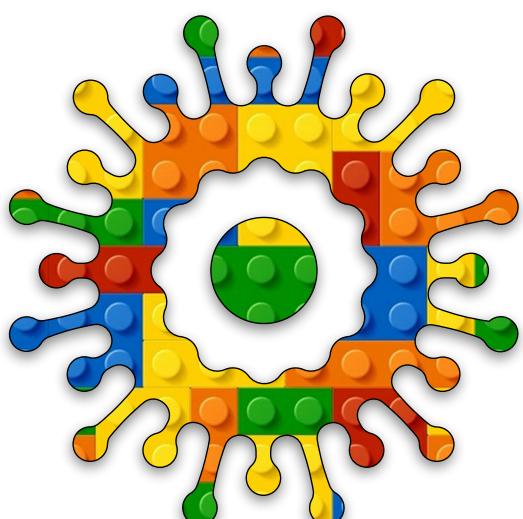


# Creative Coding Module D Unit #3

## array of objects





### Module D Unit #3 array of objects

- Sketch D3.1 bubbles
- Sketch D3.2 classy bubbles
- Sketch D3.3 time for an argument
- Sketch D3.4 drawing the bubbles
- Sketch D3.5 random motion
- Sketch D3.6 many bubbles
- Sketch D3.7 more bubbles
- Sketch D3.8 spacing them out
- Sketch D3.9 random
- Sketch D3.10 the shimmering fog
- Sketch D3.11 simpler times
- Sketch D3.12 this is what you get
- Sketch D3.13 mouse click
- Sketch D3.14 when push comes to shove
- Sketch D3.15 a bit of a drag



## Introduction to array of objects

In this unit, we will mash together arrays and classes with a dash of mouse functions.

Recap: Arrays are extremely useful and powerful in so many ways. An array is a list of elements inside square brackets separated by a comma. The numbering of each element starts at zero; they are called the index number. Let's create an array called num and put some numbers in it.

```
let num = [42, 56, 63, 79, 88]
```

There are five numbers (elements) in the array. They are, in order: the first one is **42**, the second one is **56**, the third one is **63**, the fourth one is **79**, and the fifth one is **88**.

However, to reference the first element (**42**), we say it is at index **0**. The second element (**56**) is at index **1**, the third (**63**) is index **2**, the fourth (**79**) is index **3**, and the fifth (**88**) is at index **4**. This may seem a bit silly, but it is the way it is, so you will need to get used to it.



## Sketch D3.1 bubbles

! create a new sketch.

We are going to make an array of objects. We start with an empty array called **bubbles**.

```
let bubbles = []  
  
function setup()  
{  
    createCanvas(400, 400)  
}  
  
function draw()  
{  
    background(220)  
}
```



## Sketch D3.2 classy bubbles

We are going to put **bubble** objects in that array, but first we need to create a **Bubble** class to create some **bubbles**.

```
let bubbles = []\n\nfunction setup()\n{\n    createCanvas(400, 400)\n}\n\nfunction draw()\n{\n    background(220)\n}\n\nclass Bubble\n{\n    constructor()\n    {\n    }\n\n    move()\n    {\n    }\n\n    show()\n    {\n    }\n}
```

```
    }  
}
```



## Sketch D3.3 time for an argument

We need three arguments for the bubble: the `x`, `y` (coordinates), and `r` (radius). The radius will be randomly generated. The `x` and `y` positions will be created with a click or drag of the mouse.

```
let bubbles = []  
  
function setup()  
{  
    createCanvas(400, 400)  
}  
  
function draw()  
{  
    background(220)  
}  
  
class Bubble  
{  
    constructor(x, y, r)  
    {  
        this.x = x  
        this.y = y  
        this.r = r  
    }  
  
    move()  
    {  
    }  
}
```

```
show()
{
    circle(this.x, this.y, this.r)
}
}
```

## Notes

We are not moving anything because we haven't drawn a bubble. So there is nothing to see just yet; all will be revealed soon.



## Sketch D3.4 drawing the bubbles

Let us draw one bubble to start with.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    bubble = new Bubble(100, 100, 100)

}

function draw()
{
    background(220)
    bubble.show()

}

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

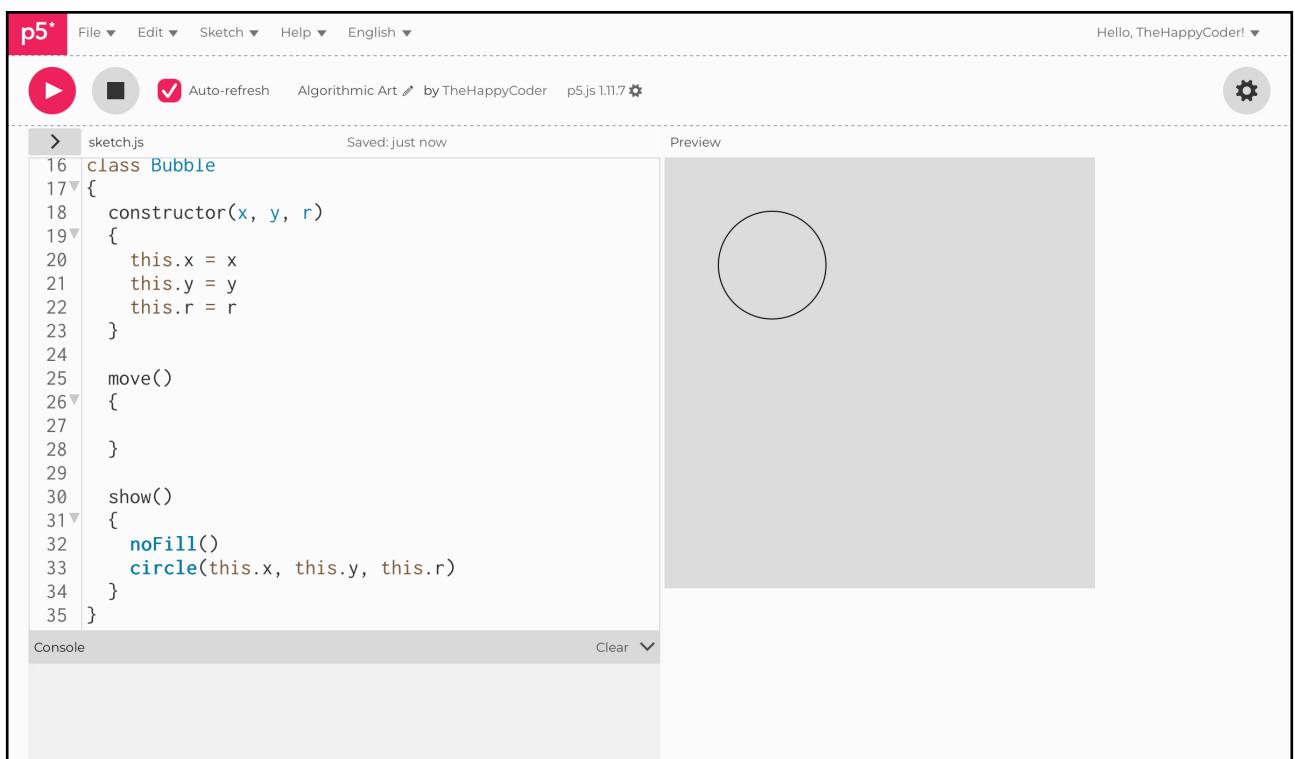
    move()
    {
    }
}
```

```
show()
{
    noFill()
    circle(this.x, this.y, this.r)
}
}
```

## Notes

We finally have something to see, our first bubble.

Figure D3.4



The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and user information ('Hello, TheHappyCoder!'). Below the toolbar, the file name is 'sketch.js' and it was 'Saved: just now'. The code editor displays the following JavaScript code:

```
sketch.js
16 class Bubble
17 {
18   constructor(x, y, r)
19   {
20     this.x = x
21     this.y = y
22     this.r = r
23   }
24
25   move()
26   {
27
28   }
29
30   show()
31   {
32     noFill()
33     circle(this.x, this.y, this.r)
34   }
35 }
```

To the right of the code editor is a preview window titled 'Preview' which shows a single, empty circle.



## Sketch D3.5 random motion

Let's move it around in a random manner.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    bubble = new Bubble(100, 100, 100)
}

function draw()
{
    background(220)
    bubble.show()
    bubble.move()
}

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

    move()
    {
        this.x = this.x + random(-5, 5)
    }
}
```

```
this.y = this.y + random(-5, 5)  
}  
  
show()  
{  
    noFill()  
    circle(this.x, this.y, this.r)  
}  
}
```

## Notes

What you should see is a circle wobbling around the canvas (or wandering off it eventually).

Figure D3.5

The screenshot shows the p5.js code editor interface. At the top, there are buttons for play/pause, stop, auto-refresh (which is checked), and settings. The title bar says "sketch.js" and "Saved: just now". The right side of the interface is a preview window showing a single white circle on a gray background. The code in the editor is as follows:

```
18<?
19  constructor(x, y, r)
20  {
21    this.x = x
22    this.y = y
23    this.r = r
24  }
25
26  move()
27  {
28    this.x = this.x + random(-5, 5)
29    this.y = this.y + random(-5, 5)
30  }
31
32  show()
33  {
34    noFill()
35    circle(this.x, this.y, this.r)
36  }
37 }
```

Below the code editor is a "Console" section which is currently empty.



## Sketch D3.6 many bubbles

We can create many of them with a nice `for()` loop and put them into the `bubbles` array. Then we have another `for()` loop to pull them out. Here we are just creating and drawing one bubble for the moment.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 1; i++)
    {
        bubbles[i] = new Bubble(100, 100, 100)
    }
}

function draw()
{
    background(220)
    for (let i = 0; i < 1; i++)
    {
        bubbles[i].show()
        bubbles[i].move()
    }
}

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

```
    this.y = y
    this.r = r
}

move()
{
    this.x = this.x + random(-5, 5)
    this.y = this.y + random(-5, 5)
}

show()
{
    noFill()
    circle(this.x, this.y, this.r)
}
}
```

## Notes

We have not really changed anything; you should have the same effect.

Figure D3.6

The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by menu items: File, Edit, Sketch, Help, and English. To the right, it says "Hello, TheHappyCoder!" with a gear icon. The main area has tabs for "sketch.js" and "Preview". The code editor contains the following JavaScript code:

```
24> sketch.js          Saved: just now
25  constructor(x, y, r)
26  {
27    this.x = x
28    this.y = y
29    this.r = r
30  }
31
32  move()
33  {
34    this.x = this.x + random(-5, 5)
35    this.y = this.y + random(-5, 5)
36  }
37
38  show()
39  {
40    noFill()
41    circle(this.x, this.y, this.r)
42  }
43 }
```

The preview window shows a single gray circle centered on a gray background, representing the output of the running code.



## Sketch D3.7 more bubbles

We only drew one **bubble** above. There is only one **bubble** in the array of **bubbles**. We can change that by increasing it from **1** to **10**. In the second **for()** loop, we can call the length of the array rather than hardcoded it just in case we change the length of the array.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 10; i++)
    {
        bubbles[i] = new Bubble(100, 100, 100)
    }
}

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

class Bubble
{
    constructor(x, y, r)
    {
```

```
    this.x = x
    this.y = y
    this.r = r
}

move()
{
    this.x = this.x + random(-5, 5)
    this.y = this.y + random(-5, 5)
}

show()
{
    noFill()
    circle(this.x, this.y, this.r)
}
}
```

Figure D3.7

The screenshot shows the p5.js code editor interface. At the top, there are navigation menus: File ▾, Edit ▾, Sketch ▾, Help ▾, and English ▾. On the right, it says "Hello, TheHappyCoder! ▾". Below the menu bar, there are icons for play, stop, and refresh, followed by "Auto-refresh" with a checkmark, "Algorithmic Art" by TheHappyCoder, and p5.js 1.11.7. A gear icon is also present.

The main area has tabs for "sketch.js" and "Preview". The "sketch.js" tab is active, showing the following code:

```
24<?
25 constructor(x, y, r)
26 {
27   this.x = x
28   this.y = y
29   this.r = r
30 }
31
32 move()
33 {
34   this.x = this.x + random(-5, 5)
35   this.y = this.y + random(-5, 5)
36 }
37
38 show()
39 {
40   noFill()
41   circle(this.x, this.y, this.r)
42 }
43 }
```

The "Preview" tab shows a circular pattern of overlapping circles, indicating the output of the current sketch.



## Sketch D3.8 spacing them out

We started each one in the same spot. Let's space them along a line using the loop. We start at **10** and multiply the index **i** by **40**.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 10; i++)
    {
        let x = 10 + 40 * i
        bubbles[i] = new Bubble(x, 100, 100)
    }
}

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

class Bubble
{
    constructor(x, y, r)
    {
```

```
    this.x = x
    this.y = y
    this.r = r
}

move()
{
    this.x = this.x + random(-5, 5)
    this.y = this.y + random(-5, 5)
}

show()
{
    noFill()
    circle(this.x, this.y, this.r)
}
}
```

## Notes

Notice we haven't touched anything in the class itself.

Figure D3.8

The screenshot shows the p5.js code editor interface. At the top, there are buttons for play/pause, stop, auto-refresh (which is checked), and settings. The title bar says "sketch.js" and "Saved: just now". The right side of the interface is a preview window showing a series of overlapping circles. The code in the editor is as follows:

```
25<?
26 constructor(x, y, r)
27 {
28     this.x = x
29     this.y = y
30     this.r = r
31 }
32
33 move()
34 {
35     this.x = this.x + random(-5, 5)
36     this.y = this.y + random(-5, 5)
37 }
38
39 show()
40 {
41     noFill()
42     circle(this.x, this.y, this.r)
43 }
44 }
```

Below the code editor is a "Console" section which is currently empty.



## Sketch D3.9 random

Let us create a set of bubbles of random positions and sizes.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 10; i++)
    {
        let x = random(width)
        let y = random(height)
        let r = random(20, 100)
        bubbles[i] = new Bubble(x, y, r)
    }
}

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

class Bubble
{
    constructor(x, y, r)
```

```
{  
    this.x = x  
    this.y = y  
    this.r = r  
}  
  
move()  
{  
    this.x = this.x + random(-5, 5)  
    this.y = this.y + random(-5, 5)  
}  
  
show()  
{  
    noFill()  
    circle(this.x, this.y, this.r)  
}  
}
```

Figure D3.9

The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by menu items: File, Edit, Sketch, Help, and English. To the right, it says "Hello, TheHappyCoder!" with a gear icon. The main area has tabs for "sketch.js" and "Preview". The code editor contains the following JavaScript code:

```
27> sketch.js
28
29  constructor(x, y, r)
30  {
31      this.x = x
32      this.y = y
33      this.r = r
34  }
35
36  move()
37  {
38      this.x = this.x + random(-5, 5)
39      this.y = this.y + random(-5, 5)
40  }
41
42  show()
43  {
44      noFill()
45      circle(this.x, this.y, this.r)
46  }

```

The "Preview" window shows a gray canvas with several circles drawn on it. There are two large circles in the center-left, one medium-sized circle above them, and several smaller circles scattered around the perimeter.



## Sketch D3.10 the shimmering fog

That is all well and good, but what can we do to make it more interesting? With a few tweaks, we can do something quite slightly unexpected.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    for (let i = 0; i < 1000; i++)
    {
        let x = random(width)
        let y = random(height)
        let r = random(20, 100)
        bubbles[i] = new Bubble(x, y, r)
    }
}

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

class Bubble
{
    constructor(x, y, r)
```

```
{  
    this.x = x  
    this.y = y  
    this.r = r  
}  
  
move()  
{  
    this.x = this.x + random(-3, 3)  
    this.y = this.y + random(-3, 3)  
}  
  
show()  
{  
    noStroke()  
    fill(51, 10)  
    circle(this.x, this.y, this.r)  
}  
}
```

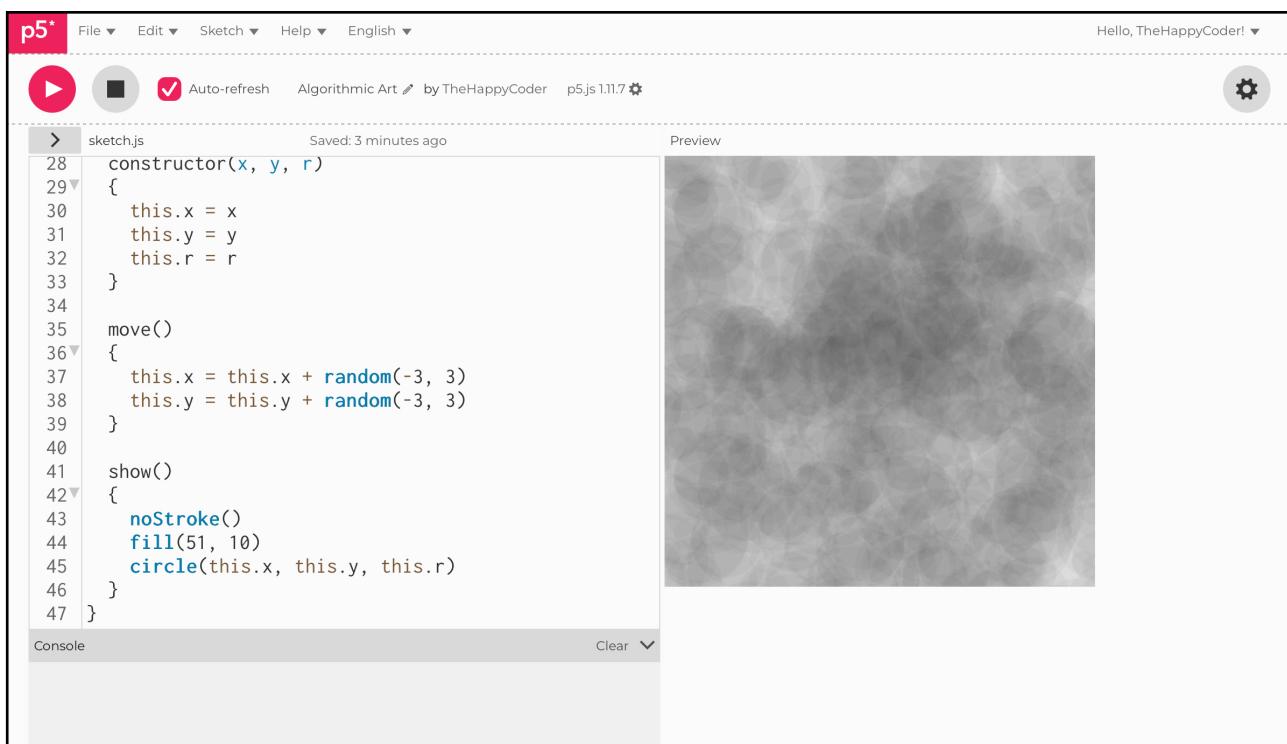
## Notes

Sometimes it doesn't take much.

## Challenge

Play with it.

Figure D3.10



The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by menu items: File, Edit, Sketch, Help, and English. To the right, it says "Hello, TheHappyCoder!" with a dropdown arrow. Below the toolbar, the code editor has a file named "sketch.js" with the extension ".js". It shows the following code:

```
28 | constructor(x, y, r)
29 | {
30 |   this.x = x
31 |   this.y = y
32 |   this.r = r
33 |
34 |
35 | move()
36 | {
37 |   this.x = this.x + random(-3, 3)
38 |   this.y = this.y + random(-3, 3)
39 |
40 |
41 | show()
42 | {
43 |   noStroke()
44 |   fill(51, 10)
45 |   circle(this.x, this.y, this.r)
46 |
47 }
```

The code defines a class with methods for construction, movement, and drawing. The draw method uses the `circle` function to create a series of overlapping circles at random positions, resulting in a textured, organic pattern in the preview window.



## Sketch D3.11 simpler times

Let's go back to a simpler sketch. A bit of deleting and a couple of lines of code will bring you back to a familiar starting point.

! Remove the lines of code highlighted in blue and commented out (//), and add the **noFill()** in the **show()** function.

```
let bubbles = []
let bubble

function setup()
{
    createCanvas(400, 400)
    // for (let i = 0; i < 1000; i++)
    // {
    //     let x = random(width)
    //     let y = random(height)
    //     let r = random(20, 100)
    //     bubbles = new Bubble(x, y, r)
    // }
}

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

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

    move()
    {
        this.x = this.x + random(-3, 3)
        this.y = this.y + random(-3, 3)
    }

    show()
    {
        // noStroke()
        // fill(51, 10)
        noFill()
        circle(this.x, this.y, this.r)
    }
}
```



## Sketch D3.12 this is what you get

You should have this...

```
let bubbles = []
let bubble

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

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

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

    move()
}
```

```
{  
    this.x = this.x + random(-3, 3)  
    this.y = this.y + random(-3, 3)  
}  
  
show()  
{  
    noFill()  
    circle(this.x, this.y, this.r)  
}  
}
```

## Notes

Nothing should happen! No bubbles, we want to draw bubbles where we click the mouse.



## Sketch D3.13 mouse click

Creating a new bubble at the click of a mouse, we create a function called **mousePressed()** to do this.

```
let bubbles = []
let bubble

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

function mousePressed()
{
    bubble = new Bubble(mouseX, mouseY, 20)
    bubbles[0] = bubble
}

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

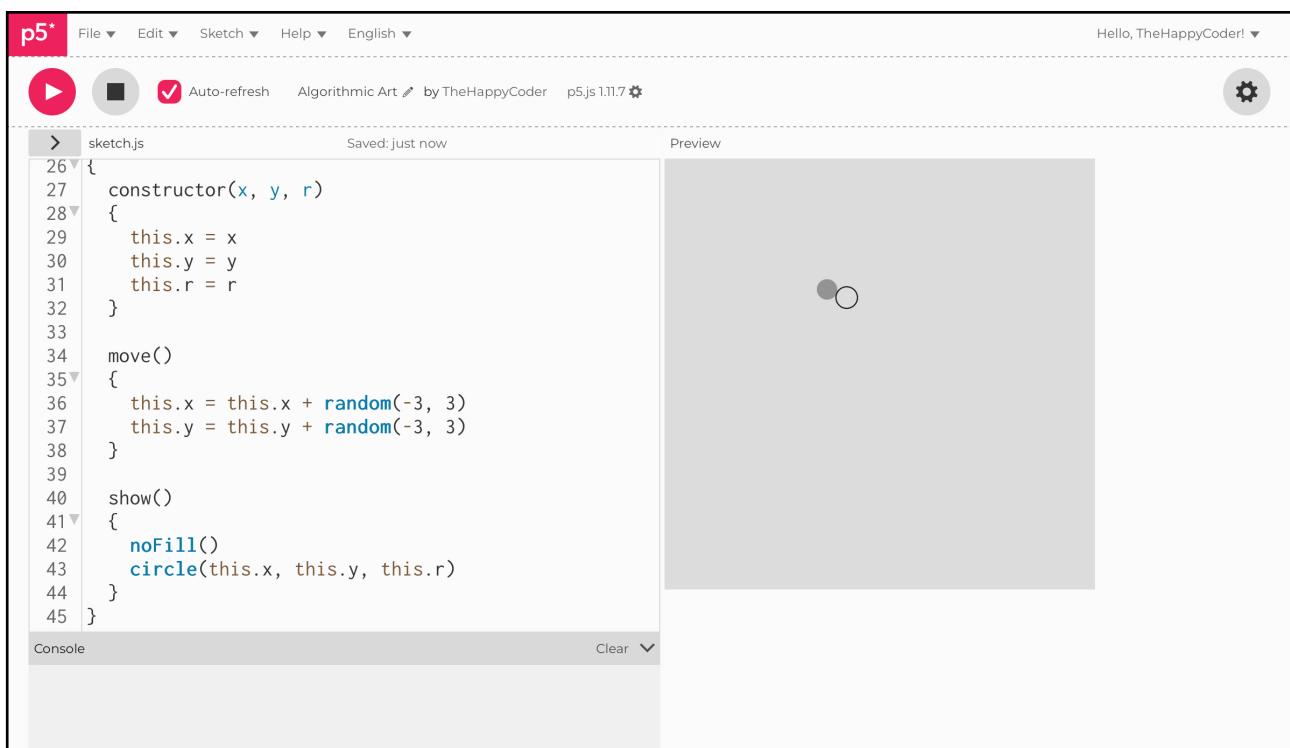
class Bubble
{
    constructor(x, y, r)
```

```
{  
    this.x = x  
    this.y = y  
    this.r = r  
}  
  
move()  
{  
    this.x = this.x + random(-3, 3)  
    this.y = this.y + random(-3, 3)  
}  
  
show()  
{  
    noFill()  
    circle(this.x, this.y, this.r)  
}  
}
```

## Notes

We create a new **bubble** every time we click on the canvas, but only one at a time. We want to keep each one, not discard the last one.

Figure D3.13



The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, auto-refresh (which is checked), and user information ('Hello, TheHappyCoder!'). Below the toolbar, the file name 'sketch.js' is displayed along with a note 'Saved: just now'. The code area contains the following JavaScript code:

```
> sketch.js
26>
27  constructor(x, y, r)
28  {
29    this.x = x
30    this.y = y
31    this.r = r
32  }
33
34  move()
35  {
36    this.x = this.x + random(-3, 3)
37    this.y = this.y + random(-3, 3)
38  }
39
40  show()
41  {
42    noFill()
43    circle(this.x, this.y, this.r)
44  }
45 }
```

The preview window on the right shows a single gray circle that is slightly offset from its original position, indicating it has moved.



## Sketch D3.14 when push comes to shove

We use a `push()` function to push each **bubble** into the array. Now we will keep all the **bubbles** we create with the mouse.

```
let bubbles = []
let bubble

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

function mousePressed()
{
  bubble = new Bubble(mouseX, mouseY, 20)
  bubbles.push(bubble)
}

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

class Bubble
{
  constructor(x, y, r)
```

```
{  
    this.x = x  
    this.y = y  
    this.r = r  
}  
  
move()  
{  
    this.x = this.x + random(-3, 3)  
    this.y = this.y + random(-3, 3)  
}  
  
show()  
{  
    noFill()  
    circle(this.x, this.y, this.r)  
}  
}
```

## Notes

Now when you click on the canvas, each bubble remains on the canvas because they have been added to the array of bubbles.

Figure D3.14

The screenshot shows the p5.js code editor interface. At the top, there are buttons for play/pause, refresh, and settings, along with the text "Hello, TheHappyCoder! ▾". The code editor window has tabs for "sketch.js" and "Preview". The "sketch.js" tab contains the following JavaScript code:

```
26> {
27  constructor(x, y, r)
28  {
29    this.x = x
30    this.y = y
31    this.r = r
32  }
33
34  move()
35  {
36    this.x = this.x + random(-3, 3)
37    this.y = this.y + random(-3, 3)
38  }
39
40  show()
41  {
42    noFill()
43    circle(this.x, this.y, this.r)
44  }
45 }
```

The "Preview" window shows a gray canvas with several white circles of varying sizes scattered across it. One large dark gray circle is located in the upper right quadrant, and several smaller white circles are scattered around it.



## Sketch D3.15 a bit of a drag

Finally, using `mouseDragged()` rather than just `mousePressed()`, we can make lots of **bubbles** as we click and drag the mouse.

```
let bubbles = []
let bubble

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

function mouseDragged()
{
    bubble = new Bubble(mouseX, mouseY, 20)
    bubbles.push(bubble)
}

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

class Bubble
{
    constructor(x, y, r)
```

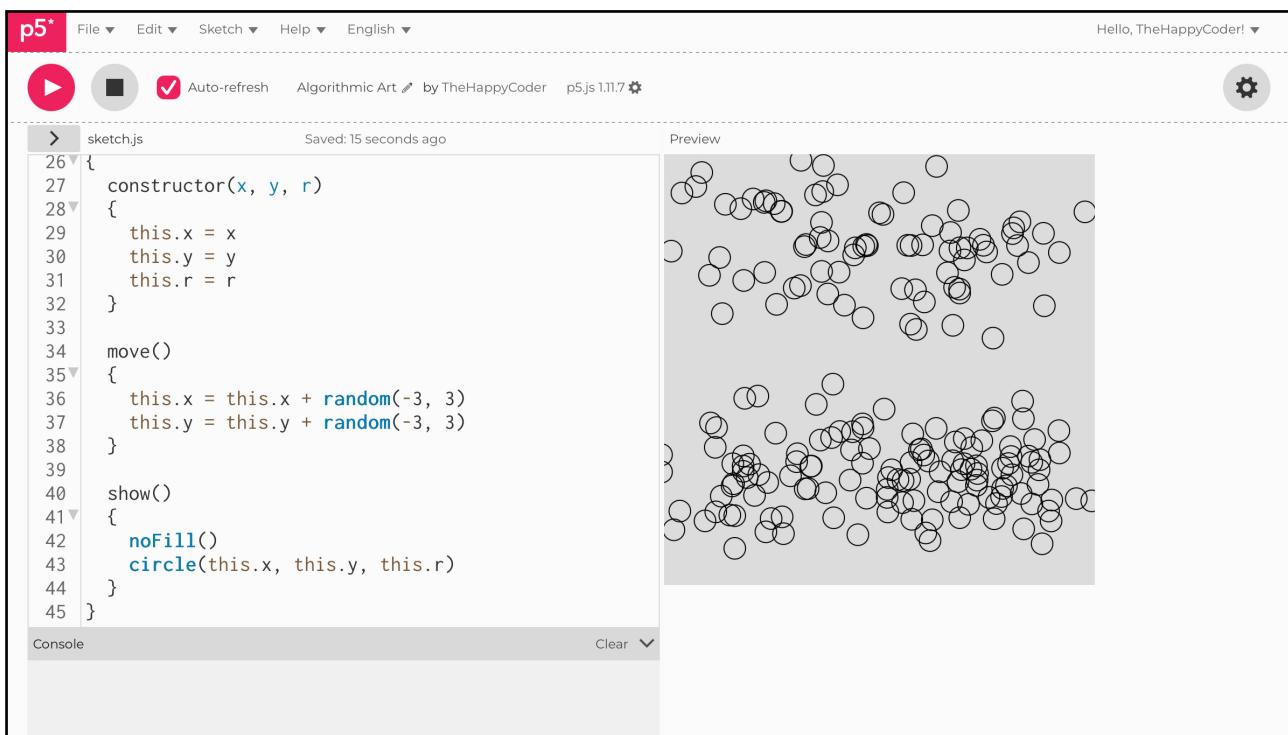
```
{  
    this.x = x  
    this.y = y  
    this.r = r  
}  
  
move()  
{  
    this.x = this.x + random(-3, 3)  
    this.y = this.y + random(-3, 3)  
}  
  
show()  
{  
    noFill()  
    circle(this.x, this.y, this.r)  
}  
}
```



## Challenges

1. What could you create with a few tweaks?
2. How would you create random shapes?

Figure D3.15



The screenshot shows the p5.js code editor interface. At the top, there are navigation menus: File ▾, Edit ▾, Sketch ▾, Help ▾, English ▾, and a user profile Hello, TheHappyCoder! ▾. Below the menu bar are standard controls: a play button, a stop button, an auto-refresh checkbox (which is checked), and the text Algorithmic Art by TheHappyCoder p5.js 1.11.7. On the right side of the interface is a preview window showing a gray background with numerous small, semi-transparent overlapping circles of varying sizes.

```
> sketch.js
Saved: 15 seconds ago
Preview

26
27 constructor(x, y, r)
28 {
29   this.x = x
30   this.y = y
31   this.r = r
32 }
33
34 move()
35 {
36   this.x = this.x + random(-3, 3)
37   this.y = this.y + random(-3, 3)
38 }
39
40 show()
41 {
42   noFill()
43   circle(this.x, this.y, this.r)
44 }
45 }
```

Console Clear ▾