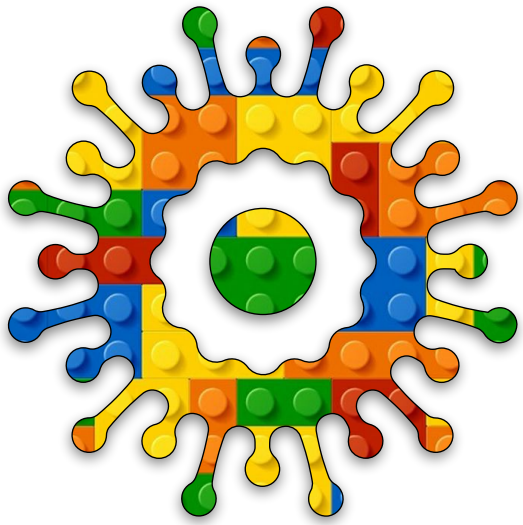
Creative Coding Module D Unit #1 functions & classes



Content

Module D Unit #1 functions and classes

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.

We can draw 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 D1.1 starting sketch

We start a new sketch, as usual.

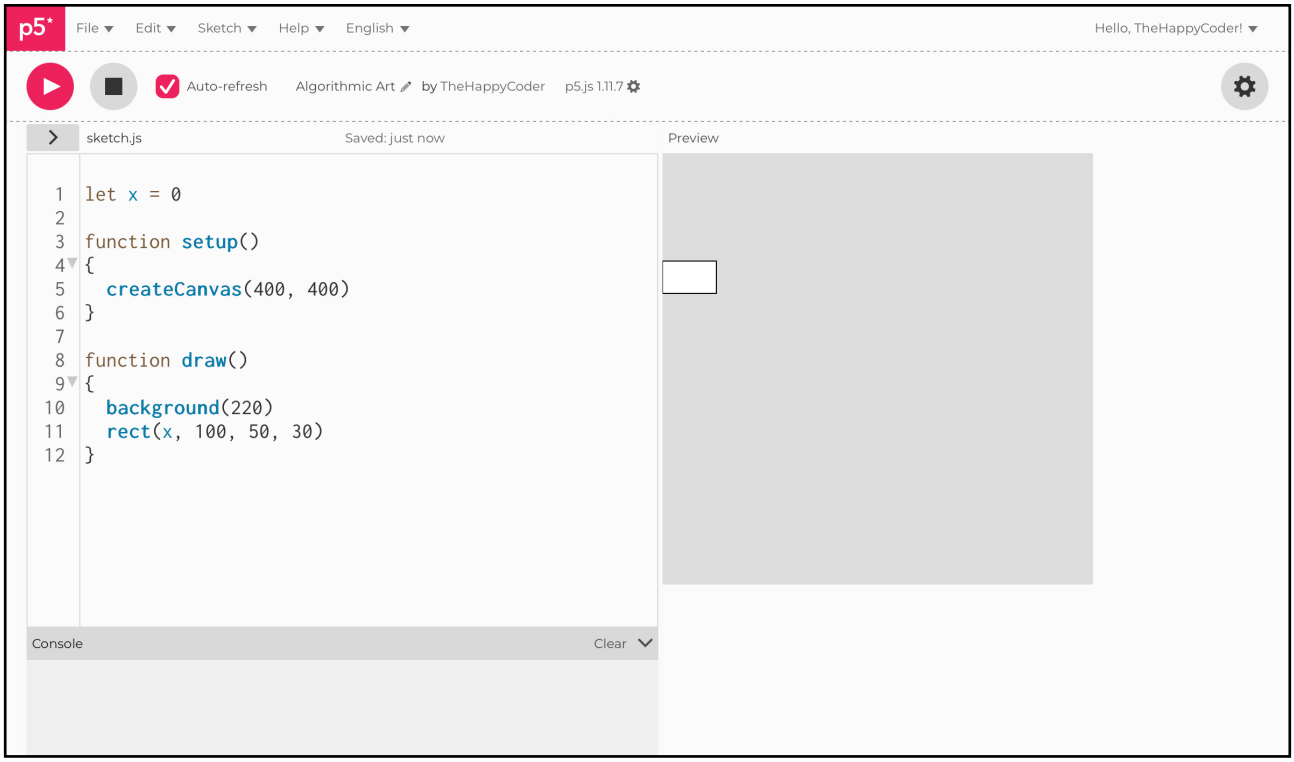
```
function setup()
{
  createCanvas(400, 400)
}

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

Sketch D1.2 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 D1.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
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 D1.3 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 from the right-hand edge of the canvas, never to be seen again.

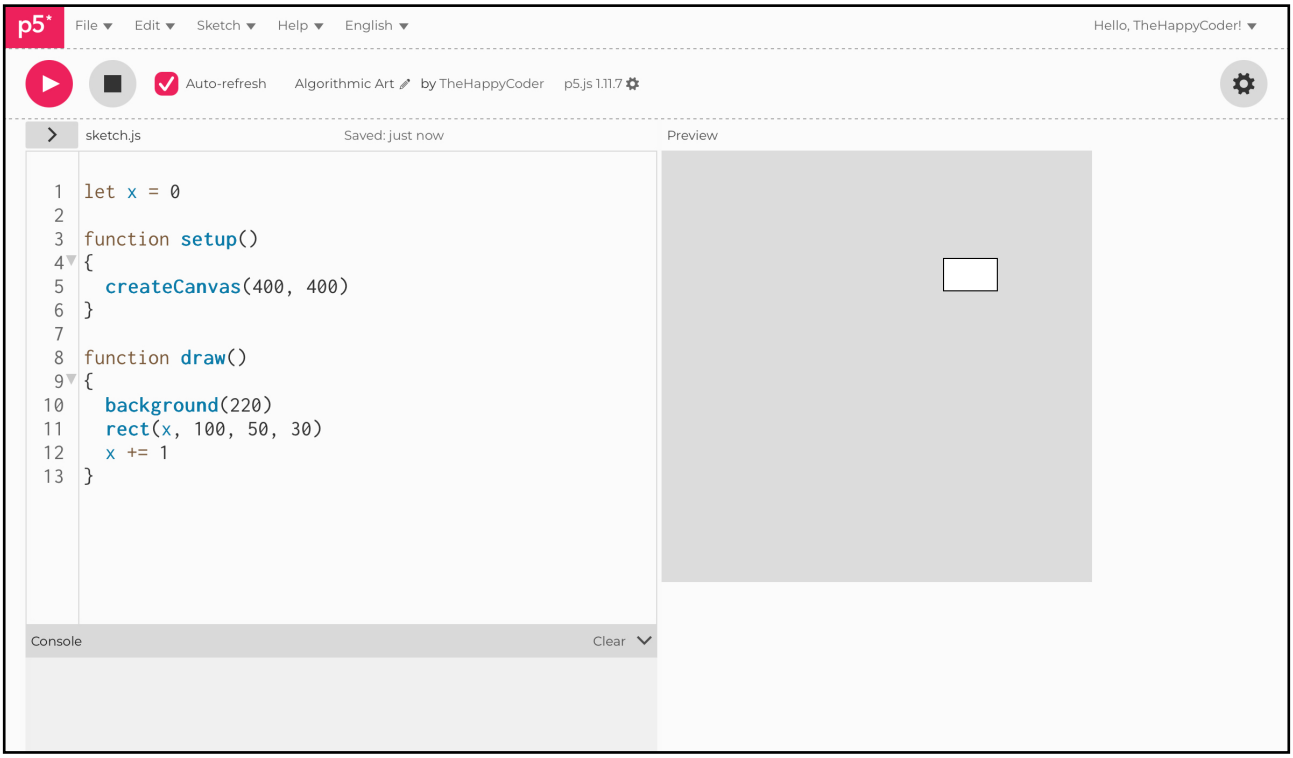
Challenge

Make it move faster.

Code Explanation

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

Figure D1.3


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
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 D1.4 it reappears

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
  }
}
```

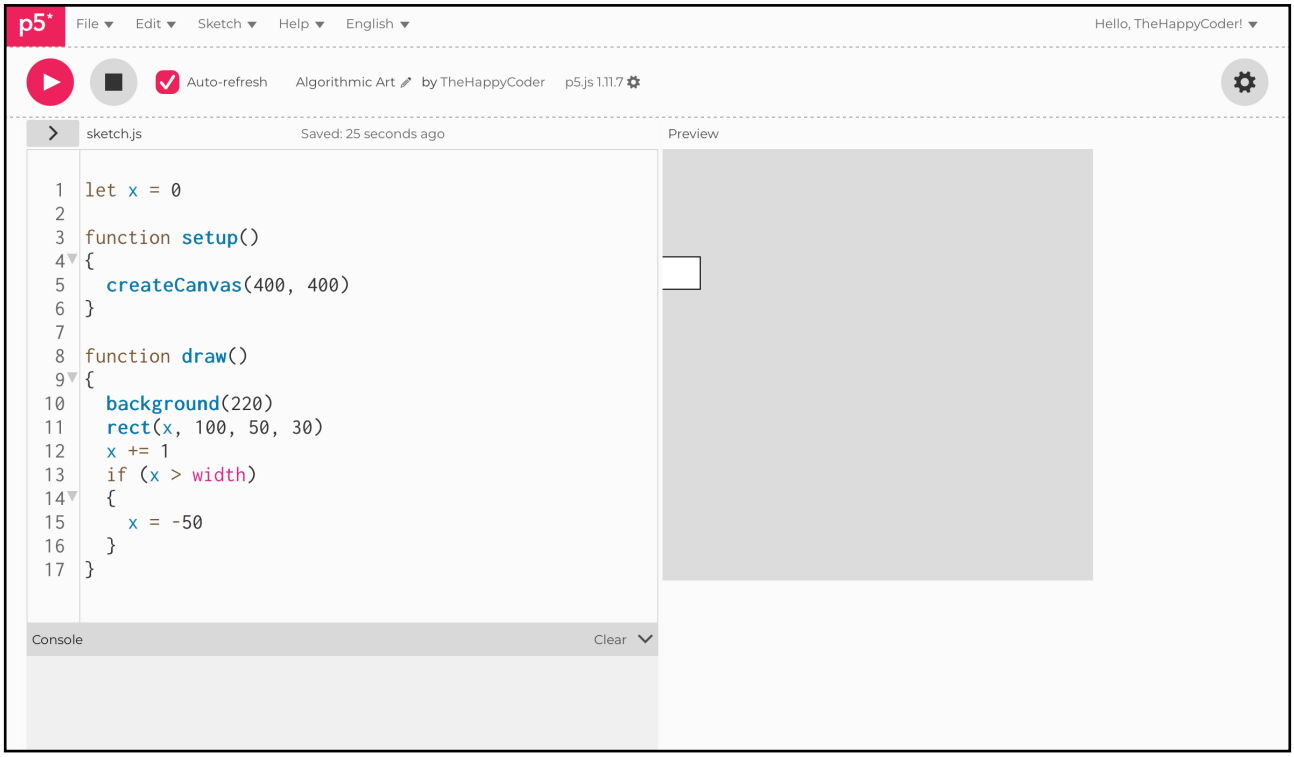
Notes

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 D1.4


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: 25 seconds ago
Preview
1 let x = 0
2
3 function setup()
4 {
5 createCanvas(400, 400)
6 }
7
8 function draw()
9 {
10 background(220)
11 rect(x, 100, 50, 30)
12 x += 1
13 if (x > width)
14 {
15 x = -50
16 }
17 }
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 D1.5 function single car

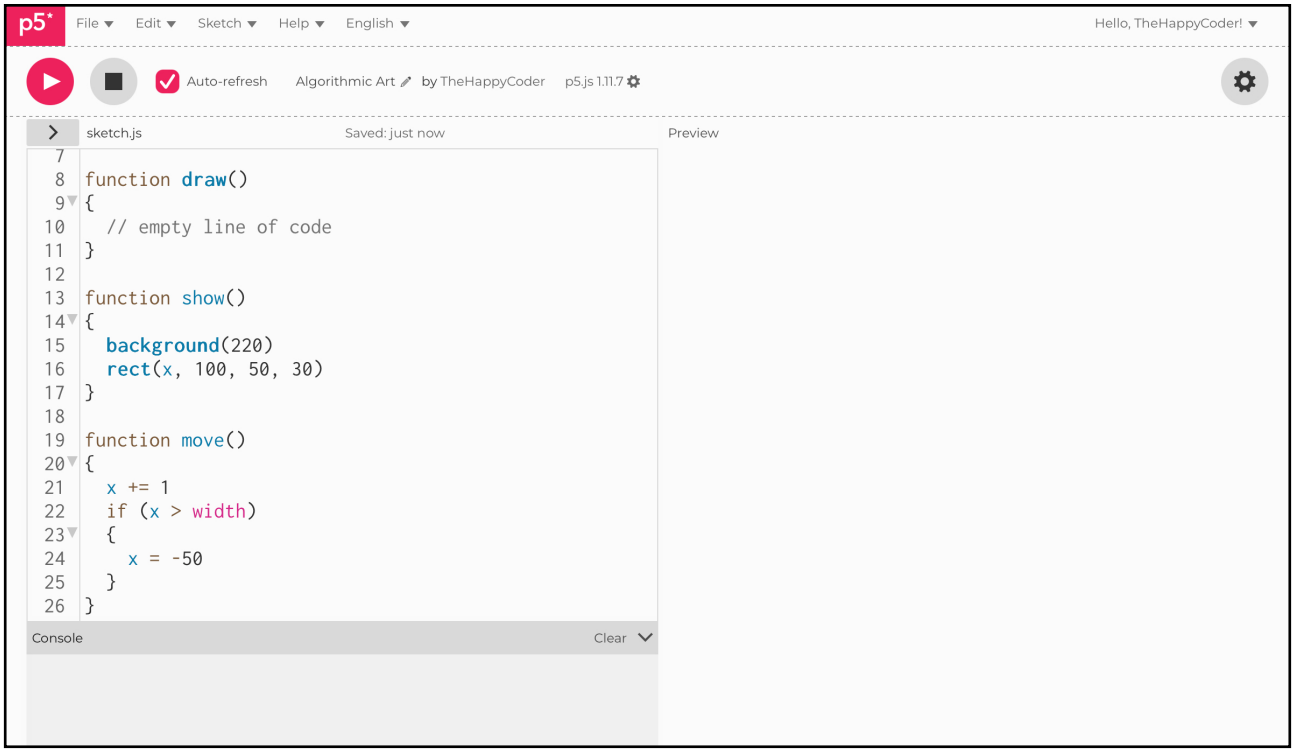
We have moved the code that was in the `draw()` function and split it between the two new functions: `show()` and `move()`. We have used the same code as in the previous sketch, just rearranged it.

```
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

Nothing to see. You can just cut and paste to save time; however, you will notice that you get nothing, not even a canvas.

Figure D1.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: just now
Preview
7
8 function draw()
9 {
10 // empty line of code
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 }
26 }
Console
Clear

Sketch D1.6 function single car

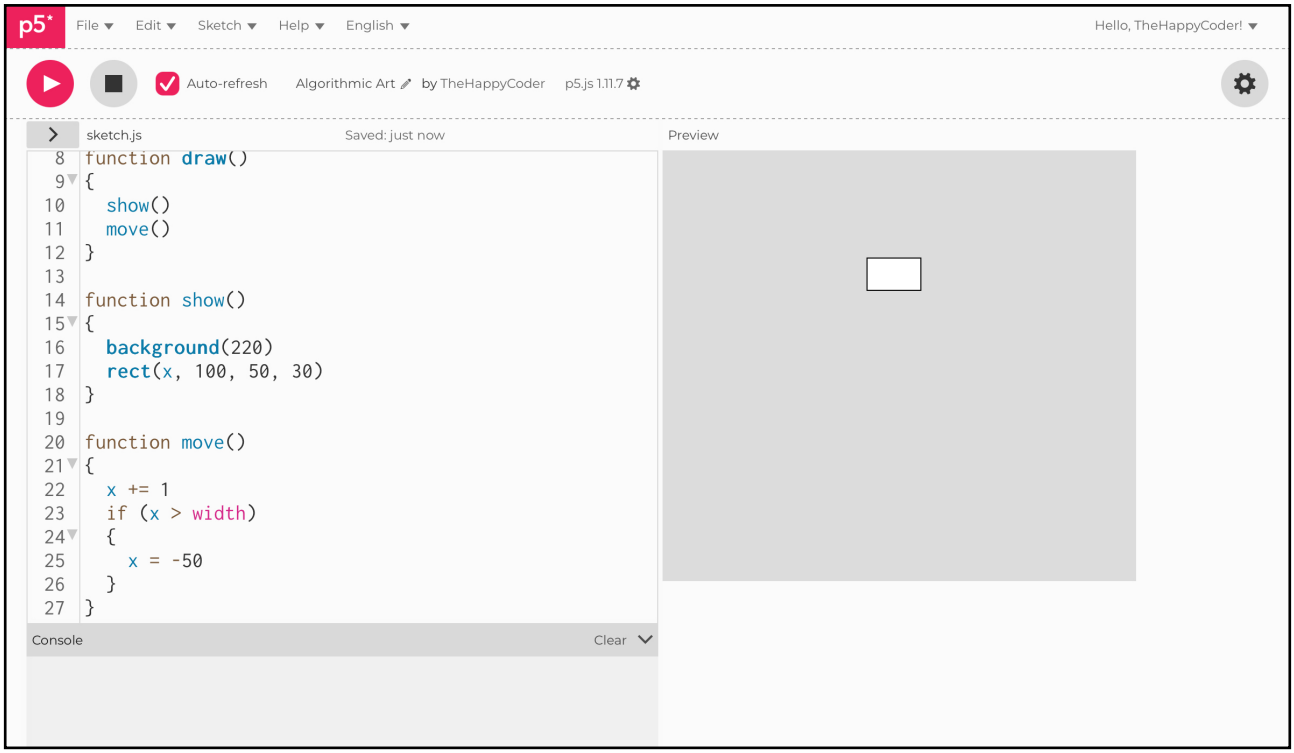
To get the two new functions to do anything, we need to call them from inside the `draw()` function, and we do it as shown below.

```
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 D1.6


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: just now
Preview
8 function draw()
9 {
10 show()
11 move()
12 }
13
14 function show()
15 {
16 background(220)
17 rect(x, 100, 50, 30)
18 }
19
20 function move()
21 {
22 x += 1
23 if (x > width)
24 {
25 x = -50
26 }
27 }
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 D1.7 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. In this case, we call it `car`.

```
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. Look at the code carefully and see how you code the car as an object rather than just as a rectangle. Below is a more detailed explanation of the code; it isn't as scary as it might look.

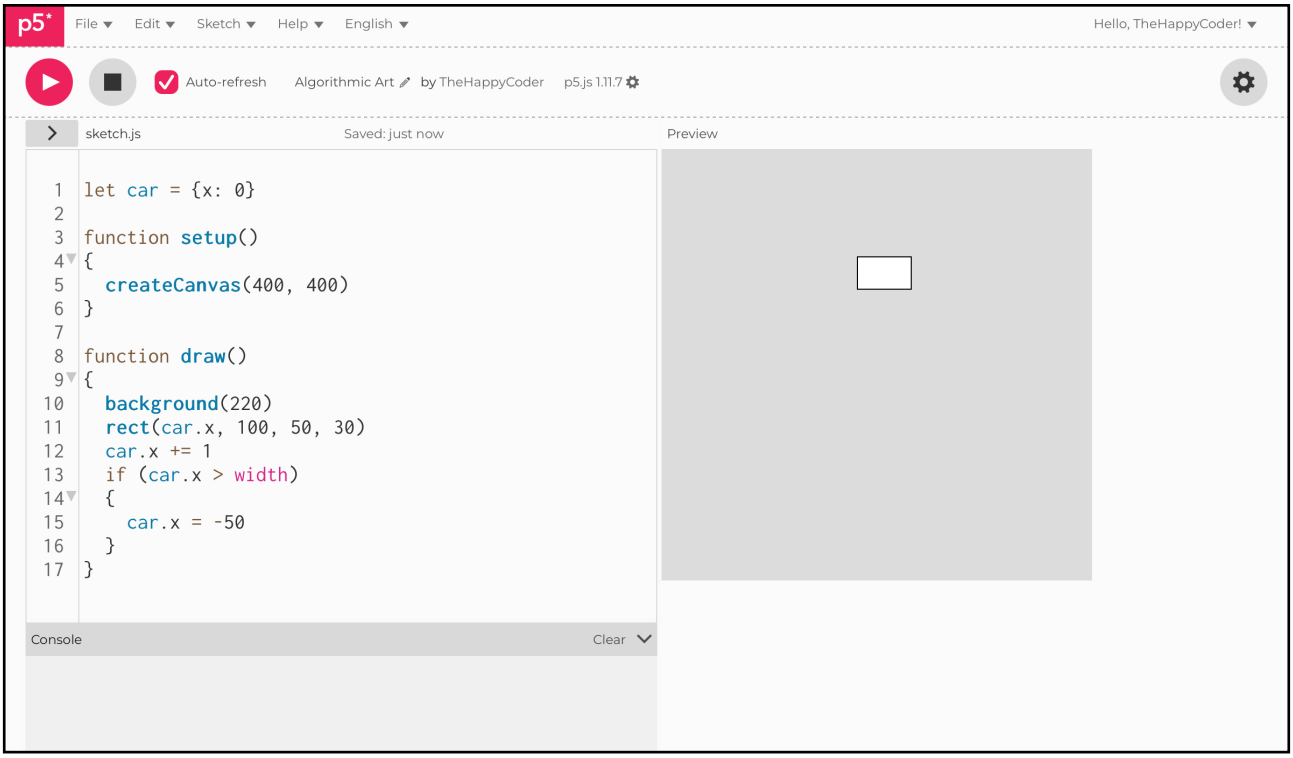
🌻 Challenges

1. Give it a `y` component
2. Introduce the `show()` and `move()` functions like we did previously (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 D1.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
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 D1.8 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
  }
}
```

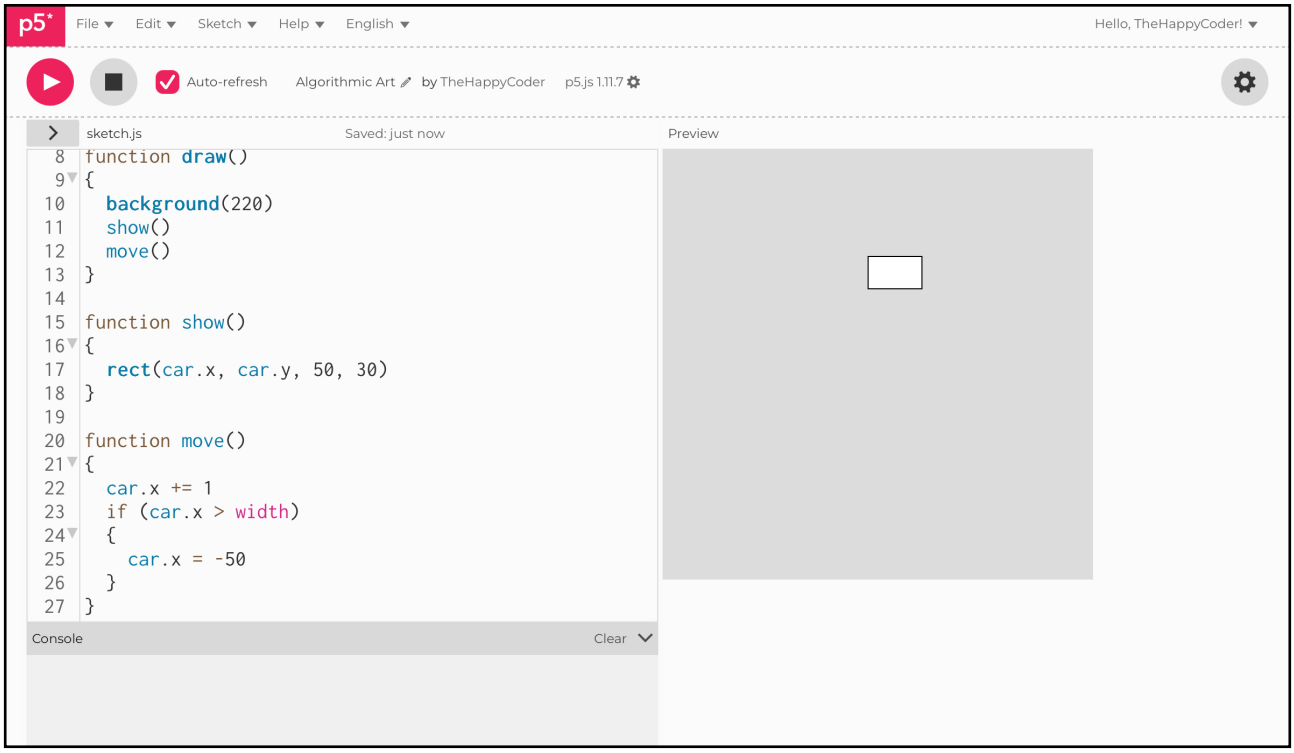
🗒️ Notes

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.

Figure D1.8


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: just now
Preview
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
}
}
Console
Clear

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 in understanding the approach.

It does take a bit of getting used to. I will try to illustrate this with a simple example. Imagine you have a template (or blueprint) 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 (`fig.1`), 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. You can see the `show()` and `move()` functions we had before, but you can have any number of functions.

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 (or blueprint) 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 `fig.2`. Where you create the car or cars from the class and call the functions from within the class.

Figure 1: class structure

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

  show()
  {
    // what it looks like
  }

  move()
  {
    // how it will move/change
  }
}
```

Figure 2: main elements in sketch

```
let car

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

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

Sketch D1.9 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
  }
}
```

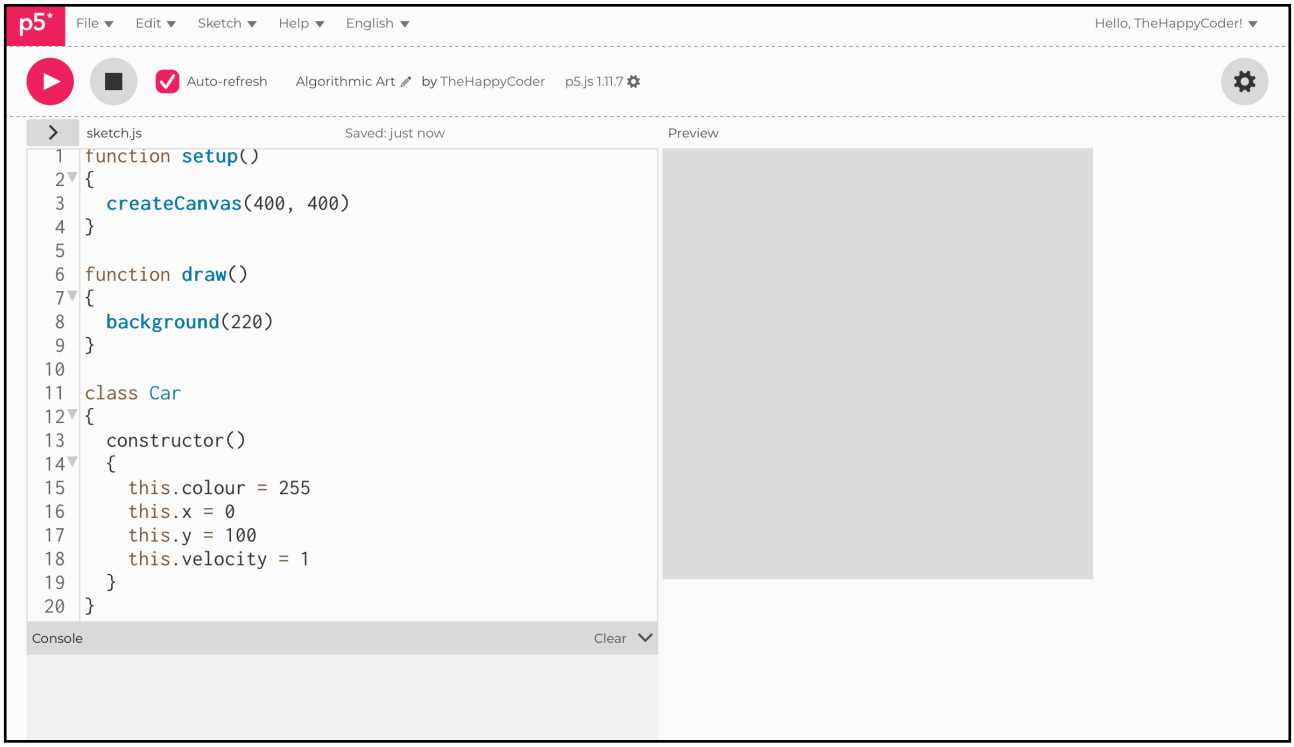
Notes

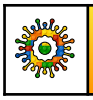
When we give attributes to an object in a class, we always use `this.` before the attribute. There is nothing to see at this point.

🛠️ 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

Figure D1.9


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
}
class Car
{
constructor()
{
this.colour = 255
this.x = 0
this.y = 100
this.velocity = 1
}
}
Console
Clear



Sketch D1.10 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)
  }
}
```

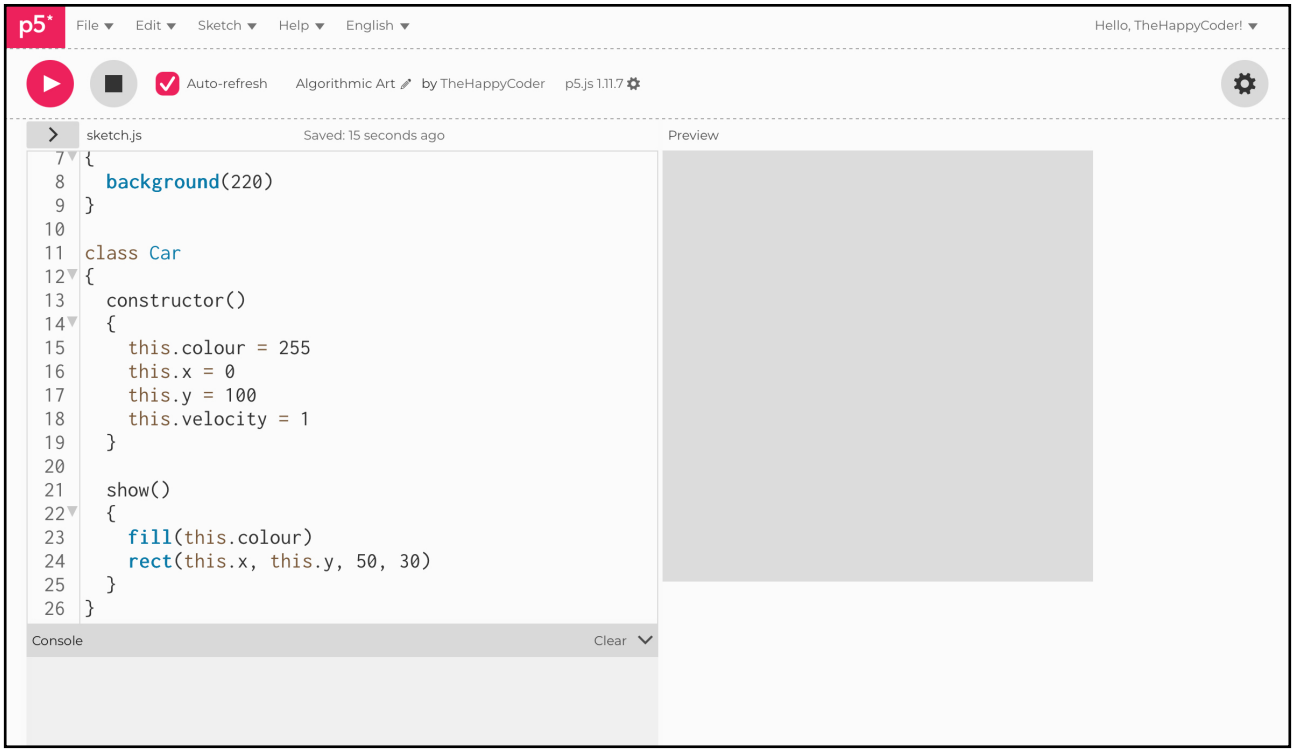
🗒️ Notes

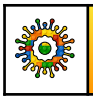
It pulls the information from the `constructor()` function. Still nothing to see yet.

🛠️ 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)

Figure D1.10


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: 15 seconds ago
Preview
{
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)
}
}
Console
Clear



Sketch D1.11 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
    }
  }
}
```

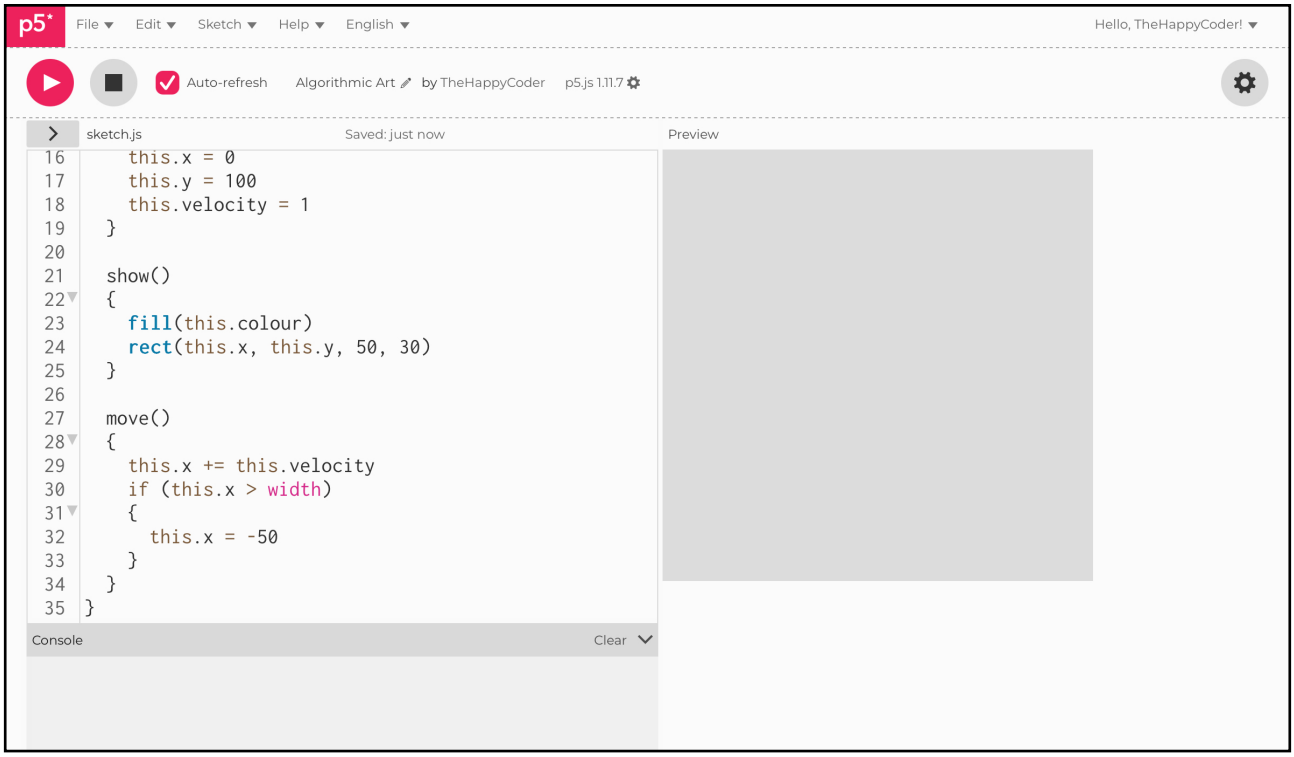
🗒️ Notes

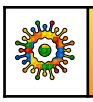
This is exactly the same as with the previous examples of a moving car. However, we created a variable for the `velocity` rather than just having a value (`1`). This allows us to alter it later. As before, still nothing to see.

🛠️ 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

Figure D1.11


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: just now
Preview
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
}
}
}
Console
Clear



Sketch D1.12 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
    }
  }
}
```

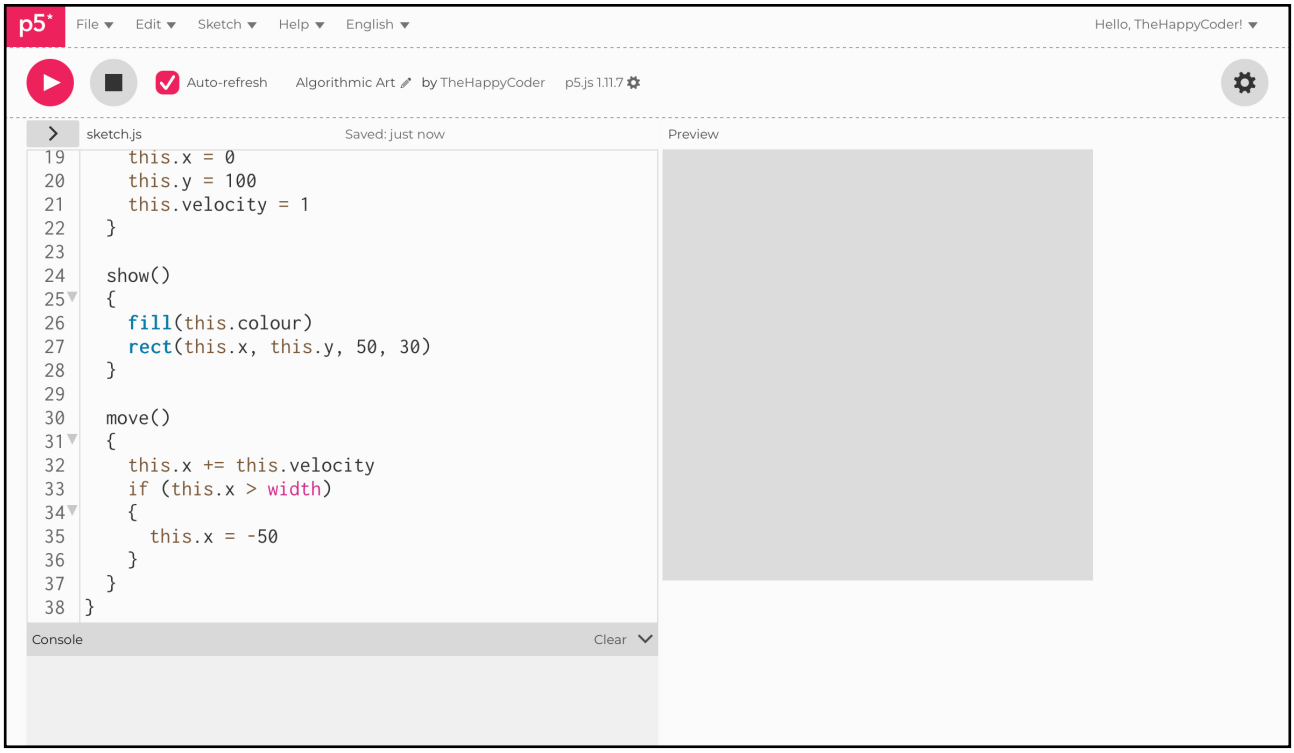
🗒️ Notes

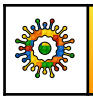
Be aware that the variable name for the car is a lowercase `c`, and the name of the class is an uppercase `C`. They both have the same name, which I admit is a little confusing, but they are totally separate entities. One is a variable name, the other is a class name. Still nothing to see here.

🛠️ Code Explanation

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

Figure D1.12


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: just now
Preview
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
}
}
}
Console
Clear



Sketch D1.13 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.

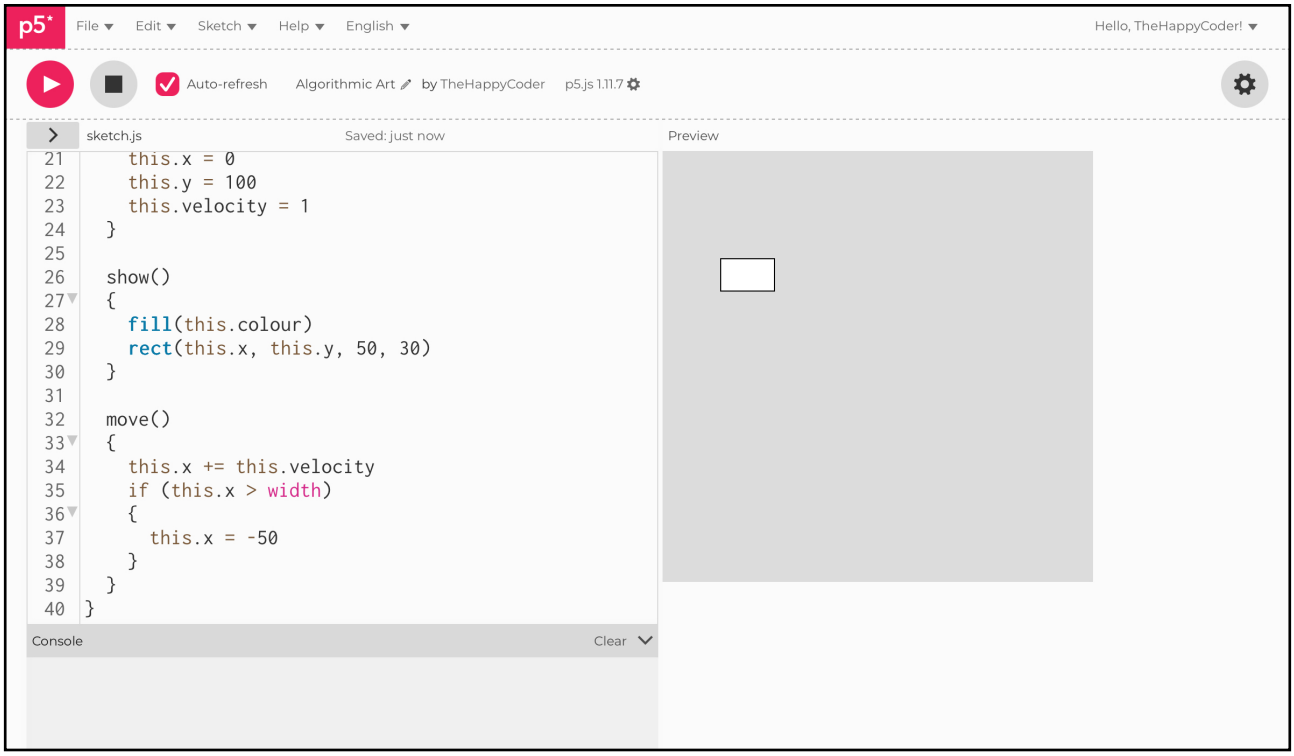
🌻 Challenges

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

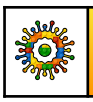
`car.show()`	For this car we show it according to the show() function
`car.move()`	For this car we move it according to the move() function

Figure D1.13


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: just now
Preview
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
}
}
}
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 hundreds of them, where each one can be created separately, independently.



Sketch D1.14 car attributes

When we create the car, we can specify its attributes rather than hard-code them in the `constructor()` function. We have given 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 that we can create a second (or more) car with different features.

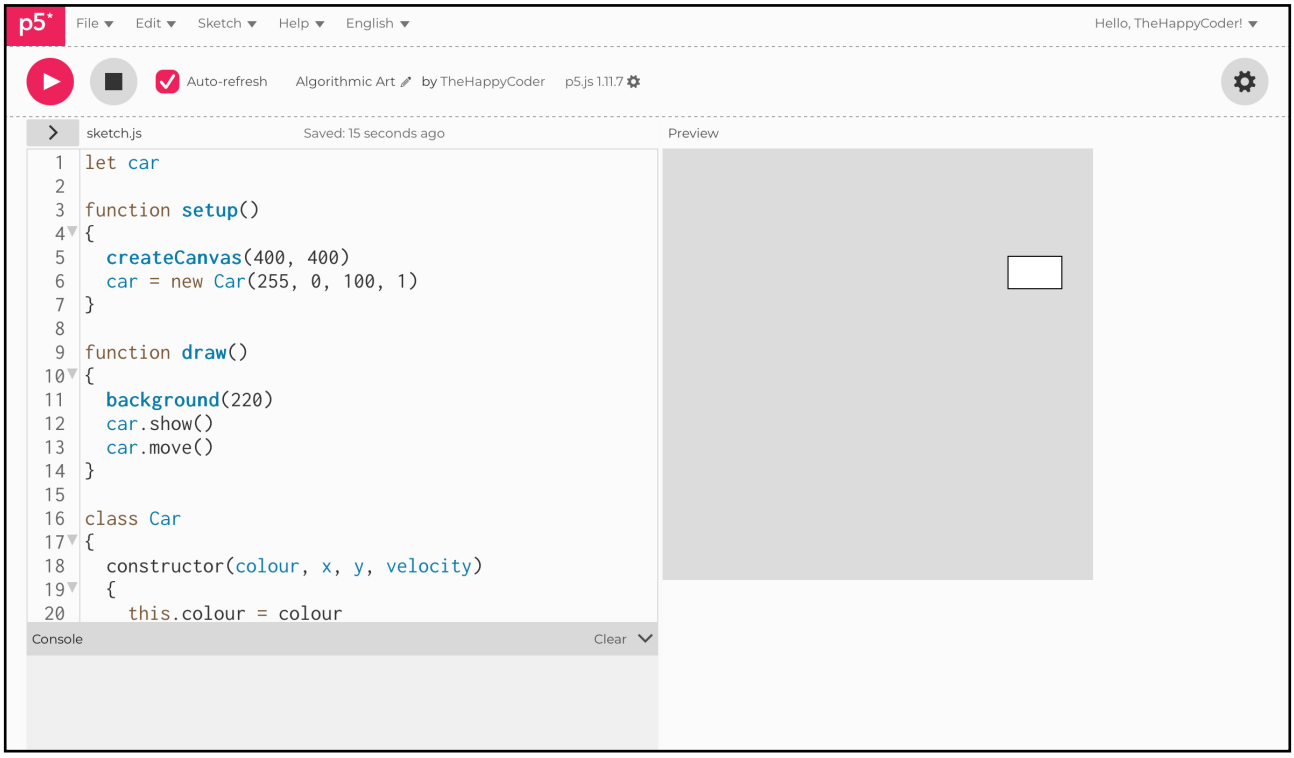
🌻 Challenge

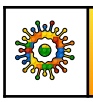
Change the values/features of the car.

🛠️ 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 D1.14


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: 15 seconds 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
{
constructor(colour, x, y, velocity)
{
this.colour = colour
Console
Clear



Sketch D1.15 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
    }
  }
}
```

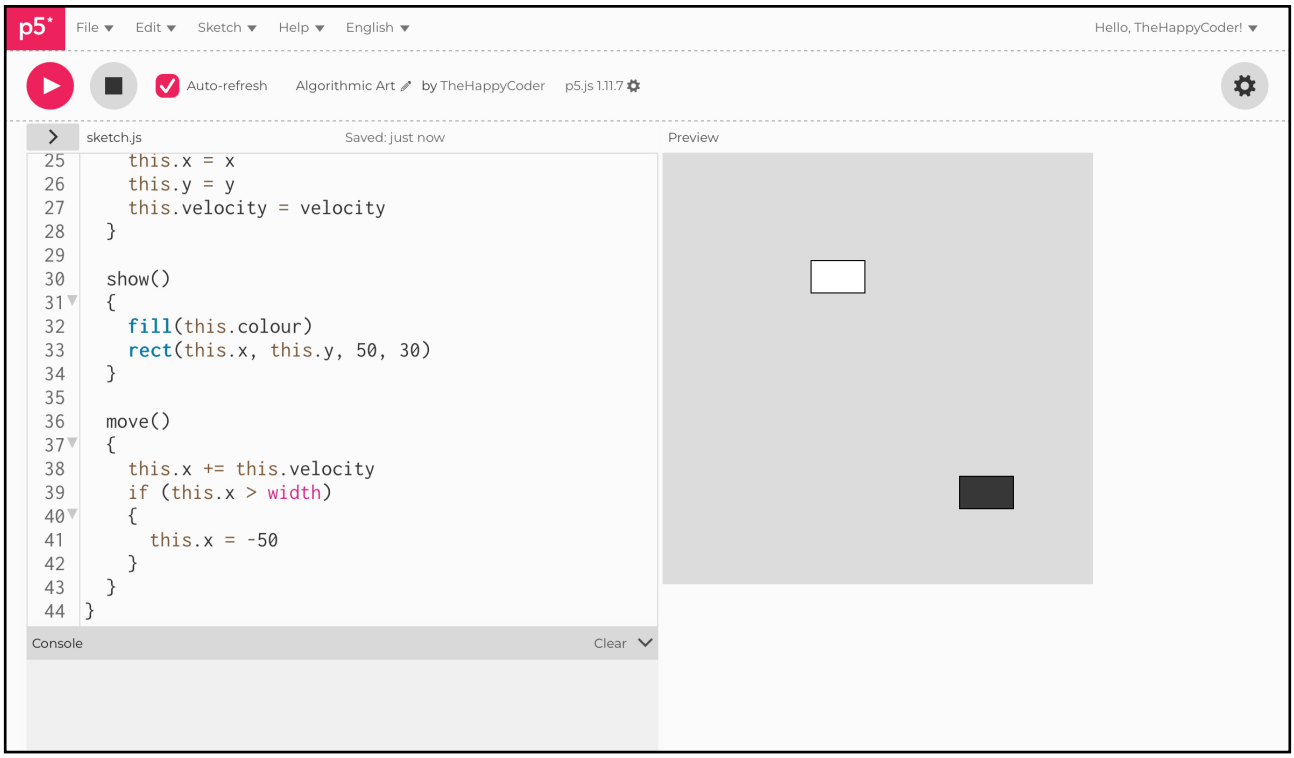
🗒️ Notes

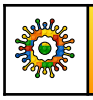
With just a few lines of code, we have created a second car. You can see the simple logic.

🌻 Challenge

Add a third car.

Figure D1.15


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: just now
Preview
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
}
}
}
Console
Clear



Sketch D1.16 lots and lots of cars

Here is a quick peek 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()` functions; 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
    }
  }
}
```

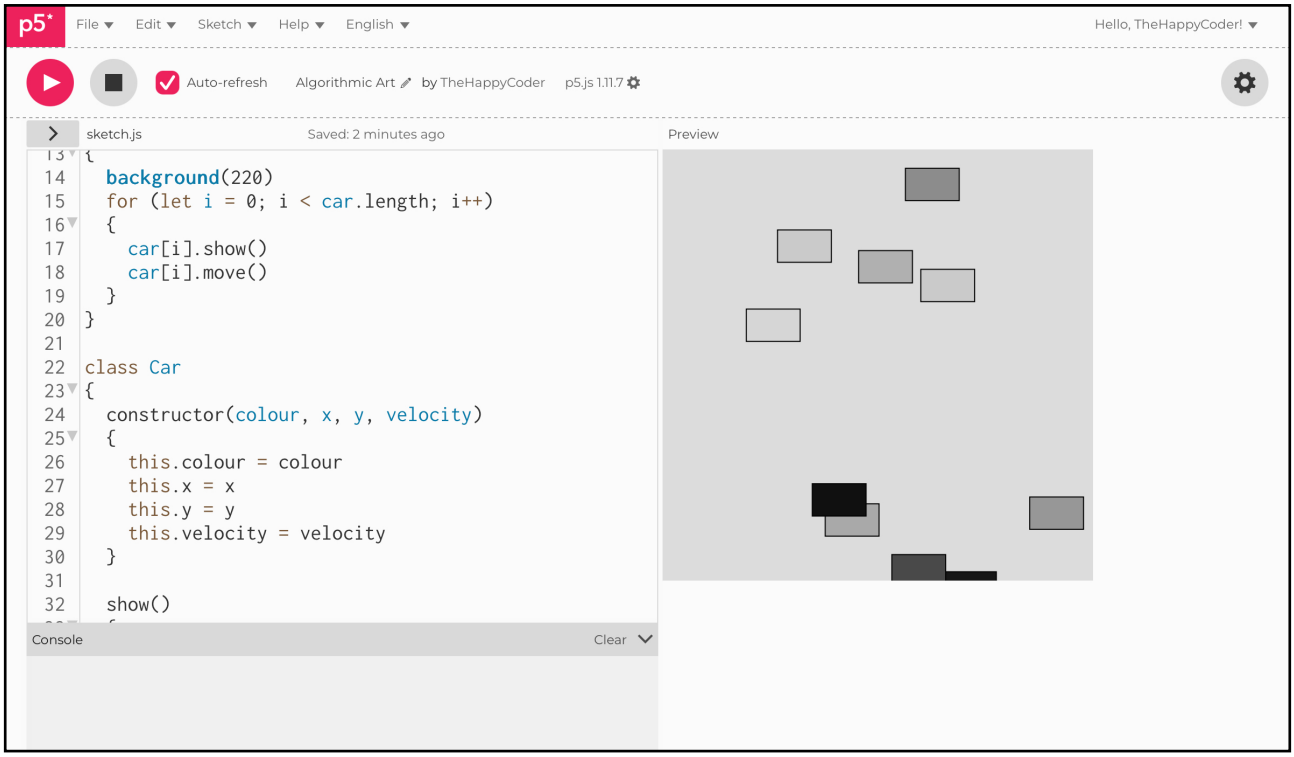
🗒️ Notes

I think that is pretty elegant!

🌻 Challenge

Just have a play.

Figure D1.16


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.7
sketch.js
Saved: 2 minutes ago
Preview
{
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()
Console
Clear