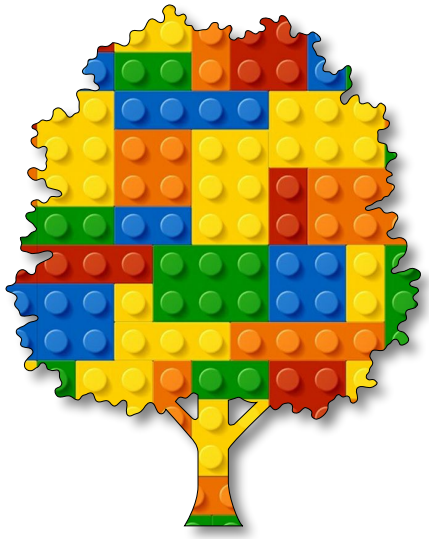
Algorithmic Art

Module B

Unit #8

irregular shapes



Content

Module B Unit #8 vertex and bézier

Introduction to irregular shapes

It is one thing to draw regular shapes using the functions circle, square, rectangle, and triangle, etc., but what if you want to draw something that isn't a regular shape? Then you can use the `vertex()` function to draw any shape. This is great for drawing random patterns or shapes and manipulating them. The `bezier()` function allows us to draw curves.

We start with `beginShape()` and add vertex co-ordinates for `x` and `y`, and then add `endShape()` at the end.

There are many types; here are two:

- vertex()
- bezier()

Vertex lines

A vertex is a point in space; here we are using it in 2D, but you can also do this in 3D (next module). It needs the x and y coordinates. You can have as many vertices as you would like. To draw the lines between the vertices, you need to use the functions (commands) `beginShape()` and `endShape()`.

Sketch B8.1 drawing a vertex line

We are going to start very simply by drawing a line. We specify the end coordinates of the line, and it draws a line between them; there is no line function. Although this seems a little protracted, it has great value when you use the coordinates in creative ways, drawing complex shapes and patterns.

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

function draw()
{
  background(220)
  beginShape()
  vertex(100, 100)
  vertex(300, 300)
  endShape()
}
```

🗒️ Notes

Using the `vertex()` function is a useful tool to draw irregular polygons, in other words, shapes other than rectangles and triangles. A vertex simply draws a dot (pixel) at the co-ordinate given. The `beginShape()` and `endShape()` functions join the dots up. In this example, to get you started, two dots are used, so it gives you one line.

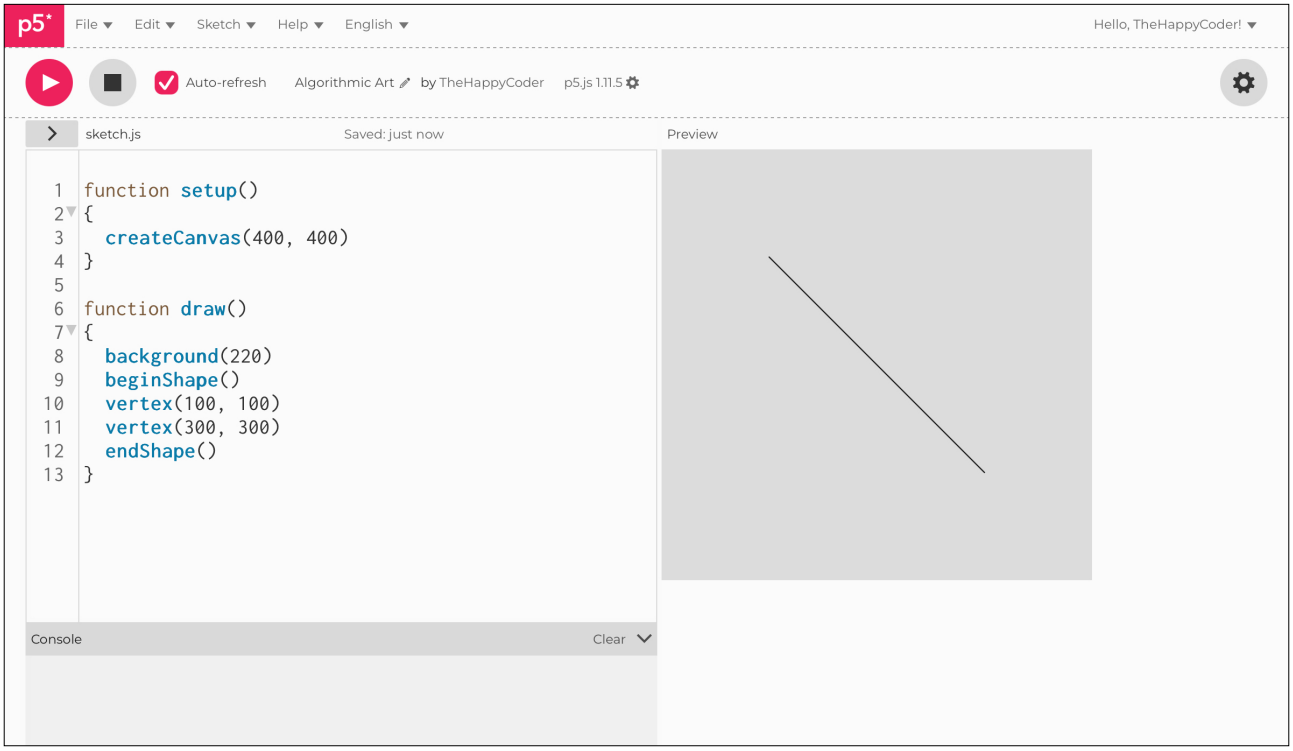
🌻 Challenge

Try it without the `beginShape()` and `endShape()`.

🛠️ Code Explanation

`beginShape()`	Starts adding vertices to an irregular shape
`vertex(100, 100)`	Defines a vertex with x and y coordinates
`endShape()`	Stops adding vertices to an irregular shape

Figure B8.1


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
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
beginShape()
vertex(100, 100)
vertex(300, 300)
endShape()
}
Console
Clear

Sketch B8.2 drawing a square using vertex

Drawing a simple square using the vertex coordinates, the order is critical. It draws a line from one vertex to the next one, in order, on the list. So don't get them mixed up. Notice that we have five vertices, whereas a square should have only four; do you know why?

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

function draw()
{
  background(220)
  beginShape()
  vertex(100, 100)
  vertex(300, 100)
  vertex(300, 300)
  vertex(100, 300)
  vertex(100, 100)
  endShape()
}
```

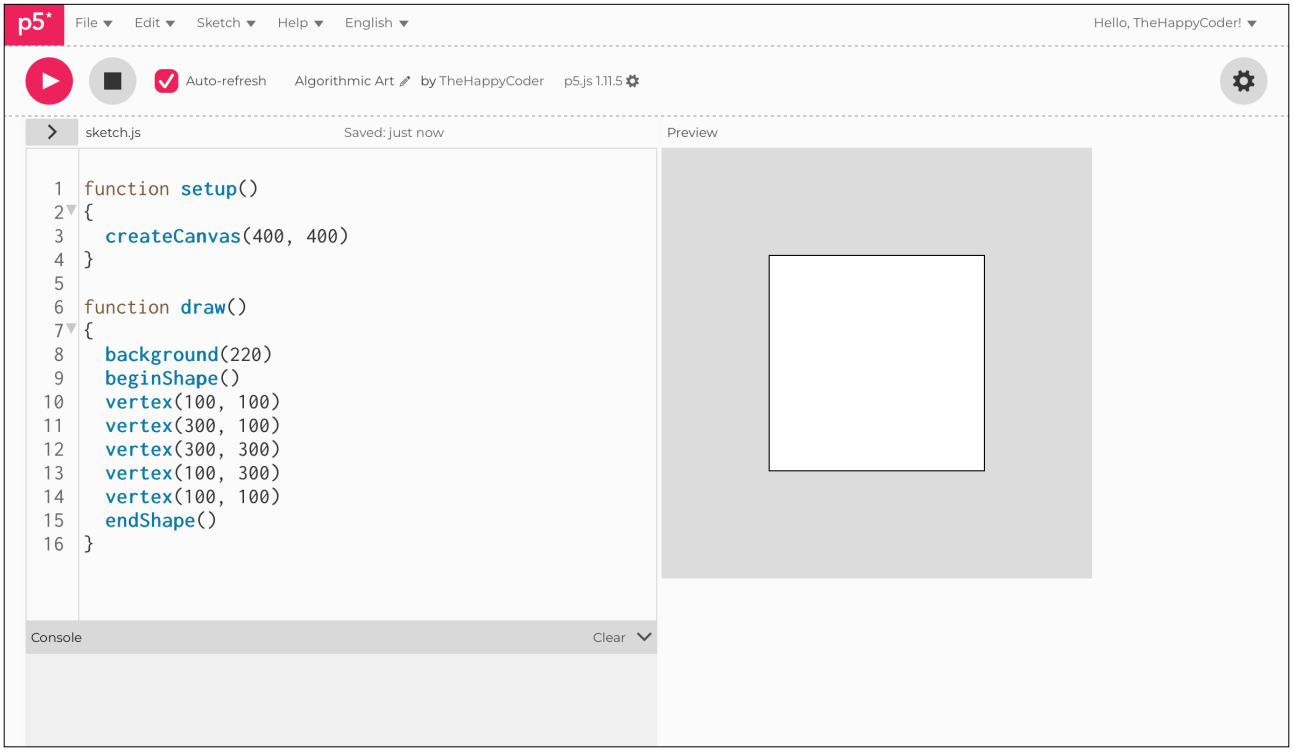
🗒️ Notes

Now to draw a square using more dots and therefore more lines. For this to work, you need to close the shape by drawing the last dot where the first one was drawn. In the next example, there is a shortcut to this. Notice that it fills the shape automatically.

🌻 Challenges

1. Reorder the vertices and see what happens.
2. Try to draw a couple of rectangles.

Figure B8.2


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
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
beginShape()
vertex(100, 100)
vertex(300, 100)
vertex(300, 300)
vertex(100, 300)
vertex(100, 100)
endShape()
}
Console
Clear

Sketch B8.3 that was CLOSE

❗ Remove the last vertex point.
This joins up the first and last vertices automatically. We can remove the last vertex so that we only have four and add the CLOSE command. Although we have four vertices for the square, it doesn't draw the final line until you add the word CLOSE to the endShape().

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

function draw()
{
  background(220)
  beginShape()
  vertex(100, 100)
  vertex(300, 100)
  vertex(300, 300)
  vertex(100, 300)
  endShape(CLOSE)
}
```

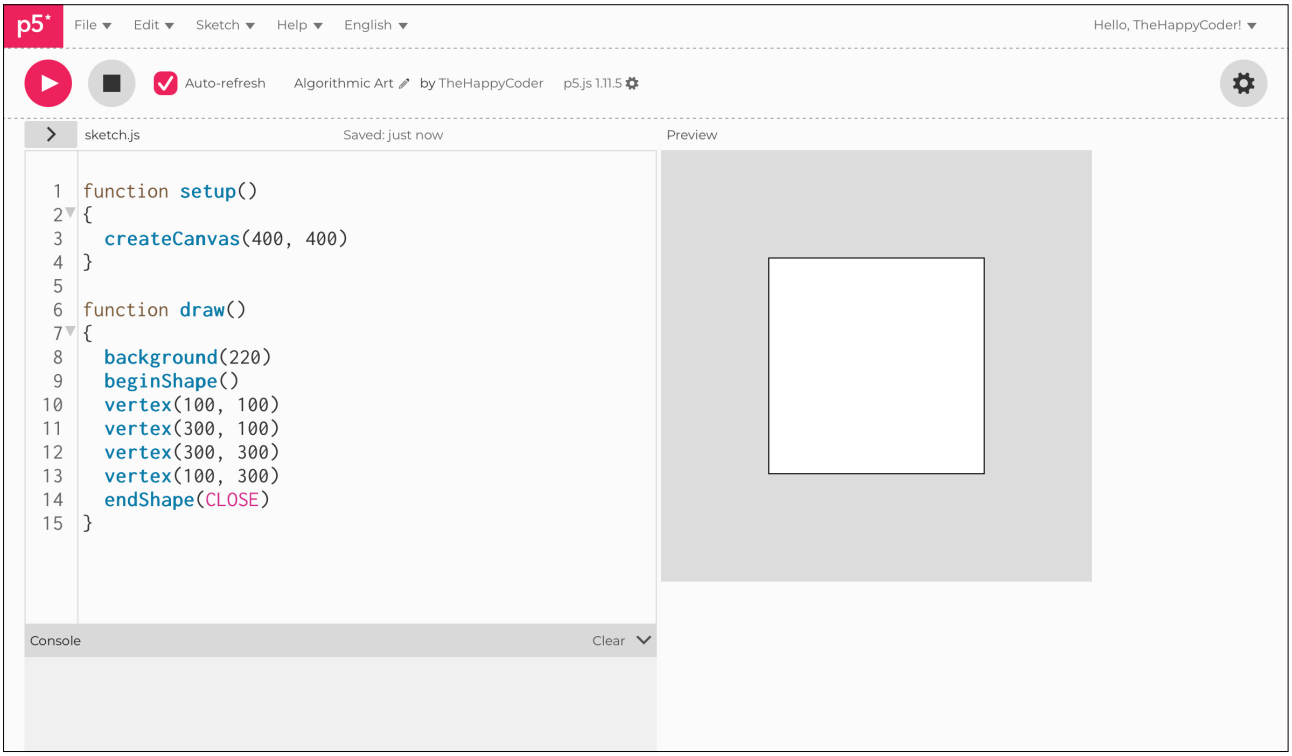
🗒️ Notes
Using endShape(CLOSE) means you don't have to draw the last dot to join it all up. Notice it is in capital letters.

🌻 Challenge
Try the above without the CLOSE argument and see the difference.

🛠️ Code Explanation

endShape(CLOSE)	Draws the final line between the first and last vertices

Figure B8.3


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 }
5
6 function draw()
7 {
8 background(220)
9 beginShape()
10 vertex(100, 100)
11 vertex(300, 100)
12 vertex(300, 300)
13 vertex(100, 300)
14 endShape(CLOSE)
15 }
Console
Clear

Sketch B8.4 making a more irregular shape

Adding in a couple of vertices.

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

function draw()
{
  background(220)
  beginShape()
  vertex(100, 100)
  vertex(300, 100)
  vertex(200, 150)
  vertex(300, 300)
  vertex(100, 300)
  vertex(200, 250)
  endShape(CLOSE)
}
```

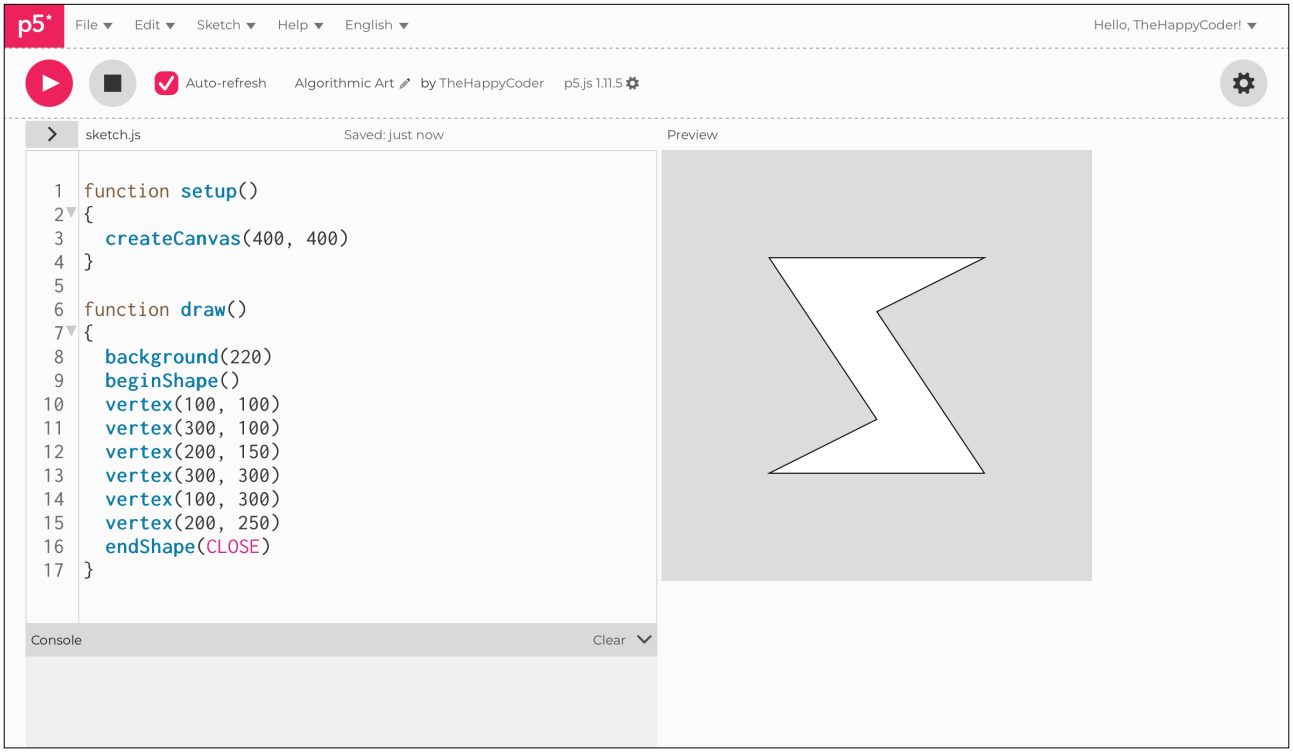
🗒️ Notes

A slightly more interesting shape.

🌻 Challenge

Make your own shape, a star perhaps!

Figure B8.4


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
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
beginShape()
vertex(100, 100)
vertex(300, 100)
vertex(200, 150)
vertex(300, 300)
vertex(100, 300)
vertex(200, 250)
endShape(CLOSE)
}
Console
Clear

Sketch B8.5 translating

We will now translate to the centre of the canvas. There is going to be some reorganising of the coordinates, as you might expect. Have a go yourself first and see if you can do it. Using pen and paper can help to work out the new coordinates.

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

function draw()
{
  background(220)
  translate(width/2, height/2)
  beginShape()
  vertex(-100, -100)
  vertex(100, -100)
  vertex(0, -50)
  vertex(100, 100)
  vertex(-100, 100)
  vertex(0, 50)
  endShape(CLOSE)
}
```

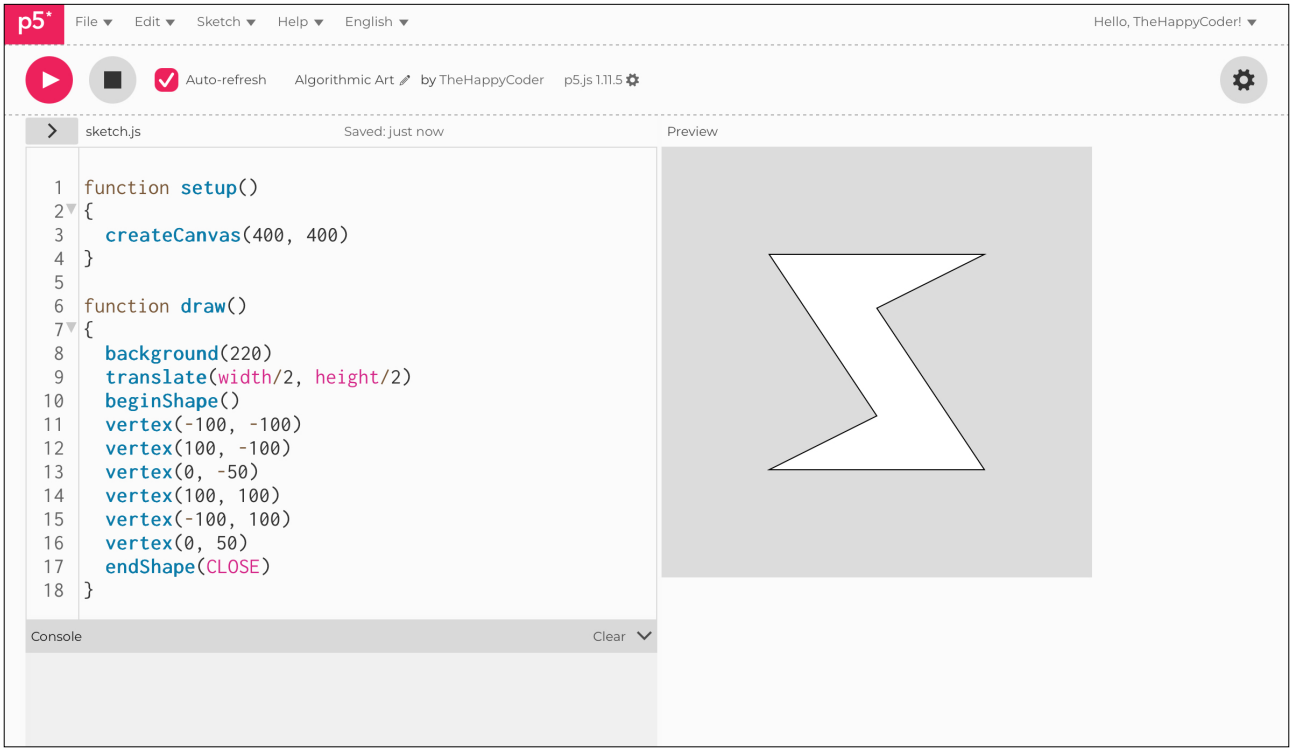
Notes

We just subtracted 200.

Challenge

Create a variable to subtract.

Figure B8.5


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
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
translate(width/2, height/2)
beginShape()
vertex(-100, -100)
vertex(100, -100)
vertex(0, -50)
vertex(100, 100)
vertex(-100, 100)
vertex(0, 50)
endShape(CLOSE)
}
Console
Clear

Sketch B8.6 rotating

We can now rotate the shape.

```
let angle = 0

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

function draw()
{
  background(220)
  translate(width/2, height/2)
  rotate(angle)
  beginShape()
  vertex(-100, -100)
  vertex(100, -100)
  vertex(0, -50)
  vertex(100, 100)
  vertex(-100, 100)
  vertex(0, 50)
  endShape(CLOSE)
  angle += 0.05
```

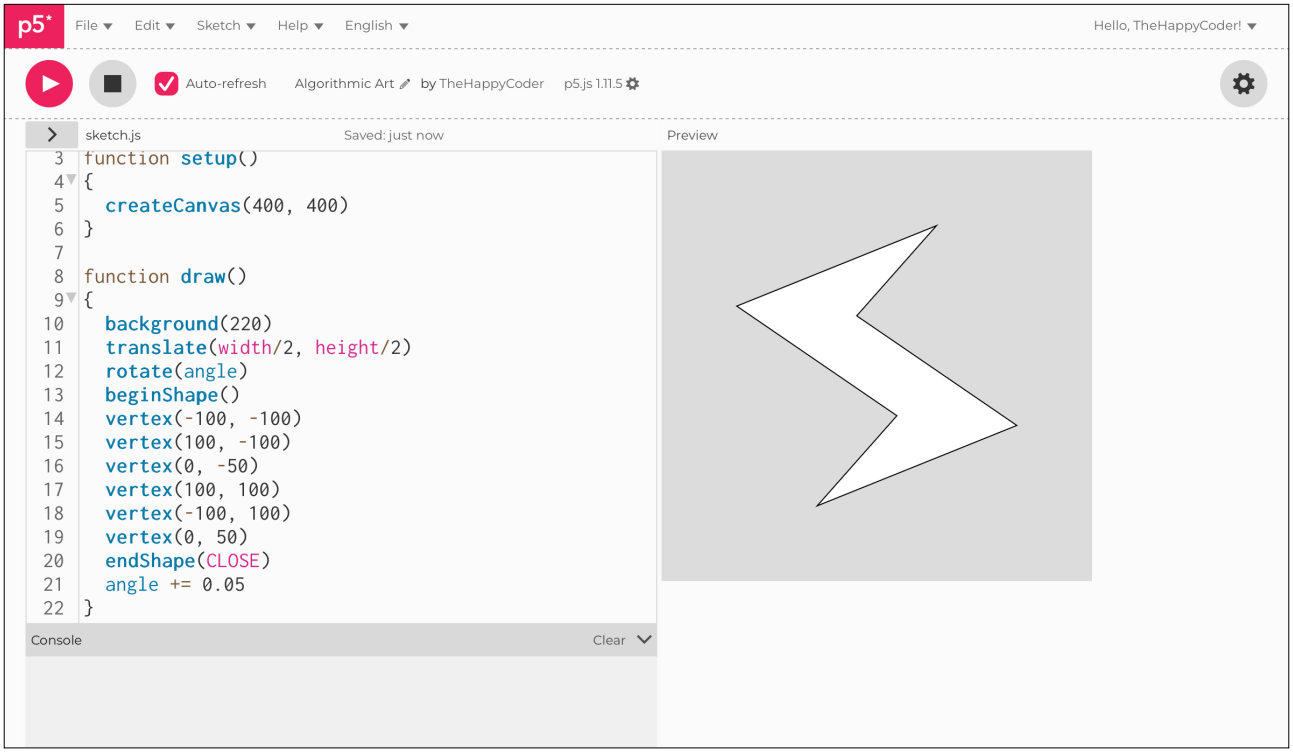
🗒️ Notes

It rotates about the centre of the canvas; also, we are using radians for the angle.

🌻 Challenge

Change to degrees.

Figure B8.6


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
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
translate(width/2, height/2)
rotate(angle)
beginShape()
vertex(-100, -100)
vertex(100, -100)
vertex(0, -50)
vertex(100, 100)
vertex(-100, 100)
vertex(0, 50)
endShape(CLOSE)
angle += 0.05
}
Console
Clear

Sketch B8.7 many sided shape

❗ starting a new sketch

Using a `for()` loop and the `x` and `y` coordinates for a circle, we can draw any-sided shape we want. Here, we are drawing a `10-sided` shape.

```
let radius = 150
let sides = 10
let x = 0
let y = 0

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

function draw()
{
  background(220)
  translate(width/2, height/2)
  beginShape()
  for (let angle = 0; angle < 360; angle += 360/sides)
  {
    x = radius * sin(angle)
    y = radius * cos(angle)
    vertex(x, y)
  }
  endShape(CLOSE)
}
```

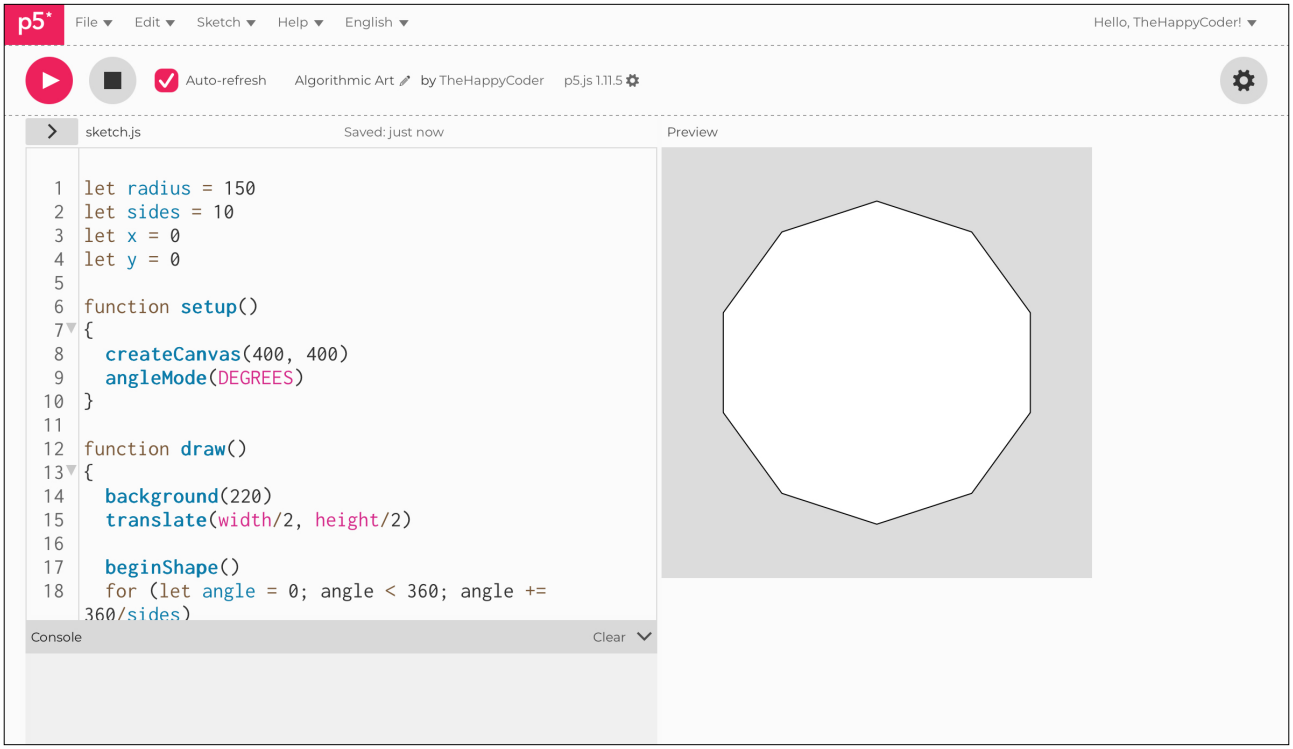
🗒️ Notes

We specify the number of sides and divide `360` by the number of sides we want.

🌻 Challenges

1. Try other numbers of sides.
2. What happens if you put in 4.5 rather than a whole number?
3. Have a slider to change the number of sides.

Figure B8.7


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 radius = 150
let sides = 10
let x = 0
let y = 0
function setup()
{
createCanvas(400, 400)
angleMode(DEGREES)
}
function draw()
{
background(220)
translate(width/2, height/2)
beginShape()
for (let angle = 0; angle < 360; angle +=
360/sides)
Console
Clear

Sketch B8.8 a vertex doodle

❗ new sketch
A bit of a doodle.

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

function draw()
{
  background(255)
  beginShape()
  for (let y = 50; y < 350; y += 5)
  {
    strokeWeight(random(2))
    for (let x = 50; x < 355; x += 5)
    {
      vertex(x, y)
      y += random(-2, 2)
      fill(random(50), 10)
      endShape()
    }
    beginShape()
  }
  endShape()
  noLoop()
}
```

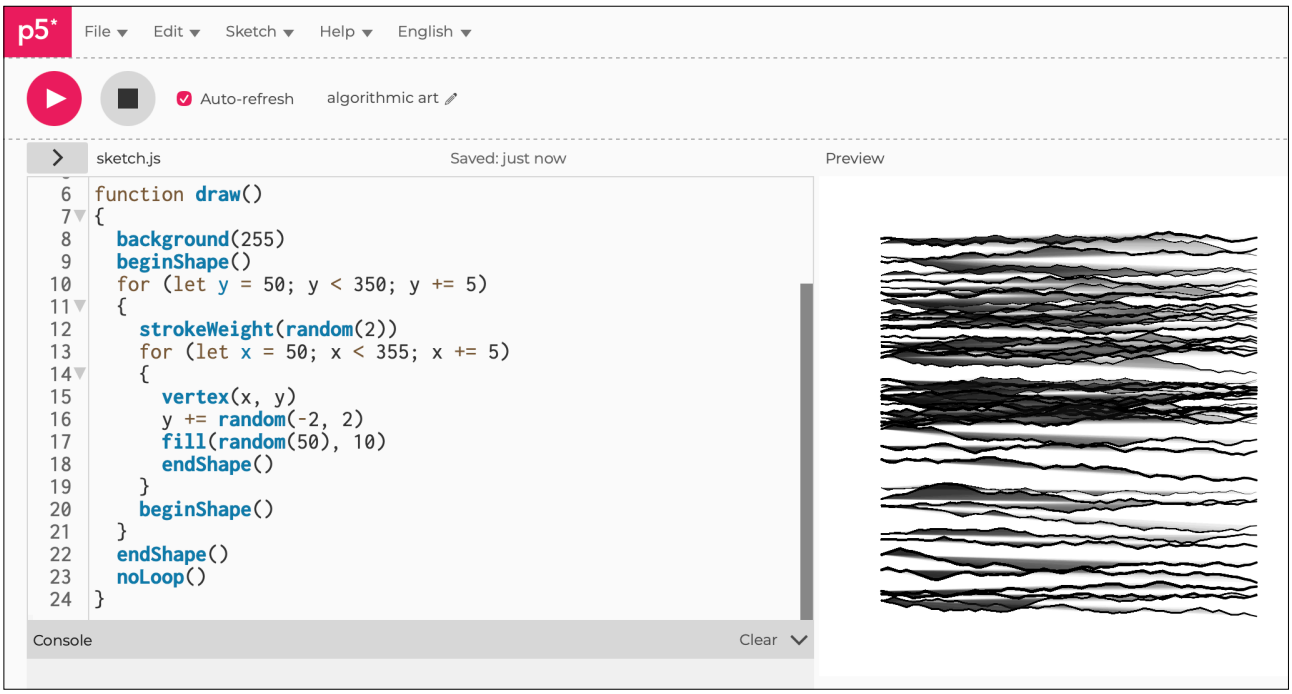
🗒️ Notes

Just using some of the lessons we have learned. This is a matter of playing and experimenting.

Challenges

1. Play with the values, try colours; HSB might be nice.
2. Animated seascape?

Figure B8.8



Bezier curves

Not only can we draw straight lines with a `vertex()` function, but we can also do curved lines with bezier. The `bezier()` function has four arguments. The first and last are the fixed end points, think `line()`. The middle two affect the line by drawing it towards those points. They don't join the line but pull it.

There is maths behind all of this, but at this stage, it is not necessary to understand it, just use it. If you want to know more, just search for it.

Sketch B8.9 a Bézier curve

❗ starting a brand new sketch.

We fix four points (pairs of co-ordinates) that serve as the arguments for the `bezier()` curve function.

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

function draw()
{
  background(220)
  noFill()
  strokeWeight(3)
  stroke('blue')
  let x1 = 50
  let y1 = 50
  let x2 = 50
  let y2 = 350
  let x3 = 350
  let y3 = 350
  let x4 = 350
  let y4 = 50
  bezier(x1, y1, x2, y2, x3, y3, x4, y4)
  strokeWeight(10)
  stroke('darkred')
  point(x1, y1)
  point(x2, y2)
  point(x3, y3)
  point(x4, y4)
}
```

🗒️ Notes

The first and last are fixed, and the middle two cause the line to curve. I have coloured the points and the line so that you can see their positions and influence.

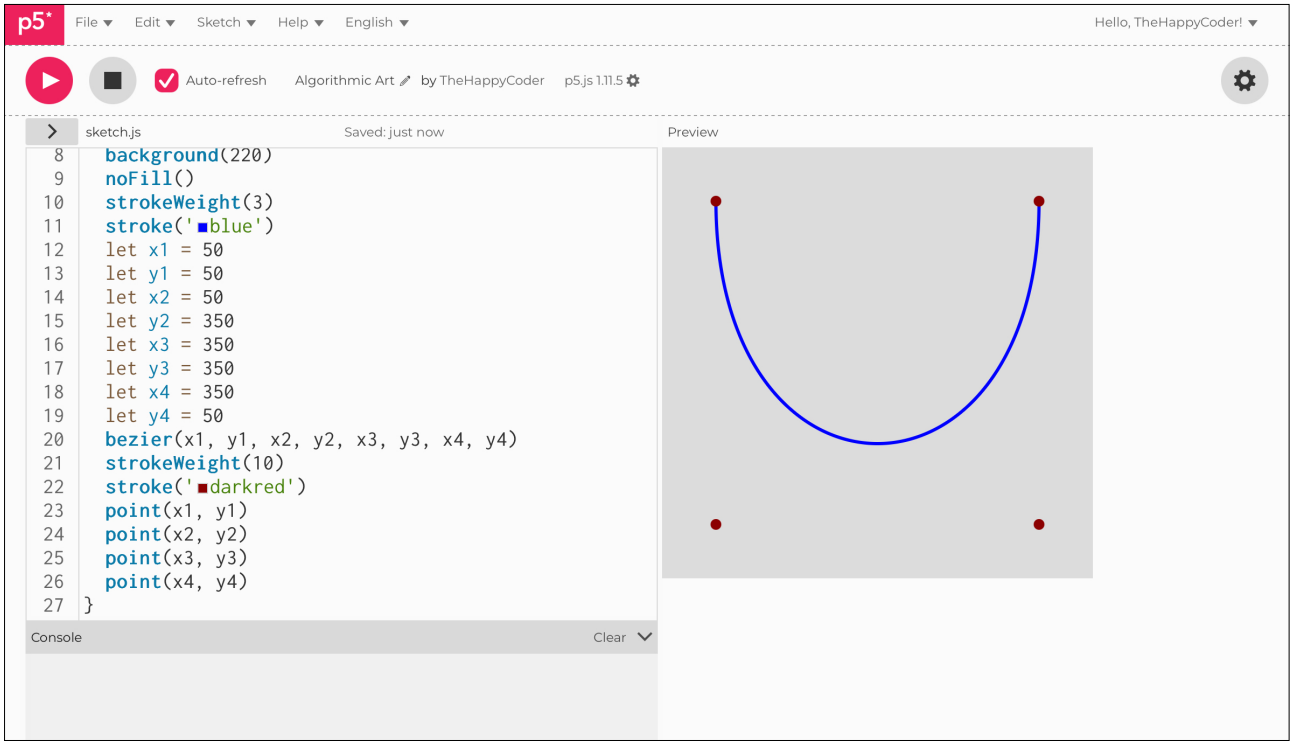
🌻 Challenge

Try different positions for the middle two points.

🛠️ Code Explanation

`bezier(x1, y1, x2, y2, x3, y3, x4, y4)`	The bezier() curve function with the four sets of vertices.

Figure B8.9



Sketch B8.10 the dancing mouse

As you move your mouse, it affects the curve of the line. We have also changed the last y co-ordinate.

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

function draw()
{
  background(220)
  noFill()
  strokeWeight(3)
  stroke('blue')
  let x1 = 50
  let y1 = 50
  let x2 = 50
  let y2 = 350
  let x3 = mouseX
  let y3 = mouseY
  let x4 = 350
  let y4 = 350
  bezier(x1, y1, x2, y2, x3, y3, x4, y4)
  strokeWeight(10)
  stroke('darkred')
  point(x1, y1)
  point(x2, y2)
  point(x3, y3)
  point(x4, y4)
}
```

🗒️ Notes

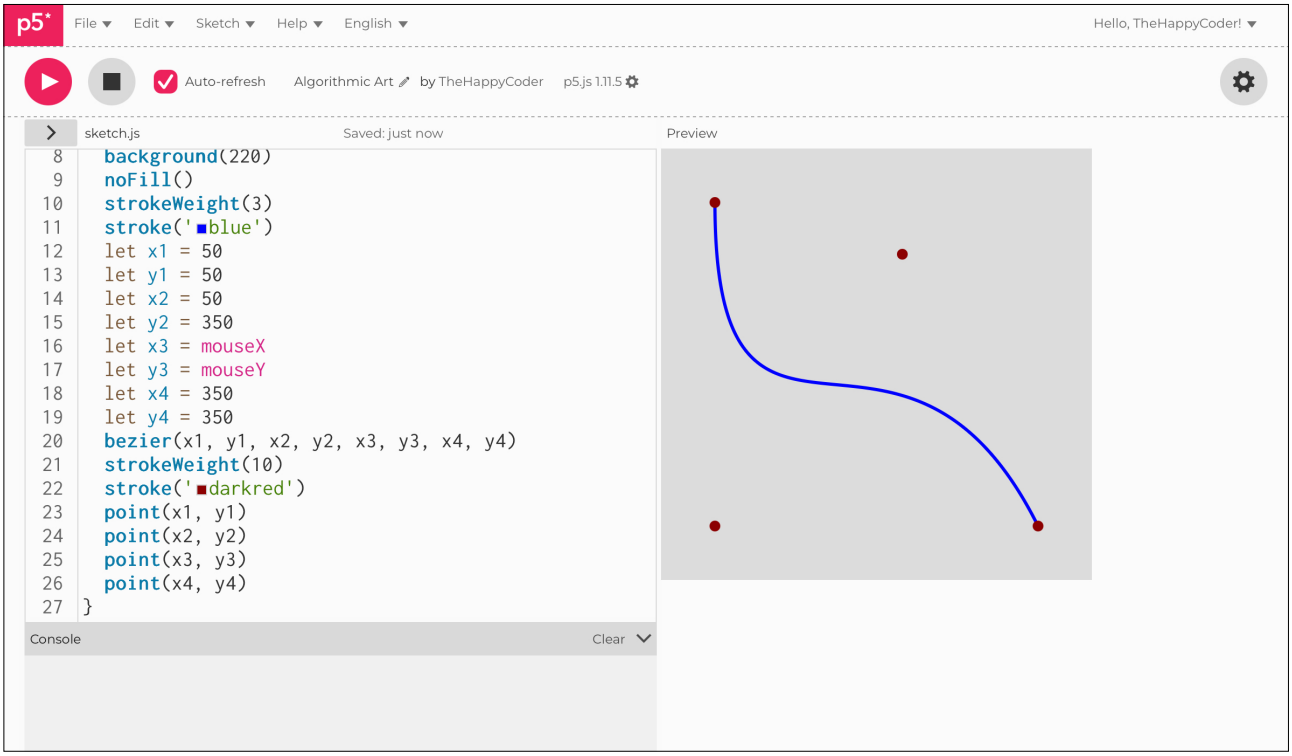
Gives you a sense of how one of the middle points may affect the curve.

🌻 Challenges

1. They'll see the other points and the difference.
2. Have both the middle points be controlled by the mouse.

3. You could code the values straight into the Bézier curve function.

Figure B8.10



Sketch B8.11 bezier doodle part 1

❗ new sketch.
We are going to create a simple pattern.

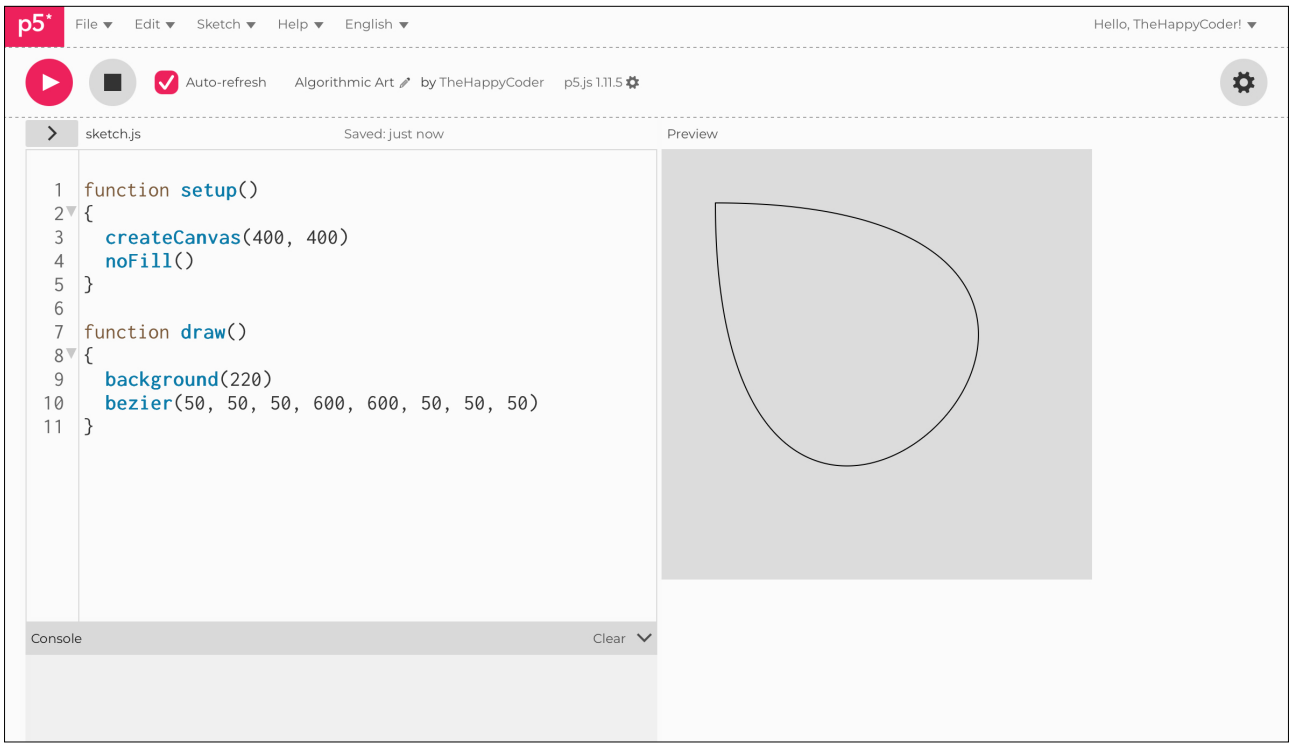
```
function setup()
{
  createCanvas(400, 400)
  noFill()
}

function draw()
{
  background(220)
  bezier(50, 50, 50, 600, 600, 50, 50, 50)
}
```

🗒️ Notes
We will code the coordinates straight into the function.

🌻 Challenge
Remove the `noFill()`.

Figure B8.11


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 noFill()
5 }
6
7 function draw()
8 {
9 background(220)
10 bezier(50, 50, 50, 600, 600, 50, 50, 50)
11 }
Console
Clear

Sketch B8.12 bezier doodle part 2

We create a `for()` loop that increments in steps of `10`; this is added to the `bezier()` function, which is inside the loop.

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

function draw()
{
  background(220)
  for(let i = 0; i < 100; i += 10)
  {
  bezier(50 + i, 50 + i, 50 + i, 600 + i, 600 + i, 50 + i, 50 + i, 50 + i)
  }
  noLoop()
}
```

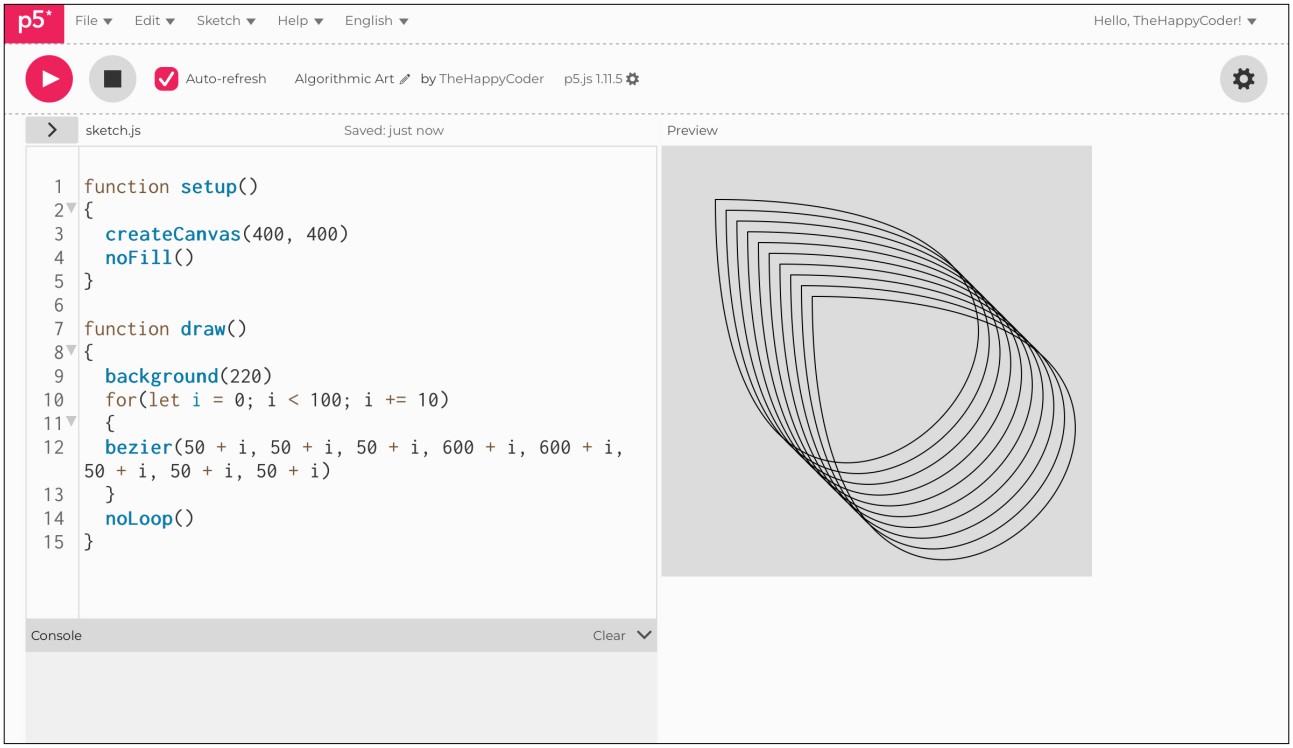
🗒️ Notes

Creates a repeated pattern; the `noLoop()` isn't entirely necessary.

🌻 Challenge

Try something else.

Figure B8.12


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
function setup()
{
createCanvas(400, 400)
noFill()
}
function draw()
{
background(220)
for(let i = 0; i < 100; i += 10)
{
bezier(50 + i, 50 + i, 50 + i, 600 + i, 600 + i, 50 + i, 50 + i, 50 + i)
}
noLoop()
}
Console
Clear

Sketch B8.13 another doodle with bezier

Changing this slightly, we can create the following pattern:

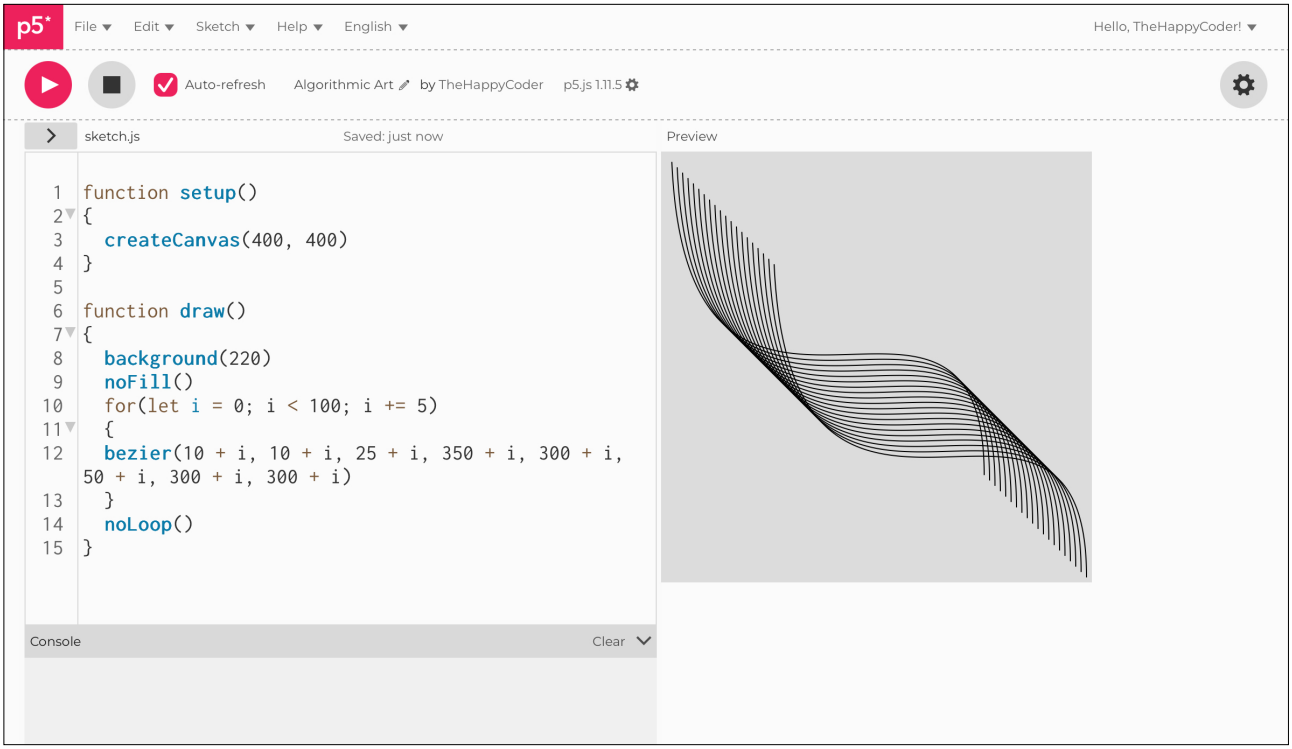
```
function setup()
{
  createCanvas(400, 400)
  noFill()
}

function draw()
{
  background(220)
  for (let i = 0; i < 100; i += 5)
  {
  bezier(10 + i, 10 + i, 25 + i, 350 + i, 300 + i, 50 + i, 300 + i, 300 + i)
  }
  noLoop()
}
```

Notes

Another nice pattern.

Figure B8.13


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 }
5
6 function draw()
7 {
8 background(220)
9 noFill()
10 for(let i = 0; i < 100; i += 5)
11 {
12 bezier(10 + i, 10 + i, 25 + i, 350 + i, 300 + i, 50 + i, 300 + i, 300 + i)
13 }
14 noLoop()
15 }
Console
Clear

Sketch B8.14 lovely jubbly

Another little doodle.

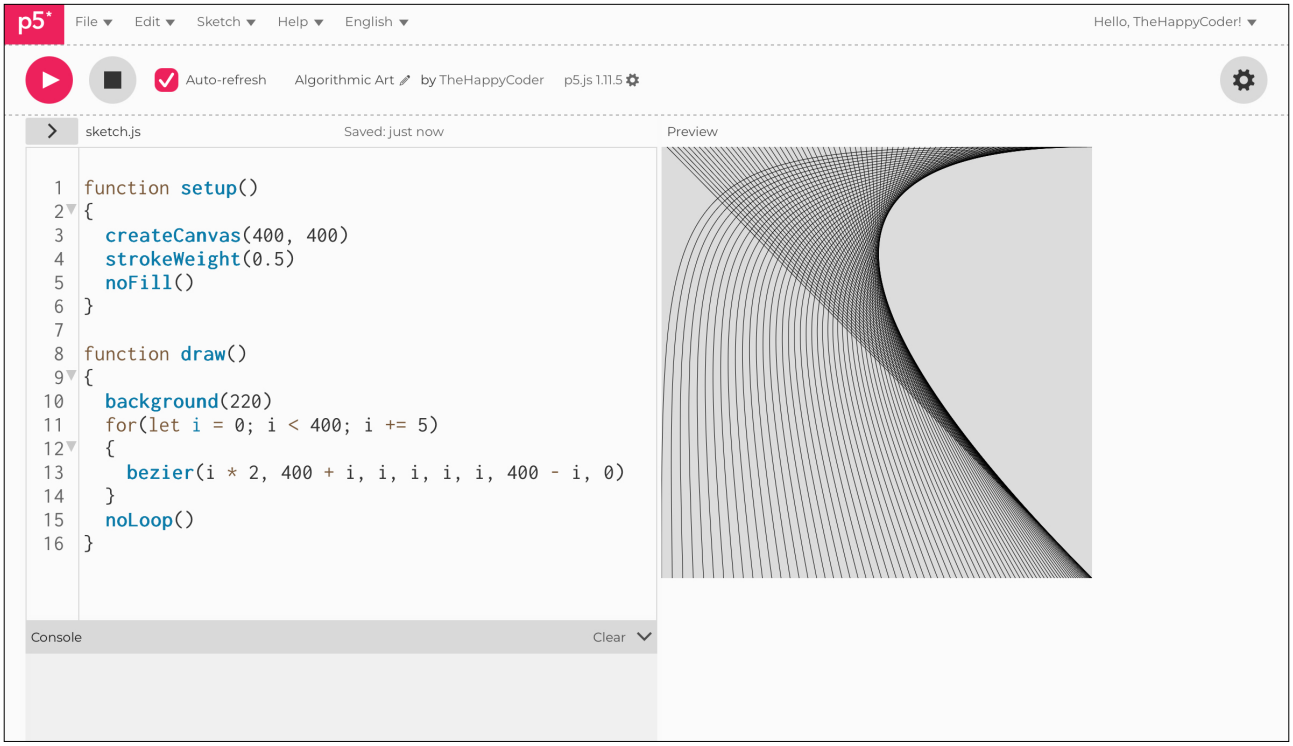
```
function setup()
{
  createCanvas(400, 400)
  strokeWeight(0.5)
  noFill()
}

function draw()
{
  background(220)
  for(let i = 0; i < 400; i += 5)
  {
    bezier(i * 2, 400 + i, i, i, i, i, 400 - i, 0)
  }
  noLoop()
}
```

Notes

Just have fun playing.

Figure B8.14


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
function setup()
{
createCanvas(400, 400)
strokeWeight(0.5)
noFill()
}
function draw()
{
background(220)
for(let i = 0; i < 400; i += 5)
{
bezier(i * 2, 400 + i, i, i, i, i, 400 - i, 0)
}
noLoop()
}
Console
Clear

Sketch B8.15 a bit of a splash

Another (final) Bezier doodle.

```
function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(3)
}

function draw()
{
  background(255)
  for(let i = 0; i < 100; i += 5)
  {
    let x1 = random(10, 390)
    let y1 = random(10, 390)

    let x2 = random(100, 200)
    let y2 = random(100, 200)

    let x3 = random(200, 300)
    let y3 = random(200, 300)

    let x4 = random(10, 390)
    let y4 = random(10, 390)
    stroke(random(255), 0, 0)
    bezier(x1, y1, x2, y2, x3, y3, x4, y4)
  }
  noLoop()
  rect(0, 0, width, height)
}
```

Notes

White background with a simple frame around it.

🌻 Challenge

Play with colours, alpha, strokeWeight(), etc.

Figure B8.15