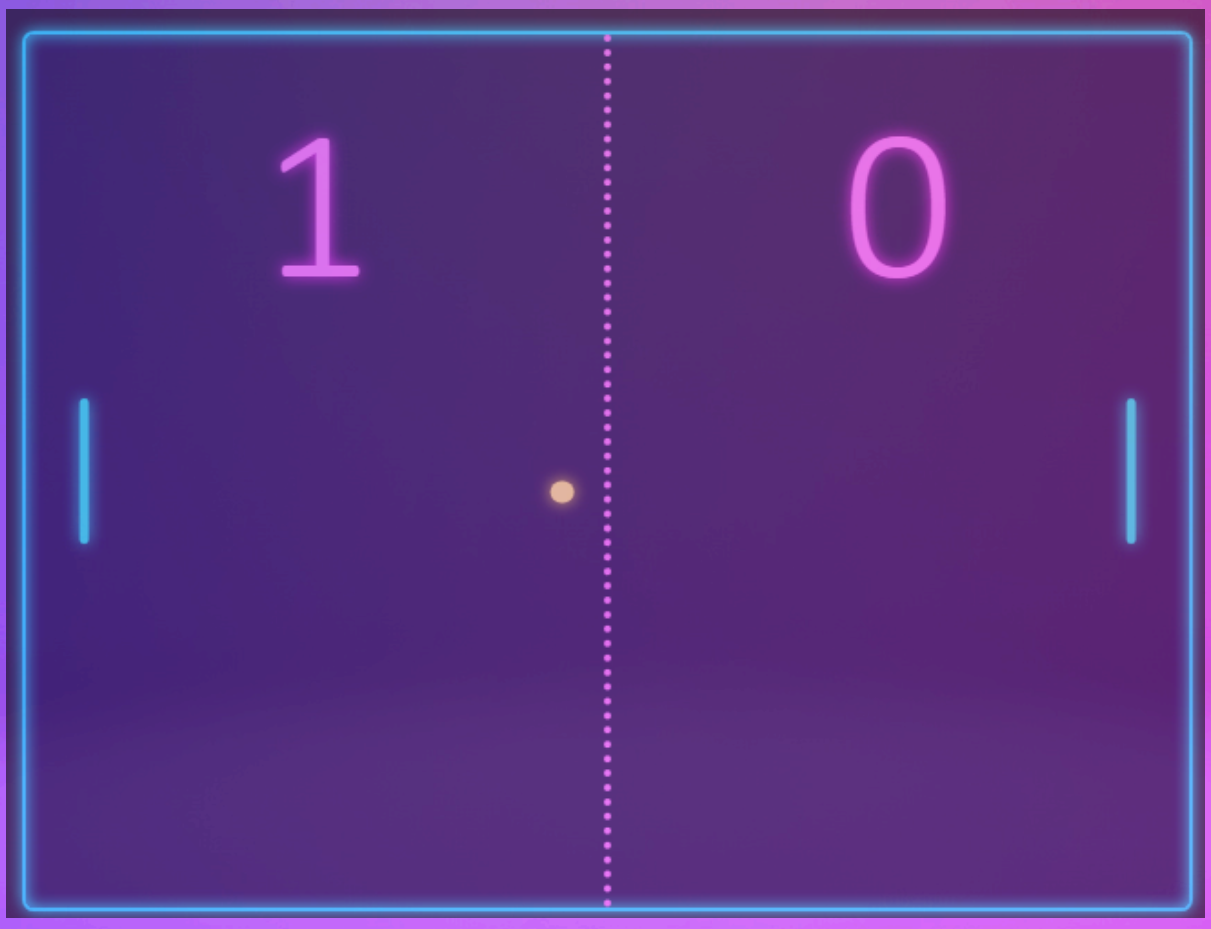
PING PONG

Creador por

SANTIAGO PAEZ EVELY



DISEÑO Y MAQUETACION

CASTILLO ESPINOSA MAURICIO EMMANUEL

Manual de uso del juego “PING PONG”

Introducción

El juego “PING PONG” es una recreación digital del clásico juego de mesa donde dos jugadores compiten por golpear una pelota con sus palas, evitando que salga de su campo.
Está desarrollado completamente con HTML, CSS y JavaScript, sin necesidad de bibliotecas externas.
Su funcionamiento combina:

- Diseño visual (CSS) para la apariencia del juego.
- Estructura (HTML) para los elementos del tablero.
- Lógica (JavaScript) para el movimiento, detección de colisiones y sistema de puntos.

Documentos utilizados

1. index.html:
2. Contiene toda la estructura visual (tablero, palas, pelota y marcador) y la lógica del juego en JavaScript.
3. Estilos CSS (internos):
4. Están dentro del mismo archivo HTML, dentro de la etiqueta <style>.
5. Definen colores, tamaños, sombras, posiciones y estilo general del juego.

Explicación del código HTML

```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>
```

🔹 ¿Para qué sirve cada elemento?

- .escenario → Es el área donde se juega. Contiene todos los elementos (pelota, palas, marcador, red).
- .marcador1 y .marcador2 → Muestran los puntos de cada jugador.
- .pala1 y .pala2 → Son las paletas controladas por los jugadores (izquierda y derecha).
- .red → Línea punteada que divide el campo en dos mitades.
- .pelota → Es la bola que se mueve y rebota durante el juego.

Explicación del CSS

El CSS define cómo se ve el juego.

1. Ajustes generales

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

Elimina márgenes y rellenos por defecto.
Da fondo negro al cuerpo de la página para resaltar los elementos del juego.

🔹 2. Escenario del juego

```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;
}
```

- Crea un área de 800x600 píxeles.
- La centra en la pantalla (margin: calc(50vh - 300px)).
- El borde y las sombras dan efecto neón.
- position: relative permite posicionar los objetos dentro (palas, pelota) con coordenadas absolutas.

3. Marcadores

```css
.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; }
```

- Muestran los puntos con tipografía grande.
- Uno está a la izquierda (marcador1) y otro a la derecha (marcador2).

4. Palas

```css
.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; }
```

- Cada pala tiene 6px de ancho y 100px de alto.
- Se colocan verticalmente a los lados del campo.
- Su color cyan y brillo las hacen visibles sobre el fondo oscuro.

5. Red

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

- Línea punteada en el centro del tablero (como en el ping pong real).

6. Pelota

```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;
}
```

- Es un círculo amarillo (por border-radius: 50%).
- Su posición inicial está centrada.
- Tiene un brillo que la hace resaltar.

Explicación del código JavaScript

El JavaScript controla la lógica, movimiento y colisiones del juego.

1. Clase Bordes

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

- Define los límites del campo (izquierda, derecha, arriba y abajo).
- Sirve para detectar rebotes y cuándo la pelota sale del área.

2. Clase ObjetoMovil

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

- Clase base para objetos que se mueven (palas y pelota).
- GetBordes() devuelve los límites del objeto para calcular colisiones.
- Resetear() actualiza su tamaño si cambia.

🔷 3. Clase Pelota

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

Funciones principales:

- Mover() → Calcula nueva posición según velocidad y dirección.
- ComprobarRebote() → Si toca techo o suelo, cambia dirección vertical.
- RebotarX() → Invierte dirección horizontal (cuando golpea una pala).
- ComprobarGol() → Detecta si la pelota sale del tablero (anota punto).
- Resetear() → Devuelve la pelota al centro.

🔹 4. Clase Pala

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

- Controla la posición y movimiento de la pala.
- Usa las teclas:
  - Jugador 1: "a" (subir) y "z" (bajar).
  - Jugador 2: "ArrowUp" y "ArrowDown".
- Frenar() aplica una desaceleración cuando el jugador suelta las teclas.
- ComprobarColision() verifica si la pala y la pelota se tocan.

🔹 5. Clase Marcador

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

- Lleva el conteo de puntos de cada jugador.
- Cada vez que la pelota sale del tablero, el jugador contrario gana un punto.

🔹 6. Funciones del juego

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

- Se ejecuta en cada cuadro de animación.
- Actualiza movimiento, colisiones y marcador.

🔹 7. Colisiones

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

- Detecta si la pelota toca una pala.
- Si es así, cambia su dirección y aplica efecto de rebote según la fuerza del golpe.

8. Reinicio del juego

```
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));
}
```

- Después de un gol, reinicia posiciones y da nueva dirección a la pelota.

9. Bucle del juego

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

- Calcula el tiempo entre cuadros (deltaTime) para que el movimiento sea fluido.
- requestAnimationFrame actualiza el juego de forma constante (animación continua).

Cómo se juega

- Jugador 1:
  - Subir: tecla A
  - Bajar: tecla Z
- Jugador 2:
  - Subir: tecla ↑ (flecha arriba)
  - Bajar: tecla ↓ (flecha abajo)
- Cuando la pelota pasa una pala, el jugador contrario gana un punto.
- El juego no tiene fin automático, pero puedes competir hasta el marcador que desees.

🏁 Conclusión

El juego PING PONG es un excelente ejemplo de cómo se combinan tres lenguajes web para crear una aplicación interactiva:

- HTML estructura los elementos del juego (tablero, pelota, palas, marcador).
- CSS define su estilo, color y disposición visual, logrando un diseño atractivo tipo neón.