Algorithmic Art Workbook #2

10 PRINT Patterns



Table of Contents

MIT Licence 4
Workbook #2 Quick Content Summary 5
Module A Unit #5: squares and rectangles 7
Sketch A5.1 a simple square 8
Sketch A5.2 from the centre 10
Sketch A5.3 rotate 12
Sketch A5.4 translate 14
Sketch A5.5 no fill and an angle 16
Sketch A5.6 rotating it slowly 18
Sketch A5.7 adding a second square 20
Sketch A5.8 push and pop 22
Sketch A5.9 a rectangle 24
Sketch A5.10 random rectangles 26
Module A Unit #6: ellipses and triangles 29
Sketch A6.1 drawing an ellipse 30
Sketch A6.2 mouseX and mouseY 32
Sketch A6.3 stretchy time 34
Sketch A6.4 triangle 36
Sketch A6.5 width and height 38
Sketch A6.6 rebuilding the triangle 40
Sketch A6.7 a length 42
Sketch A6.8 a bit more pointy 44
Sketch A6.9 a hundred of them 46
Sketch A6.10 push pop stop 48
Sketch A6.11 rotate the triangle 50
Sketch A6.12 split the triangles up 52
Sketch A6.13 random angles 54
Sketch A6.14 random length 56
Sketch A6.15 random colouring 58
Module A Unit #7: pixels 61
Sketch A7.1 what's the point? 62
Sketch A7.2 a line of pixels 64
Infinite Loop problem 66
Infinite Loop solution 67
Sketch A7.3 nested loop 68
Sketch A7.4 colouring the pixel 70
Sketch A7.5 random pixel colour 72
Sketch A7.6 gradual colour 74
Sketch A7.7 colour mode RGB 76

Sketch A7.8 in both directions 78
Module A Unit #8: 10PRINT 81
Sketch A8.1 our starting sketch 82
Sketch A8.2 adding some variables 83
Sketch A8.3 the line function 84
Sketch A8.4 another variable 85
Sketch A8.5 using the variable 86
Sketch A8.6 adding the spacing 88
Sketch A8.7 stop at the edge 90
Sketch A8.8 let’s go down 91
Sketch A8.9 stop when we get to the bottom 93
Sketch A8.10 sloping the other way 94
Sketch A8.11 the other line 50% 96
Sketch A8.12 variations on a theme 98
Sketch A8.13 A row of squares 100
Sketch A8.14 rows and columns 102
Sketch A8.15 random size squares 104
Sketch A8.16 using rectMode(CENTER) 106
Sketch A8.17 random colours 108



MIT Licence

Copyright ©2026 Warren George

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Brief summary:

Permissions (what you can do)

Commercial use
Modification
Distribution
Private use

Limitations (what is not covered)

Liability
Warranty



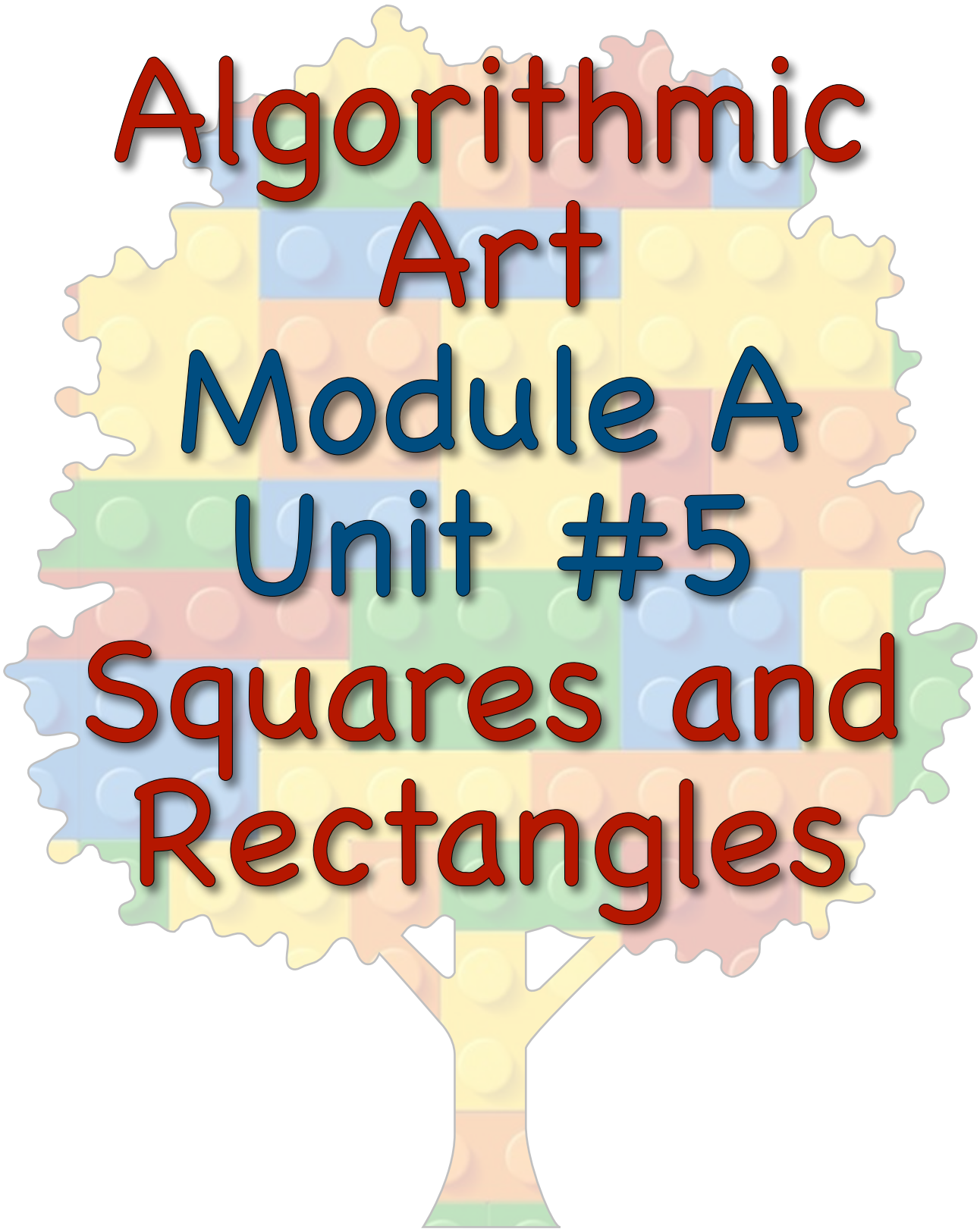
Workbook #2 Quick Content Summary

This second workbook continues where the first one left off. In this one, we explore some more simple shapes: squares, rectangles, ellipses, triangles, and a venture into pixels (points).

The final unit is an exercise putting what we have learned into practice. It is a pattern generator familiar to those who were coding in the late 60s or early 70s, where the computers were very basic indeed. Even though it is very simple, it is quite elegant.

I would encourage you to explore these shapes and ideas further for yourself. See what you can create and what works, and more importantly, what doesn't. I cannot stress enough the importance of learning through exploring. Don't just copy the code verbatim (although that is also quite good); change it, adapt it, try ideas out, and see what happens.

Memory muscle is also important. It seems very repetitive when doing the same thing over and over again, but it builds memory muscle, just like when learning to play a musical instrument. Aim to do some coding every day; the more, the better.



Algorithmic Art
Module A
Unit #5
Squares and Rectangles



Module A Unit #5: squares and rectangles

There is a lot to pack into this unit. Here we introduce the square and rectangle and show how we can manipulate these shapes, rotating and translating them. I will be introducing a lot of new concepts to you, but they will be used a lot in the units and examples to come. Just plough your way through the sketches and play with the code; this approach will help you understand what it is doing. I will give you the framework, but it is up to you to explore.

Key concepts:

```
square()
rotate()
angleMode()
translate()
push()
pop()
rectMode()
rectangle()
comments
```

Sketch A5.1 a simple square

! Our starting sketch

We can draw a simple square in the middle of the canvas. The `square()` function has three arguments: the `x` co-ordinate, the `y` co-ordinate and the `length` of the sides.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  square(200, 200, 100)
}
```

Notes

The only problem is, it isn't in the centre of the canvas. This is because the co-ordinates of the square are taken from the top left-hand corner, not the centre of the square, but we can fix that quite easily.

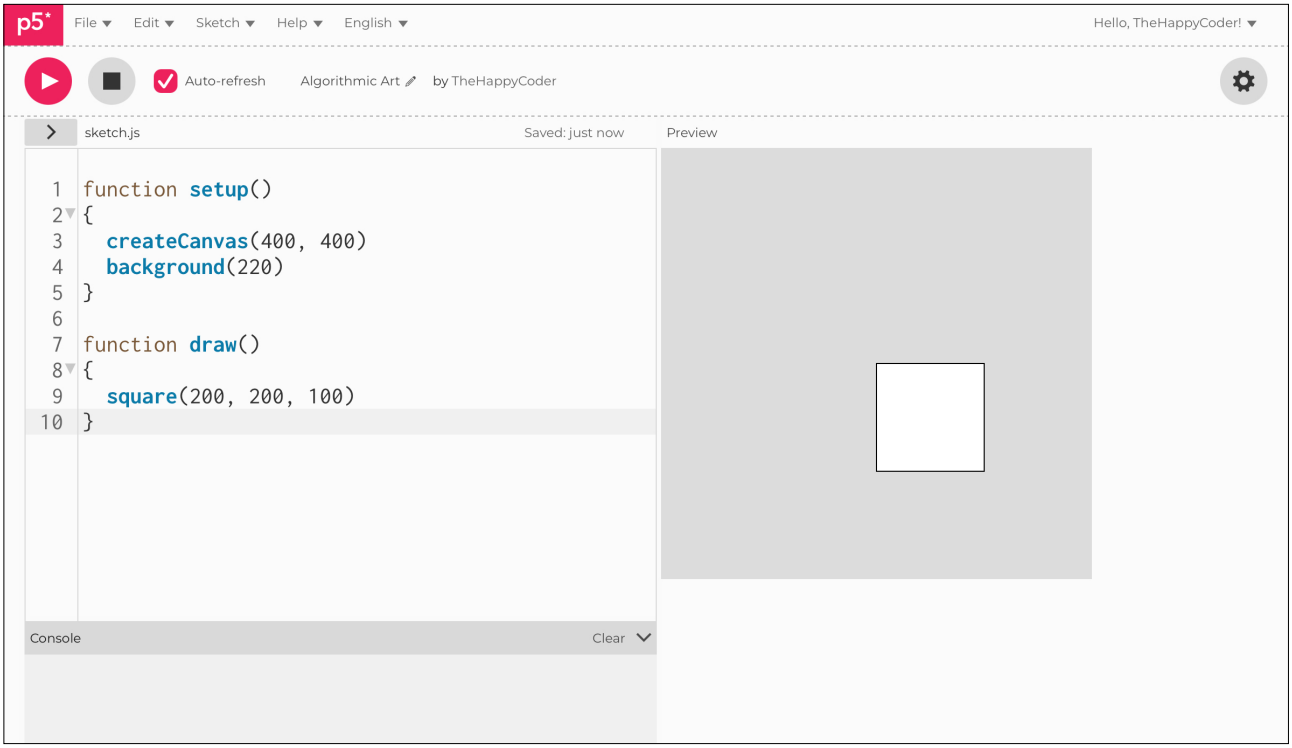
Challenges

1. What would be the co-ordinates to put the square in the centre of the canvas?
2. Draw more squares.

Code Explanation

`square(200, 200, 100)`	Draws a square at (200, 200) with a side length of 100 pixels.

Figure A5.1


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 background(220)
5 }
6
7 function draw()
8 {
9 square(200, 200, 100)
10 }
Console
Clear



Sketch A5.2 from the centre

Using a function called `rectMode()`, we can move the co-ordinates to the centre.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
  rectMode(CENTER)
}

function draw()
{
  square(200, 200, 100)
}
```

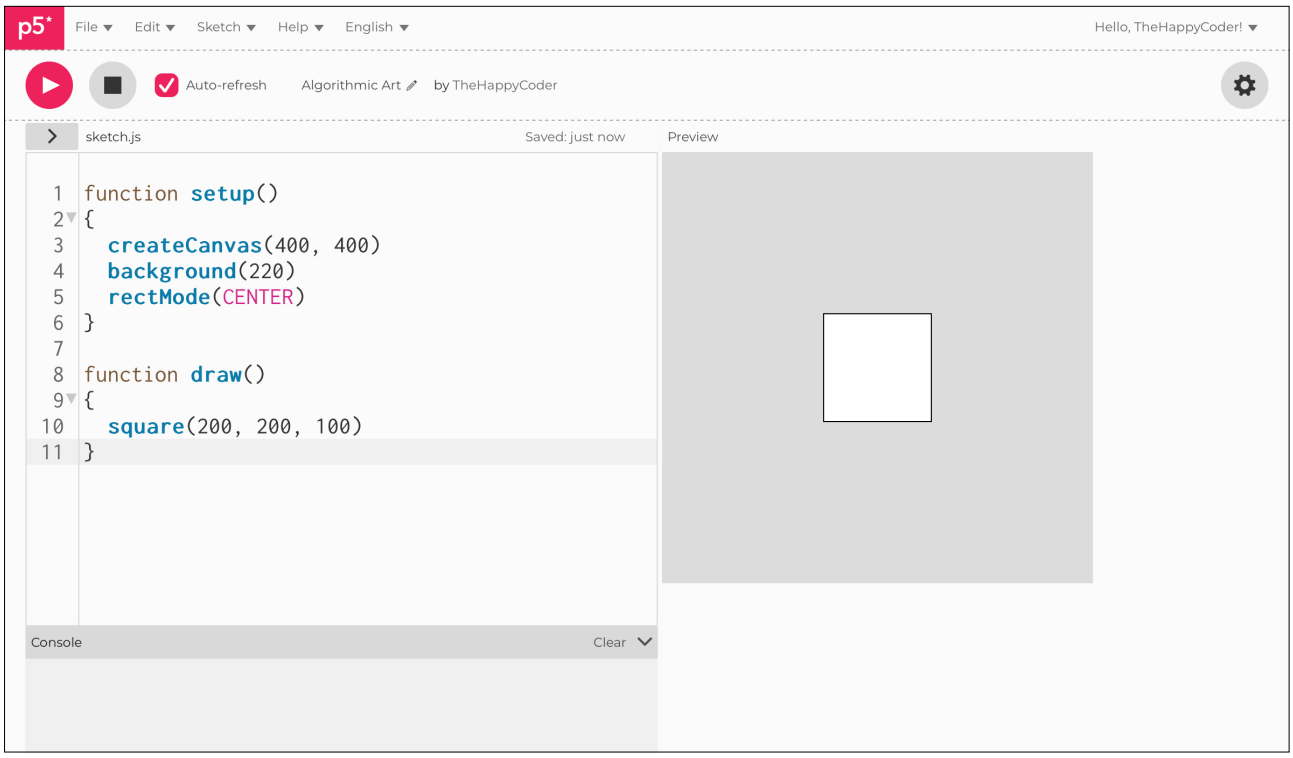
Notes

Now it is nicely in the centre!

Code Explanation

`rectMode(CENTER)`	This function needs the information in the brackets (which needs to be uppercase and US spelling).

Figure A5.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 background(220)
5 rectMode(CENTER)
6 }
7
8 function draw()
9 {
10 square(200, 200, 100)
11 }
Console
Clear



Sketch A5.3 rotate

We can rotate the square now because we have the centre in the centre, if you get my meaning; otherwise, it would rotate around the corner. By default, the angle of rotation is measured in radians; however, we can change that to degrees with angleMode().

```
function setup()
{
  createCanvas(400, 400)
  background(220)
  rectMode(CENTER)
  angleMode(DEGREES)
}

function draw()
{
  rotate(45)
  square(200, 200, 100)
}
```

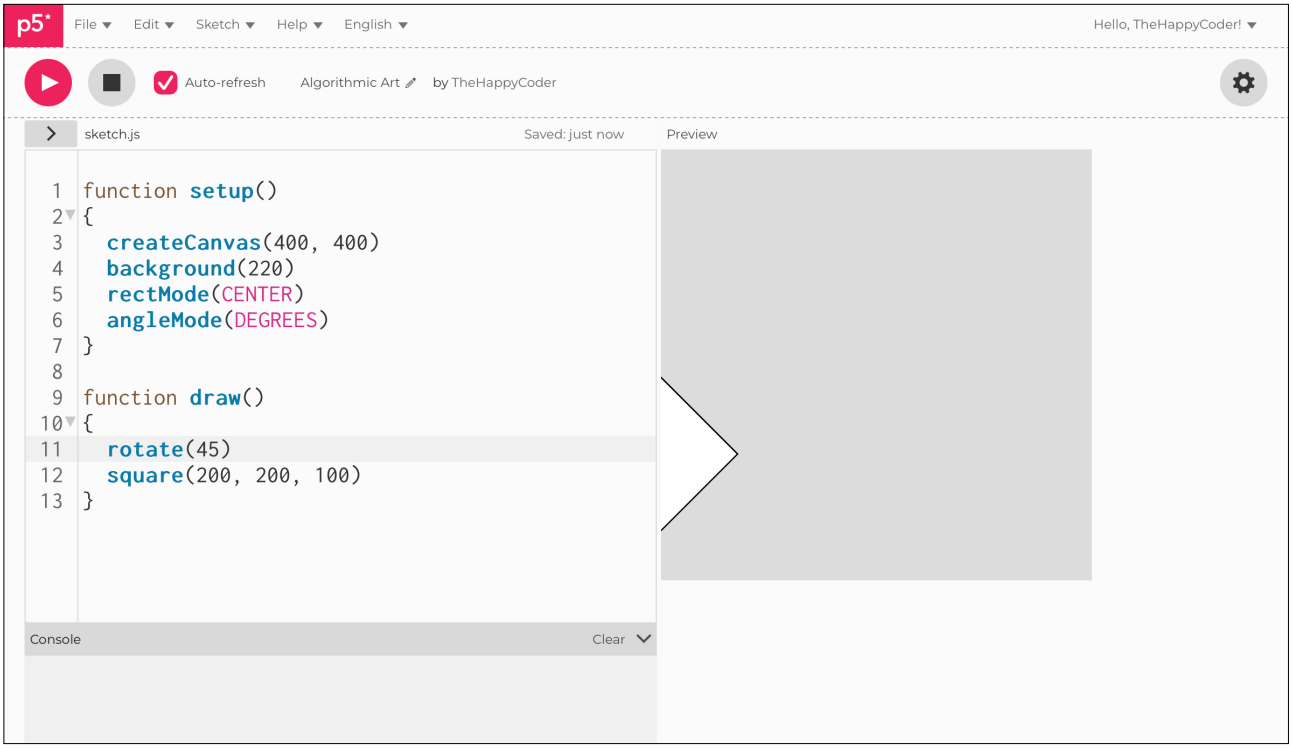
Notes

As you will see, there is a problem. It has rotated about the corner of the canvas, not the centre of the square. The good news is we have a solution!

Code Explanation

angleMode(DEGREES)	Converts from radians to degrees.
rotate(45)	Rotate by 45°.

Figure A5.3


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 background(220)
5 rectMode(CENTER)
6 angleMode(DEGREES)
7 }
8
9 function draw()
10 {
11 rotate(45)
12 square(200, 200, 100)
13 }
Console
Clear



Sketch A5.4 translate

In order to solve this problem, we have to move the origin of the canvas from the top-left-hand corner to the centre of the canvas. To do this, we use the `translate()` function to move it. We then have to change the co-ordinates of the square to `(0, 0)`.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
  rectMode(CENTER)
  angleMode(DEGREES)
}

function draw()
{
  translate(200, 200)
  rotate(45)
  square(0, 0, 100)
}
```

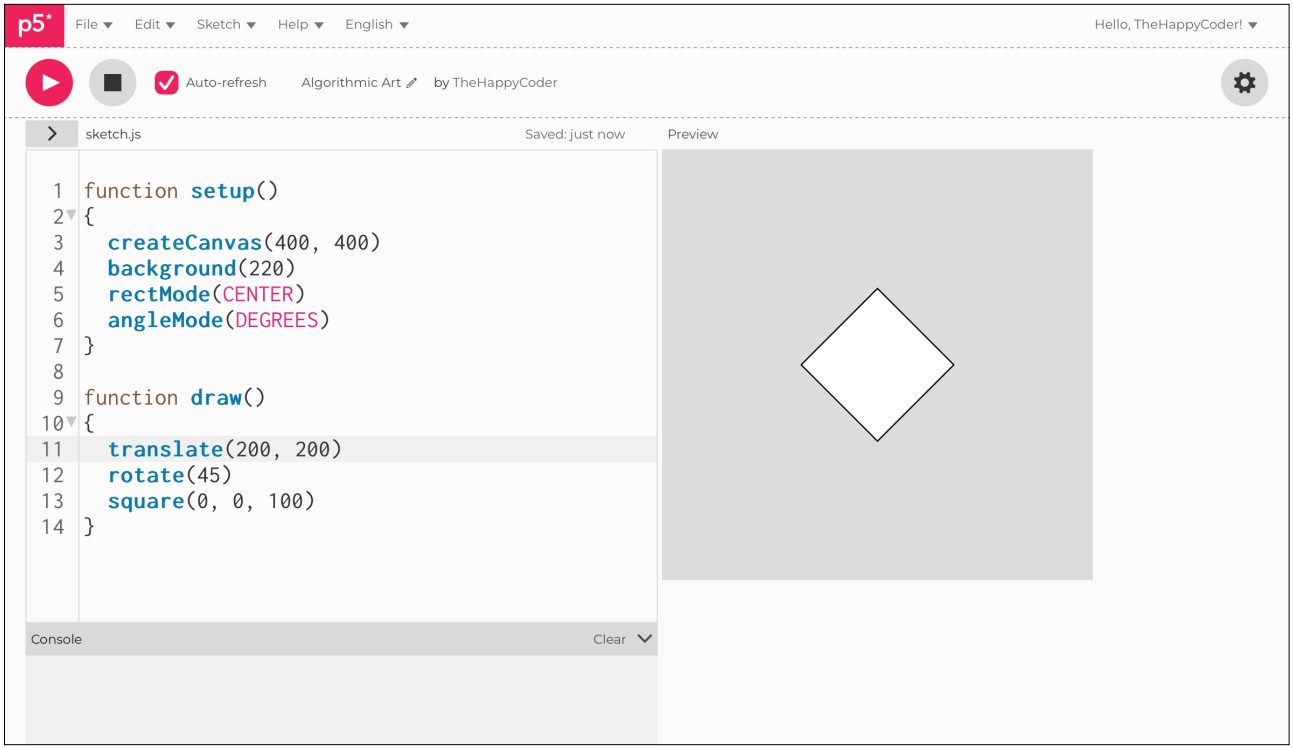
Notes

It is now rotating around the centre of the square.

Code Explanation

`translate(200, 200)`	Moves the origin `(0, 0)` from the top left corner to the centre of the canvas.
`square(0, 0, 100)`	The coordinates now reflect the changes we have made through translating the origin.

Figure A5.4


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 background(220)
5 rectMode(CENTER)
6 angleMode(DEGREES)
7 }
8
9 function draw()
10 {
11 translate(200, 200)
12 rotate(45)
13 square(0, 0, 100)
14 }
Console
Clear



Sketch A5.5 no fill and an angle

We will use a variable for the angle called, strangely enough, `angle`. We can add another function to clear any fill colour (default is white) from the square called `noFill()`.

```
let angle = 45

function setup()
{
  createCanvas(400, 400)
  background(220)
  rectMode(CENTER)
  angleMode(DEGREES)
  noFill()
}

function draw()
{
  translate(200, 200)
  rotate(angle)
  square(0, 0, 100)
}
```

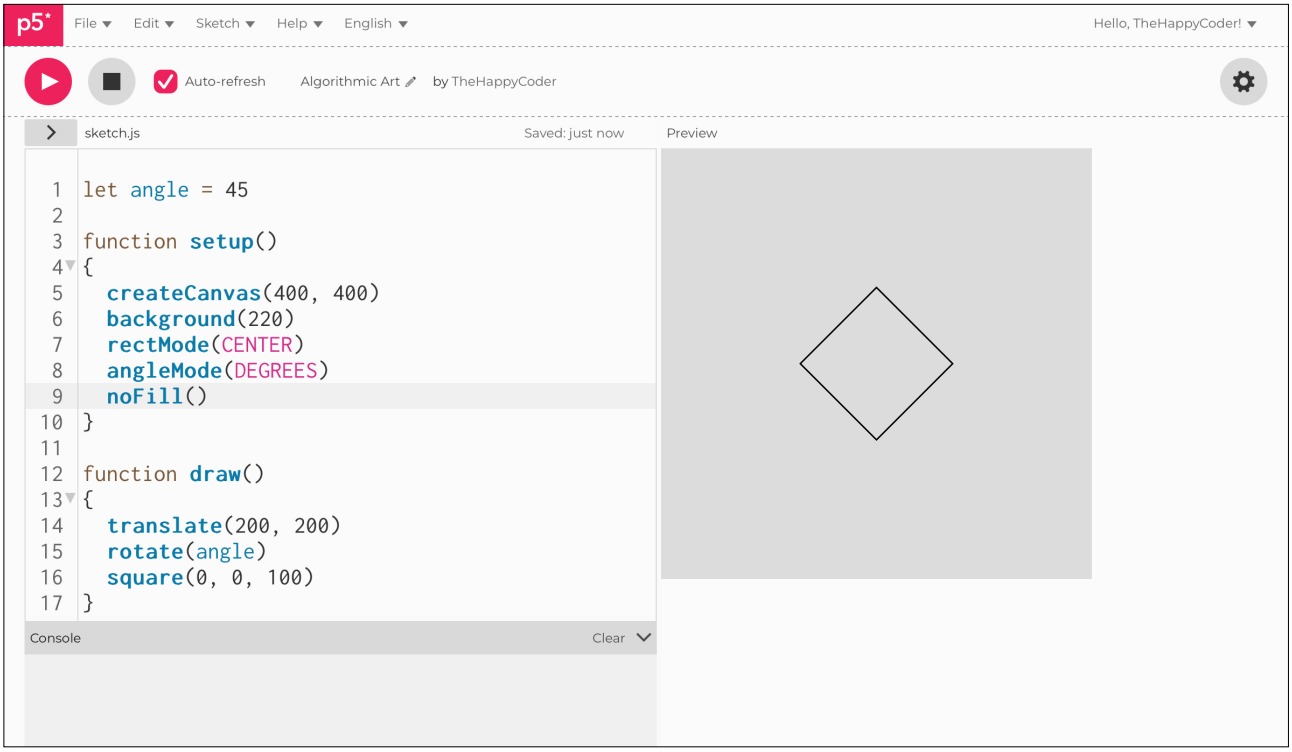
Notes

We get the same effect of a square at 45° but this time without the default fill colour.

Code Explanation

`noFill()`	Simply means there is no colour at all, making it completely transparent.

Figure A5.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
let angle = 45
function setup()
{
createCanvas(400, 400)
background(220)
rectMode(CENTER)
angleMode(DEGREES)
noFill()
}
function draw()
{
translate(200, 200)
rotate(angle)
square(0, 0, 100)
}
Console
Clear

Sketch A5.6 rotating it slowly

We move the `background()` into the `draw()` function. We can use the `//` symbol to effectively remove the line of code. We add `1` to the angle on each iteration of the `draw()` loop. The effect is that the square slowly rotates.

! comment out the `background()` in the `setup()` function

```
let angle = 45

function setup()
{
  createCanvas(400, 400)
  // background(220)
  rectMode(CENTER)
  angleMode(DEGREES)
  noFill()
}

function draw()
{
  background(220)
  translate(200, 200)
  rotate(angle)
  square(0, 0, 100)
  angle++
}
```

Notes

Commenting out (`//`) is a very useful way of leaving code in but where you don't want to use it, or use it at a later date. The computer will ignore any lines that start with `//`. You can leave notes for yourself or someone else to help explain what your code is doing.

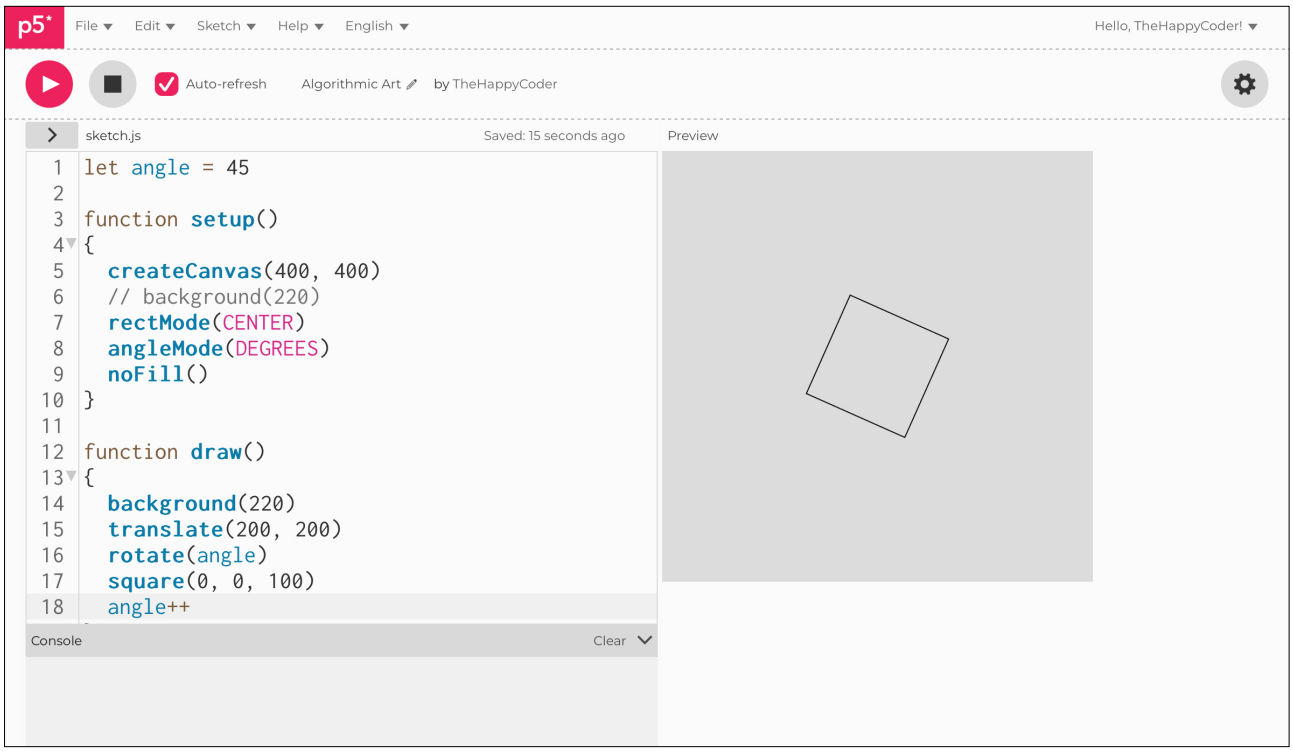
Challenge

Try commenting out other lines of code.

Code Explanation

`//`	The // symbols mean that the computer does not see anything on that line as code (as if invisible).

Figure A5.6 slowly rotates





Sketch A5.7 adding a second square

! Remove the background() in the setup() function.

If we add another square, the rotation is applied to it as well. Here, there are two squares, but they are both rotating together; hence, it looks as if there is just one square.

```
let angle = 45

function setup()
{
  createCanvas(400, 400)
  rectMode(CENTER)
  angleMode(DEGREES)
  noFill()
}

function draw()
{
  background(220)
  translate(200, 200)
  rotate(angle)
  square(0, 0, 100)
  square(0, 0, 100)
  angle++
}
```

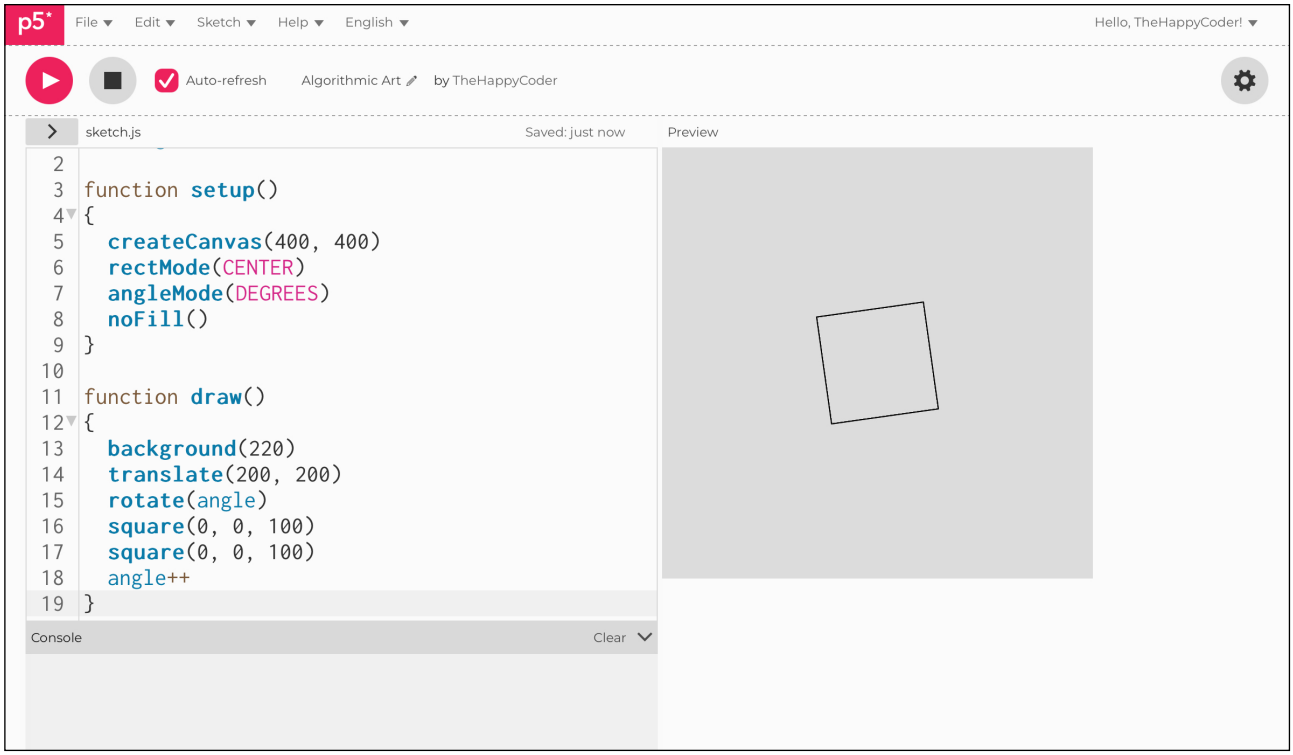
Notes

All you get is what looks like a square slowly rotating as before. What you have is both rotating at the same time; there are two squares. We want to stop one and let the other rotate.

Challenges

1. Change the speed of the rotation (`angle += 2`)
2. Change the direction

Figure A5.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
2
3 function setup()
4 {
5 createCanvas(400, 400)
6 rectMode(CENTER)
7 angleMode(DEGREES)
8 noFill()
9 }
10
11 function draw()
12 {
13 background(220)
14 translate(200, 200)
15 rotate(angle)
16 square(0, 0, 100)
17 square(0, 0, 100)
18 angle++
19 }
Console
Clear



Sketch A5.8 push and pop

If we want to separate the two squares and have one rotate and the other not, we use two functions (as a pair) called `push()` and `pop()`. They are like bookends where you take the current state, change it, (rotate, for instance), and then return the code back to its previous state. What happens between `push()` and `pop()` stays between `push()` and `pop()`.

```
let angle = 45

function setup()
{
  createCanvas(400, 400)
  rectMode(CENTER)
  angleMode(DEGREES)
  noFill()
}

function draw()
{
  background(220)
  translate(200, 200)
  push()
  rotate(angle)
  square(0, 0, 100)
  pop()
  square(0, 0, 100)
  angle++
}
```

Notes

We now have one static square and a rotating square.

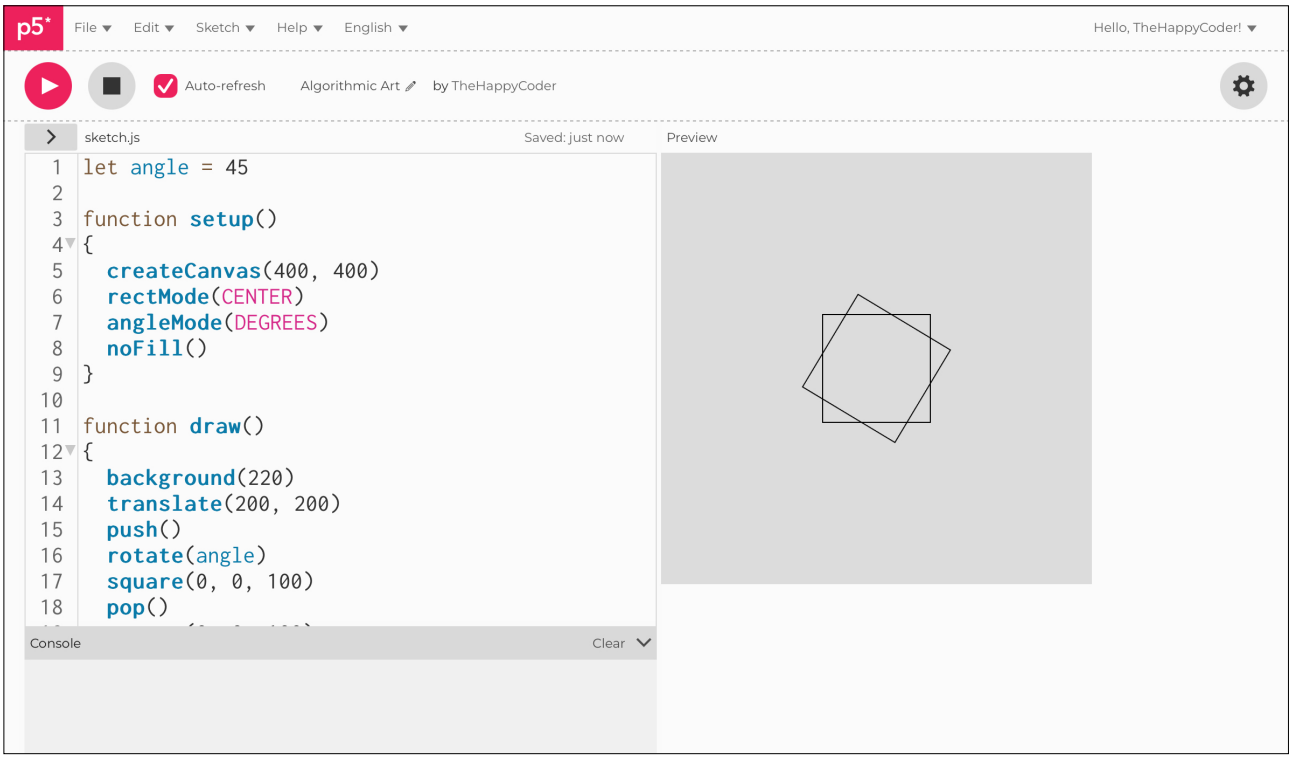
Challenge

Add a third square and have it rotate backwards (`-angle`), don't forget `push()` and `pop()`.

Code Explanation

push()	The push() function collects the state of everything at that moment in the code.
pop()	After you have added other functionality, the pop() function returns the original state when you introduced the push() function.

Figure A5.8



Sketch A5.9 a rectangle

! Start a new sketch

Instead of a square, we have a rectangle. To achieve this, we add another argument and call the `rect()` function. The first two arguments are the coordinates, the third is the width, and the fourth is the height of the rectangle.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
  rectMode(CENTER)
}

function draw()
{
  rect(200, 200, 300, 50)
}
```

Notes

If you only give `rect()` three arguments, it will draw a square.

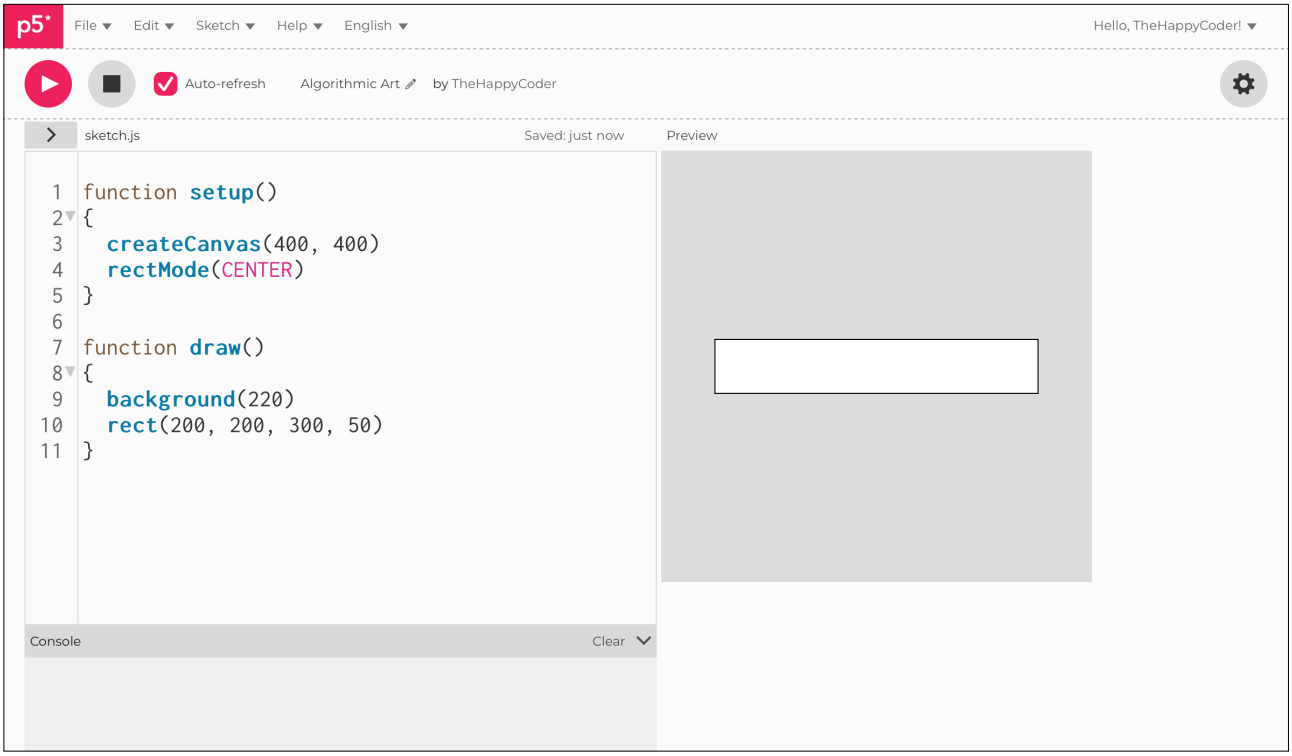
Challenge

Try other dimensions for sizes.

Code Explanation

`rect(200, 200, 300, 50)`	The rectangle has four arguments: the x-coordinate, the y-coordinate, the width, and the height of the rectangle.

Figure A5.9


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 rectMode(CENTER)
5 }
6
7 function draw()
8 {
9 background(220)
10 rect(200, 200, 300, 50)
11 }
Console
Clear



Sketch A5.10 random rectangles

Let's go a little mad with random and create a nice pattern with the rectangles. We randomise the coordinates between 100 and 300, and give the width and height a random dimension of 50. We make the background white.

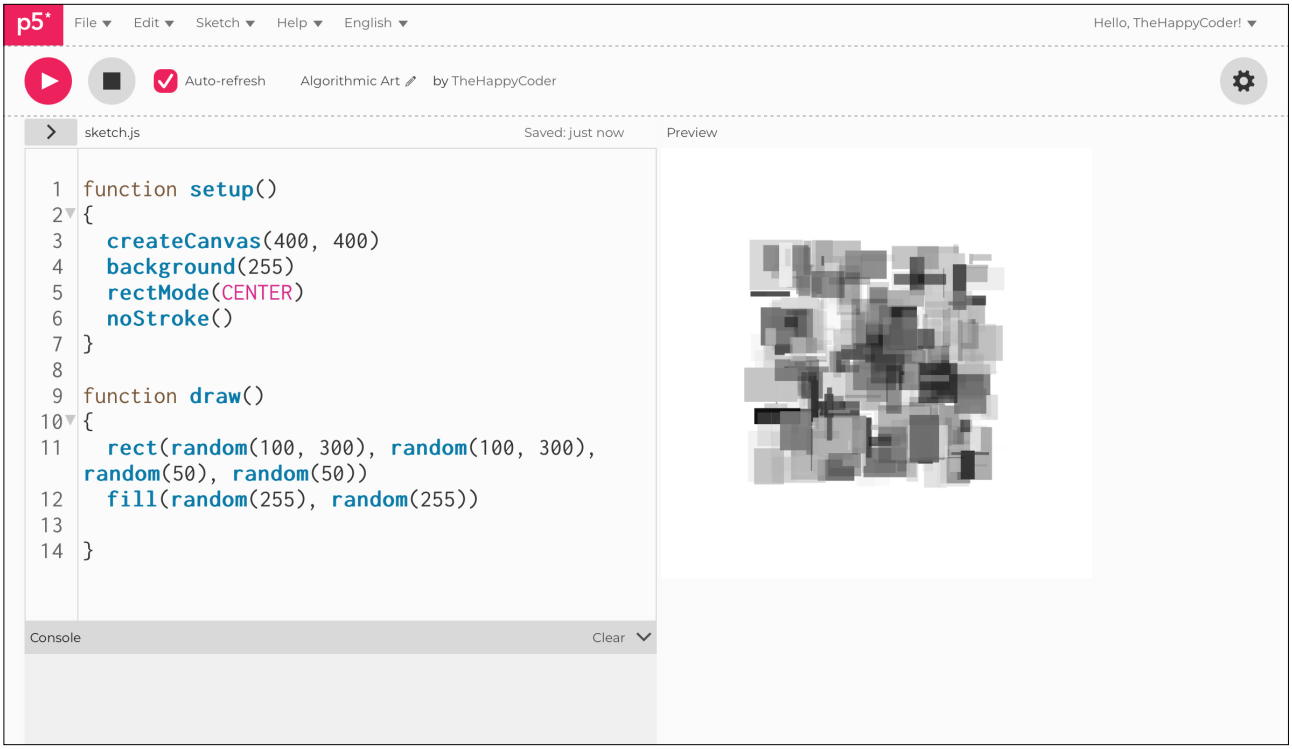
```
function setup()
{
  createCanvas(400, 400)
  background(255)
  rectMode(CENTER)
  noStroke()
}

function draw()
{
  rect(random(100, 300), random(100, 300), random(50), random(50))
  fill(random(255), random(255))
}
```

Challenges

1. Change the random dimensions.
2. Have it draw just 100 rectangles.
3. Add colour.

Figure A5.10


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 background(255)
5 rectMode(CENTER)
6 noStroke()
7 }
8
9 function draw()
10 {
11 rect(random(100, 300), random(100, 300), random(50), random(50))
12 fill(random(255), random(255))
13
14 }
Console
Clear



Algorithmic Art
Module A
Unit #6
Ellipses and Triangles



Module A Unit #6: ellipses and triangles

As well as introducing two new shapes, we will also explore how to interact with shapes with the mouse. We will be able to move and stretch these shapes, offering more opportunities to create interesting designs that are more than static images.

Key concepts:

ellipse()
triangle()
mouseX
mouseY
move
stretch

Sketch A6.1 drawing an ellipse

! We start with a new sketch with an ellipse.

The `ellipse()` function has four arguments: the x and y coordinates, and the width and height of the ellipse.

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

function draw()
{
  background(220)
  ellipse(200, 200, 300, 100)
}
```

Notes

If you use only three dimensions, you will draw a circle.

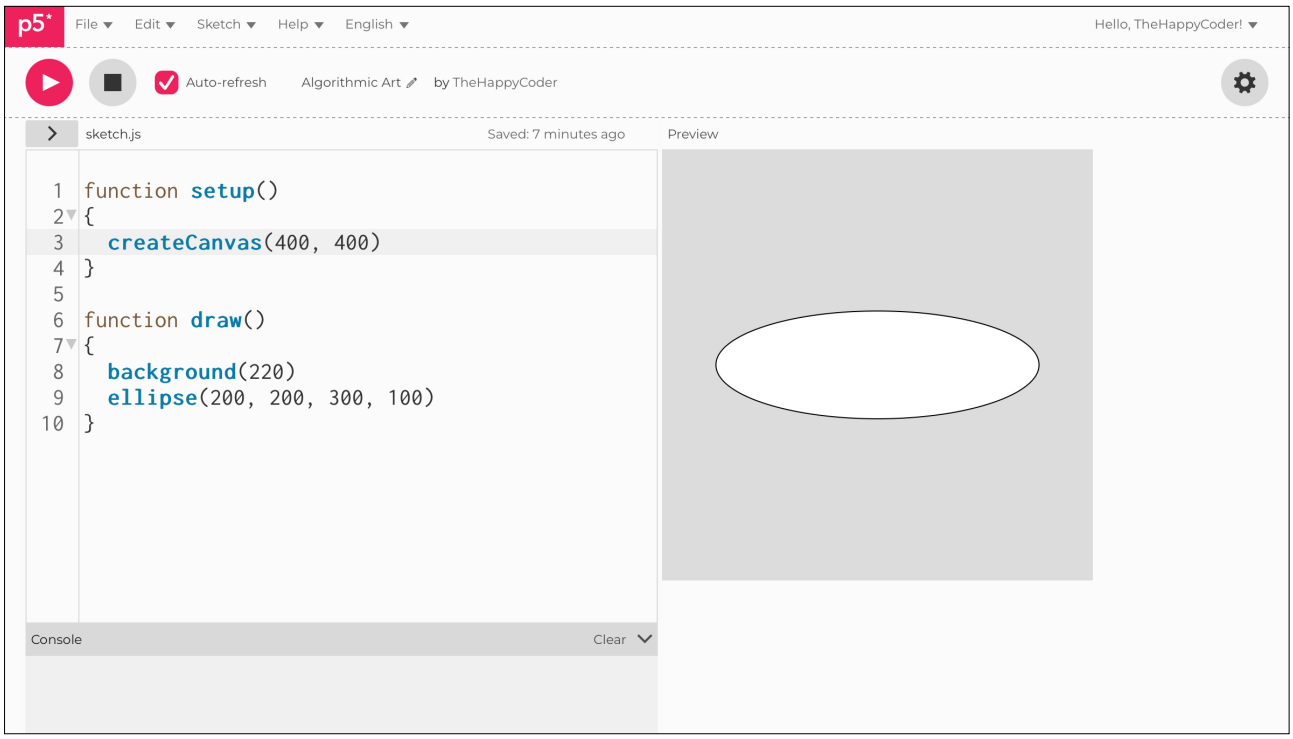
Challenge

Try other dimensions for the height and width.

Code Explanation

`ellipse(200, 200, 300, 100)`	We have an ellipse at the centre of the canvas (200, 200) and a width of 300 and height of 100.

Figure A6.1


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: 7 minutes ago
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 ellipse(200, 200, 300, 100)
10 }
Console
Clear



Sketch A6.2 mouseX and mouseY

We can use some built-in variables; one set of variables is called mouseX and mouseY. They return the coordinates of the mouse cursor position on the canvas. We can illustrate it with our ellipse.

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

function draw()
{
  background(220)
  ellipse(mouseX, mouseY, 300, 100)
}
```

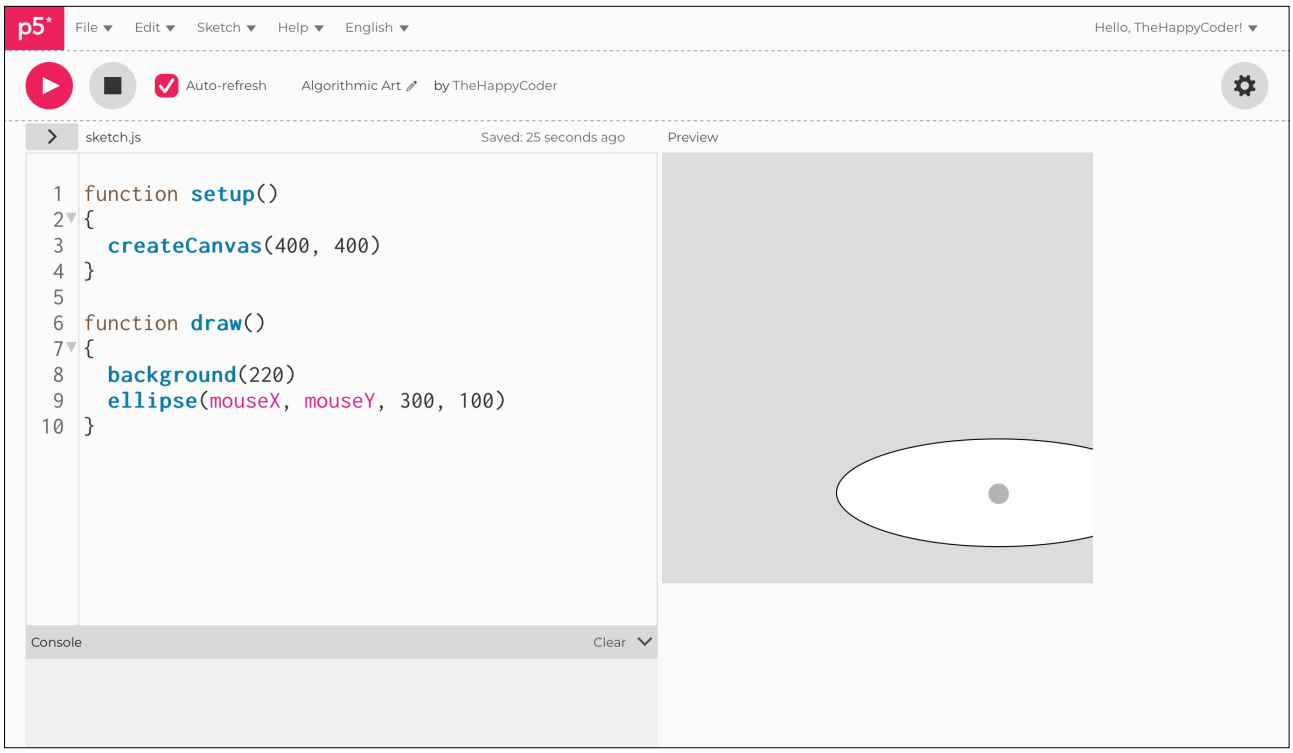
Notes

As you move your mouse cursor around the canvas, the ellipse follows it.

Code Explanation

`ellipse(mouseX, mouseY, 300, 100)`	mouseX returns the x coordinate, and mouseY returns the y coordinate of the cursor.

Figure A6.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: 25 seconds ago
Preview
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
ellipse(mouseX, mouseY, 300, 100)
}
Console
Clear



Sketch A6.3 stretchy time

If we move mouseX and mouseY to the dimensions of the ellipse, we can see how we can change the shape of the ellipse through moving our cursor.

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

function draw()
{
  background(220)
  ellipse(200, 200, mouseX, mouseY)
}
```

Notes

You can now stretch the ellipse by moving your mouse.

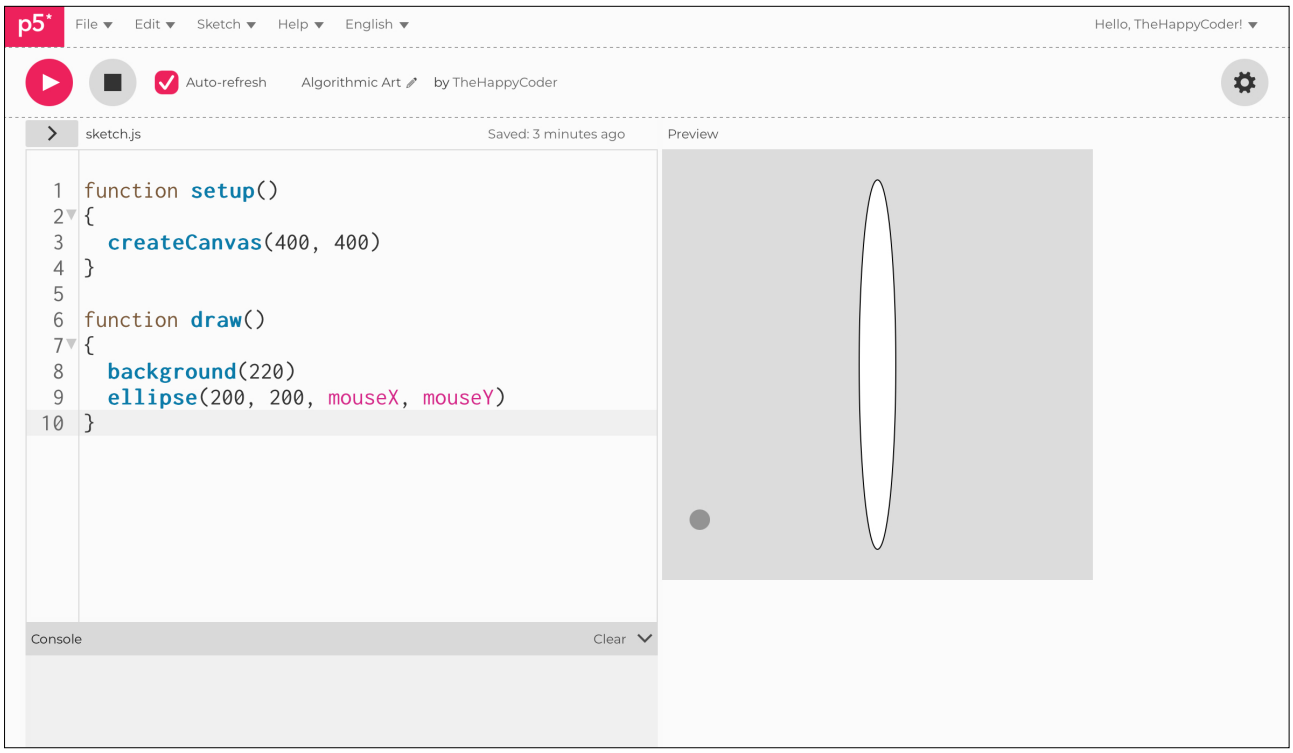
Challenge

Try it with the rectangle.

Code Explanation

`ellipse(200, 200, mouseX, mouseY)`	The mouseX and mouseY are the width and height, respectively.

Figure A6.3


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: 3 minutes ago
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 ellipse(200, 200, mouseX, mouseY)
10 }
Console
Clear

Sketch A6.4 triangle

Instead of an ellipse, let's make a triangle. A triangle has coordinates for all three corners, so the function `triangle()` has six arguments. There are a lot of numbers to get your head around; often, I sketch it out on a piece of paper first so I get the right coordinates for the right corner.

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

function draw()
{
  background(220)
  triangle(100, 100, 100, 300, 300, 300)
}
```

Notes

We have three sets of coordinates: (`100, 100`), (`100, 300`) and (`300, 300`).

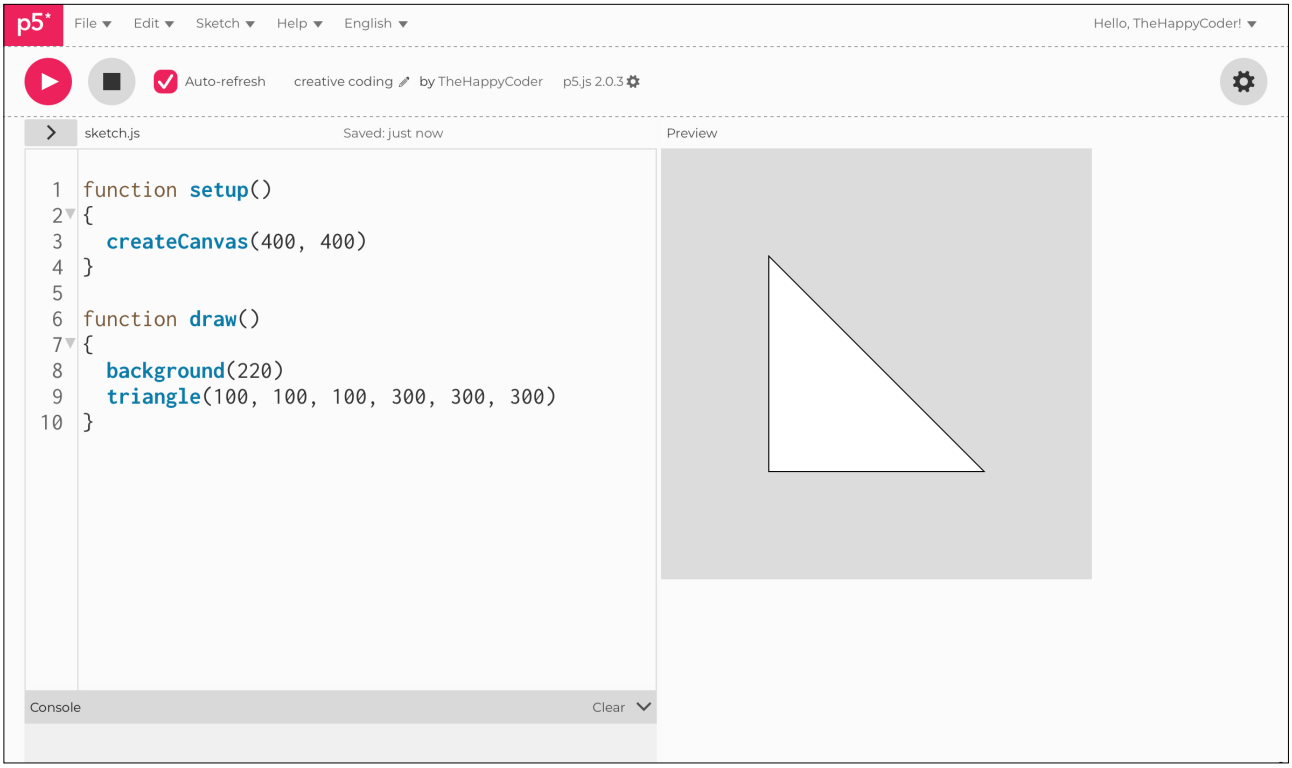
Challenge

Draw more triangles, make a simple pattern.

Code Explanation

`triangle(100, 100, 100, 300, 300, 300)`	A triangle with three sets of coordinates for each corner (vertex).

Figure A6.4


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding
by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 triangle(100, 100, 100, 300, 300, 300)
10 }
Console
Clear



Sketch A6.5 width and height

We can make use of another built-in variable, `width` and `height`. These variables return the width and height of the canvas. So in our case, the value of `width` is `400` and the value of `height` is also `400`. If we change the dimensions of the canvas, these values also change. If we divide the `width` and `height` by 2, we always get the centre of the canvas, whatever its dimensions.

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

function draw()
{
  background(220)
  translate(width/2, height/2)
  triangle(100, 100, 100, 300, 300, 300)
}
```

Notes

We have translated the origin of the canvas to its centre rather than the top left-hand corner. This has pushed the triangle down and to the left by `200` pixels.

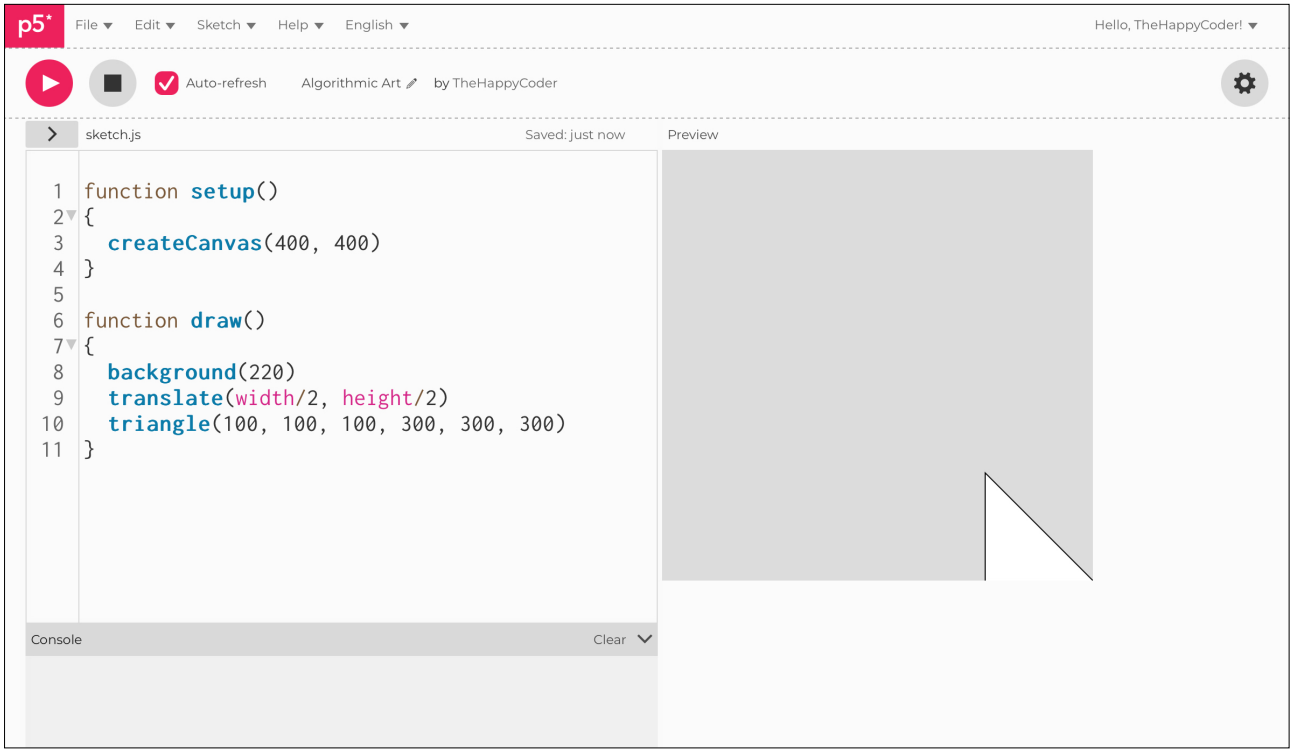
Challenge

Change the dimensions of the canvas.

Code Explanation

`translate(width/2, height/2)`	The width/2 is 200, and the height/2 is also 200.

Figure A6.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 translate(width/2, height/2)
10 triangle(100, 100, 100, 300, 300, 300)
11 }
Console
Clear



Sketch A6.6 rebuilding the triangle

This is where a physical drawing often helps. We can redraw the triangle with new coordinates. Remember that the origin is now in the centre of the canvas, and we measure the x and y coordinates from the centre; hence, −100 is to the left of the centre for x and upwards for the y coordinates, and vice versa for 100.

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

function draw()
{
  background(220)
  translate(width/2, height/2)
  triangle(-100, -100, -100, 100, 100, 100)
}
```

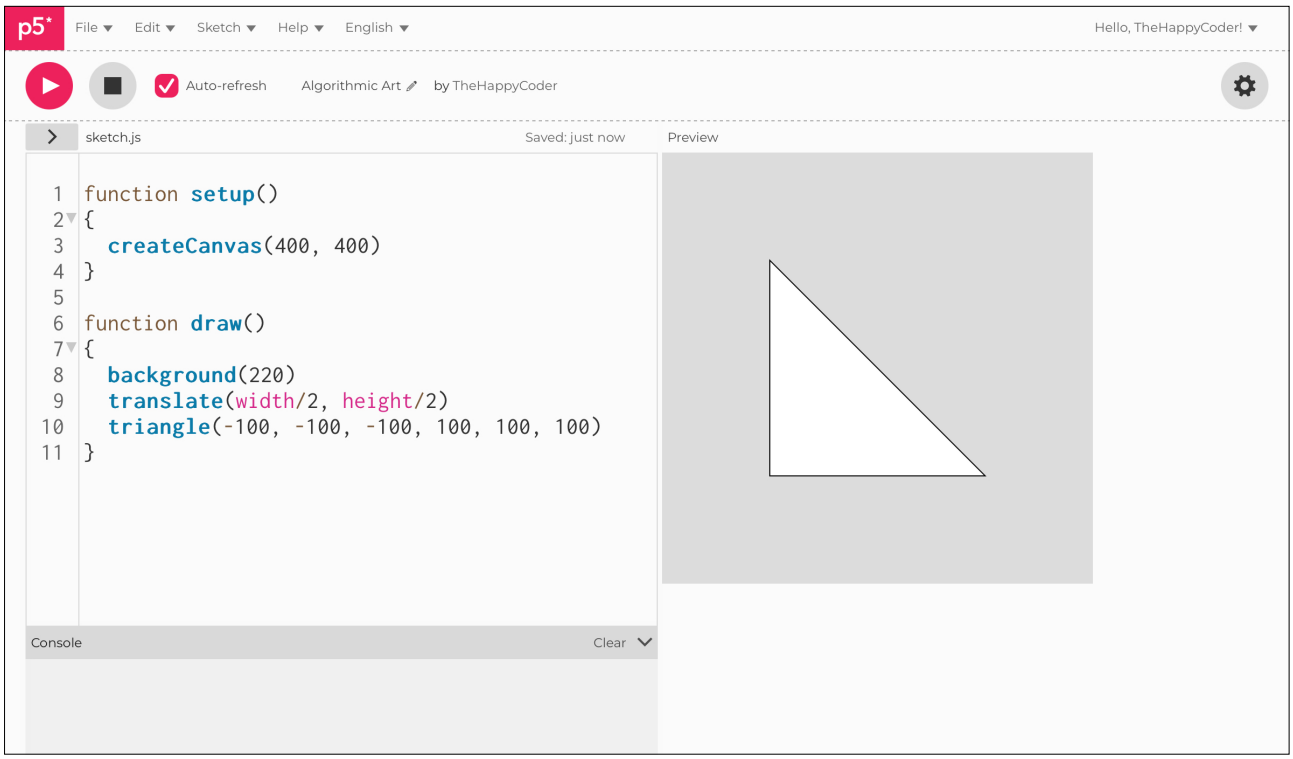
Notes

We have the same triangle as previously.

Challenge

Change the coordinates to draw different types of triangles.

Figure A6.6


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 translate(width/2, height/2)
10 triangle(-100, -100, -100, 100, 100, 100)
11 }
Console
Clear



Sketch A6.7 a length

Let's give the dimension a variable called `len` (for length).

```
let len = 100

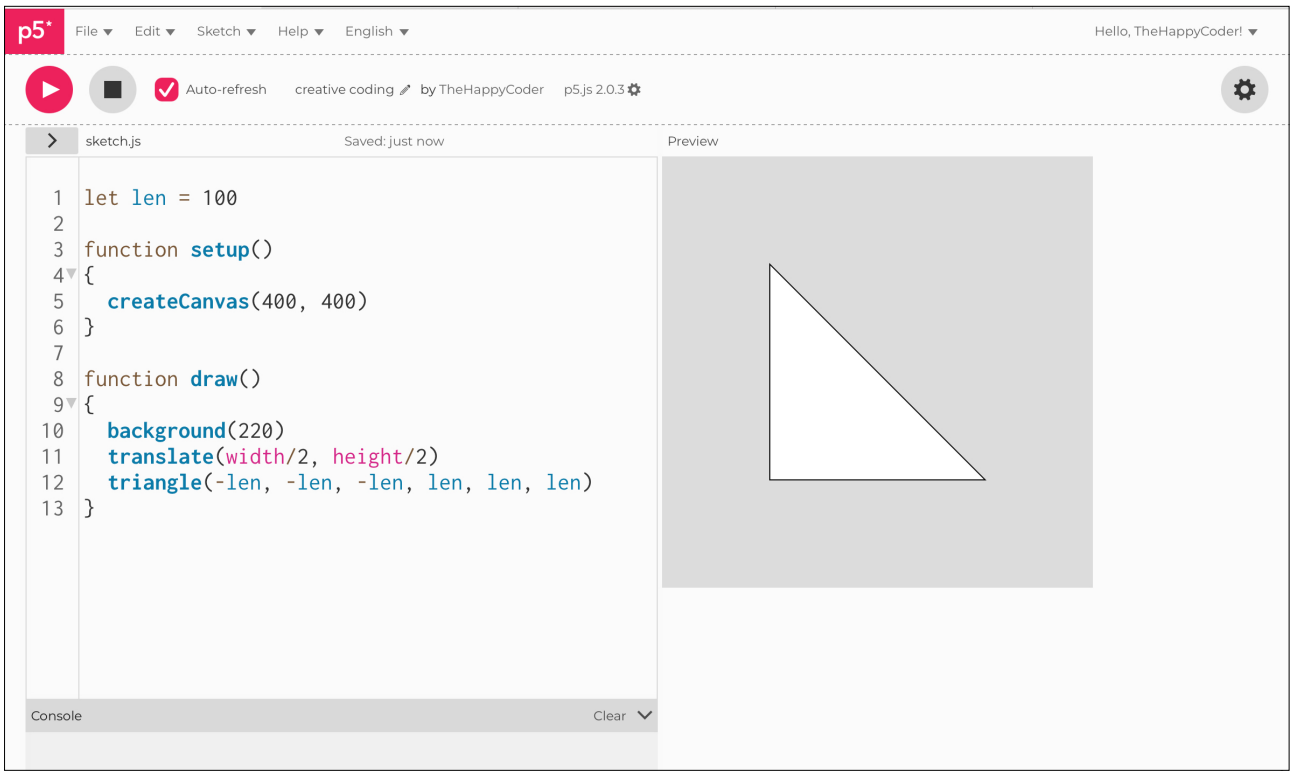
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  triangle(-len, -len, -len, len, len, len)
}
```

Notes

This can be handy if we change the dimension and therefore only need to change it once.

Figure A6.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding
by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: just now
Preview
let len = 100
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
translate(width/2, height/2)
triangle(-len, -len, -len, len, len, len)
}
Console
Clear



Sketch A6.8 a bit more pointy

Make it a bit smaller and more pointy so that it isn't an equilateral triangle.

```
let len = 50

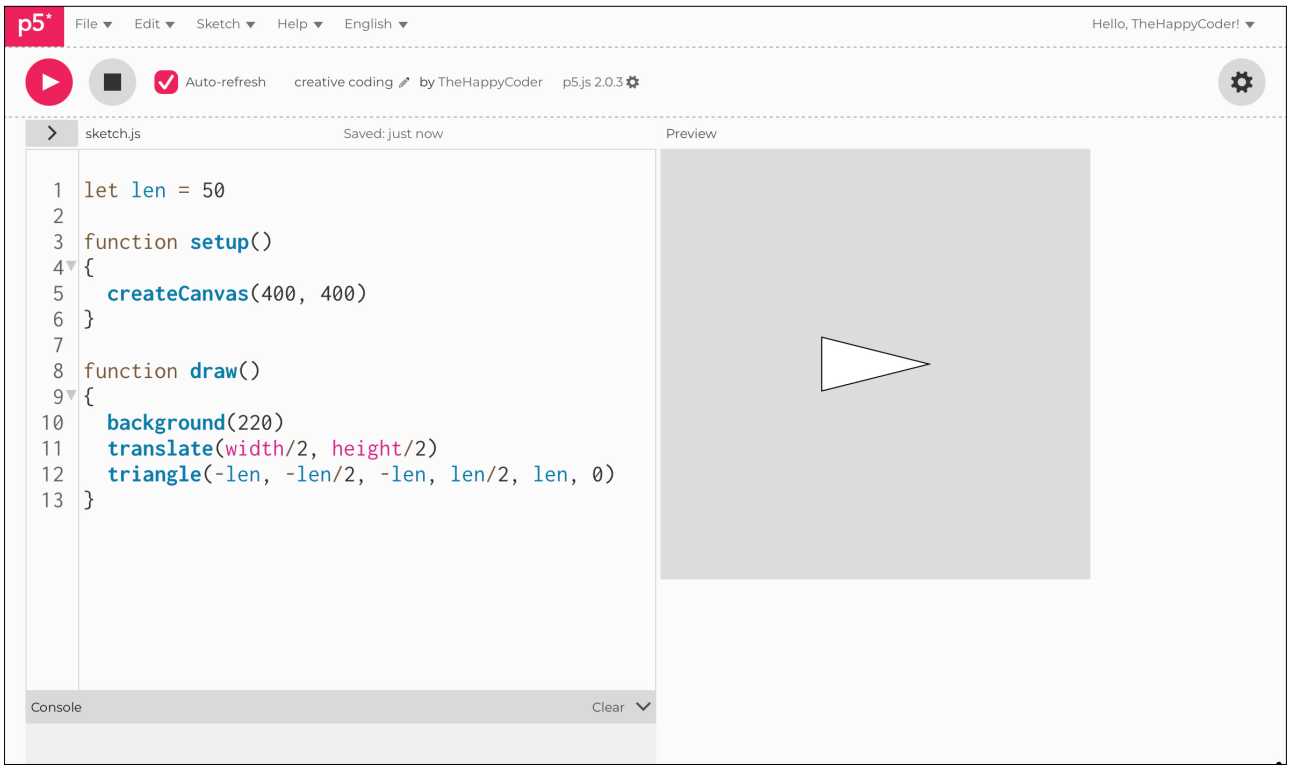
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  triangle(-len, -len/2, -len, len/2, len, 0)
}
```

Notes

We can see what direction it is pointing in.

Figure A6.8


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding
by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: just now
Preview
let len = 50
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
translate(width/2, height/2)
triangle(-len, -len/2, -len, len/2, len, 0)
}
Console
Clear



Sketch A6.9 a hundred of them

Now we create a `for()` loop to draw `100` triangles. We put those two lines of code inside the `for()` loop.

```
let len = 50

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

function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    translate(width/2, height/2)
    triangle(-len, -len/2, -len, len/2, len, 0)
  }
}
```

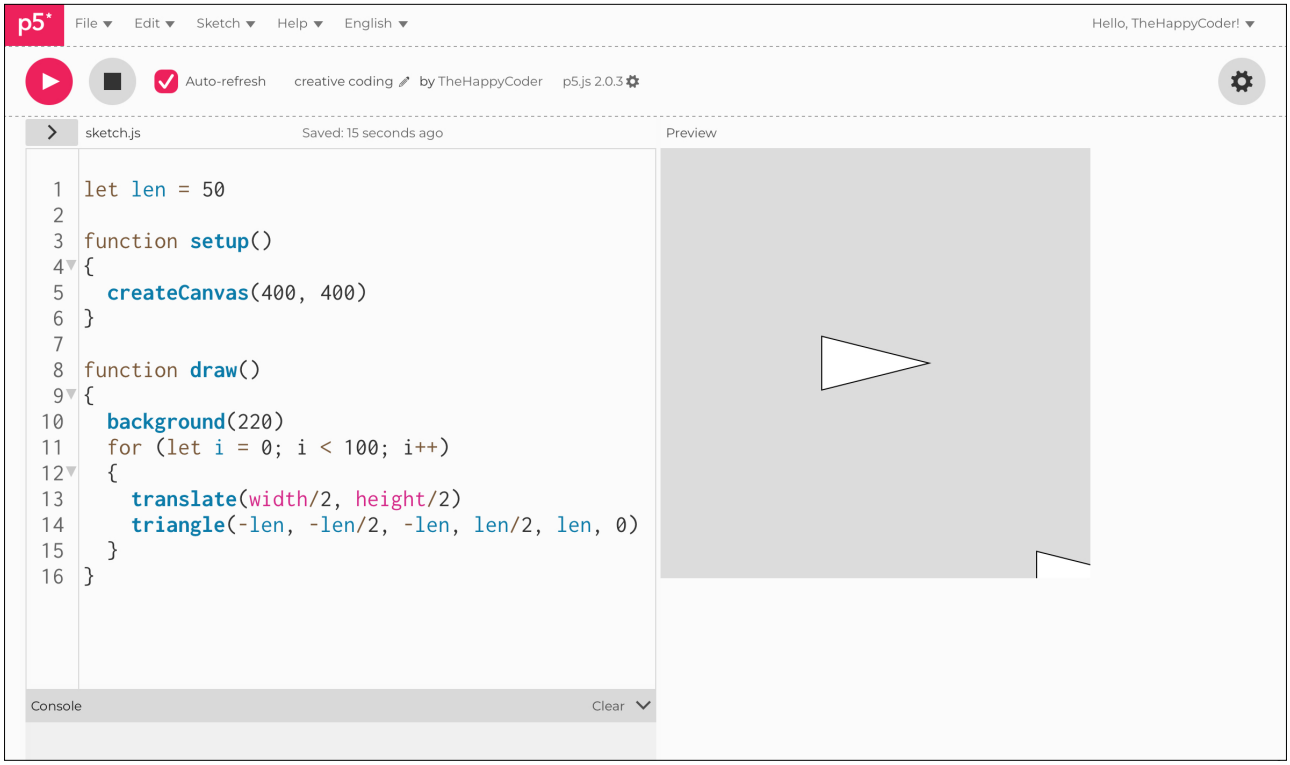
Notes

The problem is that the `translate()` function is accumulative, which means it shifts the origin by that amount each iteration in the `for()` loop. We need to use `push()` and `pop()`, and also we need to stop repeating the `for()` in `draw()` with `noLoop()`.

Code Explanation

`for (let i = 0; i < 100; i++)`	A for() loop where the variable i, which is defined and initialised to 0, is then incremented by 1 until it reaches 100.

Figure A6.9


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding
by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: 15 seconds ago
Preview
let len = 50
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
for (let i = 0; i < 100; i++)
{
translate(width/2, height/2)
triangle(-len, -len/2, -len, len/2, len, 0)
}
}
Console
Clear



Sketch A6.10 push pop stop

We add `push()` and `pop()` to stop the `translate()` from adding itself 100 times, and also the `noLoop()` to stop the `draw()` function once it has drawn the 100 triangles.

```
let len = 50

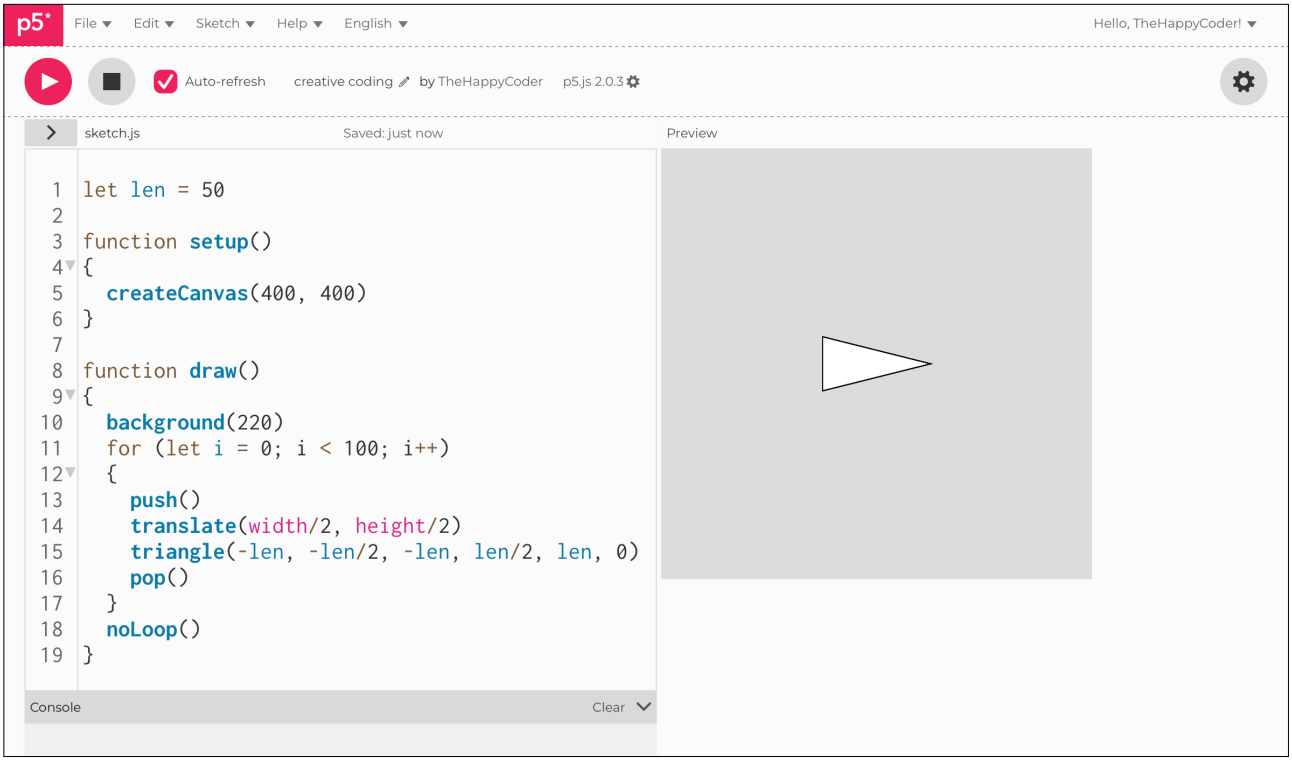
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(width/2, height/2)
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

Notes

We have `100` triangles, but they are all in one place. We could also put all of this code in the `setup()` function and therefore no need to use `noLoop()`.

Figure A6.10


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: just now
Preview
let len = 50
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
for (let i = 0; i < 100; i++)
{
push()
translate(width/2, height/2)
triangle(-len, -len/2, -len, len/2, len, 0)
pop()
}
noLoop()
}
Console
Clear



Sketch A6.11 rotate the triangle

We are rotating all the triangles by 45° degrees.

```
let len = 50
let angle = 45

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

function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(width/2, height/2)
    rotate(angle)
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

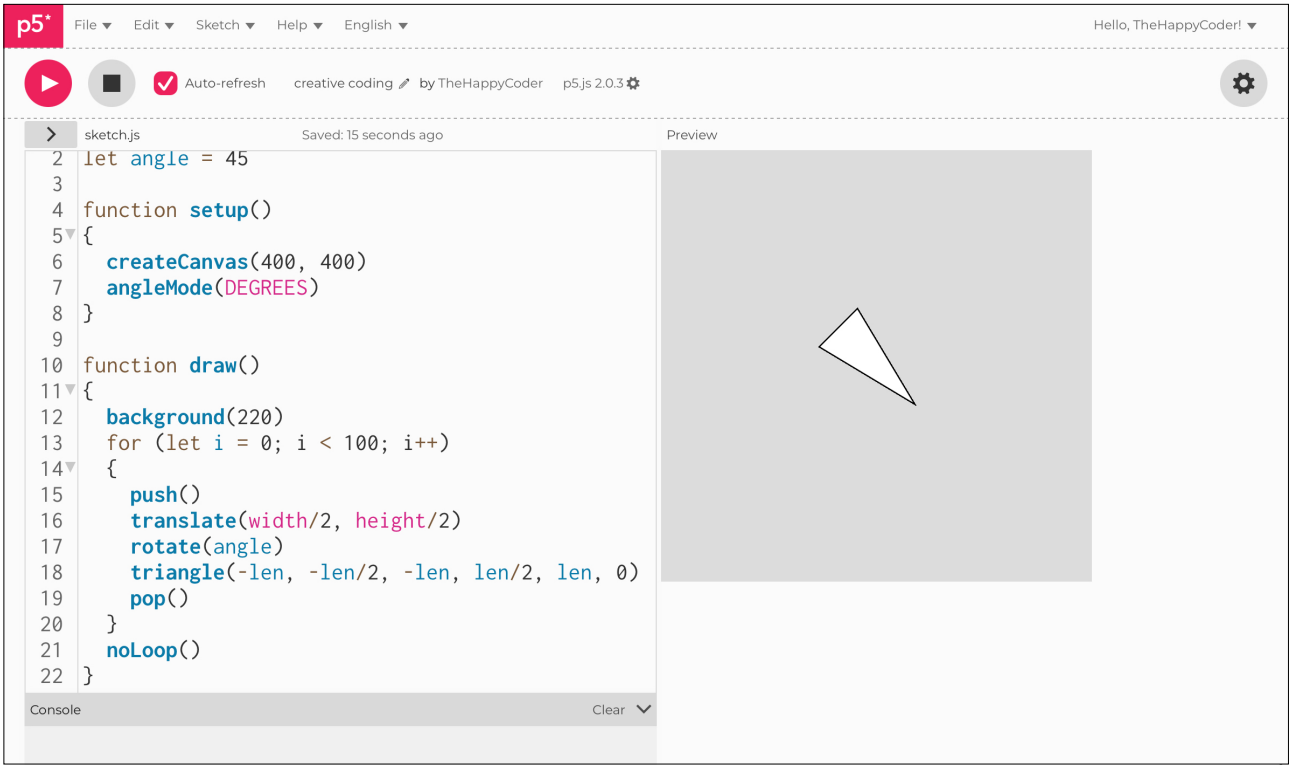
Notes

We have 100 triangles, all rotated by 45°.

Challenge

Try other angles.

Figure A6.11


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding
by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: 15 seconds ago
Preview
let angle = 45
function setup()
{
createCanvas(400, 400)
angleMode(DEGREES)
}
function draw()
{
background(220)
for (let i = 0; i < 100; i++)
{
push()
translate(width/2, height/2)
rotate(angle)
triangle(-len, -len/2, -len, len/2, len, 0)
pop()
}
noLoop()
}
Console
Clear



Sketch A6.12 split the triangles up

But we want to see all the triangles; to do that, we can translate them randomly across the whole canvas (width and height).

```
let len = 50
let angle = 45

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

function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(random(width), random(height))
    rotate(angle)
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

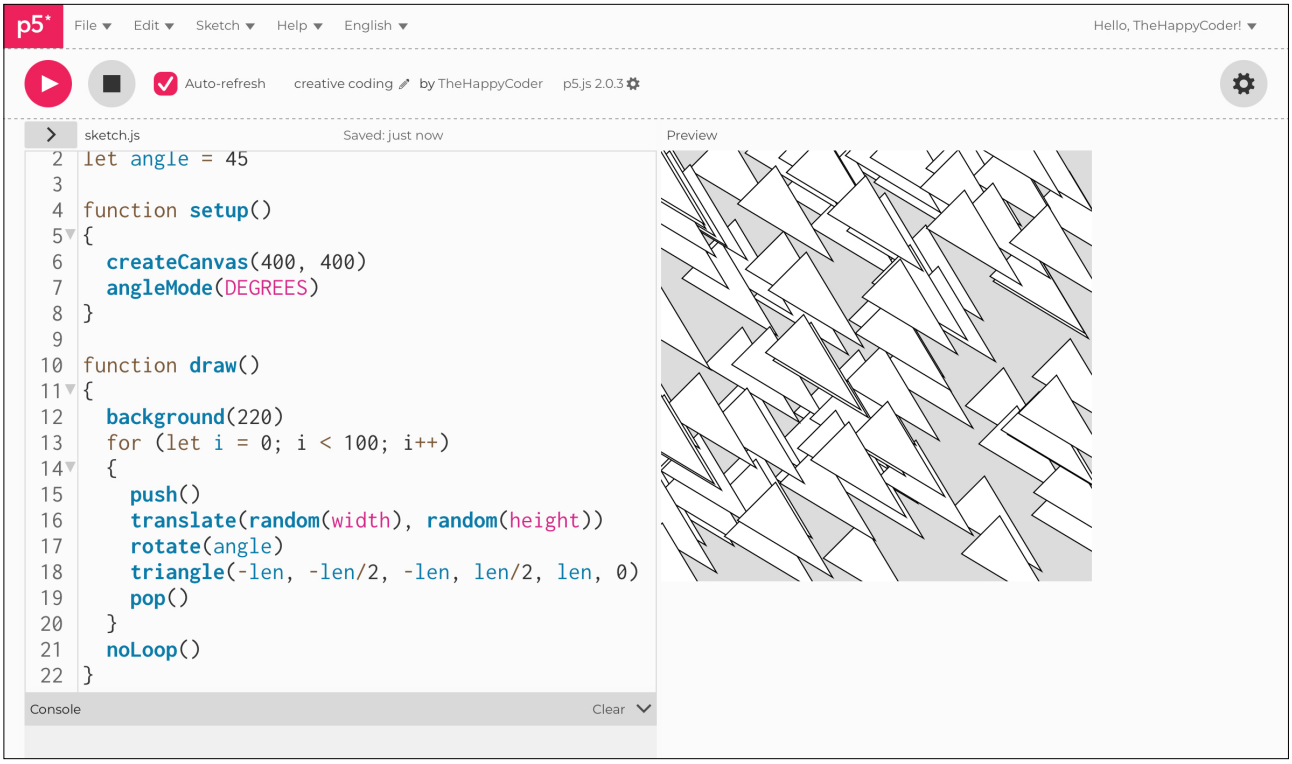
Notes

Now each triangle drawn has its own random position.

Challenges

1. Try pushing them all into a square in the middle.
2. Make the triangles smaller or random in size.
3. Can you work out how to rotate each one separately?

Figure A6.12


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding
by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: just now
Preview
let angle = 45
function setup()
{
createCanvas(400, 400)
angleMode(DEGREES)
}
function draw()
{
background(220)
for (let i = 0; i < 100; i++)
{
push()
translate(random(width), random(height))
rotate(angle)
triangle(-len, -len/2, -len, len/2, len, 0)
pop()
}
noLoop()
}
Console
Clear

Sketch A6.13 random angles

This is how you can randomly set the angles. Change the angle to a full 360° and rotate randomly within that angle.

```
let len = 50
let angle = 360

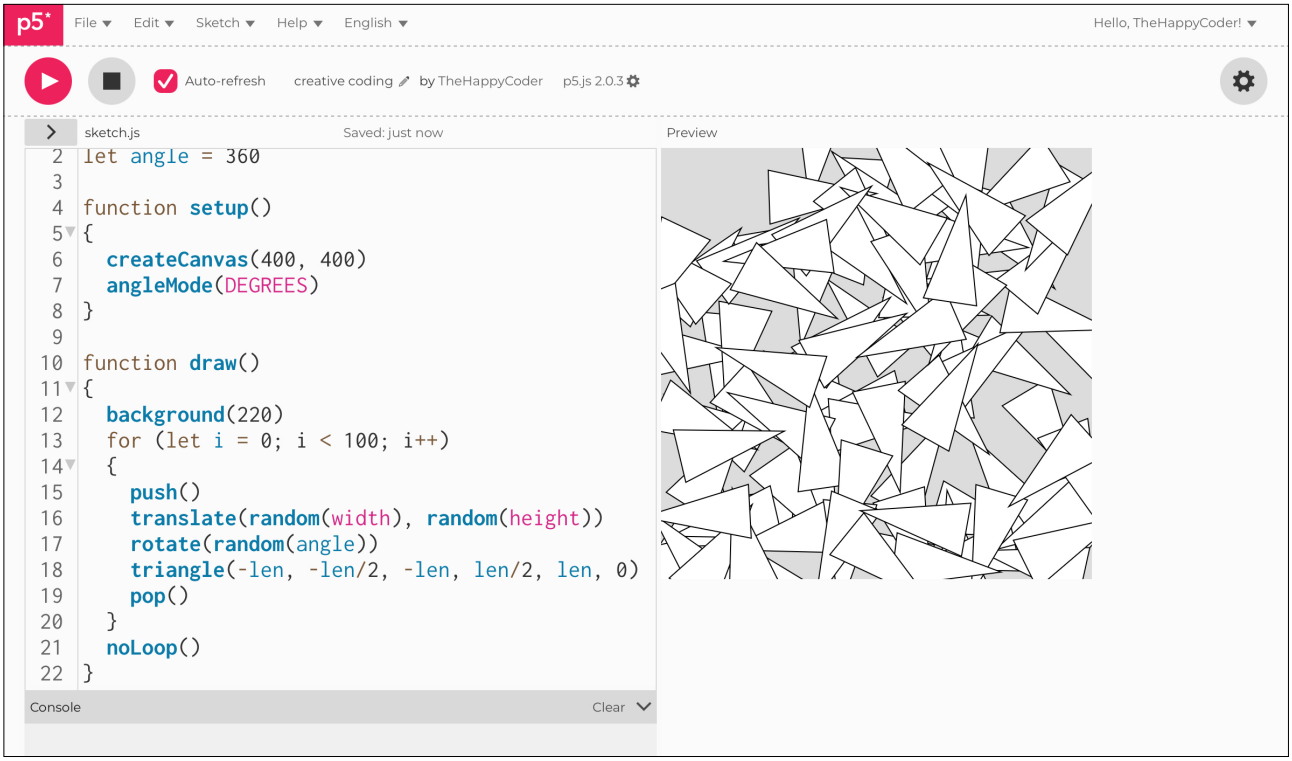
function setup()
{
  createCanvas(400, 400)
  angleMode(DEGREES)
}

function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(random(width), random(height))
    rotate(random(angle))
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

Notes

Rotating randomly from 0° to 360° degrees

Figure A6.13


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding
by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: just now
Preview
let angle = 360
function setup()
{
createCanvas(400, 400)
angleMode(DEGREES)
}
function draw()
{
background(220)
for (let i = 0; i < 100; i++)
{
push()
translate(random(width), random(height))
rotate(random(angle))
triangle(-len, -len/2, -len, len/2, len, 0)
pop()
}
noLoop()
}
Console
Clear



Sketch A6.14 random length

Now for the size of the triangle. This will keep the proportions the same. We will have triangles between 5 and 50 long.

```
let len = 50
let angle = 360

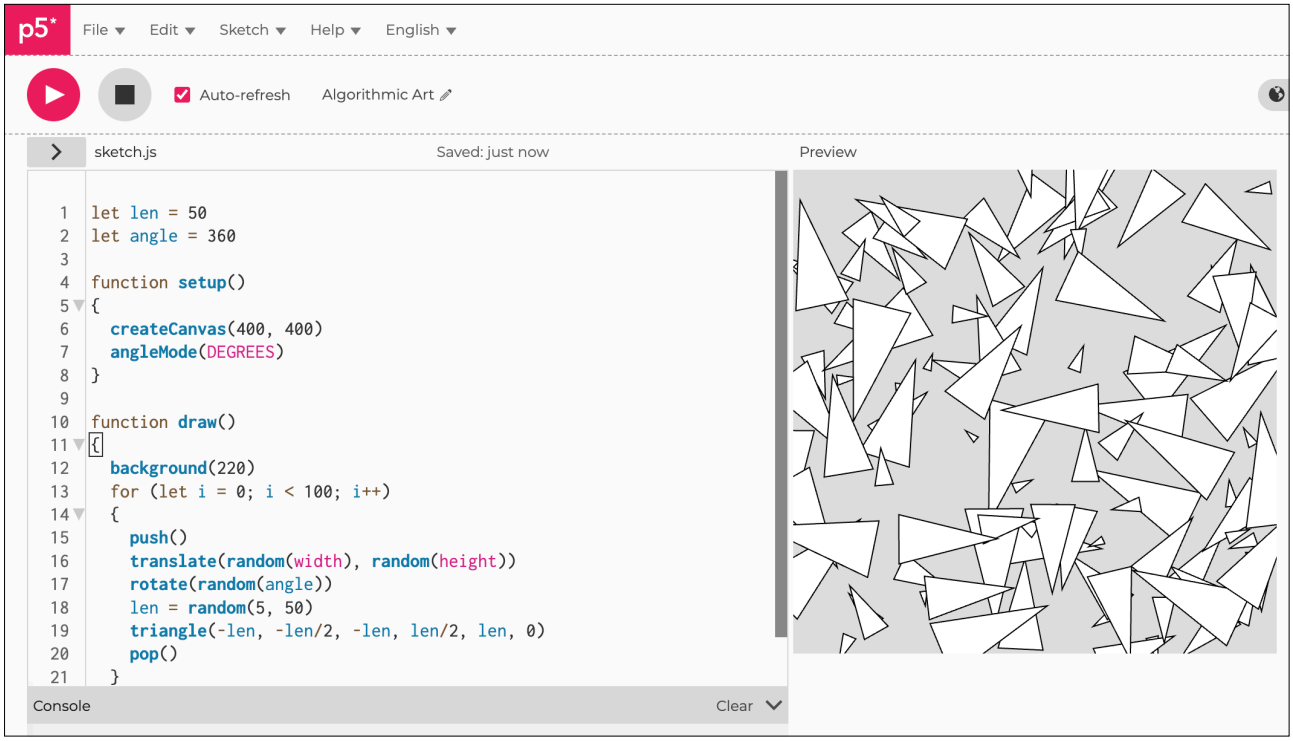
function setup()
{
  createCanvas(400, 400)
  angleMode(DEGREES)
}

function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(random(width), random(height))
    rotate(random(angle))
    len = random(5, 50)
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

Notes

There is a lot to play with here.

Figure A6.14


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
let len = 50
let angle = 360
function setup()
{
createCanvas(400, 400)
angleMode(DEGREES)
}
function draw()
{
background(220)
for (let i = 0; i < 100; i++)
{
push()
translate(random(width), random(height))
rotate(random(angle))
len = random(5, 50)
triangle(-len, -len/2, -len, len/2, len, 0)
pop()
}
Console
Clear



Sketch A6.15 random colouring

Make the background white and give each triangle a random colour (and alpha).

```
let len = 50
let angle = 360

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

function draw()
{
  background(255)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(random(width), random(height))
    rotate(random(angle))
    len = random(5, 50)
    fill(random(255), random(255), random(255), random(255))
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

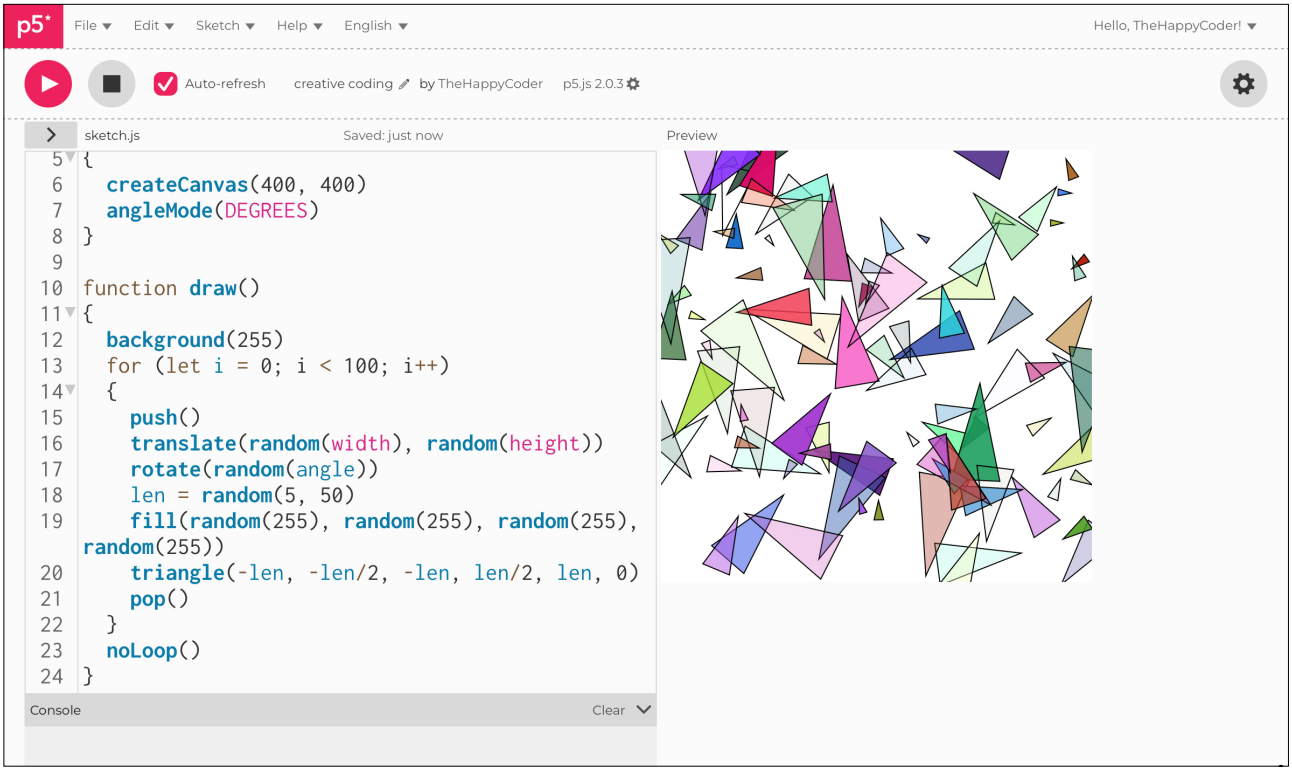
Notes

Starting to look interesting.

Challenges

1. What else could you develop alongside these ideas?
2. Different shapes, colours, etc.

Figure A6.15


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
creative coding
by TheHappyCoder
p5.js 2.0.3
sketch.js
Saved: just now
Preview
{
createCanvas(400, 400)
angleMode(DEGREES)
}
function draw()
{
background(255)
for (let i = 0; i < 100; i++)
{
push()
translate(random(width), random(height))
rotate(random(angle))
len = random(5, 50)
fill(random(255), random(255), random(255), random(255))
triangle(-len, -len/2, -len, len/2, len, 0)
pop()
}
noLoop()
}
Console
Clear

Algorithmic Art Module A Unit #7 Pixels





Module A Unit #7: pixels

Every image on your screen or on the canvas is made up of pixels. They are too tiny to be seen individually; even so, we can draw pixels using the point() function on the canvas. This is a very simple and early introduction to pixels as a shape we can draw. There is so much more to the pixels of the canvas or an image on the canvas, but that is for a bit later on.

One concept which is based on the for() loop is something called a nested loop. This is useful for pixel arrays and 3D objects and shapes.

Key concepts:

pixels
point()
nested loops
colorMode()

Sketch A7.1 what's the point?

! Our starting sketch

We have drawn a pixel at position (0, 0). Can you see it?
A `point()` shape is just one pixel shape.

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

function draw()
{
  background(220)
  point(0, 0)
}
```

Notes

You can't see it! Look closer at Fig. A7.1b, can you see it now?

Challenge

Give it a `strokeWeight(50)` to see it clearly.

Code Explanation

`point(0, 0)`	This draws a pixel at coordinates (0, 0).

Figure A7.1a

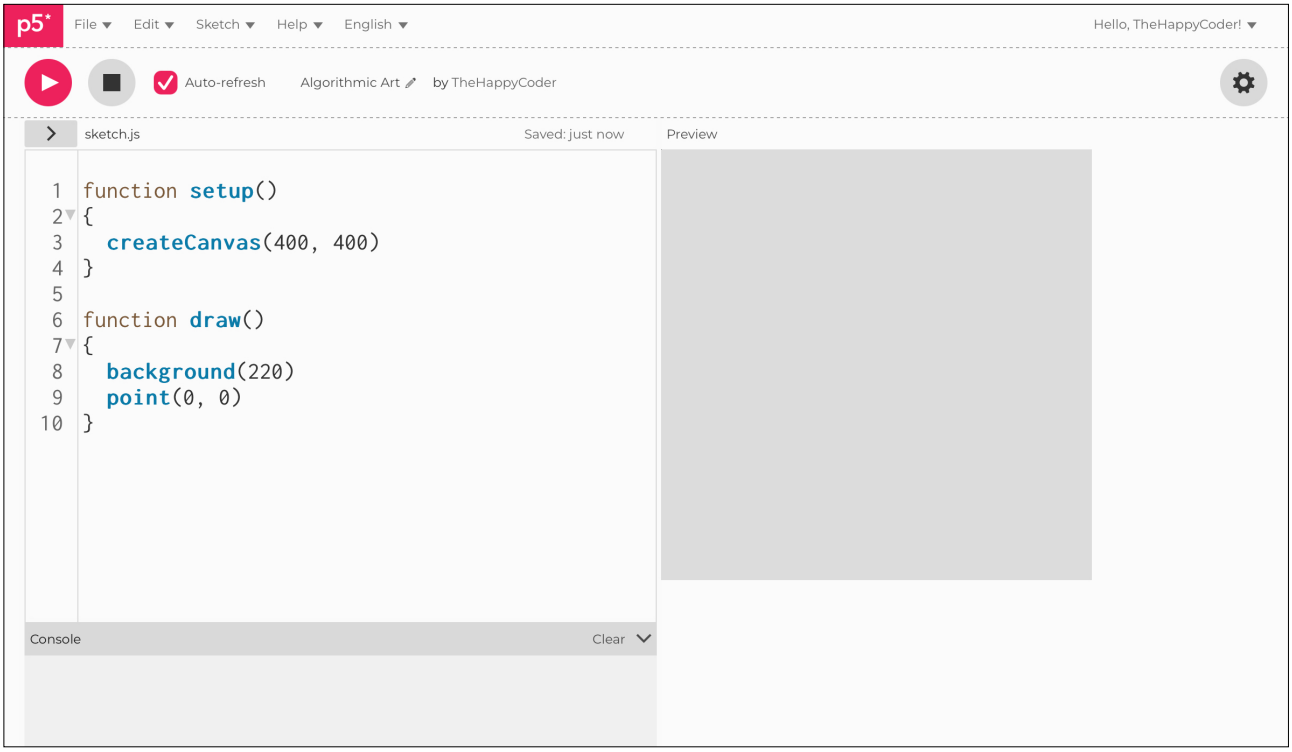
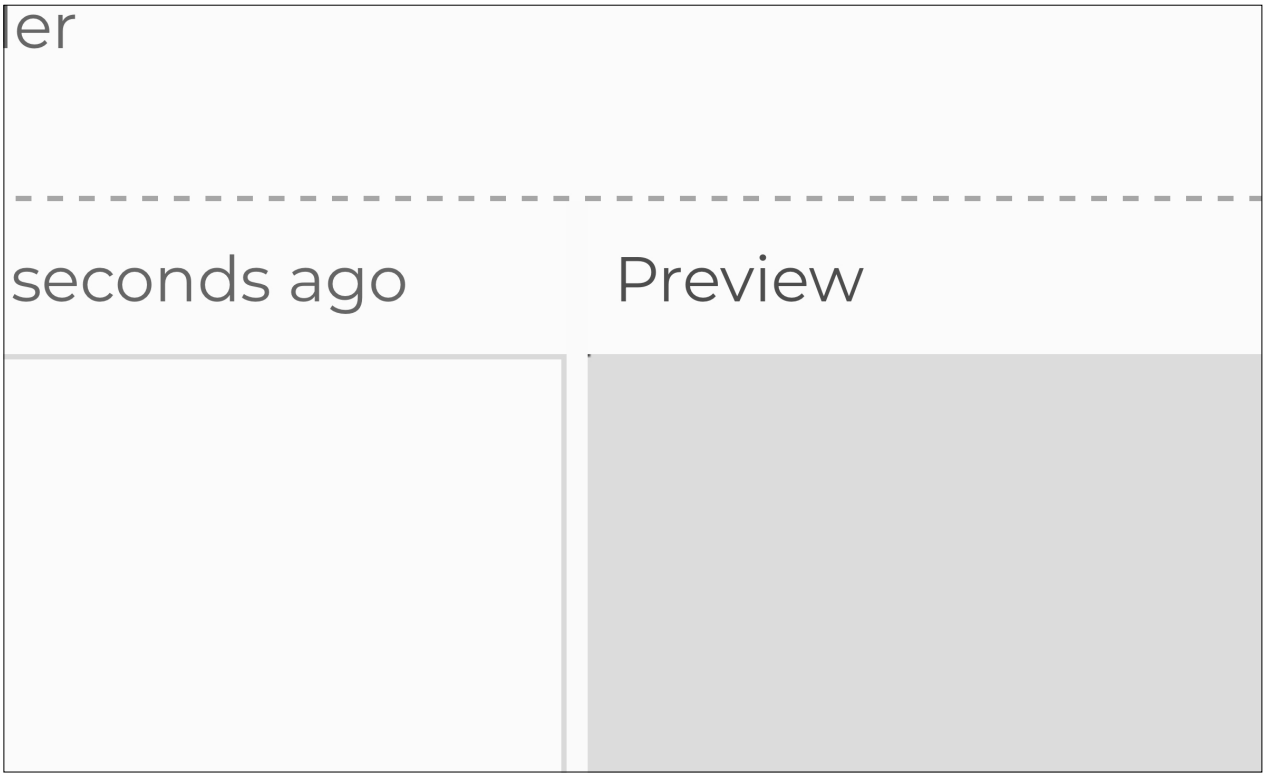


Figure A7.1b: pixel point



Sketch A7.2 a line of pixels

! Please make sure that you are using no higher than version 2.2.3, there seems to be a bug with version 2.3.x at present (time of writing)

We use a for() loop to draw a line across the top of the canvas. Remember to change point(0, 0) to point (x, 0).

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

function draw()
{
  background(220)
  for (let x = 0; x < width; x++)
  {
    point(x, 0)
  }
}
```

Notes

You should be able to see a thin line going across the top of the canvas. We used x instead of i for this for() loop.

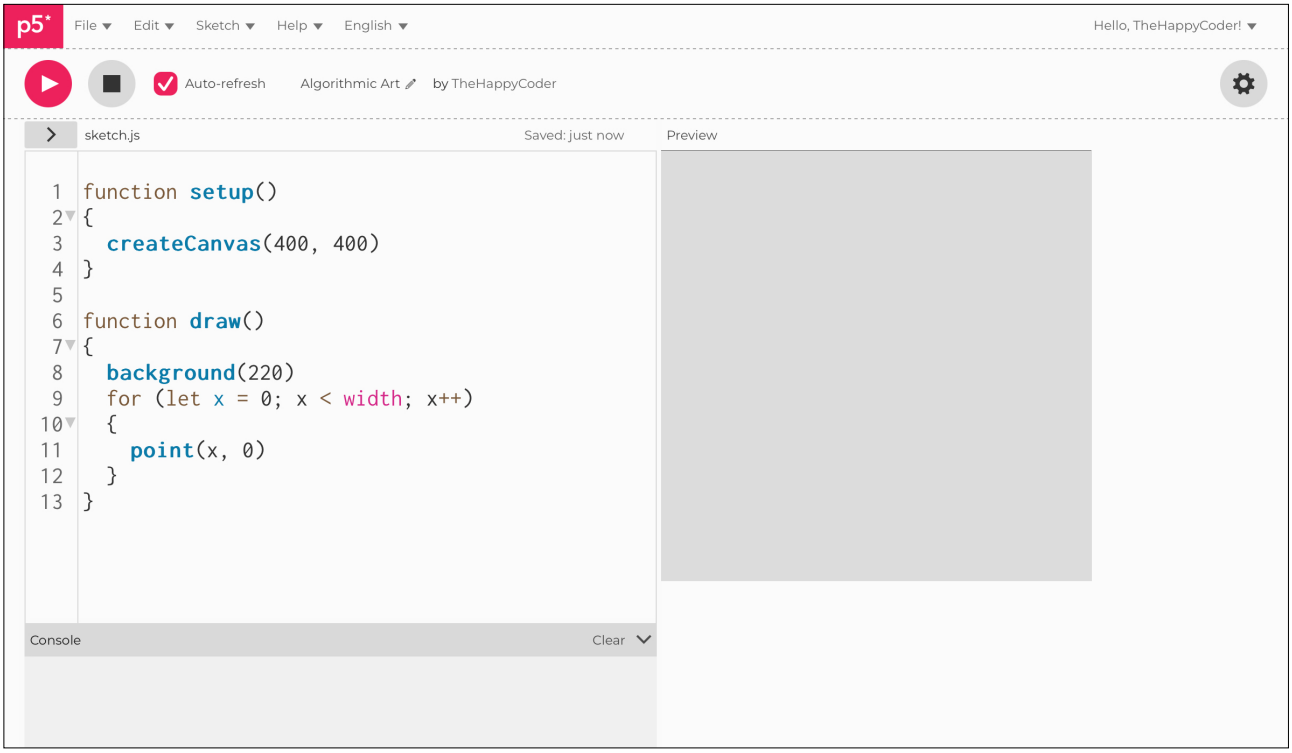
Challenge

Give it a higher strokeWeight(10) or so.

Code Explanation

for (let x = 0; x < width; x++)	A for() loop for the x value.
point(x, 0)	Draws a pixel for each x value.

Figure A7.2

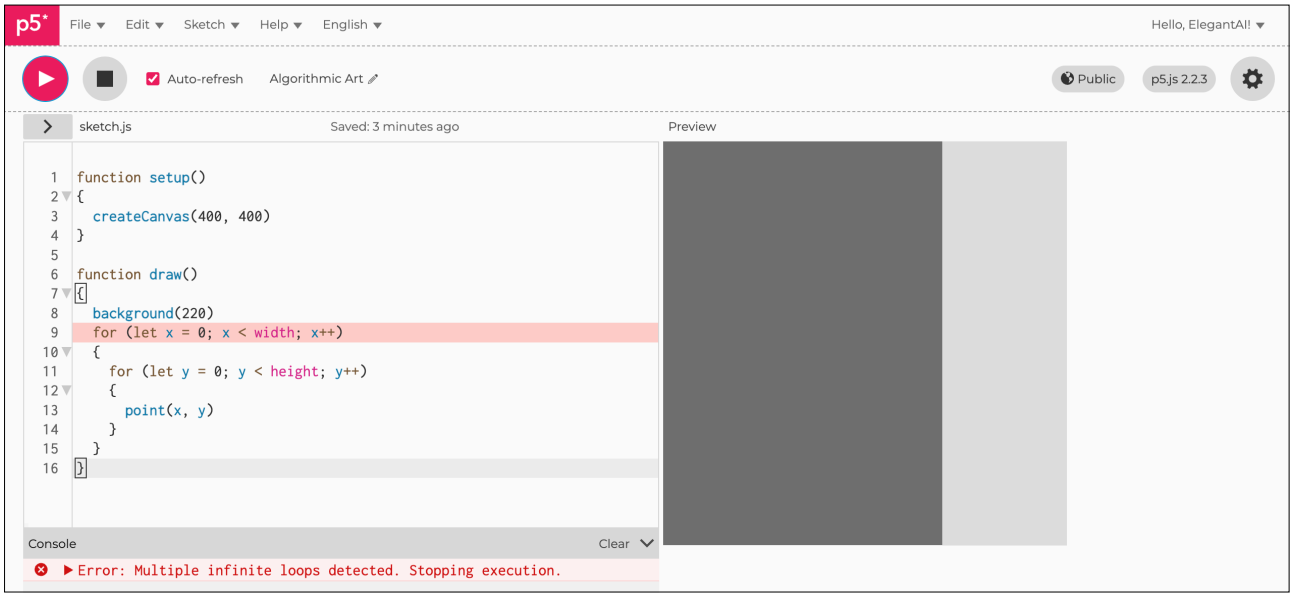

p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 for (let x = 0; x < width; x++)
10 {
11 point(x, 0)
12 }
13 }
Console
Clear



Infinite Loop problem

An issue that is coming up (especially as you progress through the workbooks) regularly is the infinite loop. This happens when you create a loop that never ends. What happens is that your sketch crashes and/or freezes and you have to close it all down and reopen p5.js editor again. To protect against that happening they have a built-in protection. This is what you might encounter if you do not cancel it. Here we do not have an infinite loop but just a very large one which it thinks might be an infinite loop. Notice that it stops 2/3 of the way across.

Figure 20: Infinite Loop Detected



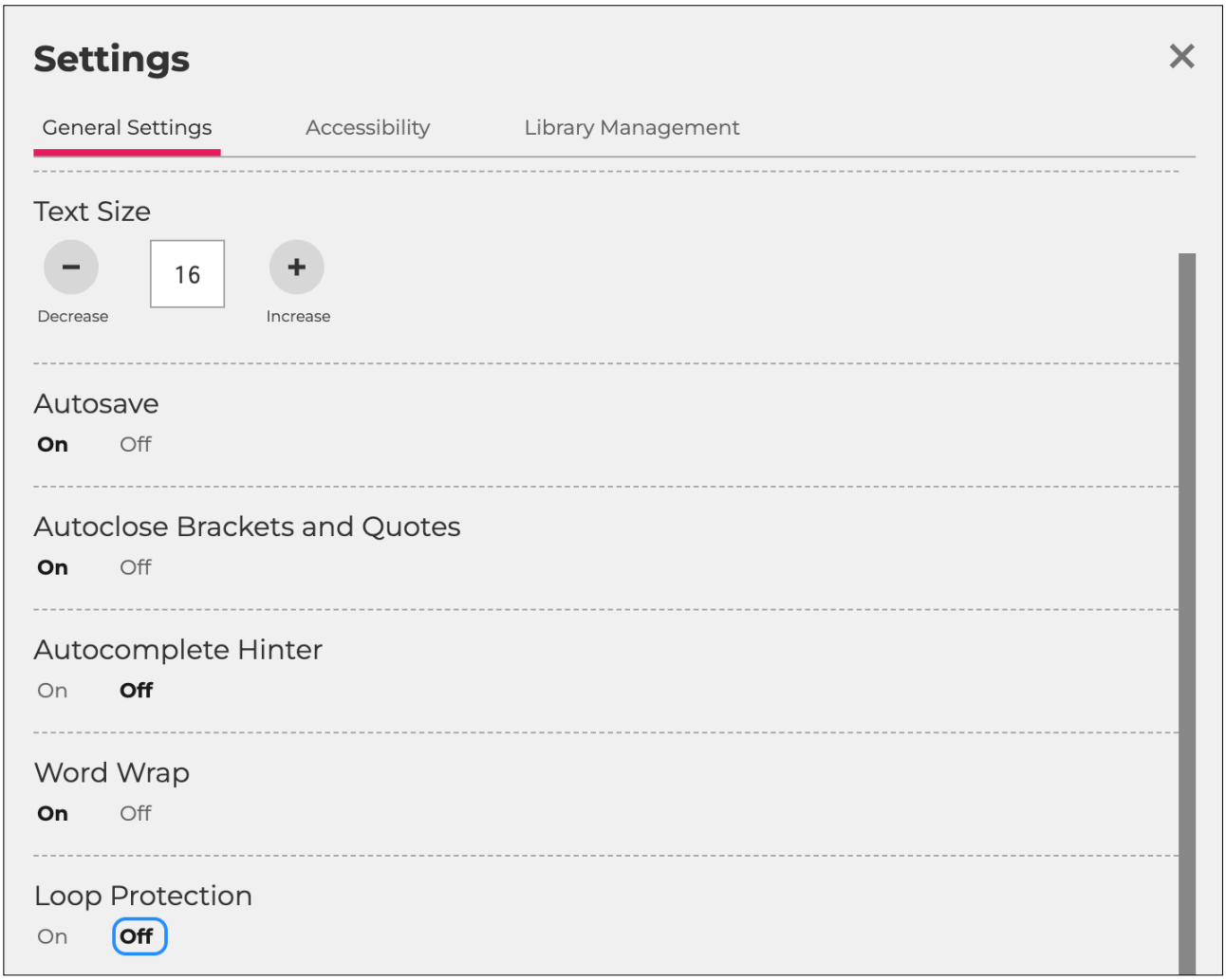


Infinite Loop solution

The solution is quite simple, go to the `Settings` icon (the `cog` on the top right) and scroll down until you come to the `Loop Protection` and toggle it `Off`. Every time you change versions of `p5.js` it is reset to `On` by default, so you need to toggle it `Off` each time.

Also, for the time being suggest sticking with version `2.2.3` for now while they iron out all the bugs.

Figure 21: Loop Protection toggle





Sketch A7.3 nested loop

Now we can use another `for()` loop inside the original `for()` loop to draw all the `y` components for each `x` component of the coordinates for each pixel. This I call a `nested loop`.

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

function draw()
{
  background(220)
  for (let x = 0; x < width; x++)
  {
    for (let y = 0; y < height; y++)
    {
      point(x, y)
    }
  }
}
```

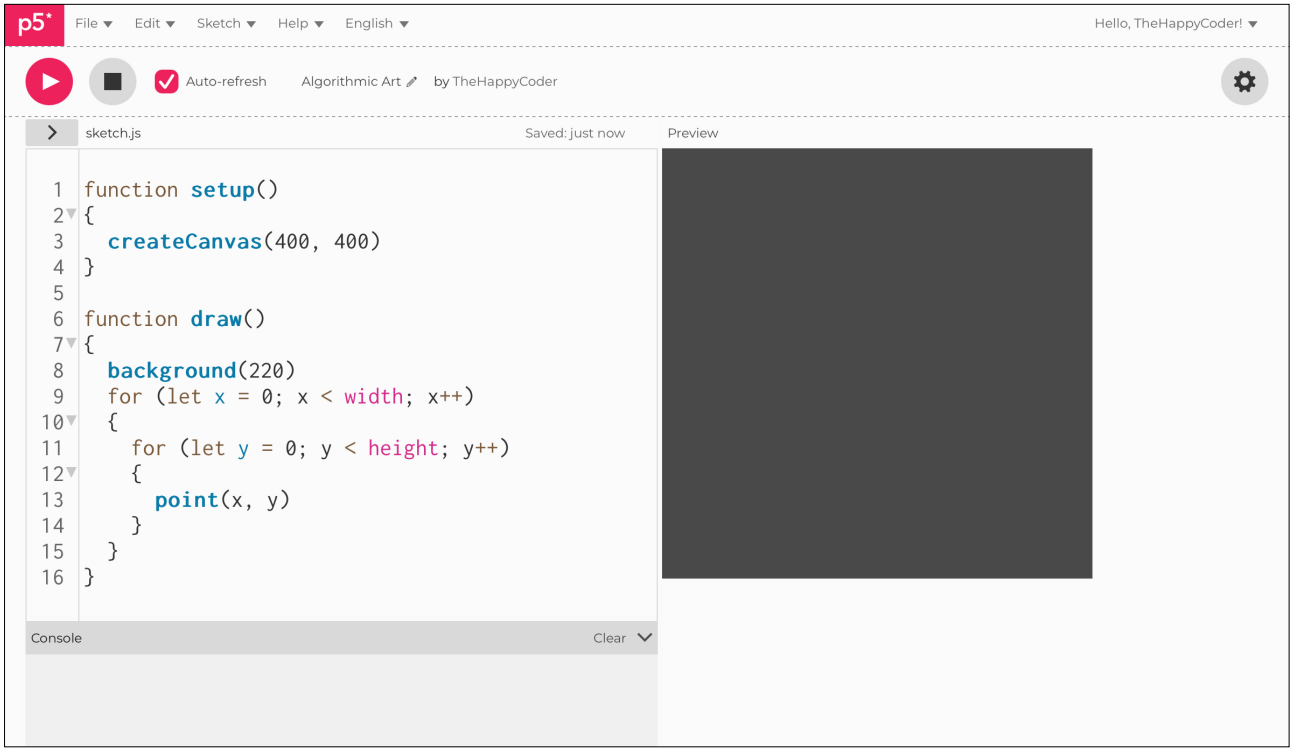
Notes

We have drawn a column of pixels working from left to right; this draws every pixel black. Remember to change `point(x, 0)` to `point(x, y)`.

Code Explanation

`for (let y = 0; y < height; y++)`	A for() loop for all the y values.
`point(x, y)`	Draw the pixel at (x, y) coordinates.

Figure A7.3


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
for (let x = 0; x < width; x++)
{
for (let y = 0; y < height; y++)
{
point(x, y)
}
}
}
Console
Clear



Sketch A7.4 colouring the pixel

We can give every pixel a colour using `stroke()`.

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

function draw()
{
  background(220)
  for (let x = 0; x < width; x++)
  {
    for (let y = 0; y < height; y++)
    {
      stroke('lightblue')
      point(x, y)
    }
  }
}
```

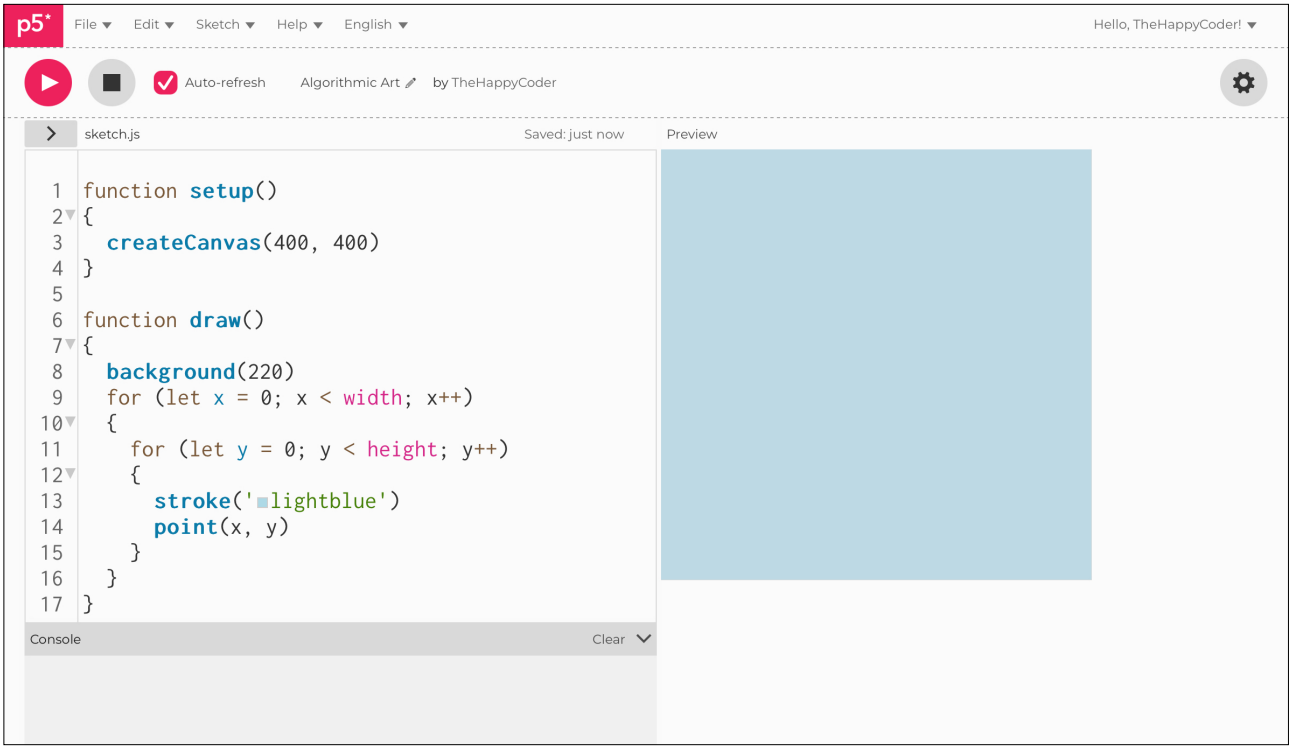
Notes

Our canvas is now a light blue colour.

Challenge

Try other colours.

Figure A7.4


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
for (let x = 0; x < width; x++)
{
for (let y = 0; y < height; y++)
{
stroke('lightblue')
point(x, y)
}
}
}
Console
Clear



Sketch A7.5 random pixel colour

Give each pixel a random greyscale colour.

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

function draw()
{
  background(220)
  for (let x = 0; x < width; x++)
  {
    for (let y = 0; y < height; y++)
    {
      stroke(random(255))
      point(x, y)
    }
  }
  noLoop()
}
```

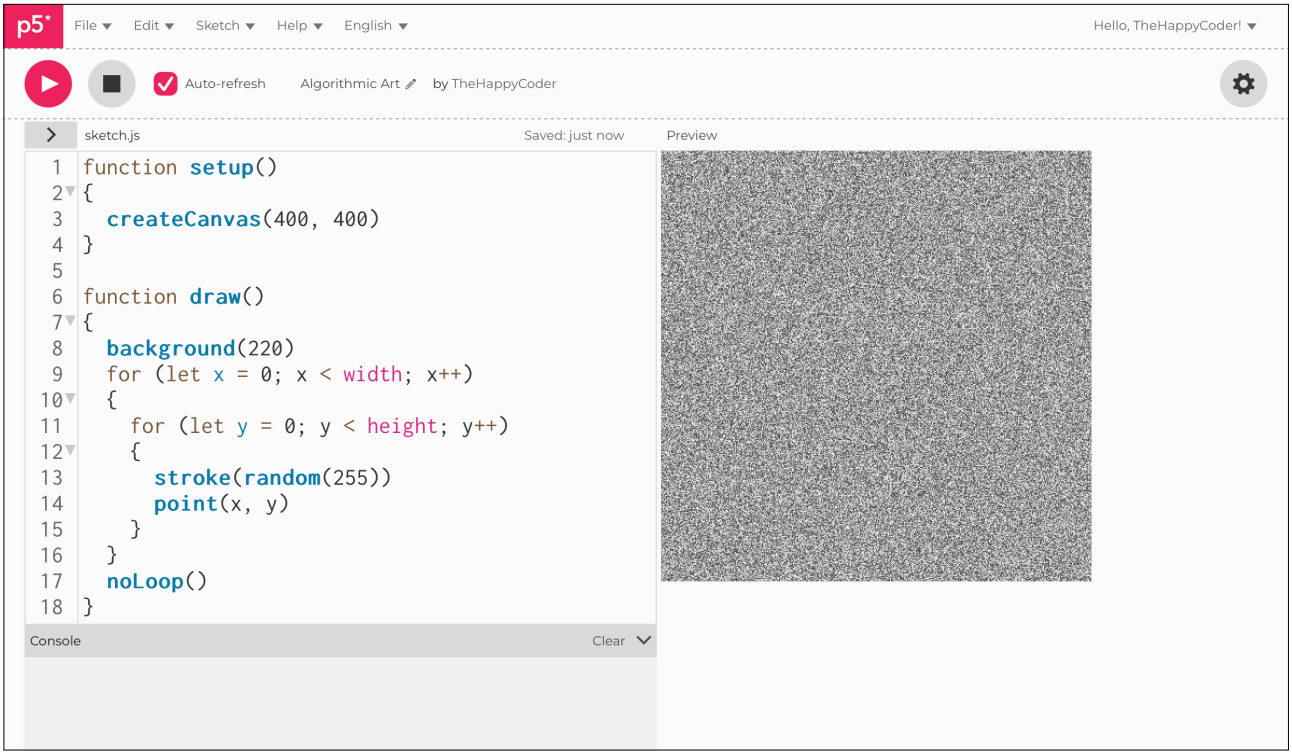
Notes

We get a random image, a bit like static on old TVs.

Challenge

Try random RGB colours.

Figure A7.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 for (let x = 0; x < width; x++)
10 {
11 for (let y = 0; y < height; y++)
12 {
13 stroke(random(255))
14 point(x, y)
15 }
16 }
17 noLoop()
18 }
Console
Clear



Sketch A7.6 gradual colour

Adding some colour to the pixels gradually.

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

function draw()
{
  background(220)
  for (let x = 0; x < width; x++)
  {
    for (let y = 0; y < height; y++)
    {
      stroke(x, 0, 0)
      point(x, y)
    }
  }
  noLoop()
}
```

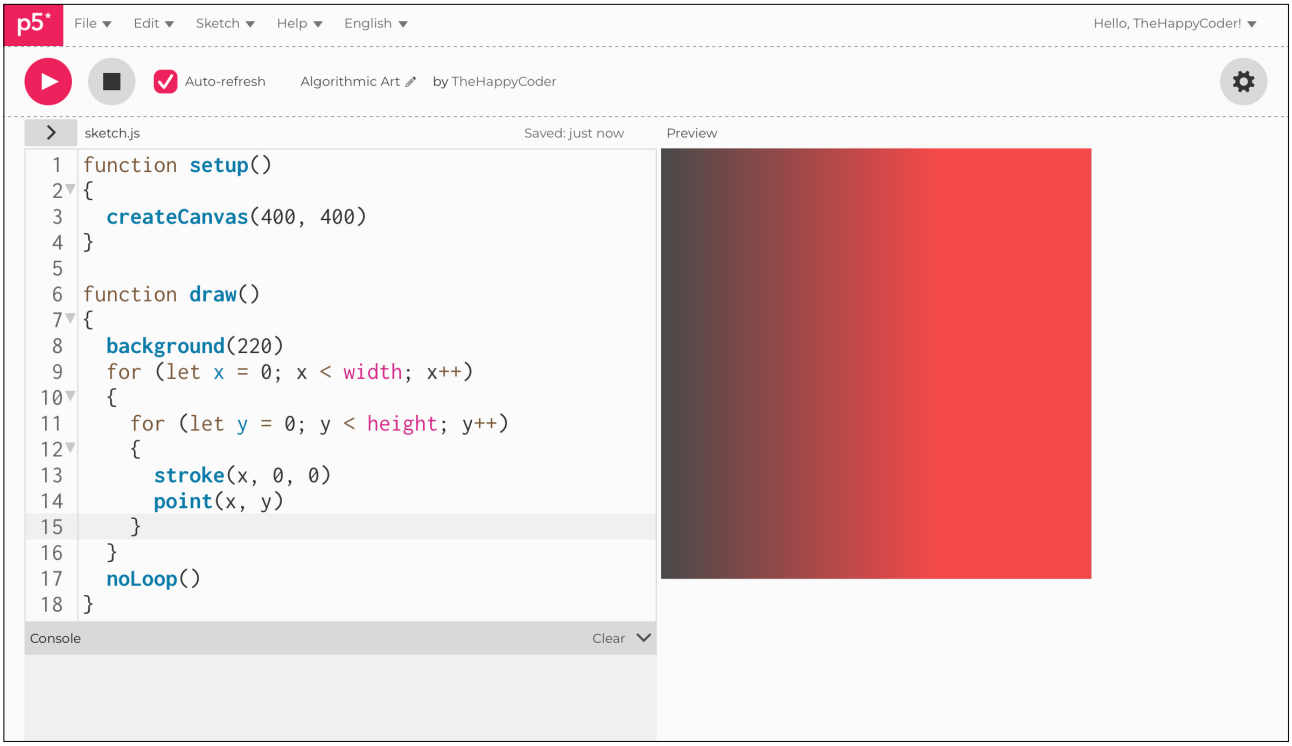
Notes

We increase the `x` value of red from `0` to `400`, but the RGB values stop at `255`; however, there is a way round this...

Challenge

Play with the values of the `stroke()` function.

Figure A7.6


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 for (let x = 0; x < width; x++)
10 {
11 for (let y = 0; y < height; y++)
12 {
13 stroke(x, 0, 0)
14 point(x, y)
15 }
16 }
17 noLoop()
18 }
Console
Clear

Sketch A7.7 colour mode RGB

The default `colorMode()` is RGB, so that is why we don't specify (more on other modes later). In this mode, we can specify the values of the red, green, and blue; they are extrapolated to any value you choose. We can give it an extra argument of `400`, as this is the dimension of the canvas.

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

function draw()
{
  background(220)
  for (let x = 0; x < width; x++)
  {
    for (let y = 0; y < height; y++)
    {
      stroke(x, 0, 0)
      point(x, y)
    }
  }
  noLoop()
}
```

Notes

Now we have a more graduated colouring of the pixels with a range of `0` to `400` (rather than `0` to `255`).

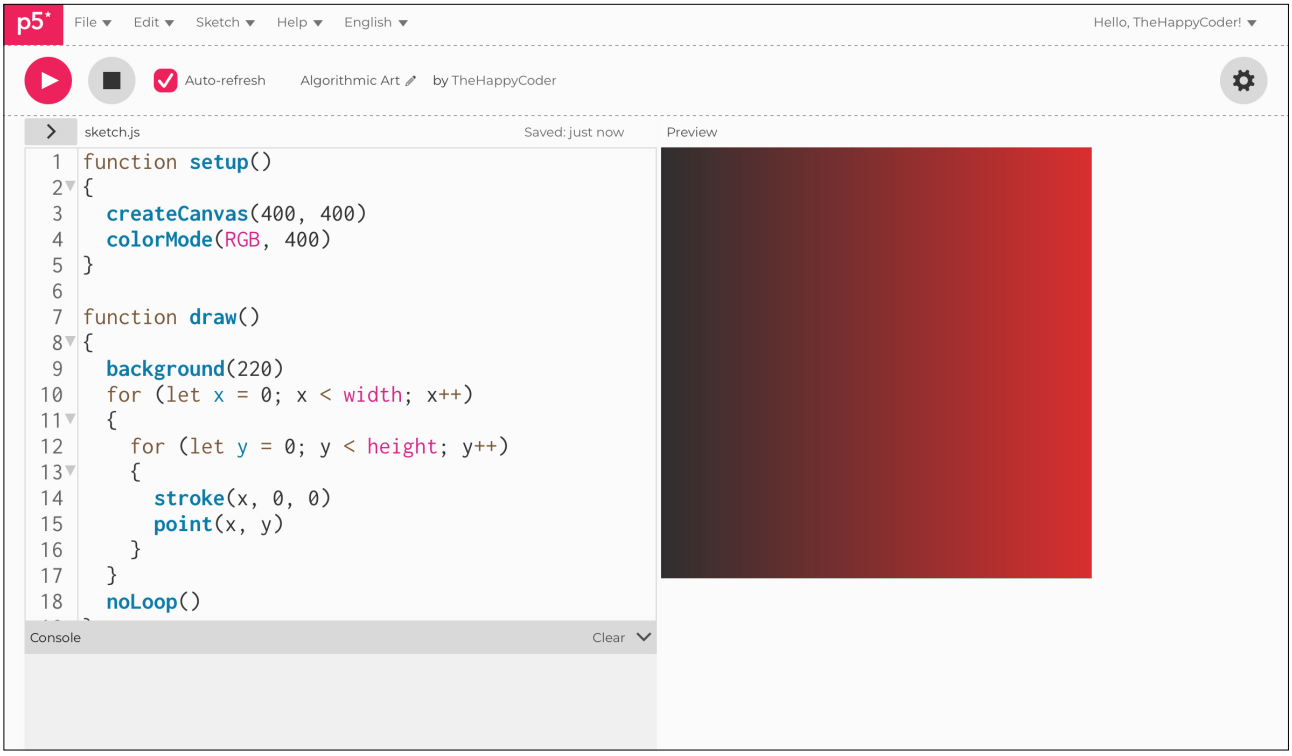
Challenges

1. Try other colours.
2. How would you start with a high value for red and work backwards?Clue: `stroke(400 - x, 0, x)`.

Code Explanation

`colorMode(RGB, 400)`	The colorMode() function allows us to change to other modes of colour and also change their properties.

Figure A7.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 colorMode(RGB, 400)
5 }
6
7 function draw()
8 {
9 background(220)
10 for (let x = 0; x < width; x++)
11 {
12 for (let y = 0; y < height; y++)
13 {
14 stroke(x, 0, 0)
15 point(x, y)
16 }
17 }
18 noLoop()
Console
Clear



Sketch A7.8 in both directions

If we add in the y values as well.

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

function draw()
{
  background(220)
  for (let x = 0; x < width; x++)
  {
    for (let y = 0; y < height; y++)
    {
      stroke(x, 0, y)
      point(x, y)
    }
  }
  noLoop()
}
```

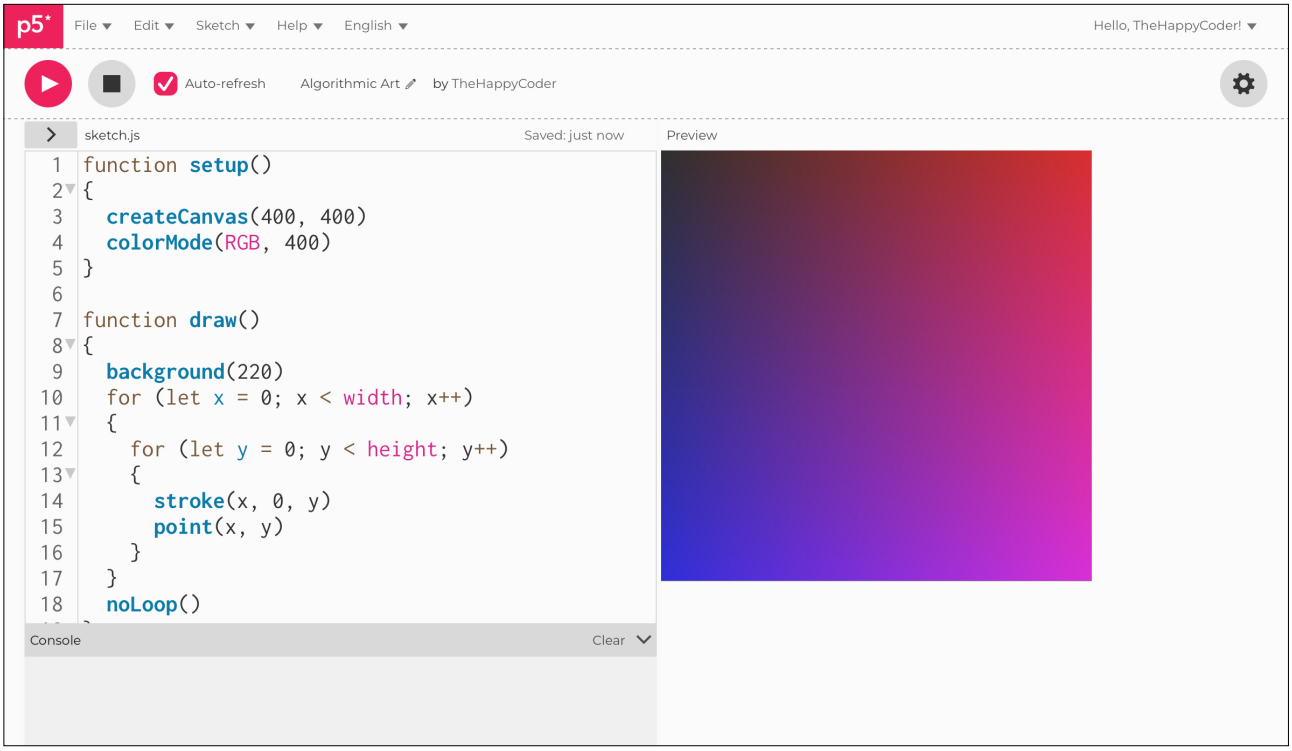
Notes

There is a lot you can do to play with these values.

Challenges

1. Try: `stroke(400 - x, 400 - y, y)`.
2. Experiment further.

Figure A7.8


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 colorMode(RGB, 400)
5 }
6
7 function draw()
8 {
9 background(220)
10 for (let x = 0; x < width; x++)
11 {
12 for (let y = 0; y < height; y++)
13 {
14 stroke(x, 0, y)
15 point(x, y)
16 }
17 }
18 noLoop()
Console
Clear



Algorithmic Art
Module A
Unit #8
10PRINT



Module A Unit #8: 10PRINT

It comes from an abbreviation of a single-line algorithm written in BASIC in the early 1980s: `10 PRINT CHR$(205.5+RND(1)); : GOTO 10`.

If you remember coding in BASIC, then you are possibly older than you look. This single line of code produces a random pattern, and this section uses this principle for creating random patterns.

This unit also introduces nested loops. There is such a huge scope to play with this, creating interesting designs and patterns with different shapes, colours, thicknesses, etc., so it is a great place to experiment.

Sketch A8.1 our starting sketch

Drawing a line from (0, 0) to (100, 100).

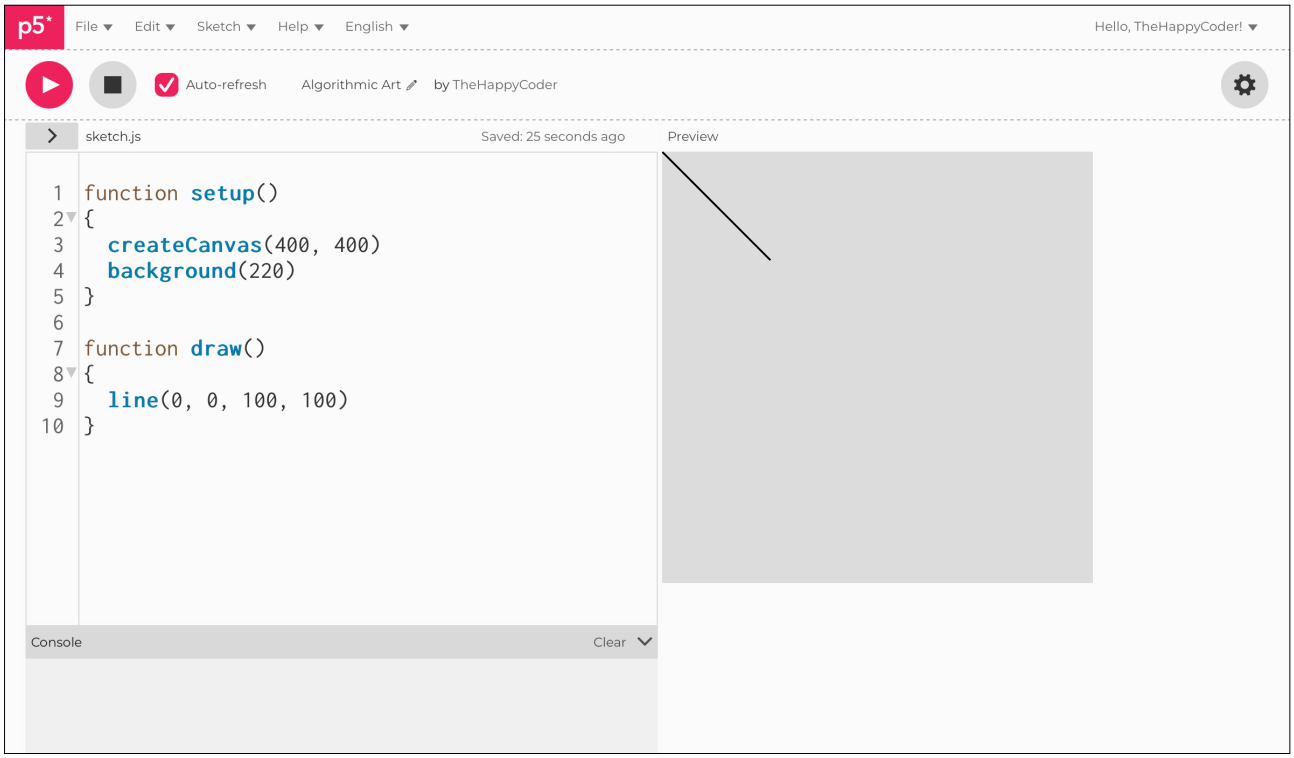
```
function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  line(0, 0, 100, 100)
}
```

Notes

This is our starting point.

Figure A8.1





Sketch A8.2 adding some variables

Adding some variables for x and y.

```
let x = 0
let y = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  line(0, 0, 100, 100)
}
```

Notes

We will need them later.



Sketch A8.3 the line function

This does nothing for the line; yet, all will be revealed in due course.

```
let x = 0
let y = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  line(x, y, x + 100, y + 100)
}
```

Notes

You should have the same effect.



Sketch A8.4 another variable

Instead of typing 100, we will create a variable called spacing and give that the value of 100, which will give us exactly the same result.

```
let x = 0
let y = 0
let spacing = 100

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
}
```

Notes

We now have easy control of the main variables.



Sketch A8.5 using the variable

Let's now change the value of the `spacing` variable to `10` and give the canvas a nice yellow background. Also, make the line dark red.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
}
```

Notes

Not a lot to show yet.

Challenge

Use different colours or line thickness.

Figure A8.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let spacing = 10
function setup()
{
createCanvas(400, 400)
background(255, 255, 0)
stroke(100, 0, 0)
}
function draw()
{
line(x, y, x + spacing, y + spacing)
}
Console
Clear



Sketch A8.6 adding the spacing

To draw evenly spaced objects in a row, we'll loop through them. You could use a fixed value like 10, but using a variable allows you to change it once at the beginning, and it will automatically apply to all instances of that variable.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
  x += spacing
}
```

Notes

This gives us a nice row of lines.

Figure A8.6


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let spacing = 10
function setup()
{
createCanvas(400, 400)
background(255, 255, 0)
stroke(100, 0, 0)
}
function draw()
{
line(x, y, x + spacing, y + spacing)
x += spacing
}
Console
Clear

Sketch A8.7 stop at the edge

However, the lines are being drawn beyond the canvas continually. What I would like to do is to start a new line when it gets to the right-hand edge. To do this, we use an `if()` statement. An `if()` statement checks to see if something has happened, and if it has, we tell it to do something different.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
  x += spacing
  if (x >= width)
  {
    x = 0
  }
}
```

Notes

The symbol `>=` means greater than or equal to. So if the value of `x` is greater than or equal to the width, it goes back to zero. So what you are seeing is the red line starting again from the left-hand edge every time it reaches the right-hand edge. You can't see it happen because it is writing it over and over again on the same line in a continuous loop.

Challenge

You can check to see if it works by putting a smaller number instead of width.

Code Explanation

`if (x >= width)`	Checks to see if x is greater than or equal to the width.



Sketch A8.8 let's go down

But what I would like is to start a new line after it gets to the edge, and so we need to nudge it down by the same amount. To do this, we add a `spacing` value to `y`.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
  x += spacing
  if (x >= width)
  {
    x = 0
    y += spacing
  }
}
```

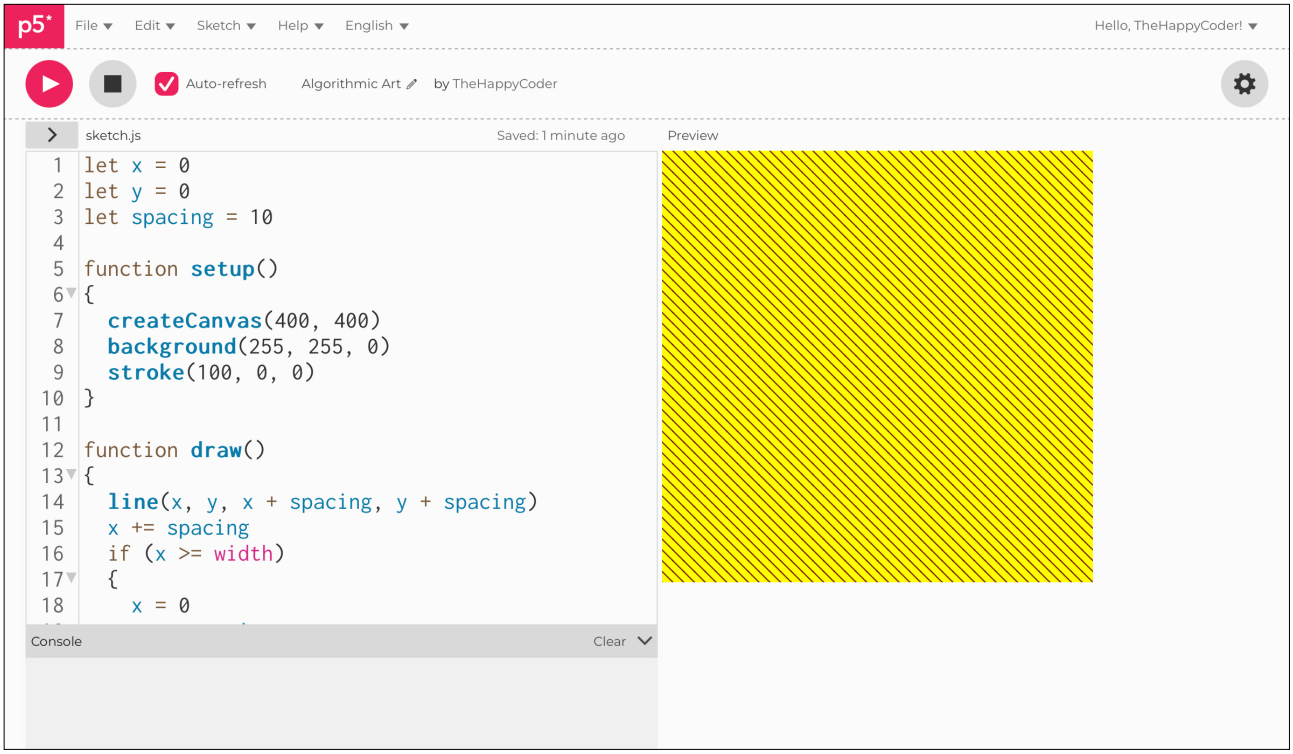
Notes

Every time we get to the edge, it starts a new line; however, it doesn't stop when it gets to the bottom. It is not critical, but neater if it did stop.

Code Explanation

`y += spacing`	Adds the value of spacing each time, accumulating as it goes.

Figure A8.8


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: 1 minute ago
Preview
let x = 0
let y = 0
let spacing = 10
function setup()
{
createCanvas(400, 400)
background(255, 255, 0)
stroke(100, 0, 0)
}
function draw()
{
line(x, y, x + spacing, y + spacing)
x += spacing
if (x >= width)
{
x = 0
Console
Clear



Sketch A8.9 stop when we get to the bottom

We use noLoop() when we get to the bottom; it stops the draw() function from looping round. You should get the same result as before.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
  x += spacing
  if (x >= width)
  {
    x = 0
    y += spacing
  }
  if (y >= height)
  {
    noLoop()
  }
}
```

Notes

Nothing new to see yet.

Code Explanation

if (y >= height)	Checks to see if the y value has reached the bottom of the canvas.
noLoop()	Instructs any loop to stop.



Sketch A8.10 sloping the other way

Let's change it so that it draws the line sloping the other way. We can use nearly the same code.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x + spacing, y, x, y + spacing)
  x += spacing
  if (x >= width)
  {
    x = 0
    y += spacing
  }
  if (y >= height)
  {
    noLoop()
  }
}
```

Notes

Now we have some changes, not a lot but some; the lines are sloping the opposite way. We just reorganised the line function. The logic is there, just follow it through.

Figure A8.10


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: 1 minute ago
Preview
8 background(255, 255, 0)
9 stroke(100, 0, 0)
10 }
11
12 function draw()
13 {
14 line(x + spacing, y, x, y + spacing)
15 x += spacing
16 if (x >= width)
17 {
18 x = 0
19 y += spacing
20 }
21 if (y >= height)
22 {
23 noLoop()
24 }
25 }
Console
Clear

Sketch A8.11 the other line 50%

We want to draw one line sloping one way 50% of the time and then sloping the other way 50% of the time in a random order. To achieve that, we can simply pick a number at random from 0 to 1, and if it is less than 0.5, do one of them; else, do the other.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  if (random(1) < 0.5)
  {
    line(x, y, x + spacing, y + spacing)
  }
  else
  {
    line(x + spacing, y, x, y + spacing)
  }
  x += spacing
  if (x >= width)
  {
    x = 0
    y += spacing
  }
  if (y >= height)
  {
    noLoop()
  }
}
```

Notes

We have an `if()...else` statement. You could use two `if()` statements, but this is a bit more elegant.

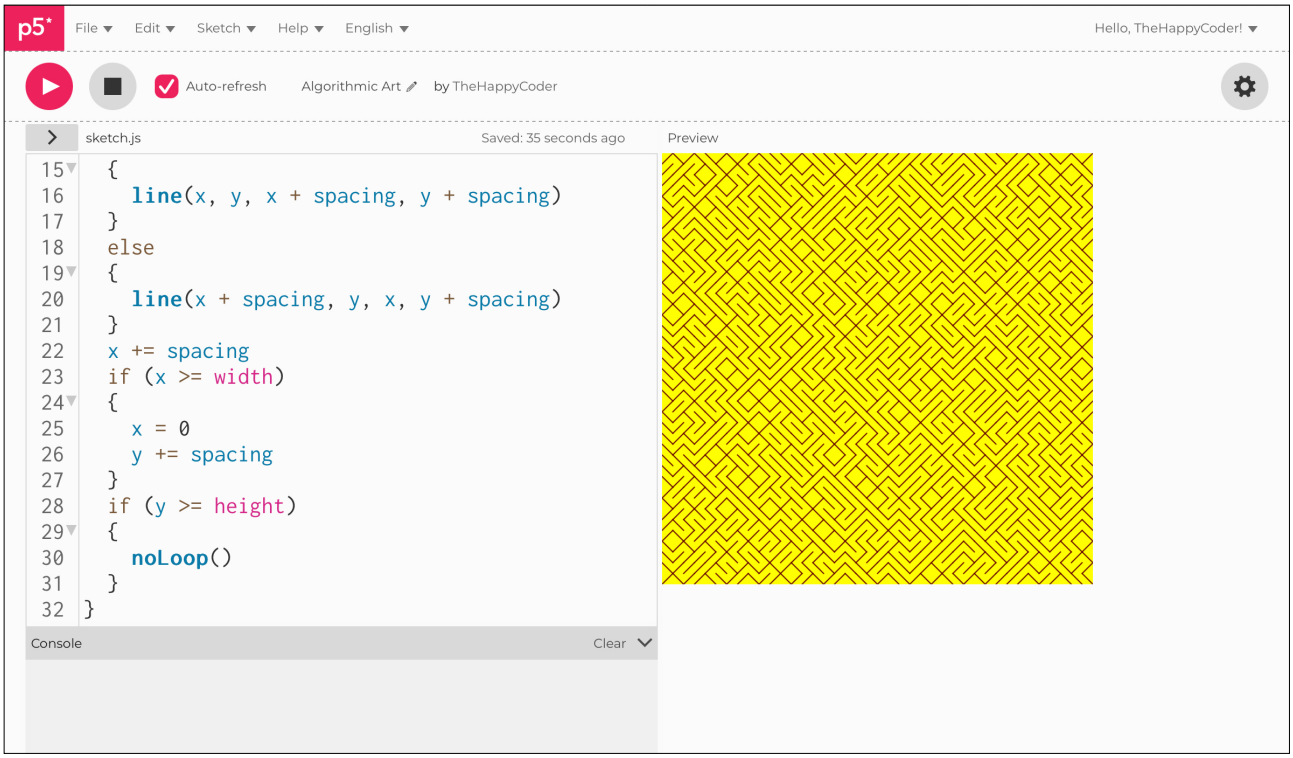
Challenge

1. What happens when you change the percentage, e.g. `0.5` to `0.8`?
2. Use other colours (random colours).
3. Thicker strokeWeight().

Code Explanation

`if (random(1) < 0.5)`	If the random element is less than 0.5, do this, or else do that.
`else`	A useful addition to the if() statement.

Figure A8.11



Sketch A8.12 variations on a theme

! Start a completely newish sketch

Instead of a line, we are going to draw a `square`. We will draw it at the top corner. The `square` takes three arguments: the first two are the `x` and `y` co-ordinates, and the third is the length of the side (`20`). We will use the variable `spacing` as it will come in handy.

```
let x = 0
let y = 0
let spacing = 20

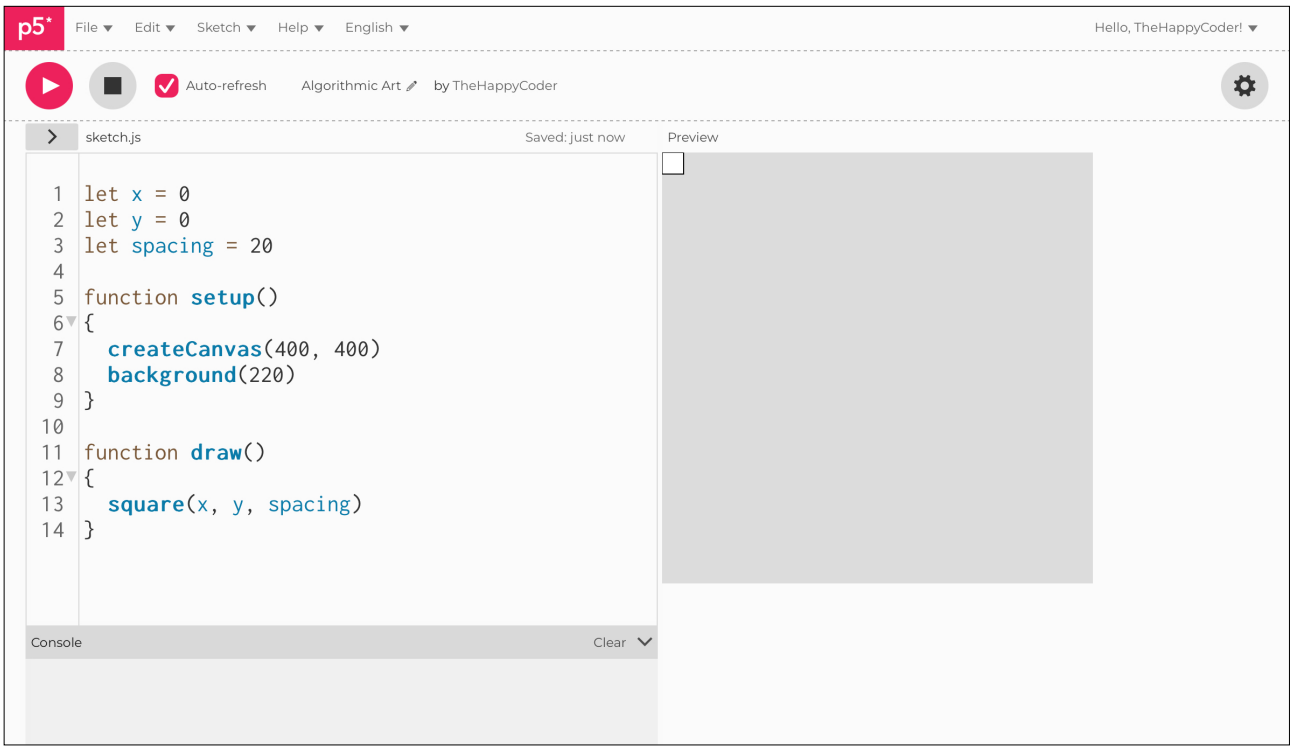
function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  square(x, y, spacing)
}
```

Notes

We get a single, solitary square.

Figure A8.12


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let spacing = 20
function setup()
{
createCanvas(400, 400)
background(220)
}
function draw()
{
square(x, y, spacing)
}
Console
Clear



Sketch A8.13 A row of squares

We are going to draw a row of them using a `for()` loop. A quick reminder, the `for()` loop has three parts to it. The first part (`let i = 0;`) creates a variable called `i` and gives it a value of `0`. The second part tells the loop when to stop, in this case when `i` gets to `width` (`i < width;`). The third part is how many steps (jumps) it takes (`i += spacing`). We set the `spacing` to `20`, so we are jumping in `20`s from `0` to `400` (the `width`).

```
let x = 0
let y = 0
let spacing = 20

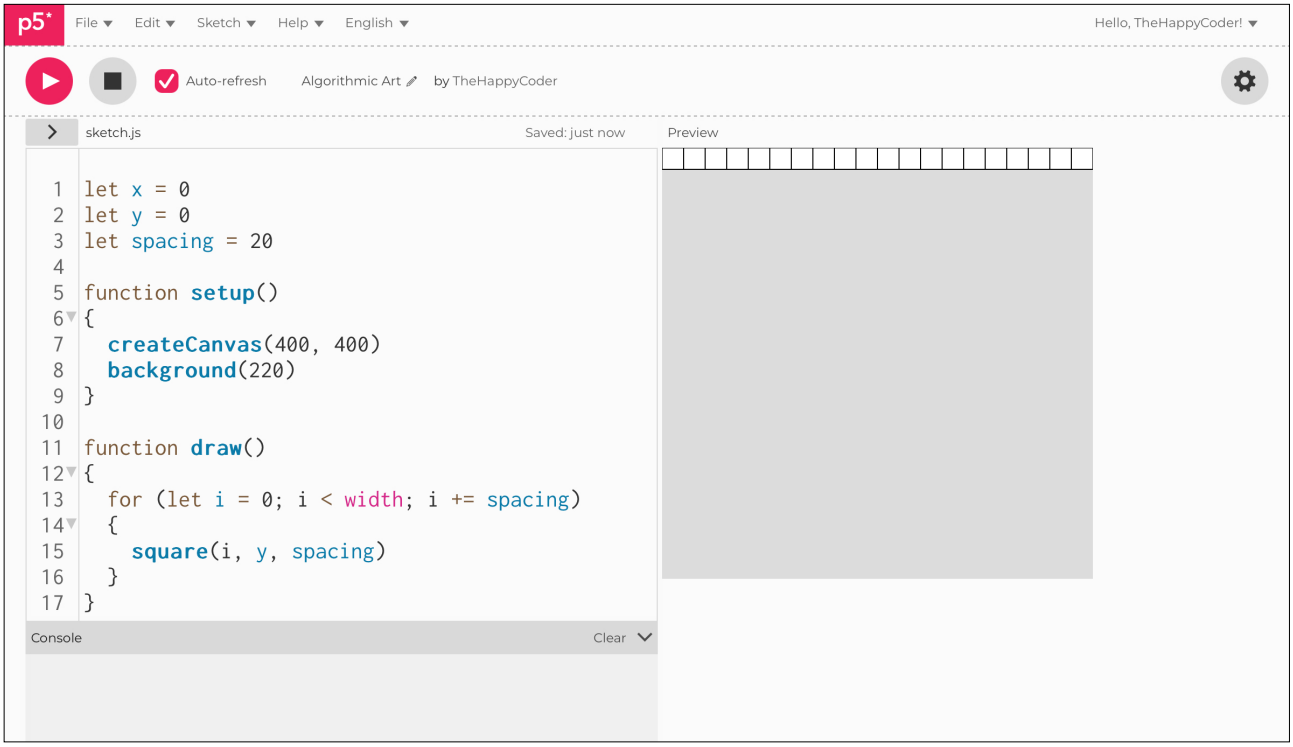
function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  for (let i = 0; i < width; i += spacing)
  {
    square(i, y, spacing)
  }
}
```

Notes

Not solitary anymore. We could've used `x` instead of `i`.

Figure A8.13


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let spacing = 20
function setup()
{
createCanvas(400, 400)
background(220)
}
function draw()
{
for (let i = 0; i < width; i += spacing)
{
square(i, y, spacing)
}
}
Console
Clear



Sketch A8.14 rows and columns

Adding columns as well as rows to fill the canvas, here we have a nested loop for the y value.

```
let x = 0
let y = 0
let spacing = 20

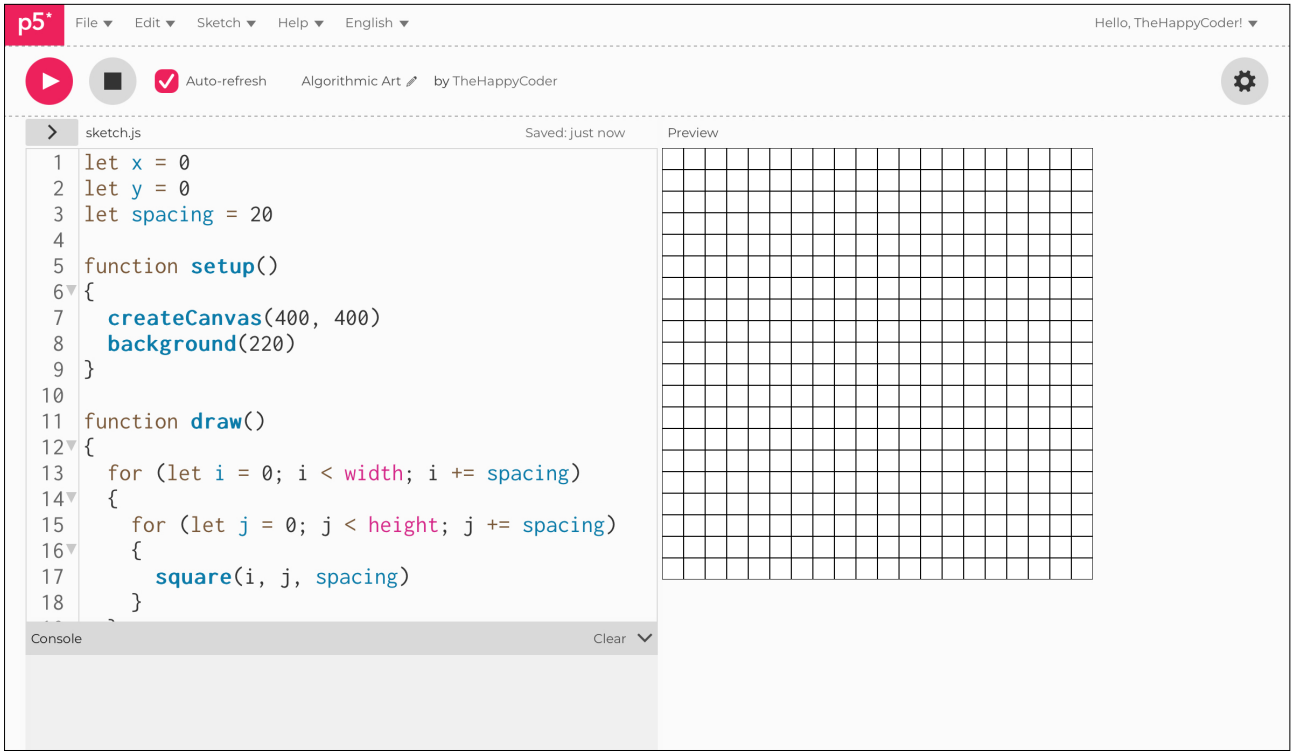
function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  for (let i = 0; i < width; i += spacing)
  {
    for (let j = 0; j < height; j += spacing)
    {
      square(i, j, spacing)
    }
  }
}
```

Notes

A canvas full of squares. We don't need a `noLoop()` because it stops drawing them at the end of the loops. This is also where variables become very useful. The `i` and `j` are just convention; we could've used `x` and `y` as in a previous example. If you need a third variable, we could use `k` (or `z`) for a third loop.

Figure A8.14


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let spacing = 20
function setup()
{
createCanvas(400, 400)
background(220)
}
function draw()
{
for (let i = 0; i < width; i += spacing)
{
for (let j = 0; j < height; j += spacing)
{
square(i, j, spacing)
}
Console
Clear

Sketch A8.15 random size squares

We can have some more fun with random, instead of having the same size squares. What if we change the size every time we draw one? We will need a `noLoop()` function now to stop it waggling all the time.

```
let x = 0
let y = 0
let spacing = 20

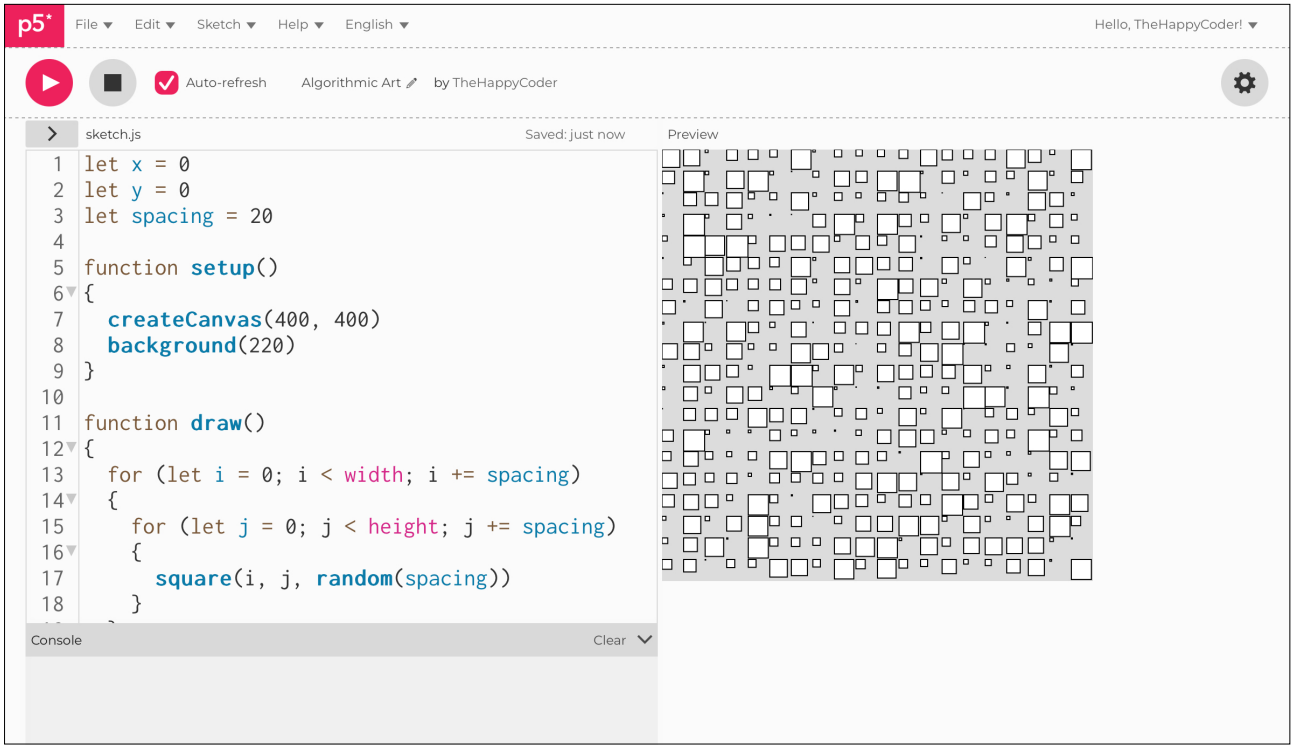
function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  for (let i = 0; i < width; i += spacing)
  {
    for (let j = 0; j < height; j += spacing)
    {
      square(i, j, random(spacing))
    }
  }
  noLoop()
}
```

Notes

Nice, although it still doesn't look quite right.

Figure A8.15


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let spacing = 20
function setup()
{
createCanvas(400, 400)
background(220)
}
function draw()
{
for (let i = 0; i < width; i += spacing)
{
for (let j = 0; j < height; j += spacing)
{
square(i, j, random(spacing))
}
Console
Clear



Sketch A8.16 using rectMode(CENTER)

A little bit of refactoring. At the moment, the co-ordinates of the `square` are at the top-left corner of the `square`. To change that so that the co-ordinates are in the centre of the `square`, we use a function called `rectMode(CENTER)`. Also, to give it a border, we start with `spacing` for `i` and `j` rather than `0`.

```
let x = 0
let y = 0
let spacing = 20

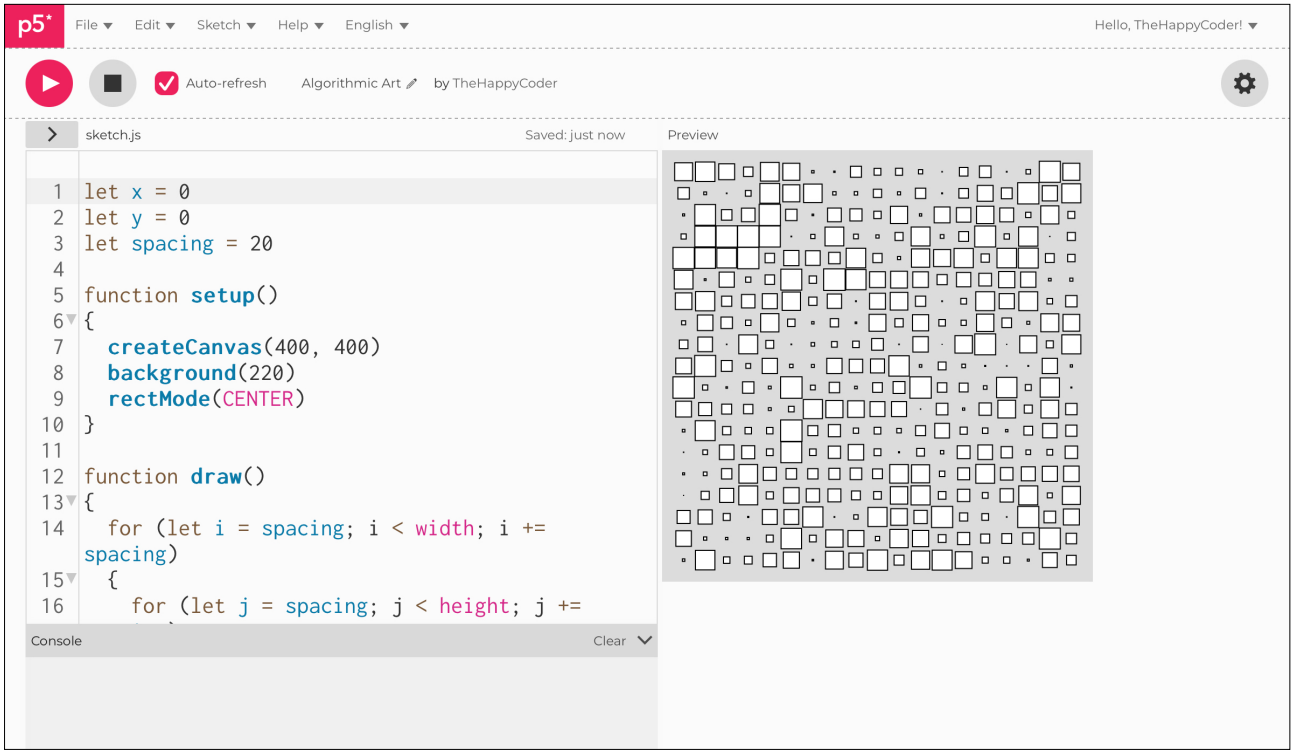
function setup()
{
  createCanvas(400, 400)
  background(220)
  rectMode(CENTER)
}

function draw()
{
  for (let i = spacing; i < width; i += spacing)
  {
    for (let j = spacing; j < height; j += spacing)
    {
      square(i, j, random(spacing))
    }
  }
  noLoop()
}
```

Notes

Looks a lot better, but we can improve more.

Figure A8.16


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let spacing = 20
function setup()
{
createCanvas(400, 400)
background(220)
rectMode(CENTER)
}
function draw()
{
for (let i = spacing; i < width; i +=
spacing)
{
for (let j = spacing; j < height; j +=
Console
Clear



Sketch A8.17 random colours

A bit more fun will be to colour them each randomly. We will also make the background white to show up the colours, adding a small gap between `random (spacing - 2)`.

```
let x = 0
let y = 0
let spacing = 20

function setup()
{
  createCanvas(400, 400)
  background(255)
  rectMode(CENTER)
}

function draw()
{
  for (let i = spacing; i < width; i += spacing)
  {
    for (let j = spacing; j < height; j += spacing)
    {
      fill(random(255), random(255), random(255))
      square(i, j, random(spacing - 2))
    }
  }
  noLoop()
}
```

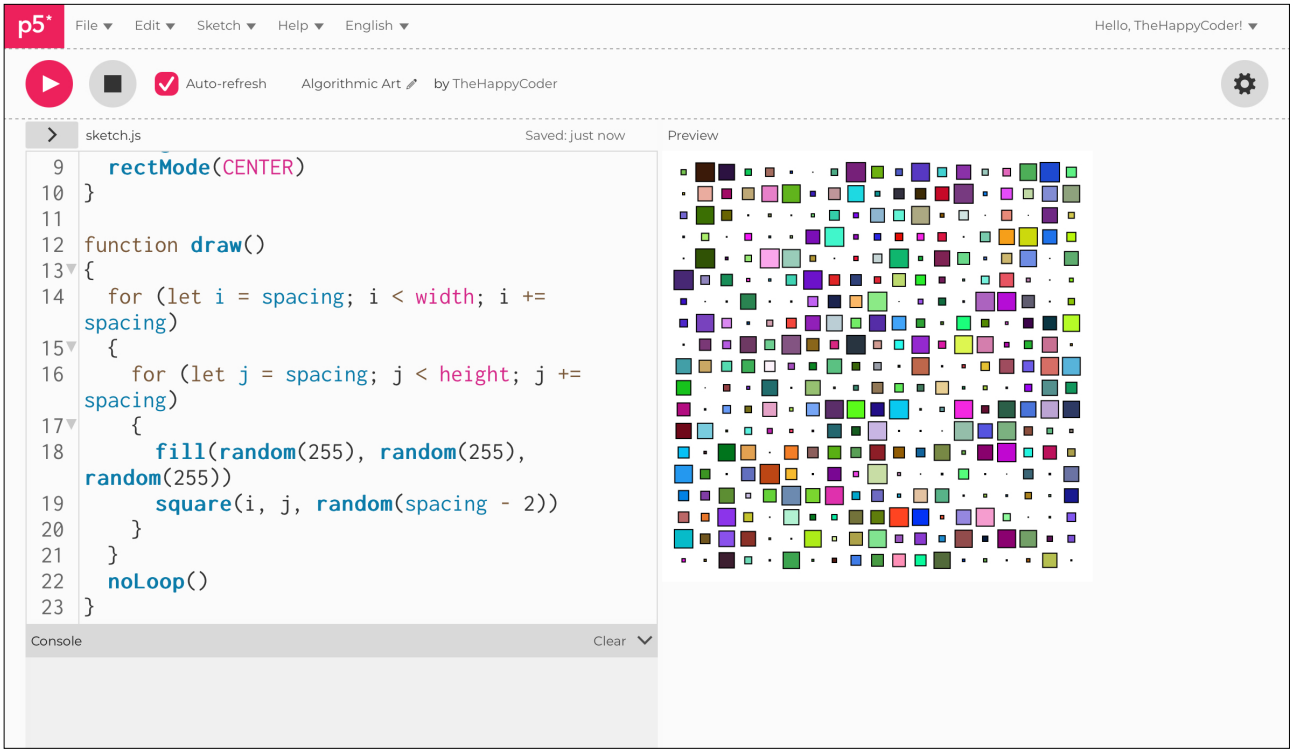
Notes

Looking good.

Challenge

Try some other shapes.

Figure A8.17


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
sketch.js
Saved: just now
Preview
rectMode(CENTER)
}
function draw()
{
for (let i = spacing; i < width; i += spacing)
{
for (let j = spacing; j < height; j += spacing)
{
fill(random(255), random(255), random(255))
square(i, j, random(spacing - 2))
}
}
noLoop()
}
Console
Clear