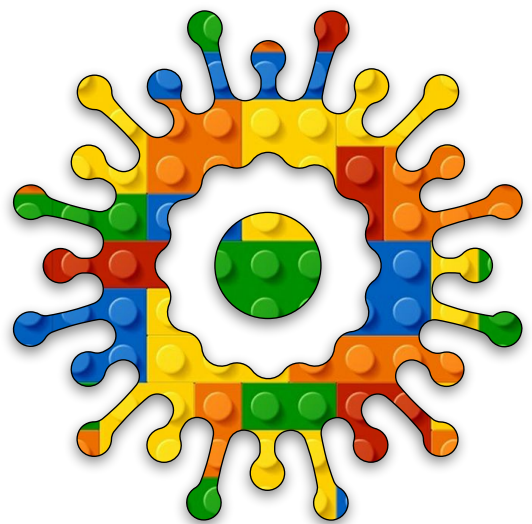


Creative
Coding
Module D
Unit #8
3D arrays
part 2





Module D Unit #8 3D arrays part 2

Sketch D8.1	an array of cubes
Sketch D8.2	rotating the array
Sketch D8.3	spaced out
Sketch D8.4	an array in the y direction
Sketch D8.5	a cube of cubes
Sketch D8.6	adding a splash of colour
Sketch D8.7	a bubble factory
Sketch D8.8	not a cube
Sketch D8.9	now a circle
Sketch D8.10	creating a spiral
Sketch D8.11	more spirally



Introduction to 3D arrays part 2

We are going to develop further the concept of a 3D array and create some interesting shapes and patterns. First off is a cube of cubes, and then we will have a spiral of bubbles!



Sketch D8.1 an array of cubes

Starting a new sketch with many of the previous elements. Making an array of **cubes** where we have **0** to **100** in steps of **20** for a cube size of **20**, giving us **five** cubes. We do this with a simple **for()** loop.

```
let cubes = []
let cube

function setup()
{
  createCanvas(400, 400, WEBGL)
  for (let i = 0; i < 100; i += 20)
  {
    cube = new Cube(i, 0, 0)
    cubes.push(cube)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < cubes.length; i++)
  {
    cubes[i].show()
  }
}

class Cube
{
  constructor(x, y, z)
  {
    this.x = x
```

```
    this.y = y
    this.z = z
  }

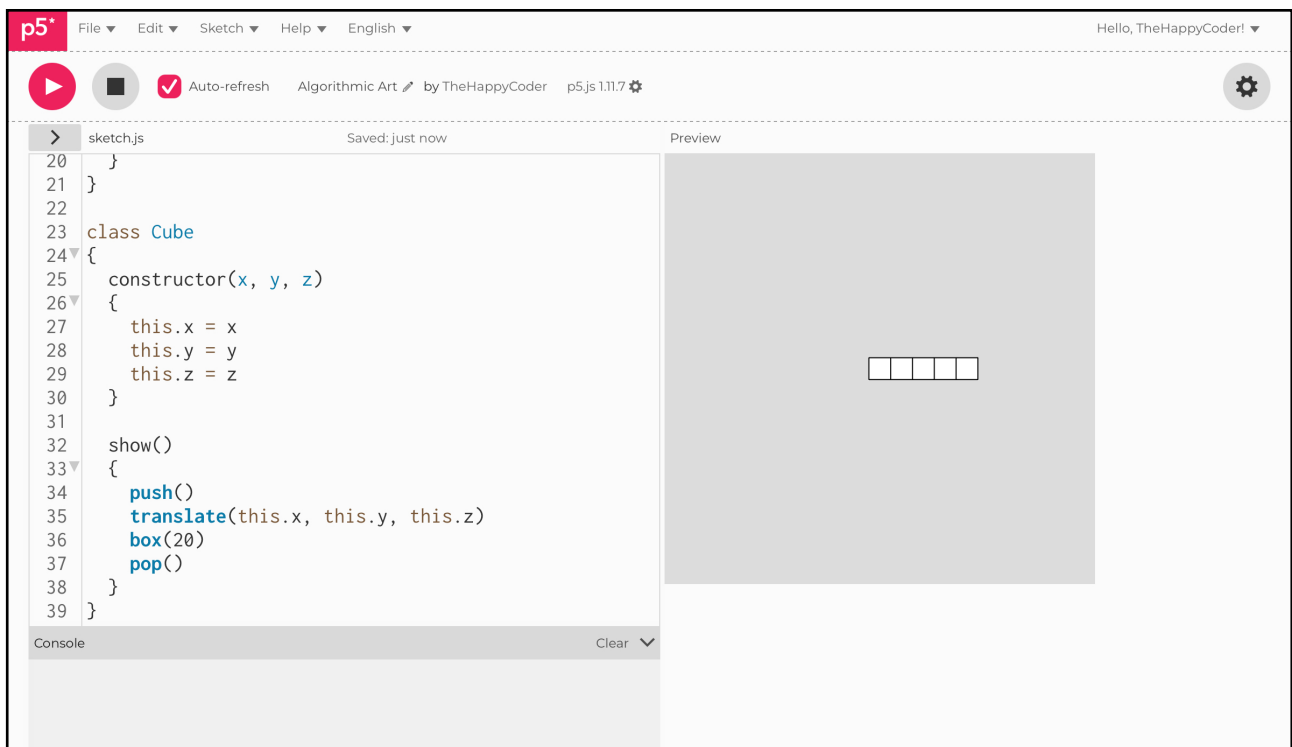
  show()
  {
    push()
    translate(this.x, this.y, this.z)
    box(20)
    pop()
  }
}
```



Notes

As you can see, we have a problem. We have drawn the cubes from (0, 0), which is the centre of the canvas. We need to start at -100 rather than 0. Also, we can only see from the front; it would be nice to rotate the array of cubes to see them properly.

Figure D8.1





Sketch D8.2 rotating the array

Now we rotate the array about the centre of the canvas. Also, we have now got **ten** cubes.

```
let cubes = []
let cube
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = -100; i < 100; i += 20)
  {
    cube = new Cube(i, 0, 0)
    cubes.push(cube)
  }
}

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  for (let i = 0; i < cubes.length; i++)
  {
    cubes[i].show()
  }
  angle += 0.5
}
```

```
class Cube
{
  constructor(x, y, z)
  {
    this.x = x
    this.y = y
    this.z = z
  }

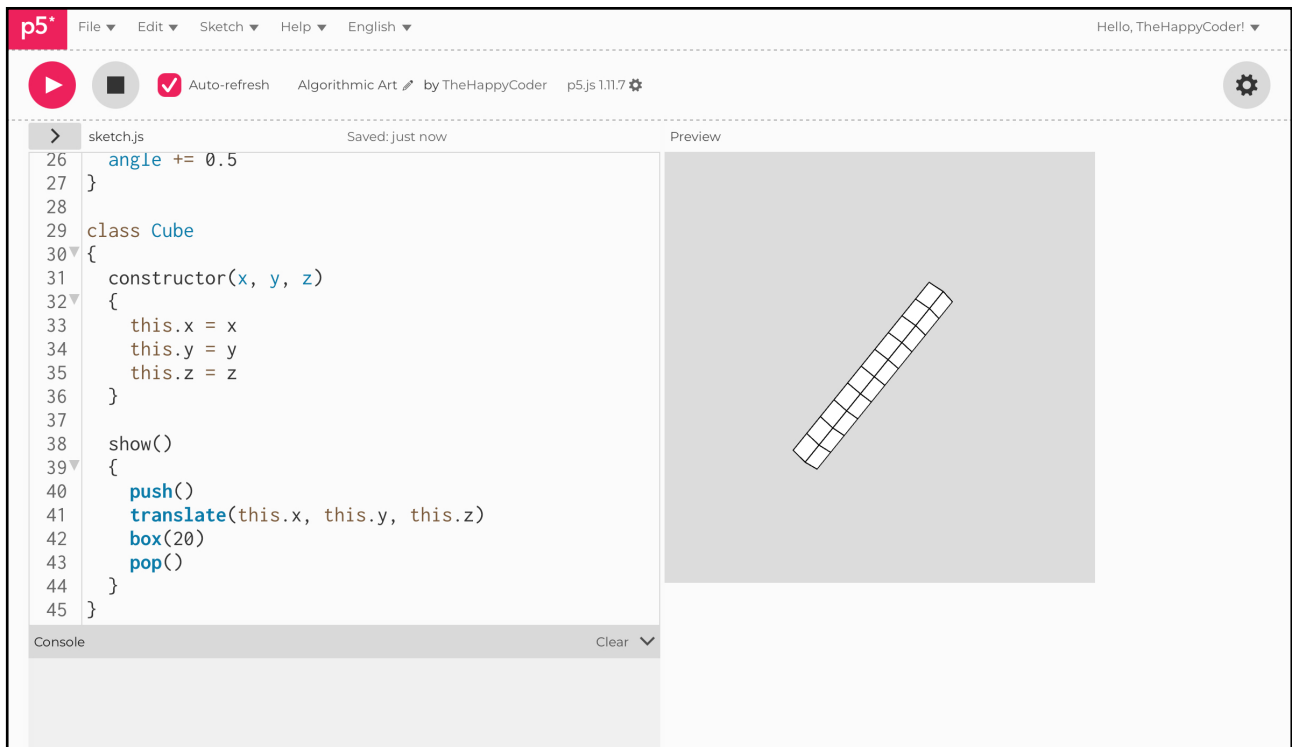
  show()
  {
    push()
    translate(this.x, this.y, this.z)
    box(20)
    pop()
  }
}
```



Notes

A long stick of cubes or a long cuboid

Figure D8.2





Sketch D8.3 spaced out

If we space them out a little bit more by increasing the steps from **20** to **25**, this will also mean that we have only **eight** cubes.

```
let cubes = []
let cube
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = -100; i < 100; i += 25)
  {
    cube = new Cube(i, 0, 0)
    cubes.push(cube)
  }
}

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  for (let i = 0; i < cubes.length; i++)
  {
    cubes[i].show()
  }
  angle += 0.5
}
```

```
class Cube
{
  constructor(x, y, z)
  {
    this.x = x
    this.y = y
    this.z = z
  }

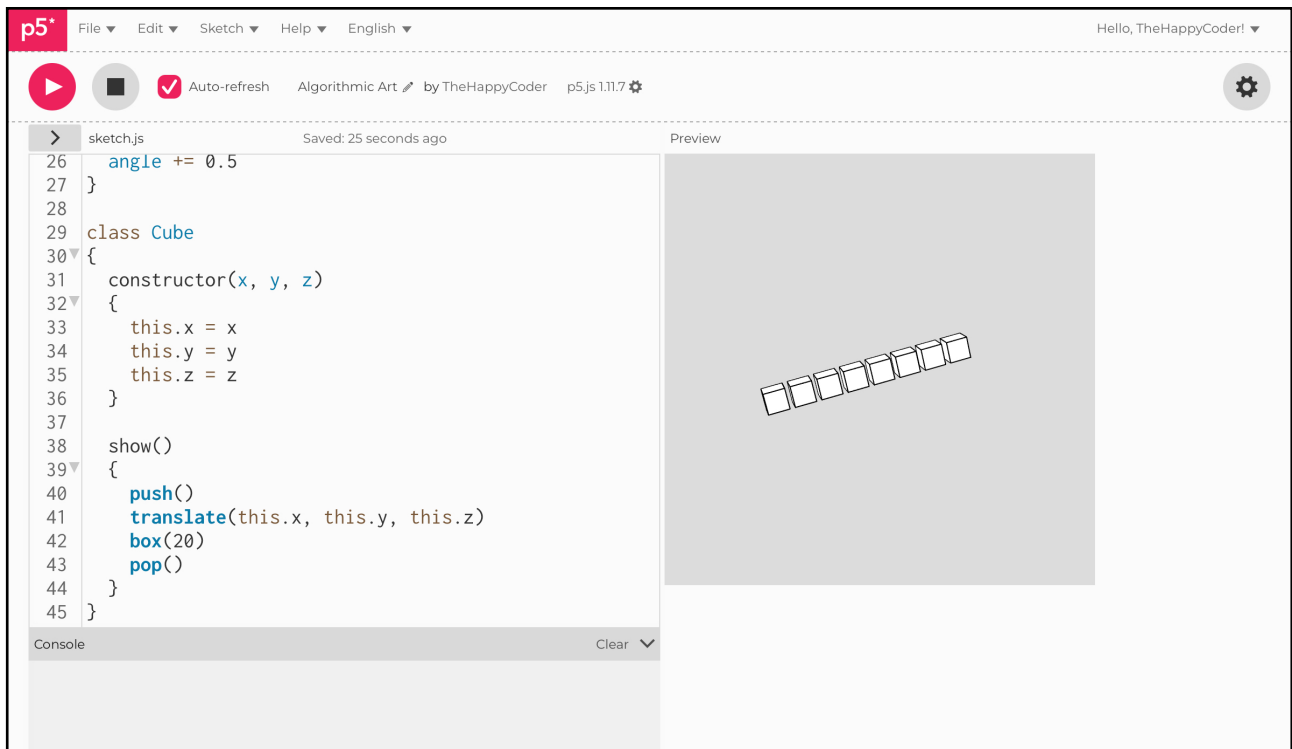
  show()
  {
    push()
    translate(this.x, this.y, this.z)
    box(20)
    pop()
  }
}
```



Notes

A bit more spaced out.

Figure D8.3





Sketch D8.4 an array in the y direction

Yet we can do even better because we can create an array in the **y** direction also. For that, we need a nested loop.

```
let cubes = []
let cube
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = -100; i < 100; i += 25)
  {
    for (let j = -100; j < 100; j += 25)
    {
      cube = new Cube(i, j, 0)
      cubes.push(cube)
    }
  }
}

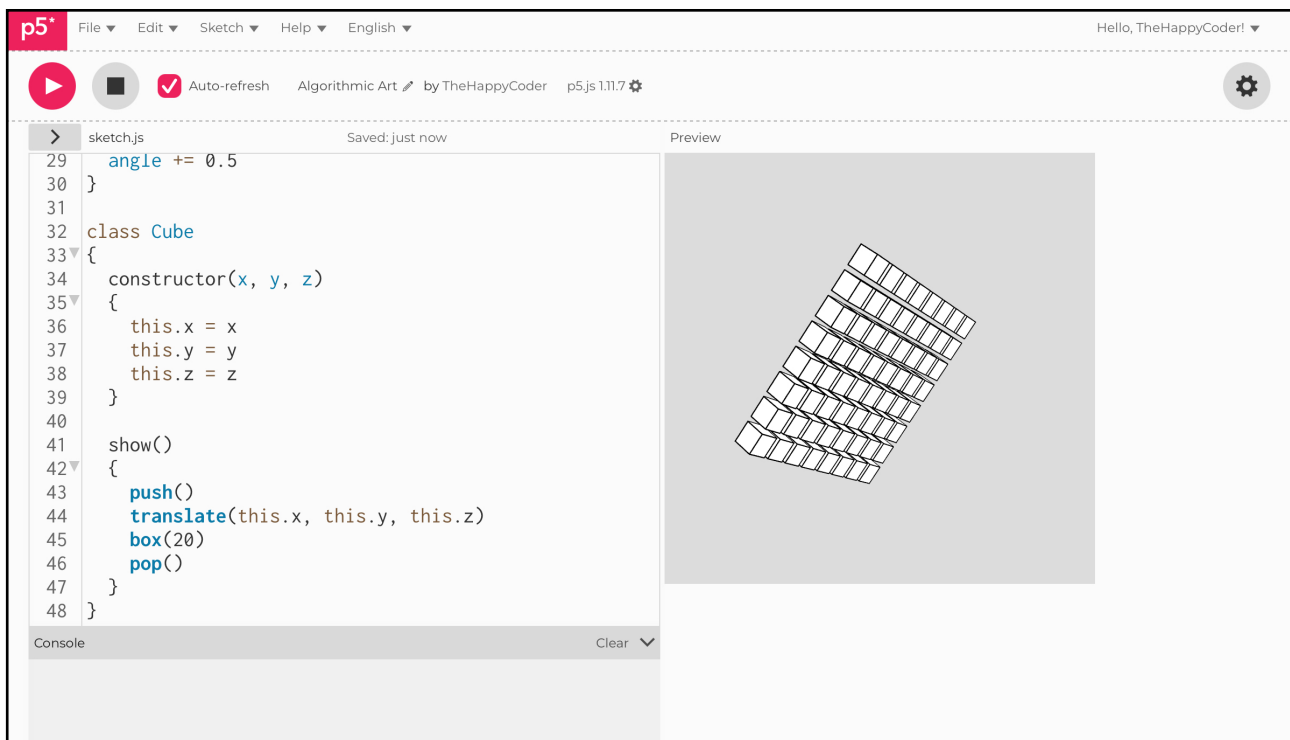
function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  for (let i = 0; i < cubes.length; i++)
  {
    cubes[i].show()
  }
}
```

```
    angle += 0.5
}

class Cube
{
    constructor(x, y, z)
    {
        this.x = x
        this.y = y
        this.z = z
    }

    show()
    {
        push()
        translate(this.x, this.y, this.z)
        box(20)
        pop()
    }
}
```

Figure D8.4





Sketch D8.5 a cube of cubes

But why stop there? Let's add the **third** dimension, the **z** direction; we just add another nested loop.

```
let cubes = []
let cube
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = -100; i < 100; i += 25)
  {
    for (let j = -100; j < 100; j += 25)
    {
      for (let k = -100; k < 100; k += 25)
      {
        cube = new Cube(i, j, k)
        cubes.push(cube)
      }
    }
  }
}

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  for (let i = 0; i < cubes.length; i++)
```

```
{
    cubes[i].show()
}
angle += 0.5
}

class Cube
{
    constructor(x, y, z)
    {
        this.x = x
        this.y = y
        this.z = z
    }

    show()
    {
        push()
        translate(this.x, this.y, this.z)
        box(20)
        pop()
    }
}
```



Notes

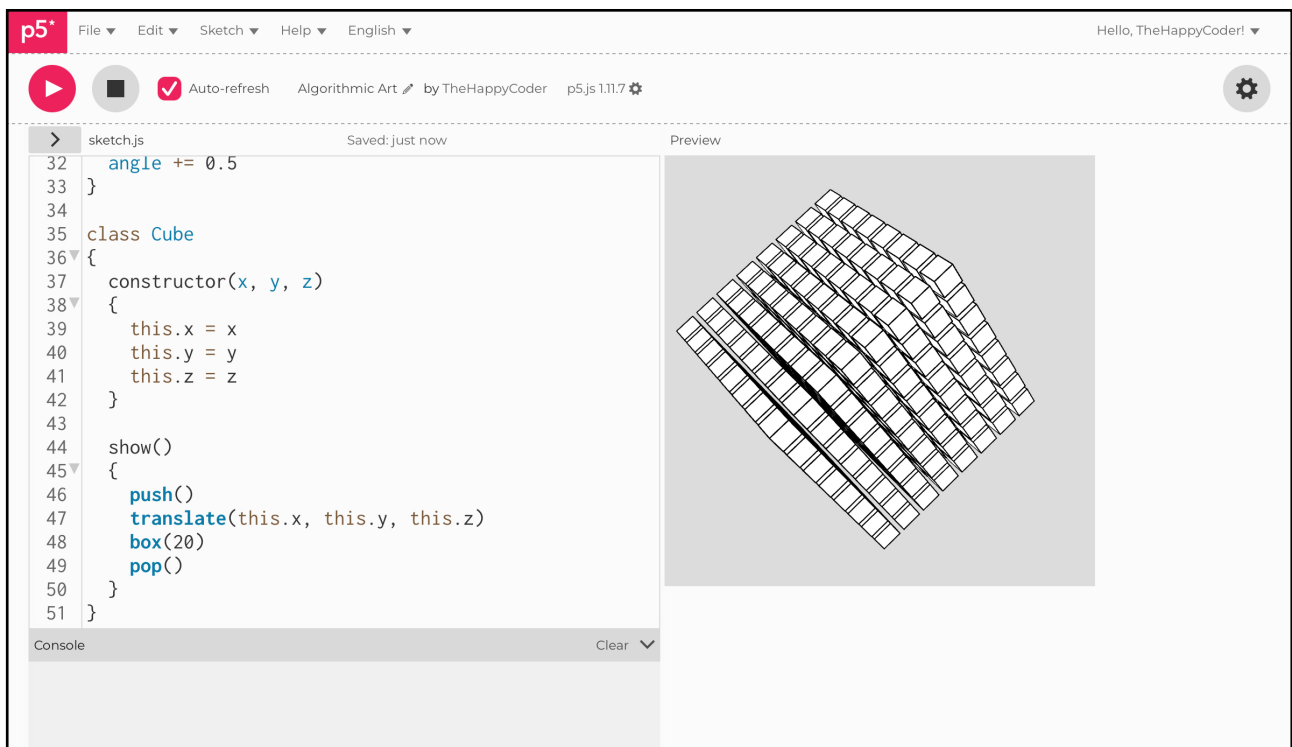
Inside the `cubes[]` array, we have a 3D array.



Challenge

Have a look inside the `cubes[]` array with `console.log()`.

Figure D8.5





Sketch D8.6 adding a splash of colour

We can add individual colours (random) to each cube. We use a function called `color(a, b, c)` with three arguments for the red, green, and blue. We add a variable (`c`) to carry this information to the new cube being created.

```
let cubes = []
let cube
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = -100; i < 100; i += 25)
  {
    for (let j = -100; j < 100; j += 25)
    {
      for (let k = -100; k < 100; k += 25)
      {
        let c = color(random(255), random(255), random(255))
        cube = new Cube(i, j, k, c)
        cubes.push(cube)
      }
    }
  }
}

function draw()
{
  background(220)
  rotateX(angle)
```

```

    rotateY(angle)
    rotateZ(angle)
    for (let i = 0; i < cubes.length; i++)
    {
        cubes[i].show()
    }
    angle += 0.5
}

class Cube
{
    constructor(x, y, z, c)
    {
        this.x = x
        this.y = y
        this.z = z
        this.c = c
    }

    show()
    {
        push()
        fill(this.c)
        translate(this.x, this.y, this.z)
        box(20)
        pop()
    }
}

```



Notes

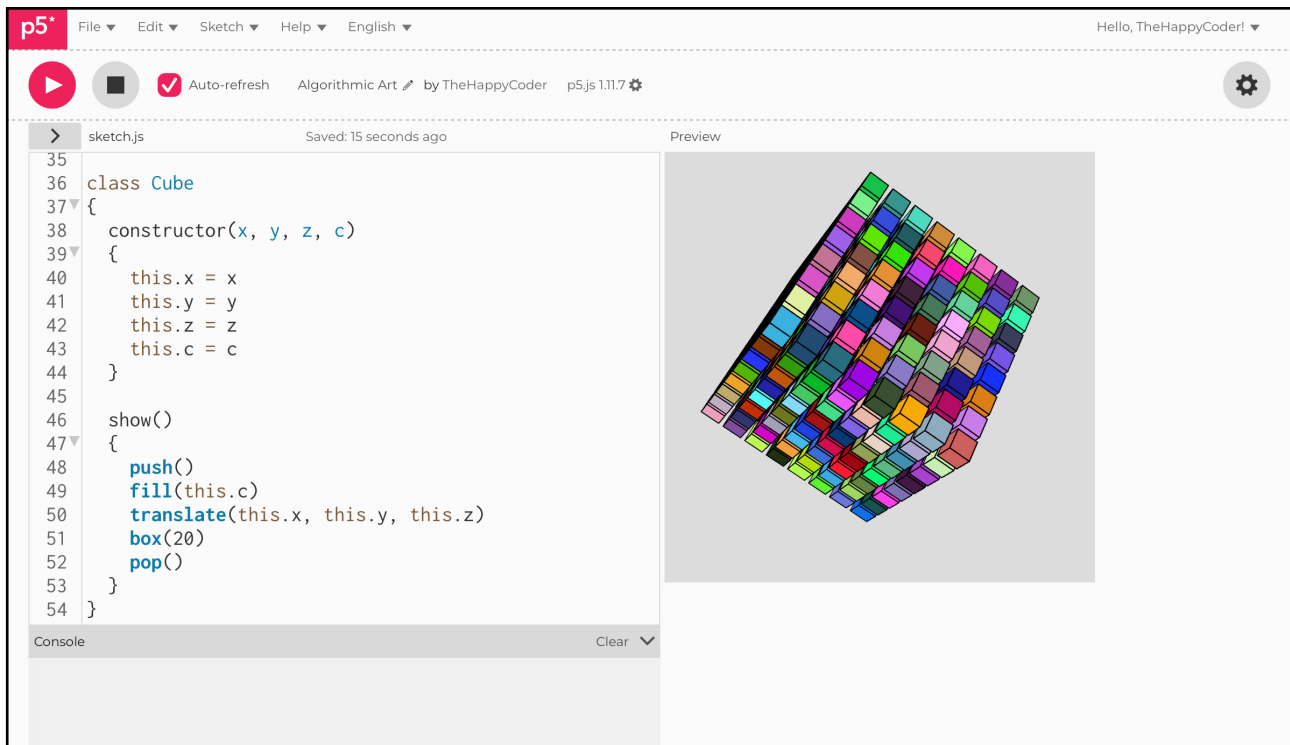
We give each individual cube its own colour.



Challenges

1. Add some `alpha` (fourth argument) to the colour to make it transparent and remove the `stroke()`.
2. Change the shape to a sphere or cylinder.

Figure D8.6





Sketch D8.7 a bubble factory

A bit of refactoring. We are going to use a sphere, so we will change the name of the class, variable, and array to something more appropriate and familiar: **bubble**. Also added **sphere(10)**, **lights()**, and **noStroke()**.

```
let bubbles = []
let bubble
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = -100; i < 100; i += 25)
  {
    for (let j = -100; j < 100; j += 25)
    {
      for (let k = -100; k < 100; k += 25)
      {
        let c = color(random(255), random(255), random(255))
        bubble = new Bubble(i, j, k, c)
        bubbles.push(bubble)
      }
    }
  }
}

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
```

```
rotateZ(angle)
for (let i = 0; i < bubbles.length; i++)
{
  bubbles[i].show()
}
angle += 0.5
}
```

```
class Bubble
{
  constructor(x, y, z, c)
  {
    this.x = x
    this.y = y
    this.z = z
    this.c = c
  }

  show()
  {
    push()
    fill(this.c)
    translate(this.x, this.y, this.z)
    lights()
    noStroke()
    sphere(10)
    pop()
  }
}
```



Notes

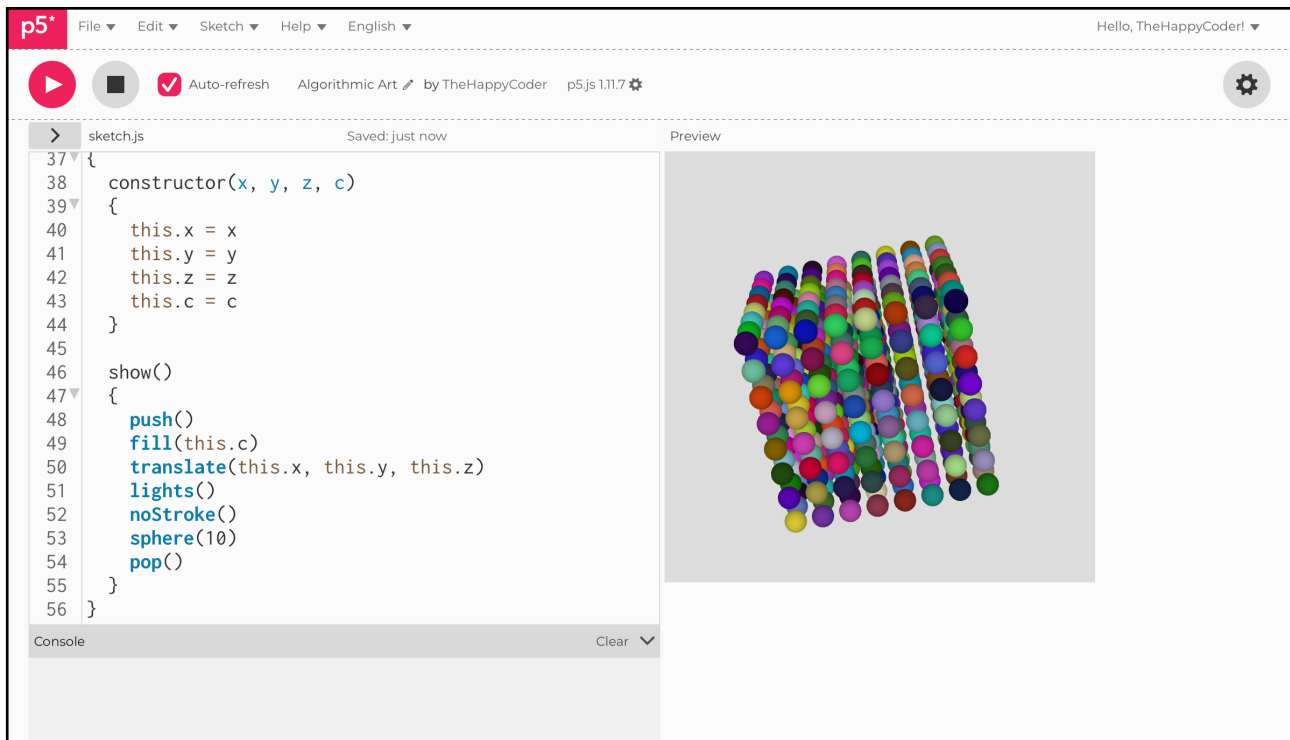
The `noStroke()` is just so that you can see the colours.



Challenge

Call the variable, class, and array anything you want.

Figure D8.7





Sketch D8.8 not a cube

Our next step is putting all those bubbles in a circle rather than a cube, but first remove all the code highlighted and commented, then change the **constructor()** function.

```
let bubbles = []
let bubble
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  // for (let i = -100; i < 100; i += 25)
  // {
  //   for (let j = -100; j < 100; j += 25)
  //   {
  //     for (let k = -100; k < 100; k += 25)
  //     {
  //       let c = color(random(255), random(255), random(255))
  //       bubble = new Bubble(i, j, k, c)
  //       bubbles.push(bubble)
  //     }
  //   }
  // }

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
```

```
rotateZ(angle)
for (let i = 0; i < bubbles.length; i++)
{
    bubbles[i].show()
}
angle += 0.5
}

class Bubble
{
    constructor(x, y, z)
    {
        this.x = x
        this.y = y
        this.z = z
        // this.c = c
    }

    show()
    {
        push()
        // fill(this.c)
        translate(this.x, this.y, this.z)
        lights()
        noStroke()
        sphere(10)
        pop()
    }
}
```



Notes

Don't try running this just yet!



Sketch D8.9 now a circle

Creating a circle of bubbles.

```
let bubbles = []
let bubble
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = 0; i < 360; i += 30)
  {
    let x = 100 * sin(i)
    let y = 100 * cos(i)
    bubble = new Bubble(x, y, 0)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
  }
  angle += 0.5
}
```

```
class Bubble
{
  constructor(x, y, z)
  {
    this.x = x
    this.y = y
    this.z = z
  }

  show()
  {
    push()
    translate(this.x, this.y, this.z)
    lights()
    fill('yellow')
    noStroke()
    sphere(10)
    pop()
  }
}
```



Notes

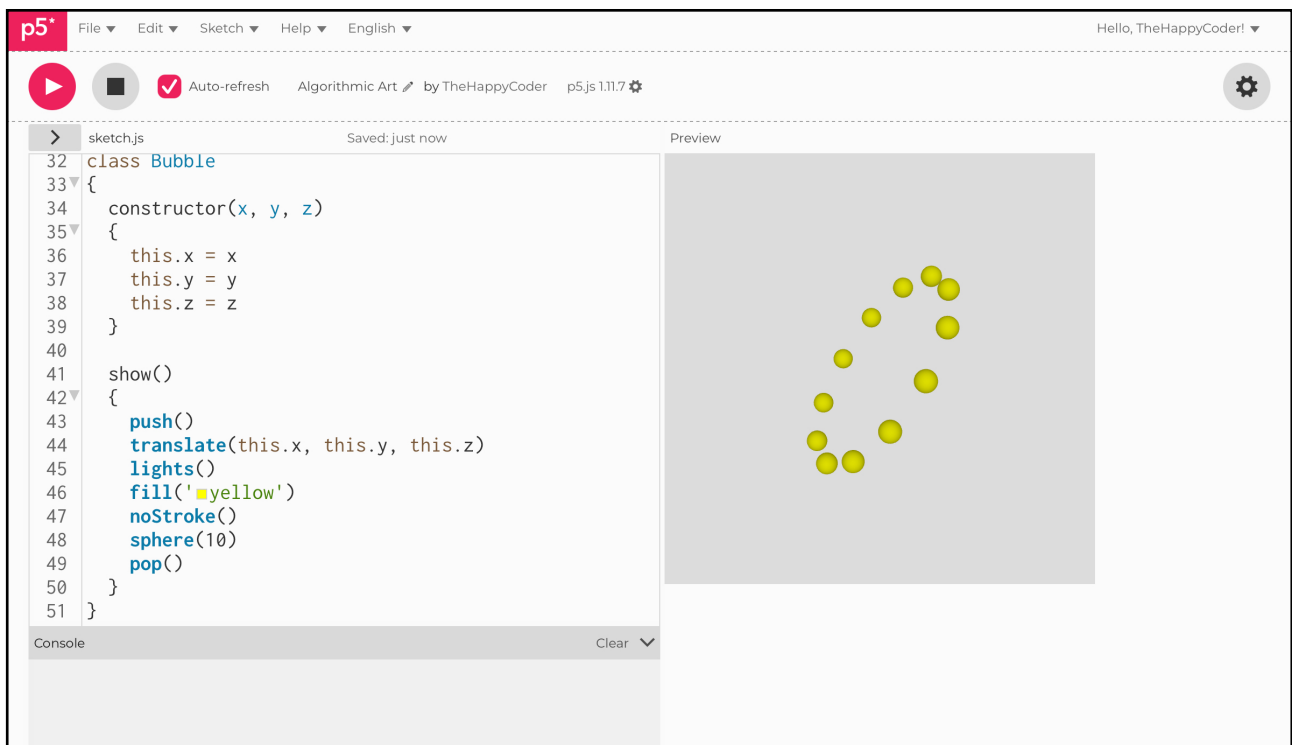
We have a nice ring of spheres.



Challenge

Add other lights or materials.

Figure D8.9





Sketch D8.10 creating a spiral

Next, let's try to make a spiral so that the array of bubbles also has a **z** component.

```
let bubbles = []
let bubble
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = 0; i < 360; i += 30)
  {
    let x = 100 * sin(i)
    let y = 100 * cos(i)
    bubble = new Bubble(x, y, i)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
  }
  angle += 0.5
}
```

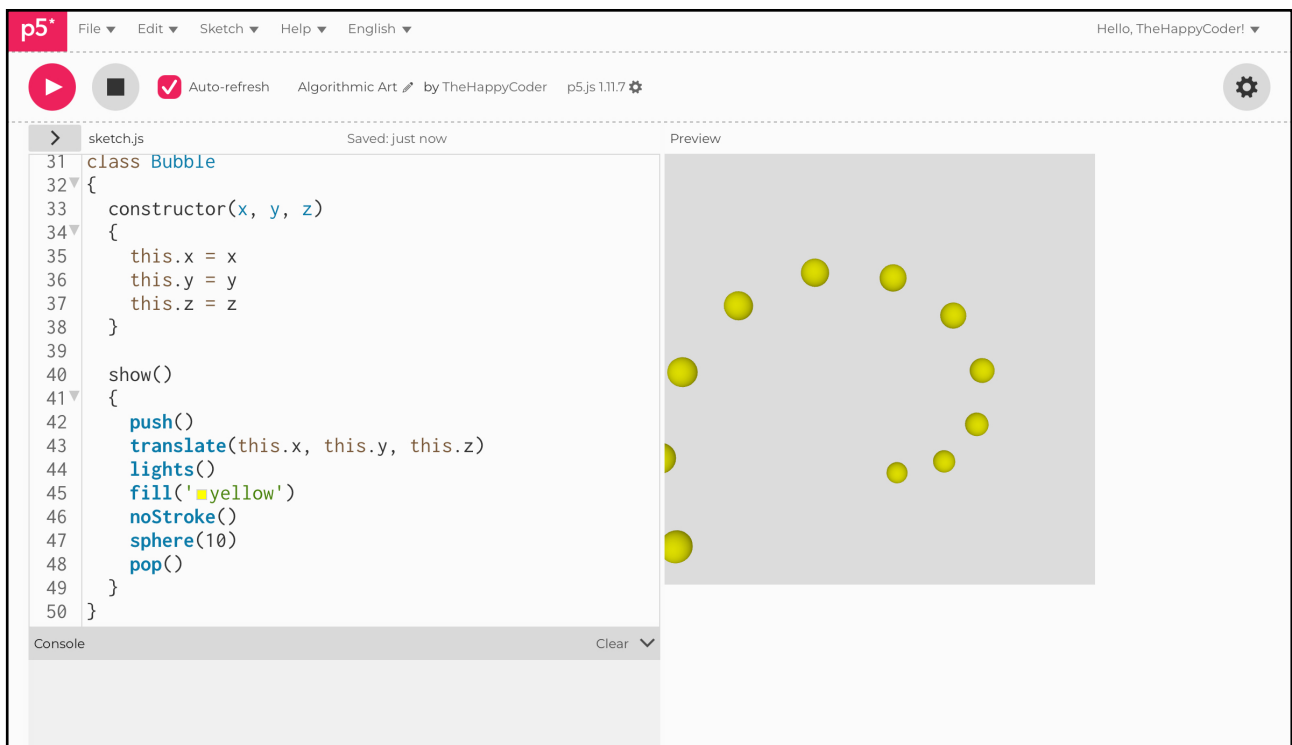
```
}  
  
class Bubble  
{  
  constructor(x, y, z)  
  {  
    this.x = x  
    this.y = y  
    this.z = z  
  }  
  
  show()  
  {  
    push()  
    translate(this.x, this.y, this.z)  
    lights()  
    fill('yellow')  
    noStroke()  
    sphere(10)  
    pop()  
  }  
}
```



Notes

It's a spiral Jim, but not as we know it. Let's see if we can improve.

Figure D8.10





Sketch D8.11 more spirally

We can make some improvements to this to make it more of a spiral.

```
let bubbles = []
let bubble
let angle = 0

function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  for (let i = -360; i < 360; i += 10)
  {
    let x = 100 * sin(i)
    let y = 100 * cos(i)
    bubble = new Bubble(x, y, i/4)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
  }
  angle += 0.5
}
```

```
class Bubble
{
  constructor(x, y, z)
  {
    this.x = x
    this.y = y
    this.z = z
  }

  show()
  {
    push()
    translate(this.x, this.y, this.z)
    lights()
    fill('yellow')
    noStroke()
    sphere(10)
    pop()
  }
}
```



Notes

Much better.



Challenges

1. Can you randomise the diameter of the sphere?
2. Add random colours to the spheres.
3. Change the radius of the spiral so that it starts small and increases.
4. How would you extend it?
5. How would you rotate it about an axis?

Figure D8.11

