# Systems of particle systems

Let's review for a moment where we are. We know how to talk about an individual Particle object. We also know how to talk about a system of `Particle` objects, and this we call a "particle system." And we've defined a particle system as a collection of independent objects. But isn't a particle system itself an object? If that's the case (which it is), there's no reason why we couldn't also have a collection of many particle systems, i.e. a system of systems.

This line of thinking could of course take us even further, and you might lock yourself in a basement for days sketching out a diagram of a system of systems of systems of systems of systems of systems. Of systems. After all, this is how the world works. An organ is a system of cells, a human body is a system of organs, a neighbourhood is a system of human bodies, a city is a system of neighbourhoods, and so on and so forth. While this is an interesting road to travel down, it's a bit beyond where we need to be right now. It is, however, quite useful to know how to write a program that keeps track of many particle systems, each of which keep track of many particles. Let's take the following scenario:

- You start with a blank screen.
- You click the mouse and generate a particle system at the mouse's location.
- Each time you click the mouse, a new particle system is created at the mouse's location.
  Try it out yourself:

How would we accomplish this in code? In the previous article, we stored a single reference to a `ParticleSystem` object in the variable `ps`.

```
var ps = new ParticleSystem(new PVector(width/2, 50));
```

```
draw = function() {
  background(50, 50, 50);
  ps.addParticle();
  ps.run();
};
```

Now that we have multiple systems, a potentially ever increasing number, we don't want to store them in individually named variables. Instead, we'll use an array to keep track of all the systems. We'll start it as empty when the program begins:

```
var systems = [];
```

Whenever the mouse is pressed, a new `ParticleSystem` object is created and pushed onto the array:

```
mousePressed = function() {
  systems.push(new ParticleSystem(new PVector(mouseX, mouseY)));
};
```

And in `draw()`, instead of referencing a single `ParticleSystem` object, we now look through all the systems in the array and call `run()` on each of them.

```
draw = function() {
  background(50, 50, 50);
  for(var i = 0; i < systems.length; i++){
    systems[i].addParticle();
    systems[i].run();
  }
};
```

Now, try it yourself again, with the code we've written:

```
// A single Particle object
var Particle = function(position) {
   this.acceleration = new PVector(0, 0.05);
   this.velocity = new PVector(random(-1, 1), random(-1, 0));
   this.position = position.get();
   this.timeToLive = 255;
};

Particle.prototype.run = function() {
   this.update();
   this.display();
};

Particle.prototype.update = function(){
   this.velocity.add(this.acceleration);
   this.position.add(this.velocity);
   this.timeToLive -= 2;
};

Particle.prototype.display = function() {
   stroke(255, 255, 255, this.timeToLive);
   strokeWeight(2);
   fill(210, 210, 255, this.timeToLive);
   ellipse(this.position.x, this.position.y, 12, 12);
};
Particle.prototype.isDead = function() {
   if (this.timeToLive < 0) {
```

```
        return true;
    } else {
        return false;
    }
};

var ParticleSystem = function(position) {
    this.origin = position.get();
    this.particles = [];
};

ParticleSystem.prototype.addParticle = function() {
    this.particles.push(new Particle(this.origin));
};

ParticleSystem.prototype.run = function() {
  for (var i = this.particles.length-1; i >= 0; i--) {
      var p = this.particles[i];
      p.run();
      if (p.isDead()) {
        this.particles.splice(i, 1);
      }
    }
};
var systems = [];

mousePressed = function() {
```

```
    systems.push(new ParticleSystem(new PVector(mouseX, mouseY)));
};

draw = function() {
    background(91, 88, 184);

    for(var i = 0; i < systems.length; i++){
        systems[i].addParticle();
        systems[i].run();
    }
};
```