

---

# Crear Videojuegos y Simulaciones con JS

Omar Jordán Jordán (Omwekiatl) - 2025



# Introducción

Se expone la realización de un minijuego web utilizando JS puro, es decir, sin librerías que simplifiquen el proceso de desarrollo, este material es educativo, enfocado en la enseñanza de JS así como de las bases para la creación de videojuegos y simulaciones, incluye: main loop con su modelo de lógica y renderizado separados, cálculo vectorial, detección de colisiones circulares, entradas por teclado y mouse, obtención de datos HTML, dibujado de sprites con sub-imágenes, animación, movimiento 2D, game over.

Ejecutable:

[https://omwekiatl.xyz/logic\\_simula/](https://omwekiatl.xyz/logic_simula/)

Repositorio:

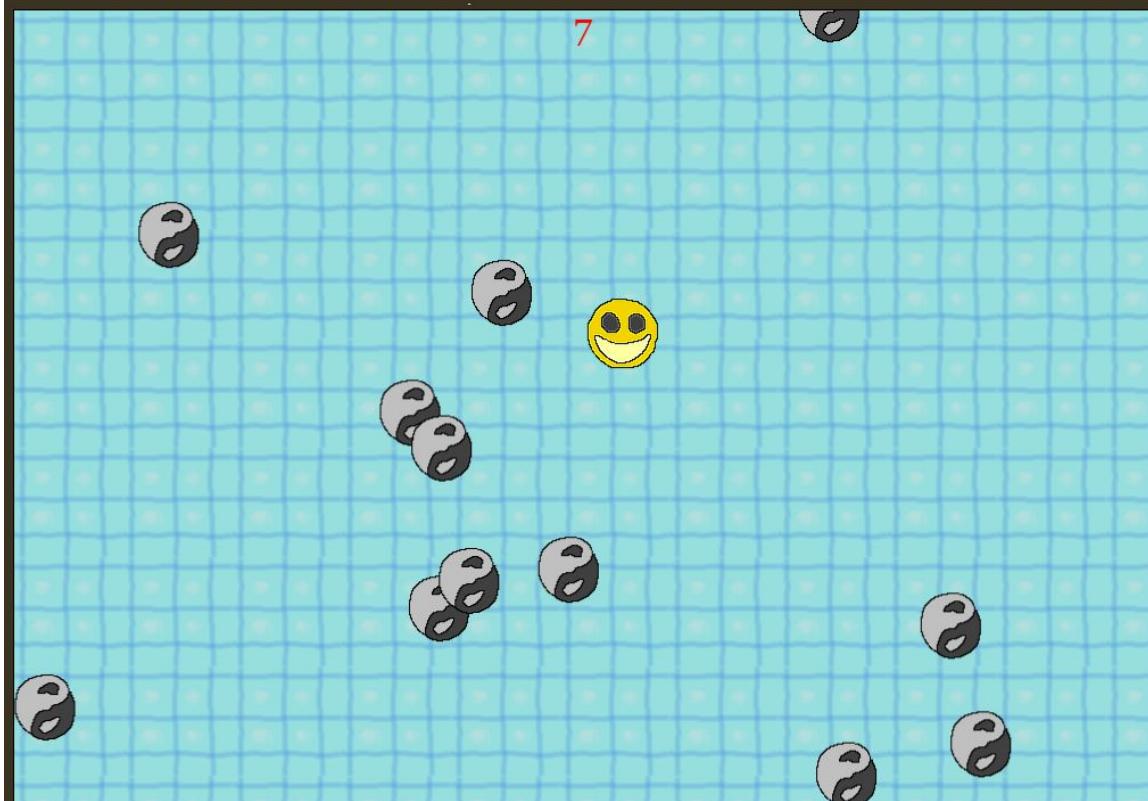
[https://github.com/OmarsaurioJordan/PracticasWeb/tree/main/logic\\_simula](https://github.com/OmarsaurioJordan/PracticasWeb/tree/main/logic_simula)

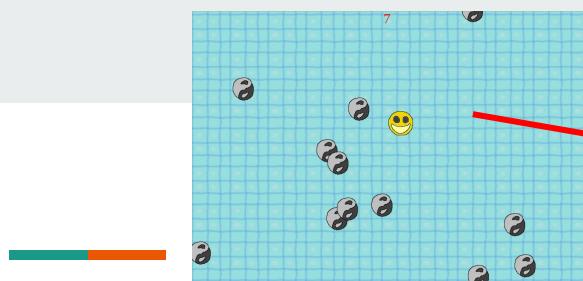
# Gameplay

El jugador deberá sobrevivir el mayor tiempo posible, moviéndose por el escenario para evadir oponentes, los cuales se lanzan de un lado a otro en línea recta, dirigidos pseudo aleatoriamente y siendo cada vez más de ellos

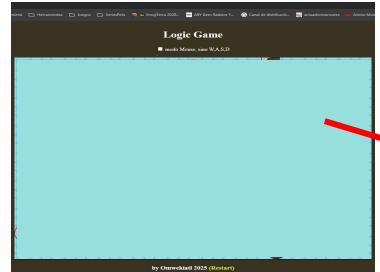
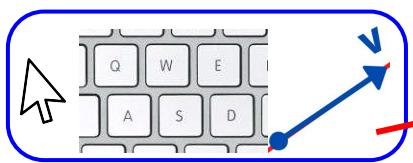
## Logic Game

■ modo Mouse, sino W,A,S,D





## Estructura



LOGIC\_SI... E

scripts

JS juego.js

JS Objeto.js

JS Player.js

JS Rock.js

JS Sprites.js

JS tools.js

sounds

final.wav

golpe.wav

musica.wav

tiro.wav

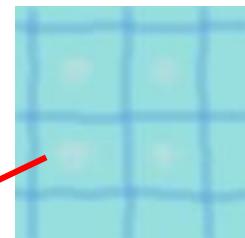
sprites

fondo.png

player.png

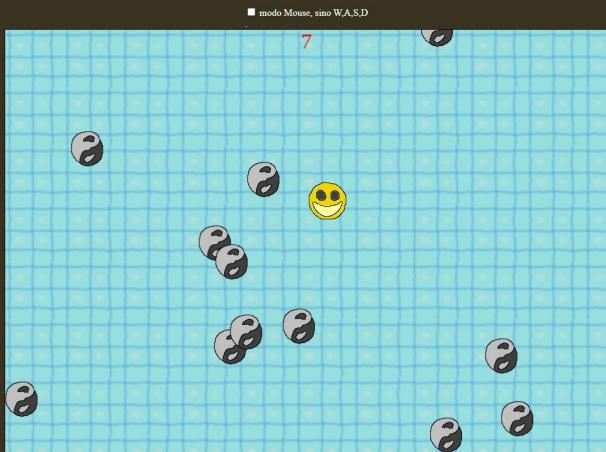
rock.png

index.php



index.php

# HTML canvas



```
1  <?php
2      $modoMouse = isset($_GET['modoMouse']) ? " checked" : "";
3  ?>
4
5
6
7      <body>
8          <h1>Logic Game</h1>
9          <div>
10             <input type="checkbox" id="modoMouse" <?php echo $modoMouse; ?>>
11             <label for="modoMouse"> modo Mouse, sino W,A,S,D<br></label>
12         </div>
13         <canvas id="lienzo" width="1000" height="700"
14             style="border:1px solid black;"></canvas>
15         <h3>by Omwekiatl 2025 <a href="" id="restart">(Restart)</a></h3>
16     </body>
17     <script src="scripts/tools.js" defer></script>
18     <script src="scripts/Sprites.js" defer></script>
19     <script src="scripts/Objeto.js" defer></script>
20     <script src="scripts/Player.js" defer></script>
21     <script src="scripts/Rock.js" defer></script>
22     <script src="scripts/juego.js" defer></script>
23 </html>
```

juego.php

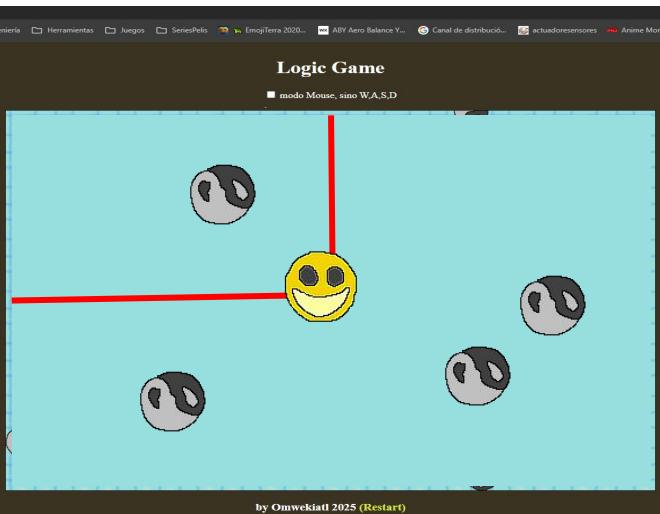
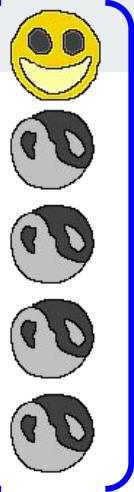
# Juego inicializa 1

```
1 // obtener informacion del lienzo
2 const canvas = document.getElementById("lienzo");
3 const ctx = canvas.getContext("2d");
4 const width = canvas.width;
5 const height = canvas.height; 
6 // lectura de comandos desde HTML
7 let modoMouse = document.getElementById("modoMouse").checked;
8 document.getElementById("modoMouse").addEventListener("change", (event) => {
9     modoMouse = event.target.checked;
10    document.getElementById("restart").href = modoMouse ?
11        "index.php?modoMouse=1" : "index.php";
12 });
13 // activar rutinas de lectura de comandos
14 newMouseListener(canvas);
15 newKeyboardListener();
```



juego.php

# Juego inicializa 2

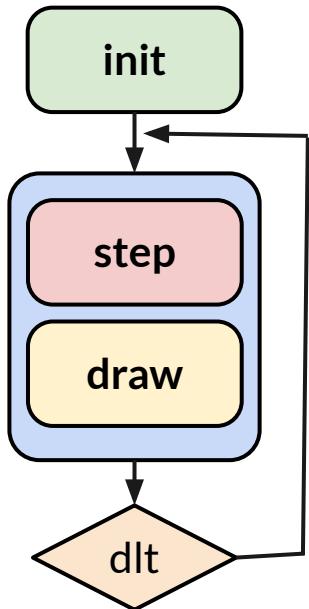


```
38 // estructuras de datos de la simulacion
39 let objetos = [] // contiene todos los objetos del juego
40 let puntaje = 0 // segundos sobreviviendo
41 let respawn_rock = {
42     reloj: 0, // segundos para crear nuevo Rock
43     espera: 4
44 };
45
46 // crear al Player
47 objetos.push(new Player({
48     x: width / 2,
49     y: height / 2
50 }));
51
52 // crear las Rock iniciales
53 for (let i = 0; i < 10; i++) {
54     objetos.push(new Rock());
55 }
```

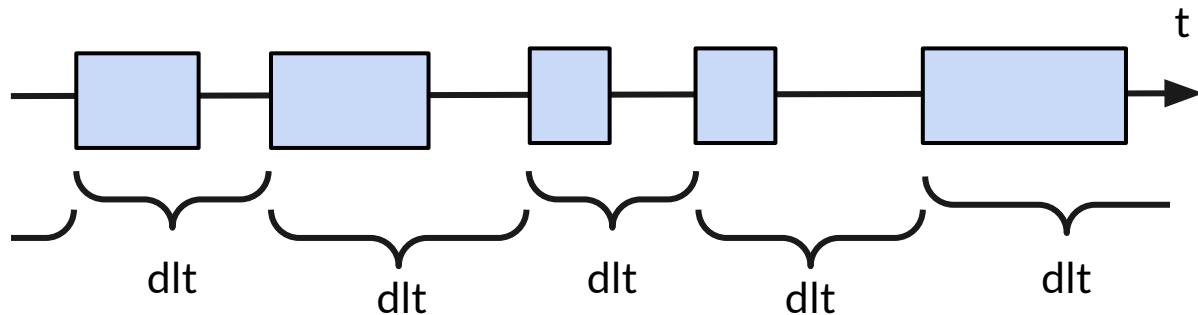


## juego.php

### Juego main loop

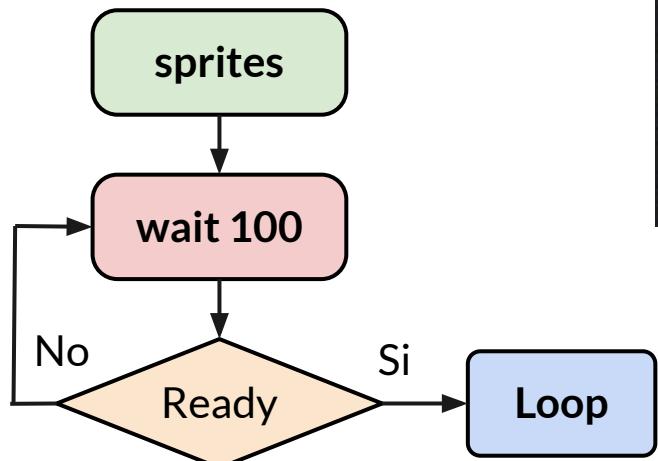


```
57 // el main loop del juego
58 let lastTime = 0;
59 function loop(currentTime) {
60     let dlt = (currentTime - lastTime) / 1000;
61     lastTime = currentTime;
62     if (!Number.isFinite(dlt)) dlt = 0;
63     // ejecutar todo usando el delta de tiempo
64     step(dlt);
65     draw();
66     // pedir que se re ejecute de nuevo
67     requestAnimationFrame(loop);
68 }
```

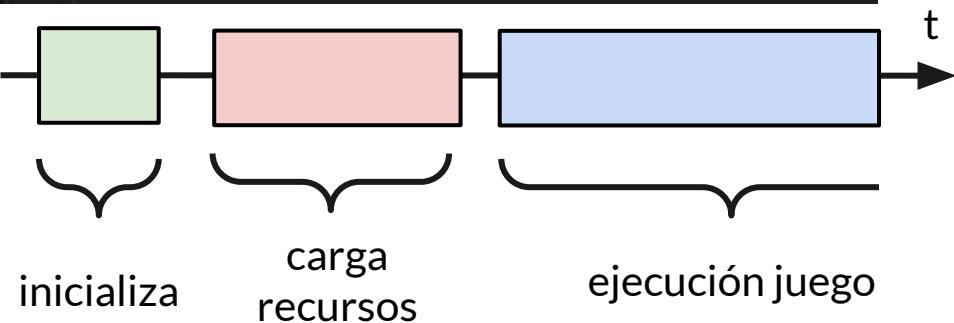


## juego.php

# Juego arranque

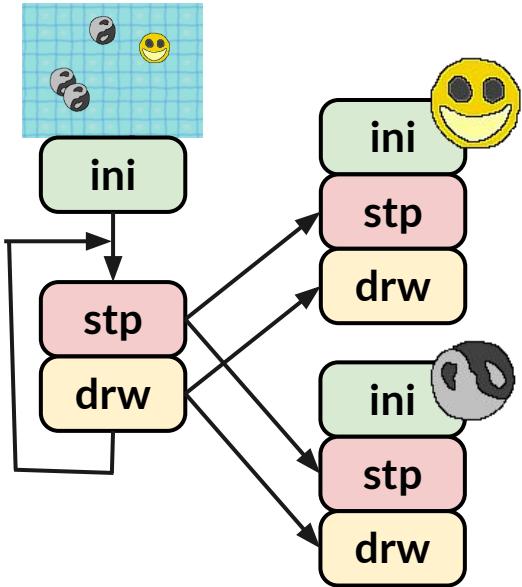


```
102 const sprites = new Sprites();
103 sprites.loadImg("player", 64, 64);
104 sprites.loadImg("rock", 64, 64);
105 sprites.loadImg("fondo", 96, 96);
106 setTimeout(arranque, 100);
107 function arranque() {
108     if (sprites.getReady()) {
109         loop();
110     }
111     else {
112         setTimeout(arranque, 100);
113     }
114 }
```



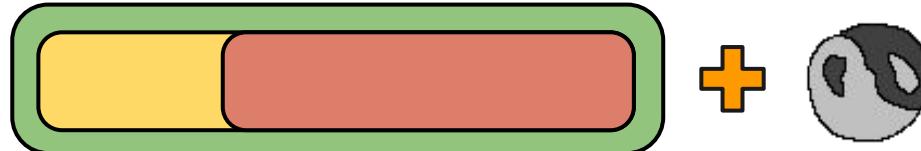
juego.php

## Juego step



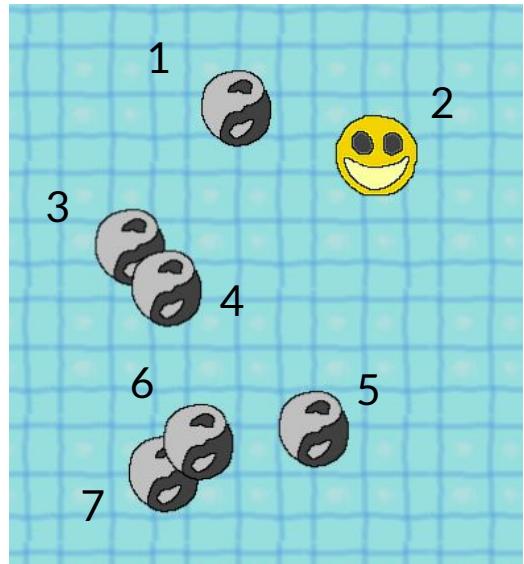
```
70 // se calcula la logica
71 function step(dlt) {
72     // ejecutar la logica de cada objeto
73     objetos.forEach(obj => obj.step(dlt));
74     // obtener al objeto Player y ver si vive
75     let ply = objetos.filter(obj => obj instanceof Player)[0];
76     if (ply.vida != 0) {
77         // conteo de puntos
78         puntaje += dlt;
79         // hacer aparecer mas Rock enemigos
80         respawn_rock.reloj -= dlt;
81         if (respawn_rock.reloj <= 0) {
82             respawn_rock.reloj += respawn_rock.espera +
83                 Math.random();
84             objetos.push(new Rock());
85         }
86     }
87 }
```

4.5 ± 0.5      reloj      0

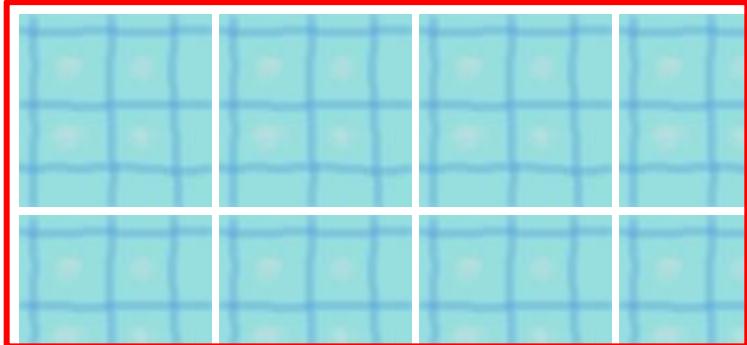


juego.php

## Juego draw

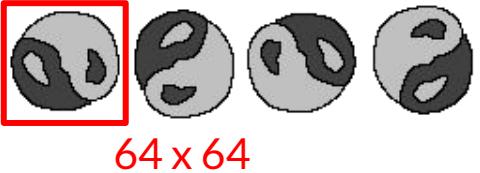
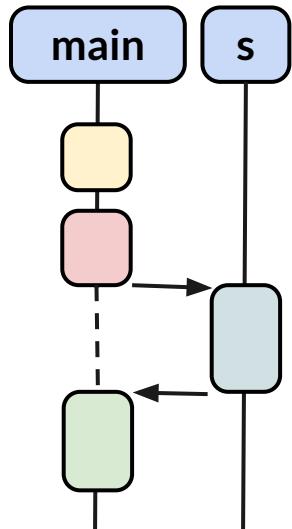


```
89 // se dibuja todo
90 function draw() {
91     // dibujar todo el suelo
92     sprites.drawTiled(ctx, "fondo", width, height);
93     // ordenar en Y todos los objetos
94     objetos.sort((a, b) => a.depth - b.depth);
95     // dibujar todos los objetos
96     objetos.forEach(obj => obj.draw());
97     // dibujar la interfaz grafica
98     Sprites.drawTexto(ctx, Math.floor(puntaje),
99                         {x: width / 2, y: 0});
100 }
```



## Sprites.js

# Sprites load



```
1 class Sprites {
2     constructor() {
3         this.cargas = 0;
4         this.sprites = [];
5     }
6
7     loadImg(filename, width, height) {
8         let img = new Image();
9         img.src = "sprites/" + filename + ".png";
10        img.onload = () => {
11            this.cargas += 1;
12        };
13        this.sprites.push({
14            name: filename,
15            imagen: img,
16            width: width,
17            height: height
18        });
19    }
20
21    getReady() {
22        return this.cargas == this.sprites.length;
23    }
24 }
```

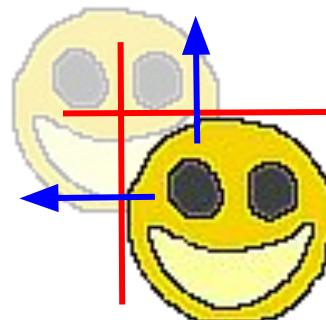
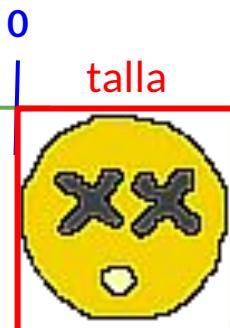
## Sprites.js

---

### Sprites draw

```
30     drawSprite(ctx, posicion, spritename, subimg) {
31         let spr = this.sprites.filter(s => s.name == spritename)[0];
32         ctx.drawImage(spr.imagen, // sprite
33             subimg * spr.width, 0, // pos sprite
34             spr.width, spr.height, // talla sprite
35             posicion.x - spr.width / 2, // posicion
36             posicion.y - spr.height / 2,
37             spr.width, spr.height); // escala
38     }
```

subimg

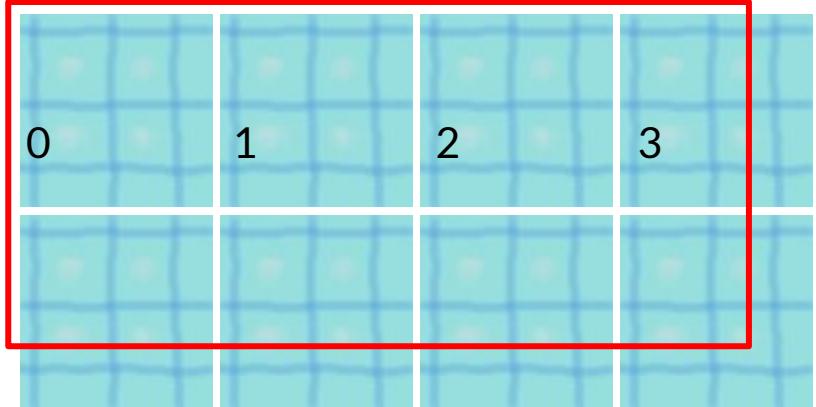


## Sprites.js

# Sprites tiled

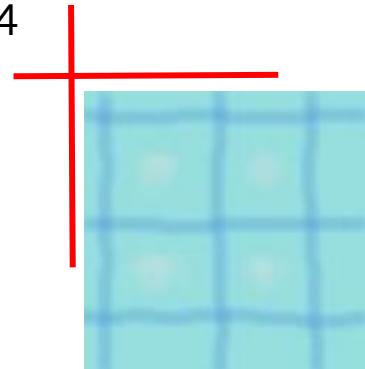
12.1

43.68



```
40     drawTiled(ctx, spritename, widthMax, heightMax) {  
41         let spr = this.sprites.filter(s => s.name == spritename)[0];  
42         let wMax = Math.ceil(widthMax / spr.width);  
43         let hMax = Math.ceil(heightMax / spr.height);  
44         for (let w = 0; w < wMax; w++) {  
45             for (let h = 0; h < hMax; h++) {  
46                 ctx.drawImage(spr.imagen, // sprite  
47                               w * spr.width, h * spr.height); // posicion  
48             }  
49         }  
50     }
```

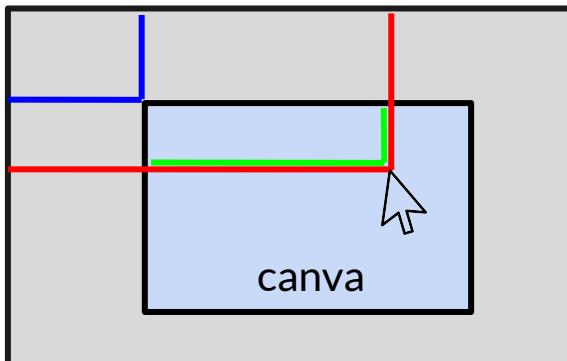
$$\frac{43.68}{12.1} = 3.6 \rightarrow = 4$$



## tools.js

# Mouse

left, top      clientX, clientY



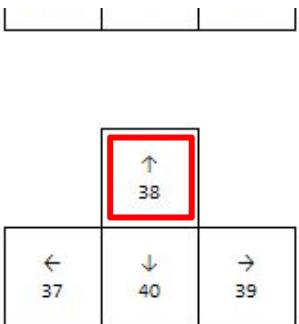
mouseX, mouseY

```
1 // posicion del mouse
2 let mouseData = {
3     x: 0, // posicion en el lienzo, sin afectarse por escalamiento
4     y: 0,
5     pulsado: false, // true si el clic izquierdo esta apretado
6 };
7
8 function newMouseListener(miCanvas) {
9     miCanvas.addEventListener("mousemove", (event) => {
10         let rect = miCanvas.getBoundingClientRect();
11         mouseData.x = Math.round(event.clientX - rect.left);
12         mouseData.y = Math.round(event.clientY - rect.top);
13     });
14     document.addEventListener("mousedown", (event) => {
15         if (event.button != 0) { return null; }
16         mouseData.pulsado = true;
17     });
18     document.addEventListener("mouseup", (event) => {
19         if (event.button != 0) { return null; }
20         mouseData.pulsado = false;
21     });
22 }
```

## tools.js

# Keyboard

		Enter 13
?	Shift 16	
Win 92	Menu 93	Ctrl 17

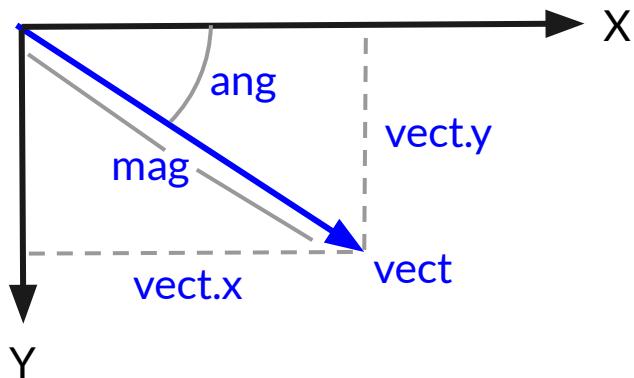


```
event.key ==  
"ArrowUp"  
event.code == 38
```

```
25 let keyData = {  
26   "w": false,  
27   "s": false,  
28   "a": false,  
29   "d": false  
30 }  
31  
32 function newKeyboardListener() {  
33   window.addEventListener('keydown', (event) => {  
34     if (event.key in keyData) {  
35       keyData[event.key] = true;  
36     }  
37   });  
38   window.addEventListener('keyup', (event) => {  
39     if (event.key in keyData) {  
40       keyData[event.key] = false;  
41     }  
42   });  
43 }
```

tools.js

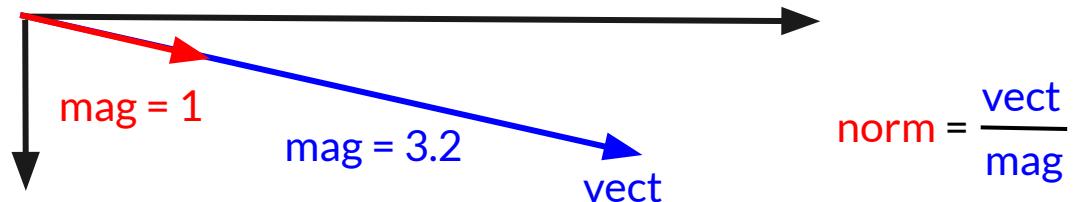
## Vectores 1



$$mag = \sqrt{vect.x^2 + vect.y^2}$$

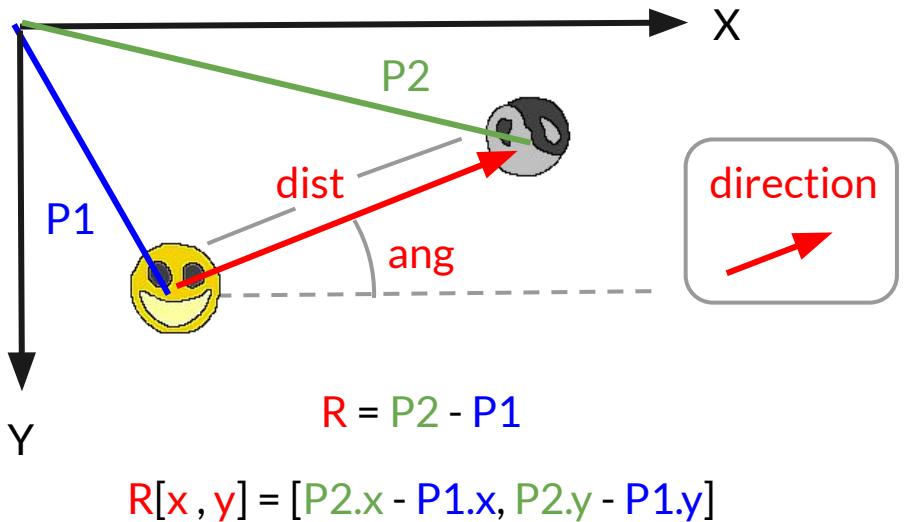
$$ang = \tan^{-1}(vect.y / vect.x)$$

```
47  function angulo(vect) {  
48      return Math.atan2(vect.y, vect.x);  
49  }  
51  function magnitud(vect) {  
52      return Math.sqrt(  
53          Math.pow(vect.x, 2) + Math.pow(vect.y, 2));  
54  }  
56  function normalize(vect) {  
57      let mag = magnitud(vect);  
58      if (mag == 0) return vect;  
59      return {  
60          x: vect.x / mag,  
61          y: vect.y / mag  
62      };  
63  }
```



tools.js

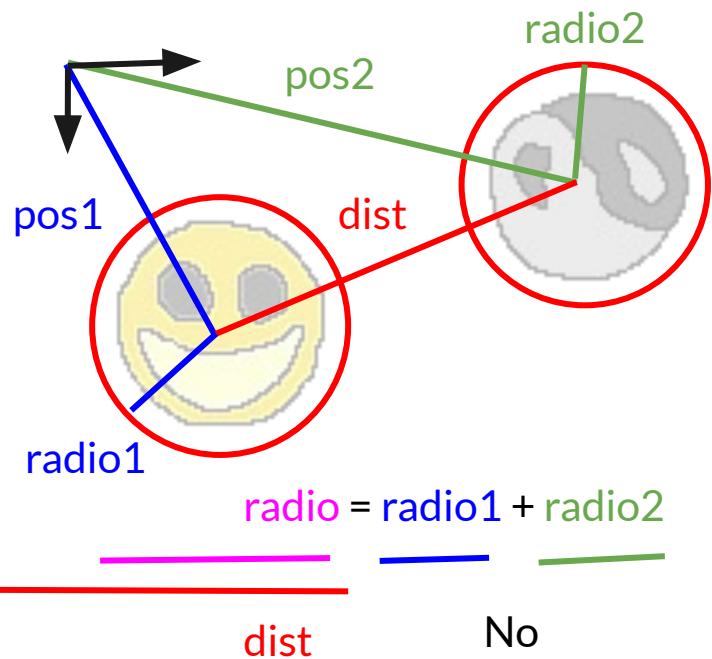
## Vectores 2



```
65     function pointDirection(pos1, pos2) {  
66         return normalize({  
67             x: pos2.x - pos1.x,  
68             y: pos2.y - pos1.y  
69         });  
70     }  
72     function pointAngle(pos1, pos2) {  
73         return angulo({  
74             x: pos2.x - pos1.x,  
75             y: pos2.y - pos1.y  
76         });  
77     }  
79     function pointDistance(pos1, pos2) {  
80         return magnitud({  
81             x: pos2.x - pos1.x,  
82             y: pos2.y - pos1.y  
83         });  
84     }
```

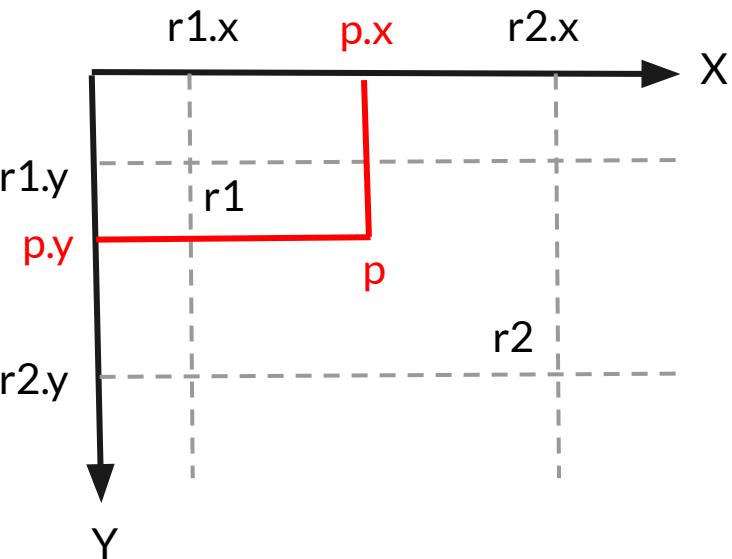
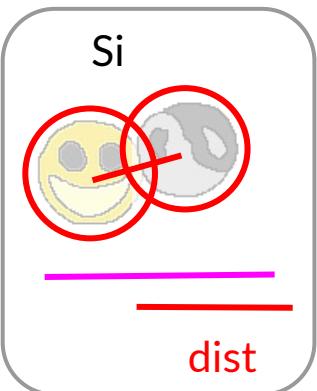
tools.js

## Vectores 3



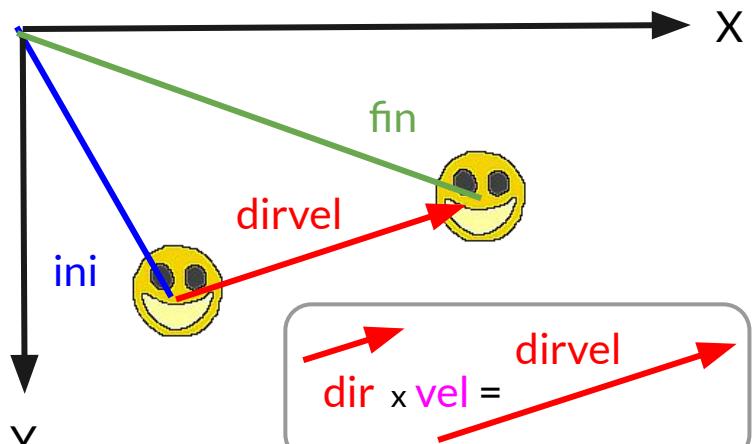
$$\text{radio} = \text{radio1} + \text{radio2}$$

```
86 function pointInCircle(pos1, pos2, radio) {  
87     let dist = pointDistance(pos1, pos2);  
88     return dist < radio;  
89 }  
91 function pointInRectangle(pos, rec1, rec2) {  
92     return pos.x > rec1.x && pos.x < rec2.x &&  
93         pos.y > rec1.y && pos.y < rec2.y;  
94 }
```



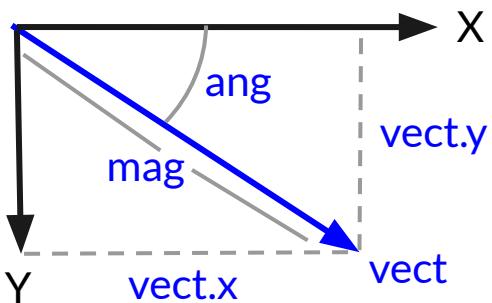
tools.js

## Vectores 4



```
96 function moveDirVel(pos, dir, vel) {  
97     return {  
98         x: pos.x + vel * dir.x,  
99         y: pos.y + vel * dir.y  
100    };  
101 }  
  
103 function moveAngVel(pos, angRad, vel) {  
104     return {  
105         x: pos.x + vel * Math.cos(angRad),  
106         y: pos.y + vel * Math.sin(angRad)  
107    };  
108 }
```

$$\text{dirvel}[x, y] = [\text{vel} \cos(\text{ang}), \text{vel} \sin(\text{ang})]$$



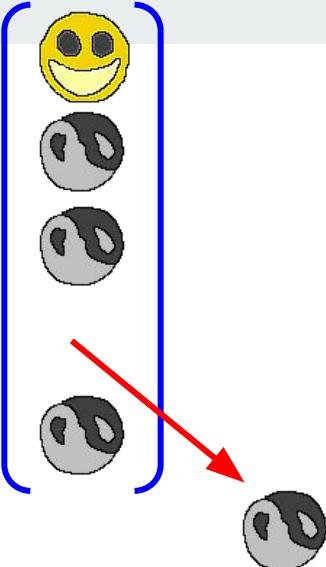
$$\text{vect}.x = \text{mag} \cos(\text{ang})$$

$$\text{vect}.y = \text{mag} \sin(\text{ang})$$

## Ente.js

### Ente

Ente
pos[x,y]
destroy()

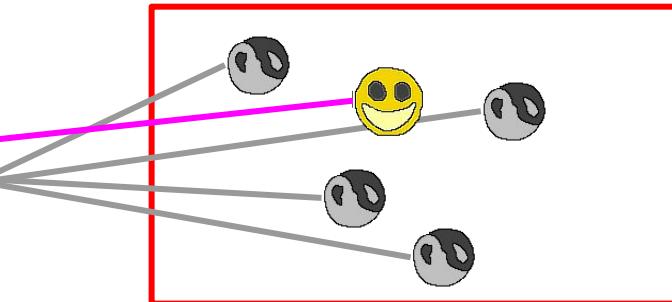


```
1 class Ente {  
2     constructor(posicion) {  
3         this.pos = {...posicion};  
4         this.depth = 0;  
5     }  
6     destroy() {  
7         let i = objetos.indexOf(this);  
8         if (i != -1) {  
9             objetos.splice(i, 1);  
10        }  
11    }  
12 }  
13 }
```

Player
vida
colisión()

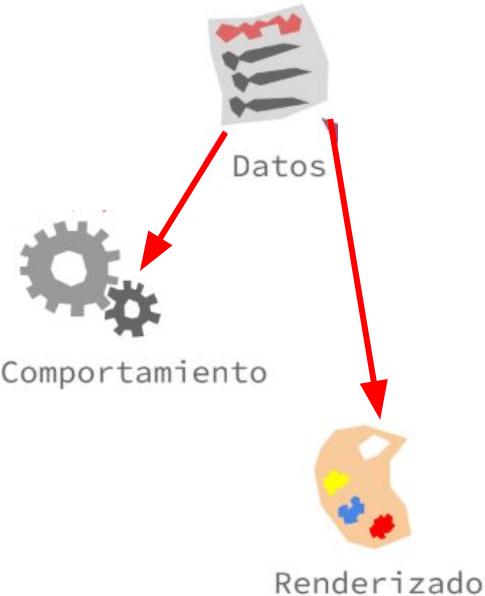
  

Rock
ángulo
reiniciar()



## Player.js

# Player MVC



```
step(dlt) {
    if (this.vida != 0) {
        // moverse con teclas
        let dir = modoMouse ?
            this.getDirMouse() : this.getDirKeys();
        this.pos = moveDirVel(this.pos, dir,
            Player.VELOCIDAD * dlt);
        this.lmites();
        // calcular colision
        let otro = this.isColision();
        if (otro !== null) {
            this.vida--;
            this.sonar();
            otro.destroy();
        }
    }
}

draw() {
    sprites.drawSprite(ctx, this.pos, "player", this.vida);
}
```

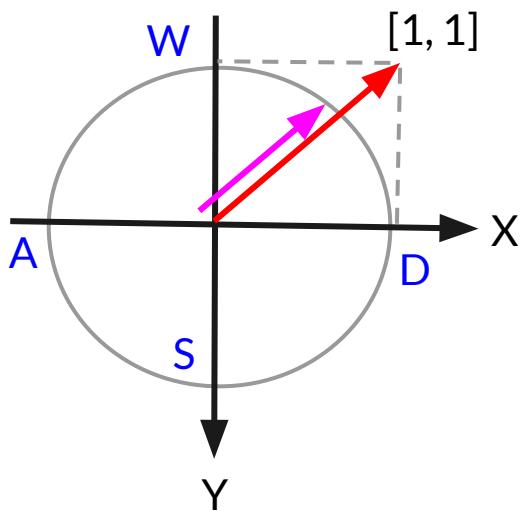
```
class Player extends Objeto {
    static VELOCIDAD = 400;
    static RADIO = 26;

    constructor(posicion) {
        super(posicion);
        this.depth = 1;
        this.vida = 4;
    }
}
```

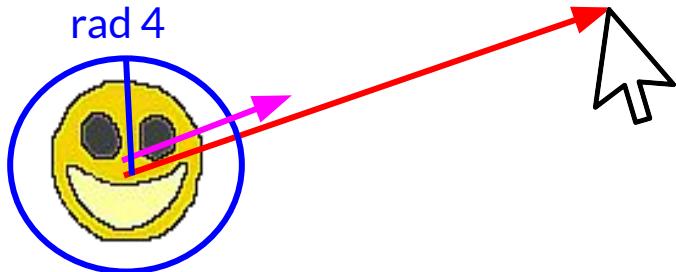


## Player.js

# Player manejo

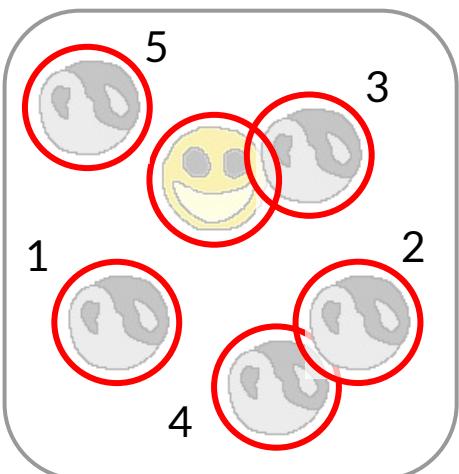


```
44 getDirKeys() {
45     let dir = {x:0, y:0};
46     if (keyData["w"]) dir.y--;
47     if (keyData["s"]) dir.y++;
48     if (keyData["a"]) dir.x--;
49     if (keyData["d"]) dir.x++;
50     return normalize(dir);
51 }
53 getDirMouse() {
54     if (pointDistance(this.pos, mouseData) < 4) {
55         return {x: 0, y: 0};
56     }
57 }
58 }
```



## Player.js

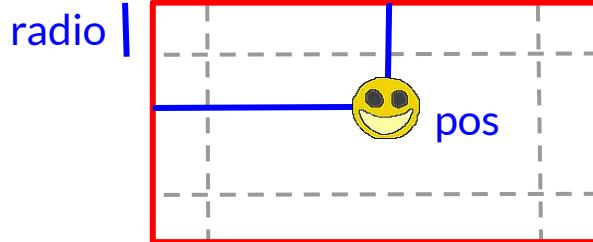
# Player colisión



- 1 - No
- 2 - No
- 3 - Si
- 4 -
- 5 -

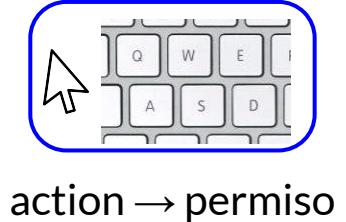
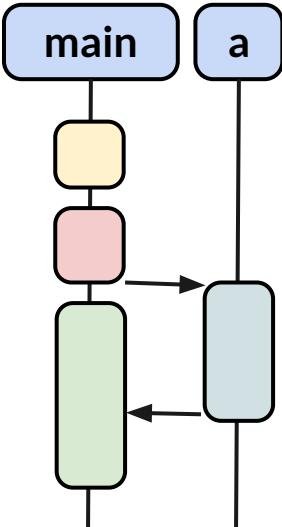
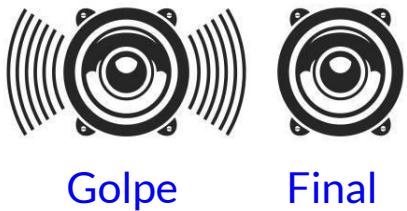
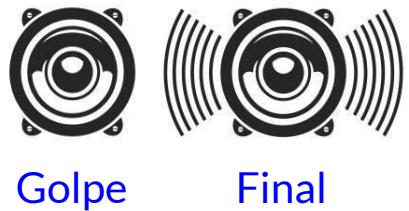
```
60     isColision() {  
61         for (let obj of objetos) {  
62             if (obj == this) { continue; }  
63             if (!obj.getActivo()) { continue; }  
64             if (pointInCircle(this.pos, obj.pos,  
65                 Player.RADIO + Rock.RADIO)) {  
66                 return obj;  
67             }  
68         }  
69         return null;  
70     }  
71     limites() {  
72         this.pos.x = Math.max(Player.RADIO,  
73             Math.min(width - Player.RADIO, this.pos.x));  
74         this.pos.y = Math.max(Player.RADIO,  
75             Math.min(height - Player.RADIO, this.pos.y));  
76     }  
77 }
```

M objetos  
N detectandose  
 $\text{Colis} = N * (M - 1)$   
 $\text{Optim} = \sum_{i=1}^N (M - i)$



## Player.js

# Player sonido



```
5 constructor(posicion) {  
6     super(posicion);  
7     this.depth = 1;  
8     this.vida = 4;  
9     this.sndGolpe = new Audio("sounds/golpe.wav");  
10    this.sndFinal = new Audio("sounds/final.wav");  
11}  
12  
13 sonar() {  
14     if (this.vida == 0) {  
15         this.sndGolpe.pause();  
16         this.sndFinal.currentTime = 0;  
17         this.sndFinal.play();  
18     }  
19     else {  
20         this.sndFinal.pause();  
21         this.sndGolpe.currentTime = 0;  
22         this.sndGolpe.play();  
23     }  
24 }
```

```
let otro = this.isColision();  
if (otro !== null) {  
    this.vida--;  
    this.sonar();  
    otro.destroy();  
}
```

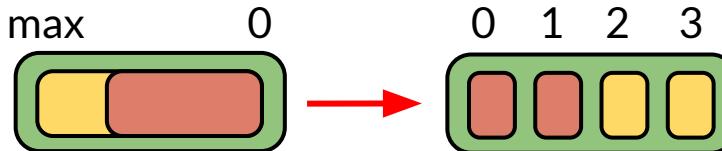
Rock.js

setTimeout()



sin dlt

## Rock animación



```
stepAnima(dlt) {  
    this.anima.reloj -= dlt;  
    while (this.anima.reloj <= 0) {  
        this.anima.reloj += this.anima.relojMax;  
        this.anima.paso++;  
        if (this.anima.paso > this.anima.pasoMax) {  
            this.anima.paso = 0;  
        }  
    }  
}
```

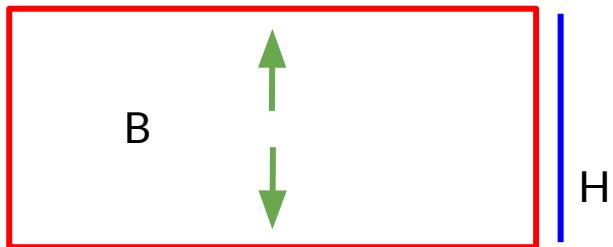
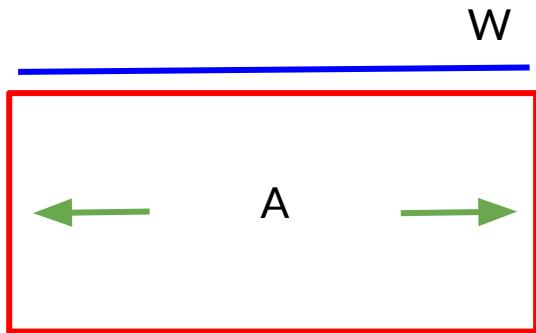
```
class Rock extends Objeto {  
    static VELOCIDAD = 300;  
    static RADIO = 10;  
    static RELOJ_INICIA = 2;  
    static ARCO_AZAR_RAD = Math.PI / 8;  
    constructor() {  
        super({x: 0, y: 0});  
        this.relojInicia = 0; // temporizador empezar a moverse  
        this.anguloRad = 0; // direccion de movimiento  
        this.anima = {  
            relojMax: 0.1, // tiempo que dura un paso de animacion  
            reloj: 0, // contador para hacer cambios  
            pasoMax: 3, // numero total de subimagenes - 1  
            paso: 0 // subimagen actual de la animacion  
        }  
        this.reiniciar();  
        this.relojInicia = Math.random() * Rock.RELOJ_INICIA;  
        this.sndTiro = new Audio("sounds/tiro.wav");  
    }  
}
```



```
draw() {  
    sprites.drawSprite(ctx, this.pos, "rock", this.anima.paso);  
}
```

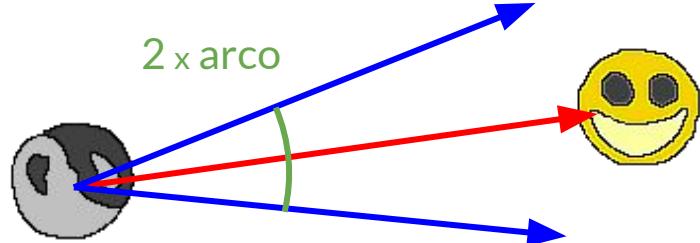
## Rock.js

# Rock inicializa



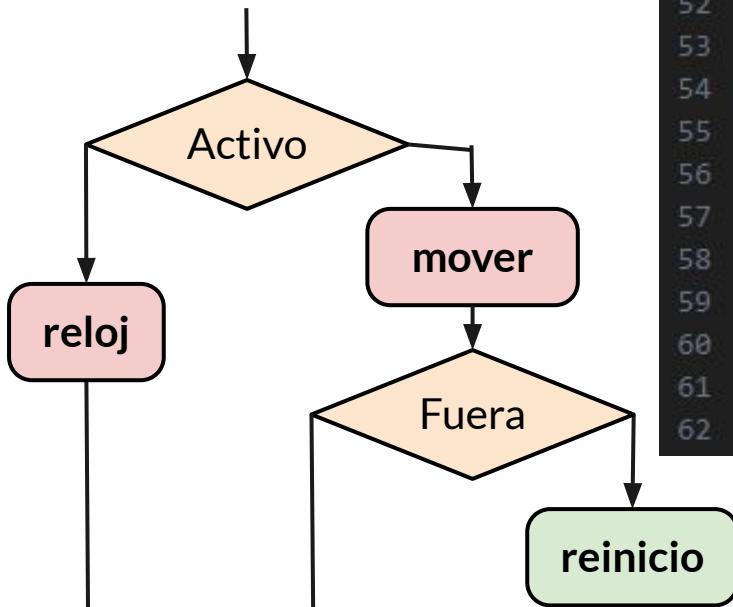
```
22     reiniciar() {
23         this.posBordesAzar();
24         this.relojInicia = Rock.RELOJ_INICIA;
25         let ply = objetos.filter(obj => obj instanceof Player)[0];
26         this.anguloRad = pointAngle(this.pos, ply.pos) +
27             (Math.random() * 2 - 1) * Rock.ARCO_AZAR_RAD;
28     }
29
30     posBordesAzar() {
31         if (Math.random() < width / (width + height)) { // width
32             this.pos.x = Math.random() * width;
33             this.pos.y = Math.random() < 0.5 ? 0 : height;
34         } else { // height
35             this.pos.x = Math.random() < 0.5 ? 0 : width;
36             this.pos.y = Math.random() * height;
37         }
38     }
39 }
```

si  $W = H$ :  
50% A-B  
`random()`  
select 50%



## Rock.js

# Rock movimiento

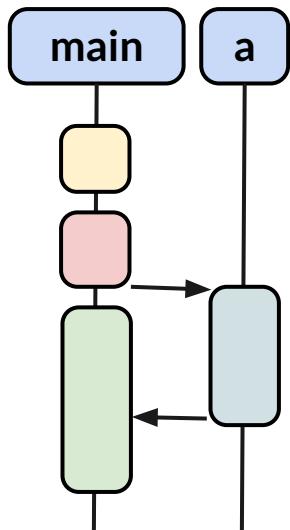


```
41     getActivo() {
42         return this.relojInicia == 0;
43     }
44
45     step(dlt) {
46         this.stepAnima(dlt);
47         if (this.getActivo()) {
48             this.pos = moveAngVel(this.pos,
49                 this.anguloRad, Rock.VELOCIDAD * dlt);
50             if (!pointInRectangle(this.pos,
51                 {x: 0, y: 0}, {x: width, y: height})) {
52                 this.reiniciar();
53             }
54         } else {
55             this.relojInicia = Math.max(0, this.relojInicia - dlt);
56             if (this.relojInicia == 0) {
57                 this.sndTiro.currentTime = 0;
58                 this.sndTiro.play();
59             }
60         }
61     }
62 }
```

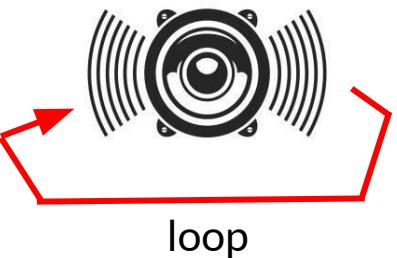


## juego.php

# Música



```
20 const musica = new Audio("sounds/musica.wav");
21 musica.volume = 0.333;
22 musica.loop = true;
23 musica.addEventListener("canplaythrough", () => {
24     setTimeout(miMusica, 20);
25 });
26 function miMusica() {
27     let tecla = Object.values(keyData).some(key => key);
28     if (mouseData.pulsado || tecla) {
29         musica.play();
30     } else {
31         setTimeout(miMusica, 20);
32     }
33 }
34 }
```



action → permiso



## Conclusiones

Además de la enseñanza, el presente trabajo es en sí una plantilla o librería, útil para hacer sencillos videojuegos web, donde no se precise de grandes librerías especializadas, como plantilla está sujeta a mejoras. En definitiva, el videojuego aquí expuesto cumple a cabalidad con su gameplay y demuestra lo fácil que es hacer un mini-juego con JS.

Como trabajo futuro se plantea hacer un sistema de login con bases de datos, para poder registrar un puntaje global, que el sistema muestre al menos los 10 mejores puntajes de usuarios, el mejor puntaje del usuario actual y su puesto en el ranking global, así como poder ver si fué finalizado con teclado o mouse.

Contacto:

[ojorcio@gmail.com](mailto:ojorcio@gmail.com)