**JUEGO DE MEMORIA**

**Metodología resolución del problema**

Descripción Breve

Juego de cartas donde el objetivo es avanzar de nivel consiguiendo el par de todas las cartas sin perder las vidas entregadas y en el menor tiempo posible.

Gustavo Adolfo Sandoval Sandoval

Pablo Andrés Lucero Alvarado

**Contenido**

[**Descripción del problema** 1](#_Descripción_del_problema)

[**Autor** 1](#_Toc168604980)

[**Algoritmo** 1](#_Toc168604981)

[**Diccionario de datos** 2](#_Toc168604982)

[**Código fuente** 2](#_Toc168604983)

[**Manual de usuario y paso a paso** 11](#_Toc168604984)

# **Descripción del problema**

Crear un sistema de donde se nos muestren un cierto número de cartas con su respectivo par y que, al hacer coincidir esos 2 pares, se den la vuelta permanentemente y que sumen puntos hasta encontrar todos los pares y llegar a 100 puntos. Así como que se muestre un contador que correrá cuando presionemos el botón de comienzo y que podemos avanzar o retroceder de nivel, así como reintentar el nivel si lo perdemos. Se tendrán un cierto número de vidas en cada uno de los 3 niveles, si perdemos todas las vidas perderemos el nivel, en el caso contrario ganaremos.

# **Autor**

Pablo Andrés Lucero Alvarado

# **Algoritmo**

# 

# **Diccionario de datos**

num1, num2 = int

puntos = int

vidas = int

aleatorio = int

# **Código fuente**

**HTML**

<!DOCTYPE html>

<html lang="en">

<head>

    <!-- Metadatos y configuraciones -->

    <meta charset="UTF-8">

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

    <title>Juego de Memoria</title>

    <!-- Enlace a la hoja de estilos externa -->

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div id="game-container">

        <!-- Encabezado del juego con información del jugador -->

        <div id="header">

            <h1>Juego de Memoria</h1>

            <div id="level-info">Nivel: Fácil</div>

            <div id="time">Tiempo: 0s</div>

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

            <div id="lives">Vidas: 10</div>

        </div>

        <!-- Contenedor del tablero del juego -->

        <div id="game-board"></div>

        <!-- Contenedor para mensajes durante el juego -->

        <div id="message-container"></div>

        <!-- Botón para comenzar el juego -->

        <button id="start-button">Comenzar</button>

        <!-- Botón para reintentar el nivel (oculto al inicio) -->

        <button id="retry-button" style="display: none;">Reintentar</button>

        <!-- Botón para volver al nivel anterior (oculto al inicio) -->

        <button id="prev-level-button" style="display: none;">Nivel Anterior</button>

        <!-- Botón para avanzar al siguiente nivel -->

        <button id="next-level-button">Siguiente Nivel</button>

    </div>

    <!-- Enlace al archivo de script que contiene la lógica del juego -->

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

</body>

</html>

**JavaScript**

// Variables para controlar el estado del juego

let firstClick = false;

let score = 0;

let level = 1;

let totalPairs = 6; // Fácil

const basePointsPerPair = 100;

let pointsPerPair = Math.floor(basePointsPerPair / totalPairs);

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let flipSpeed = 500; // Velocidad de volteo en milisegundos

let timer = null;

let startTime = null;

let lives = 10;

// Constante para la puntuación máxima

const maxScore = 100;

// Función que se ejecuta cuando el documento HTML ha sido completamente cargado

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

    // Obtener elementos del DOM

    const gameBoard = document.getElementById("game-board");

    const scoreContainer = document.getElementById("score");

    const messageContainer = document.getElementById("message-container");

    const retryButton = document.getElementById("retry-button");

    const prevLevelButton = document.getElementById("prev-level-button");

    const nextLevelButton = document.getElementById("next-level-button");

    const startButton = document.getElementById("start-button");

    const levelInfo = document.getElementById("level-info");

    const timeContainer = document.getElementById("time");

    const livesContainer = document.getElementById("lives");

    // Configurar el nivel actual en el encabezado del juego

    levelInfo.innerText = `Nivel: ${getLevelName(level)}`;

    // Crear el tablero del juego

    createBoard();

    // Agregar event listener al botón de inicio

    startButton.addEventListener("click", () => {

        // Iniciar el juego

        revealAllCards();

        setTimeout(() => {

            hideAllCards();

            startGameTimer();

            startButton.style.display = 'none';

        }, 1000);

    });

    // Agregar event listener al botón de reintentar

    retryButton.addEventListener("click", () => {

        // Reiniciar el juego

        resetGame();

    });

    // Agregar event listener al botón de nivel anterior

    prevLevelButton.addEventListener("click", () => {

        if (level > 1) {

            level--;

            startNewLevel();

        }

    });

    // Agregar event listener al botón de siguiente nivel

    nextLevelButton.addEventListener("click", () => {

        if (level < 3) {

            level++;

            startNewLevel();

        }

    });

    // Función para crear el tablero del juego

    function createBoard() {

        // Limpiar el tablero

        gameBoard.innerHTML = '';

        // Generar valores para las cartas y crearlas

        const cardValues = generateCardValues(totalPairs);

        cardValues.forEach(value => {

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

            card.classList.add('card');

            card.innerHTML = `

                <div class="front"></div>

                <div class="back">${value}</div>

            `;

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

                // Voltear la carta al hacer clic

                if (!lockBoard && !card.classList.contains('flipped')) {

                    flipCard(card);

                    if (!firstCard) {

                        firstCard = card;

                    } else {

                        secondCard = card;

                        checkForMatch();

                    }

                }

            });

            gameBoard.appendChild(card);

        });

        // Configurar la disposición de las cartas en el tablero

        const columns = (level === 3) ? totalPairs : (level === 2) ? Math.ceil(totalPairs / 2) : totalPairs;

        const rows = (level === 2) ? 2 : Math.ceil(totalPairs / columns);

        gameBoard.style.gridTemplateColumns = `repeat(${columns}, 50px)`;

        gameBoard.style.gridTemplateRows = `repeat(${rows}, 50px)`;

    }

    // Función para revelar todas las cartas al comenzar el juego

    function revealAllCards() {

        const cards = document.querySelectorAll('.card');

        cards.forEach(card => card.classList.add('flipped'));

    }

    // Función para ocultar todas las cartas al comenzar el juego

    function hideAllCards() {

        const cards = document.querySelectorAll('.card');

        cards.forEach(card => card.classList.remove('flipped'));

    }

    // Función para generar los valores de las cartas del juego

    function generateCardValues(pairsCount) {

        const values = [];

        const alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        // Generar pares de letras para cada nivel

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

            if (level === 1 || (level === 2 && pairsCount === 6) || (level === 3 && i < pairsCount / 2)) {

                const letter = alphabet[i % alphabet.length];

                values.push(letter, letter);

            } else if (level === 2 || (level === 3 && i >= pairsCount / 2)) {

                const letter1 = alphabet[i % alphabet.length];

                const letter2 = alphabet[(i + 1) % alphabet.length];

                values.push(letter1 + letter2, letter1 + letter2);

            }

        }

        return shuffleArray(values); // Barajar los valores

    }

    // Función para barajar un array

    function shuffleArray(array) {

        for (let i = array.length - 1; i > 0; i--) {

            const j = Math.floor(Math.random() \* (i + 1));

            [array[i], array[j]] = [array[j], array[i]];

        }

        return array;

    }

    // Función para voltear una carta

    function flipCard(card) {

        card.classList.add('flipped');

    }

    // Función para verificar si las dos cartas coinciden

    function checkForMatch() {

        if (firstCard.querySelector('.back').textContent === secondCard.querySelector('.back').textContent) {

            disableCards();

            incrementScore();

            if (checkAllPairsMatched()) {

                endGame();

            }

        } else {

            loseLife();

            unflipCards();

        }

    }

    // Función para deshabilitar dos cartas coincidentes

    function disableCards() {

        firstCard.removeEventListener('click', flipCard);

        secondCard.removeEventListener('click', flipCard);

        resetBoard();

    }

   // Función para incrementar la puntuación

    function incrementScore() {

    score += pointsPerPair;

    if (checkAllPairsMatched()) {

        score = maxScore;

    }

    scoreContainer.innerText = `Puntuación: ${score}`;

    }

    // Función para verificar si se han encontrado todos los pares

    function checkAllPairsMatched() {

        return document.querySelectorAll('.card.flipped').length === totalPairs \* 2;

    }

    // Función para perder una vida

    function loseLife() {

        lives--;

        livesContainer.innerText = `Vidas: ${lives}`;

        if (lives <= 0) {

            loseGame();

        }

    }

    // Función para voltear las dos cartas si no coinciden

    function unflipCards() {

        lockBoard = true;

        setTimeout(() => {

            firstCard.classList.remove('flipped');

            secondCard.classList.remove('flipped');

            resetBoard();

        }, flipSpeed); // Tiempo corto para mostrar la carta antes de voltear

    }

    // Función para reiniciar el tablero del juego

    function resetBoard() {

        [firstCard, secondCard] = [null, null];

        lockBoard = false;

    }

    // Función para iniciar el temporizador del juego

    function startGameTimer() {

        startTime = new Date();

        timer = setInterval(updateTimer, 1000);

    }

    // Función para detener el temporizador del juego

    function stopGameTimer() {

        clearInterval(timer);

        timer = null;

    }

    // Función para actualizar el temporizador del juego

    function updateTimer() {

        const now = new Date();

        const elapsedTime = Math.floor((now - startTime) / 1000);

        timeContainer.innerText = `Tiempo: ${elapsedTime}s`;

    }

    // Función para finalizar el juego

    function endGame() {

        messageContainer.innerHTML = '<h2>¡Felicitaciones! Has completado el nivel.</h2>';

        retryButton.style.display = "block";

        stopGameTimer();

    }

    // Función para perder el juego

    function loseGame() {

        messageContainer.innerHTML = '<h2 class="lose">¡Has perdido! Inténtalo de nuevo.</h2>';

        messageContainer.classList.add('lose'); // Cambio el color a rojo

        retryButton.style.display = "block";

        stopGameTimer();

    }

    // Función para reiniciar el juego

    function resetGame() {

        score = 0;

        lives = getInitialLives(level);

        scoreContainer.innerText = `Puntuación: ${score}`;

        livesContainer.innerText = `Vidas: ${lives}`;

        messageContainer.innerHTML = '';

        retryButton.style.display = "none";

        startButton.style.display = "block";

        firstClick = false;

        firstCard = null;

        secondCard = null;

        lockBoard = false;

        stopGameTimer();

        timeContainer.innerText = '';

        createBoard(); // Crear nuevamente el tablero

    }

    // Función para iniciar un nuevo nivel

    function startNewLevel() {

        updateLevelSettings();

        levelInfo.innerText = `Nivel: ${getLevelName(level)}`;

        resetGame();

    }

    // Función para actualizar la configuración del nivel

    function updateLevelSettings() {

        switch (level) {

            case 1:

                totalPairs = 8; // Fácil

                break;

            case 2:

                totalPairs = 12; // Medio

                break;

            case 3:

                totalPairs = 16; // Difícil

                break;

        }

        pointsPerPair = Math.floor(basePointsPerPair / totalPairs); // Actualizamos los puntos por par

        flipSpeed = Math.max(100, 500 - (level - 1) \* 50); // Incrementar la velocidad en cada nivel, hasta un mínimo de 100ms

        prevLevelButton.style.display = level > 1 ? "block" : "none";

    }

    // Función para obtener el nombre del nivel

    function getLevelName(level) {

        switch (level) {

            case 1:

                return 'Fácil';

            case 2:

                return 'Medio';

            case 3:

                return 'Difícil';

        }

    }

    // Función para obtener las vidas iniciales según el nivel

    function getInitialLives(level) {

        switch (level) {

            case 1:

                return 10;

            case 2:

                return 12;

            case 3:

                return 14;

        }

    }

});

# **Manual de usuario y paso a paso**



**5**

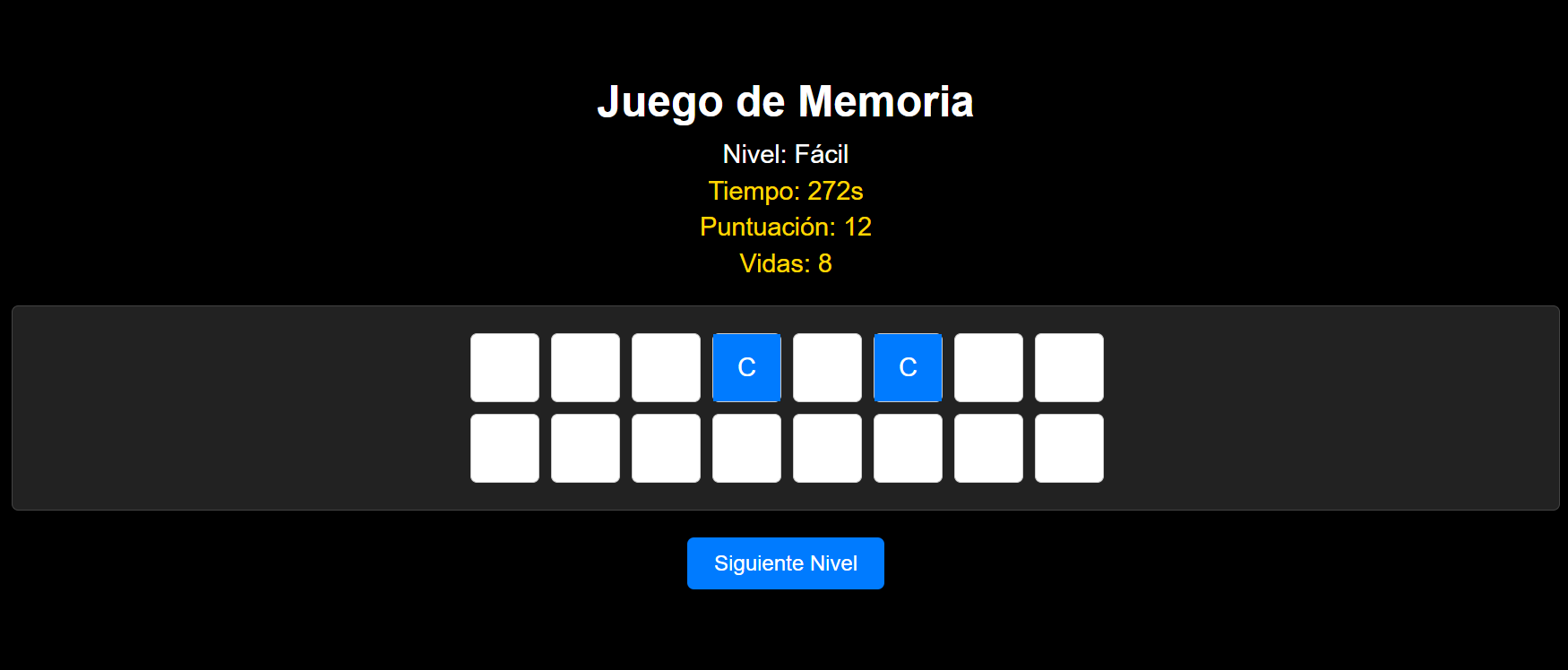
**4**

**3**

**2**

**1**

1. **Nivel de dificultad, sube en cada nivel (fácil, medio, difícil)**
2. **Puntuación que subirá cuando encuentres un par de cartas**
3. **Numero de vidas que se reducirá cuando falles al no encontrar un par**
4. **Botón para comenzar el juego, clic para comenzar**
5. **Botón para pasar al siguiente nivel, clic para cambiar**



**7**

**6**

1. **Selecciona una carta**
2. **Intenta encontrar el par de esa carta**

**Sigue intentando hasta encontrar todos los pares sin perder tus vidas y en el menor tiempo posible.**



**8**

**9**

**8. Cuando encontremos todos los pares se nos mostrará un mensaje de “felicitaciones”.**

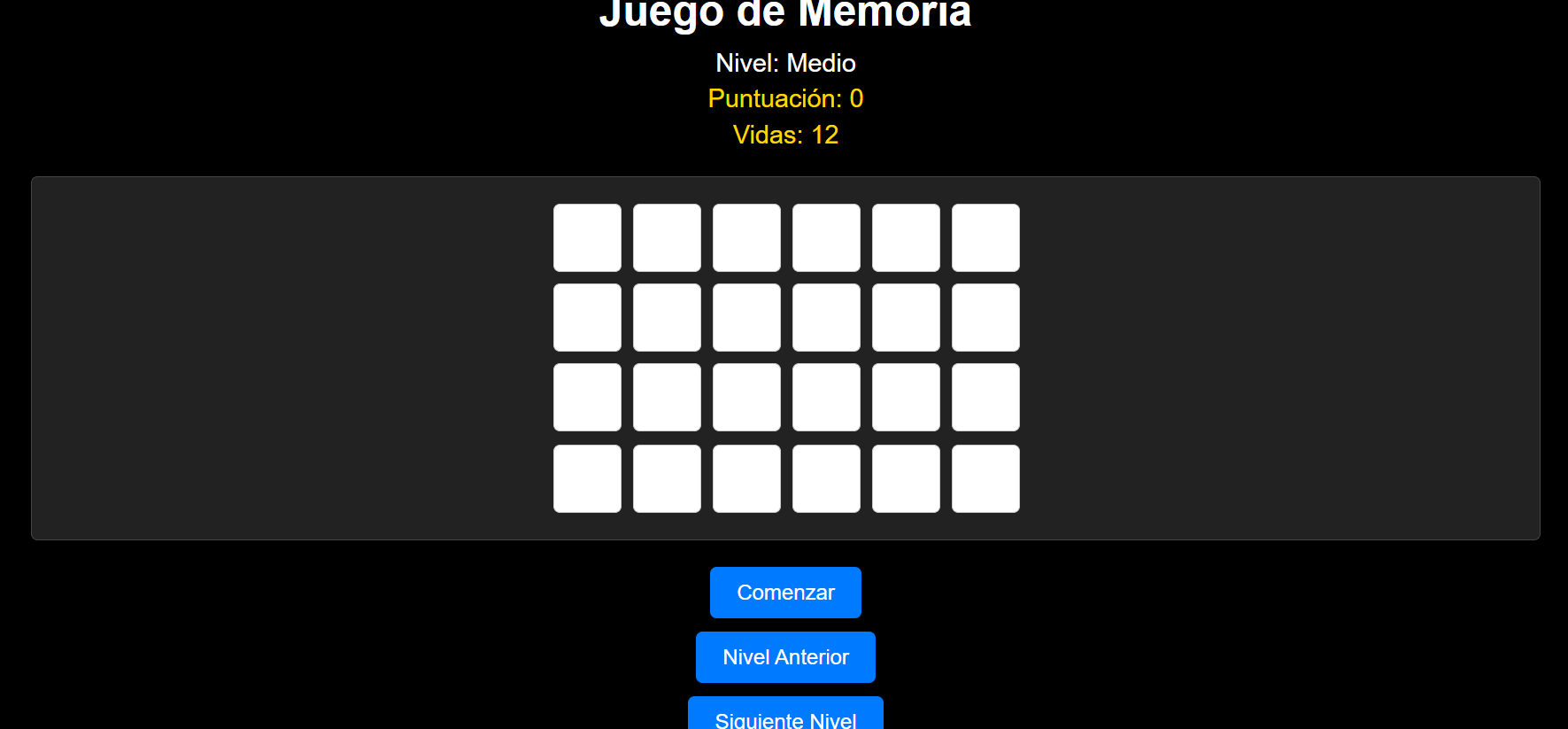
**9**. **Al completar el nivel podremos usar el botón reintentar para volver a jugar el nivel.**

****

**10**

**10. Caso en el que, si perdemos todas nuestras vidas se nos mostrara un mensaje de “has perdido”.**

**Al completar o perder un nivel podremos avanzar de nivel para seguir probando nuestra memoria.**



**11**

**11. Presione el botón de “Nivel Anterior” para retroceder de nivel**