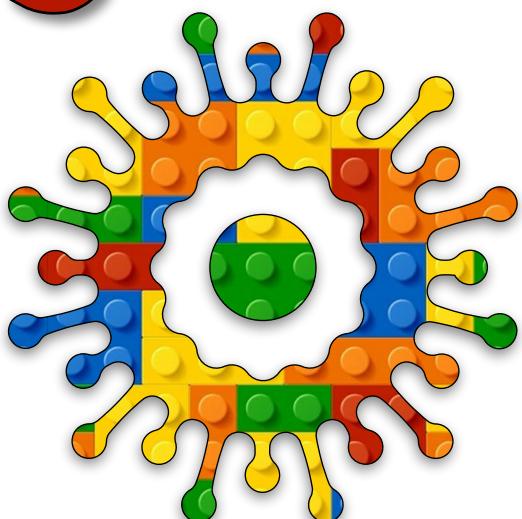


Creative Coding

Module B

Unit #9

arcs and mapping





Module B Unit #9 arcs

- Sketch B9.1 the 90° arc
- Sketch B9.2 the negative 90° arc
- Sketch B9.3 drawing a complete circle
- Sketch B9.4 random
- Sketch B9.5 accentuating the randomness
- Sketch B9.6 spiral
- Sketch B9.7 being a bit more OPEN
- Sketch B9.8 OPEN sesame
- Sketch B9.9 strikes a CHORD
- Sketch B9.10 a piece of the PIE
- Sketch B9.11 the making of a PAC-MAN
- Sketch B9.12 10PRINT arcs
- Sketch B9.13 mapping



Introduction to arcs and mapping

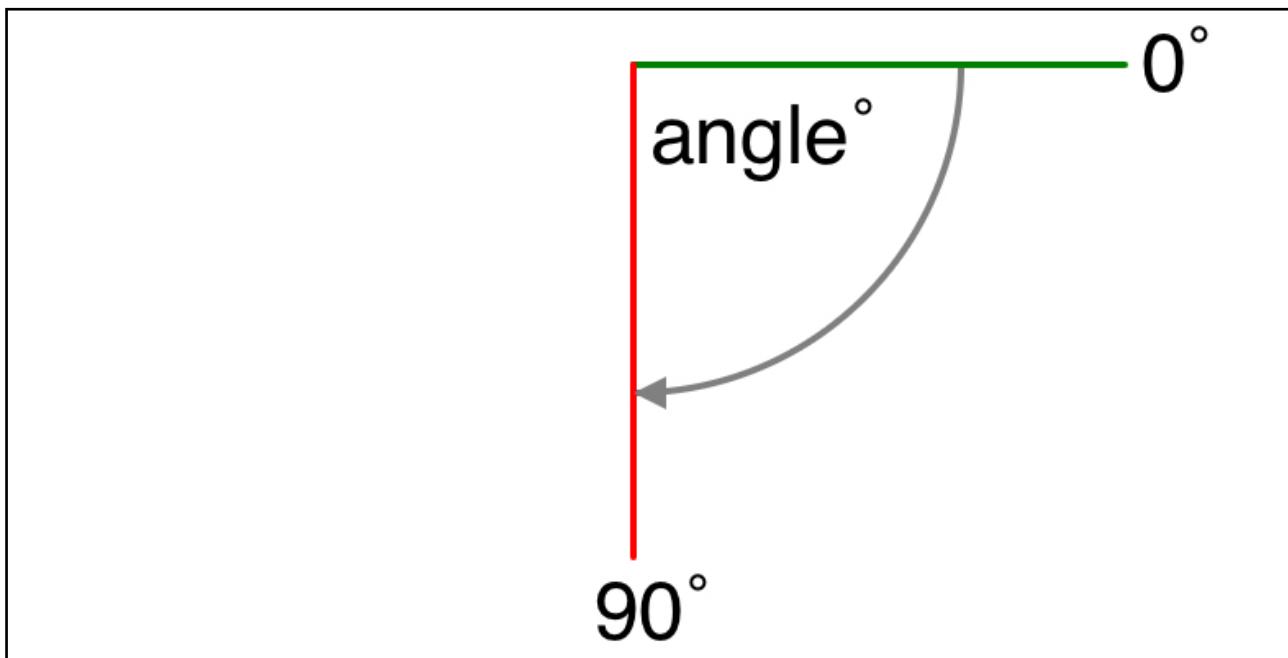
The `arc()` function has six arguments (three pairs), so if we consider this example:

`arc(100, 200, 150, 300, 0, 180)`

The first two arguments are the `x` and `y` coordinates (`x` is `100`, `y` is `200`). The second two are the dimensions of the ellipse (`150` wide, `300` tall). The third pair of arguments are the angles from and to (starts at 0° and stops at 180°).

The angle is described clockwise from the horizontal 0° , to the new angle 90° (see Fig.1). It is measured in `radians` by default; hence, we will be using the `angleMode()` to change to `degrees` to make it more intuitive.

Figure 1: arc angle orientation



Key concepts

- arcs
- `ellipseMode()`
- mapping



Sketch B9.1 the 90° arc

A simple 90° arc.

```
function setup()
{
    createCanvas(400, 400)
    background(220)
    angleMode(DEGREES)
    arc(200, 200, 200, 200, 0, 90)
}
```

Notes

Because the width and height are the same, it scribes a circle, or a quarter of one in this case.

Challenge

Try other angles, starting and stopping.

Code Explanation

arc(200, 200, 200, 200, 0, 90)	Arc drawn at position (200, 200) with a width and height of 200, starting at 0° and finishing at 90°
--------------------------------	--

Figure B9.1

The screenshot shows the p5.js web editor interface. At the top, there's a red header bar with the p5 logo, followed by a menu bar with File, Edit, Sketch, Help, and English. On the right side of the header, it says "Hello, TheHappyCoder!" with a dropdown arrow. Below the header is a toolbar with icons for play, stop, auto-refresh (which is checked), algorithmic art, and p5.js version 1.11.5.

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

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     arc(200, 200, 200, 200, 0, 90)
7 }
```

The "Preview" tab shows a gray canvas with a white square at the center. A black quarter-circle arc is drawn starting from the bottom-left corner of the square and extending upwards and to the right, ending at a 90-degree angle.

At the bottom of the editor, there's a "Console" tab and a "Clear" button.



Sketch B9.2 the negative 90° arc

We can start anywhere; here we start at -90° and arc round to 0° .

```
function setup()
{
    createCanvas(400, 400)
    background(220)
    angleMode(DEGREES)
    arc(200, 200, 200, 200, -90, 0)
}
```

Notes

A different way of using the angles.

Challenge

Try other negative values.

Code Explanation

`arc(200, 200, 200, 200, -90, 0)`

Starts at -90° and moves round to 0° clockwise

Figure B9.2

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/pause, stop, auto-refresh (which is checked), algorithmic art by TheHappyCoder, and p5.js 1.11.5. The main workspace is divided into two sections: 'sketch.js' on the left and 'Preview' on the right. In 'sketch.js', the following code is written:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     arc(200, 200, 200, 200, -90, 0)
7 }
```

The 'Preview' section shows a gray square canvas with a white quarter-circle arc starting from the bottom-left corner and extending towards the top-right.



Sketch B9.3 drawing a complete circle

Although there are a number of easy ways to draw a circle, we can do the same with arcs; here we increase the final angle by one in a `for()` loop.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
  angleMode(DEGREES)
  for (let i = 0; i < 360; i++)
  {
    arc(200, 200, 200, 200, i - 1, i)
  }
}
```

Notes

We start with `i - 1` because we want to draw the arc at each increment; otherwise, we get to the end of the loop and just draw the final arc from 0° to 360° .

Challenge

What happens if you start with 0° ?

Code Explanation

<code>for (let i = 0; i < 360; i++)</code>	A <code>for()</code> loop starting at 0° and finishing at 360° in increments of 1°
--	---

Figure B9.3

The screenshot shows the p5.js IDE interface. At the top, there's a red header bar with the 'p5*' logo, followed by menu items: File ▾, Edit ▾, Sketch ▾, Help ▾, English ▾, and a user greeting 'Hello, TheHappyCoder! ▾'. Below the header is a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and a gear for settings. The main workspace is divided into two sections: 'sketch.js' on the left and 'Preview' on the right. In the 'sketch.js' section, the code is as follows:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     for (let i = 0; i < 360; i++)
7     {
8         arc(200, 200, 200, 200, i - 1, i)
9     }
10 }
```

The 'Preview' section shows a gray square canvas containing a large, hollow white circle with concentric arcs drawn at regular intervals around its circumference. At the bottom of the interface, there's a 'Console' tab and a 'Clear ▾' button.



Sketch B9.4 random

We can now add some randomness to the circle so that each arc has a slightly different value.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
  angleMode(DEGREES)
  for (let i = 0; i < 360; i++)
  {
    arc(200, 200, 200 + random(-10, 10), 200 + random(-10, 10), i
- 1, i)
  }
}
```

Notes

Each diameter (width and height) will vary randomly between **-10** and **+10**.

Challenge

1. Add in a **noFill()**.
2. Try **i - 5** (or more). Why do you think you get that effect?

Figure B9.4

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!" with a dropdown arrow. Below the toolbar, the title "sketch.js" is shown next to a "Saved: just now" message. On the far right, there's a gear icon.

The main area has two tabs: "sketch.js" (which is active) and "Preview". The "sketch.js" tab contains the following code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     for (let i = 0; i < 360; i++)
7     {
8         arc(200, 200, 200 + random(-10, 10), 200 +
random(-10, 10), i - 1, i)
9     }
10 }
```

The "Preview" tab shows a circular pattern of concentric arcs centered at (200, 200). Each arc has a radius of 200 plus a random offset between -10 and 10 pixels. The segments of the arcs are slightly offset from each other, creating a textured, organic look.



Sketch B9.5 accentuating the randomness

This just pushes the boat out.

```
function setup()
{
    createCanvas(400, 400)
    background(220)
    angleMode(DEGREES)
    noFill()
    for (let i = 0; i < 360; i++)
    {
        strokeWeight(2)
        arc(200, 200, 250 + random(-30, 30), 250 + random(-30, 30), i
- 15, i)
    }
}
```

Notes

Added **noFill()** and **strokeWeight()**.

Challenge

Add some colour and play with the values.

Figure B9.5

The screenshot shows the p5.js IDE 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 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     background(220)
5     angleMode(DEGREES)
6     noFill()
7     for (let i = 0; i < 360; i++)
8     {
9         strokeWeight(2)
10        arc(200, 200, 250 + random(-30, 30), 250 +
11            random(-30, 30), i - 15, i)
12    }
}
```

The preview window shows a light gray canvas with a black concentric circle pattern. The inner circle is composed of many small, overlapping arcs, creating a textured appearance.



Sketch B9.6 spiral

We can make it do a spiral.

```
function setup()
{
    createCanvas(400, 400)
    background(220)
    angleMode(DEGREES)
    noFill()
    for (let i = 0; i < 810; i++)
    {
        strokeWeight(2)
        arc(180, 180, i / 2, i / 2, i - 1, i)
    }
}
```

Notes

Moved the centre of the spiral so it fits on the canvas.

Figure B9.6

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" 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 title bar reads "sketch.js" and "Saved: just now". To the right of the title bar is a "Preview" button.

The main area has two panes. The left pane contains the following JavaScript code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     noFill()
7     for (let i = 0; i < 810; i++)
8     {
9         strokeWeight(2)
10        arc(180, 180, i / 2, i / 2, i - 1, i)
11    }
12 }
```

The right pane shows a gray canvas with a black spiral line drawn on it, starting from the center and expanding outwards.

At the bottom, there's a "Console" tab and a "Clear" dropdown menu.



Sketch B9.7 being a bit more OPEN

! start with a new sketch

There are a number of attributes that you can use that change the appearance of an arc. The first one we will look at is called **OPEN**.

```
function setup()
{
    createCanvas(400, 400)
    background(220)
    angleMode(DEGREES)
    arc(200, 200, 200, 200, 0, 270)
}
```



Notes

This is an arc through to 270° .

Figure B9.7

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 main sections: a code editor, a preview canvas, and a console.

Code Editor: The file is named 'sketch.js'. It contains the following code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     arc(200, 200, 200, 200, 0, 270)
7 }
```

Preview: The preview window shows a gray square canvas with a white circle centered at (200, 200) with a radius of 200 pixels. A white arc is drawn from the bottom-left quadrant (0 degrees) to the top-right quadrant (270 degrees).

Console: The console is empty, with a 'Clear ▾' button above it.



Sketch B9.8 OPEN sesame

Adding the word **OPEN** at the end of the function (**7th** argument).

```
function setup()
{
    createCanvas(400, 400)
    background(220)
    angleMode(DEGREES)
    arc(200, 200, 200, 200, 0, 270, OPEN)
}
```

Notes

It has a dramatic effect.

Challenge

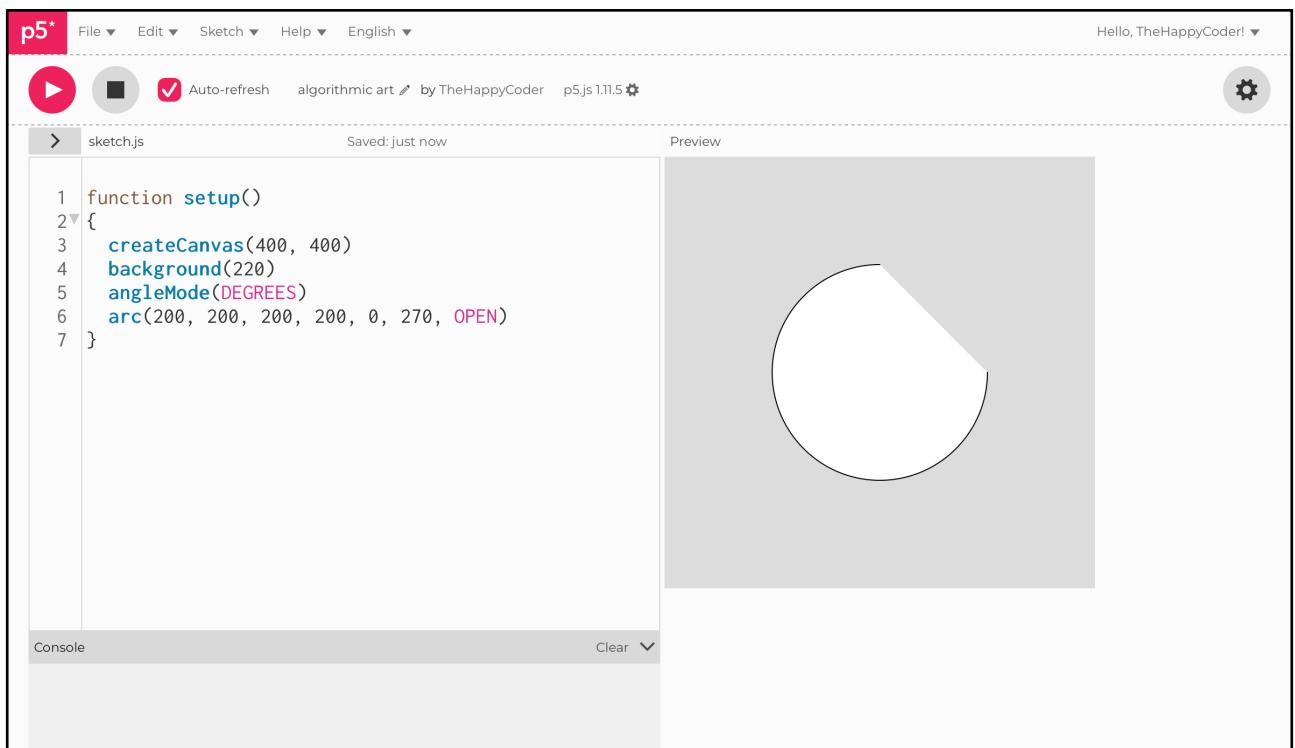
Try it with other angles.

Code Explanation

`arc(200, 200, 200, 200, 0, 270,
OPEN)`

Adding the attribute **OPEN** as the
seventh argument

Figure B9.8



The screenshot shows the p5.js IDE interface. At the top, there's a red header bar with the p5 logo and navigation links: File ▾, Edit ▾, Sketch ▾, Help ▾, English ▾, and a user greeting "Hello, TheHappyCoder! ▾". Below the header is a toolbar with icons for play/pause, stop, auto-refresh (which is checked), algorithmic art by TheHappyCoder, and p5.js 1.11.5. The main workspace is divided into two sections: "sketch.js" on the left and "Preview" on the right. In the sketch.js editor, the following code is written:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     arc(200, 200, 200, 200, 0, 270, OPEN)
7 }
```

The Preview window shows a gray square canvas with a white circle drawn on it. The circle is positioned such that its bottom-right quadrant is cut off, creating a wedge-like shape. The rest of the circle is white against the gray background. At the bottom of the interface, there's a "Console" tab and a "Clear" button.



Sketch B9.9 strikes a CHORD

We will try a different attribute, the **CHORD**.

```
function setup()
{
    createCanvas(400, 400)
    background(220)
    angleMode(DEGREES)
    arc(200, 200, 200, 200, 0, 270, CHORD)
}
```

Notes

It has the same effect but also draws the outline.

Code Explanation

arc(200, 200, 200, 200, 0, 270, CHORD)	Adding the attribute CHORD as the seventh argument
---	---

Figure B9.9

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" and "algorithmic art by TheHappyCoder". The version "p5.js 1.11.5" is also visible. On the right, a user profile says "Hello, TheHappyCoder!". Below the toolbar, the title "sketch.js" is shown along with the message "Saved: just now". To the right of the title is a "Preview" button. The main area contains the following code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     arc(200, 200, 200, 200, 0, 270, CHORD)
7 }
```

The preview window shows a gray square canvas with a white circle drawn on it. A portion of the circle from the 0 to 270 degree range is filled with white, while the rest of the circle and the circumference are black. The bottom left corner of the interface has a "Console" tab and a "Clear" button.



Sketch B9.10 a piece of the PIE

Another attribute, this time **PIE**.

```
function setup()
{
    createCanvas(400, 400)
    background(220)
    angleMode(DEGREES)
    arc(200, 200, 200, 200, 0, 270, PIE)
}
```



Notes

More like the original but with an outline.

Figure B9.10

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 message 'Hello, TheHappyCoder! ▾' with a gear icon. On the left, there's a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and a link to 'algorithmic art' by TheHappyCoder. Below the toolbar, the file name 'sketch.js' is shown along with a note 'Saved: just now'. The code editor contains the following JavaScript code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4     background(220)
5     angleMode(DEGREES)
6     arc(200, 200, 200, 200, 0, 270, PIE)
7 }
```

To the right of the code editor is a preview window titled 'Preview' which displays a pie chart. The chart has a radius of 200 pixels centered at (200, 200). It consists of three segments: a large white segment from 0 to 270 degrees, a small black segment from 270 to 360 degrees, and another large white segment from 360 back to 0 degrees. Below the preview is a 'Console' section with a 'Clear' button.



Sketch B9.11 the making of a PAC-MAN

I will leave you to work out the logic, just a bit of fun; art can be anything.

```
let bite = 10
let angleA
let angleB

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

function draw()
{
    background('darkblue')
    fill('yellow')
    angleA = bite * sin(millis()/2) + bite + 0.01
    angleB = -angleA
    arc(200, 200, 200, 200, angleA, angleB, PIE)
    fill('darkblue')
    circle(225, 150, 25)
}
```

Notes

The sine of an angle fluctuates between **-1** and **+1**, that is why we multiply by the bit and add it; it doubles and then cancels itself out. We have a small addition (**0.01**) so that it doesn't close all the way to **zero**.

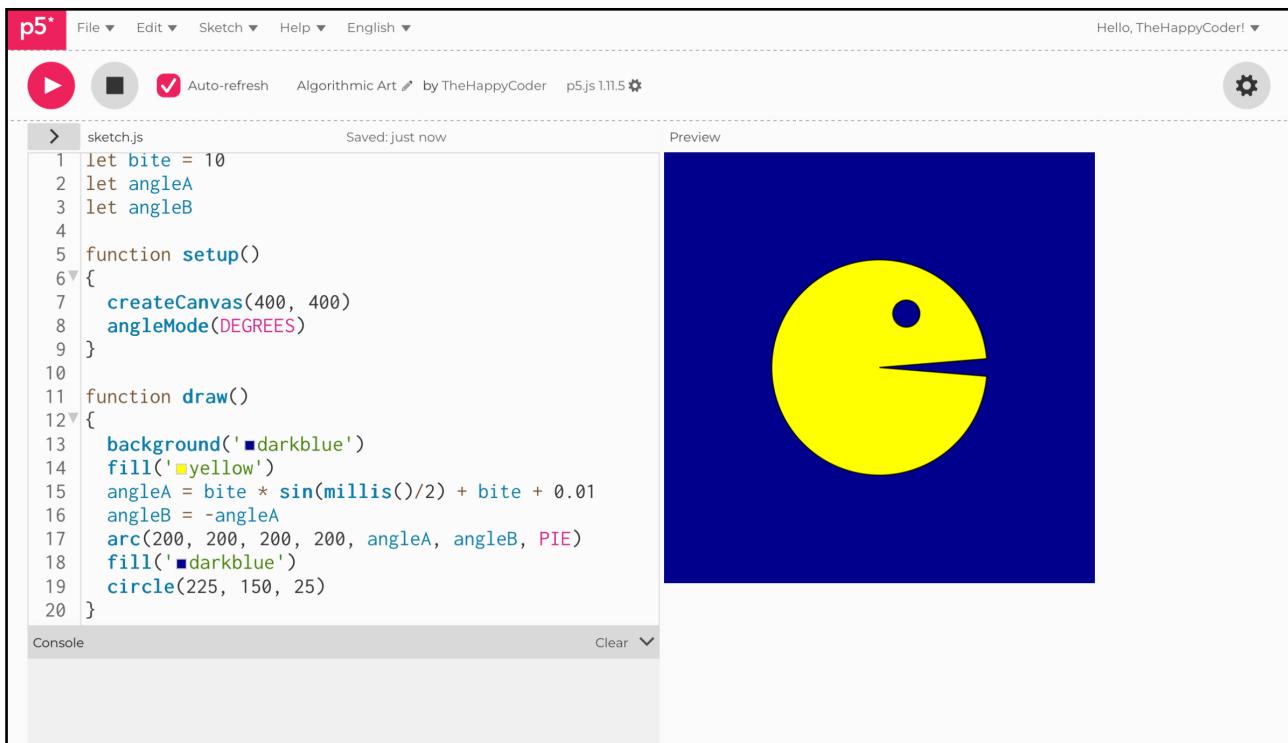
Challenge

You could use `frameCount` which also adds up very quickly but you may need to multiply by `15` or more/less

Code Explanation

millis()	Counts the number of milliseconds since the sketch started running
----------	--

Figure B9.11



The screenshot shows the p5.js code editor interface. The top bar includes the p5 logo, file navigation (File, Edit, Sketch, Help, English), and a user profile (Hello, TheHappyCoder!). The main area has tabs for 'sketch.js' and 'Preview'. The code in 'sketch.js' is as follows:

```
sketch.js
Saved: just now
1 let bite = 10
2 let angleA
3 let angleB
4
5 function setup()
6 {
7   createCanvas(400, 400)
8   angleMode(DEGREES)
9 }
10
11 function draw()
12 {
13   background('darkblue')
14   fill('yellow')
15   angleA = bite * sin(millis()/2) + bite + 0.01
16   angleB = -angleA
17   arc(200, 200, 200, 200, angleA, angleB, PIE)
18   fill('darkblue')
19   circle(225, 150, 25)
20 }
```

The 'Preview' window shows a yellow circle with a bite taken out of its right side, set against a dark blue background.



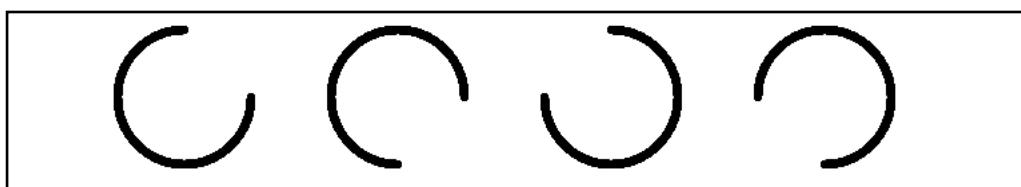
Sketch B9.12 10PRINT arcs

These are our four arcs.

Arc 1: `arc(x, y, spacing, spacing, 0, 270)`
Arc 2: `arc(x, y, spacing, spacing, 90, 0)`
Arc 3: `arc(x, y, spacing, spacing, 270, 180)`
Arc 4: `arc(x, y, spacing, spacing, 180, 90)`

They start and finish in different places.

Figure 2: our four arcs



We are going to randomly select them, where a quarter of the time any single one is selected. As a basis, we will use the **10PRINT** code from a previous unit, which I have adapted slightly (rather than going through the whole thing again).

We will create a random number (between **0** and **1**), then divide the probability into four equal parts:

- 中 less than **0.25**
- 中 between **0.25** and **0.5**
- 中 between **0.5** and **0.75**
- 中 more than **0.75**

I have highlighted the main elements that are very different.

```
let x = 0
let y = 0
let spacing = 25

function setup()
{
    createCanvas(400, 400)
    ellipseMode(CORNER)
    angleMode(DEGREES)
    background(220)
    noFill()
    x = spacing
    y = spacing
}

function draw()
{
    if (random(1) <= 0.25)
    {
        arc(x, y, spacing, spacing, 0, 270)
    }
    else if (random(1) >= 0.25 && random(1) <= 0.5)
    {
        arc(x, y, spacing, spacing, 90, 0)
    }
    else if (random(1) >= 0.5 && random(1) <= 0.75)
    {
        arc(x, y, spacing, spacing, 270, 180)
    }
}
```

```

{
    arc(x, y, spacing, spacing, 180, 90)
}
x += spacing
if (x >= width - spacing)
{
    x = spacing
    y += spacing
}
if (y >= height - spacing)
{
    noLoop()
}
}

```

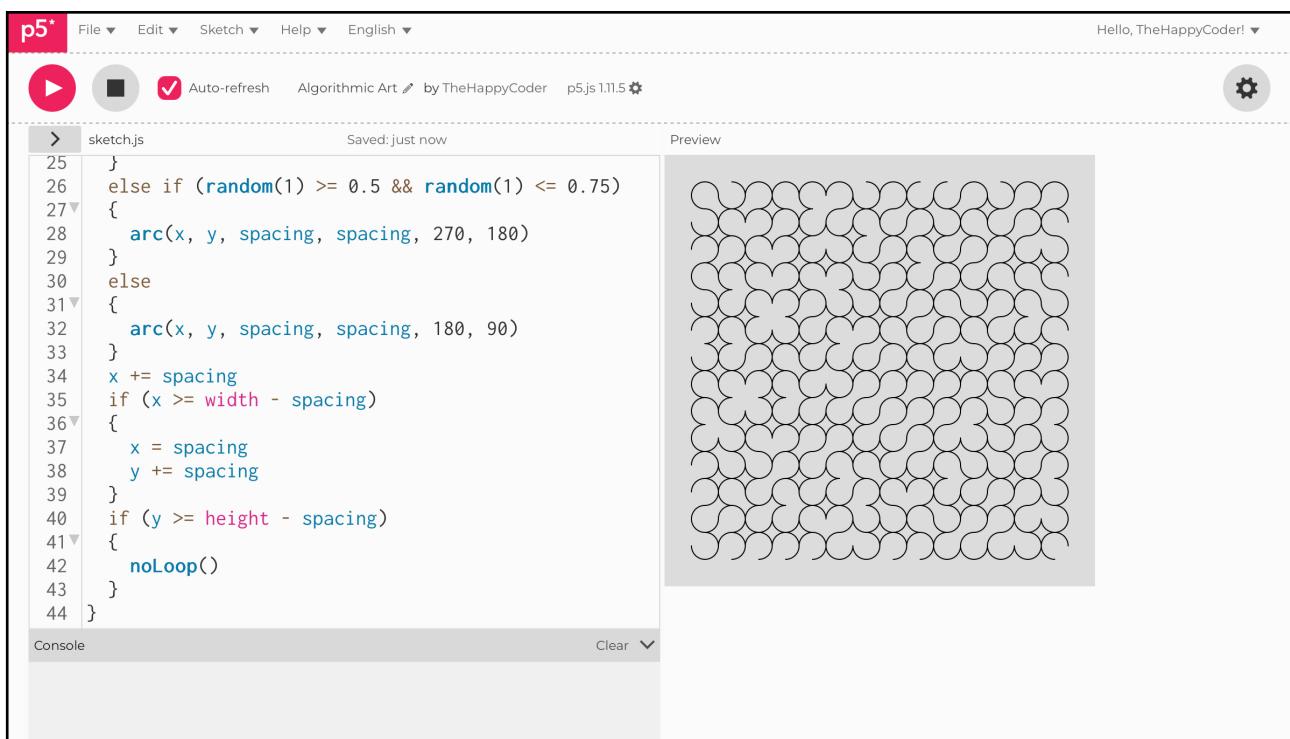
Notes

The **ellipseMode()** function allows you to move the centre of the circle (in terms of coordinates).

Code Explanation

ellipseMode(CORNER)	Puts the coordinates of the circle top left hand rather than in the centre
if (random(1) <= 0.25)	If less than 0.25 draw first arc
else if (random(1) >= 0.25 && random(1) <= 0.5)	If between than 0.25 and 0.5 draw second arc
else if (random(1) >= 0.5 && random(1) <= 0.75)	If between than 0.5 and 0.75 draw third arc
else	If none of the above then draw the fourth arc

Figure B9.12



The screenshot shows the p5.js code editor interface. At the top, there are buttons for play/pause, stop, auto-refresh (which is checked), and settings. The title bar says "sketch.js" and "Saved: just now". The right side of the interface is labeled "Preview" and shows a grid pattern of overlapping circles, which is the output of the p5.js code in the left panel. The code itself is a script named "sketch.js" containing the following pseudocode:

```
25 }  
26 else if (random(1) >= 0.5 && random(1) <= 0.75)  
27 {  
28   arc(x, y, spacing, spacing, 270, 180)  
29 }  
30 else  
31 {  
32   arc(x, y, spacing, spacing, 180, 90)  
33 }  
34 x += spacing  
35 if (x >= width - spacing)  
36 {  
37   x = spacing  
38   y += spacing  
39 }  
40 if (y >= height - spacing)  
41 {  
42   noLoop()  
43 }  
44 }
```

Below the code editor is a "Console" section which is currently empty. At the bottom right of the editor area is a "Clear" button.



Sketch B9.13 mapping

We can use a function called `map()` to map the horizontal (`mouseX`) position of the mouse across the canvas to the angle of the arc. The `map()` function takes five arguments.

The first one is the input variable you want to map, in this case it is the `mouseX` value which will range from `0` to `400`. The second two are start and end values for that input value (`0` and `400`), and the final two values are the values you want to map to, in this case the `angle` of the arc which we want in a range of `0` to `360`.

In effect, we are mapping `0` to `400` onto `0` to `360`.

```
let angle = 0

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

function draw()
{
    background(220)
    fill(255)
    angle = map(mouseX, 0, width, 0, 360)
    if (angle > 360)
    {
        angle = 360
    }
    if (angle < 0)
    {
        angle = 0
    }
}
```

```
}

arc(200, 200, 200, 200, 0, angle, PIE)
fill(0)
textSize(32)
text(floor(angle) + '°', 100, 100)

}
```

Notes

We have also put limitations on how far to the left and right of the canvas we can go; otherwise, you will have values bigger than 360° and negative numbers.

Code Explanation

angle = map(mouseX, 0, width, 0, 360)	Mapping the mouse position (0-400) onto the angle (0-360)
---------------------------------------	---

Figure B9.13

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 "Algorithmic Art" and "by TheHappyCoder". The version "p5.js 1.11.5" is also visible. On the right side, there's a user profile "Hello, TheHappyCoder!" and a settings gear icon.

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

```
8 }
9
10 function draw()
11 {
12   background(220)
13   fill(255)
14   angle = map(mouseX, 0, width, 0, 360)
15   if (angle > 360)
16   {
17     angle = 360
18   }
19   if (angle < 0)
20   {
21     angle = 0
22   }
23   arc(200, 200, 200, 200, 0, angle, PIE)
24   fill(0)
25   textSize(32)
26   text(floor(angle) + '°', 100, 100)
27 }
```

The "Preview" tab shows a circular arc centered at (200, 200) with a radius of 200 pixels. The arc starts at 0 degrees and ends at 276 degrees, as indicated by the text "276°" displayed in the center of the arc.