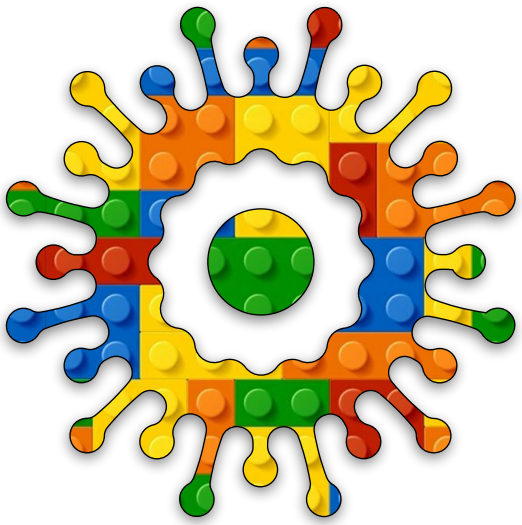
Creative Coding
Module B
Unit #10
phyllotaxis



Content

Module B Unit #10 Phyllotaxis

Introduction to Phyllotaxis

In nature, there is something called the golden ratio and the golden angle, which is a mathematical explanation for the pattern on, say, a sunflower. We are going to recreate the pattern that you can see in nature. If you want to know more, I suggest that you Google it and read up about it for yourself.

It will probably cover a lot of maths, but it is simple maths that can be coded, as you will see towards the end of this unit. But first, let's draw some circles and spirals to get us started.

Sketch B10.1 start with a point

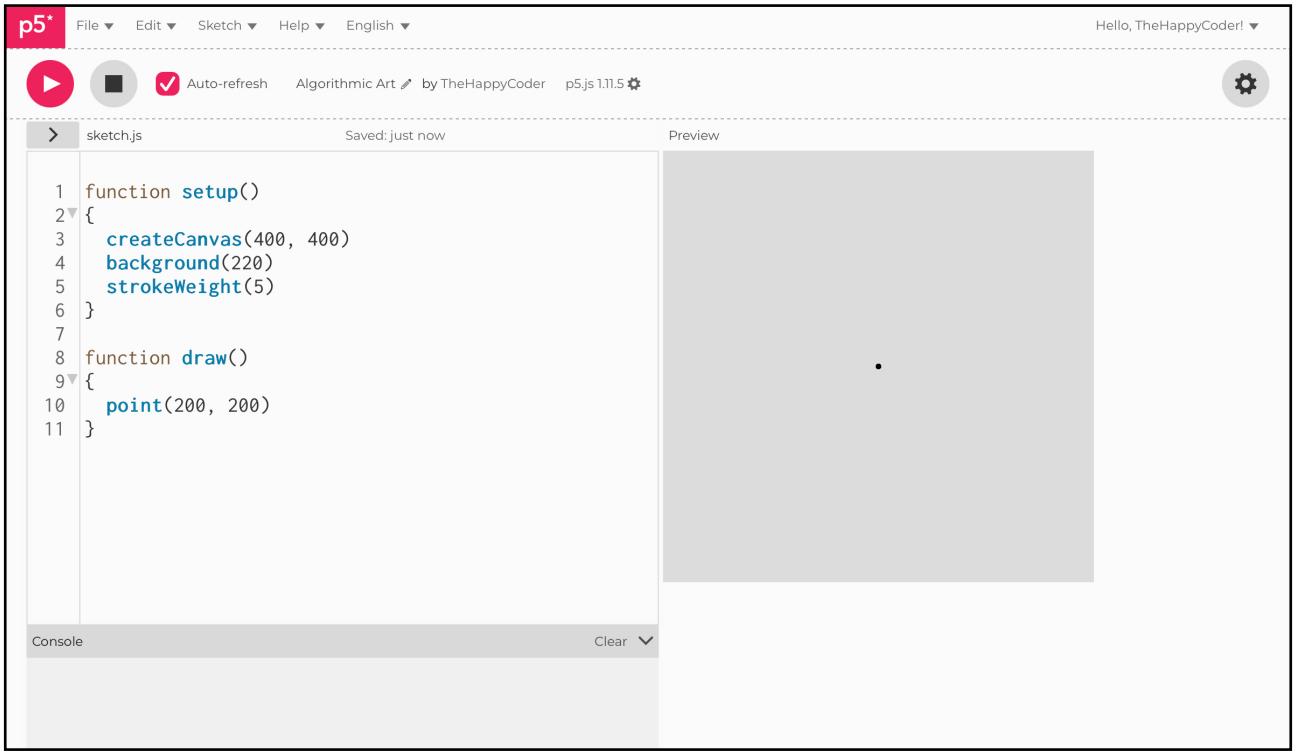
A simple point in the centre of the canvas.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
  strokeWeight(5)
}

function draw()
{
  background(220)
  point(200, 200)
}
```

Notes

Giving it a heavy stroke weight so we can see it.

Figure B10.1


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 background(220)
5 strokeWeight(5)
6 }
7
8 function draw()
9 {
10 point(200, 200)
11 }
Console
Clear

Sketch B10.2 the x and y variables

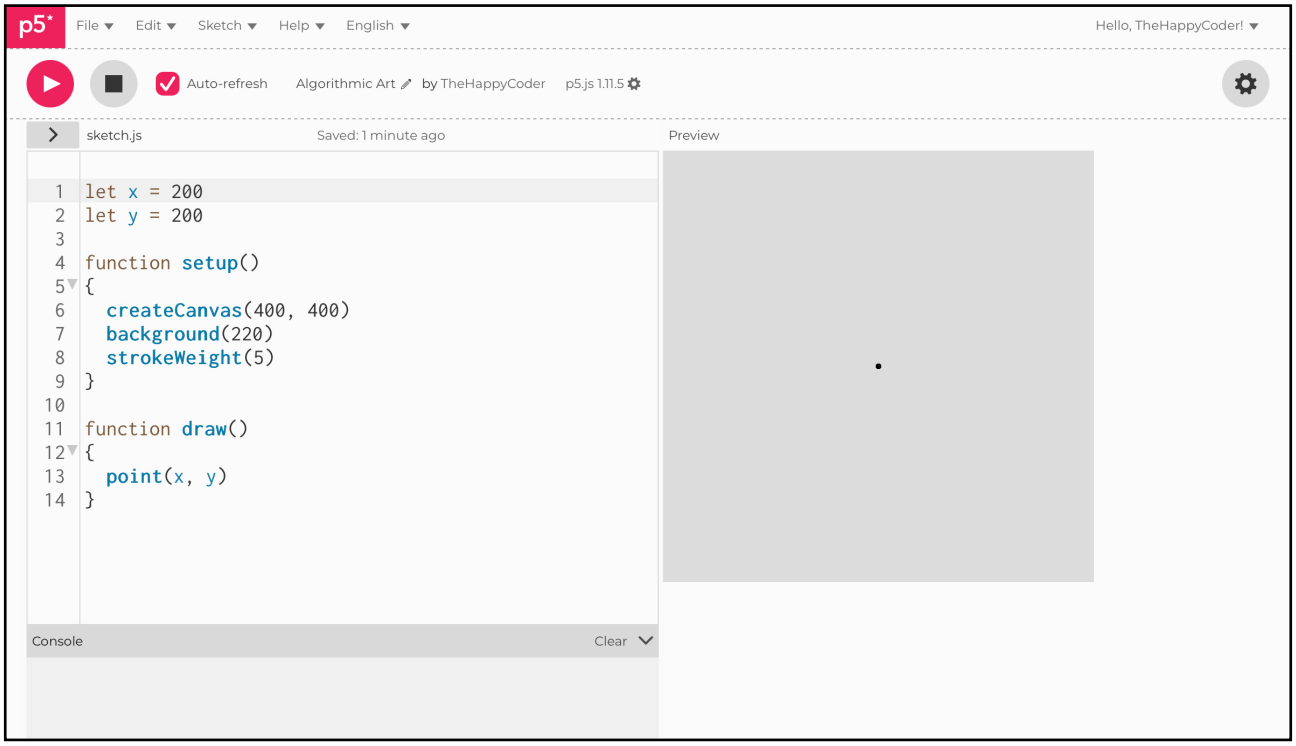
We will give it an x and y variable but still draw it in the centre.

```
let x = 200
let y = 200

function setup()
{
  createCanvas(400, 400)
  background(220)
  strokeWeight(5)
}

function draw()
{
  point(x, y)
}
```

Notes

You should still have the point in the centre, just as before.

Figure B10.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 1 minute ago
Preview
let x = 200
let y = 200
function setup()
{
createCanvas(400, 400)
background(220)
strokeWeight(5)
}
function draw()
{
point(x, y)
}
Console
Clear

Sketch B10.3 translate

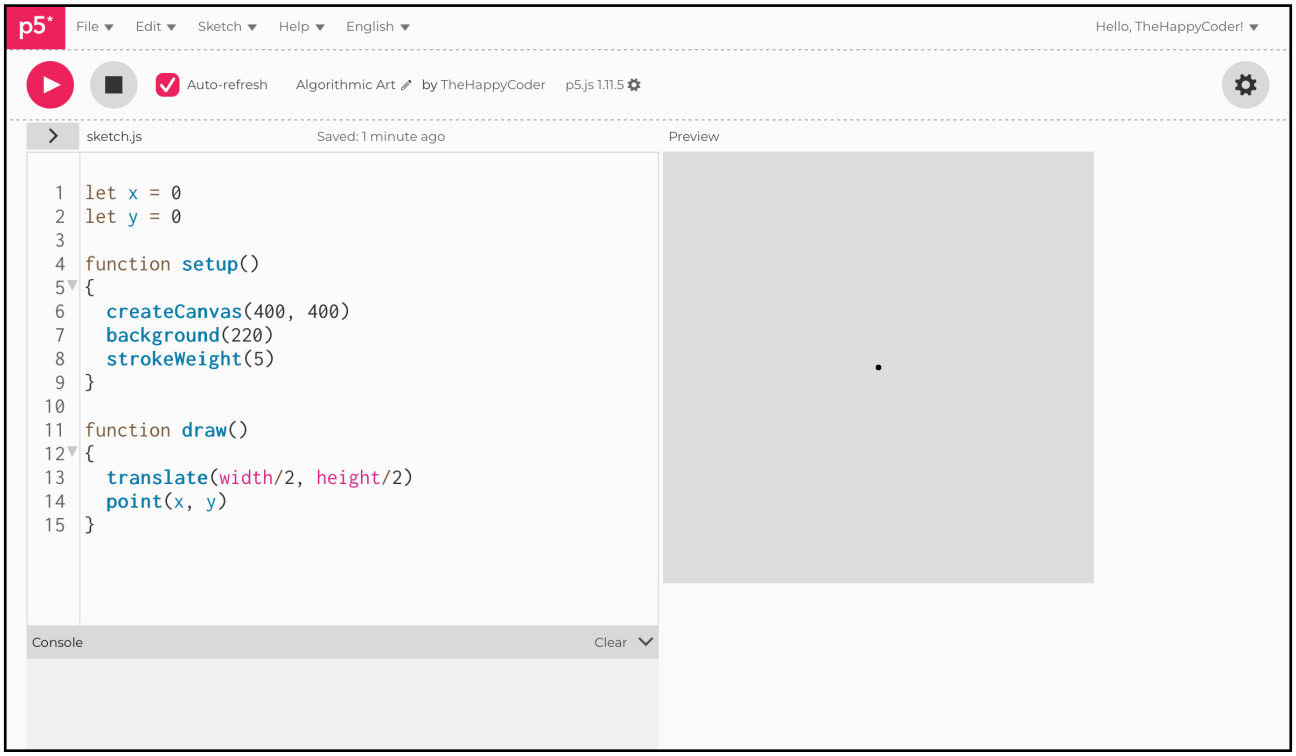
We will `translate` it into the centre using the `x`, `y` variables.

```
let x = 0
let y = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
  strokeWeight(5)
}

function draw()
{
  translate(width/2, height/2)
  point(x, y)
}
```

Notes

Exactly as before.

Figure B10.3


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 1 minute ago
Preview
let x = 0
let y = 0
function setup()
{
createCanvas(400, 400)
background(220)
strokeWeight(5)
}
function draw()
{
translate(width/2, height/2)
point(x, y)
}
Console
Clear

Sketch B10.4 background

❗ comment out `background()` in `setup()`

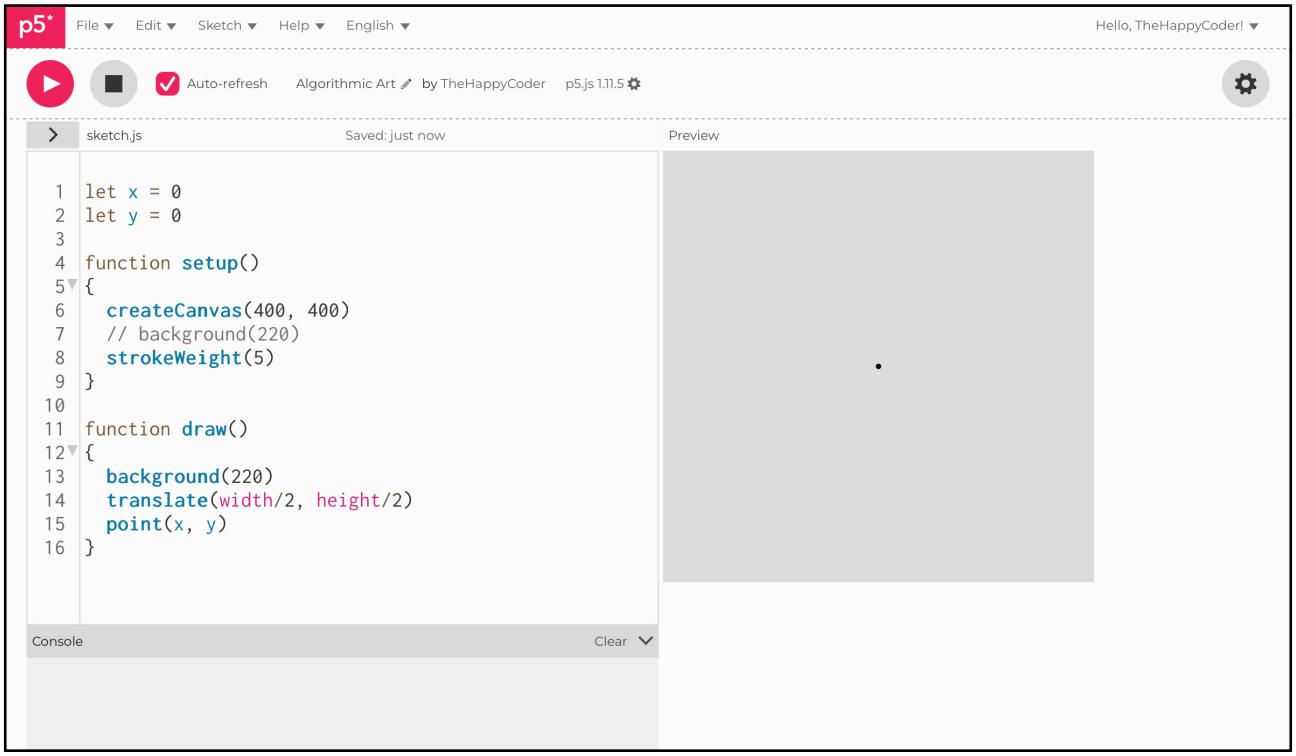
We are going to put the background in the `draw()` function so that it draws the point moving.

```
let x = 0
let y = 0

function setup()
{
  createCanvas(400, 400)
  // background(220)
  strokeWeight(5)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  point(x, y)
}
```

🗒️ Notes

One small step at a time, nothing you haven't already seen.

Figure B10.4


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
function setup()
{
createCanvas(400, 400)
// background(220)
strokeWeight(5)
}
function draw()
{
background(220)
translate(width/2, height/2)
point(x, y)
}
Console
Clear

Sketch B10.5 angle in radians

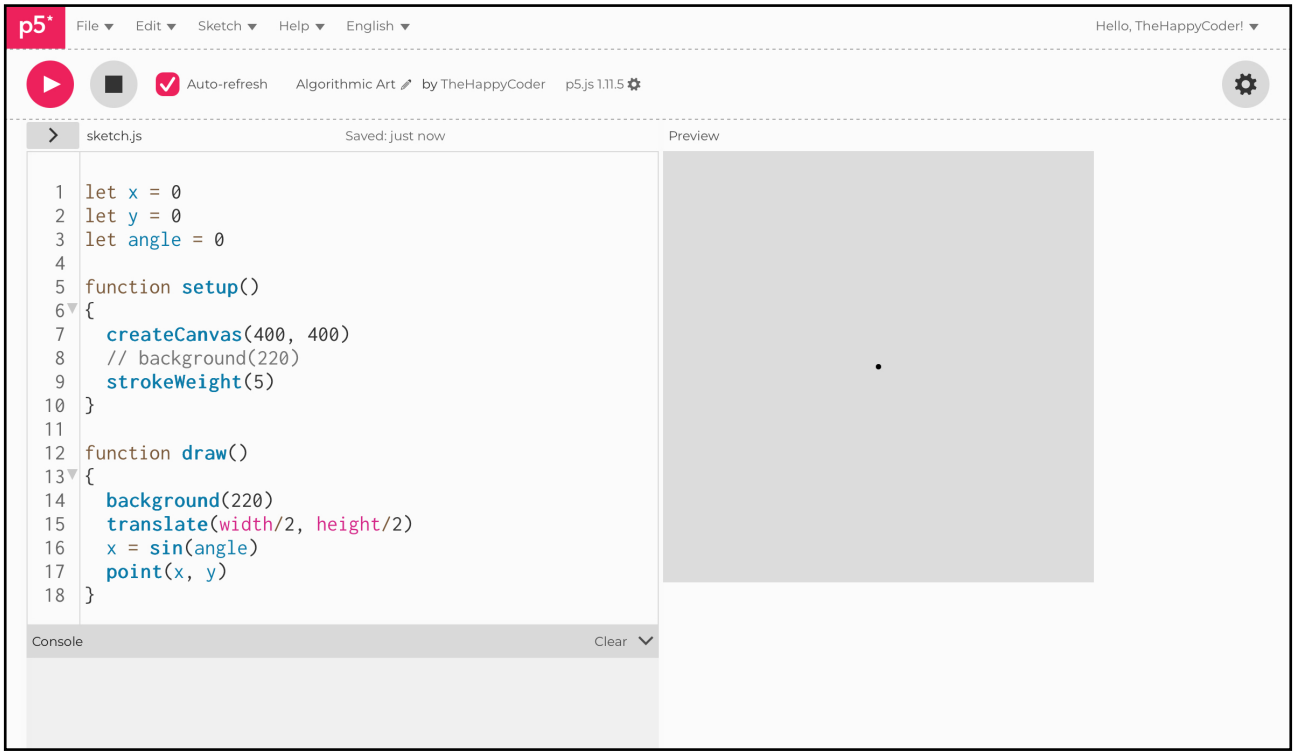
We want to oscillate the point about the centre using a sine wave motion. We first need to create a variable called angle. That angle is in radians, and the sine of those radians returns values between −1 and +1.

```
let x = 0
let y = 0
let angle = 0

function setup()
{
  createCanvas(400, 400)
  // background(220)
  strokeWeight(5)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  x = sin(angle)
  point(x, y)
}
```

Notes

A lot of coding just to draw a dot on the canvas.

Figure B10.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let angle = 0
function setup()
{
createCanvas(400, 400)
// background(220)
strokeWeight(5)
}
function draw()
{
background(220)
translate(width/2, height/2)
x = sin(angle)
point(x, y)
}
Console
Clear

Sketch B10.6 incremental angle

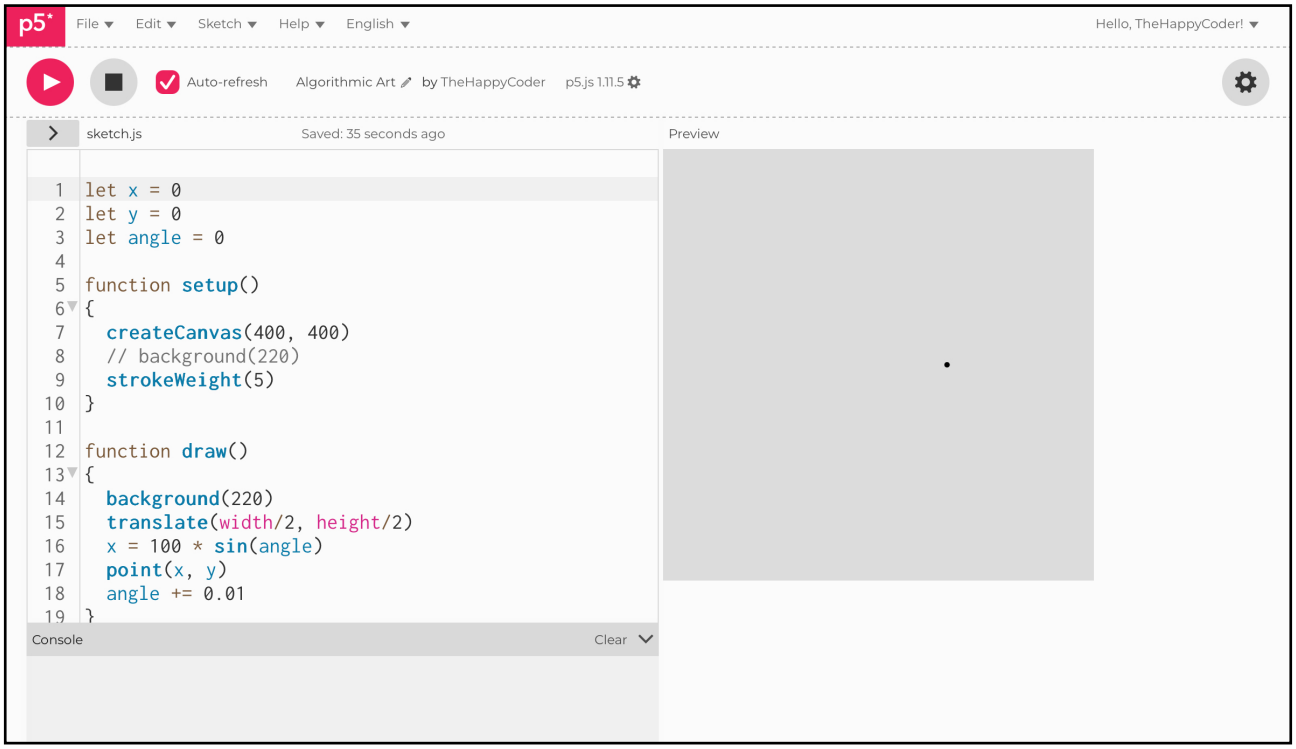
What we need to do is move it by increasing the angle in every loop of draw. We will do it by a very small amount each time. To add a number to a variable, we can use a shortcut method using `+=`. In this case, we are going to be adding `0.01` to the value of the angle each iteration of the loop. We start with `0`, and then it keeps on increasing. The sine of the angle always stays between `-1` and `+1`.

```
let x = 0
let y = 0
let angle = 0

function setup()
{
  createCanvas(400, 400)
  // background(220)
  strokeWeight(5)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  x = 100 * sin(angle)
  point(x, y)
  angle += 0.01
}
```

Notes

We multiply by `100` (the `*` symbol means multiply) because otherwise the point will only move one pixel! What you should see is it oscillating in a line from around `100` to `300` halfway down the canvas.

Figure B10.6


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 35 seconds ago
Preview
let x = 0
let y = 0
let angle = 0
function setup()
{
createCanvas(400, 400)
// background(220)
strokeWeight(5)
}
function draw()
{
background(220)
translate(width/2, height/2)
x = 100 * sin(angle)
point(x, y)
angle += 0.01
}
Console
Clear

Sketch B10.7 circular motion

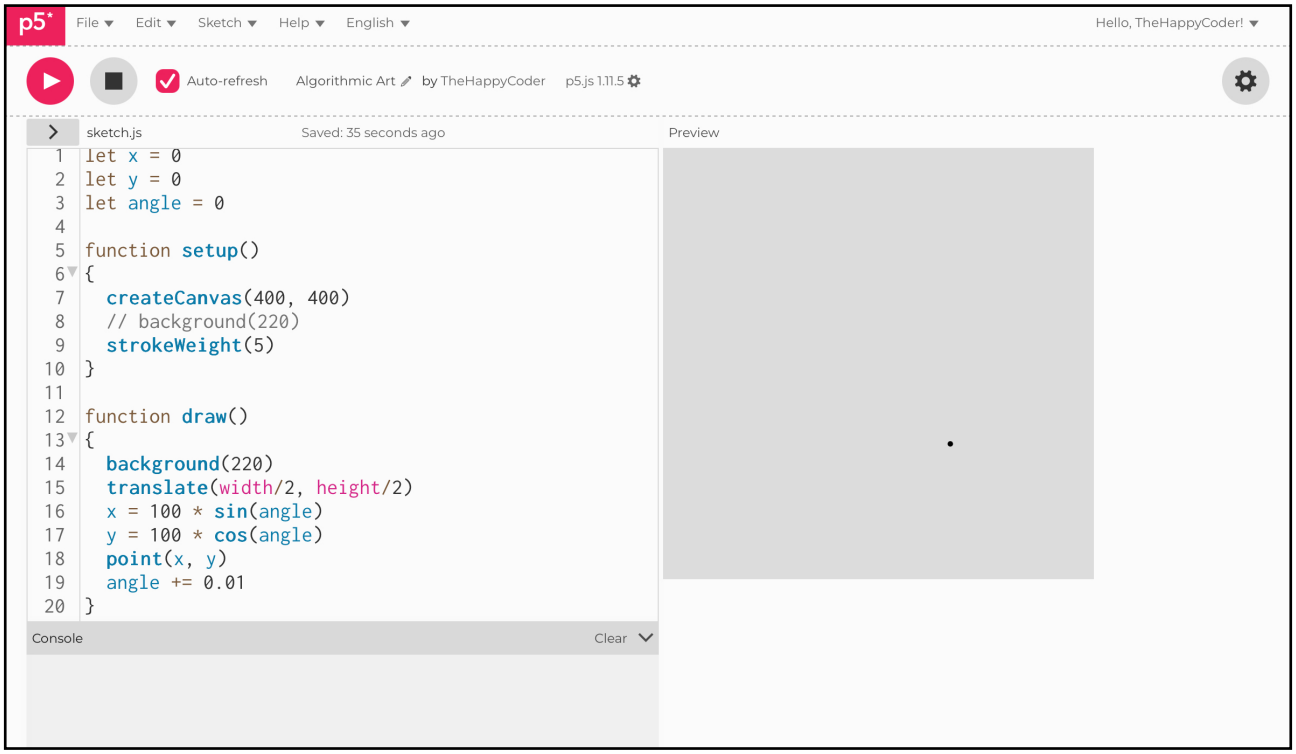
Next, we try to get it to move in a circle.

```
let x = 0
let y = 0
let angle = 0

function setup()
{
  createCanvas(400, 400)
  // background(220)
  strokeWeight(5)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  x = 100 * sin(angle)
  y = 100 * cos(angle)
  point(x, y)
  angle += 0.01
}
```

Notes

You should see the dot moving in a circle with a radius of 100.

Figure B10.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 35 seconds ago
Preview
let x = 0
let y = 0
let angle = 0
function setup()
{
createCanvas(400, 400)
// background(220)
strokeWeight(5)
}
function draw()
{
background(220)
translate(width/2, height/2)
x = 100 * sin(angle)
y = 100 * cos(angle)
point(x, y)
angle += 0.01
}
Console
Clear

Sketch B10.8 drawing a circle

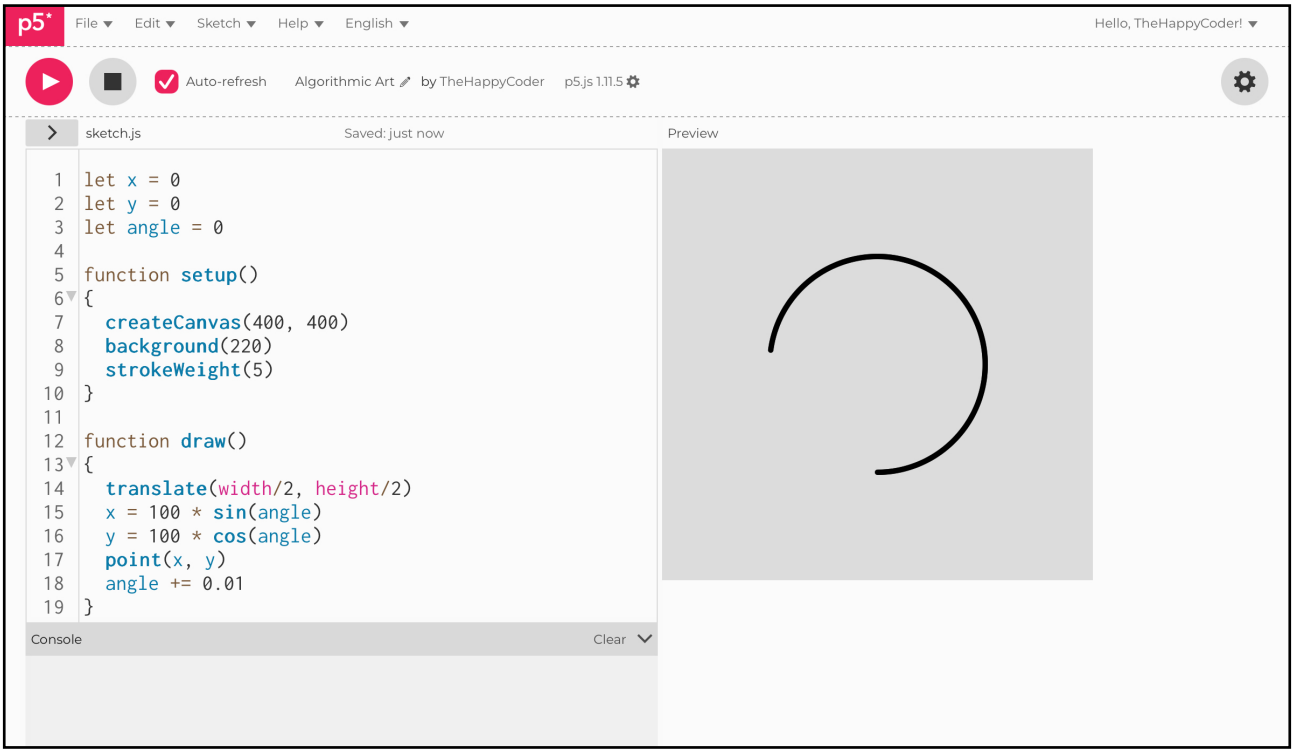
❗ Remove the `background()` function in `draw()` and reinstate it in `setup()` as before, and we will see the dot scribe a permanent circle on the canvas.

```
let x = 0
let y = 0
let angle = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
  strokeWeight(5)
}

function draw()
{
  translate(width/2, height/2)
  x = 100 * sin(angle)
  y = 100 * cos(angle)
  point(x, y)
  angle += 0.01
}
```

🗒️ Notes

You will get a circle drawn on the canvas and it will keep on drawing.

Figure B10.8


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let x = 0
let y = 0
let angle = 0
function setup()
{
createCanvas(400, 400)
background(220)
strokeWeight(5)
}
function draw()
{
translate(width/2, height/2)
x = 100 * sin(angle)
y = 100 * cos(angle)
point(x, y)
angle += 0.01
}
Console
Clear

Sketch B10.9 vertex circle

Going to make quite a few changes.

- The variable `r` for the radius
- The `for()` loop for the angle
- The `vertex()` instead of a `point()`
- The need to `beginShape()` and `endShape()`

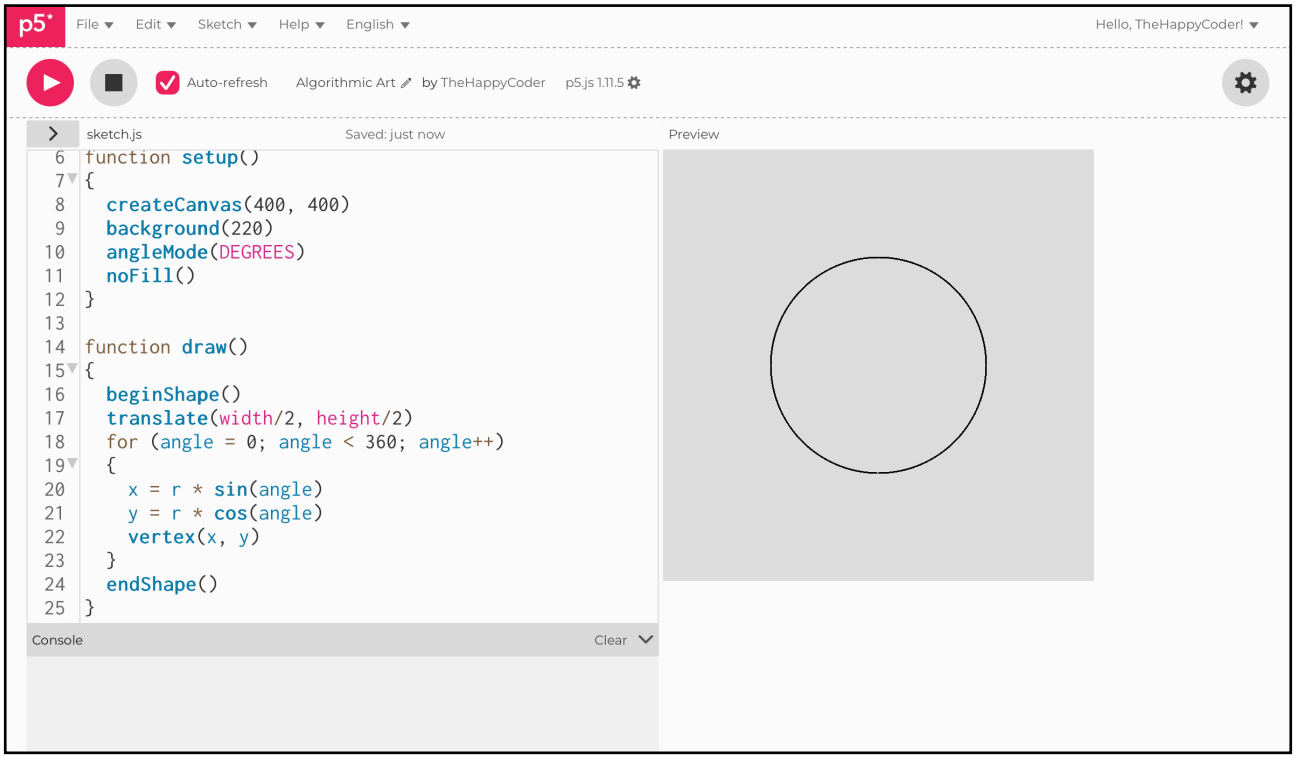
```
let x = 0
let y = 0
let r = 100
let angle = 0

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

function draw()
{
  beginShape()
  translate(width/2, height/2)
  for (angle = 0; angle < 360; angle++)
  {
    x = r * sin(angle)
    y = r * cos(angle)
    vertex(x, y)
  }
  endShape()
```

```
}
```

Notes

A few changes, you might say, all fairly obvious.

Figure B10.9


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
6 function setup()
7 {
8 createCanvas(400, 400)
9 background(220)
10 angleMode(DEGREES)
11 noFill()
12 }
13
14 function draw()
15 {
16 beginShape()
17 translate(width/2, height/2)
18 for (angle = 0; angle < 360; angle++)
19 {
20 x = r * sin(angle)
21 y = r * cos(angle)
22 vertex(x, y)
23 }
24 endShape()
25 }
Console
Clear

Sketch B10.10 random loop

Let us try something a bit different, making the radius random. We need a `noLoop()` at the end, though.

```
let x = 0
let y = 0
let r = 100
let angle = 0

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

function draw()
{
  beginShape()
  translate(width/2, height/2)
  for (angle = 0; angle < 360; angle++)
  {
    x = random(r) * sin(angle)
    y = r * cos(angle)
    vertex(x, y)
  }
  endShape()
  noLoop()
}
```

🗒️ Notes

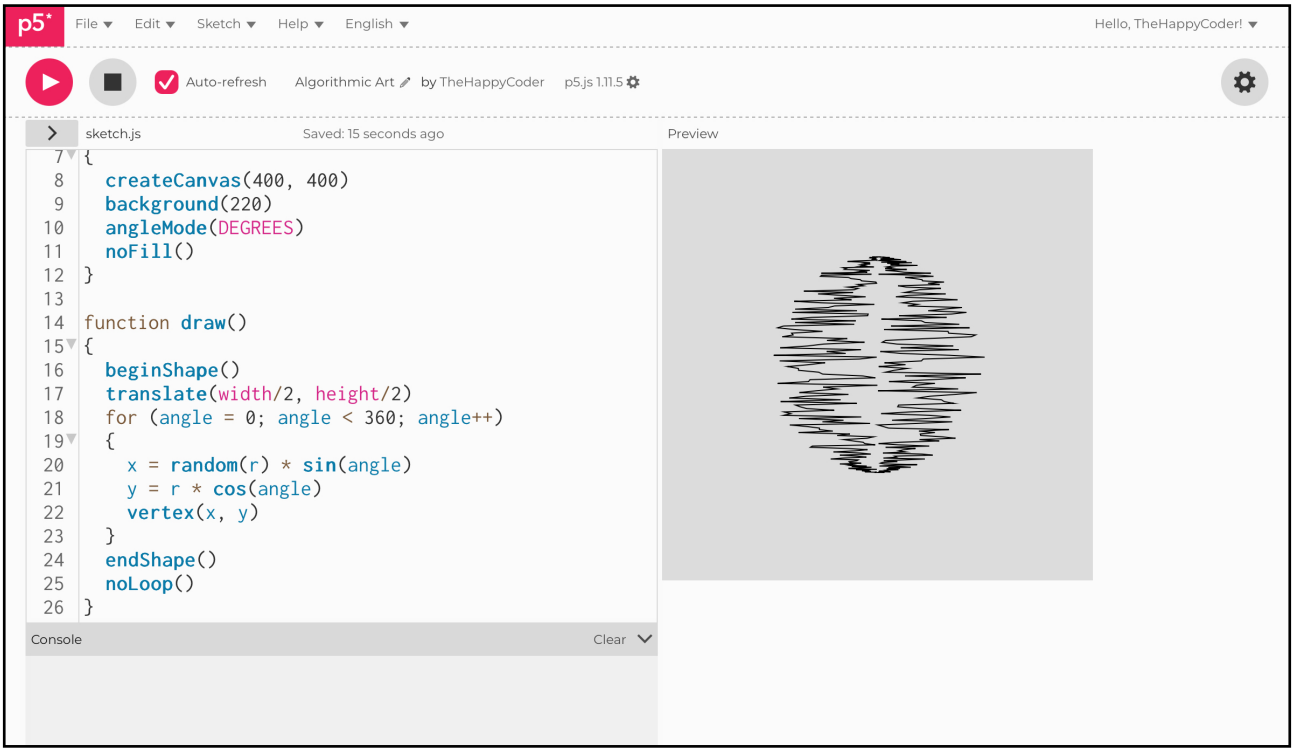
We have only changed one variable to be random.

🌻 Challenges

1. Change the `y` to a random value as well.
2. Change the random value for `x` to be between `50` and `100`.
3. Try the following:

```
x = random(-r, r) * sin(angle)
y = r*cos(angle)
```

Figure B10.10



Sketch B10.11 spiral

Take it back to a circle (remove the random). We can alter the radius so that it starts at zero and increases incrementally. We also need to decide how many times we want to go round the circle; here I have multiplied by 20 times.

```
let x = 0
let y = 0
let r = 0
let angle = 0

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

function draw()
{
  beginShape()
  translate(width/2, height/2)
  for (angle = 0; angle < 360 * 20; angle++)
  {
    x = r * sin(angle)
    y = r * cos(angle)
    vertex(x, y)
    r += 0.02
  }
  endShape()
  noLoop()
```

```
}
```

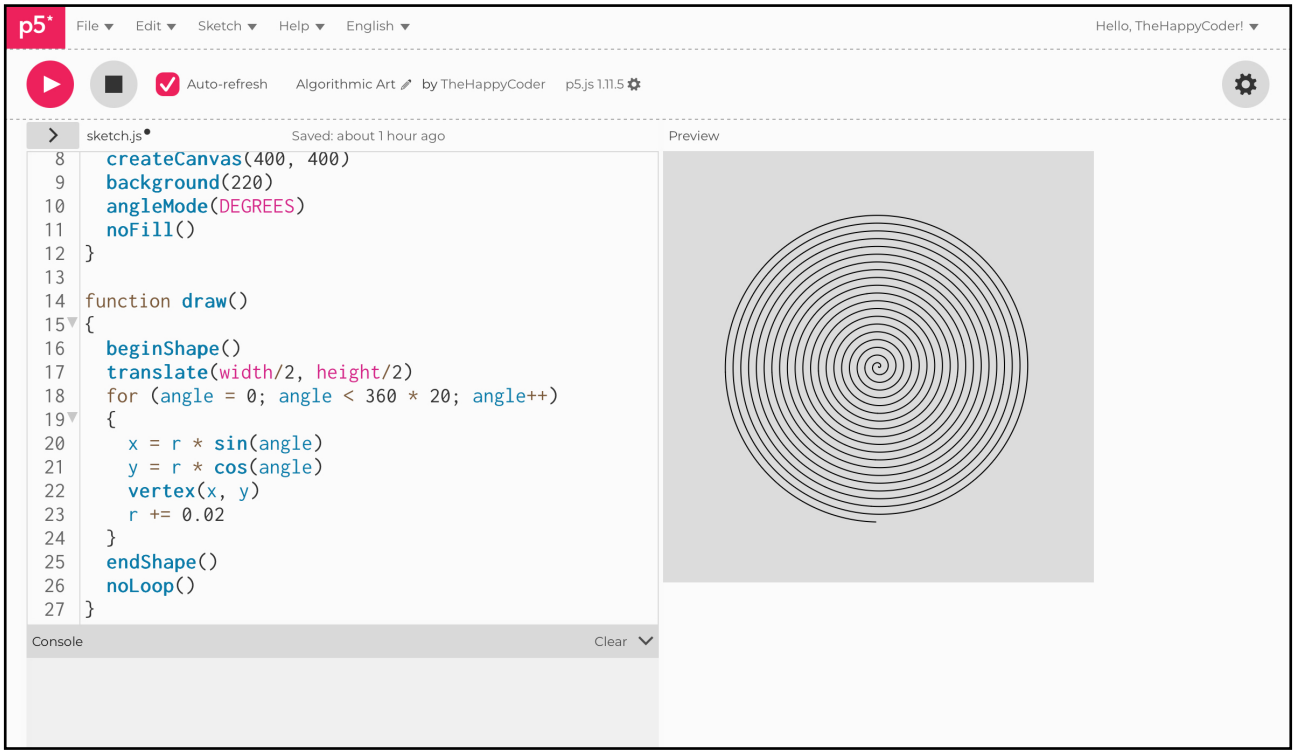
🗒️ Notes

A very simple spiral.

🌻 Challenges

1. Change the number of revolutions.
2. Change the increase in radius.
3. Add some randomness.

Figure B10.11



The phyllotaxis

To replicate the patterns on a flower (such as a sunflower or something similar), we need to follow the maths. First, we identify the variables:

- `index`: each floret (circle) has an `index` (0, 1, 2, 3, etc.)
- `constant`: a scaling factor, something like `8`
- `radius`: is the `constant` times the square root of the `index`
- `angle`: is the `index` times the golden ratio (`137.508°`)

The angle `137.508°` is the golden angle, which is approximated by the ratios of Fibonacci numbers.

Sketch B10.12 constant variables

❗ starting a brand new sketch

Let's keep it simple and build from there. First, we need the variables and constants. We will give them values of `zero` till we know what to do with them.

```
let constant = 0
let radius = 0
let index = 0
let x = 0
let y = 0
let angle = 0

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

function draw()
{

}
```

Notes

This does nothing, and we have an empty `draw()` function for the moment.

Sketch B10.13 draw a circle

Now create a loop to draw the circles.

```
let constant = 0
let radius = 0
let index = 0
let x = 0
let y = 0
let angle = 0

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

function draw()
{
  translate(width/2, height/2)
  for (let i = 0; i < index; i++)
  {
    x = radius * cos(angle)
    y = radius * sin(angle)
    circle(x, y, 10)
  }
}
```

Notes

Nothing to see because everything is zero.

Sketch B10.14 angleMode

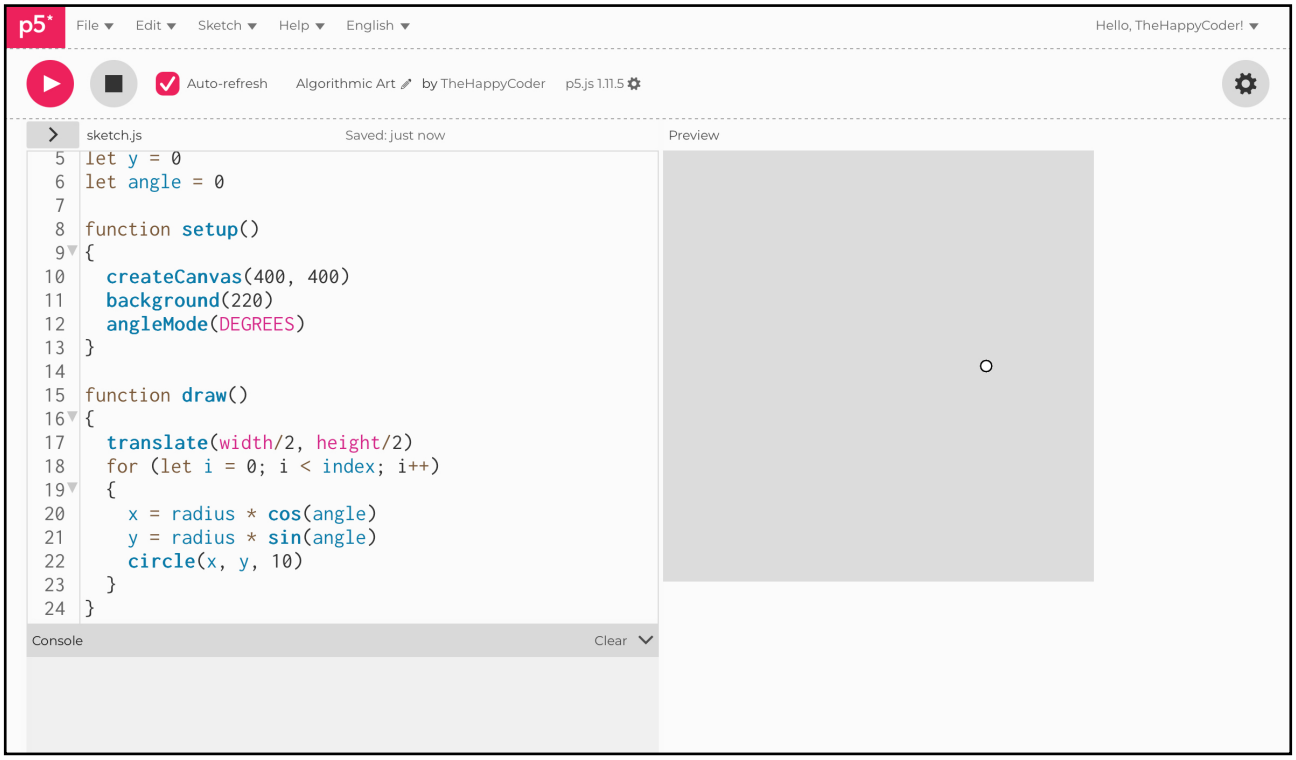
Let's put some values on this so we can see something. Also, we need to work in degrees, so we will add `angleMode()` and so the `index` needs to be `360`. The radius will be `100`.

```
let constant = 0
let radius = 100
let index = 360
let x = 0
let y = 0
let angle = 0

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

function draw()
{
  translate(width/2, height/2)
  for (let i = 0; i < index; i++)
  {
    x = radius * cos(angle)
    y = radius * sin(angle)
    circle(x, y, 10)
  }
}
```

Notes

At least something has happened, just not a lot!

Figure B10.14



Sketch B10.15 using the golden angle

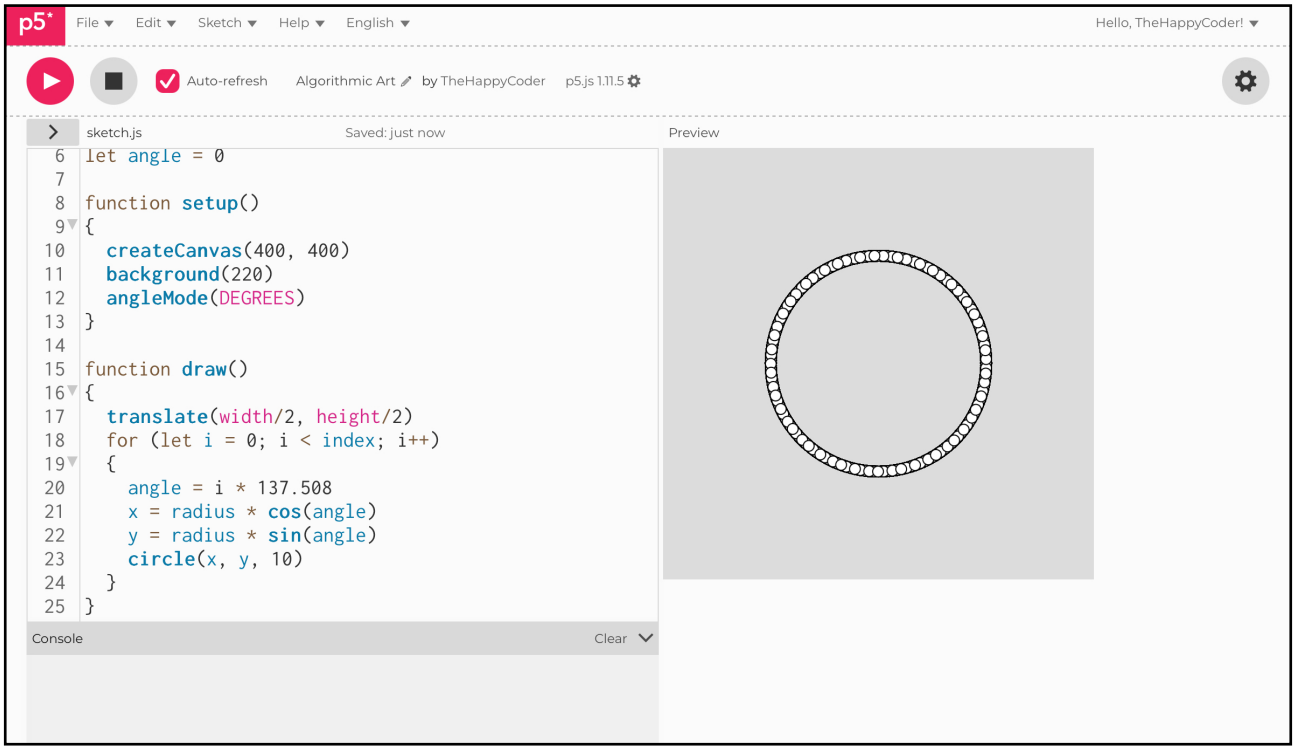
We now need to introduce an angle that changes and add the equation where the angle is the index times 137.508.

```
let constant = 0
let radius = 100
let index = 360
let x = 0
let y = 0
let angle = 0

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

function draw()
{
  translate(width/2, height/2)
  for (let i = 0; i < index; i++)
  {
    angle = i * 137.508
    x = radius * cos(angle)
    y = radius * sin(angle)
    circle(x, y, 10)
  }
}
```

🗒️ Notes

Because the `angle` increments are `1°` times `137.508`, it goes round several times, which is what we want; however, the `radius` is constant, which we don't want.

Figure B10.15



Sketch B10.16 the final piece of the puzzle

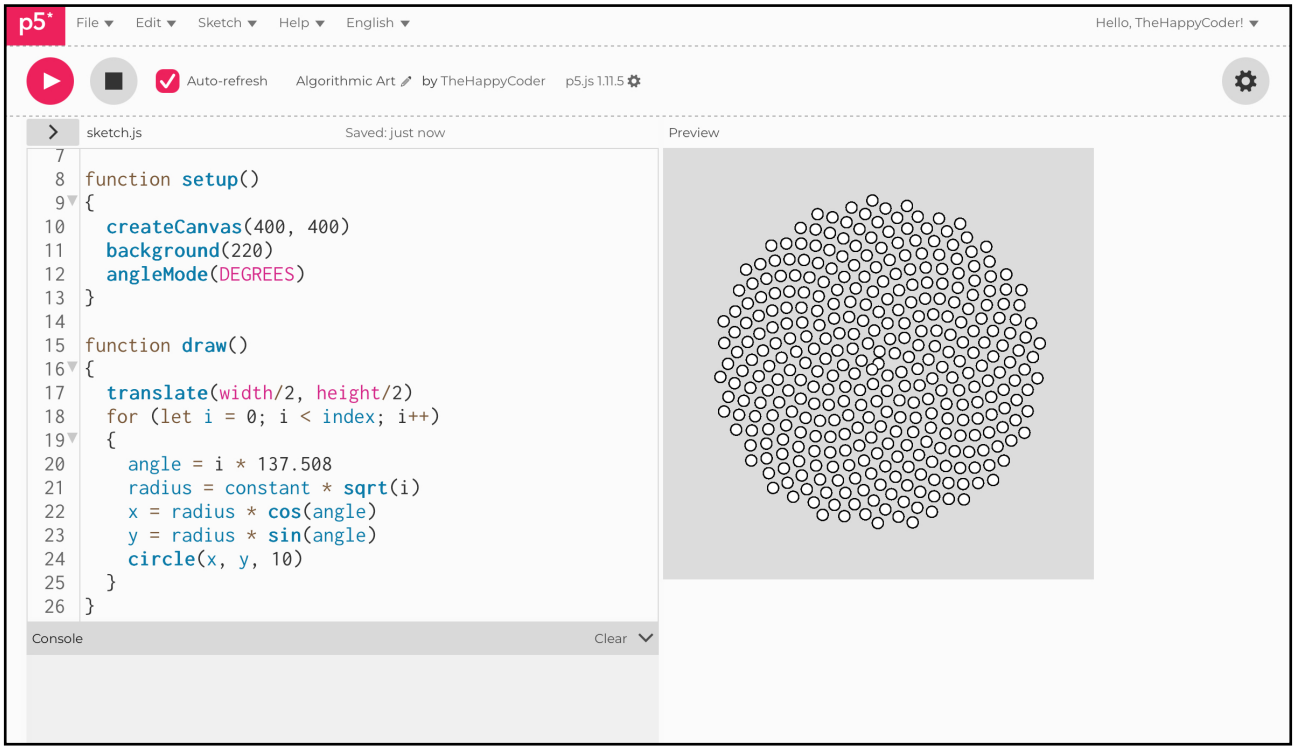
The radius is the constant times the square root of the `index`; in this case, it is `i`. We can use a function called `sqrt()` for this. But we need to give the `constant` a value; I will pick `8` because I have already played with some values; you can try others.

```
let constant = 8
let radius = 100
let index = 360
let x = 0
let y = 0
let angle = 0

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

function draw()
{
  translate(width/2, height/2)
  for (let i = 0; i < index; i++)
  {
    angle = i * 137.508
    radius = constant * sqrt(i)
    x = radius * cos(angle)
    y = radius * sin(angle)
    circle(x, y, 10)
  }
}
```

Notes

Wow, we get a rather nice pattern.

Figure B10.16



Sketch B10.17 greyness

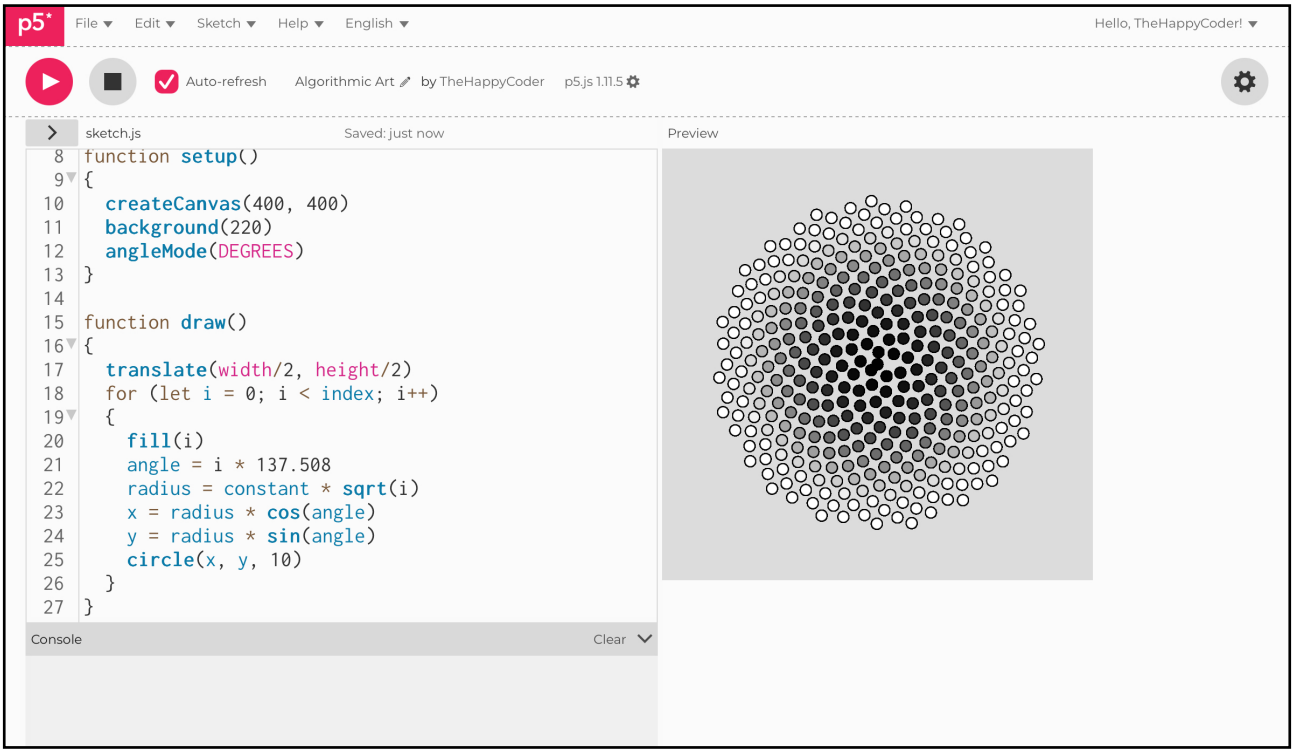
So we can now give it some colour of sorts. Let us start really simply with grey before we move on to other more dynamic colours.

```
let constant = 8
let radius = 100
let index = 360
let x = 0
let y = 0
let angle = 0

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

function draw()
{
  translate(width/2, height/2)
  for (let i = 0; i < index; i++)
  {
    fill(i)
    angle = i * 137.508
    radius = constant * sqrt(i)
    x = radius * cos(angle)
    y = radius * sin(angle)
    circle(x, y, 10)
  }
}
```

🗒️ Notes

We have something, but we can do even better.

Figure B10.17



Sketch B10.18 mapping the colour

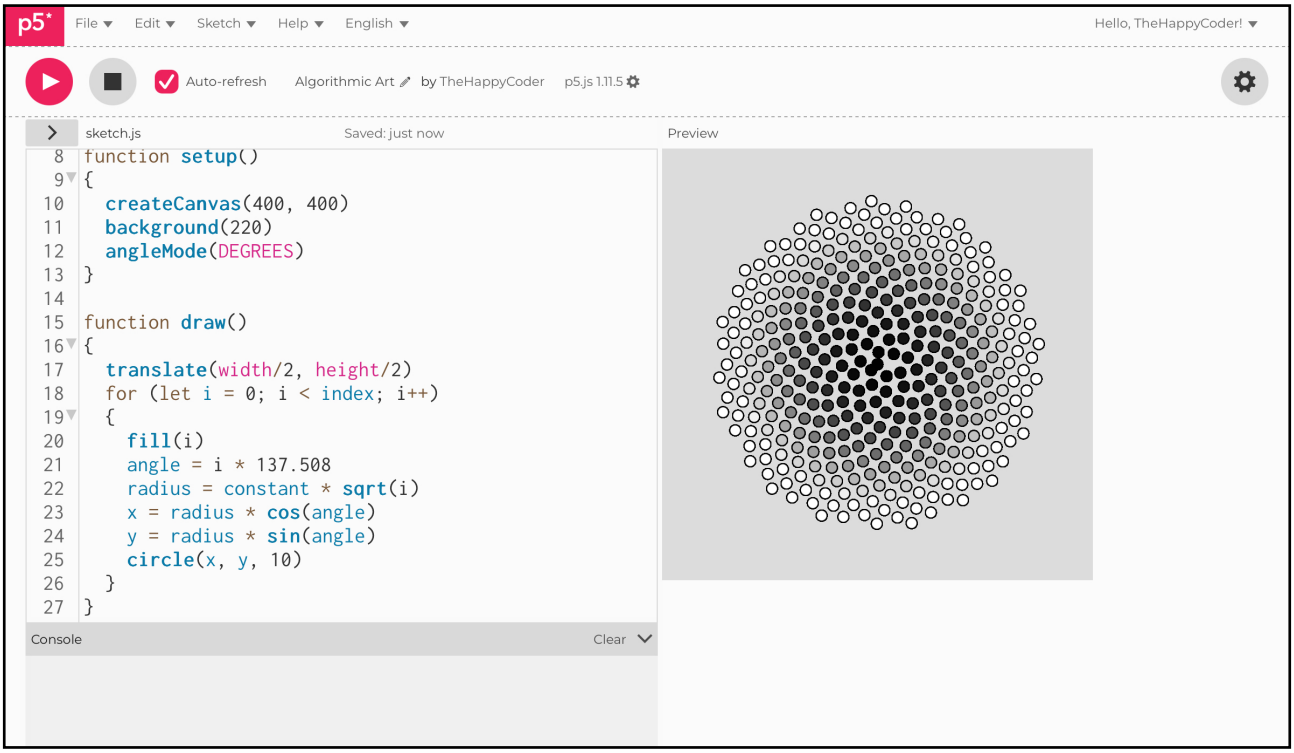
Mapping the colour to the index value.

```
let constant = 8
let radius = 100
let index = 360
let x = 0
let y = 0
let angle = 0

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

function draw()
{
  translate(width/2, height/2)
  for (let i = 0; i < index; i++)
  {
    let colour = map(i, 0, index, 0, 255)
    fill(colour)
    angle = i * 137.508
    radius = constant * sqrt(i)
    x = radius * cos(angle)
    y = radius * sin(angle)
    circle(x, y, 10)
  }
}
```

Notes

Showing promise.

Figure B10.18



Sketch B10.19 colour HSB

Let us make it colour with HSB rather than RGB. RGB is the default mode, but HSB gives you different options.

```
let constant = 8
let radius = 100
let index = 360
let x = 0
let y = 0
let angle = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
  angleMode(DEGREES)
  colorMode(HSB)
}

function draw()
{
  translate(width/2, height/2)
  for (let i = 0; i < index; i++)
  {
    let colour = map(i, 0, index, 0, 255)
    fill(colour, 100, 100)
    angle = i * 137.508
    radius = constant * sqrt(i)
    x = radius * cos(angle)
    y = radius * sin(angle)
    circle(x, y, 10)
  }
```

```
}
```

🗒️ Notes

Now that is so much nicer.

🌻 Challenges

1. See what you can produce.
2. Play with the values.

Figure B10.19