**Average Othello**

**Tony Nguyen**

**Johnathan Kneice**

**Luis Garcia**

**Henry Dyer**

**Table of Contents**

[**1. Introduction**](#_heading=h.gjdgxs) **3**

[1.1. Purpose](#_heading=h.30j0zll) 3

[1.2. Intended Audience](#_heading=h.1fob9te) 3

[1.3. Scope](#_heading=h.3znysh7) 3

[1.4. Definitions](#_heading=h.2et92p0) 3

[1.5. Reference](#_heading=h.tyjcwt) 4

[**2. Overall description**](#_heading=h.3dy6vkm) **5**

[2.1. Game Rules](#_heading=h.1t3h5sf) 5

[2.2. Product function](#_heading=h.4d34og8) 5

[2.3. User characteristics](#_heading=h.2s8eyo1) 6

[2.4. Dependencies on other systems or software](#_heading=h.17dp8vu) 6

[**3. Specific requirements**](#_heading=h.3rdcrjn) **6**

[3.1. User interfaces](#_heading=h.26in1rg) 6

[3.2. Hardware interfaces](#_heading=h.lnxbz9) 7

[3.3. Software interfaces](#_heading=h.35nkun2) 7

[3.4. Communication interfaces](#_heading=h.1ksv4uv) 8

[**4. Functional Requirements**](#_heading=h.44sinio) **8**

[4.1. Start and End of Game](#_heading=h.2jxsxqh) 8

[4.2. Input and Output Processes](#_heading=h.z337ya) 9

[4.3 Playing Against an AI Opponent](#_heading=h.3j2qqm3) 9

[4.4 Playing Against a Local Opponent](#_heading=h.1y810tw) 10

[4.5 Playing Against a Network Opponent](#_heading=h.4i7ojhp) 10

[4.6 Appearance](#_heading=h.2xcytpi) 10

[4.7 Menus](#_heading=h.1ci93xb) 11

[**5. Non-functional Requirements**](#_heading=h.3whwml4) **11**

[5.1. Performance requirements](#_heading=h.2bn6wsx) 11

[5.2. Safety Requirements](#_heading=h.qsh70q) 12

[5.3. Security Requirements](#_heading=h.3as4poj) 12

[5.4. Software quality requirements](#_heading=h.1pxezwc) 12

# 1. Introduction

This introduction should cover the overview of this SRS document. Furthermore, this document has a table of definitions and abbreviations.

## 1.1. Purpose

The purpose of this document is to provide a detailed description of the requirements for the “Average Othello” application.

## 1.2. Intended Audience

This document is intended for people with a basic knowledge of Java and a basic knowledge of software engineering.

## 1.3. Scope

“Average Othello” is a desktop application that allows the user to play a virtual version of the board game Othello (and Reversi). The application should emulate the experience of playing Othello using the modern ruleset created by Goro Hasegawa in 1971. It should also provide the option to play with the older Reversi ruleset if the user so chooses.

Othello is a game played by two players on an 8 × 8 grid-based board. Each player is assigned a color, typically either black or white. Players take turns placing one piece of their color on the board in a spot on the grid adjacent to another piece of either color that has already been placed. If any piece of an opponent’s color is placed between two pieces of a player’s color–that is, a straight line can be drawn between two pieces of a player’s color, and there are pieces of an opponent’s color that fall on that line–then all pieces of the opponent’s color should be replaced with pieces of the player’s color. The objective of the game is to have more pieces of the board in your color than your opponent once all the spots on the board are filled with pieces.

The application should display an 8 × 8 grid-based board similar to that used in real-world games of Othello, where users should be able to place their pieces to play the game. The game should be playable using the mouse and left-click button to place pieces on the virtual game board. Upon the start of a game, the user should be given the option to play against an AI opponent, a real opponent using a singular instance of the application, and a real opponent with a second instance of the application running, where the two applications communicate via a network connection.

## 1.4. Definitions

| **Term** | **Definition** |
| --- | --- |
| Reversi | Another name for Othello that uses an 8 × 8 board and 32 pieces for each player.  Reversi is the original game on which Othello is based. Othello is a specific rule-set variant of Reversi, though Reversi and Othello are often used as synonyms of each other. |
| User | Someone who is operating the game piece and the one who is interacting with the software  Someone who is interacting with the software. A user is sometimes also a player, but not always. For example, a user who is clicking a button on the application’s main menu is not a player because a game for the user to play has not yet been started. |
| Player | Someone who is interacting with the software in order to play a game of Othello. |
| AI (Artificial Intelligence) | Computer technology that tries to mimic human intelligence. In this document, it refers to the way the software simulates a player’s decision-making by placing pieces on the board in an algorithmically-decided manner in order to allow human players to play the game without another human present. |
| Network | A way to link multiple computers to be able to interact at the same time. |
| Local multiplayer | Two or more players/users using the software on one device simultaneously. |
| Multiplayer | Two or more players/users using the software. |
| Single player | One player/user using the software. |
| OS (Operating System) | System software providing services for computer programs. |

## 1.5. Reference

**[1] World Othello Federation. “OFFICIAL RULES FOR THE GAME OTHELLO” *World Othello Federation.* 1 Sep 2022.**

**[2] Susan Weber. “Behind the Othello Board Game: History and Gameplay” *LOVE to KNOW*. 9 Sep 2022**

**[3] Home Rec World Editorial Team. “Othello vs. Reversi: Are They the Same?” *Home Rec World.*  8 Sep 2022.**

# 2. Overall description

This section will give an overview of the general software. This overview will also go into detail on the functions and interactions of the inner workings of the program Othello.

## 2.1. Game Rules

A game following the Othello ruleset begins with four pieces placed in the middle of the board. The pieces are placed in a two-by-two square in the center of the board, with the black pieces in the top-right and bottom-left corners, and the white pieces in the top-left and bottom-right corners. The player who goes first plays using the black pieces, whereas the player who goes second uses the white pieces.

During a player's turn, that player must place a piece onto one of the spots on the board that does not already have a piece. A player may only place a piece on a spot if that spot meets the following conditions: one, the spot is adjacent to a spot that already has a piece placed onto it; two, that a horizontal, vertical, or diagonal line can be drawn between the piece being played and another existing piece of that color; and three, that there is at least one or more piece of the opposing color between those two pieces on that line. When a piece is played in this manner, all pieces of the opposing color that are between the piece played and any other pieces of the player's color are flipped over and turned into pieces of the player's color. That player's turn then ends, and the opposing player takes their turn. If a player is unable to place a piece in this manner on their turn, then their turn immediately ends, and the opposing player takes their turn.

Once all of the spots on the board have been filled, the game ends, and the winner is determined based on the number of pieces of each color on the board. The player with the most pieces of their color on the board wins.

A game following the Reversi ruleset plays in the same way, with two key differences. Firstly, in the event that a player is unable to play a piece during their turn, the game ends immediately, and the player with the most pieces on the board wins. Secondly, the pieces placed on the board at the beginning of the game are different than in the Othello ruleset. In Reversi, the starting player begins by placing their first piece on any space within the center four squares of the board, regardless of the normal rules for playing a piece. The opposing player then places their piece in one of the three spaces in the center four squares that is still available. Then, play continues as normal, where each player is subjected to the restrictions of piece placement described above.

## 2.2. Product function

With this program running on computers, the users will be able to operate their game pieces by mouse. Please note that the user should be able to place their pieces in a free space adjacent to any other piece. After placing the user’s piece, the user should no longer be able to affect the board until the opponent makes their move. Please note that the player using black pieces will always go first. Another player can join an online game through IP/TCP.

## 2.3. User characteristics

The users/players will consist of people with the bare minimum of computer skills such as being able to use the keyboard, mouse, and having general reading skills. The second kind of users will be people who understand the rules of Othello, which has an 8+ age rating.

## 2.4. Dependencies on other systems or software

We will be using Java JDK 16.0.2 as the main framework for this software.

# 3. Specific requirements

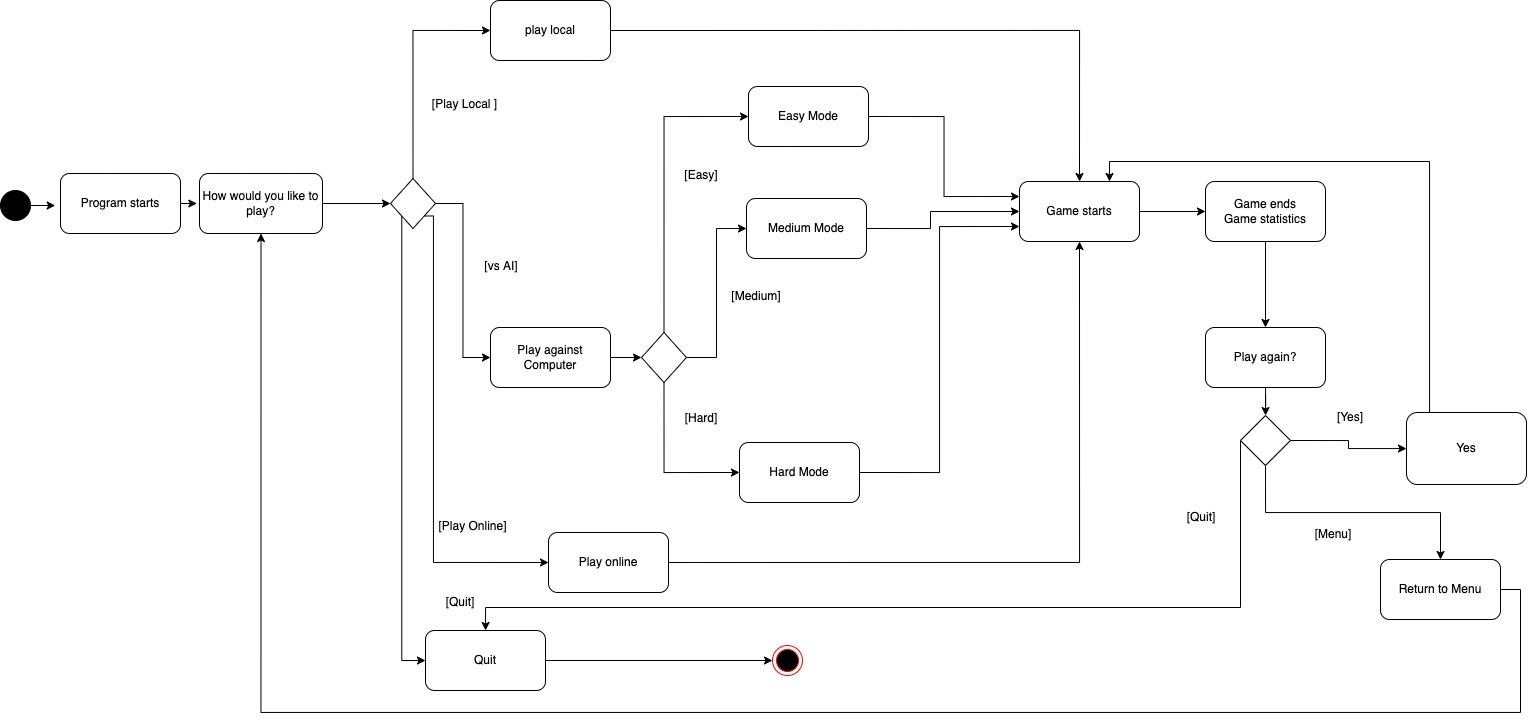
This section contains the functionality and requirements of our program. This section also gives an idea of what the interface will look like.

## 3.1. User interfaces

When the user first starts the program, it should show the user an 8 × 8 light green background and a dark green line grid board with two white pieces in the spots 4D and 5E.\* Two black pieces will also be placed in the spots 5D and 4E. Legal spaces that are allowed for the player will be marked with transparent gray circles. Next, a modal pop-up describes "How do you want to play?" with the four options Local Multiplayer, Online Multiplayer, play against AI, and Quit. Selecting one of these will start the game unless you select quit which will close the program. When the user selects Online Multiplayer, the user's IP address will pop up, and a bar to enter another IP address of the computer the user wants to connect to. The user can place pieces by clicking with your mouse on a location on the board. When no more moves can be made, a pop-up will announce the winner or a tie. A pop-up window will ask if you want to play again, return to the menu or quit.

\*We will be using the following grid to refer to spots in the game board

| **1** |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |
| **4** |  |  |  | **W** | **B** |  |  |  |
| **5** |  |  |  | **B** | **W** |  |  |  |
| **6** |  |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  |  |  |  |
| **8** |  |  |  |  |  |  |  |  |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** |



## 3.2. Hardware interfaces

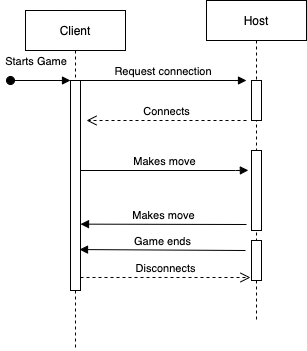
Our program requires a monitor, keyboard, mouse, and speaker. Touch screens will be our stretch goals.

## 3.3. Software interfaces

Our program will be supported on Java JDK 16.0.2.

## 3.4. Communication interfaces

We will have an online mode for two people to play if they both have the program running on their computers. One computer will act as the server allowing the other computer to connect. We will be using TCP/IP in order to achieve online play.



# 4. Functional Requirements

This section contains the functional requirements of the software.

## 4.1. Start and End of Game

4.1.1. The program shall ask the user whether they want to play against an AI, against a human on the same computer, or against a human on another computer.

4.1.2. The program shall ask the user whether they would like to play with the Othello or Reversi ruleset.

4.1.3. The program shall end the game when all places on the board have been filled with pieces if the user has chosen the Othello ruleset.

4.1.4. The program shall end the game when one player is unable to make a move, or when all places on the board have been filled with pieces if the user has chosen the Reversi ruleset.

4.1.5. The program shall display the number of pieces of each color on the board, and throughout the game, the counter will increase or decrease based on the players either gaining or losing pieces. It would then stop counting the pieces when the game reaches its end, in which the player with the most pieces would be considered the winner.

4.1.6. The program shall ask the user whether or not they would like to play another game once the winner of the last game has been displayed.

4.1.7. The program shall start a new game if the user selects to play another game after the end of the game.

## 4.2. Input and Output Processes

4.2.1. The program shall display a placed piece on the board immediately after the player has chosen to place a piece in a permitted space.

4.2.2. The program shall not allow players to place pieces in spaces that are not allowed by the ruleset of the game.

4.2.3. The program shall pass a turn to the other player if the current player has no valid moves if the Othello ruleset has been chosen.

4.2.4. The program shall provide a short visual indication if a player's turn has been passed.

4.2.5. The program shall check which pieces should change color after a piece is placed, and immediately change those colors after a piece is placed.

4.2.6. The opponent should provide a visual indication of which spaces are available for a piece to be placed on during a player's turn.

4.2.7.The player shall be unable to interact with the board while it is the other player’s turn.

4.2.8. The program shall play a short sound effect when the player plays a piece on the board.

## 4.3 Playing Against an AI Opponent

4.3.1. The program shall allow the user to choose between an Easy, Normal, and Hard difficulty setting for their AI opponent.

4.3.2. The program shall decide a location and place a piece on the board within one second of a player placing their piece on the board.

4.3.3. The program shall only place pieces in spaces permitted by the ruleset of the game, subject to the same limitations as human players.

4.3.4. The program shall randomize who goes first. Whoever goes first has the black piece. Whoever goes second will have the white pieces.

## 4.4 Playing Against a Local Opponent

4.4.1. The program shall alternate between black and white pieces being placed on the board upon player input.

## 4.5 Playing Against a Network Opponent

4.5.1. The program shall form a network connection with another instance of the program that is running.

4.5.2. The program shall allow one player in one instance to place black pieces on the board, and shall allow the other player in the other instance to place white pieces on the board.

4.5.3. The player who goes first is randomized, and the colors are determined by who goes first. Whoever goes first gets the black pieces, and the other player gets the white pieces.

4.5.4. The program shall only allow a player to place a piece after the other player has already placed a piece. However, there will be a twenty-five-second time limit which, when elapsed, will skip the player's turn.

4.5.5. The program shall display the local and public IP addresses of the user who initiated the game.

## 4.6 Appearance

4.6.1. The program shall display a green 8 × 8 grid board while in a game.

4.6.2. The program shall provide a visual indication of which moves are permitted to the player.

4.6.3. The program shall display each piece that is currently on the board in its corresponding position on the grid.

4.6.4. The program shall not change the color of pieces on the board unless a permitted piece placement would cause those pieces to change color.

4.6.5. The program shall display a counter of how many pieces of each color are currently on the board.

4.6.6. The program shall update the counter of how many pieces of each color are on the board immediately after a new piece is placed and the color of old pieces is changed as a result.

## 4.7 Menus

4.7.1. Upon the start of the program, the program shall display a "Main Menu" with four options for the user to choose from: Start Game, Join Game, Options and a quit.

4.7.2. The program shall initiate the sequence for starting a game once the user selects "Start Game."

4.7.3. When the “Join Game” option is selected, the program shall prompt the user for an IP address, provided by the game host.

4.7.4. The program shall attempt to connect to a host using a given IP address once an IP address is provided after “Join Game” is selected.

4.7.5. When the “Options” option is selected, the user shall be presented with a menu that allows them to change the piece color of the player who goes first (black by default) and the player who goes second (white by default). That menu shall also allow the user to switch between Othello and Reversi rulesets.

# 5. Non-functional Requirements

This section contains the traits of the software.

## 5.1. Performance requirements

5.1.1 The program shall be run on local Java code.

5.1.2 The program shall depend on the strength of the internet connection for loading times.

5.1.3 The AI will not take more than a second to make a move.

## 5.2. Safety Requirements

5.2.1 This program is unlikely to do any damage since it will not be interacting with many OS features.

## 5.3. Security Requirements

5.3.1 This program will not be encrypting any moves, which would allow a malicious attacker to sabotage the game.

## 5.4. Software quality requirements

5.4.1 The program shall start in the correct way and end in the correct way.

5.4.2 The program shall have an aesthetically pleasing interface for the user.

5.4.3 The program shall satisfy the user’s expectations for a game of Othello.

5.4.4 The program shall respond to the user’s actions immediately after they’re done.

5.4.5 The program shall not crash more than once per 12 hours of play.