HE2B ESI

DEV4 : Project

Report

22 Pommes

Author:
Yahya Ouamar
54915

July 23, 2023

Contents

1 Introduction

The purpose of this project was to develop an engaging and strategic game, 22 Pommes, as part of my DEV4 course. The game, intended for two players aged 10 and up, introduces strategic elements and tactical decision-making processes. The main goal of the game is to either gather exactly 11 green apples and 11 red apples in the player's baskets or force the opponent to accumulate more than 11 apples of any color. In addition to the standard version of the game, I have also implemented an advanced version named the "Verger Allongé" variant, which adds complexity and strategy by altering the initial layout of the tokens.

2 Game Logic and Mechanics

The core of 22 Pommes lies in its game logic and mechanics. The game manages players, the game board, and player moves.

2.1 Board Initialization

The game board is initialized depending on the game mode. In the Verger Allongé variant, the initial layout of the tokens is altered to add an extra layer of challenge and strategy to the gameplay. A deterministic algorithm ensures the balanced distribution of red and green apple tokens on the board.

2.2 Scoring and Win Conditions

The game maintains the score for each player, tracking the number of red and green apples they have gathered. It checks after each move whether a player has met the win conditions – having exactly 11 apples of each color, or forcing their opponent to have more than 11 apples of either color.

3 Randomization

The board configuration is randomly shuffled using the Mersenne Twister algorithm

```
(std::mt19937)
```

a widely used pseudorandom number generation algorithm due to its good statistical properties and long period. The randomization is seeded with a random device

```
(std::random_device)
```

for an unpredictable seed. This randomization is used to shuffle the board and also to decide whose turn it is to start.

4 Data Structure Choice

A flat vector

```
(std::vector<std::string>)
```

is used to model the 2D game board. This simplifies the operations of accessing and modifying the elements, especially with random access, at the cost of slightly more complex indexing

```
(row * cols + col instead of [row][col])
```

5 Game State

The state of the game is kept in an instance of the Game class, which includes the board configuration, the positions of the 'harvester', and the turn of the player. Additionally, each player's collected apples are stored in their Player objects.

6 Move Validation

The validity of a move is determined by whether the target cell is in the same row or column as the 'harvester' and whether it has not been visited before (marked by "X").

7 Observer Pattern

The Game class implements the Observer design pattern. This pattern is used when an object (subject) needs to inform other objects (observers) about changes in its state.

8 Modelization Choices

The game uses simple integers to represent the position of the 'harvester' and string values to represent the state of each cell on the board. Player information is encapsulated into a Player class, which tracks the number of apples of each color that a player has collected.

In terms of algorithmic complexity, most operations (move validation, making a move, checking for game end) are constant-time (O(1)), ensuring efficient performance even if the size of the board is increased.

9 Future Work

This game provides a solid foundation for future enhancements. Potential areas for development include adding AI opponents for single-player mode, improving user interface for an optimal player experience, and exploring the implementation of other game variants.