

Analysing and Solving Problems

by Mariam Hussain

Submission date: 27-Jul-2023 07:43AM (UTC+0100)

Submission ID: 210732143

File name: 44343_Mariam_Hussain_Analysing_and_Solving_Problems_503743_451381501.docx (82.07K)

Word count: 957

Character count: 5064

Scrabble Game Computer-Based Solution Requirements

Mariam Hussain

Analysing and Solving Problems

ACCA4019

Abida Perveen

25th July, 2023

Scrabble Game Computer-Based Solution Requirements

¹ Rules of Scrabble

¹ Scrabble is a word game played on a 15x15 grid board, where ¹ players take turns forming words using letter tiles. Each tile has a letter and a corresponding point value. The rules are as follows:

- Players draw seven letter tiles from a pool to start the game.
- The first word must be placed on the center square of the board.
- Each subsequent word must connect horizontally or vertically to existing words on the board.
- ¹ Words must be formed from left to right (or top to bottom) and cannot be placed diagonally.
- Players score points based on the letters used and any premium squares (double or triple letter/word scores) covered by the word.
- Unused tiles can be exchanged for new ones, but this ends the player's turn.
- The game continues until all tiles are used, no more words can be formed, or when one player uses all their tiles.

Requirement Features List (Prioritized)

Must-Have Requirements

1. User Interface (UI)
 - Display the game board with a 15x15 grid, allowing players to place their words.
 - Show the letter tiles in the player's rack and let them drag and drop tiles onto the board.
 - Provide buttons for actions like exchanging tiles, passing turns, and reshuffling the rack.
2. Word Validation

- Implement a dictionary to verify the validity of words played by players.
- Highlight invalid words and prevent players from placing them on the board.
- Apply rules for connecting words correctly to the existing tiles on the board.

3. Scoring System

- Assign point values to each letter tile based on the official Scrabble letter distribution.
- Incorporate premium squares (double/triple letter/word scores) to calculate scores for each word.
- Update and display the scores of players after each turn.

Should-Have Requirements

1. Single-Player Mode

- Create an AI opponent with varying difficulty levels for solo play.
- Enable users to switch between single-player and multiplayer modes.

2. Multiplayer Functionality

- Implement online multiplayer capabilities to allow users to play with friends or random opponents.
- Enable real-time synchronization of the game state between players.

Could-Have Requirements

1. Hint/Assist Feature

- Give players the option to receive hints or suggestions for possible high-scoring words.
- Limit the usage of hints to maintain the game's competitive aspect.

2. Customization Options:

- Allow players to choose different board layouts, themes, and tile designs for a personalized experience.

Selected Must-Have Requirements and Their Priorities

a. User Interface (UI) - Priority: High

The User Interface is paramount since it directly impacts the game's usability and user experience. A well-designed and intuitive UI ensures players can easily interact with the game, enhancing their enjoyment.

b. Word Validation - Priority: High

The accuracy of word validation is crucial to maintain the integrity of the game.

Implementing a reliable dictionary and enforcing proper word connection rules will prevent cheating and keep the gameplay fair for all participants.

c. Scoring System - Priority: Medium

While scoring is an essential aspect of Scrabble, it is a medium priority compared to UI and word validation. A scoring system is vital to track player progress and determine the winner, but it can still function effectively even if it lacks certain advanced features.

1. Variables for the Scrabble Game Implementation:

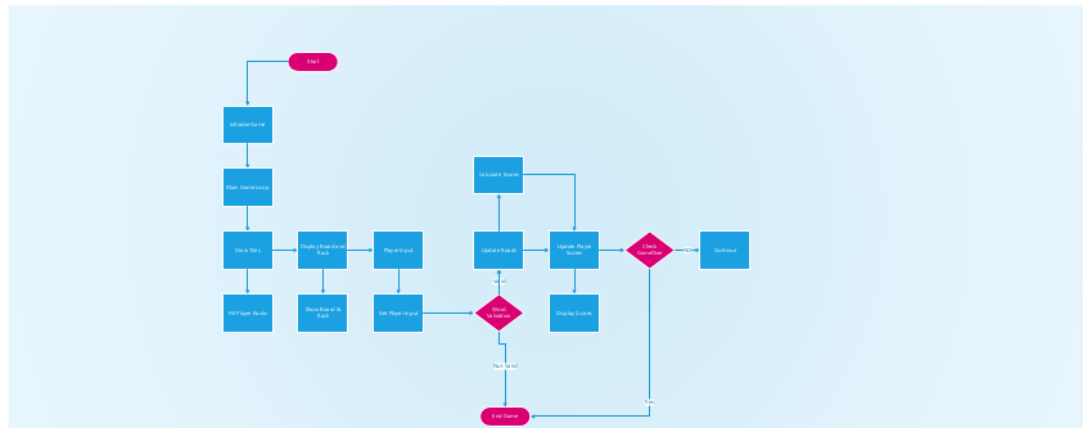
- a. **GameBoard (2D Array):** Represents the 15x15 game board grid, storing the letters and their respective scores at each cell.
- b. **LetterPool (Array):** Holds all the letter tiles available for players to draw from during the game.
- c. **PlayerRacks (Array of Arrays):** Stores the letter tiles for each player's rack.
- d. **CurrentPlayer (Integer):** Indicates the index of the current player.
- e. **WordDictionary (Set or Array):** Contains a list of valid words to validate player inputs.
- f. **PlayerScores (Array):** Keeps track of each player's score during the game.
- g. **GameOver (Boolean):** Signals at the end of the game when set to true.
- h. **PremiumSquares (2D Array):** Marks the game board's premium squares (double/triple letter/word scores).
- i. **BonusWordScores (Array):** Stores the scores of each bonus word formed in a single turn.
- j. **ExchangePool (Array):** Holds the letter tiles returned by players during tile exchange.

Flowchart for Scrabble Game Algorithm

Major Components

- a. **Start:** The game begins with initializing the game board, letter pool, player racks, player scores, and other variables.
- b. **Main Game Loop:** The loop iterates through players' turns until the game ends.
- c. **Draw Tiles:** Each player draws tiles from the letter pool to fill their rack.
- d. **Display Board and Rack:** The current state of the game board and the player's rack are displayed on the UI.
- e. **Player Input:** The player inputs their word choice, exchange option, or pass-turn decision.
- f. **Word Validation:** The input word is checked against the word dictionary for validity. If valid, the Word's score is calculated based on premium squares.
- g. **Update Board:** The valid Word is placed on the board and updated with new letters.

- h. **Calculate Score:** The player's score is updated based on the Word's score and any bonus word scores.
- i. **Check Game Over:** The game checks if it has ended by evaluating tile availability and valid word formation possibilities.
- j. **End Game:** The game ends, and the final scores are displayed.



Pseudo Code for Two Sub Processes

a. Draw Tiles (Sub Process)

```

Procedure DrawTiles(PlayerIndex)
  While PlayerRacks[PlayerIndex] is not full:
    Randomly select a letter tile from LetterPool.
    Add the selected tile to PlayerRacks[PlayerIndex].
    Remove the selected tile from LetterPool.
  End While
End Procedure
  
```

b. Word Validation (Sub Process)

```

Function IsValidWord(Word):
  If Word is present in WordDictionary:
    Return true
  Else:
    Return false
  End If
End Function
  
```


Analysing and Solving Problems

ORIGINALITY REPORT

5%

SIMILARITY INDEX

2%

INTERNET SOURCES

0%

PUBLICATIONS

4%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to University of Wales, Lampeter

Student Paper

3%

2

indofreak.net

Internet Source

2%

Exclude quotes Off

Exclude bibliography Off

Exclude matches Off