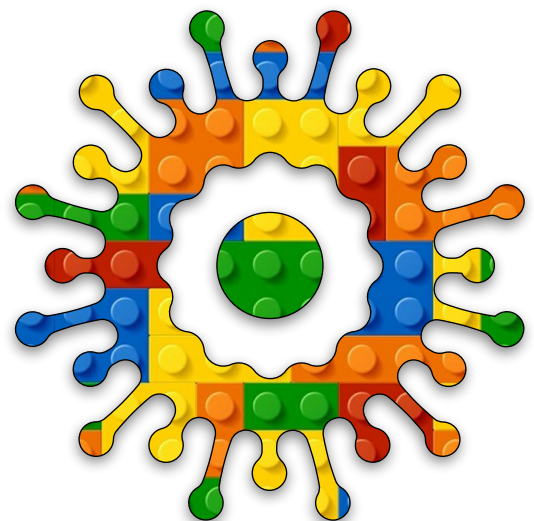


Algorithmic Art

Module A

Unit #3

smoke and
fire





Section A Unit #3 smoke and fire

Sketch A3.1	starting sketch
Sketch A3.2	adding a particle class
Sketch A3.3	adding move and show
Sketch A3.4	draw a circle
Sketch A3.5	particle p
Sketch A3.6	an array of particles
Sketch A3.7	move the particle
Sketch A3.8	many particles
Sketch A3.9	no smoke without fire
Sketch A3.10	too many particles
Sketch A3.11	remove dead particles
Sketch A3.12	backwards array
Sketch A3.13	adding more particles
Sketch A3.14	some adaptations



Introduction to smoke and fire

Creating a simple smoke or fire effect using a simple particle system with p5.js.



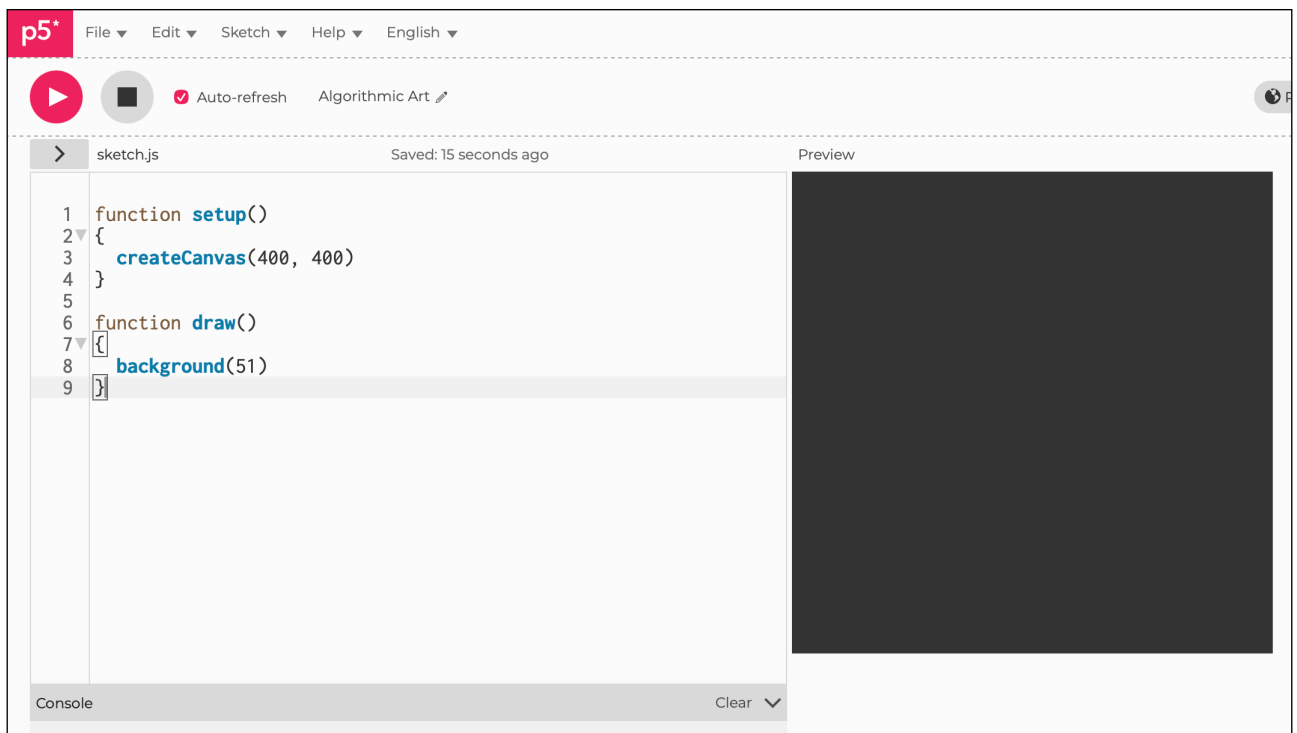
Sketch A3.1 starting sketch

This is our basic sketch; this gives us a nice dark background for a change.

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

function draw()
{
  background(51)
}
```

Figure A3.1





Sketch A3.2 adding a particle class

We are going to add a particle class with a constructor function. We want the particles to start near the bottom and in the middle, then travel up and fade.

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

```
function draw()
{
  background(51)
}
```

```
class Particle
{
  constructor()
  {
    this.x = 200
    this.y = 380
  }
}
```



Notes

Nothing has changed.



Sketch A3.3 adding move and show

We want a `show()` function and a `move()` function.

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

```
function draw()
{
  background(51)
}
```

```
class Particle
{
  constructor()
  {
    this.x = 200
    this.y = 380
  }
```

```
  show()
  {

  }
```

```
  move()
  {

  }
```

```
}
```



Sketch A3.4 draw a circle

We draw a circle at these **x** and **y** co-ordinates filling it with white (**255**).

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

function draw()
{
  background(51)
}

class Particle
{
  constructor()
  {
    this.x = 200
    this.y = 380
  }

  show()
  {
    stroke(255)
    fill(255, 10)
    circle(this.x, this.y, 16)
  }

  move()
  {
  }
}
```

}



Sketch A3.5 particle p

Let's create one particle called **p** so we can see it.

```
let p

function setup()
{
  createCanvas(400, 400)
  p = new Particle()
}

function draw()
{
  background(51)
  p.show()
}

class Particle
{
  constructor()
  {
    this.x = 200
    this.y = 380
  }

  show()
  {
    stroke(255)
    fill(255, 10)
    circle(this.x, this.y, 16)
  }
}
```

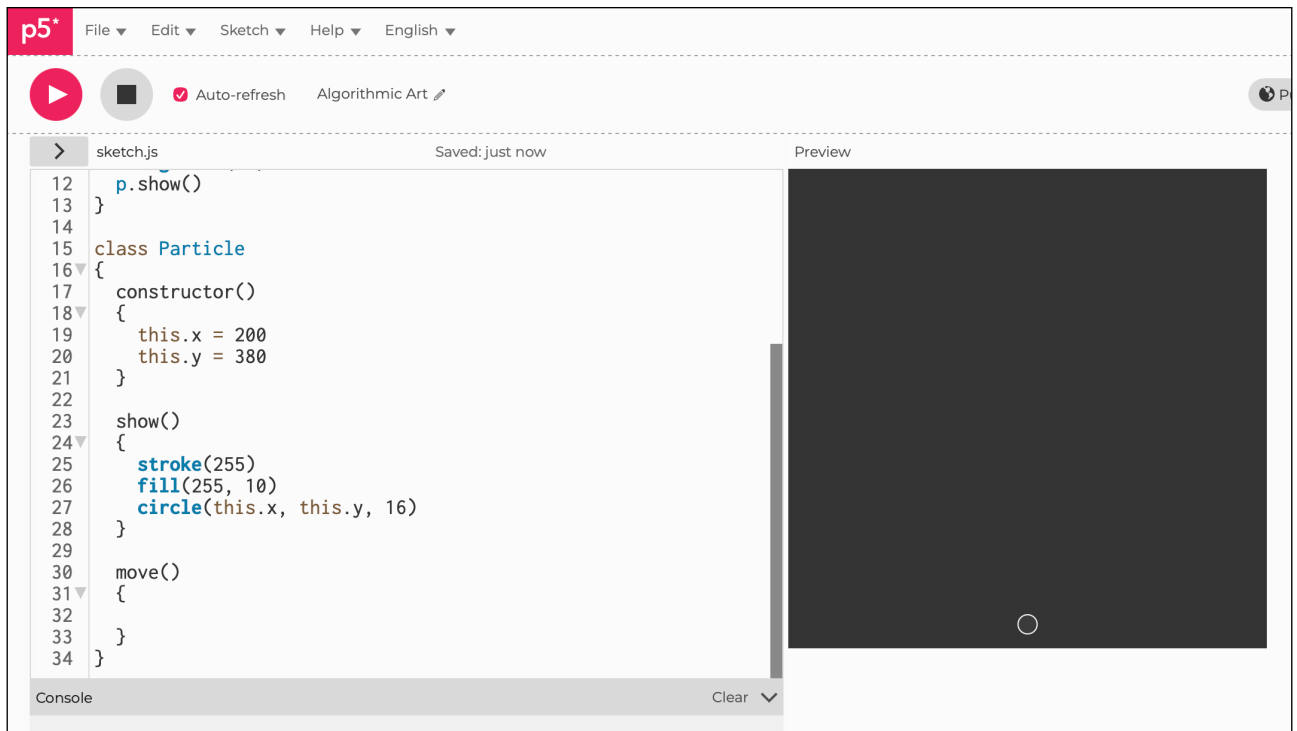
```
move()  
{  
  
}  
}
```



Notes

At last, we have something to see!

Figure A3.5





Sketch A3.6 an array of particles

We want to create many particles, so we create an array and `push()` them into the array using a `for()` loop. We will start with just one particle.

```
let p
let particles = []

function setup()
{
  createCanvas(400, 400)
  p = new Particle()
  particles.push(p)
}

function draw()
{
  background(51)
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].show()
  }
}

class Particle
{
  constructor()
  {
    this.x = 200
    this.y = 380
  }
}
```

```
show()  
{  
  stroke(255)  
  fill(255, 10)  
  circle(this.x, this.y, 16)  
}  
  
move()  
{  
  
}  
}
```



Notes

You should have exactly the same as before.



Sketch A3.7 move the particle

Let's give it some movement, velocity. We create two variables called **velX** and **velY**.

```
let p
let particles = []

function setup()
{
  createCanvas(400, 400)
  p = new Particle()
  particles.push(p)
}

function draw()
{
  background(51)
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].move()
    particles[i].show()
  }
}

class Particle
{
  constructor()
  {
    this.x = 200
    this.y = 380
    this.velX = random(-1, 1)
    this.velY = random(-5, -1)
```

```
}

show()
{
  stroke(255)
  fill(255, 10)
  circle(this.x, this.y, 16)
}

move()
{
  this.x += this.velX
  this.y += this.velY
}
}
```



Notes

The particle (circle) will drift upwards. Every time you replay the sketch, the particle will have a different (random) velocity.

The `+=` symbol is shorthand for:

```
this.x = this.x + this.velX
this.y = this.y + this.velY
```



Sketch A3.8 many particles

We want more than one particle. To do this quickly, we cut and paste the following code from `setup()` into `draw()`.

! you delete those lines of code in `setup()`.

```
let p
let particles = []

function setup()
{
  createCanvas(400, 400)
  // p = new Particle()
  // particles.push(p)
}

function draw()
{
  background(51)
  p = new Particle()
  particles.push(p)
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].movement()
    particles[i].show()
  }
}

class Particle
{
  constructor()
  {
```

```
this.x = 200
this.y = 380
this.velX = random(-1, 1)
this.velY = random(-5, -1)
}

show()
{
  stroke(255)
  fill(255, 10)
  circle(this.x, this.y, 16)
}

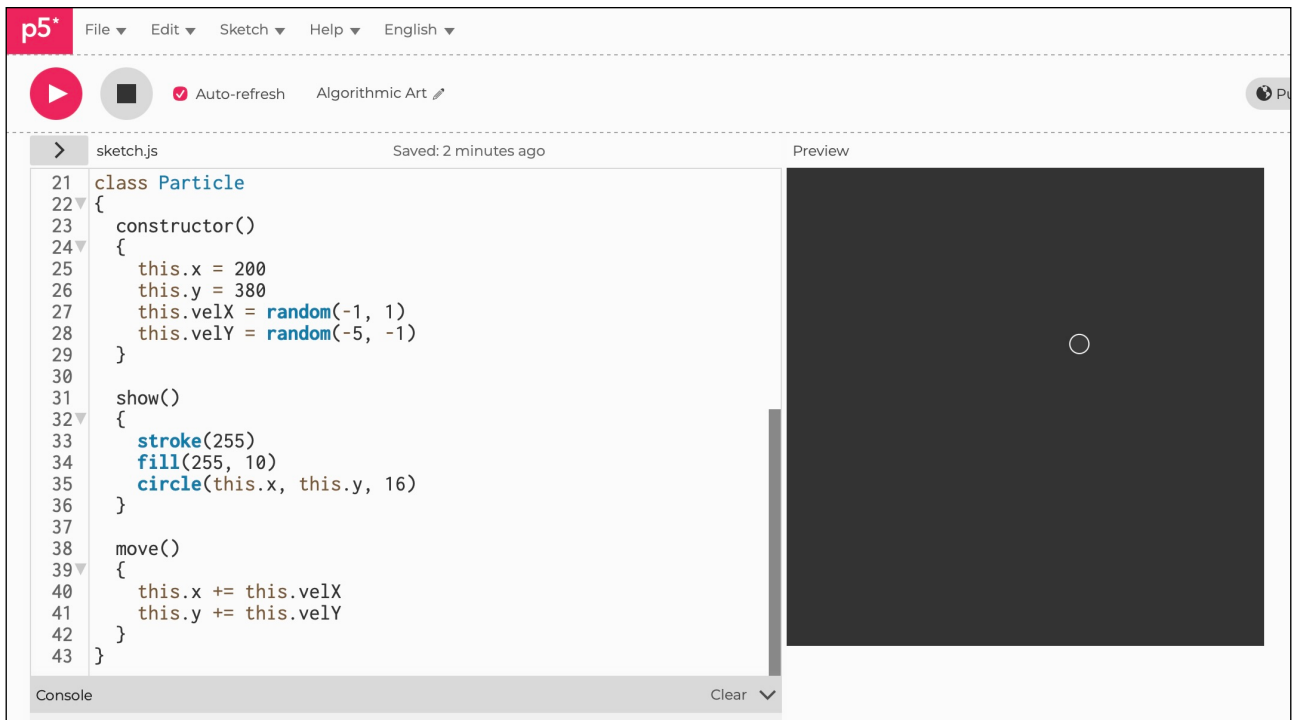
move()
{
  this.x += this.velX
  this.y += this.velY
}
}
```



Notes

You get a stream of particles.

Figure A3.8





Sketch A3.9 no smoke without fire

This is a pleasing effect but still not like smoke or fire. So we will introduce some **alpha**. This is so we can fade it and remove it altogether when it has disappeared; otherwise, the programme will run slower and slower as we create more and more articles.

! We replace **stroke(255)** with **noStroke()**.

```
let p
let particles = []

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

function draw()
{
  background(51)
  p = new Particle()
  particles.push(p)
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].movement()
    particles[i].show()
  }
}

class Particle
{
  constructor()
  {
    this.x = 200
```

```
this.y = 380
this.velX = random(-1, 1)
this.velY = random(-5, -1)
this.alpha = 255
}

show()
{
  noStroke()
  fill(255, this.alpha)
  circle(this.x, this.y, 16)
}

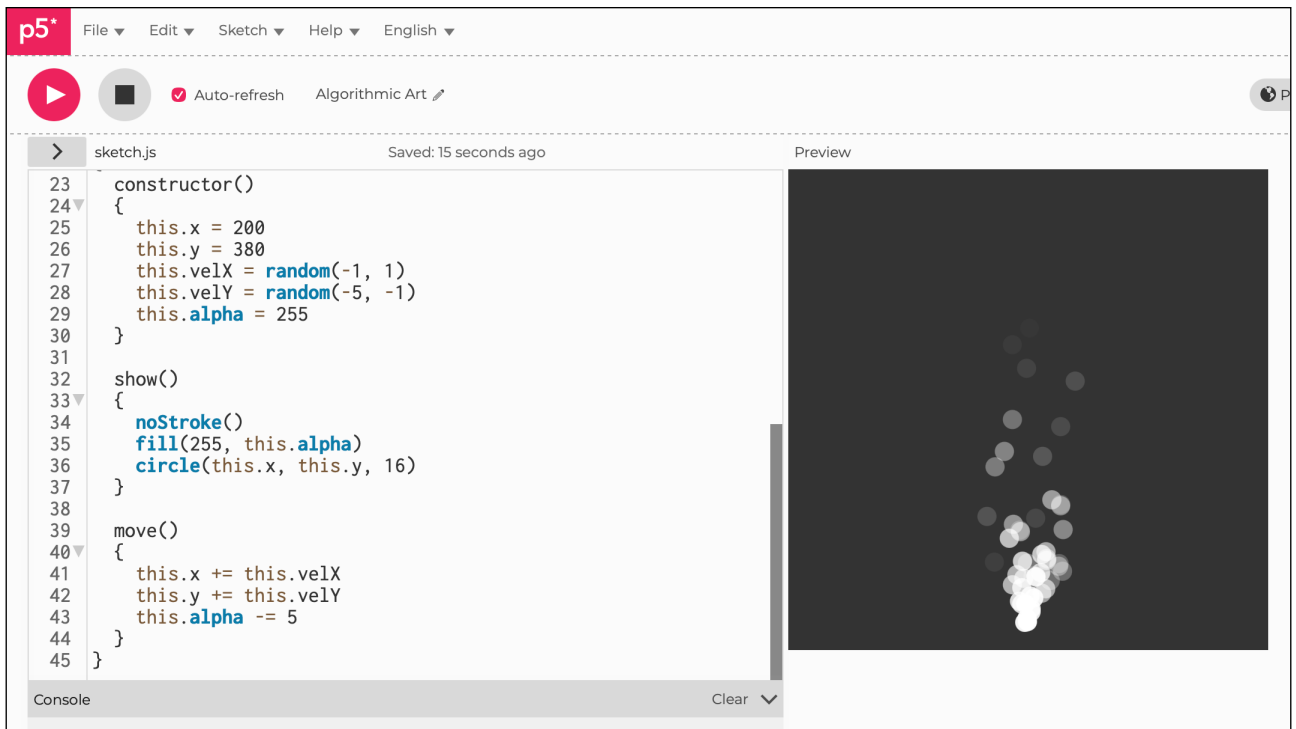
move()
{
  this.x += this.velX
  this.y += this.velY
  this.alpha -= 5
}
}
```



Notes

The line of code: `this.alpha -= 5` means `this.alpha = this.alpha - 5`

Figure A3.9





Sketch A3.10 too many particles

We need to address the issue. If we include a line to see how many particles we have created and still exist, we can use `console.log()` in `draw()`. We can give it the parameter of `particles.length`, and as you let it run for a short while, the number just keeps on going up.

```
let p
let particles = []

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

function draw()
{
  background(51)
  p = new Particle()
  particles.push(p)
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].move()
    particles[i].show()
  }
  console.log(particles.length)
}

class Particle
{
  constructor()
  {
    this.x = 200
```

```
    this.y = 380
    this.velX = random(-1, 1)
    this.velY = random(-5, -1)
    this.alpha = 255
  }

  show()
  {
    noStroke()
    fill(255, this.alpha)
    circle(this.x, this.y, 16)
  }

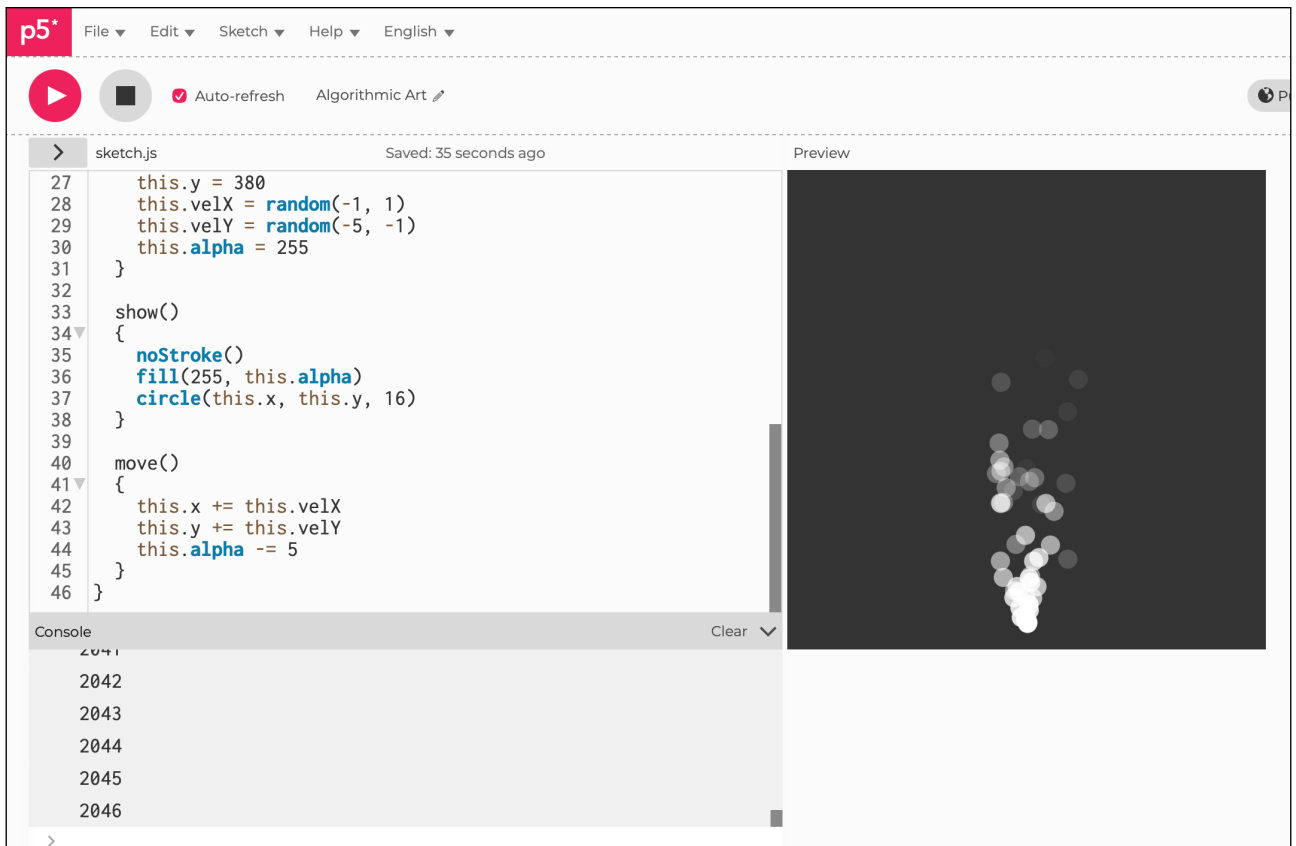
  move()
  {
    this.x += this.velX
    this.y += this.velY
    this.alpha -= 5
  }
}
```



Notes

After running this for a few minutes, you will get the idea. We will keep it there for the moment and delete it later after we have removed the particles that have faded to zero. You will notice that it is fine for a few minutes and then starts to run very slowly.

Figure A3.10





Sketch A3.11 remove dead particles

To remove the particles that are less than zero (**fade**), we use the **splice()** function in a new function that checks to see if it is zero or less. We will call this function **finished()**.

```
let p
let particles = []

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

function draw()
{
  background(51)
  p = new Particle()
  particles.push(p)
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].move()
    particles[i].show()
    if (particles[i].finished())
    {
      particles.splice(i, 1)
    }
  }
  console.log(particles.length)
}

class Particle
{

```

```
constructor()
{
  this.x = 200
  this.y = 380
  this.velX = random(-1, 1)
  this.velY = random(-5, -1)
  this.alpha = 255
}
```

```
show()
{
  noStroke()
  fill(255, this.alpha)
  circle(this.x, this.y, 16)
}
```

```
move()
{
  this.x += this.velX
  this.y += this.velY
  this.alpha -= 5
}
```

```
finished()
{
  return this.alpha < 0
}
```

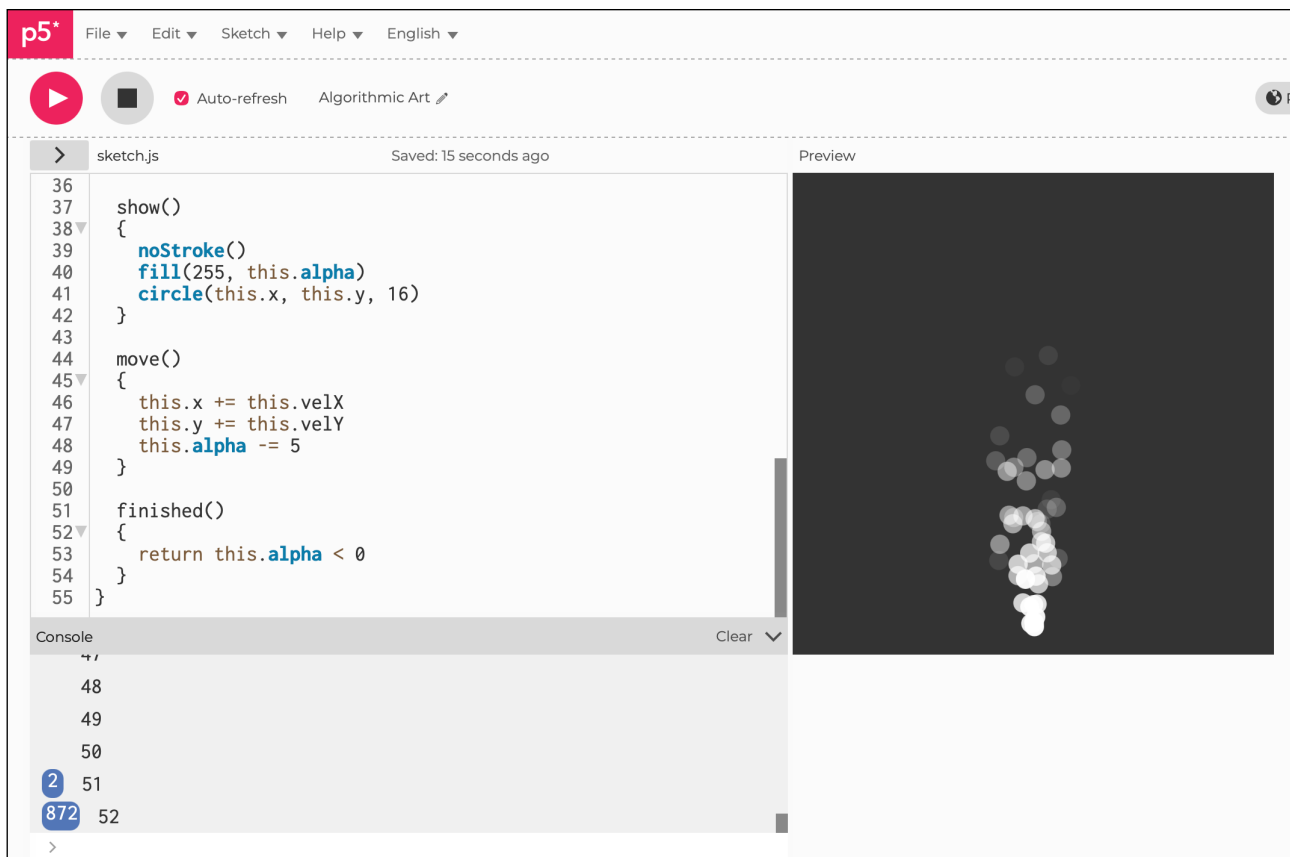
```
}
```



Notes

The code: `return this.alpha < 0` means the function `finished()` will return `true` when the alpha is less than zero. Think boolean (true/false). In my case, the counter stops at `52`.

Figure A3.11





Sketch A3.12 backwards array

Although this looks fine, there is a problem. When you are using `splice()`, we are doing something a bit odd when we delete something from an array counting from the front. This can mean that you jump past the next one after you have deleted a particle.

The way around it is to start from the end and check through the array backwards. We start with the last element in the array, which is at `particles.length - 1` because the numbering system starts from `0` (not `1`), count backwards (`i--`) while `i` is greater than zero (`> 0`). It may take a bit of thinking through, but the logic is very sound. It is a good idea to explore the nature of arrays and how they are spliced.

```
let p
let particles = []

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

function draw()
{
  background(51)
  p = new Particle()
  particles.push(p)
  for (let i = particles.length - 1; i > 0; i--)
  {
    particles[i].move()
    particles[i].show()
    if (particles[i].finished())
    {
      particles.splice(i, 1)
    }
  }
}
```

```
    }  
    console.log(particles.length)  
}
```

```
class Particle
```

```
{  
  constructor()  
  {  
    this.x = 200  
    this.y = 380  
    this.velX = random(-1, 1)  
    this.velY = random(-5, -1)  
    this.alpha = 255  
  }  
  
  show()  
  {  
    noStroke()  
    fill(255, this.alpha)  
    circle(this.x, this.y, 16)  
  }  
  
  move()  
  {  
    this.x += this.velX  
    this.y += this.velY  
    this.alpha -= 5  
  }  
  
  finished()  
  {  
    return this.alpha < 0  
  }  
}
```

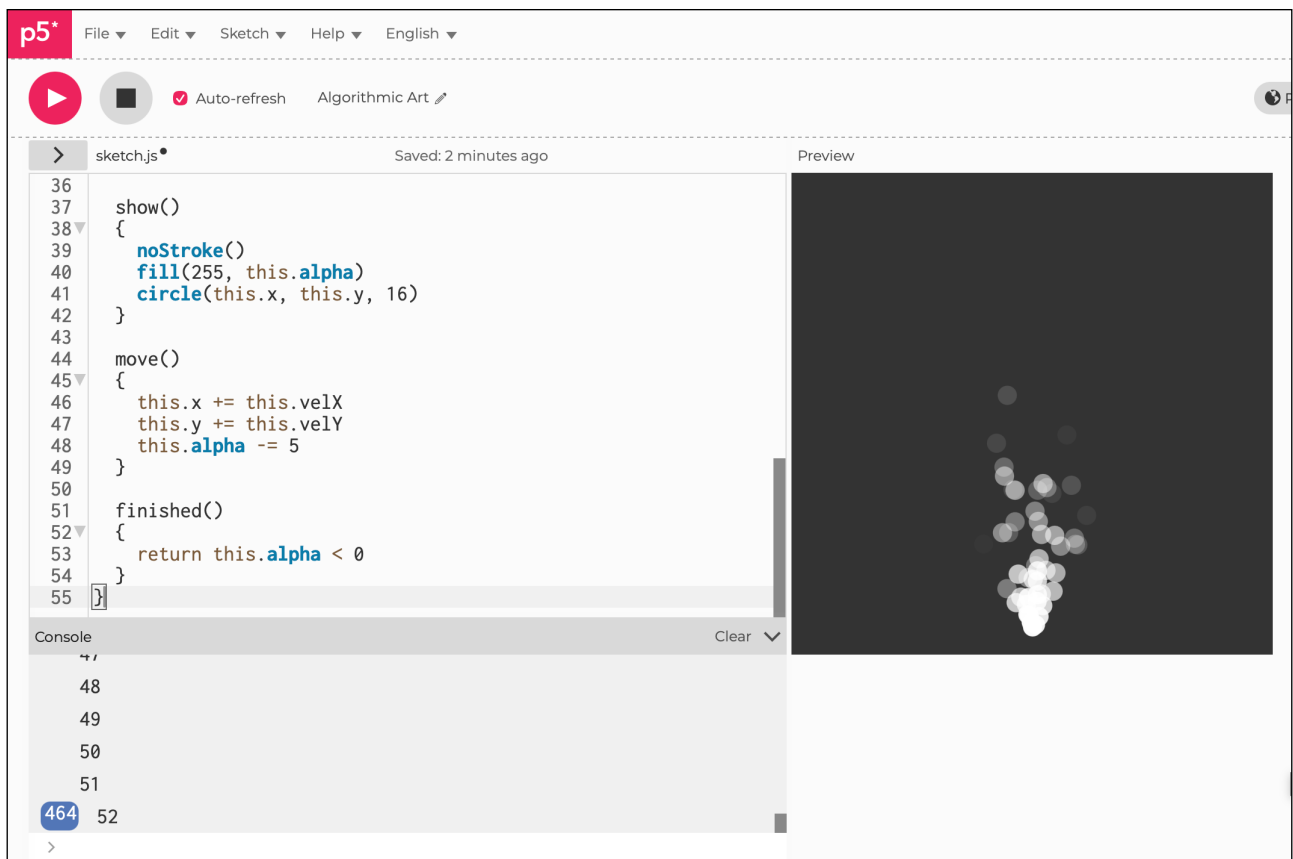
```
}
```



Notes

When you run this with the `console.log()`, you will see that it has the same effect; you may see a slight improvement. It didn't stick at **51**, much cleaner and better coding.

Figure A3.12





Sketch A3.13 adding more particles

We only added **one** particle per frame in **draw()**, now let's add, say, **five** particles per frame.

! remove **console.log()**.

```
let p
let particles = []

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

function draw()
{
  background(51)
  for (let i = 0; i < 5; i++)
  {
    p = new Particle()
    particles.push(p)
  }
  for (let i = particles.length - 1; i > 0; i--)
  {
    particles[i].move()
    particles[i].show()
    if (particles[i].finished())
    {
      particles.splice(i, 1)
    }
  }
  // console.log(particles.length)
```

```

}

class Particle
{
  constructor()
  {
    this.x = 200
    this.y = 380
    this.velX = random(-1, 1)
    this.velY = random(-5, 5)
    this.alpha = 255
  }

  show()
  {
    noStroke()
    fill(255, this.alpha)
    circle(this.x, this.y, 16)
  }

  move()
  {
    this.x += this.velX
    this.y += this.velY
    this.alpha -= 5
  }

  finished()
  {
    return this.alpha < 0
  }
}

```



Notes

It looks so much better.

Figure A3.13





Sketch A3.14 some adaptations

Just some little adjustments.

```
let p
let particles = []

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

function draw()
{
  background(51)
  for (let i = 0; i < 5; i++)
  {
    p = new Particle()
    particles.push(p)
  }
  for (let i = particles.length - 1; i > 0; i--)
  {
    particles[i].move()
    particles[i].show()
    if (particles[i].finished())
    {
      particles.splice(i, 1)
    }
  }
}

class Particle
{
```

```

constructor()
{
  this.x = 200
  this.y = 380
  this.velX = random(-1, 1)
  this.velY = random(-6, -2)
  this.alpha = 255
  this.radius = 20
}

show()
{
  noStroke()
  fill(255, this.alpha)
  circle(this.x, this.y, this.radius)
}

move()
{
  this.x += this.velX
  this.y += this.velY
  this.alpha -= 5
  this.radius -= 0.001
}

finished()
{
  return this.alpha < 0
}
}

```



Notes

Even better.

🌻 Challenges

1. Change the diameter randomly.
2. Add some colour and then fade.

Figure A3.14

