**title: דוגמה למשחק React אינטראקטיבי - Memory Game description: משחק זיכרון מלא ב-React עם hooks, ניהול state מתקדם, אנימציות ומעקב ניקוד tags: [react, game-development, hooks, state-management, animations, component-design, interactive] priority: גבוהה language: he**

**דוגמה למשחק React אינטראקטיבי - Memory Game**

**מבוא**

React הוא כלי מצוין לבניית משחקים אינטראקטיביים בדפדפן. במדריך זה נבנה משחק זיכרון (Memory Game) מלא עם React Hooks, ניהול state מתקדם, אנימציות CSS ומעקב אחר ביצועים. המשחק ידגים עקרונות חשובים בפיתוח משחקים עם React כולל ניהול מצב מורכב, אופטימיזציה ועיצוב רכיבים מודולרי.

**מבנה הפרויקט**

**1. רכיב המשחק הראשי (Game Component)**

import React, { useState, useEffect, useCallback } from 'react';

import './MemoryGame.css';

const MemoryGame = () => {

// ניהול State

const [cards, setCards] = useState([]);

const [flipped, setFlipped] = useState([]);

const [matched, setMatched] = useState([]);

const [moves, setMoves] = useState(0);

const [gameWon, setGameWon] = useState(false);

const [score, setScore] = useState(0);

const [timer, setTimer] = useState(0);

const [isPlaying, setIsPlaying] = useState(false);

const [difficulty, setDifficulty] = useState('medium');

// קונפיגורציה לפי רמת קושי

const difficultyConfig = {

easy: { pairs: 6, timeBonus: 1000, scoreMultiplier: 1 },

medium: { pairs: 8, timeBonus: 800, scoreMultiplier: 1.5 },

hard: { pairs: 12, timeBonus: 600, scoreMultiplier: 2 }

};

// אמוג'ים למשחק

const emojis = ['🎮', '🎯', '🎲', '🎨', '🎭', '🎪', '🎬', '🎤',

'🎧', '🎸', '🥇', '🏆', '⚡', '🔥', '💎', '🌟'];

// אתחול המשחק

const initializeGame = useCallback(() => {

const config = difficultyConfig[difficulty];

const selectedEmojis = emojis.slice(0, config.pairs);

const gameCards = [...selectedEmojis, ...selectedEmojis]

.map((emoji, index) => ({

id: index,

emoji,

uniqueId: `${emoji}-${index}`

}))

.sort(() => Math.random() - 0.5);

setCards(gameCards);

setFlipped([]);

setMatched([]);

setMoves(0);

setGameWon(false);

setScore(0);

setTimer(0);

setIsPlaying(true);

}, [difficulty]);

// טיימר למשחק

useEffect(() => {

let interval;

if (isPlaying && !gameWon) {

interval = setInterval(() => {

setTimer(prev => prev + 1);

}, 1000);

}

return () => clearInterval(interval);

}, [isPlaying, gameWon]);

// בדיקת ניצחון

useEffect(() => {

if (matched.length > 0 && matched.length === cards.length) {

handleGameWon();

}

}, [matched, cards]);

// טיפול בלחיצה על קלף

const handleCardClick = useCallback((id) => {

if (gameWon || flipped.includes(id) || matched.includes(id)) {

return;

}

if (flipped.length === 0) {

setFlipped([id]);

} else if (flipped.length === 1) {

setFlipped([...flipped, id]);

setMoves(moves + 1);

// בדיקת התאמה

const firstCard = cards.find(card => card.id === flipped[0]);

const secondCard = cards.find(card => card.id === id);

if (firstCard.emoji === secondCard.emoji) {

// התאמה!

setTimeout(() => {

setMatched([...matched, flipped[0], id]);

setFlipped([]);

calculateScore(true);

}, 500);

} else {

// אין התאמה

setTimeout(() => {

setFlipped([]);

calculateScore(false);

}, 1000);

}

}

}, [flipped, matched, cards, moves, gameWon]);

// חישוב ניקוד

const calculateScore = (isMatch) => {

const config = difficultyConfig[difficulty];

if (isMatch) {

const timeBonus = Math.max(0, config.timeBonus - timer \* 10);

const matchScore = 100 \* config.scoreMultiplier;

setScore(prev => prev + matchScore + timeBonus);

} else {

setScore(prev => Math.max(0, prev - 10));

}

};

// טיפול בניצחון

const handleGameWon = () => {

setGameWon(true);

setIsPlaying(false);

const finalScore = score + Math.max(0, 5000 - timer \* 50 - moves \* 20);

setScore(finalScore);

};

// פורמט זמן

const formatTime = (seconds) => {

const mins = Math.floor(seconds / 60);

const secs = seconds % 60;

return `${mins.toString().padStart(2, '0')}:${secs.toString().padStart(2, '0')}`;

};

return (

<div className="memory-game">

<Header

score={score}

moves={moves}

timer={formatTime(timer)}

difficulty={difficulty}

onDifficultyChange={setDifficulty}

onNewGame={initializeGame}

/>

<GameBoard

cards={cards}

flipped={flipped}

matched={matched}

onCardClick={handleCardClick}

difficulty={difficulty}

/>

{gameWon && (

<WinModal

score={score}

moves={moves}

time={formatTime(timer)}

onPlayAgain={initializeGame}

/>

)}

</div>

);

};

export default MemoryGame;

**2. רכיב הכותרת (Header Component)**

const Header = ({ score, moves, timer, difficulty, onDifficultyChange, onNewGame }) => {

return (

<div className="game-header">

<h1>משחק הזיכרון</h1>

<div className="stats">

<div className="stat">

<span className="stat-label">ניקוד:</span>

<span className="stat-value">{score}</span>

</div>

<div className="stat">

<span className="stat-label">מהלכים:</span>

<span className="stat-value">{moves}</span>

</div>

<div className="stat">

<span className="stat-label">זמן:</span>

<span className="stat-value">{timer}</span>

</div>

</div>

<div className="controls">

<select

value={difficulty}

onChange={(e) => onDifficultyChange(e.target.value)}

className="difficulty-select"

>

<option value="easy">קל (6