

Taller p5

Parte I: p5.js

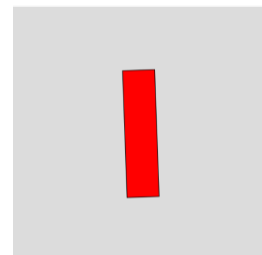
Vamos a aprender y practicar un poco con JavaScript, y las librerías p5 y p5.quadrille. Para hacerlo, son importantes las siguientes recursos:

1. [p5.js web editor](#) para escribir los programas
2. [p5.js website](#) que es el sitio de referencia oficial de la librería p5.js

Vamos a seguir la parte inicial del video [Learn p5.js for Creative Coding – 5 Beginner Projects](#) para realizar los siguientes ejercicios.

Ejercicio 1: rectángulo rotando (0 - 3:20)

Haga un programa que rote un rectángulo de dimensiones 50 x 200, en el centro del canvas, en el sentido contrario a las manecillas del reloj. Su color debe ser rojo (o su color favorito).



```
index.html •
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.11.1/p5.js">
5   </script>
6     <script
7   src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.11.1/addons/p5.sound.min.js"
8   ></script>
9     <link rel="stylesheet" type="text/css" href="style.css">
10    <meta charset="utf-8" />
11  </head>
12  <body>
13    <main>
14    </main>
15    <script src="sketch.js"></script>
16    <script src="block.js"></script>
17  </body>
18 </html>
```

```

block.js
1 class Block {
2   constructor(x,y){
3     this.x = x;
4     this.y = y;
5     this.angle = 0;
6     this.c = color(255, 0, 0);
7   }
8
9   display(){
10    fill(this.c);
11    stroke(this.c);
12    translate(this.x,this.y);
13    //rotate(this.angle);
14    rect(0,0,200,50);
15  }
16
17  move(){
18    let distance = dist(mouseX,mouseY,this.x,this.y);
19    if(distance < distMouse){
20      this.angle -= 1;
21    }
22  }
23 }

```

Ejercicio 2: rectángulo rotando con control del mouse (3:25 - 5:12)

Modifique el programa anterior para que el rectángulo rote solo cuando el mouse esté cerca de él (menos de 50 pixeles de su centro)

```

sketch.js
1 let distMouse = 50;
2 let b;
3
4 function setup() {
5   createCanvas(400, 400);
6   rectMode(CENTER);
7   angleMode(DEGREES);
8
9   b = new Block(width/2,height/2);
10 }
11
12 function draw() {
13   background(220);
14   b.move();
15   b.display();
16 }

```

```

blockjs
class Block {
  constructor(x,y){
    this.x = x;
    this.y = y;
    this.angle = 0;
    this.c = color(255, 0, 0);
  }

  display(){
    fill(this.c);
    stroke(this.c);
    translate(this.x,this.y);
    rotate(this.angle);
    rect(0,0,200,50);
  }

  move(){
    let distance = dist(mouseX,mouseY,this.x,this.y);
    if(distance < distMouse){
      this.angle -= 1;
    }
  }
}

```

Ejercicio 3: creación de una clase (5:15 - 8:30)

Modifique el programa anterior para definir la clase **Block**, que representa un bloque. Defina esta nueva clase en el archivo *block.js*. El programa debe tener la misma funcionalidad que en el ejercicio anterior.

La clase **Block** debe tener los siguientes atributos:

- Variables de instancia:
 - x (abscisa)
 - y (ordenada)
 - size (el bloque será cuadrado)
 - angle
- Métodos:
 - move: rota el rectángulo cuando el mouse se acerca
 - display: dibuja el rectángulo de acuerdo a su posición y ángulo

```

< blockjs
1 class Block {
2   constructor(x,y,size){
3     this.x = x;
4     this.y = y;
5     this.angle = 0;
6     this.size = size;
7     this.c = color(255, 0, 0);
8   }
9
10  display(){
11    fill(this.c);
12    stroke(this.c);
13    translate(this.x,this.y);
14    rotate(this.angle);
15    rect(0,0,this.size,this.size);
16  }
17
18  move(){
19    let distance = dist(mouseX,mouseY,this.x,this.y);
20    if(distance < distMouse){
21      this.angle -= 1;
22    }
23  }
24 }

```

```
sketch.js

let distMouse = 50;
let b;

function setup() {
  createCanvas(400, 400);
  rectMode(CENTER);
  angleMode(DEGREES);

  b = new Block(width/2, height/2, 50);
}

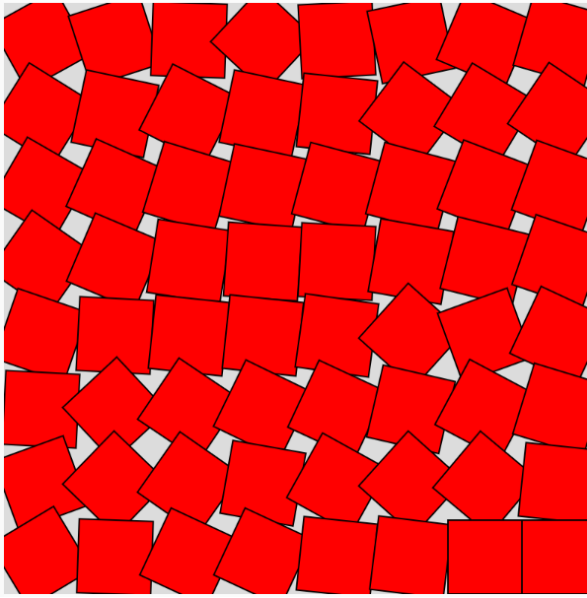
function draw() {
  background(220);
  b.move();
  b.display();
}
```

Ejercicio 4: creación de un arreglo de objetos (8:30 - 12:15)

Modifique su programa para que ahora construya, no un solo bloque, sino una matriz de bloques cuadrados. En el archivo sketch.js debe definir un par de funciones:

- `createBlocks()`: se encarga de crear una matriz de bloques
- `moveBlocks()`: se encarga de rotar y dibujar cada uno de los bloques de la matriz.

La primera función se debe invocar dentro del *setup*, y la segunda, dentro de la función *draw*. Los resultados deben ser como se muestra en la figura siguiente:



Sketch Files + < sketch.js*

JS Block.js
index.html
JS sketch.js
style.css

```
1 let distMouse = 50;
2 let size = 50;
3 let b;
4
5 function setup() {
6   createCanvas(400, 400);
7   rectMode(CENTER);
8   angleMode(DEGREES);
9   cols = width/size
10  rows = height/size
11
12  blocks = createBlocks(cols,rows,size);
13 }
14
15 function draw() {
16   background(220);
17   moveBlocks(blocks,cols,rows);
18 }
19
20 function createBlocks(cols,rows,size){
21   let Blocks = [];
22   for(let i = 0; i< cols; i++){
23     Blocks[i] = []
24     for(let j = 0; j<rows; j++){
25       Blocks[i][j] = new Block(size/2 +
26         i*size,size/2 + j*size,size);
27     }
28   }
29   return Blocks;
30 }
31
32 function moveBlocks(Blocks,cols,rows){
33   for(let i = 0; i< cols; i++){
34     for(let j = 0; j<rows; j++){
35       Blocks[i][j].move();
36       Blocks[i][j].display();
37     }
38   }
39 }
```

Preview

< Block.js

```
1 class Block{
2   constructor(x,y,size){
3     this.x = x;
4     this.y = y;
5     this.angle = 0;
6     this.size = size;
7     this.colour = color(255,0,0);
8   }
9
10  display(){
11    fill(this.colour);
12    stroke(0); // Borde negro
13    strokeWeight(2); // Grosor del
borde
14    push();
15    translate(this.x,this.y);
16    rotate(this.angle);
17    rect(0,0,this.size,this.size);
18    pop();
19  }
20
21  move(){
22    let distance =
dist(mouseX,mouseY,this.x,this.y);
23    if(distance < distMouse){
24      this.angle -= 1;
25    }
26  }
27
28 }
```

Preview

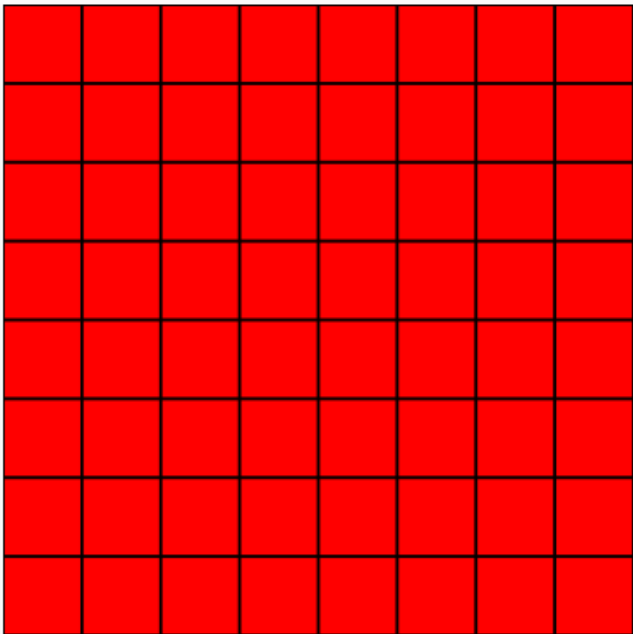
Ejercicio 5: movimiento completo de 90° (12:15 - 13:50)

Modifique su programa para que ahora los bloques giren 90° a la izquierda, apenas se aproxime en mouse. Aunque el mouse se aleje, el bloque debe completar su giro de 90° a la izquierda.

< Block.js

Preview

```
1 class Block {
2   constructor(x, y, size) {
3     this.x = x;
4     this.y = y;
5     this.angle = 0;
6     this.targetAngle = 0; // Ángulo al que
    debe girar
7     this.size = size;
8     this.colour = color(255, 0, 0);
9   }
10
11  display() {
12    fill(this.colour);
13    stroke(0); // Borde negro
14    strokeWeight(2); // Grosor del borde
15    push();
16    translate(this.x, this.y);
17    rotate(this.angle);
18    rect(0, 0, this.size, this.size);
19    pop();
20  }
21
22  move() {
23    let distance = dist(mouseX, mouseY,
    this.x, this.y);
24    if (distance < distMouse && this.angle
    === this.targetAngle) {
25      this.targetAngle -= 90;
26    }
27    if (this.angle > this.targetAngle) {
28      this.angle -= 1;
29    }
30  }
31 }
```



Ejercicio 6: movimiento de bloques solo si el mouse se mueve (13:52 - 18:20)

Modifique su programa para que ahora los bloques se queden en su posición inicial, siempre que el mouse se detenga.

```
Block.js • Prev
1 class Block {
2   constructor(x, y, size) {
3     this.x = x;
4     this.y = y;
5     this.angle = 0;
6     this.targetAngle = 0; // Ángulo al que
    debe girar
7     this.size = size;
8     this.colour = color(255, 0, 0);
9   }
10
11  display() {
12    fill(this.colour);
13    stroke(0); // Borde negro
14    strokeWeight(2); // Grosor del borde
15    push();
16    translate(this.x, this.y);
17    rotate(this.angle);
18    rect(0, 0, this.size, this.size);
19    pop();
20  }
21
22  move() {
23    let distance = dist(mouseX, mouseY,
24      this.x, this.y);
25    if (distance < distMouse && this.angle
26      === this.targetAngle) {
27      this.targetAngle -= 90;
28    }
29    if (this.angle > this.targetAngle) {
30      this.angle -= 1;
31    }
32  }
33 }
```

Ejercicio 7: Cambiar apariencia y tamaño de los bloques (18:20 - 21:35)

Modifique su programa para que ocurran los siguientes cambios:

- El tamaño de cada cuadrícula de la matriz y cada bloque sea de solo 10
- Cambie el color de la cuadrícula a negro, en la función *draw*. (cero)
- Hacer blancos los bordes de los bloques y que tengan un color. Es decir agregue el atributo 'color' en la clase **Block**. Ese color debe ser oscuro (70).
- Disminuir un poco el tamaño de los bloques para que no se dibujen pegadas unas de otras. Use la variable global 'offset' para controlar esa disminución de tamaño.

- Hacer que los bloques cambien de color momentaneamente cuando el mouse pase cerca o sobre ellos. Deben recuperar su color original apenas completen su giro de 90°.

```

1 class Block {
2   constructor(x, y, size) {
3     this.x = x;
4     this.y = y;
5     this.angle = 0;
6     this.targetAngle = 0; // Ángulo al que debe girar
7     this.size = size;
8     this.colour = 70; // Color actual
9     this.originalColour = this.colour; // Guarda el color original
10    this.prevMouse = createVector(mouseX, mouseY); // Guarda la posición previa del mouse
11  }
12
13  display() {
14    noFill();
15    stroke(0); // Borde negro
16    strokeWeight(2); // Grosor del borde
17    push();
18    translate(this.x, this.y);
19    rotate(this.angle);
20    rect(0, 0, this.size - offset, this.size - offset);
21    pop();
22  }
23
24  move() {
25    let distance = dist(mouseX, mouseY, this.x, this.y);
26    let mouseMoved = mouseX !== this.prevMouse.x || mouseY !== this.prevMouse.y;
27
28    // Si el mouse está cerca y se está moviendo, establece el ángulo objetivo 90° más a la izquierda
29    if (distance < distMouse && this.angle === this.targetAngle && mouseMoved) {
30      this.targetAngle -= 90;
31      this.colour = color(255); // Cambia a blanco temporalmente
32    }
33
34    // Realiza el giro gradual hacia el ángulo objetivo
35    if (this.angle > this.targetAngle) {
36      this.angle -= 1;
37    }
38
39    // Restaurar color cuando se complete el giro
40    if (this.angle === this.targetAngle) {
41      this.colour = this.originalColour;
42    }
43
44    // Actualiza la posición anterior del mouse
45    this.prevMouse.set(mouseX, mouseY);
46  }

```

Ejercicio 8: cambiar los bloques por líneas que formen una x en cada cuadrícula (21:35)

Modifique su programa para que en la cuadrícula se desplieguen líneas que formen una x en cada celda de la cuadrícula. ¿podría crear una nueva clase, en vez de modificar la clase **Block**?

```

1 class Block {
2   constructor(x, y, size) {
3     this.x = x;
4     this.y = y;
5     this.angle = 0;
6     this.targetAngle = 0; // Ángulo al que debe girar
7     this.size = size;
8     this.colour = 70; // Color actual
9     this.originalColour = this.colour; // Guarda el color original
10    this.prevMouse = createVector(mouseX, mouseY); // Guarda la posición previa del mouse
11    this.cross = new Cross(x, y, size); // Agrega una instancia de Cross
12  }
13
14  display() {
15    noFill();
16    stroke(0); // Borde negro
17    strokeWeight(2); // Grosor del borde
18    push();
19    translate(this.x, this.y);
20    rotate(this.angle);
21    rect(0, 0, this.size - offset, this.size - offset);
22    pop();
23
24    // Muestra la X
25    this.cross.display();
26  }
27
28  move() {
29    let distance = dist(mouseX, mouseY, this.x, this.y);
30    let mouseMoved = mouseX !== this.prevMouse.x || mouseY !== this.prevMouse.y;
31
32    // Si el mouse está cerca y se está moviendo, establece el ángulo objetivo 90° más a la izquierda
33    if (distance < distMouse && this.angle === this.targetAngle && mouseMoved) {
34      this.targetAngle -= 90;
35      this.colour = color(255); // Cambia a blanco temporalmente
36    }
37
38    // Realiza el giro gradual hacia el ángulo objetivo
39    if (this.angle > this.targetAngle) {
40      this.angle -= 1;
41    }
42
43    // Restaurar color cuando se complete el giro
44    if (this.angle === this.targetAngle) {
45      this.colour = this.originalColour;
46    }
47
48    // Actualiza la posición anterior del mouse
49    this.prevMouse.set(mouseX, mouseY);
50  }
51 }

```

```

1 class Cross {
2   constructor(x, y, size) {
3     this.x = x;
4     this.y = y;
5     this.size = size;
6   }
7
8   display() {
9     stroke(0);
10    strokeWeight(1);
11    line(this.x - this.size / 2, this.y - this.size / 2, this.x + this.size / 2, this.y + this.size / 2);
12    line(this.x + this.size / 2, this.y - this.size / 2, this.x - this.size / 2, this.y + this.size / 2);
13  }
14 }
15

```

Parte II: p5.quadrille

Implementar el juego tic tac toe en cualquiera de las dos formas que se proponen en [Quadrille API - Object Oriented Programming](#).