

Algorithmic
Art
Workbook #9
Display of
Fireworks 2



Table of Contents

MIT Licence	3
Workbook #9 Quick Content Summary	4
Module E Unit #4: fireworks continued	6
Sketch E4.15 cleaned up a bit	7
Sketch E4.16 from setup() to draw()	9
Sketch E4.17 not so many	12
Sketch E4.18 random height	15
Sketch E4.19 top of travel	18
Sketch E4.20 null	21
Sketch E4.21 boolean	24
Sketch E4.22 explode() function	27
Sketch E4.23 adding gravity	31
Sketch E4.24 two types of particle	35
Sketch E4.25 true	39
Sketch E4.26 random velocity	43
Sketch E4.27 slow down	47
Sketch E4.28 fade	51
Sketch E4.29 improve fade adjustment	55
Sketch E4.30 increase gravity reduce particles	60
Sketch E4.31 removing spent particles	64
Sketch E4.32 splice the faded	68
Sketch E4.33 it is done()	72
Sketch E4.34 sparks and rockets	76
Sketch E4.35 backwards array	81
Sketch E4.36 trails	86
Sketch E4.37 colour	91
Sketch E4.38 adding the colour in	96
Sketch E4.39 colorMode(RGB)	101



MIT Licence

Copyright ©2026 Warren George

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Brief summary:

Permissions (what you can do)

Commercial use
Modification
Distribution
Private use

Limitations (what is not covered)

Liability
Warranty



Workbook #9 Quick Content Summary

We continue where we left off from the last workbook. We are creating a firework display. This is a long unit, hence why it has spilled over into this workbook. As I mentioned in the previous workbook, when you have completed the unit, save the sketch for the next workbook, where we will be using this example to break up the sketch into more manageable files.



Algorithmic Art

Module E

Unit #4

Firework

Display Part 2



Module E Unit #4: fireworks continued

Keep on going with this but as always there are various concepts and ideas that you may struggle at first but the more you do you over time you will start to see the logic and pattern repeating itself. This is very much a crash course and I will be creating a series on **Physics Simulations** that delve a bit deeper.



Sketch E4.15 cleaned up a bit

Which should look like this now.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)
  fireworks.push(new Firework())
  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, -8)
    this.acc = createVector(0, 0)
  }

  show()
  {
    point(this.pos.x, this.pos.y)
  }
}
```

```
    applyForce(force)
  {
    this.acc.add(force)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }
}

class Firework
{
  constructor()
  {
    this.firework = new Particle(random(width), height)
  }

  show()
  {
    this.firework.show()
  }

  move()
  {
    this.firework.applyForce(gravity)
    this.firework.move()
  }
}
```

Notes

This is just a check you have the same code as me.



Sketch E4.16 from setup() to draw()

If we put the line of code creating an array of fireworks from `setup()` into `draw()`, we get lots of fireworks.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)
  // fireworks.push(new Firework())
  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  fireworks.push(new Firework())
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, -8)
    this.acc = createVector(0, 0)
  }

  show()
  {

```

```

    point(this.pos.x, this.pos.y)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }
}

class Firework
{
  constructor()
  {
    this.firework = new Particle(random(width), height)
  }

  show()
  {
    this.firework.show()
  }

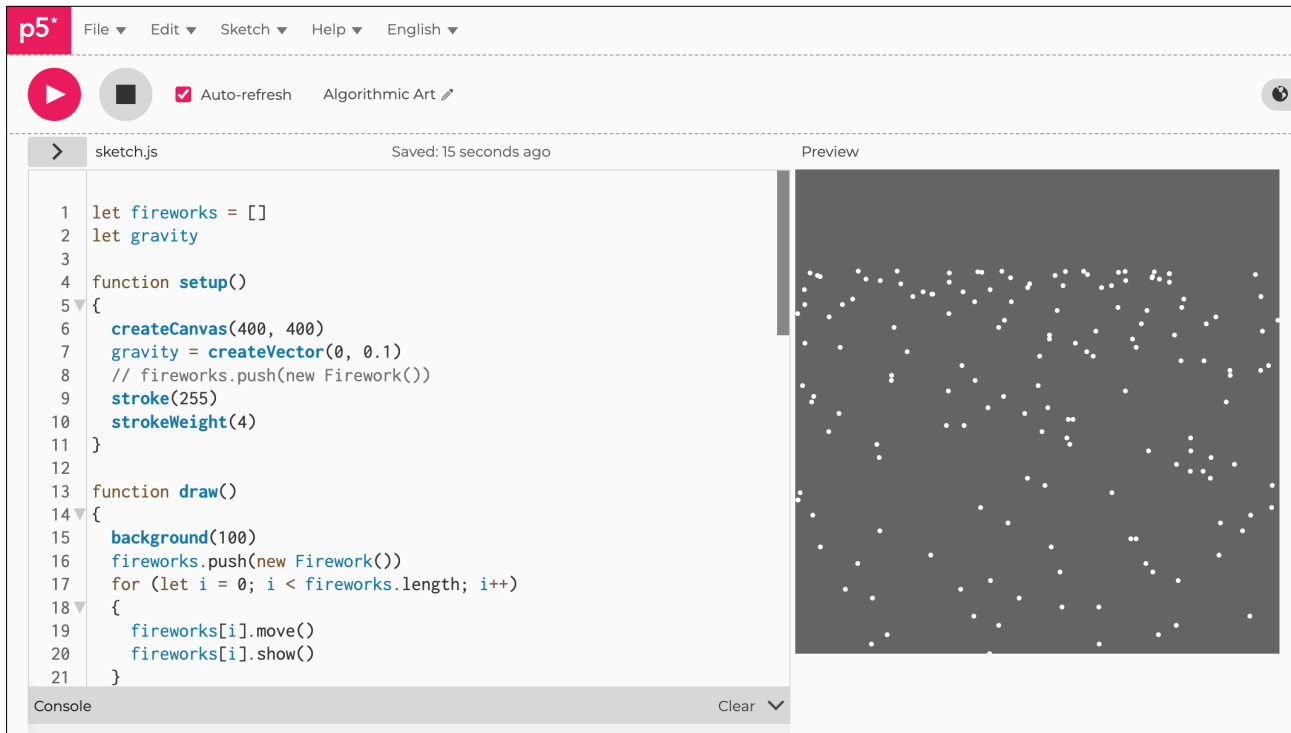
  move()
  {
    this.firework.applyForce(gravity)
    this.firework.move()
  }
}

```

Notes

There are now lots of particles climbing into the sky and falling back down again.

Figure E4.16





Sketch E4.17 not so many

We want more than one but not hundreds, so if we put the creation of a firework in the `draw()` function, we can limit how many it draws by using a random selection. In this case, every frame there is a **10%** chance of creating a firework.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }

  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, -8)
    this.acc = createVector(0, 0)
  }
}
```

```

}

show()
{
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height)
    }

    show()
    {
        this.firework.show()
    }

    move()
    {
        this.firework.applyForce(gravity)
        this.firework.move()
    }
}

```

Notes

A little less manic.

Figure E4.17





Sketch E4.18 random height

Let's randomise the height as well, so in the Particle class we do the following. We can play with these later.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, random(-8, -4))
    this.acc = createVector(0, 0)
  }
}
```

```

    show()
    {
        point(this.pos.x, this.pos.y)
    }

    applyForce(force)
    {
        this.acc.add(force)
    }

    move()
    {
        this.vel.add(this.acc)
        this.pos.add(this.vel)
        this.acc.mult(0)
    }
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height)
    }

    show()
    {
        this.firework.show()
    }

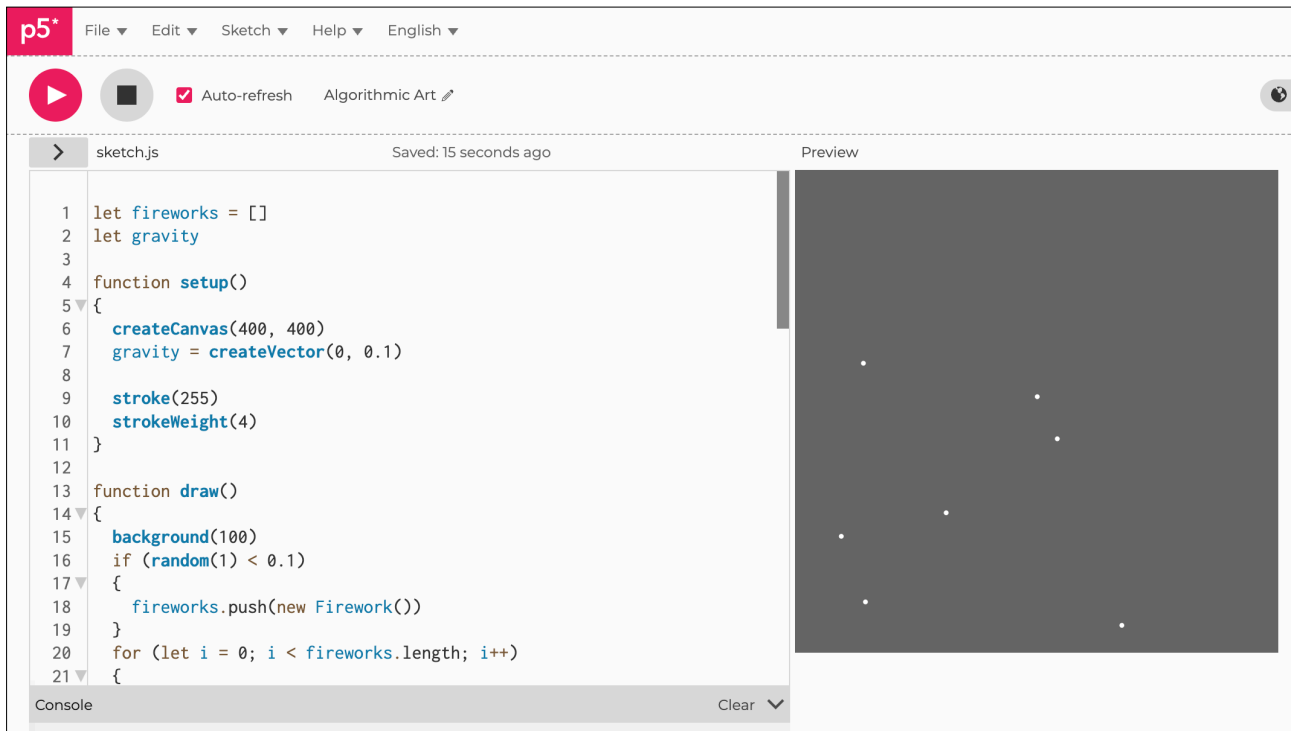
    move()
    {
        this.firework.applyForce(gravity)
        this.firework.move()
    }
}

```

Notes

A bit more random.

Figure E4.18





Sketch E4.19 top of travel

We want to explode the firework at the point it reaches the top of its travel. This is when the velocity is zero. As it goes up, it is negative, becomes zero, and then positive on the way down. First, we only draw the firework if it exists by using the if statement (`this.firework`) in both `move()` and `show()`.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, random(-8, -4))
    this.acc = createVector(0, 0)
  }
}
```

```

}

show()
{
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height)
    }

    show()
    {
        if (this.firework)
        {
            this.firework.show()
        }
    }

    move()
    {
        if (this.firework)

```

```

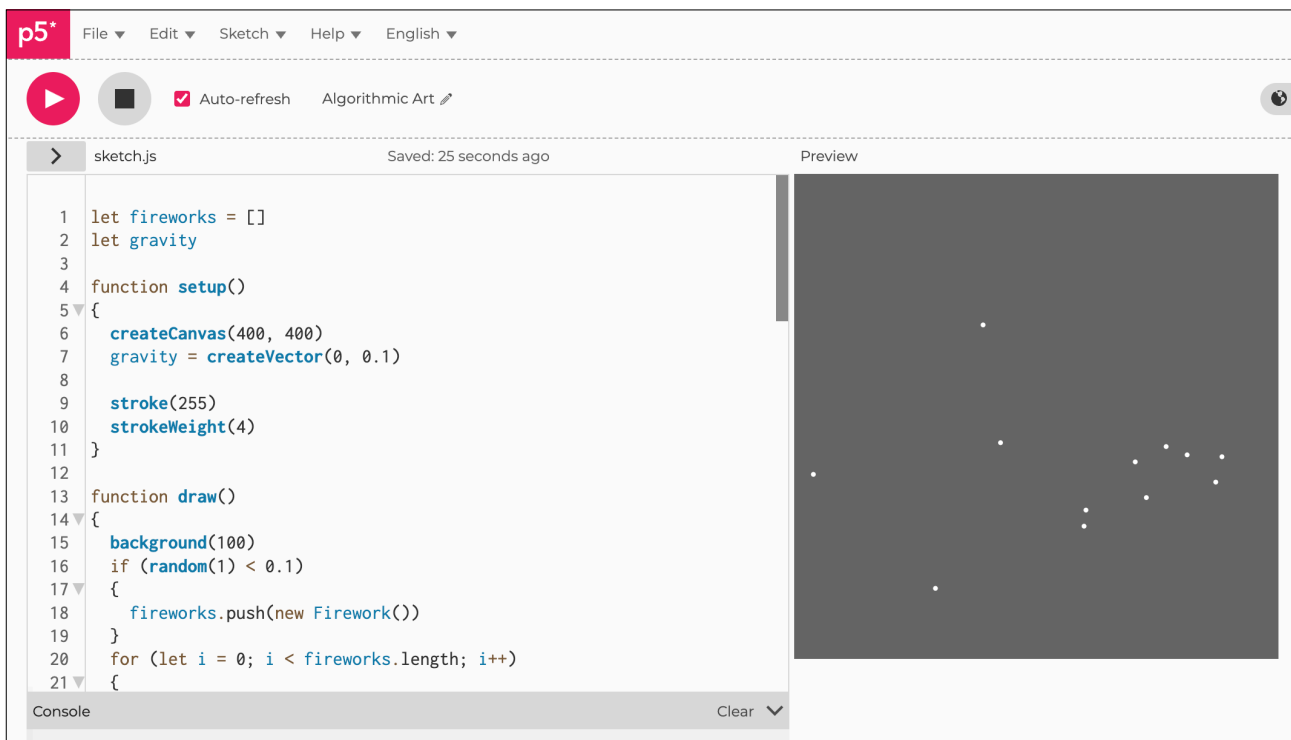
    {
      this.firework.applyForce(gravity)
      this.firework.move()
    }
  }
}

```

Notes

This is pretty much the same.

Figure E4.19





Sketch E4.20 null

Now, to make it null, which will make it disappear. This isn't permanent because we really want it to explode, but we will get to that in a moment.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, random(-8, -4))
    this.acc = createVector(0, 0)
  }
}
```

```

show()
{
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height)
    }

    show()
    {
        if (this.firework)
        {
            this.firework.show()
            if (this.firework.vel.y >= 0)
            {
                this.firework = null
            }
        }
    }
}

```

```
move()
{
  if (this.firework)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.firework = null
    }
  }
}
```

Notes

As it reaches its highest point, it then disappears.



Sketch E4.21 boolean

You get the idea, but we don't want to just remove them; we want to explode them. So we will change what happens at the top of the to a boolean expression. It will have the same effect. We will call the boolean variable **exploded**.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, random(-8, -4))
    this.acc = createVector(0, 0)
```

```

}

show()
{
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height)
        this.exploded = false
    }

    show()
    {
        if (!this.exploded)
        {
            this.firework.show()
            if (this.firework.vel.y >= 0)
            {
                this.exploded = true
            }
        }
    }
}

```

```
}

move()
{
  if (!this.exploded)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.exploded = true
    }
  }
}
}
```

Notes

You could start with it being true, and then you would have to reverse all the logic in the rest of the sketch. It's a preference thing.



Sketch E4.22 explode() function

As we want to have the firework explode, we will create a function called `explode()`. We want it to explode when it reaches the very top (when it is null or true). In this function, we are going to make **100** particles.

! Removed the following lines of code:

```
if (this.firework.vel.y >= 0)
{
  this.exploded = true
}
```

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
```

```

    constructor(x, y)
    {
        this.pos = createVector(x, y)
        this.vel = createVector(0, random(-8, -4))
        this.acc = createVector(0, 0)
    }

    show()
    {
        point(this.pos.x, this.pos.y)
    }

    applyForce(force)
    {
        this.acc.add(force)
    }

    move()
    {
        this.vel.add(this.acc)
        this.pos.add(this.vel)
        this.acc.mult(0)
    }
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height)
        this.exploded = false
        this.particles = []
    }

    show()
    {
        if (!this.exploded)
        {

```

```

    this.firework.show()
  }
  for (let i = 0; i < this.particles.length; i++)
  {
    this.particles[i].show()
  }
}

move()
{
  if (!this.exploded)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.exploded = true
      this.explode()
    }
  }
}

explode()
{
  for (let i = 0; i < 100; i++)
  {
    let p = new Particle(this.firework.pos.x, this.firework.pos.y)
    this.particles.push(p)
  }
}
}

```

Notes

What you see is the particles going up, and then the **100** particles just stay put. We need to apply a force to those particles and move them.

Figure E4.22





Sketch E4.23 adding gravity

Applying gravity to the 100 particles.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y)
  {
    this.pos = createVector(x, y)
    this.vel = createVector(0, random(-8, -4))
    this.acc = createVector(0, 0)
  }
}
```

```

show()
{
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height)
        this.exploded = false
        this.particles = []
    }

    show()
    {
        if (!this.exploded)
        {
            this.firework.show()
        }
        for (let i = 0; i < this.particles.length; i++)
        {
            this.particles[i].show()
        }
    }
}

```

```

move()
{
  if (!this.exploded)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.exploded = true
      this.explode()
    }
  }
}

for (let i = 0; i < this.particles.length; i++)
{
  this.particles[i].applyForce(gravity)
  this.particles[i].move()
}

}

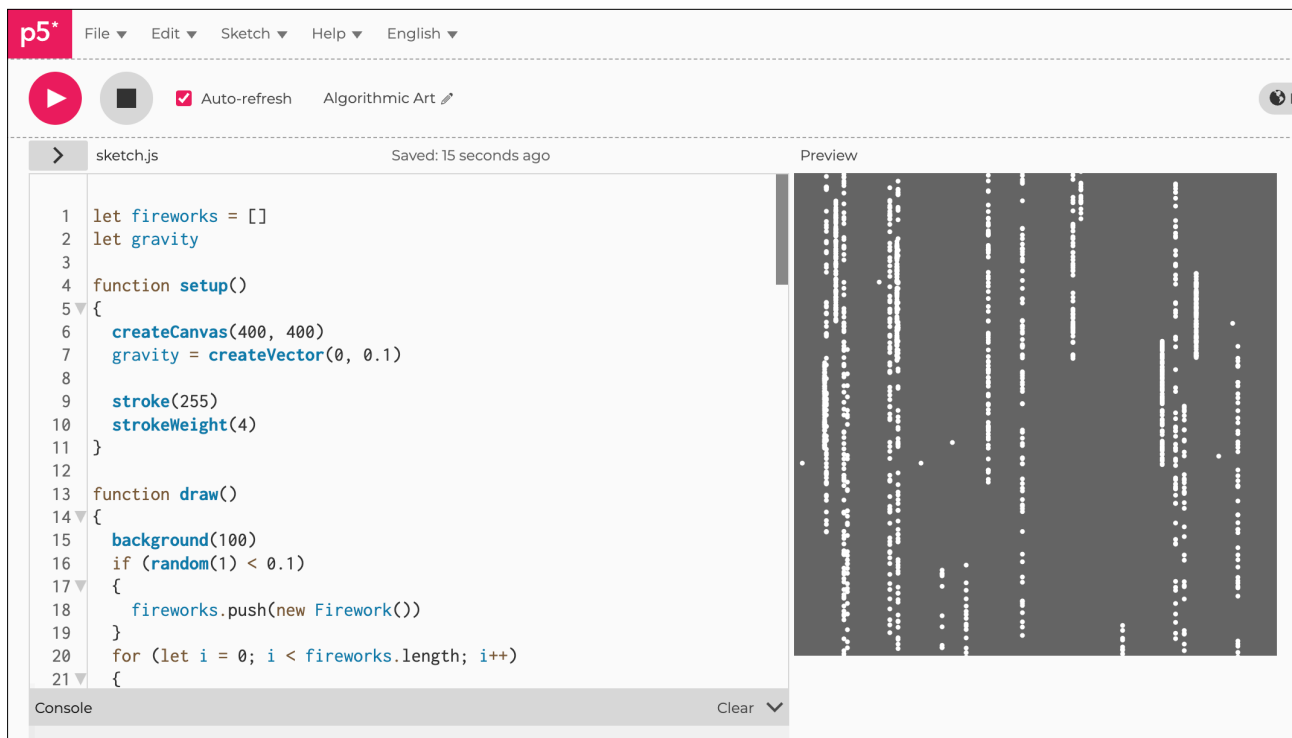
explode()
{
  for (let i = 0; i < 100; i++)
  {
    let p = new Particle(this.firework.pos.x, this.firework.pos.y)
    this.particles.push(p)
  }
}
}

```

Notes

They are exploding but only moving vertically.

Figure E4.23





Sketch E4.24 two types of particle

In the Particle class, we add another argument. This is because we have two types of particle: one for the going-up bit (rocket firework) and the explode bit. This allows us to give different vector values to the two types of particle. `firework` becomes a boolean variable.

We use `p5.Vector.random2D()` so that we have a random vector in both the horizontal and the vertical.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
```

```

    this.pos = createVector(x, y)
    this.firework = firework
    if (firework)
    {
        this.vel = createVector(0, random(-8, -4))
    }
    else
    {
        this.vel = p5.Vector.random2D()
    }

    this.acc = createVector(0, 0)
}

show()
{
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height)
        this.exploded = false
        this.particles = []
    }
}

```

```

}

show()
{
  if (!this.exploded)
  {
    this.firework.show()
  }
  for (let i = 0; i < this.particles.length; i++)
  {
    this.particles[i].show()
  }
}

move()
{
  if (!this.exploded)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.exploded = true
      this.explode()
    }
  }
  for (let i = 0; i < this.particles.length; i++)
  {
    this.particles[i].applyForce(gravity)
    this.particles[i].move()
  }
}

explode()
{
  for (let i = 0; i < 100; i++)
  {
    let p = new Particle(this.firework.pos.x, this.firework.pos.y)
  }
}

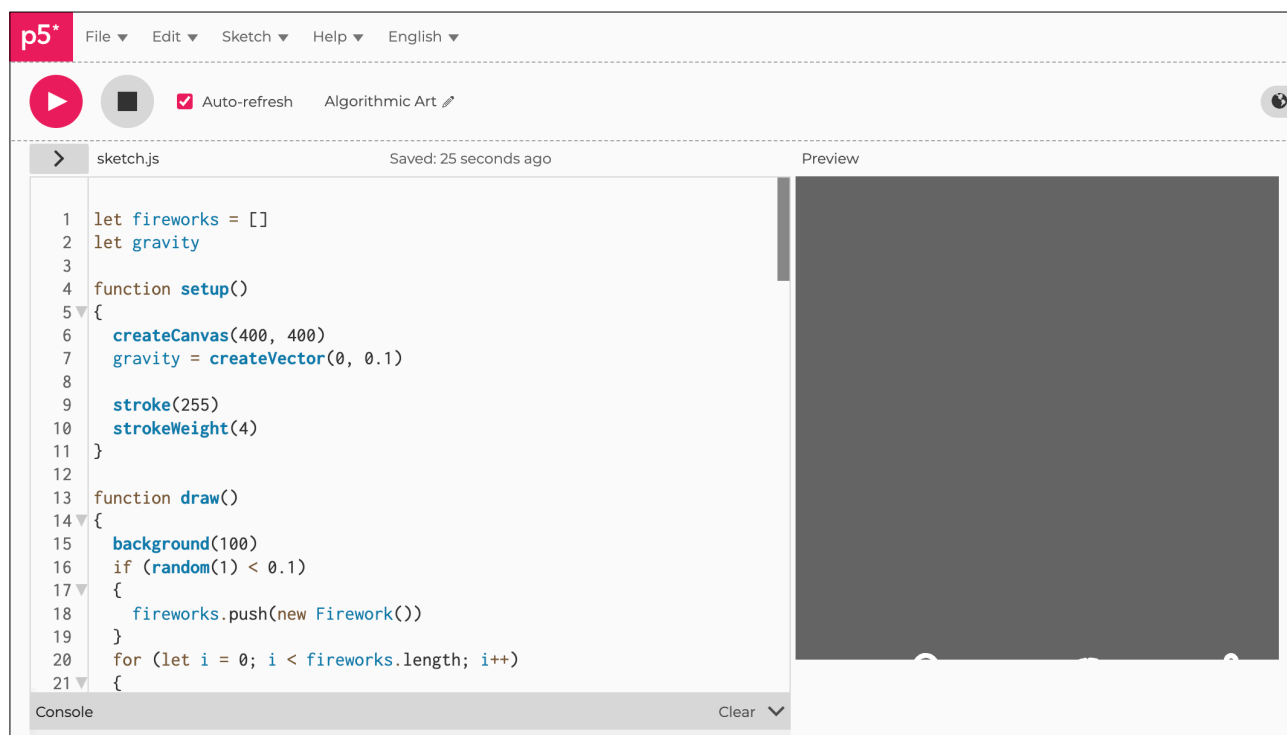
```

```
    this.particles.push(p)
  }
}
}
```

Notes

You get them bubbling at the bottom of the canvas. We will raise them into the air.

Figure E4.24





Sketch E4.25 true

In the Firework class, we need to add the boolean to be true for the particle.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    if (firework)
    {
      this.vel = createVector(0, random(-8, -4))
    }
  }
}
```

```

    }
    else
    {
        this.vel = p5.Vector.random2D()
    }
    this.acc = createVector(0, 0)
}

show()
{
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height, true)
        this.exploded = false
        this.particles = []
    }

    show()
    {
        if (!this.exploded)

```

```

    {
        this.firework.show()
    }
    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].show()
    }
}

move()
{
    if (!this.exploded)
    {
        this.firework.applyForce(gravity)
        this.firework.move()
        if (this.firework.vel.y >= 0)
        {
            this.exploded = true
            this.explode()
        }
    }
    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
    }
}

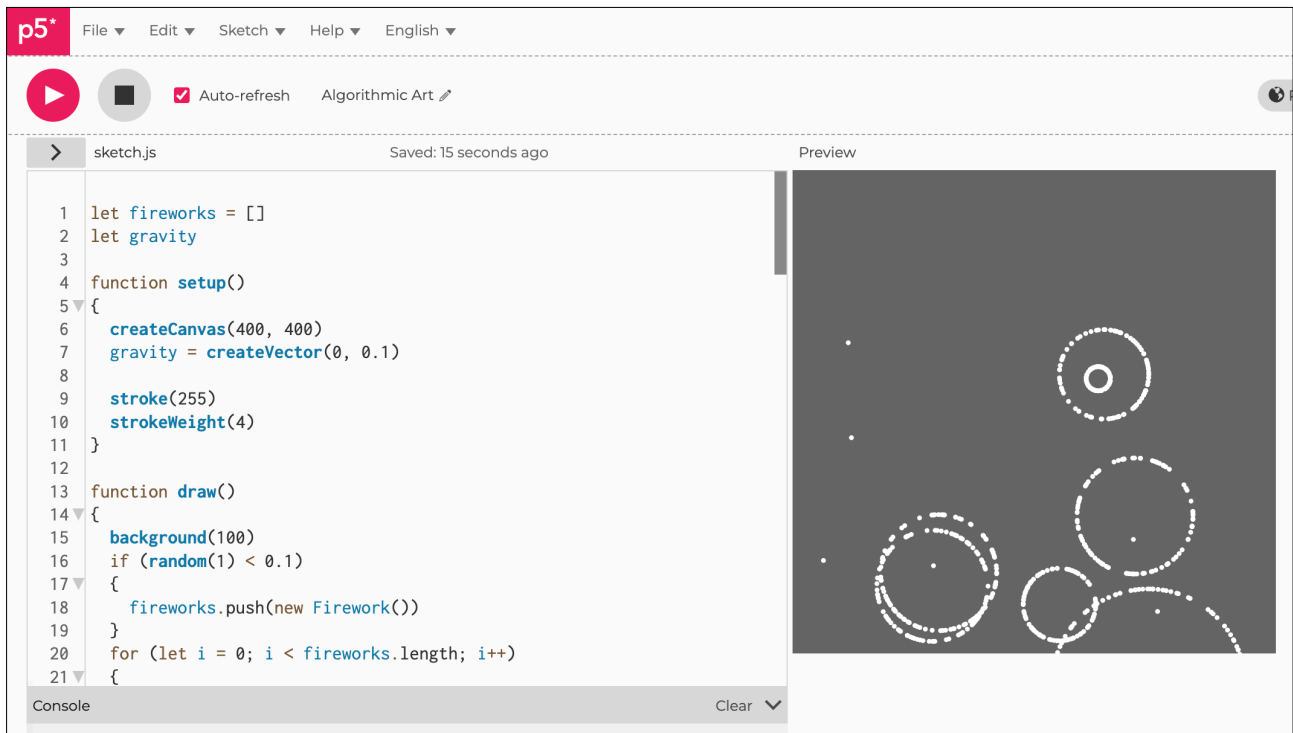
explode()
{
    for (let i = 0; i < 100; i++)
    {
        let p = new Particle(this.firework.pos.x, this.firework.pos.y)
        this.particles.push(p)
    }
}
}

```

Notes

When you run it, you get something like this. Also, it starts running slower and slower as you create more and more particles. It looks rather nice as it is.

Figure E4.25





Sketch E4.26 random velocity

We have nice circles, but we want not just random direction but also random velocity.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    if (firework)
    {
      this.vel = createVector(0, random(-8, -4))
    }
  }
}
```

```

    }
    else
    {
        this.vel = p5.Vector.random2D()
        this.vel.mult(random(1, 3))
    }
    this.acc = createVector(0, 0)
}

show()
{
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height, true)
        this.exploded = false
        this.particles = []
    }

    show()
    {

```

```

    if (!this.exploded)
    {
        this.firework.show()
    }
    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].show()
    }
}

move()
{
    if (!this.exploded)
    {
        this.firework.applyForce(gravity)
        this.firework.move()
        if (this.firework.vel.y >= 0)
        {
            this.exploded = true
            this.explode()
        }
    }
    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
    }
}

explode()
{
    for (let i = 0; i < 100; i++)
    {
        let p = new Particle(this.firework.pos.x, this.firework.pos.y)
        this.particles.push(p)
    }
}
}

```

Notes

They now explode a little bit more realistically.

Figure E4.26





Sketch E4.27 slow down

We want to keep track of those firework particles and cause them to not just fly away, also to slow down.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    if (this.firework)
    {
```

```

    this.vel = createVector(0, random(-8, -4))
  }
  else
  {
    this.vel = p5.Vector.random2D()
    this.vel.mult(random(1, 3))
  }
  this.acc = createVector(0, 0)
}

show()
{
  point(this.pos.x, this.pos.y)
}

applyForce(force)
{
  this.acc.add(force)
}

move()
{
  if (!this.firework)
  {
    this.vel.mult(0.85)
  }

  this.vel.add(this.acc)
  this.pos.add(this.vel)
  this.acc.mult(0)
}
}

class Firework
{
  constructor()
  {
    this.firework = new Particle(random(width), height, true)
    this.exploded = false
  }
}

```

```

    this.particles = []
}

show()
{
    if (!this.exploded)
    {
        this.firework.show()
    }
    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].show()
    }
}

move()
{
    if (!this.exploded)
    {
        this.firework.applyForce(gravity)
        this.firework.move()
        if (this.firework.vel.y >= 0)
        {
            this.exploded = true
            this.explode()
        }
    }
    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
    }
}

explode()
{
    for (let i = 0; i < 100; i++)
    {

```

```

    let p = new Particle(this.firework.pos.x, this.firework.pos.y)
    this.particles.push(p)
  }
}
}

```

Notes

This has something of a bizarre effect, because the particles (or sparks) would fade quite soon after the initial explosion.

Figure E4.27





Sketch E4.28 fade

To get the fade effect, we create a new variable called **lifespan**. It will decrease by **4** on each frame and it will be drawn as a **stroke()** value.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
    if (this.firework)
```

```

    {
        this.vel = createVector(0, random(-8, -4))
    }
    else
    {
        this.vel = p5.Vector.random2D()
        this.vel.mult(random(1, 3))
    }
    this.acc = createVector(0, 0)
}

show()
{
    stroke(255, this.lifespan)
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    if (!this.firework)
    {
        this.vel.mult(0.85)
        this.lifespan -= 4
    }
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
}

class Firework
{
    constructor()

```

```

{
  this.firework = new Particle(random(width), height, true)
  this.exploded = false
  this.particles = []
}

show()
{
  if (!this.exploded)
  {
    this.firework.show()
  }
  for (let i = 0; i < this.particles.length; i++)
  {
    this.particles[i].show()
  }
}

move()
{
  if (!this.exploded)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.exploded = true
      this.explode()
    }
  }
  for (let i = 0; i < this.particles.length; i++)
  {
    this.particles[i].applyForce(gravity)
    this.particles[i].move()
  }
}

explode()

```

```

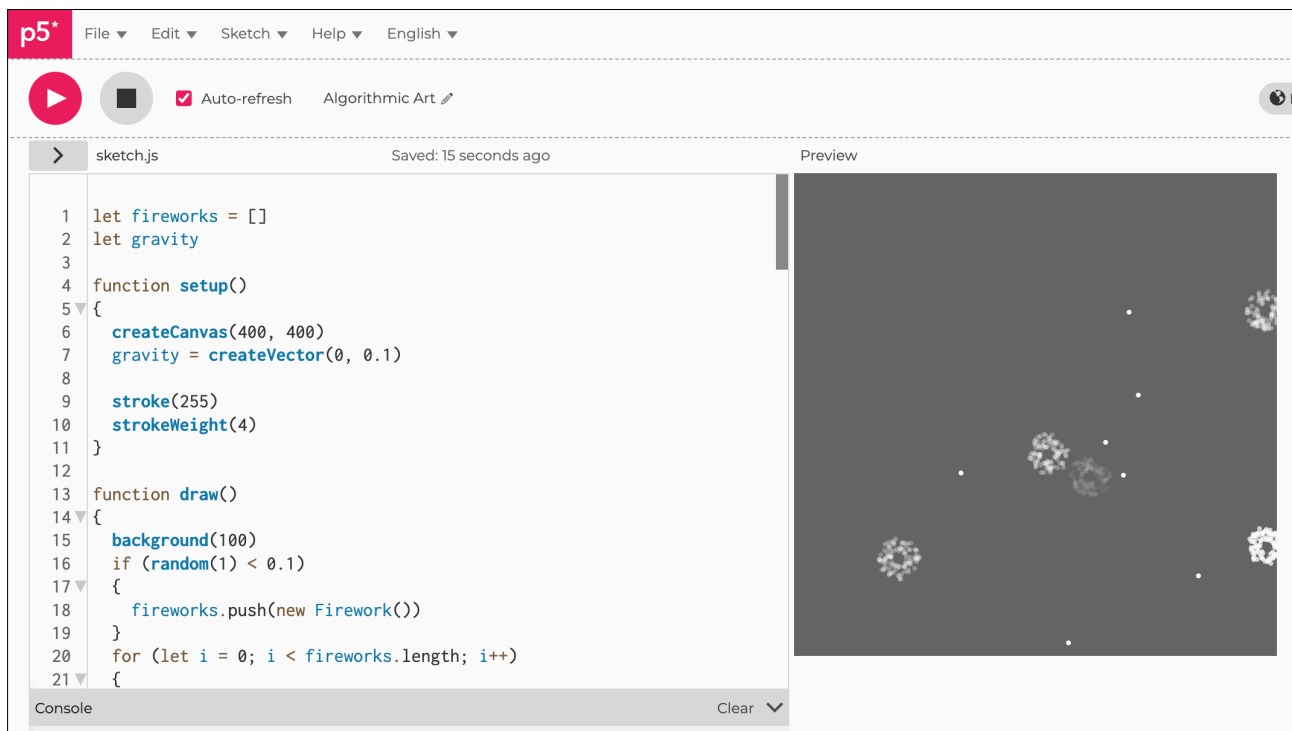
{
  for (let i = 0; i < 100; i++)
  {
    let p = new Particle(this.firework.pos.x, this.firework.pos.y)
    this.particles.push(p)
  }
}
}

```

Notes

The **lifespan** variable becomes the **alpha** value and causes them to fade.

Figure E4.28





Sketch E4.29 improve fade adjustment

We want to have only the sparks fireworks fade, so a little bit of adjustment to a few parameters for velocity as well to give a more pleasing effect. You can play with these yourself.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.1)

  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.1)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
  }
}
```

```

    if (this.firework)
    {
        this.vel = createVector(0, random(-12, -8))
    }
    else
    {
        this.vel = p5.Vector.random2D()
        this.vel.mult(random(2, 10))
    }
    this.acc = createVector(0, 0)
}

show()
{
    if (!this.firework)
    {
        strokeWeight(2)
        stroke(255, this.lifespan)
    }
    else
    {
        strokeWeight(4)
        stroke(255)
    }

    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    if (!this.firework)
    {
        this.vel.mult(0.85)
        this.lifespan -= 4
    }
}

```

```

    }
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }
}

class Firework
{
  constructor()
  {
    this.firework = new Particle(random(width), height, true)
    this.exploded = false
    this.particles = []
  }

  show()
  {
    if (!this.exploded)
    {
      this.firework.show()
    }
    for (let i = 0; i < this.particles.length; i++)
    {
      this.particles[i].show()
    }
  }

  move()
  {
    if (!this.exploded)
    {
      this.firework.applyForce(gravity)
      this.firework.move()
      if (this.firework.vel.y >= 0)
      {
        this.exploded = true
        this.explode()
      }
    }
  }
}

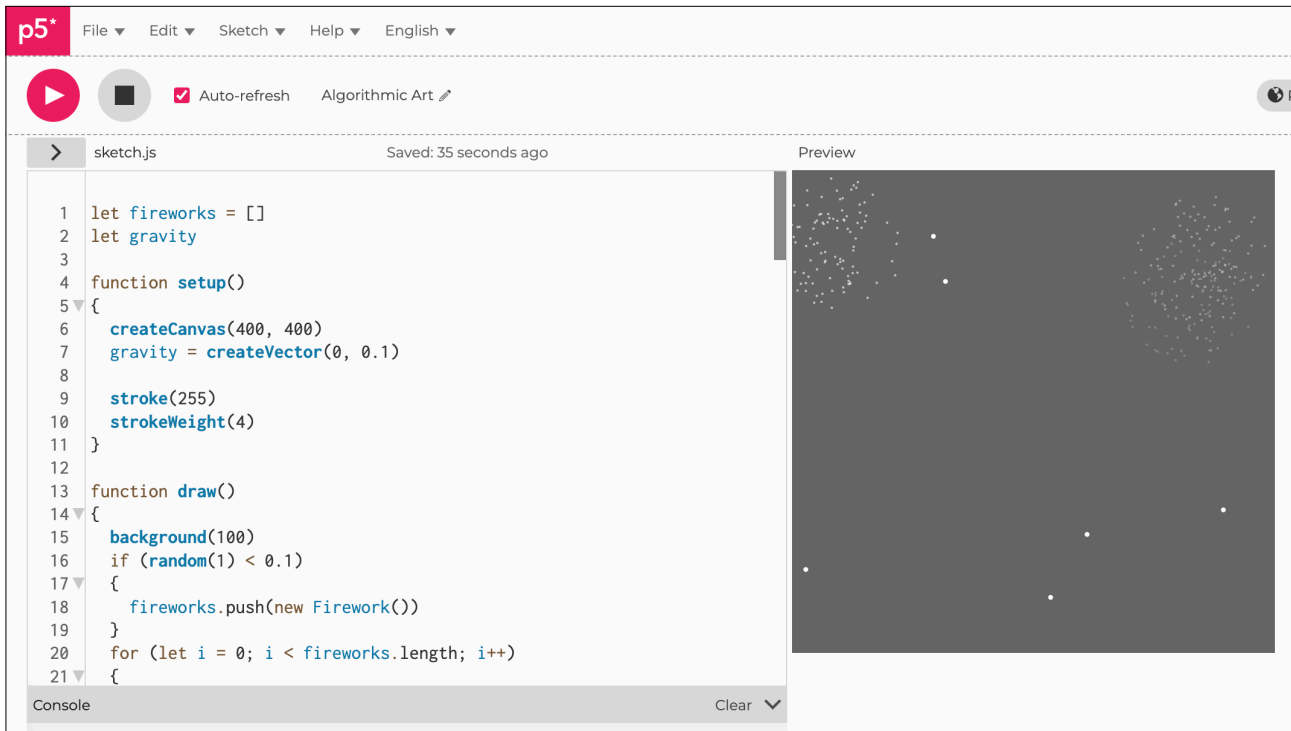
```

```
    }  
  }  
  for (let i = 0; i < this.particles.length; i++)  
  {  
    this.particles[i].applyForce(gravity)  
    this.particles[i].move()  
  }  
}  
  
explode()  
{  
  for (let i = 0; i < 100; i++)  
  {  
    let p = new Particle(this.firework.pos.x, this.firework.pos.y)  
    this.particles.push(p)  
  }  
}  
}
```

Notes

Definitely more like a firework display, but more to do.

Figure E4.29





Sketch E4.30 increase gravity reduce particles

We need to increase gravity a little more, and also reduce the number of particles being generated.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
    if (this.firework)
    {
```

```

    this.vel = createVector(0, random(-12, -8))
  }
  else
  {
    this.vel = p5.Vector.random2D()
    this.vel.mult(random(2, 10))
  }
  this.acc = createVector(0, 0)
}

show()
{
  if (!this.firework)
  {
    strokeWeight(2)
    stroke(255, this.lifespan)
  }
  else
  {
    strokeWeight(4)
    stroke(255)
  }
  point(this.pos.x, this.pos.y)
}

applyForce(force)
{
  this.acc.add(force)
}

move()
{
  if (!this.firework)
  {
    this.vel.mult(0.85)
    this.lifespan -= 4
  }
  this.vel.add(this.acc)
}

```

```

    this.pos.add(this.vel)
    this.acc.mult(0)
  }
}

class Firework
{
  constructor()
  {
    this.firework = new Particle(random(width), height, true)
    this.exploded = false
    this.particles = []
  }

  show()
  {
    if (!this.exploded)
    {
      this.firework.show()
    }
    for (let i = 0; i < this.particles.length; i++)
    {
      this.particles[i].show()
    }
  }

  move()
  {
    if (!this.exploded)
    {
      this.firework.applyForce(gravity)
      this.firework.move()
      if (this.firework.vel.y >= 0)
      {
        this.exploded = true
        this.explode()
      }
    }
  }
}

```

```

    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
    }
}

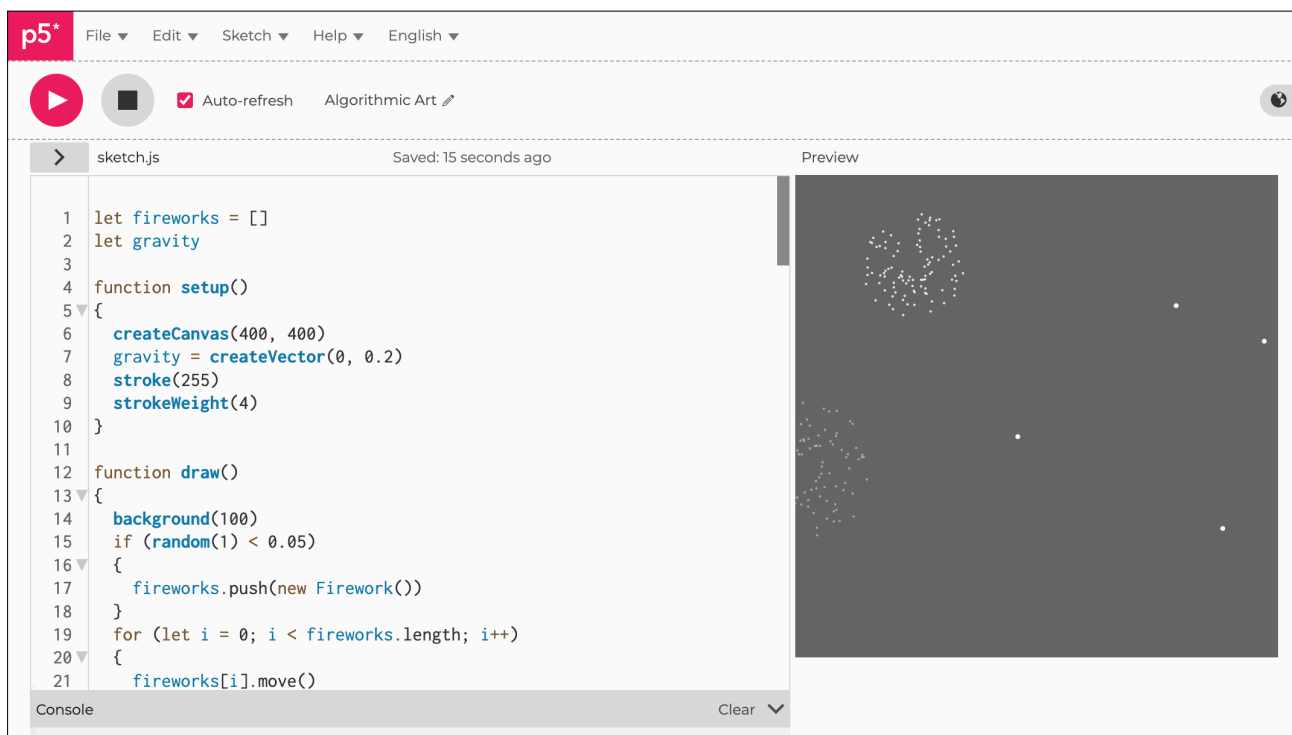
explode()
{
    for (let i = 0; i < 100; i++)
    {
        let p = new Particle(this.firework.pos.x, this.firework.pos.y)
        this.particles.push(p)
    }
}
}

```

Notes

Still improving the effect.

Figure E4.30





Sketch E4.31 removing spent particles

We create lots of particles, but we never remove them from the array. This is why it starts to run very slowly after a short while. We have to delete those particles which have faded.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
    if (this.firework)
```

```

    {
      this.vel = createVector(0, random(-12, -8))
    }
    else
    {
      this.vel = p5.Vector.random2D()
      this.vel.mult(random(2, 10))
    }
    this.acc = createVector(0, 0)
  }

  show()
  {
    if (!this.firework)
    {
      strokeWeight(2)
      stroke(255, this.lifespan)
    }
    else
    {
      strokeWeight(4)
      stroke(255)
    }
    point(this.pos.x, this.pos.y)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  move()
  {
    if (!this.firework)
    {
      this.vel.mult(0.85)
      this.lifespan -= 4
    }
  }

```

```

    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }
}

class Firework
{
  constructor()
  {
    this.firework = new Particle(random(width), height, true)
    this.exploded = false
    this.particles = []
  }

  show()
  {
    if (!this.exploded)
    {
      this.firework.show()
    }
    for (let i = 0; i < this.particles.length; i++)
    {
      this.particles[i].show()
    }
  }

  move()
  {
    if (!this.exploded)
    {
      this.firework.applyForce(gravity)
      this.firework.move()
      if (this.firework.vel.y >= 0)
      {
        this.exploded = true
        this.explode()
      }
    }
  }
}

```

```
    }  
    for (let i = this.particles.length - 1; i >= 0; i--)  
    {  
        this.particles[i].applyForce(gravity)  
        this.particles[i].move()  
    }  
}  
  
explode()  
{  
    for (let i = 0; i < 100; i++)  
    {  
        let p = new Particle(this.firework.pos.x, this.firework.pos.y)  
        this.particles.push(p)  
    }  
}  
}
```

Notes

This should make no difference to the running of the sketch. If it does, then just check though that line of code.



Sketch E4.32 splice the faded

We need to splice, remove in this case the particle that is done (faded), done will be another function which we create in the Particle class very shortly.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
    if (this.firework)
    {
```

```

    this.vel = createVector(0, random(-12, -8))
  }
  else
  {
    this.vel = p5.Vector.random2D()
    this.vel.mult(random(2, 10))
  }
  this.acc = createVector(0, 0)
}

show()
{
  if (!this.firework)
  {
    strokeWeight(2)
    stroke(255, this.lifespan)
  }
  else
  {
    strokeWeight(4)
    stroke(255)
  }
  point(this.pos.x, this.pos.y)
}

applyForce(force)
{
  this.acc.add(force)
}

move()
{
  if (!this.firework)
  {
    this.vel.mult(0.85)
    this.lifespan -= 4
  }
  this.vel.add(this.acc)
}

```

```

    this.pos.add(this.vel)
    this.acc.mult(0)
  }
}

class Firework
{
  constructor()
  {
    this.firework = new Particle(random(width), height, true)
    this.exploded = false
    this.particles = []
  }

  show()
  {
    if (!this.exploded)
    {
      this.firework.show()
    }
    for (let i = 0; i < this.particles.length; i++)
    {
      this.particles[i].show()
    }
  }

  move()
  {
    if (!this.exploded)
    {
      this.firework.applyForce(gravity)
      this.firework.move()
      if (this.firework.vel.y >= 0)
      {
        this.exploded = true
        this.explode()
      }
    }
  }
}

```

```
    for (let i = this.particles.length - 1; i >= 0; i--)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
        if (this.particles[i].done())
        {
            this.particles.splice(i, 1)
        }
    }
}

explode()
{
    for (let i = 0; i < 100; i++)
    {
        let p = new Particle(this.firework.pos.x, this.firework.pos.y)
        this.particles.push(p)
    }
}
}
```

Notes

You will get an error message if you run the code.



Sketch E4.33 it is done()

Adding the `done()` function in the `Particle` class, we use the boolean return as `true`, which means that it will splice (remove) that particle when it is done (`lifespan` is zero or less).

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
    if (this.firework)
```

```

{
  this.vel = createVector(0, random(-12, -8))
}
else
{
  this.vel = p5.Vector.random2D()
  this.vel.mult(random(2, 10))
}
this.acc = createVector(0, 0)
}

show()
{
  if (!this.firework)
  {
    strokeWeight(2)
    stroke(255, this.lifespan)
  }
  else
  {
    strokeWeight(4)
    stroke(255)
  }
  point(this.pos.x, this.pos.y)
}

applyForce(force)
{
  this.acc.add(force)
}

move()
{
  if (!this.firework)
  {
    this.vel.mult(0.85)
    this.lifespan -= 4
  }
}

```

```
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}
```

```
done()
{
    if (this.lifespan < 0)
    {
        return true
    }
    else
    {
        return false
    }
}
```

```
}
```

```
class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height, true)
        this.exploded = false
        this.particles = []
    }

    show()
    {
        if (!this.exploded)
        {
            this.firework.show()
        }
        for (let i = 0; i < this.particles.length; i++)
        {
            this.particles[i].show()
        }
    }
}
```

```

move()
{
  if (!this.exploded)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.exploded = true
      this.explode()
    }
  }
  for (let i = this.particles.length - 1; i >= 0; i--)
  {
    this.particles[i].applyForce(gravity)
    this.particles[i].move()
    if (this.particles[i].done())
    {
      this.particles.splice(i, 1)
    }
  }
}

explode()
{
  for (let i = 0; i < 100; i++)
  {
    let p = new Particle(this.firework.pos.x, this.firework.pos.y)
    this.particles.push(p)
  }
}
}

```

Notes

It should work fine now.



Sketch E4.34 sparks and rockets

We need to remove the rocket part of the firework now (we did the sparks). We want to delete the rocket firework when it has exploded and there are no more particles. First, we create a `done()` function for the firework.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = 0; i < fireworks.length; i++)
  {
    fireworks[i].move()
    fireworks[i].show()
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
    if (this.firework)
```

```

    {
      this.vel = createVector(0, random(-12, -8))
    }
    else
    {
      this.vel = p5.Vector.random2D()
      this.vel.mult(random(2, 10))
    }
    this.acc = createVector(0, 0)
  }

  show()
  {
    if (!this.firework)
    {
      strokeWeight(2)
      stroke(255, this.lifespan)
    }
    else
    {
      strokeWeight(4)
      stroke(255)
    }
    point(this.pos.x, this.pos.y)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  move()
  {
    if (!this.firework)
    {
      this.vel.mult(0.85)
      this.lifespan -= 4
    }
  }

```

```

    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}

done()
{
    if (this.lifespan < 0)
    {
        return true
    }
    else
    {
        return false
    }
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height, true)
        this.exploded = false
        this.particles = []
    }

    show()
    {
        if (!this.exploded)
        {
            this.firework.show()
        }
        for (let i = 0; i < this.particles.length; i++)
        {
            this.particles[i].show()
        }
    }
}

```

```

move()
{
  if (!this.exploded)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.exploded = true
      this.explode()
    }
  }
  for (let i = this.particles.length - 1; i >= 0; i--)
  {
    this.particles[i].applyForce(gravity)
    this.particles[i].move()
    if (this.particles[i].done())
    {
      this.particles.splice(i, 1)
    }
  }
}

explode()
{
  for (let i = 0; i < 100; i++)
  {
    let p = new Particle(this.firework.pos.x, this.firework.pos.y)
    this.particles.push(p)
  }
}

```

```

done()
{
  if (this.exploded && this.particles.length === 0)
  {
    return true
  }
}

```

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

Notes

Looking good.



Sketch E4.35 backwards array

Now we need to go to the main sketch and firstly go through the array backwards and delete it (splice it). This may seem odd, but there is logic to this, just not immediately intuitive.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
}

function draw()
{
  background(100)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }

  for (let i = fireworks.length - 1; i >= 0; i--)
  {
    fireworks[i].move()
    fireworks[i].show()

    if (fireworks[i].done())
    {
      fireworks.splice(i, 1)
    }
  }
}

class Particle
{
  constructor(x, y, firework)
  {
```

```

    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
    if (this.firework)
    {
        this.vel = createVector(0, random(-12, -8))
    }
    else
    {
        this.vel = p5.Vector.random2D()
        this.vel.mult(random(2, 10))
    }
    this.acc = createVector(0, 0)
}

show()
{
    if (!this.firework)
    {
        strokeWeight(2)
        stroke(255, this.lifespan)
    }
    else
    {
        strokeWeight(4)
        stroke(255)
    }
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    if (!this.firework)

```

```

    {
        this.vel.mult(0.85)
        this.lifespan -= 4
    }
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}

done()
{
    if (this.lifespan < 0)
    {
        return true
    }
    else
    {
        return false
    }
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height, true)
        this.exploded = false
        this.particles = []
    }

    show()
    {
        if (!this.exploded)
        {
            this.firework.show()
        }
        for (let i = 0; i < this.particles.length; i++)

```

```

    {
        this.particles[i].show()
    }
}

move()
{
    if (!this.exploded)
    {
        this.firework.applyForce(gravity)
        this.firework.move()
        if (this.firework.vel.y >= 0)
        {
            this.exploded = true
            this.explode()
        }
    }
    for (let i = this.particles.length - 1; i >= 0; i--)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
        if (this.particles[i].done())
        {
            this.particles.splice(i, 1)
        }
    }
}

explode()
{
    for (let i = 0; i < 100; i++)
    {
        let p = new Particle(this.firework.pos.x, this.firework.pos.y)
        this.particles.push(p)
    }
}

done()

```

```
{  
  if (this.exploded && this.particles.length === 0)  
  {  
    return true  
  }  
  else  
  {  
    return false  
  }  
}  
}
```

Notes

You should see the same as before, but it should not slow down. You can check by adding this line of code at the end of `draw()`:

```
console.log(fireworks.length)
```

...and remove it once you are satisfied!



Sketch E4.36 trails

Let's add some trails to the fireworks.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
  background(0)
}

function draw()
{
  background(0, 25)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = fireworks.length - 1; i >= 0; i--)
  {
    fireworks[i].move()
    fireworks[i].show()
    if (fireworks[i].done())
    {
      fireworks.splice(i, 1)
    }
  }
}

class Particle
{
  constructor(x, y, firework)
  {
    this.pos = createVector(x, y)
```

```

    this.firework = firework
    this.lifespan = 255
    if (this.firework)
    {
        this.vel = createVector(0, random(-12, -8))
    }
    else
    {
        this.vel = p5.Vector.random2D()
        this.vel.mult(random(2, 10))
    }
    this.acc = createVector(0, 0)
}

show()
{
    if (!this.firework)
    {
        strokeWeight(2)
        stroke(255, this.lifespan)
    }
    else
    {
        strokeWeight(4)
        stroke(255)
    }
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{
    if (!this.firework)
    {

```

```

        this.vel.mult(0.85)
        this.lifespan -= 4
    }
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}

done()
{
    if (this.lifespan < 0)
    {
        return true
    }
    else
    {
        return false
    }
}
}

class Firework
{
    constructor()
    {
        this.firework = new Particle(random(width), height, true)
        this.exploded = false
        this.particles = []
    }

    show()
    {
        if (!this.exploded)
        {
            this.firework.show()
        }
        for (let i = 0; i < this.particles.length; i++)
        {

```

```

        this.particles[i].show()
    }
}

move()
{
    if (!this.exploded)
    {
        this.firework.applyForce(gravity)
        this.firework.move()
        if (this.firework.vel.y >= 0)
        {
            this.exploded = true
            this.explode()
        }
    }
    for (let i = this.particles.length - 1; i >= 0; i--)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
        if (this.particles[i].done())
        {
            this.particles.splice(i, 1)
        }
    }
}

explode()
{
    for (let i = 0; i < 100; i++)
    {
        let p = new Particle(this.firework.pos.x, this.firework.pos.y)
        this.particles.push(p)
    }
}

done()
{

```

```

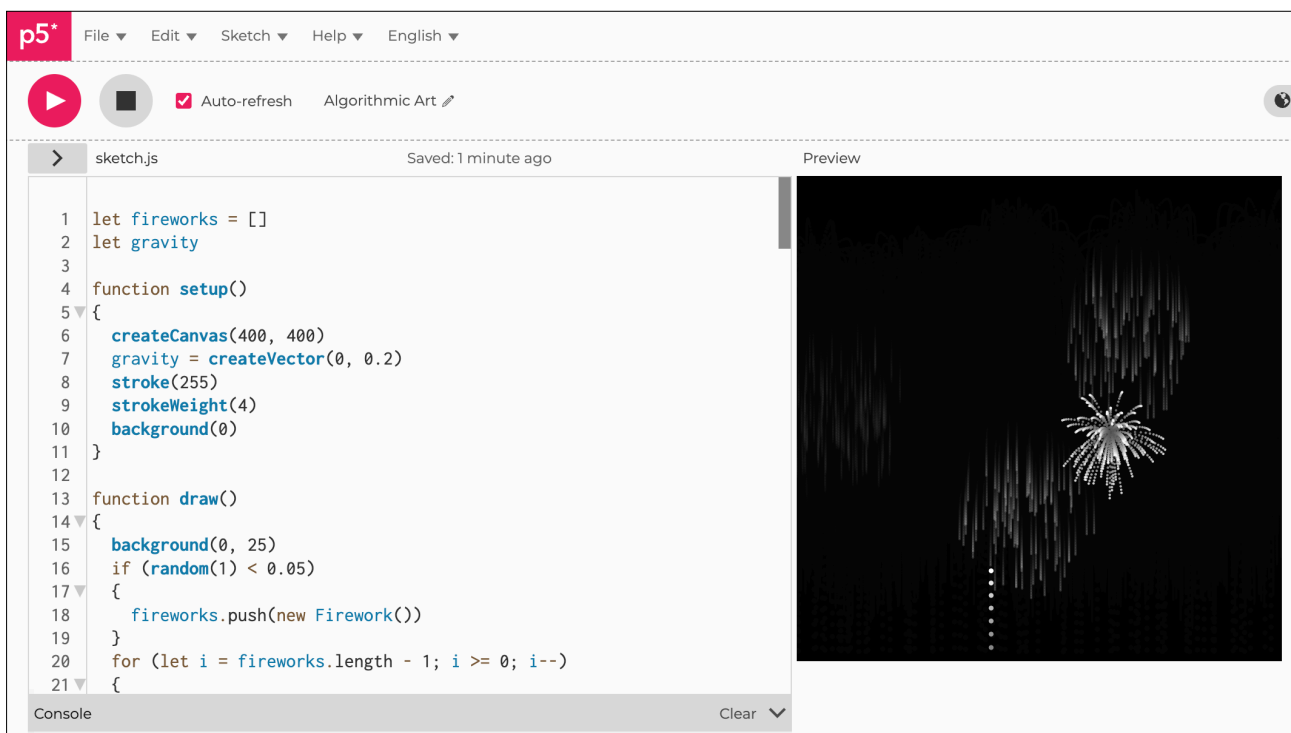
    if (this.exploded && this.particles.length === 0)
    {
        return true
    }
    else
    {
        return false
    }
}
}

```

Notes

We now have something resembling a fireworks display.

Figure E4.36





Sketch E4.37 colour

We are going to add colour with **HSB** rather than **RGB** (just a bit nicer). In the firework.js sketch, we introduce a variable called **hu**, which is the first argument for the **colourMode(HSB)**. It is going to generate a random colour.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
  background(0)
}

function draw()
{
  background(0, 25)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = fireworks.length - 1; i >= 0; i--)
  {
    fireworks[i].move()
    fireworks[i].show()
    if (fireworks[i].done())
    {
      fireworks.splice(i, 1)
    }
  }
}

class Particle
{
```

```

constructor(x, y, firework)
{
  this.pos = createVector(x, y)
  this.firework = firework
  this.lifespan = 255
  if (this.firework)
  {
    this.vel = createVector(0, random(-12, -8))
  }
  else
  {
    this.vel = p5.Vector.random2D()
    this.vel.mult(random(2, 10))
  }
  this.acc = createVector(0, 0)
}

show()
{
  if (!this.firework)
  {
    strokeWeight(2)
    stroke(255, this.lifespan)
  }
  else
  {
    strokeWeight(4)
    stroke(255)
  }
  point(this.pos.x, this.pos.y)
}

applyForce(force)
{
  this.acc.add(force)
}

move()

```

```

{
  if (!this.firework)
  {
    this.vel.mult(0.85)
    this.lifespan -= 4
  }
  this.vel.add(this.acc)
  this.pos.add(this.vel)
  this.acc.mult(0)
}

done()
{
  if (this.lifespan < 0)
  {
    return true
  }
  else
  {
    return false
  }
}
}

class Firework
{
  constructor()
  {
    this.hu = random(255)
    this.firework = new Particle(random(width), height, this.hu, true)
    this.exploded = false
    this.particles = []
  }

  show()
  {
    if (!this.exploded)
    {

```

```

    this.firework.show()
  }
  for (let i = 0; i < this.particles.length; i++)
  {
    this.particles[i].show()
  }
}

move()
{
  if (!this.exploded)
  {
    this.firework.applyForce(gravity)
    this.firework.move()
    if (this.firework.vel.y >= 0)
    {
      this.exploded = true
      this.explode()
    }
  }
  for (let i = this.particles.length - 1; i >= 0; i--)
  {
    this.particles[i].applyForce(gravity)
    this.particles[i].move()
    if (this.particles[i].done())
    {
      this.particles.splice(i, 1)
    }
  }
}

```

```

explode()

```

```

{
  for (let i = 0; i < 100; i++)
  {

```

```

    let p = new Particle(this.firework.pos.x, this.firework.pos.y, this.hu,
false)

```

```

    this.particles.push(p)

```

```

  }

```

```

}

done()
{
  if (this.exploded && this.particles.length === 0)
  {
    return true
  }
  else
  {
    return false
  }
}
}
}

```

Notes

This will break it. Don't panic, we will resolve it shortly.

Figure E4.37





Sketch E4.38 adding the colour in

In the Particle class, we add it in this way.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
  background(0)
}

function draw()
{
  background(0, 25)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = fireworks.length - 1; i >= 0; i--)
  {
    fireworks[i].move()
    fireworks[i].show()
    if (fireworks[i].done())
    {
      fireworks.splice(i, 1)
    }
  }
}

class Particle
{
  constructor(x, y, hu, firework)
  {
    this.pos = createVector(x, y)
```

```

    this.firework = firework
    this.lifespan = 255
    this.hu = hu
    if (this.firework)
    {
        this.vel = createVector(0, random(-12, -8))
    }
    else
    {
        this.vel = p5.Vector.random2D()
        this.vel.mult(random(2, 10))
    }
    this.acc = createVector(0, 0)
}

show()
{
    colorMode(HSB)
    if (!this.firework)
    {
        strokeWeight(2)
        stroke(this.hu, 255, 255, this.lifespan)
    }
    else
    {
        strokeWeight(4)
        stroke(this.hu, 255, 255)
    }
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()
{

```

```

    if (!this.firework)
    {
        this.vel.mult(0.85)
        this.lifespan -= 4
    }
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
}

done()
{
    if (this.lifespan < 0)
    {
        return true
    }
    else
    {
        return false
    }
}
}

class Firework
{
    constructor()
    {
        this.hu = random(255)
        this.firework = new Particle(random(width), height, this.hu, true)
        this.exploded = false
        this.particles = []
    }

    show()
    {
        if (!this.exploded)
        {
            this.firework.show()

```

```

    }
    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].show()
    }
}

move()
{
    if (!this.exploded)
    {
        this.firework.applyForce(gravity)
        this.firework.move()
        if (this.firework.vel.y >= 0)
        {
            this.exploded = true
            this.explode()
        }
    }
    for (let i = this.particles.length - 1; i >= 0; i--)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
        if (this.particles[i].done())
        {
            this.particles.splice(i, 1)
        }
    }
}

explode()
{
    for (let i = 0; i < 100; i++)
    {
        let p = new Particle(this.firework.pos.x, this.firework.pos.y, this.hu,
false)
        this.particles.push(p)
    }
}

```

```

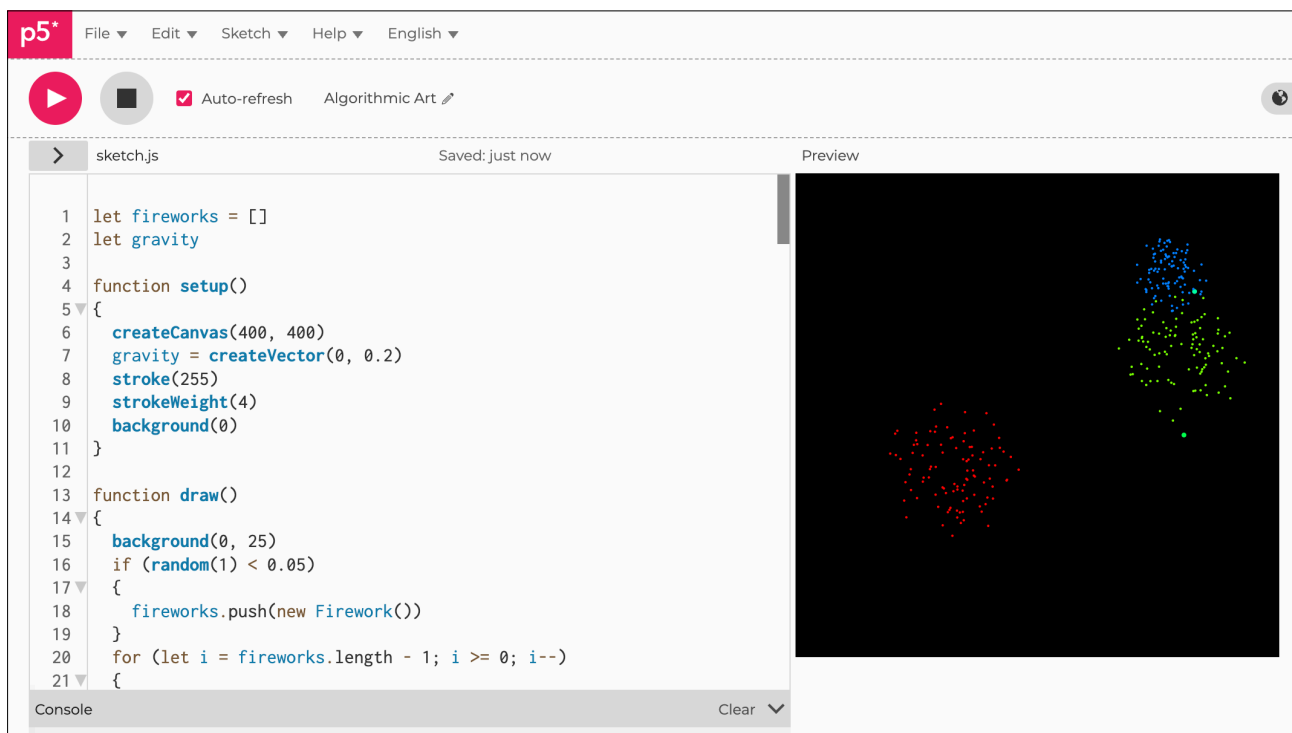
done()
{
  if (this.exploded && this.particles.length === 0)
  {
    return true
  }
  else
  {
    return false
  }
}
}

```

Notes

And we are back again, this time in colour.

Figure E4.38





Sketch E4.39 colorMode(RGB)

But we have lost the trails, so we need to change the `colorMode()` to `RGB`.

```
let fireworks = []
let gravity

function setup()
{
  createCanvas(400, 400)
  gravity = createVector(0, 0.2)
  stroke(255)
  strokeWeight(4)
  background(0)
}

function draw()
{
  colorMode(RGB)
  background(0, 25)
  if (random(1) < 0.05)
  {
    fireworks.push(new Firework())
  }
  for (let i = fireworks.length - 1; i >= 0; i--)
  {
    fireworks[i].move()
    fireworks[i].show()
    if (fireworks[i].done())
    {
      fireworks.splice(i, 1)
    }
  }
}

class Particle
{
  constructor(x, y, hu, firework)
  {
```

```

    this.pos = createVector(x, y)
    this.firework = firework
    this.lifespan = 255
    this.hu = hu
    if (this.firework)
    {
        this.vel = createVector(0, random(-12, -8))
    }
    else
    {
        this.vel = p5.Vector.random2D()
        this.vel.mult(random(2, 10))
    }
    this.acc = createVector(0, 0)
}

show()
{
    colorMode(HSB)
    if (!this.firework)
    {
        strokeWeight(2)
        stroke(this.hu, 255, 255, this.lifespan)
    }
    else
    {
        strokeWeight(4)
        stroke(this.hu, 255, 255)
    }
    point(this.pos.x, this.pos.y)
}

applyForce(force)
{
    this.acc.add(force)
}

move()

```

```

{
  if (!this.firework)
  {
    this.vel.mult(0.85)
    this.lifespan -= 4
  }
  this.vel.add(this.acc)
  this.pos.add(this.vel)
  this.acc.mult(0)
}

done()
{
  if (this.lifespan < 0)
  {
    return true
  }
  else
  {
    return false
  }
}
}

class Firework
{
  constructor()
  {
    this.hu = random(255)
    this.firework = new Particle(random(width), height, this.hu, true)
    this.exploded = false
    this.particles = []
  }

  show()
  {
    if (!this.exploded)
    {

```

```

        this.firework.show()
    }
    for (let i = 0; i < this.particles.length; i++)
    {
        this.particles[i].show()
    }
}

move()
{
    if (!this.exploded)
    {
        this.firework.applyForce(gravity)
        this.firework.move()
        if (this.firework.vel.y >= 0)
        {
            this.exploded = true
            this.explode()
        }
    }
    for (let i = this.particles.length - 1; i >= 0; i--)
    {
        this.particles[i].applyForce(gravity)
        this.particles[i].move()
        if (this.particles[i].done())
        {
            this.particles.splice(i, 1)
        }
    }
}

explode()
{
    for (let i = 0; i < 100; i++)
    {
        let p = new Particle(this.firework.pos.x, this.firework.pos.y, this.hu,
false)
        this.particles.push(p)
    }
}

```

```

}

done()
{
  if (this.exploded && this.particles.length === 0)
  {
    return true
  }
  else
  {
    return false
  }
}
}

```

Notes

Now that looks even better. We are done. You have your fireworks display.

Figure E4.39

