# ****MarbleMind****

**DOCUMENTATION**

**Project:** Two-Player Counterclockwise Marble Game  
**Author(s):** Harsh Agnihotri

**Table of Contents**

1. Project Overview
2. Code Structure
3. Approach
4. Libraries and Frameworks
5. Installation and Setup
6. Usage Instructions
7. Testing
8. Troubleshooting and FAQs
9. Future Enhancements
10. Acknowledgements and Credits
11. Appendices

**Project Overview**

**Introduction**

The "Two-Player Counterclockwise Marble Game" is a digital recreation of a strategic board game designed for two players. The game is played on a 4x4 grid where players take turns to place marbles with the goal of aligning four consecutive marbles in a row to win. After every turn, all marbles move counterclockwise, adding a layer of complexity to the game. The game provides a dynamic and interactive experience through its simple interface and clear game logic.

**Features**

* 4x4 grid gameplay for two players.
* Counterclockwise marble movement after each turn.
* Dynamic UI with interactive elements.
* Win condition check after every move.
* Timer for each round.

**Target Audience**

This game is targeted at casual gamers who enjoy turn-based strategy games and want a simple yet challenging experience. It is also aimed at developers learning game logic and Flutter app development.

**Goals**

The game solves the problem of creating a digital version of a classic board game with an additional counterclockwise movement mechanic. It aims to deliver a fun, engaging, and educational experience while showcasing Flutter’s capabilities.

**Code Structure**

**Directory Layout**

/src - Source code files

/assets - Static files like images, icons

/lib - Libraries and reusable components

/test - Test cases and scripts

**Key Files and Folders**

* **main.dart:** Entry point of the application where the game UI is initialized and the game loop begins.
* **game\_logic.dart:** Contains the core game logic for marbles’ placement, movement, and win condition checking.
* **board\_widget.dart:** Manages the visual representation of the grid, handling user inputs and UI updates.

**Approach**

**Problem Definition**

The problem this project aims to solve is the lack of a simple, engaging, two-player strategy game with unique mechanics, such as counterclockwise marble movement, which adds an additional layer of challenge to the game.

**Design Choices**

* **Architecture:** The game uses the MVVM (Model-View-ViewModel) design pattern, separating the game’s logic (Model), the user interface (View), and the state management (ViewModel).
* **State Management:** The provider package is used for state management to handle the game state changes, such as player turns, marble positions, and win conditions.

**Algorithm Overview**

* **Counterclockwise Movement:** After each player's turn, all marbles move one position in a counterclockwise direction. This ensures that the board’s state is constantly updated, introducing a dynamic aspect to the game.
* **Win Condition Check:** After every move, the game checks for four consecutive marbles in any row, column, or diagonal to determine if a player has won.

**Libraries and Frameworks**

**List of Dependencies**

* **Flutter (version: 3.10.5)**: Framework used for building cross-platform applications.
* **VS Code**: Code editor used for development due to its extensive Flutter support and useful plugins.
* **Android Development**:
  + **Android Studio (or Android SDK)**: Essential for building, running, and testing the app on Android devices or emulators.
  + **ADB (Android Debug Bridge)**: Used for debugging and running the app on physical or virtual devices.

**Reason for Selection**

* **Flutter**: Chosen for its ability to create beautiful, performant, and responsive UIs for both Android and iOS platforms using a single codebase.
* **VS Code**:
  + Offers a lightweight development environment.
  + Provides excellent Flutter extensions for debugging, linting, and code auto-completion.
* **Android Development**:
  + Android Studio: Provides comprehensive tools for Android-specific development, including device emulators and performance profiling.
  + ADB: Simplifies app testing and debugging on Android devices.

**Installation and Setup**

**Prerequisites**

* **Flutter SDK** installed on your system.
* **IDE** like Visual Studio Code or IntelliJ IDEA with Flutter plugin installed.

**Installation Steps**

1. Clone the repository:

git clone <repository\_url>

1. Navigate to the project directory:

cd project\_directory

1. Install dependencies:

flutter pub get

1. Run the application:

flutter run

**Usage Instructions**

**Running the Application**

* To start the app, execute flutter run in the project directory.

**Key Functionalities**

* **Start Game:** On the home screen, click "Start Game" to begin a new session.
* **Place Marbles:** Players alternate placing marbles by tapping an empty cell on the 4x4 grid.
* **Counterclockwise Movement:** After each turn, marbles automatically move counterclockwise around the grid.
* **Win Condition Check:** The game checks for four consecutive marbles in a row, column, or diagonal after every move and declares the winner.

**Testing**

**Testing Strategy**

* **Unit Tests:** Test the core game logic, including marble placement and counterclockwise movement.
* **Integration Tests:** Ensure that the UI correctly reflects the game state after each move.

**Test Files**

### /test/game\_logic\_test.dart: Contains unit tests for marble movement and win condition checking. Explanation of Tests

1. **Initial Board Setup:**
   * This test checks if the board is empty initially, ensuring that the game starts with no marbles placed.
2. **Place Marble on Board:**
   * This test checks if marbles are placed correctly on the grid when a player clicks on a cell.
3. **Counterclockwise Marble Movement:**
   * This test checks if marbles move in a counterclockwise direction after a turn. It places marbles in the first column and then moves them counterclockwise, ensuring the positions are updated correctly.
4. **Win Condition (Row):**
   * This test checks if the game correctly detects a win condition for a row. It simulates a scenario where a player places four marbles in a row and checks if the game recognizes this as a win.
5. **Win Condition (Column):**
   * This test checks for a vertical win condition, where a player places four marbles in a column and expects the game to detect it as a win.
6. **Win Condition (Diagonal):**
   * This test checks for a diagonal win condition, where a player places four marbles diagonally and the game detects the win.
7. **Alternating Turns:**
   * This test checks if the turns alternate between the two players (X and O).
8. **Prevent Placing Marble on Occupied Cell:**
   * This test ensures that players cannot place a marble in an already occupied cell, enforcing game rules.

**Execution**

Run tests with the following command:

flutter test

**Troubleshooting and FAQs**

**Common Issues**

* **App Crashes on Launch:** Ensure Flutter SDK is correctly installed and up to date.

**FAQs**

* **How do I restart the game?** Simply click the "Restart Game" button after a game ends.

**Future Enhancements**

* **AI for Single-Player Mode:** Implement an AI opponent with different difficulty levels.
* **Game History View:** Enable players to view a history of past moves and outcomes.

**Acknowledgements and Credits**

* **Flutter:** Used as the framework for developing the app.
* **Provider:** State management library.

**Appendices**

**Code Snippets**

**Example: Counterclockwise Movement Logic**

void moveMarblesCounterclockwise() {

// Iterate over the grid and shift each marble one position counterclockwise.

}

**Glossary**

* **State Management:** A way of managing the state of the application to ensure the UI updates reactively when the game state changes.
* **Counterclockwise Movement:** A mechanic where marbles shift one position in a counterclockwise direction after each turn.