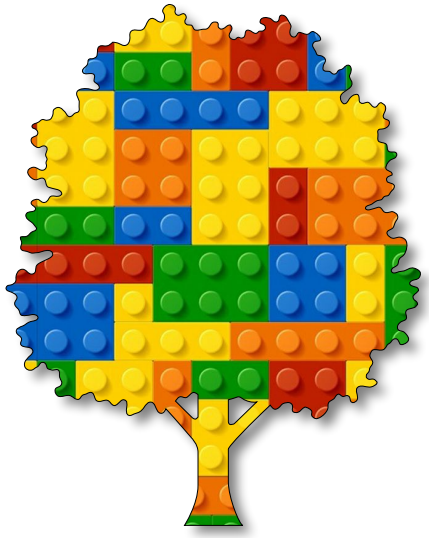
Algorithmic Art

Module C

Unit #6

cube wave



Content

Module C Unit #6 Cube Wave

Introduction to 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)
}
```

Sketch C6.2 rectangle

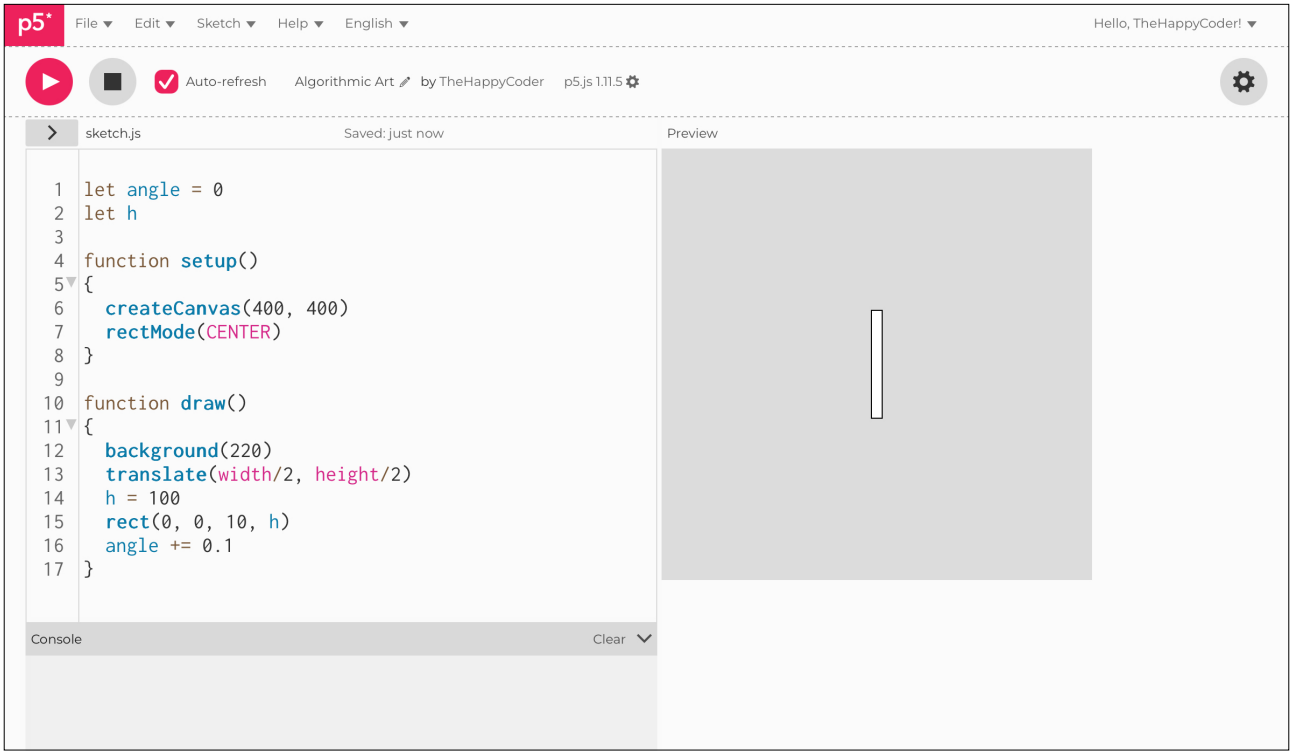
We are adding an `angle` variable and incrementing it by `0.1`. We have a rectangle translated to the centre with a height variable `h` set to `100`.

```
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
}
```

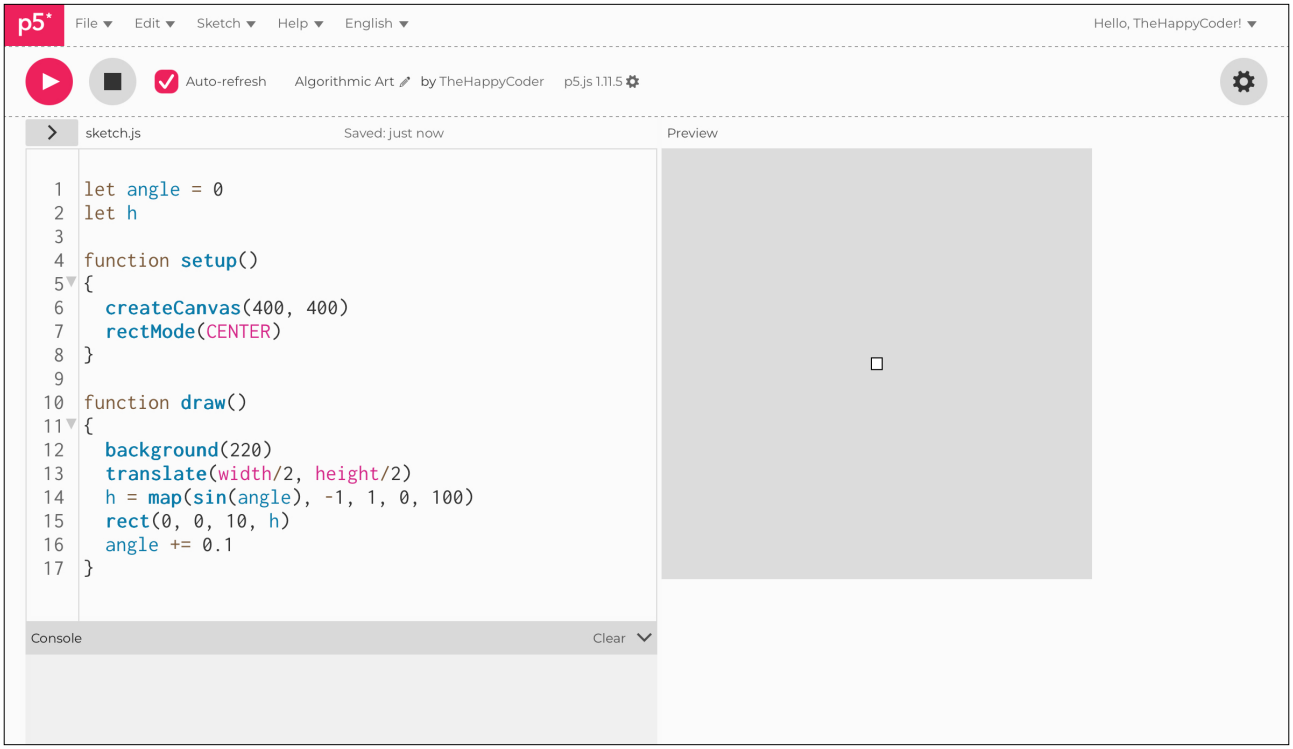
🗒️ Notes

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

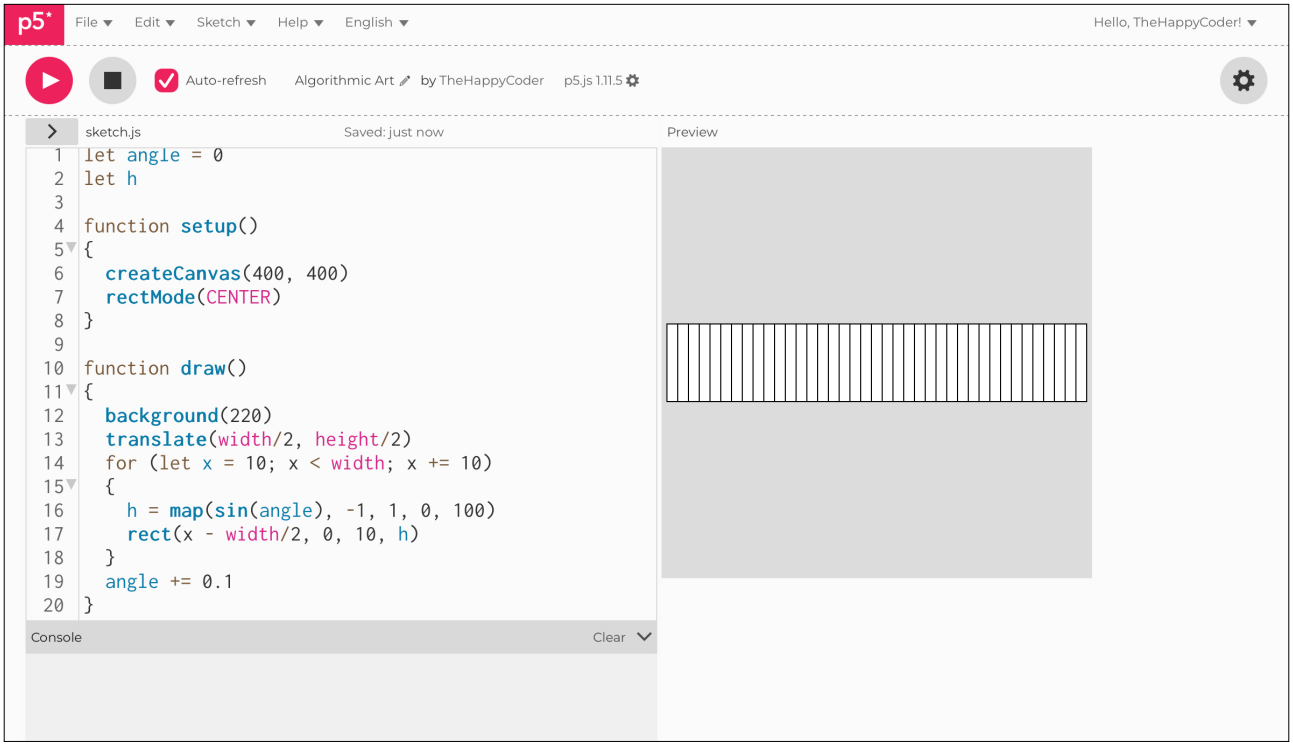
Now we will make a row of these moving starting at the left-hand edge, adding an `x` variable for the rectangle and looping through every `10` pixels. We use `x - width/2` because we have translated everything to the centre.

```
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

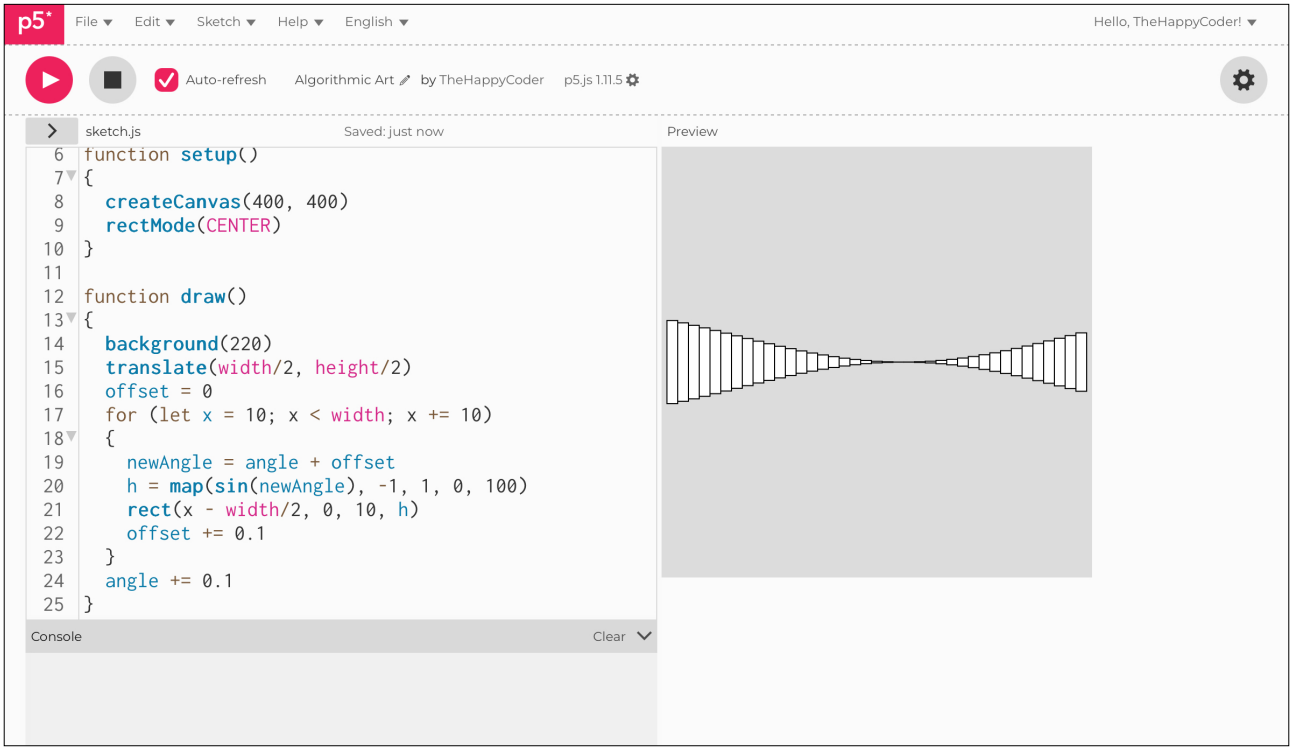
Now we can add an `offset` so that we get a nice wave pattern rather than all together.

```
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
}
```

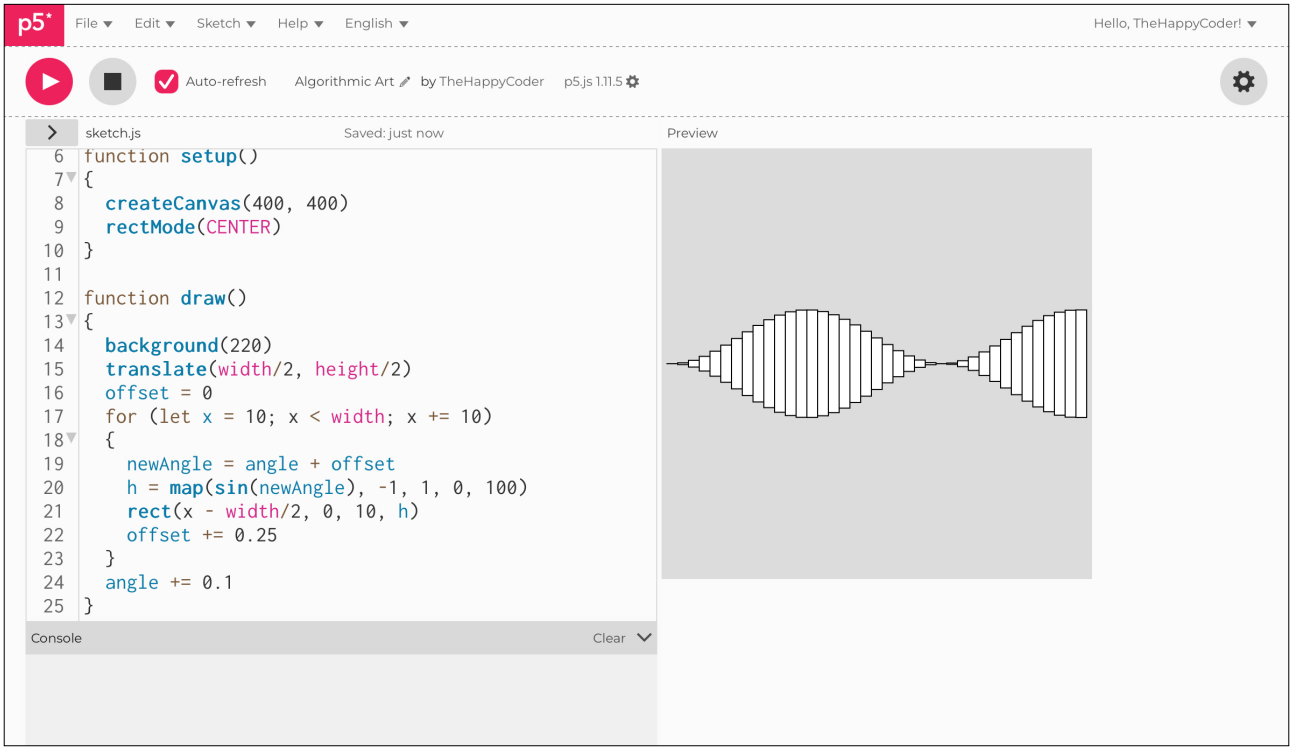
🗒️ Notes

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
6 function setup()
7 {
8 createCanvas(400, 400)
9 rectMode(CENTER)
10 }
11
12 function draw()
13 {
14 background(220)
15 translate(width/2, height/2)
16 offset = 0
17 for (let x = 10; x < width; x += 10)
18 {
19 newAngle = angle + offset
20 h = map(sin(newAngle), -1, 1, 0, 100)
21 rect(x - width/2, 0, 10, h)
22 offset += 0.25
23 }
24 angle += 0.1
25 }
Console
Clear

Sketch C6.7 tidy up

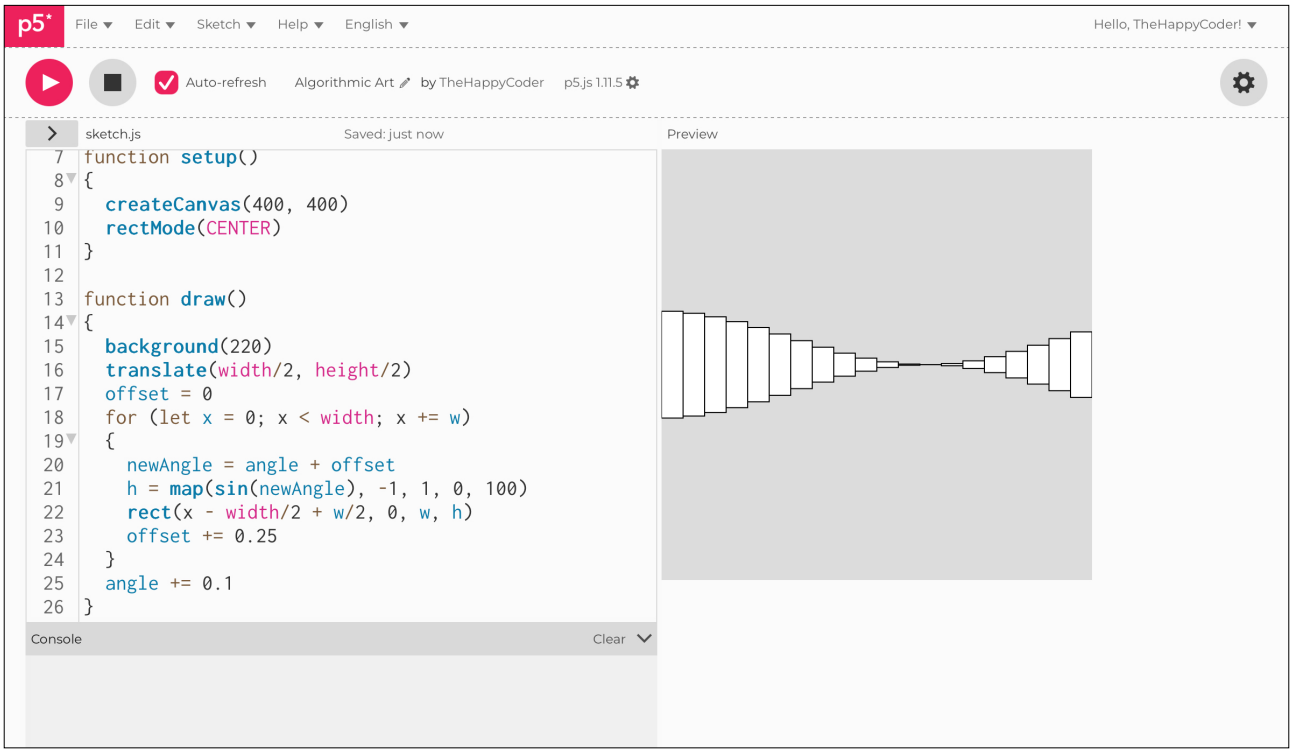
We are going to add a few more features as we prepare to jump from 2D to 3D. We create a variable for the width called w and set x to start at 0, not 10, in the for() loop. This just tidies things up a bit.

```
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

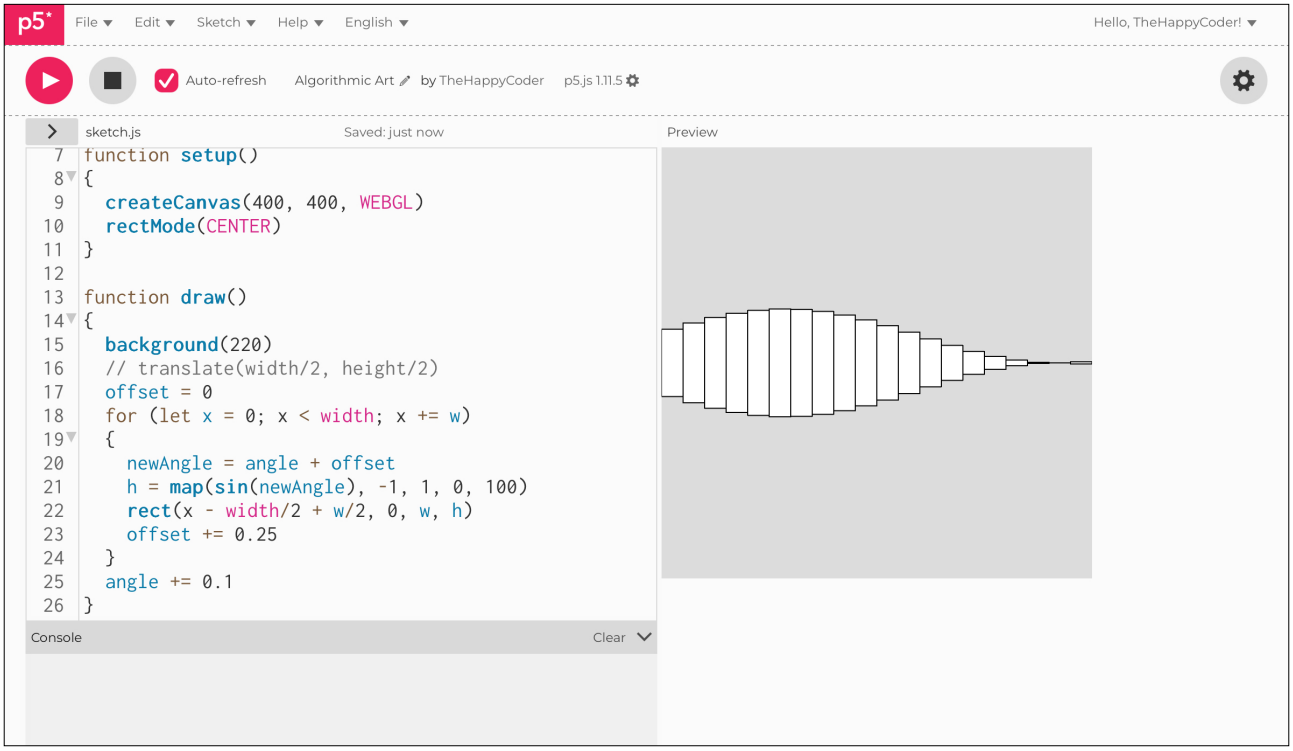
So far everything has been 2D; now we add `WEBGL` and remove the `translate()` (because it automatically puts it in the centre). `WEBGL` is just the render, so we can still draw something in 2D.

```
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

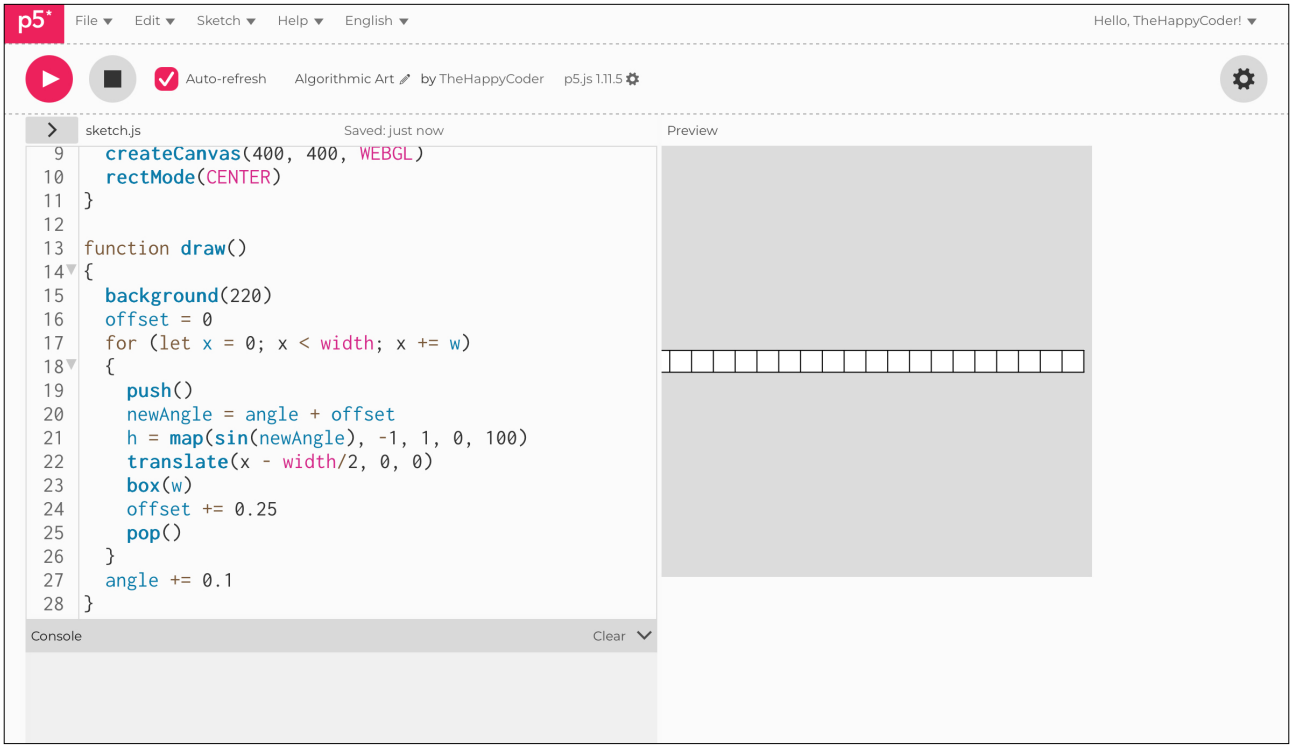
We replace the `rect()` with `box(w)` and have each box drawn at a new `x` position as part of the loop. To make sure we aren't compounding the translate, we use `push()` and `pop()` to reset it each iteration of the `for()` loop.

```
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

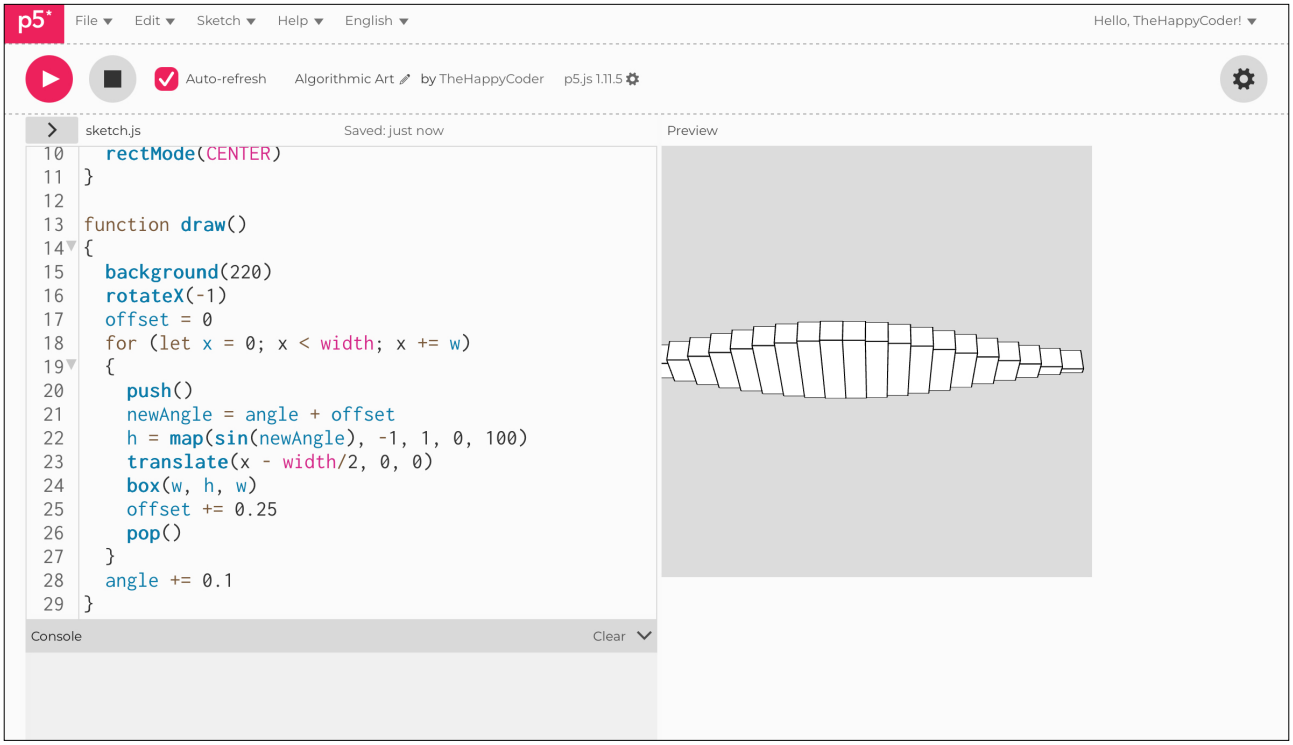
To prove they are cubes, we will rotate by −1, but we want the sine wave to affect the height, so we will put that back in and also have the z dimension as w.

```
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
}
```

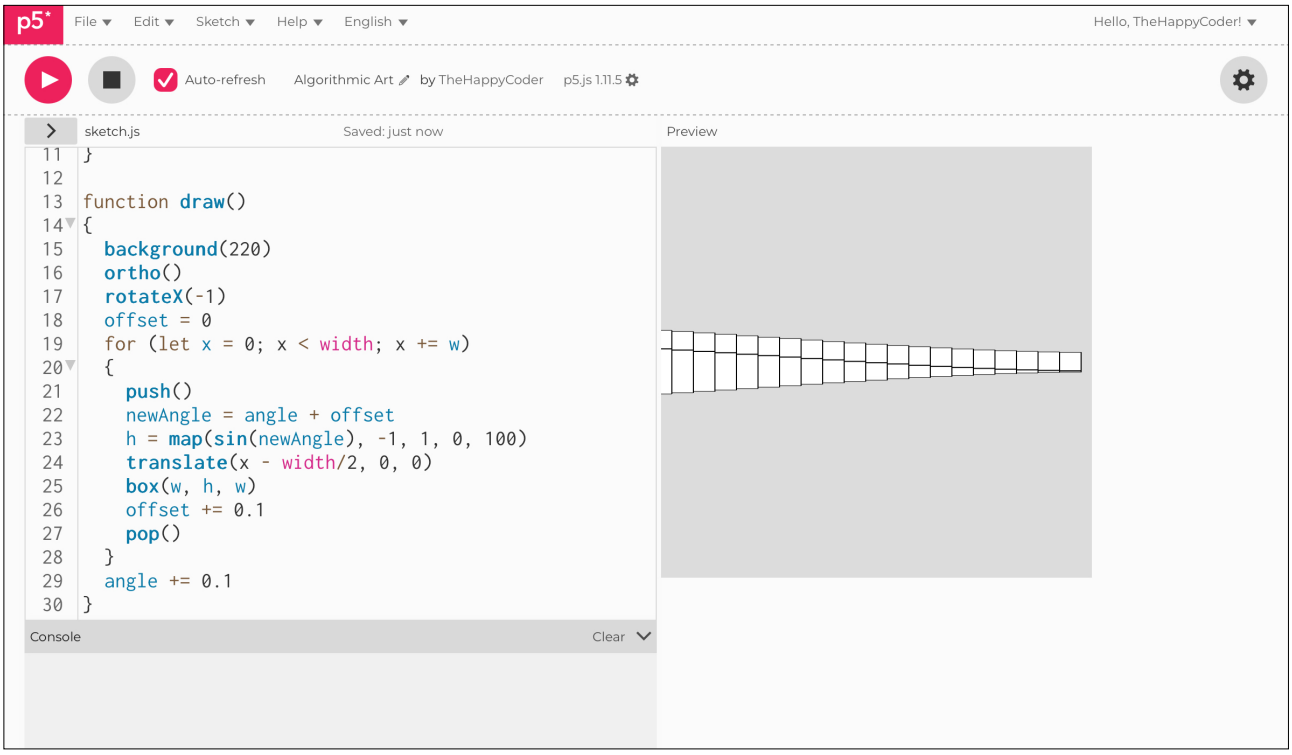
🗒️ Notes

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)
      pop()
```

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

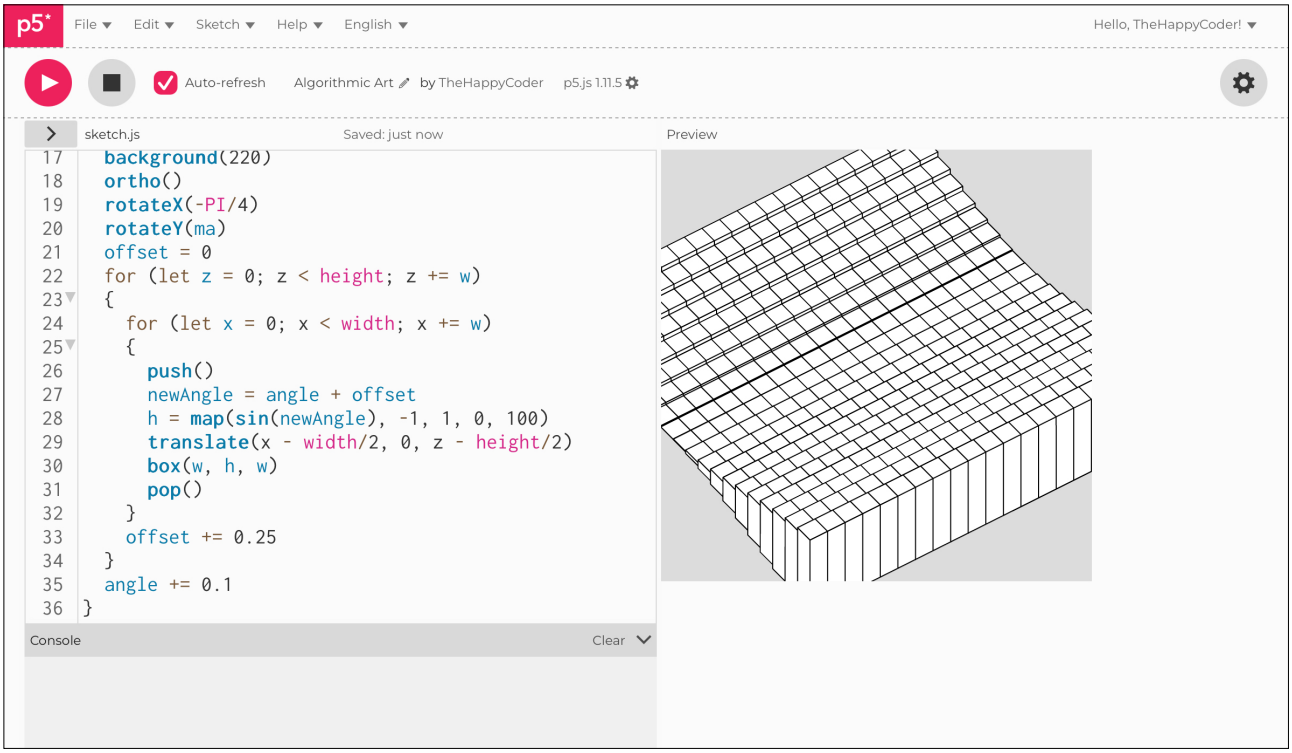
🗒️ Notes

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.

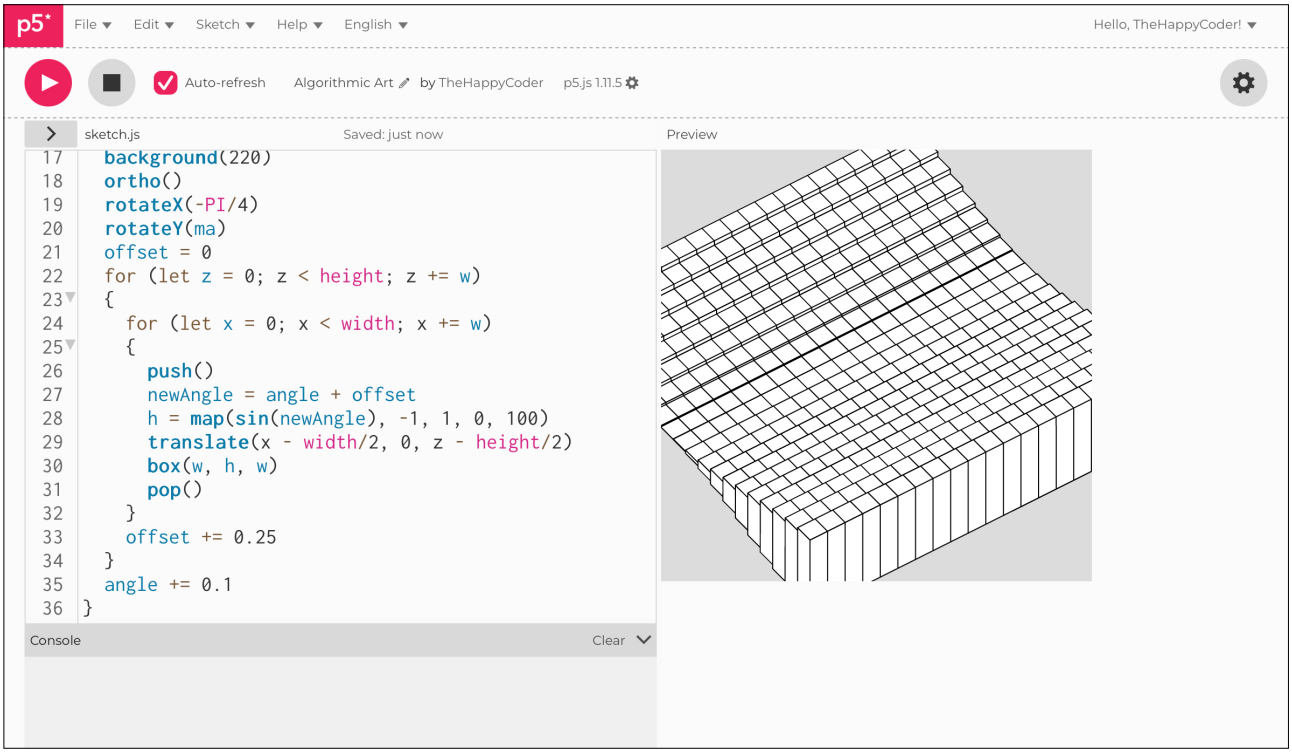
🌻 Challenge

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 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!

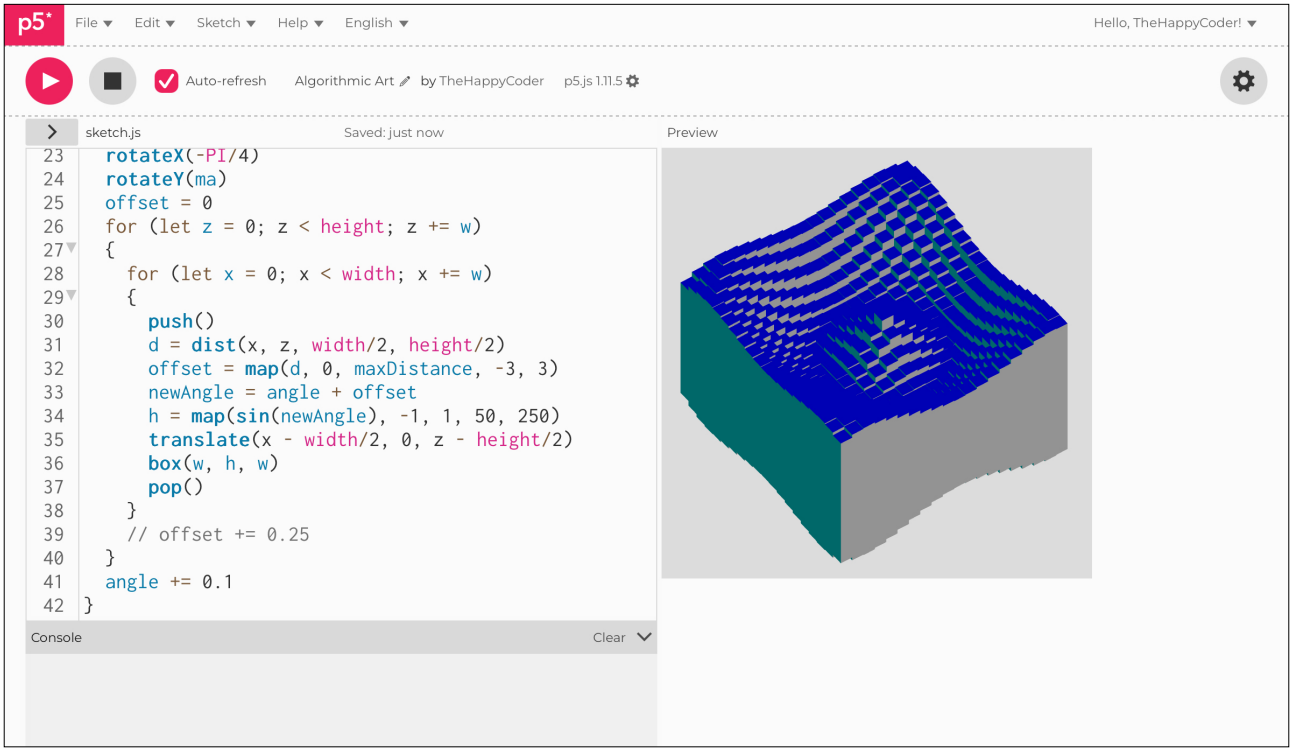
🌻 Challenge

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 saved, the type of file, how many frames and any options stated