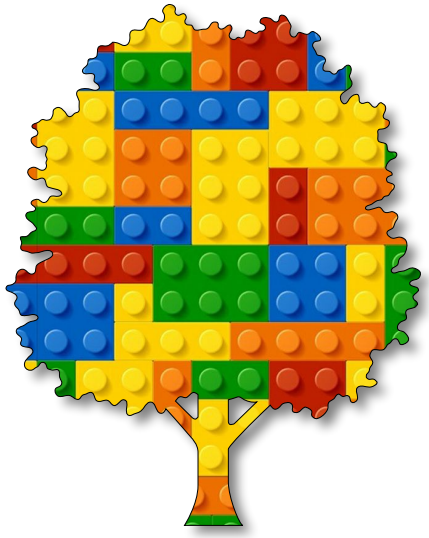
Algorithmic Art

Module D

Unit #5

the overlap



Content

Module D Unit #5 the overlap

Introduction to the overlap

We will cover more aspects of arrays where we can add and remove elements from an array. Keep hold of what you did in the last unit as we will use it again and build on it.

Sketch D5.1 removing some code

! using the sketch from the previous unit.
We want to change the colour of the bubbles if they overlap each other rather than when we move the mouse. We start by deleting the following lines of code (// in blue).

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 20)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    // if (bubbles[i].rollover(mouseX, mouseY))
    // {
    //   bubbles[i].changeColour(255)
    // }
    // else
    // {
    //   bubbles[i].changeColour(0)
    // }
    bubbles[i].show()
    bubbles[i].move()
  }
  // if (bubbles.length > 50)
```

```
  // {
  //   bubbles.splice(0, 1)
  // }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
    this.brightness = 0
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
  }

  changeColour(bright)
  {
    this.brightness = bright
  }

//   rollover(px, py)
//   {
//     let d = dist(px, py, this.x, this.y)
//     if (d < this.r/2)
//     {
//       return true
```

```
//     }
//     else
//     {
//       return false
//     }
//   }
}
```

🗒️ Notes

We want the bare bones.

Sketch D5.2 this is what is left

This is what we have left.

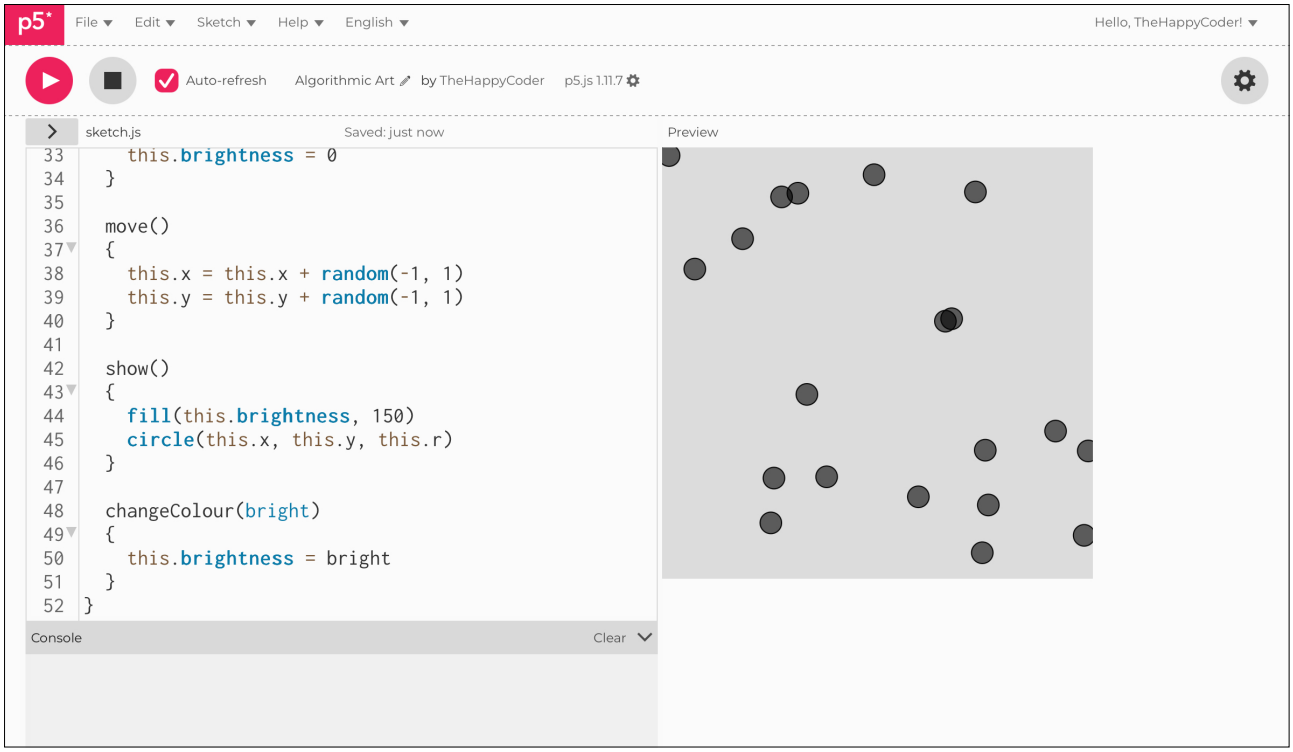
```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 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
    this.brightness = 0
  }
```

```
  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
  }

  changeColour(bright)
  {
    this.brightness = bright
  }
}
```

Notes

We get our 20 wandering bubbles.

Figure D5.2



Sketch D5.3 the intersection

This is where it gets tricky. We want to check if a particular bubble in the array is overlapping with another bubble (but not itself). Before we do that bit, we need to have a function that checks for the intersection of the two bubbles, for instance, whether the distance between the centres is less than the addition of the two radii. If you draw two circles touching, their distance centre to centre is equal to the radius of one plus the radius of the other.

We have two bubbles, one bubble is `this.___` and the other bubble is `other.___`.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 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
    this.brightness = 0
  }

  intersects(other)
  {
    let d = dist(this.x, this.y, other.x, other.y)
    return d < this.r + other.r
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
  }

  changeColour(bright)
  {
    this.brightness = bright
  }
}
```

🗒️ Notes

We create another function in the Bubbles class called `intersect()` and we have an argument called `other` (which we will come to shortly). We return the distance `d` only if it is less than the two radii. That means they are overlapping or intersecting. We haven't created the `other` array just yet, but we will do so next.

🛠️ Code Explanation

`return d < this.r + other.r`	We only want d if it less than than both radii combined

Sketch D5.4 the other bubble

We create another array called `other`.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 20)
    bubbles.push(bubble)
  }
}

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

    }
  }
}

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

```
    this.y = y
    this.r = r
    this.brightness = 0
  }

  intersects(other)
  {
    let d = dist(this.x, this.y, other.x, other.y)
    return d < this.r + other.r
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
  }

  changeColour(bright)
  {
    this.brightness = bright
  }
}
```

🗒️ Notes

The `let...of` allows us to effectively copy the `bubbles` array and call it `other`.

Sketch D5.5 not the same

We need to check that we are not comparing the same bubble.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 20)
    bubbles.push(bubble)
  }
}

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


      }
    }
  }
}

class Bubble
{
```

```
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
    this.brightness = 0
  }

  intersects(other)
  {
    let d = dist(this.x, this.y, other.x, other.y)
    return d < this.r + other.r
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
  }

  changeColour(bright)
  {
    this.brightness = bright
  }
}
```

🗒️ Notes

This is a neat bit of code that compares any element from the `other` array with any element from the `bubbles` array. It uses the `!==` (not the same) comparison.

🛠️ Code Explanation

`if (bubbles[i] !== other)`	Cycles through all the elements in the bubbles array looking for the ones that are NOT in the other array

Sketch D5.6 do we have intersection?

We also need to know if it intersects. So now we have the full condition: it cannot be comparing itself to itself (`bubble !== other`) and it also must (`&&`) intersect.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 20)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
    for (let other of bubbles)
    {
      if (bubbles[i] !== other && bubbles[i].intersects(other))
      {

      }
    }
  }
}

class Bubble
```

```
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
    this.brightness = 0
  }

  intersects(other)
  {
    let d = dist(this.x, this.y, other.x, other.y)
    return d < this.r + other.r
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
  }

  changeColour(bright)
  {
    this.brightness = bright
  }
}
```

🗒️ Notes

This may seem very confusing, but it is extremely logical. Work through it slowly, reading it out loud as if it is a sentence; sometimes it makes more sense that way.

Sketch D5.7 false by default

We are now going to add a boolean expression for overlapping. By default, the boolean will be called `false` and only change that to `true` when it isn't the same AND it intersects.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 20)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
    let overlapping = false
    for (let other of bubbles)
    {
      if (bubbles[i] !== other && bubbles[i].intersects(other))
      {
        overlapping = true
      }
    }
  }
}
```

```
class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
    this.brightness = 0
  }

  intersects(other)
  {
    let d = dist(this.x, this.y, other.x, other.y)
    return d < this.r + other.r
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
  }

  changeColour(bright)
  {
    this.brightness = bright
  }
}
```

🗒️ Notes

A boolean variable is just either `true` or `false`. It is either overlapping or it is not; we start off `false` (not overlapping), until it is, then it is `true`.

Sketch D5.8 use the info

We need to do something with that information. Change the colour.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 20)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
    let overlapping = false
    for (let other of bubbles)
    {
      if (bubbles[i] !== other && bubbles[i].intersects(other))
      {
        overlapping = true
      }
    }
    if (overlapping)
    {
      bubbles[i].changeColour(255)
    }
```

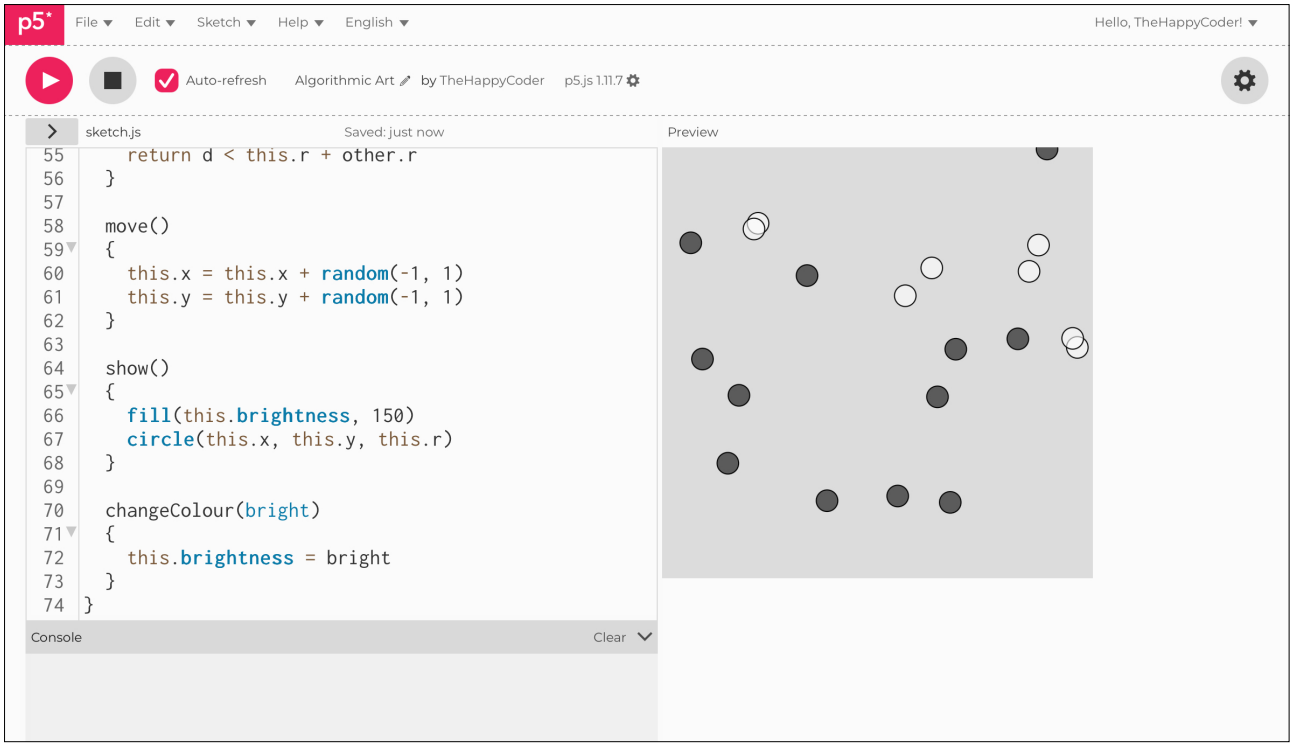
```
    else
    {
      bubbles[i].changeColour(0)
    }
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
    this.brightness = 0
  }

  intersects(other)
  {
    let d = dist(this.x, this.y, other.x, other.y)
    return d < this.r + other.r
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r)
  }

  changeColour(bright)
  {
    this.brightness = bright
```

```
  }
}
```

Notes

If overlapping is `true`, then colour it white (`255`); otherwise, it is `false` and so colour it black (`0`). However, if you look closely, there is something wrong; they aren't overlapping but are still changing. This is because we have set the radius to `20`, but when we draw the circle, it is the diameter.

Figure D5.8



Sketch D5.9 simple change

Making a very simple change.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    bubble = new Bubble(x, y, 20)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
    let overlapping = false
    for (let other of bubbles)
    {
      if (bubbles[i] !== other && bubbles[i].intersects(other))
      {
        overlapping = true
      }
    }
    if (overlapping)
    {
      bubbles[i].changeColour(255)
    }
```

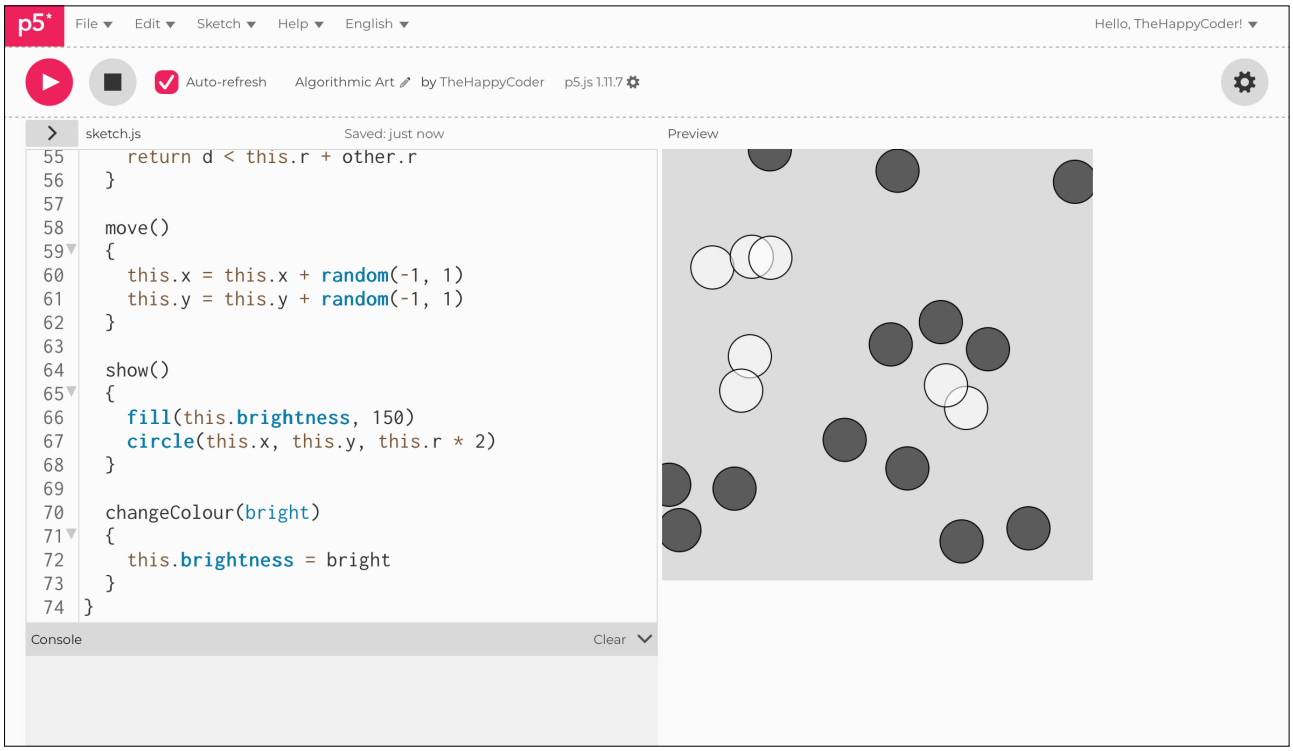
```
    else
    {
      bubbles[i].changeColour(0)
    }
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
    this.brightness = 0
  }

  intersects(other)
  {
    let d = dist(this.x, this.y, other.x, other.y)
    return d < this.r + other.r
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r * 2)
  }

  changeColour(bright)
  {
    this.brightness = bright
```

```
  }
}
```

Notes

Double the radius; there is a reason for doing this.

Figure D5.9



Sketch D5.10 random sizes

The reason for doing it that way is so that you could have randomly sized bubbles.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    let x = random(width)
    let y = random(height)
    let r = random(10, 30)
    bubble = new Bubble(x, y, r)
    bubbles.push(bubble)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
    let overlapping = false
    for (let other of bubbles)
    {
      if (bubbles[i] !== other && bubbles[i].intersects(other))
      {
        overlapping = true
      }
    }
    if (overlapping)
    {
      bubbles[i].changeColour(255)
```

```
    }
    else
    {
      bubbles[i].changeColour(0)
    }
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
    this.brightness = 0
  }

  intersects(other)
  {
    let d = dist(this.x, this.y, other.x, other.y)
    return d < this.r + other.r
  }

  move()
  {
    this.x = this.x + random(-1, 1)
    this.y = this.y + random(-1, 1)
  }

  show()
  {
    fill(this.brightness, 150)
    circle(this.x, this.y, this.r * 2)
  }

  changeColour(bright)
  {
```

```
    this.brightness = bright
  }
}
```

🗒️ Notes

Now it works for any size of radius for any bubble.

Figure D5.10