TIC-TAC-TOE

PROJECT DOCUMENTATION

1. Introduction

- **1.1 Overview**

The Tic-Tac-Toe game is a classic two-player game played on a 3x3 grid. Players take turns marking spaces with their respective symbols, typically "X" and "O," with[1] the goal of achieving a row, column, or diagonal of three of their own symbols. This digital version, "Tic-Tac-Toe: Bluey Black Enhanced," provides an engaging and visually appealing rendition of the game, incorporating a modern design and user-friendly interface.

- **1.2 Purpose**

This project aims to create an interactive and enjoyable digital version of Tic-Tac-Toe that can be played directly in a web browser. It focuses on delivering a clean, intuitive gaming experience with a stylish aesthetic.

2. Project Goals

- **2.1 Core Objectives**
    - Implement the complete rules and logic of the Tic-Tac-Toe game.
    - Create a user-friendly interface for players to interact with the game.
    - Provide clear visual feedback to players during gameplay.
    - Accurately track and display the game's score (wins for each player).
    - Allow players to easily restart the game.
- **2.2 Aesthetic Goals**
    - Develop a visually appealing dark theme with vibrant highlights.
    - Use clear and stylish typography.
    - Incorporate smooth transitions and hover effects for enhanced interactivity.

3. Features and Working

- **3.1 Game Features**
    - **Two-Player Gameplay:** Supports turn-based gameplay for two players (X and O).

- **Interactive Board:** A 3x3 grid where players can click on empty cells to make their move.
- **Turn Management:** Clearly indicates the current player's turn.
- **Win Detection:** Automatically detects a win when a player achieves three in a row (horizontally, vertically, or diagonally).
- **Draw Detection:** Recognizes a draw when all cells are filled and no player has won.
- **Score Tracking:** Keeps track of and displays the number of wins for each player (X and O).
- **Game Restart:** Provides a button to easily restart the game, resetting the board and keeping the score.
- **Visual Feedback:** Updates the game board with X or O symbols, highlights the winning player, and provides status messages.

- **3.2 Game Working**

1. **Initialization:**
    - The game board is created as a 3x3 grid of clickable cells.
    - The initial player is set to "X."
    - The game status message indicates Player X's turn.
    - The score is initialized to 0 for both players.
2. **Player Turn:**
    - The current player clicks on an empty cell.
    - The cell is marked with the player's symbol (X or O).
    - The game checks for a win or a draw.
3. **Win Condition:**
    - If a win is detected (three in a row), the winning player is declared.
    - The score is updated.
    - The board is disabled to prevent further moves.
4. **Draw Condition:**
    - If all cells are filled and no win is detected, the game is declared a draw.

5. **Turn Switch:**
    - If neither a win nor a draw occurs, the turn switches to the other player.
    - The game status message is updated to reflect the current player's turn.
6. **Game Restart:**
    - Clicking the "Restart" button resets the board to its initial empty state, sets the current player to "X," and updates the status message. The score is retained.

4. Technology Used

- **4.1 HTML (Hypertext Markup Language)**
    - Used to structure the web page, including the game board, headings, status messages, and buttons.
    - Provides the basic layout and elements of the game.
- **4.2 CSS (Cascading Style Sheets)**
    - Used for styling the game, including the color scheme, fonts, layout, and visual effects.
    - Creates the dark theme, styles the board and cells, and adds hover effects and transitions.
- **4.3 JavaScript**
    - Used to implement the game logic, handle user interactions, update the display, and manage the game state.
    - Handles player turns, win/draw conditions, score tracking, and the restart functionality.

5. Implementation Details

- **5.1 HTML Structure**
    - The main elements are:
        - <h1> for the game title.
        - to display the scores.
        - to represent the Tic-Tac-Toe grid.
        - to show game messages.

        - <button class="restart-btn"> to restart the game.
    - The game board ( ) is structured using a CSS Grid Layout.
    - Each cell in the board is a element with the class "cell."
- **5.2 CSS Styling**
    - **Color Scheme:** A dark background with blue and purple accents is used to create a visually appealing theme.
    - **Grid Layout:** CSS Grid is used to create the 3x3 game board, making it easy to arrange the cells.
    - **Typography:** The "Segoe UI" font is used for a clean and modern look.
    - **Visual Effects:** Hover effects, transitions, and text shadows are used to enhance the user experience.
- **5.3 JavaScript Logic**
    - **Variables:**
        - board: References the game board element.
        - statusText: References the status message element.
        - xScore, oScore: References the score display elements.
        - currentPlayer: Stores the current player ('X' or 'O').
        - cells: An array of length 9 to represent the state of each cell on the board.
        - score: An object to track the scores of players X and O.
    - **Functions:**
        - createBoard(): Dynamically creates the 3x3 grid by appending elements to the board. Each cell gets a click event listener.
        - handleClick(e): Handles cell clicks, updates the cells array, checks for wins/draws, and switches players.
        - checkWin(): Checks for a winning combination using predefined winning patterns.
        - disableBoard(): Removes the click event listener from all cells, preventing further moves after a win.
        - updateScore(): Updates the displayed scores.

- restartGame(): Resets the game board and status.

6. Theme

- **6.1 Color Palette**
  - Background: Dark radial gradient (#0a0f1a to #000010)
  - Text: Light colors (#cde9ff, #88ccff, #aaccff)
  - Player X Color: Light blue (#33ccff)
  - Player O Color: Light purple (#ff99ff)
  - Board Cells: Dark gray (#0b1320, #152237 on hover)
  - Accent: Bright blue for title and button highlights (#66ccff, #00bfff, #007acc)
- **6.2 Typography**
  - Font: Segoe UI
  - Heading: Larger font size with text shadow for emphasis
  - Status and Score: Clear and readable font sizes
- **6.3 Visual Style**
  - Modern and clean design
  - Smooth transitions and hover effects for interactivity
  - Subtle gradients and shadows for depth

7. Code Snippets

- **7.1 Creating the Board**

JavaScript

```
function createBoard() {
  board.innerHTML = '';
  cells.forEach((_, index) => {
    const cell = document.createElement('div');
    cell.classList.add('cell');
    cell.dataset.index = index;
    cell.addEventListener('click', handleClick);
    board.appendChild(cell);
```

```
    });
}
```

- **7.2 Handling Cell Clicks**

JavaScript

```
function handleClick(e) {
    const index = e.target.dataset.index;
    if (cells[index] !== '') return;

    cells[index] = currentPlayer;
    e.target.textContent = currentPlayer;
    e.target.classList.add(currentPlayer.toLowerCase());

    if (checkWin()) {
        statusText.textContent = `🎉 Player ${currentPlayer} Wins!`;
        score[currentPlayer]++;
        updateScore();
        disableBoard();
        return;
    }

    if (cells.every(cell => cell !== '')) {
        statusText.textContent = "It's a Draw!";
        return;
    }

    currentPlayer = currentPlayer === 'X' ? 'O' : 'X';
    statusText.textContent = `Player ${currentPlayer}'s Turn`;
}
```

- **7.3 Checking for a Win**

JavaScript

```
function checkWin() {
   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]
   ];
   return winPatterns.some(pattern =>
      pattern.every(i => cells[i] === currentPlayer)
   );
}
```

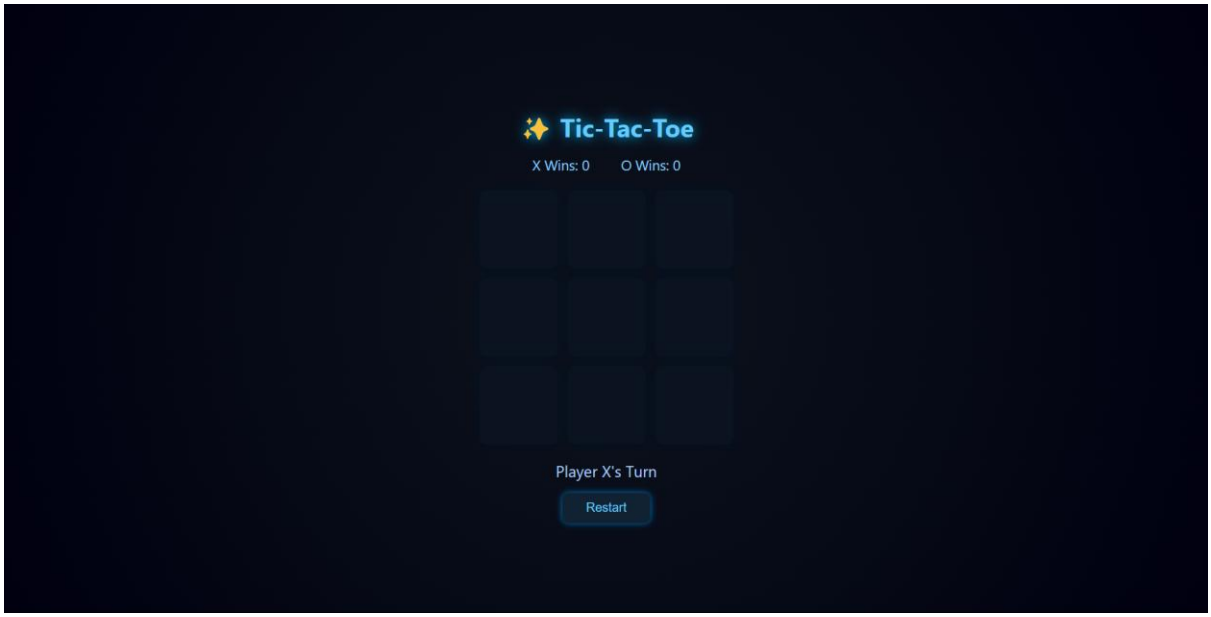
- **7.4 Restarting the Game**

JavaScript

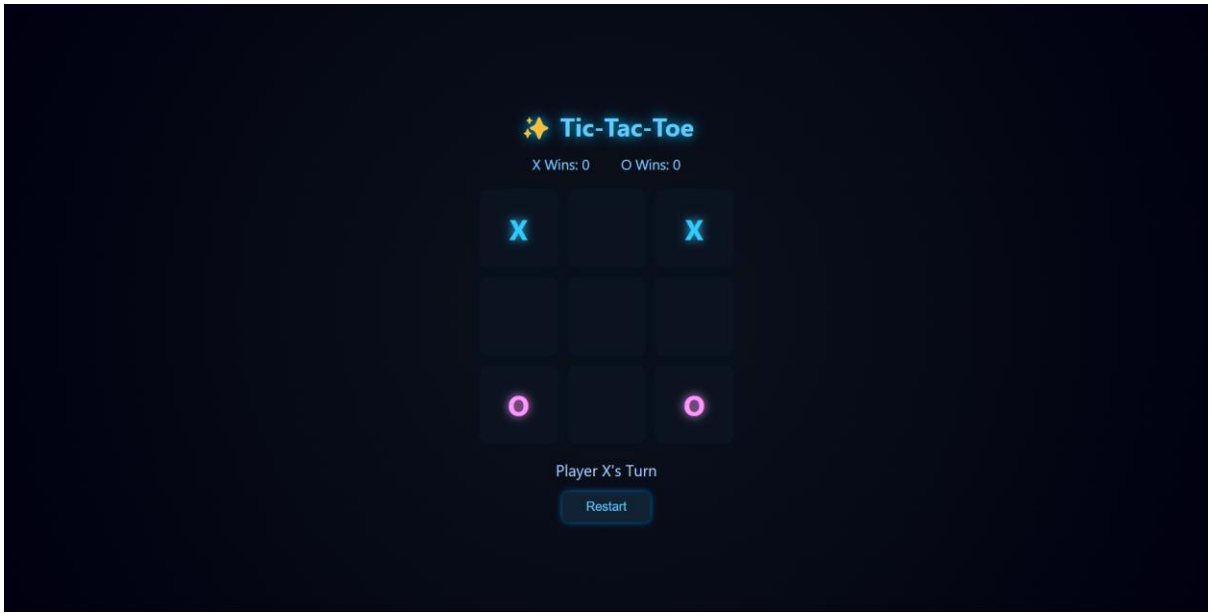
```
function restartGame() {
   cells = Array(9).fill('');
   currentPlayer = 'X';
   statusText.textContent = `Player ${currentPlayer}'s Turn`;
   createBoard();
}
```

8. Output Photos

- **8.1 Initial State**
    - The game displays the title "Tic-Tac-Toe."
    - The scoreboard shows "X Wins: 0" and "O Wins: 0."
    - The 3x3 grid is empty, with dark gray cells.
    - The status message reads "Player X's Turn."
    - The "Restart" button is visible.

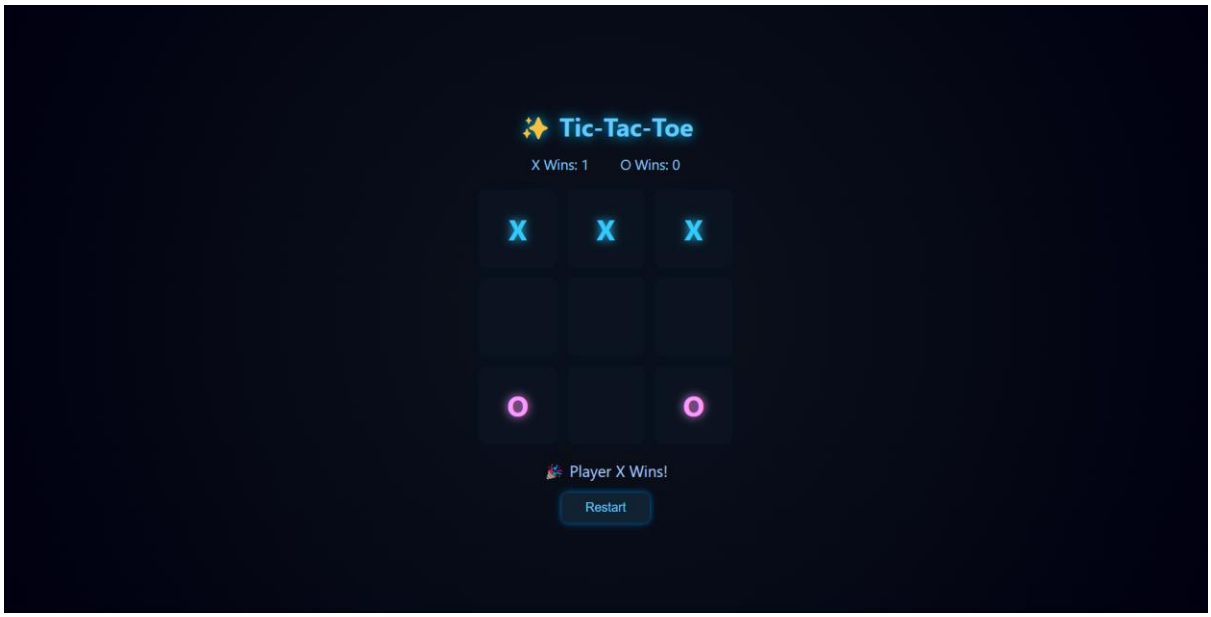


- **8.2 During Gameplay**
    - As players click on cells, "X" and "O" symbols appear in the respective cells, with "X" in light blue and "O" in light purple.
    - The status message alternates between "Player X's Turn" and "Player O's Turn."

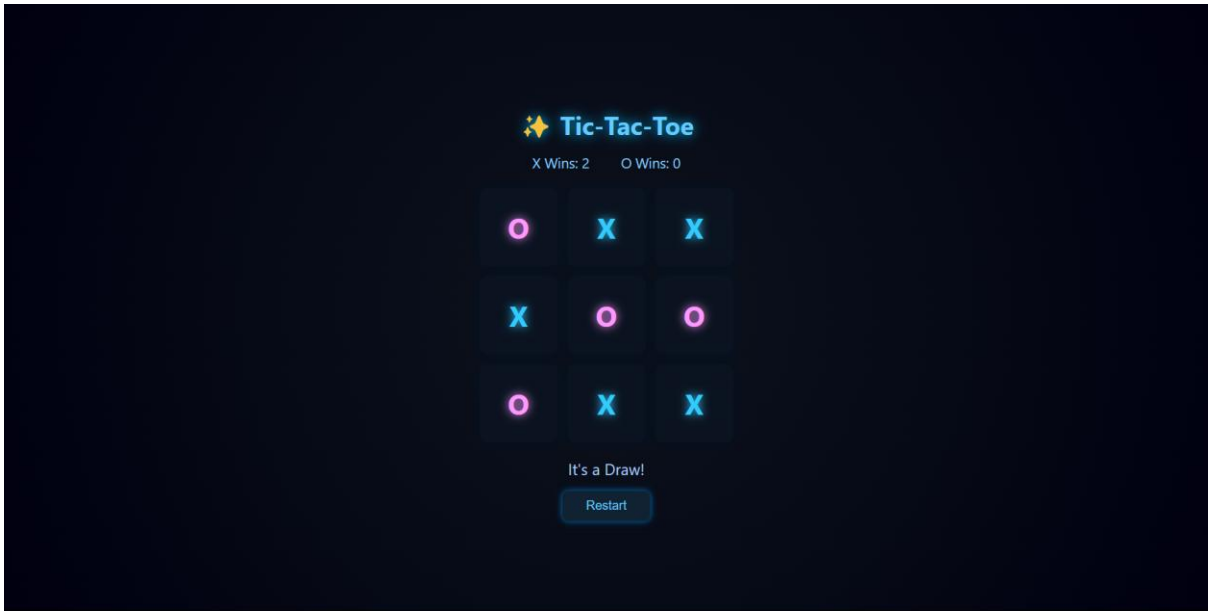


- **8.3 Win State**
    - When a player wins, the status message displays "Player [X/O] Wins!".
    - The winning row, column, or diagonal may be visually highlighted (this is not explicitly in the code but could be added).
    - The scoreboard is updated to reflect the winner's increased score.

- The board is disabled.



- **8.4 Draw State**
  - If all cells are filled and no player has won, the status message displays "It's a Draw!".



9. Limitations and Feature Enhancements

- **9.1 Limitations**
  - **Local Gameplay Only:** The game is designed for two players on the same device. It does not support online multiplayer.
  - **Basic AI:** There is no AI opponent for single-player mode.

    - **No Player Name Input:** Players are simply referred to as "X" and "O" without the option to enter their names.
    - **No Visual Highlighting of Winning Cells:** The winning combination (row, column, or diagonal) is not visually highlighted.
- **9.2 Feature Enhancements**
    - **Single-Player Mode:** Implement an AI opponent with varying difficulty levels.
    - **Online Multiplayer:** Allow players to play against each other over a network.
    - **Player Name Input:** Enable players to enter their names.
    - **Visual Highlighting of Winning Cells:** Highlight the winning combination of cells.
    - **Game History:** Keep a record of previous games.
    - **Animations:** Add more elaborate animations for wins, draws, and turns.
    - **Sound Effects:** Incorporate sound effects for actions like cell clicks, wins, and draws.
    - **Difficulty Levels:** For single-player mode, provide different AI difficulty levels.
    - **Theming Options:** Allow users to choose from different color themes or customize the appearance.

10. Conclusion

The "Tic-Tac-Toe: Bluey Black Enhanced" project successfully delivers a functional and visually appealing implementation of the classic Tic-Tac-Toe game. It meets the core objectives of providing a user-friendly, interactive, and engaging gaming experience. While it has some limitations, it lays a solid foundation for future enhancements, such as single-player mode, online multiplayer, and improved aesthetics. The project demonstrates effective use of HTML, CSS, and JavaScript to create a dynamic web application.