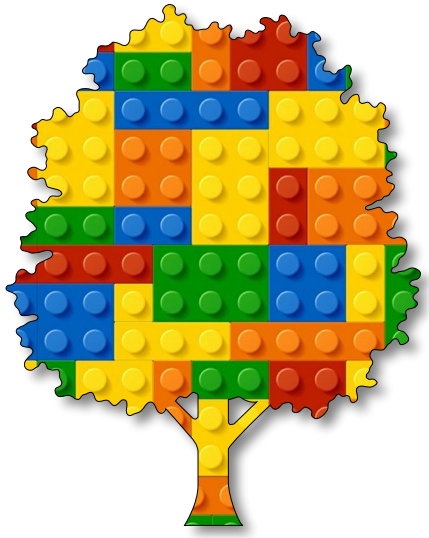
Algorithmic Art

Module D
Unit #2
in another class



Content

Module D Unit #2 in another class

Introduction to another class

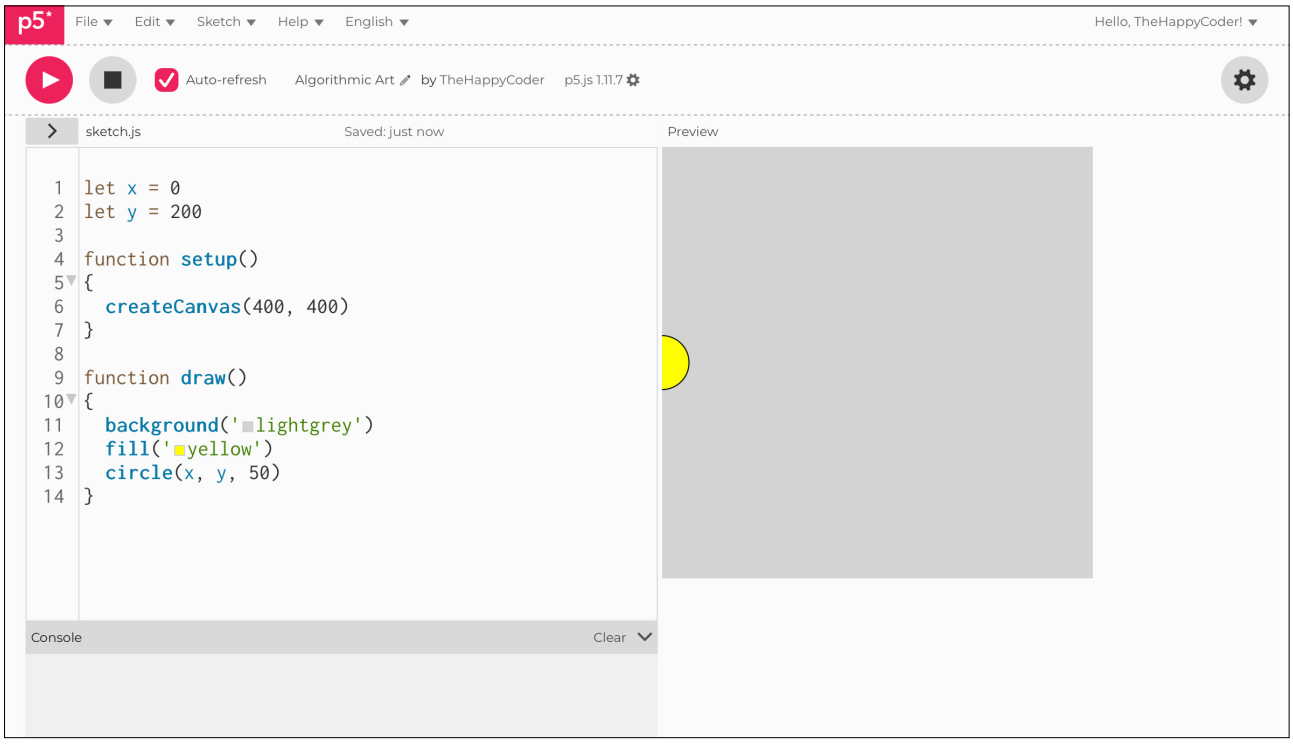
This is another simple example of using classes to create something. The previous unit introduced classes as well as comparing them with functions and objects. Here we add a few more features to create an explosion of balls cascading or raining down. We will draw on this when we create a fireworks display.

Sketch D2.1 starting again

! starting a new sketch.
A circle (ball) at (x, y) coordinates.

```
let x = 0
let y = 200

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

function draw()
{
  background('lightgrey')
  fill('yellow')
  circle(x, y, 50)
}
```

Figure D2.1



Sketch D2.2 a single ball

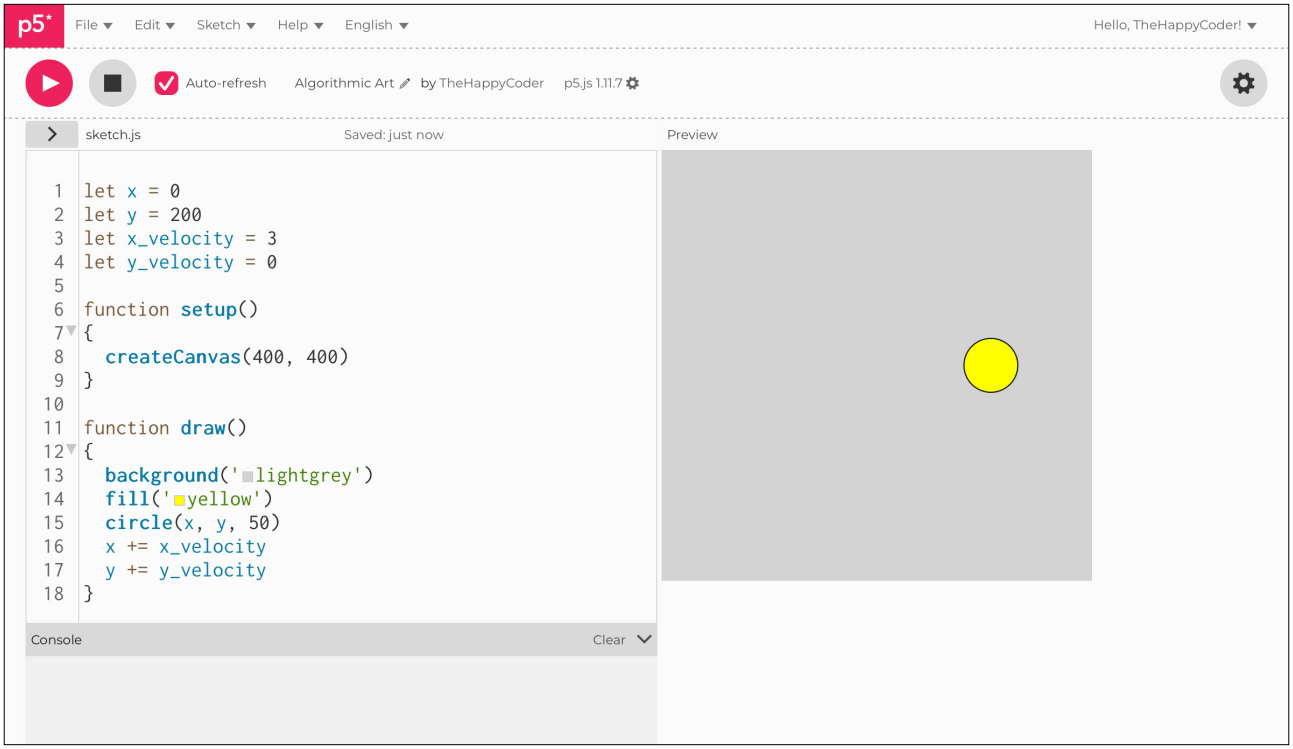
Now a ball moving across the canvas with a x velocity of 3 and a y velocity of 0.

```
let x = 0
let y = 200
let x_velocity = 3
let y_velocity = 0

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

function draw()
{
  background('lightgrey')
  fill('yellow')
  circle(x, y, 50)
  x += x_velocity
  y += y_velocity
}
```

Notes

Another naming convention is to use an underscore if splitting workday and letters in a variable name.

Figure D2.2


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
let y = 200
let x_velocity = 3
let y_velocity = 0
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background('lightgrey')
fill('yellow')
circle(x, y, 50)
x += x_velocity
y += y_velocity
}
Console
Clear

Sketch D2.3 adding a bit of gravity

We give the y_velocity a bit of movement upwards (hence the negative value), but by adding a bit of gravity, it causes the ball to fall downwards to the ground after a while.

```
let x = 0
let y = 200
let x_velocity = 3
let y_velocity = -3
let gravity = 0.07

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

function draw()
{
  background('lightgrey')
  fill('yellow')
  circle(x, y, 50)
  x += x_velocity
  y += y_velocity
  y_velocity += gravity
}
```

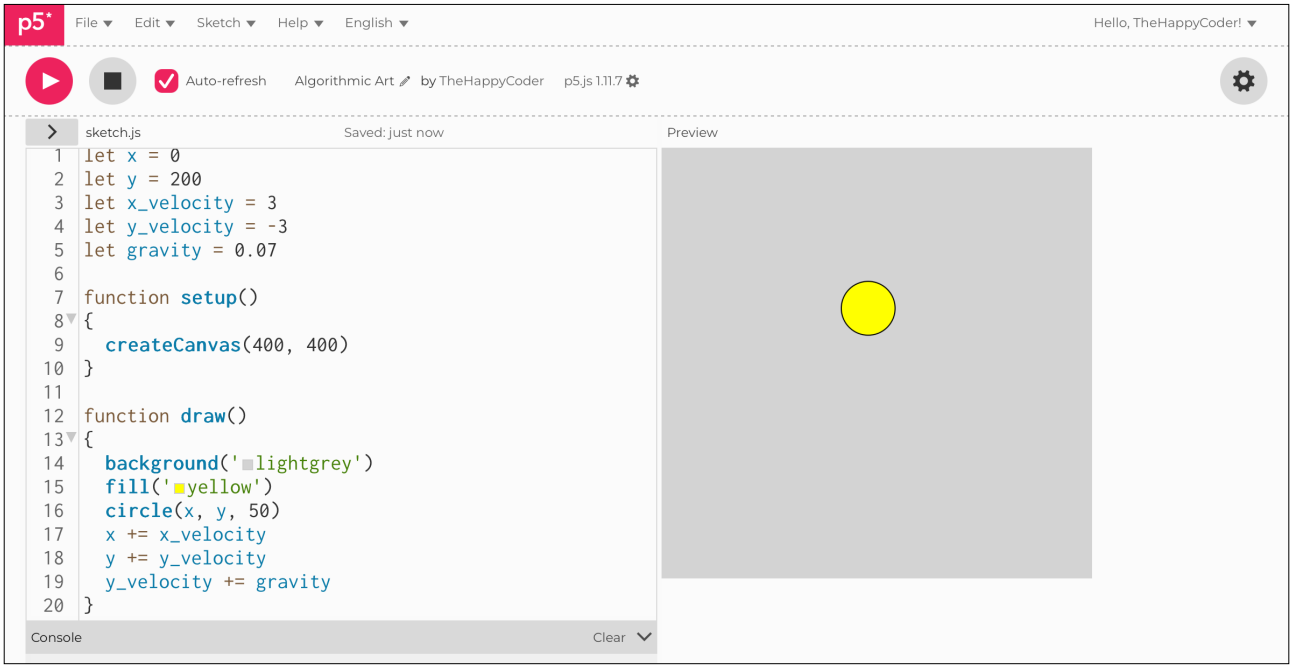
🗒️ Notes

The ball goes up briefly and then starts to fall downwards once the velocity becomes positive.

🌻 Challenges

1. Try different y velocities and gravity values.
2. Wind could be a variable in the x direction.

Figure D2.3


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
let y = 200
let x_velocity = 3
let y_velocity = -3
let gravity = 0.07
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background('lightgrey')
fill('yellow')
circle(x, y, 50)
x += x_velocity
y += y_velocity
y_velocity += gravity
}
Console
Clear

Sketch D2.4 creating a ball class

We have just built the basic structure for the class. We have our `constructor()` function; we will also need a `show()` function and a `move()` function. I want to emphasise now that the names of these functions are made up; you can call each one of them whatever names you want. It is convention to have a constructor function, but the others can have other names and often do.

```
let x = 0
let y = 200
let x_velocity = 3
let y_velocity = -3
let gravity = 0.07

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

function draw()
{
  background('lightgrey')
  fill('yellow')
  circle(x, y, 50)
  x += x_velocity
  y += y_velocity
  y_velocity += gravity
}
```

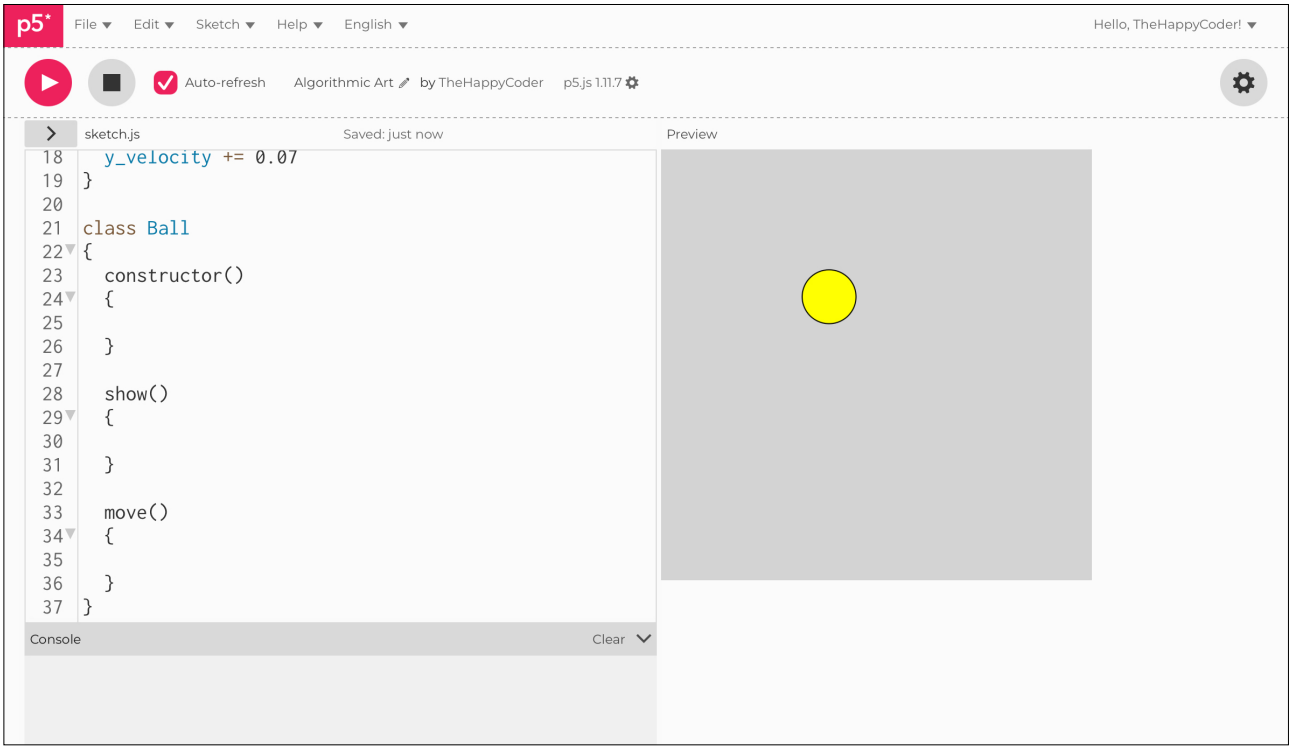
```
class Ball
{
  constructor()
  {

  }

  show()
  {
```

```
  }

  move()
  {

  }
}
```

🗒️ Notes

Nothing new will happen.

Figure D2.4



Sketch D2.5 the constructor() function

❗ remove the commented lines of code.
First, we will fill in the `constructor()` function using the variables highlighted. At the moment, you will get an error message if you try to run it now.

```
// let x = 0
// let y = 200
// let x_velocity = 3
// let y_velocity = -3
// let gravity = 0.07

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

function draw()
{
  background('lightgrey')
  fill('yellow')
  circle(x, y, 50)
  x += x_velocity
  y += y_velocity
  y_velocity += gravity
}

class Ball
{
  constructor()
  {
    this.x = 0
    this.y = 200
    this.x_velocity = 3
    this.y_velocity = -3
    this.gravity = 0.07
  }
```

```
  show()
  {

  }

  move()
  {

  }
}
```

🗒️ Notes

The this._____ means it is referring to a specific ball, not necessarily to all of them. You should get an error message re: the x variable.

Sketch D2.6 the show() function

❗ remove commented out lines of code.
We can now fill in the show() function.

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

function draw()
{
  background('lightgrey')
  // fill('yellow')
  // circle(x, y, 50)
  x += x_velocity
  y += y_velocity
  y_velocity += gravity
}

class Ball
{
  constructor()
  {
    this.x = 0
    this.y = 200
    this.x_velocity = 3
    this.y_velocity = -3
    this.gravity = 0.07
  }

  show()
  {
    fill('yellow')
    circle(this.x, this.y, 50)
  }

  move()
```

```
  {


  }
}
```

🗒️ Notes

The `x` and `y` coordinates become `this.x` and `this.y` for each ball we create. Still an error message, but don't worry.

Sketch D2.7 the move() function

❗ remove the commented lines of code.
And finally, the `move()` function.

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

function draw()
{
  background('lightgrey')
  // x += x_velocity
  // y += y_velocity
  // y_velocity += gravity
}

class Ball
{
  constructor()
  {
    this.x = 0
    this.y = 200
    this.x_velocity = 3
    this.y_velocity = -3
    this.gravity = 0.07
  }

  show()
  {
    fill('yellow')
    circle(this.x, this.y, 50)
  }

  move()
  {
    this.x += this.x_velocity
```

```
    this.y += this.y_velocity
    this.y_velocity += this.gravity
  }
}
```

🗒️ Notes

You shouldn't get an error message now, but neither will you see anything. We need to create the ball next and call those functions.

Sketch D2.8 can we have our ball back, please

We have finally got our ball back, phew!

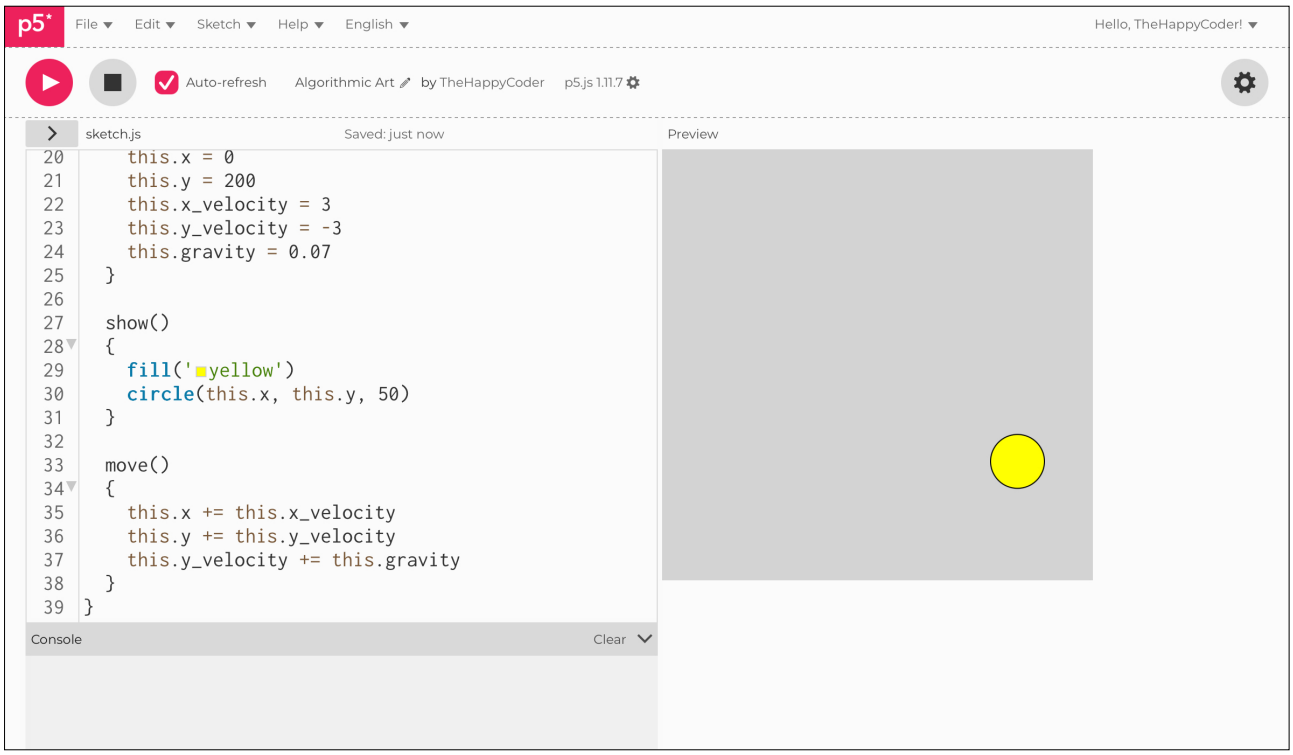
```
let ball

function setup()
{
  createCanvas(400, 400)
  ball = new Ball()
}

function draw()
{
  background('lightgrey')
  ball.show()
  ball.move()
}

class Ball
{
  constructor()
  {
    this.x = 0
    this.y = 200
    this.x_velocity = 3
    this.y_velocity = -3
    this.gravity = 0.07
  }

  show()
  {
    fill('yellow')
    circle(this.x, this.y, 50)
  }

  move()
  {
```

```
    this.x += this.x_velocity
    this.y += this.y_velocity
    this.y_velocity += this.gravity
  }
}
```

🗒️ Notes

Everything should now be as before if all is well.

Figure D2.8



Sketch D2.9 many balls

❗ Remove and replace `ball = new Ball()` with the `for()` loop.
We can have more than one ball, so let's have `20` and for that we create an array to store them by pushing each new ball into that array, warning you that you will get an error message.

```
let ball
let balls = []
let number = 20

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < number; i++)
  {
    balls.push(new Ball())
  }
}

function draw()
{
  background('lightgrey')
  ball.show()
  ball.move()
}

class Ball
{
  constructor()
  {
    this.x = 0
    this.y = 200
    this.x_velocity = 3
    this.y_velocity = -3
    this.gravity = 0.07
  }

  show()
```

```
  {
    fill('yellow')
    circle(this.x, this.y, 50)
  }

  move()
  {
    this.x += this.x_velocity
    this.y += this.y_velocity
    this.y_velocity += this.gravity
  }
}
```

🗒️ Notes

We have broken the sketch, but have no fear, we will fix that. Error message!

Sketch D2.10 pulling the balls out

We need to pull each of the balls from the array and `show()` and `move()` each one. We use the `for...of` loop.

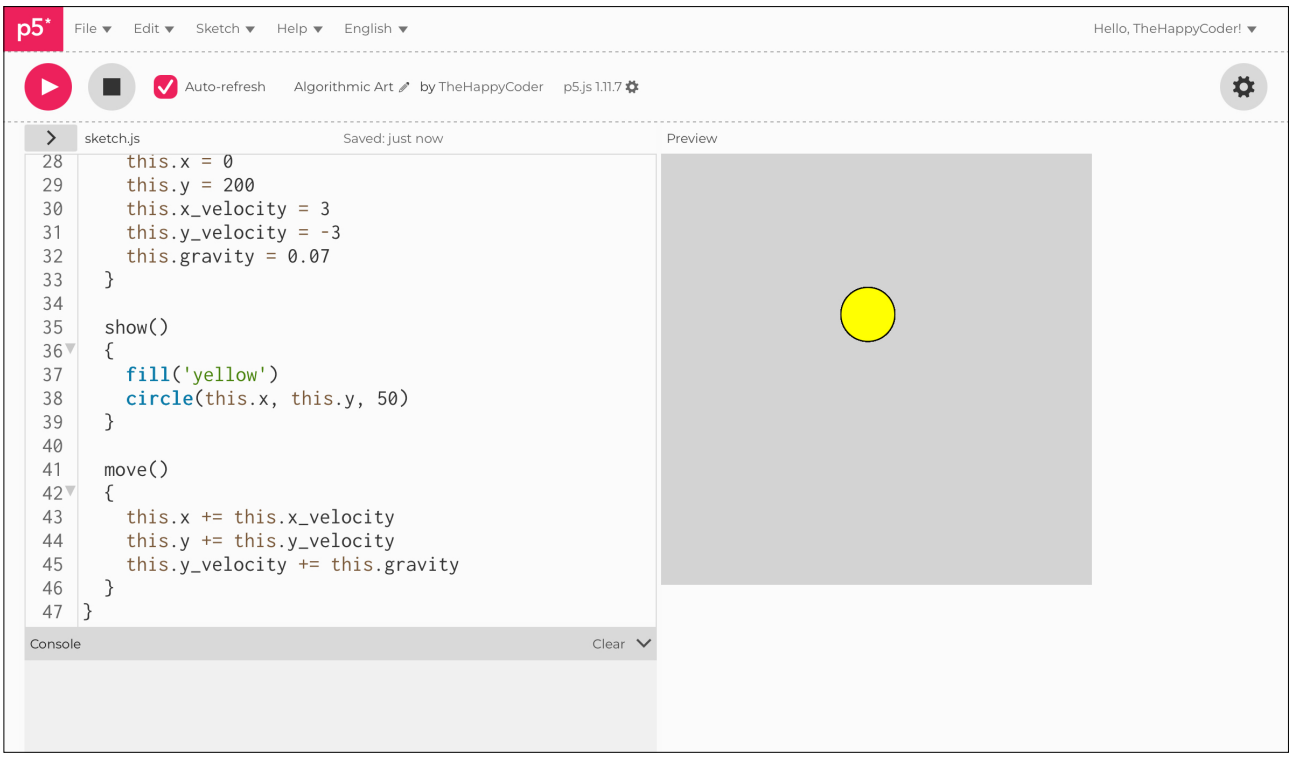
```
let ball
let balls = []
let number = 20

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < number; i++)
  {
    balls.push(new Ball())
  }
}

function draw()
{
  background('lightgrey')
  for (ball of balls)
  {
    ball.show()
    ball.move()
  }
}

class Ball
{
  constructor()
  {
    this.x = 0
    this.y = 200
    this.x_velocity = 3
    this.y_velocity = -3
    this.gravity = 0.07
  }
```

```
  show()
  {
    fill('yellow')
    circle(this.x, this.y, 50)
  }

  move()
  {
    this.x += this.x_velocity
    this.y += this.y_velocity
    this.y_velocity += this.gravity
  }
}
```

🗒️ Notes

We have fixed the sketch, but there were supposed to be 20 balls, not just one! To see them, we need to mix them up a bit; they are occupying the same space.

Figure D2.10



Sketch D2.11 random velocity

To mix it up, we can introduce some randomness to their velocities for both x and y.

```
let ball
let balls = []
let number = 20

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < number; i++)
  {
    balls.push(new Ball())
  }
}

function draw()
{
  background('lightgrey')
  for (ball of balls)
  {
    ball.show()
    ball.move()
  }
}

class Ball
{
  constructor()
  {
    this.x = 0
    this.y = 200
    this.x_velocity = 3 * random(1, 2)
    this.y_velocity = -3 * random(-1, 1)
    this.gravity = 0.07
  }
```

```
  show()
  {
    fill('yellow')
    circle(this.x, this.y, 50)
  }

  move()
  {
    this.x += this.x_velocity
    this.y += this.y_velocity
    this.y_velocity += this.gravity
  }
}
```

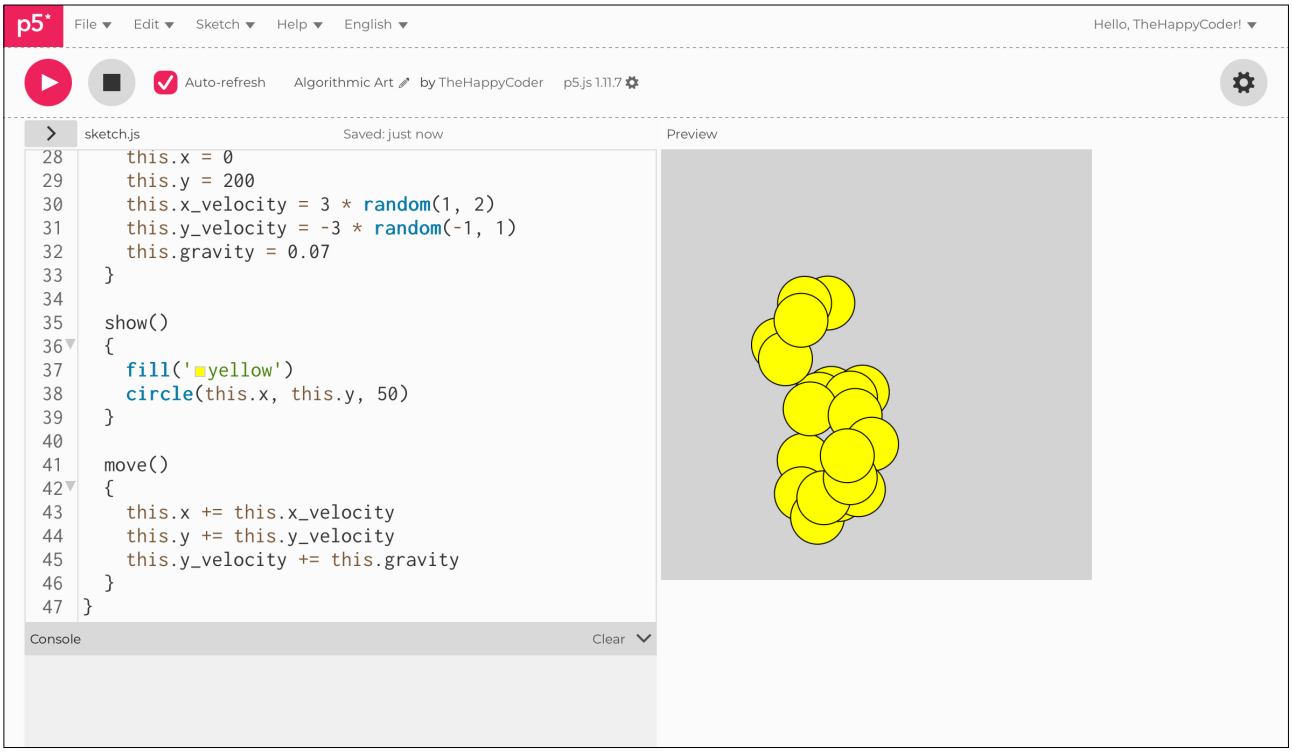
🗒️ Notes

They can now wander off in different directions.

🌻 Challenge

Try other random values.

Figure D2.11



Sketch D2.12 refactoring

A bit of refactoring for a slightly different effect.

```
let ball
let balls = []
let number = 200

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < number; i++)
  {
    balls.push(new Ball())
  }
}

function draw()
{
  background('lightgrey')
  for (ball of balls)
  {
    ball.show()
    ball.move()
  }
}

class Ball
{
  constructor()
  {
    this.x = 200
    this.y = 200
    this.x_velocity = 3 * random(-1, 1)
    this.y_velocity = -3 * random(-1, 2)
    this.gravity = 0.07
  }
```

```
  show()
  {
    fill('yellow')
    circle(this.x, this.y, 20)
  }

  move()
  {
    this.x += this.x_velocity
    this.y += this.y_velocity
    this.y_velocity += this.gravity
  }
}
```

Notes

We will have 200 smaller balls emitting from the centre; you could have something like a firework-type display.

Figure D2.12