Intelligent Machines Workbook #2 Robotics and p5.js

Table of Contents

MIT Licence

Brief summary:

Permissions (what you can do)

Commercial use
Modification
Distribution
Private use

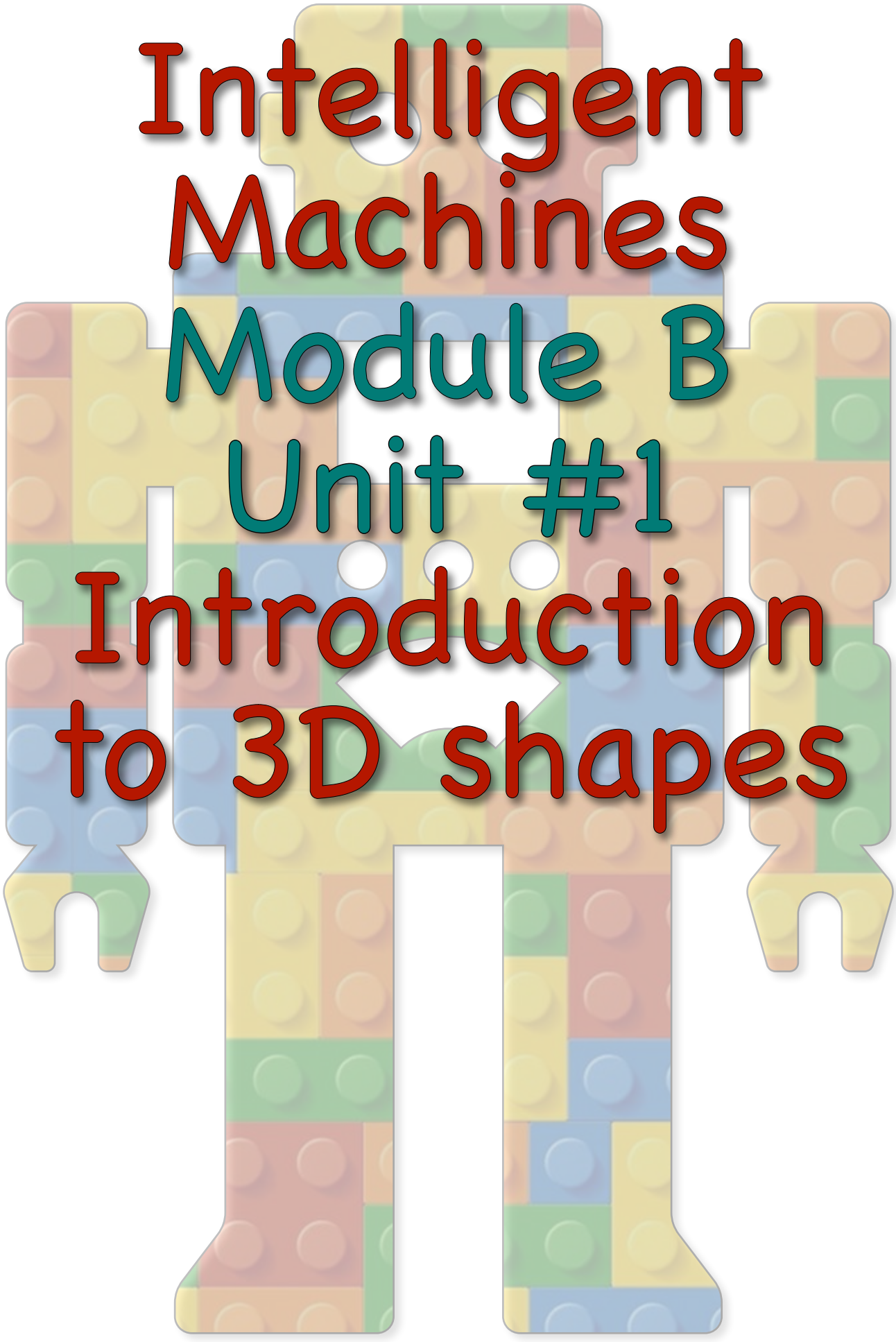
Limitations (what is not covered)

Liability
Warranty

Workbook #2 Quick Content Summary

xxxx

Intelligent Machines Module B Unit #1 Introduction to 3D shapes



 Module B Unit #1: p5.js 3D shapes

In p5.js, WEBGL is one of the two available rendering modes, allowing you to create 3D graphics and interactive experiences alongside its default 2D mode.
Key features of WEBGL in p5.js:

3D shapes
Draw basic shapes like boxes (cuboids), spheres, cones, cylinders, and more using functions like `box()`, `sphere()`, etc.

Custom geometry
Create complex models from code or load them from 3D file formats like `OBJ` and `STL`.

Materials and lighting
Define materials like `basicMaterial()` or `specularMaterial()` and use lights like `ambientLight()` to affect object appearance.

Camera control
Change the viewpoint using functions like `perspective()`, `ortho()`, and rotate the camera with `rotateX()`, `rotateY()`, `rotateZ()`.

Textures
Add textures to surfaces for enhanced realism and detail.
Shaders: For advanced users, write custom shaders to achieve unique visual effects.

We will draw some of the basic shapes and learn about the co-ordinates, translation, and rotation. There are a number of very key differences between the default 2D and the 3D environment.

One of the main changes is how we use the coordinates. All coordinates are based around the centre of the 3D space. There is an `x`, `y`, and `z` component to any coordinates; if only two are specified, then it takes the `x` and `y` and relegates the `z` to zero.

The `x` component is left and right, `y` is top and bottom, and the `z` is front and back.

Sketch B1.1 standard sketch

Starting with our standard sketch.

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

function draw()
{
  background(220)
}
```

Sketch B1.2 adding the WEBGL

The first thing to notice is that we have the third argument in the `createCanvas()` function: `WEBGL`.

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

function draw()
{
  background(220)
}
```

Notes

The first thing you notice is that nothing happens. What you now have is effectively a 3D canvas or space. One where you can draw in the `x`, `y`, and `z` directions.

Code Explanation

`createCanvas(400, 400, WEBGL)`	WEGL allows to render an image in 3D

Sketch B1.3 drawing a box

What we need is a 3D shape to put into this space. We will start by drawing a cube. To draw a cube, we use the box() function; it will have dimensions of 100 pixels by 100 pixels by 100 pixels. We only need to give the single dimension, as it assumes the other two are the same. We will add the other two when we draw a cuboid later.

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

function draw()
{
  background(220)
  box(100)
}
```

Notes

One thing you will notice is that I have not specified where to draw it. I have given it no coordinates, yet it has drawn it in the centre of the canvas. Also, it doesn't look very 3D...ish. To the first point, the centre of the canvas is in the centre in all three dimensions, the x, y, and z axes. Secondly, we need to rotate it to see it.

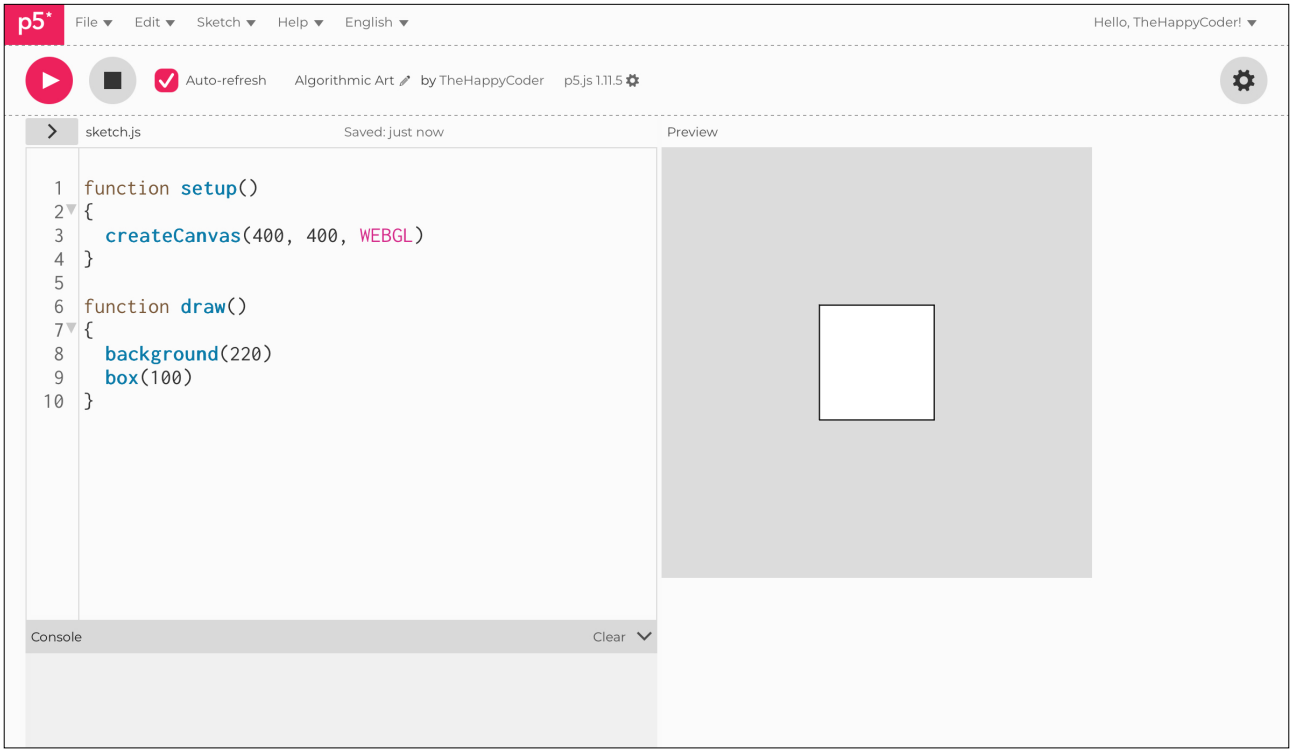
Challenge

Add the other two dimensions.

Code Explanation

box(100)	A rectangle with each side of equal length (100)

Figure B1.3


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, WEBGL)
4 }
5
6 function draw()
7 {
8 background(220)
9 box(100)
10 }
Console
Clear

Sketch B1.4 rotating

We cannot simply rotate it; we have to specify which axis we want to rotate it on, whether it is the x, y, or z axis, and by some angle (measured in radians, for now). The functions we use are: rotateX(), rotateY(), and rotateZ(), which I hope are self-explanatory. We will rotate the box along the x axis by 2 radians.

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

function draw()
{
  background(220)
  rotateX(2)
  box(100)
}
```

Notes

We can now see that it is a cube, at least from one angle.

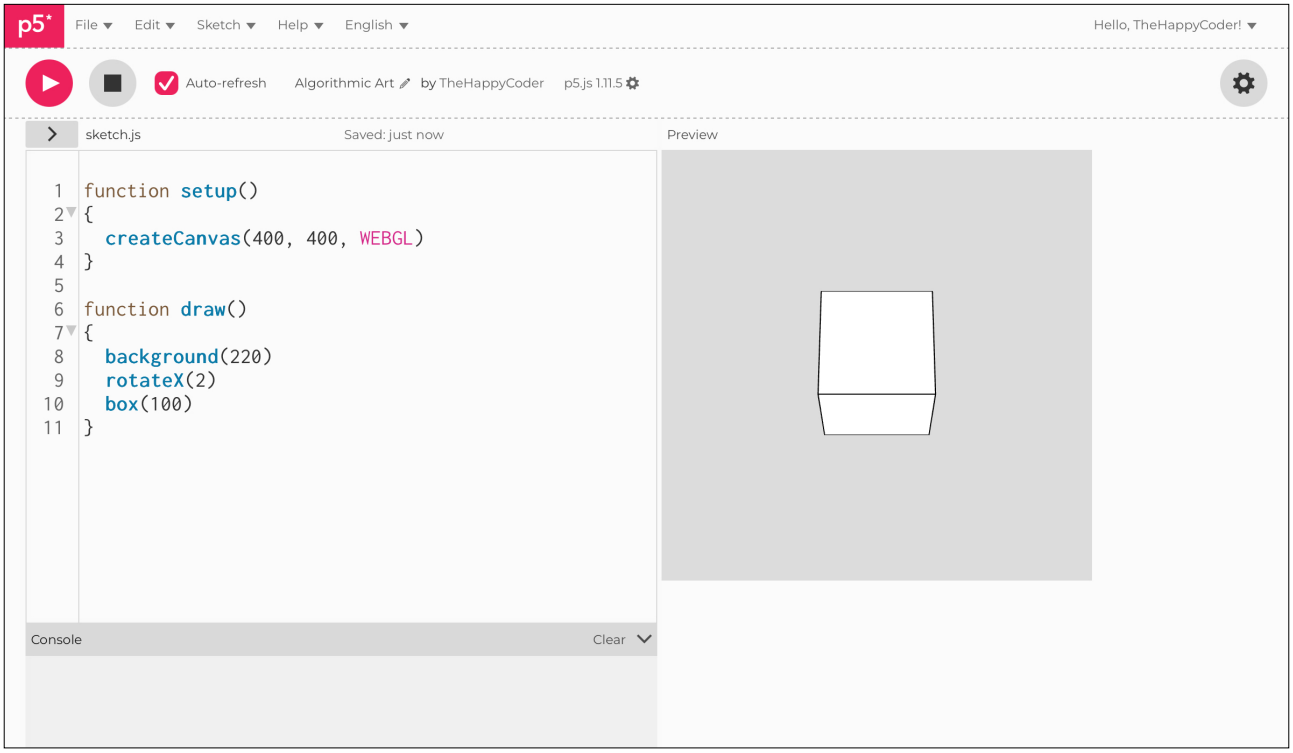
Challenge

Try other angles.

Code Explanation

`rotateX(2)`	Rotate along the x axis by 2 radians

Figure B1.4


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
1 function setup()
2 {
3 createCanvas(400, 400, WEBGL)
4 }
5
6 function draw()
7 {
8 background(220)
9 rotateX(2)
10 box(100)
11 }
Console
Clear

 Sketch B1.5 45° rotation

To make life a little easier to understand, I will change the angle mode to degrees, which is a bit more intuitive.

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

function draw()
{
  background(220)
  rotateX(45)
  box(100)
}
```

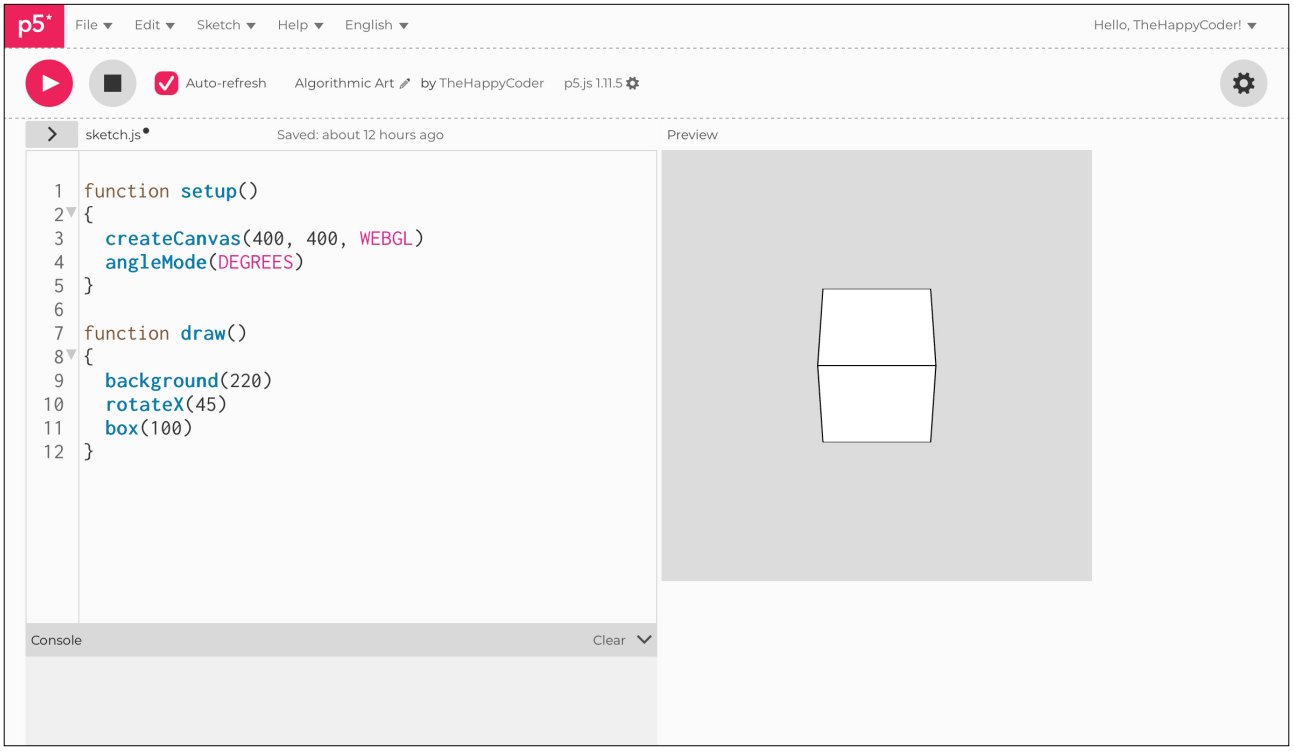
Notes

Almost the same

Code Explanation

`rotateX(45)`	Rotating through 45° along the x axis

Figure B1.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: about 12 hours ago
Preview
1 function setup()
2 {
3 createCanvas(400, 400, WEBGL)
4 angleMode(DEGREES)
5 }
6
7 function draw()
8 {
9 background(220)
10 rotateX(45)
11 box(100)
12 }
Console
Clear

Sketch B1.6 the x and y axis

To show how 3D it is, we are going to rotate along all three axes.

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

function draw()
{
  background(220)
  rotateX(45)
  rotateY(45)
  rotateZ(45)
  box(100)
}
```

Notes

That is definitely more 3D...ish.

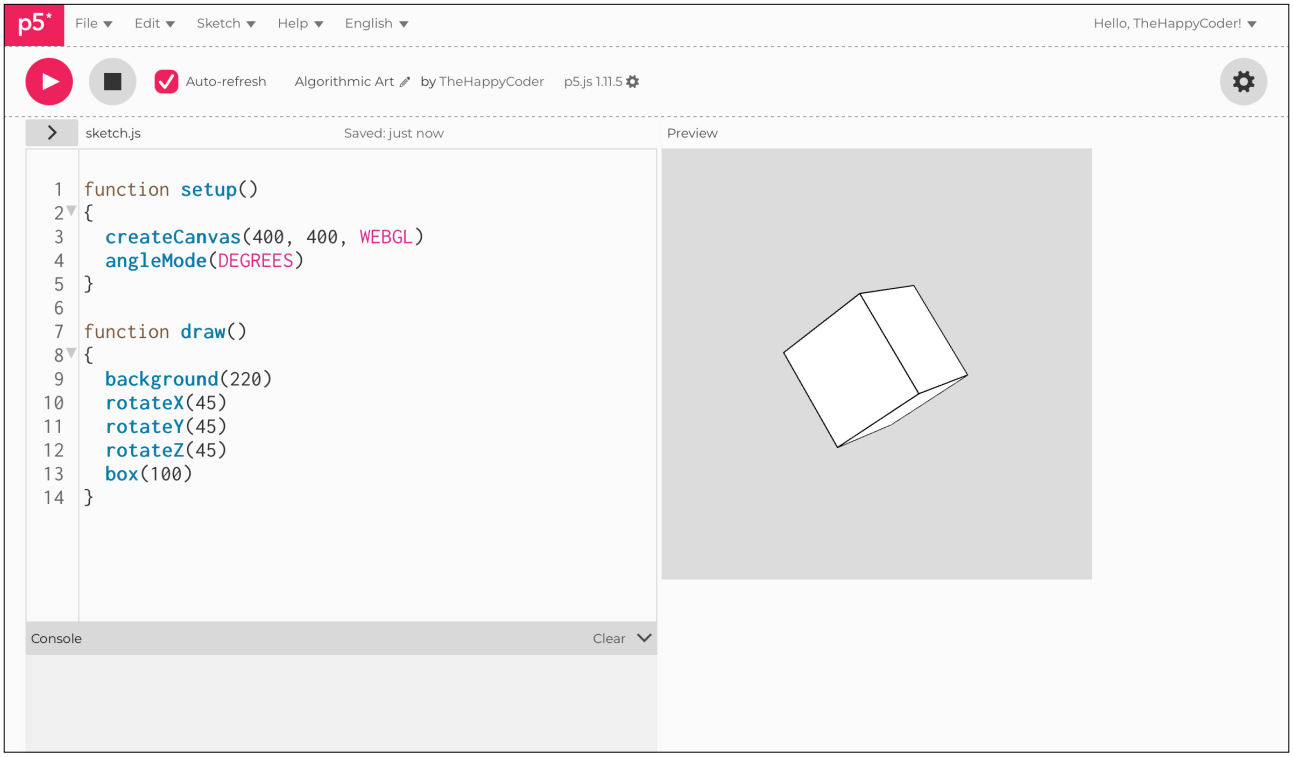
Challenge

Try different angles.

Code Explanation

`rotateY(45)`	Rotating through 45° along the y axis
`rotateZ(45)`	Rotating through 45° along the z axis

Figure B1.6


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, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(45)
rotateY(45)
rotateZ(45)
box(100)
}
Console
Clear

Sketch B1.7 incrementing the rotation

Even better still, we can animate the rotation by introducing a variable (`angle`) and incrementing it.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  angle++
}
```

Notes

You should see it rotating in all directions. We have incremented the angle of rotation by `1°` on all three axes.

Challenges

1. Try `-angle` for one of them.
2. Move it faster: `angle += 3`.

Sketch B1.8 drawing a cuboid

That was just a cube; we can add other dimensions to make it a cuboid.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100, 150, 50)
  angle++
}
```

Notes

Nicely rotating cuboid.

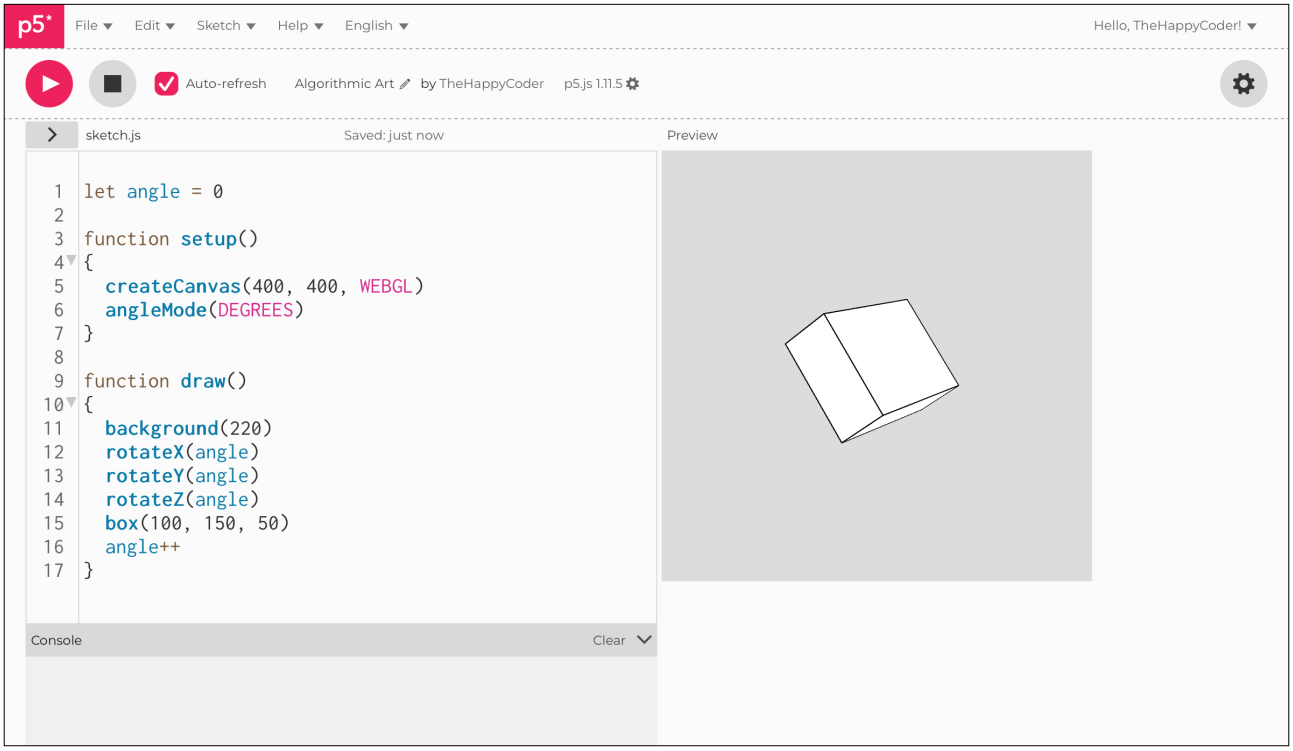
Challenge

Try other dimensions.

Code Explanation

`box(100, 150, 50)`	Draws a cuboid 100 wide, 150 long and 50 deep

Figure B1.8


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 angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100, 150, 50)
angle++
}
Console
Clear

Sketch B1.9 drawing a plane

We will draw another shape, a simple plane, where it has two dimensions. Do you notice that there is a line going across it? This is because all the shapes that are created are made up of triangles.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  plane(100, 150)
  angle++
}
```

Notes

A floating plane.

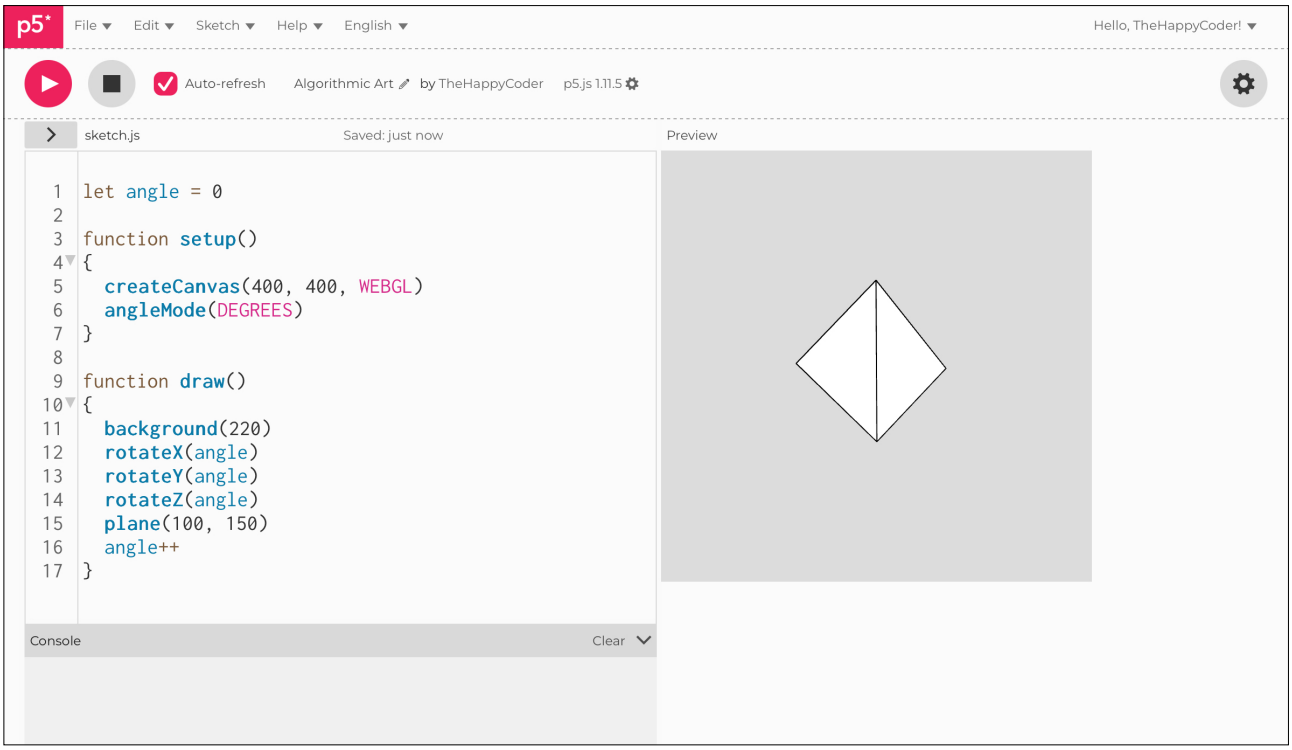
Challenge

Add noStroke().

Code Explanation

plane(100, 150)	Creates a flat plane 100 by 150

Figure B1.9


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 angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
plane(100, 150)
angle++
}
Console
Clear

Sketch B1.10 drawing a cylinder

A cylinder has two dimensions, one for the radius and the other for the length.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  cylinder(100, 150)
  angle++
}
```

Notes

The first is the diameter and the second is the length. If you use noStroke() you lose a lot of definition; we will use lights to address that issue (in another unit).

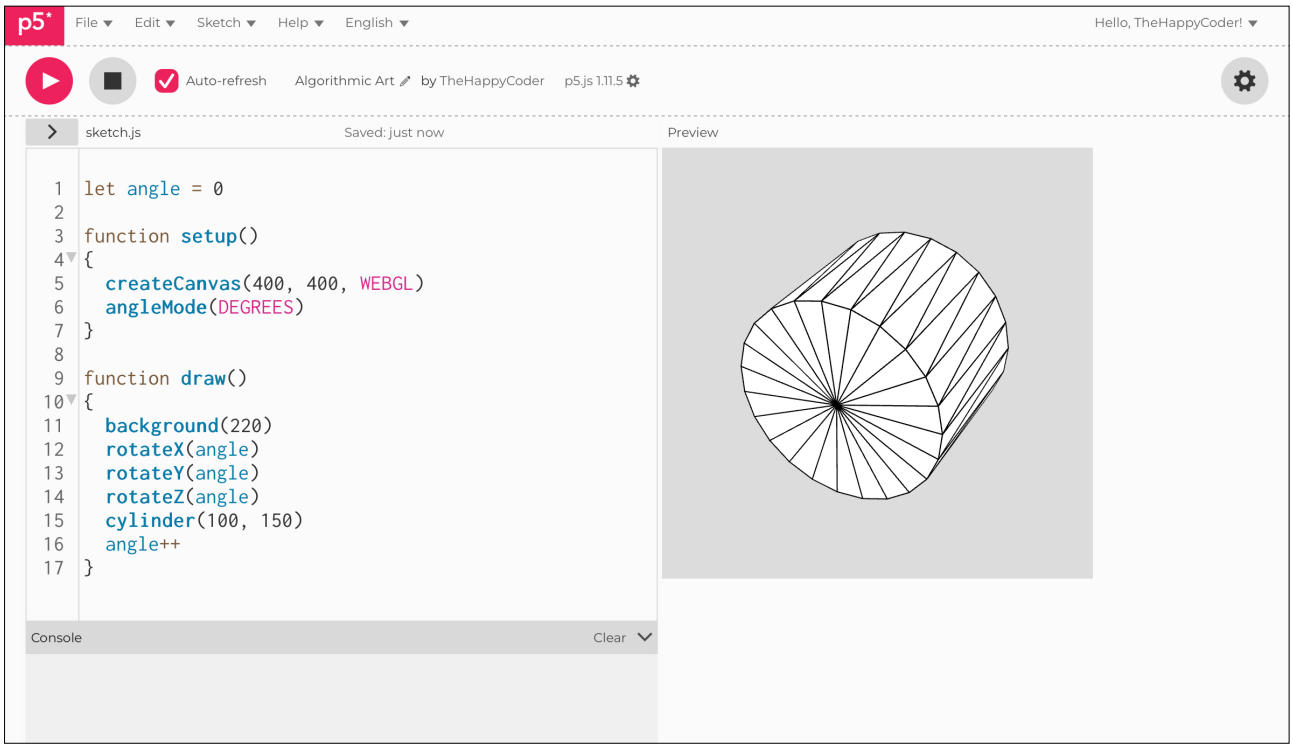
Challenge

Alter the dimensions.

Code Explanation

cylinder(100, 150)	A cylinder with a radius 100 and length 150

Figure B1.10


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 angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
cylinder(100, 150)
angle++
}
Console
Clear

Sketch B1.11 drawing a sphere

Here is a sphere with a single dimension, the radius.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  sphere(100)
  angle++
}
```

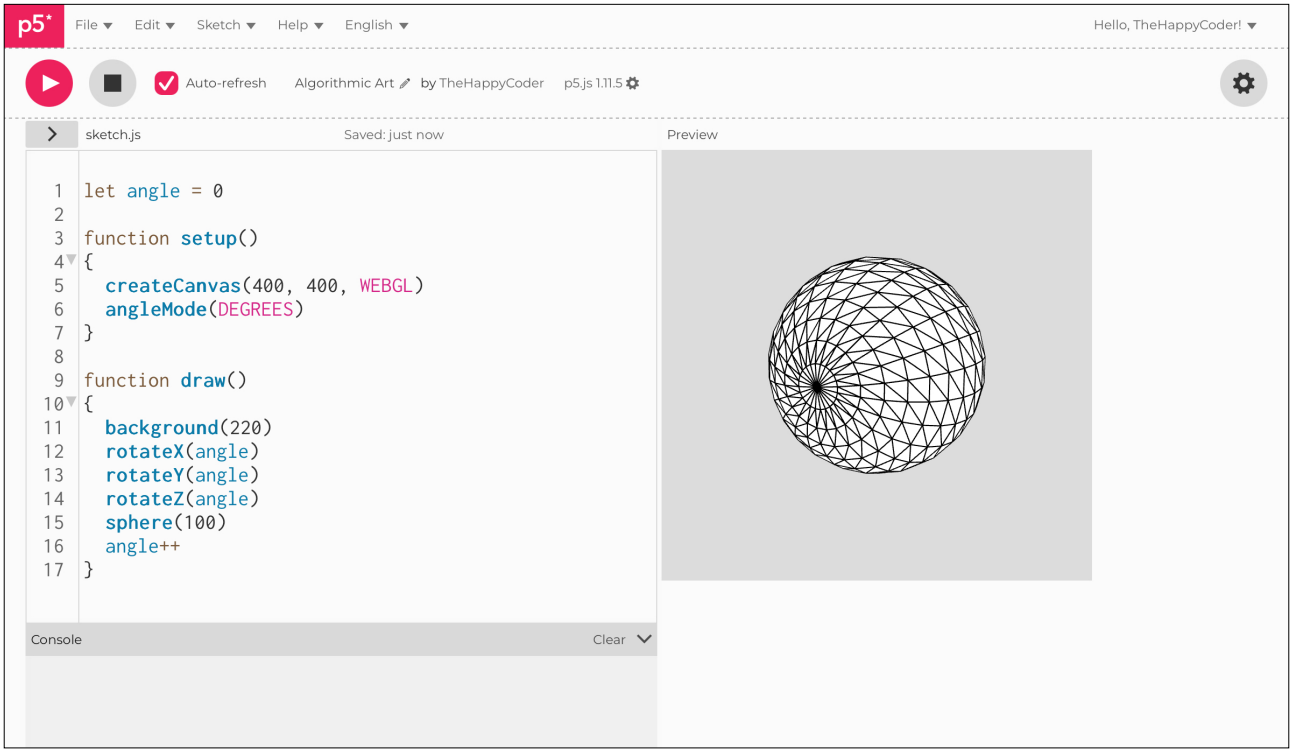
Notes

We will explore how to make lots of shapes in different positions at a later date.

Code Explanation

sphere(100)	Sphere with a radius of 100

Figure B1.11


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
let angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
sphere(100)
angle++
}
Console
Clear

Sketch B1.12 drawing an ellipsoid

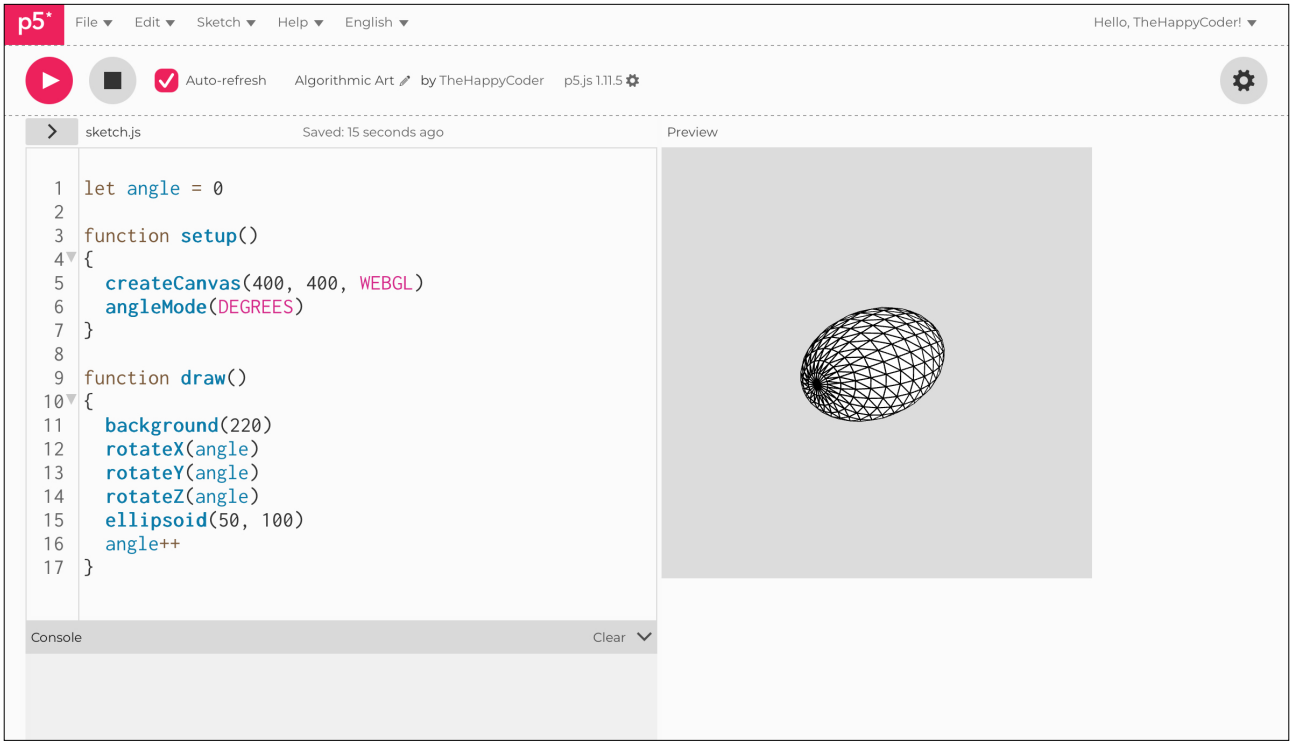
An ellipsoid is a sphere but with two dimension. A radius in one direction and a perpendicular radius.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  ellipsoid(50, 100)
  angle++
}
```

Code Explanation

`ellipsoid(50, 100)`	Drawing an ellipsoid 50 wide, 100 long

Figure B1.12


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 15 seconds ago
Preview
let angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
ellipsoid(50, 100)
angle++
}
Console
Clear

 Sketch B1.13 drawing a torus

Now for a torus (doughnut/donut), which has two dimensions: the torus radius and tube radius (thickness of the doughnut).

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  torus(100, 50)
  angle++
}
```

Notes

The radius of the torus (taken at the centre of the tube) is the first argument, and the radius of the actual tube itself is the second.

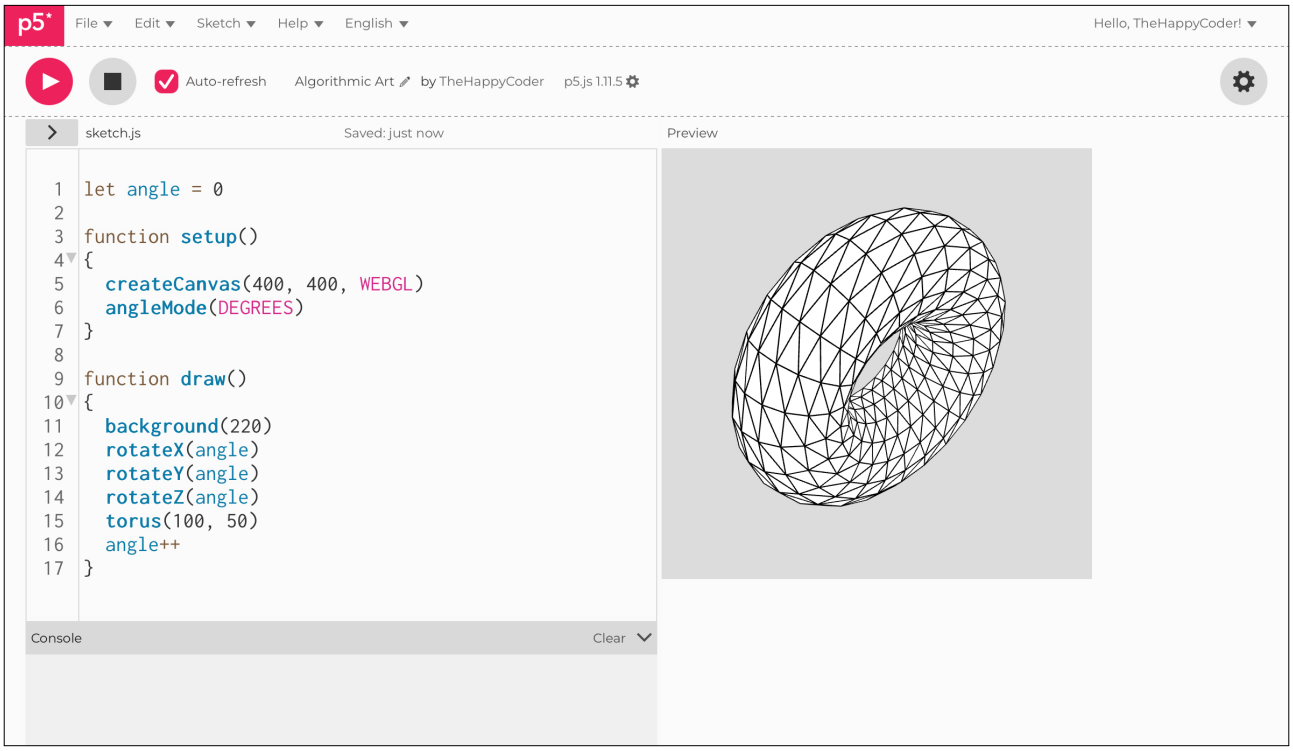
Challenge

Play with the values to get a feel for the shape.

Code Explanation

`torus(100, 50)`	A torus with a radius of 100 and a tube dimension of 50

Figure B1.13


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
let angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
torus(100, 50)
angle++
}
Console
Clear

Sketch B1.14 drawing a cone

A cone which has two dimensions, also, the first is the radius of the base and the second is the length (or height) of the cone.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  cone(100, 200)
  angle++
}
```

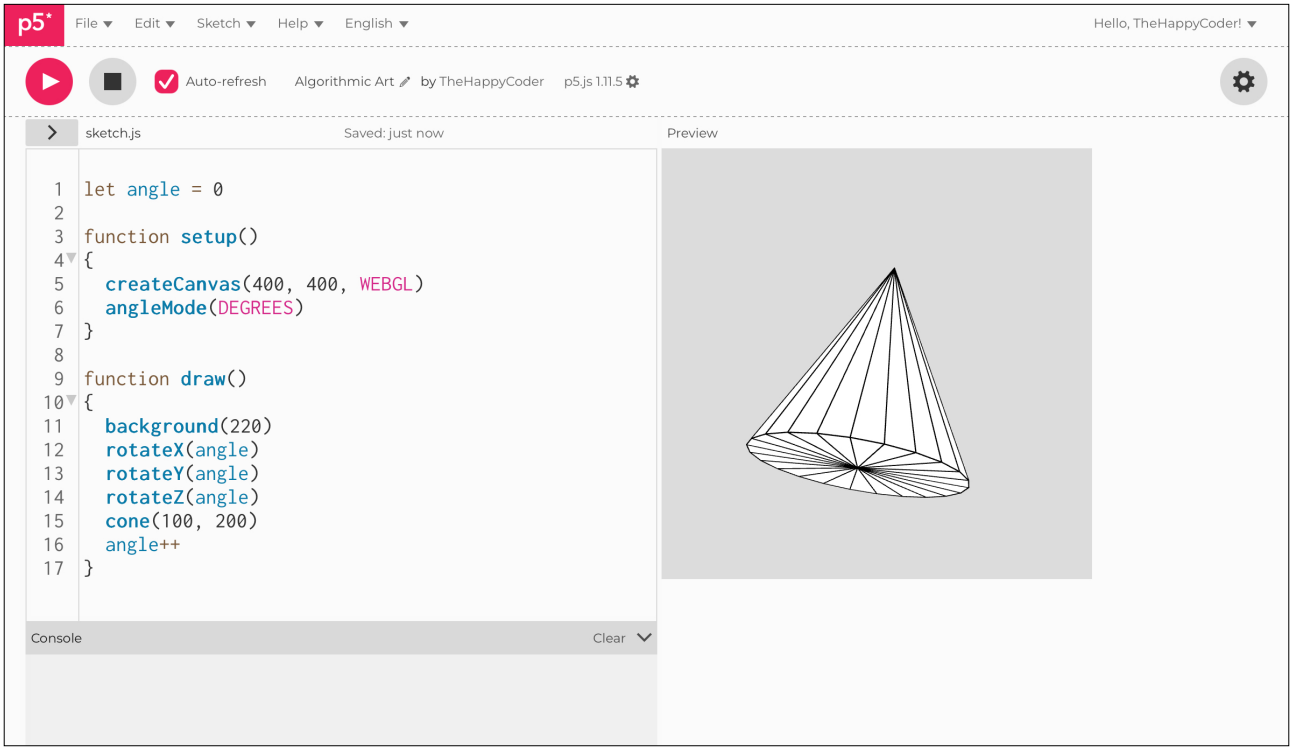
Challenges

1. Have more than one shape at the same time.
2. Use noFill().

Code Explanation

cone(100, 200)	Draws a cone with a radius 100 and length 200

Figure B1.14


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
let angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
cone(100, 200)
angle++
}
Console
Clear

Sketch B1.15 adding a bit of colour

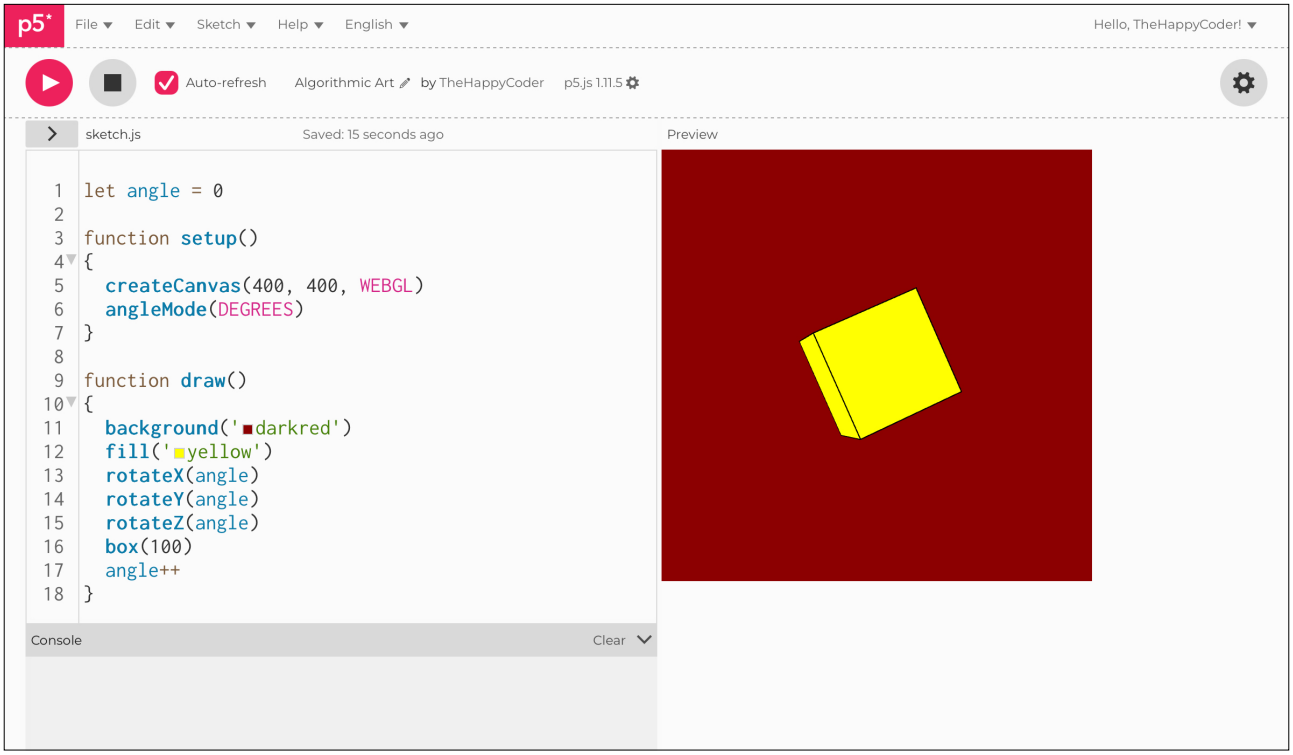
❗ remove cone()
Let's go back to our simple cube. We can add colour with `fill()` and give the `background()` some colour for good measure.

```
let angle = 0

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

function draw()
{
  background('darkred')
  fill('yellow')
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  angle++
}
```

Notes
This gives us a nice dark red background and a yellow cube. We will look at materials later.

Challenge
Explore other colours.

Figure B1.15


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 15 seconds ago
Preview
let angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background('darkred')
fill('yellow')
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100)
angle++
}
Console
Clear

Sketch B1.16 translate

To move a shape in 3D (WEBGL), we translate the origin of the space. It takes a bit of getting used to as we are translating relative to the centre of the 3D space rather than from the top left as in the 2D canvas.

```
let angle = 0

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

function draw()
{
  background('darkred')
  fill('yellow')
  translate(100, 100)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  angle++
}
```

Notes

This moves it down and to the right; notice that it still rotates about the new origin.

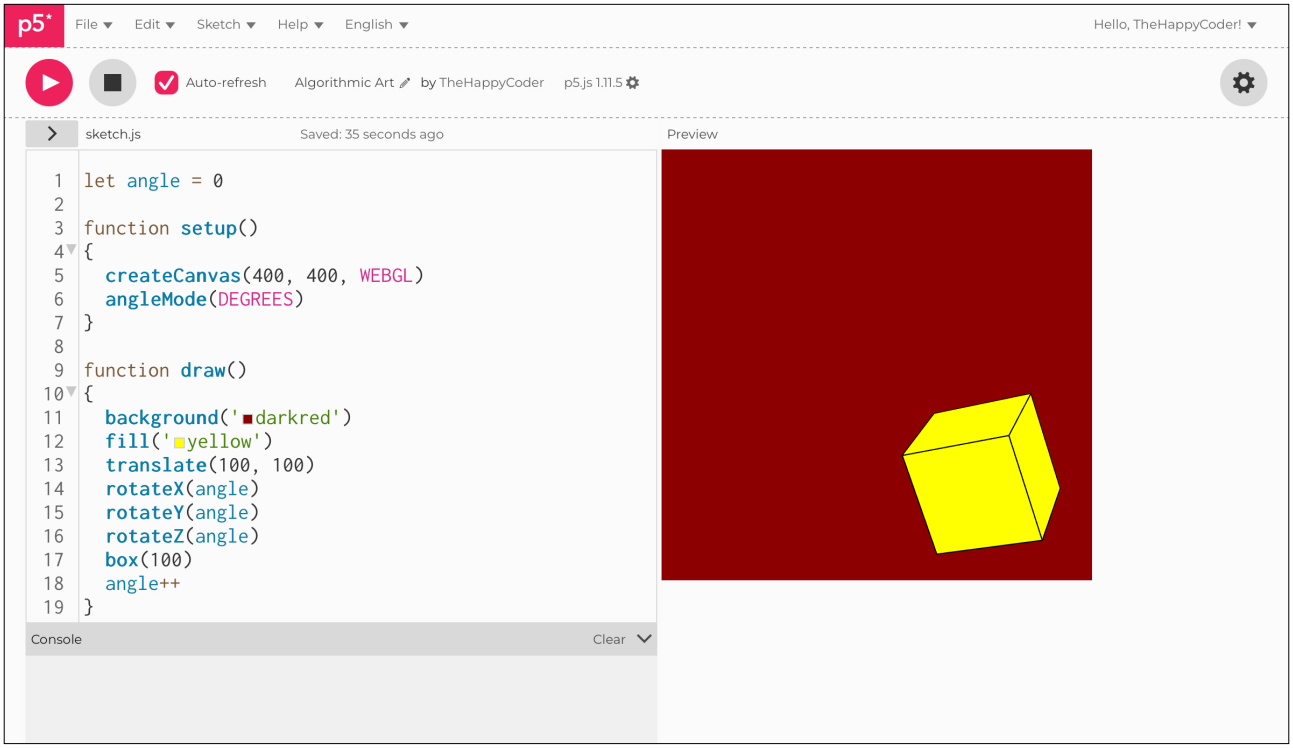
Challenges

1. Try other translations.
2. Do you know how to translate it to the left?

Code Explanation

`translate(100, 100)`	Translating relative to the original origin

Figure B1.16


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 35 seconds ago
Preview
let angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background('darkred')
fill('yellow')
translate(100, 100)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100)
angle++
}
Console
Clear

 Sketch B1.17 translate ‘tother way

Translating it to the top left-hand corner

```
let angle = 0

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

function draw()
{
  background('darkred')
  fill('yellow')
  translate(-100, -100)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  angle++
}
```

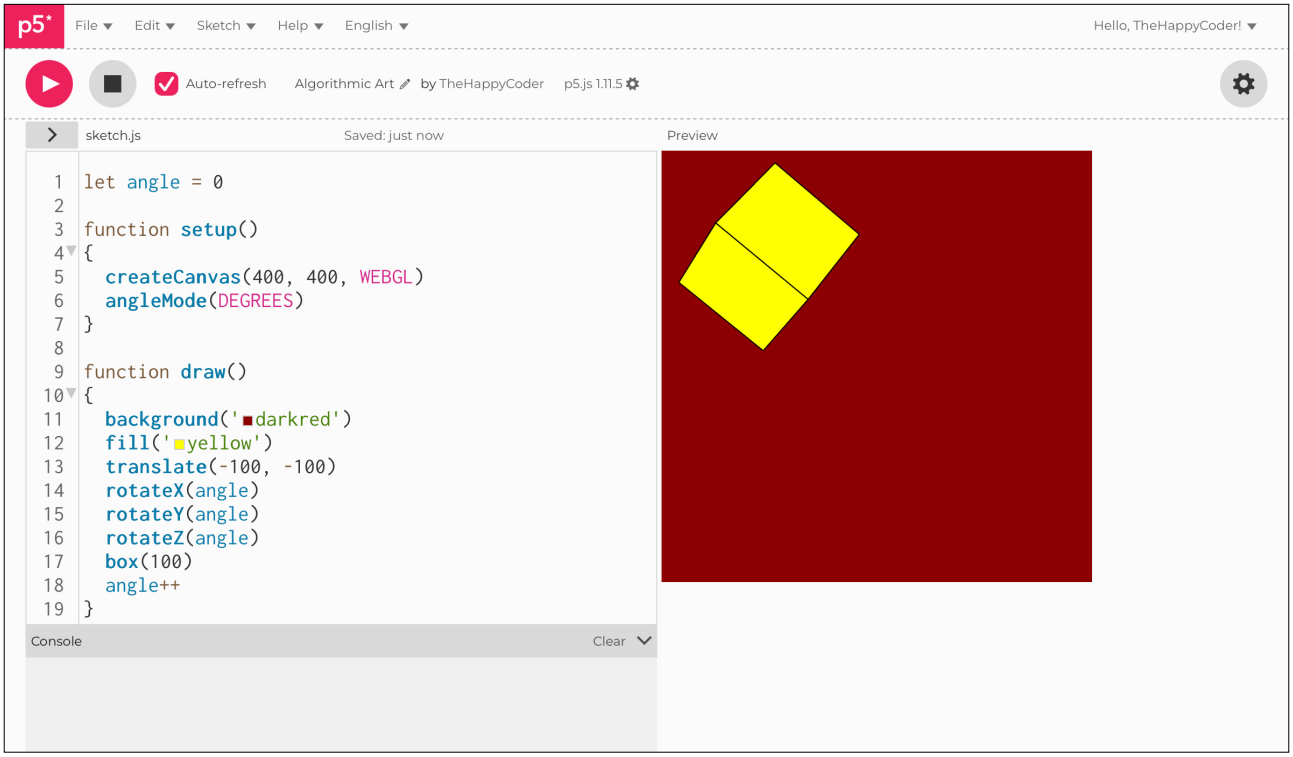
Notes

We need to translate it negatively relative to the old origin.

Code Explanation

`translate(-100, -100)`	Translating relative to the original origin

Figure B1.17


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 angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background('darkred')
fill('yellow')
translate(-100, -100)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100)
angle++
}
Console
Clear

Sketch B1.18 translate along the z axis

But what about the z axis? We will now translate along the z axis only; we still need all three arguments for translate.

```
let angle = 0

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

function draw()
{
  background('darkred')
  fill('yellow')
  translate(0, 0, 500)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  angle++
}
```

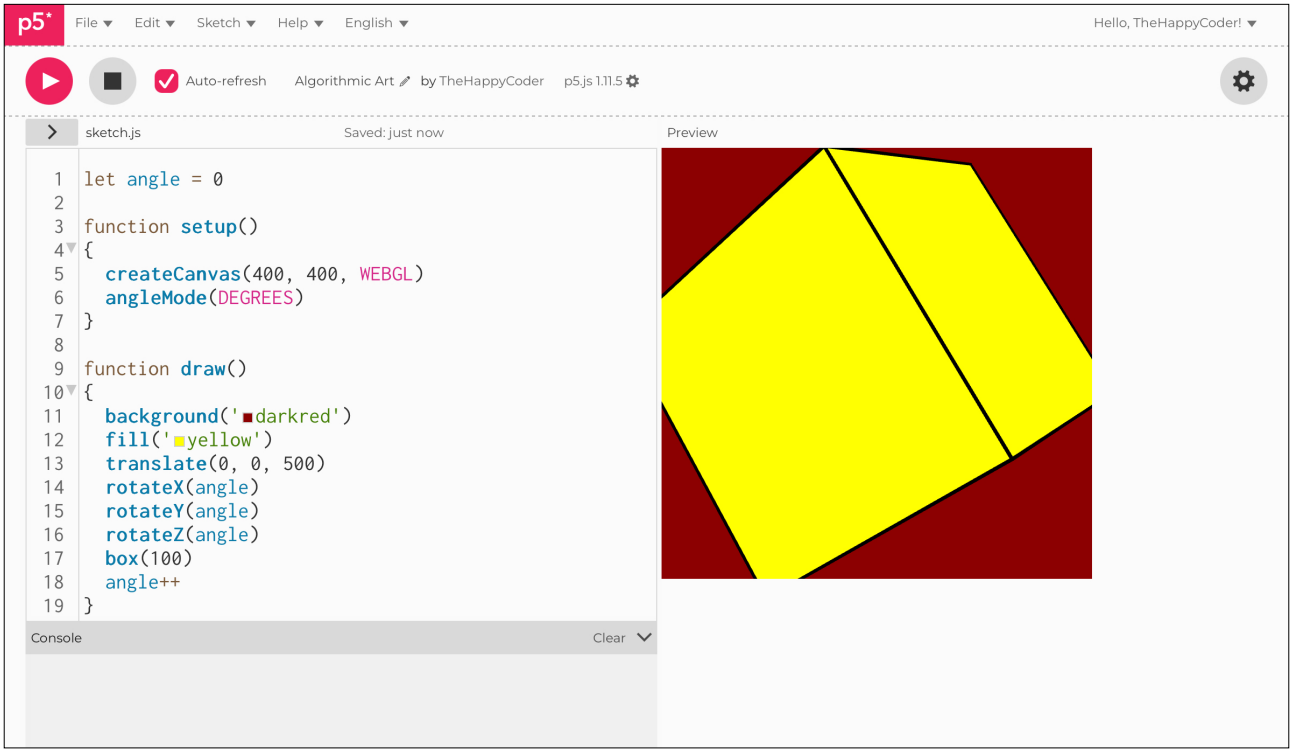
Notes

This has moved it a lot closer; a positive value moves it towards you.

Code Explanation

`translate(0, 0, 500)`	Translated only in the z direction, positive value brings it forwards

Figure B1.18


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 angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background('darkred')
fill('yellow')
translate(0, 0, 500)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100)
angle++
}
Console
Clear

Sketch B1.19 translate the other way

Giving it a negative value moves it further away; you are translating the space, not the shape, remember. So if you add other shapes, they too will be affected by the same amount.

```
let angle = 0

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

function draw()
{
  background('darkred')
  fill('yellow')
  translate(0, 0, -500)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  angle++
}
```

Notes

You can see how you can position the shape relative to the origin in all three planes (axes).

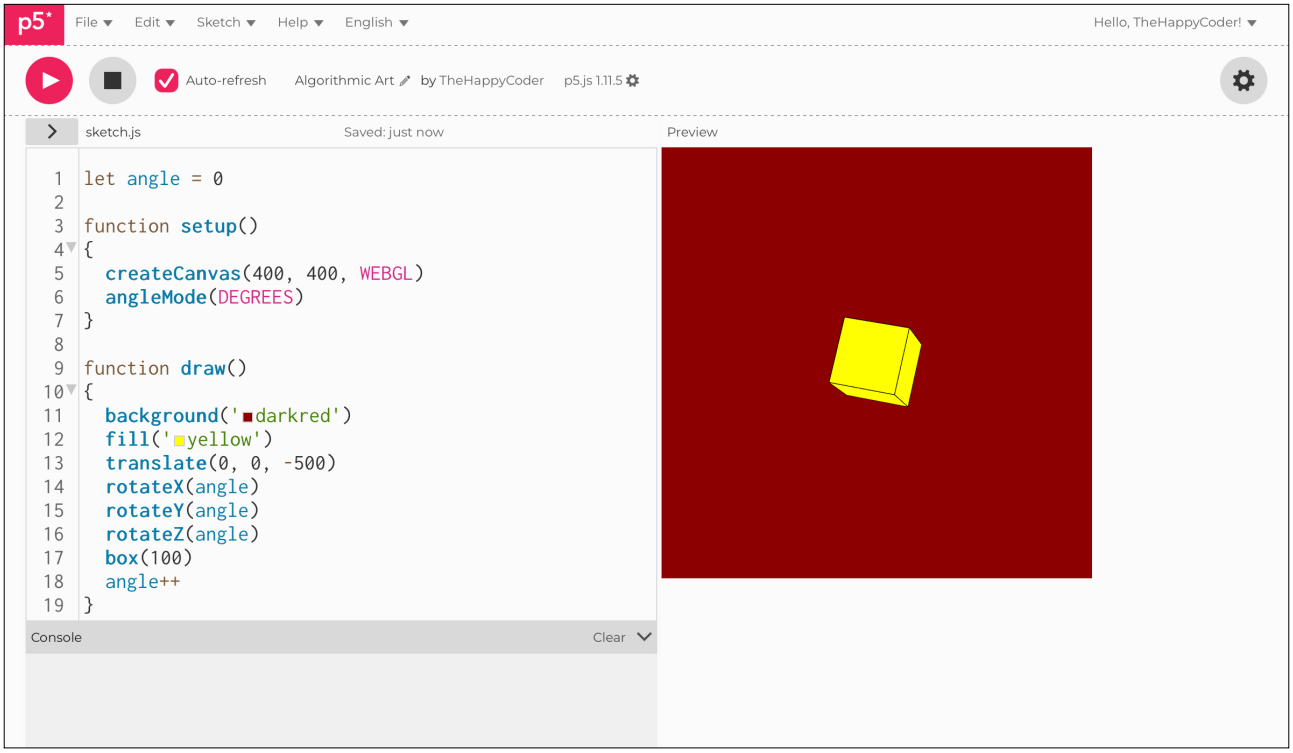
Challenge

Play with all three values.

Code Explanation

`translate(0, 0, -500)`	Moves it further away by 500

Figure B1.19


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 angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background('darkred')
fill('yellow')
translate(0, 0, -500)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100)
angle++
}
Console
Clear

Sketch B1.20 more than one shape

But what happens if you have more than one shape occupying the same coordinates? Let's find out. We will add a cone and a torus to the box.

❗ Remove `translate()`

```
let angle = 0

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

function draw()
{
  background('darkred')
  fill('yellow')
  // translate(0, 0, -500)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  torus(100, 25)
  cone(100, 200)
  angle++
}
```

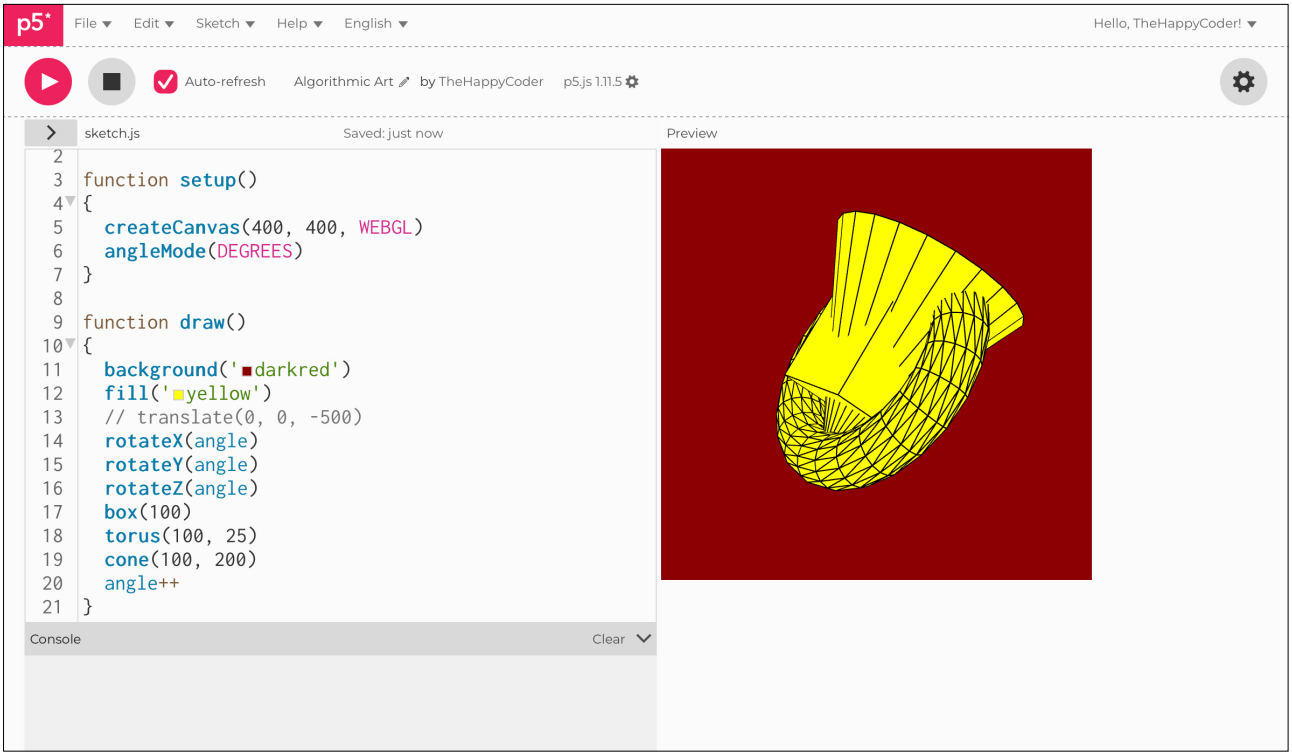
Notes

All three are meshed together.

Challenge

Try `noFill()` and `stroke('white')`.

Figure B1.20


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, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background('darkred')
fill('yellow')
// translate(0, 0, -500)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100)
torus(100, 25)
cone(100, 200)
angle++
}
Console
Clear

Sketch B1.21 pushing and popping

Using `push()` and `pop()` we can rotate and translate individual shapes (or groups of shapes).

```
let angle = 0

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

function draw()
{
  background('darkred')
  fill('yellow')
  // translate(0, 0, -500)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  push()
  translate(0, 0, 100)
  rotateX(angle * 2)
  torus(100, 25)
  pop()
  cone(100, 200)
  angle++
}
```

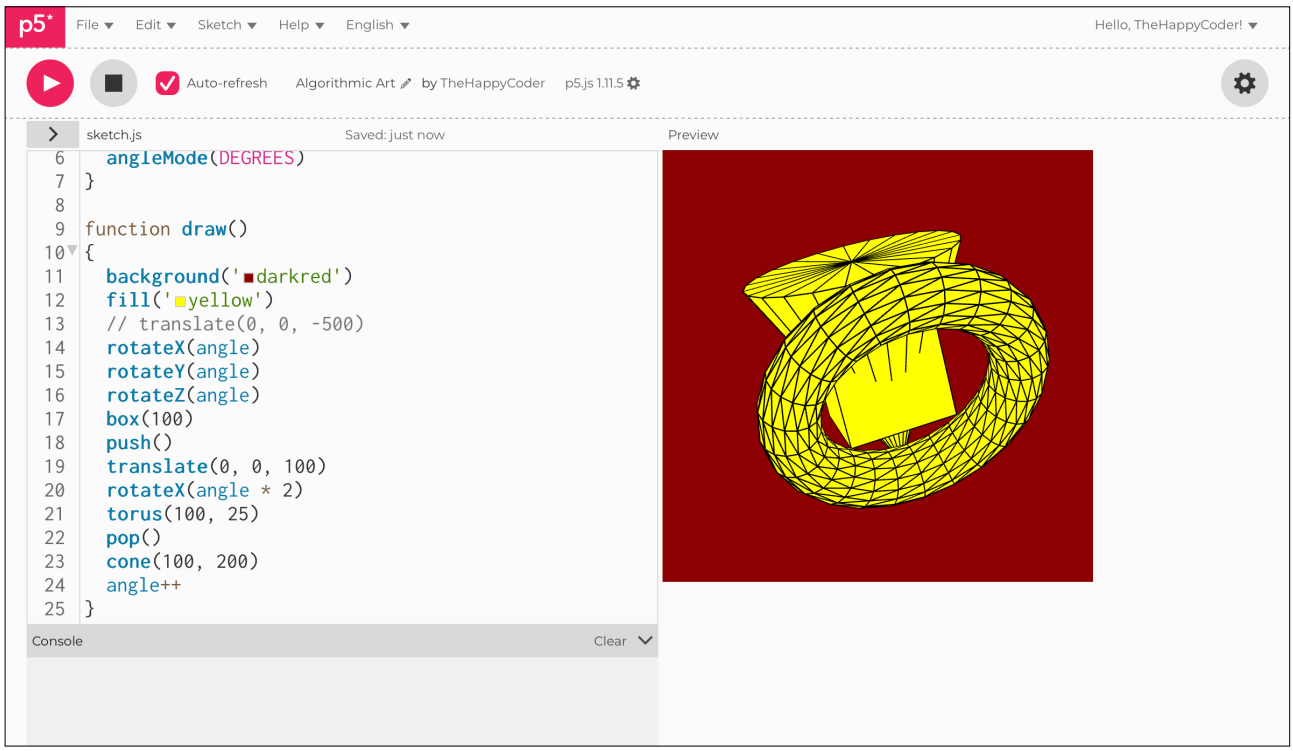
Notes

The torus moves independently of the other two shapes, and yet all three are still moving together. There is a lot of scope to do much more.

Challenge

Just play and see what you can create.

Figure B1.21


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
angleMode(DEGREES)
}
function draw()
{
background('darkred')
fill('yellow')
// translate(0, 0, -500)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100)
push()
translate(0, 0, 100)
rotateX(angle * 2)
torus(100, 25)
pop()
cone(100, 200)
angle++
}
Console
Clear

Intelligent Machines
Module B
Unit #2
LED on/off with p5.js

 Module A Unit #2: LED with p5.js

Previously, we were able to control the LED with code inside the sketch, either to blink, switch it on or off. We could even use the serial port to send messages to switch it on or off or control its brightness.

Here, we are going to take this a step further and use p5.js to do just that. In the first unit, we are going to simply switch it on with a click of the mouse on the canvas. This will lead us onto being able to connect our Arduino with AI. The Machine Learning tool (ml5.js) will be able to directly engage with it, both sending data to it and also receiving data from its many sensors.

This can get a bit complicated because we are coding in two different places: the p5.js web editor and the Arduino IDE. To, hopefully, make it clear which is which, I have colour-coded them separately. We will start with the Arduino code. There are no components to add, just the Arduino Nano 33 BLE connected, as before, to your computer.

Sketch B2.1 LED

! Arduino sketch

Using pin 13 for the LED, waits for the input words either HIGH or LOW to turn the LED on (HIGH) or off (LOW).

Arduino sketch

```
const int ledPin = 13;

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "HIGH")
    {
      digitalWrite(ledPin, HIGH);
    }
    else if (command == "LOW")
    {
      digitalWrite(ledPin, LOW);
    }
  }
}
```

Notes

All we need to do now (as far as the Arduino is concerned) is wait for the incoming `command`.

Code Explanation

`const int ledPin = 13;`	LED is connected to pin 13
`Serial.begin(9600);`	Match the baud rate in p5.js
`String command = Serial.readStringUntil('\n');`	Read until newline
`command.trim();`	Remove leading/trailing whitespace

Talking to the USB port

Head over to the p5.js web editor. Here, we will need to do a number of things. The first is getting the web editor to talk to the port. This requires some extra code and software. The great thing is that someone has done the hard work for us, and so we can use a webserial library created for just this purpose.

We simply add it into the index.html file.

```
<script src="https://unpkg.com/p5-webserial@0.1.1/build/p5.webserial.js"></script>
```

One issue is that you have to use Chrome or Edge as the web browser (not Safari).

Because you will be communicating on the same port as the Arduino Nano 33 BLE, you probably can't update the Arduino (upload) if the p5.js web editor is connected to it. If, for some reason, you need to change the code on the Arduino, then you first have to stop the p5.js web browser before proceeding with the Arduino upload, and conversely, you can't have the serial monitor open if you are communicating from the web editor. You'll find out soon enough.

Although we have added a library to take care of the heavy lifting, we do need some code to make the web editor talk to the Arduino and vice versa. To do that, we will create another file called port.js and add that to the index.html file.

The port.js functions

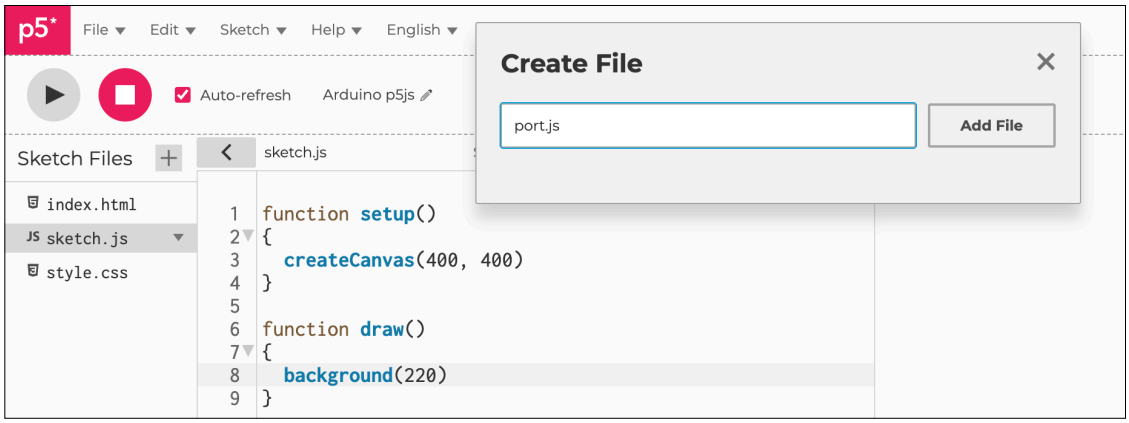
As part of the library, we have to create a number of functions to fulfil a number of actions. They are relatively self-evident. Rather than trying to analyse each line of code, I will summarise each one in turn.

function navigation()
function makePortButton()
function choosePort()
function openPort()
function portError(err)
function serialEvent()
function portConnect()
function portDisconnect()
function closePort()

Creating the port.js file

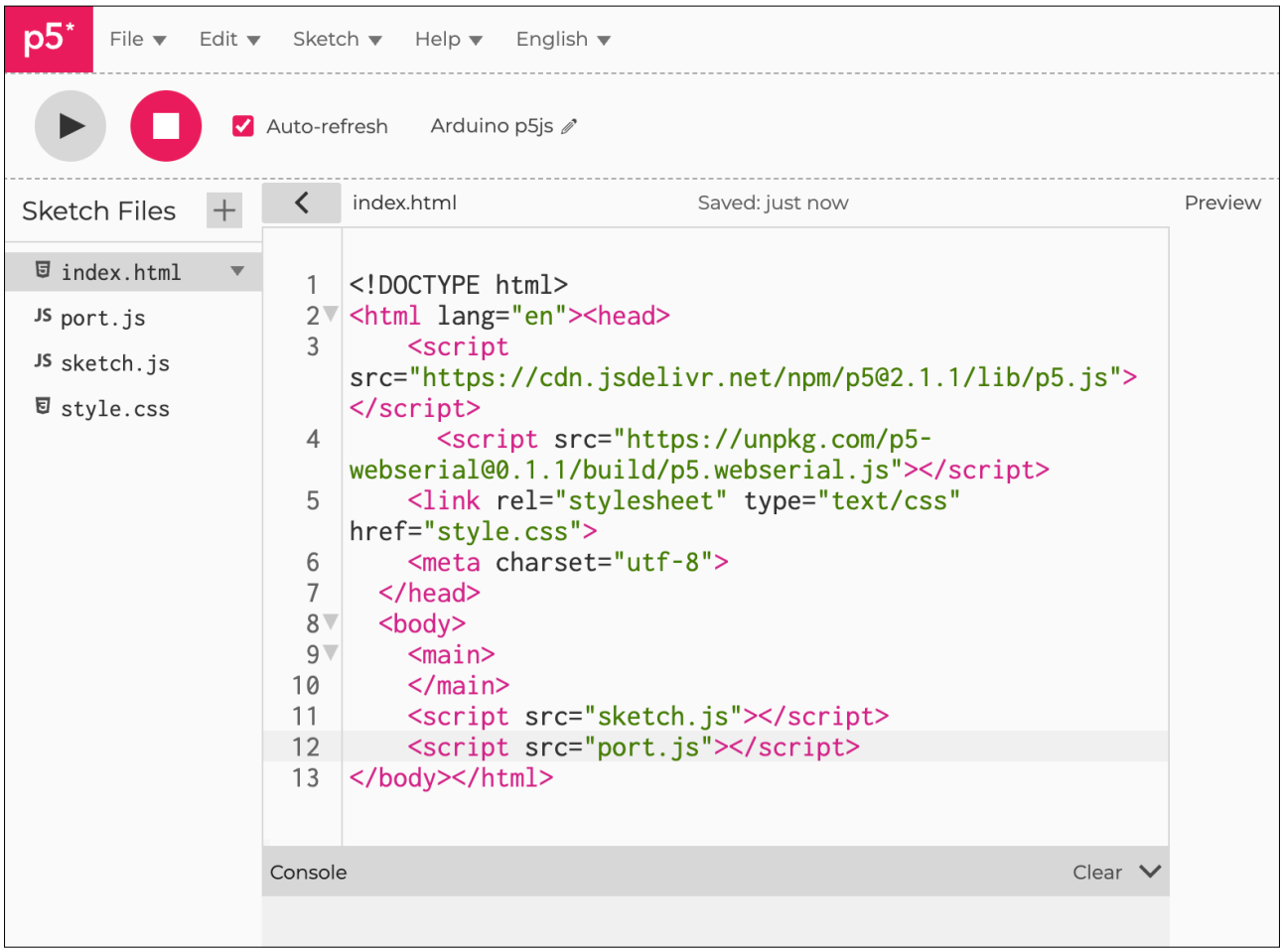
Creating the `port.js` file as we have done for other files previously.

Figure 1: port.js file



Adding the webserial library and the port.js file to the index.html file.

Figure 2: webserial and port.js to index.html



Sketch B2.2 index html

In the index.html file, we need to add the reference to include the webserial. This will allow us to connect the website with the port we want to use when connected to our Arduino.

index.html

```
<!DOCTYPE html>
<html lang="en"><head>
    <script src="https://cdn.jsdelivr.net/npm/p5@2.1.1/lib/p5.js"></script>
      <script src="https://unpkg.com/p5-webserial@0.1.1/build/
p5.webserial.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8">
  </head>
  <body>
    <main>
    </main>
    <script src="sketch.js"></script>
    <script src="port.js"></script>
</body></html>
```

Notes

It will look like figure 2.

Sketch B2.3 function navigation()

! port.js

The main function holding all the others together is the navigation() function. The first challenge is to check whether the browser is suitable for this.

port.js

```
function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}
```

Notes

There are lots of elements to this. In one sense, you really don't need to know the details, but being aware is what coding is all about.

Code Explanation

`if (!navigator.serial)`	Checks to see if the browser is compatible
`alert ("WebSerial is not supported in this browser. Try Chrome")`	If not then it sends a message to you that you need to consider a different browser
`serial.getPorts()`	Gets a list of available ports it has permission to use
`serial.on("noport", makePortButton)`	We go to the make a button function when we haven't connected with a port yet
`serial.on("portavailable", openPort)`	If a port is now connected start the process of sending/receiving data
`serial.on("requesterror", portError)`	Means there might be a problem or error of some kind
`serial.on("close", makePortButton)`	This is when the port closes for any reason, intentional or in error

Sketch B2.4 making a button

When you start off, you will need to choose a port. We have a button that we click on to take us to the port menu, and from there, select the port the Arduino is on.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}
```

Notes

You can position the button anywhere you like. When you click on the button, it will take you to another function called `choosePort()`, which we haven't created yet.

Code Explanation

`portButton = createButton("choose port")`	Creating a button
`portButton.position(10, 10)`	Putting it somewhere other than the default
`portButton.mousePressed(choosePort)`	When clicked the button action takes us to a function called choosePort

Sketch B2.5 choosing a port

There is a useful function that shows the available ports.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}
```

Notes

This operation gives you a drop-down menu (later) for available ports; you will just have to find the one that the Arduino is connected to.

Code Explanation

`serial.requestPort()`	Gives a selection of available ports

Sketch B2.6 port available

This opens the port that is connected to the Arduino.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
  {
```

```
    console.log("port open")
    if (portButton) portButton.hide()
  }
}
```

Code Explanation

`serial.open().then(initiateSerial)`	The connection has successfully been made to the microcontroller

Sketch B2.7 error report

If there is a problem with the connection or the port, an error message is displayed.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
  {
```

```
    console.log("port open")
    if (portButton) portButton.hide()
  }
}
```

```
function portError(err)
{
  alert("Serial port error: " + err)
}
```

Notes

Helps to debug.

Code Explanation

`alert("Serial port error: " + err)`	This puts a message on the screen

Sketch B2.8 confirmation

This lets you know all is well.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
  {
```

```
    console.log("port open")
    if (portButton) portButton.hide()
  }
}

function portError(err)
{
  alert("Serial port error: " + err)
}
```

```
function portConnect()
{
  console.log("port connected")
  serial.getPorts()
}
```

Sketch B2.9 disconnected

Fairly obvious, it lets you know when it has disconnected from the port.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
  {
```

```
    console.log("port open")
    if (portButton) portButton.hide()
  }
}

function portError(err)
{
  alert("Serial port error: " + err)
}

function portConnect()
{
  console.log("port connected")
  serial.getPorts()
}
```

```
function portDisconnect()
{
  serial.close()
  console.log("port disconnected")
}
```

Code Explanation

`serial.close()`	This stops the communication with the port

Sketch B2.10 closing the port

Does the same function as above but without the messaging.

port.js

```
const serial = new p5.WebSerial()
let portButton

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
  {
```

```
    console.log("port open")
    if (portButton) portButton.hide()
  }
}

function portError(err)
{
  alert("Serial port error: " + err)
}

function portConnect()
{
  console.log("port connected")
  serial.getPorts()
}

function portDisconnect()
{
  serial.close()
  console.log("port disconnected")
}

function closePort()
{
  serial.close()
}
```

Let's test it out

Now we have the opening of the port code working (hopefully), and you have connected and uploaded your Arduino code, we can start to interact with it. All we are going to do is click the mouse, and it will switch the LED `on` or `off`.

For dramatic effect, we will have a circle change colour accordingly.

Sketch B2.11 starting sketch

! Now in the sketch.js file,
Sticking a circle in the middle of the canvas.

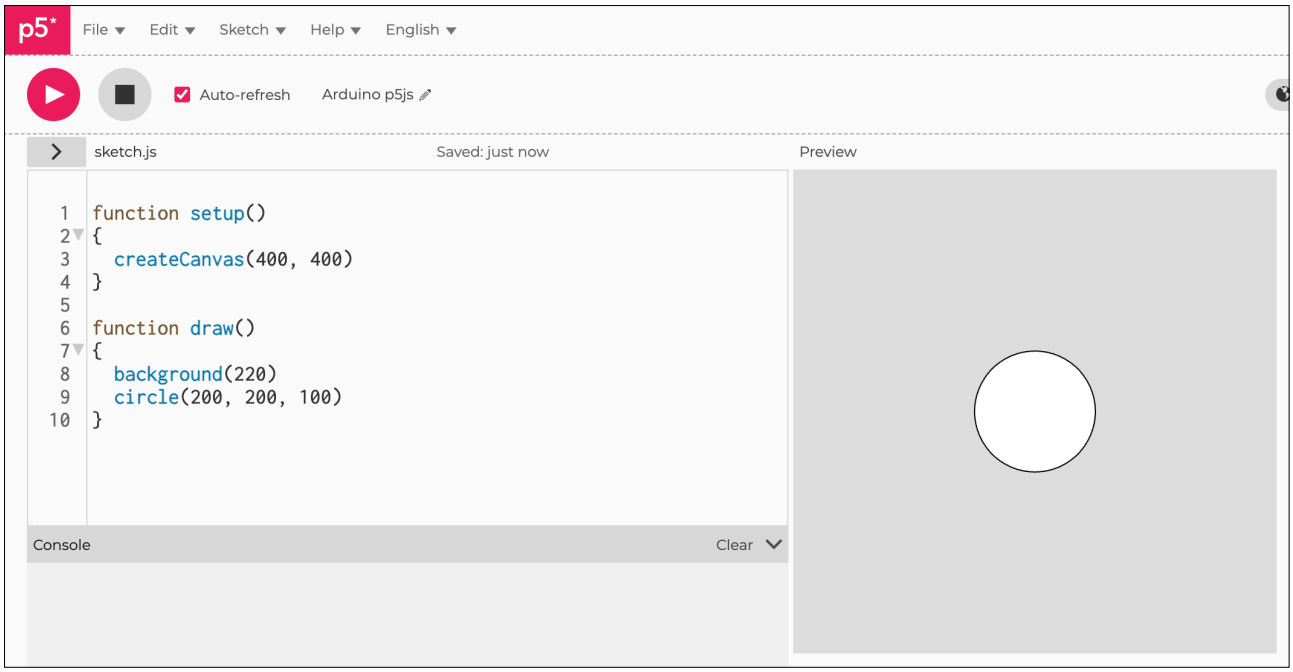
sketch.js

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

function draw()
{
  background(220)
  circle(200, 200, 100)
}
```

Notes

Just the usual.

Figure B2.11



Sketch B2.12 LED on/off

Now, when you click on the canvas, the LED is on; when you release the mouse button, the LED is `off`. We have also included the `navigation()` function in setup to activate the serial communication.

sketch.js

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

function draw()
{
  background(220)
  if (mouseIsPressed)
  {
    serial.write("HIGH\n")
  }
  else
  {
    serial.write("LOW\n")
  }
  circle(200, 200, 100)
}
```

Notes

The `\n` means new line, which corresponds with the Arduino code that means the command has finished.

Code Explanation

`serial.write("HIGH\n")`	Sends data followed by a new line command

Sketch B2.13 the circle response

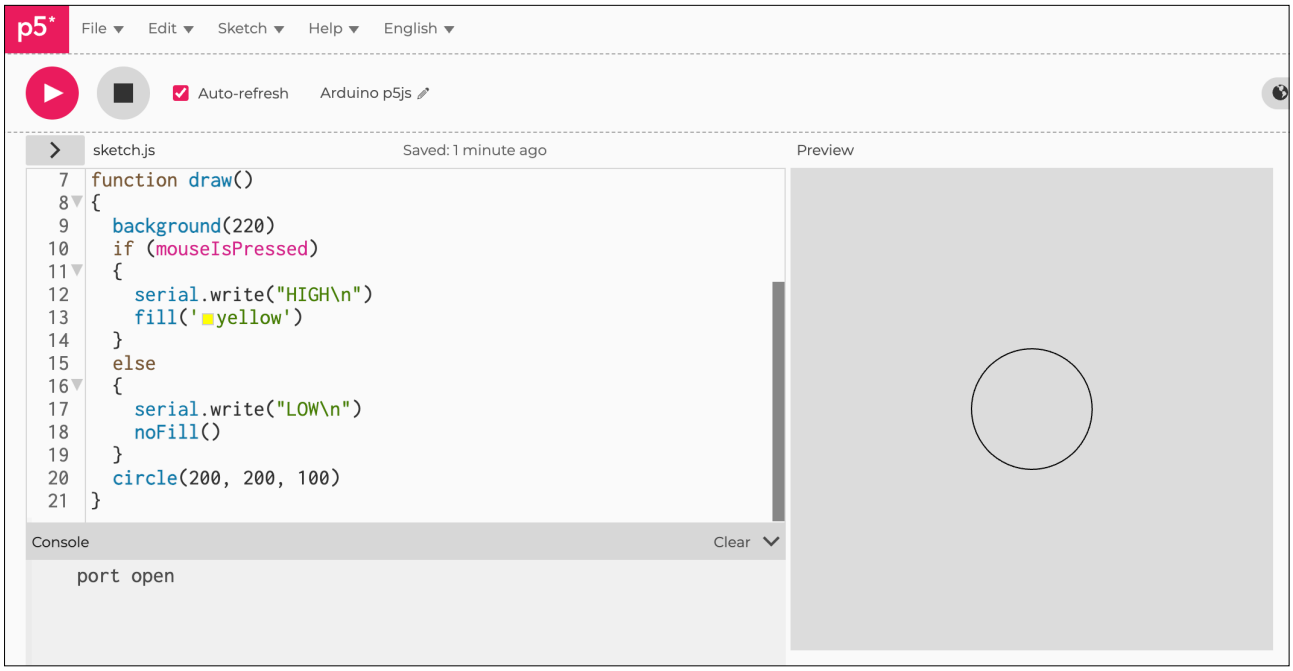
Now when you click on the canvas, the LED goes on/off, and the circle changes from grey to yellow.

sketch.js

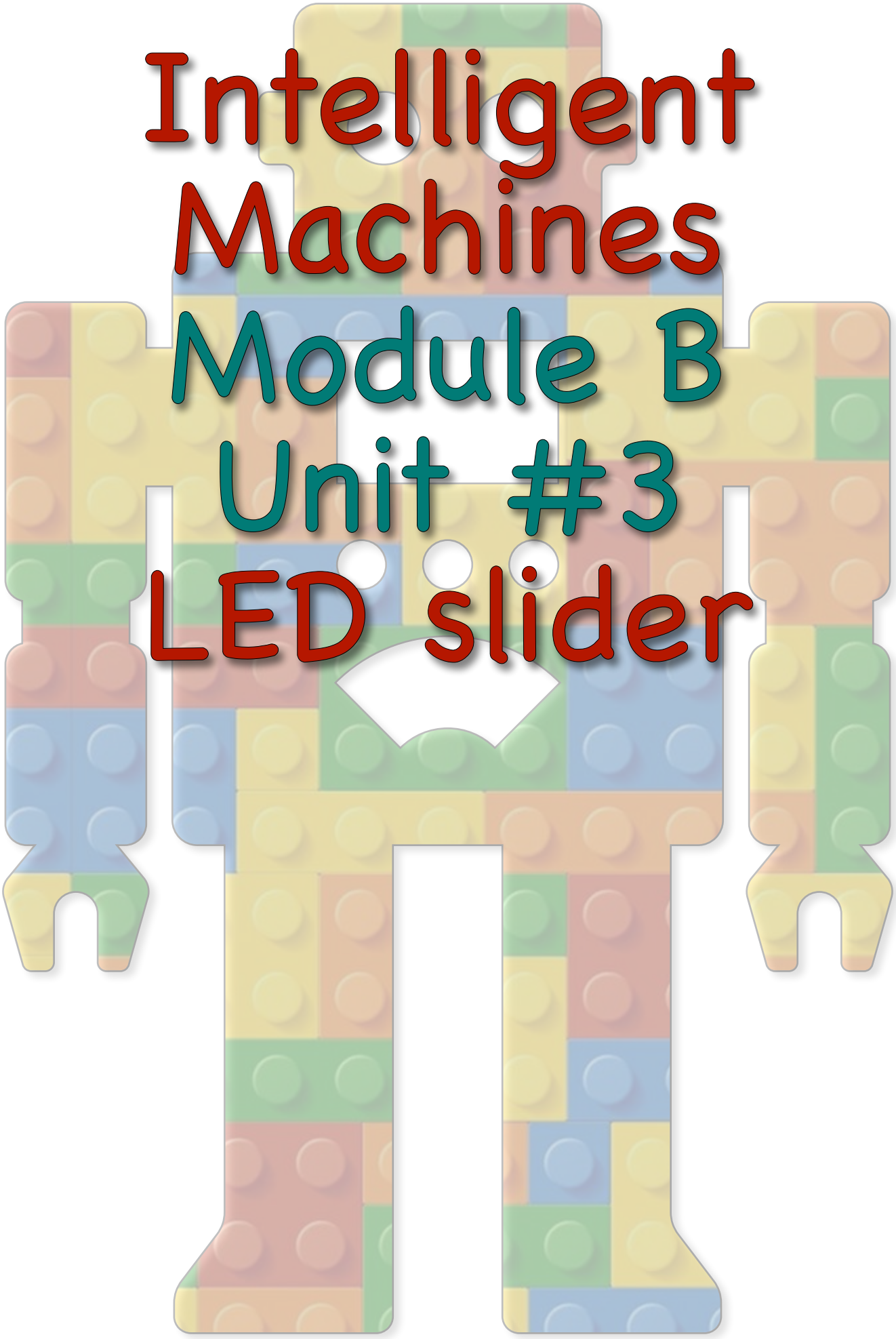
```
function setup()
{
  createCanvas(400, 400)
  navigation()
}

function draw()
{
  background(220)
  if (mouseIsPressed)
  {
    serial.write("HIGH\n")
    fill('yellow')
  }
  else
  {
    serial.write("LOW\n")
    noFill()
  }
  circle(200, 200, 100)
}
```

Figure B2.13a: mouse pressed

Figure B2.13b: mouse not pressed



Intelligent Machines
Module B
Unit #3
LED slider



Module B Unit #3: LED slider

In this example, we are going to alter the brightness of the LED using a slider in p5.js.

The `index.html` file remains unaltered.
The `port.js` file remains unaltered.

Sketch B3.1 receiving a value

! Our Arduino sketch

Remove everything in the `Serial.available()` loop and replace it with the code highlighted below. We are going to be sending an integer value from `p5.js` to the Arduino, and it will alter the brightness of the LED. To receive the value, we create an integer called `val`.

Arduino

```
const int ledPin = 13;
int val = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    val = Serial.read();
    analogWrite(ledPin, val);
  }
}
```

Notes

Now upload the sketch.

Code Explanation

`int val = 0;`	Creating the integer variable
`val = Serial.read();`	Reading the integer value form the p5.js sketch
`analogWrite(ledPin, val);`	Write that value to the LED

Sketch B3.2 starting sketch

❗ The main sketch.js
We include the navigation() function in our basic sketch.

sketch.js

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

function draw()
{
  background(220)
}
```

Notes

We need to include the navigation() function each time.

 Sketch B3.3 slider

Adding the slider.

sketch.js

```
let slider

function setup()
{
  createCanvas(400, 400)
  navigation()
  slider = createSlider(0, 255, 100)
}

function draw()
{
  background(220)
}
```

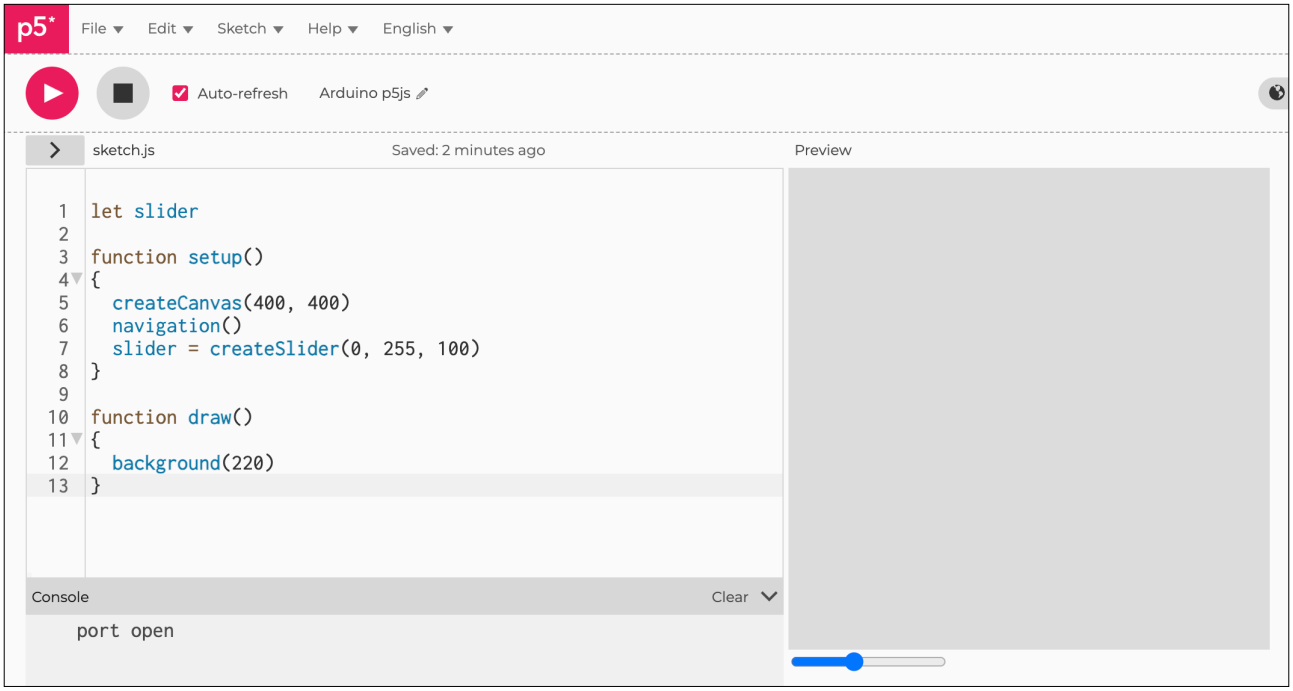
Notes

This just pops the slider on the bottom of the canvas.

Code Explanation

`let slider`	Naming the slider object
`slider = createSlider(0, 255, 100)`	Creating a slider with values from 0-255 with a default setting of 100

Figure B3.3


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Arduino p5js
sketch.js
Saved: 2 minutes ago
Preview
let slider
function setup()
{
createCanvas(400, 400)
navigation()
slider = createSlider(0, 255, 100)
}
function draw()
{
background(220)
}
Console
Clear
port open

Sketch B3.4 slider value

We take the value of the slider and send it (via serial) to the Arduino.

sketch.js

```
let slider
let val

function setup()
{
  createCanvas(400, 400)
  navigation()
  slider = createSlider(0, 255, 100)
}

function draw()
{
  background(220)
  val = slider.value()
  serial.write(val)
}
```

Notes

If you try this now, assuming you have connected successfully to the Arduino through a port, the LED on the board should correspond in brightness with the slider.

Code Explanation

`let val`	The val variable
`val = slider.value()`	Read the value of the slider
`serial.write(val)`	Send the value of val to the serial port

Sketch B3.5 visual

Just to add a bit of visualisation, we fill a circle according to the value and write the value on the canvas for good measure.

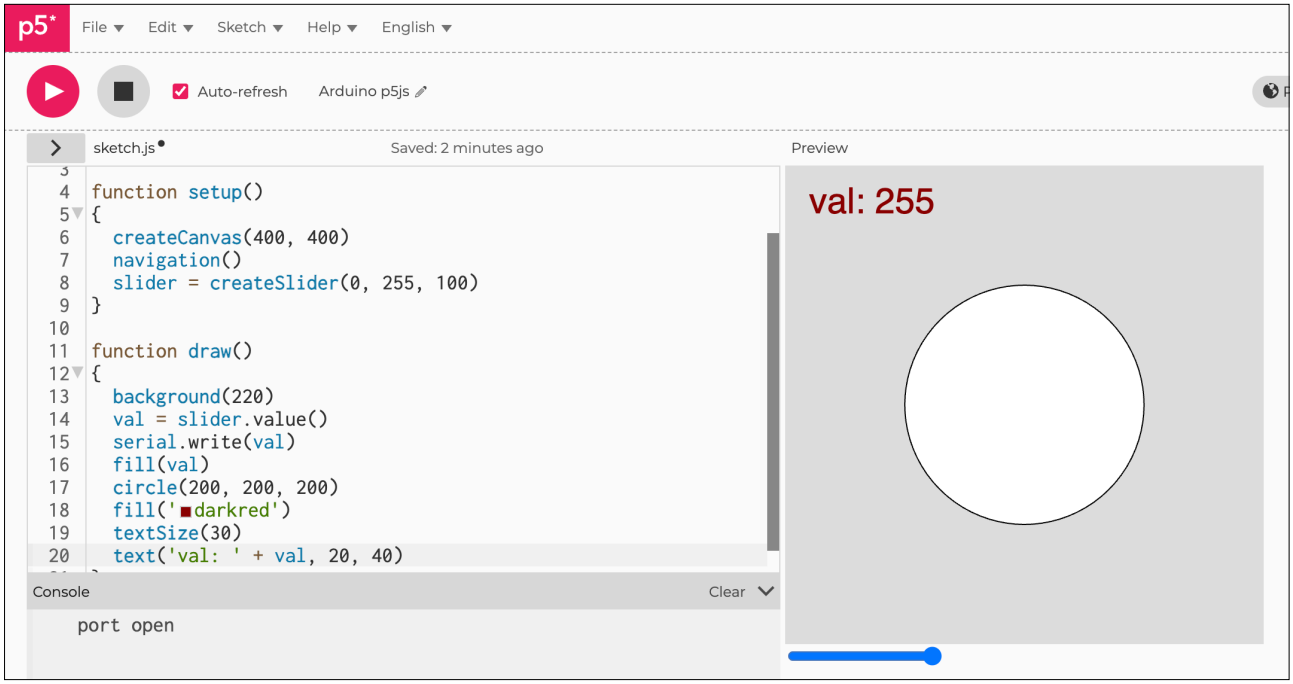
sketch.js

```
let slider
let val

function setup()
{
  createCanvas(400, 400)
  navigation()
  slider = createSlider(0, 255, 100)
}

function draw()
{
  background(220)
  val = slider.value()
  serial.write(val)
  fill(val)
  circle(200, 200, 200)
  fill('darkred')
  textSize(30)
  text('val: ' + val, 20, 40)
}
```

Notes

Adding a circle and text to the p5.js sketch.

Figure B3.5

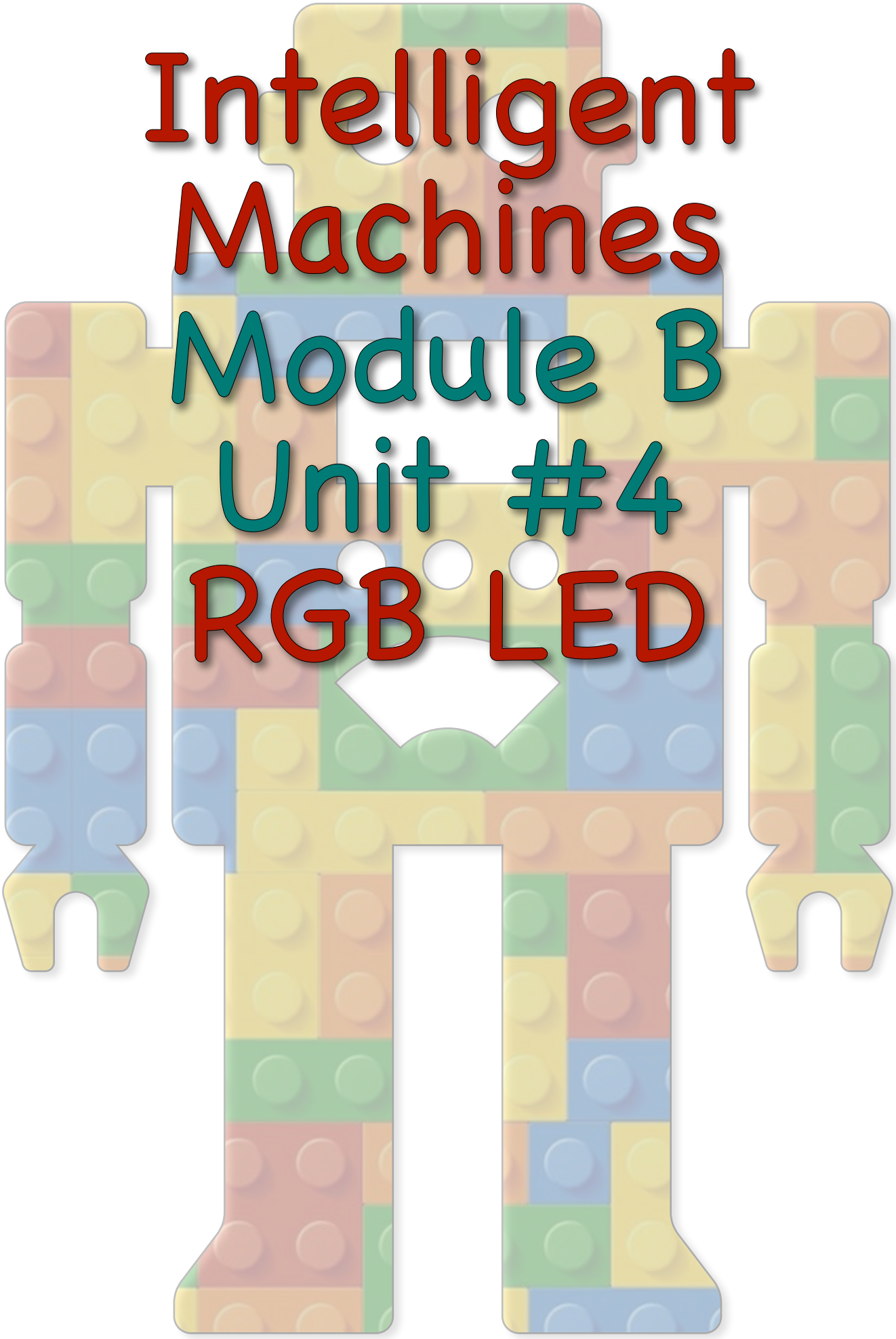

p5*
File
Edit
Sketch
Help
English
Auto-refresh
Arduino p5js
sketch.js
Saved: 2 minutes ago
Preview
function setup()
{
createCanvas(400, 400)
navigation()
slider = createSlider(0, 255, 100)
}
function draw()
{
background(220)
val = slider.value()
serial.write(val)
fill(val)
circle(200, 200, 200)
fill('darkred')
textSize(30)
text('val: ' + val, 20, 40)
Console
Clear
port open
val: 255

Intelligent Machines

Module B

Unit #4

RGB LED



 Module A Unit #4: RGB LEDs

We are going to light up the RGB LED as the mouse passes over (hovers) over a circle with that colour in it. The port.js sketch remains the same.

Sketch B4.1 starting again

! The Arduino sketch

We will start with a basic sketch initialising the three RGB LED colours. Basically, removing everything inside the `if (Serial.available() > 0)` loop.

Arduino sketch

```
const int ledPinRed = LEDR;
const int ledPinGreen = LEDG;
const int ledPinBlue = LEDB;

void setup()
{
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinBlue, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {

  }
}
```

Notes

Setting up all three built-in RGB LED.

Sketch B4.2 command

We create the variable command (as a string variable). Read all the data coming in as a string until we get a new line. We will be sent a string from p5.js: RED, BLUE, or GREEN.

Arduino sketch

```
const int ledPinRed = LEDR;
const int ledPinGreen = LEDG;
const int ledPinBlue = LEDB;

void setup()
{
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinBlue, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    String command = Serial.readStringUntil('\n');
    command.trim();
  }
}
```

Notes

Trim off any spaces.

Code Explanation

`String command = Serial.readStringUntil('\n');`	Receives a word not a number
`command.trim();`	Removes any spaces

Sketch B4.3 first the red

If we receive the string RED (text), then we switch the red LED on.

Arduino

```
const int ledPinRed = LEDR;
const int ledPinGreen = LEDG;
const int ledPinBlue = LEDB;

void setup()
{
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinBlue, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "RED")
    {
      digitalWrite(ledPinRed, LOW);
    }
  }
}
```

Notes

Remember that for the RGB LED, the logic is reversed: LOW is on! When it reads the word RED, it switches the LED on.

Sketch B4.4 and for the other colours

Same with the GREEN and BLUE.

Sketch

```
const int ledPinRed = LEDR;
const int ledPinGreen = LEDG;
const int ledPinBlue = LEDB;

void setup()
{
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinBlue, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "RED")
    {
      digitalWrite(ledPinRed, LOW);
    }
    if (command == "GREEN")
    {
      digitalWrite(ledPinGreen, LOW);
    }
    if (command == "BLUE")
    {
      digitalWrite(ledPinBlue, LOW);
    }
  }
}
```

Notes

The same for the other colours.

Sketch B4.5 everything off

If no string is sent, then we do not want any colour from the RGB LED on.

Sketch

```
const int ledPinRed = LEDR;
const int ledPinGreen = LEDG;
const int ledPinBlue = LEDB;

void setup()
{
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinBlue, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "RED")
    {
      digitalWrite(ledPinRed, LOW);
    }
    if (command == "GREEN")
    {
      digitalWrite(ledPinGreen, LOW);
    }
    if (command == "BLUE")
    {
      digitalWrite(ledPinBlue, LOW);
    }
    else if (command == "HIGH")
    {
      digitalWrite(ledPinRed, HIGH);
      digitalWrite(ledPinGreen, HIGH);
```

```
      digitalWrite(ledPinBlue, HIGH);
    }
  }
}
```

Notes

For the final condition, use the `else if()`. If you upload now, you might get all three `on` (white) at the same time.

Sketch B4.6 starting sketch

! The main sketch.js

We will draw three circles and colour them according to the RGB LEDs.

sketch.js

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

function draw()
{
  background(220)
  fill('red')
  circle(150, 100, 80)
  fill('green')
  circle(150, 200, 80)
  fill('blue')
  circle(150, 300, 80)
}
```

Notes

This should be fairly self-explanatory.

Figure B4.6


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Arduino p5js
sketch.js
Saved: 1 minute ago
Preview
function setup()
{
createCanvas(400, 400)
navigation()
}
function draw()
{
background(220)
fill('red')
circle(150, 100, 80)
fill('green')
circle(150, 200, 80)
fill('blue')
circle(150, 300, 80)
}
Console
Clear
port open

Sketch B4.7 mouse distance

We need to calculate the distance of the mouse to the centre of each circle.

sketch.js

```
let redDistance
let greenDistance
let blueDistance

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

function draw()
{
  background(220)
  fill('red')
  circle(150, 100, 80)
  fill('green')
  circle(150, 200, 80)
  fill('blue')
  circle(150, 300, 80)
  redDistance = dist(mouseX, mouseY, 150, 100)
  greenDistance = dist(mouseX, mouseY, 150, 200)
  blueDistance = dist(mouseX, mouseY, 150, 300)
}
```

Notes

Nothing new to see. The `dist()` function measures the distance between two sets of co-ordinates.

Sketch B4.8 check the distance

Now that we have calculated the distance between the mouse and the centre of each circle, we need to do something with that information. We will send a string when the mouse is inside the circle.

sketch.js

```
let redDistance
let greenDistance
let blueDistance

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

function draw()
{
  background(220)
  fill('red')
  circle(150, 100, 80)
  fill('green')
  circle(150, 200, 80)
  fill('blue')
  circle(150, 300, 80)
  redDistance = dist(mouseX, mouseY, 150, 100)
  greenDistance = dist(mouseX, mouseY, 150, 200)
  blueDistance = dist(mouseX, mouseY, 150, 300)
  if (redDistance <= 40)
  {
    serial.write("RED\n")
  }
}
```

Notes

If you run this sketch, you will see the red part of the LED come on once when the mouse is inside the circle; however, it stays on when the mouse moves outside the circle. We will fix that issue in due course.

Sketch B4.9 and the other colours

Now we will add the other colours. But all that we get is white as each colour adds to the mix.

sketch.js

```
let redDistance
let greenDistance
let blueDistance

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

function draw()
{
  background(220)
  fill('red')
  circle(150, 100, 80)
  fill('green')
  circle(150, 200, 80)
  fill('blue')
  circle(150, 300, 80)
  redDistance = dist(mouseX, mouseY, 150, 100)
  greenDistance = dist(mouseX, mouseY, 150, 200)
  blueDistance = dist(mouseX, mouseY, 150, 300)
  if (redDistance <= 40)
  {
    serial.write("RED\n")
  }
  else if (greenDistance <= 40)
  {
    serial.write("GREEN\n")
  }
  else if (blueDistance <= 40)
  {
    serial.write("BLUE\n")
```

```
  }
}
```

Notes

The commands RED, GREEN, and BLUE are sent to the Arduino.

Sketch B4.10 the off switch

We need the RGB LED to reset, so that we can see the colours individually. Also, we need them to be off altogether when not hovering over a circle.

sketch.js

```
let redDistance
let greenDistance
let blueDistance

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

function draw()
{
  background(220)
  fill('red')
  circle(150, 100, 80)
  fill('green')
  circle(150, 200, 80)
  fill('blue')
  circle(150, 300, 80)
  redDistance = dist(mouseX, mouseY, 150, 100)
  greenDistance = dist(mouseX, mouseY, 150, 200)
  blueDistance = dist(mouseX, mouseY, 150, 300)
  if (redDistance <= 40)
  {
    serial.write("RED\n")
  }
  else if (greenDistance <= 40)
  {
    serial.write("GREEN\n")
  }
  else if (blueDistance <= 40)
  {
    serial.write("BLUE\n")
```

```
  }
  else
  {
    serial.write("HIGH\n")
  }
}
```

Notes

Now it should work perfectly. The RGB LED should display the colour of the circle you hover over, and when you are not over any circle, then nothing should be on.

Sketch B4.11 a bit of a tweak

Just to finish off, let's add a circle that will display the colour of the RGB LED that is on. Just for good measure.

sketch.js

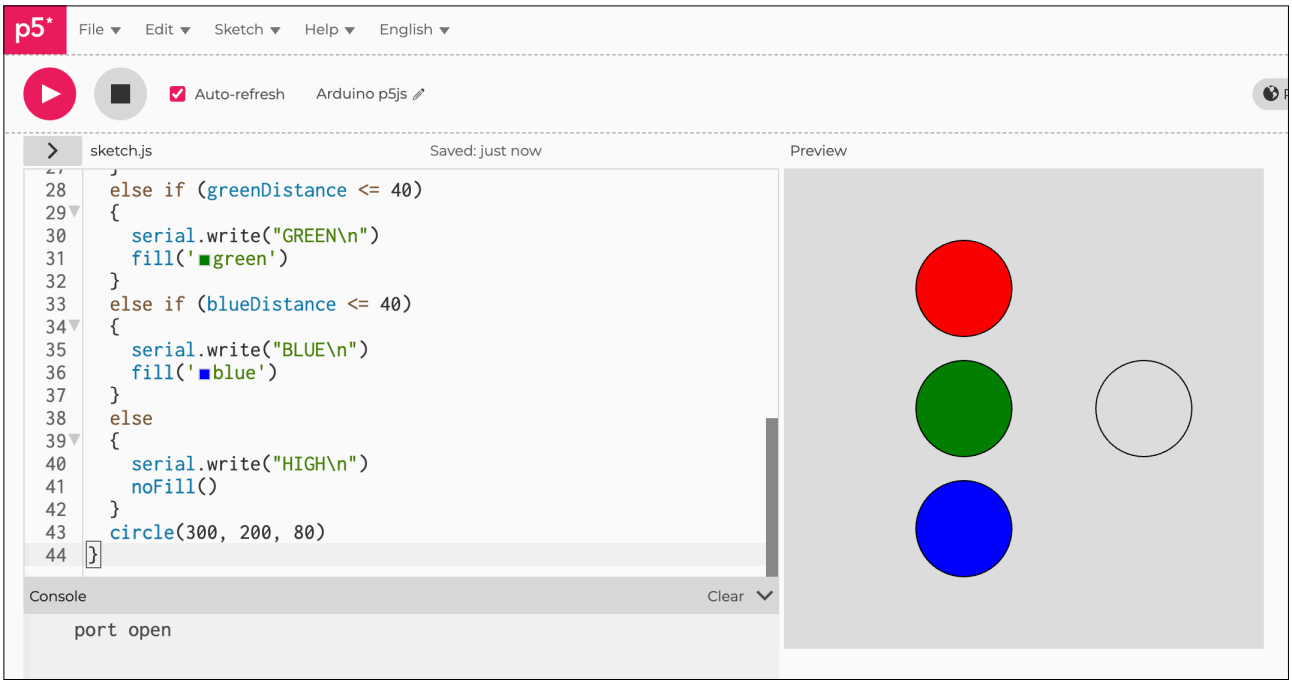
```
let redDistance
let greenDistance
let blueDistance

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

function draw()
{
  background(220)
  fill('red')
  circle(150, 100, 80)
  fill('green')
  circle(150, 200, 80)
  fill('blue')
  circle(150, 300, 80)
  redDistance = dist(mouseX, mouseY, 150, 100)
  greenDistance = dist(mouseX, mouseY, 150, 200)
  blueDistance = dist(mouseX, mouseY, 150, 300)
  if (redDistance <= 40)
  {
    serial.write("RED\n")
    fill('red')
  }
  else if (greenDistance <= 40)
  {
    serial.write("GREEN\n")
    fill('green')
  }
  else if (blueDistance <= 40)
```

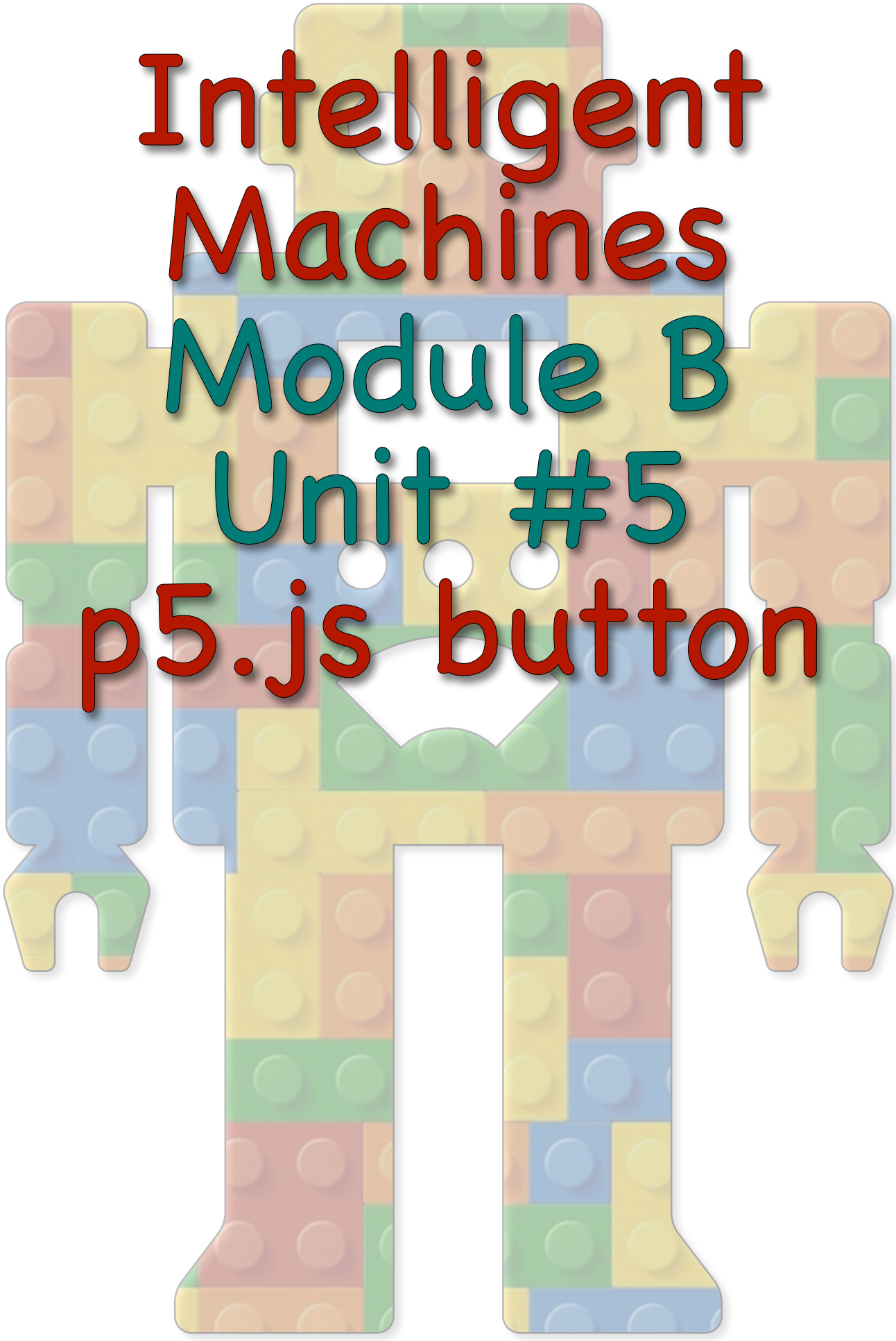
```
  {
    serial.write("BLUE\n")
    fill('blue')
  }
  else
  {
    serial.write("HIGH\n")
    noFill()
  }
  circle(300, 200, 80)
}
```

Notes

Now we have an indicator.

Figure B4.11





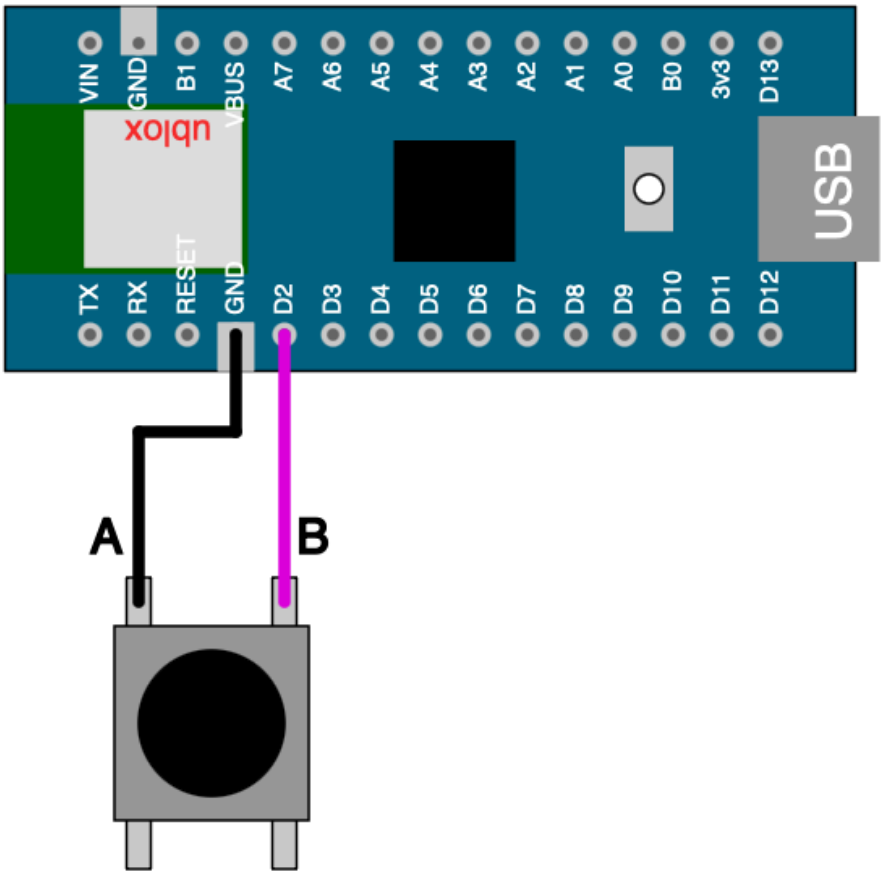
Intelligent Machines
Module B
Unit #5
p5.js button

Module A Unit #5: p5.js and a button

As we have covered in a previous unit, a button is connected to pin D2. Here we are going to send data from the Arduino to the sketch. Here we will simply change the colour from black to white by pressing the button.

A reminder of the circuit diagram, on the same side of the button, one pin goes to ground and the other to digital pin D2 (next to it).

Figure 1: Circuit diagram



Sketch B5.1 button

! The Arduino sketch

We have added the button to digital pin 2. We called the variable buttonPin and used the pull-up resistor. We read that pin and print it to the serial monitor. We used that to control the colour of the circle.

Arduino sketch

```
const int buttonPin = 2;
int val;

void setup()
{
  Serial.begin(9600);
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop()
{
  val = digitalRead(buttonPin);
  Serial.println(val);
}
```

Notes

I have highlighted the main changes.

Code Explanation

`const int buttonPin = 2;`	Attaching the button to pin 2
`pinMode(buttonPin, INPUT_PULLUP);`	Using the internal resistor
`val = digitalRead(buttonPin);`	Reading the value of the button

Sketch B5.2 the port.js sketch

! A few changes to the port.js sketch

We have to add a serialEvent() function that will catch the data. We push the data to a variable called circleColour.

port.js

```
const serial = new p5.WebSerial()
let portButton
let inData
let circleColour

function navigation()
{
  if (!navigator.serial)
  {
    alert ("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("data", serialEvent)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
  serial.requestPort()
}
```

```
function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
  {
    console.log("port open")
    if (portButton) portButton.hide()
  }
}

function portError(err)
{
  alert("Serial port error: " + err)
}

function portConnect()
{
  console.log("port connected")
  serial.getPorts()
}

function portDisconnect()
{
  serial.close()
  console.log("port disconnected")
}

function closePort()
{
  serial.close()
}

function serialEvent()
{
  inData = serial.readStringUntil('\n')
  if (inData)
  {
```

```
    circleColour = 255 - inData * 255
  }
}
```

Notes

This is necessary because we are sending data to the p5.js sketch, whereas before we were sending data from the p5.js sketch.

Code Explanation

`let circleColour`	A variable to hold the colour of the circle
`serial.on("data", serialEvent)`	Call the serialEvent() function
`inData = serial.readStringUntil('\n')`	Keeps reading until there is a new line /n
`if (inData)`	If it detects any data
`circleColour = 255 - inData * 255`	The inData value will be either 1 or 0

Sketch B5.3 the sketch.js

! In sketch.js

All we are going to do with the data is fill a circle with the final colour value.

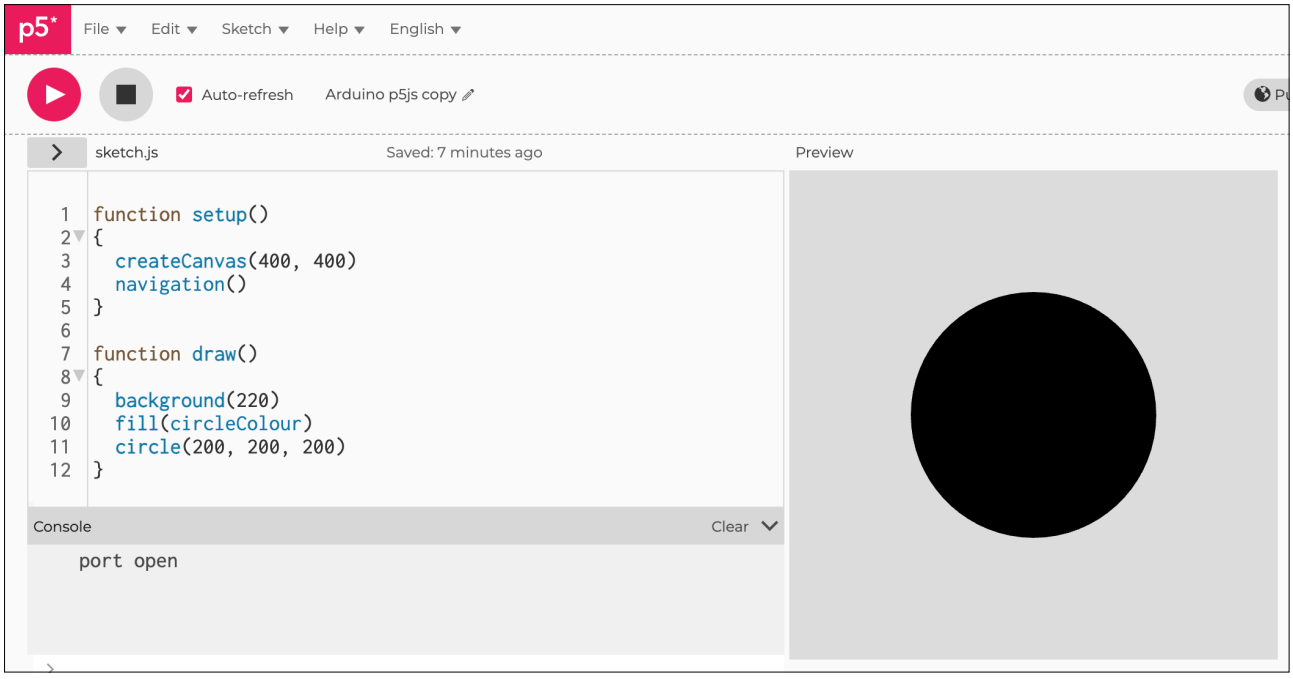
sketch.js

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

function draw()
{
  background(220)
  fill(circleColour)
  circle(200, 200, 200)
}
```

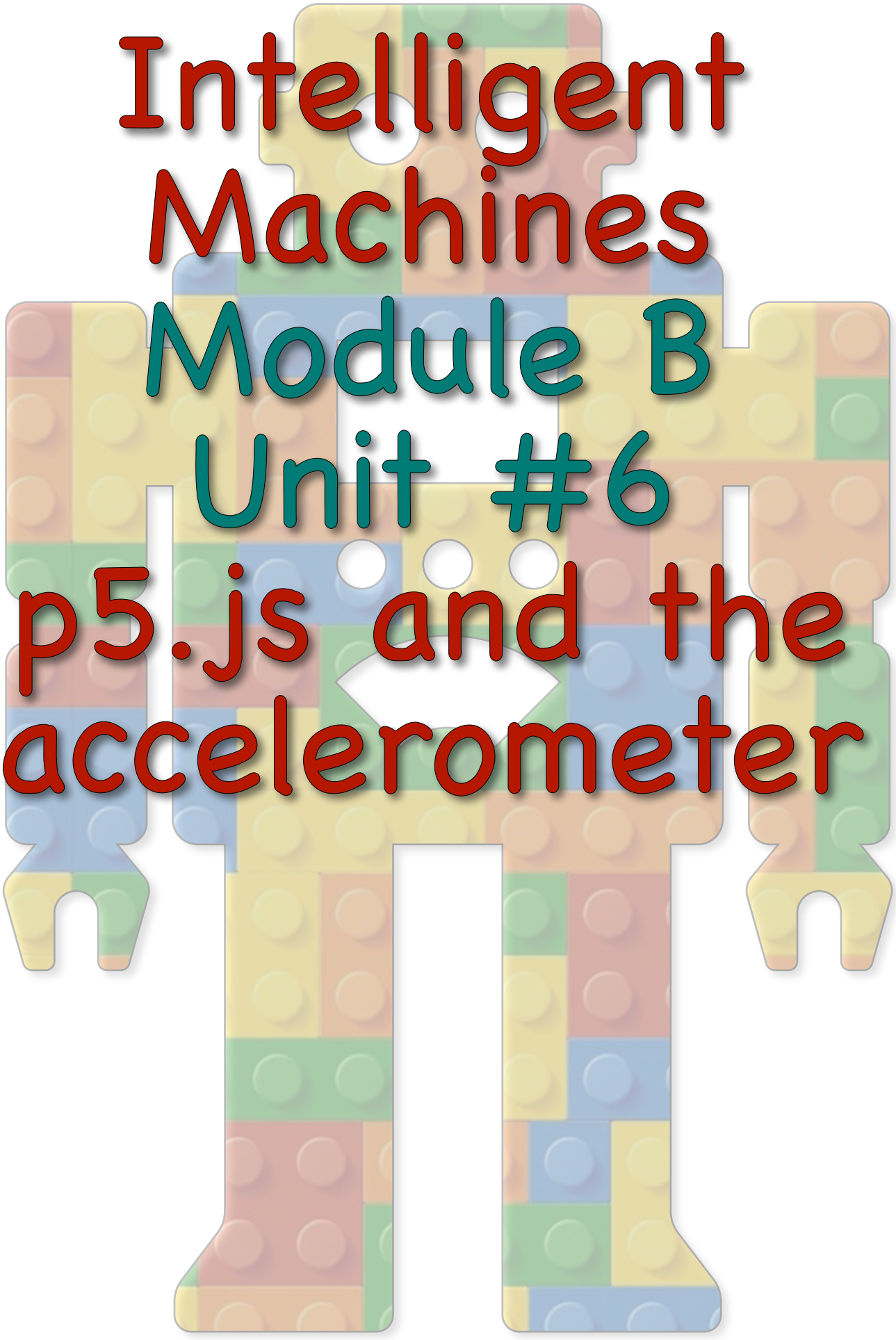
Notes

When you press the button, the circle goes from black to white.

Figure B5.3



Intelligent Machines
Module B
Unit #6
p5.js and the accelerometer



Module A Unit #6: p5.js and the accelerometer

We have three axes: the x, y, and z on the Nano, which we can replicate with p5.js using WebGL 3D rendering mode. One issue is that the x-axis and y-axis are self-explanatory. The z-axis is the tilt according to a vertical axis through the centre of the board.

On the p5.js WebGL rendering, we have a 3D environment, an x, y, and z plane. Here, we need to use the x-plane and the z-plane, not the y-plane. Not going to explain why in detail but just work it out in your head.

So we can rotate a 3D object in two planes. If we want to turn it (think heading), then we could use the gyroscope, but that could get a bit complicated because it measures rotational acceleration and not angle. There could be a way around that by having a reference angle and then keeping track of the angular movement.

Sketch B6.1 the accelerometer

! The Arduino sketch

Here we are sending the values of x, y, and z to p5.js. We will send all three, even though we will only use two (the x and y).

Arduino sketch

```
#include "Arduino_BMI270_BMM150.h"
float x;
float y;
float z;

void setup()
{
  Serial.begin(9600);
  if (!IMU.begin())
  {
    Serial.println("Failed to initialise IMU");
    while (true);
  }
}

void loop()
{
  if (IMU.accelerationAvailable())
  {
    IMU.readAcceleration(x, y, z);
  }
    Serial.print(x);
    Serial.print(",");
    Serial.print(y);
    Serial.print(",");
    Serial.println(z);
}
```

Notes

We have installed and included the library for the IMU used for v2. We grab the x, y, and z values (even though we don't use the z-axis values) and print them, which sends them to the serial monitor.

Sketch B6.2 tweaking port.js

❗ The port.js sketch

Instead of inData, we will have a string and call it inString. We create an empty array and call it list[].

port.js

```
const serial = new p5.WebSerial()
let portButton
let inString
let list = []
let x
let y
let z

function navigation()
{
  if (!navigator.serial)
  {
    alert("WebSerial is not supported in this browser. Try Chrome")
  }
  serial.getPorts()
  serial.on("noport", makePortButton)
  serial.on("portavailable", openPort)
  serial.on("requesterror", portError)
  serial.on("data", serialEvent)
  serial.on("close", makePortButton)
}

function makePortButton()
{
  portButton = createButton("choose port")
  portButton.position(10, 10)
  portButton.mousePressed(choosePort)
}

function choosePort()
{
  if (portButton) portButton.show()
```

```
  serial.requestPort()
}

function openPort()
{
  serial.open().then(initiateSerial)
  function initiateSerial()
  {
    console.log("port open")
    if (portButton) portButton.hide()
  }
}

function portError(err)
{
  alert("Serial port error: " + err)
}

function portConnect()
{
  console.log("port connected")
  serial.getPorts()
}

function portDisconnect()
{
  serial.close()
  console.log("port disconnected")
}

function closePort()
{
  serial.close()
}

function serialEvent()
{
  inString = serial.readStringUntil("\r\n")
```

```
  if (inString)
  {
    list = splitTokens(inString, ",")
    if (list.length > 2)
    {
      x = float(list[0])
      y = float(list[1])
      z = float(list[2])
    }
  }
}
```

Notes

This is a bit more complicated because we are drawing the data as strings from an array of elements separated by commas, and there are spaces which we don't need.

Code Explanation

`let list = []`	An array to hold the three values
`inString = serial.readStringUntil("\r\n")`	Read values until new line or carriage return
`if (inString)`	If there is data
`list = splitTokens(inString, ",")`	Separate the data according to a comma (,)
`if (list.length > 2)`	If there are more than two elements in the array, we want there to be three
`x = float(list[0])`	Returns the first element, the x value
`y = float(list[1])`	Returns the second element, the y value
`z = float(list[2])`	Returns the third element, the z value

 Sketch B6.3 the main sketch

❗ In the main sketch.js, we start with the usual addition of the navigation() function.

sketch.js

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

function draw()
{
  background(220)
}
```

Sketch B6.4 adding the 3D render (WEBGL)

We want a 3D render, and to do that, we add the reference when creating the canvas. We use WEBGL, which we simply add when we create the canvas.

sketch.js

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

function draw()
{
  background(220)
}
```

Notes

WEBGL has a different co-ordinate system; the origin is now in the centre of the canvas.

Code Explanation

`createCanvas(400, 400, WEBGL)`	This gives us the 3D renderer we need to draw in 3D

Sketch B6.5 drawing the Arduino Nano

The origin is now in the centre of the canvas. We want to draw a representation of the Arduino. We do this as a separate function called nano().

sketch.js

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

function draw()
{
  background(220)
  nano()
}

function nano()
{
  stroke(0, 90, 90)
  fill(0, 130, 130)
  box(300, 10, 120)

  stroke(0)
  fill(80)
  translate(30, -6, 0)
  box(60, 0, 60)

  stroke(80)
  fill(180)
  translate(80, 0, 0)
  box(60, 15, 60)

  translate(-245, 0, 0)
  box(35, 15, 40)
}
```

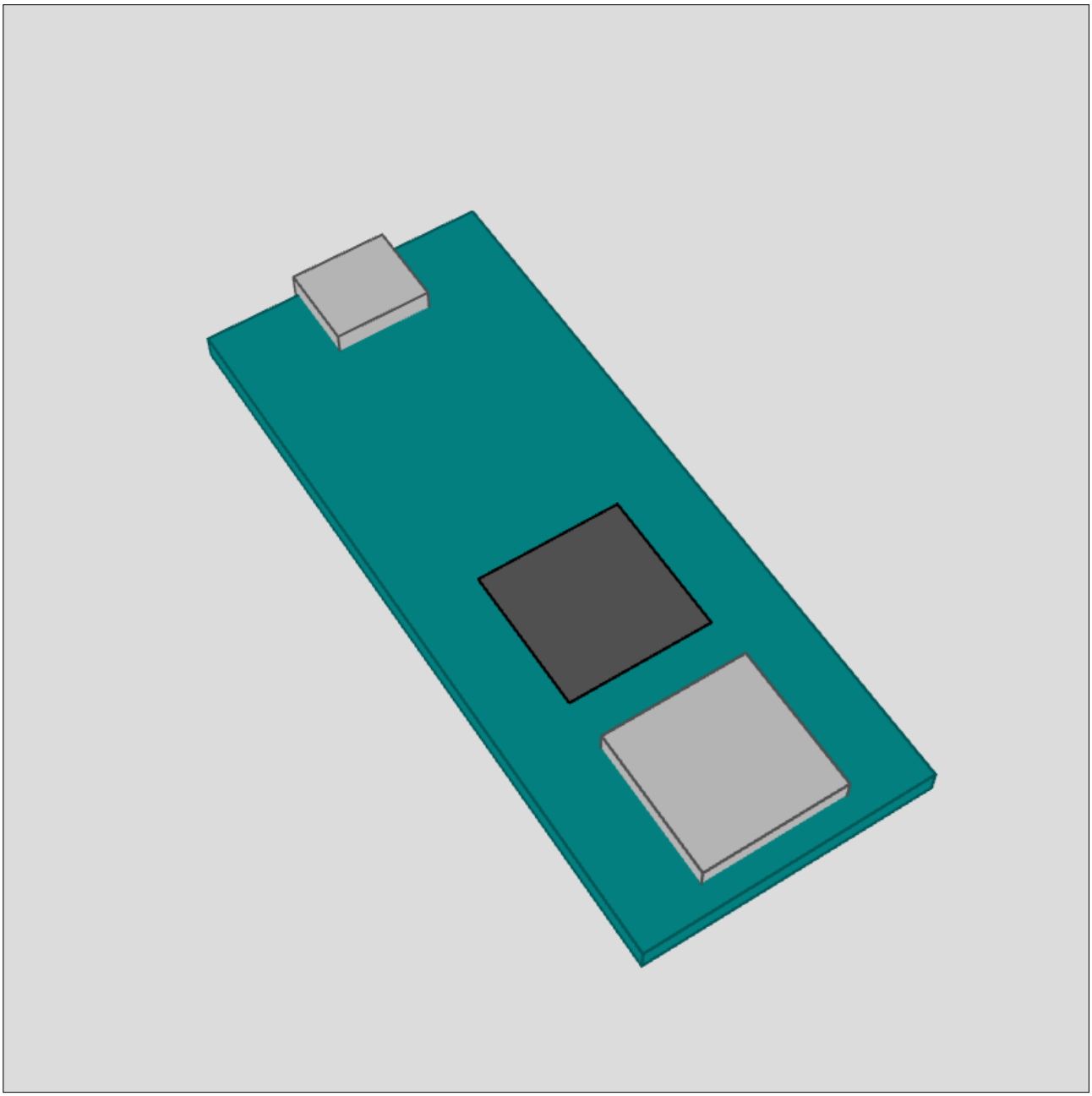
Notes

In the image below, I have rotated it so you can see the features; yours will be flat.

Challenge

Can you add more features?

Figure B6.4



Sketch B6.6 rotate

We take the x and y values from the IMU, rotate the x-axis about the z-plane and the y-axis about the x-plane. This seems very counterintuitive, but it provides the best correlation, in other words it just works.

sketch.js

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

function draw()
{
  background(220)
  rotateX(y)
  rotateZ(x)
  nano()
}

function nano()
{
  stroke(0, 90, 90)
  fill(0, 130, 130)
  box(300, 10, 120)

  stroke(0)
  fill(80)
  translate(30, -6, 0)
  box(60, 0, 60)

  stroke(80)
  fill(180)
  translate(80, 0, 0)
  box(60, 15, 60)

  translate(-245, 0, 0)
  box(35, 15, 40)
```

```
}
```

Notes

This, however, only uses the values between -1 and 1.

Sketch B6.7 mapping

The values are mapped to degrees (a bit more intuitive). For that, we need to have `angleMode(DEGREES)` included.

sketch.js

```
function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  navigation()
}

function draw()
{
  background(220)
  x = map(x, -1, 1, 90, -90)
  y = map(y, -1, 1, -90, 85)
  rotateX(y)
  rotateZ(x)
  nano()
}

function nano()
{
  stroke(0, 90, 90)
  fill(0, 130, 130)
  box(300, 10, 120)

  stroke(0)
  fill(80)
  translate(30, -6, 0)
  box(60, 0, 60)

  stroke(80)
  fill(180)
  translate(80, 0, 0)
  box(60, 15, 60)
```

```
  translate(-245, 0, 0)
  box(35, 15, 40)
}
```

Notes

I used 85 in my case so that it would lie flat.

Sketch B6.8 smooth

To smooth out the motion (make it a little less jittery), I have slowed the frame rate down to 10 frames per second.

sketch.js

```
function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  navigation()
  frameRate(10)
}

function draw()
{
  background(220)
  x = map(x, -1, 1, 90, -90)
  y = map(y, -1, 1, -90, 85)
  rotateX(y)
  rotateZ(x)
  nano()
}

function nano()
{
  stroke(0, 90, 90)
  fill(0, 130, 130)
  box(300, 10, 120)

  stroke(0)
  fill(80)
  translate(30, -6, 0)
  box(60, 0, 60)

  stroke(80)
  fill(180)
  translate(80, 0, 0)
  box(60, 15, 60)
```

```
  translate(-245, 0, 0)
  box(35, 15, 40)
}
```

Notes

It sort of works.

Challenge

Can you think of other ways you could smooth the movement?

Figure B6.8