MANUAL TÉCNICO DEL JUEGO "LA NEW ERA"

MANUAL Y JUEGO ELABORADO POR :

YAZU BAUTISTA

DISEÑO Y MAQUETACION: MAURICIO ROJAS

TRADUCIDO POR: ALEXANDRA DIAZ

INTRODUCCIÓN

"La New Era" es un videojuego de lucha libre 1 vs 1 diseñado con HTML, CSS y JavaScript, tres lenguajes fundamentales del desarrollo web.

El jugador controla a Penta El Zero M, y el sistema maneja al enemigo Fénix. El objetivo es derrotar al oponente antes de que se acabe el tiempo, utilizando ataques (puñetazos y patadas) mientras se controla la posición en el ring.

El juego demuestra cómo se integran los tres lenguajes:

- HTML crea la estructura (personajes, barras de salud, botones).
- CSS define el diseño visual, colores, posiciones, fondo, transiciones y tamaños.
- JavaScript otorga la lógica y movimiento: golpes, daño, detección de colisiones, tiempo y resultado final.

ARCHIVOS REQUERIDOS

El proyecto se compone de tres archivos principales y varias imágenes que conforman la interfaz visual.

Archivos base:

- lucha.html → define toda la estructura del juego.
- lucha.css → establece los estilos visuales, posiciones y tamaños.
- lucha.js → programa la jugabilidad, movimientos y resultados.

IMAGENES UTILIXIZADAS

JUGADOR (PENTA EL CERO M)

penta.png

penta1.png

penta2.png

ENEMIGO(FENIX)

fenix.png

fenix1.png

fenix2.png

CINTURÓN DE CAMPEON

cinto.png

FONDO DE RIN

rin.jpg

ESTRUCTURA HTML (LUCHA HTML)

El HTML es el esqueleto del juego. Define qué elementos existen y dónde se colocan.

🔹 Encabezado (<head>)

```
<head>
  <meta charset="UTF-8" />
  <title>LA NEW ERA</title>
  <link rel="stylesheet" href="lucha.css" />
</head>
```

- <meta charset="UTF-8">: Permite usar tildes, eñes y caracteres especiales sin errores.
- <title>: El nombre que aparece en la pestaña del navegador.
- <link rel="stylesheet" href="lucha.css">: Conecta la hoja de estilos externa (CSS) al HTML.

Cuerpo (<body>)

```
<body>
  <h1>LA NEW ERA</h1>
```

- Muestra el título principal en pantalla.
- Se usa <h1> porque es el encabezado más importante y se puede estilizar con CSS.

Arena de combate

```
<div class="arena">
  <div id="player" class="luchador"></div>
  <div id="enemy" class="luchador enemy"></div>

  <img id="belt" class="hidden" src="cintO.png" />
  <div id="result" class="hidden"></div>
</div>
```

- : Es el área de lucha, el “ring”.
- #player: Contiene la imagen del jugador.
- #enemy: Contiene la imagen del enemigo.
- #belt: Muestra el cinturón cuando el jugador gana.
- #result: Aparece al final mostrando el mensaje “¡Ganaste!” o “Perdiste”.

Los valores se controlan con CSS y JS.

El atributo class="hidden" hace que un elemento esté oculto hasta que el JavaScript lo muestre.

Barras de vida y texto de salud

```
<div class="info">
  <div class="health-bar-container">
    <div id="player-health-bar" class="health-bar"></div>
  </div>
  <div class="health-bar-text">Jugador: <span id="player-health">100</span></div>

  <div class="health-bar-text">Enemigo: <span id="enemy-health">100</span></div>
  <div class="health-bar-container">
    <div id="enemy-health-bar" class="health-bar enemy-bar"></div>
  </div>
</div>
```

- .health-bar-container: Es el marco o contenedor de la barra de salud.
- .health-bar: Es la barra que se va vaciando con el daño.
- #player-health y #enemy-health: Muestran los valores numéricos actualizados desde JS.
- .enemy-bar: Se usa para cambiar el color del enemigo.

Temporizador y reinicio

- #timer: Muestra el tiempo restante del combate, actualizado por JS.
- #restart-btn: Permite reiniciar el juego al hacer clic.

```
<p id="timer">Tiempo restante: 60s</p>
<button id="restart-btn">Reiniciar Juego</button>
```

Al final, se carga la lógica del juego con:

```
<script src="lucha.js"></script>
```

ESTILOS CSS (LUCHA.CSS)

El CSS da apariencia y ambiente. Define colores, tamaños, bordes y animaciones.

Fondo y texto general

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: url('ring.jpg') no-repeat center center fixed;
  background-size: cover;
  color: white;
}
```

- font-family define el tipo de letra global.
- text-align: center centra todos los textos.
- background usa la imagen ring.jpg como fondo, sin repetir (no-repeat) y centrada.
- background-size: cover hace que cubra toda la pantalla.

color: white asegura que el texto sea visible sobre el fondo oscuro.

Arena

```css
.arena {
  width: 1100px;
  height: 500px;
  border: 4px solid #333;
  border-radius: 12px;
  background: rgba(0, 0, 0, 0.3);
  overflow: hidden;
  margin: 20px auto;
  position: relative;
}
```

- width y height: definen el tamaño del ring.
- border crea un marco negro.
- border-radius: 12px suaviza las esquinas.
- background: rgba(0, 0, 0, 0.3) aplica una capa oscura semitransparente (30% opaca).
- overflow: hidden evita que los personajes se salgan visualmente del área.
- margin: auto la centra en pantalla.
- position: relative permite colocar luchadores dentro con position: absolute.

Luchadores

```css
.luchador {
  position: absolute;
  bottom: 0;
  width: 220px;
  height: 340px;
  background-size: contain;
  background-repeat: no-repeat;
  transition: transform 0.2s ease, background-image 0.2s ease;
}
```

- position: absolute: posiciona libremente dentro del ring.
- bottom: 0: los mantiene sobre el suelo.
- width y height: definen tamaño del personaje.
- background-size: contain: ajusta la imagen completa al tamaño sin recortar.
- transition: suaviza los cambios cuando se mueven o atacan.

Posiciones iniciales:

```css
#player { left: 60px; background-image: url('penta.png'); }
#enemy { right: 60px; background-image: url('fenix.png'); }
```

- Enemigo a la derecha (right: 60px).
- Cada uno con su imagen inicial

Barras de vida

```css
.health-bar-container {
  width: 300px;
  height: 20px;
  background: #111;
  margin: 8px auto;
  border-radius: 8px;
}
.health-bar {
  height: 100%;
  background: #4caf50;
  width: 100%;
  transition: width 0.3s ease;
}
.enemy-bar {
  background: #e53935;
```

- El contenedor es negro (#111) con bordes redondeados.
- La barra verde representa la salud actual, y su ancho disminuye con el daño.
- El enemigo usa rojo para distinguirse.
- transition suaviza el efecto de daño.

Temporizador y botón

```css
#timer {
  font-size: 24px;
  margin-top: 20px;
}
#restart-btn {
  background: #27ae60;
  color: white;
  border: none;
  padding: 12px 30px;
  font-size: 18px;
  border-radius: 10px;
  cursor: pointer;
  transition: background 0.3s;
}
#restart-btn:hover {
  background: #2ecc71;
}
```

- #timer: muestra claramente el tiempo.
- #restart-btn: verde con texto blanco, grande y redondeado para destacar.
- :hover: cambia el color cuando el cursor pasa encima, dándole sensación interactiva.

LÓGICA DEL JUEGO (LUCHAJS)

El archivo JavaScript controla toda la jugabilidad.

🔹 Variables principales

```
const player = document.getElementById("player");
const enemy = document.getElementById("enemy");
const playerHealthBar = document.getElementById("player-health-bar");
```

Guarda las referencias de los elementos HTML para manipularlos.
Así se pueden cambiar imágenes, posiciones y salud con código.

Movimiento

```
document.addEventListener("keydown", (e) => {
  if (e.key === "ArrowLeft") movePlayer(-20);
  if (e.key === "ArrowRight") movePlayer(20);
});
```

Detecta teclas presionadas:

- Izquierda/Derecha mueven al jugador 20 píxeles por vez.
- Este valor (20) se elige para dar fluidez sin que se desplace demasiado rápido.

Ataques del jugador

```
if (e.key === "a") playerAttack("punch");
if (e.key === "s") playerAttack("kick");
```

- A ejecuta un puñetazo (cambia imagen a penta1.png).
- S ejecuta una patada (cambia imagen a penta2.png).
- Cada ataque tiene un pequeño “cooldown” para no abusar de golpes.

Colisiones y daño

```
if (Math.abs(playerX - enemyX) < 120) {
  enemyHealth -= 10;
}
```

- Si la distancia entre los luchadores es menor que 120 píxeles, hay impacto.
- El daño estándar es 10 puntos por golpe.
- Este valor se puede ajustar para hacerlo más difícil o fácil.

Ataque automático del enemigo

```
setInterval(() => {
  if (!gameOver) enemyAttack();
}, 800);
```

- Cada 0.8 segundos el enemigo intenta atacar.
- Se usa setInterval para repetir acciones.
- !gameOver evita ataques una vez terminado el juego.

Temporizador

```
const timer = setInterval(() => {
  timeLeft--;
  document.getElementById("timer").textContent = `Tiempo restante: ${timeLeft}s`;
  if (timeLeft <= 0) endGame();
}, 1000);
```

- Resta 1 segundo por cada ciclo de 1000 ms (1 segundo real).
- Cuando llega a 0, se llama a endGame().

Final del juego

```
function endGame() {
  clearInterval(timer);
  if (playerHealth > enemyHealth) resultDiv.textContent = "¡Ganaste!";
  else if (playerHealth < enemyHealth) resultDiv.textContent = "Perdiste";
  else resultDiv.textContent = "Empate";
  resultDiv.classList.remove("hidden");
}
```

- Detiene el tiempo.
- Compara las vidas para decidir el resultado.
- Muestra el mensaje final y revela el cinturón si se gana.

Instrucciones

1. Guarda todos los archivos y las imágenes en una misma carpeta.
2. Abre lucha.html en tu navegador.
3. Usa las flechas izquierda/derecha para moverte.
4. Presiona A para puñetazo, S para patada.
5. Gana reduciendo la vida del enemigo antes de que el tiempo termine.
6. Presiona Reiniciar Juego para jugar de nuevo.

Personalización

Puedes modificar libremente:

- Daño: cambia el valor de enemyHealth -= 10;.
- Velocidad del enemigo: cambia el número del setInterval().
- Tiempo de combate: modifica el valor inicial timeLeft = 60;.
- Fondo o personajes: reemplaza imágenes en el CSS.

Conclusión

"La New Era" combina los tres lenguajes base de la web para crear una experiencia jugable e interactiva.

El HTML estructura los elementos, el CSS da identidad visual, y el JavaScript aporta vida y dinamismo.

Cada valor numérico (tamaño, distancia, tiempo, daño) está pensado para un equilibrio entre jugabilidad, claridad visual y facilidad de comprensión.