Documentation of Tic Tac Toe Game

Christof Renner and Felix Hahl

March 26, 2024

Contents

1 Introduction

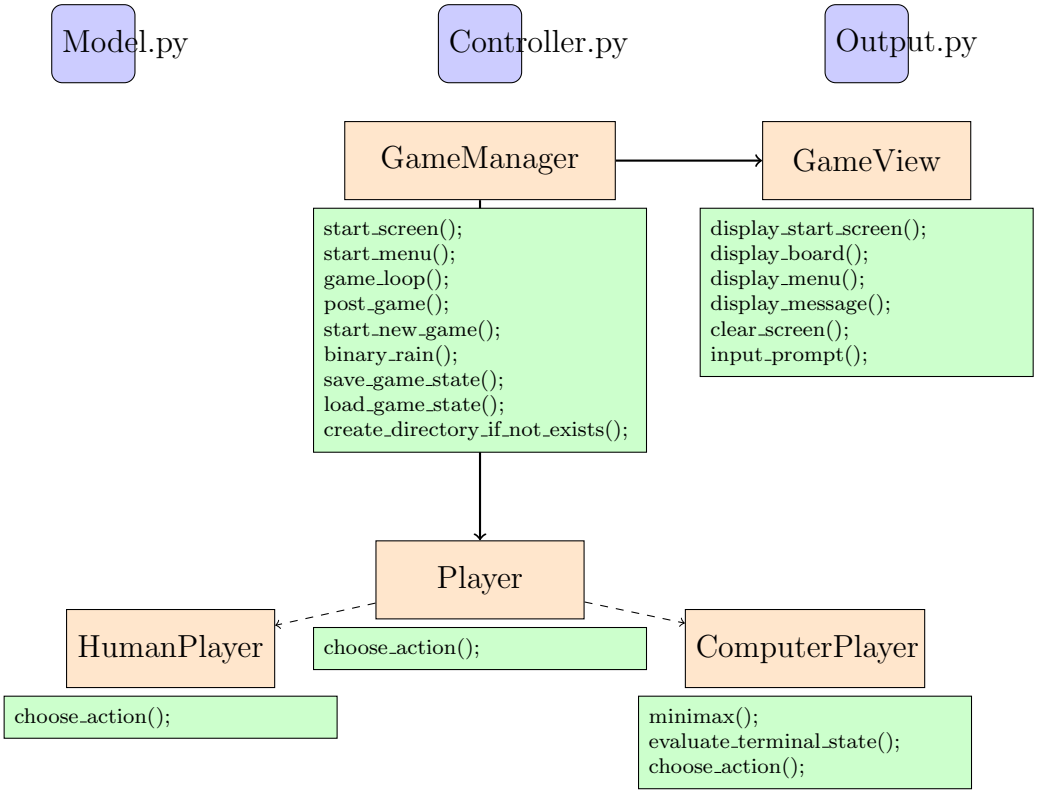
This document aims to provide a comprehensive documentation of the Tic Tac Toe game, covering its architecture, AI implementation, serialization mechanisms, testing strategies, contributions, and future prospects.

2 Architecture

2.1 Overview

The game's architecture is designed to separate concerns, enhance code reusability, and simplify maintenance. It is divided into three main components: the Model, the View, and the Controller, following the MVC design pattern.

2.2 UML Diagrams



3 Serialization and Deserialization

Serialization and deserialization mechanisms are crucial for saving and loading game states. This functionality is implemented through Python's JSON module, allowing game states to be saved to disk and retrieved later.

3.1 Implementation Details

Here we would provide code snippets showing how the game state is serialized into JSON format and deserialized back into game objects.

```
import json

def save_game_state(self):
    game_state = {
        'board': self.board.board,
        'num_players': self.num_players,
        'current_turn': self.board.current_player()
    }
    with open(filename, 'w') as file:
        json.dump(game_state, file)
```

Listing 1: Serialization Example

4 Game AI

The game AI uses the Minimax algorithm to determine the optimal move for the computer player. It evaluates the game board at each step to make the most advantageous move.

4.1 Minimax Algorithm

The Minimax algorithm is a recursive algorithm used for minimizing the possible loss for a worst-case scenario. When dealing with gains, it is maximizing the minimum gain.

5 Tests

Our testing strategy focused on achieving high code coverage and ensuring the reliability of the game's core functionalities. We utilized a combination

of unit tests and integration tests.

5.1 Achieving 90% Line Coverage

To ensure that at least 90% of our business logic was covered by tests, we employed the following strategies:

- Writing unit tests for all models and utility functions.
- Implementing integration tests to cover scenarios involving user interactions.

6 Contributions

This section details the contributions of each team member to the project. For example, John Doe was responsible for implementing the game AI, while Jane Doe focused on the serialization/des