UNIVERSIDAD
DON BOSCO

Desarrollo de Aplic. Web con Soft. Interpret. en el Cliente

Investigación aplicada 2

Estudiante	Carnet
Francisco Ernesto Ruano Torres	RT243331
Gabriela Stephanie Figueroa Calderón	FC250109
Cesar Alejandro Lara Franco	LL202677
Cesar Armando Meléndez	MA250099
Josue Ernesto Mena Colato	MC250517

Docente: Mauricio Vladimir Fuentes Salguero

Introducción

Al cargar la página lo primero que aparece es un menú con el título y dos botones: uno para iniciar y otro para salir. Al comenzar, se muestran las rondas de juego, con un contador regresivo que aparece en pantalla usando setInterval, el cual va cambiando cada segundo de 3 a 2, luego a 1 y finalmente muestra “¡Ya!”, momento en el que se habilitan las imágenes para que el jugador elija su opción. La computadora hace su elección al azar, y de acuerdo con reglas predefinidas se decide quién gana la ronda. El resultado se muestra junto con las imágenes seleccionadas por ambos. Todo esto se repite hasta completar las cinco rondas, acumulando puntajes tanto del jugador como de la CPU. Una vez finalizado el ciclo, el área del juego se oculta y se despliega una tabla de resultados con el historial de cada ronda, además de un mensaje que anuncia si el jugador ganó, perdió o empató en el marcador global.

En conjunto, el HTML define las secciones visibles, el CSS da estilos para que la experiencia sea clara y atractiva, y el JavaScript es el motor que controla la lógica del juego, las rondas, el conteo regresivo y la determinación de ganadores.

Función setInterval:

```
let count = 3;

const interval = setInterval(() => {

  count--;

  if (count === 0) {

    clearInterval(interval);

    document.getElementById('countdown').textContent = '¡Ya!';

    updateCountdownStyle('ya');

    document.getElementById('choices').classList.remove('hidden');

  } else {

    document.getElementById('countdown').textContent = count;

    updateCountdownStyle(count);

  }}, 1000);
```

En este juego la función setInterval se utiliza para crear el efecto de conteo regresivo antes de que el jugador pueda elegir su opción. Básicamente, setInterval ejecuta una porción de código cada cierto intervalo de tiempo. En este caso, cada 1000 milisegundos y va disminuyendo el número del contador en pantalla, mostrando primero el 3, luego el 2, después el 1, hasta llegar a "¡Ya!". Cuando el contador alcanza cero, se detiene el intervalo con clearInterval para que no siga ejecutándose, y en ese momento aparecen las imágenes de muro, conejo y flecha para que el jugador haga su elección. De esta manera, setInterval permite que el conteo suceda de forma automática y visible, sin necesidad de que el jugador haga nada más que esperar a que termine la cuenta atrás.

Detalles del código

OBJETIVO DEL JUEGO:

- Juegas contra la CPU.
- Cada ronda, eliges una de las 3 opciones: Muro, Conejo o Flecha.
- Se juegan 5 rondas.
- Gana el que tenga más victorias.
- Cada elección se acompaña de una imagen.
- Hay un conteo regresivo antes de elegir.
- Al final, se muestra una tabla de resultados.

ESTRUCTURA DEL CÓDIGO

El proyecto tiene 1 solo archivo HTML, que contiene:

1. HTML (estructura visual)
2. CSS (estilo y presentación)
3. JavaScript (lógica del juego)

ARCHIVOS NECESARIOS EN TU CARPETA:

Tu carpeta debe tener:
juego.html
muro.png
conejo.png
flecha.png

HTML (estructura visual)

Menú principal

```
<div id="menu">
  <h1>¡Muro, Conejo o Flecha!</h1>
  <button onclick="startGame()">Jugar</button>
  <button onclick="exitGame()">Salir</button>
</div>
```

- Muestra el título del juego y 2 botones.
- Botón "Jugar" inicia el juego.
- Botón "Salir" simula cerrar (redirecciona por seguridad).

Área del juego:

```
<div id="game" class="hidden">
  <h2>Ronda <span id="round-number">1</span> de 5</h2>
  <div id="countdown">Preparados...</div>
  <div class="choices hidden" id="choices">
    <img src="muro.png" onclick="playerSelect('muro')">
    <img src="conejo.png" onclick="playerSelect('conejo')">
    <img src="flecha.png" onclick="playerSelect('flecha')">
  </div>
  <div id="round-result" class="hidden">
    <p id="round-text"></p>
    <img id="player-img" src="">
    <img id="cpu-img" src="">
  </div>
  <button id="start-round" onclick="startRound()">Comenzar Ronda</button>
</div>
```

- Muestra la ronda actual.
- Muestra el conteo hasta que el jugador pueda elegir.
- Las opciones son imágenes clicables.
- Al terminar la ronda, muestra quién ganó y qué eligió cada uno.

Resultados finales:

```
<div id="results" class="hidden">
  <h2>Tabla de Resultados</h2>
  <table>
    <thead>...</thead>
    <tbody id="results-body"></tbody>
  </table>
  <h3 id="final-winner"></h3>
  <button onclick="startGame()">Volver a jugar</button>
  <button onclick="goToMenu()">Regresar al menú</button>
</div>
```

- Tabla que muestra qué eligió cada jugador por ronda.
- Muestra quién ganó en total.
- Permite volver a jugar o regresar al menú.

CSS (estilos visuales)

Los estilos hacen que el juego se vea bonito y claro. Ejemplos:

- .hidden: oculta elementos usando display: none
- img: se les da un tamaño uniforme
- .choices img:hover: hace zoom al pasar el mouse
- .count-1, .count-2, .count-3, .count-ya: estilos de colores y tamaños para el conteo visual

Ejemplo:

```
.count-3 { font-size: 72px; color: red; }
```

JavaScript (lógica del juego)

Aquí es donde se hace toda la magia. Vamos por partes:

1. Variables

```
const options = ['muro', 'conejo', 'flecha'];
const images = {
  muro: 'muro.png',
```

```
  conejo: 'conejo.png',
  flecha: 'flecha.png'
};

let currentRound = 1;
let playerScore = 0;
let cpuScore = 0;
let results = [];
```

- options: lista de opciones posibles.
- images: enlaces a las imágenes locales.
- playerScore, cpuScore, currentRound, results: estado del juego.

2. Función startGame()

```
function startGame() {
  document.getElementById('menu').classList.add('hidden');
  document.getElementById('results').classList.add('hidden');
  document.getElementById('game').classList.remove('hidden');
  reset();
}
```

- Oculta el menú, oculta los resultados, muestra el juego y reinicia datos.

3. Función exitGame()

```
function exitGame() {
  alert('Gracias por jugar. ¡Hasta la próxima!');
  window.location.href = 'https://www.google.com';
}
```

- Redirige a Google porque los navegadores no permiten cerrar ventanas abiertas manualmente por razones de seguridad.

4. startRound(): inicia la ronda con conteo

```
function startRound() {
  ...
  let count = 3;
  const interval = setInterval(() => {
    count--;
    if (count === 0) {
```

```
      clearInterval(interval);
      document.getElementById('countdown').textContent = '¡Ya!';
      updateCountdownStyle('ya');
      document.getElementById('choices').classList.remove('hidden');
    } else {
      document.getElementById('countdown').textContent = count;
      updateCountdownStyle(count);
    }
  }, 1000);
}
```

- Hace un conteo regresivo: 3, 2, 1, ¡Ya!
- Luego muestra las opciones para que el jugador elija.

5. playerSelect(choice): cuando el jugador hace clic

```
function playerSelect(choice) {
  const cpuChoice = options[Math.floor(Math.random() * 3)];
  let winner = 'Empate';
  if (choice !== cpuChoice) {
    if (
      (choice === 'muro' && cpuChoice === 'flecha') ||
      (choice === 'conejo' && cpuChoice === 'muro') ||
      (choice === 'flecha' && cpuChoice === 'conejo')
    ) {
      playerScore++;
      winner = 'Jugador';
    } else {
      cpuScore++;
      winner = 'CPU';
    }
  }
  results.push({ round: currentRound, player: choice, cpu: cpuChoice, winner });
  showRoundResult(choice, cpuChoice, winner);
  if (currentRound === 5) {
    setTimeout(showResults, 1500);
  } else {
    currentRound++;
    document.getElementById('round-number').textContent = currentRound;
```

```
    setTimeout(() => {
      document.getElementById('start-round').classList.remove('hidden');
      document.getElementById('start-round').disabled = false;
    }, 1500);
  }
}
```

- La CPU elige al azar.
- Se comparan elecciones para determinar el ganador.
- Se guarda el resultado.
- Si es la última ronda, se muestra la tabla final.

6. Mostrar resultado de la ronda

```
function showRoundResult(player, cpu, winner) {
  document.getElementById('player-img').src = images[player];
  document.getElementById('cpu-img').src = images[cpu];
  document.getElementById('round-text').textContent = `Ganador de la ronda: ${winner}`;
  document.getElementById('round-result').classList.remove('hidden');
  document.getElementById('choices').classList.add('hidden');
}
```

7. Mostrar resultados finales

```
function showResults() {
  document.getElementById('game').classList.add('hidden');
  document.getElementById('results').classList.remove('hidden');
  const tbody = document.getElementById('results-body');
  tbody.innerHTML = '';
  results.forEach((res, index) => {
    const row = document.createElement('tr');
    row.innerHTML = `
      <td>${index + 1}</td>
      <td>${res.player}</td>
      <td>${res.cpu}</td>
      <td>${res.winner}</td>
    `;
    tbody.appendChild(row);
  });
```

```
  const final = playerScore > cpuScore ? '¡Ganaste!' :
                playerScore < cpuScore ? 'Perdiste.' : 'Empate.';
  document.getElementById('final-winner').textContent = `Resultado final: ${final}`;
}
```

- Construye dinámicamente la tabla de resultados.
- Muestra quién ganó en total.

Repositorio de GitHub

https://github.com/AlejandroFranco1101/muro-conejo-flecha.git