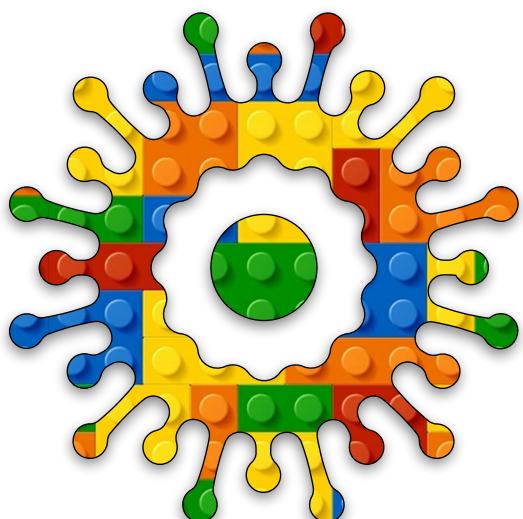


Creative Coding

Module A

Unit #7

pixels





Module A Unit #7 pixels

- Sketch A7.1 what's the point?
- Sketch A7.2 a line of pixels
- Sketch A7.3 nested loop
- Sketch A7.4 colouring the pixel
- Sketch A7.5 random pixel colour
- Sketch A7.6 gradual colour
- Sketch A7.7 colour mode RGB
- Sketch A7.8 in both directions



Introduction to 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:

- 中 pixels
- 中 point()
- 中 nested loops
- 中 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

No! 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 is draws a pixel at coordinates (0, 0)
-------------	---

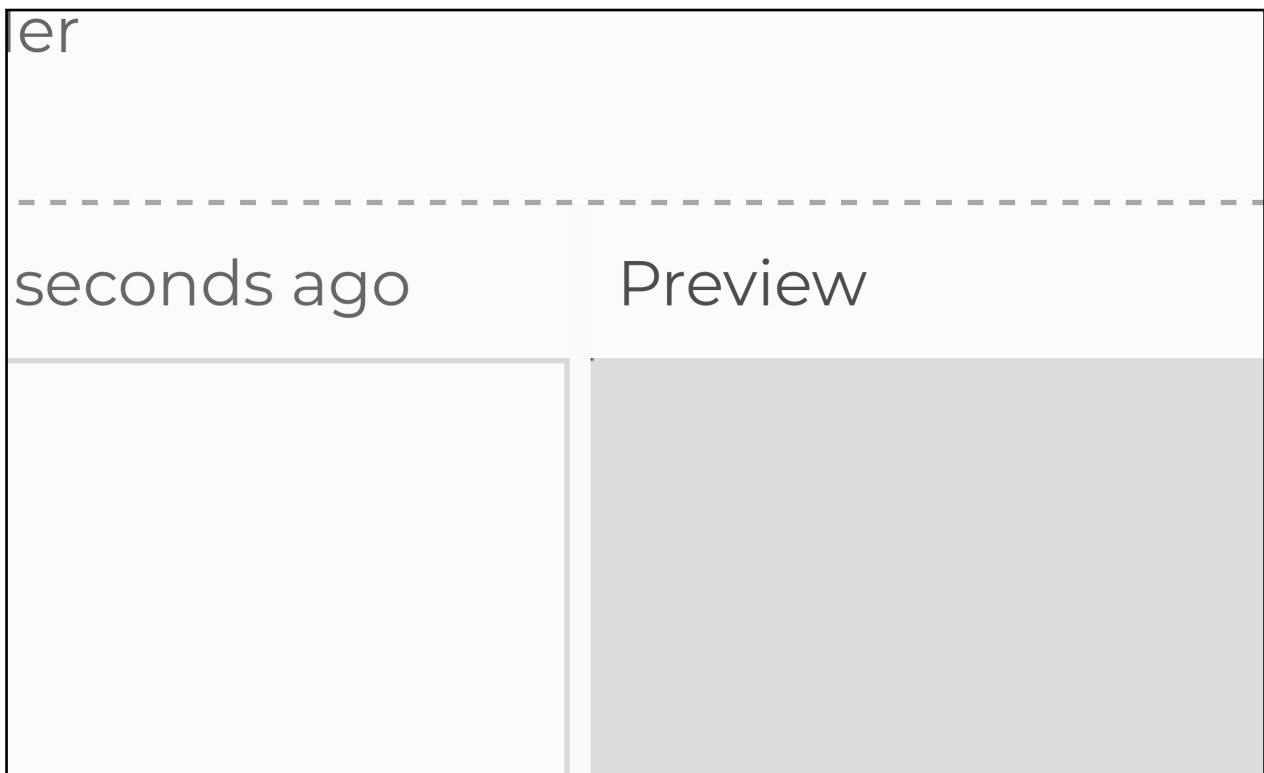
Figure A7.1a

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 and 'Algorithmic Art' with a small icon. The main workspace is titled 'sketch.js' and contains the following code:

```
1 function setup()
2 {
3   createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8   background(220)
9   point(0, 0)
10 }
```

To the right of the code is a preview window showing a light gray canvas. At the bottom of the editor, there's a 'Console' tab and a 'Clear ▾' button.

Figure A7.1b: pixel point





Sketch A7.2 a line of pixels

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

<code>for (let x = 0; x < width; x++)</code>	A <code>for()</code> loop for the <code>x</code> value
<code>point(x, 0)</code>	Draw pixel for each <code>x</code> value

Figure A7.2

The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by menu items: File, Edit, Sketch, Help, and English. To the right, it says "Hello, TheHappyCoder!" and has a gear icon. The main area has tabs for "sketch.js" and "Preview". The code editor contains the following JavaScript code:

```
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 }
```

The preview window shows a light gray canvas with a series of small black dots arranged vertically along the left edge, representing the output of the code.



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)
        }
    }
}
```

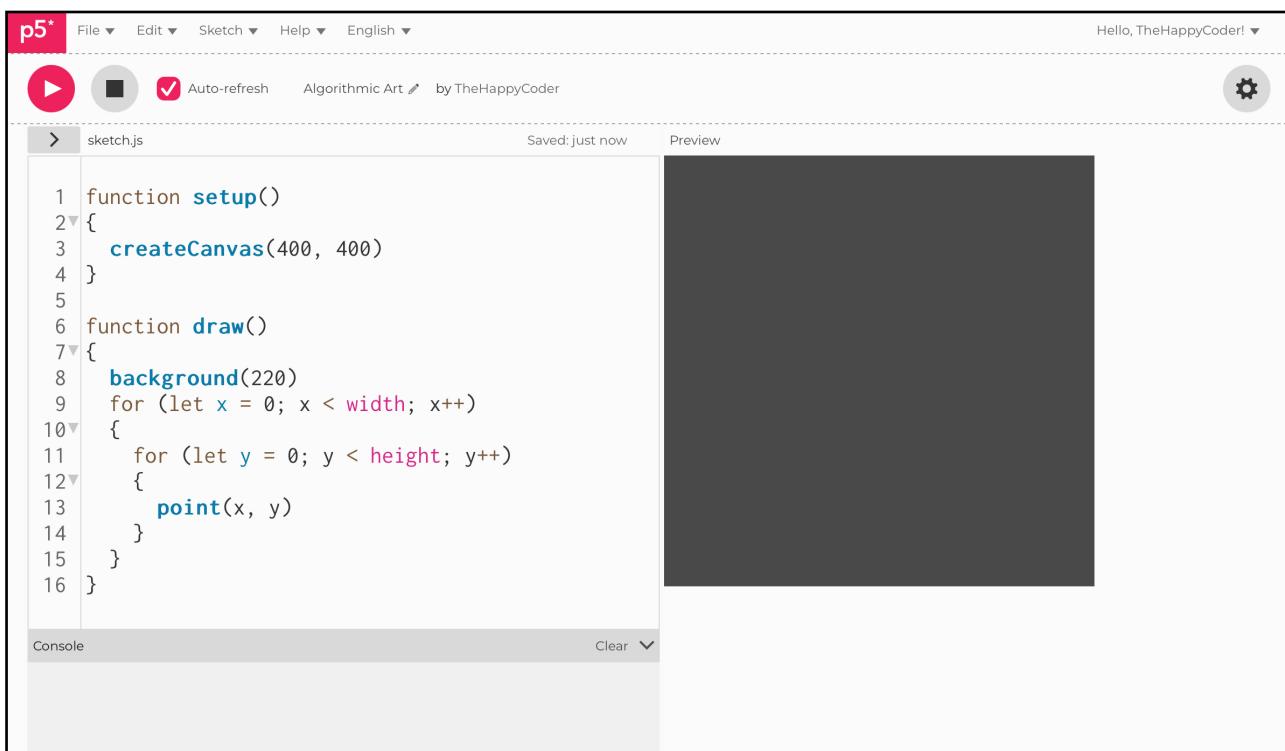
Notes

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

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

Figure A7.3



The screenshot shows the p5.js code editor interface. At the top, there's a red header bar with the p5 logo. Below it, the menu bar includes File, Edit, Sketch, Help, and English. On the right side of the header, it says "Hello, TheHappyCoder!" with a dropdown arrow. The main workspace has a title bar "sketch.js" and status bars "Saved: just now" and "Preview". The code area contains the following JavaScript code:

```
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      point(x, y)
14    }
15  }
16 }
```

The preview window on the right shows a solid dark gray square, which is the result of the code running.



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

The screenshot shows the p5.js IDE interface. At the top, there's a red header bar with the p5 logo and menu items: File, Edit, Sketch, Help, and English. To the right of the header is a message: "Hello, TheHappyCoder! ▾". Below the header is a toolbar with icons for play, stop, auto-refresh (which is checked), and a gear for settings. The main workspace is titled "sketch.js" and shows the following code:

```
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('lightblue')
14            point(x, y)
15        }
16    }
17 }
```

To the right of the code editor is a preview window showing a light blue square canvas. At the bottom of the interface are two tabs: "Console" and "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()
}
```

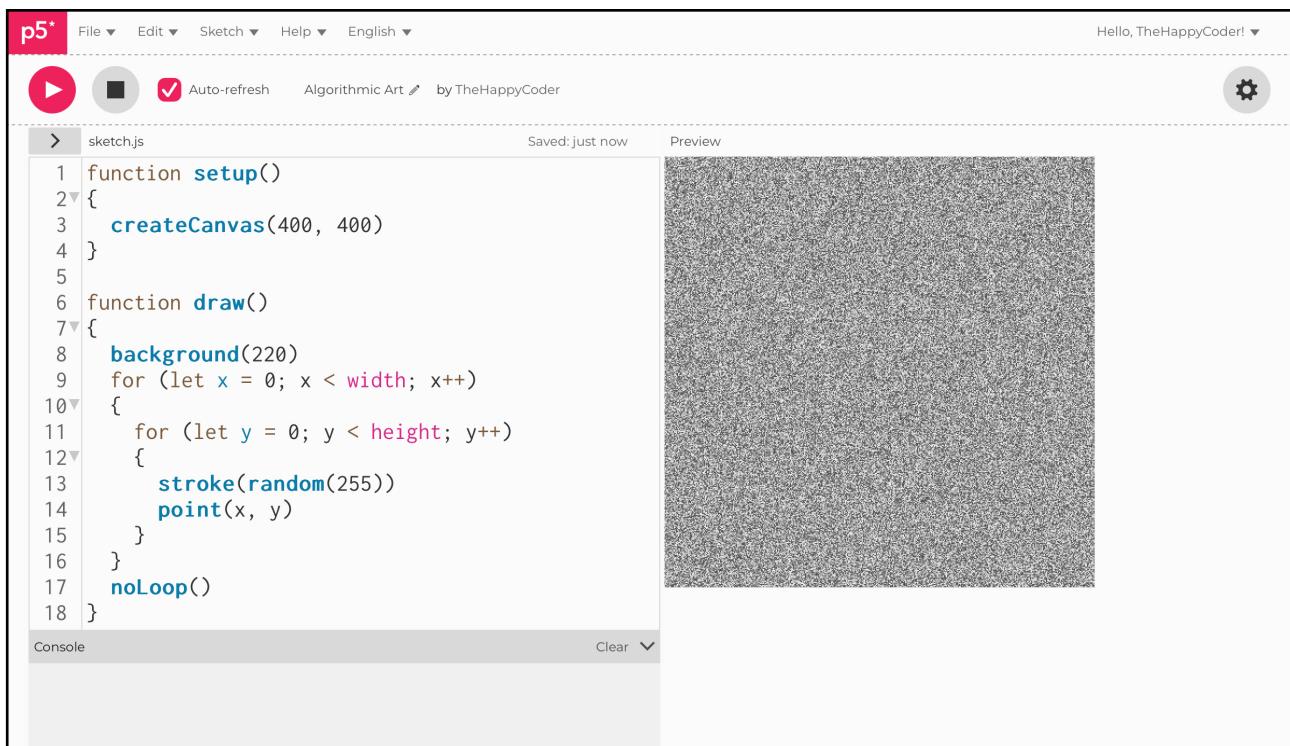
Notes

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

Challenge

Try random RGB colours.

Figure A7.5



The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by menu items: File, Edit, Sketch, Help, and English. To the right, it says "Hello, TheHappyCoder!" and has 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
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 }
```

The preview window on the right displays a 400x400 pixel square filled with a dense, uniform noise pattern in grayscale.



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()
}
```

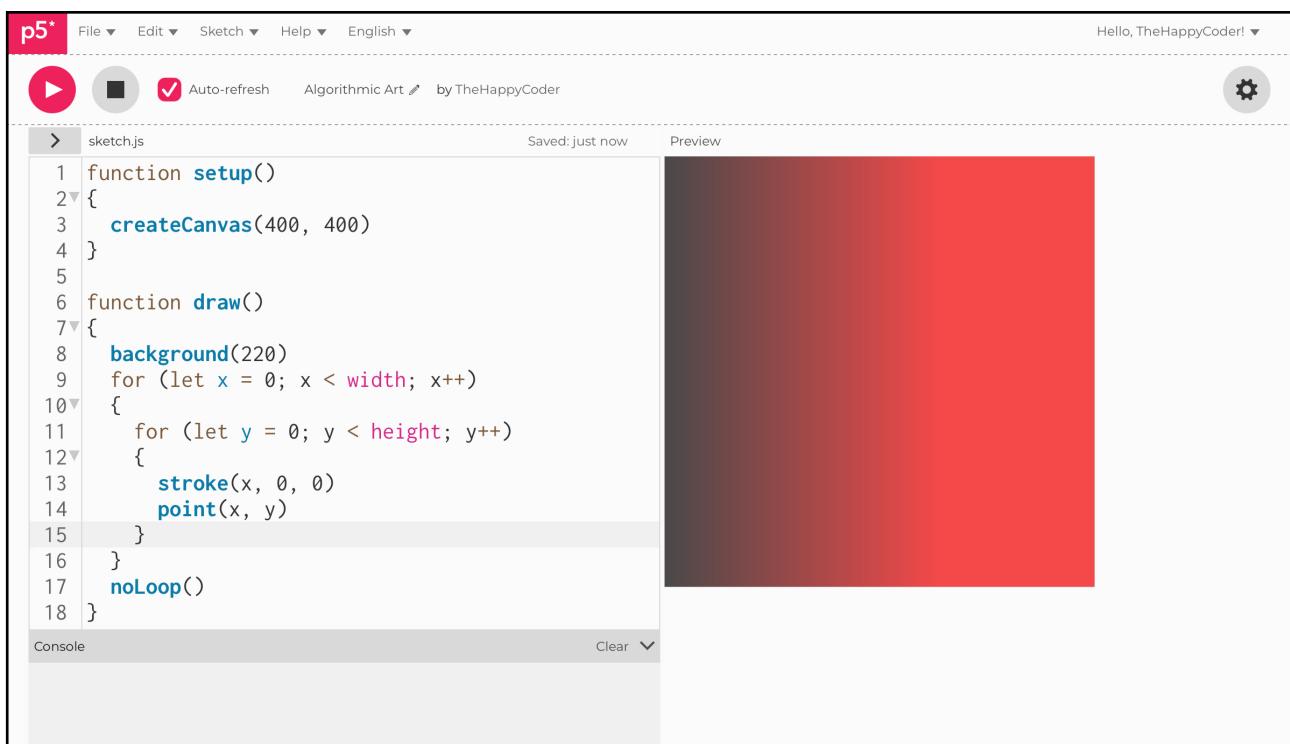
Notes

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



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" and "Algorithmic Art" by TheHappyCoder. On the right, it says "Hello, TheHappyCoder!" with a gear icon. Below the toolbar, the code editor window has a title bar with "sketch.js" and "Saved: just now". To the right of the code editor is a preview canvas showing a vertical gradient from dark gray at the top to bright red at the bottom. The code editor itself contains the following JavaScript code:

```
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()
}
```

At the bottom of the code editor, there are "Console" and "Clear" buttons.



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`).

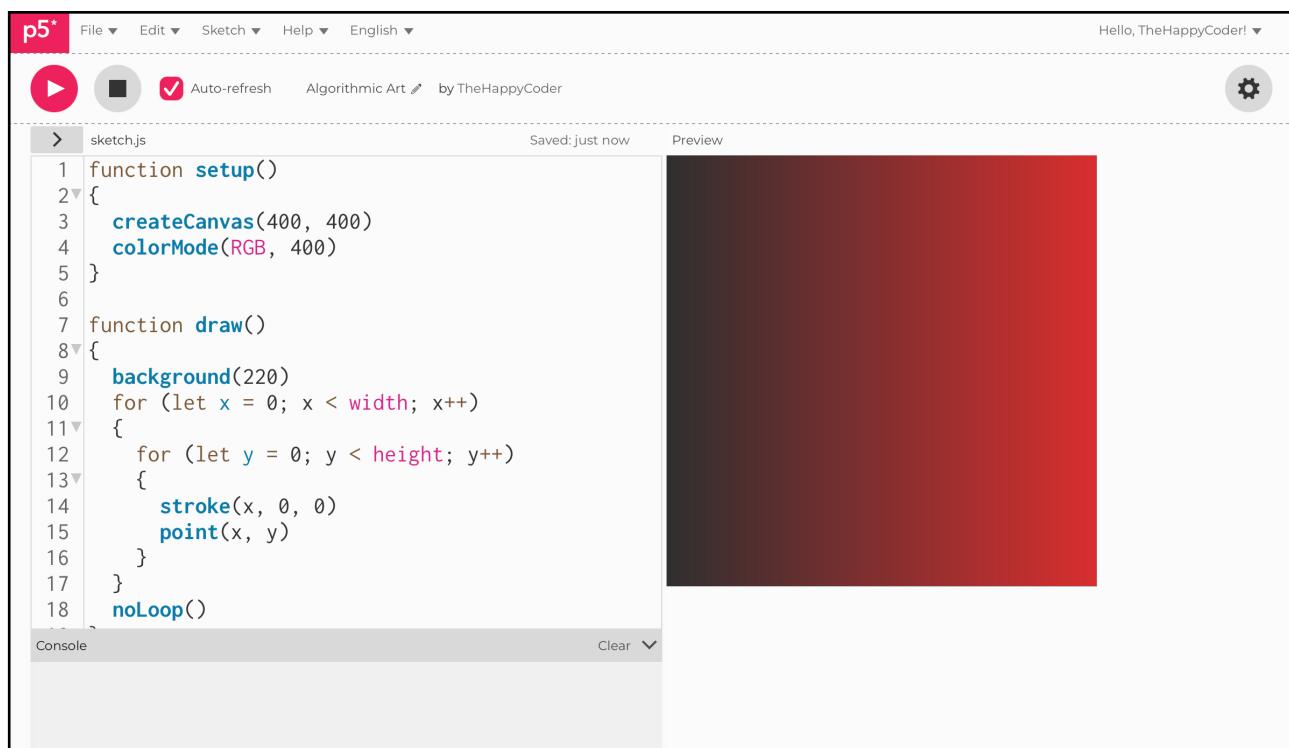
Challenges

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



The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by the text "Auto-refresh" and "Algorithmic Art" by TheHappyCoder. 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" and "Preview". The code editor area contains the following JavaScript code:

```
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()
}
```

Below the code editor is a "Console" section with a "Clear" button. To the right of the code editor is a preview window showing a black square with a diagonal red gradient from bottom-left to top-right.



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()
}
```

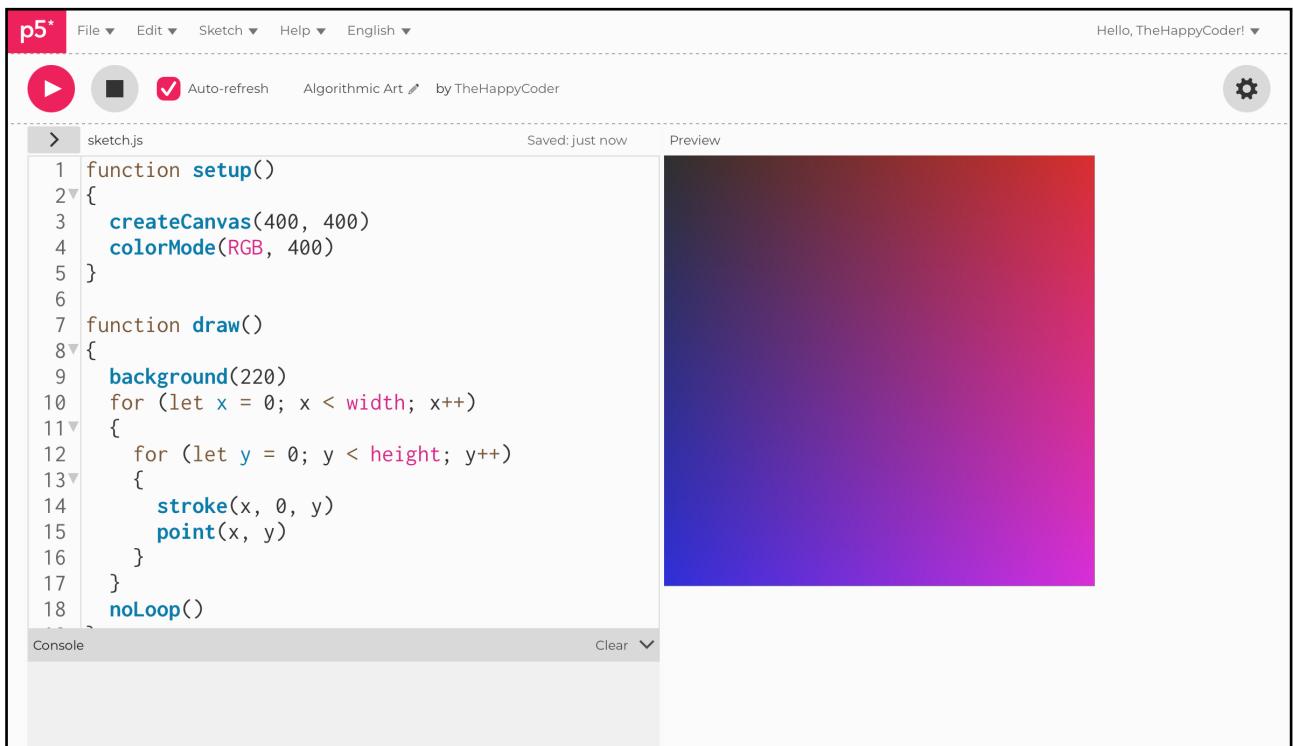
Notes

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



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 user profile 'Hello, TheHappyCoder!'. Below the menu is a toolbar with a play button, a square, a refresh icon with a checkmark labeled 'Auto-refresh', and a gear icon. The main area has tabs for 'sketch.js' and 'Preview'. The code editor contains the following JavaScript code:

```
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()
}
```

The 'Preview' window on the right displays a 400x400 pixel canvas with a smooth radial gradient from dark purple at the top-left to bright yellow at the bottom-right. The preview window has a 'Saved: just now' timestamp and a 'Preview' tab.



Next. . .

In the final unit (#8) of this module, we will put much of what we have learned into creating a classic pattern called 10PRINT.