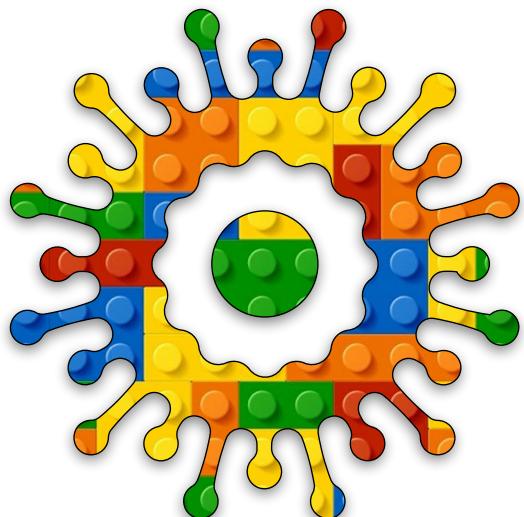


Creative Coding

Module B

Unit #10

phyllotaxis





Module B Unit #10 Phyllotaxis

- Sketch B10.1 start with a point
- Sketch B10.2 x and y variables
- Sketch B10.3 translate
- Sketch B10.4 background
- Sketch B10.5 angle in radians
- Sketch B10.6 incremental angle
- Sketch B10.7 circular motion
- Sketch B10.8 drawing a circle
- Sketch B10.9 vertex circle
- Sketch B10.10 random loop
- Sketch B10.11 spiral
- Sketch B10.12 constant variables
- Sketch B10.13 draw a circle
- Sketch B10.14 angleMode
- Sketch B10.15 the golden angle
- Sketch B10.16 the final piece of the puzzle
- Sketch B10.17 greyness
- Sketch B10.18 mapping the colour
- Sketch B10.19 colour HSB



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

The screenshot shows the p5.js web editor interface. At the top, there's a navigation bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. On the right side of the bar, it says 'Hello, TheHappyCoder! ▾'. Below the bar are three buttons: a play button, a square button, and an auto-refresh button with a checkmark. To the right of these buttons is the text 'Algorithmic Art by TheHappyCoder p5.js 1.11.5'. On the far right is a gear icon.

The main area has tabs for 'sketch.js' (which is selected), 'Saved: just now', and 'Preview'. The code editor contains the following JavaScript code:

```
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 }
```

To the right of the code editor is a preview window showing a light gray canvas. At the bottom left is a 'Console' tab and a 'Clear ▾' button. The bottom right corner of the preview window has a small black dot.



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

The screenshot shows the p5.js web editor interface. At the top, there's a navigation bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. On the right side of the bar, it says 'Hello, TheHappyCoder! ▾' and has a gear icon. Below the bar, there are icons for play, stop, and refresh, followed by 'Auto-refresh' with a checked checkbox. To the right of these are 'Algorithmic Art' and 'p5.js 1.11.5' with a gear icon.

The main area is divided into two sections: 'sketch.js' on the left and 'Preview' on the right. In 'sketch.js', the code is:

```
let x = 200
let y = 200

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

}

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

In the 'Preview' section, a single black dot is visible at the coordinates (200, 200) on a light gray background.

At the bottom left is a 'Console' tab, and at the bottom center is a 'Clear ▾' button.



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

The screenshot shows the p5.js web editor interface. At the top, there's a navigation bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. On the right side of the bar, it says 'Hello, TheHappyCoder! ▾' and has a gear icon. Below the bar, there are three buttons: a play button, a stop button, and an 'Auto-refresh' button with a checkmark. The main area is titled 'sketch.js' and shows the following code:

```
1 let x = 0
2 let y = 0
3
4 function setup()
5 {
6   createCanvas(400, 400)
7   background(220)
8   strokeWeight(5)
9 }
10
11 function draw()
12 {
13   translate(width/2, height/2)
14   point(x, y)
15 }
```

To the right of the code is a 'Preview' window showing a light gray canvas with a single black dot at its center. At the bottom left is a 'Console' window with a 'Clear' dropdown menu.



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

The screenshot shows the p5.js code editor interface. At the top, there are menu options: File ▾, Edit ▾, Sketch ▾, Help ▾, and English ▾. On the right side, it says "Hello, TheHappyCoder! ▾". Below the menu is a toolbar with icons for play, stop, and refresh, followed by "Auto-refresh" checked, "Algorithmic Art" by TheHappyCoder, and p5.js 1.11.5. To the right of the toolbar is a gear icon.

The main area has tabs for "sketch.js" and "Preview". The "sketch.js" tab contains the following code:

```
1 let x = 0
2 let y = 0
3
4 function setup()
5 {
6   createCanvas(400, 400)
7   // background(220)
8   strokeWeight(5)
9 }
10
11 function draw()
12 {
13   background(220)
14   translate(width/2, height/2)
15   point(x, y)
16 }
```

The "Preview" window shows a light gray canvas with a single black dot at its center, indicating the output of the current sketch.



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

The screenshot shows the p5.js code editor interface. At the top, there's a menu bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. To the right of the menu is a greeting 'Hello, TheHappyCoder! ▾' and a gear icon for settings. Below the menu, there are three buttons: a play button, a square button, and an auto-refresh button with a checkmark. The title bar says 'sketch.js' and 'Saved: just now'. The preview window on the right is currently empty.

```
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)
}
```

At the bottom left is a 'Console' tab and a 'Clear' dropdown menu.



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

The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, auto-refresh (which is checked), and other options like 'Algorithmic Art' and 'Help'. The title bar says 'Hello, TheHappyCoder! ▾' and the file name is 'sketch.js'. Below the toolbar, the code editor has two main sections: 'sketch.js' on the left containing the code, and 'Preview' on the right showing the resulting output.

```
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
}
```

The preview window shows a single black dot at the center of a 400x400 canvas, indicating that the code has not yet run or is in its initial state.



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

The screenshot shows the p5.js code editor interface. At the top, there are navigation menus: File, Edit, Sketch, Help, and English. On the right, it says "Hello, TheHappyCoder!". Below the menu bar, there are buttons for play/pause, stop, auto-refresh (which is checked), and settings. The file name is "sketch.js" and it was saved 35 seconds ago. The code editor contains the following JavaScript code:

```
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
}
```

The preview window on the right shows a gray canvas with a single black dot at the center, indicating the current state of the sketch.



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

The screenshot shows the p5.js code editor interface. At the top, there are menu options: File, Edit, Sketch, Help, and English. On the right, it says "Hello, TheHappyCoder! ▾". Below the menu is a toolbar with icons for play, stop, auto-refresh (which is checked), and settings. The file name is "sketch.js" and it was saved "just now". The preview window on the right shows a black sine wave curve on a light gray background. The code editor contains the following JavaScript code:

```
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
}
```

Below the code editor is a "Console" section with a "Clear" button.



Sketch B10.9 vertex circle

Going to make quite a few changes.

- 中 The variable `r` for the radius
- 中 The `for()` loop for the angle
- 中 The `vertex()` instead of a `point()`
- 中 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

The screenshot shows the p5.js code editor interface. At the top, there are menu options: File, Edit, Sketch, Help, and English. On the right, it says "Hello, TheHappyCoder!". Below the menu is a toolbar with icons for play, stop, and refresh, followed by "Auto-refresh" checked, "Algorithmic Art" by TheHappyCoder, and p5.js 1.11.5. To the right of the toolbar is a gear icon.

The code editor window has a title bar "sketch.js" and a status bar "Saved: just now" and "Preview". The code itself is:

```
> sketch.js
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 }
```

The preview window on the right shows a single black outline of a circle centered at the top-left of a 400x400 pixel canvas.



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

The screenshot shows the p5.js code editor interface. At the top, there are menu options: File ▾, Edit ▾, Sketch ▾, Help ▾, English ▾, and a user greeting "Hello, TheHappyCoder! ▾". On the left, there's a toolbar with a play button, a square, and a refresh icon with a checkmark. Below the toolbar, the file name "sketch.js" is listed, along with the message "Saved: 15 seconds ago". The code editor area contains the following JavaScript code:

```
7 > sketch.js
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 = random(r) * sin(angle)
21     y = r * cos(angle)
22     vertex(x, y)
23   }
24   endShape()
25   noLoop()
26 }
```

To the right of the code editor is a preview window showing a circular pattern composed of many short, black, wavy lines radiating from the center, resembling a stylized sun or a complex fractal shape.



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()
```

}

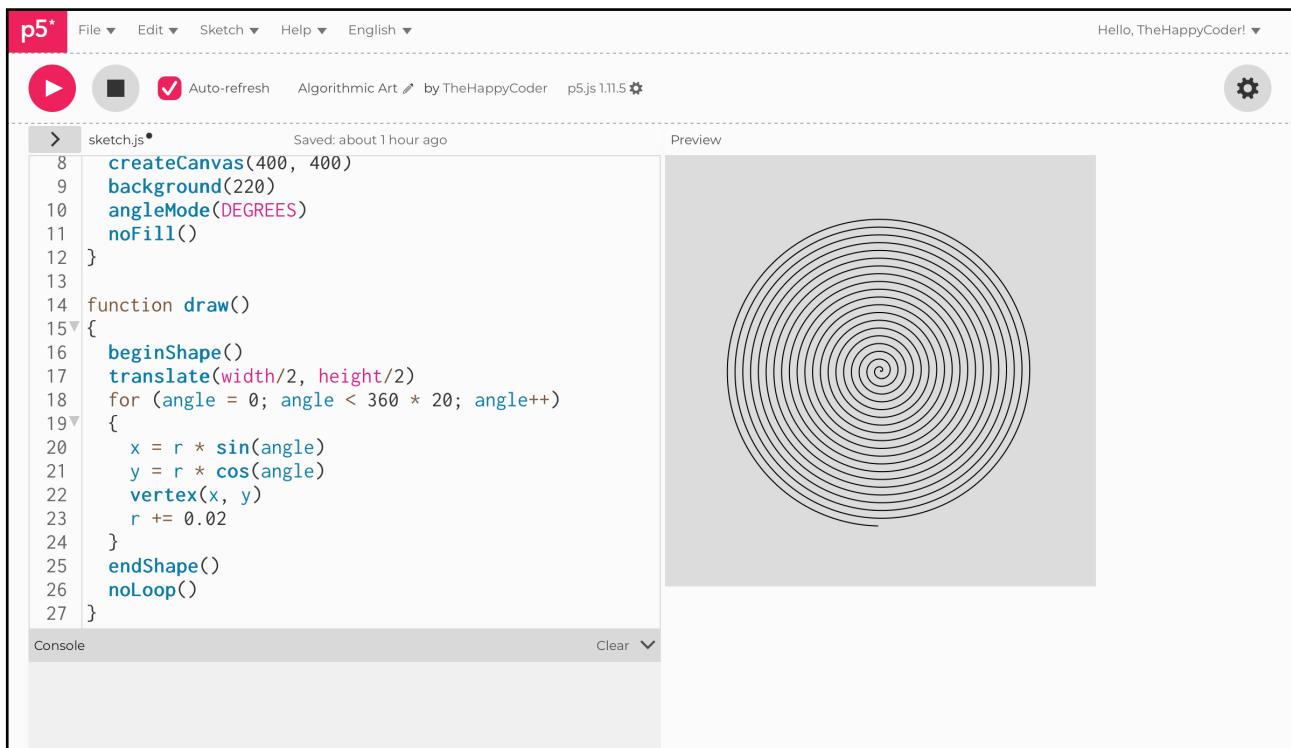
Notes

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 screenshot shows the p5.js code editor interface. At the top, there are menu options: File ▾, Edit ▾, Sketch ▾, Help ▾, English ▾, and a user greeting "Hello, TheHappyCoder! ▾". Below the menu is a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and settings. The main area has tabs for "sketch.js" and "Preview". The code in "sketch.js" is as follows:

```
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 * 20; angle++)
19  {
20    x = r * sin(angle)
21    y = r * cos(angle)
22    vertex(x, y)
23    r += 0.02
24  }
25  endShape()
26  noLoop()
27 }
```

The "Preview" window shows a circular spiral pattern on a light gray background, starting from the center and expanding outwards.



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:

- 中 **index**: each floret (circle) has an **index** (0, 1, 2, 3, etc.)
- 中 **constant**: a scaling factor, something like **8**
- 中 **radius**: is the **constant** times the square root of the **index**
- 中 **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

The screenshot shows the p5.js code editor interface. The top bar includes the p5 logo, file navigation (File, Edit, Sketch, Help, English), a user greeting (Hello, TheHappyCoder!), and a gear icon for settings. The main area has tabs for 'sketch.js' and 'Preview'. The code editor displays the following JavaScript code:

```
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)
  }
}
```

The preview window on the right shows a single small black circle centered at the bottom of the 400x400 canvas.



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

The screenshot shows the p5.js IDE interface. At the top, there's a menu bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. On the right, it says 'Hello, TheHappyCoder! ▾'. Below the menu is a toolbar with a play button, a square, a refresh icon, and a gear icon. The main area has tabs for 'sketch.js' and 'Preview'. The code editor contains the following JavaScript code:

```
sketch.js
Saved: just now
6 let angle = 0
7
8 function setup()
9 {
10   createCanvas(400, 400)
11   background(220)
12   angleMode(DEGREES)
13 }
14
15 function draw()
16 {
17   translate(width/2, height/2)
18   for (let i = 0; i < index; i++)
19   {
20     angle = i * 137.508
21     x = radius * cos(angle)
22     y = radius * sin(angle)
23     circle(x, y, 10)
24   }
25 }
```

The preview window shows a large circle with a series of smaller concentric circles drawn around its perimeter.



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.

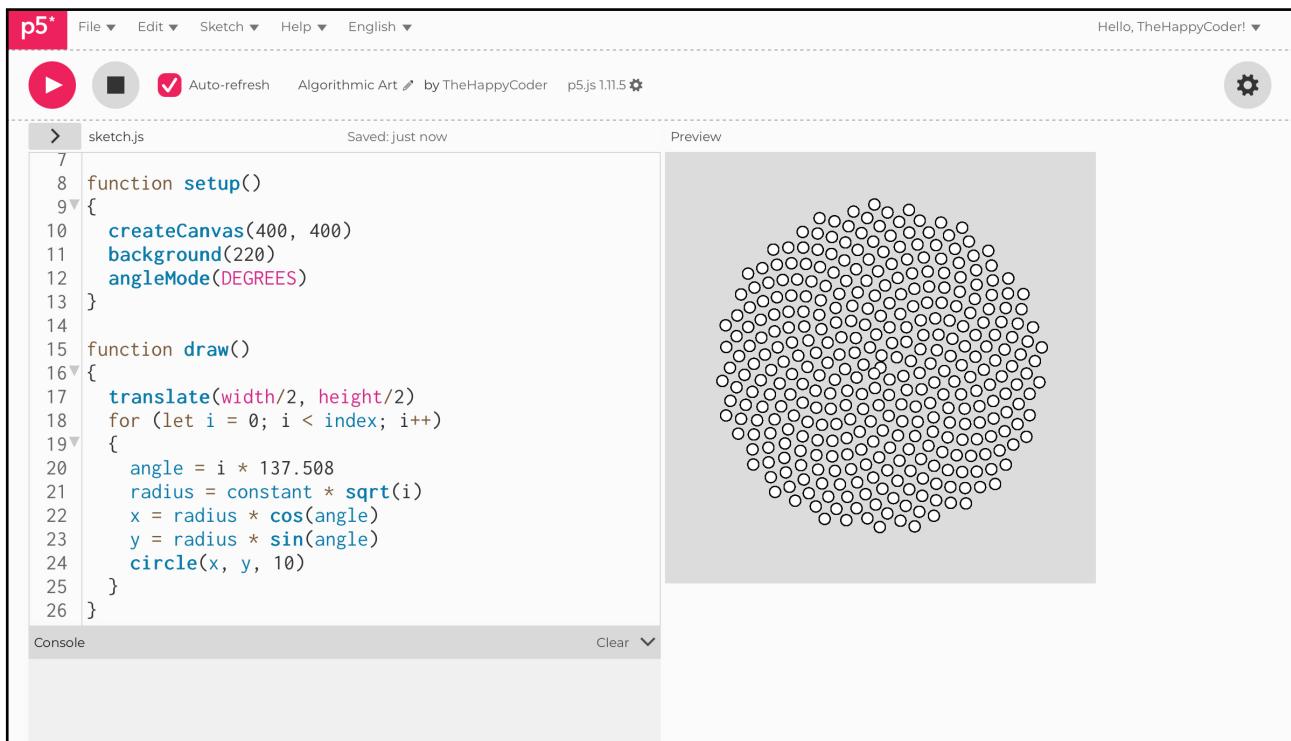
```
let constant = 8  
let radius = 100  
let index = 360  
let x = 0  
let y = 0  
let angle = 0  
  
function setup()  
{  
    createCanvas(400, 400)  
    background(220)  
    angleMode(DEGREES)  
}  
  
function draw()  
{  
    translate(width/2, height/2)  
    for (let i = 0; i < index; i++)  
    {  
        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



The screenshot shows the p5.js code editor interface. At the top, there are menu options: File, Edit, Sketch, Help, and English. On the right, it says "Hello, TheHappyCoder! ▾". Below the menu is a toolbar with icons for play, stop, auto-refresh (which is checked), algorithmic art by TheHappyCoder, and p5.js 1.11.5. The main area has tabs for sketch.js and Preview. The code in sketch.js is as follows:

```
sketch.js
Saved: just now
Preview

7
8 function setup()
9 {
10   createCanvas(400, 400)
11   background(220)
12   angleMode(DEGREES)
13 }
14
15 function draw()
16 {
17   translate(width/2, height/2)
18   for (let i = 0; i < index; i++)
19   {
20     angle = i * 137.508
21     radius = constant * sqrt(i)
22     x = radius * cos(angle)
23     y = radius * sin(angle)
24     circle(x, y, 10)
25   }
26 }
```

The Preview window shows a heart-shaped pattern of small circles, with more circles on the left side and fewer on the right, creating a gradient effect.



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)
}

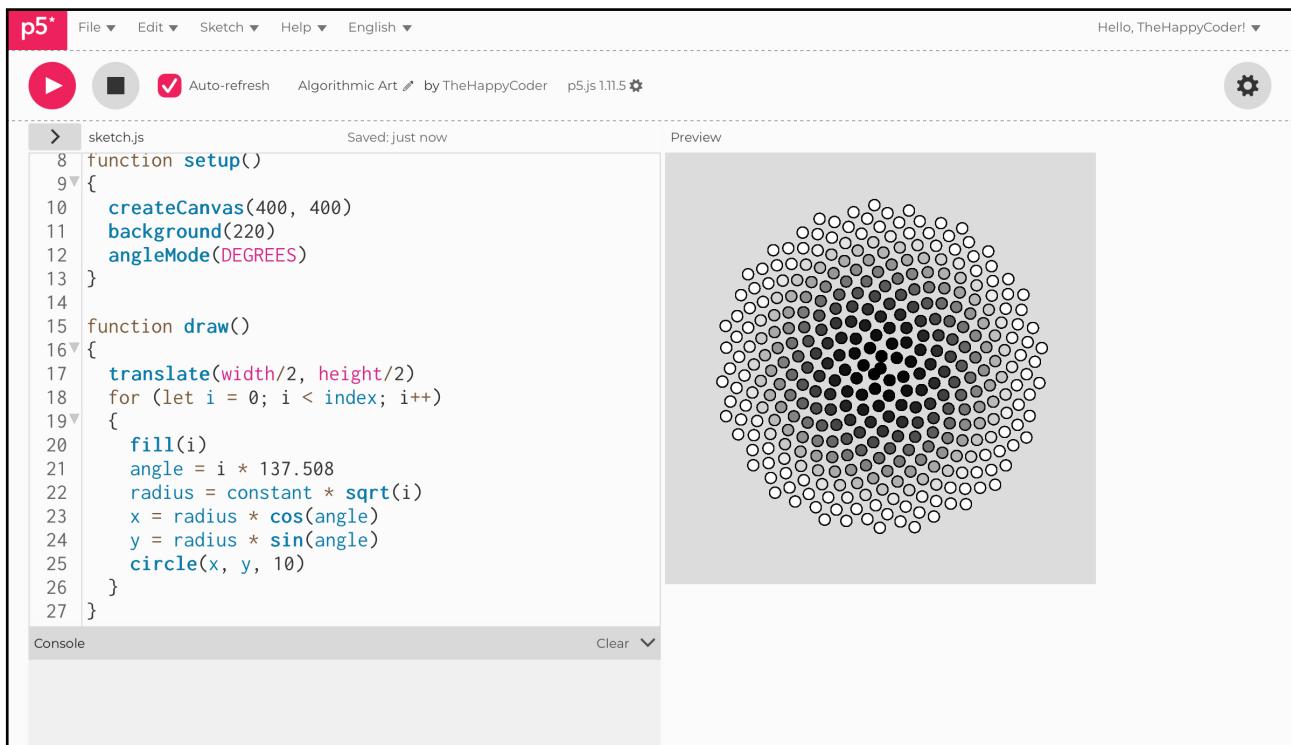
function draw()
{
    translate(width/2, height/2)
    for (let i = 0; i < index; i++)
    {
        fill(i)
        angle = i * 137.508
        radius = constant * sqrt(i)
        x = radius * cos(angle)
        y = radius * sin(angle)
        circle(x, y, 10)
    }
}
```



Notes

We have something, but we can do even better.

Figure B10.17



The screenshot shows the p5.js code editor interface. At the top, there are menu options: File ▾, Edit ▾, Sketch ▾, Help ▾, English ▾, and a user greeting "Hello, TheHappyCoder! ▾". On the left, there's a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and a gear for settings. The main area has tabs for "sketch.js" and "Preview". The code in "sketch.js" is as follows:

```
sketch.js
Saved: just now
8 function setup()
9 {
10   createCanvas(400, 400)
11   background(220)
12   angleMode(DEGREES)
13 }
14
15 function draw()
16 {
17   translate(width/2, height/2)
18   for (let i = 0; i < index; i++)
19   {
20     fill(i)
21     angle = i * 137.508
22     radius = constant * sqrt(i)
23     x = radius * cos(angle)
24     y = radius * sin(angle)
25     circle(x, y, 10)
26   }
27 }
```

The "Preview" window on the right displays a circular pattern of small circles arranged in concentric rings, with the innermost ring being the darkest and the outermost being the lightest.



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)

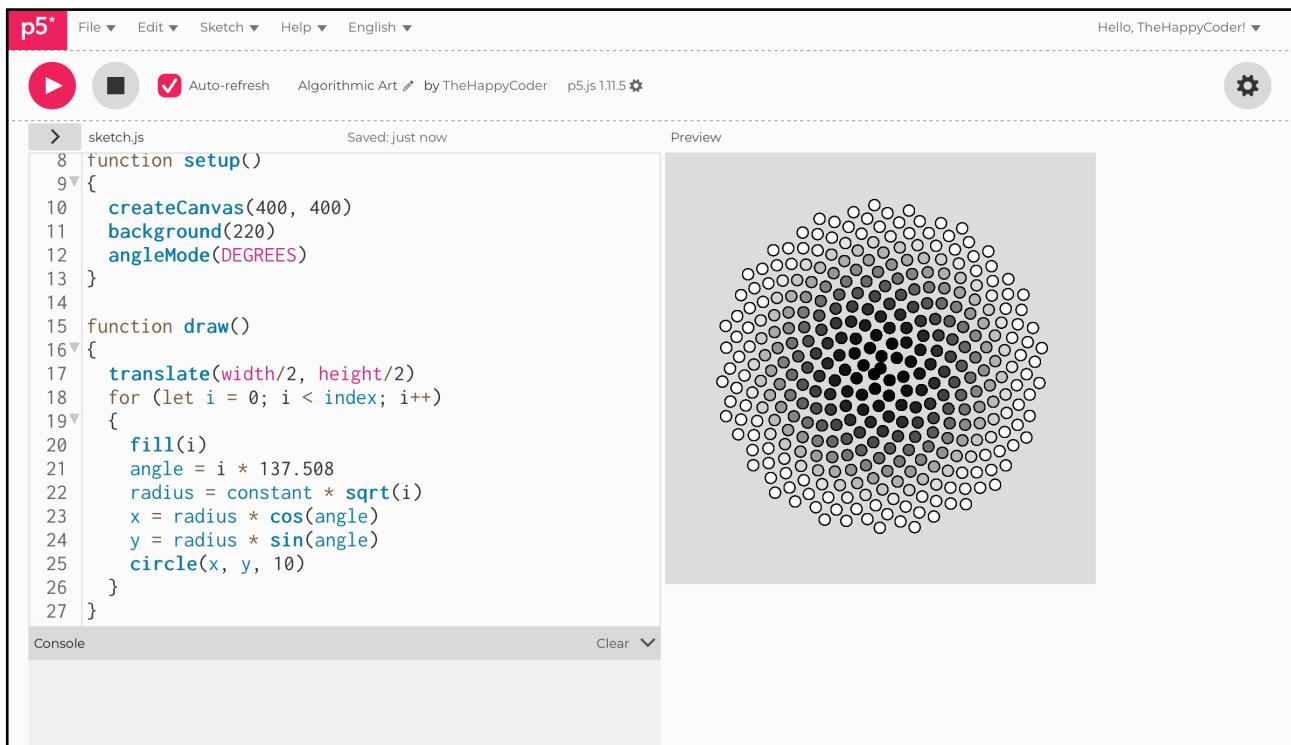
        angle = i * 137.508
        radius = constant * sqrt(i)
        x = radius * cos(angle)
        y = radius * sin(angle)
        circle(x, y, 10)
    }
}
```



Notes

Showing promise.

Figure B10.18



The screenshot shows the p5.js code editor interface. At the top, there are menu options: File ▾, Edit ▾, Sketch ▾, Help ▾, English ▾, and a user greeting "Hello, TheHappyCoder! ▾". On the left, there's a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and a gear for settings. Below the toolbar, the file name is "sketch.js" and it says "Saved: just now". The code editor displays the following JavaScript code:

```
sketch.js
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)
  }
}
```

The preview window on the right shows a circular pattern of small circles arranged in concentric rings, with the density of circles increasing towards the center. The circles are filled with grayscale values corresponding to their index in the loop.



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)
    }
}
```

}

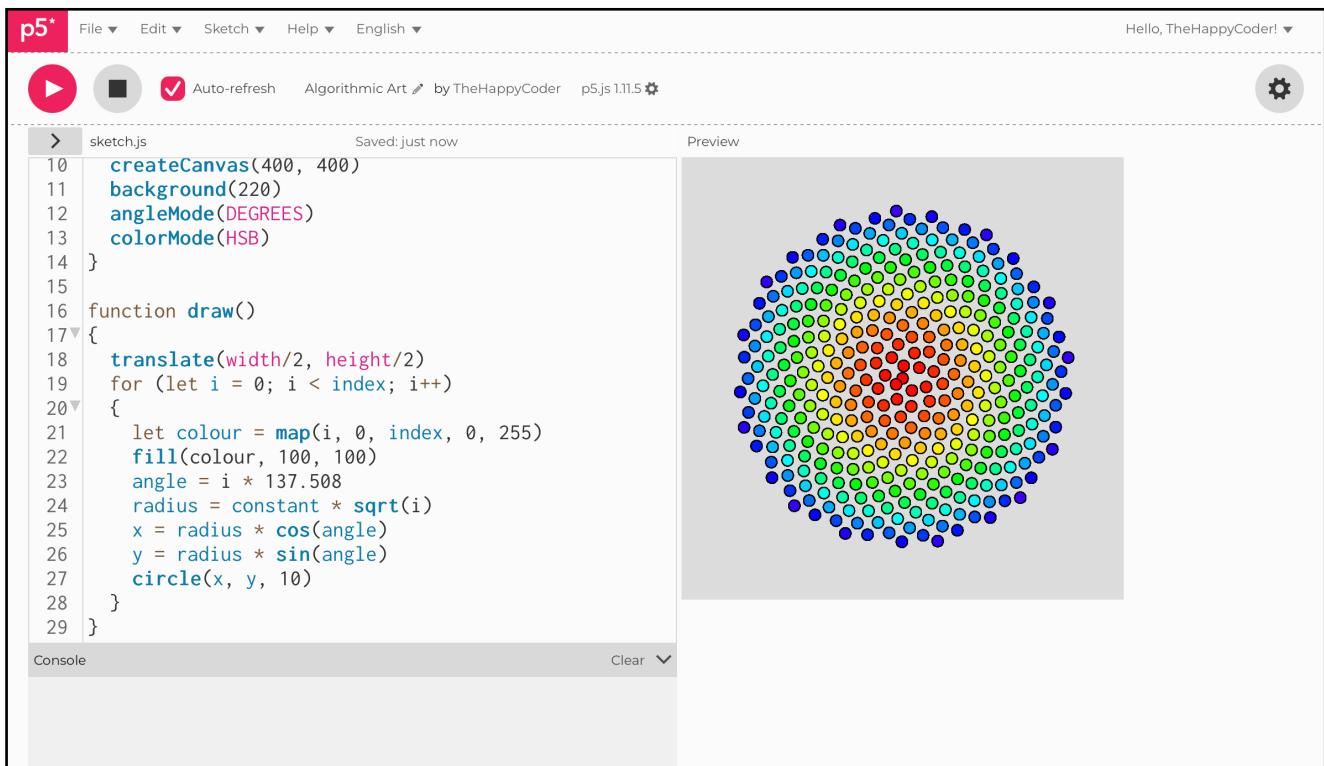
Notes

Now that is so much nicer.

Challenges

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

Figure B10.19



The screenshot shows the p5.js code editor interface. At the top, there are tabs for File, Edit, Sketch, Help, and English. On the right, it says "Hello, TheHappyCoder! ▾". Below the tabs, there are buttons for play/pause, stop, auto-refresh (which is checked), and a gear icon for settings. The title bar says "sketch.js" and "Saved: just now". The preview window on the right shows a circular pattern of colored dots arranged in concentric rings, transitioning from blue at the outer edges to red in the center. The code editor itself has a code area and a console area below it.

```
> sketch.js
10 createCanvas(400, 400)
11 background(220)
12 angleMode(DEGREES)
13 colorMode(HSB)
14 }
15
16 function draw()
17 {
18   translate(width/2, height/2)
19   for (let i = 0; i < index; i++)
20   {
21     let colour = map(i, 0, index, 0, 255)
22     fill(colour, 100, 100)
23     angle = i * 137.508
24     radius = constant * sqrt(i)
25     x = radius * cos(angle)
26     y = radius * sin(angle)
27     circle(x, y, 10)
28   }
29 }
```

Console Clear ▾