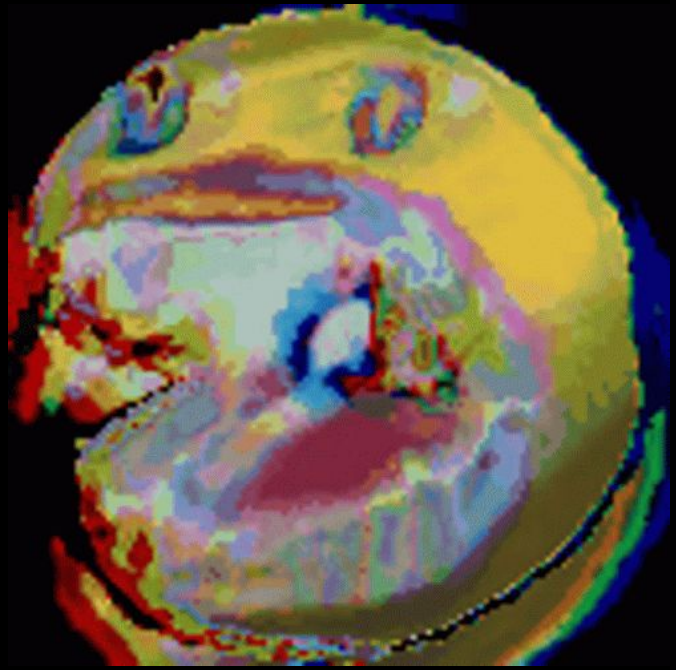


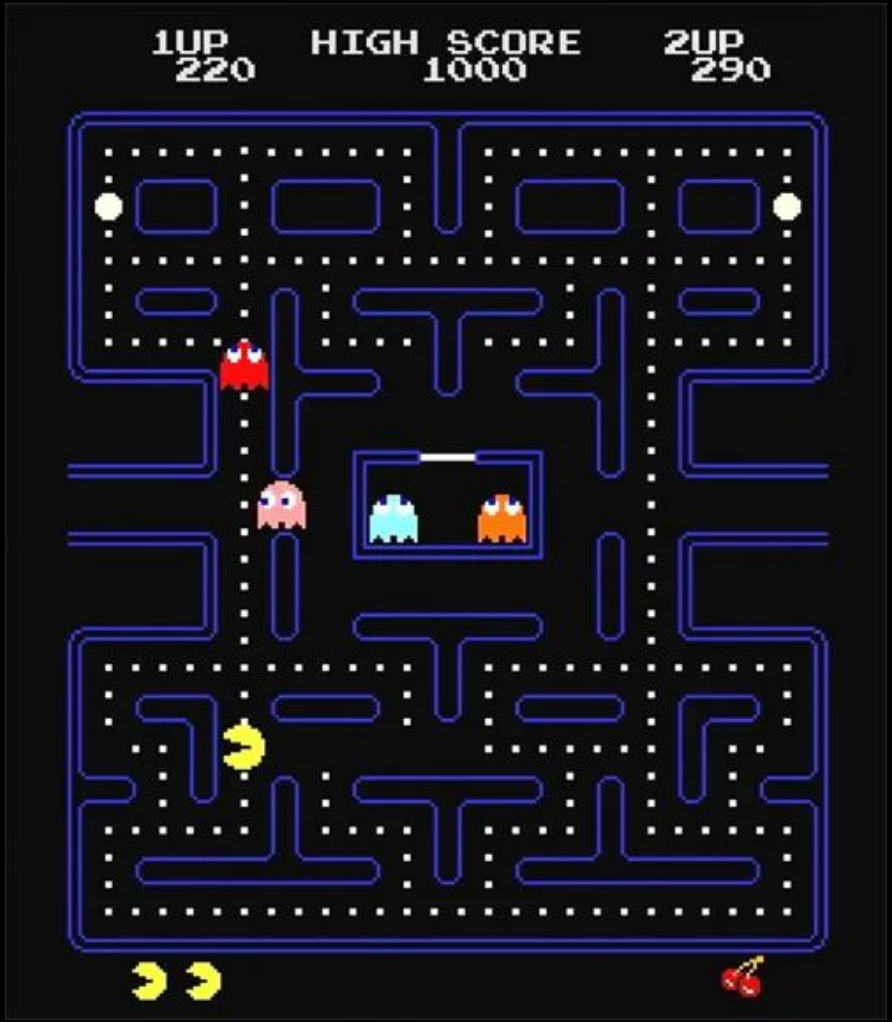
Presentación Web

Grupo 8 uwu

1ST - JAAF

2ND - VECB

3RD - THTH


1UP
220
HIGH SCORE
1000
2UP
290

Can I use canvas ? Settings

50 results found

☑ Caniuse (5) ☑ MDN (45)

Canvas (basic support) - LS

Method of generating fast, dynamic graphics using JavaScript.

Usage: Global

% of all users ?

95.5% + 1.06% = 96.56%

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-8		[1] 2-3.5		[1] 3.1-3.2				[1] 2.1-2.3								
9-10	12-98	3.6-97	4-99	4-15.3	10-82	3.2-15.3		3-4.4.4	12-12.1				4-15.0			
11	99	98	100	15.4	83	15.4	[2] all	99	64	100	98	12.12	16.0	10.4	7.12	2.5
		99-100	101-103	TP												

Creación de mapa

“#” -> Bloques

“.” -> Monedas

“ “ -> Vacíos

“o” -> Power-ups

```
1  const MAP_STRING = `###########################
2  #............##............#
3  #.####.#####.##.#####.####.#
4  #o####.#####.##.#####.####o#
5  #.####.#####.##.#####.####.#
6  #..........................#
7  #.####.##.########.##.####.#
8  #.####.##.########.##.####.#
9  #......##....##....##......#
10 ######.##### ## #####.######
11 ######.##### ## #####.######
12 ######.##          ##.######
13 ######.## ######## ##.######
14 ######.## ######## ##.######
15        .   ########   .
16 ######.## ######## ##.######
17 ######.## ######## ##.######
18 ######.##          ##.######
19 ######.## ######## ##.######
20 ######.## ######## ##.######
21 #............##............#
22 #.####.#####.##.#####.####.#
23 #.####.#####.##.#####.####.#
24 #o..##.......  .......##..o#
25 ###.##.##.########.##.##.###
26 ###.##.##.########.##.##.###
27 #......##....##....##......#
28 #.##########.##.##########.#
29 #.##########.##.##########.#
30 #..........................#
31 ###########################`;
```

Creación de mapa: Celdas

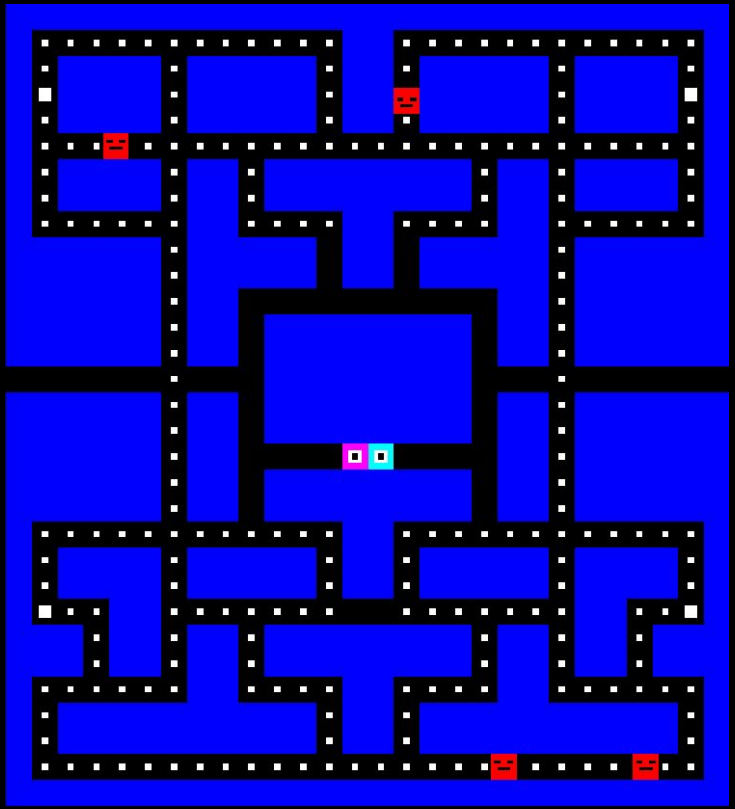
```
// Basic object
class Object {
  constructor(x0, y0) {
    this.width = cellSize;
    this.height = cellSize;
    this.x = x0;
    this.y = y0;
  }
}
```

Creación de mapa: Celdas

```
126 class Map {
127   constructor(map, ctx) {
128     this.barriers = [];
129     this.coins = [];
130     this.powerUps = [];
131     this.ctx = ctx;
132
133     const mapArray = map.split('\n');
134     for (let row = 0; row < gameHeight; row++) {
135       for (let col = 0; col < gameWidth; col++) {
136         switch (mapArray[row][col]) {
137           case '#':
138             this.barriers.push(new Object(col * cellSize, row * cellSize));
139             break;
140           case '.':
141             this.coins.push(new Object(col * cellSize, row * cellSize));
142             break;
143           case 'o':
144             this.powerUps.push(new Object(col * cellSize, row * cellSize));
145             break;
146         }
147       }
148     }
149   }
```

Creación de mapa: Draw

```
draw() {
    // barriers
    map.barriers.forEach((barrier) => {
      this.ctx.fillStyle = 'blue';
      this.ctx.fillRect(barrier.x, barrier.y, barrier.width, barrier.height);
    });
    // coins
    map.coins.forEach((coin) => {
      this.ctx.fillStyle = 'white';
      this.ctx.fillRect(
        coin.x + 3,
        coin.y + 3,
        coin.width - 6,
        coin.height - 6
      );
    });
    // powerUps
    map.powerUps.forEach((coin) => {
      this.ctx.fillStyle = 'white';
      this.ctx.fillRect(
        coin.x + 2,
        coin.y + 2,
        coin.width - 4,
        coin.height - 4
      );
    });
  }
```



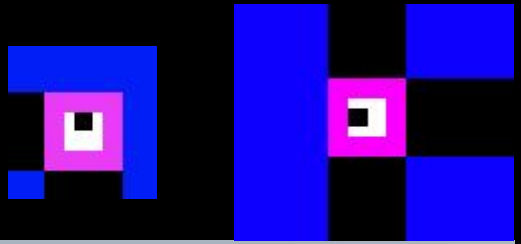
Jugadores

```
49  class Player extends Object {
50    constructor(x0, y0, color, ctx) {
51      super(x0, y0);
52      this.color = color;
53      this.vx = 0;
54      this.vy = 0;
55      this.hasNextVelocity = false;
56      this.nextVx = 0;
57      this.nextVy = 0;
58      this.ctx = ctx;
59    }
```

Jugadores: Movimiento

```
60  move() {
61      this.x += this.vx;
62      this.y += this.vy;
63      // teleport horizontal
64      if (gameWidth * cellSize < this.x + this.width) this.x = cellSize;
65      else if (this.x < 0) this.x = gameWidth * cellSize - cellSize;
66    }
67    setVelocity() {
68      this.hasNextVelocity = false;
69      this.vx = this.nextVx;
70      this.vy = this.nextVy;
71    }
72    setNextVelocity(vx, vy) {
73      this.hasNextVelocity = true;
74      this.nextVx = vx;
75      this.nextVy = vy;
76    }
```

Player: Draw



```
draw() {
    // body
    this.ctx.fillStyle = this.color;
    this.ctx.fillRect(this.x, this.y, this.width, this.height);
    // eye
    this.ctx.fillStyle = 'white';
    this.ctx.fillRect(this.x + 2, this.y + 2, 4, 4);
    this.ctx.fillStyle = 'black';
    this.ctx.fillRect(this.x + 3 + this.vx, this.y + 3 + this.vy, 2, 2);
  }
```

Enemigos: "Jugador simplificado"

```
class Enemy extends Object {
  constructor(x0, y0, color, ctx) {
    super(x0, y0);
    this.color = color;
    this.vx = 1;
    this.vy = 0;
    this.ctx = ctx;
  }
  draw() {
    // body
    this.ctx.fillStyle = this.color;
    this.ctx.fillRect(this.x, this.y, this.width, this.height);
    // eye
    this.ctx.fillStyle = 'black';
    this.ctx.fillRect(this.x + 1, this.y + 2 + this.vy, 2, 1);
    this.ctx.fillRect(this.x + 5, this.y + 2 + this.vy, 2, 1);
    this.ctx.fillRect(this.x + 2, this.y + 4 + this.vy, 4, 1);
  }
  move() {
    this.x += this.vx;
    this.y += this.vy;
    // teleport horizontal
    if (gameWidth * cellSize < this.x + this.width) this.x = cellSize;
    else if (this.x < 0) this.x = gameWidth * cellSize - cellSize;
  }
```

Enemigos: Movimiento

```
115 randomizeMovement() {
116     const choices = [-1, 0, 1];
117     const nonZeroChoices = [-1, 1];
118     this.vx = choices[Math.floor(Math.random() * choices.length)];
119
120     const verticalChoices = this.vx === 0 ? nonZeroChoices : choices;
121     this.vy =
122       verticalChoices[Math.floor(Math.random() * verticalChoices.length)];
123   }
```

Game: “La lógica del juego”

```
179  class Game {
180    constructor(map, players, enemies, canvas, ctx) {
181      this.map = map;
182      this.players = players;
183      this.enemies = enemies;
184      this.score = 0;
185      this.powerUpActive = false;
186      this.canvas = canvas;
187      this.ctx = ctx;
188      // Used to avoid multiple game over alerts
189      this.gameOver = false;
190      // Used to avoid multiple game won alters
191      this.gameWon = false;
192      // Used to cancel previous timer when new powerUp is eaten while previous is active
193      this.powerUpTimer = null;
194      // Used to check when to start the game
195      this.waitingPlayers = true;
196    }
```

Game: Colisiones

```
197  checkCollision(first, second) {
198      return (
199        first.x < second.x + second.width &&
200        second.x < first.x + first.width &&
201        first.y < second.y + second.height &&
202        second.y < first.y + first.height
203      );
204    }
```

Game: Colisiones

```
checkBarrierCollision(other) {
    return this.map.barriers.some((barrier) ⇒
      this.checkCollision(barrier, other)
    );
  }
```

Game: Colisiones

```
checkBarrierCollision(other) {
    return this.map.barriers.some((barrier) ⇒
      this.checkCollision(barrier, other)
    );
  }
```

```
checkCoinsEaten(player) {
    this.map.coins = this.map.coins.filter((coin) ⇒ {
      if (this.checkCollision(coin, player)) {
        this.score += 100;
        return false;
      }
      return true;
    });
  }
```

Game: Colisiones

```
checkBarrierCollision(other) {
    return this.map.barriers.some((barrier) ⇒
      this.checkCollision(barrier, other)
    );
  }
```

```
checkCoinsEaten(player) {
    this.map.coins = this.map.coins.filter((coin) ⇒ {
      if (this.checkCollision(coin, player)) {
        this.score += 100;
        return false;
      }
      return true;
    });
  }
```

```
checkPowerUpsEaten(player) {
    this.map.powerUps = this.map.powerUps.filter((coin) ⇒ {
      if (this.checkCollision(coin, player)) {
        clearTimeout(this.powerUpTimer);
        this.score += 500;
        this.powerUpActive = true;
        this.powerUpTimer = setTimeout(
          () ⇒ (this.powerUpActive = false),
          10_000
        );
        return false;
      }
      return true;
    });
  }
```

Game: Colisiones

```
checkBarrierCollision(other) {
    return this.map.barriers.some((barrier) ⇒
      this.checkCollision(barrier, other)
    );
  }
```

```
checkCoinsEaten(player) {
    this.map.coins = this.map.coins.filter((coin) ⇒ {
      if (this.checkCollision(coin, player)) {
        this.score += 100;
        return false;
      }
      return true;
    });
  }
```

```
checkPowerUpsEaten(player) {
    this.map.powerUps = this.map.powerUps.filter((coin) ⇒ {
      if (this.checkCollision(coin, player)) {
        clearTimeout(this.powerUpTimer);
        this.score += 500;
        this.powerUpActive = true;
        this.powerUpTimer = setTimeout(
          () ⇒ (this.powerUpActive = false),
          10_000
        );
        return false;
      }
      return true;
    });
  }
```

```
checkEnemyCollision(player) {
  this.enemies.forEach((e) ⇒ {
    if (this.checkCollision(e, player)) {
      if (this.powerUpActive) {
        e.x = cellSize * 13;
        e.y = cellSize * 11;
        this.score += 1000;
      } else if (!this.gameOver) {
        this.gameOver = true;
        alert('GAME OVER!!!');
        window.location.reload();
      }
    }
  });
}
```

Game.tick(): Frame lógico del juego

```
tick() {
    // check if all coins and powerups have been eaten
    this.checkWin();
    this.enemies.forEach((e) => {
      this.enemyMovement(e);
    });
    this.players.forEach((p) => {
      // attemp direction change
      const attempChange = this.simulateMovement(p, p.nextVx, p.nextVy);
      if (!this.checkBarrierCollision(attempChange)) p.setVelocity();
      // move if possible
      const nextMove = this.simulateMovement(p, p.vx, p.vy);
      if (!this.checkBarrierCollision(nextMove)) p.move();
      // eat coin if posible
      this.checkCoinsEaten(p);
      // eat power if posible
      this.checkPowerUpsEaten(p);
      // die if possible
      this.checkEnemyCollision(p);
    });
  }
```

Game.draw(): Frame visual del juego

```
297  draw() {
298      this.ctx.filter = this.powerUpActive ? 'invert(.75)' : 'invert(0)';
299      this.ctx.clearRect(0, 0, this.canvas.width, this.canvas.height);
300      this.ctx.fillStyle = 'black';
301      this.ctx.fillRect(0, 0, this.canvas.width, this.canvas.height);
302      this.drawObjects();
303      this.drawUI();
304    }
```

Game.draw(): Frame visual del juego

```
305  drawObjects() {
306      this.map.draw();
307      this.players.forEach((p) ⇒ p.draw());
308      this.enemies.forEach((e) ⇒ e.draw());
309    }
310    drawUI() {
311      ctx.fillStyle = 'white';
312      ctx.font = '16px Consolas';
313      ctx.fillText(`SCORE: ${this.score}`, 0, (gameHeight + 2) * cellSize);
314    }
```

¿Dónde está RxJS?

¿FIN?

Instancia de juego

```
341  const map = new Map(MAP_STRING, ctx);
342  const p1 = new Player(cellSize * 13, cellSize * 17, 'magenta', ctx);
343  const p2 = new Player(cellSize * 14, cellSize * 17, 'cyan', ctx);
344  const enemies = [
345    new Enemy(cellSize, cellSize, 'red', ctx),
346    new Enemy(cellSize * 16, cellSize, 'red', ctx),
347    new Enemy(cellSize, cellSize * 29, 'red', ctx),
348    new Enemy(cellSize * 16, cellSize * 29, 'red', ctx),
349  ];
350  const game = new Game(map, [p1, p2], enemies, canvas, ctx);
```

RxJS: Control de Input

```
353   const keyDowns$ = rxjs.fromEvent(window, 'keydown');
```

RxJS: Control de Input

Player 1

```
keyDowns$.subscribe((kd) => {
  switch (kd.key.toLowerCase()) {
    case 'w':
      p1.setNextVelocity(0, -1);
      break;
    case 'd':
      p1.setNextVelocity(1, 0);
      break;
    case 's':
      p1.setNextVelocity(0, 1);
      break;
    case 'a':
      p1.setNextVelocity(-1, 0);
      break;
```

Player 2

```
case 'arrowup':
  p2.setNextVelocity(0, -1);
  break;
case 'arrowright':
  p2.setNextVelocity(1, 0);
  break;
case 'arrowdown':
  p2.setNextVelocity(0, 1);
  break;
case 'arrowleft':
  p2.setNextVelocity(-1, 0);
  break;
```

RxJS: Control de Input

Main Menu

```
380  case '1':
381    // Remove player 2
382    game.waitingPlayers = false;
383    game.players = game.players.slice(0, 1);
384    break;
385  case '2':
386    // Do not remove any player
387    game.waitingPlayers = false;
388    break;
```

RxJS: Game.run()

```
run() {
  // Game main loop
  rxjs.interval(1000 / 120).subscribe((n) => {
    if (!this.waitingPlayers) {
      // logic
      this.tick();
      // draw
      this.draw();
    } else {
        this.drawMenu();
    }
  });
}
```

FIN :)