TAREA 2

Partículas conectadas

Programación con p5.js

¿Qué vamos a hacer?

En esta tarea vas a crear una simulación de **partículas en movimiento** que se conectan entre sí cuando están cerca. El resultado es una red animada que cambia continuamente, similar a las visualizaciones que aparecen en fondos de pantalla o en gráficos de datos en movimiento.

El programa introduce dos conceptos nuevos: los **arrays** (listas de datos) y los **objetos**. Los verás aparecer de forma natural a medida que el programa los necesite. El bucle **for** que ya conoces será la herramienta central para manejar las partículas.

> **Antes de empezar**
> Este tutorial presupone que has trabajado con el editor de p5.js y conoces setup(), draw(), background(), circle(), stroke(), fill(), random() y el bucle for. Si alguno de estos conceptos te resulta poco familiar, repasa la Tarea 1 antes de continuar.

Paso 1 — Una partícula que se mueve

Empezamos con lo más sencillo: **un único punto** que se desplaza por el lienzo. Tiene una posición (`x`, `y`) y una velocidad (`vx`, `vy`). En cada fotograma sumamos la velocidad a la posición.

```
// Variables globales: posición y velocidad de una partícula
let x, y;
let vx, vy;

function setup() {
  createCanvas(windowWidth, windowHeight);
  // Posición inicial: centro del lienzo
  x = width / 2;
  y = height / 2;
  // Velocidad inicial aleatoria (entre -3 y 3 píxeles por fotograma)
  vx = random(-3, 3);
  vy = random(-3, 3);
}

function draw() {
  background(255);   // fondo blanco
  // Mover la partícula sumando la velocidad a la posición
  x = x + vx;
  y = y + vy;
  // Rebotar al tocar los bordes
  if (x < 0 || x > width)  { vx = -vx; }
  if (y < 0 || y > height) { vy = -vy; }
  // Dibujar la partícula
  noStroke();
  fill(0);
  circle(x, y, 6);
}
```

¿Cómo funciona el rebote?
El rebote funciona invirtiendo la velocidad: si la partícula sale por el lado derecho (x > width), vx se vuelve negativa y la partícula empieza a moverse hacia la izquierda. Es como una pelota que choca contra una pared.

Experimenta
Prueba a cambiar el rango de random(-3, 3) a random(-8, 8). ¿Qué efecto tiene? ¿Y si usas random(0, 3) sin valores negativos?

Paso 2 — Muchas partículas: el array

Concepto: array

Ahora queremos 50 partículas en lugar de una. Podríamos crear 50 variables (`x1`, `x2`, `x3`…), pero eso es inviable. Un **array** es una *lista numerada* que agrupa muchos valores bajo un mismo nombre:

```
let nombres = ['Ana', 'Luis', 'María'];  // array de textos
let edades  = [17, 16, 18];              // array de números

// Acceder a un elemento: nombre_del_array[índice]
// El índice empieza en 0, no en 1
console.log(nombres[0]);  // 'Ana'
console.log(edades[2]);   // 18
```

En nuestro programa necesitamos guardar la posición y velocidad de cada partícula. La solución más directa es usar **cuatro arrays paralelos**: uno para cada dato. El índice `i` identifica a la misma partícula en todos ellos:

```
let nb = 50;          // número de partículas
let px = [];          // posición x de cada partícula
let py = [];          // posición y
let pvx = [];         // velocidad horizontal
let pvy = [];         // velocidad vertical

function setup() {
  createCanvas(windowWidth, windowHeight);
  // Inicializar cada partícula con un bucle for
  for (let i = 0; i < nb; i = i + 1) {
    px[i]  = width / 2;
    py[i]  = height / 2;
    pvx[i] = random(-3, 3);
    pvy[i] = random(-3, 3);
  }
}

function draw() {
  background(255);
  noStroke();
  fill(0);
  for (let i = 0; i < nb; i = i + 1) {
    px[i]  = px[i] + pvx[i];
    py[i]  = py[i] + pvy[i];
    if (px[i] < 0 || px[i] > width)  { pvx[i] = -pvx[i]; }
    if (py[i] < 0 || py[i] > height) { pvy[i] = -pvy[i]; }
    circle(px[i], py[i], 6);
  }
}
```

El bucle for como herramienta de gestión
Fíjate en que el bucle for aparece dos veces: una en setup() para crear todas las partículas, y otra en draw() para moverlas y dibujarlas en cada fotograma. Es exactamente el mismo patrón: recorrer todas las partículas de 0 a nb-1.

Prueba a cambiar `nb` a 200. El programa funciona igual, solo con cambiar un número. Esa es la ventaja del array combinado con el bucle.

Paso 3 — Conectar las partículas cercanas

Concepto: bucle for anidado y dist()

Para dibujar una línea entre dos partículas cercanas necesitamos comparar **cada partícula con todas las demás**. Eso requiere un bucle dentro de otro bucle: el bucle exterior recorre la primera partícula (`i`), y el interior recorre la segunda (`j`). En lenguaje natural: «para cada partícula i, compruebo su distancia con cada partícula j».

```
for (let j=0; j<nb; j++){
  let d = dist(px[i], py[i], px[j],py[j])
  if (d < dMin){
    stroke(0)
    strokeWeight(0.5)
    line(px[i], py[i], px[j],py[j])
  }
```

¿Qué hace dist()?
dist(x1, y1, x2, y2) es una función de p5.js que devuelve la distancia en píxeles entre los puntos (x1, y1) y (x2, y2). Calcula internamente lo mismo que el teorema de Pitágoras.

Nota de rendimiento
Con nb = 350 partículas, el bucle anidado realiza 350 × 350 = 122.500 comparaciones por fotograma. Si el programa va lento, reduce nb a 150 o 200.

Paso 4 — Hazlo tuyo

El programa está completo. Ahora es tu turno de tomar decisiones para diferenciarlo. Aquí tienes ideas concretas, ordenadas de menor a mayor dificultad:

Color

- Cambia el fondo a negro (`background(0)`) y las partículas a blanco (`fill(255)`). Las líneas quedan muy distintas.
- Usa `colorMode(HSB, 360, 100, 100)` y asigna a cada partícula un tono aleatorio en el constructor (`this.col = random(360)`). Úsalo en `show()` con `fill(this.col, 80, 90)`.
- Haz que las líneas sean más transparentes cuanto mayor sea la distancia entre las partículas: `stroke(0, map(d, 0, dMin, 200, 0))` (el segundo parámetro es el alfa).

Comportamiento

- Cambia el punto de inicio de todas las partículas a una posición aleatoria en lugar del centro: `new Particle(random(width), random(height))`.
- Varía `dMin` con la posición horizontal del ratón: `dMin = map(mouseX, 0, width, 30, 200)`. Las líneas de conexión cambian en tiempo real.

Entrega

1. Guarda el proyecto en el editor (necesitas cuenta para guardarlo online).
2. Copia el enlace del proyecto (botón `Share` en el editor).
3. Entrega el enlace respondiendo a las preguntas de la ficha siguiente.

Ficha de entrega

Pregunta	Tu respuesta
Título de tu composición	
¿Qué efecto produce cambiar dMin? Describe un valor concreto que hayas probado.	
¿Qué variación añadiste respecto al programa base? (si añadiste alguna)	
¿Qué fue lo más difícil de esta tarea?	
¿Qué añadirías si tuvieras más tiempo?	

Resumen de lo aprendido en esta tarea

Elemento	Para qué sirve en esta tarea
`array []`	Lista numerada que agrupa muchos valores bajo un mismo nombre.
`array[i]`	Accede al elemento i del array (el índice empieza en 0).
`dist(x1,y1,x2,y2)`	Calcula la distancia en píxeles entre dos puntos.
`for anidado`	Un bucle for dentro de otro: compara cada elemento con todos los demás.