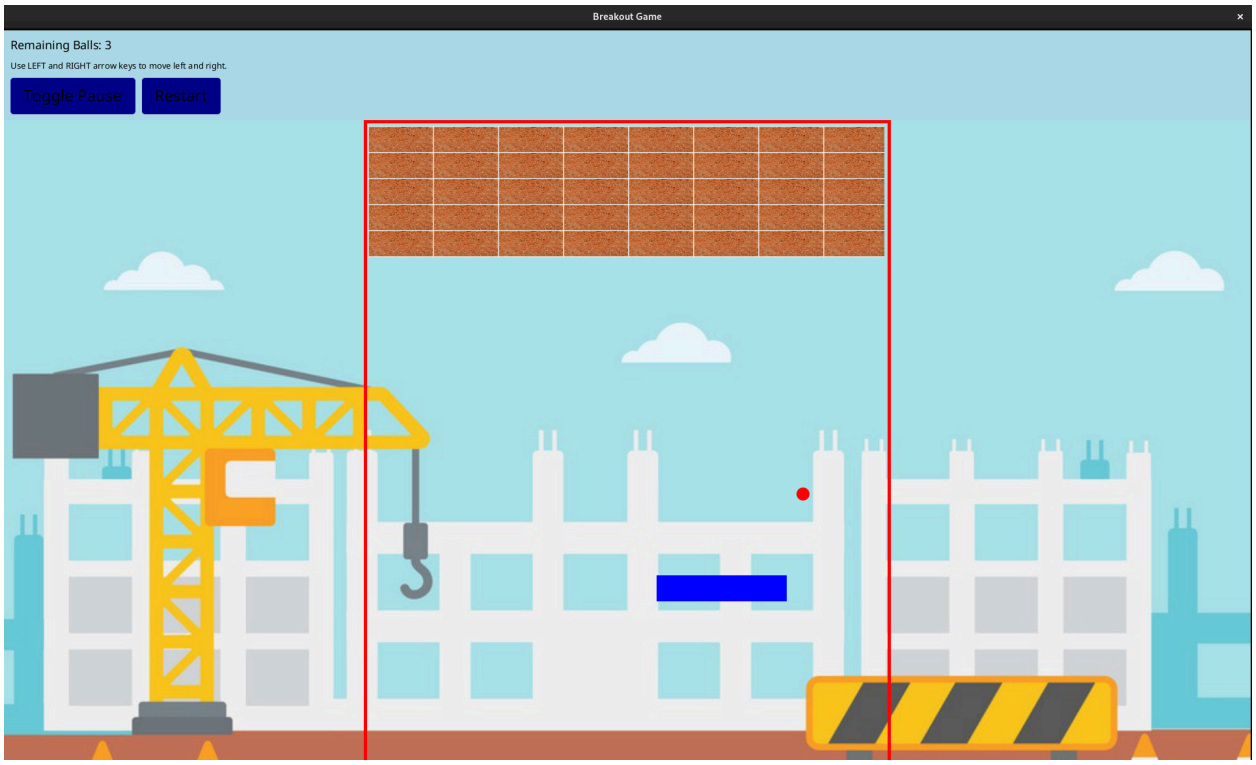
Testing Manual for Breakout Game



Introduction

Hello, I hope you've been enjoying our game so far. In this testing manual, you will find our testing strategy, testing plan, and testing scenarios we've used throughout the development of our game. We will go in-depth on why you might be experiencing your error and explain our program's behaviors alongside examples. Our testing scenarios have been verified by us, so please attempt to recreate our results and don't take our word for it or do.

Test Strategy for `Breakout Game`

Objectives

The main objectives of the testing strategy are to:

- Ensure the Breakout Game fulfills all specified functional requirements.
- Check the application's behavior under a range of scenarios and edge cases.
- Assess the user interface for both usability and accessibility.
- Verify the application's performance efficiency across varying load conditions.

Risk and Issues

Our testing, although comprehensive, does not cover all edge cases and possible scenarios. We've done our best to include all scenarios we believe our program should be able to handle and not and now it's up to you the user to verify our results.

Test Environment

- **Operating System**: Windows 10, macOS, or Linux
- **Java Version**: JDK 11 or later
- **JavaFX Version**: 11 or later
-

Prerequisites:

- Java Development Kit (JDK) installed on your system.
- JavaFX SDK installed.

Running the Program:

1. **Navigate to the Source Directory**.
2. **Compile the Java files**: Use the `javac` command to compile all Java files in the specified package directory.
3. **Run the JavaFX Application**: Use the `java` command to run the main application class. Ensure that you specify the fully qualified class name, including the package.

Notes:

JavaFX Modules: Since JavaFX is not included in the JDK by default, ensure you have the JavaFX SDK installed and properly configured. If you are using Java 11 or later, you might need to add the JavaFX modules explicitly:

```
java --module-path /path/to/javafx-sdk/lib --add-modules
javafx.controls,javafx.fxml edu.rpi.cs.csci4963.u24.chane5.finalproject.java
```

Summary:

To start the program:

1. Ensure you have Java and JavaFX installed on your system.
2. Compile and run the `BreakoutApplication`class.

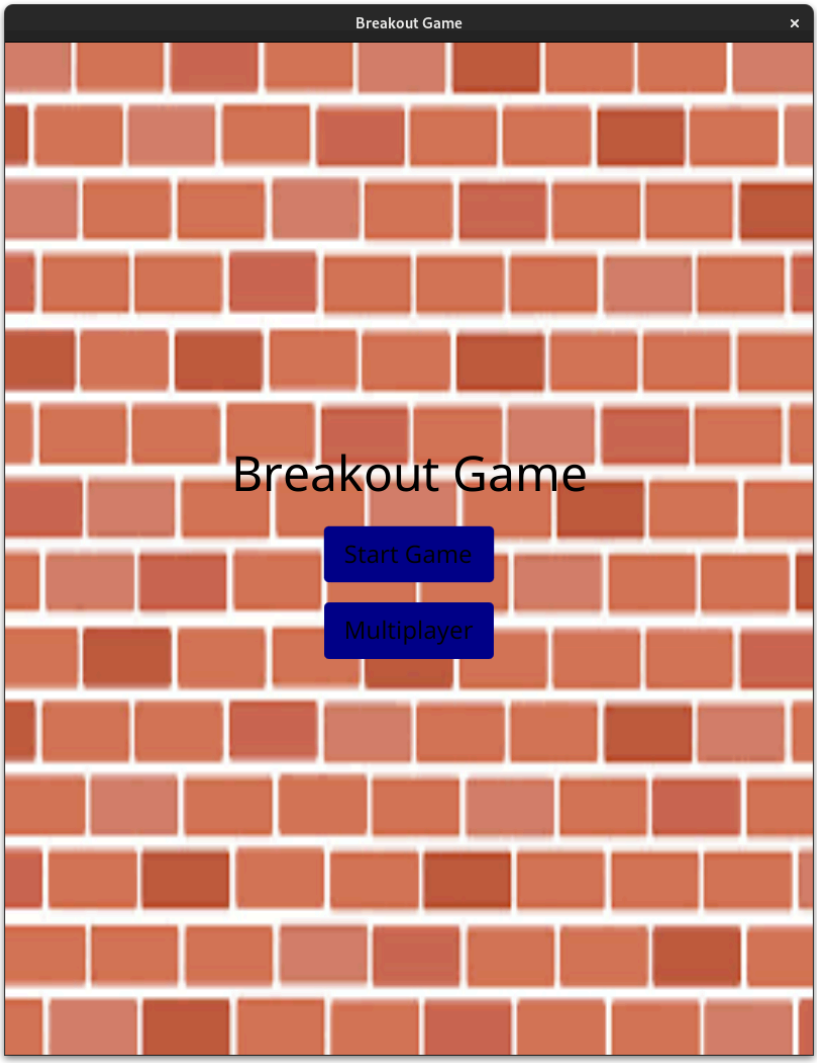
Test Cases:

Functional Tests:

Application Launch:

- **Test Case**: Launch the BreakoutApplication.
- **Steps**:
    1. Open a terminal or command prompt.
    2. Navigate to the directory containing the compiled classes.
    3. Run the application using the command provided in the setup instructions.
- **Expected Result**: The application window should open without errors.

  The main window should look like so:

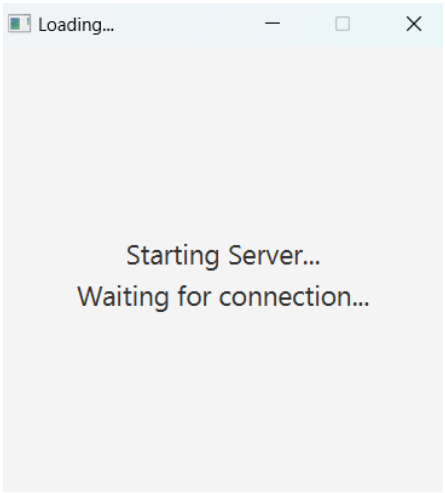


Hosting Game

Steps

1. From the start page click the multiplayer button.
2. Enter port to host on.
3. Wait for other players to connect.

Expected Result:

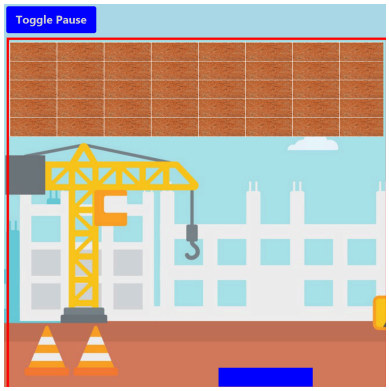


Joining Game

Steps

1. From the start page click the multiplayer button.
2. Wait for the other player to start hosting server
3. Input server IP address (if on same network input localhost) and server port it's being hosted on and connect

Expected Result:

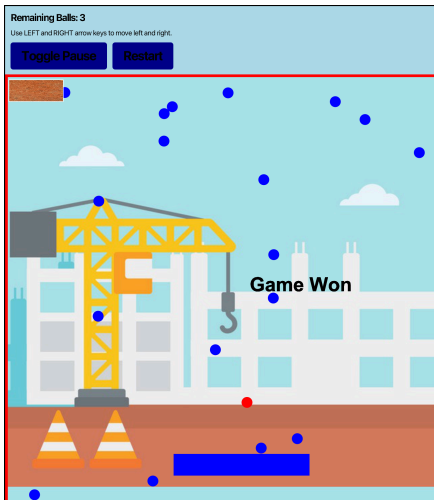


Win Game

Steps:

1. **Join a game**
2. **Play the game until all bricks are broken**
3. **Game Won title displays**

Expected Result:



Game Over

Steps

1. **Join a multiplayer or single player game**
2. **Play until the remaining balls goes to zero**

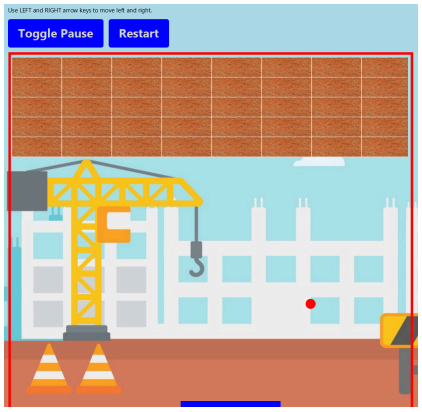
Expected Result: The game should stop and the Game Over title should be displayed

Pause Game

Steps

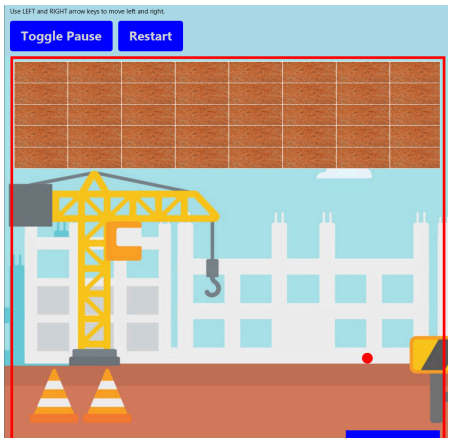
1. **Start or join a multiplayer or single player game**
2. **Click the pause button**

Expected Outcome: The game should pause after the button is pressed



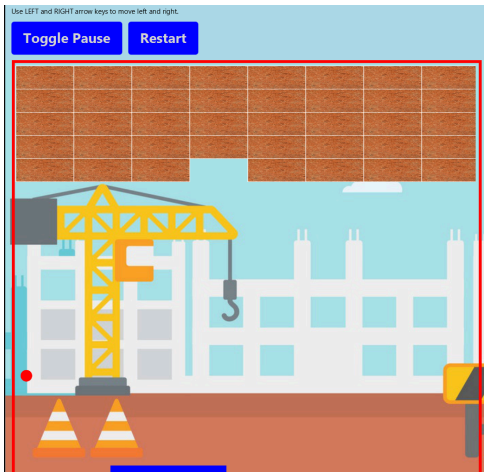
Ball-Paddle Collision Detection

- **Objective: Verify that the ball correctly bounces off the paddle.**
- **Steps:**
    1. **Start a game.**
    2. **Move the paddle to intercept the ball.**
- **Expected Outcome: The ball should bounce back at an angle determined by where it hits the paddle.**



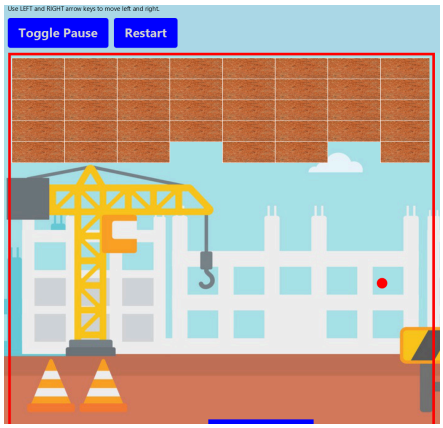
Ball-Wall Collision Detection

- Objective: Ensure the ball bounces correctly off the walls.
- Steps:
    1. Start a game.
    2. Observe the ball hitting the left, right, and top walls.
- Expected Outcome: The ball should bounce back without passing through the walls.



Brick Collision Detection

- Objective: Confirm that the ball breaks bricks when it collides with them.
- Steps:
    1. Start a game.
    2. Direct the ball towards bricks.
- Expected Outcome: The bricks should break, and the ball should continue its trajectory.

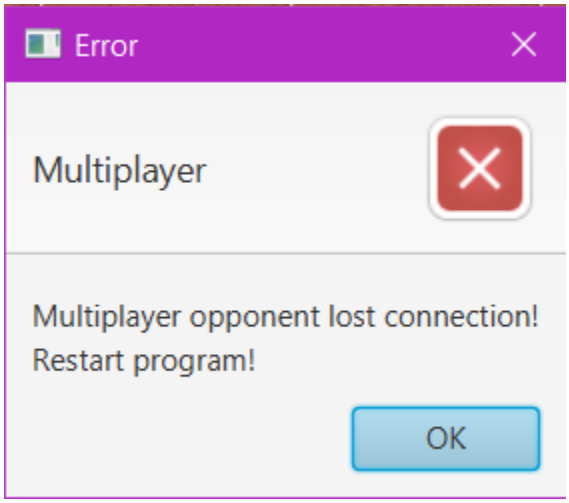


Connection Stability

- **Objective: Verify that the game maintains a stable connection between peers.**
- **Steps:**
    1. **Start a peer-to-peer game.**
    2. **Play the game for 3 minutes.**
- **Expected Outcome: The connection should remain stable without disconnections.**

Peer Disconnection

- **Objective: Test the game's behavior when one peer disconnects.**
- **Steps:**
    1. **Start a peer-to-peer game.**
    2. **Have one player disconnect mid-game.**
- **Expected Outcome: The game should notify the remaining player of the disconnection and handle the situation gracefully.**



Testing Suite Description:

The testing suite for the JavaFX Breakout game, which features both single-player and peer-to-peer multiplayer modes, was meticulously designed and executed to validate the game's functionality, performance, and user experience. Unit tests were implemented to verify the correctness of individual game components, such as paddle movement, ball collision detection, and brick-breaking mechanics. Integration tests confirmed the seamless interaction between the JavaFX front-end and the game's back-end logic, ensuring that game states were correctly managed and displayed in both single-player and multiplayer modes. The peer-to-peer multiplayer functionality was rigorously tested to verify network communication, synchronization

of game states between peers, and the handling of potential network issues, such as latency or disconnections. Performance tests evaluated the game's ability to maintain smooth gameplay under varying conditions, such as different screen sizes or network speeds. Additionally, end-to-end tests simulated typical player journeys, confirming that the game provided a consistent and enjoyable experience across all modes, from launching the game to completing levels or matches.

Conclusion

I hope the testing manual provided insight into the workings of my program and if you had encountered an error provided clarity for the issue you are having. I hope that my test cases have verified the correctness of the program's outputs and completely erased all doubt. By following the steps provided in the user manual, I can almost ensure that the program will output valid results and properly stimulate your desired imputed seeds.