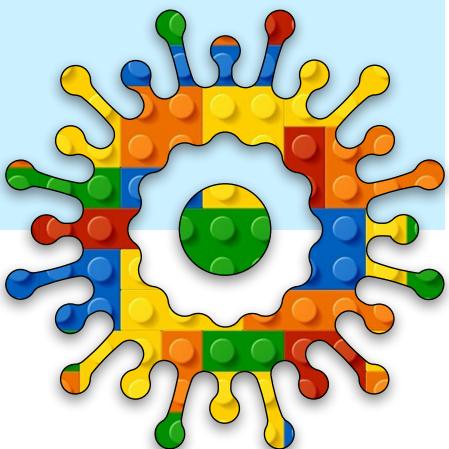


Physics Simulation

Module A

Unit #1

Oscillations





Contents

Module A Unit #1 Oscillation

Sketch A1.1	polar co-ordinates
Sketch A1.2	spiral
Sketch A1.3	vertex
Sketch A1.4	mapping the mouse
Sketch A1.5	SHM translated
Sketch A1.6	SHM breathe
Sketch A1.7	SHM period
Sketch A1.8	graphing sine wave
Sketch A1.9	an array of spheres
Sketch A1.10	a random value
Sketch A1.11	smaller and symmetrical
Sketch A1.12	amplitude, period and phase
Sketch A1.13	five waves
Sketch A1.14	adding waves
Sketch A1.15	using phase
Sketch A1.16	a few tweaks



Introduction to Unit #1 Oscillation

Creating a simulated spring and pendulum through exploring circular motion. It includes the introduction of polar co-ordinates, vectors, forces, etc., on a particle with a mass.



Sketch A1.1 polar co-ordinates

This is our starting sketch to illustrate polar to **Cartesian** co-ordinates - drawing the circle. We translate the co-ordinates to the centre of the canvas and use the angle (sine and cosine) to rotate round and draw a point at each **0.02** radians.

```
let x
let y
let angle = 0
let radius = 150

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

function draw()
{
    translate(200, 200)
    x = radius * cos(angle)
    y = radius * sin(angle)
    point(x, y)
    angle += 0.02
}
```

Notes

This draws a nice circle one point at a time.

Figure A1.1

The screenshot shows the p5.js web-based code editor interface. At the top, there's a navigation bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. On the right side of the header, it says 'Hello, TheHappyCoder!' and has buttons for 'Public' and 'p5.js 1.11.10'. Below the header, there are play/pause and refresh icons, and a checkbox for 'Auto-refresh' which is checked.

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

```
1 let x
2 let y
3 let angle = 0
4 let radius = 150
5
6 function setup()
7 {
8   createCanvas(400, 400)
9   background(220)
10  strokeWeight(5)
11 }
12
13 function draw()
14 {
15   translate(200, 200)
16   x = radius * cos(angle)
17   y = radius * sin(angle)
18   point(x, y)
19   angle += 0.02
20 }
```

The 'Preview' section shows a gray canvas with a black curved line forming a quarter-circle arc starting from the bottom-left and ending at the top-right.



Sketch A1.2 spiral

Following the same principle but this time creating a spiral by increasing the radius incrementally, starting with a slightly smaller **radius**.

```
let x  
let y  
let angle = 0  
let radius = 50  
  
function setup()  
{  
    createCanvas(400, 400)  
    background(220)  
    strokeWeight(5)  
}  
  
function draw()  
{  
    translate(200, 200)  
    x = radius * cos(angle)  
    y = radius * sin(angle)  
    point(x, y)  
    angle += 0.02  
    radius += 0.05  
}
```

Notes

We now have a nice spiral emerging.

Figure A1.2

The screenshot shows the p5.js code editor interface. At the top, there's a menu bar with File, Edit, Sketch, Help, and English. On the right side, it says "Hello, TheHappyCoder" and has Public and p5.js 1.11.10 buttons. Below the menu is a toolbar with a play button, a square button, Auto-refresh (which is checked), Physics, and a preview button. The main area is divided into two sections: a code editor on the left and a preview canvas on the right.

sketch.js

```
1 let x
2 let y
3 let angle = 0
4 let radius = 50
5
6 function setup()
7 {
8   createCanvas(400, 400)
9   background(220)
10  strokeWeight(5)
11 }
12
13 function draw()
14 {
15   translate(200, 200)
16   x = radius * cos(angle)
17   y = radius * sin(angle)
18   point(x, y)
19   angle += 0.02
20   radius += 0.05
21 }
```

The preview canvas shows a black helix curve drawn on a light gray background. The helix starts at the bottom center and spirals upwards towards the top center of the canvas.



Sketch A1.3 vertex

Drawing the circle by joining the points with the `vertex()` co-ordinates (changing the radius back to `150`, removing the angle, and moving the background back into `draw()` function.

```
let x
let y
let radius = 150

function setup()
{
    createCanvas(400, 400)
    strokeWeight(5)
    noFill()
}

function draw()
{
    background(220)
    translate(200, 200)
    beginShape()
    for (let i = 0; i < TWO_PI; i += 0.01)
    {
        x = radius * cos(i)
        y = radius * sin(i)
        vertex(x, y)
    }
    endShape(CLOSE)
}
```

Notes

We have a circle but the points on the circle are joined by lines.

Figure A1.3

The screenshot shows the p5.js web-based code editor interface. At the top, there's a navigation bar with 'File', 'Edit', 'Sketch', 'Help', and language settings ('English'). On the right side of the header, it says 'Hello, TheHappyCoder!' and shows a 'Public' button, the version 'p5.js 1.11.10', and a gear icon for settings.

The main area has two panes. The left pane is labeled 'sketch.js' and contains the following p5.js code:

```
4
5 function setup()
6 {
7   createCanvas(400, 400)
8   strokeWeight(5)
9   noFill()
10 }
11
12 function draw()
13 {
14   background(220)
15   translate(200, 200)
16   beginShape()
17   for (let i = 0; i < TWO_PI; i += 0.01)
18   {
19     x = radius * cos(i)
20     y = radius * sin(i)
21     vertex(x, y)
22   }
23   endShape(CLOSE)
24 }
```

The right pane is labeled 'Preview' and shows a large, hollow circle with a thick black outline, centered on a light gray background.



Sketch A1.4 mapping the mouse

Mapping the **increment** to **mouseX**.

! There is a slight risk of an infinite loop if your mouse strays too far. Be warned, the programme will crash, so save it before going any further.

```
let x
let y
let radius = 150

function setup()
{
    createCanvas(400, 400)
    strokeWeight(5)
    noFill()
}

function draw()
{
    background(220)
    translate(200, 200)
    let increment = map(mouseX, 0, width, PI, PI/8)
    beginShape()
    for (let i = 0; i < TWO_PI; i += increment)
    {
        x = radius * cos(i)
        y = radius * sin(i)
        vertex(x, y)
    }
    endShape(CLOSE)
}
```



Notes

As you move your mouse across the canvas the number of increments increases (the mapping takes care of this). This increases the number of points (vertices). You will start with a flat line and move through to nearly a circle.

Figure A1.4

The screenshot shows the p5.js code editor 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!' and has a 'Public' button and 'p5.js 1.11.10' link. Below the header, there are buttons for play/pause, stop, auto-refresh (which is checked), and physics. The main area is divided into two sections: 'sketch.js' on the left and 'Preview' on the right.

sketch.js

```
6<?
7  createCanvas(400, 400)
8  strokeWeight(5)
9  noFill()
10 }
11
12 function draw()
13 {
14  background(220)
15  translate(200, 200)
16  let increment = map(mouseX, 0, width, PI,
17  PI/8)
18  beginShape()
19  {
20    for (let i = 0; i < TWO_PI; i += increment)
21    {
22      x = radius * cos(i)
23      y = radius * sin(i)
24      vertex(x, y)
25    }
26  endShape(CLOSE)
27 }
```

Preview

The preview shows a large, irregular polygonal shape drawn with a thick black stroke. The shape is roughly circular but has several sharp, angular protrusions and indentations, giving it a serrated appearance. A small dark gray circle is positioned at the bottom right of the canvas.



Sketch A1.5 SHM translated

! Start a new sketch.

We are going to explore simple harmonic motion (**SHM**). The basic circle sketch with the centre translated to the middle and with a radius **r** of **100**.

```
let r = 100

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

function draw()
{
    background(220)
    translate(200, 200)
    circle(0, 0, r * 2)
}
```

Notes

It's just a circle!

Figure A1.5

The screenshot shows the p5.js web editor 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!' and has icons for 'Public' and 'p5.js 1.11.10'. Below the header, there are buttons for play/pause, auto-refresh (with a checked checkbox), and physics. The main area is divided into two sections: 'sketch.js' on the left and 'Preview' on the right. In 'sketch.js', the following code is written:

```
let r = 100
function setup()
{
  createCanvas(400, 400)
}
function draw()
{
  background(220)
  translate(200, 200)
  circle(0, 0, r * 2)
}
```

In the 'Preview' section, a large white circle is drawn on a gray background at the center of the canvas.



Sketch A1.6 SHM breathe

Now we are going to let the circle **breathe** to the waveform of the sine wave. The sine of an angle returns a value between **-1** and **1**. To get round the problem of having a circle of radius between **-1** and **1**, we use the **map()** function so its radius is between **50** and **150**.

```
let r = 100
let angle = 0

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

function draw()
{
    background(220)
    translate(200, 200)
    r = map(sin(angle), -1, 1, 50, 150)
    circle(0, 0, r * 2)
    angle += 0.05
}
```

Notes

You should see it inflate and deflate.

Challenges

1. Try different angle increments.
2. Different mapping values.
3. Colour: **fill(r * 2, 200 - r, 200)**

Figure A1.6

The screenshot shows the p5.js web-based code editor interface. At the top, there's a navigation bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. On the right side of the header, it says 'Hello, TheHappyCoder!' and has icons for 'Public' and 'p5.js 1.11.10'. Below the header, there are three main sections: a code editor, a preview canvas, and a console.

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

```
let r = 100
let angle = 0

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

function draw()
{
  background(220)
  translate(200, 200)
  r = map(sin(angle), -1, 1, 50, 150)
  circle(0, 0, r * 2)
  angle += 0.05
}
```

Preview: The preview window shows a gray square containing a single white circle centered at approximately (200, 200) with a radius of about 100 pixels.

Console: There is a 'Console' section at the bottom left with a 'Clear' button.



Sketch A1.7 SHM period

Introducing a period, how long is the cycle of one sine wave? It is 360° or 2π .

The frame rate of your screen is around 60 frames per second; you can check this by including the line of code `console.log(frameRate())` in the `draw()` function. To do the maths so that we have 1 wave per second, we divide $360/60$, which will give us the amount of rotation per frame to make a couple of cycles in 1 second, which comes out at 6° . Phew! In the sketch, we will continue to use radians, hence $2\pi/60$ (`TWO_PI/60`).

```
let r = 100
let angle = 0

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

function draw()
{
    background(220)
    translate(200, 200)
    r = map(sin(angle), -1, 1, 50, 150)
    circle(0, 0, r * 2)
    let increment = TWO_PI/60
    angle += increment
}
```

Notes

This is just another way of simulating movement, we are simulating this digitally.

 Challenges

1. Change the line of code to: `let increment = TWO_PI/frameRate()`
2. Change the time period using `millis()`.
3. Add an angular velocity so it gets faster over time.
4. Try it in 3D.

Figure A1.7

The screenshot shows the p5.js web-based code editor interface. At the top, there's a navigation bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. On the right side of the header, it says 'Hello, TheHappyCoder!' and has a 'Public' button and a version 'p5.js 1.11.10'. Below the header, there are three main sections: a code editor on the left, a preview canvas in the center, and a console at the bottom.

Code Editor:

```
let r = 100
let angle = 0

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

function draw()
{
  background(220)
  translate(200, 200)
  r = map(sin(angle), -1, 1, 50, 150)
  circle(0, 0, r * 2)
  let increment = TWO_PI/frameRate()
  angle += increment
}
```

Preview: The preview canvas displays a single black circle centered at (200, 200) with a radius of approximately 100 pixels, rendered against a light gray background.

Console: The console at the bottom is currently empty, with a 'Clear' button next to it.



Sketch A1.8 graphing sine wave

! Start a new sketch for this bit.

Creating an oscillating sphere. A simple oscillating sphere attached to an elastic string.

```
let angle = 0
let r = 16

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

function draw()
{
    background(220)
    translate(200, 200)
    let y = map(sin(angle), -1, 1, -150, 150)
    line(0, 0, 0, y)
    circle(0, y, r * 2)
    angle += 0.04
}
```

Notes

A bit like an everlasting yo-yo.

Figure A1.8

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 side, it says "Hello, TheHappyCoder!" and has buttons for "Public" and "p5.js 1.11.10". Below the menu is a toolbar with icons for play, stop, and refresh, followed by "Auto-refresh" and "Physics". The main area is divided into two sections: "sketch.js" on the left and "Preview" on the right. In "sketch.js", the code is as follows:

```
1 let angle = 0
2 let r = 16
3
4 function setup()
5 {
6   createCanvas(400, 400)
7 }
8
9 function draw()
10{
11   background(220)
12   translate(200, 200)
13   let y = map(sin(angle), -1, 1, -150, 150)
14   line(0, 0, 0, y)
15   circle(0, y, r * 2)
16   angle += 0.04
17 }
```

In the "Preview" section, a single white circle is shown at the end of a vertical black line, representing a simple pendulum at its lowest point.



Sketch A1.9 an array of spheres

Adding in an array of spheres, adding lines to each one, and adjusting so it fits nicely on the canvas. The **floor()** function returns an integer.

```
let angles = []
let angleV = 0.04
let r = 16

function setup()
{
  createCanvas(400, 400)
  let total = floor(width / (r * 2))
  for (let i = 0; i < total; i++)
  {
    angles[i] = 0
  }
}

function draw()
{
  background(220)
  translate(200, 200)
  for (let i = 0; i < angles.length; i++)
  {
    let y = map(sin(angles[i]), -1, 1, -150, 150)
    let x = map(i, 0, angles.length, -180, 210)
    line(x, 0, x, y)
    circle(x, y, r * 2)
    angles[i] += angleV
  }
}
```



Notes

Lots of yo-yos, all moving in unison.

Figure A1.9

The screenshot shows the p5.js code editor interface. At the top, there's a menu bar with 'File', 'Edit', 'Sketch', 'Help', and 'English'. On the right, it says 'Hello, TheHappyCoder!' and has 'Public' and 'p5.js 1.11.0' buttons. Below the menu is a toolbar with icons for play, stop, and refresh, followed by 'Auto-refresh' and 'Physics' dropdowns. The main area is divided into two sections: 'sketch.js' on the left and 'Preview' on the right. In 'sketch.js', the code defines a variable 'r' and a function 'draw()'. The 'draw()' function sets the background to 220, translates to (200, 200), loops through an array of angles, and for each angle, calculates x and y coordinates using trigonometric functions (sin and map) and draws a circle at those coordinates. The 'angles' array is initialized in the setup() function. The 'Preview' section shows a vertical column of 10 circles, each connected by a thin vertical line to a common horizontal baseline.

```
let r = 50;
let total = floor(width / (r * 2));
let angles = [];

function setup() {
  createCanvas(400, 400);
  for (let i = 0; i < total; i++) {
    angles[i] = 0;
  }
}

function draw() {
  background(220);
  translate(200, 200);
  for (let i = 0; i < angles.length; i++) {
    let y = map(sin(angles[i]), -1, 1, -150, 150);
    let x = map(i, 0, angles.length, -180, 210);
    line(x, 0, x, y);
    circle(x, y, r * 2);
    angles[i] += angleV;
  }
}
```



Sketch A1.10 a random value

Make an array of angles set to a random value between **-0.1** and **0.1**.

```
let angles = []
let angleV = []
let r = 16

function setup()
{
    createCanvas(400, 400)
    let total = floor(width / (r * 2))
    for (let i = 0; i < total; i++)
    {
        angles[i] = 0
        angleV[i] = random(-0.1, 0.1)
    }
}

function draw()
{
    background(220)
    translate(200, 200)
    for (let i = 0; i < angles.length; i++)
    {
        let y = map(sin(angles[i]), -1, 1, -150, 150)
        let x = map(i, 0, angles.length, -180, 210)
        line(x, 0, x, y)
        circle(x, y, r * 2)
        angles[i] += angleV[i]
    }
}
```



Notes

Not so unison, randomly phased



Challenge

1. Change `angleV[i] = random(-0.1, 0.1)` to `angleV[i] = i / 100` and keep it for the next sketch.
2. Play around with the values.

Figure A1.10

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!" and has buttons for "Public" and "p5.js 1.11.10". Below the menu is a toolbar with icons for play, stop, auto-refresh (which is checked), physics, and preview.

The code editor window contains a file named "sketch.js" with the following content:

```
sketch.js
Saved: just now
1 for (let i = 0; i < total; i++)
2 {
3     angles[i] = 0
4     angleV[i] = random(-0.1, 0.1)
5 }
6
7 function draw()
8 {
9     background(220)
10    translate(200, 200)
11    for (let i = 0; i < angles.length; i++)
12    {
13        let y = map(sin(angles[i]), -1, 1, -150,
14 150)
14        let x = map(i, 0, angles.length, -180, 210)
15        line(x, 0, x, y)
16        circle(x, y, r * 2)
17        angles[i] += angleV[i]
18    }
19 }
```

The preview window on the right shows a simulation of several circular pendulums. Each pendulum consists of a small white circle at the end of a thin vertical line. The pendulums are arranged in a roughly triangular pattern, with some lines extending beyond the frame. The circles are white with black outlines.



Sketch A1.11 smaller and symmetrical

Removing the lines and making the spheres smaller **10**, adding vertex lines to join the circles. Remove: **line(x, 0, x, y)**

```
let angles = []
let angleV = []
let r = 10

function setup()
{
    createCanvas(400, 400)
    let total = floor(width / (r * 2))
    for (let i = 0; i < total; i++)
    {
        angles[i] = map(i, 0, total, 0, TWO_PI)
        angleV[i] = 0.04
    }
}

function draw()
{
    background(220)
    translate(200, 200)
    beginShape()
    for (let i = 0; i < angles.length; i++)
    {
        let y = map(sin(angles[i]), -1, 1, -150, 150)
        let x = map(i, 0, angles.length, -300, 300)
        fill(255)
        circle(x, y, r * 2)
        noFill()
    }
}
```

```
    vertex(x, y)
    angles[i] += angleV[i]
}
endShape()
```

Notes

We have a bit of a sine wave

Challenge

Replace `angleV[i] = 0.04` with `angleV[i] = random(0.04)`.

Figure A1.11

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, there are buttons for Play, Stop, Auto-refresh (which is checked), Physics, Public (which is also checked), and p5.js 1.11.10. The main area has tabs for sketch.js and Preview. The code in sketch.js is as follows:

```
13 }
14 }
15
16 function draw()
17 {
18   background(220)
19   translate(200, 200)
20   beginShape()
21   for (let i = 0; i < angles.length; i++)
22   {
23     let y = map(sin(angles[i]), -1, 1, -150, 150)
24     let x = map(i, 0, angles.length, -300, 300)
25     fill(255)
26     circle(x, y, r * 2)
27     noFill()
28     vertex(x, y)
29     angles[i] += angleV[i]
30   }
31   endShape()
32 }
```

The Preview window shows a series of white circles connected by thin black lines, forming a curved path that starts at the top left, dips down, and then curves back up towards the top right. The circles are evenly spaced along the curve.



Sketch A1.12 amplitude, period and phase

! Start a new sketch.

There are three parts to a wave: **amplitude** (height of the wave), **period** (length of one wave), and **phase** (offset). In this sketch, we are going to add waves together. First, using the final sketch from the last example, we will make it **object-orientated**.

```
let wave

function setup()
{
    createCanvas(400, 400)
    wave = new Wave(150, 400, 0)
}

function draw()
{
    background(220)
    for (let x = 0; x < width; x += 10)

    {
        let y = wave.calculate(x)
        circle(x, y + height/2, 10)
    }
}

class Wave
{
    constructor(amp, period, phase)
    {
        this.amplitude = amp
        this.period = period
    }
}
```

```
    this.phase = phase
}

calculate(x)
{
    return sin(this.phase + TWO_PI*x/this.period)*this.amplitude
}
}
```

Notes

More like a full sine wave

Challenge

Try different values for amplitude, period, and phase:

e.g. `wave = new Wave(100, 300, PI/3).`

Figure A1.12

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 side, it says "Hello, TheHappyCoder!" and has buttons for "Public" and "p5.js 1.11.10". Below the menu bar, there are icons for play, stop, and refresh, followed by "Auto-refresh" and "Physics". The main area is divided into two sections: "sketch.js" on the left and "Preview" on the right. In "sketch.js", the code defines a class "Wave" with methods "calculate" and "constructor". The "calculate" method uses the formula $\sin(\text{phase} + \text{PI} * \text{x} / \text{period}) * \text{amplitude}$. In the "Preview" section, a sine wave is visualized as a series of small circles connected by lines, forming a smooth curve.

```
sketch.js
Saved: just now
Preview

13> sketch.js
14  {
15    let y = wave.calculate(x)
16    circle(x, y + height/2, 10)
17  }
18
19 class Wave
20{
21  constructor(amp, period, phase)
22  {
23    this.amplitude = amp
24    this.period = period
25    this.phase = phase
26  }
27
28  calculate(x)
29  {
30    return sin(this.phase +
TWO_PI*x/this.period)*this.amplitude
31  }
32 }
```



Sketch A1.13 five waves

Here we will introduce five random waves, we create them in the `setup()` function, each with a different **amplitude**, **period**, and **phase**.

```
let waves = []

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 5; i++)
    {
        waves[i] = new Wave(random(20, 80), random(100, 400),
random(TWO_PI))
    }
}

function draw()
{
    background(220)
    for (let x = 0; x < width; x += 10)
    {
        for (let wave of waves)
        {
            let y = wave.calculate(x)
            circle(x, y + height/2, 10)
        }
    }
}

class Wave
{
    constructor(amp, period, phase)
```

```
{  
    this.amplitude = amp  
    this.period = period  
    this.phase = phase  
}  
  
calculate(x)  
{  
    return sin(this.phase + TWO_PI*x/this.period)*this.amplitude  
}  
}
```

Notes

This looks a bit messy, but wait.

Figure A1.13

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!" and "Public". Below the menu, there are buttons for play/pause, auto-refresh, and physics simulation. The code editor has a left panel for "sketch.js" and a right panel for "Preview". The code in "sketch.js" is as follows:

```
19 let y = wave.calculate(x)
20   circle(x, y + height/2, 10)
21 }
22 }
23 }
24
25 class Wave {
26 {
27   constructor(amp, period, phase) {
28     this.amplitude = amp
29     this.period = period
30     this.phase = phase
31   }
32
33   calculate(x) {
34     return sin(this.phase +
35 TWO_PI*x/this.period)*this.amplitude
36   }
37 }
38 }
```

The "Preview" window shows a visual representation of a sine wave. It consists of numerous small white circles arranged in a wavy pattern across a light gray background. The wave starts at the top left, dips down, rises to a peak, and then dips again, creating a continuous oscillation.



Sketch A1.14 adding waves

Now adding them together, moving the drawing of the circle so it is drawn after adding the waves together.

```
let waves = []

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 5; i++)
    {
        waves[i] = new Wave(random(20, 80), random(100, 400),
random(TWO_PI))
    }
}

function draw()
{
    background(220)
    for (let x = 0; x < width; x += 10)
    {
        let y = 0
        for (let wave of waves)
        {
            y += wave.calculate(x)
        }
        circle(x, y + height/2, 10)
    }
}

class Wave
{
```

```
constructor(amp, period, phase)
{
    this.amplitude = amp
    this.period = period
    this.phase = phase
}

calculate(x)
{
    return sin(this.phase + TWO_PI*x/this.period)*this.amplitude
}
}
```

Notes

Every time you refresh the sketch you will get something very different.

Figure A1.14

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, there are buttons for play/pause, auto-refresh, physics, public, and p5.js 1.11.10. The code editor has a left panel for code and a right panel for preview. The code in the left panel is:

```
> sketch.js Saved: just now Preview
20     y += wave.calculate(x)
21   }
22   circle(x, y + height/2, 10)
23 }
24 }
25
26 class Wave
27 {
28   constructor(amp, period, phase)
29   {
30     this.amplitude = amp
31     this.period = period
32     this.phase = phase
33   }
34
35   calculate(x)
36   {
37     return sin(this.phase +
TWO_PI*x/this.period)*this.amplitude
38   }
39 }
```

The preview window on the right shows a series of small circles forming a sine wave curve. The wave starts at the bottom left, goes up to a peak, down to a trough, and back up again. The circles are white with black outlines.



Sketch A1.15 using phase

Getting it in motion by using the phase.

```
let waves = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 5; i++)
  {
    waves[i] = new Wave(random(20, 80), random(100, 400),
random(TWO_PI))
  }
}

function draw()
{
  background(220)
  for (let x = 0; x < width; x += 10)
  {
    let y = 0
    for (let wave of waves)
    {
      y += wave.calculate(x)
    }
    circle(x, y + height/2, 10)
  }
  for (let wave of waves)
  {
    wave.phase += 0.1
  }
}
```

```
class Wave
{
    constructor(amp, period, phase)
    {
        this.amplitude = amp
        this.period = period
        this.phase = phase
    }

    calculate(x)
    {
        return sin(this.phase + TWO_PI*x/this.period)*this.amplitude
    }
}
```

Notes

This creates a moving wave

Challenges

1. Add more waves.
2. Add colour and alpha.
3. Change the values for period, amplitude, and phase.
4. Use vertex to draw lines in between.

Figure A1.15

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 side, it says "Hello, TheHappyCoder!" and shows "Public" and "p5.js 1.11.10". Below the menu bar, there are icons for play, stop, and refresh, followed by "Auto-refresh" and "Physics". The main area has tabs for "sketch.js" and "Preview". The code in "sketch.js" is as follows:

```
24  for (let wave of waves)
25  {
26      wave.phase += 0.1
27  }
28 }
29
30 class Wave
31 {
32     constructor(amp, period, phase)
33     {
34         this.amplitude = amp
35         this.period = period
36         this.phase = phase
37     }
38
39     calculate(x)
40     {
41         return sin(this.phase +
TWO_PI*x/this.period)*this.amplitude
42     }
43 }
```

The "Preview" window shows a sine wave plotted with small white circles at each data point. The wave oscillates between two horizontal lines, creating a smooth curve.



Sketch A1.16 a few tweaks

For clarity, this is an updated version of the previous sketch. The following changes: add `update()` to the class and use it at the end of `draw()`.

```
let waves = []

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 5; i++)
    {
        waves[i] = new Wave(random(20, 80), random(100, 400),
random(TWO_PI))
    }
}

function draw()
{
    background(220)
    for (let x = 0; x < width; x += 10)
    {
        let y = 0
        for (let wave of waves)
        {
            y += wave.calculate(x)
        }
        circle(x, y + height/2, 10)
    }
    for (let wave of waves)
    {
        wave.update()
    }
}
```

```
}

class Wave
{
    constructor(amp, period, phase)
    {
        this.amplitude = amp
        this.period = period
        this.phase = phase
    }

    calculate(x)
    {
        return sin(this.phase + TWO_PI*x/this.period)*this.amplitude
    }

    update()
    {
        this.phase += 0.05
    }
}
```

***** Notes

A simple tweak.

Figure A1.16

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 side, it says "Hello, TheHappyCoder!" and has buttons for "Public" and "p5.js 1.11.10". Below the menu bar, there are icons for play, stop, and refresh, followed by "Auto-refresh" and "Physics" 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 following code is written:

```
29
30 class Wave {
31   constructor(amp, period, phase) {
32     this.amplitude = amp
33     this.period = period
34     this.phase = phase
35   }
36
37   calculate(x) {
38     return sin(this.phase +
39 TWO_PI*x/this.period)*this.amplitude
40   }
41
42   update() {
43     this.phase += 0.05
44   }
45 }
```

In the "Preview" section, a sine wave is visualized as a series of small circles connected by lines, forming a smooth curve. The wave starts at the bottom left, reaches a peak, crosses the x-axis, reaches another peak, and then crosses the x-axis again towards the end.