Algorithmic Art Workbook #5 3D Shapes and Lights

Table of Contents

MIT Licence 5
Workbook #5 Quick Content Summary 6
Module C Unit #1: primitive 3D shapes 8
Sketch C1.1 standard sketch 9
Sketch C1.2 adding the WEBGL 10
Sketch C1.3 drawing a box 11
Sketch C1.4 rotating 13
Sketch C1.5 45° rotation 15
Sketch C1.6 the x and y axis 17
Sketch C1.7 incrementing the rotation 19
Sketch C1.8 drawing a cuboid 20
Sketch C1.9 drawing a plane 22
Sketch C1.10 drawing a cylinder 24
Sketch C1.11 drawing a sphere 26
Sketch C1.12 drawing an ellipsoid 28
Sketch C1.13 drawing a torus 30
Sketch C1.14 drawing a cone 32
Sketch C1.15 adding a bit of colour 34
Sketch C1.16 translate 36
Sketch C1.17 translate ‘tother way 38
Sketch C1.18 translate along the z axis 40
Sketch C1.19 translate the other way 42
Sketch C1.20 more than one shape 44
Sketch C1.21 pushing and popping 46
Module C Unit #2: orbiting and smoothing 49
Sketch C2.1 default WEBGL sketch 50
Sketch C2.2 default details 51
Sketch C2.3 smoothing the curves 53
Sketch C2.4 a single cube 55
Sketch C2.5 many cubes 56
Sketch C2.6 controlling the orbit 58
Sketch C2.7 frame count 60
Sketch C2.8 alternative frame count 62
Module C Unit #3: lights and materials #1 65
Sketch C3.1 draw a sphere 66
Sketch C3.2 a normal material 67
Sketch C3.3 a more ambient material 69
Sketch C3.4 an ambient light 71
Sketch C3.5 ambient blue 73

Sketch C3.6 a green light on a white material 75
Sketch C3.7 a point of light 77
Sketch C3.8 moving the light 79
Sketch C3.9 three points of light 81
Sketch C3.10 metallic finish 83
Sketch C3.11 more subtle 85
Sketch C3.12 box and rotate 87
Module C Unit #4: lights and materials #2 90
Sketch C4.1 new sketch new lights 91
Sketch C4.2 a more directional light 93
Sketch C4.3 just lights! 95
Sketch C4.4 basic torus 97
Sketch C4.5 more detail added 99
Sketch C4.6 orbiting object 101
Sketch C4.7 rotating directional light 103
Sketch C4.8 final trick up our sleeve 105
Module C Unit #5: graphics texture 108
Sketch C5.1 starting sketch 109
Sketch C5.2 applying texture 111
Sketch C5.3 separate canvas 113
Sketch C5.4 adding graphics 115
Sketch C5.5 drawing on the sides 117
Sketch C5.6 a bit of fun tweaking 119
Sketch C5.7 creating text 121
Sketch C5.8 words on a plane 123
Sketch C5.9 text on a cylinder 125
Module C Unit #6: the cube wave 128
Sketch C6.1 starting sketch 129
Sketch C6.2 rectangle 130
Sketch C6.3 breathing 132
Sketch C6.4 a row of rectangles 134
Sketch C6.5 wavy pattern 136
Sketch C6.6 incremental offset 138
Sketch C6.7 tidy up 140
Sketch C6.8 adding WEBGL 142
Sketch C6.9 adding a box 144
Sketch C6.10 rotate 146
Sketch C6.11 orthographic 148
Sketch C6.12 the magic number 150
Sketch C6.13 ortho() parameters 152
Sketch C6.14 ripples 154



MIT Licence

Copyright ©2026 Warren George

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Brief summary:

Permissions (what you can do)

Commercial use
Modification
Distribution
Private use

Limitations (what is not covered)

Liability
Warranty



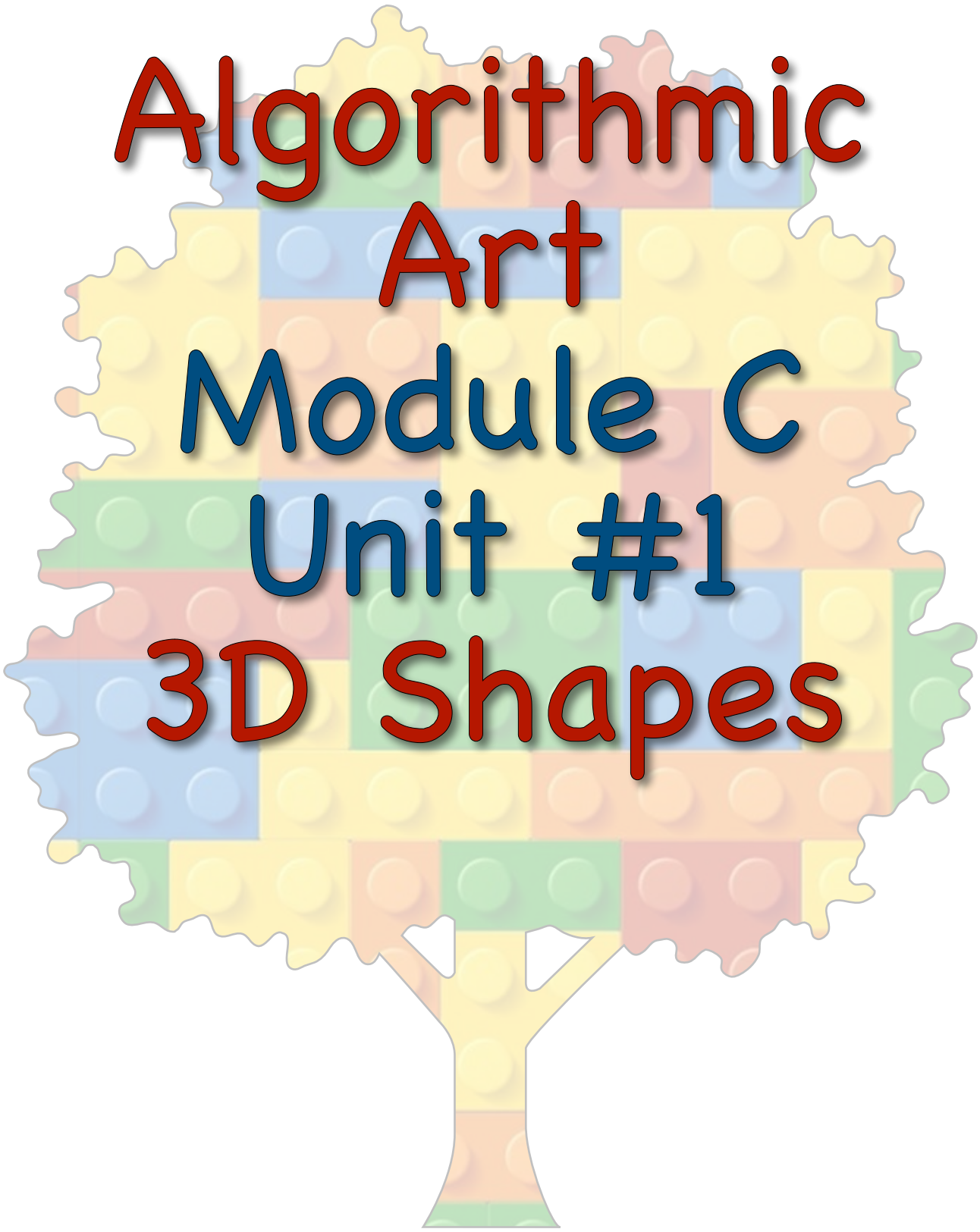
Workbook #5 Quick Content Summary

We have only been using 2D shapes until now. Here you will be introduced to 3D shapes. This is quite different from using 2D shapes, but the principles are just the same. We will explore not just the shapes available but also some materials and lighting effects.

The code is in the yellow boxes; any new code is highlighted in blue, so you don't have to type out the whole thing again each time. Set up an account (highly recommended) so you can save your work as you go along rather than starting again each time. It will also remind you where you were up to.

Most browsers work, but Chrome is best as there is some functionality missing in some browsers. I suspect it has been tested on Chrome to guarantee it works in the Chrome browser.

Stay with the 2.2.3 version for now (September 2026).



Algorithmic Art
Module C
Unit #1
3D Shapes

Module C Unit #1: primitive 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 C1.1 standard sketch

Starting with our standard sketch.

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

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



Sketch C1.2 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 C1.3 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.

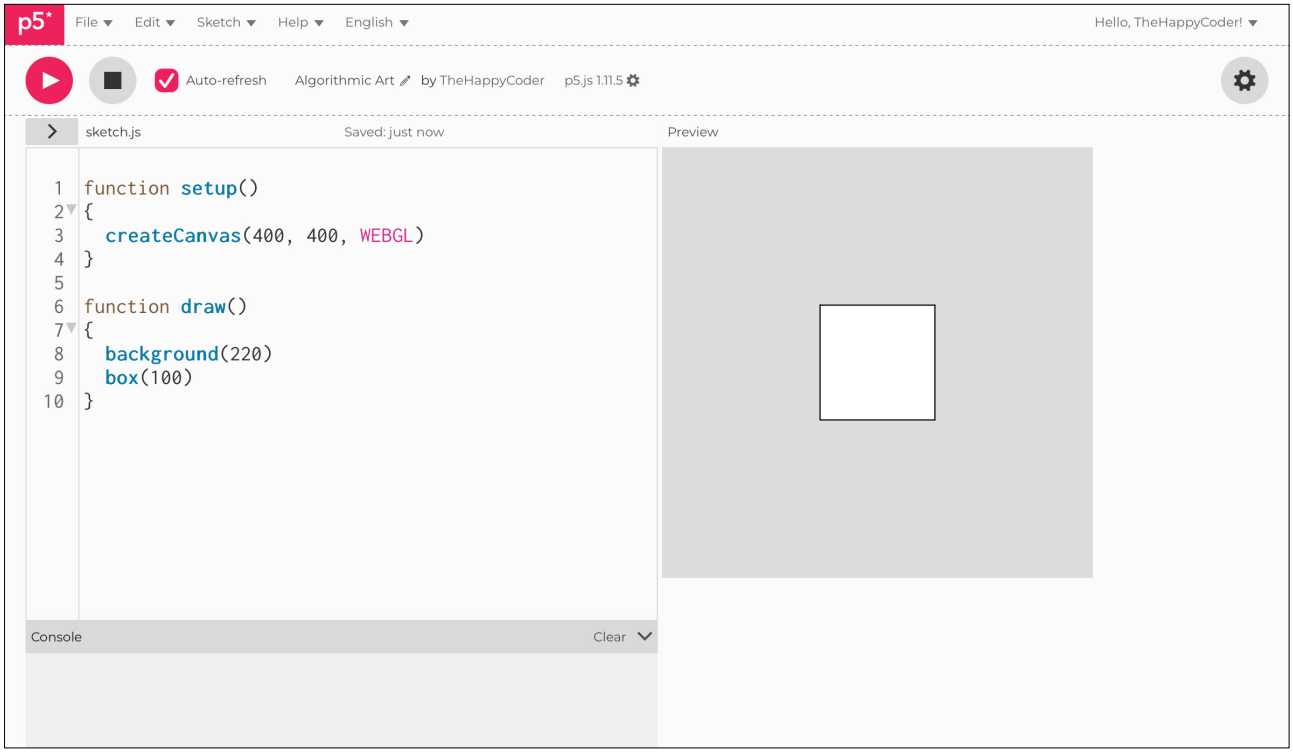
Challenge

Add the other two dimensions.

Code Explanation

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

Figure C1.3


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 C1.4 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.

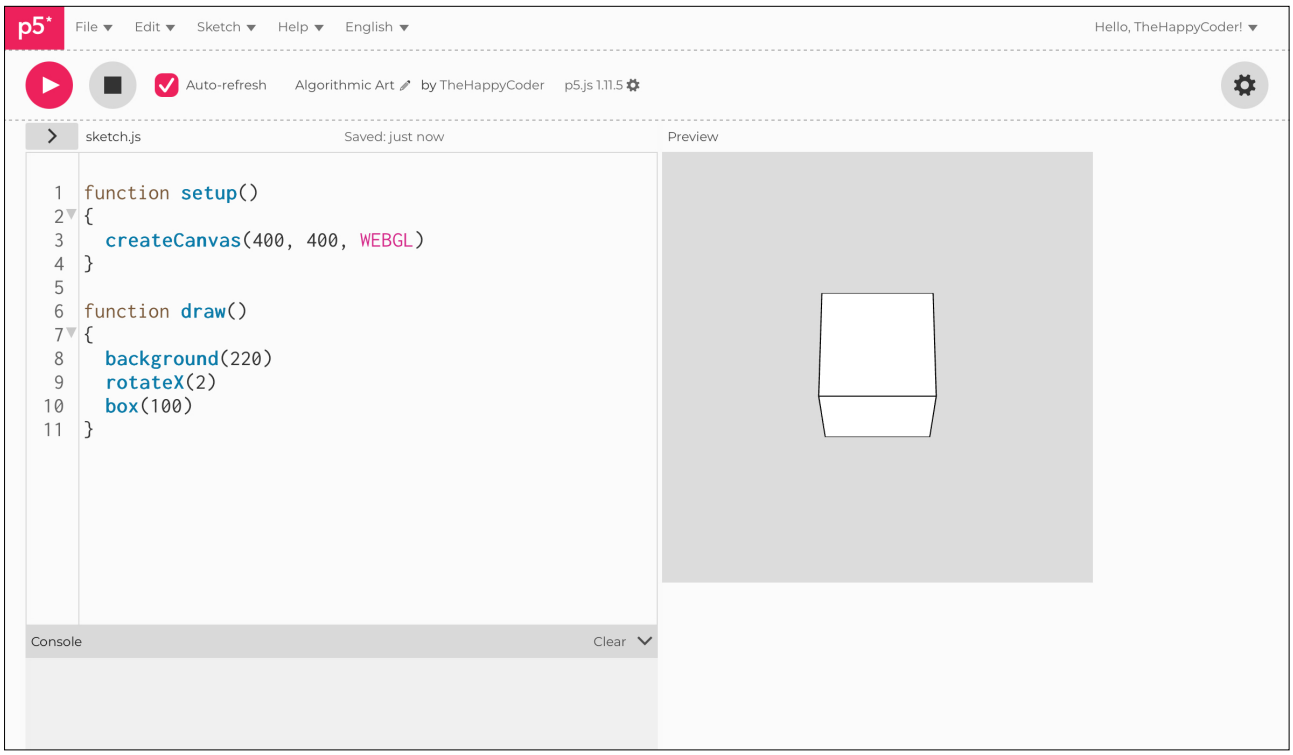
Challenge

Try other angles.

Code Explanation

rotateX(2)	Rotate along the x-axis by 2 radians.

Figure C1.4


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 C1.5 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)
}
```

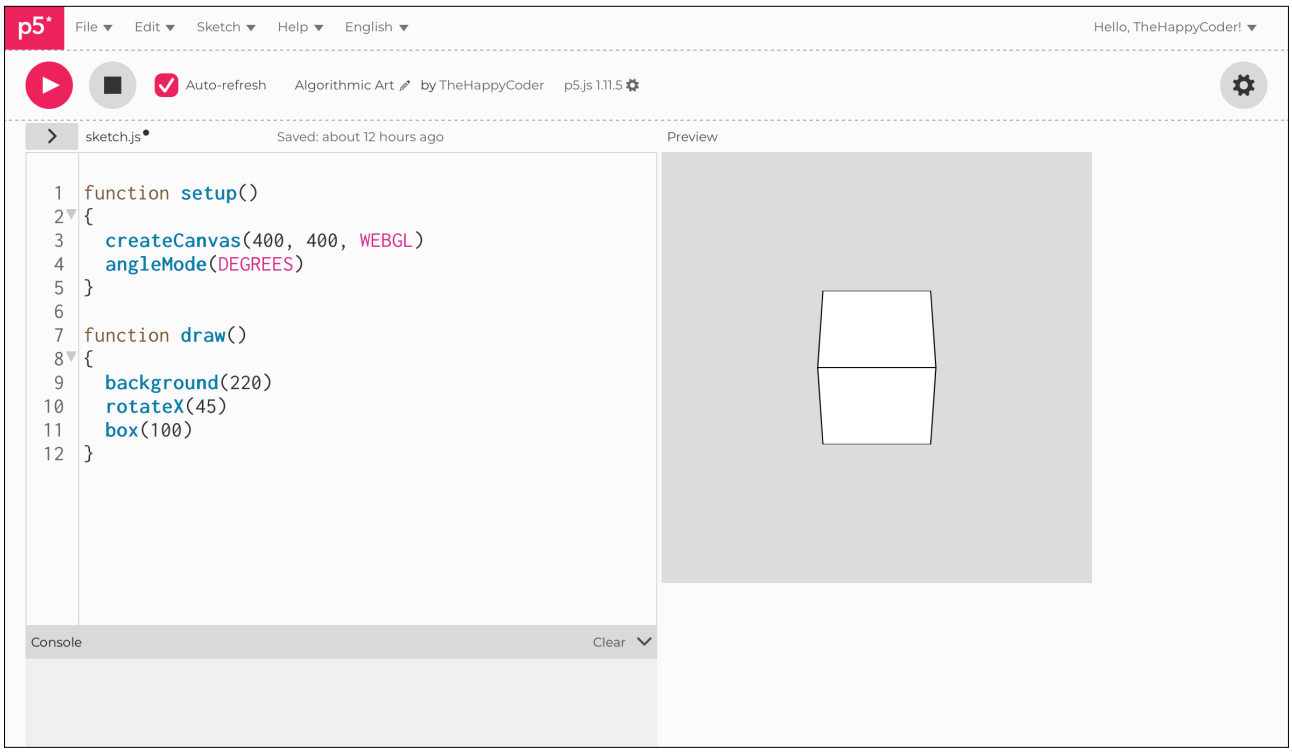
Notes

Almost the same

Code Explanation

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

Figure C1.5


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 C1.6 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.

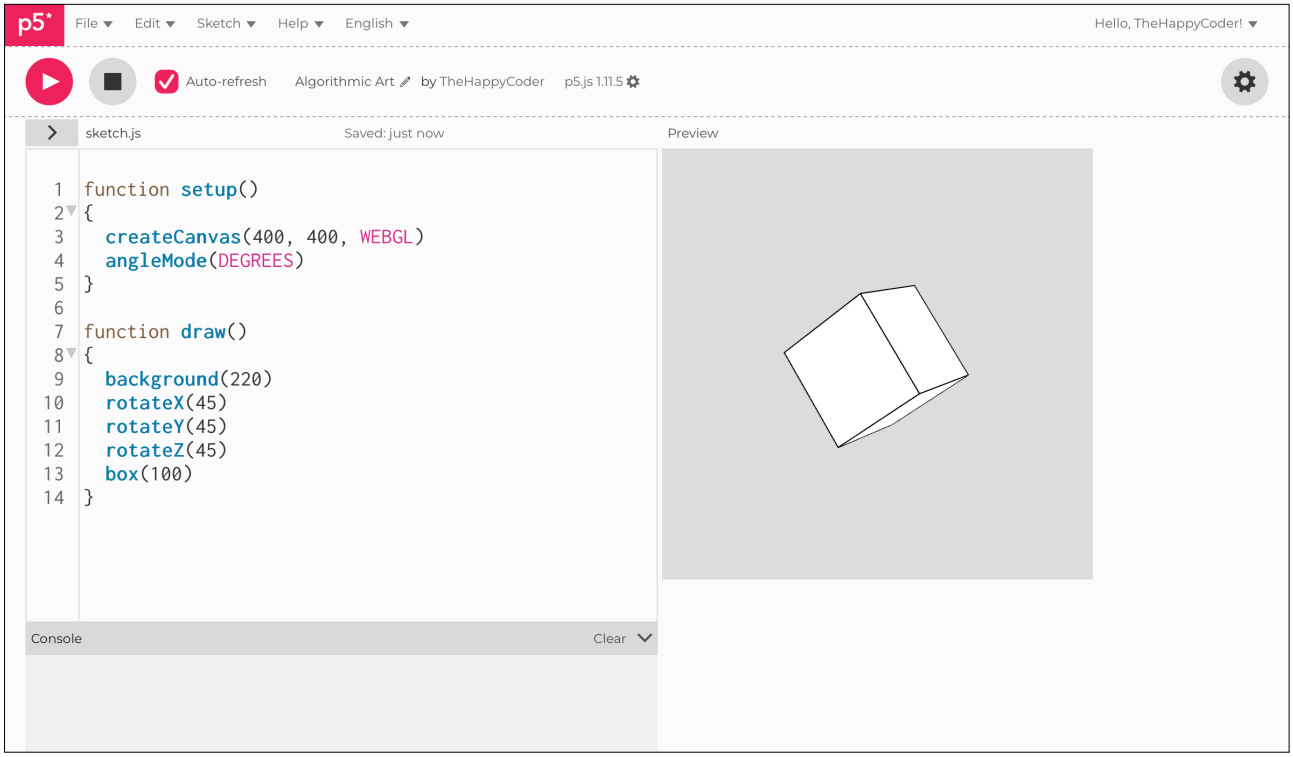
Challenge

Try different angles.

Code Explanation

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

Figure C1.6


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 C1.7 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 C1.8 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.

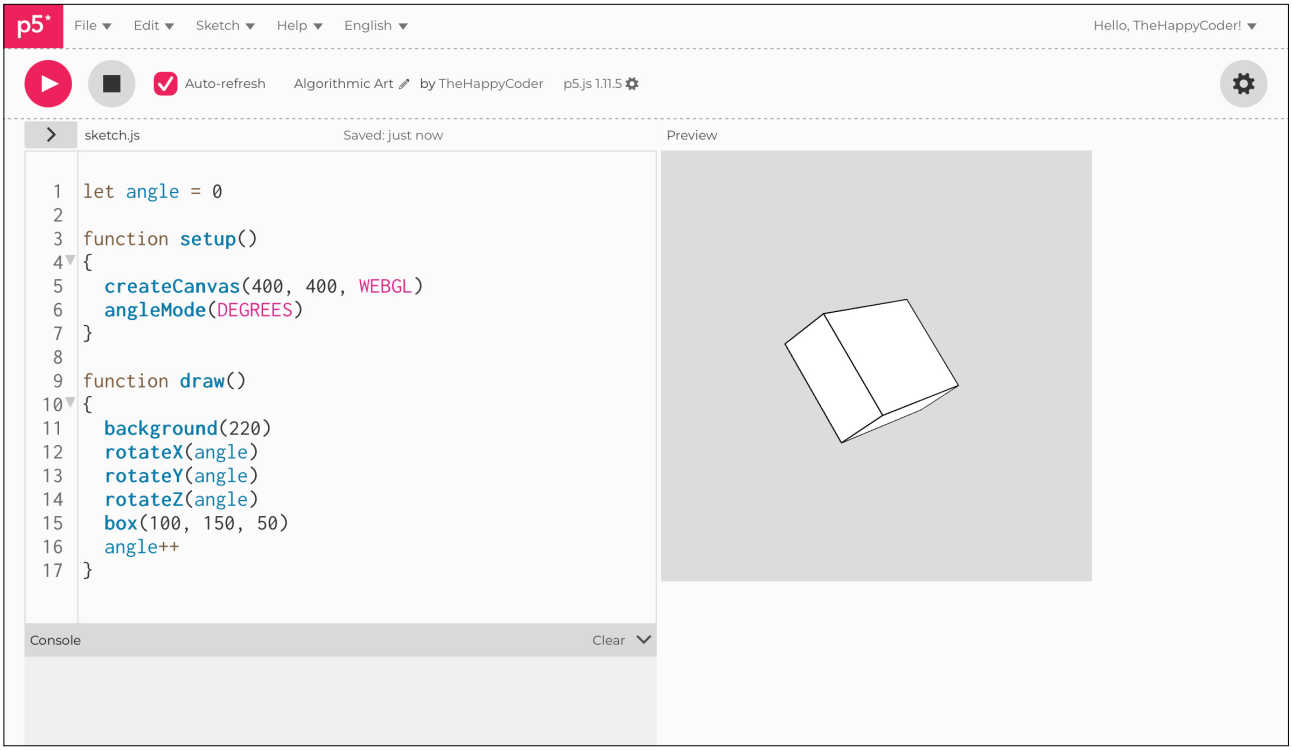
Challenge

Try other dimensions.

Code Explanation

`box(100, 150, 50)`	It draws a cuboid 100 wide, 150 long, and 50 deep.

Figure C1.8


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 C1.9 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.

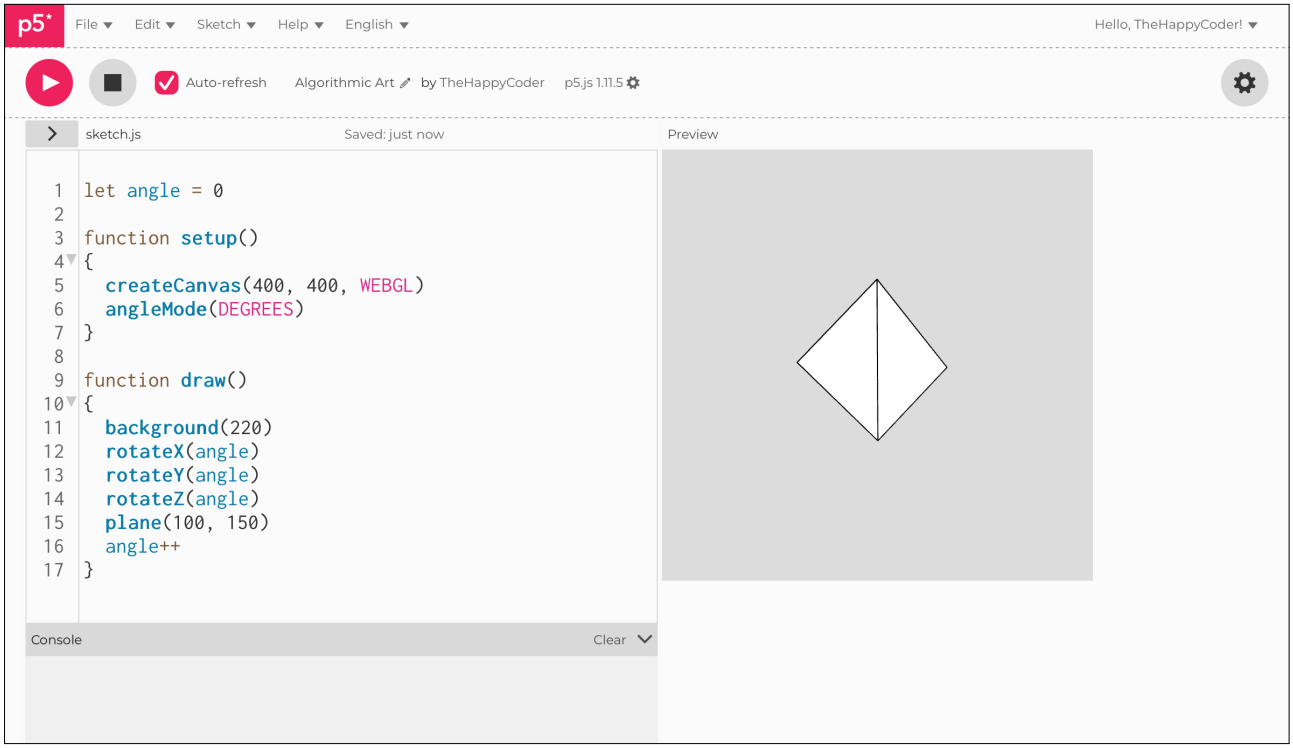
Challenge

Add noStroke().

Code Explanation

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

Figure C1.9


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 C1.10 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).

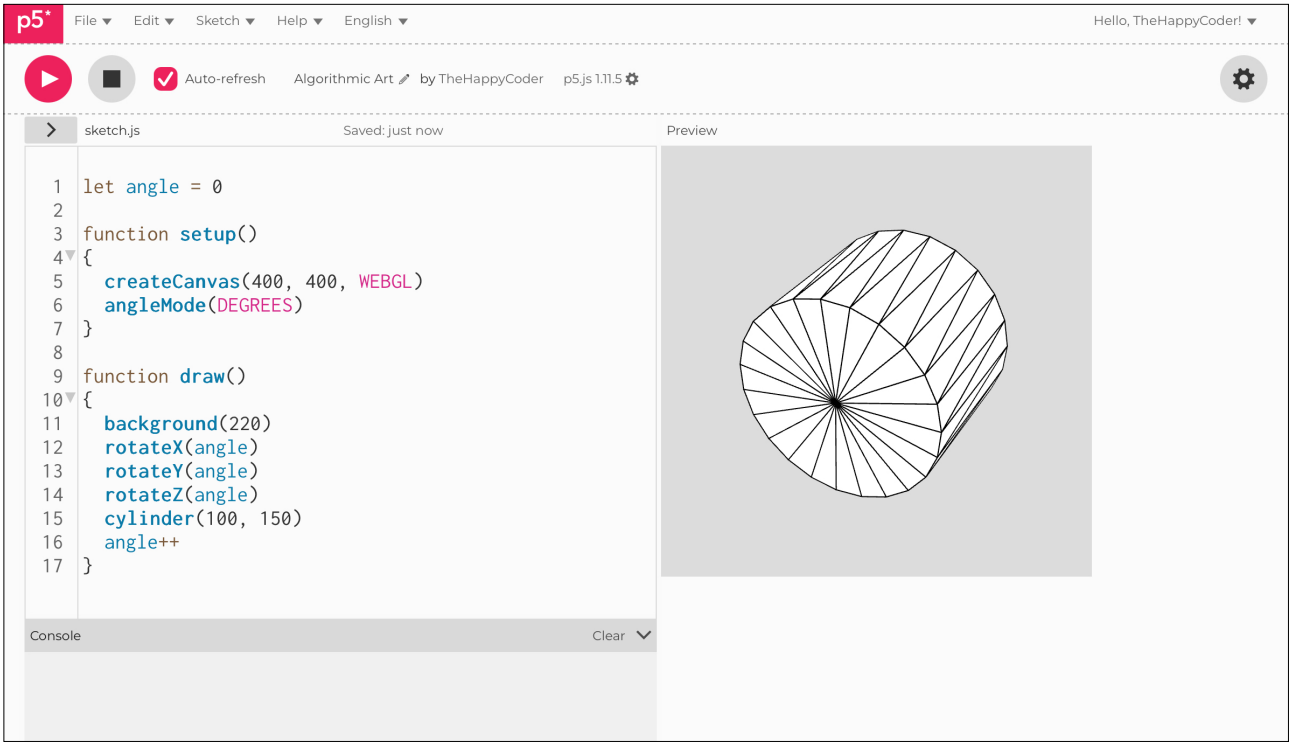
Challenge

Alter the dimensions.

Code Explanation

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

Figure C1.10


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 C1.11 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++
}
```

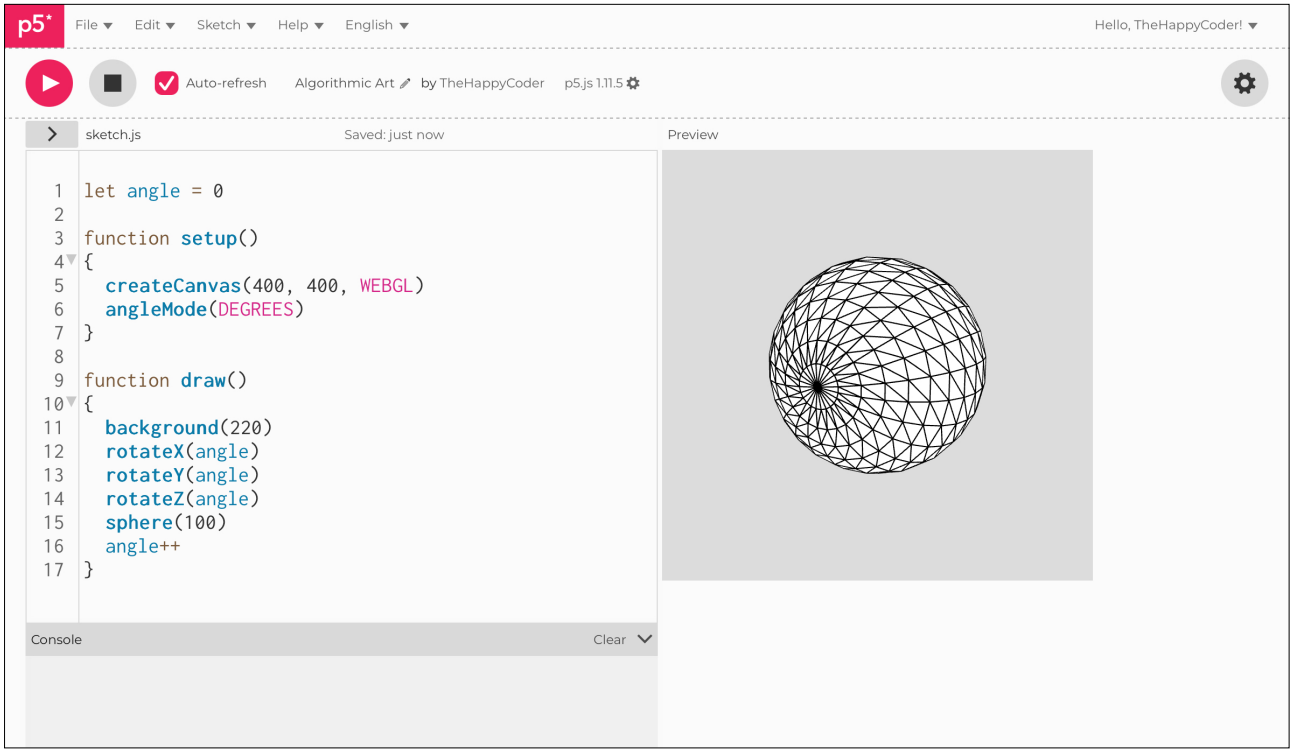
Notes

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 C1.11


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 C1.12 drawing an ellipsoid

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

```
let angle = 0

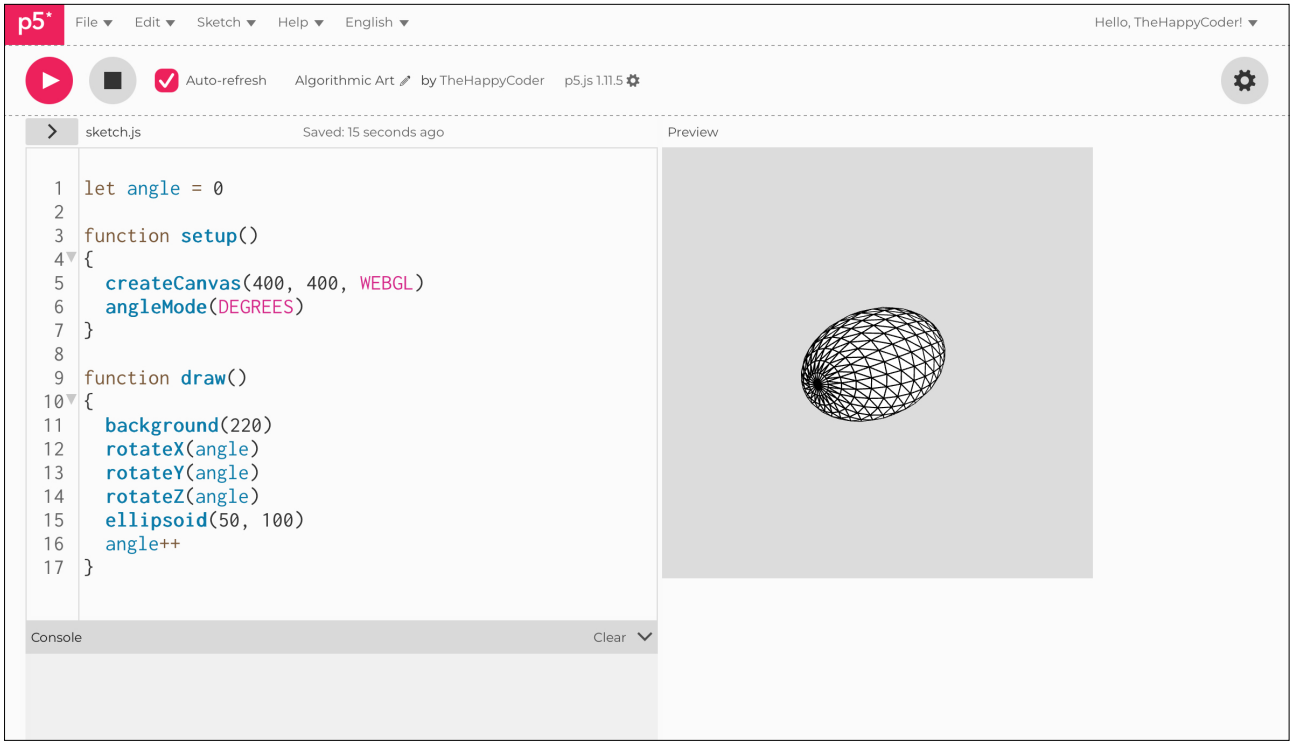
function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
}

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 C1.12


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 C1.13 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.

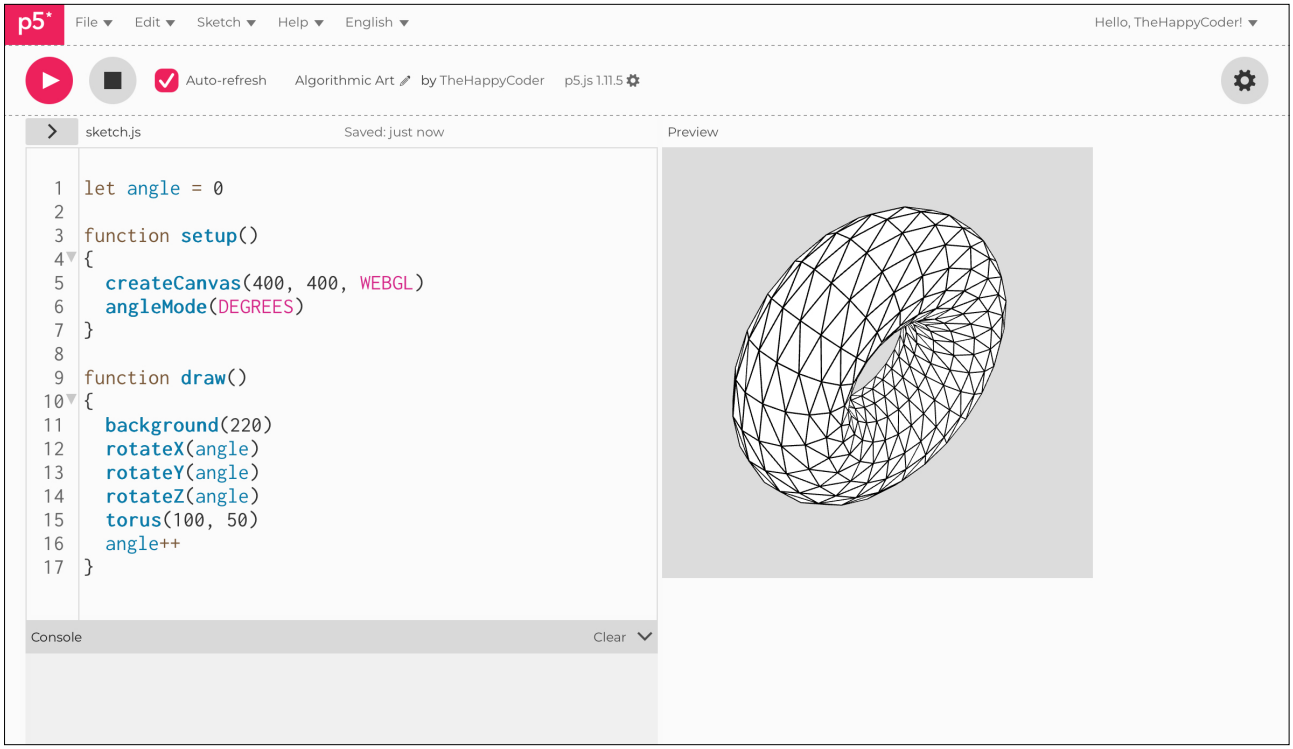
Challenge

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 C1.13


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 C1.14 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++
}
```

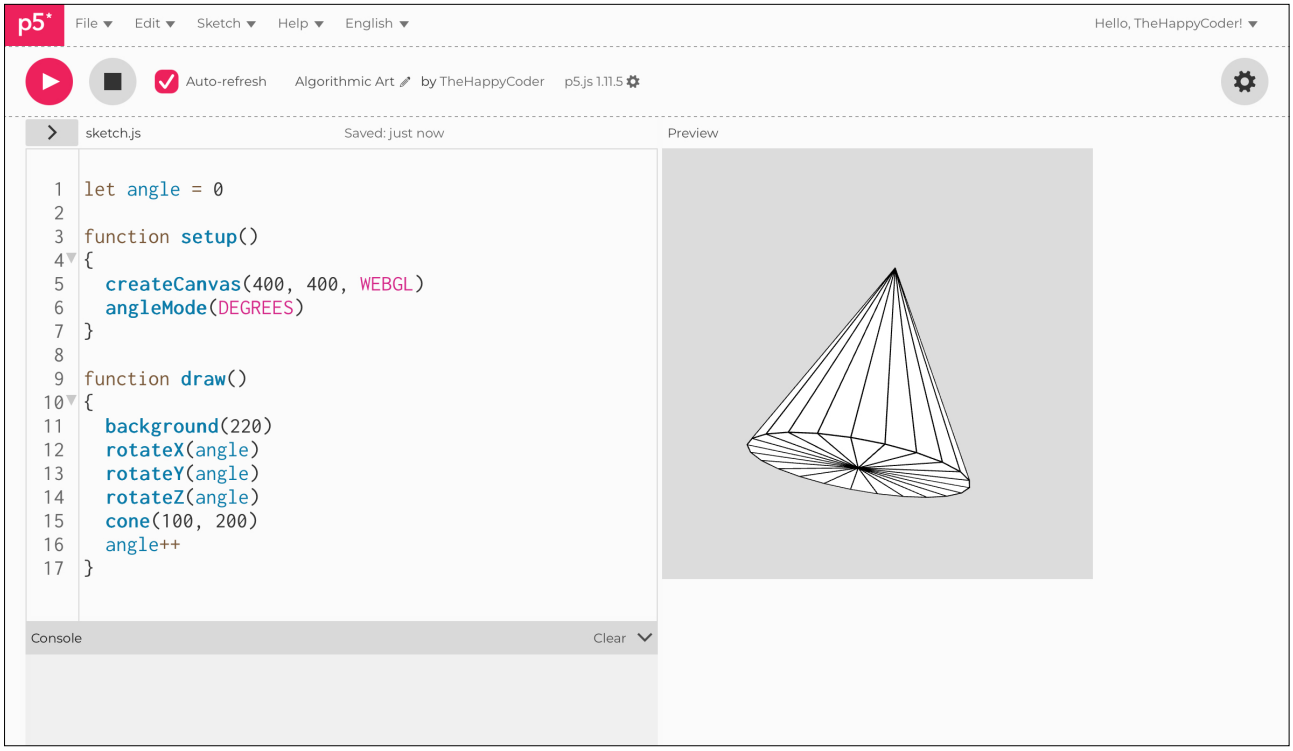
Challenges

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

Code Explanation

cone(100, 200)	Draw a cone with a radius of 100 and a length of 200.

Figure C.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 C1.15 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++
}
```

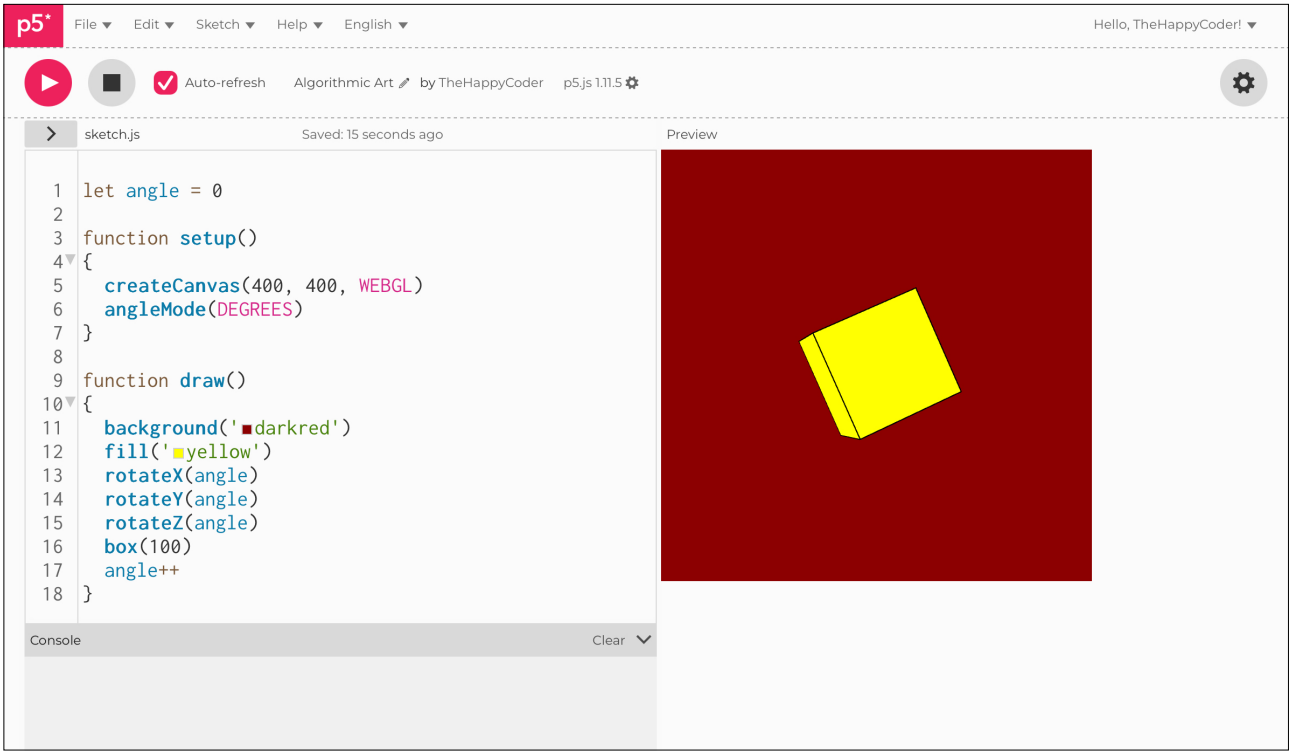
Notes

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

Challenge

Explore other colours.

Figure C1.15


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 C1.16 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.

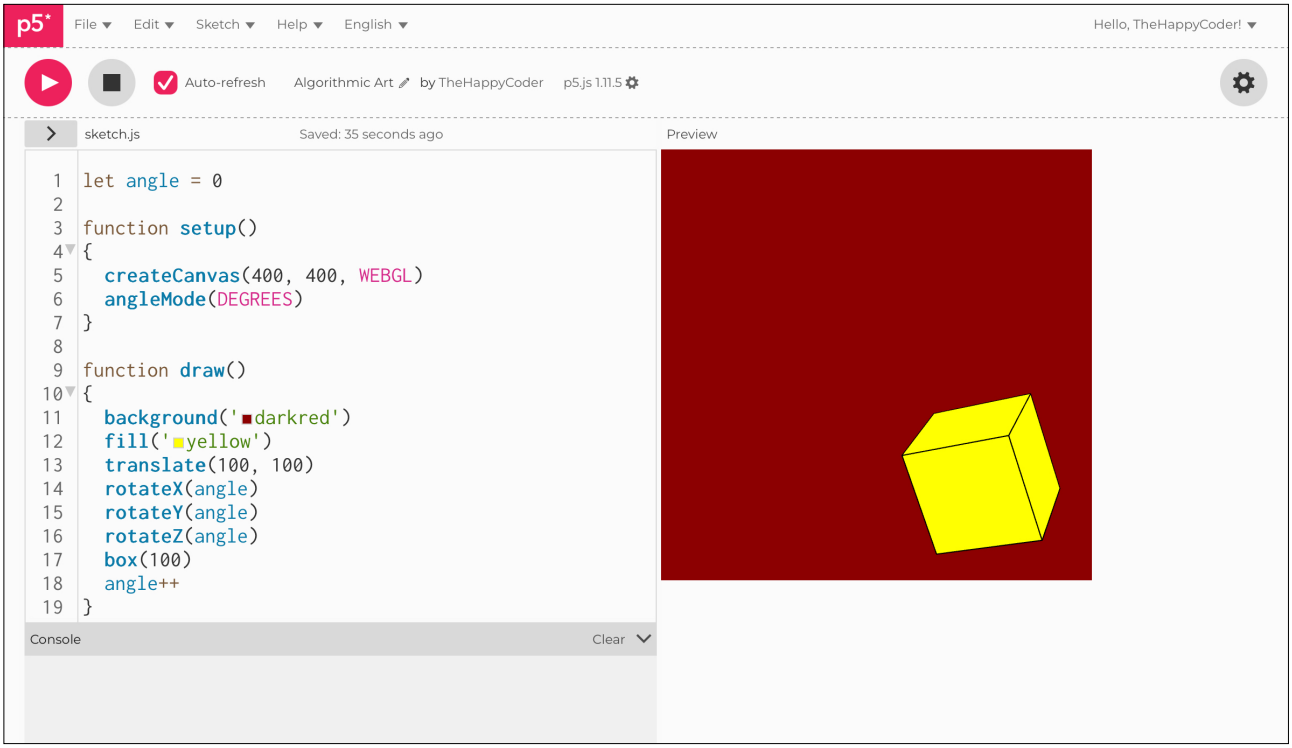
Challenges

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 C1.16


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 C1.17 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++
}
```

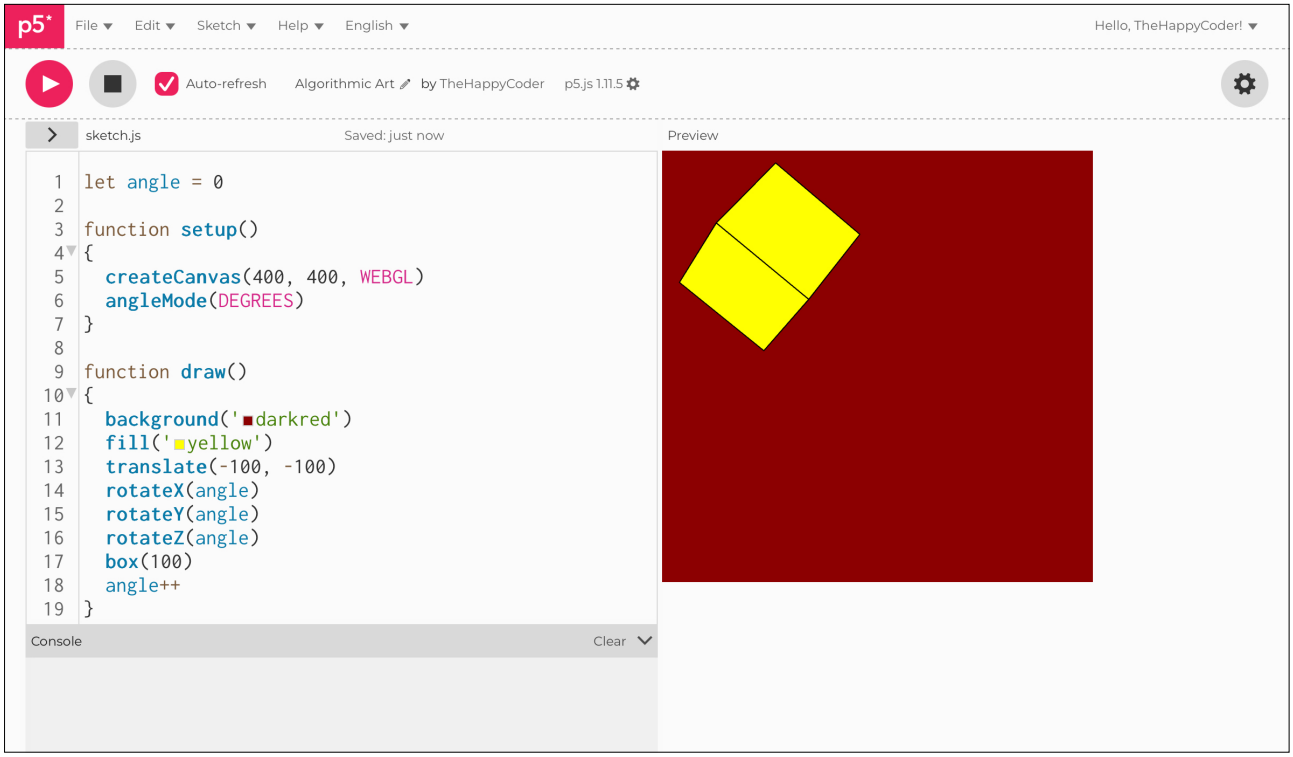
Notes

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

Code Explanation

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

Figure C1.17


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 C1.18 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++
}
```

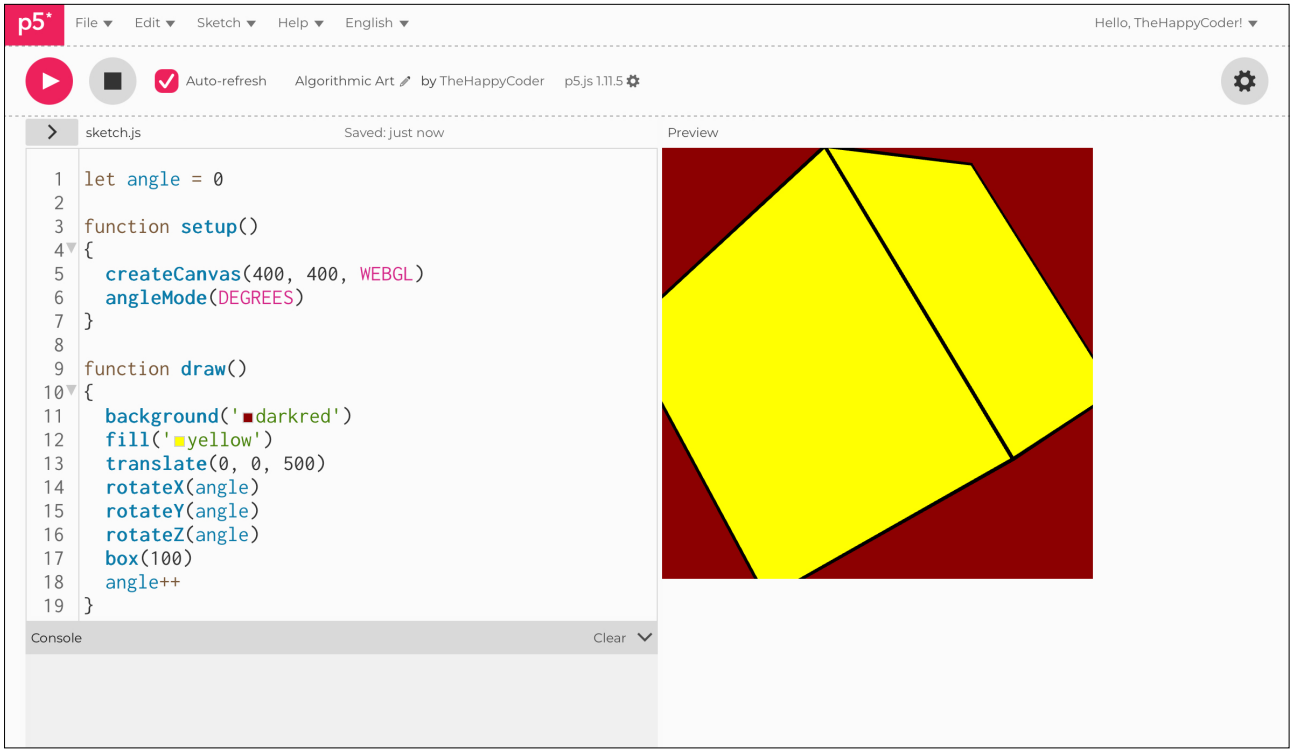
Notes

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, a positive value brings it forwards.

Figure C1.18


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 C1.19 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).

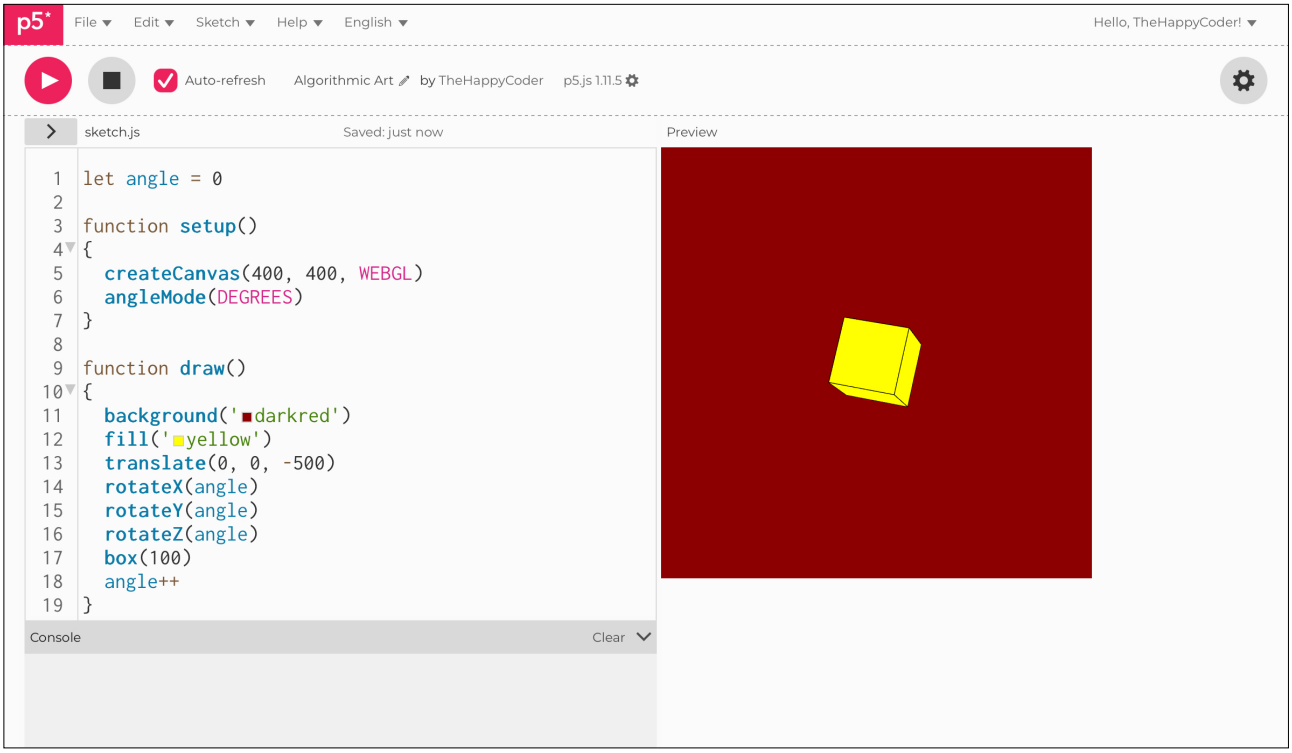
Challenge

Play with all three values.

Code Explanation

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

Figure C1.19


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 C1.20 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++
}
```

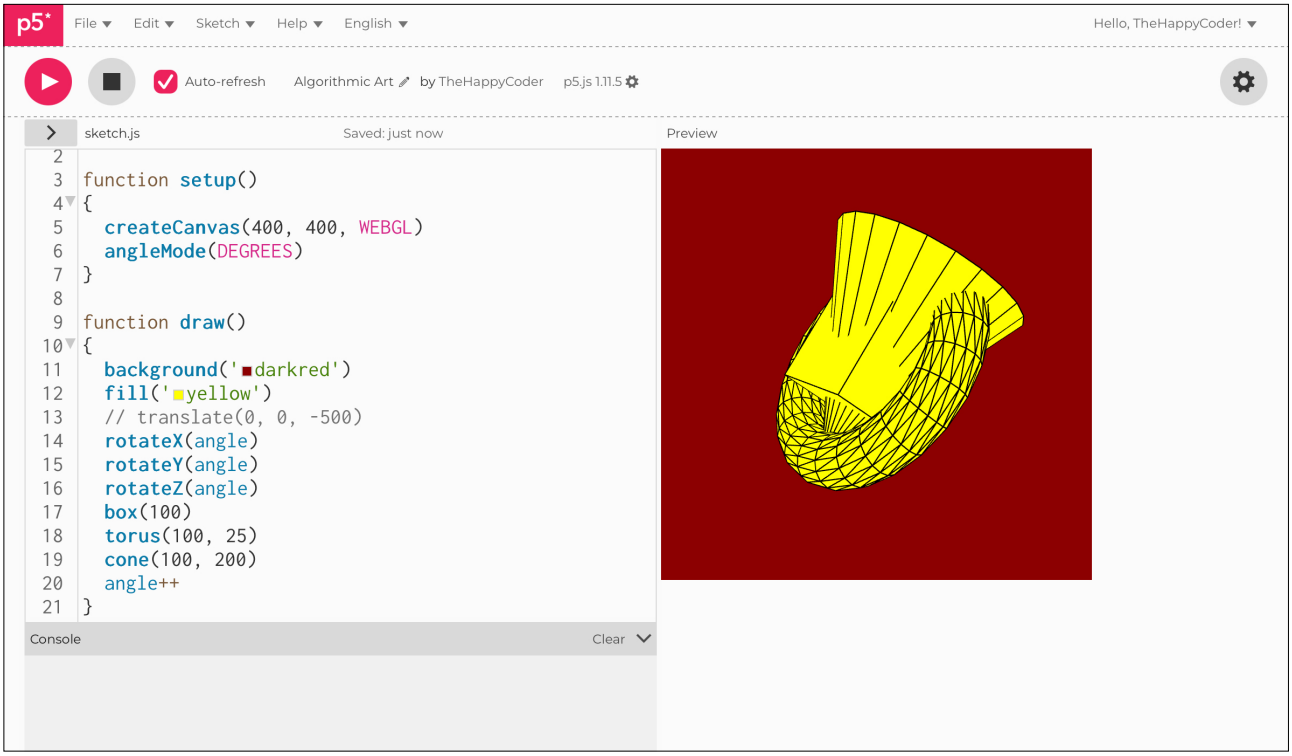
Notes

All three are meshed together.

Challenge

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

Figure C1.20


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 C1.21 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')
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  push()
  translate(0, 0, 100)
  rotateX(angle * 2)
  torus(100, 25)
  pop()
  cone(100, 200)
  angle++
}
```

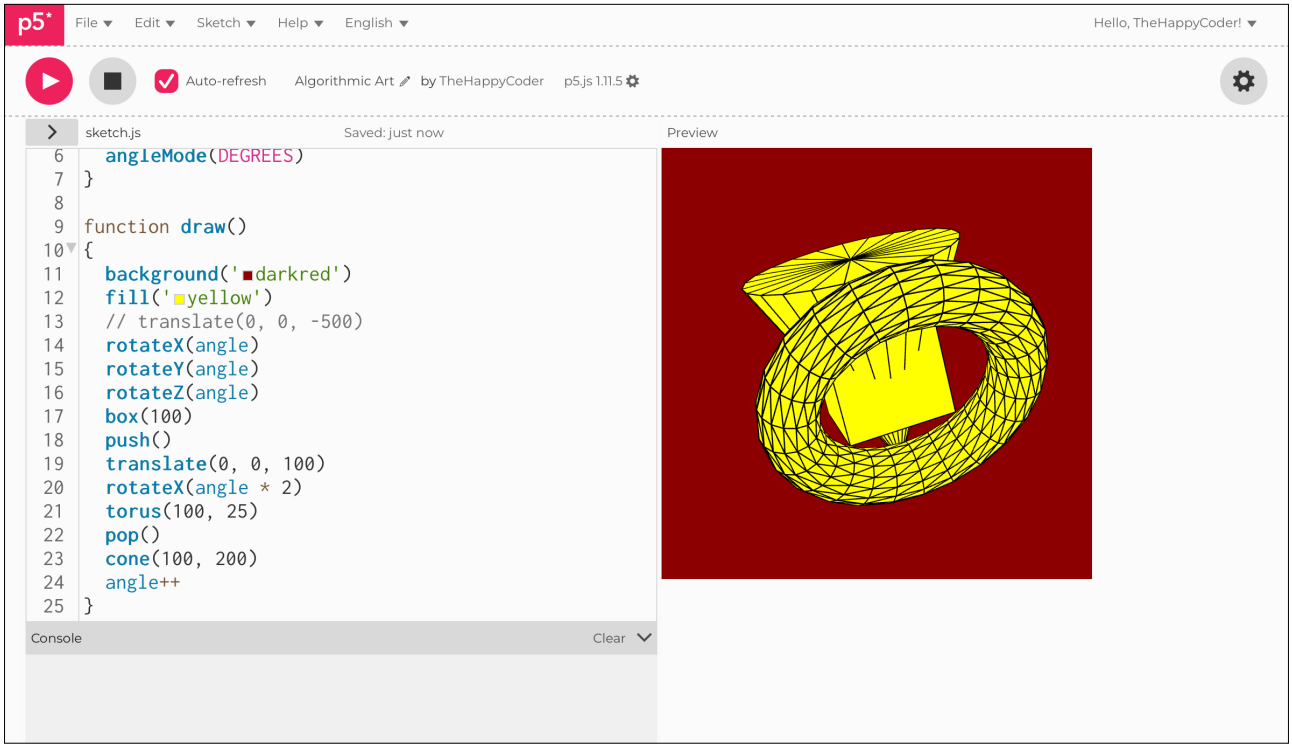
Notes

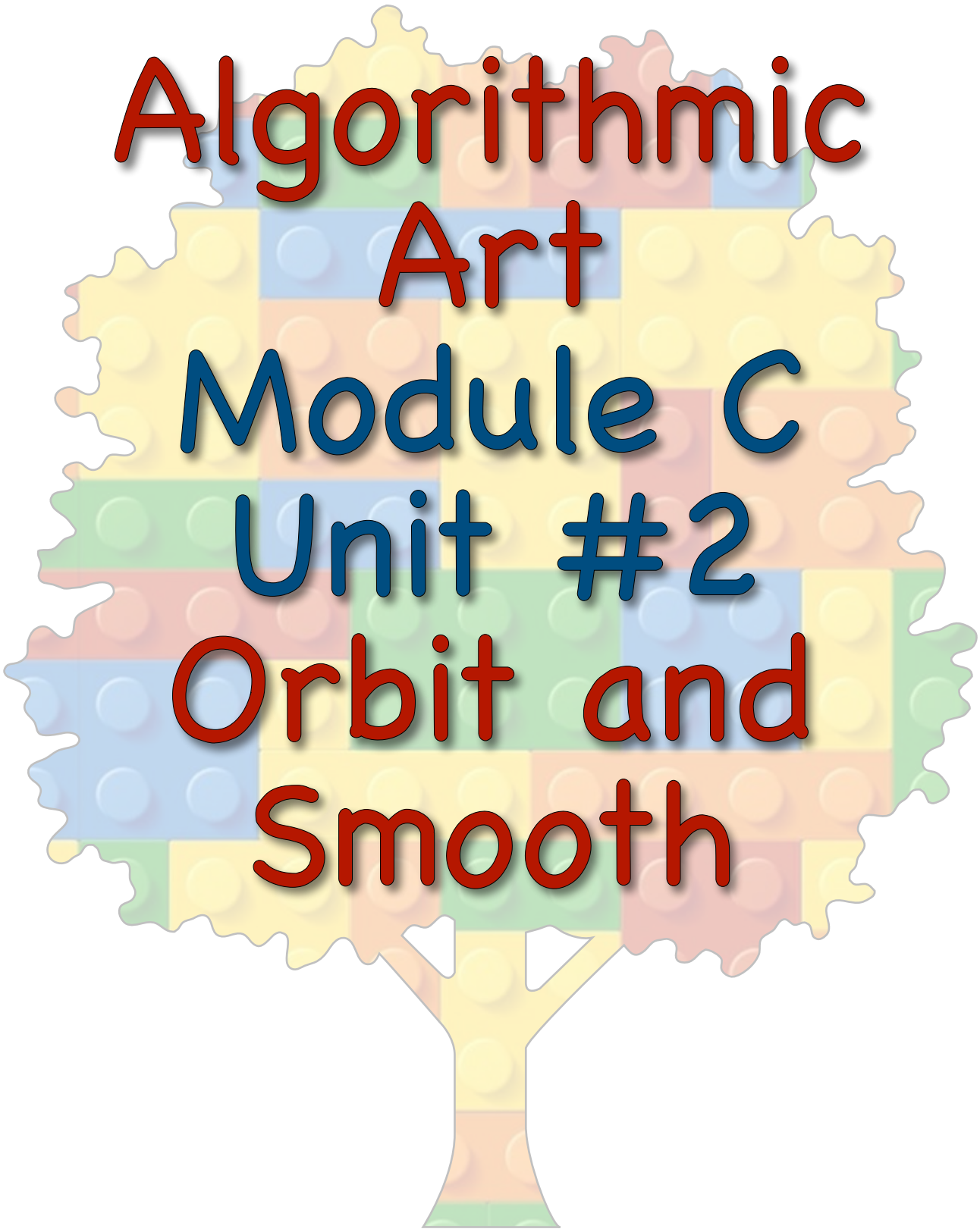
The torus moves independently of the other two shapes, and yet all three are still moving together. There is a lot of scope to do much more.

Challenge

Just play and see what you can create.

Figure C1.21


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



Algorithmic Art
Module C
Unit #2
Orbit and Smooth



Module C Unit #2: orbiting and smoothing

You will notice that the shapes aren't very smooth as they are drawn with a series of interconnecting triangles. This can be improved by increasing the number of triangles; it will mean that there is a lot more geometry for the code to keep track of, so don't go mad, keep them to a minimum where possible to still give a pleasing effect. In these units, we won't really need to worry as we are not making big demands on it, but it may start running slowly.



Sketch C2.1 default WEBGL sketch

! Start a new WEBGL sketch.

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

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

Sketch C2.2 default details

! Add the usual suspects

With 3D shapes, you may notice that the shapes with curves, for instance, the sphere and the torus, are not what you call smooth. The triangles that construct the shape add a certain angular nature to them. This is because the default detail is 24 for detailX and 16 for detailY.

You can increase the amount of detail for those shapes, but you cannot then draw the lines. If you remove the lines with noStroke(), then you can do it, giving you a much smoother-looking shape. To do this, we add two more arguments to our torus() shape. The third argument is detailX, and the fourth is detailY.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  torus(100, 25, 24, 16)
  angle++
}
```

Notes

Here we just add the default values for the torus().

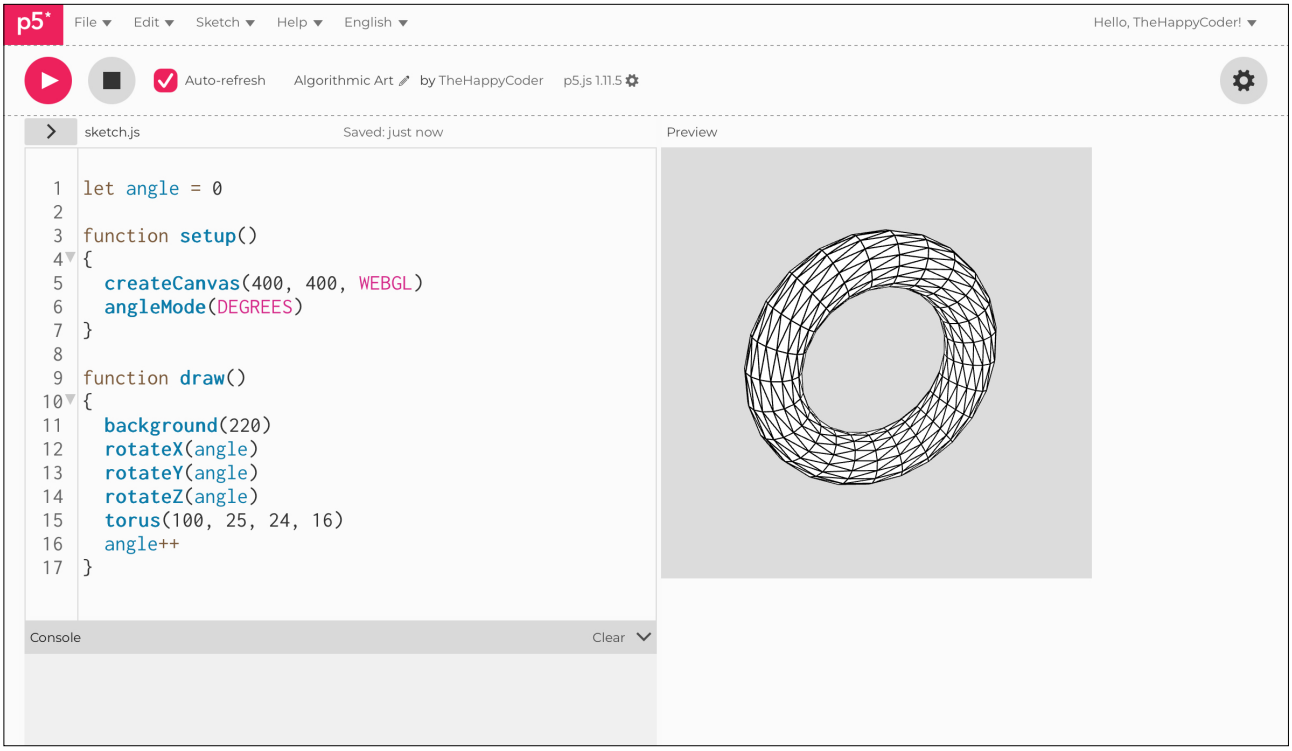
Challenge

What happens if you increase those values?

Code Explanation

torus(100, 25, 24, 16)	Drawing a torus with a detailX of 24 and a detailY of 16.

Figure C2.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
torus(100, 25, 24, 16)
angle++
}
Console
Clear



Sketch C2.3 smoothing the curves

We are going to remove the lines, noStroke(), and increase the values by a factor of two.

```
let angle = 0

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

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  noStroke()
  torus(100, 25, 48, 32)
  angle++
}
```

Notes

This creates a much smoother-looking shape. However, because we haven't explored materials and light yet, the full benefit is not readily appreciated.

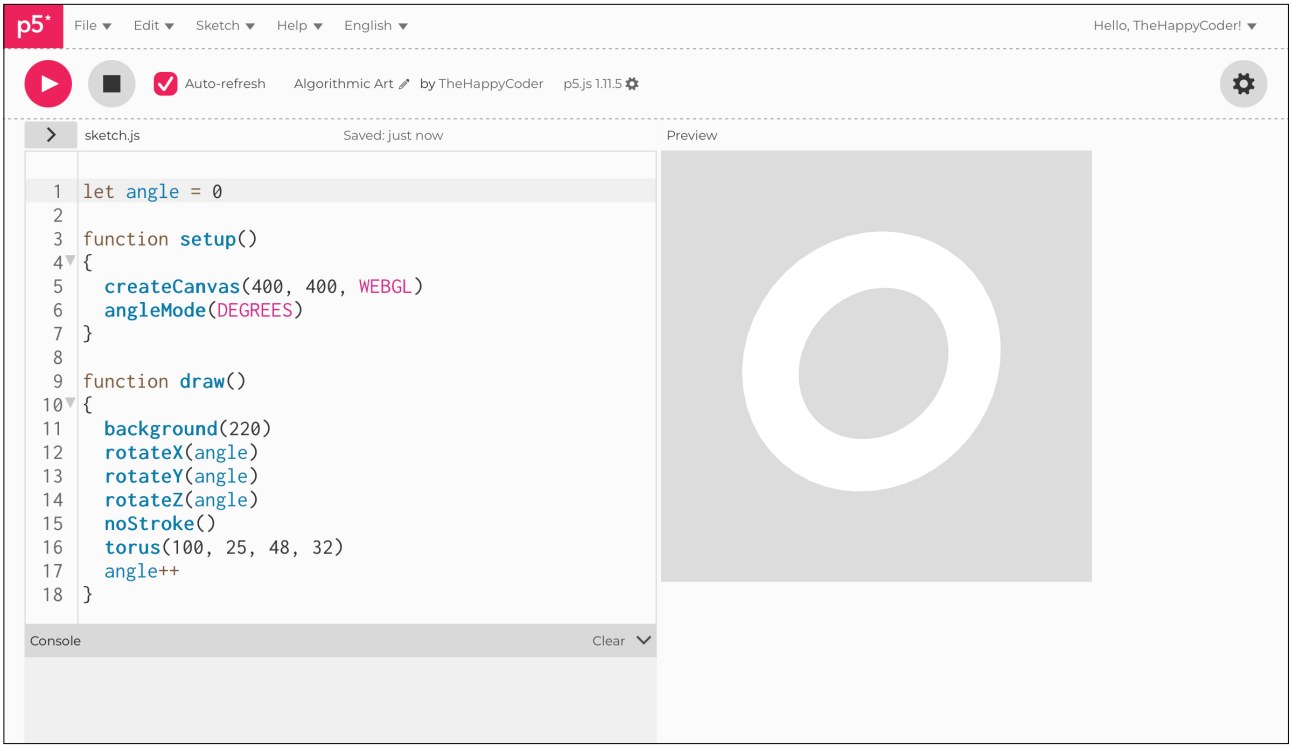
Challenges

1. See what difference it makes if you change the detail by half (12 and 8).
2. Try even smaller values to see the effect, as well as much larger values.

Code Explanation

`torus(100, 25, 48, 32)`	Drawing a torus with a detailX of 24 and a detailY of 16.

Figure C2.3


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
let angle = 0
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(angle)
rotateY(angle)
rotateZ(angle)
noStroke()
torus(100, 25, 48, 32)
angle++
}
Console
Clear

Sketch C2.4 a single cube

! Starting a new sketch

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

function draw()
{
  background(220)
  box(50)
}
```

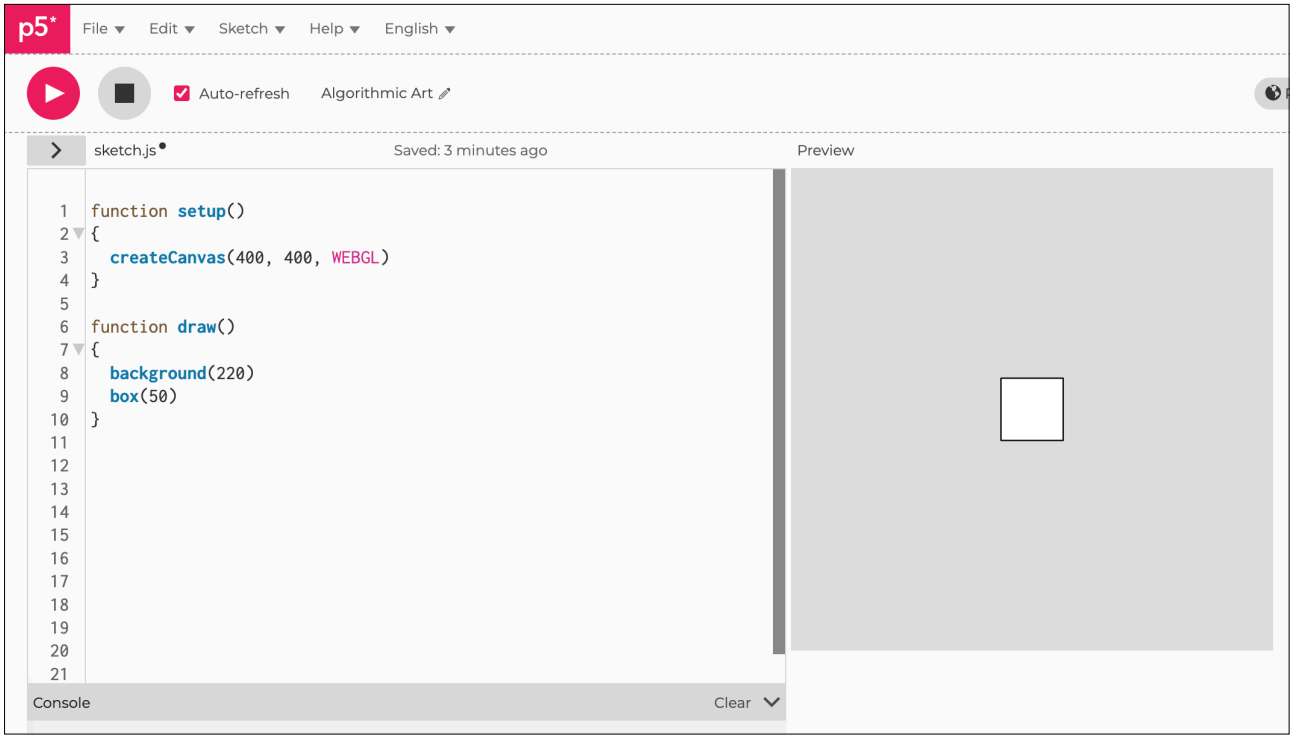
Notes

A simple box to start with.

Challenge

Other shapes are available.

Figure C2.4





Sketch C2.5 many cubes

Making many cubes.

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

function draw()
{
  background(220)
  box(50)
  push()
  translate(50, 50, 0)
  box(50)
  pop()
  push()
  translate(-50, 50, 0)
  box(50)
  pop()
  push()
  translate(50, -50, 0)
  box(50)
  pop()
  push()
  translate(-50, -50, 0)
  box(50)
  pop()
}
```

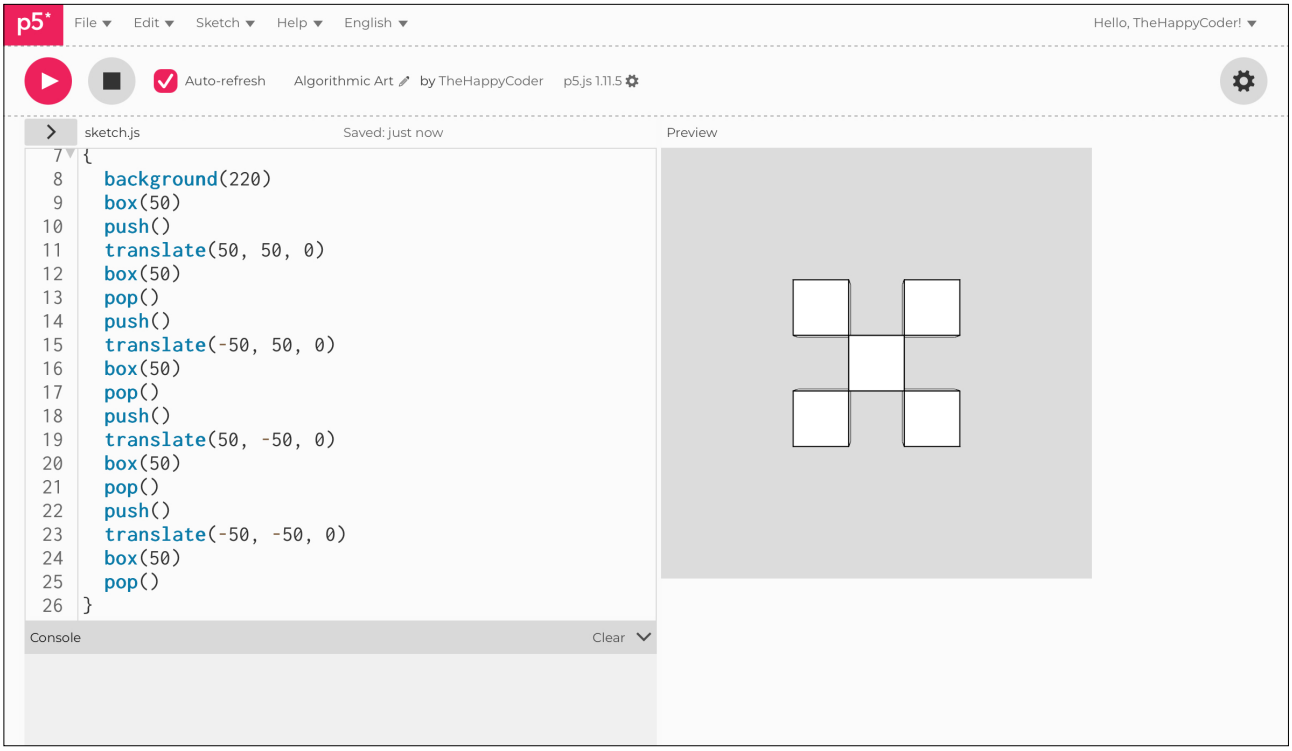
Notes

Using `push()` and `pop()` to translate each other cube separately.

Challenge

Make each box rotate separately.

Figure C2.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
7 {
8 background(220)
9 box(50)
10 push()
11 translate(50, 50, 0)
12 box(50)
13 pop()
14 push()
15 translate(-50, 50, 0)
16 box(50)
17 pop()
18 push()
19 translate(50, -50, 0)
20 box(50)
21 pop()
22 push()
23 translate(-50, -50, 0)
24 box(50)
25 pop()
26 }
Console
Clear



Sketch C2.6 controlling the orbit

Now add the orbit control; click and drag your mouse over the canvas.

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

function draw()
{
  background(220)
  orbitControl()
  box(50)
  push()
  translate(50, 50, 0)
  box(50)
  pop()
  push()
  translate(-50, 50, 0)
  box(50)
  pop()
  push()
  translate(50, -50, 0)
  box(50)
  pop()
  push()
  translate(-50, -50, 0)
  box(50)
  pop()
}
```

Notes

The `orbitControl()` function allows you to move the canvas so that you orbit the shapes.

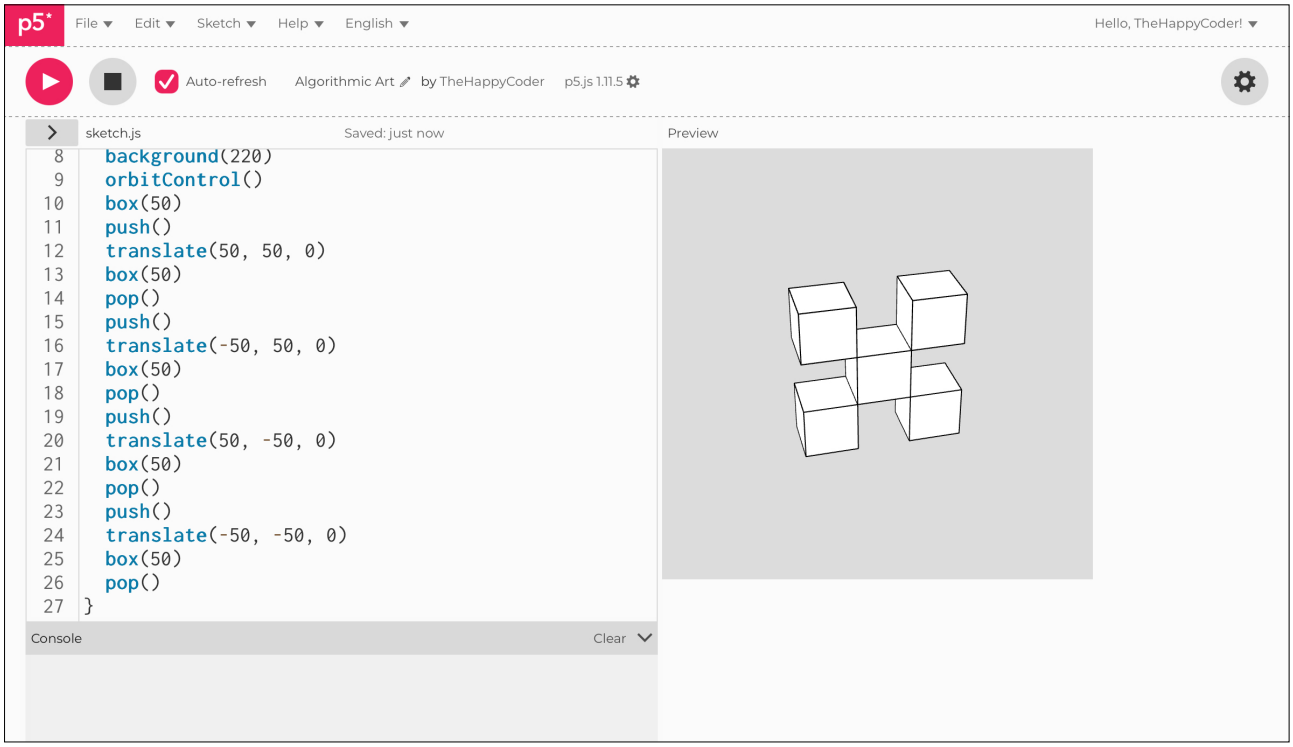
Challenge

Try other shapes.

Code Explanation

`orbitControl()`	A function that allows you to orbit the shapes.

Figure C2.6



Sketch C2.7 frame count

! Start another new sketch

Another quick way to rotate is to use the `frameCount`. Here we can rotate a box just using the `frameCount` variable. It counts how many frames have elapsed since starting the sketch. Note it is a variable, not a function. Also, use angle mode with degrees; otherwise, it will spin out of control.

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

}

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

Notes

Rotates nicely enough

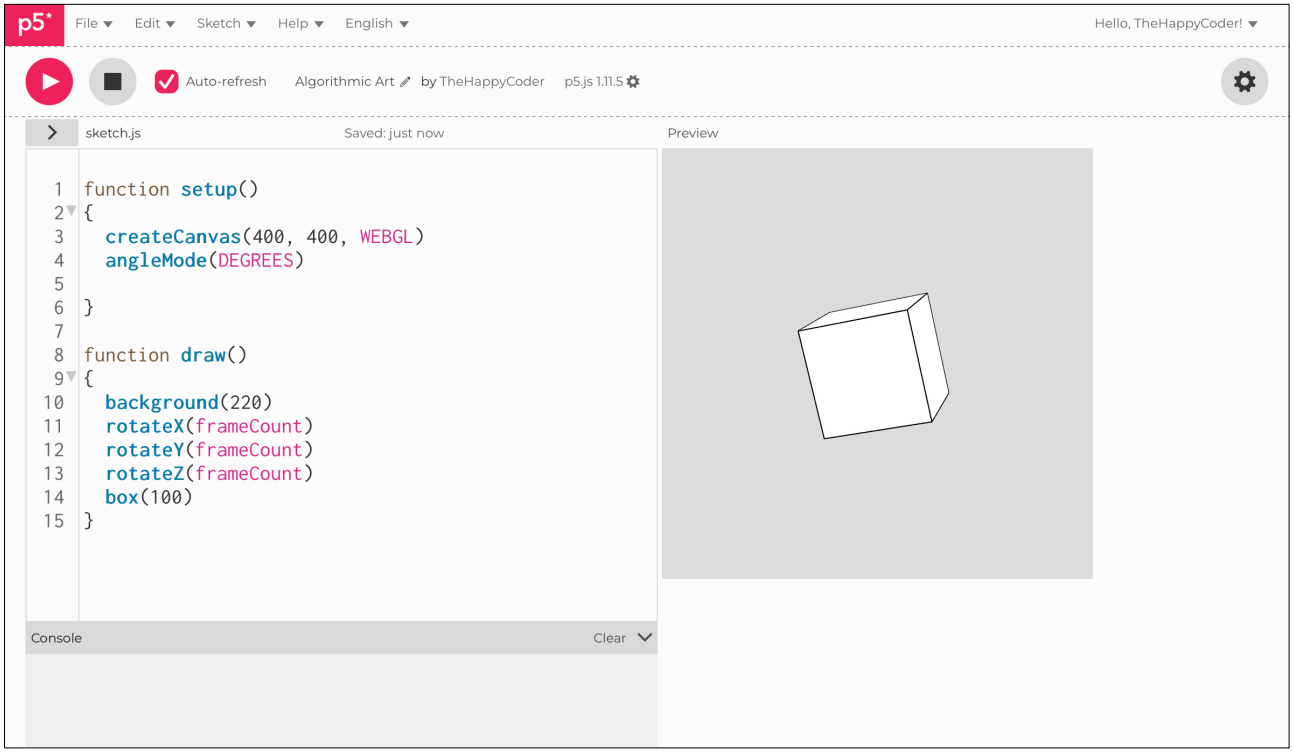
Challenge

Remove `angleMode(DEGREES)`

Code Explanation

`rotateX(frameCount)`	Rotates every time the draw() function loops.

Figure C2.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
rotateX(frameCount)
rotateY(frameCount)
rotateZ(frameCount)
box(100)
}
Console
Clear



Sketch C2.8 alternative frame count

An alternative is to remove `angleMode()` and just divide the frame count by `50` to reduce the spin rate.

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

function draw()
{
  background(220)
  rotateX(frameCount/50)
  rotateY(frameCount/50)
  rotateZ(frameCount/50)
  box(100)
}
```

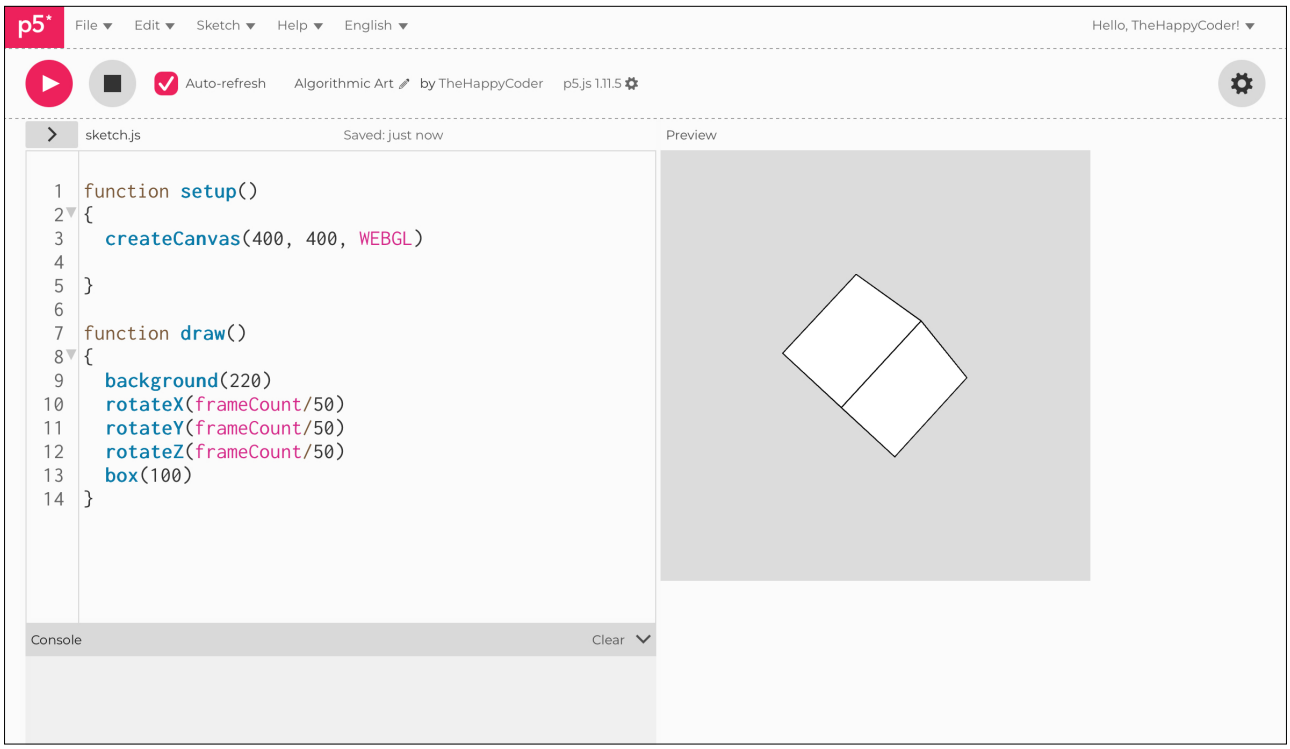
Notes

Rotates at a more steady rate.

Challenge

Other rates are available.

Figure C2.8


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
1 function setup()
2 {
3 createCanvas(400, 400, WEBGL)
4
5 }
6
7 function draw()
8 {
9 background(220)
10 rotateX(frameCount/50)
11 rotateY(frameCount/50)
12 rotateZ(frameCount/50)
13 box(100)
14 }
Console
Clear



Algorithmic Art
Module C
Unit #3
Lights and Materials
Part 1



Module C Unit #3: lights and materials #1

Because we are working in 3D, we can position lights as a separate entity rather than just fill a shape with a colour (which you can do as well).

We can create many pleasing results with just a little practice. There are a number of light options as well as materials. Using the right combination, for not all work as you might expect, you can create some pleasing effects.

There are a number of materials we can use with WebGL:

```
normalMaterial()
ambientMaterial()
specularMaterial()
emissiveMaterial()
```

For the lights, we have a few to choose from:

```
ambientLight()
pointLight()
directionalLight()
lights()
```

You have a lot to choose from and play with.

Sketch C3.1 draw a sphere

Our starting sketch, the humble sphere.

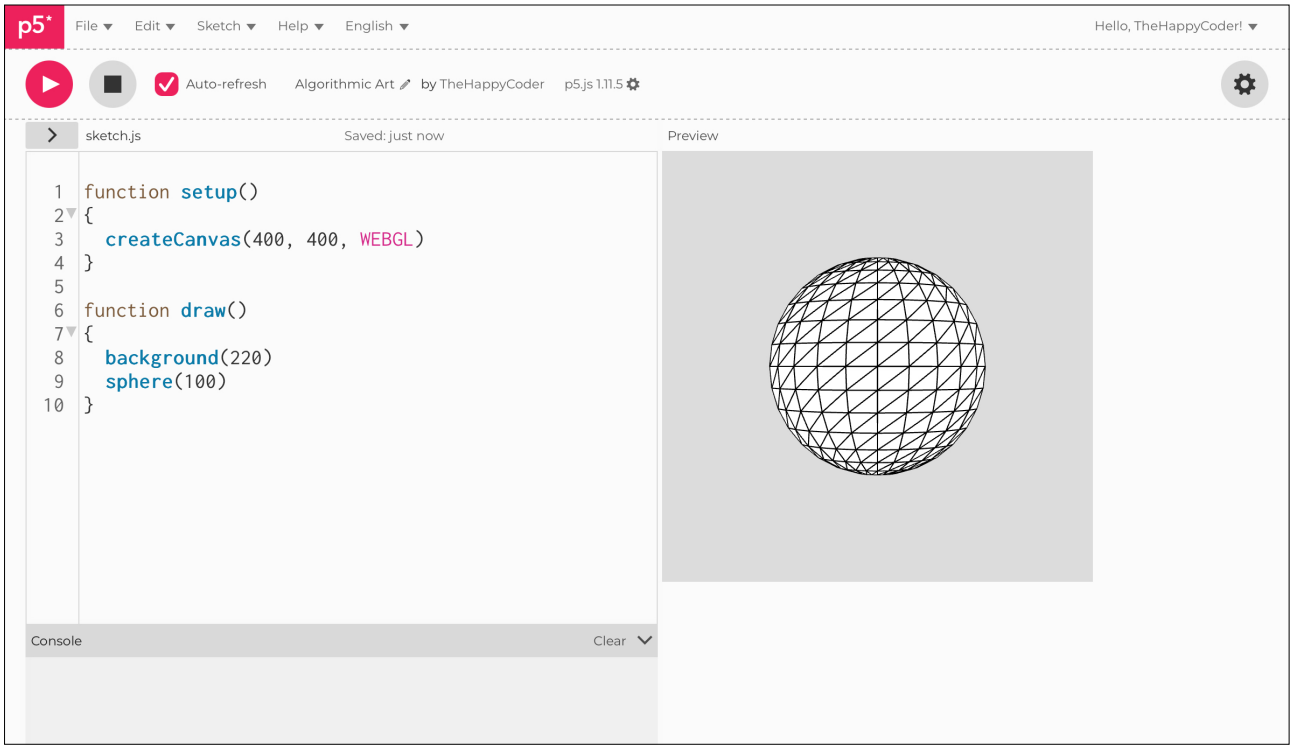
```
function setup()
{
  createCanvas(400, 400, WEBGL)
}

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

Notes

The sphere is made up of lots of triangles.

Figure C3.1





Sketch C3.2 a normal material

Using `normalMaterial()` is just a quick way to give objects a solid colour to make sure everything is working OK.

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

function draw()
{
  background(220)
  normalMaterial()
  sphere(100)
}
```

Notes

It gives you a strange, almost psychedelic tint to the colours. There isn't much you can do with it other than use it, handy though.

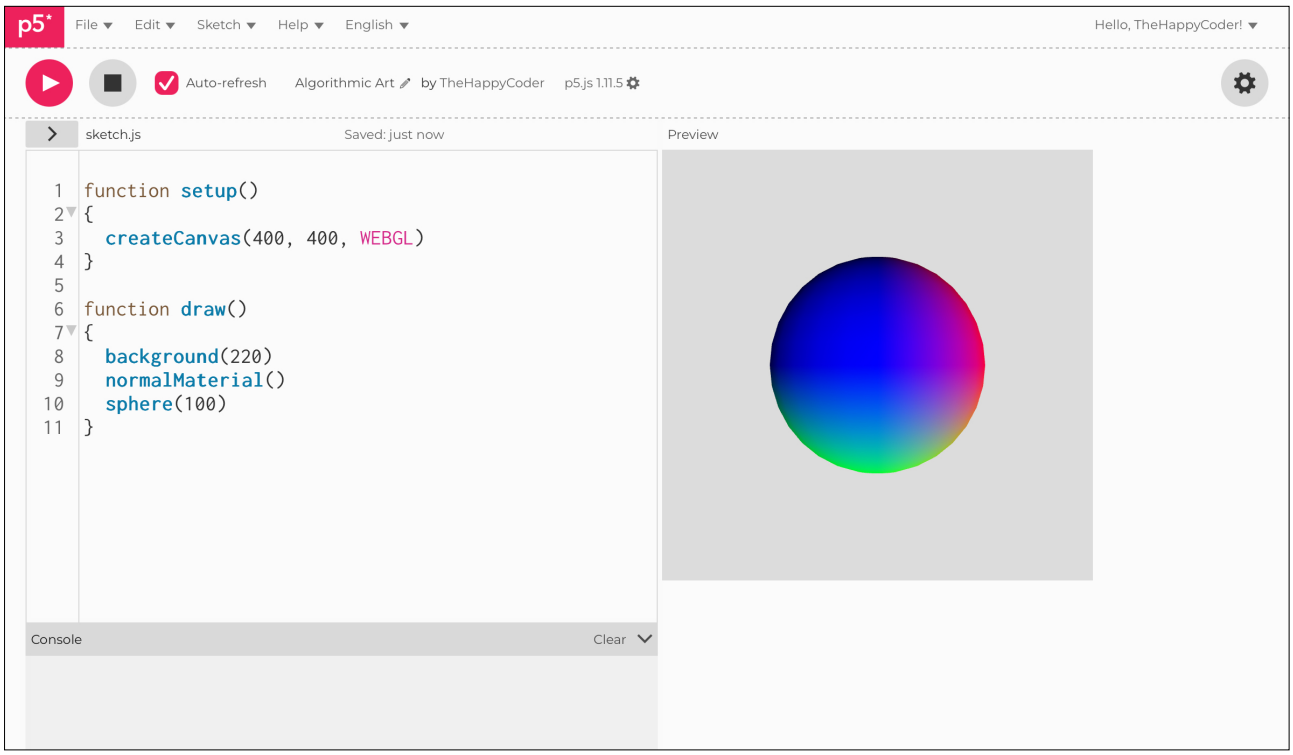
Challenges

1. Try other shapes.
2. Add `orbitControl()` to spin the shapes around.

Code Explanation

`normalMaterial()`	A basic material you can put on the shapes.

Figure C3.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
normalMaterial()
sphere(100)
}
Console
Clear

Sketch C3.3 a more ambient material

Another material we could give it is called `ambientMaterial()`. We have to give it a colour, and in this case, we will give it a `yellow` colour.

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

function draw()
{
  background(220)
  ambientMaterial('yellow')
  sphere(100)
}
```

Notes

Oh dear, that didn't work.

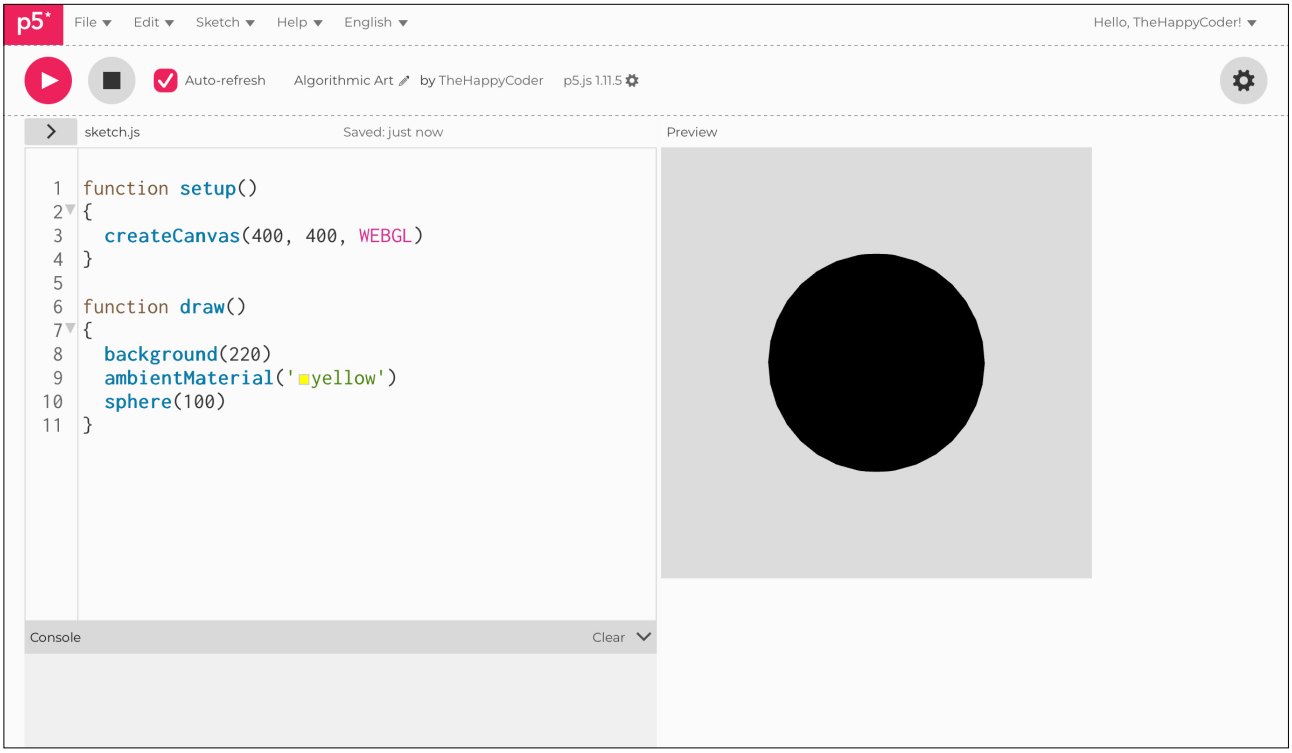
Challenge

Can you imagine why?

Code Explanation

`ambientMaterial('yellow')`	Gives the shape a material colour.

Figure C3.3


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
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
ambientMaterial('yellow')
sphere(100)
}
Console
Clear



Sketch C3.4 an ambient light

There is a good reason it didn't work; we need to give it some `ambientLight()`. For this to show the true colour of the `ambientMaterial()`, we need to use white light.

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

function draw()
{
  background(220)
  ambientLight('white')
  ambientMaterial('yellow')
  sphere(100)
}
```

Notes

We have shone a white light, which means the sphere is now illuminated and its true colour can be seen.

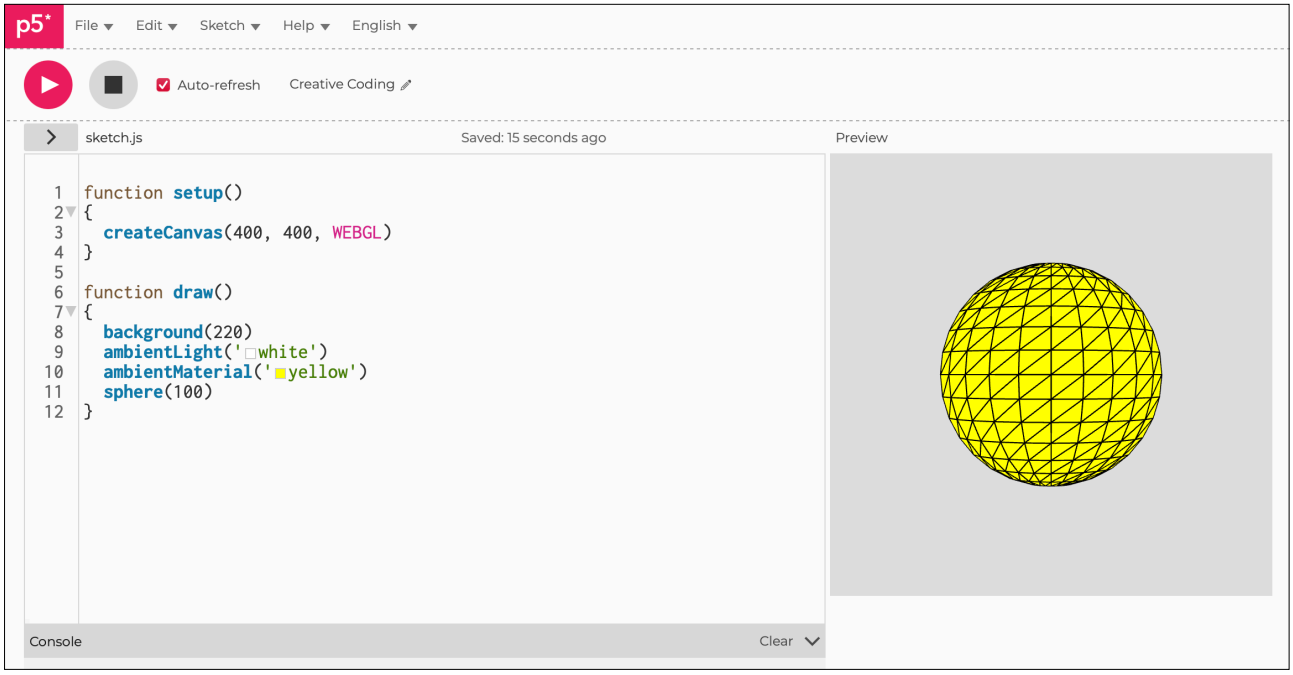
Challenge

Try other colour combinations.

Code Explanation

`ambientLight('white')`	A white ambient light.

Figure C3.4


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Creative Coding
sketch.js
Saved: 15 seconds ago
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
ambientLight('white')
ambientMaterial('yellow')
sphere(100)
}
Console
Clear



Sketch C3.5 ambient blue

But see what happens if we make the `ambientLight()` blue.

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

function draw()
{
  background(220)
  ambientLight('blue')
  ambientMaterial('yellow')
  sphere(100)
}
```

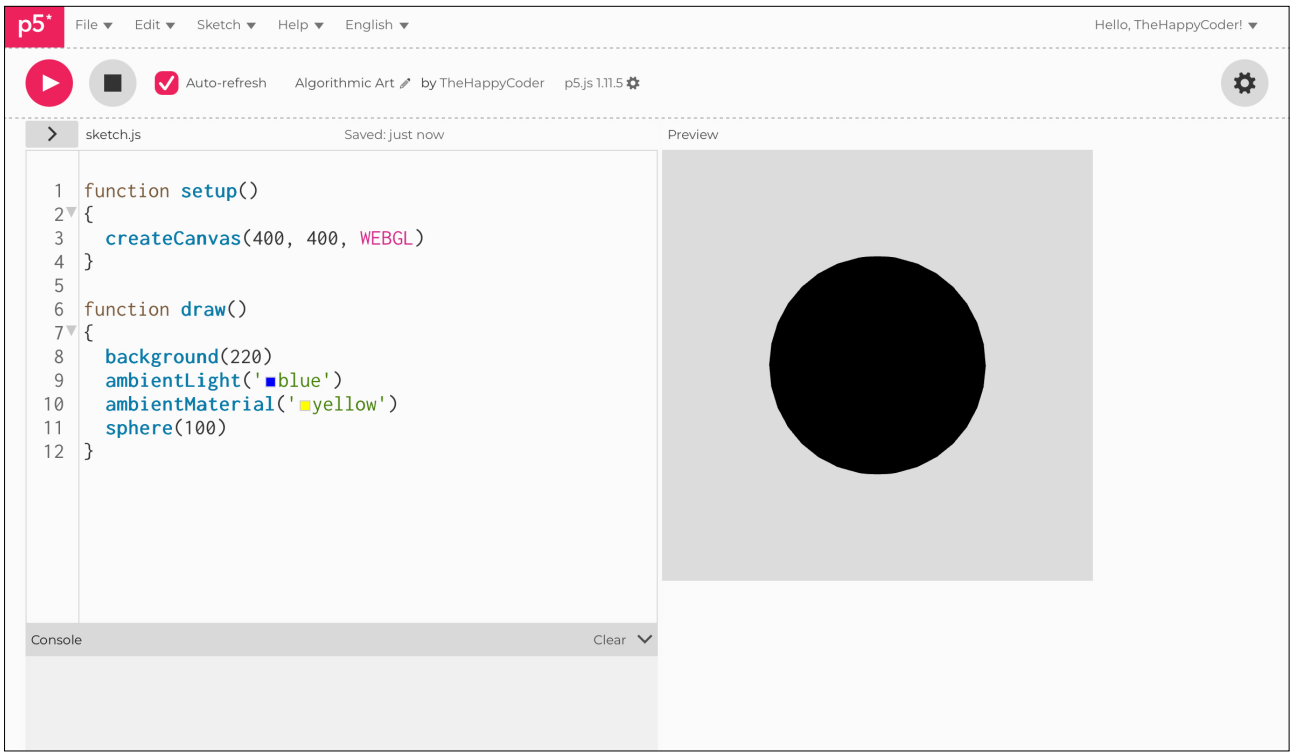
Notes

It is a black sphere; you have to be careful which colours you mix.

Challenges

1. Try `ambientLight('green')`
2. Try other colours

Figure C3.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
ambientLight('blue')
ambientMaterial('yellow')
sphere(100)
}
Console
Clear



Sketch C3.6 a green light on a white material

What would happen if we made the material white and shone a green light on it?

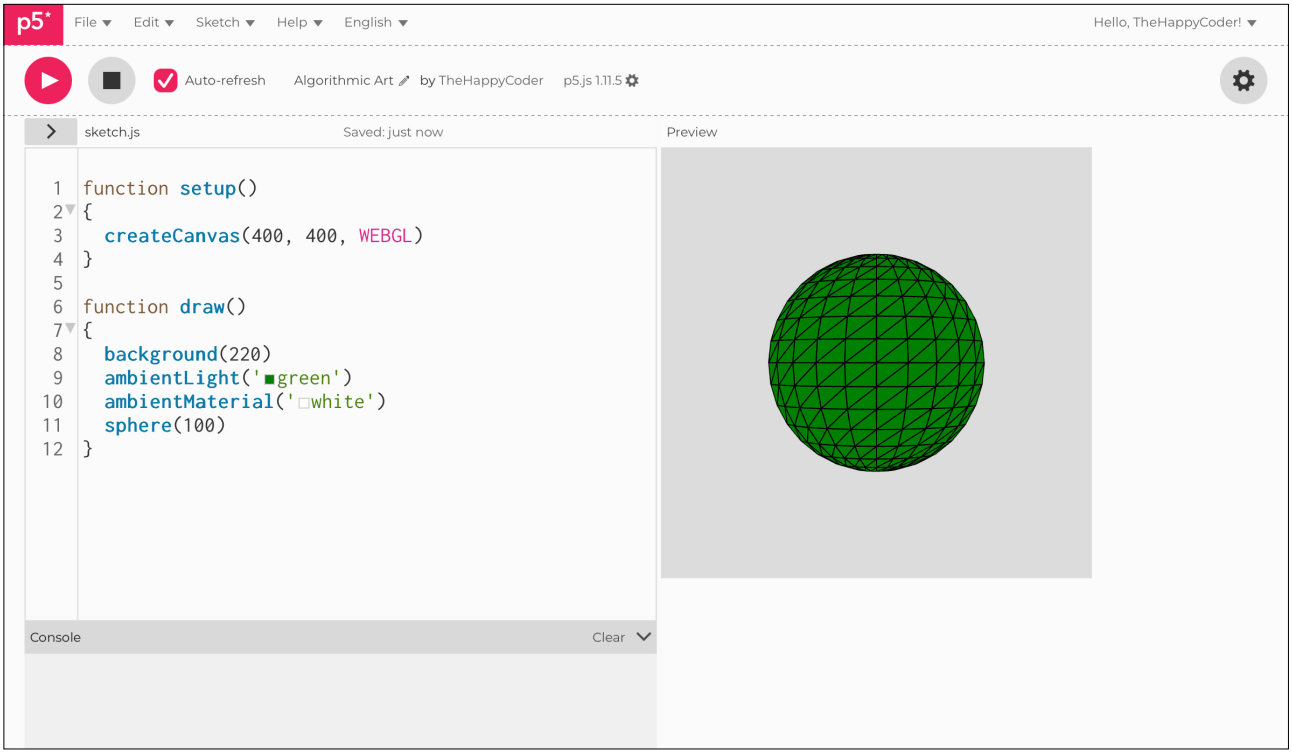
```
function setup()
{
  createCanvas(400, 400, WEBGL)
}

function draw()
{
  background(220)
  ambientLight('green')
  ambientMaterial('white')
  sphere(100)
}
```

Notes

It all depends on how much of one colour is in the other. If there is none of one in the other (light or material), then it will turn black. As you probably guess, ambient light shines from all directions, not from a source, and it is not coming from a particular direction, so this is why you get a consistent colour from every angle.

Figure C3.6


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
1 function setup()
2 {
3 createCanvas(400, 400, WEBGL)
4 }
5
6 function draw()
7 {
8 background(220)
9 ambientLight('green')
10 ambientMaterial('white')
11 sphere(100)
12 }
Console
Clear

Sketch C3.7 a point of light

! Remove the ambientLight() and the ambientMaterial().

An alternative is to have a point of light, and for this, we need to still give it a colour but also a position. The function pointLight() has six arguments: the first three are the RGB colours, and the second set of three are x, y, and z positions relative to the centre of the canvas space.

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

function draw()
{
  background(220)
  pointLight(0, 0, 255, -250, -200, 350)
  sphere(100)
}
```

Notes

We have positioned the light source to the left, slightly above and in front of the sphere. Remember that the positions are relative to the origin of the 3D space.

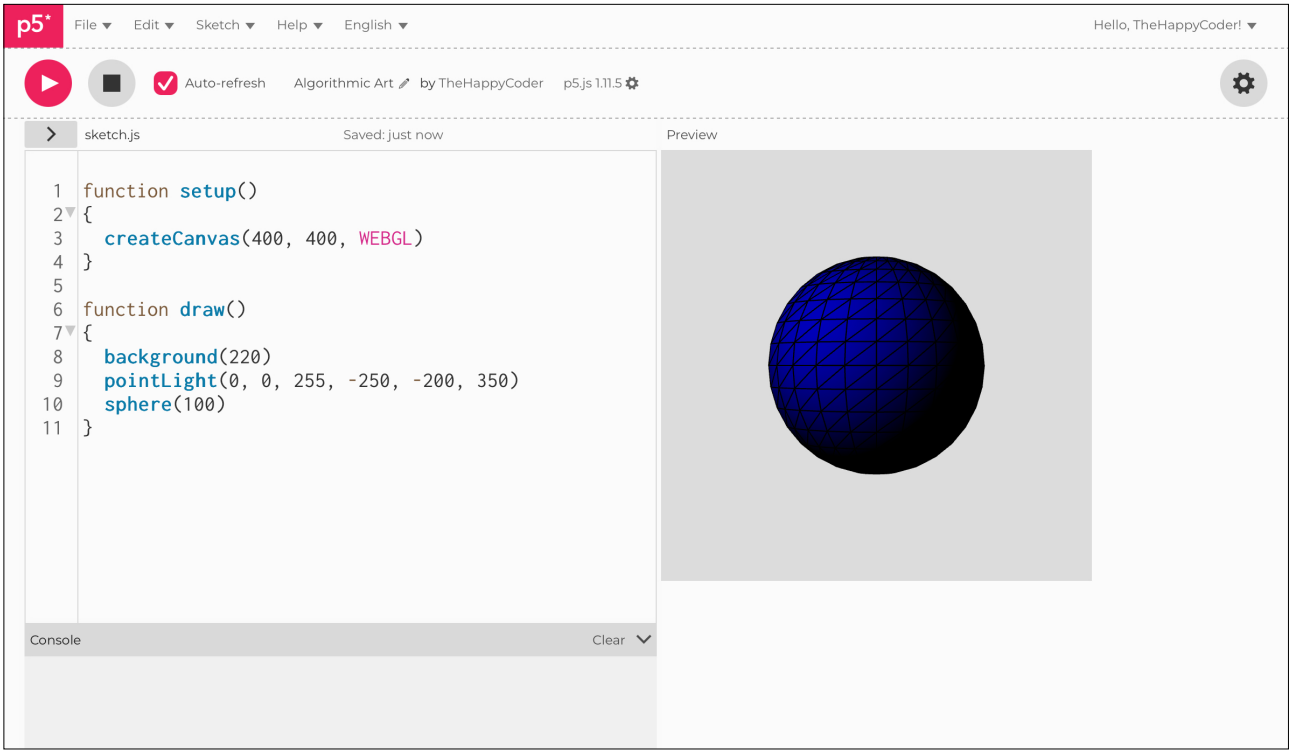
Challenge

Try other positions.

Code Explanation

`pointLight(0, 0, 255, -250, -200, 350)`	This has a colour part (three arguments) and a position part (also three arguments).

Figure C3.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
pointLight(0, 0, 255, -250, -200, 350)
sphere(100)
}
Console
Clear



Sketch C3.8 moving the light

If we give the x and y position the mouseX and mouseY, so you can see the point of light moving as you move the mouse. We subtract 200 because the mouse still thinks it is on a 2D canvas.

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

function draw()
{
  background(220)
  pointLight(0, 0, 255, mouseX - 200, mouseY - 200, 200)
  sphere(100)
}
```

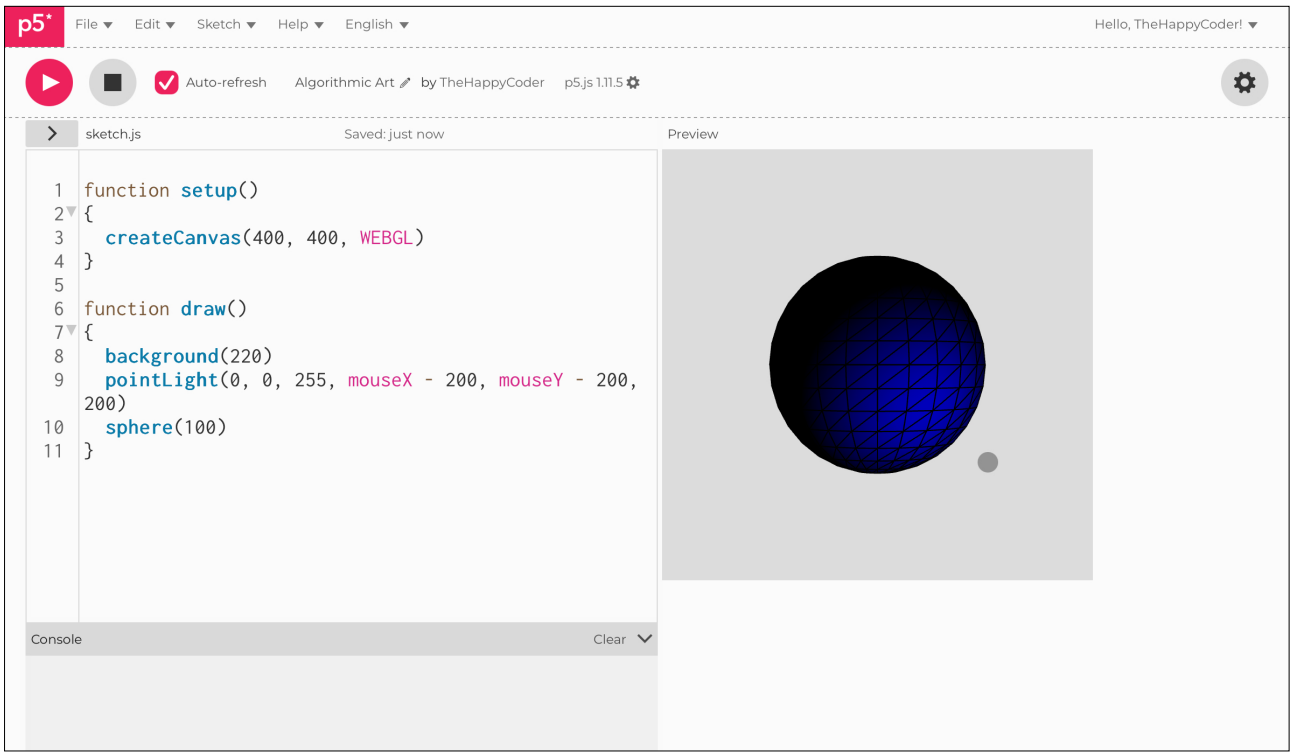
Notes

The light source follows the mouse. The results are a bit dim, though, but you will see that you can use them effectively.

Challenge

Have mouseX or mouseY move the z position.

Figure C3.8


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
1 function setup()
2 {
3 createCanvas(400, 400, WEBGL)
4 }
5
6 function draw()
7 {
8 background(220)
9 pointLight(0, 0, 255, mouseX - 200, mouseY - 200, 200)
10 sphere(100)
11 }
Console
Clear



Sketch C3.9 three points of light

Here, we are going to have three points of light and a different colour for each one. Adding a `noStroke()` removes the detail lines.

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

function draw()
{
  background(220)
  noStroke()
  pointLight(0, 0, 255, -200, -200, 200)
  pointLight(255, 0, 0, 200, 200, 200)
  pointLight(0, 255, 0, 200, -200, 200)
  sphere(100)
}
```

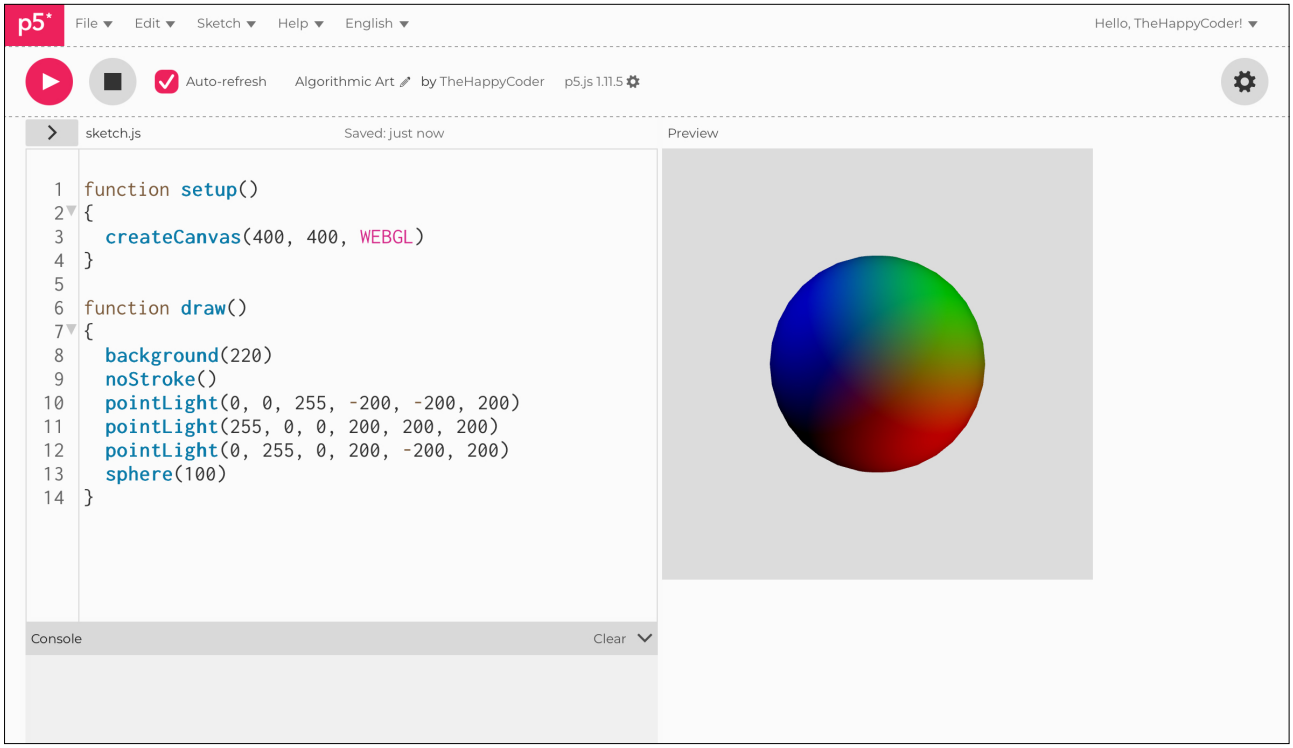
Notes

Creates an interesting effect, which is something you could play with.

Challenge

Add in `ambientLight(100)`.

Figure C3.9


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
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
noStroke()
pointLight(0, 0, 255, -200, -200, 200)
pointLight(255, 0, 0, 200, 200, 200)
pointLight(0, 255, 0, 200, -200, 200)
sphere(100)
}
Console
Clear

Sketch C3.10 metallic finish

We have a matte colour by default, but we can make the surface metallic or reflective. We still need to give it a base colour (in this case, white).

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

function draw()
{
  background(220)
  noStroke()
  pointLight(0, 0, 255, -200, -200, 200)
  pointLight(255, 0, 0, 200, 200, 200)
  pointLight(0, 255, 0, 200, -200, 200)
  specularMaterial(255)
  sphere(100)
}
```

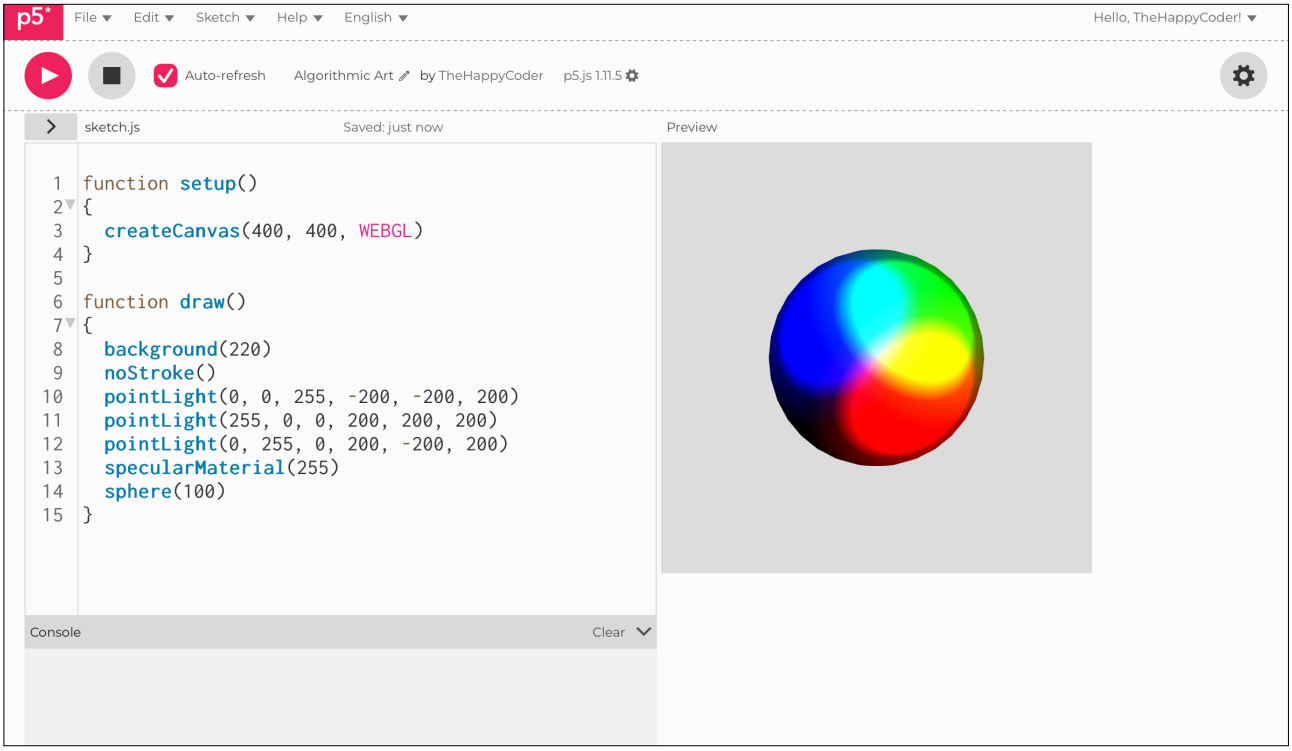
Notes

A bit overwhelming, but we can tone it down a bit.

Code Explanation

`specularMaterial(255)`	Brings a more reflective surface colour to the shape.

Figure C3.10


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
1 function setup()
2 {
3 createCanvas(400, 400, WEBGL)
4 }
5
6 function draw()
7 {
8 background(220)
9 noStroke()
10 pointLight(0, 0, 255, -200, -200, 200)
11 pointLight(255, 0, 0, 200, 200, 200)
12 pointLight(0, 255, 0, 200, -200, 200)
13 specularMaterial(255)
14 sphere(100)
15 }
Console
Clear



Sketch C3.11 more subtle

By reducing the amount of red, blue, and green, we can create a more subtle effect.

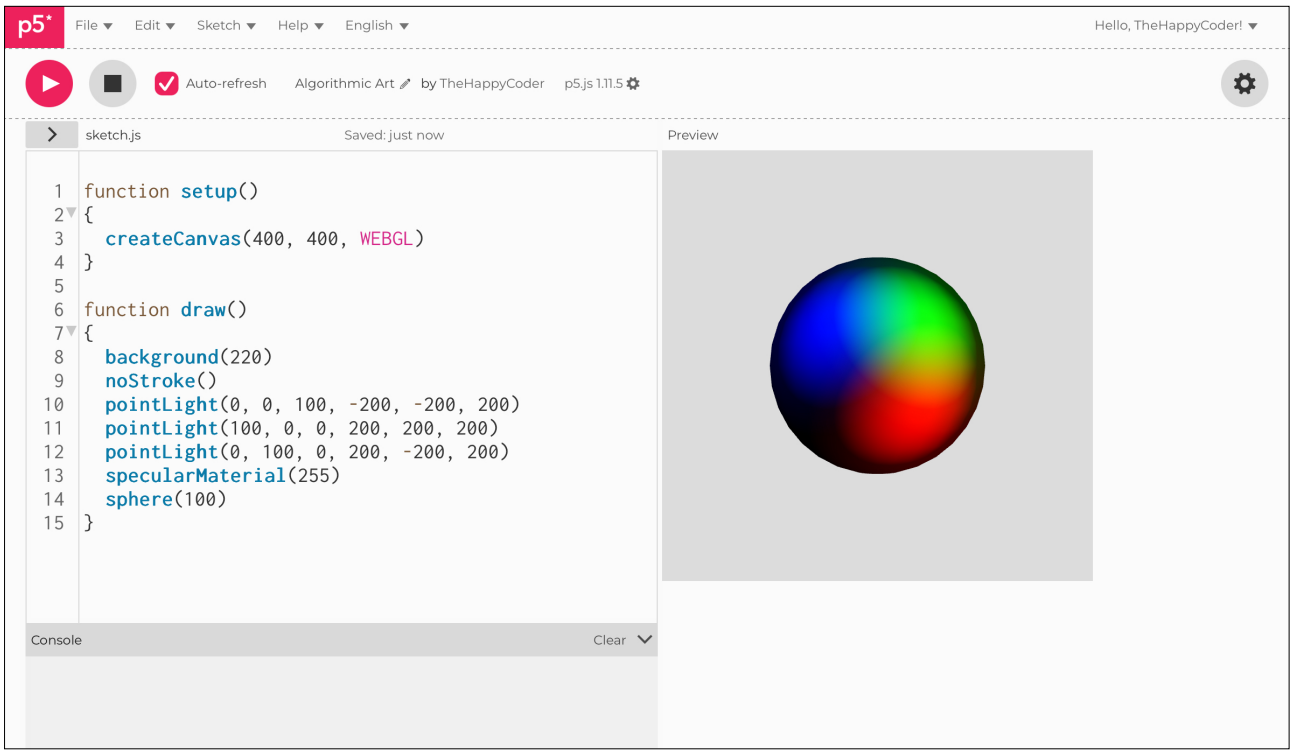
```
function setup()
{
  createCanvas(400, 400, WEBGL)
}

function draw()
{
  background(220)
  noStroke()
  pointLight(0, 0, 100, -200, -200, 200)
  pointLight(100, 0, 0, 200, 200, 200)
  pointLight(0, 100, 0, 200, -200, 200)
  specularMaterial(255)
  sphere(100)
}
```

Notes

That looks a bit more natural.

Figure C3.11


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
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
noStroke()
pointLight(0, 0, 100, -200, -200, 200)
pointLight(100, 0, 0, 200, 200, 200)
pointLight(0, 100, 0, 200, -200, 200)
specularMaterial(255)
sphere(100)
}
Console
Clear

Sketch C3.12 box and rotate

! Making a few changes to the sketch.

For a bit of fun, let's change back to a box and rotate.

```
let angle = 0

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

function draw()
{
  background(220)
  noStroke()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  pointLight(0, 0, 100, -200, -200, 200)
  pointLight(100, 0, 0, 200, 200, 200)
  pointLight(0, 100, 0, 200, -200, 200)
  specularMaterial(255)
  box(100)
  angle++
}
```

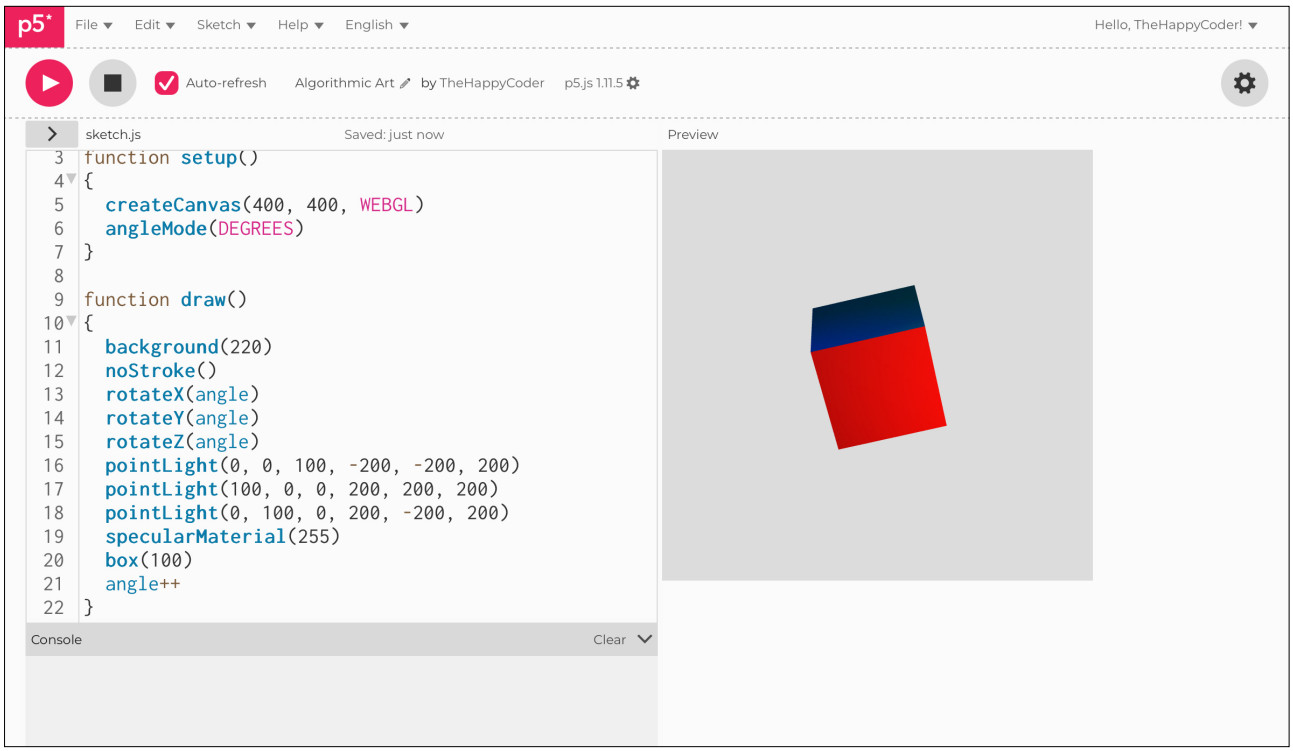
Notes

You can see the more reflective nature of the material with the cube rotating.

Challenges

1. Try stronger values for the red, green, and blue.
2. Try other shapes and positions of the light source.
3. Give the material different colours.

Figure C3.12


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
}
function draw()
{
background(220)
noStroke()
rotateX(angle)
rotateY(angle)
rotateZ(angle)
pointLight(0, 0, 100, -200, -200, 200)
pointLight(100, 0, 0, 200, 200, 200)
pointLight(0, 100, 0, 200, -200, 200)
specularMaterial(255)
box(100)
angle++
}
Console
Clear



Algorithmic Art
Module C
Unit #4
Lights and Materials
Part 2



Module C Unit #4: lights and materials #2

Quick recap:

Since we're working in 3D, we can position lights as separate entities rather than simply filling a shape with a colour (which is also possible). With just a little practice, we can create many pleasing results. There are many light options and materials to choose from, and by combining the right ones, you can create some pleasing effects that don't always work as you might expect.

```
normalMaterial()
ambientMaterial()
specularMaterial()
emissiveMaterial()
ambientLight()
pointLight()
directionalLight()
lights()
```

Sketch C4.1 new sketch new lights

! Start a new sketch

We are going to introduce directional light. It is similar to the point light because it has six arguments: the first three are the RGB colours and the next three are the vectors which tell you the direction but not the position. They have one of three values: `1`, `0`, and `-1`. Let's start with a basic sketch again and demonstrate it. We get a white sphere with no detail.

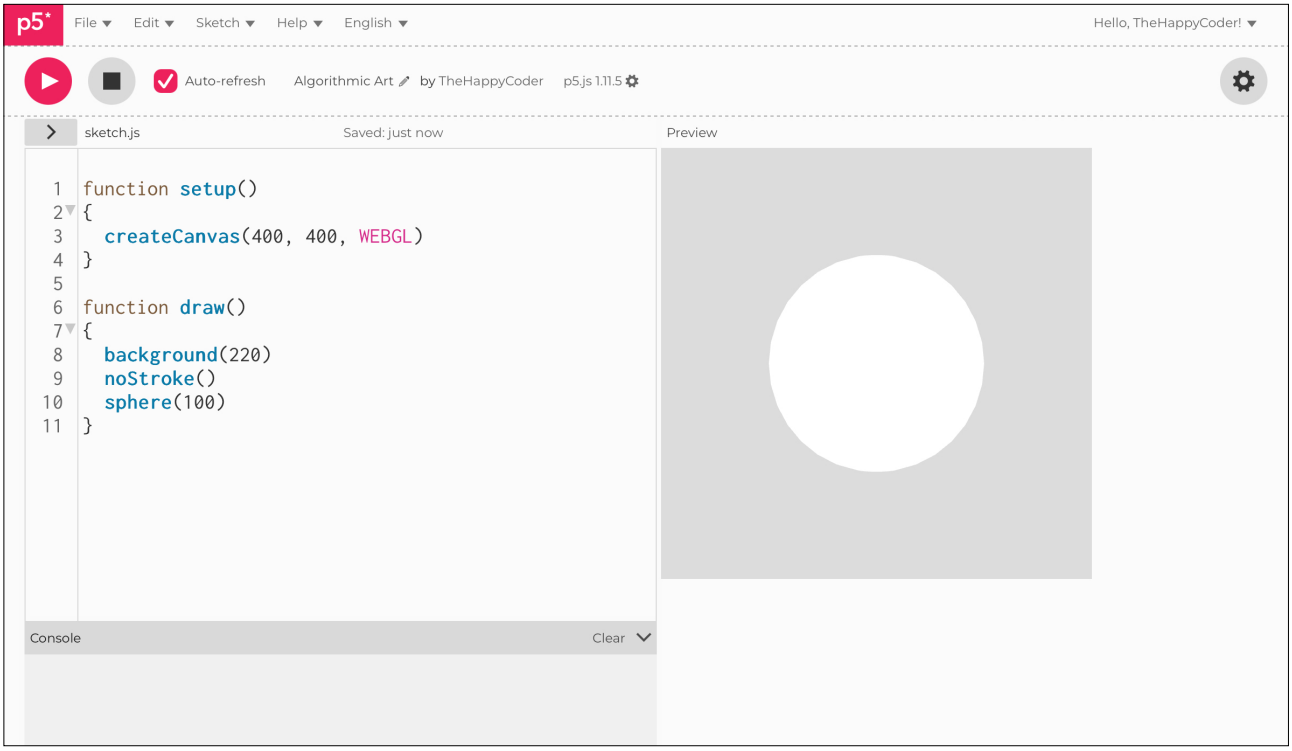
```
function setup()
{
  createCanvas(400, 400, WEBGL)
}

function draw()
{
  background(220)
  noStroke()
  sphere(100)
}
```

Notes

Just a white, indistinct sphere.

Figure C4.1


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400, WEBGL)
4 }
5
6 function draw()
7 {
8 background(220)
9 noStroke()
10 sphere(100)
11 }
Console
Clear

Sketch C4.2 a more directional light

Now, add the directional light (a nice yellow). The `directionalLight()` has six arguments.

The first three are the RGB (depending on the `colourMode()`), the next three require a little attention but are quite logical. They are:
The x direction with values between −1 and +1
The y direction with values between −1 and +1
The z direction with values between −1 and +1

As an example for the x direction, a value of +1 shines from the left-hand side, whereas −1 the light shines from the right-hand side (see fig. C4.2). Then, the y direction +1 is from above, whereas −1 is from below; for the z value +1 is from behind, −1 is from in front. A value of zero seems to switch them off altogether.

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

function draw()
{
  background(220)
  directionalLight(255, 255, 0, -1, 0, 0)
  noStroke()
  sphere(100)
}
```

Notes

The light in this instance is coming from the right (x is −1). It is best to play with these values to get a feel for them.

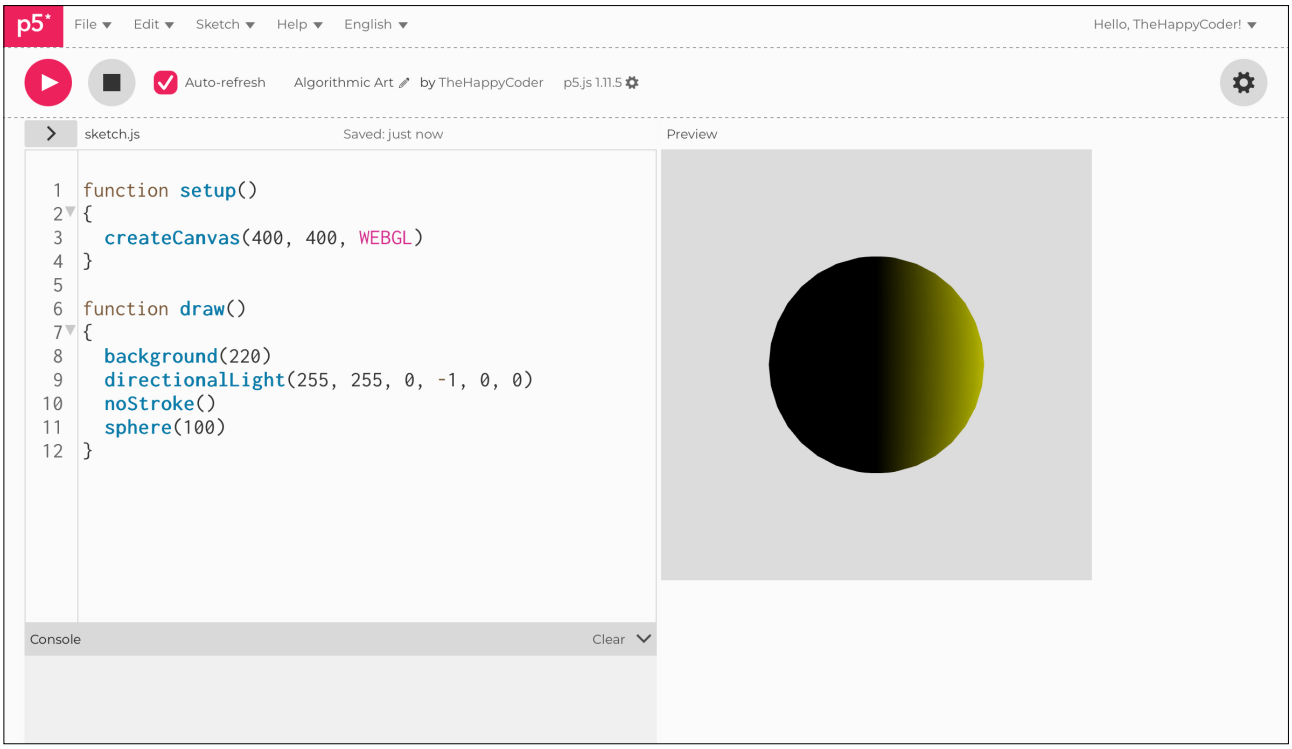
Challenges

1. Change the values from 1 to 0 to −1.
2. Change the strength of the colour.
3. Add more directional lights from different directions (maximum number is five).
4. Add more directional lights from the same direction.
5. Use different colours on the same position.

Code Explanation

`directionalLight(255, 255, 0, -1, 0, 0)`	A yellow directional light pointing from the right.

Figure C4.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400, WEBGL)
4 }
5
6 function draw()
7 {
8 background(220)
9 directionalLight(255, 255, 0, -1, 0, 0)
10 noStroke()
11 sphere(100)
12 }
Console
Clear

Sketch C4.3 just lights!

! Removing the sphere() replacing with a box() and adding some rotation and shiny material.

One final option is a function called lights() which is a mixture of ambient and directional light. It's a quick throw-in so that you don't have to think about direction or position.

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

function draw()
{
  background(220)
  lights()
  specularMaterial(50)
  rotateX(frameCount/50)
  rotateY(frameCount/50)
  rotateZ(frameCount/50)
  noStroke()
  box(100)
}
```

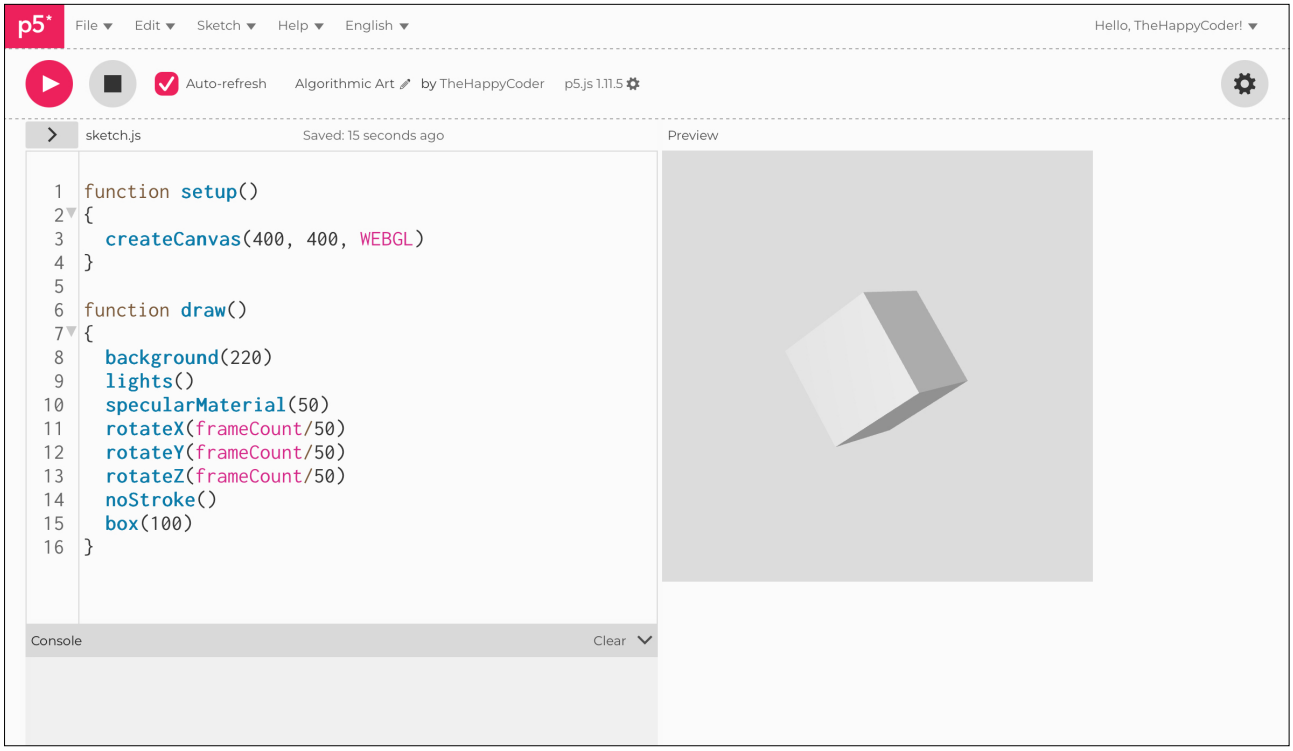
Notes

This shows how you can quickly show your 3D graphics at work.

Code Explanation

lights()	It has ambient and directional light built in.

Figure C4.3


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
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
lights()
specularMaterial(50)
rotateX(frameCount/50)
rotateY(frameCount/50)
rotateZ(frameCount/50)
noStroke()
box(100)
}
Console
Clear



Sketch C4.4 basic torus

Here we have a torus with the default level of detail (24, 16).

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

function draw()
{
  background(220)
  lights()
  specularMaterial(50)
  rotateX(frameCount/50)
  rotateY(frameCount/50)
  rotateZ(frameCount/50)
  noStroke()
  torus(100, 25)
}
```

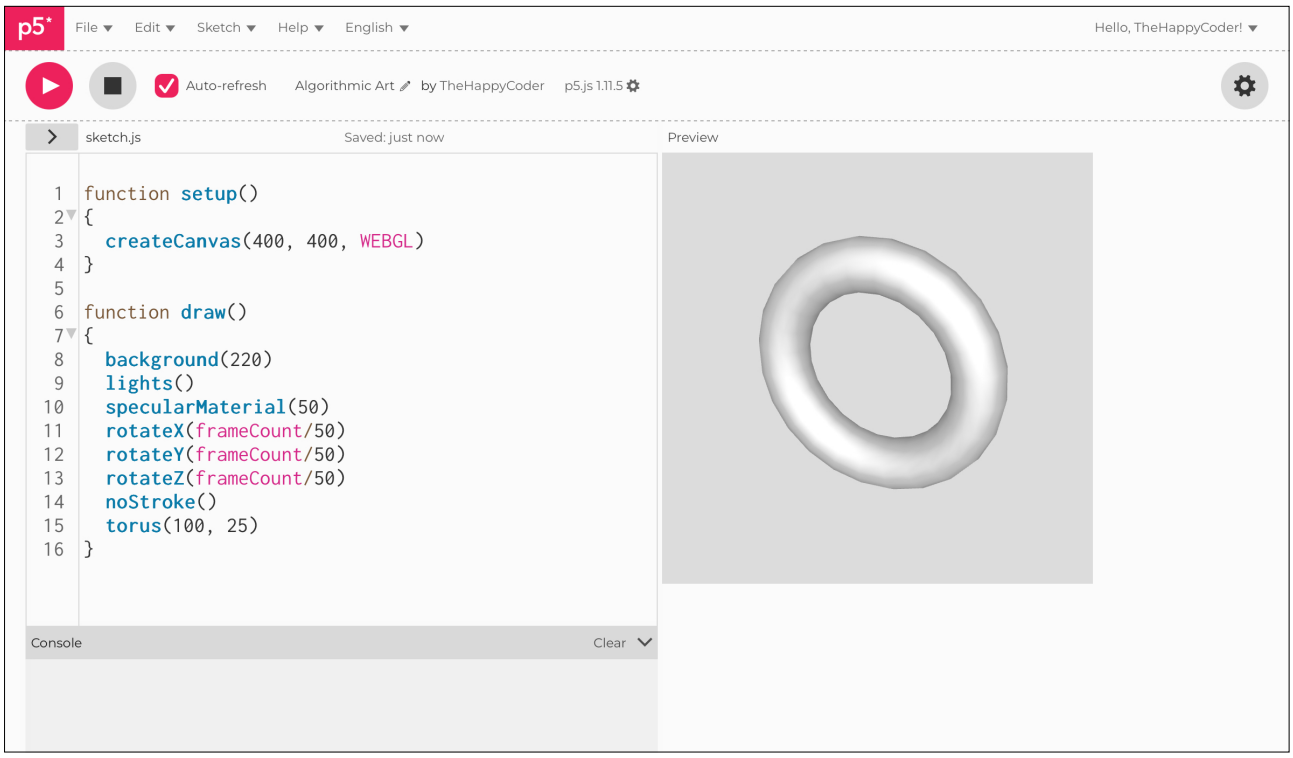
Notes

We have our standard torus.

Challenge

Make the background darker to highlight the effect.

Figure C4.4


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
lights()
specularMaterial(50)
rotateX(frameCount/50)
rotateY(frameCount/50)
rotateZ(frameCount/50)
noStroke()
torus(100, 25)
}
Console
Clear

Sketch C4.5 more detail added

Increasing the amount of detail to see the difference compared to the default.

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

function draw()
{
  background(220)
  lights()
  specularMaterial(50)
  rotateX(frameCount/50)
  rotateY(frameCount/50)
  rotateZ(frameCount/50)
  noStroke()
  torus(100, 25, 48, 32)
}
```

Notes

It makes it much smoother

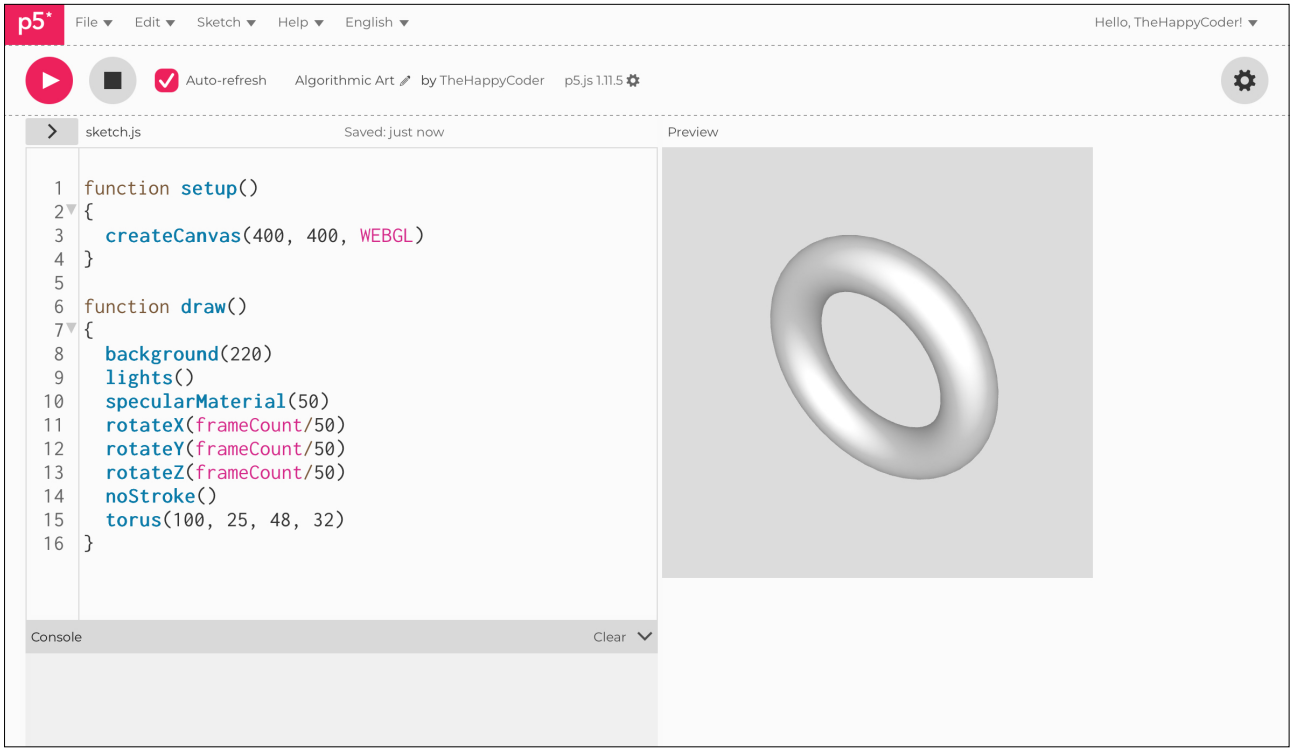
Challenges

1. Try higher values
2. Try it with other shapes

Code Explanation

`torus(100, 25, 48, 32)`	A torus shape with double the detail.

Figure C4.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
}
function draw()
{
background(220)
lights()
specularMaterial(50)
rotateX(frameCount/50)
rotateY(frameCount/50)
rotateZ(frameCount/50)
noStroke()
torus(100, 25, 48, 32)
}
Console
Clear

Sketch C4.6 orbiting object

! New sketch

We have a large sphere in the centre and a smaller sphere orbiting it. The `translate()` function scribes a circular motion, think circle equation with `sin()` and `cos()`. The radius of the orbit is `120`.

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

function draw()
{
  background(220)
  ambientLight(255)
  push()
  translate(-120 * sin(frameCount / 30), 0, -120 * cos(frameCount / 30))
  sphere(10)
  pop()
  sphere(50)
}
```

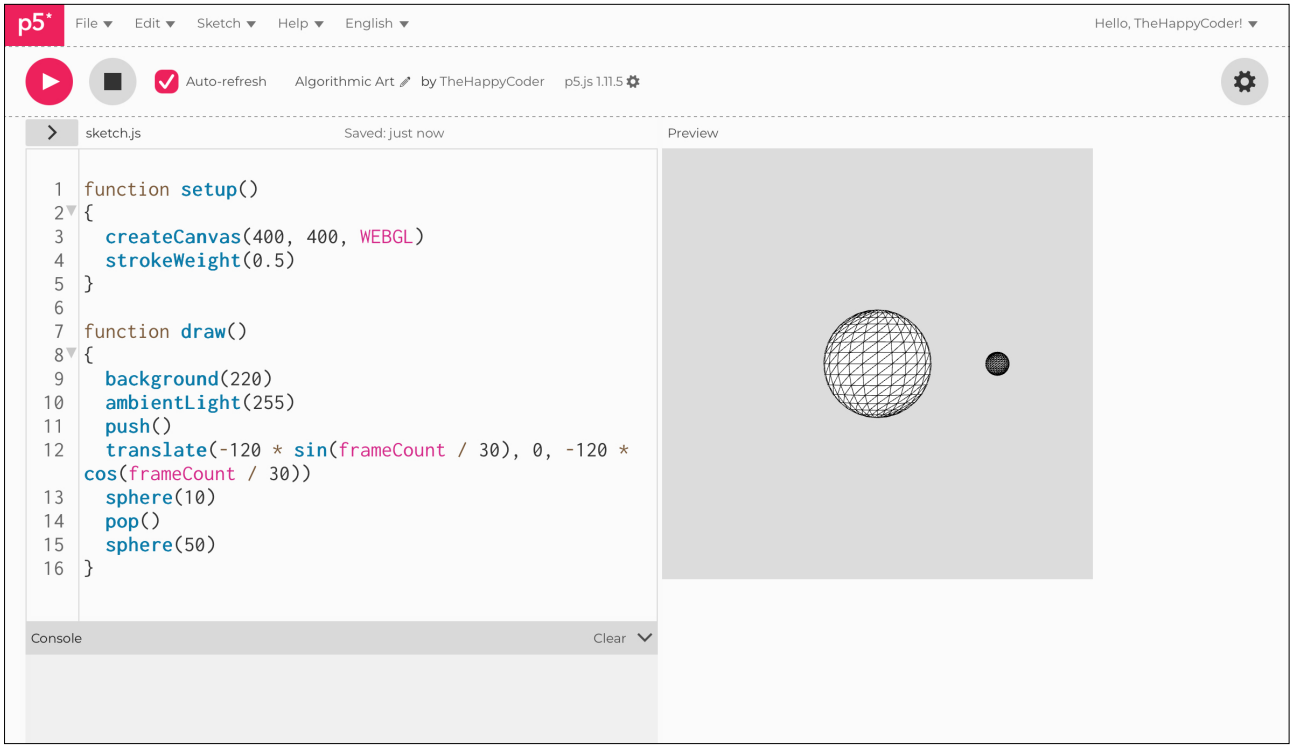
Notes

The rotation is on the `x` axis and the `z` axis, if it were on the `x` and `y` axis it would make more sense. We `push()` and `pop()` it so that the large sphere doesn't start orbiting as well.

Code Explanation

`translate(-120 * sin(frameCount / 30), 0, -120 * cos(frameCount / 30))`	This is the motion of the small sphere; it describes a circular motion.

Figure C4.6


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)
strokeWeight(0.5)
}
function draw()
{
background(220)
ambientLight(255)
push()
translate(-120 * sin(frameCount / 30), 0, -120 *
cos(frameCount / 30))
sphere(10)
pop()
sphere(50)
}
Console
Clear



Sketch C4.7 rotating directional light

We now add in the directional light, which mirrors the movement of the small sphere. We reduce the ambient light and give it a black background.

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

function draw()
{
  background(0)
  ambientLight(100)
  directionalLight(200, 200, 0, sin(frameCount / 30), 0, cos(frameCount / 30))
  push()
  translate(-120 * sin(frameCount / 30), 0, -120 * cos(frameCount / 30))
  sphere(10)
  pop()
  sphere(50)
}
```

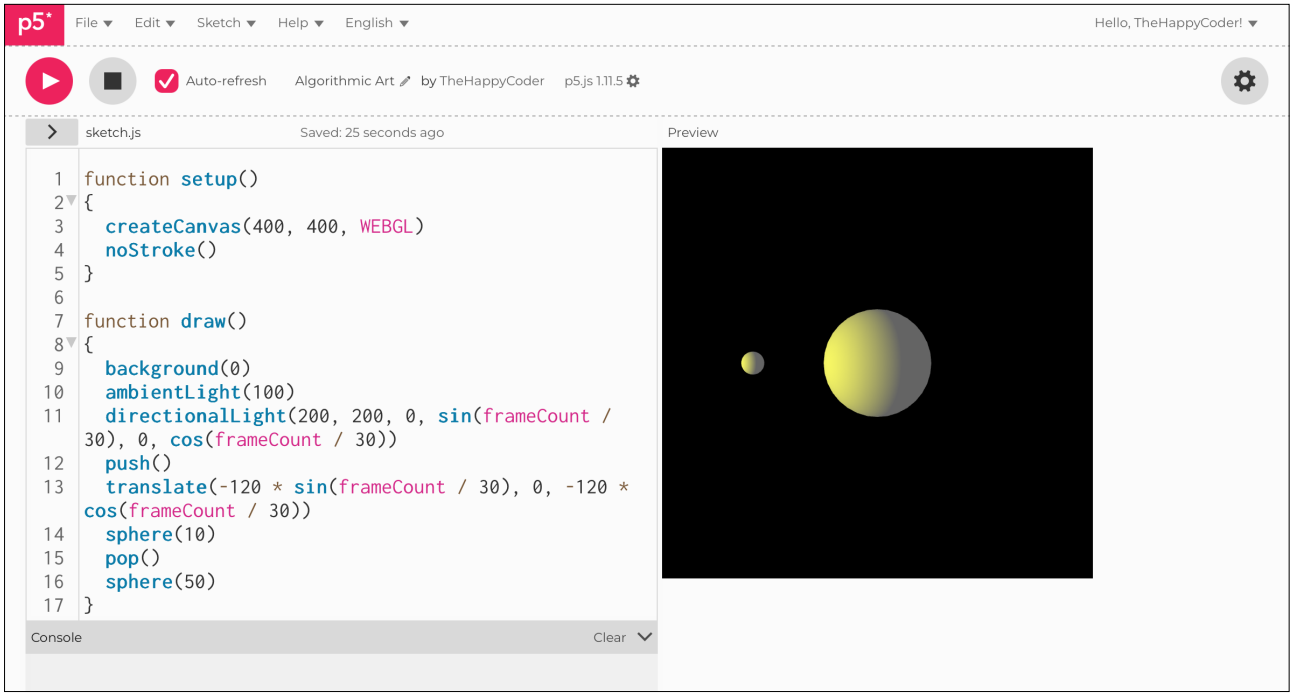
Notes

The vectors (the last three arguments) change direction according to the position of the sphere; they still return values between −1 and +1.

Code Explanation

`directionalLight(200, 200, 0,` `sin(frameCount / 30), 0,` `cos(frameCount / 30))`	The directional light changes direction as it navigates the larger sphere.

Figure C4.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: 25 seconds ago
Preview
function setup()
{
createCanvas(400, 400, WEBGL)
noStroke()
}
function draw()
{
background(0)
ambientLight(100)
directionalLight(200, 200, 0, sin(frameCount / 30), 0, cos(frameCount / 30))
push()
translate(-120 * sin(frameCount / 30), 0, -120 * cos(frameCount / 30))
sphere(10)
pop()
sphere(50)
}
Console
Clear



Sketch C4.8 final trick up our sleeve

Adding in a new material called `emissiveMaterial()`, this gives the appearance of glowing but doesn't actually emit light. It takes three arguments.

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

function draw()
{
  background(0)
  ambientLight(100)
  directionalLight(200, 200, 0, sin(frameCount / 30), 0, cos(frameCount / 30))
  push()
  translate(-120 * sin(frameCount / 30), 0, -120 * cos(frameCount / 30))
  emissiveMaterial(200, 200, 0)
  sphere(10)
  pop()
  sphere(50)
}
```

Notes

We give it the same colour as the directional light; if you don't, you may get the light on the small sphere as well.

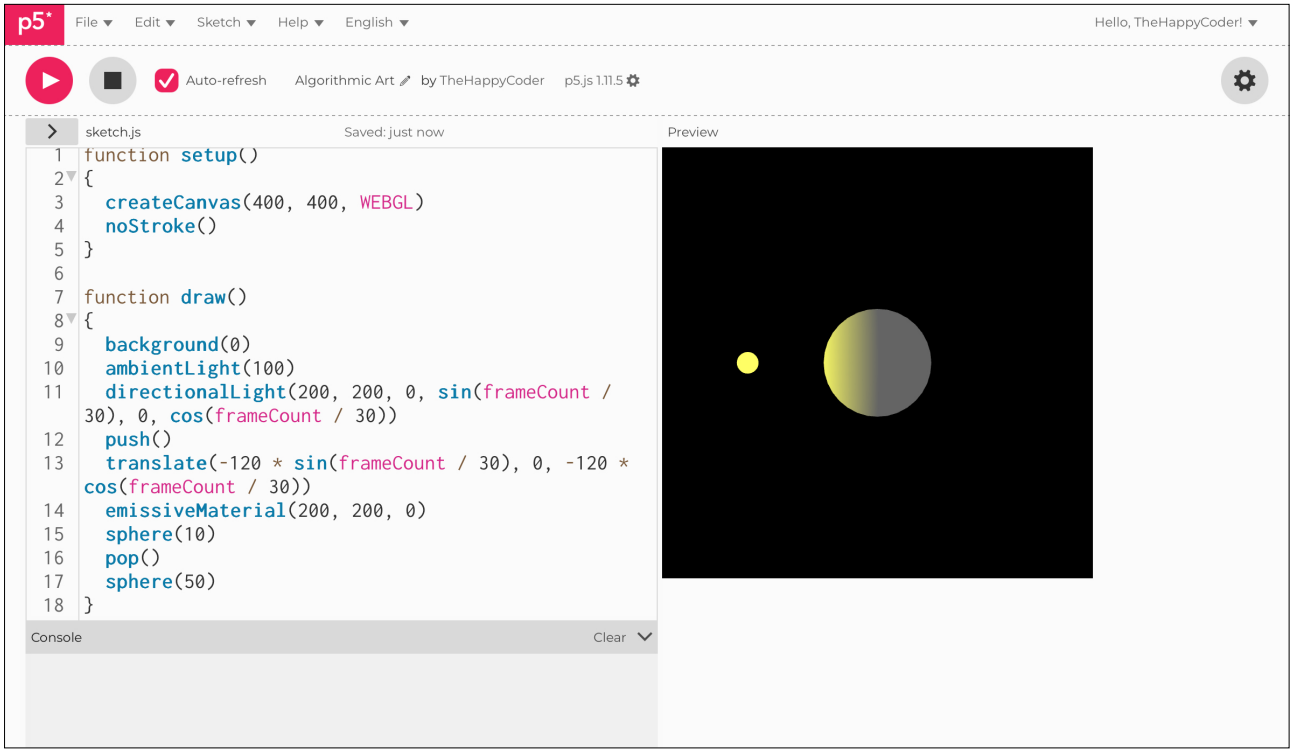
Challenge

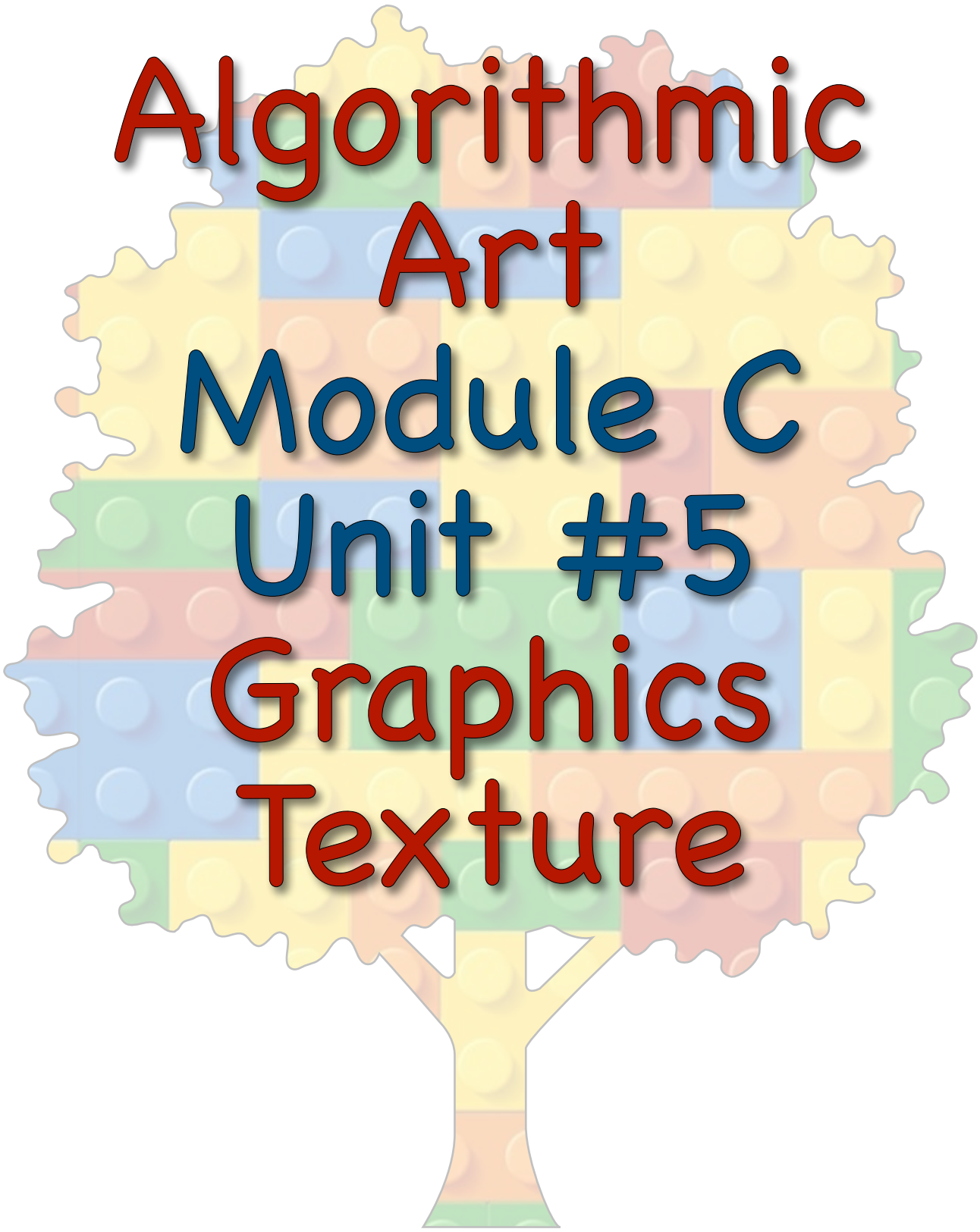
I will leave it to you to explore further.

Code Explanation

`emissiveMaterial(200, 200, 0)`	Makes the shape glow; you select the colour you want.

Figure C4.8


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
function setup()
{
createCanvas(400, 400, WEBGL)
noStroke()
}
function draw()
{
background(0)
ambientLight(100)
directionalLight(200, 200, 0, sin(frameCount / 30), 0, cos(frameCount / 30))
push()
translate(-120 * sin(frameCount / 30), 0, -120 * cos(frameCount / 30))
emissiveMaterial(200, 200, 0)
sphere(10)
pop()
sphere(50)
}
Console
Clear


Algorithmic
Art
Module C
Unit #5
Graphics
Texture



Module C Unit #5: graphics texture

This is a powerful function that is also extremely fun (in my opinion). It gives you the opportunity to put images, photos, videos and even draw on the faces of a shape while it is moving. For this unit, we are just keeping it simple to add shapes, text and colour for now.



Sketch C5.1 starting sketch

Our starting sketch gives us a nice rotating cube.

```
let angle = 0

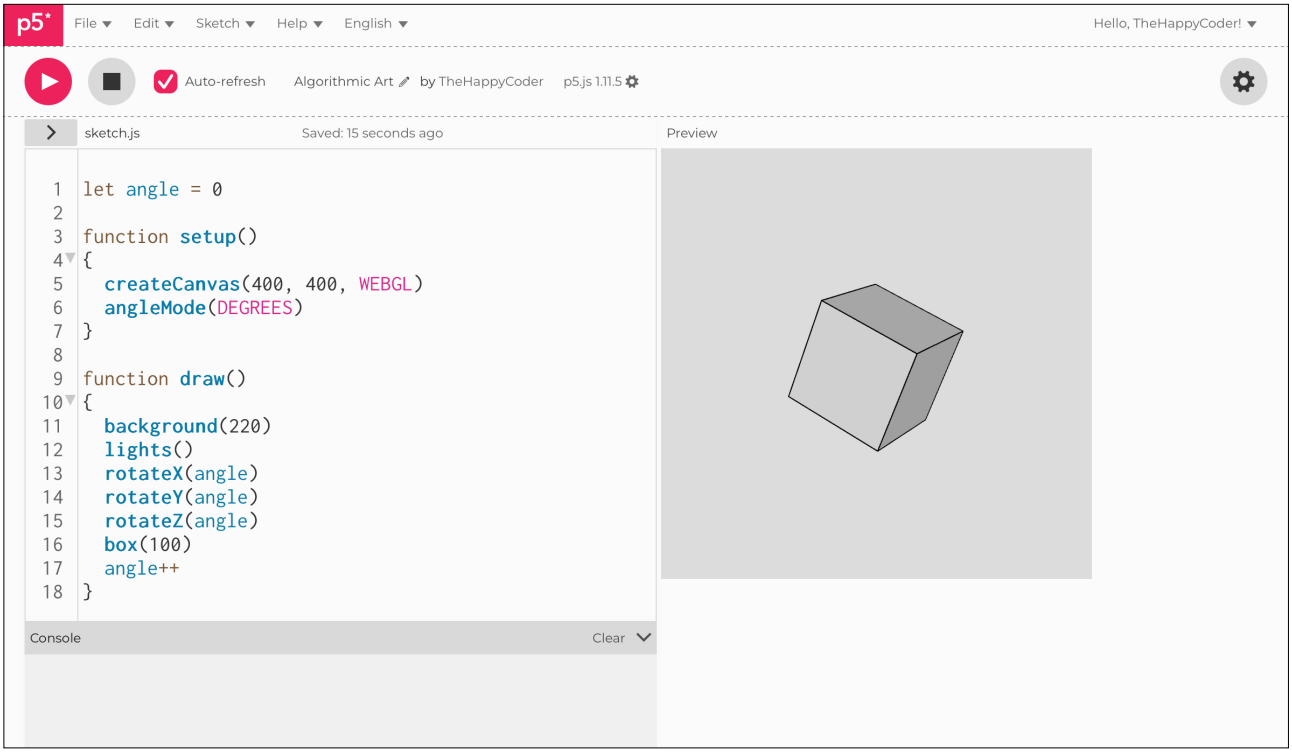
function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
}

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

Notes

A simple rotating cube

Figure C5.1


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)
lights()
rotateX(angle)
rotateY(angle)
rotateZ(angle)
box(100)
angle++
}
Console
Clear

Sketch C5.2 applying texture

We create a square object 400 by 400 and apply it as a texture to the box. The texture() function wraps an image onto a regular, primitive shape. Effectively, we have created and added another canvas onto the sides of the cube.

```
let angle = 0
let graphics

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

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

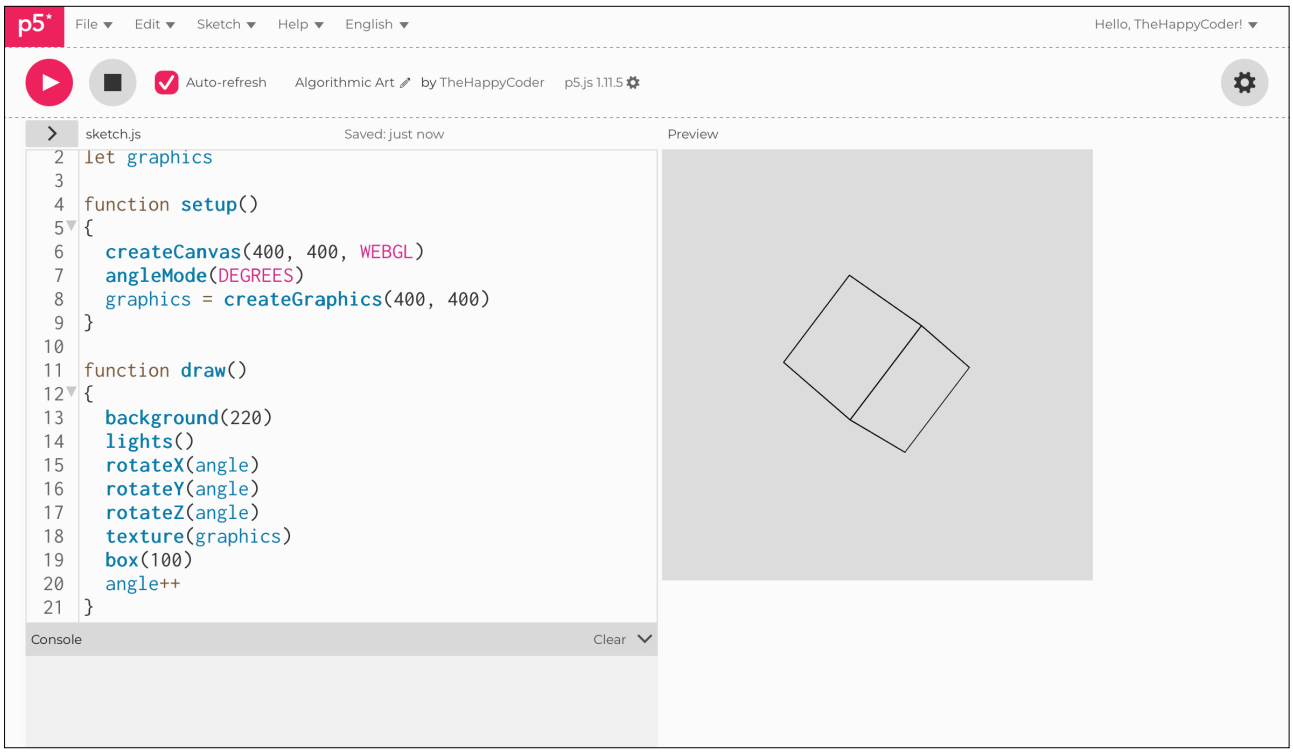
Notes

We haven't done anything with the texture as such, so it is a blank canvas. It is difficult to try to explain in words what is happening; that is why it is better to do it and see what it can produce. It makes it more intuitive than academic.

Code Explanation

`let graphics`	Variable to hold the graphics object.
`graphics = createGraphics(400, 400)`	Creating the graphics object and giving it a size of 400 x 400.
`texture(graphics)`	Putting the texture onto the cube with the graphics object.

Figure C5.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let graphics
function setup()
{
createCanvas(400, 400, WEBGL)
angleMode(DEGREES)
graphics = createGraphics(400, 400)
}
function draw()
{
background(220)
lights()
rotateX(angle)
rotateY(angle)
rotateZ(angle)
texture(graphics)
box(100)
angle++
}
Console
Clear



Sketch C5.3 separate canvas

Now we can manipulate the texture as a separate canvas. Starting simple, we give it a yellow background.

```
let angle = 0
let graphics

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

function draw()
{
  background(220)
  graphics.background(200, 200, 0)
  lights()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  texture(graphics)
  box(100)
  angle++
}
```

Notes

We now have a yellow cube; we haven't filled it with a colour but treated each side as a canvas and given it a background colour. We call the new `background()` function onto the `graphics` object.

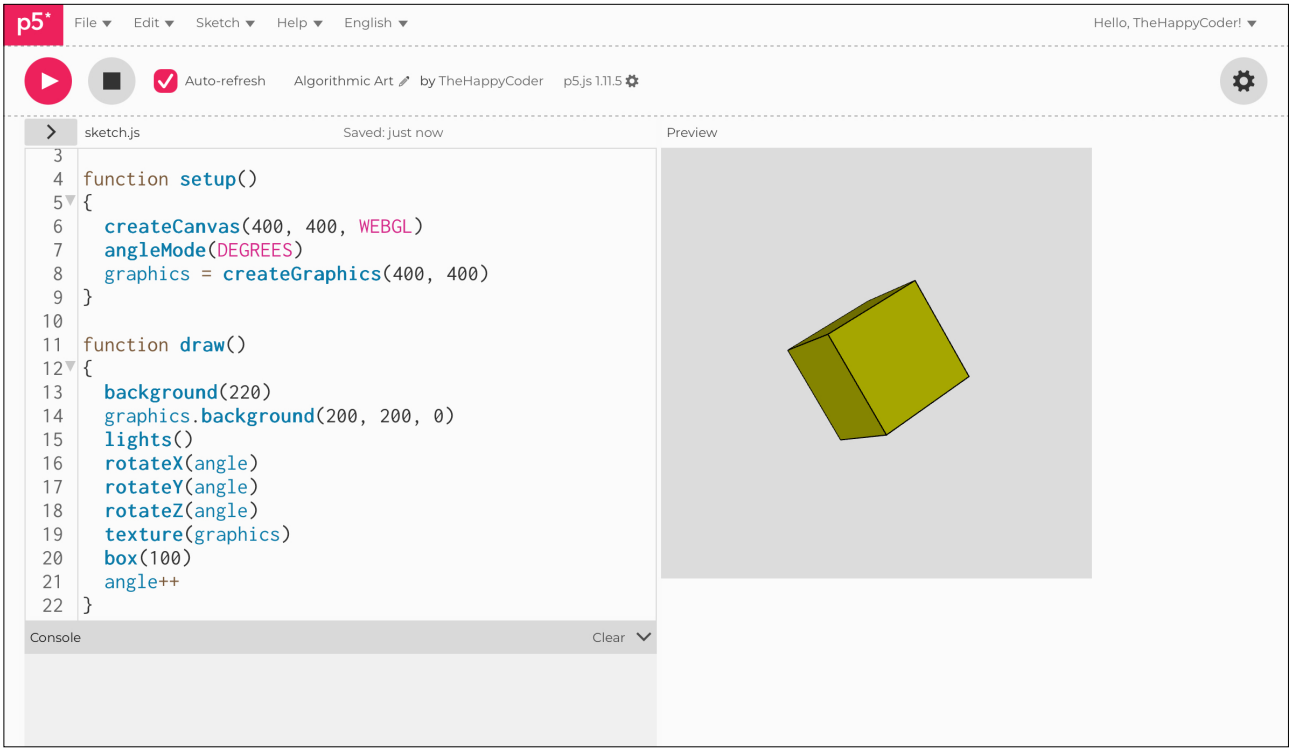
Challenges

1. Change the colour.
2. Try `graphics.fill()` to see if that does anything.

Code Explanation

`graphics.background(200, 200, 0)`	Gives the graphics canvas a background.

Figure C5.3


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
3
4 function setup()
5 {
6 createCanvas(400, 400, WEBGL)
7 angleMode(DEGREES)
8 graphics = createGraphics(400, 400)
9 }
10
11 function draw()
12 {
13 background(220)
14 graphics.background(200, 200, 0)
15 lights()
16 rotateX(angle)
17 rotateY(angle)
18 rotateZ(angle)
19 texture(graphics)
20 box(100)
21 angle++
22 }
Console
Clear

Sketch C5.4 adding graphics

We can draw shapes on the side of the cube; in this case, we can draw a circle which appears on all the surfaces of the box.

```
let angle = 0
let graphics

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

function draw()
{
  background(220)
  graphics.background(200, 200, 0)
  graphics.circle(100, 100, 150)
  lights()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  texture(graphics)
  box(100)
  angle++
}
```

Notes

Notice that the circle really does appear on each side of the cube; this shows that the texture is applied separately to each side of the cube. The size of the canvas (and circle) is scaled to fit.

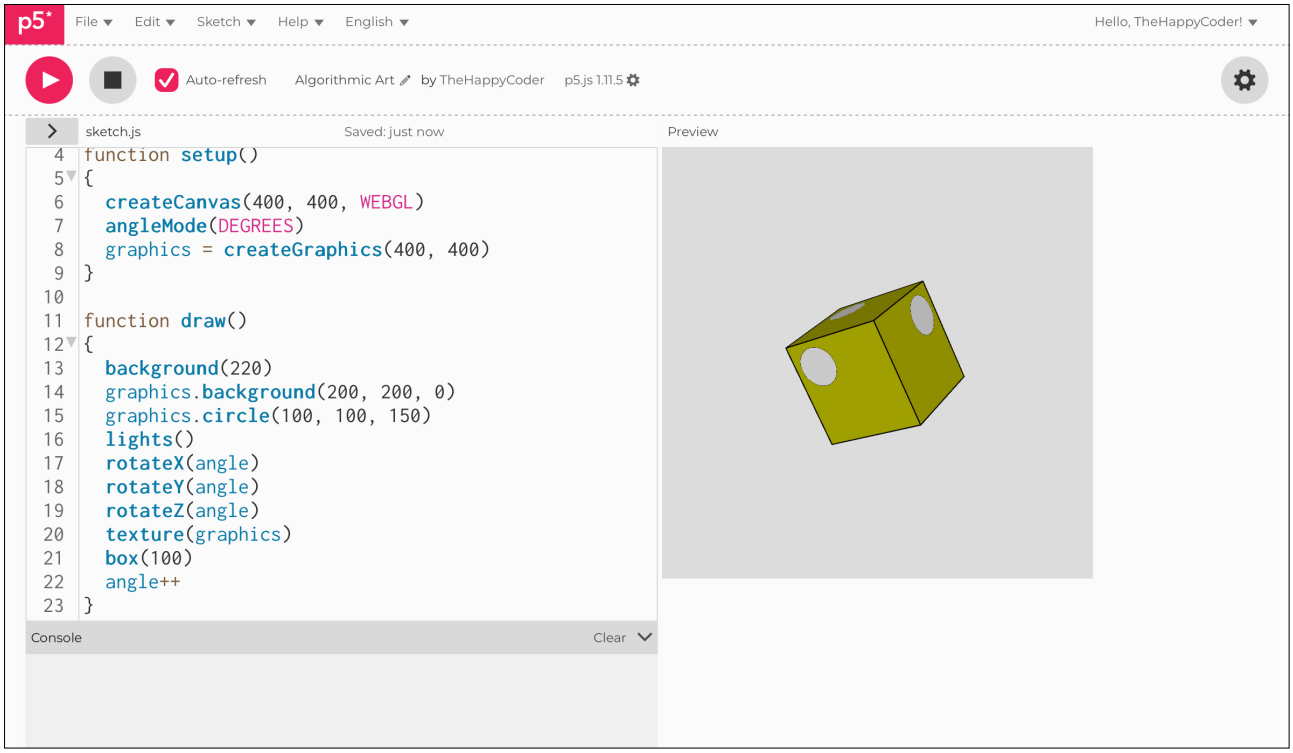
Challenges

1. Change the size of the graphics to (100, 400).
2. Try other 2D shapes.

Code Explanation

`graphics.circle(100, 100, 150)`	Creating a circle on the graphics canvas, 100 in, 100 down and radius 150.

Figure C5.4



Sketch C5.5 drawing on the sides

We can fill the circle with a colour (red in this case), and we can also have the circle move across the sides of the box as you move your mouse across the main canvas.

```
let angle = 0
let graphics

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

function draw()
{
  background(220)
  graphics.background(200, 200, 0)
  graphics.fill(255, 0, 0)
  graphics.circle(mouseX, mouseY, 150)
  lights()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  texture(graphics)
  box(100)
  angle++
}
```

Notes

The circle follows the mouse across the main canvas.

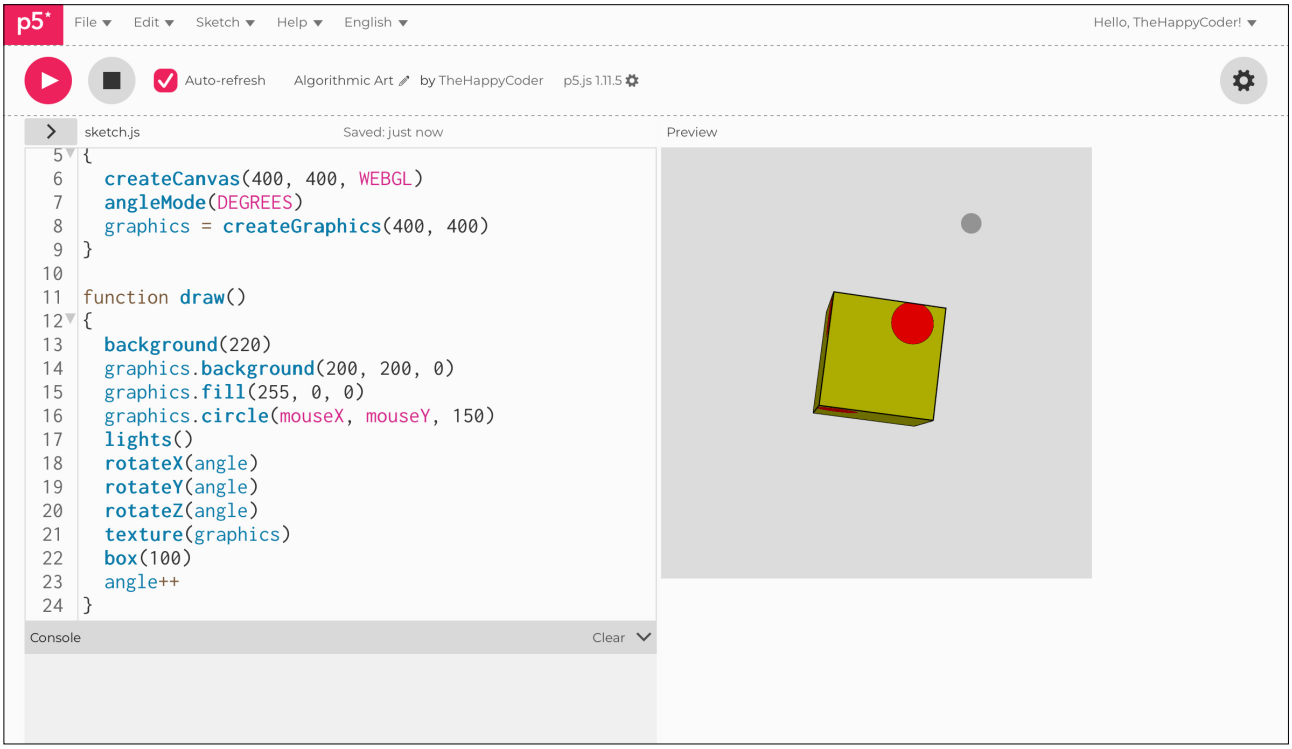
Challenge

Could you get the circle to bounce off the edges?

Code Explanation

`graphics.fill(255, 0, 0)`	Fills the circle with red colour.
`graphics.circle(mouseX, mouseY, 150)`	Gives the x and y coordinates of the mouse.

Figure C5.5





Sketch C5.6 a bit of fun tweaking

Watch what happens when we comment out the graphics background and add in `noStroke()`.

```
let angle = 0
let graphics

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

function draw()
{
  background(220)
  // graphics.background(200, 200, 0)
  graphics.fill(255, 0, 0)
  graphics.circle(mouseX, mouseY, 150)
  noStroke()
  lights()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  texture(graphics)
  box(100)
  angle++
}
```

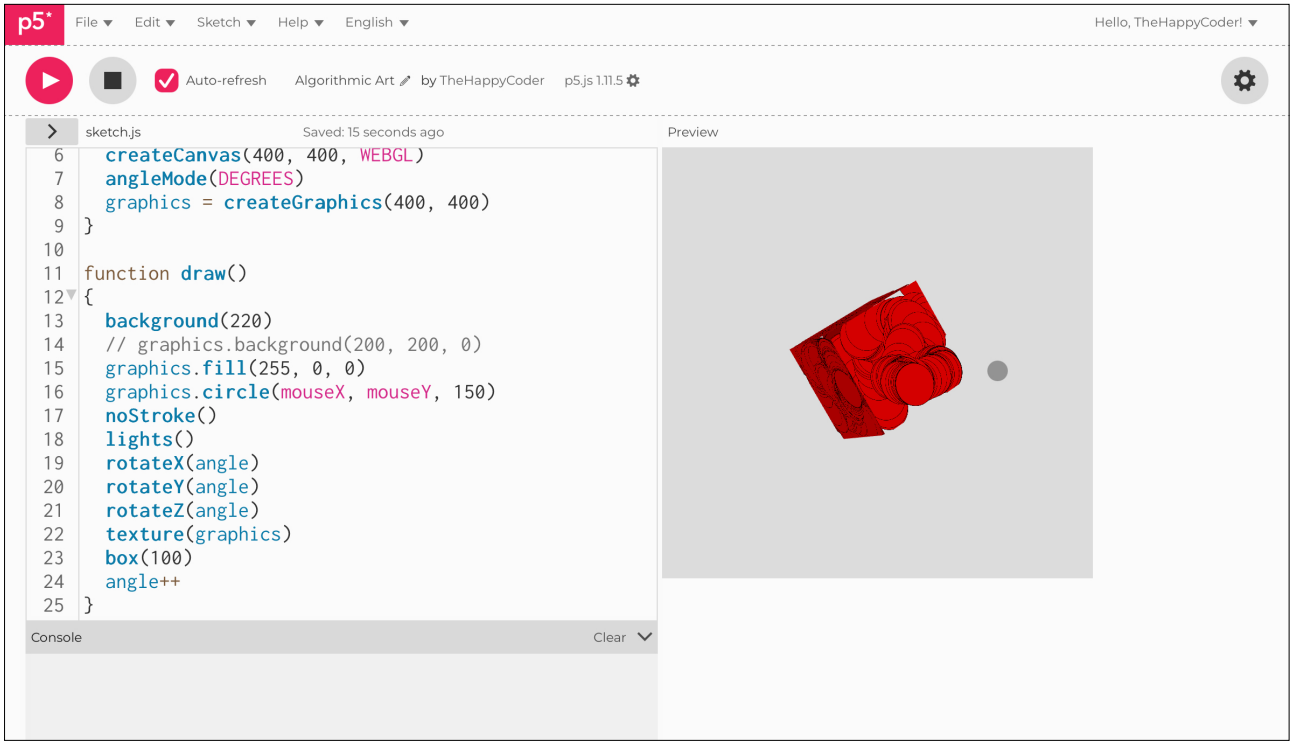
Notes

We have an invisible cube to paint on. The `noStroke()` only works on the cube, not the circle.

Challenge

How would you use `noStroke()` on the circle?

Figure C5.6


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
6 createCanvas(400, 400, WEBGL)
7 angleMode(DEGREES)
8 graphics = createGraphics(400, 400)
9 }
10
11 function draw()
12 {
13 background(220)
14 // graphics.background(200, 200, 0)
15 graphics.fill(255, 0, 0)
16 graphics.circle(mouseX, mouseY, 150)
17 noStroke()
18 lights()
19 rotateX(angle)
20 rotateY(angle)
21 rotateZ(angle)
22 texture(graphics)
23 box(100)
24 angle++
25 }
Console
Clear



Sketch C5.7 creating text

We can also create text on the cube.

```
let angle = 0
let graphics

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  graphics = createGraphics(400, 400)
  graphics.textAlign(CENTER, CENTER)
}

function draw()
{
  background(220)
  // graphics.background(200, 200, 0)
  graphics.textSize(75)
  graphics.text('HAPPY', 200, 200)
  noStroke()
  lights()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  texture(graphics)
  box(100)
  angle++
}
```

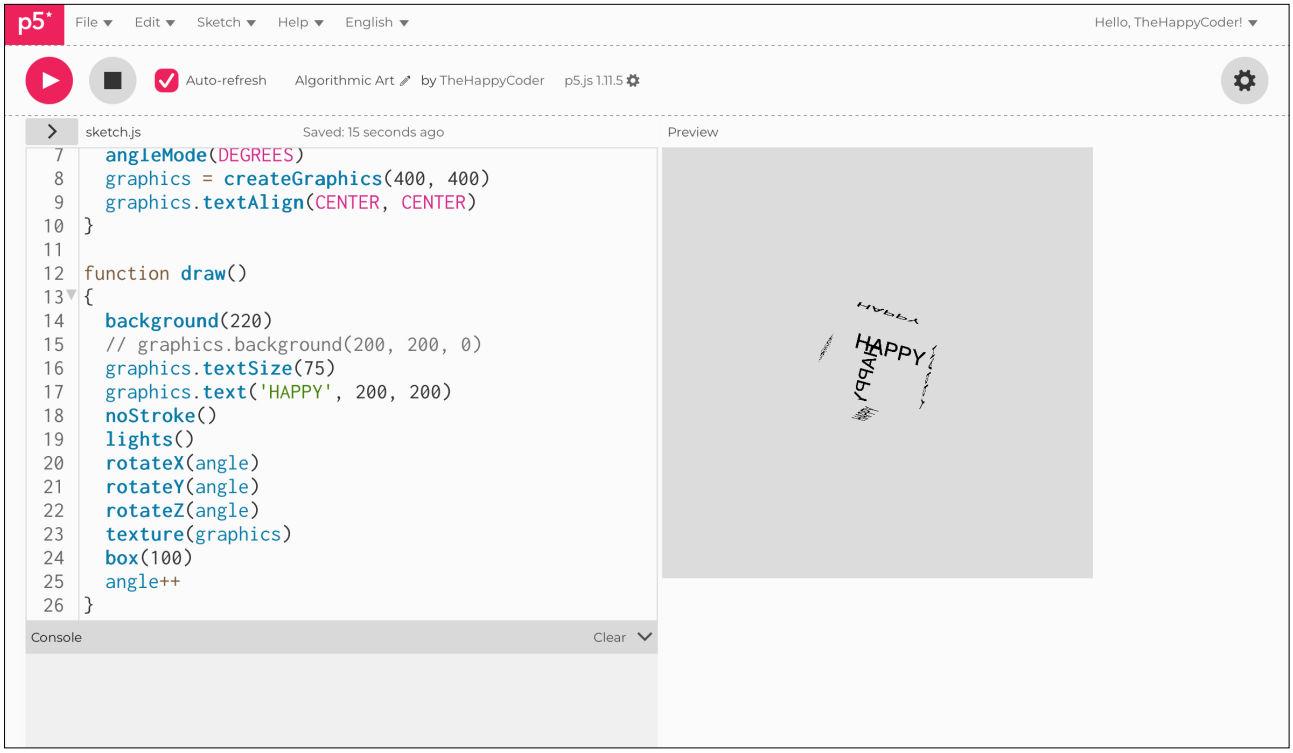
Notes

Notice that we had to prefix the `textAlign()` with the `graphics` reference.

Challenge

Could you rotate the text at the same time?

Figure C5.7


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
7 angleMode(DEGREES)
8 graphics = createGraphics(400, 400)
9 graphics.textAlign(CENTER, CENTER)
10 }
11
12 function draw()
13 {
14 background(220)
15 // graphics.background(200, 200, 0)
16 graphics.textSize(75)
17 graphics.text('HAPPY', 200, 200)
18 noStroke()
19 lights()
20 rotateX(angle)
21 rotateY(angle)
22 rotateZ(angle)
23 texture(graphics)
24 box(100)
25 angle++
26 }
Console
Clear



Sketch C5.8 words on a plane

If we put it on a plane...

```
let angle = 0
let graphics

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  graphics = createGraphics(400, 400)
  graphics.textAlign(CENTER, CENTER)
}

function draw()
{
  background(220)
  // graphics.background(200, 200, 0)
  graphics.textSize(75)
  graphics.text('HAPPY', 200, 200)
  noStroke()
  lights()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  texture(graphics)
  plane(100)
  angle++
}
```

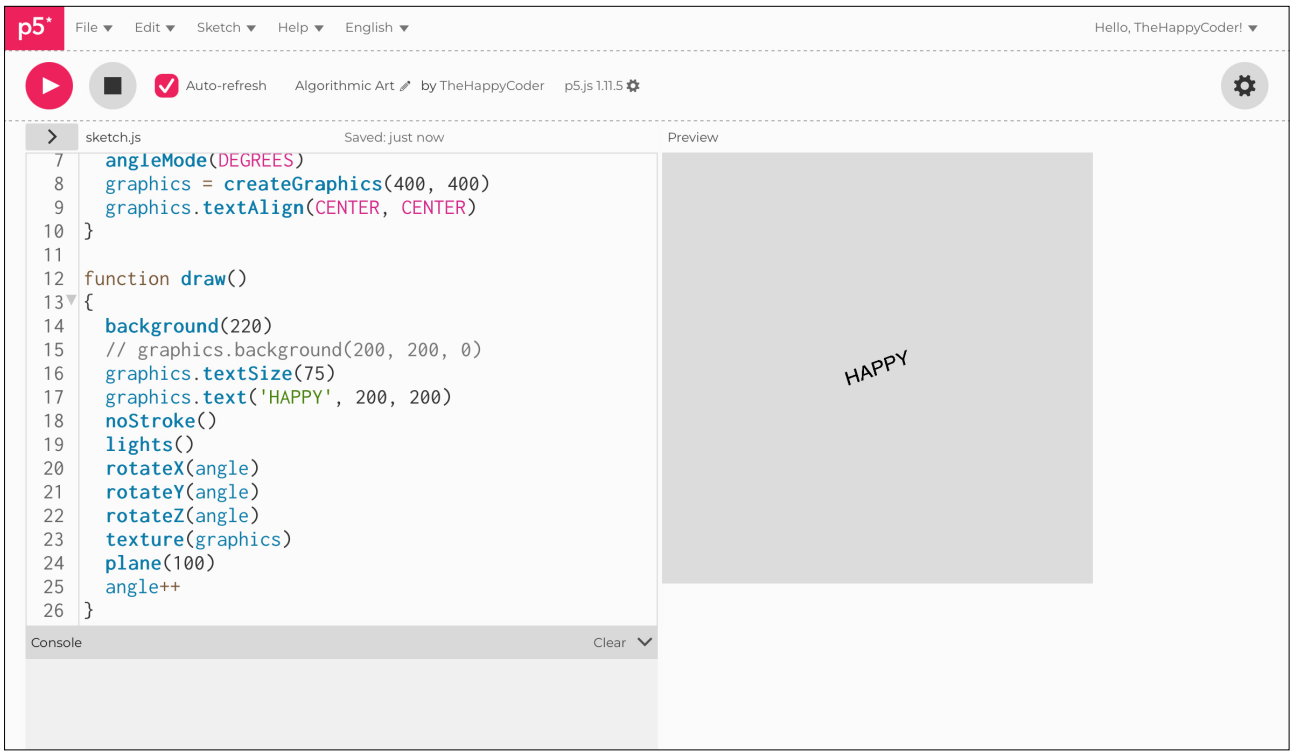
Notes

What is interesting is that the text always ends up the right way (not upside down or reversed) when you use all three axes.

Challenge

See what happens when you only have the x or y axis of rotation.

Figure C5.8


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
angleMode(DEGREES)
graphics = createGraphics(400, 400)
graphics.textAlign(CENTER, CENTER)
}
function draw()
{
background(220)
// graphics.background(200, 200, 0)
graphics.textSize(75)
graphics.text('HAPPY', 200, 200)
noStroke()
lights()
rotateX(angle)
rotateY(angle)
rotateZ(angle)
texture(graphics)
plane(100)
angle++
}
HAPPY
Console
Clear



Sketch C5.9 text on a cylinder

What about round the side of a cylinder? We need to make some adaptations to get this to work reasonably. You can play with everything later.

```
let angle = 0
let graphics

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  graphics = createGraphics(400, 400)
  graphics.textAlign(CENTER, CENTER)
}

function draw()
{
  background(220)
  graphics.background(220)
  graphics.textSize(50)
  graphics.text('HAPPY CODER', 200, 200)
  noStroke()
  // lights()
  // rotateX(angle)
  rotateY(-angle)
  // rotateZ(angle)
  texture(graphics)
  cylinder(50, 200)
  angle++
}
```

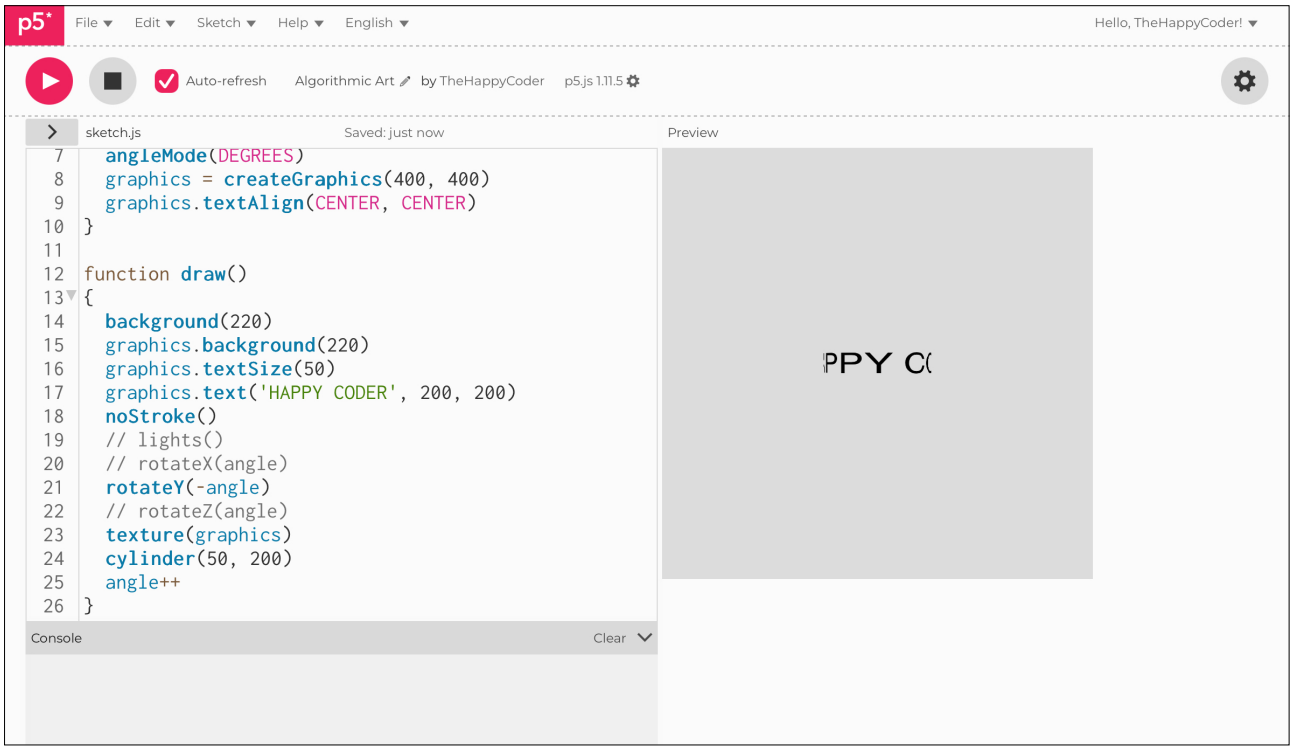
Notes

Quite a bit of refactoring. I won't go into all of it, but I will allow you to experiment with what works or doesn't work.

Challenges

1. Try other shapes.
2. Add some colour.
3. Could you get the text to move up and down?

Figure C5.9





Algorithmic Art
Module C
Unit #6
Cube Wave



Module C Unit #6: the cube wave

This is a coding challenge from the Coding Train YouTube channel. It is a good illustration of what you can do with WebGL. We will add to what we have already learned, including creating a nice little GIF.



Sketch C6.1 starting sketch

This is our starting sketch. We are not WebGL just yet.

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

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

Notes

Just the usual basic sketch



Sketch C6.2 rectangle

We are adding an `angle` variable and incrementing it by `0.1`. We have a rectangle translated to the centre with a height variable `h` set to `100`.

```
let angle = 0
let h

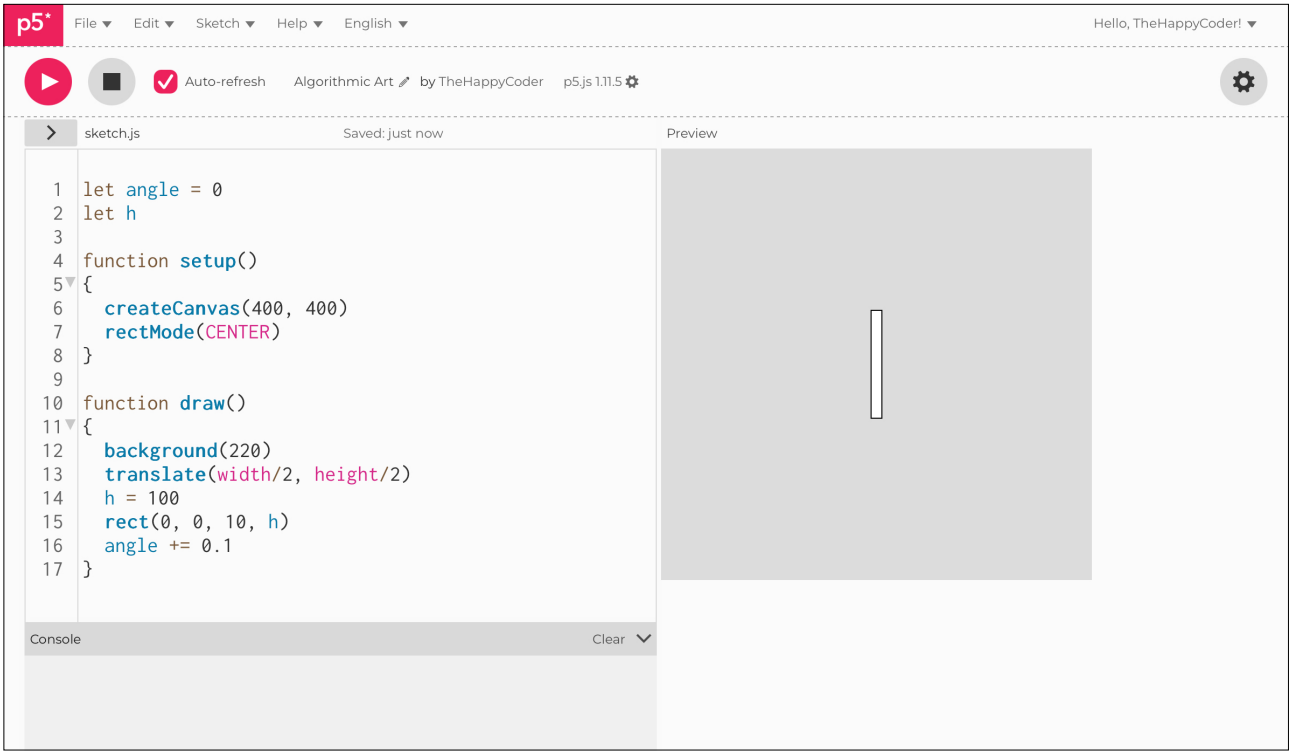
function setup()
{
  createCanvas(400, 400)
  rectMode(CENTER)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  h = 100
  rect(0, 0, 10, h)
  angle += 0.1
}
```

Notes

Nothing happens; this is just the build-up.

Figure C6.2


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
let angle = 0
let h
function setup()
{
createCanvas(400, 400)
rectMode(CENTER)
}
function draw()
{
background(220)
translate(width/2, height/2)
h = 100
rect(0, 0, 10, h)
angle += 0.1
}
Console
Clear

Sketch C6.3 breathing

We are going to add a sine wave motion to the vertical dimensions of the rectangle. The output from a sine wave is +1 to −1. To increase the visibility of the movement, we will `map()` it to `100`.

```
let angle = 0
let h

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

function draw()
{
  background(220)
  translate(width/2, height/2)
  h = map(sin(angle), -1, 1, 0, 100)
  rect(0, 0, 10, h)
  angle += 0.1
}
```

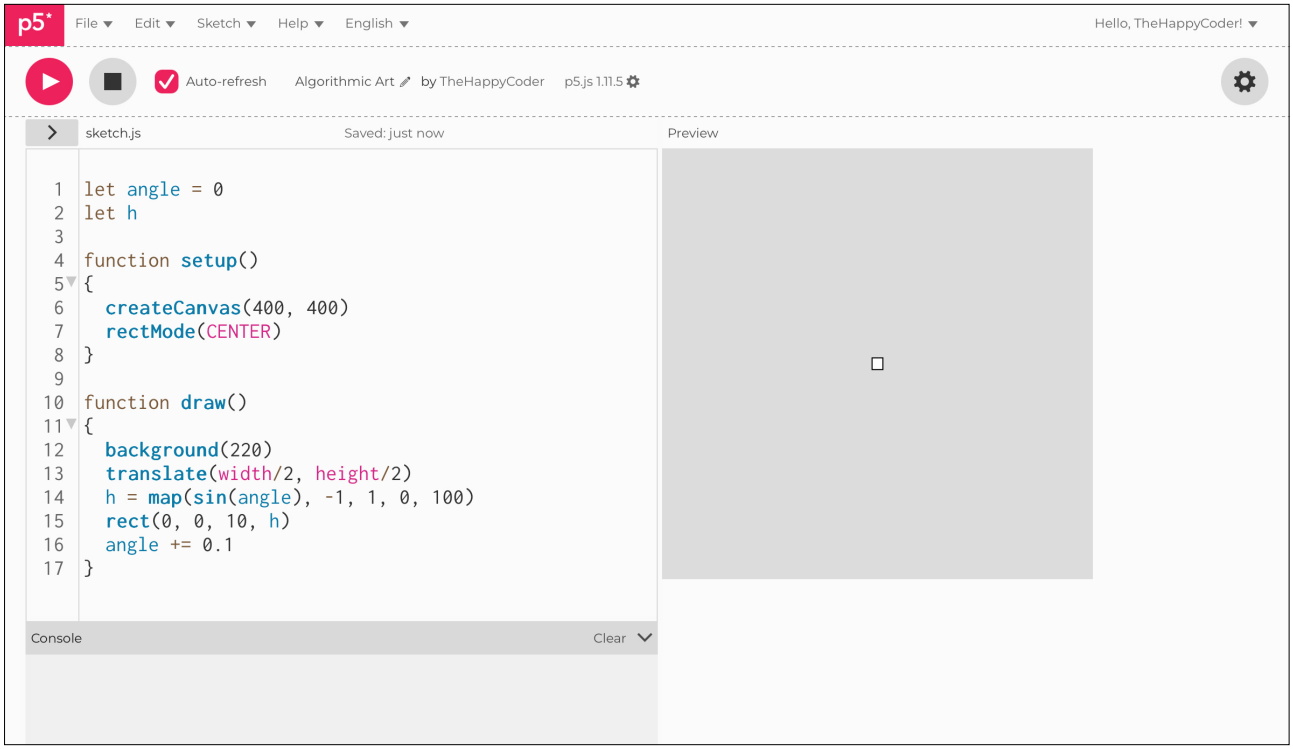
Notes

What you should see is the rectangle 'breathing' as it expands and contracts vertically. Called Simple Harmonic Motion (SHM).

Code Explanation

`h = map(sin(angle), -1, 1, 0, 100)`	The value of h is mapped from the sin() output (−1, 1) to (0, 100).

Figure C6.3


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
let angle = 0
let h
function setup()
{
createCanvas(400, 400)
rectMode(CENTER)
}
function draw()
{
background(220)
translate(width/2, height/2)
h = map(sin(angle), -1, 1, 0, 100)
rect(0, 0, 10, h)
angle += 0.1
}
Console
Clear



Sketch C6.4 a row of rectangles

Now we will make a row of these moving starting at the left-hand edge, adding an `x` variable for the rectangle and looping through every `10` pixels. We use `x - width/2` because we have translated everything to the centre.

```
let angle = 0
let h

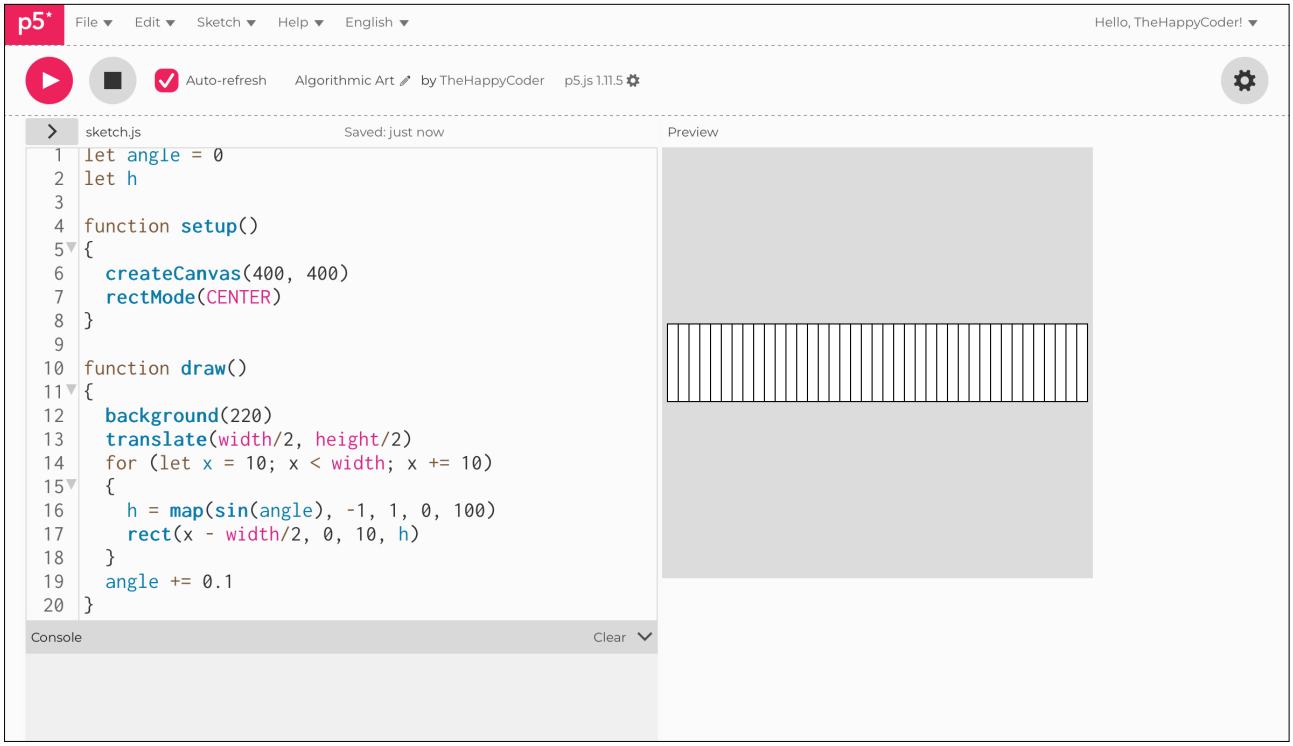
function setup()
{
  createCanvas(400, 400)
  rectMode(CENTER)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  for (let x = 10; x < width; x += 10)
  {
    h = map(sin(angle), -1, 1, 0, 100)
    rect(x - width/2, 0, 10, h)
  }
  angle += 0.1
}
```

Notes

We have a uniform sine wave, where they are all moving as one.

Figure C6.4


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
let angle = 0
let h
function setup()
{
createCanvas(400, 400)
rectMode(CENTER)
}
function draw()
{
background(220)
translate(width/2, height/2)
for (let x = 10; x < width; x += 10)
{
h = map(sin(angle), -1, 1, 0, 100)
rect(x - width/2, 0, 10, h)
}
angle += 0.1
}
Console
Clear



Sketch C6.5 wavy pattern

Now we can add an offset so that we get a nice wave pattern rather than all together.

```
let angle = 0
let h
let newAngle
let offset

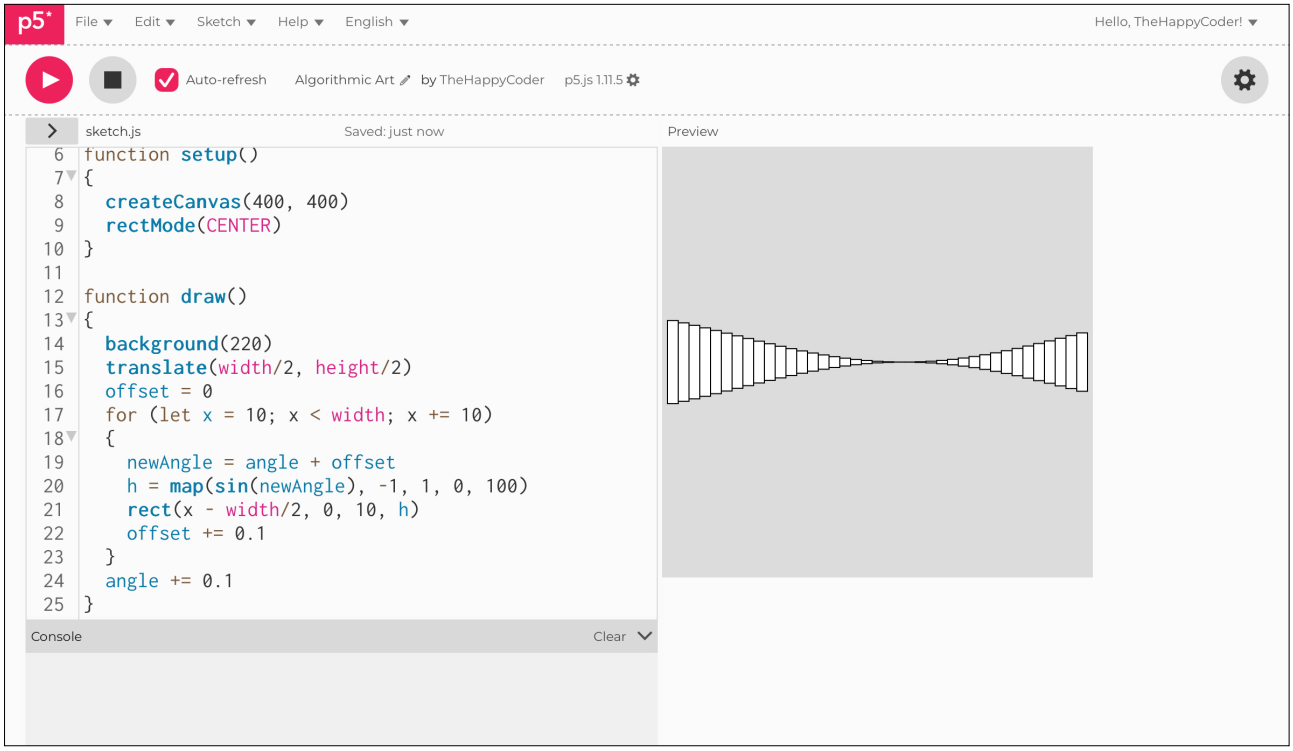
function setup()
{
  createCanvas(400, 400)
  rectMode(CENTER)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  offset = 0
  for (let x = 10; x < width; x += 10)
  {
    newAngle = angle + offset
    h = map(sin(newAngle), -1, 1, 0, 100)
    rect(x - width/2, 0, 10, h)
    offset += 0.1
  }
  angle += 0.1
}
```

Notes

For this, we needed a new variable for the angle, newAngle, as the original is being added to all the time.

Figure C6.5


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400)
rectMode(CENTER)
}
function draw()
{
background(220)
translate(width/2, height/2)
offset = 0
for (let x = 10; x < width; x += 10)
{
newAngle = angle + offset
h = map(sin(newAngle), -1, 1, 0, 100)
rect(x - width/2, 0, 10, h)
offset += 0.1
}
angle += 0.1
}
Console
Clear



Sketch C6.6 incremental offset

Changing the `offset` increment to `0.25`.

```
let angle = 0
let h
let newAngle
let offset

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

function draw()
{
  background(220)
  translate(width/2, height/2)
  offset = 0
  for (let x = 10; x < width; x += 10)
  {
    newAngle = angle + offset
    h = map(sin(newAngle), -1, 1, 0, 100)
    rect(x - width/2, 0, 10, h)
    offset += 0.25
  }
  angle += 0.1
}
```

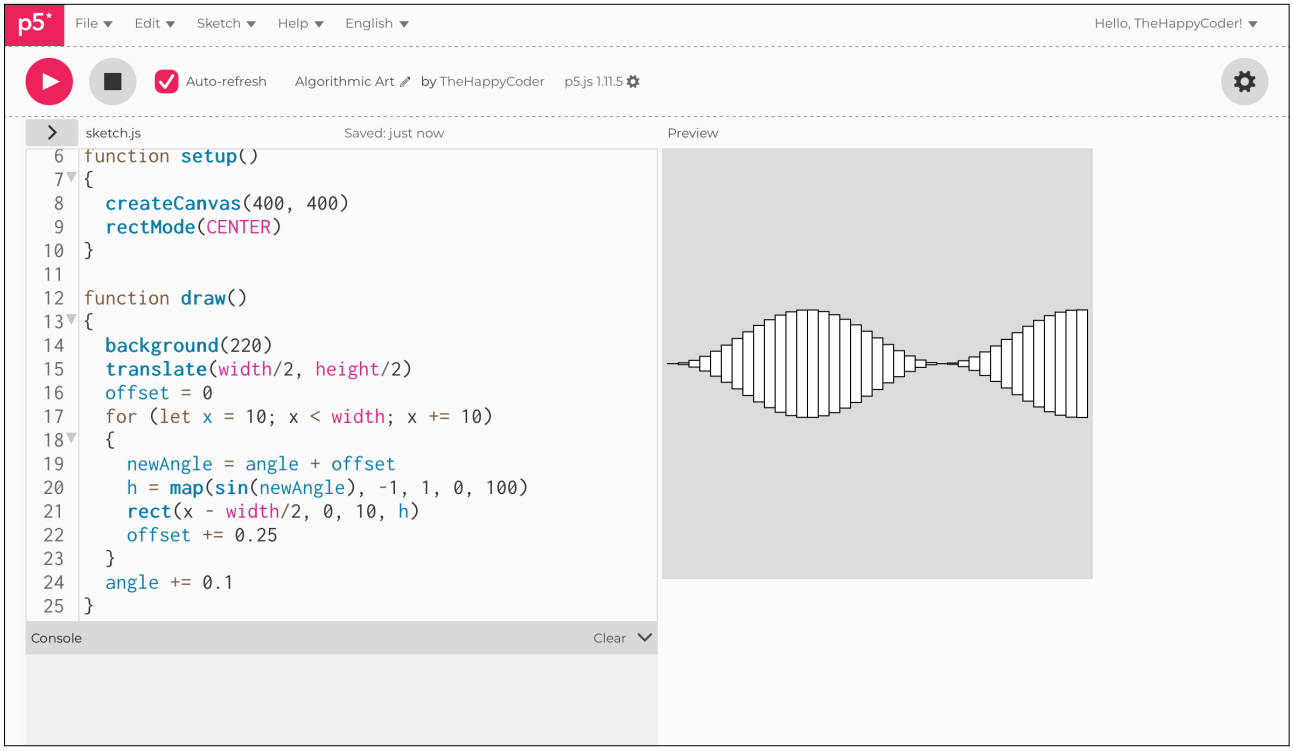
Notes

An even nicer sine wave.

Challenge

Try different offsets.

Figure C6.6


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)
rectMode(CENTER)
}
function draw()
{
background(220)
translate(width/2, height/2)
offset = 0
for (let x = 10; x < width; x += 10)
{
newAngle = angle + offset
h = map(sin(newAngle), -1, 1, 0, 100)
rect(x - width/2, 0, 10, h)
offset += 0.25
}
angle += 0.1
}
Console
Clear

Sketch C6.7 tidy up

We are going to add a few more features as we prepare to jump from 2D to 3D. We create a variable for the width called w and set x to start at 0, not 10, in the for() loop. This just tidies things up a bit.

```
let angle = 0
let h
let newAngle
let offset
let w = 20

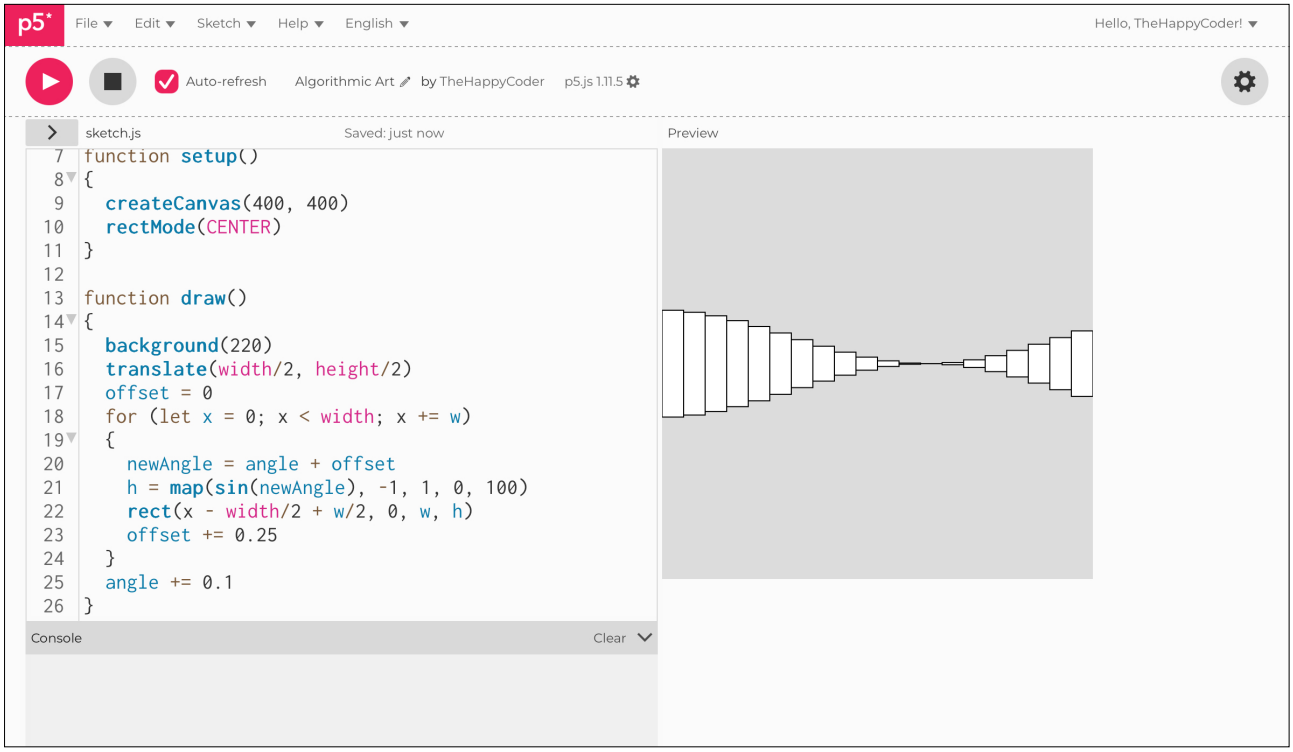
function setup()
{
  createCanvas(400, 400)
  rectMode(CENTER)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  offset = 0
  for (let x = 0; x < width; x += w)
  {
    newAngle = angle + offset
    h = map(sin(newAngle), -1, 1, 0, 100)
    rect(x - width/2 + w/2, 0, w, h)
    offset += 0.25
  }
  angle += 0.1
}
```

Notes

Similar result.

Figure C6.7


p5*
File
Edit
Sketch
Help
English
Hello, TheHappyCoder!
Auto-refresh
Algorithmic Art
by TheHappyCoder
p5.js 1.11.5
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400)
rectMode(CENTER)
}
function draw()
{
background(220)
translate(width/2, height/2)
offset = 0
for (let x = 0; x < width; x += w)
{
newAngle = angle + offset
h = map(sin(newAngle), -1, 1, 0, 100)
rect(x - width/2 + w/2, 0, w, h)
offset += 0.25
}
angle += 0.1
}
Console
Clear

Sketch C6.8 adding WEBGL

So far everything has been 2D; now we add `WEBGL` and remove the `translate()` (because it automatically puts it in the centre). `WEBGL` is just the render, so we can still draw something in 2D.

```
let angle = 0
let h
let newAngle
let offset
let w = 20

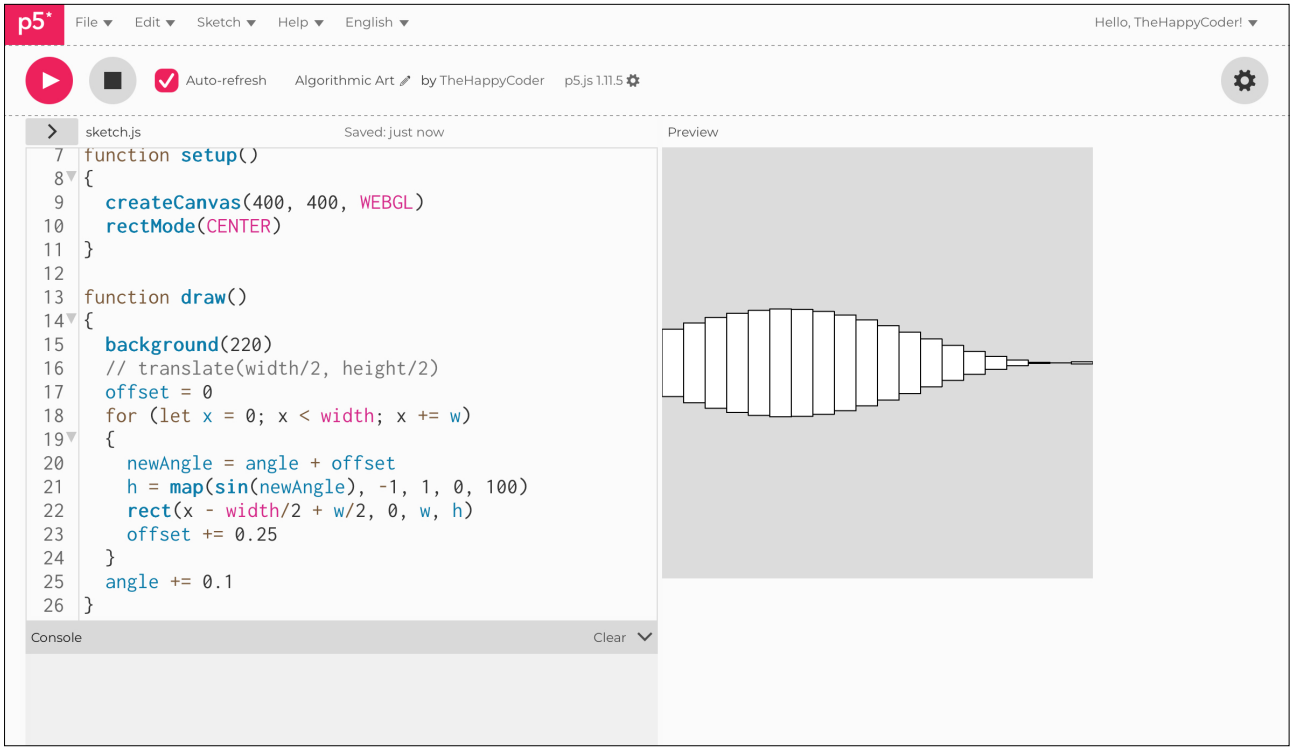
function setup()
{
  createCanvas(400, 400, WEBGL)
  rectMode(CENTER)
}

function draw()
{
  background(220)
  // translate(width/2, height/2)
  offset = 0
  for (let x = 0; x < width; x += w)
  {
    newAngle = angle + offset
    h = map(sin(newAngle), -1, 1, 0, 100)
    rect(x - width/2 + w/2, 0, w, h)
    offset += 0.25
  }
  angle += 0.1
}
```

Notes

Pretty much the same result again.

Figure C6.8


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
function setup()
{
createCanvas(400, 400, WEBGL)
rectMode(CENTER)
}
function draw()
{
background(220)
// translate(width/2, height/2)
offset = 0
for (let x = 0; x < width; x += w)
{
newAngle = angle + offset
h = map(sin(newAngle), -1, 1, 0, 100)
rect(x - width/2 + w/2, 0, w, h)
offset += 0.25
}
angle += 0.1
}
Console
Clear

Sketch C6.9 adding a box

We replace the `rect()` with `box(w)` and have each box drawn at a new `x` position as part of the loop. To make sure we aren't compounding the translate, we use `push()` and `pop()` to reset it each iteration of the `for()` loop.

```
let angle = 0
let h
let newAngle
let offset
let w = 20

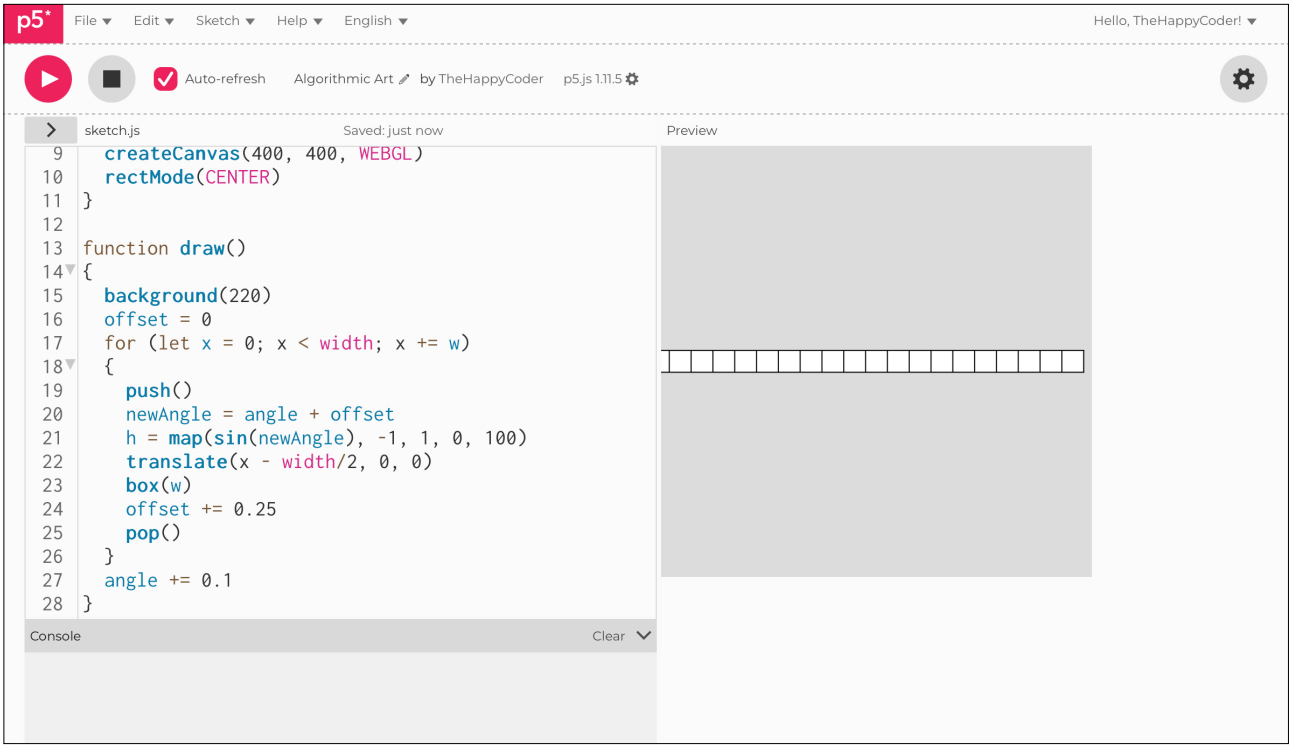
function setup()
{
  createCanvas(400, 400, WEBGL)
  rectMode(CENTER)
}

function draw()
{
  background(220)
  offset = 0
  for (let x = 0; x < width; x += w)
  {
    push()
    newAngle = angle + offset
    h = map(sin(newAngle), -1, 1, 0, 100)
    translate(x - width/2, 0, 0)
    box(w)
    offset += 0.25
    pop()
  }
  angle += 0.1
}
```

Notes

It looks a little off-centre, but we will live with that for now.

Figure C6.9


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
createCanvas(400, 400, WEBGL)
rectMode(CENTER)
}
function draw()
{
background(220)
offset = 0
for (let x = 0; x < width; x += w)
{
push()
newAngle = angle + offset
h = map(sin(newAngle), -1, 1, 0, 100)
translate(x - width/2, 0, 0)
box(w)
offset += 0.25
pop()
}
angle += 0.1
}
Console
Clear

Sketch C6.10 rotate

To prove they are cubes, we will rotate by −1, but we want the sine wave to affect the height, so we will put that back in and also have the z dimension as w.

```
let angle = 0
let h
let newAngle
let offset
let w = 20

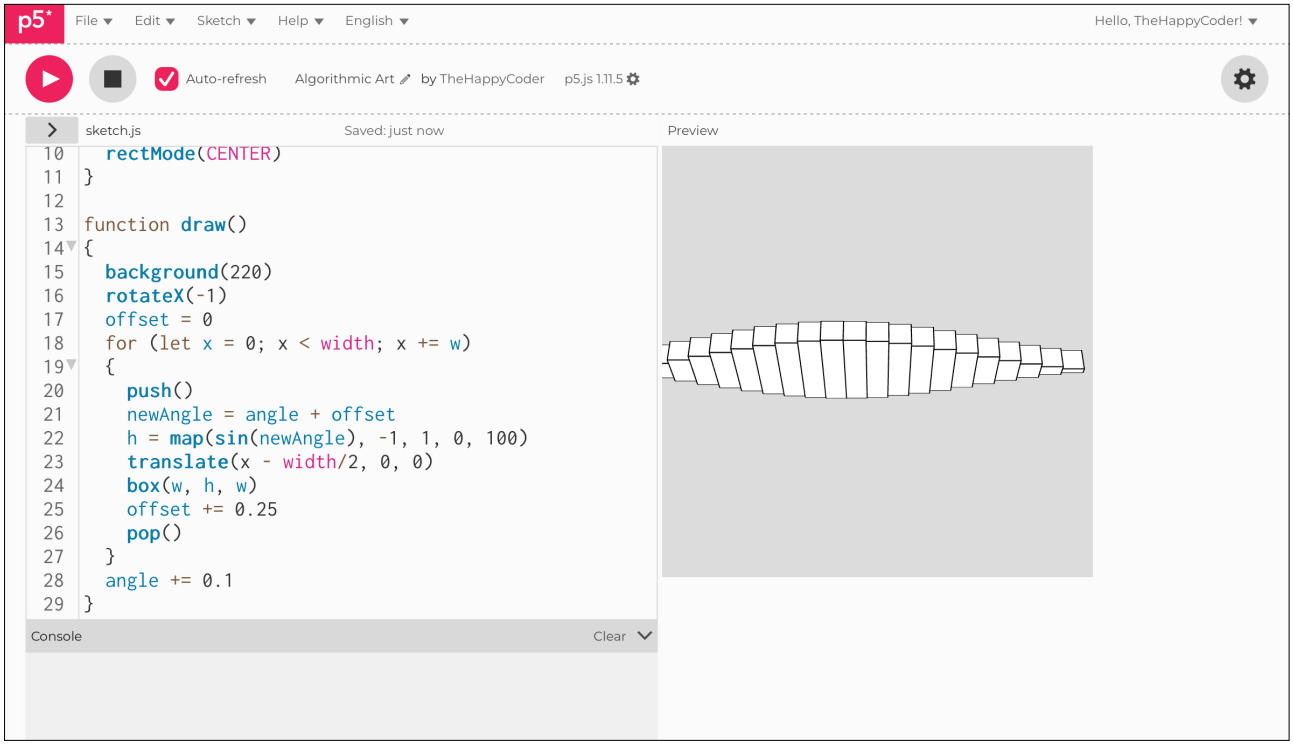
function setup()
{
  createCanvas(400, 400, WEBGL)
  rectMode(CENTER)
}

function draw()
{
  background(220)
  rotateX(-1)
  offset = 0
  for (let x = 0; x < width; x += w)
  {
    push()
    newAngle = angle + offset
    h = map(sin(newAngle), -1, 1, 0, 100)
    translate(x - width/2, 0, 0)
    box(w, h, w)
    offset += 0.25
    pop()
  }
  angle += 0.1
}
```

Notes

You can see the motion of the cubes.

Figure C6.10


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
rectMode(CENTER)
}
function draw()
{
background(220)
rotateX(-1)
offset = 0
for (let x = 0; x < width; x += w)
{
push()
newAngle = angle + offset
h = map(sin(newAngle), -1, 1, 0, 100)
translate(x - width/2, 0, 0)
box(w, h, w)
offset += 0.25
pop()
}
angle += 0.1
}
Console
Clear



Sketch C6.11 orthographic

We are going to introduce a different perspective called orthographic. The default is perspective projection, which looks more realistic, but we will introduce orthographic projection. It makes the boxes look a bit flatter.

```
let angle = 0
let h
let newAngle
let offset
let w = 20

function setup()
{
  createCanvas(400, 400, WEBGL)
  rectMode(CENTER)
}

function draw()
{
  background(220)
  ortho()
  rotateX(-1)
  offset = 0
  for (let x = 0; x < width; x += w)
  {
    push()
    newAngle = angle + offset
    h = map(sin(newAngle), -1, 1, 0, 100)
    translate(x - width/2, 0, 0)
    box(w, h, w)
    offset += 0.1
    pop()
  }
  angle += 0.1
}
```

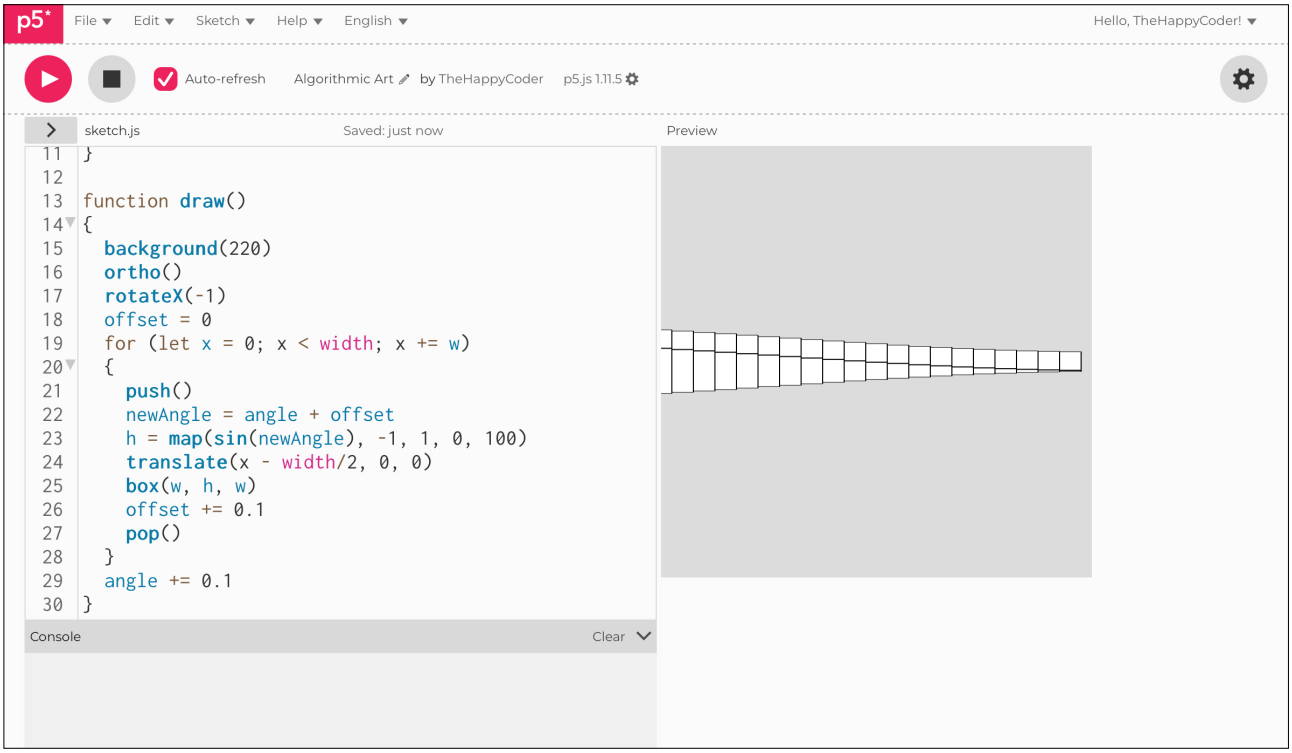
Notes

The difference between the projections is something I will leave you to research if you don't already know.

Code Explanation

`ortho()`	Changes from the default projection to orthographic.

Figure C6.11





Sketch C6.12 the magic number

We are now going to rotate along the x-axis by 45 degrees (π/4) and rotate along the y axis by a magic number (ma), which is arctan of one divided by the square root of two. This is a number found from those experimenting with different projections, so just accept it. Also, we have a new loop around the x for() loop to have a z value and move the offset into that loop.

```
let angle = 0
let h
let newAngle
let offset
let w = 20
let ma

function setup()
{
  createCanvas(400, 400, WEBGL)
  rectMode(CENTER)
  ma = atan(1/sqrt(2))
}

function draw()
{
  background(220)
  ortho()
  rotateX(-PI/4)
  rotateY(ma)
  offset = 0
  for (let z = 0; z < height; z += w)
  {
    for (let x = 0; x < width; x += w)
    {
      push()
      newAngle = angle + offset
      h = map(sin(newAngle), -1, 1, 0, 100)
      translate(x - width/2, 0, z - height/2)
      box(w, h, w)
      // offset += 0.25
      pop()
```

```
    }
    offset += 0.25
  }
  angle += 0.1
}
```

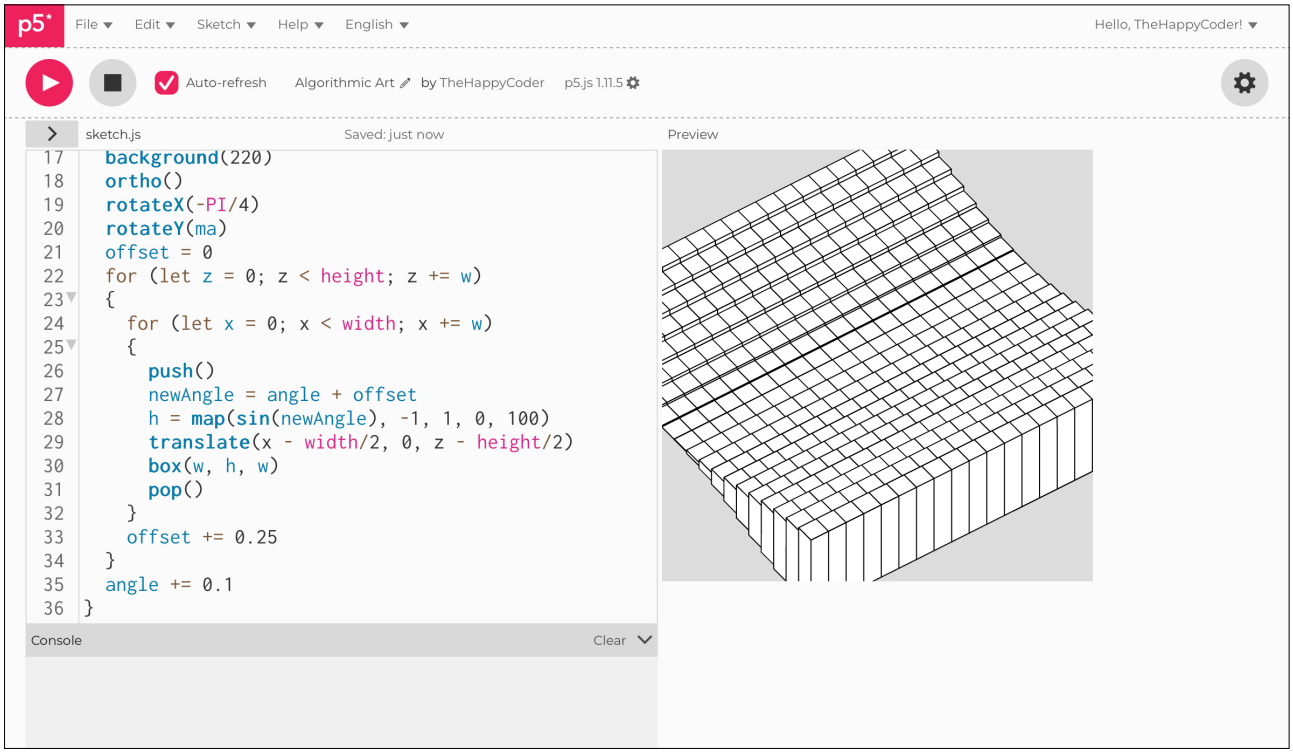
Notes

In p5.js, you can use the constant π by using PI. This is useful if you are using radians and don't want to work out the actual number. The atan() function is currently not supported with v2.x.

Code Explanation

`ma = atan(1/sqrt(2))`	The magic number.
`rotateX(-PI/4)`	Rotating x by π/4 or 45°.

Figure C6.12



Sketch C6.13 ortho() parameters

We can add more parameters to `ortho()` to allow what you can see. There is a lot involved in these parameters to explain, but it is to do with the depth and width of the viewing angle, etc. I've chosen these values as they put the boxes in a nice view. I suggest twiddling the numbers to get what you want.

```
let angle = 0
let h
let newAngle
let offset
let w = 20
let ma

function setup()
{
  createCanvas(400, 400, WEBGL)
  rectMode(CENTER)
  ma = atan(1/sqrt(2))
}

function draw()
{
  background(220)
  ortho(-320, 300, -300, 300)
  rotateX(-PI/4)
  rotateY(ma)
  offset = 0
  for (let z = 0; z < height; z += w)
  {
    for (let x = 0; x < width; x += w)
    {
      push()
      newAngle = angle + offset
      h = map(sin(newAngle), -1, 1, 0, 100)
      translate(x - width/2, 0, z - height/2)
      box(w, h, w)

      pop()
```

```
    }
    offset += 0.25
  }
  angle += 0.1
}
```

Notes

You can have up to six arguments for the `ortho()` projection; the fifth and sixth are near and far. This is all to do with a camera's view of the object shape.

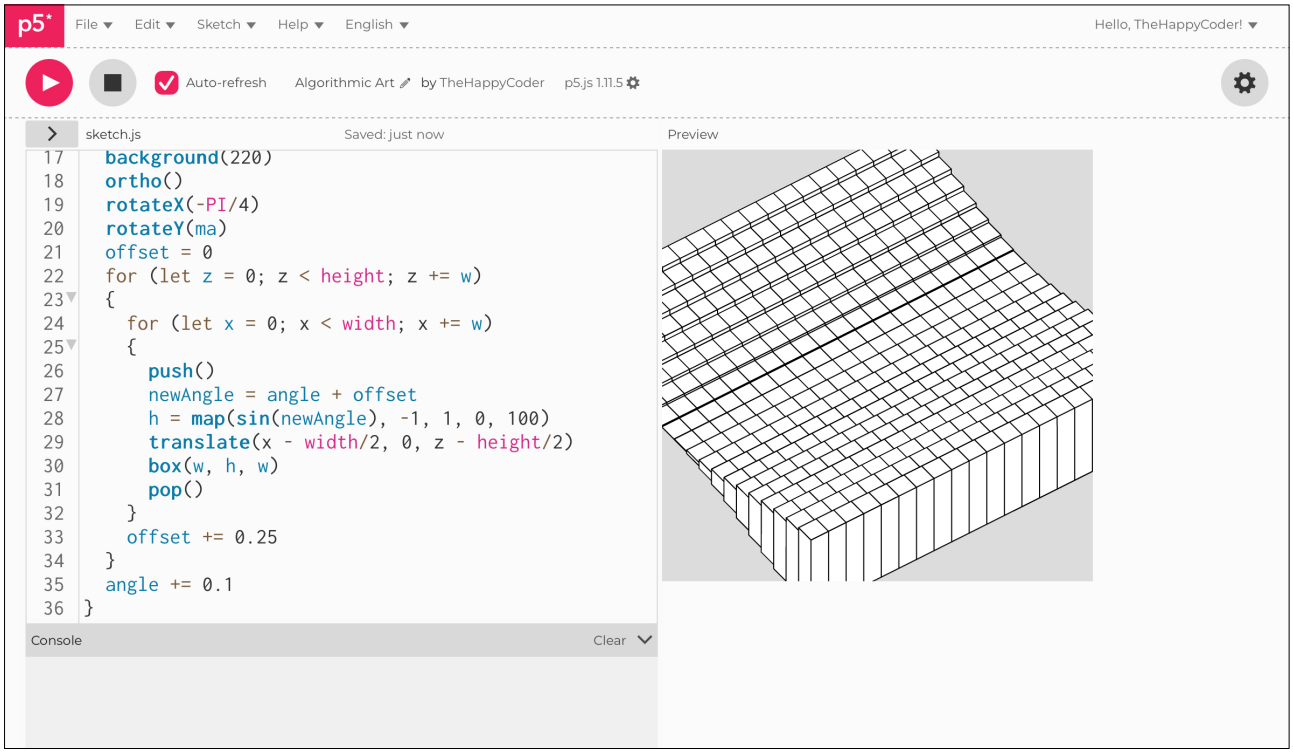
Challenge

Have a play with the values to see what difference they make.

Code Explanation

`ortho(-320, 300, -300, 300)`	They indicate frustum values for left, right, bottom, and top.

Figure C6.13



Sketch C6.14 ripples

At the moment, we have the whole thing moving as a wave, but what we want is a ripple-type effect across the surface. Where the individual boxes move. So we need to calculate the distance between the `x`, `z`, and the `centre`.

We create a `maxDistance` in the `x` and `z` directions, then move the `offset` into the loop so we get the ripple effect across the loop. Also, a few tweaks and adding `normalMaterial`.

```
let angle = 0
let h
let newAngle
let offset
let w = 20
let ma
let maxDistance
let d

function setup()
{
  createCanvas(400, 400, WEBGL)
  rectMode(CENTER)
  ma = atan(1/sqrt(2))
  maxDistance = dist(0, 0, 200, 200)
}

function draw()
{
  background(220)
  normalMaterial()
  ortho(-320, 300, -300, 300)
  rotateX(-PI/4)
  rotateY(ma)
  offset = 0
  for (let z = 0; z < height; z += w)
  {
    for (let x = 0; x < width; x += w)
    {
      push()
```

```
      d = dist(x, z, width/2, height/2)
      offset = map(d, 0, maxDistance, -3, 3)
      newAngle = angle + offset
      h = map(sin(newAngle), -1, 1, 50, 250)
      translate(x - width/2, 0, z - height/2)
      box(w, h, w)
      pop()
    }
    // offset += 0.25
  }
  angle += 0.1
}
```

Notes

Pretty neat, eh!

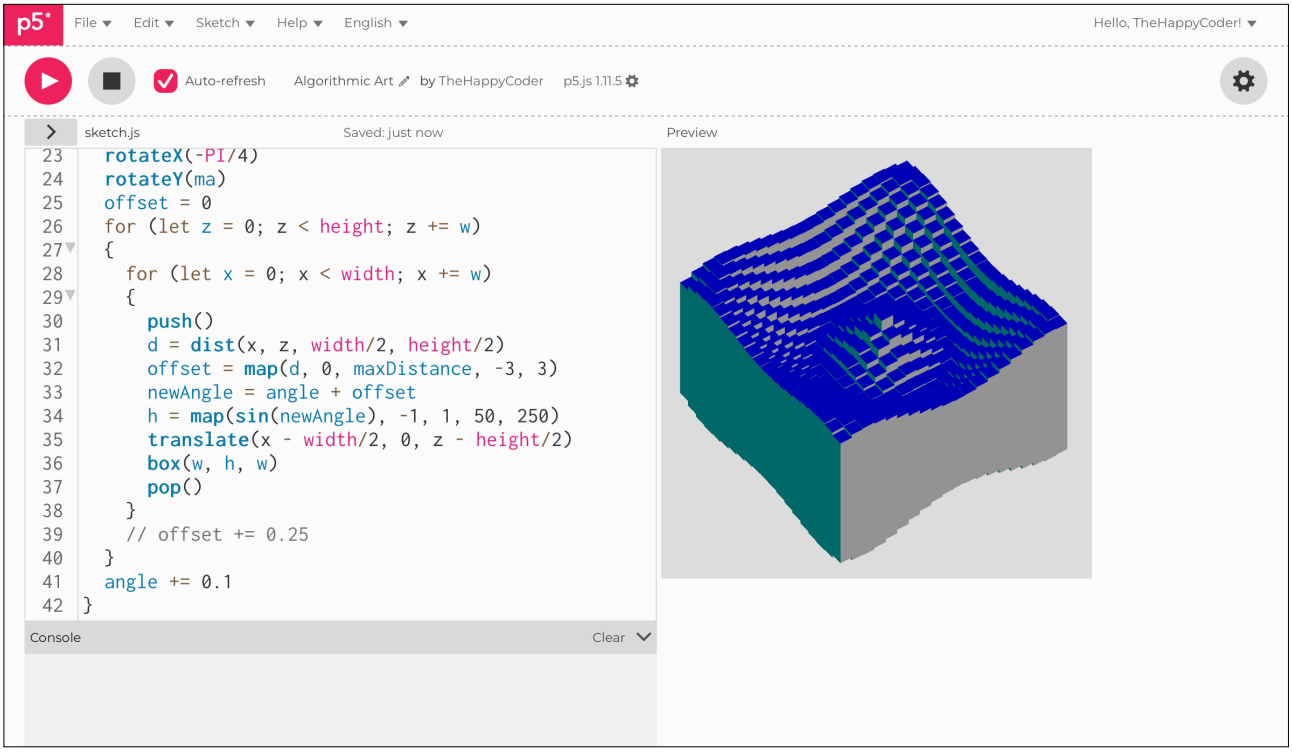
Challenge

Try other materials and lights.

Code Explanation

`d = dist(x, z, width/2, height/2)`	Measures the distance between x, z and the centre of the space.

Figure C6.14


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
23 rotateX(-PI/4)
24 rotateY(ma)
25 offset = 0
26 for (let z = 0; z < height; z += w)
27 {
28 for (let x = 0; x < width; x += w)
29 {
30 push()
31 d = dist(x, z, width/2, height/2)
32 offset = map(d, 0, maxDistance, -3, 3)
33 newAngle = angle + offset
34 h = map(sin(newAngle), -1, 1, 50, 250)
35 translate(x - width/2, 0, z - height/2)
36 box(w, h, w)
37 pop()
38 }
39 // offset += 0.25
40 }
41 angle += 0.1
42 }
Console
Clear

Sketch C6.15 a gif

Now to create a GIF. A GIF is a short video that repeats endlessly. We can do this using our knowledge of the number of frames it takes to do a complete cycle through the sine wave so it will look as if it is seamlessly continuous.

We will use 60 frames and when you press the space bar (key), which is what the (" ") means, it will generate a .gif file according to the name you have given it.

I have also swapped the rotateX and rotateY around to give a different (better) effect.

! Please note that: If you are doing this on a tablet (iPad, etc.), I found that there wasn't enough memory for 60 frames, but less than 60 frames doesn't really work. Also, you might want to slow it down a bit for the .gif, maybe 0.05 rather than 0.1.

```
let angle = 0
let h
let newAngle
let offset
let w = 20
let ma
let maxDistance
let d
let frames = 60

function setup()
{
  createCanvas(400, 400, WEBGL)
  rectMode(CENTER)
  ma = atan(1/sqrt(2))
  maxDistance = dist(0, 0, 200, 200)
}

function keyPressed()
{
  if (key == " ")
  {
    const options = {
      units: "frames",
      delay: 0
```

```
    }
    saveGif("cubewave.gif", frames, options)
  }
}

function draw()
{
  background(220)
  normalMaterial()
  ortho(-320, 300, -300, 300)
  rotateY(ma)
  rotateX(-PI/4)
  offset = 0
  for (let z = 0; z < height; z += w)
  {
    for (let x = 0; x < width; x += w)
    {
      push()
      d = dist(x, z, width/2, height/2)
      offset = map(d, 0, maxDistance, -3, 3)
      newAngle = angle + offset
      h = map(sin(newAngle), -1, 1, 50, 250)
      translate(x - width/2, 0, z - height/2)
      box(w, h, w)
      pop()
    }
  }
  angle += 0.1
}
```

Notes

Play with the values regarding the GIF and see if you can get a better response. Remember that to save the GIF, you need to press the space bar.

Challenge

1. Use lights, colours and other materials.
2. Use other shapes, e.g. cylinders.
3. Expand the ripple effect.
4. Rotate the whole thing.

Code Explanation

`saveGif("cubewave.gif", frames, options)`	We give the name of the file to save, the type of file, how many frames, and any options stated.