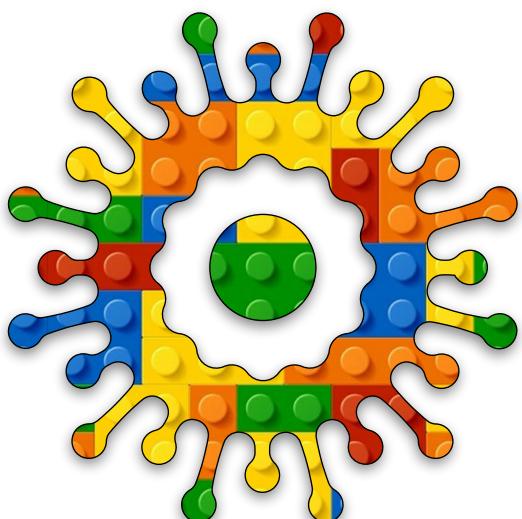


**Creative
Coding
Module B
Unit #5**

**HSB
colours**





Module B Unit #5 HSB colours

- Sketch B5.1 using HSB for the colour
- Sketch B5.2 HSB colour chart
- Sketch B5.3 circle of colour
- Sketch B5.4 centring the circle
- Sketch B5.5 angle of attack
- Sketch B5.6 sine and cosine
- Sketch B5.7 one final thing
- Sketch B5.8 100 rectangles
- Sketch B5.9 HSB saturation
- Sketch B5.10 HSB brightness



Introducing HSB colours

HSB is another colour format, where **HSB** stands for **Hue**, **Saturation**, and **Brightness**. We can only access this format using the **colourMode()** function. It creates pleasant colours but is less intuitive than RGB. The **Hue** has values from **0** to **360**, **Saturation** has values from **0** to **100**, and **Brightness** also has values from **0** to **100**.



Sketch B5.1 using HSB for the colour

HSB (or HSL) stands for Hue, Saturation, and Brightness (or Lightness).

中 Hue

The first argument is the hue, which is on a colour circle from **0** to **360**, where **0** is red, **120** is green, and **240** is blue.

中 Saturation

The second argument is the saturation, which is a percentage from **0** to **100**. It is the amount of colour, so **0** is white to **100** being full colour.

中 Brightness

The third argument is the level of brightness, which is from **0** to **100** also, where **0** is completely dark, and **100** fully bright.

! Start a new sketch and add in the highlighted lines of code. For this, we have to change the `colorMode()` to HSB. Here we get a green background and a blue circle.

```
function setup()
{
    createCanvas(400, 400)
    colorMode(HSB)

}

function draw()
{
    background(0, 50, 100)
    fill(200, 50, 100)
    circle(100, 100, 100)
}
```



Notes

The default is **RGB**, so that is why we call the **colorMode()** for **HSB**. You can switch back by calling the **colorMode(RGB)**. HSB produces some very pleasing colours.



Challenge

Play with the numbers.



Code Explanation

<code>colorMode(HSB)</code>	Sets the mode for HSB values
<code>background(0, 50, 100)</code>	A pale red background
<code>fill(200, 50, 100)</code>	A pale blue circle

Figure B5.1

The screenshot shows the p5.js IDE 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. Below the menu is a toolbar with icons for play, stop, auto-refresh (which is checked), and help. The title bar says 'sketch.js' and 'Saved: just now'. The main area is divided into two sections: 'Preview' on the right showing a red square with a blue circle in the center, and a code editor on the left containing the following JavaScript code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     colorMode(HSB)
5 }
6
7 function draw()
8 {
9     background(0, 50, 100)
10    fill(200, 50, 100)
11    circle(100, 100, 100)
12 }
```

At the bottom of the interface is a 'Console' section with a 'Clear ▾' button.



Sketch B5.2 HSB colour chart

We iterate through the colours **10** steps at a time from **0** to **360**. This is the first argument.

```
function setup()
{
    createCanvas(360, 400)
    colorMode(HSB)
    noStroke()
}

function draw()
{
    for (let i = 0; i < 360; i++)
    {
        fill(i, 100, 100)
        rect(i, 1, 1, 400)
    }
}
```



Notes

We start with red at **0** and end with red at **360**.



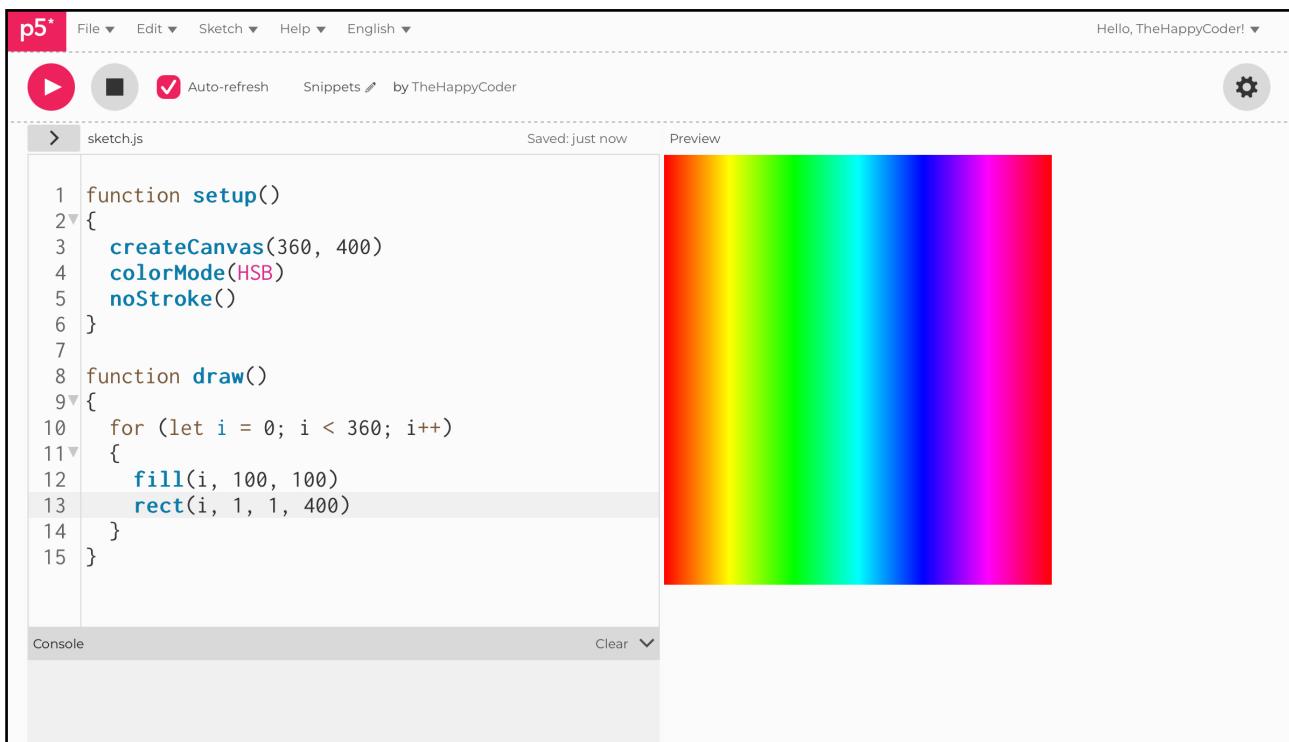
Challenge

You could try to create a circular colour palette.

🛠️ Code Explanation

for (let i = 0; i < 360; i++)	Iterates from 0 to 360 one at a time
fill(i, 100, 100)	Increases the hue value one at a time
rect(i, 1, 1, 400)	Draws a rectangle incrementing each rectangle one pixel at a time

Figure B5.2



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 icons for play, stop, and auto-refresh, followed by 'sketch.js' and 'Saved: just now'. On the far right is a 'Preview' button.

The main area contains the code for 'sketch.js':

```
1 function setup()
2 {
3     createCanvas(360, 400)
4     colorMode(HSB)
5     noStroke()
6 }
7
8 function draw()
9 {
10    for (let i = 0; i < 360; i++)
11    {
12        fill(i, 100, 100)
13        rect(i, 1, 1, 400)
14    }
15 }
```

The preview window on the right displays a vertical color gradient from yellow at the left to magenta at the right, with a thin black border around each colored rectangle.



Sketch B5.3 circle of colour

! new sketch

To get white, you put the saturation to zero.

```
let x = 0
let y = 0

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

function draw()
{
    background(0, 0, 100)
    circle(x, y, 20)
}
```

Notes

Notice how we have given the background a white colour.

Figure B5.3

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 settings gear icon. Below the menu, there are three buttons: a play button, a stop button, and an 'Auto-refresh' button with a checkmark. The title bar says 'sketch.js' and 'Saved: just now'. The main area contains the following code:

```
1 let x = 0
2 let y = 0
3
4 function setup()
5 {
6   createCanvas(400, 400)
7   colorMode(HSB)
8 }
9
10 function draw()
11 {
12   background(0, 0, 100)
13   circle(x, y, 20)
14 }
```

Below the code editor is a preview window showing a blank white canvas. At the bottom of the editor is a 'Console' tab and a 'Clear ▾' button.



Sketch B5.4 centring the circle

It is hard to see, so we will translate the circle to the centre.

```
let x = 0
let y = 0

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

function draw()
{
    background(0, 0, 100)
    translate(width/2, height/2)
    circle(x, y, 20)
}
```

Figure B5.4

The screenshot shows the p5.js web editor interface. At the top, there's a navigation bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', 'English ▾', and a user greeting 'Hello, TheHappyCoder! ▾'. Below the bar are 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 editor contains the following JavaScript code:

```
1 let x = 0
2 let y = 0
3
4 function setup()
5 {
6   createCanvas(400, 400)
7   colorMode(HSB)
8 }
9
10 function draw()
11 {
12   background(0, 0, 100)
13   translate(width/2, height/2)
14   circle(x, y, 20)
15 }
```

The preview window shows a single white circle centered at the bottom of a 400x400 canvas.



Sketch B5.5 angle of attack

We want to draw a series of small circles around a larger circle, spaced out evenly. We will use a `for()` loop to increment 8° from 0° to 360° . We will do this in `degrees` using `angleMode(DEGREES)`.

```
let x = 0
let y = 0

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

function draw()
{
    background(0, 0, 100)
    translate(width/2, height/2)
    for (let angle = 0; angle < 360; angle += 8)
    {
        fill(angle, 100, 100)
        circle(x, y, 20)
    }
}
```

Notes

Notice all this seems to do is fill the circle red; this is because the 360° (and 0°) value of Hue is red. What happens is that it spins through all the colours very fast in the `for()` loop.

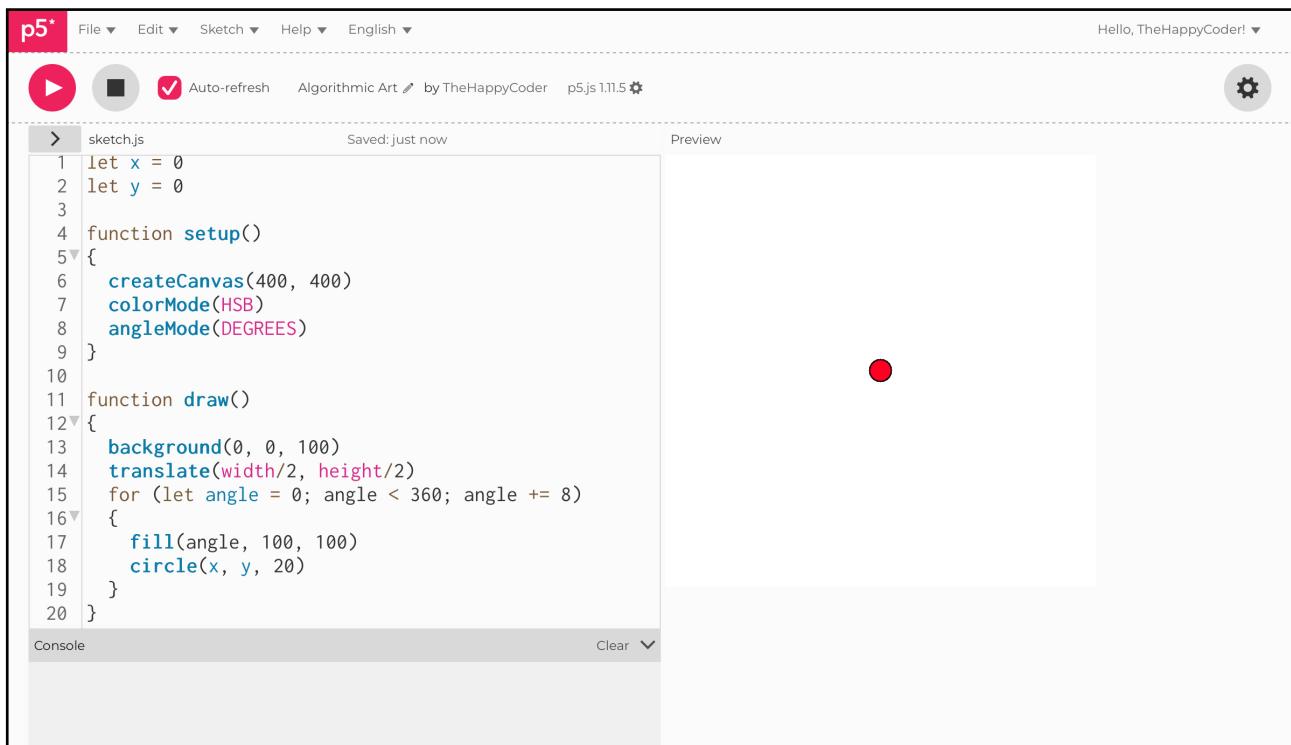
 **Challenge**

Change the **360** to **180**.

 **Code Explanation**

<code>for (let angle = 0; angle < 360; angle += 8)</code>	Spin through all the angles from 0° - 360° by 8° at a time
<code>fill(angle, 100, 100)</code>	Fill the hue value from the angle

Figure B5.5



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 help. The title bar says "Hello, TheHappyCoder! ▾". The code editor window has a file named "sketch.js" with the following content:

```
> sketch.js
1 let x = 0
2 let y = 0
3
4 function setup()
5 {
6   createCanvas(400, 400)
7   colorMode(HSB)
8   angleMode(DEGREES)
9 }
10
11 function draw()
12 {
13   background(0, 0, 100)
14   translate(width/2, height/2)
15   for (let angle = 0; angle < 360; angle += 8)
16   {
17     fill(angle, 100, 100)
18     circle(x, y, 20)
19   }
20 }
```

The preview area on the right shows a canvas with a single red dot at the center. The console area below the code editor is empty.



Sketch B5.6 sine and cosine

We use the principles discussed a few units back, where the coordinates on a circle can be described using **sine** and **cosine** for **x** and **y** about the centre of the circle.

```
let x = 0
let y = 0
let radius = 180

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

function draw()
{
    background(0, 0, 100)
    translate(width/2, height/2)
    for (let angle = 0; angle < 360; angle += 8)
    {
        fill(angle, 100, 100)
        x = radius * sin(angle)
        y = radius * cos(angle)
        circle(x, y, 20)
    }
}
```

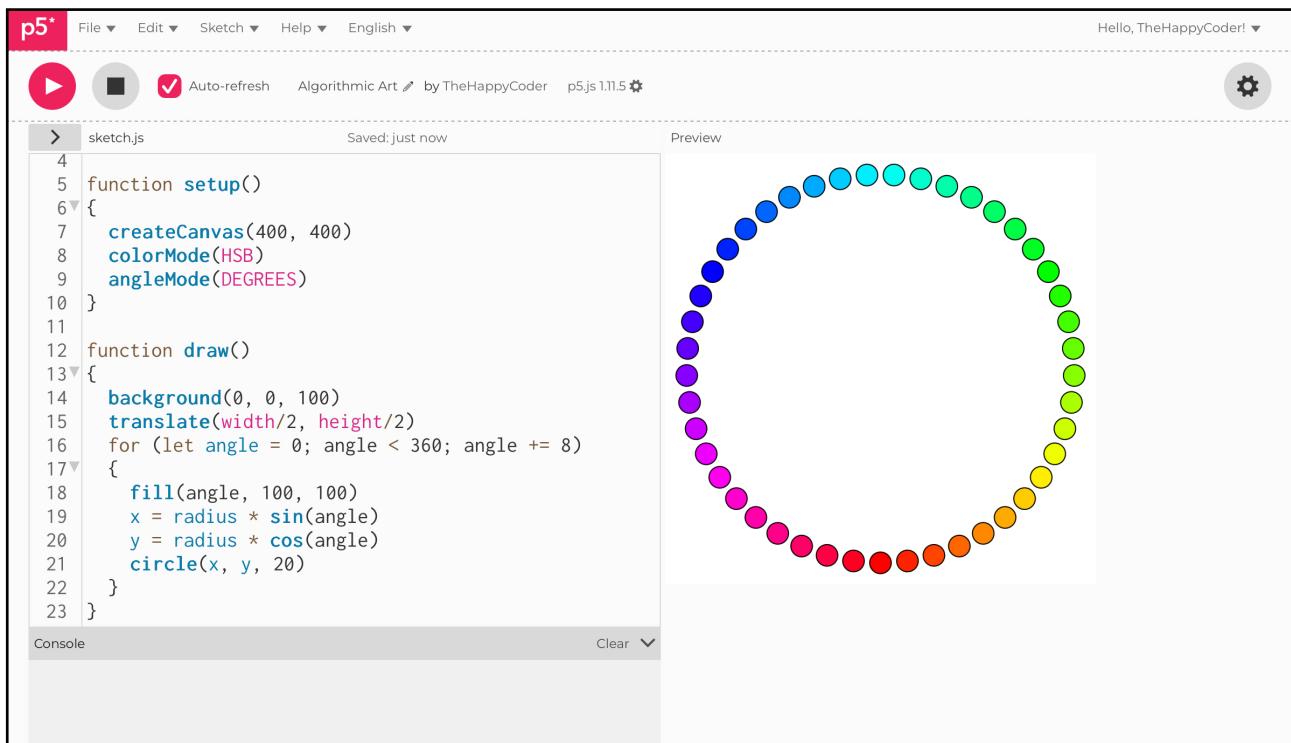
Notes

You now have the 360-degree

 Challenge

Try different radii and spacings of the circles

Figure B5.6



The screenshot shows the p5.js IDE interface. The top bar includes the p5 logo, file navigation, and a user profile. The code editor on the left contains a script named 'sketch.js' with the following code:

```
4
5 function setup()
6 {
7   createCanvas(400, 400)
8   colorMode(HSB)
9   angleMode(DEGREES)
10 }
11
12 function draw()
13 {
14   background(0, 0, 100)
15   translate(width/2, height/2)
16   for (let angle = 0; angle < 360; angle += 8)
17   {
18     fill(angle, 100, 100)
19     x = radius * sin(angle)
20     y = radius * cos(angle)
21     circle(x, y, 20)
22   }
23 }
```

The preview window on the right displays a circular pattern of 45 points, each a small circle with a different color from the HSB color space, transitioning from purple at the top-left to yellow at the bottom-right.



Sketch B5.7 one final thing

Let's draw a nice box round the canvas so we can see where the canvas starts. Even for something as simple as this, you need to think about translation and the impact on coordinates.

```
let x = 0
let y = 0
let radius = 180

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

function draw()
{
    background(0, 0, 100)
    translate(width/2, height/2)
    for (let angle = 0; angle < 360; angle += 8)
    {
        fill(angle, 100, 100)
        x = radius * sin(angle)
        y = radius * cos(angle)
        circle(x, y, 20)
    }
    noFill()
    rect(-width/2, -height/2, width, height)
}
```



Notes

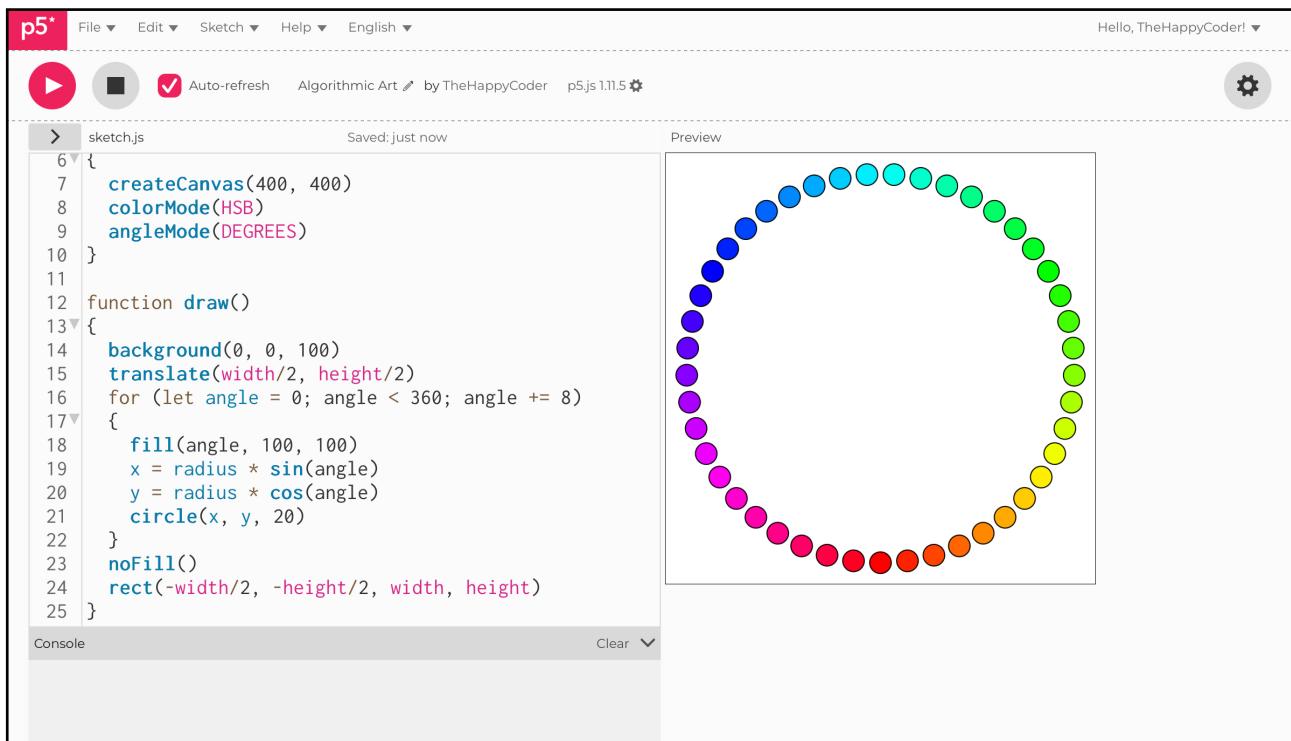
Looks a lot nicer.



Challenge

If you put it inside the loop, what would you need to change?

Figure B5.7



The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, refresh, and settings. The title bar says "Hello, TheHappyCoder! ▾". The main area has tabs for "sketch.js" and "Preview". The code editor on the left contains the following JavaScript code:

```
sketch.js
1 createCanvas(400, 400)
2 colorMode(HSB)
3 angleMode(DEGREES)
4
5 function draw()
6 {
7     background(0, 0, 100)
8     translate(width/2, height/2)
9     for (let angle = 0; angle < 360; angle += 8)
10    {
11        fill(angle, 100, 100)
12        x = radius * sin(angle)
13        y = radius * cos(angle)
14        circle(x, y, 20)
15    }
16    noFill()
17    rect(-width/2, -height/2, width, height)
18 }
```

The preview window on the right displays a circular pattern of 45 small circles arranged in an arc from the bottom-left to the top-right. The colors transition smoothly through the HSB color space, starting at purple on the left and ending at yellow-green on the right.



Sketch B5.8 100 rectangles

! Starting a new sketch where we loop through 100 rectangles, with each rectangle being 4 pixels wide and 400 pixels long; hence, we multiply the i variable by 4.

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

function draw()
{
    background(0, 0, 100)
    for (let i = 0; i < 100; i++)
    {
        rect(i * 4, 0, 4, 400)
    }
}
```

Notes

We just get a bunch of rectangles from left to right.

Figure B5.8

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 side of the header, it says 'Hello, TheHappyCoder! ▾'. Below the header, there are icons for play, stop, and refresh, followed by 'Auto-refresh' with a checked checkbox. The title 'Algorithmic Art' and 'by TheHappyCoder' are displayed next to the refresh icon. The file name 'sketch.js' is shown in the top left of the code editor area. The code itself is:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     colorMode(HSB)
5 }
6
7 function draw()
8 {
9     background(0, 0, 100)
10    for (let i = 0; i < 100; i++)
11    {
12        rect(i * 4, 0, 4, 400)
13    }
14 }
```

To the right of the code editor is a 'Preview' window showing a 400x400 grid of vertical black lines on a white background. At the bottom of the interface, there's a 'Console' tab and a 'Clear ▾' button.



Sketch B5.9 HSB saturation

Then we fill each rectangle with a hue of **0**, which is red; the saturation (second argument) we increase by **1** on each iteration of the loop, keeping the brightness (third argument) at maximum (**100**).

```
function setup()
{
    createCanvas(400, 400)
    colorMode(HSB)
    noStroke()
}

function draw()
{
    background(0, 0, 100)
    for (let i = 0; i < 100; i++)
    {
        fill(0, i, 100)
        rect(i * 4, 0, 4, 400)
    }
}
```

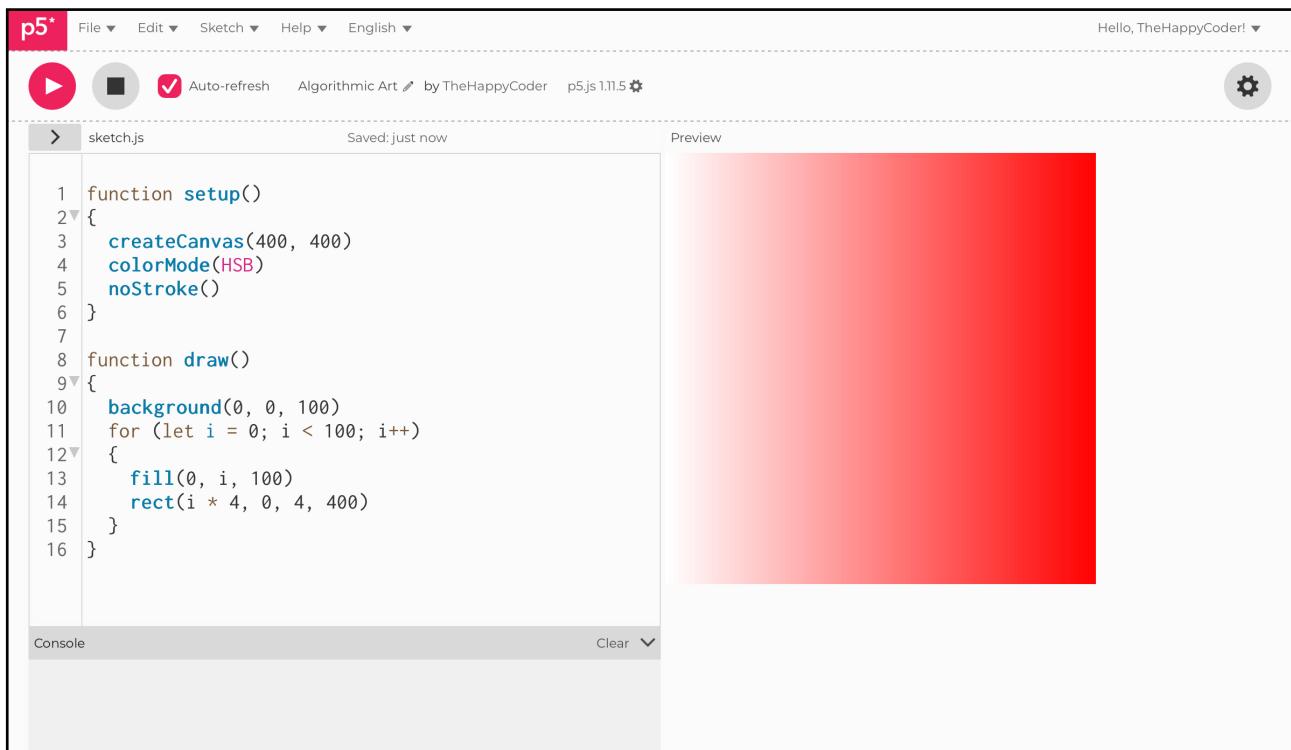
Notes

A saturation of **0** is no saturation (**white**), and **100** is the maximum; then everything in between.

Challenge

Do this with other **hues**.

Figure B5.9



The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by the text "Auto-refresh" with a checked checkbox, "Algorithmic Art" by "TheHappyCoder", and "p5.js 1.11.5". On the right, it says "Hello, TheHappyCoder!" with a dropdown arrow. Below the toolbar, the file name "sketch.js" is shown along with "Saved: just now". To the right, under "Preview", is a large red square. The main area contains the following code:

```
function setup() {
  createCanvas(400, 400)
  colorMode(HSB)
  noStroke()
}

function draw() {
  background(0, 0, 100)
  for (let i = 0; i < 100; i++) {
    fill(0, i, 100)
    rect(i * 4, 0, 4, 400)
  }
}
```

At the bottom left is a "Console" tab and at the bottom right is a "Clear" button.



Sketch B5.10 HSB brightness

Now for the brightness, this is the third argument: we change the `fill()` function, giving it `0` hue, `100` saturation, and an incrementing `i` variable for the brightness. This means that it starts off with `zero` brightness and increments to `full` brightness (100%).

```
function setup()
{
    createCanvas(400, 400)
    colorMode(HSB)
    noStroke()
}

function draw()
{
    background(0, 0, 100)
    for (let i = 0; i < 100; i++)
    {
        fill(0, 100, i)
        rect(i * 4, 0, 4, 400)
    }
}
```



Notes

Fairly obvious results.



Challenge

Create a palette of colours that you might want to use regularly.

Figure B5.10

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 icons for play, stop, auto-refresh (which is checked), and preview. The main area has a code editor on the left and a preview canvas on the right. The code editor contains the following JavaScript code:

```
1 function setup()
2 {
3   createCanvas(400, 400)
4   colorMode(HSB)
5   noStroke()
6 }
7
8 function draw()
9 {
10  background(0, 0, 100)
11  for (let i = 0; i < 100; i++)
12  {
13    fill(0, 100, i)
14    rect(i * 4, 0, 4, 400)
15  }
16 }
```

The preview canvas shows a vertical gradient from black at the top to red at the bottom, with 25 horizontal bars of varying widths.