Othello/Reversi Web Game

User Manual and Technical Documentation

Oliver Chaisty

Contents

1 Introduction

This document provides complete instructions for installing, running, and using the Othello/Reversi web game.

1.1 What is Othello/Reversi?

Othello is a board game for two players played on an 8x8 board. Players take turns placing pieces on the board with their assigned color (Dark and Light) facing up. The goal is to have the majority of your color pieces on the board at the end of the game.

1.2 Game Rules

Dark always moves first. A move consists of placing one piece on an empty square. To make a legal move, you must flip at least one opponent piece. Pieces are flipped in straight lines (horizontal, vertical, or diagonal) when they are surrounded by your pieces. If you have no legal moves, you must pass your turn. The game ends when neither player can move (board is full or no legal moves remain). The player with more pieces of their color wins.

2 Installation

2.1 Prerequisites

Before running the game, ensure you have the following installed:

1. **Python**
   - Download from: `https://www.python.org/downloads/`
   - During installation, check "Add Python to PATH"
2. **Browser**
   - Google Chrome, Firefox, Safari, or Edge (modern version)

2.2 Downloading the Game Files

You need these three files in the same directory:

Listing 1: Required Files

```
othello_game/
          ai_flask_game_engine.py  # Main game server
          components.py             # Game logic module
          templates/
              index.html           # Website
```

Create a folder on your computer and place all files inside it. (or git clone)

2.3 Installing Python Dependencies

Open a terminal/command prompt and install Flask:

```
pip install flask
```

Or on Arch Linux (the best operating system):

```
sudo pacman -S python-flask
```

3 Running the Game

3.1 Starting the Game Server

1. Open a terminal/command prompt
2. Navigate to the folder containing the game files:

```
    cd ~/othello_game
```

3. Start the Flask server:

```
    python ai_flask_game_engine.py
```

4. You should see:

```
     * Serving Flask app 'ai_flask_game_engine'
     * Debug mode: on
     * Running on http://127.0.0.1:5000
```

3.2 Accessing the Game

1. Open your web browser
2. Go to the address shown in the terminal (usually: `http://127.0.0.1:5000`)
3. The game board should appear

Keep the terminal window open while playing. Closing it will stop the game server.

4 Playing the Game

4.1 Game Interface

The interface consists of:

- **Game Board (8x8 grid)**: Click any empty square to place your piece
- **Pieces**:
  - **Dark**: Your pieces (black circles)
  - **Light**: AI's pieces (white circles)

Figure 1: Game interface showing board, pieces, and message log

- **Message Log**: Shows game status, whose turn it is, and move history
- **Turn Indicator**: Shows whose turn it is (this is a holdover from stage 2, where we had two human players. It isn't necessary in this version.)

4.2 Making a Move

1. Look for squares with a red highlight when you hover over them - these are legal moves
2. Click on any highlighted square to place your piece
3. The game will:
   - Place your piece
   - Flip any captured opponent pieces
   - Make the AI's move automatically
   - Update the board

4.3 Understanding Game Feedback

The message log provides information about:

- **Valid moves**: "Move accepted at (x, y)"
- **Invalid moves**: "Invalid move at (x, y): This is not a legal move..."
- **Turn information**: "It's Dark's turn"
- **Game status**: "Game Over: Dark wins"

4.4 Game End Conditions

The game ends when all 64 squares are filled with pieces or neither player has any legal moves. The winner is determined by counting pieces:

- More Dark pieces: "Dark wins"
- More Light pieces: "Light wins"
- Equal pieces: "Draw"

4.5 Restarting the Game

To start a new game: refresh your browser or click the browser's back button and return to the game

5 API Documentation

This section describes how the API works

5.1 API Endpoints

5.1.1 GET / - Initialize Game

Purpose: Start a new game or reset existing game

Request:

```
GET http://localhost:5000/
```

Response: HTML page with initial game board

5.1.2 GET /move - Process a Turn

Purpose: Make a move and get AI response

Request:

```
GET http://localhost:5000/move?x=3&y=4
```

Parameters:

- `x`: X-coordinate (0-7) of move
- `y`: Y-coordinate (0-7) of move

Response (JSON): One of three formats:

1. **Success (move processed)**:

```
    {
        "status": "success",
        "board": [[...]],  // 8x8 array
        "player": "Dark "  // Next player
    }
```

2. **Fail (invalid move)**:

```
    {
        "status": "fail",
        "message": "Error description"
    }
```

3. **Game Over**:

```
    {
        "status": "game_over",
        "message": "Game Over: Winner",
        "board": [[...]]  // Final board
    }
```

5.2 Board Representation

```
[
    ["None ", "None ", "None ", ...],
    ["None ", "None ", "Dark ", ...],
    ["Light", "Dark ", "None ", ...],
    ...
]
```

Values:

- `"Light"`: Light player's piece
- `"Dark "`: Dark player's piece (note: includes space)
- `"None "`: Empty square (note: includes space)

6 Test Results

This section documents the results of automated testing performed on the Othello/Reversi game implementation.

6.1 Test Suite Overview

The game includes a comprehensive test suite (`test.py`) that validates core functionality, game logic, and API endpoints. All tests must pass before considering the implementation complete.

6.2 Test Results Summary

Test	Result
Board creation	PASS
Legal move validation	PASS
Move availability detection	PASS
Piece counting	PASS
Game over conditions	PASS
Piece flipping (capture)	PASS
AI move generation	PASS
Flask API routes	PASS
Full game flow	PASS
Total	**9/9 PASSED**

6.3 Detailed Test Output

Listing 2: Test Execution Output

```
running tests
PASS: Board creation test
PASS: Legal move test
PASS: Move is available test
PASS: Count counters test
PASS: Game over test
PASS: Swap colours test
PASS: AI function test
60
60
60
PASS: Flask routes test
PASS: Full game flow test

=== Results: 9/9 passed ===
```

6.4 Test Descriptions

6.4.1 Board Creation Test

Validates that the game board initializes correctly:

- Board dimensions are 8×8
- Center pieces are correctly placed (Light at (3,3) and (4,4), Dark at (3,4) and (4,3))
- Corner squares are initially empty

6.4.2 Legal Move Test

Verifies move validation logic:

- Prevents moves on occupied squares

- Rejects moves with no captures
- Identifies known legal opening moves
- Handles out-of-bounds coordinates

6.4.3 Move Availability Test

Tests detection of available moves:

- Full board has no available moves
- Empty board has no available moves
- Starting board has moves for both players

6.4.4 Count Counters Test

Validates piece counting function:

- Empty board returns (0, 0)
- Full dark board returns (0, 64)
- Starting board returns (2, 2)
- Mixed boards count correctly

6.4.5 Game Over Test

Tests end-game detection:

- Identifies draw conditions (equal pieces)
- Correctly declares Light wins
- Correctly declares Dark wins

6.4.6 Swap Colours Test

Validates piece capture/flipping logic:

- Single pieces flip correctly
- Multi-direction captures work properly
- New piece placement is correct

6.4.7 AI Function Test

Tests computer opponent:

- Returns valid (x,y) coordinates
- Moves are legal according to game rules
- Returns (-1,-1) when no moves available

6.4.8 Flask Routes Test

Validates web API endpoints:

- Root endpoint returns HTML page
- /move endpoint handles missing parameters
- /move endpoint rejects invalid coordinates
- Proper JSON responses for all cases

6.4.9 Full Game Flow Test

Tests complete game turn:

- Finds legal move for human player
- Resets game state correctly
- Tests integration between components

6.5 Testing Methodology

The test suite follows these principles:

- **Isolation**: Each test focuses on a single function or component
- **Edge Cases**: Tests include boundary conditions (full/empty boards, out-of-bounds coordinates)
- **Integration**: Tests combined functionality (human move + AI response)
- **Automation**: All tests run with a single command

6.6 Running the Tests

To execute the test suite:

1. Navigate to the game directory:

```
cd ~/othello_game
```

2. Run the test script:

```
python test.py
```

3. Verify all tests pass (should show ”9/9 passed”)

The test suite should be run after any code modifications to ensure functionality remains intact.

End of User Manual