PING PONG

Creador por

SANTIAGO PAEZ EVELY

DISEÑO Y MAQUETACION

CASTILLO ESPINOSA

MAURICIO EMMANUEL

TRADUCIDO POR:

FERNANDO LOPEZ

Manual de uso del juego “PING PONG”

Introducción

The game “PING PONG” is a digital recreation of the classic board game where two players compete on hitting a ball with their shovels keeping it inside the zone.
It was completely developed in HTML, CSS and JavaScript, without any need for an external library.
Its operation combines:

- Visual design (CSS) For the game’s appearance.
- Structure (HTML) For the board’s elements.
- Logic (JavaScript) For movement, collisions detector and score system.

Used documents

1. index.html:
2. Contains all the visual structure (board, shoves, ball and marker) and game’s logic in JavaScript.

Styles CSS (internal):
They are inside the same HTML file, inside the <style> label.

1. Defines colors,sizes,shadows , positions and the general game style.

HTML code explication

```html
<div class="escenario">
    <div class="marcador marcador1">0</div>
    <div class="marcador marcador2">0</div>
    <div class="pala pala1"></div>
    <div class="pala pala2"></div>
    <div class="red"></div>
    <div class="pelota"></div>
</div>
```

Function of each element

- .escenario → Is the playing area. Contains all the elements (Ball,shovels ,marker , grid, etc.).
- .marcador1 y .marcador2 → Shows each players score.
- .pala1 y .pala2 → Are the shovels controlled by the players (left and right).
- .red →Dotted line that divides the field in 2.
- .pelota → Is the ball that moves and bounces during the game.
-

CSS explication

The CSS sets the game's appearance.

1. General settings

```css
* { padding: 0; margin: 0; }
body { background: black; color: white; }
```

- Eliminates the default padding and the margin.
- Gives a black background for the game elements to stand-up.

2. Game scenarios

```css
.escenario {
    width: 800px;
    height: 600px;
    margin: calc(50vh - 300px) auto 0;
    border: 3px solid rgb(0, 242, 255);
    position: relative;
    background: rgb(4, 1, 39);
    box-shadow: cyan 0 0 10px, cyan 0 0 10px inset;
    border-radius: 8px;
}
```

- Creates a 800x600 pixels area.
- It centers the screen (margin: calc(50vh - 300px)).
- The edges and the margin gives a neon effect.
- position: relative allows to place the objects like (shovels, ball) with absolute coordinates.

3. Markers

```
.marcador {
    font-size: 150px;
    position: absolute;
    top: 40px;
    color: rgb(255, 125, 246);
    text-shadow: rgb(255, 36, 240) 0 0 10px;
}
.marcador1 { left: 100px; }
.marcador2 { right: 100px; }
```

- Shows the score with a big font.
- One in the left (marcador1) and one in the right (marcador2).

4. Shovels

```
.pala {
    width: 6px;
    height: 100px;
    background: rgb(5, 255, 226);
    position: absolute;
    bottom: 50%;
    border-radius: 10px;
    box-shadow: cyan 0 0 10px;
}
.pala1 { left: 40px; }
.pala2 { left: 760px; }
```

- Each shovel has 6px of width and 100px of height.
- They are placed at the field sides.
- The cyan color and the glow, makes them to stand-up of the black background

5. Grid

```css
.red {
    position: absolute;
    top: 0;
    bottom: 0;
    left: calc(50% - 3px);
    border-left: 6px dotted rgb(255, 125, 246);
}
```

- Dotted line at the center of the dashboard.

6. Ball

```css
.pelota {
    background: rgb(255, 245, 109);
    width: 16px;
    height: 16px;
    position: absolute;
    bottom: 300px;
    left: 400px;
    border-radius: 50%;
    box-shadow: rgb(255, 225, 30) 0 0 10px;
}
```

- It’s a yellow circle (border-radius: 50%).
- The default position is in the middle.
- Has a glow that makes it to stand-up.

JavaScript code explication

JavaScript controls the game’s logic, movement and collisions.

1. Bordes class

```
class Bordes {
    constructor(minX, maxX, minY, maxY) {
        this.minX = minX;
        this.maxX = maxX;
        this.minY = minY;
        this.maxY = maxY;
    }
}
```

- Sets the field limits (left, right, up and down).
- Detects bounces and when the ball leaves the field.

2. ObjetoMovil Class

```
class ObjetoMovil {
    constructor(bordesTablero, elem, vel) { ... }
    GetBordes() { ... }
    Resetear() { ... }
}
```

- Base class of the moving objects (shovels and ball).
- GetBordes()Detects the object edges to calculate collisions.
- Resetear() updates its size if it changes.

3. Pelota Class

```
class Pelota extends ObjetoMovil {
    Mover() { ... }
    ComprobarRebote() { ... }
    RebotarX(inerciaY) { ... }
    RebotarY() { ... }
    ComprobarGol() { ... }
}
```

Main functions:

- Mover() → Calculates its position, according to speed and direction.
- ComprobarRebote() → If it touches the floor or ceiling, the vertical direction changes.
- RebotarX() →Reverses horizontal direction (when a shovel hits it).
- ComprobarGol() → Detects if the ball leaves the field (scores point).
- Resetear() → Returns the ball at the center.
-

4. Pala Class

```
class Pala extends ObjetoMovil {
    IniciarMovimiento(evento) { ... }
    FinalizarMovimiento(evento) { ... }
    Mover() { ... }
    Frenar() { ... }
    ComprobarColision(bordes2) { ... }
}
```

- Controls the shovel position and movement.
- Uses the keys:
  - Player 1: "a" (raise) y "z" (lower).
  - Player 2: "ArrowUp" y "ArrowDown".
- Frenar() Apply a slow down when the player lets go of the keys.
- ComprobarColision() verifies if the shovel and the ball touches each other.

5. Marcador Class

```
class Marcador {
    constructor(elem) { this.elem = elem; this.puntos = 0; }
    GanarPunto() { this.puntos++; this.elem.innerHTML = ""+this.puntos; }
}
```

- Keeps the count of each player's score.
- Each time the ball leaves the field the opposite player gains a point.

6. game funcions

```
function Update() {
    pelota.Mover();
    MoverPalas();
    ComprobarPalazo();
    ComprobarGol();
}
```

- The action is executed in each animation frame.
- Updates the movement, collisions and marker.

7. Collisions

```
if(pala1.ComprobarColision(pelota.GetBordes())) {
    if(pelota.dirX < 0) pelota.RebotarX(pala1.velActual * deltaTime);
}
```

- Detects if the ball hits the shovel.
- If it's so, it changes its direction and applies a bouncing effect according to the strength.

8. Restart game

```
function Reiniciar() {
    pala1.Resetear();
    pala2.Resetear();
    var dir = Math.random() * 2 * Math.PI;
    pelota.Resetear(Math.random() * 150 + 250, 400, 300, Math.cos(dir), Math.sin(dir));
}
```

- After a point, restarts the positions and gives the ball a new direction.

9. Game loop

```
function Tick() {
    deltaTime = (Date.now() - time) / 1000;
    time = Date.now();
    Update();
    requestAnimationFrame(Tick);
}
requestAnimationFrame(Tick);
```

- Calculates the time between frames (deltaTime) so that the movement is fluent.
- requestAnimationFrame updates the game constantly (continuously animation).

How to play

- Player 1:
  - Raise: A key
  - Lower: Z key
- Player 2:
  - Raise: ↑ key (arrow pointing up)
  - Lower: ↓ key (arrow pointing down)
- When the ball leaves the field without getting hit the opposite player gains a point.
- The game doesn't have an established ending, but you can play until the score you choose.

Conclusion

The game PING PONG is an excellent example of how three coding languages combine in an interactive app:

- HTML Structure of the game elements (board, ball, shovels, marker).
- CSS Sets the style, color and visual appearance, achieving an attractive neon color .
- JavaScript gives the game live: controls the movements, detects collisions, manages the score and creates an animation loop.

Thanks to the use of classes, keyboards, events, collisions with coordinates and basic physics, the result is smooth and fully functional game within a browser.