



Word Scramble Game

~Somika Dobriyal, 590027461, B-48

Abstract

This document provides comprehensive documentation for the Word Scramble Game, a C-based console application designed to provide an engaging word puzzle experience. The system allows players to unscramble words within limited attempts, with scoring based on correct guesses. The implementation follows modular programming principles with clear separation of concerns, demonstrating good software engineering practices.

Contents

- 1 Problem Definition
 - 1.1 Overview
 - 1.2 Objectives
- 2 System Design
 - 2.1 System Architecture
 - 2.2 Flowchart
 - 2.3 Data Structures
- 3 Implementation Details
 - 3.1 Key Features
 - 3.2 Code Structure
- 4 Testing & Results
 - 4.1 Test Cases
- 5 Conclusion & Future Work
 - 5.1 Conclusion
 - 5.2 Future Work
- 6 References

1 Problem Definition

1.1 Overview

The Word Scramble Game is designed to provide an entertaining and educational word puzzle experience. The game challenges players' vocabulary and problem-solving skills by presenting scrambled words that must be unscrambled within a limited number of attempts. Traditional word games often lack structured scoring systems and progressive difficulty, which this system addresses through its round-based approach with scoring mechanics.

1.2 Objectives

- To develop a console-based word puzzle game with scoring mechanics
- To implement word scrambling and unscrambling functionality
- To provide multiple attempts per word with feedback
- To implement persistent word database
- To create a user-friendly interface with clear instructions

- To ensure robust input validation and error handling

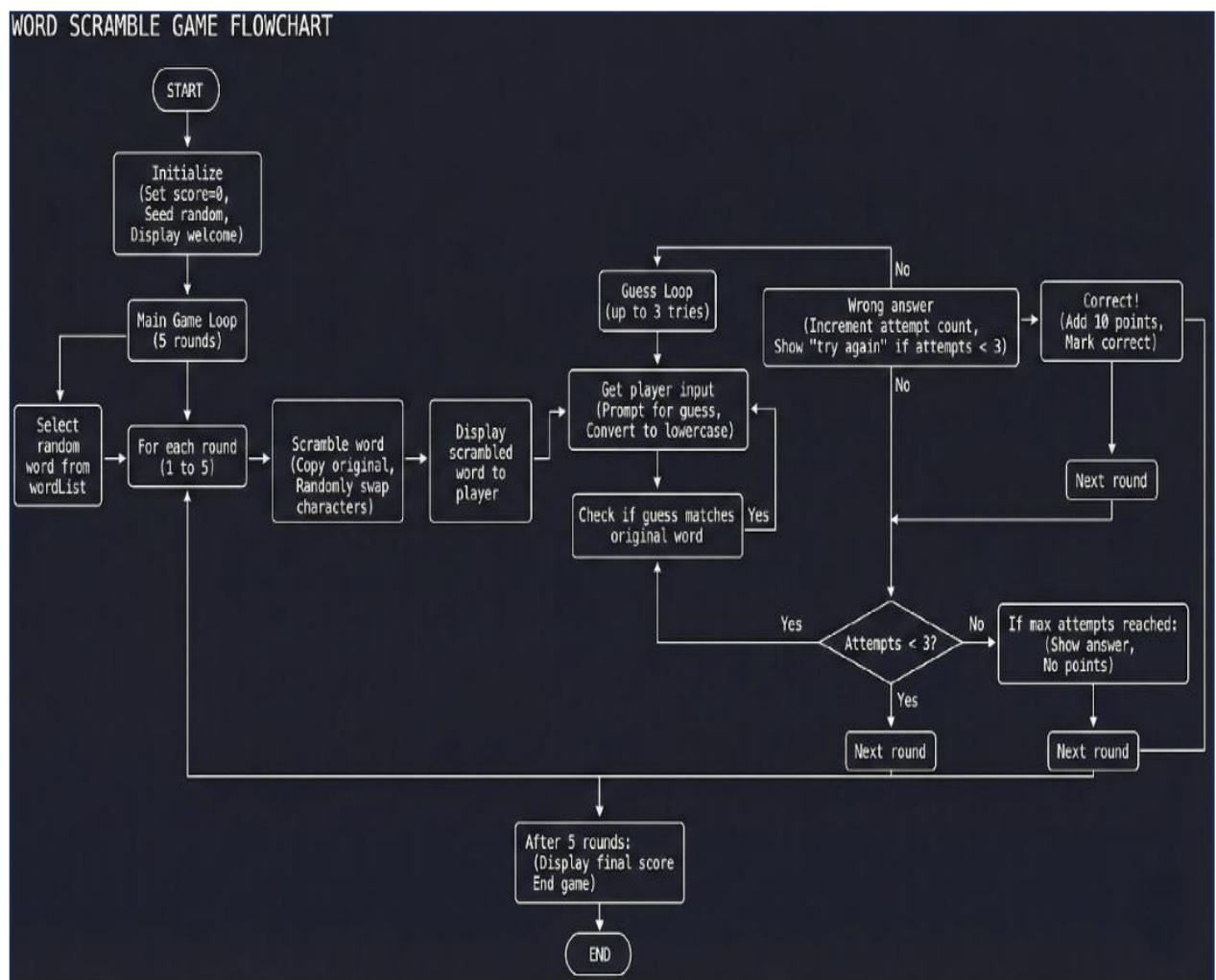
2 System Design

2.1 System Architecture

The system follows a modular architecture with clear separation of concerns:

- **Presentation Layer:** Handles user interaction through console interface
- **Game Logic Layer:** Implements word selection, scrambling, and scoring
- **Data Layer:** Manages word database storage
- **Utility Layer:** Provides helper functions for string manipulation

2.2 Flowchart



```
char wordList[MAX_WORDS][MAX_LENGTH] = {  
    "computer",  
    "programming",  
    "algorithm",  
    "function",  
    "variable",  
    "compiler",  
    "database",  
    "keyboard",  
    "internet",  
    "software"  
};
```

```
int score = 0;  
int totalRounds = 5;  
char scrambled[MAX_LENGTH];  
char guess[MAX_LENGTH];
```

3 Implementation Details

3.1 Key Features

- **Random Word Selection:** Selects words randomly from predefined database
- **Word Scrambling:** Randomly rearranges letters while maintaining original characters
- **Case-Insensitive Comparison:** Normalizes all input to lowercase for fair comparison
- **Attempt Tracking:** Limits player to 3 attempts per word
- **Scoring System:** Awards 10 points for each correct guess
- **Round Progression:** Conducts 5 rounds with cumulative scoring
- **Visual Feedback:** Provides ✓/✗ symbols for immediate feedback
- **Progress Display:** Shows current score and round information

3.2 Code Structure

3.2.1 Header Files

- **config.h:** Contains game constants and configuration
- **words.h:** Defines the word database
- **functions.h:** Contains all game logic functions

3.2.2 Core Functions

- ❖ `int selectRandomWord();`
- ❖ `void scrambleWord(char *original, char *scrambled);`
- ❖ `void toLowerCase(char *str);`
- ❖ `void displayWelcome();`
- ❖ `void getPlayerInput(char *input);`
- ❖ `int checkGuess(char *original, char *guess);`
- ❖ `void displayScore(int score, int totalWords);`

3.2.3 Main Game Loop

1. Initializes game state and random seed
2. Displays welcome message
3. Executes 5 rounds of gameplay
4. For each round: selects word, scrambles it, gets player input
5. Manages attempt counting and scoring
6. Displays final results

4 Testing & Results

4.1 Test Cases

1. **Correct Guess Recognition**
 - Input: Player enters correct unscrambled word
 - Expected: System recognizes correct answer and awards points
 - Result: Passed - Score increments by 10 points
2. **Attempt Limitation**
 - Input: Three incorrect guesses
 - Expected: System moves to next round after 3 attempts
 - Result: Passed - Game proceeds correctly
3. **Score Calculation**
 - Input: 3 correct guesses out of 5 rounds
 - Expected: Final score = 30
 - Result: Passed - Score calculated accurately

5 Conclusion & Future Work

5.1 Conclusion

The Word Scramble Game successfully meets its objectives, provides an engaging word puzzle experience. The system demonstrates good software engineering practices through modular design, clear separation of concerns, and robust error handling. The game is both entertaining and challenging, with appropriate feedback mechanisms and scoring systems that encourage repeated play.

5.2 Future Work

- **Difficulty Levels:** Implement easy/medium/hard word categories
- **Time-Based Scoring:** Add timer for additional challenge
- **Word Categories:** Organize words by themes (technology, animals, etc.)
- **Player Profiles:** Save high scores and player statistics using files
- **Hint System:** Provide letter hints for difficult words

6 References

1. Online resources for game development patterns and algorithms
2. ASCII art and text-based UI design references