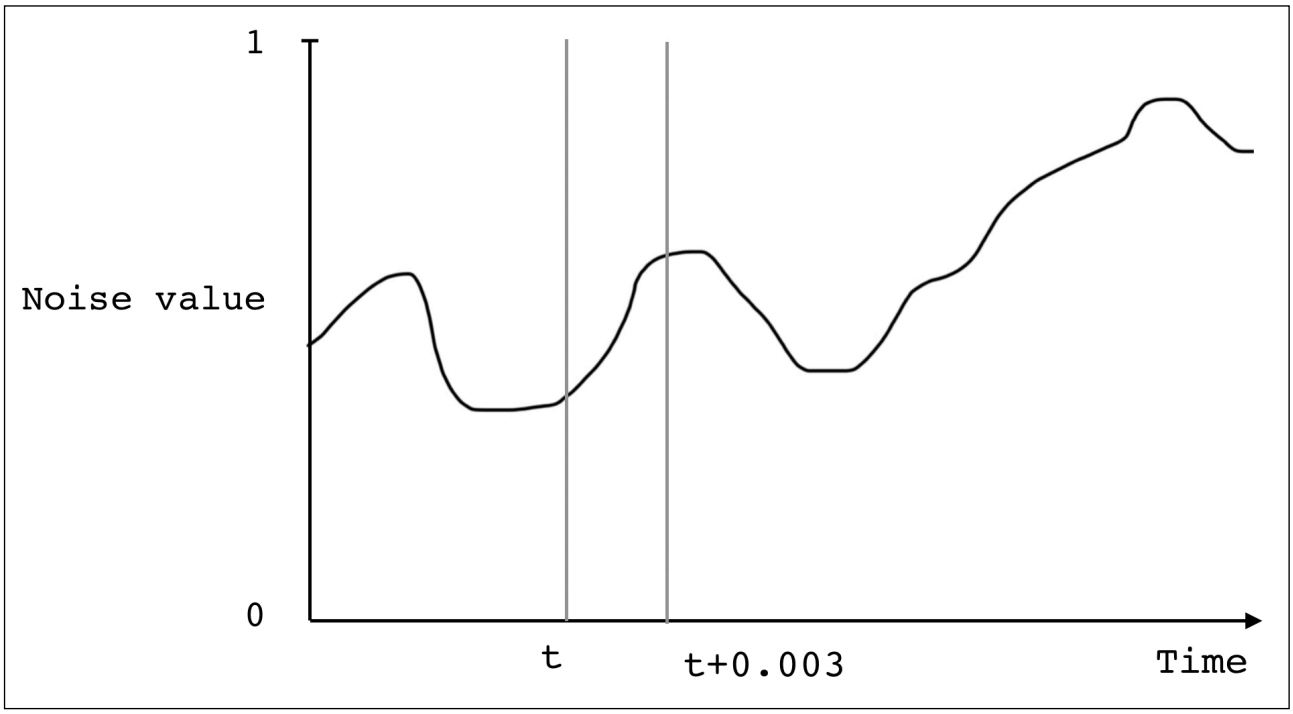
Perlin `noise()` returns a random value between 0.0 and 1.0 at a specific time. You specify the start time. It is then incremented along this smooth random time line in steps, the smaller the steps (for instance 0.005) means there are very small changes but smoother, whereas larger steps (0.03) obviously creates a much greater degree of randomness and possibly less smooth. Seems to work best between 0.005-0.03.

If this seems strange then don't let it be. If you have two variables that you want to have a different random outcome you simply grab the `noise()` starting at different times, for example 3 or 100 etc.

The beauty of this is that it is random but it bears some relationship with the previous random number at a particular point in time. So if you move it on a small increment you get a slight adjustment to the random value.

If you really want to understand how it works and who invented it (and why!) then I suggest you read up about it in Dan Shiffman's book/website 'Nature of Code' see resources tab.

Figure 1: perlin noise graph



Sketch C1.1 starting with a standard sketch

Starting with our normal basic sketch

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}
```

Sketch C1.2 randomly moving circle

We start the circle in the centre of the canvas and randomly move it. We are just going to use the `random()` function to move the circle around the canvas. You will notice that it is not very smooth or fluid, `noise` will give you something more natural.

```
let x = 200
let y = 200

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  circle(x, y, 100)
  x = x + random(-2, 2)
  y = y + random(-2, 2)
}
```

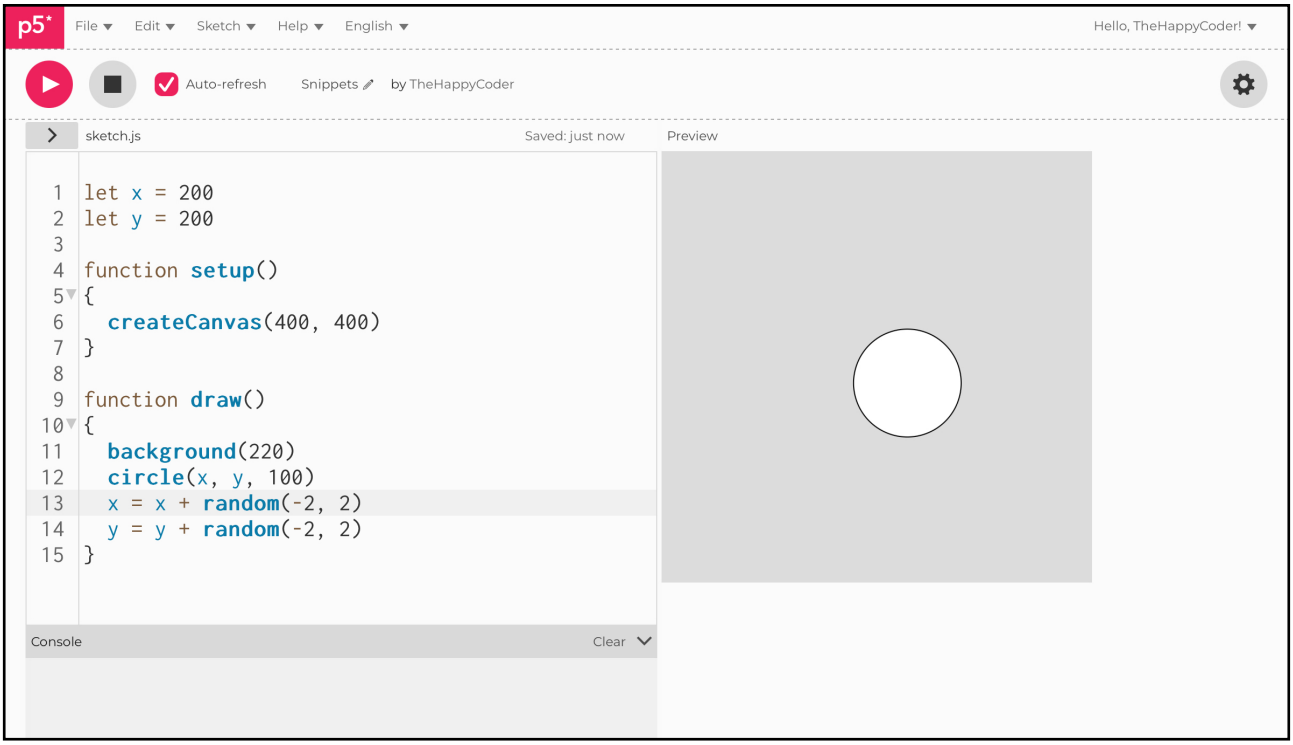
🌻 Challenges

1. Try random values of (5, -5)
2. Try different values for x and for the y

🛠️ Code Explanation

`x = x + random(-2, 2)`	Add a random value between -2 and 2 to the x value on each iteration
`y = y + random(-2, 2)`	Add a random value between -2 and 2 to the y value on each iteration

Figure C1.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 200
let y = 200
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
circle(x, y, 100)
x = x + random(-2, 2)
y = y + random(-2, 2)
}
Console
Clear

Sketch C1.3 smooth random movement

We have replaced the `random()` function with the `noise()` function. Notice that the jerkiness has gone, replaced by a much smoother movement, almost as if floating in the air, also it looks a lot more complicated. We have two start times (`3` and `10`). Because noise returns values between `0` and `1` we use `map()` to scale the movement up to the `width` and `height` of the canvas. Then on each iteration we move along the time line by `0.005`.

```
let timeX = 3
let timeY = 10

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  let x = map(noise(timeX), 0, 1, 0, width)
  let y = map(noise(timeY), 0, 1, 0, height)
  circle(x, y, 100)
  timeX = timeX + 0.005
  timeY = timeY + 0.005
}
```

Notes

To see how noise works and why it is so much better than just `random()` this short programme illustrates the smoothness of the movement.

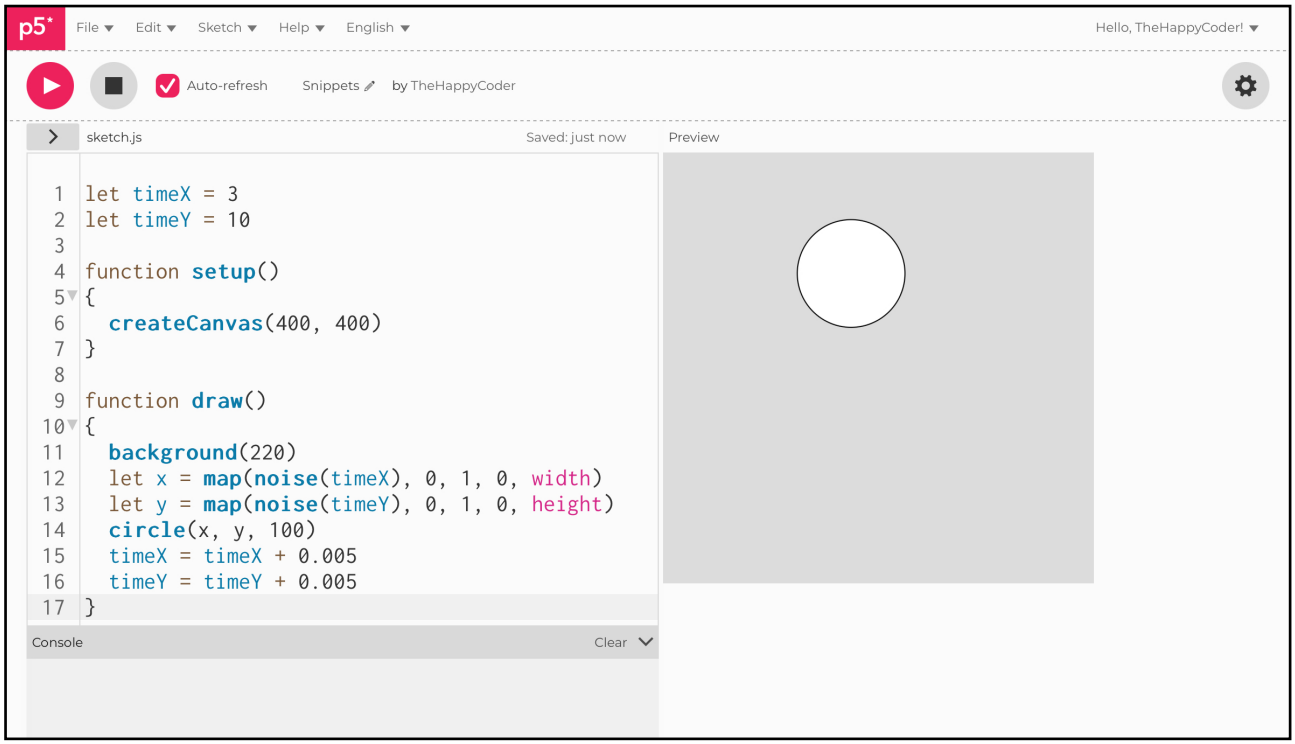
🌻 Challenges

1. Try different increments for timeX and timeY
2. What happens if you give them both the same start on the timeline?
3. Try: `timeX = timeX + 0.05`
4. Replace it with `timeX += 0.005`

🛠️ Code Explanation

`let timeX = 3`	The starting value for the x timeline
`let timeY = 10`	The starting value for the y timeline
`let x = map(noise(timeX), 0, 1, 0, width)`	Maps the timeX value to the width of the canvas
`let y = map(noise(timeY), 0, 1, 0, height)`	Maps the timeY value to the height of the canvas
`timeX = timeX + 0.005`	Adds an increment to the timeX timeline
`timeY = timeY + 0.005`	Adds an increment to the timeY timeline

Figure C1.3


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let timeX = 3
let timeY = 10
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
let x = map(noise(timeX), 0, 1, 0, width)
let y = map(noise(timeY), 0, 1, 0, height)
circle(x, y, 100)
timeX = timeX + 0.005
timeY = timeY + 0.005
}
Console
Clear

It is all about functions, objects and classes

This next section looks at coding with functions, objects and classes. They demonstrate the different ways you can code the same effect using different approaches. The context we will use is of a vehicle or vehicles moving either across the canvas or around it.

We can describe the vehicle with a `show()` function and its movement with a `move()` function. These are simple examples to highlight the differences to give you a flavour of what that might look like.

Sketch C1.4 starting sketch

We start a new sketch as usual

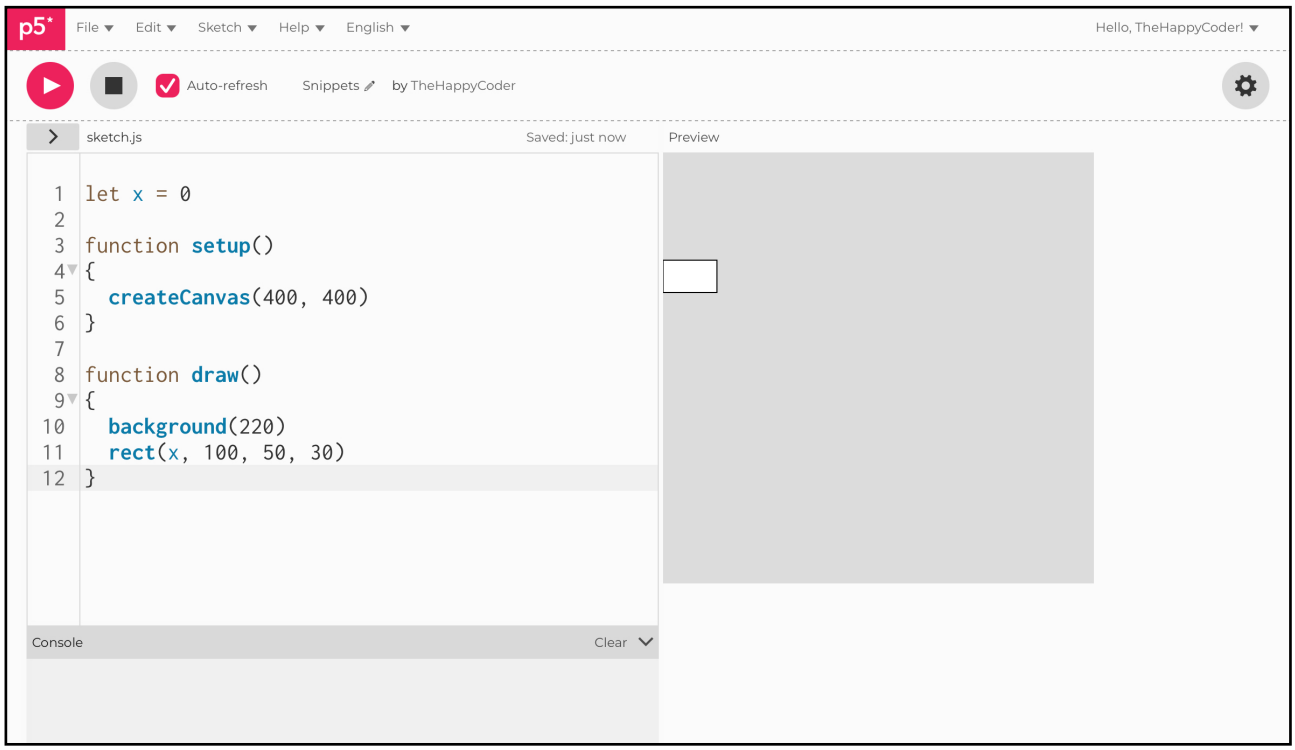
```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}
```

Sketch C1.5 a single car

We create our car as a simple rectangle, starting at the left hand edge of the canvas

```
let x = 0

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  rect(x, 100, 50, 30)
}
```

Figure C1.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
rect(x, 100, 50, 30)
}
Console
Clear

Sketch C1.6 moving the car

Now we start the car moving across the canvas

```
let x = 0

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  rect(x, 100, 50, 30)
  x += 1
}
```

Notes

It moves slowly across the canvas and disappears of the right hand edge of the canvas never to be seen again.

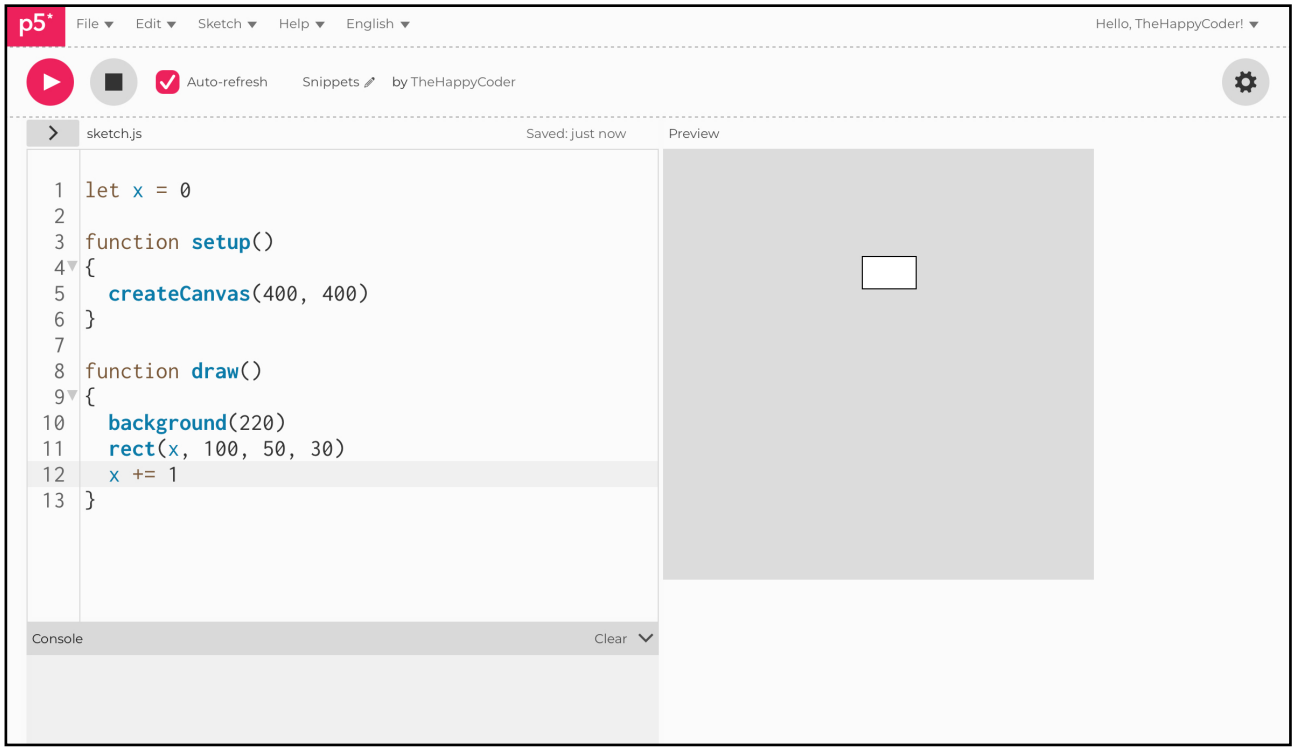
Challenge

Make it move faster

Code Explanation

`x += 1`	This adds 1 to x on each iteration

Figure C1.6


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
rect(x, 100, 50, 30)
x += 1
}
Console
Clear

Sketch C1.7 a single car

The car now reappears on the left hand edge and off it goes again

```
let x = 0

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  rect(x, 100, 50, 30)
  x += 1
  if (x > width)
  {
    x = -50
  }
}
```

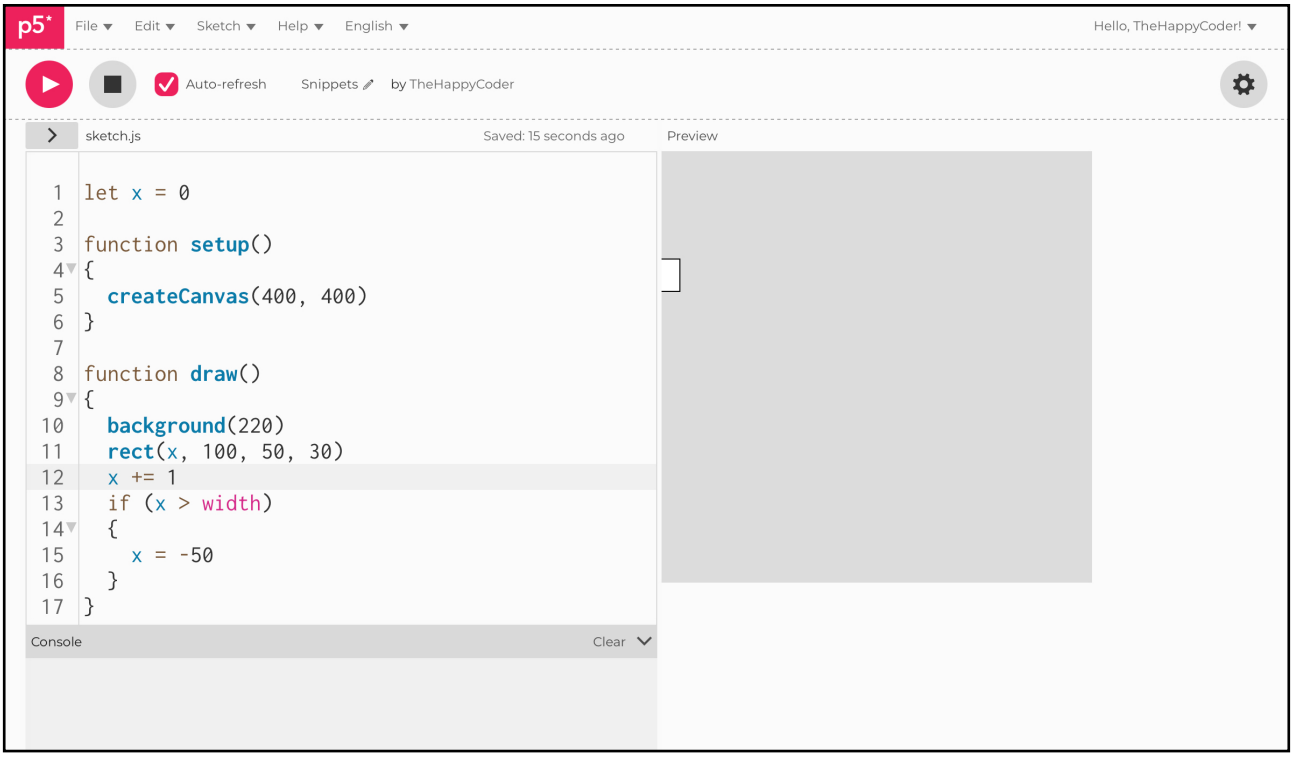
Notes

We have x as −50 so it looks like it is seamlessly continuous

Code Explanation

`if (x > width)`	Checks to see if it has reached the edge of the canvas
`x = -50`	Returns the x value to -50 if the car has gone off the edge of the canvas

Figure C1.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: 15 seconds ago
Preview
let x = 0
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
rect(x, 100, 50, 30)
x += 1
if (x > width)
{
x = -50
}
}
Console
Clear

The car as a function

We can express the same thing as before but this time we use functions, two of them to describe the car and to describe the motion. This means we have the `setup()` function as before, we keep the `draw()` function (empty for now) and add the other two functions called `show()` and `move()`, putting a lot of the stuff in those new functions.

I am using a very simple example here but bear with me as we will build on this concept when we introduce classes later.

Sketch C1.8 function single car

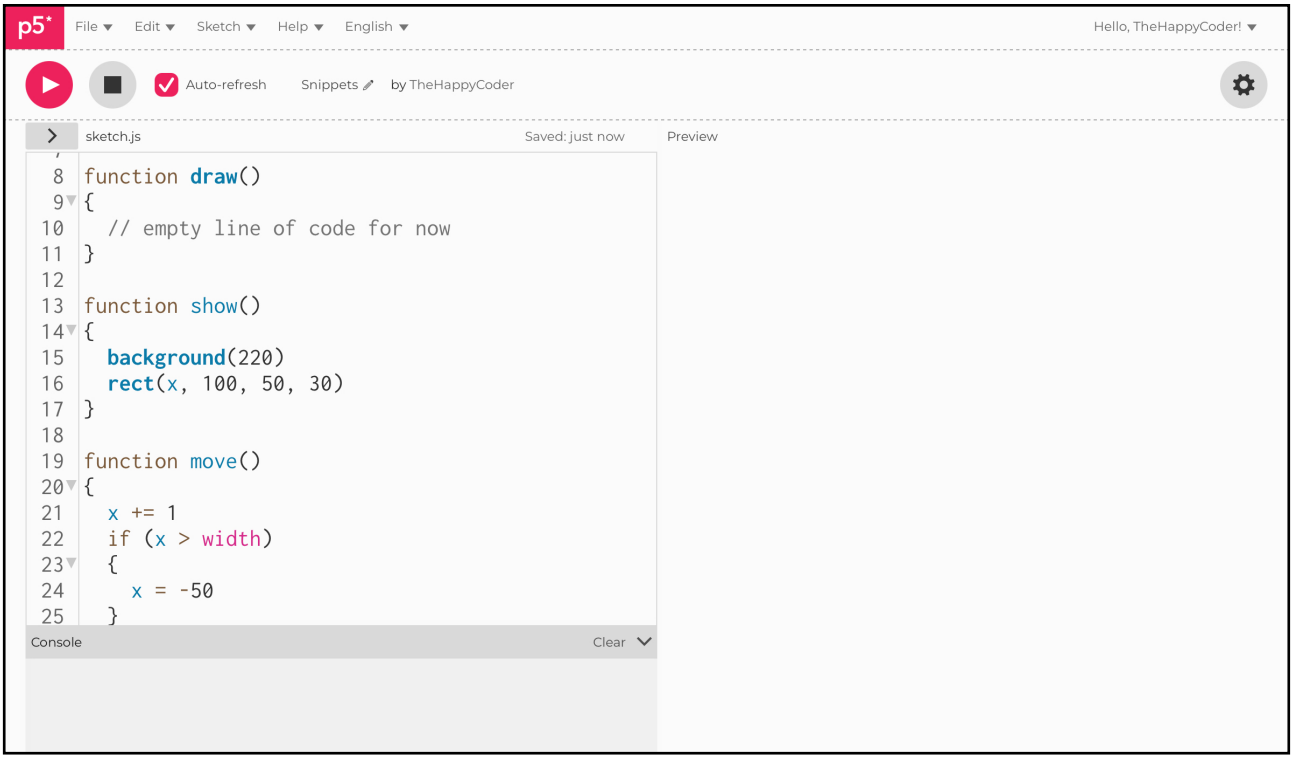
We have moved the code that was in the `draw()` function and split it between the two new functions: `show()` and `move()`, we have used the same code as in the previous sketch just moved it around.

```
let x = 0

function setup()
{
  createCanvas(440, 400)
}

function draw()
{
  // empty line of code
}

function show()
{
  background(220)
  rect(x, 100, 50, 30)
}

function move()
{
  x += 1
  if (x > width)
  {
    x = -50
  }
}
```

🗒️ Notes

You will notice that nothing happens, there isn't even a canvas. You can just cut and paste to save time.

Figure C1.8


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
8 function draw()
9 {
10 // empty line of code for now
11 }
12
13 function show()
14 {
15 background(220)
16 rect(x, 100, 50, 30)
17 }
18
19 function move()
20 {
21 x += 1
22 if (x > width)
23 {
24 x = -50
25 }
Console
Clear

Sketch C1.9 function single car

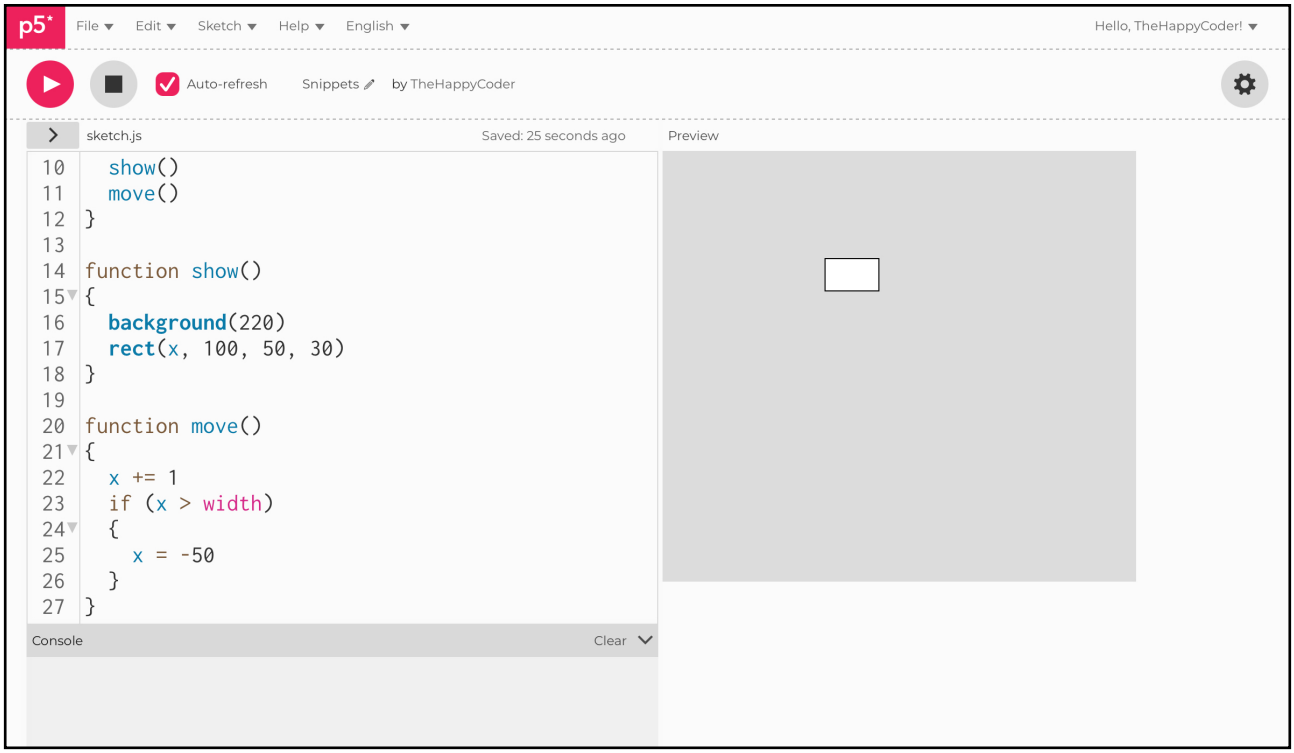
To get the two new functions to do anything we need to call them from inside the `draw()` function and we do it as shown below.

```
let x = 0

function setup()
{
  createCanvas(440, 400)
}

function draw()
{
  show()
  move()
}

function show()
{
  background(220)
  rect(x, 100, 50, 30)
}

function move()
{
  x += 1
  if (x > width)
  {
    x = -50
  }
}
```

🗒️ Notes

Now we are back where we started, but let's not stop there, there is yet another way we can do this even before we introduce classes.

Figure C1.9


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: 25 seconds ago
Preview
show()
move()
}
function show()
{
background(220)
rect(x, 100, 50, 30)
}
function move()
{
x += 1
if (x > width)
{
x = -50
}
}
Console
Clear

The car as an object using functions

This exercise is another way of doing the same thing. I include it because it shows the concept of objects in relation to functions. We could easily create two cars but it would mean doubling all the code for each car. This is another reason where classes come into their own, but we are getting ahead of ourselves here.

Sketch C1.10 car as an object

❗ Start a new sketch (highlighted differences to basic sketch)
We have added the car as an object, notice the similarity to our earlier sketch but now we have to give it a name.

```
let car = {x: 0}

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  rect(car.x, 100, 50, 30)
  car.x += 1
  if (car.x > width)
  {
    car.x = -50
  }
}
```

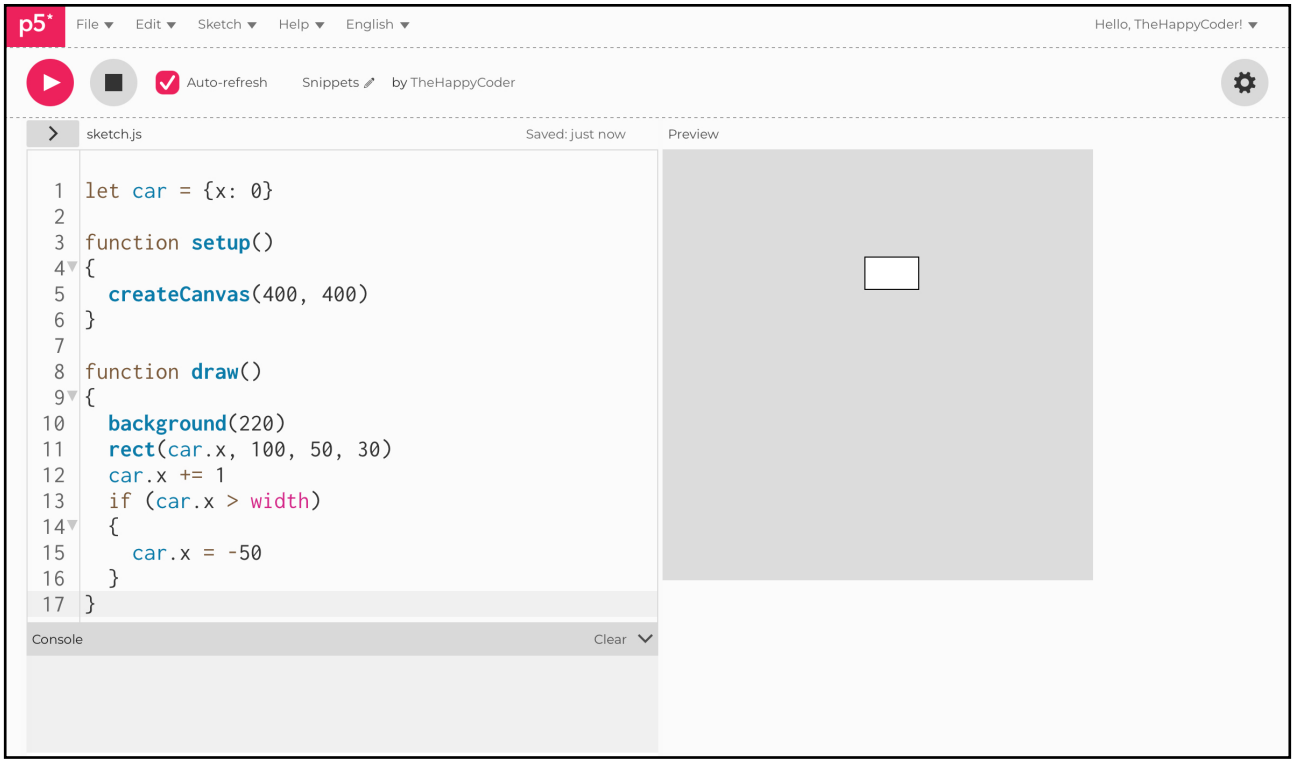
🗒️ Notes
Everything behaves just as before

🌻 Challenges
1. Give it a y component
2. Use the show() and move() functions (if struggling see next sketch)

🛠️ Code Explanation

`let car = {x: 0}`	We initialise the x component of the car object to 0
`rect(car.x, 100, 50, 30)`	The x component of the car object
`car.x += 1`	Incrementing the x component by 1 on each iteration
`if (car.x > width)`	Check when the car has gone off the edge of the canvas
`car.x = -50`	The x component is re-initialised to -50

Figure C1.10


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let car = {x: 0}
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
rect(car.x, 100, 50, 30)
car.x += 1
if (car.x > width)
{
car.x = -50
}
}
Console
Clear

Sketch C1.11 alternative car object

Now we can use the functions `show()` and `move()` as well as introducing a `y` component. The background can go into `draw()` or `show()`

```
let car = {x: 0, y: 100}

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  show()
  move()
}

function show()
{
  rect(car.x, car.y, 50, 30)
}

function move()
{
  car.x += 1
  if (car.x > width)
  {
    car.x = -50
  }
}
```

🗒️ Notes

Looks quite elegant in my opinion, it should be exactly the same as before

🛠️ Code Explanation

`let car = {x: 0, y: 100}`	Adding the y component to the car object
`rect(car.x, car.y, 50, 30)`	Adding the y component to the rectangle drawn.

Introduction to classes

Using classes is a common way for coders to organise their code. It is not essential as you could do the same thing without using classes but it is a very powerful and useful approach and one worth investing the time understanding the approach.

It does take a bit of getting used to, I will try to illustrate this with a simple example. Image you have a template to build a car. You as a consumer want some choice. The colour, the number of doors, engine size, interior style and so on. A class is like the basic template, when you order a new car they don't ask if you want doors, seats, a steering wheel, windows etc. they come as standard.

A class will have the basics and the options. So that when they make 10,000 cars they can all be slightly different depending on what the customer wants. This is a very limited comparison but you will see that you can create lots of 'cars' that all behave slightly differently. In our first example we will do just that with a sort of car.

In the diagram below (figure 2) you will see that the class is given a name, it is usual to start the class name with a capital letter. Also there are three functions in the example below. You can have as many functions as you like and can call them anything you like.

The first function I use is called the `constructor()` function. This is just the usual name given to it. This is where we hold the information about any car we are going to build. Because it is a sort of template where we can make as many cars as we want we prefix any variable with the word `this`, for instance the colour would be `this.colour`, or the starting position will be `this.x` and `this.y` and so on.

The basic structure of the main sketch is demonstrated in figure 3. Where you create the car or cars from the class and call the functions within the class.

Figure 2: in the class

```
class Car
{
  constructor()
  {
    this.something
    this.somethingelse
  }

  show()
  {
    // what the car looks like
  }

  move()
  {
    // how the car will move
}
```

Figure 3: in the main sketch

```
let car

function setup()
{
  car = new Car()
}

function draw()
{
  car.show()
  car.move()
}
```

Sketch C1.12 the constructor function

❗ new sketch

We start with our basic sketch and create a class called Car. In that class we have a `constructor()` function. This function has four elements, that give us details about the car, its `colour`, its `x` position, its `y` position and its `velocity`. This first example will not reveal the power of using classes but a very gentle introduction to creating a class.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}

class Car
{
  constructor()
  {
    this.colour = 255
    this.x = 0
    this.y = 100
    this.velocity = 1
  }
}
```

Notes

When we give attributes to an object in a class we always use `this.` before the attribute. There is nothing to see at this point.

🛠️ Code Explanation

`this.colour = 255`	For a car we define its colour
`this.x = 0`	For a car we define its x position
`this.y = 100`	For a car we define its y position
`this.velocity = 1`	For a car we define its velocity

Sketch C1.13 the show function

In the `show()` function we will describe what the car will look like.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}

class Car
{
  constructor()
  {
    this.colour = 255
    this.x = 0
    this.y = 100
    this.velocity = 1
  }

  show()
  {
    fill(this.colour)
    rect(this.x, this.y, 50, 30)
  }
}
```

🗒️ Notes

It pulls the information from the `constructor()` function. Still nothing to see yet.

🛠️ Code Explanation

`fill(this.colour)`	This will fill it with white (255)
`rect(this.x, this.y, 50, 30)`	Creates a rectangle rect(0, 100, 50, 30)

Sketch C1.14 the move function

Next we describe how the car is going to move with the `move()` function inside the `Car` class

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}

class Car
{
  constructor()
  {
    this.colour = 255
    this.x = 0
    this.y = 100
    this.velocity = 1
  }

  show()
  {
    fill(this.colour)
    rect(this.x, this.y, 50, 30)
  }

  move()
```

```
  {
    this.x += this.velocity
    if (this.x > width)
    {
      this.x = -50
    }
  }
}
```

🗒️ Notes

This is exactly the same as with the previous examples of a moving car. However, we created a variable for the `velocity` rather than just have a value (`1`), this allows us to alter it later. As before still nothing to see.

🛠️ Code Explanation

`this.x += this.velocity`	For each car we add the velocity
`if (this.x > width)`	If a car reaches the edge of the canvas
`this.x = -50`	Return that car back to the lefthand edge

Sketch C1.15 creating a car

To create a `Car` we first give this car a name, then in `setup()` we create a `new Car` from the class as a template. We currently have fixed values such as `colour`, `x`, `y` and `velocity`.

```
let car

function setup()
{
  createCanvas(400, 400)
  car = new Car()
}

function draw()
{
  background(220)
}

class Car
{
  constructor()
  {
    this.colour = 255
    this.x = 0
    this.y = 100
    this.velocity = 1
  }

  show()
  {
    fill(this.colour)
```

```
    rect(this.x, this.y, 50, 30)
  }

  move()
  {
    this.x += this.velocity
    if (this.x > width)
    {
      this.x = -50
    }
  }
}
```

🗒️ Notes

Be aware that the variable name for the car is a lowercase `c` and the name of the class is an uppercase `C`. They both have the same name which I admit is a little confusing but they are totally separate entities. One is a variable name the other is a class name. Still nothing to see here.

🛠️ Code Explanation

`car = new Car()`	Creates a new car

Sketch C1.16 to see it and move it

In order to see the car we have to call the `show()` function and to move the car we have to call the `move()` function, both in the `draw()` function. We ascribe these two functions to the new car we have created called `car`.

```
let car

function setup()
{
  createCanvas(400, 400)
  car = new Car()
}

function draw()
{
  background(220)
  car.show()
  car.move()
}

class Car
{
  constructor()
  {
    this.colour = 255
    this.x = 0
    this.y = 100
    this.velocity = 1
  }
```

```
  show()
  {
    fill(this.colour)
    rect(this.x, this.y, 50, 30)
  }

  move()
  {
    this.x += this.velocity
    if (this.x > width)
    {
      this.x = -50
    }
  }
}
```

🗒️ Notes

Finally we get to see the car and watch it move

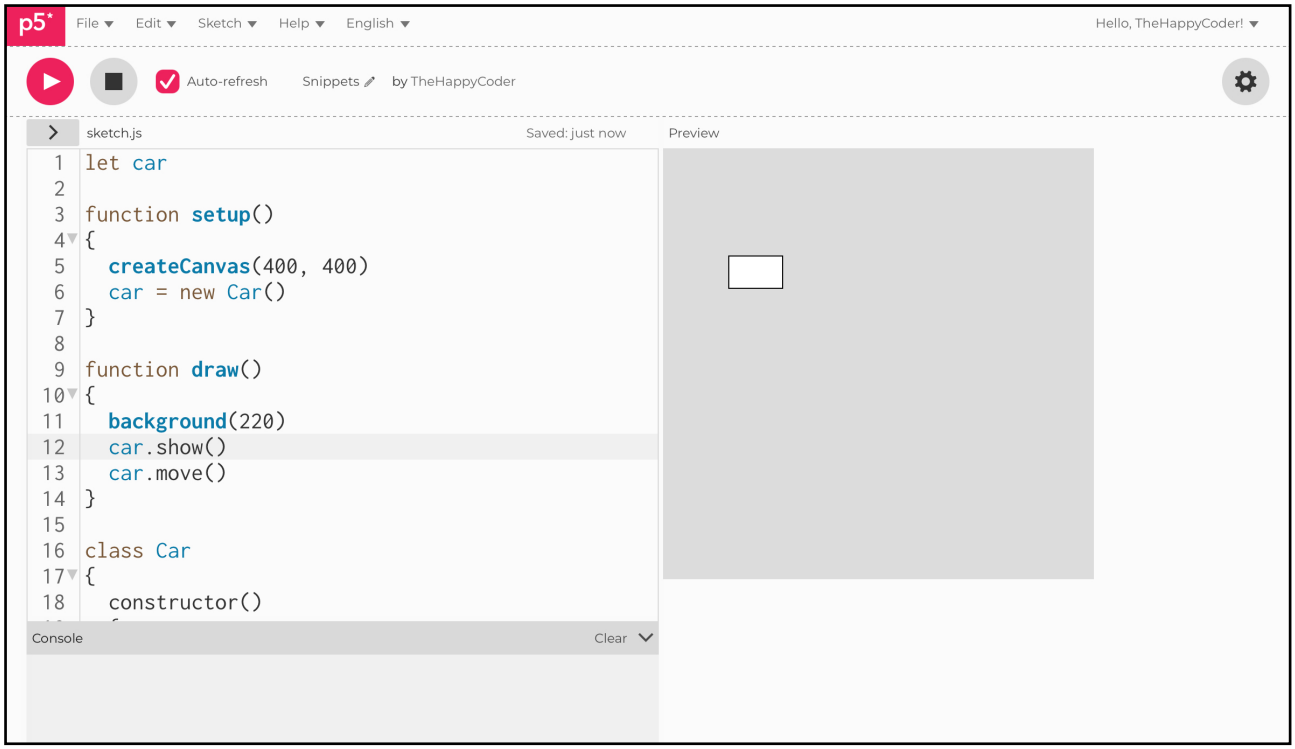
🌻 Challenges

1. Change the colour
2. Change the x value
3. Change the y value
4. Change the velocity
5. Change the name of the variable to myCar
6. Change the name of the class
7. Change the name of the constructor(), show() and move() functions

🛠️ Code Explanation

`car.show()`	For this car we show it according to the show() function
`car.move()`	For this car we move it according to the move() function

Figure C1.16


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let car
function setup()
{
createCanvas(400, 400)
car = new Car()
}
function draw()
{
background(220)
car.show()
car.move()
}
class Car
{
constructor()
Console
Clear

The power of classes

In the following sections we will consider what we can do with classes which makes all the trouble of creating them worthwhile. This is evident when we want 100's of them, where each one can be created separately, independently.

Sketch C1.17 car attributes

When we create the car we can specify its attributes rather than hard code them in the `constructor()` function. We have give the car the same values as before. They become the arguments in the `constructor()` function: `colour`, `x`, `y` and `velocity`. This is more like a template where you can now specify what you want.

```
let car

function setup()
{
  createCanvas(400, 400)
  car = new Car(255, 0, 100, 1)
}

function draw()
{
  background(220)
  car.show()
  car.move()
}

class Car
{
  constructor(colour, x, y, velocity)
  {
    this.colour = colour
    this.x = x
    this.y = y
    this.velocity = velocity
  }
```

```
  show()
  {
    fill(this.colour)
    rect(this.x, this.y, 50, 30)
  }

  move()
  {
    this.x += this.velocity
    if (this.x > width)
    {
      this.x = -50
    }
  }
}
```

🗒️ Notes

The result is exactly the same as before because we have specified the same features of our car. The beauty of this is we can create a second (or more) car with different features.

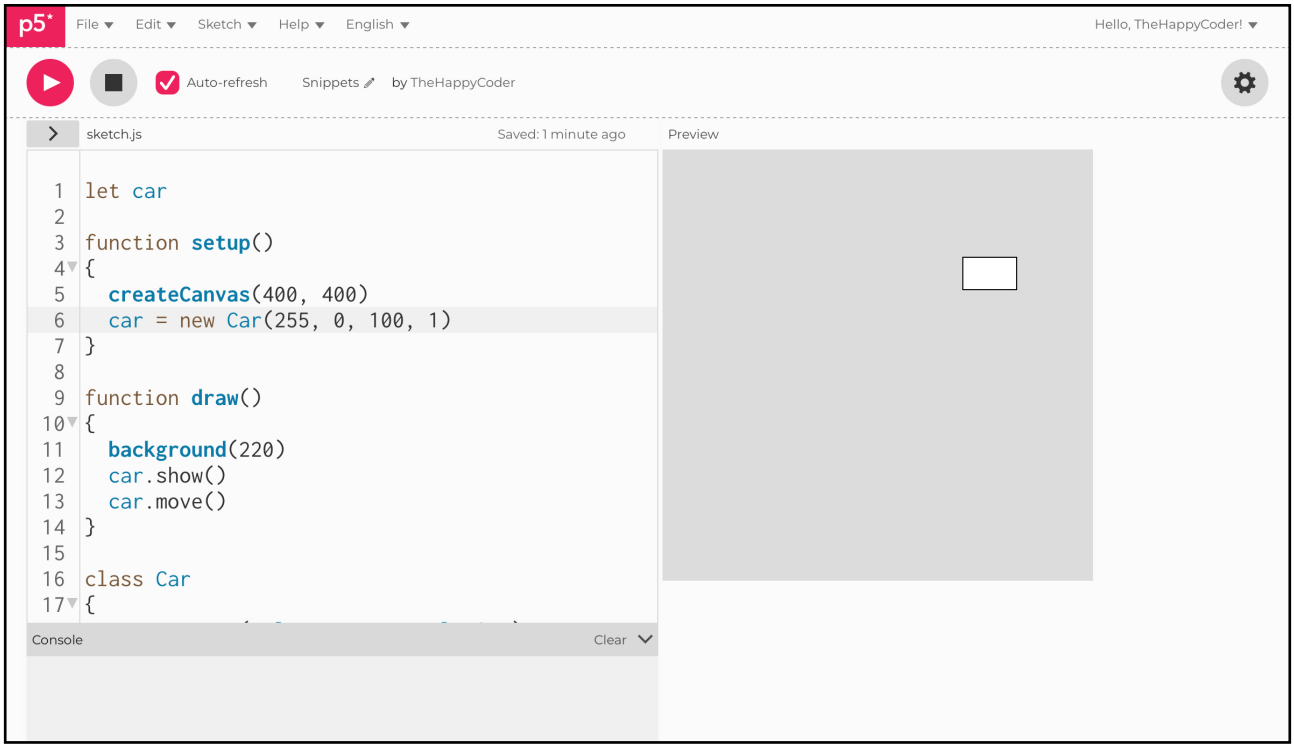
🌻 Challenge

Change the values/features of the car

🛠️ Code Explanation

`car = new Car(255, 0, 100, 1)`	We give this car some attributes
`constructor(colour, x, y, velocity)`	The attributes are received as arguments in the constructor() function
`this.colour = colour`	This car has the colour argument
`this.x = x`	This car has the x position argument
`this.y = y`	This car has the y position argument
`this.velocity = velocity`	This car has the velocity argument

Figure C1.17


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: 1 minute ago
Preview
let car
function setup()
{
createCanvas(400, 400)
car = new Car(255, 0, 100, 1)
}
function draw()
{
background(220)
car.show()
car.move()
}
class Car
{
Console
Clear

Sketch C1.18 a second car

We add a second car and give it different features.

```
let car
let car2

function setup()
{
  createCanvas(400, 400)
  car = new Car(255, 0, 100, 1)
  car2 = new Car(55, 0, 300, 2)
}

function draw()
{
  background(220)
  car.show()
  car.move()
  car2.show()
  car2.move()
}

class Car
{
  constructor(colour, x, y, velocity)
  {
    this.colour = colour
    this.x = x
    this.y = y
    this.velocity = velocity
  }
```

```
  show()
  {
    fill(this.colour)
    rect(this.x, this.y, 50, 30)
  }

  move()
  {
    this.x += this.velocity
    if (this.x > width)
    {
      this.x = -50
    }
  }
}
```

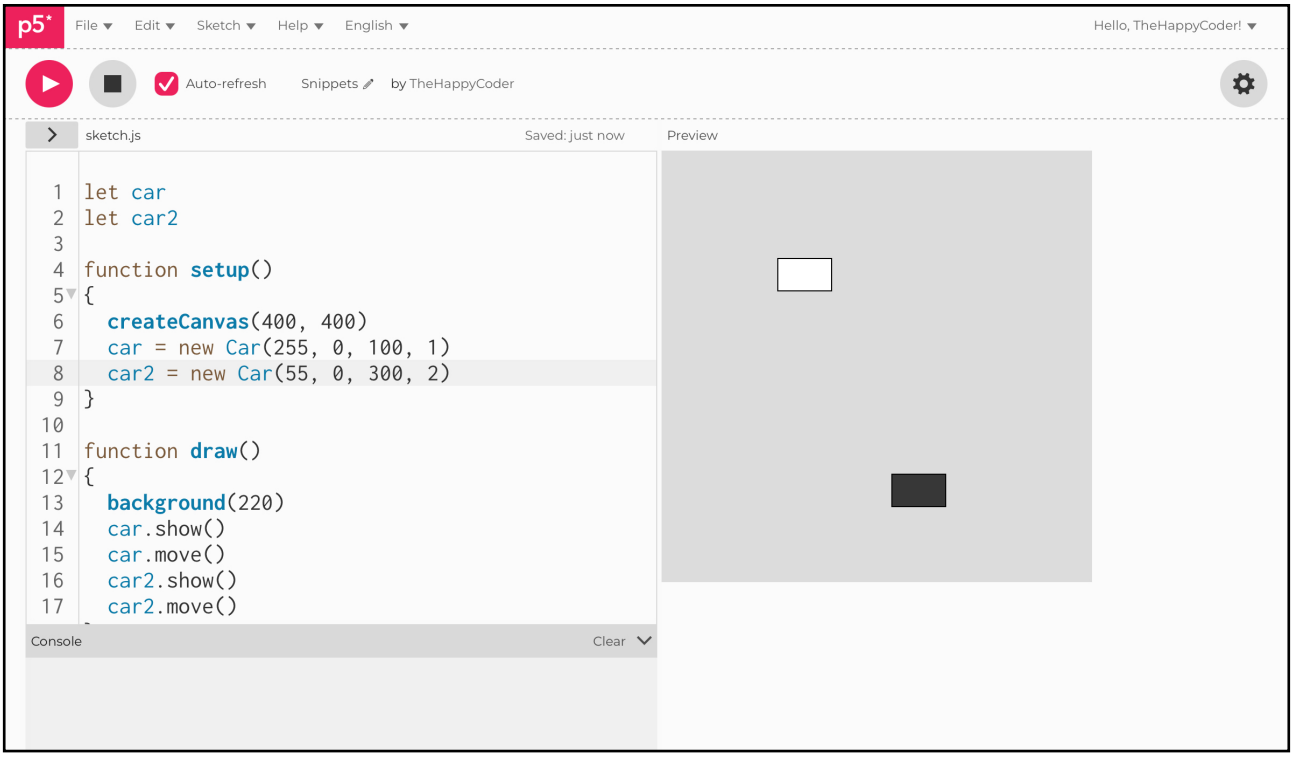
🗒️ Notes

With just a few lines of code we have created a second car. You can see the simple logic.

🌻 Challenge

Add a third car

Figure C1.18


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let car
let car2
function setup()
{
createCanvas(400, 400)
car = new Car(255, 0, 100, 1)
car2 = new Car(55, 0, 300, 2)
}
function draw()
{
background(220)
car.show()
car.move()
car2.show()
car2.move()
Console
Clear

Sketch C1.19 lots and lots of cars

Here is a quick peak at what we could do with loops to draw lots of cars. We have covered `arrays` and `for()` loops before. We create an array of cars and cycle through them with random values for all the features (except x). We then cycle through the array of cars, show and move them. All this is done in the `setup()` and draw function, we don't touch the `Car` class!

```
let car = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    car[i] = new Car(random(255), 0, random(400), random(1, 5))
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < car.length; i++)
  {
    car[i].show()
    car[i].move()
  }
}

class Car
{
  constructor(colour, x, y, velocity)
```

```
  {
    this.colour = colour
    this.x = x
    this.y = y
    this.velocity = velocity
  }

  show()
  {
    fill(this.colour)
    rect(this.x, this.y, 50, 30)
  }

  move()
  {
    this.x += this.velocity
    if (this.x > width)
    {
      this.x = -50
    }
  }
}
```

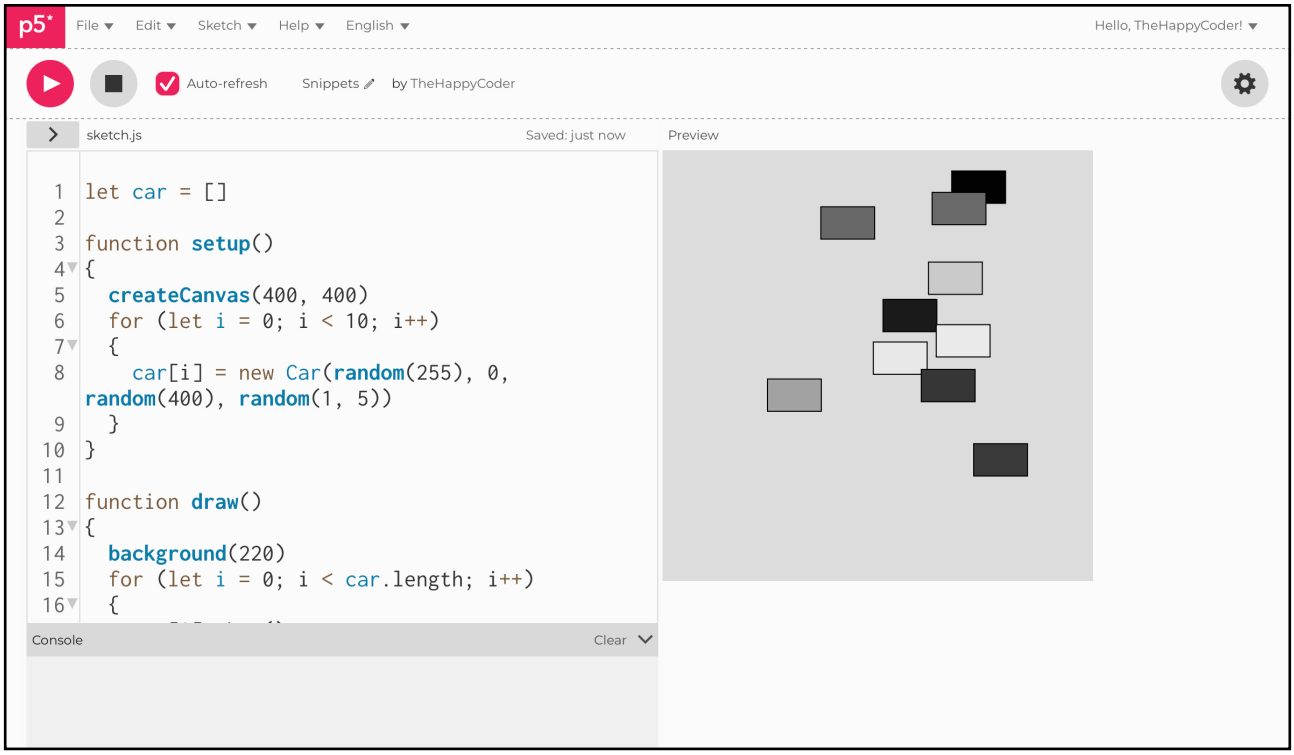
🗒️ Notes

I think that is pretty elegant!

🌻 Challenge

Just have a play

Figure C1.19


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let car = []
function setup()
{
createCanvas(400, 400)
for (let i = 0; i < 10; i++)
{
car[i] = new Car(random(255), 0,
random(400), random(1, 5))
}
}
function draw()
{
background(220)
for (let i = 0; i < car.length; i++)
{
Console
Clear

Seeking a target using classes

In this example we are going to steer a vehicle towards a moving object (your mouse). This is in preparation for the NeuroEvolution part of the AI tutorial. At the same time we will have a more detailed look at using classes. For this we will also add a file for the vehicle Class. This, seemingly, simple idea is quite a challenge.

If a vehicle is travelling in a certain direction with a particular speed how does it steer towards a target so that it realistically changes direction rather than simply stopping turning and moving. The vehicle is going to seek the target and steer to towards it.

The simple formula is described: steering = (desired velocity) - velocity.

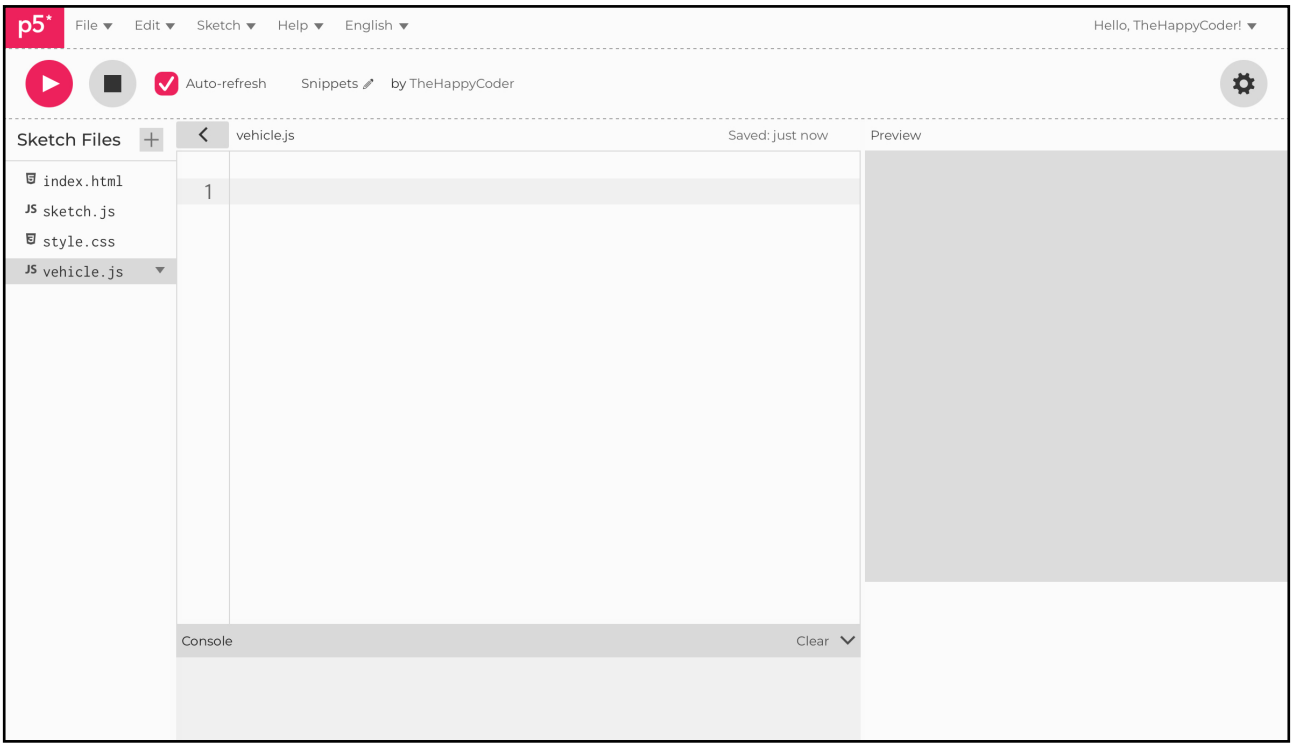
Adding files

Before we go any further we need to add a file called vehicle.js. I have already described how to add a file on page 52 of the Tutorial Brief Guide, if you are still unfamiliar I suggest going there quickly as a reminder and then return here. In brief, you click on the + sign next to Sketch Files, which gives you a drop down menu and click on the Create file and in the box give it the name vehicle.js. When you have done that go to index.html and add in the line of code <script src="vehicle.js"></script>. This last step is important otherwise your sketch won't know it is there. To access the vehicle.js sketch you click on it in the Sketch Files drop down menu. You will be flitting between that file and the main sketch.js file.

Step 1️⃣

Creating a new file called `vehicle.js`, this covered in getting stared with p5.js, make sure that the spelling and case is correct (the index.html file is case sensitive). You can call it anything you like but you must be consistent, however.

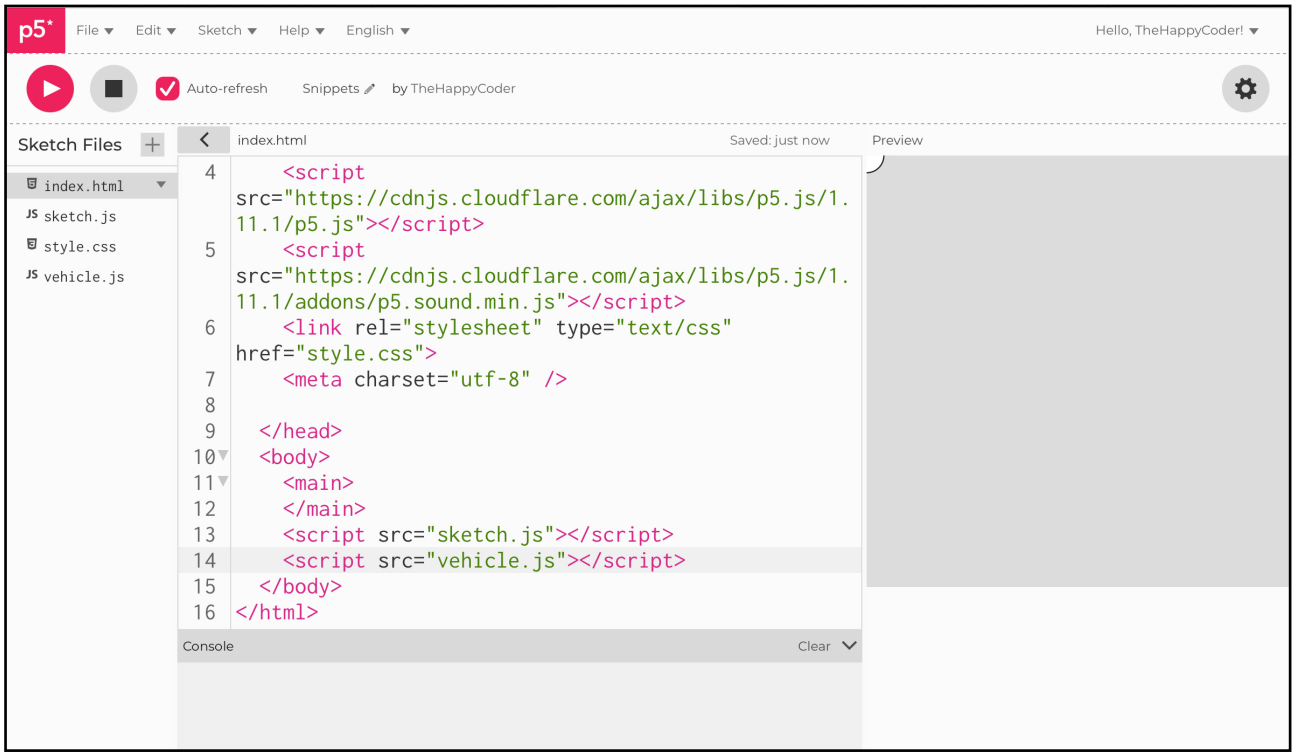
Figure 4: creating a new file called vehicle.js



Step 2️⃣

Referencing the file in the index.html file, you can simply copy and paste the line of code for sketch.js and change it to vehicle.js. You need to add it otherwise the index.html file won't know it is there and so it remains invisible (it is easy to forget to do this bit after creating the file in the first place but you soon find out!

Figure 5: referencing the file in the index.html file



Sketch C1.20 index.html

This is what the `index.html` file looks like now. This step is easy to forget after creating it.

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.10.0/p5.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.10.0/addons/p5.sound.min.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8" />

  </head>
  <body>
    <main>
    </main>
    <script src="sketch.js"></script>
    <script src="vehicle.js"></script>
  </body>
</html>
```

Notes

The spelling and case sensitivity is very important

Sketch C1.21 new sketch with a target

❗ new sketch in `sketch.js` (it based on our bog standard sketch)
I will indicate whether we are in `sketch.js` or `vehicle.js` by referencing it on the top line of the coding box. You will need to flit from one to the other at times.

We create a target for our vehicle to seek, this is simply based on the `mouseX` and `mouseY` object position.

sketch.js

```
let target


function setup()
{
  createCanvas(400, 400)
}


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

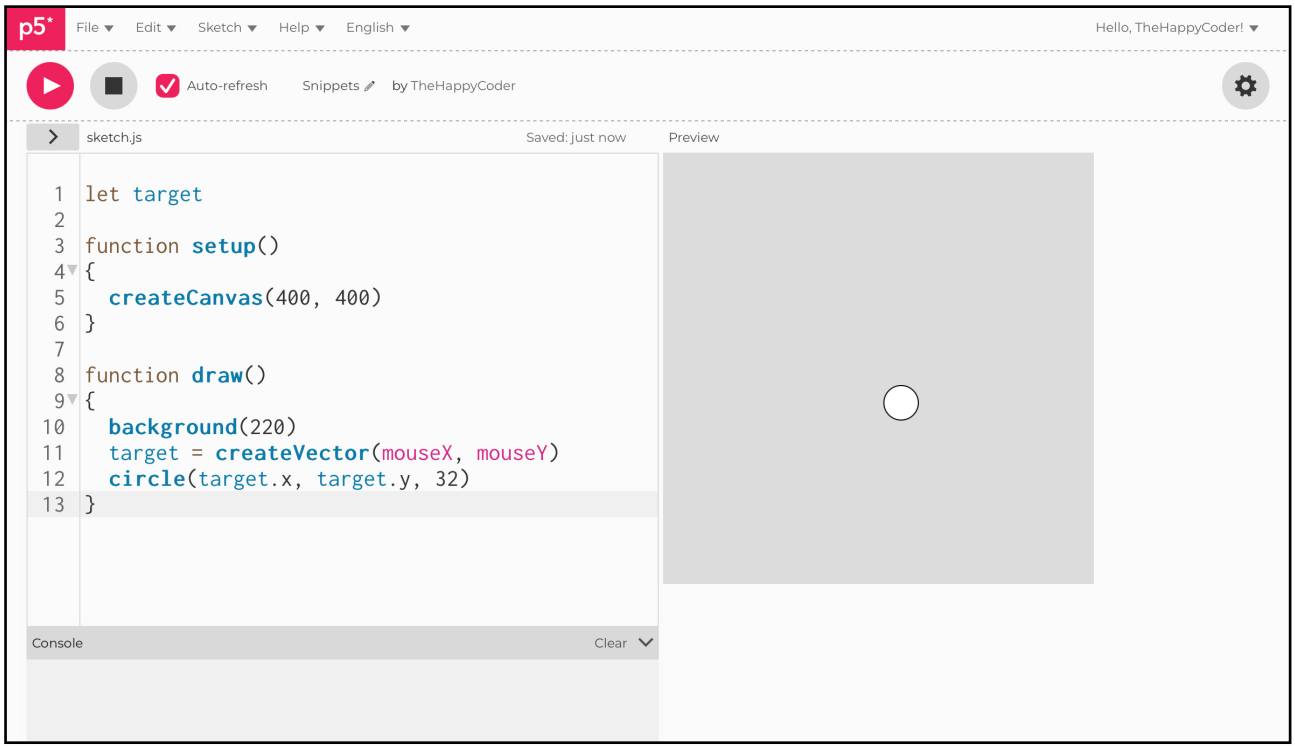
🗒️ Notes

The circle (which we are calling the target) follows the movement of your mouse

🛠️ Code Explanation

`target = createVector(mouseX, mouseY)`	We create a vector with two elements the mouseX and the mouseY
`circle(target.x, target.y, 32)`	We draw a circle to that vector, this is our target

Figure C1.21


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
sketch.js
Saved: just now
Preview
let target
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
target = createVector(mouseX, mouseY)
circle(target.x, target.y, 32)
}
Console
Clear

Sketch C1.22 the Vehicle class

❗ the `vehicle.js` sketch

Here we create our `Vehicle` class. Notice it has a capital letter for its name. The `constructor()` function has two arguments, the `x` and `y` position of the vehicle. Nothing in the `show()` and `move()` functions yet.

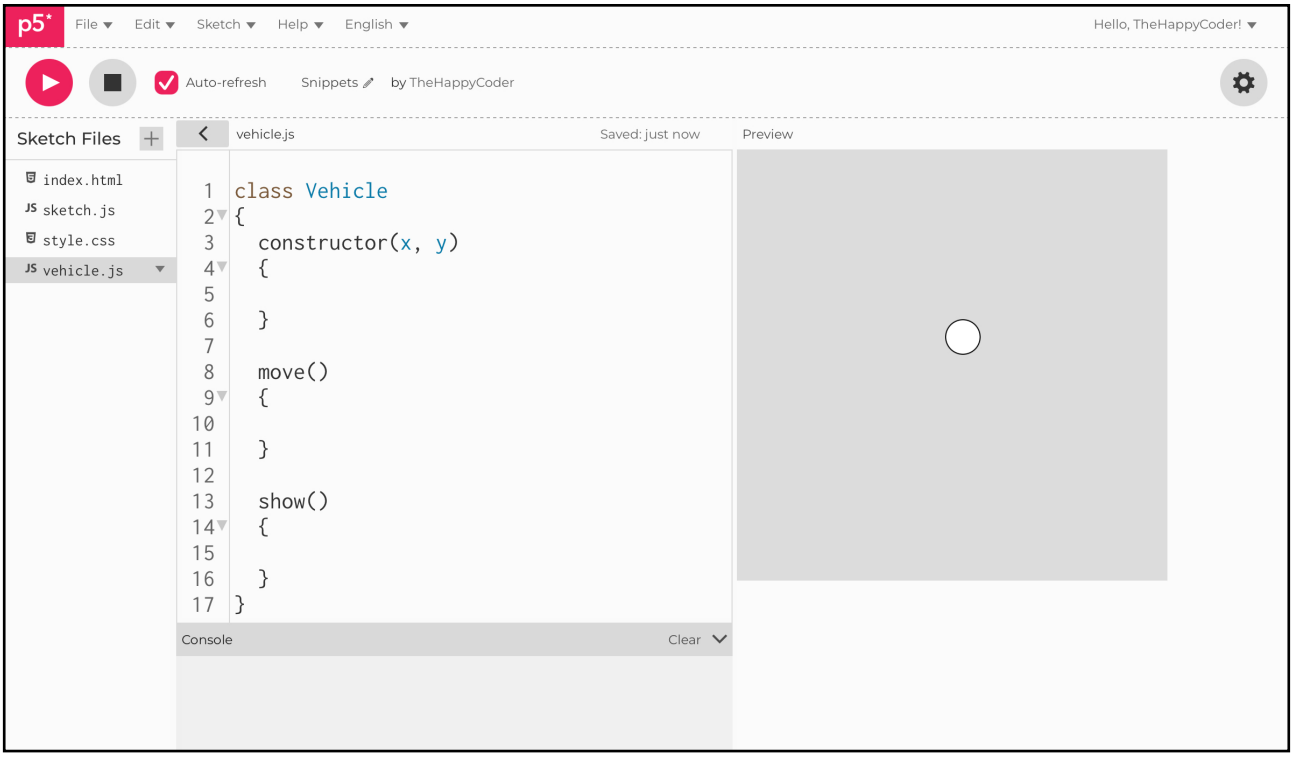
vehicle.js

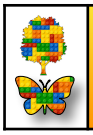
```
class Vehicle
{
  constructor(x, y)
  {

  }

  move()
  {

  }

  show()
  {

  }
}
```

🗒 Notes

This is a gentle introduction, we will build it gradually. We have our `target` (mouse) and the `Vehicle` class but nothing much else

Figure C1.22


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
Sketch Files
index.html
sketch.js
style.css
vehicle.js
vehicle.js
Saved: just now
Preview
1 class Vehicle
2 {
3 constructor(x, y)
4 {
5
6 }
7
8 move()
9 {
10
11 }
12
13 show()
14 {
15
16 }
17 }
Console
Clear



Sketch C1.23 a vehicle design

We want a triangle to represent our vehicle. We use the `triangle()` shape function to draw it. The dimensions of the triangle are based on a length `l` (which is `16`). We also create a vector for its position (`this.pos`). Then we translate the triangle by the position, this is because we have created a triangle based on the dimension of `l` and so we move it to the position we want at `x`, `y`. This is necessary for when we start to move the vehicle.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.l = 16
  }

  move()
  {

  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
  }
}
```

🗒️ Notes

We still won't see anything until we create the vehicle in the main sketch (`sketch.js`), so let's hop over there.

🌻 Challenge

1. to get a sense of the triangle's vertices draw them out on a piece of paper
2. Try different ways of using the length l

🛠️ Code Explanation

`translate(this.pos.x, this.pos.y)`	This uses the x and y coordinates and translates the triangle to that position
`triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)`	This spreads each vertex for each coroner of the triangle based on the x, y position and the length l

Sketch C1.24 a starting position

❗ go to `sketch.js`

We create a vehicle from the class template and put it in the position of (`100, 100`).

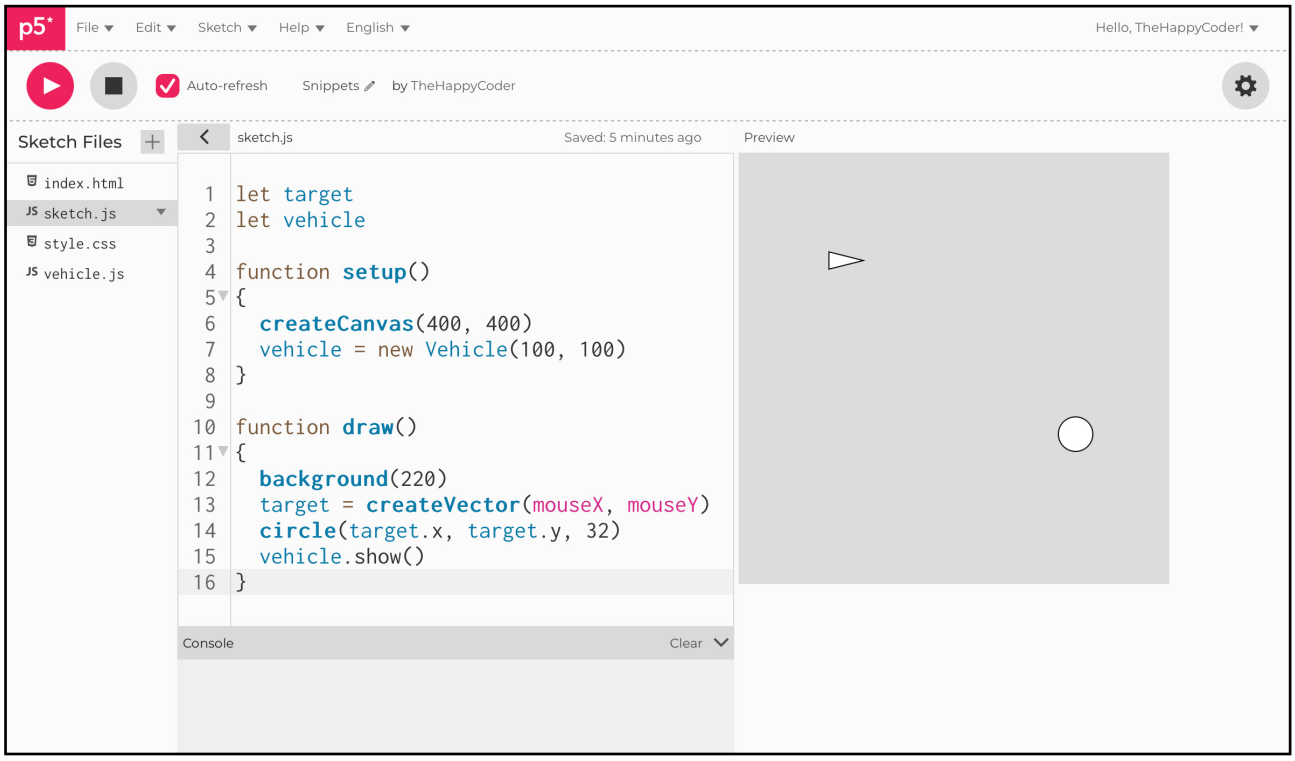
sketch.js

```
let target
let vehicle

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.show()
}
```

Notes

We have our `target` (mouse) and a `vehicle` but it isn't moving. We need to look at that next.

Figure C1.24


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
Sketch Files
index.html
sketch.js
style.css
vehicle.js
sketch.js
Saved: 5 minutes ago
Preview
let target
let vehicle
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.show()
}
Console
Clear

Movement with vectors

With vectors we describe movement a little bit differently. We could have a whole unit just on this but for now I will go through it briefly and simply. If you want to dive a little deeper then I suggest reading Dan Shiffman's book The Nature of Code which is also available as a web page (go to reference section on my website).

There are three elements to the movement of a vector object. They are:

1. position
2. velocity
3. acceleration

In short the acceleration (`acc`) is added to the velocity (`vel`) which in turn is added to the position (`pos`). We start off creating three vectors for each of these components. We may (or may not given them any initial values.

The acceleration will be the force exerted on the object by the position of the target, imagine the vehicle moving in a certain direction and the target being at some angle to its path of movement, it will want to turn towards that target. So there is a force acting on the vehicle to turn it from moving in its current direction towards the target. It could just stop turn and move but if the target is moving and the vehicle is already moving in a particular direction then the correction is a force in the direction of the target. If that makes sense. This is why later on we apply a force.

Sketch C1.25 velocity and acceleration

The velocity (vel) and acceleration (acc) vector components are added and are initialised to zero.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.l = 16
  }

  move()
  {

  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
  }
}
```

Notes

This needs more to have any impact on what we are doing, again we are building it up slowly

🛠️ Code Explanation

`this.vel = createVector(0, 0)`	Velocity is zero in the x and y direction
`this.acc = createVector(0, 0)`	Acceleration is zero in the x and y direction

Sketch C1.26 adding acc, vel and pos

It may seem a bit backwards but we first add the acceleration to the velocity, then add the velocity to the position.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.l = 16
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
  }
}
```

Notes

This means the vehicle doesn't move

🛠️ Code Explanation

`this.vel.add(this.acc)`	We are adding the acceleration to the velocity
`this.pos.add(this.vel)`	We are adding the velocity to the position

Sketch C1.27 applying a force

The acceleration is a force applied to the vehicle to turn it. We add a new function to the `Vehicle class` called `applyForce()` with an argument which will carry the value of the `force` applied. This `force` is added to the acceleration. This `force` will depend on where the vehicle is and where the target is.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.l = 16
  }

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

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
  }

  show()
  {
    translate(this.pos.x, this.pos.y)
```

```
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
  }
}
```

🗒️ Notes

Building it up gradually

🛠️ Code Explanation

`applyForce(force)`	This is the new function in the Vehicle class that will apply the force to change the acceleration
`this.acc.add(force)`	The force is a vector that will change the acceleration vector

Sketch C1.28 seeking the target

We want a function to seek the target so we add one to the Vehicle class. The `target` is already a vector (`mouse.x`, `mouse.y`). The `desired` is a vector which is the difference (subtract) between the target and where the `vehicle` is currently (`this.pos`).

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.l = 16
  }

  seek(target)
  {
    let desired = p5.Vector.sub(target, this.pos)
  }

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

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
```

```
  }


  show()
  {
    translate(this.pos.x, this.pos.y)
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
  }
}
```

🛠️ Code Explanation

`seek(target)`	The function receives an argument the target vector
`let desired = p5.Vector.sub(target, this.pos)`	Subtracts the two vectors, target and the position of the vehicle

Sketch C1.29 seek, move and show

❗ move to back to `sketch.js`

We will add three functions, `seek()`, `move()` and `show()`. These are the final changes to the main sketch.

sketch.js

```
let target
let vehicle

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.show()
  vehicle.move()
}
```

Notes

As yet we there is still nothing to see, we still have a bit more to do in the `Vehicle` class, so this is where we are heading next.

Sketch C1.30 reset acceleration

❗ head back to `vehicle.js`

We are going to reset the the acceleration otherwise it will keeps accumulating. We just want the actual current acceleration on each iteration in the `draw()` function in `sketch.js`.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.l = 16
  }

  seek(target)
  {
    let desired = p5.Vector.sub(target, this.pos)
  }

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

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
```

```
    this.acc.set(0, 0)
  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
  }
}
```

🗒️ Notes

There is a logic to this

🛠️ Code Explanation

`this.acc.set(0, 0)`	Resets the acceleration to zero so that it doesn't keep accumulating

Sketch C1.31 steering

We want to steer the vehicle towards the target, the steering vector is the difference between the desired and the actual motion of the vehicle.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.l = 16
  }

  seek(target)
  {
    let desired = p5.Vector.sub(target, this.pos)
    let steering = p5.Vector.sub(desired, this.vel)
    this.applyForce(steering)
  }

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

  move()
  {
    this.vel.add(this.acc)
```

```
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
  }
}
```

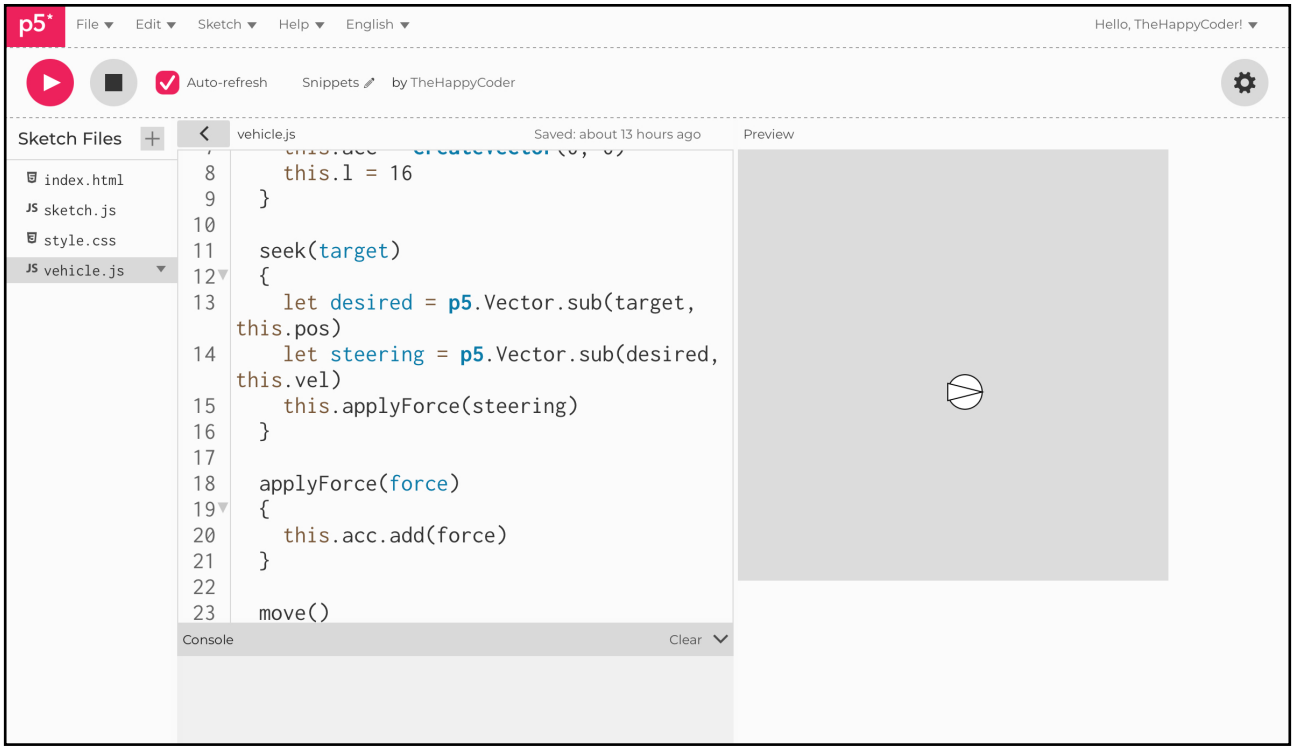
🗒️ Notes

What you get is the vehicle appearing where the target is, we are building step by step!

🛠️ Code Explanation

`let steering = p5.Vector.sub(desired, this.vel)`	The steering is a vector which is the desired vector minus the vehicle velocity vector
`this.applyForce(steering)`	This vector is the force applied to the vehicle to turn it towards the target

Figure C1.31


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
Sketch Files
index.html
sketch.js
style.css
vehicle.js
vehicle.js
Saved: about 13 hours ago
Preview
this.l = 16
}
seek(target)
{
let desired = p5.Vector.sub(target,
this.pos)
let steering = p5.Vector.sub(desired,
this.vel)
this.applyForce(steering)
}
applyForce(force)
{
this.acc.add(force)
}
move()
Console
Clear

Sketch C1.32 rotate to the heading

The function `heading()` calculates the angle between an axis and the vector. We use `push()` and `pop()` to isolate the movement of that particular vehicle.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.l = 16
  }

  seek(target)
  {
    let desired = p5.Vector.sub(target, this.pos)
    let steering = p5.Vector.sub(desired, this.vel)
    this.applyForce(steering)
  }

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

  move()
  {
    this.vel.add(this.acc)
```

```
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    push()
    rotate(this.vel.heading())
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
    pop()
  }
}
```

🗒️ Notes

We don't want to simply push the `vehicle` towards the `target` we want it to turn (`rotate`) and face the direction of travel.

🛠️ Code Explanation

`rotate(this.vel.heading())`	Calculates the angle a 2D vector makes with the positive x-axis and rotates it

Sketch C1.33 getting it working

We need to limit the magnitude of the velocity of the vehicle, we do that with `setMag()` function. We give it an arbitrary maximum velocity (`maxVelocity`) of `4`. What you should get is the vehicle moving towards the target rather than appearing instantly on top of it.

vehicle.js

```
class Vehicle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.l = 16
    this.maxVelocity = 4
  }

  seek(target)
  {
    let desired = p5.Vector.sub(target, this.pos)
    desired.setMag(this.maxVelocity)
    let steering = p5.Vector.sub(desired, this.vel)
    this.applyForce(steering)
  }

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

```
  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    push()
    rotate(this.vel.heading())
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
    pop()
  }
}
```

🗒️ Notes

What you will now see is the vehicle moving towards the target when you move the target by clicking on the canvas

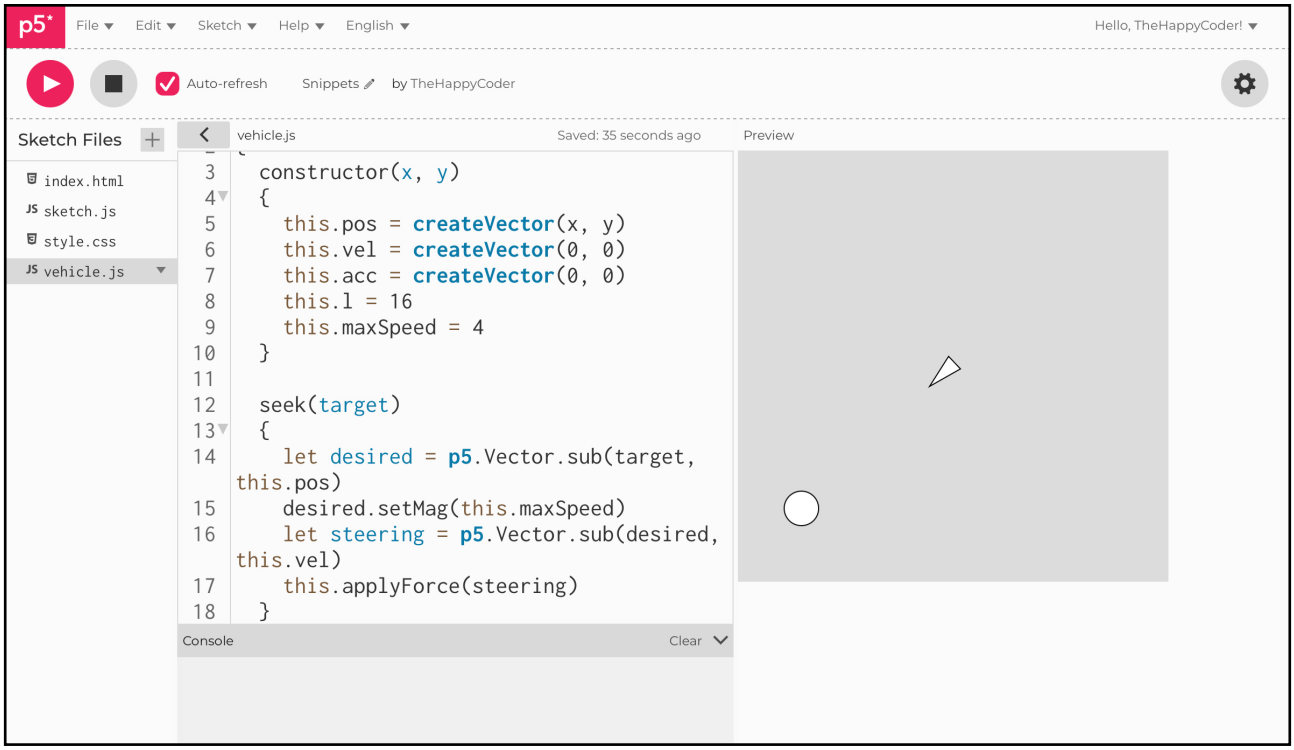
🌻 Challenge

Try other values for `maxVelocity`

🛠️ Code Explanation

`this.maxVelocity = 4`	As it says this is the maximum velocity
`desired.setMag(this.maxVelocity)`	Sets the magnitude of the desired

Figure C1.33


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
Sketch Files
index.html
sketch.js
style.css
vehicle.js
vehicle.js
Saved: 35 seconds ago
Preview
3 constructor(x, y)
4 {
5 this.pos = createVector(x, y)
6 this.vel = createVector(0, 0)
7 this.acc = createVector(0, 0)
8 this.l = 16
9 this.maxSpeed = 4
10 }
11
12 seek(target)
13 {
14 let desired = p5.Vector.sub(target, this.pos)
15 desired.setMag(this.maxSpeed)
16 let steering = p5.Vector.sub(desired, this.vel)
17 this.applyForce(steering)
18 }
Console
Clear

Sketch C1.34 the maximum force

To make it more realistic rather than just finding the quickest way to get to the target we limit the maximum force applied to the vehicle (`maxForce`) to `0.01`. To do that we use a built-in function called `limit()` which does exactly that. This means it arcs towards the target rather than just stop turn and move.

vehicle.js

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

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

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

```
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    push()
    rotate(this.vel.heading())
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
    pop()
  }
}
```

🗒️ Notes

We see a much more restrained vehicle, more sweeping arcs rather than point and move.

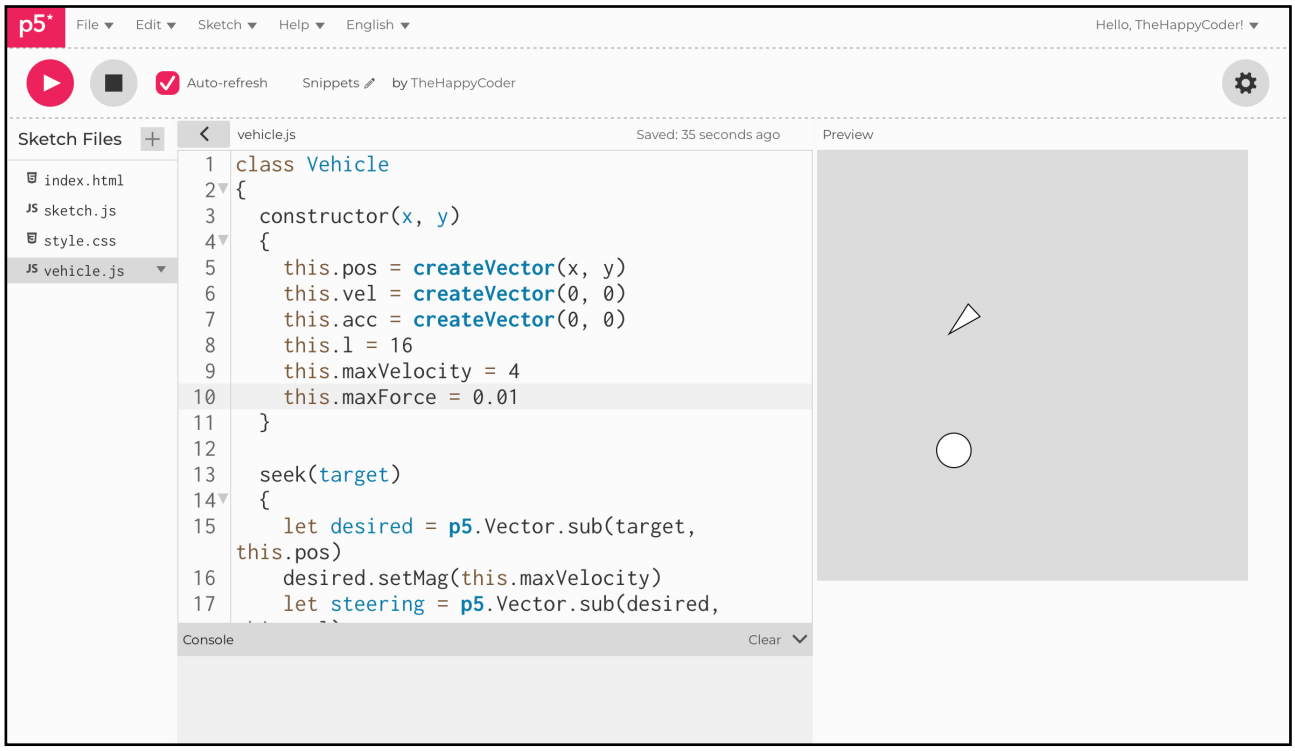
🌻 Challenges

1. Try different values of maximum velocity
2. What happens if you remove the setMag() line of code (comment out)
3. Try different values of maximum force
4. What happens if you remove the limit() line of code (comment out)

🛠️ Code Explanation

`this.maxForce = 0.01`	Helps to control the acceleration towards the target
`steering.limit(this.maxForce)`	Limits

Figure C1.34


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
Sketch Files
index.html
sketch.js
style.css
vehicle.js
vehicle.js
Saved: 35 seconds ago
Preview
1 class Vehicle
2 {
3 constructor(x, y)
4 {
5 this.pos = createVector(x, y)
6 this.vel = createVector(0, 0)
7 this.acc = createVector(0, 0)
8 this.l = 16
9 this.maxVelocity = 4
10 this.maxForce = 0.01
11 }
12
13 seek(target)
14 {
15 let desired = p5.Vector.sub(target,
this.pos)
16 desired.setMag(this.maxVelocity)
17 let steering = p5.Vector.sub(desired,
Console
Clear

Sketch C1.35 the force is now steering

Changing `desired` to `force` and remove `steering`, this whole `seek(target)` function has a revamp so that the `force` does the steering. This doesn't change the functionality of the code but you can see that there is more than one way approach the problem.

vehicle.js

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

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

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

```
  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.set(0, 0)
  }

  show()
  {
    translate(this.pos.x, this.pos.y)
    push()
    rotate(this.vel.heading())
    triangle(-this.l, -this.l/2, -this.l, this.l/2, this.l, 0)
    pop()
  }
}
```

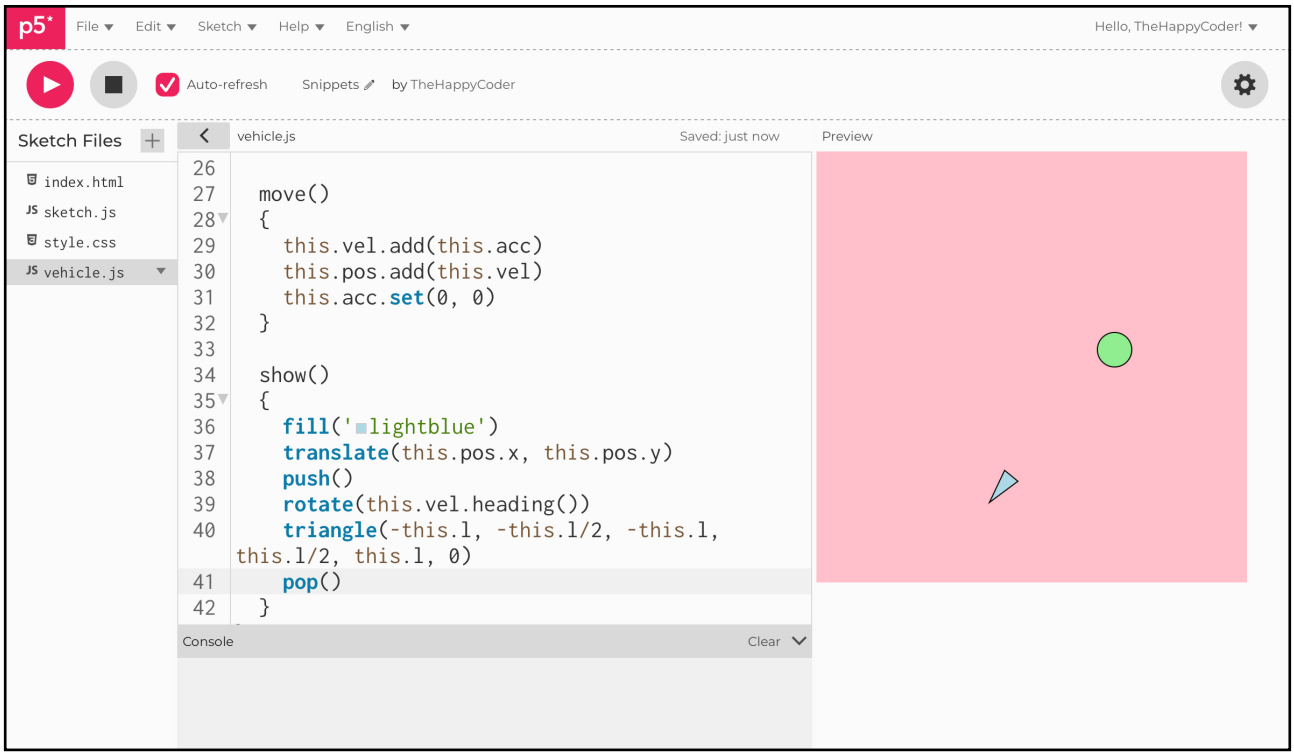
🗒️ Notes

This is an alternative for you to wrap your head around, it does look a lot more elegant

🌻 Challenges

1. Play around with the values of maxVelocity and maxForce to get something different
2. Add sliders for the two variables
3. Add some colour

Figure C1.35


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Snippets
by TheHappyCoder
Sketch Files
index.html
sketch.js
style.css
vehicle.js
vehicle.js
Saved: just now
Preview
26
27 move()
28 {
29 this.vel.add(this.acc)
30 this.pos.add(this.vel)
31 this.acc.set(0, 0)
32 }
33
34 show()
35 {
36 fill('lightblue')
37 translate(this.pos.x, this.pos.y)
38 push()
39 rotate(this.vel.heading())
40 triangle(-this.l, -this.l/2, -this.l,
this.l/2, this.l, 0)
41 pop()
42 }
Console
Clear