# TAREA 1

## Lienzo interactivo

Programación con p5.js

## ¿Qué vamos a hacer?

En esta tarea vas a crear tu primera pieza de arte generativo con p5.js. El resultado será un lienzo animado que cambia en tiempo real según la posición del ratón: las formas se mueven, el color varía y el programa reacciona a ti.

No hay un único resultado correcto. Dos personas siguiendo los mismos pasos pueden llegar a composiciones completamente distintas dependiendo de sus decisiones estéticas. Eso es exactamente lo que buscamos.

*Antes de empezar*

Abre el editor online de p5.js en editor.p5js.org en el navegador (https://editor.p5js.org/) . No necesitas crear una cuenta para trabajar, aunque sí para guardar tu proyecto. Si quieres guardar, regístrate con tu correo del centro y una contraseña para este sitio.

## Paso 0 — Antes del teclado: diseña en papel

Antes de escribir una sola línea de código, coge papel y lápiz. Esta fase es importante: te ayudará a pensar qué quieres hacer antes de preocuparte por la sintaxis.

Dibuja un rectángulo que represente tu lienzo (el canvas). Dentro de él:

- Dibuja dos o tres formas: pueden ser círculos, cuadrados o rectángulos.
- Anota unas coordenadas aproximadas para cada forma (un punto x e y desde la esquina superior izquierda).
- Piensa: ¿qué cambiaría si el ratón estuviera en la esquina izquierda? ¿Y en la derecha? ¿Y abajo?
- Anota en el papel qué propiedad de cada forma cambiarías con el ratón: ¿el tamaño? ¿el color? ¿la posición?

*¿Por qué hacemos esto en papel?*

El papel te obliga a pensar el problema antes de resolverlo. En programación, empezar a escribir código sin un plan previo suele llevar a código confuso y difícil de corregir. Un boceto de dos minutos te puede ahorrar diez minutos de errores.

# Paso 1 — La estructura base: setup() y draw()

## Concepto: el bucle de dibujo

Todo programa en p5.js tiene dos funciones principales que ya estarán escritas cuando abras el editor:

```
function setup() {
  // se ejecuta UNA SOLA VEZ al inicio del programa
}

function draw() {
  // se ejecuta CONTINUAMENTE, muchas veces por segundo
}
```

Puedes imaginarlo como una película de dibujos animados: setup() prepara el estudio de animación, y draw() dibuja cada fotograma, uno tras otro, muy rápido. Por defecto p5.js dibuja unos 60 fotogramas por segundo.

## Tu primer lienzo

Dentro de setup() vamos a crear el lienzo con createCanvas(). Los dos números son el ancho y el alto en píxeles:

```
function setup() {
  createCanvas(400,600) //anchoxalto
}

function draw() {
  background(220) //fondo gris claro
}
```

Escribe esto en el editor y pulsa el botón ▶ (Play). Deberías ver un rectángulo gris. Si lo ves, ¡todo funciona correctamente!

### *¿Qué hace background()?*

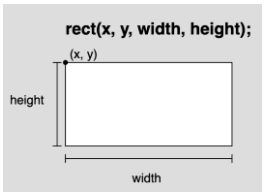
background() rellena todo el lienzo con un color. Si le pasas un solo número entre 0 y 255, obtienes un tono de gris: 0 es negro puro y 255 es blanco puro. Si no pones background() en draw(), las formas que dibujes se acumularán unas sobre otras y verás un rastro.

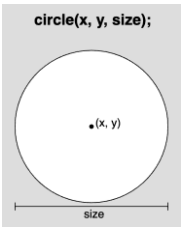
# Paso 2 — Las primeras formas

## Concepto: sistema de coordenadas

En p5.js el punto (0, 0) está en la esquina superior izquierda del lienzo. El eje X crece hacia la derecha y el eje Y crece hacia abajo (al contrario que en matemáticas).

Las funciones principales para dibujar formas son:




- ellipse(x, y, ancho, alto) — dibuja una elipse centrada en (x, y)
- rect(x, y, ancho, alto) — dibuja un rectángulo desde la esquina superior izquierda (x, y)
- circle(x, y, diametro) — dibuja un círculo centrado en (x, y)
- fill(r, g, b) — establece el color de relleno en RGB antes de dibujar
- noStroke() — elimina el borde de las formas
- stroke(r, g, b) — establece el color del borde

## Prueba: añade una forma estática

Añade esto dentro de draw(), después de background():

```
function setup() {
  createCanvas(400,600) //anchoxalto
}

function draw() {
  background (220)

  fill(255,100,50) //naranja en RGB
  noStroke() //sin borde
  ellipse(300,200,100,100) //elipse con dos radios iguales, es decir, círculo
}
```

Cambia los tres números de fill() para explorar distintos colores. Cada número va de 0 a 255 y representa Rojo, Verde y Azul respectivamente.

### *Truco para encontrar colores RGB*

Busca 'selector de color RGB' en Google. (p.ej: https://htmlcolorcodes.com/es/selector-de-color/) Verás una rueda de colores y los valores R, G, B correspondientes. Cópialos directamente a tu código.

El editor p5.js también interpreta los colores por sus nombres. Podemos, por ejemplo, configurar el fondo con esta línea, ¡usando comillas!:

```
background("CadetBlue")
```

Podemos ver los colores en este sitio web: https://htmlcolorcodes.com/es/nombres-de-los-colores/

**Nombres de Colores HTML Azul**

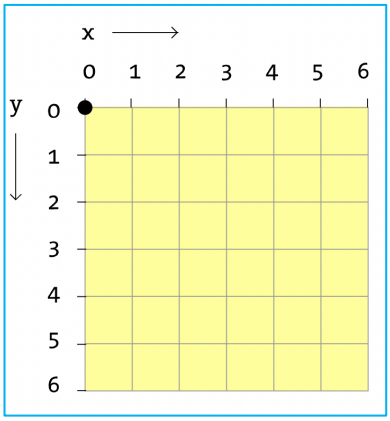
| Color | Nombre | Código Hex | Código RGB | Código HSL |
|---|---|---|---|---|
| | Aqua | 00FFFF | 0, 255, 255 | 180, 100, 50 |
| | Cyan | 00FFFF | 0, 255, 255 | 180, 100, 50 |
| | LightCyan | E0FFFF | 224, 255, 255 | 180, 100, 94 |
| | PaleTurquoise | AFEEEE | 175, 238, 238 | 180, 65, 81 |

# Paso 3 — El ratón como entrada: mouseX y mouseY

## Concepto: variables del sistema

p5.js actualiza automáticamente dos variables especiales en cada fotograma:

- mouseX — contiene la posición horizontal del ratón en el lienzo (en píxeles)
- mouseY — contiene la posición vertical del ratón en el lienzo (en píxeles)




Puedes usar estas variables en cualquier parte de draw(). Como draw() se ejecuta continuamente, el valor cambia con el ratón y el resultado se actualiza en tiempo real.

## Haz que la forma siga al ratón

Sustituye las coordenadas fijas de ellipse() por mouseX y mouseY:

```
function setup() {
  createCanvas(400,600) //anchoxalto
}

function draw() {
  background (220)

  fill("Crimson") // color carmesí en inglés
  noStroke() //sin borde
  ellipse(mouseX,mouseY,80,80) //X e Y del centro son las del ratón
}
```

Mueve el ratón por el lienzo. El círculo te sigue. Esto ocurre porque draw() redibuja el lienzo 60 veces por segundo usando el valor actualizado de mouseX y mouseY.

# Paso 4 — La función map(): traducir rangos

## Concepto: map()

mouseX va de 0 (izquierda) a 400 (derecha, si el lienzo mide 400px de ancho). Pero el tamaño de una forma puede ir de 20 a 200, o un valor de color (el R o G o B) va de 0 a 255. Necesitamos traducir un rango a otro. Para eso existe map():

```
//map(valor, inicio_original, fin_original, inicio_nuevo, fin_nuevo)
//Ejemplo si mouseX va de 0 a 400
// queremos que el tamaño vaya de 20 a 200
tamaño = map(mouseX,0,400,20,200)
```

Puedes leer map() así: "si mouseX vale 0, tamaño vale 20; si mouseX vale 400, tamaño vale 200; y cualquier valor intermedio se calcula proporcionalmente."

Aplícalo a tu código:

```
function setup() {
  createCanvas(400,600) //anchoxalto
}

function draw() {
  background (220)

  fill("Turquoise") // color turquesa en inglés
  noStroke() //sin borde

  //el tamaño depende de la posición horizontal del ratón
  tamaño = map(mouseX,0,400,20,200)

  //posición del centro del círulo fija en el centro del lienzo
  ellipse(width/2,height/2, tamaño, tamaño)
}
```

*width y height*

En lugar de escribir 400 y 600 (los valores concretos de tu lienzo), puedes usar las variables width y height que p5.js actualiza automáticamente. Así tu código funciona igual aunque cambies el tamaño del lienzo.

# Paso 5 — Color que cambia con el ratón

## Concepto: color dinámico con map()

Igual que hemos traducido la posición del ratón al tamaño de una forma, podemos traducirla a los valores de un color. Si usamos mouseX para el tono y mouseY para el brillo, el color de la forma cambia mientras movemos el ratón.

Hay dos formas habituales de manejar el color en p5.js. Te presentamos las dos para que elijas la que te resulte más intuitiva:

## Opción 1: modo RGB (Rojo, Verde, Azul)

Es el modo por defecto. Cada canal va de 0 a 255:

```
//mouseX controla el R (rojo)
//mouseY controla el canal B, azul
rojo = map(mouseX,0,width,0,255)
azul = map(mouseY,0,height,0,255)

fill(rojo,50,azul)
```

### Opción 2: modo HSB (Tono, Saturación, Brillo)

HSB es más intuitivo para crear degradados de color. El tono (hue) va de 0 a 360 y recorre el arco iris. Para usarlo hay que activarlo con colorMode() en setup():

```
function setup() {
  createCanvas(400,600) //anchoxalto
  colorMode(HSB,360,100,100) //Hue (tono), Saturación, Brillo
}

function draw() {
  background (220)

  //mouseX controla el tono
  //mouseY controla el canal B, azul
  tono = map(mouseX,0,width,0,360)

  fill(tono,80,90) //ahora en modo HSB

  noStroke() //sin borde
  ellipse(width/2,height/2, 150, 150) //radio fijo
}
```

Prueba ambas opciones y quédate con la que más te guste. No hay una respuesta correcta.

colorMode()

Los posibles parámetros a usar en la función colorMode, así como cualquier otra función que están preconfiguradas en este entorno, están ampliamente descritas en la documentación.

Es buena práctica recurrir a ella siempre que queremos aprender más.

# Paso 6 — Componer: más de una forma

Hasta ahora tenemos una forma que reacciona al ratón. Vamos a añadir formas secundarias para crear una composición. Aquí es donde entra en juego tu decisión estética personal.

Algunas ideas para formas secundarias:

- Una forma que sigue al ratón pero con la mitad de tamaño
- Una forma en la posición simétrica al ratón (width - mouseX, height - mouseY)
- Una forma fija en el centro que cambia solo de color
- Varias formas pequeñas repartidas por el lienzo con colores derivados del tono principal

Es el momento de tomar tus propias decisiones. A continuación, te mostramos dos propuestas completas distintas para que veas cómo la misma estructura puede producir

resultados muy diferentes. No copies ninguna: úsalas como inspiración para crear la tuya.

## Propuesta A — Tres lunas de colores complementarios

Esta propuesta usa el modo HSB para crear tres círculos con colores separados 120 grados en el arco iris. El ratón controla el tono principal y el tamaño general.

```
function setup() {
  createCanvas(600, 400);
  colorMode(HSB, 360, 100, 100);
}

function draw() {
  background(240, 20, 10);  // fondo casi negro azulado

  let tono    = map(mouseX, 0, width,  0, 360);
  let tamaño = map(mouseY, 0, height, 30, 180);

  noStroke();

  // Luna central: color principal
  fill(tono, 80, 90);
  ellipse(width / 2, height / 2, tamaño, tamaño);

  // Luna izquierda: 120 grados de diferencia
  fill((tono + 120) % 360, 70, 80);
  ellipse(width / 2 - 160, height / 2, tamaño * 0.6, tamaño * 0.6);

  // Luna derecha: 240 grados de diferencia
  fill((tono + 240) % 360, 70, 80);
  ellipse(width / 2 + 160, height / 2, tamaño * 0.6, tamaño * 0.6);

  // Reflejo: simetría vertical del ratón
  fill(tono, 50, 60, 0.4);
  ellipse(mouseX, mouseY, tamaño * 0.3, tamaño * 0.3);
}
```

Nota: el operador % 360 hace que el tono vuelva a 0 si supera 360, recorriendo el arco iris de forma cíclica sin salirse del rango.

```
function setup() {
  createCanvas(600, 400);
  // Modo RGB por defecto, no hace falta colorMode()
}

function draw() {
  // background con transparencia: crea un rastro que se desvanece
  background(15, 15, 30,40);

  let rojo    = map(mouseX, 0, width,  50, 255);
  let verde   = map(mouseY, 0, height, 50, 200);
  let tamaño = map(mouseX, 0, width,  20, 120);

  noFill(); //sin relleno
  strokeWeight(2); //grosor de 2 píxeles para el borde

  // Cuadrado principal en la posición del ratón
  stroke(rojo, verde, 100); //color del borde
  rect(mouseX - tamaño / 2, mouseY - tamaño / 2, tamaño, tamaño);

  // Cuadrado secundario, posición simétrica
  stroke(255 - rojo, 255 - verde, 200);
  let tam2 = tamaño * 0.6; //definimos otro tamaño algo (60%) más pequeño
  rect(width - mouseX - tam2 / 2, height - mouseY - tam2 / 2, tam2, tam2);

  // Cuadrado central fijo
  stroke(rojo, 150, verde);
  rect(width/2 - 30, height/2 - 30, 60, 60);
}
```

Nota: background(15, 15, 30, 40) usa un cuarto valor (alfa = transparencia). En lugar de borrar el lienzo completamente, lo oscurece ligeramente, creando un rastro de las posiciones anteriores del ratón. Prueba a cambiar este valor para comparar los resultados

# Ahora crea la tuya

Con todo lo que has aprendido, es el momento de tomar tus propias decisiones. Aquí tienes la estructura mínima que debe tener tu versión:

```
function setup() {
  createCanvas(600, 400);
  // Opcional: colorMode(HSB, 360, 100, 100);
}

function draw() {
  background( /* tu color de fondo */ );

  // Al menos una variable calculada con map()
  let ??? = map(mouseX, 0, width, ???, ???);

  // Al menos dos formas distintas
  // Forma principal:
  fill( /* tu color */ );
  ellipse( /* o rect() */ );

  // Forma secundaria:
  fill( /* tu color */ );
  ellipse( /* o rect() */ );
}
```

Lista de decisiones que debes tomar:

1. ¿Qué modo de color usas: RGB o HSB?
1. ¿Cuál es el color de tu fondo? ¿Oscuro o claro?
1. ¿Qué formas usas: círculos, cuadrados, rectángulos, mezcla?
1. ¿Qué controla mouseX: el color, el tamaño, la posición de una forma?
1. ¿Qué controla mouseY: el mismo concepto o algo diferente?
1. ¿Cuántas formas tienes en pantalla al mismo tiempo?
1. ¿Tu fondo borra el fotograma anterior o deja rastro (alpha)?

# Entrega

Cuando tu composición esté terminada:

- Guarda el proyecto en el editor (necesitas cuenta si quieres guardarlo online).
- Copia el enlace del proyecto (botón Compartir / Share en el editor).
- Entrega el enlace en la tarea respondiendo a estas preguntas..

### Ficha de tarea (rellena a mano o en el editor)

| | |
|---|---|
| Título de tu composición | |
| ¿Qué hace mouseX en tu programa? | |
| ¿Qué hace mouseY en tu programa? | |
| ¿Qué modo de color usaste y por qué? | |
| ¿Qué fue lo más difícil de esta tarea? | |
| ¿Qué cambiarías si tuvieras más tiempo? | |

# Reto extra (opcional)

Puedes probar a intentar algo más, aquí van unas cuantas ideas:

- Añade frameCount a tu código: es una variable de p5.js que cuenta el fotograma actual. Úsala con sin() o cos() para crear movimiento autónomo sin necesidad del ratón.
- Haz que tu composición reaccione también al botón del ratón: usa mouseIsPressed (true/false) para cambiar algo cuando se pulsa el botón.
- Cambia el tamaño del lienzo a 800x600 y añade una tercera forma que use tanto mouseX como mouseY de una forma que no hayas usado antes.

```
// Ejemplo de uso de frameCount con sin()
// sin() devuelve un valor que oscila entre -1 y 1
// map() lo convierte al rango que necesites

let oscilacion = map(sin(frameCount * 0.05), -1, 1, 20, 150);
ellipse(width / 2, height / 2, oscilacion, oscilacion);
```

# Resumen de lo aprendido en esta tarea

| Elemento | Para qué sirve |
|---|---|
| setup() | Se ejecuta una vez. Aquí se crea el lienzo con createCanvas(). |
| draw() | Se ejecuta en bucle continuo. Aquí se dibuja cada fotograma. |
| background() | Borra el lienzo (o lo oscurece si se añade transparencia). |
| ellipse() / rect() | Dibujan formas en las coordenadas indicadas. |
| fill() / stroke() | Establecen el color de relleno y borde antes de dibujar. |
| mouseX / mouseY | Variables que guardan la posición actual del ratón. |
| map() | Convierte un valor de un rango a otro proporcionalmente. |
| colorMode(HSB,...) | Cambia el modelo de color a Tono / Saturación / Brillo. |