Crear Videojuegos y Simulaciones con JS

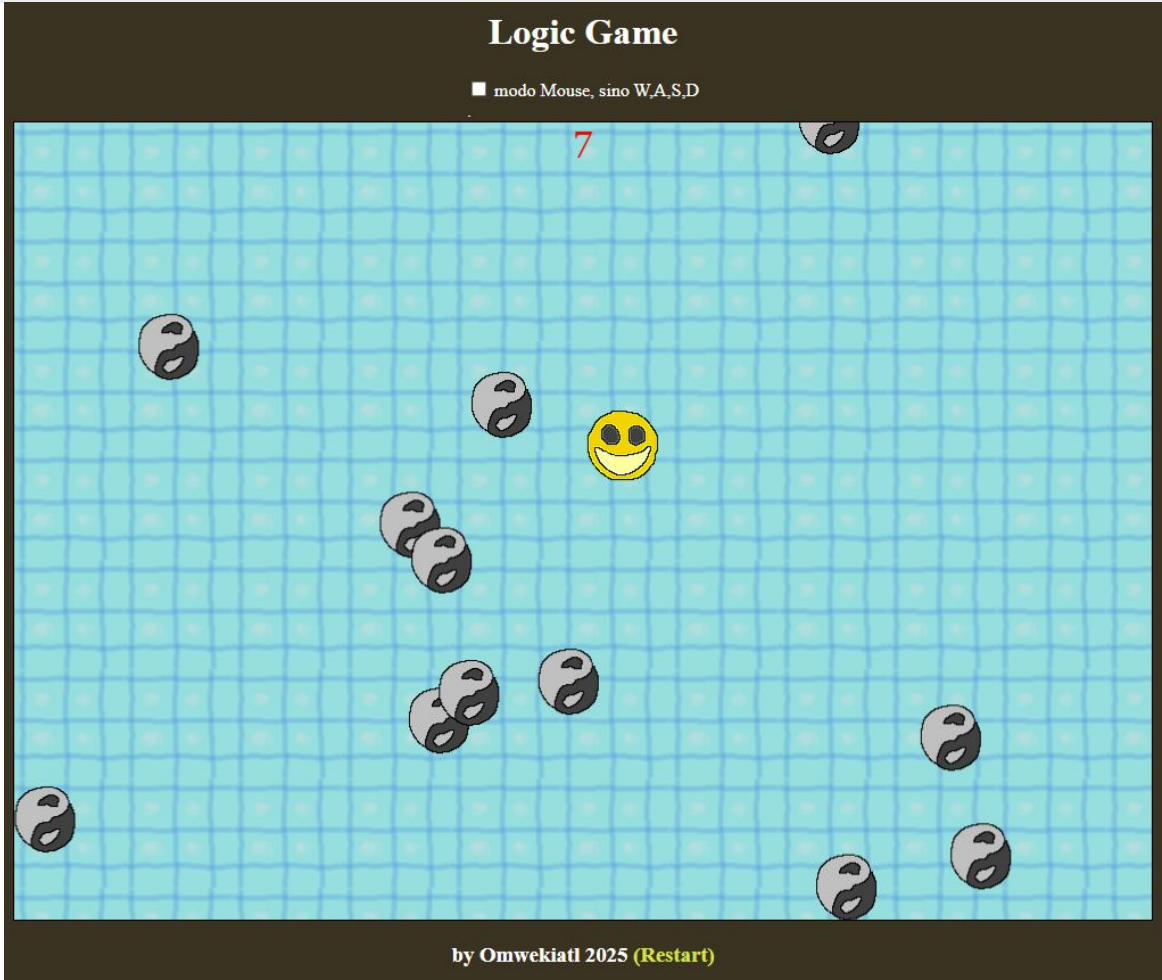
Omwekiatl - 2025

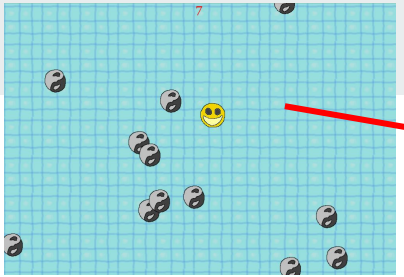
https://omwekiatl.xyz/logic_simula

https://github.com/OmarsaurioJordan/PracticasWeb/tree/main/logic_simula

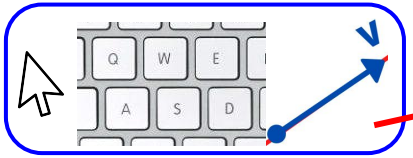
Gameplay

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



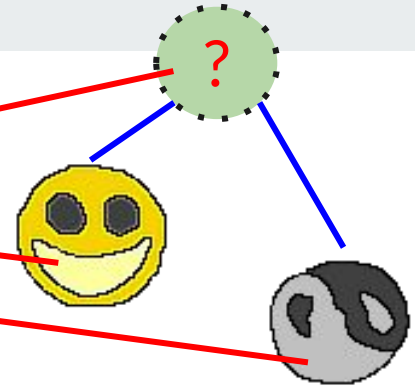


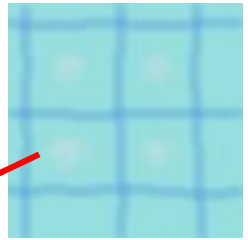
Estructura



LOGIC_SI...

- scripts
  - juego.js
  - Objeto.js
  - Player.js
  - Rock.js
  - Sprites.js
  - tools.js
- sounds
  - final.wav
  - golpe.wav
  - musica.wav
  - tiro.wav
- sprites
  - fondo.png
  - player.png
  - rock.png
- index.php



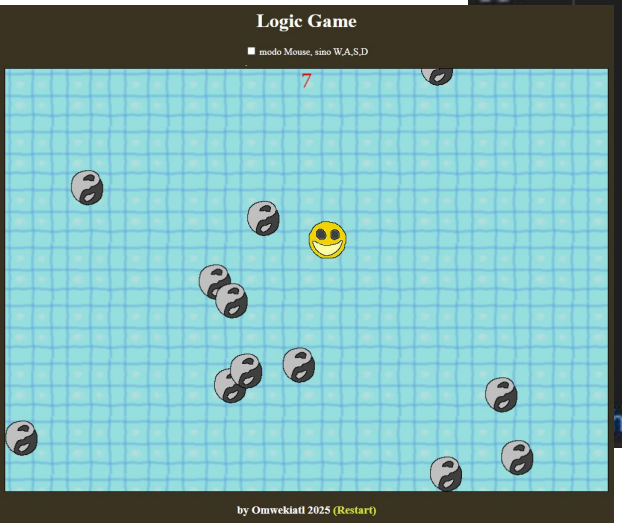


index.php

HTML canvas

```php
<?php
    $modoMouse = isset($_GET['modoMouse']) ? " checked" : "";
?>
```

```
    <body>
        <h1>Logic Game</h1>
        <div>
            <input type="checkbox" id="modoMouse" <?php echo $modoMouse; ?>>
            <label for="modoMouse"> modo Mouse, sino W,A,S,D<br>.</label>
        </div>
        <canvas id="lienzo" width="1000" height="700"
            style="border:1px solid black;"></canvas>
        <h3>by Omwekiatl 2025 <a href="" id="restart">(Restart)</a></h3>
    </body>
    <script src="scripts/tools.js" defer></script>
    <script src="scripts/Sprites.js" defer></script>
    <script src="scripts/Objeto.js" defer></script>
    <script src="scripts/Player.js" defer></script>
    <script src="scripts/Rock.js" defer></script>
    <script src="scripts/juego.js" defer></script>
ntml>
```



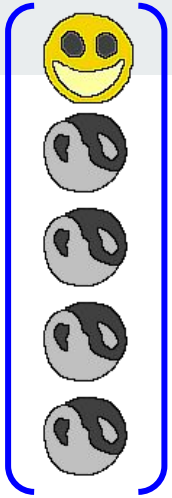
juego.php

Juego inicializa 1

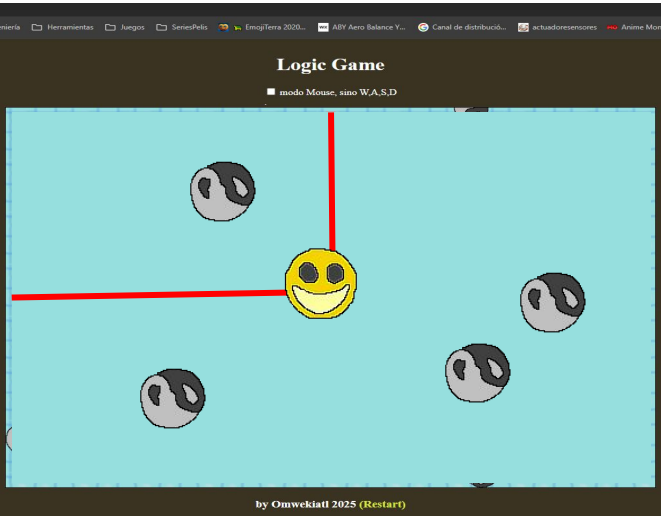
```
// obtener informacion del lienzo
const canvas = document.getElementById("lienzo");
const ctx = canvas.getContext("2d");
const width = canvas.width;
const height = canvas.height;

// lectura de comandos desde HTML
let modoMouse = document.getElementById("modoMouse").checked;
document.getElementById("modoMouse").addEventListener("change", (event) => {
    modoMouse = event.target.checked;
    document.getElementById("restart").href = modoMouse ?
        "index.php?modoMouse=1" : "index.php";
});
// activar rutinas de lectura de comandos
newMouseListener(canvas);
newKeyboardListener();
```

juego.php

Juego inicializa 2

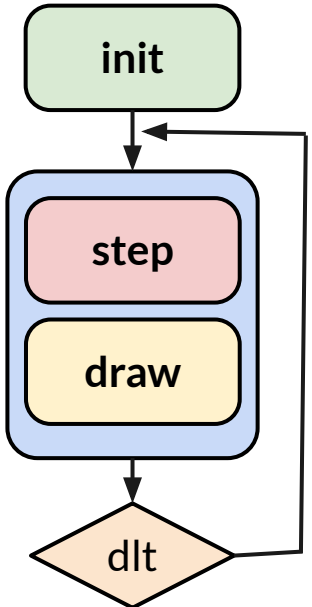


```
38  // estructuras de datos de la simulacion
39  let objetos = []; // contiene todos los objetos del juego
40  let puntaje = 0; // segundos sobreviviendo
41  let respawn_rock = {
42      reloj: 0, // segundos para crear nuevo Rock
43      espera: 4
44  };
45
46  // crear al Player
47  objetos.push(new 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  }
```

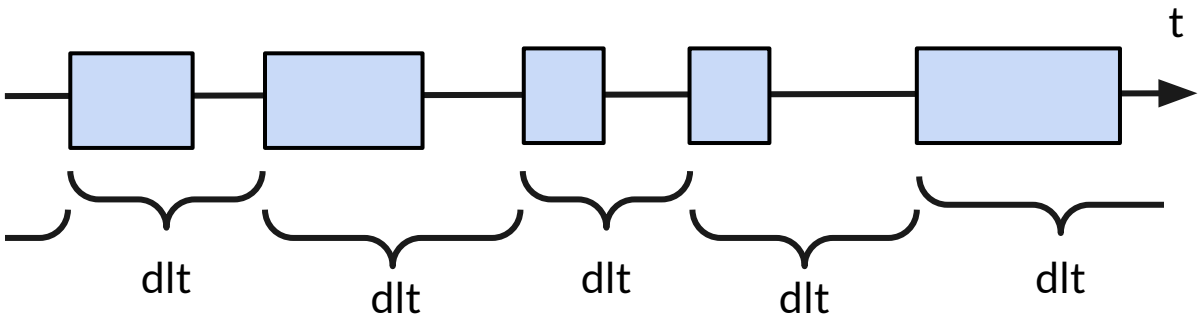


juego.php

Juego main loop



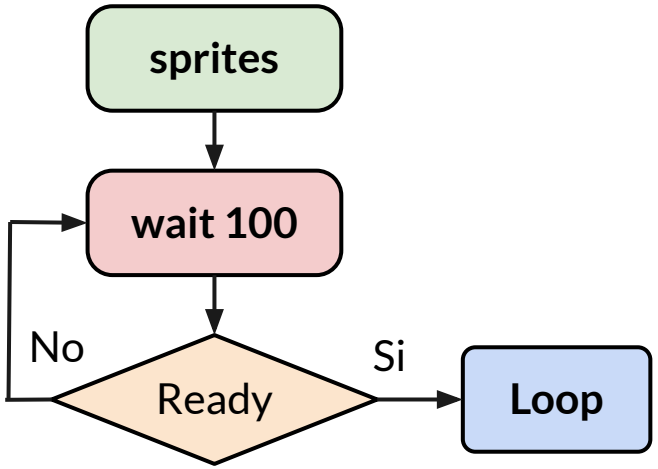

```
57  // el main loop del juego
58  let lastTime = 0;
59  function loop(currentTime) {
60      let dlt = (currentTime - lastTime) / 1000;
61      lastTime = currentTime;
62      if (!Number.isFinite(dlt)) dlt = 0;
63      // ejecutar todo usando el delta de tiempo
64      step(dlt);
65      draw();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

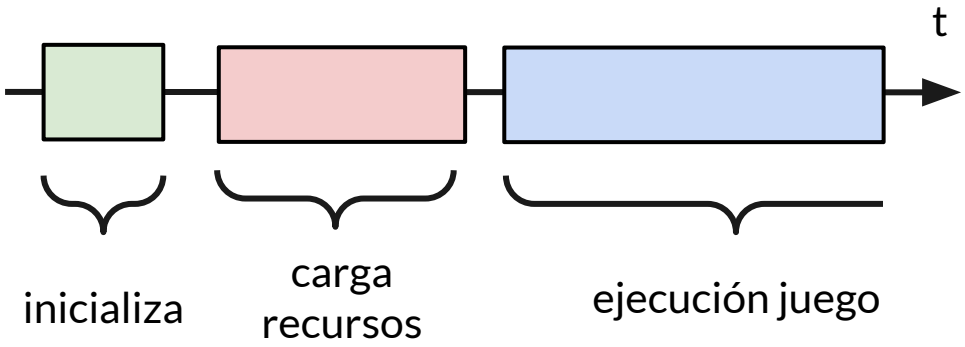


juego.php

Juego arranque

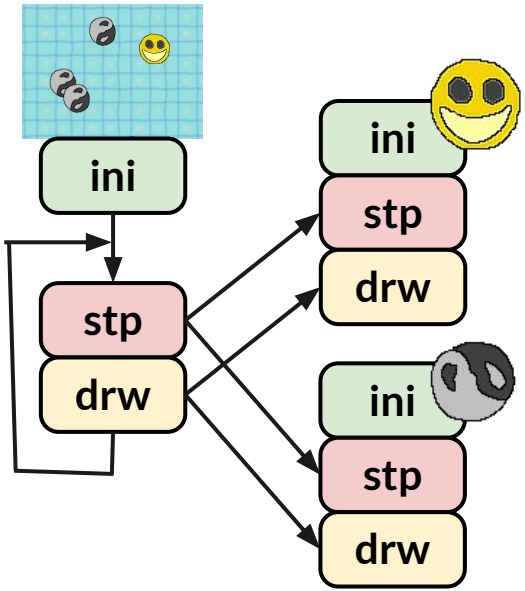
```
// iniciar el loop cuando los sprites carguen
const sprites = new Sprites();
setTimeout(arranque, 100);
function arranque() {
    if (sprites.getReady()) {
        loop();
    }
    else {
        setTimeout(arranque, 100);
    }
}
```





juego.php

Juego step

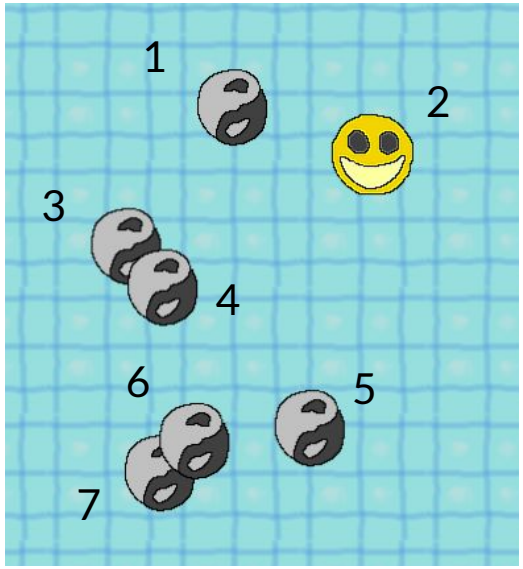


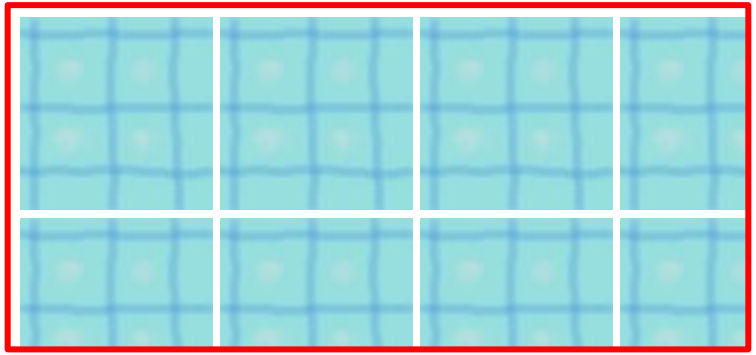

```
70  // se calcula la logica
71  function step(dlt) {
72      // ejecutar la logica de cada objeto
73      objetos.forEach(obj => obj.step(dlt));
74      // obtener al objeto Player y ver si vive
75      let ply = objetos.filter(obj => obj instanceof Player)[0];
76      if (ply.vida != 0) {
77          // 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  }
```

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 }
```



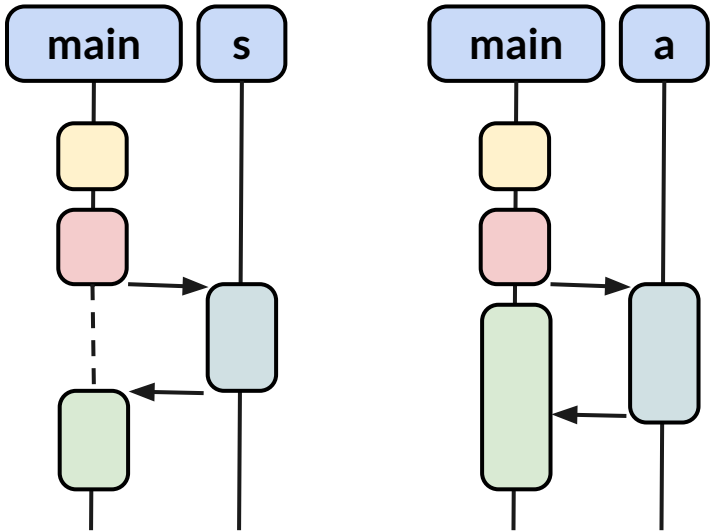


https://omwekiatl.xyz/jueguin

Sprites.js

Sprites load

64 x 64

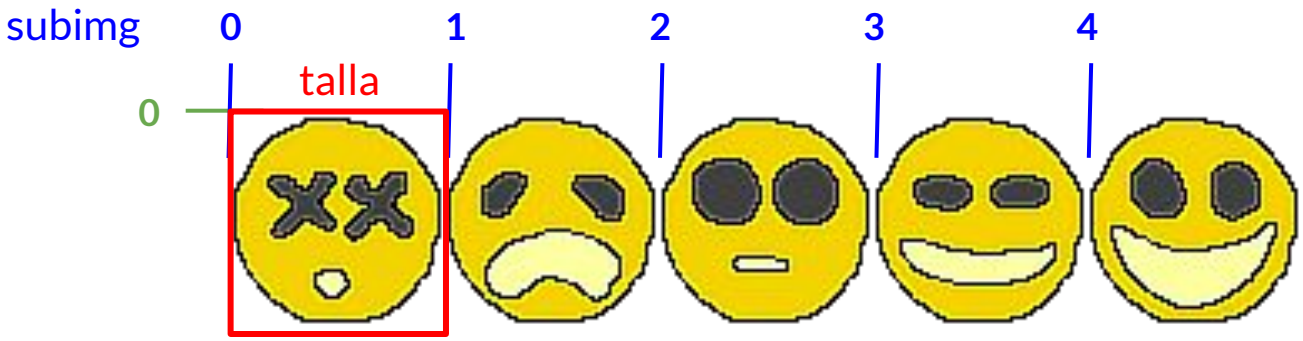


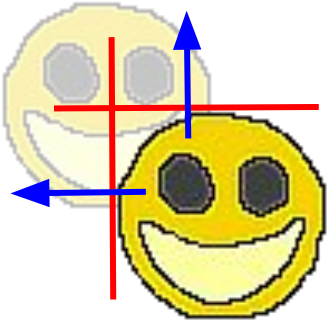

```
1   class Sprites {
3       constructor() {
4           this.cargas = 0;
5           this.sprites = [
6               this.loadImg("player", 64, 64),
7               this.loadImg("rock", 64, 64),
8               this.loadImg("fondo", 96, 96)
9           ];
10      }
12      loadImg(filename, width, height) {
13          let img = new Image();
14          img.src = "sprites/" + filename + ".png";
15          img.onload = () => {
16              this.cargas += 1;
17          };
18          return {
19              name: filename,
20              imagen: img,
21              width: width,
22              height: height
23          };
24      }
26      getReady() {
27          return this.cargas == this.sprites.length;
28      }
```

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      }
```

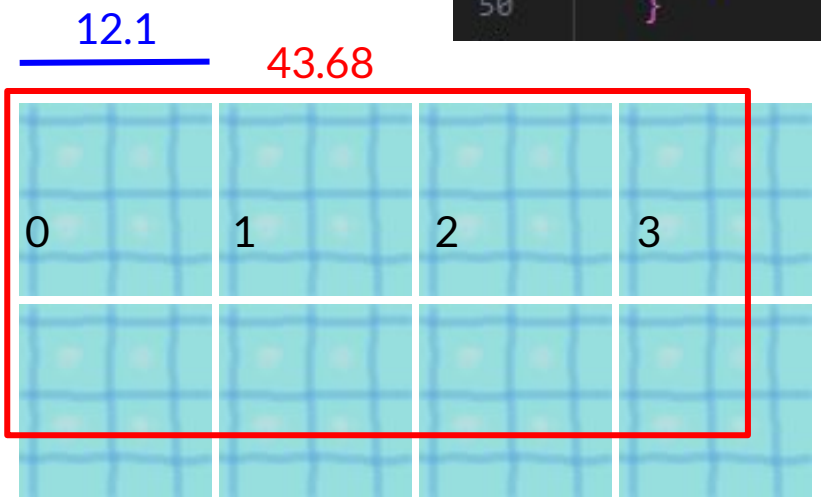




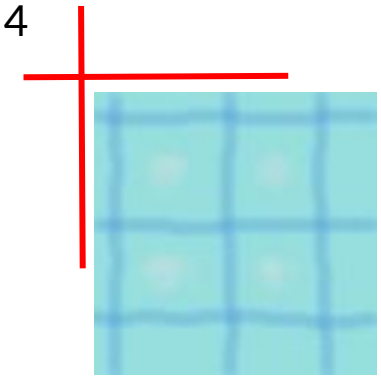
Sprites.js

Sprites tiled

```
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      }
```



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

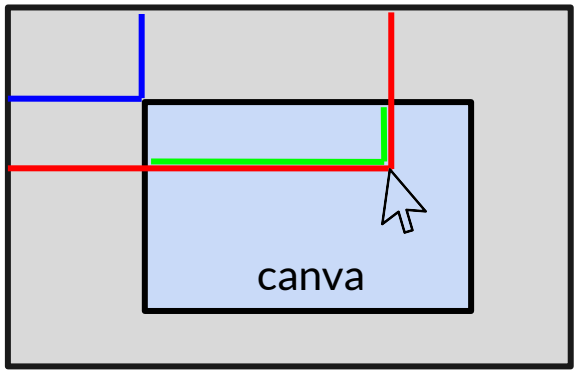


tools.js

Mouse

left, top

clientX, clientY

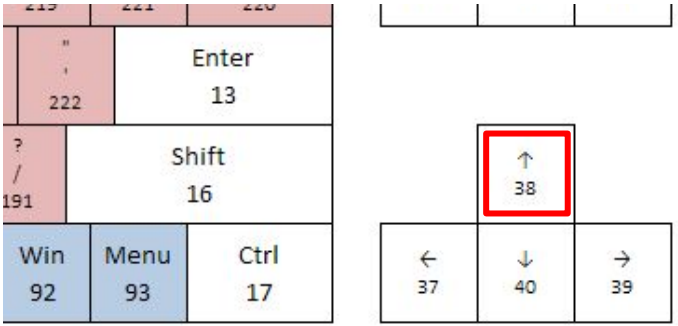


mouseX, mouseY

```
let mouseData = {
    x: 0, // posicion en el lienzo, sin afectarse por escalami
    y: 0,
    pulsado: false, // true el clic izquierdo esta apretado
};
function newMouseListener(miCanvas) {
    miCanvas.addEventListener("mousemove", (event) => {
        let rect = miCanvas.getBoundingClientRect();
        mouseData.x = Math.round(event.clientX - rect.left);
        mouseData.y = Math.round(event.clientY - rect.top);
    });
    document.addEventListener("mousedown", (event) => {
        if (event.button != 0) { return null; }
        mouseData.pulsado = true;
    });
    document.addEventListener("mouseup", (event) => {
        if (event.button != 0) { return null; }
        mouseData.pulsado = false;
    });
}
```

tools.js

Keyboard

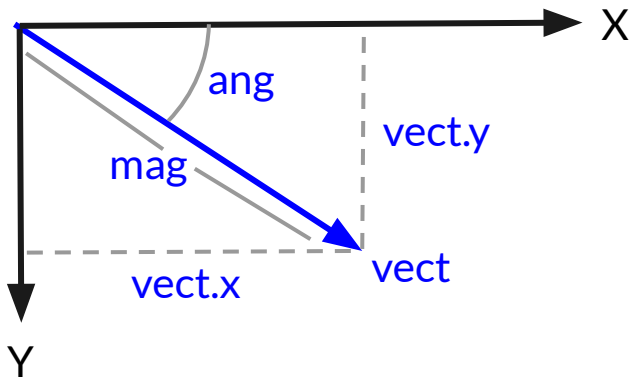


event.key ==
“ArrowUp”
event.code == 38

```
let keyData = {
    "w": false,
    "s": false,
    "a": false,
    "d": false
}
function newKeyboardListener() {
    window.addEventListener('keydown', (event) => {
        if (event.key in keyData) {
            keyData[event.key] = true;
        }
    });
    window.addEventListener('keyup', (event) => {
        if (event.key in keyData) {
            keyData[event.key] = false;
        }
    });
}
```

tools.js

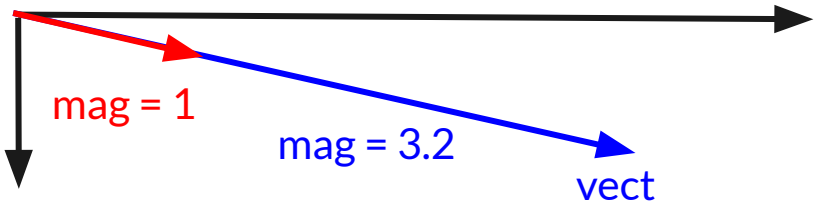
Vectores 1




```
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  }
```

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

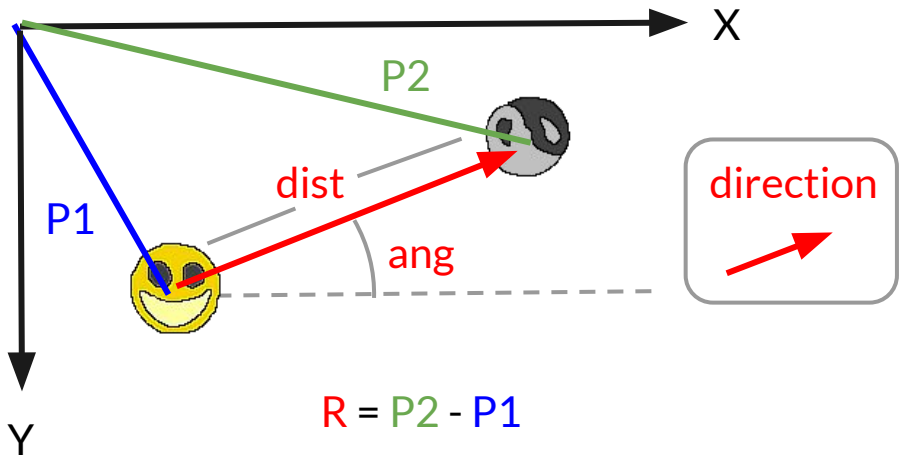
$$ang = \tan^{-1}(vect.y / vect.x)$$



$$norm = \frac{vect}{mag}$$

tools.js

Vectores 2



R = P2 - P1

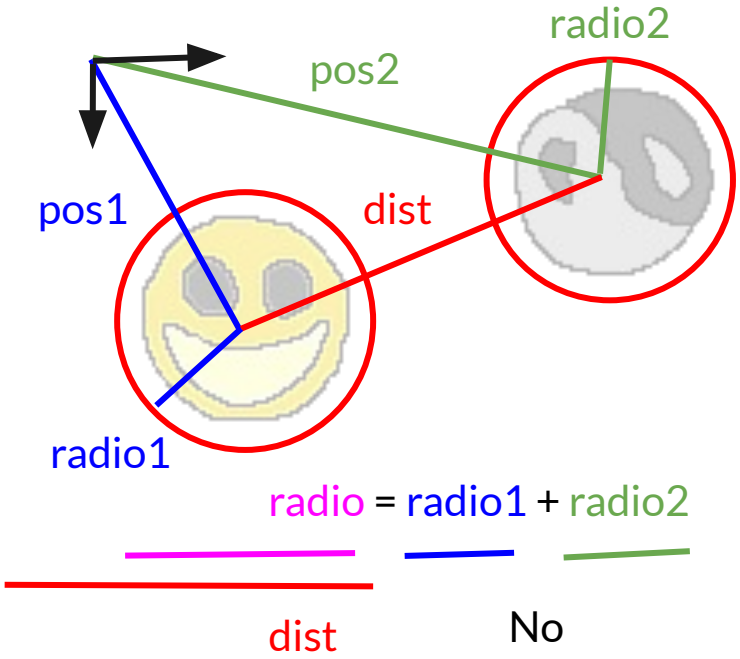
R[x , y] = [P2.x - P1.x, P2.y - P1.y]

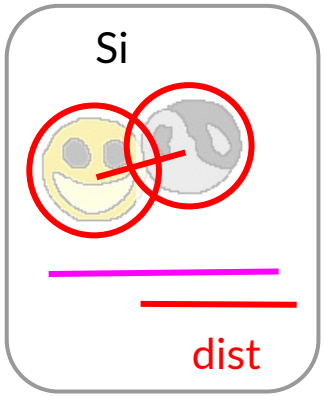
```
65  function pointDirection(pos1, pos2) {
66      return normalize({
67          x: pos2.x - pos1.x,
68          y: pos2.y - pos1.y
69      });
70  }
72  function pointAngle(pos1, pos2) {
73      return angulo({
74          x: pos2.x - pos1.x,
75          y: pos2.y - pos1.y
76      });
77  }
79 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

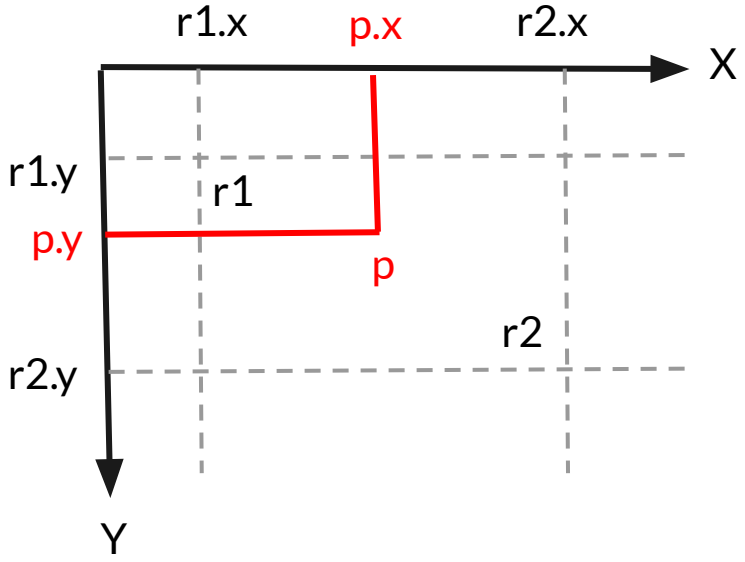
tools.js

Vectores 3

```
function pointInCircle(pos1, pos2, radio) {
    let dist = pointDistance(pos1, pos2);
    return dist < radio;
}

function pointInRectangle(pos, rec1, rec2) {
    return pos.x > rec1.x && pos.x < rec2.x &&
        pos.y > rec1.y && pos.y < rec2.y;
}
```

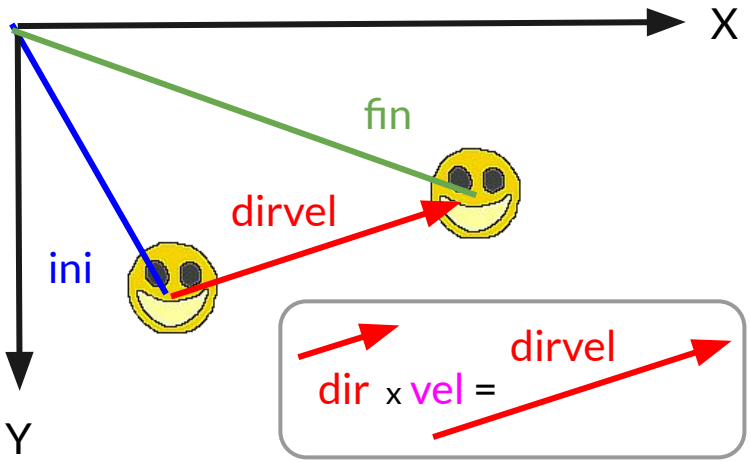






tools.js

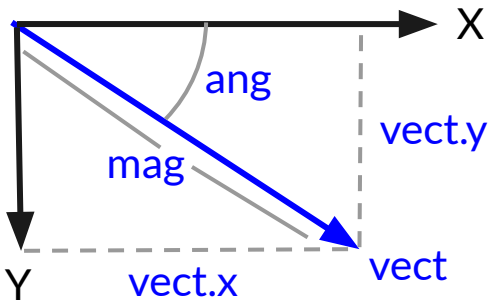
Vectores 4



fin = ini + dirvel

```
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 }
```

dirvel[x , y] = [vel x cos(ang), vel x sin(ang)]

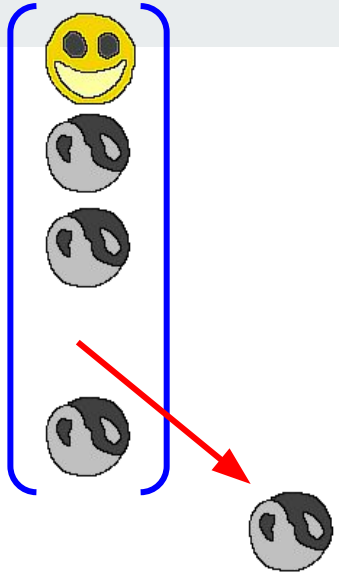


vect.x = mag x cos(ang)

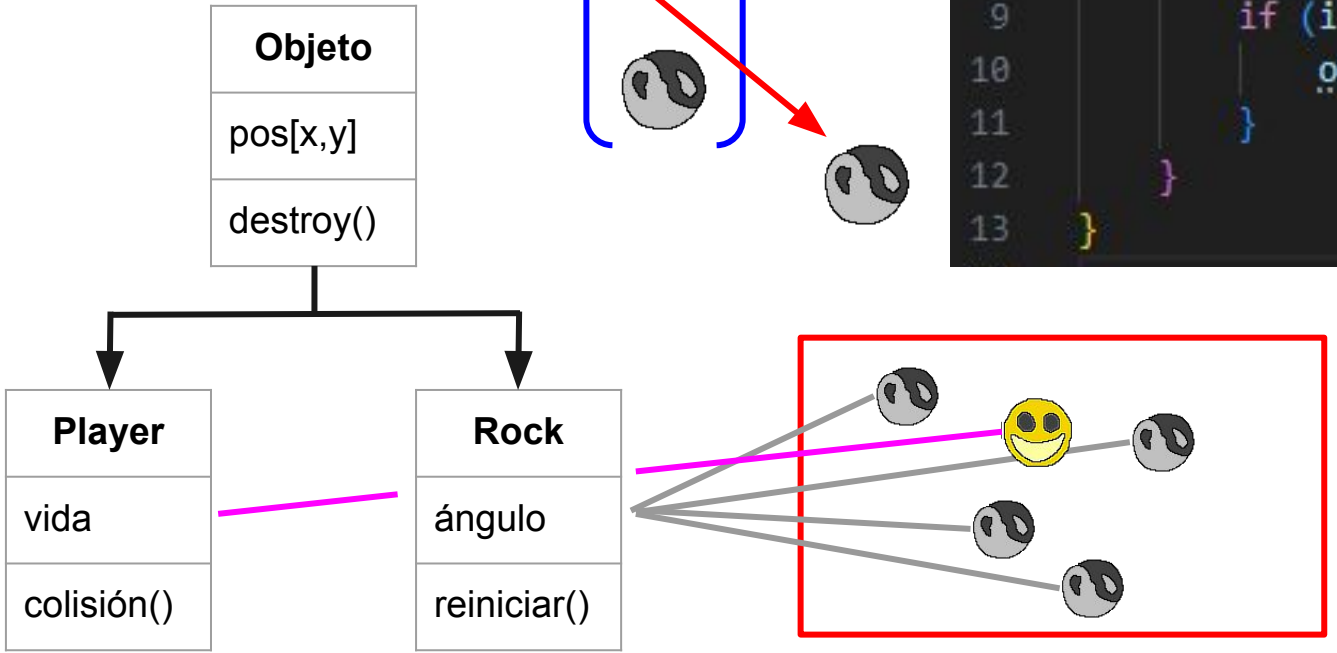
vect.y = mag x sin(ang)

Objeto.js

Objeto

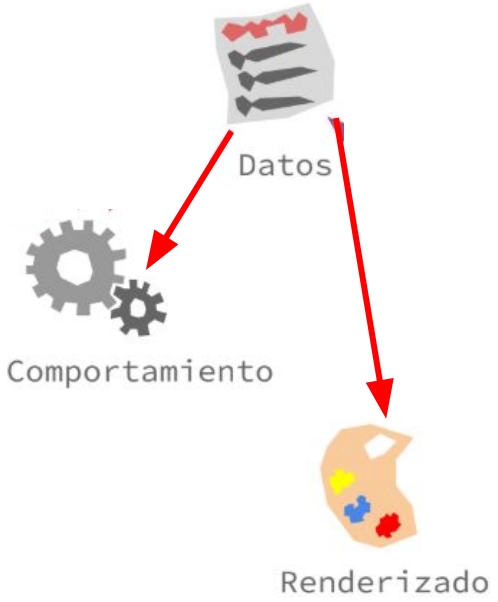


```
class Objeto {
    constructor(posicion) {
        this.pos = {...posicion};
        this.depth = 0;
    }
    destroy() {
        let i = objetos.indexOf(this);
        if (i != -1) {
            objetos.splice(i, 1);
        }
    }
}
```



Player.js

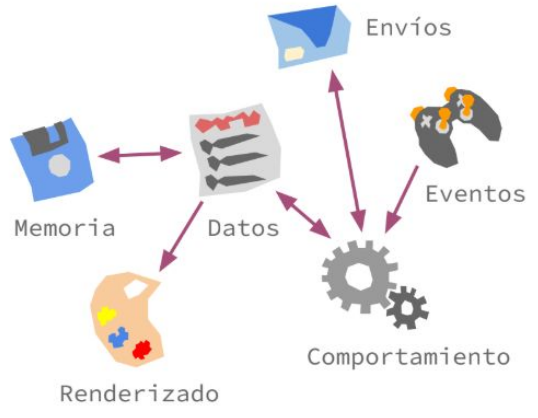
Player MVC




```
class Player extends Objeto {
    static VELOCIDAD = 400;
    static RADIO = 26;
    constructor(posicion) {
        super(posicion);
        this.depth = 1;
        this.vida = 4;
    }
```

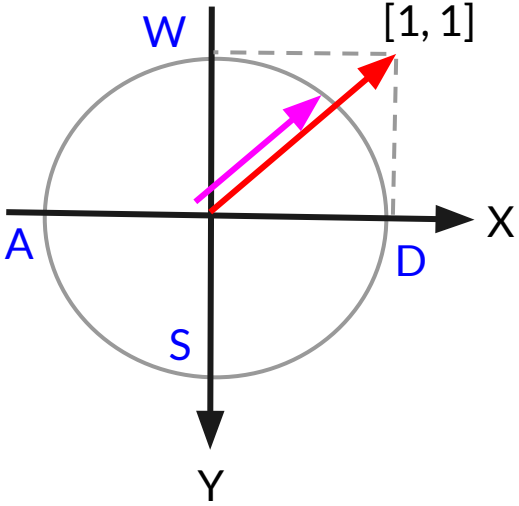
```
step(dlt) {
    if (this.vida != 0) {
        // moverse con teclas
        let dir = modoMouse ?
            this.getDirMouse() : this.getDirKeys();
        this.pos = moveDirVel(this.pos, dir,
            Player.VELOCIDAD * dlt);
        this.limites();
        // calcular colision
        let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

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

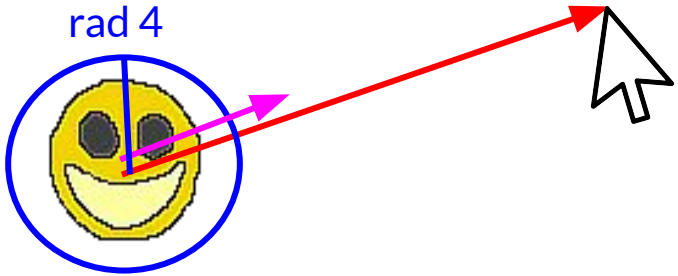


Player.js

Player manejo

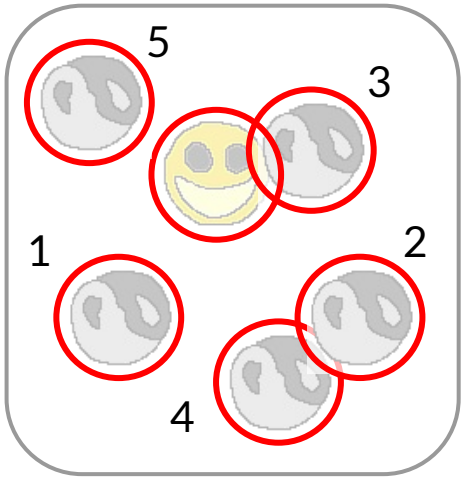



```
44    getDirKeys() {
45        let dir = {x:0, y:0};
46        if (keyData["w"]) dir.y--;
47        if (keyData["s"]) dir.y++;
48        if (keyData["a"]) dir.x--;
49        if (keyData["d"]) dir.x++;
50        return normalize(dir);
51    }
53    getDirMouse() {
54        if (pointDistance(this.pos,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



Player.js

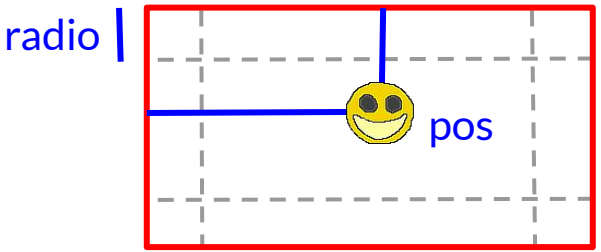
Player colisión



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      }
72      limites() {
73          this.pos.x = Math.max(Player.RADIO,
74              Math.min(width - Player.RADIO, this.pos.x));
75          this.pos.y = Math.max(Player.RADIO,
76              Math.min(height - Player.RADIO, this.pos.y));
77      }
```

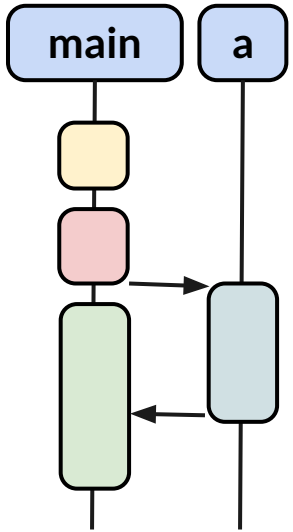
M objetos
N detectandose
Colis = N * (M - 1)
Optim = {i:1→N} ∑(M - i)



Player.js

Player sonido

Golpe Final




```
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   }
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   }
```

Golpe Final

action → permiso

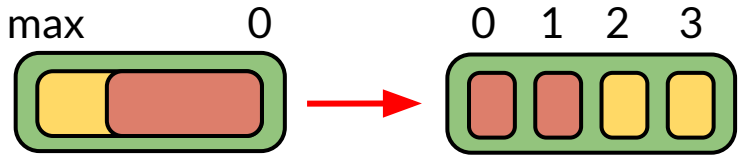
```
let otro = this.isColision();
if (otro !== null) {
    this.vida--;
    this.sonar();
    otro.destroy();
}
```

Rock.js

setTimeout()

sin dlt

Rock animación




```
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");
    }
```

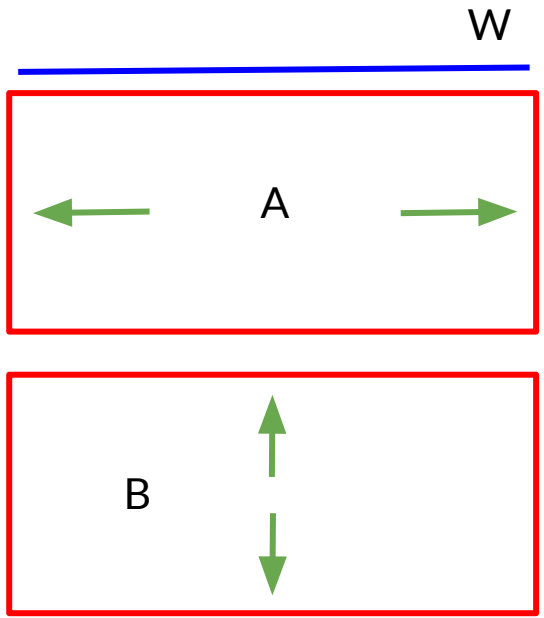
```
stepAnima(dlt) {
    this.anima.reloj -= dlt;
    while (this.anima.reloj <= 0) {
        this.anima.reloj += this.anima.relojMax;
        this.anima.paso++;
        if (this.anima.paso > this.anima.pasoMax) {
            this.anima.paso = 0;
        }
    }
}
```

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

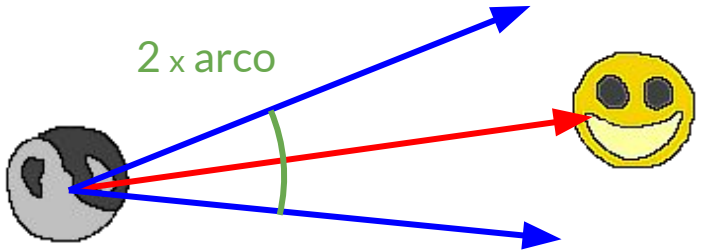
Rock.js

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      }
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          }
35          else { // height
36              this.pos.x = Math.random() < 0.5 ? 0 : width;
37              this.pos.y = Math.random() * height;
38          }
39      }
```

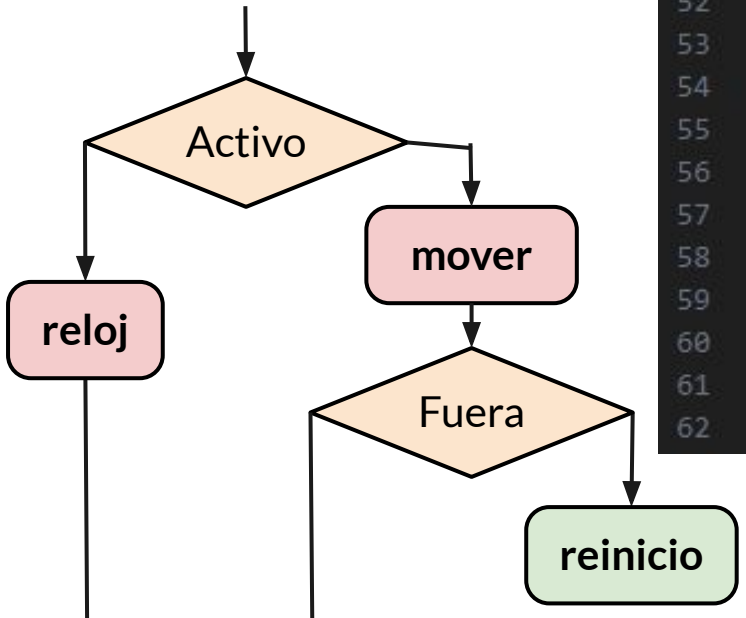


si W = H:
50% A-B
random()
select 50%



Rock.js

Rock movimiento




```
41      getActivo() {
42          return this.relojInicia == 0;
43      }
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          }
55          else {
56              this.relojInicia = Math.max(0, this.relojInicia - dlt);
57              if (this.relojInicia == 0) {
58                  this.sndTiro.currentTime = 0;
59                  this.sndTiro.play();
60              }
61          }
62      }
```

Fin