

**Tema Resumen:** Tic-Tac-Toe con AI y Temporizador

 Un temporizador de 15 segundos para cada turno.

 La capacidad de interactuar con un robot que actúa como oponente.

 Una interfaz simple y responsiva.

 Lógica AI basada en el algoritmo minimax para que el robot realice movimientos óptimos.

 Funciones para gestionar el inicio del juego, mover el robot, y detectar el fin del juego.

Index.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Tic-Tac-Toe</title>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: #2b2b2b; /\* Fondo oscuro elegante \*/

color: #f0e6d2; /\* Texto en color dorado claro \*/

margin: 0;

padding: 0;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

}

#main-container {

text-align: center;

margin-bottom: 20px; /\* Espacio debajo del contenedor principal \*/

}

#dialog, #timer, #score {

background-color: #3e3e3e; /\* Fondo de diálogo oscuro \*/

padding: 15px;

border-radius: 10px;

box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.5);

margin-bottom: 15px;

text-align: center;

font-size: 16px;

width: 90%; /\* Ancho adaptable \*/

max-width: 250px; /\* Ancho máximo \*/

box-sizing: border-box; /\* Incluye padding dentro del ancho total \*/

}

#timer {

margin-bottom: 30px; /\* Espacio adicional debajo del cuadro de tiempo restante \*/

margin-top: 20px; /\* Espacio adicional arriba del cuadro de tiempo restante \*/

}

input[type="text"], button {

width: 90%; /\* Ancho adaptable \*/

max-width: 150px; /\* Ancho máximo \*/

height: 40px; /\* Altura pequeña \*/

padding: 5px;

margin: 5px 0;

font-size: 14px;

border: 1px solid #8c7851; /\* Borde dorado oscuro \*/

border-radius: 5px;

box-sizing: border-box; /\* Incluye padding dentro del ancho total \*/

background-color: #4e4e4e; /\* Fondo del cuadro de texto y botones \*/

color: #f0e6d2; /\* Texto dorado claro \*/

}

button {

background-color: #8c7851; /\* Botones dorado oscuro \*/

color: #ffffff; /\* Texto blanco \*/

border: none;

cursor: pointer;

transition: background-color 0.3s;

display: inline-block;

text-align: center;

}

button:hover {

background-color: #a89060; /\* Efecto hover más claro \*/

}

button:active {

background-color: #705936; /\* Efecto al hacer click \*/

}

#board {

display: grid;

grid-template-columns: repeat(3, 80px); /\* Ajusta el tamaño de las celdas \*/

grid-gap: 10px;

margin-top: 20px;

}

.cell {

background-color: #4e4e4e; /\* Celdas de tablero gris oscuro \*/

color: #f0e6d2; /\* Texto dorado claro \*/

font-size: 28px;

display: flex;

align-items: center;

justify-content: center;

width: 80px; /\* Tamaño pequeño para las celdas \*/

height: 80px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.cell:hover {

background-color: #5e5e5e; /\* Efecto hover celdas \*/

}

/\* Media Queries para dispositivos móviles \*/

@media (max-width: 768px) {

#board {

grid-template-columns: repeat(3, 60px); /\* Reduce el tamaño de las celdas en pantallas más pequeñas \*/

}

.cell {

width: 60px; /\* Tamaño pequeño para las celdas \*/

height: 60px;

font-size: 24px; /\* Ajusta el tamaño del texto \*/

}

input[type="text"], button {

width: 80%; /\* Ancho adaptable en pantallas más pequeñas \*/

max-width: none; /\* Elimina el ancho máximo \*/

}

#dialog, #timer, #score {

width: 90%; /\* Ancho adaptable \*/

max-width: none; /\* Elimina el ancho máximo \*/

}

}

</style>

</head>

<body>

<div id="main-container">

<!-- Contenedor principal ahora sin imagen -->

</div>

<div id="dialog"></div>

<div id="board"></div>

<div id="timer">Tiempo restante: 15s</div>

<div id="score">Puntos: 0</div>

<input type="text" id="nameInput" placeholder="Escribe tu nombre" style="display:none;">

<button id="submitResponseButton" style="display:none;">Enviar</button>

<button id="tipButton">Dar propina</button>

<button id="wakeButton">Despertar al robot</button>

<button id="yesButton" style="display:none;">Sí</button>

<button id="noButton" style="display:none;">No</button>

<button id="resetButton">Reiniciar</button>

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

</body>

</html>

Javascript

document.addEventListener('DOMContentLoaded', () => {

const dialog = document.getElementById('dialog');

const nameInput = document.getElementById('nameInput');

const submitResponseButton = document.getElementById('submitResponseButton');

const tipButton = document.getElementById('tipButton');

const wakeButton = document.getElementById('wakeButton');

const yesButton = document.getElementById('yesButton');

const noButton = document.getElementById('noButton');

const resetButton = document.getElementById('resetButton');

const timerDisplay = document.getElementById('timer');

const scoreDisplay = document.getElementById('score');

const board = document.getElementById('board');

let timer;

let timeLeft = 15;

let playerDefeats = 0;

let score = 0;

let currentPlayer = 'X';

let boardState = [];

let gameStarted = false;

let playerName = '';

function wakeUpRobot() {

dialog.innerHTML = 'Robot: ¿Cómo te llamas?';

nameInput.style.display = 'inline';

submitResponseButton.style.display = 'inline';

submitResponseButton.addEventListener('click', handleNameResponse, { once: true });

}

function handleNameResponse() {

playerName = nameInput.value.trim();

if (playerName) {

dialog.innerHTML = `Robot: Hola ${playerName}, bienvenido al juego.<br>¿Quieres jugar conmigo?`;

nameInput.style.display = 'none';

submitResponseButton.style.display = 'none';

yesButton.style.display = 'inline';

noButton.style.display = 'inline';

yesButton.addEventListener('click', () => {

yesButton.style.display = 'none';

noButton.style.display = 'none';

startGame();

}, { once: true });

noButton.addEventListener('click', () => {

noButton.style.display = 'none';

dialog.innerHTML = `Robot: ¡Está bien, ${playerName}! Si cambias de opinión, estaré aquí.`;

}, { once: true });

} else {

alert("Por favor, ingresa tu nombre.");

}

}

function startGame() {

if (!gameStarted) {

dialog.innerHTML = `Robot: Me llamo Michel Rosales. Si pierdes, no te daré una computadora; si ganas, la tendrás en tus manos..<br>¡Vamos a jugar, ${playerName}!`;

wakeButton.style.display = 'none';

createBoard();

startTimer();

gameStarted = true;

}

}

function giveTip() {

dialog.innerHTML = 'Robot: Muchísimas gracias por tu propina!';

setTimeout(() => {

dialog.innerHTML += '<br>¡Vamos a jugar Tic-Tac-Toe!';

setTimeout(() => {

wakeButton.style.display = 'inline';

}, 2000);

}, 2000);

}

function startTimer() {

timeLeft = 15;

timerDisplay.textContent = `Tiempo restante: ${timeLeft}s`;

timer = setInterval(() => {

timeLeft--;

timerDisplay.textContent = `Tiempo restante: ${timeLeft}s`;

if (timeLeft <= 0) {

clearInterval(timer);

endGame(null);

}

}, 1000);

}

function createBoard() {

board.innerHTML = '';

boardState = ['', '', '', '', '', '', '', '', ''];

for (let i = 0; i < 9; i++) {

const cell = document.createElement('div');

cell.classList.add('cell');

cell.dataset.index = i;

cell.addEventListener('click', handleClick);

board.appendChild(cell);

}

}

function handleClick(e) {

if (!gameStarted) return;

const index = e.target.dataset.index;

if (boardState[index] === '' && currentPlayer === 'X') {

boardState[index] = 'X';

e.target.textContent = 'X';

checkWinner();

currentPlayer = 'O';

clearInterval(timer);

aiMove();

}

}

function aiMove() {

setTimeout(() => {

const bestMove = findBestMove();

if (bestMove !== -1) {

boardState[bestMove] = 'O';

document.querySelector(`.cell[data-index='${bestMove}']`).textContent = 'O';

checkWinner();

currentPlayer = 'X';

startTimer();

}

}, 500);

}

function findBestMove() {

let bestValue = -Infinity;

let move = -1;

for (let i = 0; i < boardState.length; i++) {

if (boardState[i] === '') {

boardState[i] = 'O';

let moveValue = minimax(boardState, 0, false);

boardState[i] = '';

if (moveValue > bestValue) {

bestValue = moveValue;

move = i;

}

}

}

return move;

}

function minimax(board, depth, isMaximizing) {

const scores = { 'O': 10, 'X': -10, 'tie': 0 };

const result = checkGameOver(board);

if (result !== null) return scores[result];

if (isMaximizing) {

let best = -Infinity;

for (let i = 0; i < board.length; i++) {

if (board[i] === '') {

board[i] = 'O';

best = Math.max(best, minimax(board, depth + 1, false));

board[i] = '';

}

}

return best;

} else {

let best = Infinity;

for (let i = 0; i < board.length; i++) {

if (board[i] === '') {

board[i] = 'X';

best = Math.min(best, minimax(board, depth + 1, true));

board[i] = '';

}

}

return best;

}

}

function checkGameOver(board) {

const winPatterns = [

[0, 1, 2],

[3, 4, 5],

[6, 7, 8],

[0, 3, 6],

[1, 4, 7],

[2, 5, 8],

[0, 4, 8],

[2, 4, 6]

];

for (let pattern of winPatterns) {

const [a, b, c] = pattern;

if (board[a] && board[a] === board[b] && board[a] === board[c]) {

return board[a];

}

}

return board.includes('') ? null : 'tie';

}

function checkWinner() {

const winner = checkGameOver(boardState);

if (winner) {

endGame(winner);

}

}

function endGame(winner) {

clearInterval(timer);

if (winner === 'X') {

dialog.innerHTML = `¡Ganaste, ${playerName}!`;

score++;

} else if (winner === 'O') {

dialog.innerHTML = `¡Perdiste, ${playerName}!`;

playerDefeats++;

} else {

dialog.innerHTML = `Empate, ${playerName}!`;

}

updateScore();

dialog.innerHTML += `<br><button id="retryButton">Volver a Intentar</button>`;

document.getElementById('retryButton').addEventListener('click', () => {

resetGame();

startGame();

});

nameInput.style.display = 'none';

submitResponseButton.style.display = 'none';

yesButton.style.display = 'none';

noButton.style.display = 'none';

wakeButton.style.display = 'none';

}

function updateScore() {

scoreDisplay.textContent = `Puntos: ${score}`;

}

function resetGame() {

clearInterval(timer);

dialog.innerHTML = '';

nameInput.value = '';

nameInput.style.display = 'inline';

submitResponseButton.style.display = 'none';

yesButton.style.display = 'none';

noButton.style.display = 'none';

wakeButton.style.display = 'inline';

boardState = [];

currentPlayer = 'X';

createBoard();

timerDisplay.textContent = 'Tiempo restante: 15s';

gameStarted = false;

}

tipButton.addEventListener('click', giveTip);

wakeButton.addEventListener('click', wakeUpRobot);

resetButton.addEventListener('click', resetGame);

});