Software Requirements Specification (SRS)

Logic Puzzle Game

Lukas Goering

25. Juni 2025

1 Introduction

1.1 Purpose

This document specifies the functional and non-functional requirements for a 2D logic puzzle game to be implemented using Java and JavaFX. The game involves two overlapping 3x3 grids—each filled with stones of different colors—and a central overlapping field that starts empty. The user manipulates stones under defined rules to achieve a winning configuration.

1.2 Scope

The software is a desktop-based, single-player game application. The user interacts with the system through mouse and/or keyboard input to move stones across a graphical interface. The system visually displays the state of the game board, tracks moves, and determines the win condition. It does not require any network features.

1.3 Definitions

- **Board A**: A 3x3 grid initialized with white stones.
- **Board B**: A 3x3 grid initialized with black stones.
- **Overlap Square**: The central square shared by both boards, initially empty.
- **Stone**: A movable piece, either white or black, that occupies a square on a grid.

2 Overall Description

2.1 Product Perspective

This is a standalone Java desktop application developed with JavaFX. The interface renders custom 2D graphics using the JavaFX Canvas API and standard UI controls for user feedback. The application includes a game loop that handles input, rendering, and logic processing.

2.2 Product Functions

- Initialize two 3x3 overlapping grids.
- Display black and white stones on their respective grids.
- Allow the user to move stones according to game rules.
- Detect and announce when the win condition is met.
- Track and display the number of moves made.

2.3 User Characteristics

Users are expected to be casual desktop users with basic mouse/keyboard experience. No prior knowledge of the game's internal rules is assumed.

2.4 Constraints

- Java 11 or later with JavaFX support is required.
- The application must support Windows and Linux platforms.
- The game must run as a self-contained executable (JAR or packaged distribution).

3 Functional Requirements

3.1 Board Initialization

FR1 Create and render two overlapping 3x3 grids at launch.

FR2 Fill Board A with white stones.

FR3 Fill Board B with black stones.

FR4 Leave the overlapping square empty.

3.2 Stone Movement Rules

FR5 Allow a stone to move into a horizontally or vertically adjacent field if that field is empty.

FR6 Allow a stone to jump over one adjacent stone (vertically or horizontally) into an empty field, provided both the adjacent and the next field are in line and not diagonal.

FR7 Prevent diagonal moves and invalid jumps.

FR8 Update the game board and internal state after each valid move.

FR9 Ignore or optionally notify the player of invalid moves.

3.3 Win Condition

FR10 After each move, check if all white stones occupy the initial positions of the black stones, and all black stones occupy the initial positions of the white stones.

FR11 When this condition is met, display a "You Win"message and show the total number of moves made.

3.4 Move Counter

FR12 Count each valid player move.

FR13 Display the current move count in real-time on the user interface.

3.5 Bonus Feature: Top Scores Podium

FR14 Maintain a persistent record of the top three lowest move counts achieved by players (i.e., best scores).

FR15 Display these top scores on a separate "Podiumör "High Scores"panel within the UI.

FR16 Allow the player to reset or clear the high scores via a button or menu option.

FR17 Persist high scores between sessions using local file storage.

FR18 Update the podium if a new score qualifies as one of the top three.

4 Non-Functional Requirements

- **Performance**: The game should update the interface at a minimum of 30 FPS.
- **Portability**: The game must run on both Windows and Linux environments with JavaFX.
- **Usability**: UI should be intuitive, with clearly visible stones and responsive input.
- **Maintainability**: Code should follow modular design principles using object-oriented patterns.
- **Robustness**: The system must gracefully handle unexpected input and edge cases.

5 User Interface Overview

The main window should include:

- A visual display of the overlapping 3x3 grids with colored stones.
- A label showing the current move count.
- Optional buttons: Reset, Quit, Undo, Redo

6 Future Enhancements (Out of Scope)

- Add sound effects or animations.
- Include difficulty levels (4x4 grids or custom shaped) or randomized starting configurations.
- Save/load game state to/from disk.
- Add a scoring system or timer.