SE 3XA3: Test Report
Asteroid War Game

Team 12, 3XA3-Lab3-Group12
Student 1 Eric Thai Thaie1
Student 2 Tianzheng Mai and mait6
Student 3 Linqi Jiang and jiangl21
Student 4 Junhong Chen and chenj297

April 12, 2021

Contents

List of Tables

Table 1: **Revision History**

Date	**Version**	**Notes**
April 10th	1.0	Updated Functional Requirement Evaluation
April 11th	2.0	Updated Non-functional Requirement Evaluation
April 12th	3.0	Updated Comparison to Existing Implementation and Unit Testing
April 12th	4.0	Updated Changes due to testing
April 12th	5.0	Updated Automated Testing
April 12th	6.0	Updated Trace to Requirements and Modules
April 12th	7.0	Updated Code Coverage Metrics

1 Functional Requirements Evaluation

1.1 User Interface Testing

Home Page

1. FR-UI-1

   Testing Type: Functional, Dynamic, Manual.

   Initial State: A new empty tap on the browser.

   Input: The project HTML link.

   Expected Output: The browser displays the home page of the Asteroid game.

   Test Results: Passed.

   How the test will be performed: The testers must have a stable internet connection and a valid browser on the system to access this game.

Battlefield

1. FR-UI-2

   Testing Type: Functional, Dynamic.

   Initial State: Home Page

   Input: The user selects a game mode and press space to start.

   Expected Output: The browser displays the battlefield on the game page.

   Test Result: Passed.

   How the test will be performed: After the tester selects a game mode(1 player or 2 players) and press space to start, the game will generate a battlefield where the user can control the spaceship to fly on a battlefield within an appropriate size on the screen.

Spaceship

1. FR-UI-3

   Testing Type: Functional, Dynamic.

   Initial State: Home Page

   Input: The user selects a game mode (1 player or 2 players) and press space to start

   Expected Output: If the player selected 1 player mode, the game page will display a spaceship on the battlefield. If the player selected 2 players mode, the game page will display two spaceships on the battlefield.

   Test Result: Passed.

   How the test will be performed: After the tester selects a game mode and press space to start, the game will generate one or two spaceships.

User Manual

1. FR-UI-4

   Testing Type: Functional, Dynamic.

   Initial State: Home Page

   Input: The user clicks the user manual button on the home page.

   Expected Output: The browser will display a user manual page.

   Test Result: Passed.

   How the test will be performed: After clicking on the user manual option on the home page, the screen will display a list of game instructions and rules on the user manual page.

Stop Option

1. FR-UI-5

Testing Type: Functional, Manual.

Initial State: Game Page

Input: The user press “T” to pause the game.

Expected Output: The game will pause and display a user manual window.

Test Result: Passed.

How the test will be performed: The game will pause if the user press “T”. If the user press “P”, the game will resume.

Score

1. FR-UI-6

Testing Type: Functional, Dynamic, Manual.

Initial State: Game Page

Input: The user presses the space key to start the game.

Expected Output: After the game starts, the game will display the scores of the game on the right top corner.

Test Result: Passed.

How the test will be performed: The testers need to select one player game mode or two players game mode to view if the scores are displayed on the asteroid screen.

One Players Mode

1. FR-UI-7

Testing Type: Functional, Dynamic, Manual.

Initial State: One Player Mode Game Page

Input: The user presses the space key to start the game.

Expected Output: After the game starts, the screen will display one spaceship on the battlefield. Test Result: Passed.

How the test will be performed: The testers need to press the space key to see if a spaceship displays.

Two Players Mode

1. FR-UI-8

   Testing Type: Functional, Dynamic, Manual.

   Initial State: Two Players Mode Game Page.

   Input: The user presses the space key to start the game.

   Expected Output: After the game starts, the screen will display two spaceships on the battlefield.

   Test Result: Passed.

   How the test will be performed: The testers need to press the space key to see if two spaceship display.

Contact Information

1. FR-UI-9

   Testing Type: Functional, Dynamic.

   Initial State: Home Page.

   Input: The user clicks the "contact us" button on the home page.

   Expected Output: The browser will display a page containing the contact information of all developers.

   Test Result: Passed.

   How the test will be performed: After clicking on the contact us option on the home page, the screen will display a list of contact information on the contact us page.

1.2 Game Mechanics Testing

Spaceship Speed

1. FR-GM-1

   Testing Type: Functional, Static, Manual.

   Initial State: Game Page

   Input: The tester uses the keyboard to play the game.

   Expected Output: The spaceship should fly at the same speed as the same value of speed function in game.js and game2.js.

   Test Result: Passed.

   How the test will be performed: The testers need to play the game multiple times to test if the speed of the spaceship is the same speed as the value of the speed function in game.js and game2.js.

Score Record

1. FR-GM-2

   Testing Type: Functional, Dynamic, Manual.

   Initial State: Game Page.

   Input: The tester controls the spaceship attack the asteroids.

   Expected Output: The scoring board will increase the score simultaneously when the spaceship attacks the asteroids.

   Test Result: Passed.

   How the test will be performed: The tester controls the keyboard to make the spaceship attack the asteroids and check if the scores are updated simultaneously.

Asteroid Allocation:

1. FR-GM-3

   Testing Type: Functional, Dynamic, Manual.

   Initial State: Game Page.

   Input: The user press space to start.

   Expected Output: After the game started, the asteroid will be distributed in random areas of the battlefield.

   Test Result: Passed.

   How the test will be performed: The tester plays the game manually and tests if the asteroids are distributed in random areas of the battlefield.

Lives Test:

1. FR-GM-4

   Testing Type: Functional, Dynamic, Manual.

   Initial State: Game Page (Either 1 player mode or 2 player mode)

   Input: Move the spaceship to hit an asteroid.

   Expected Output: The lives of the spaceship will be reduced by 1 and the spaceship icon of the specific player on the right top corner disappears by 1.

   Test Result: Passed.

   How the test will be performed: The tester needs to control the keyboard to move the spaceship to hit an asteroid. If the original lives are greater than 1, the number of lives will be reduced by 1 and the spaceship icon of the specific player on the right top corner disappears by 1.

Game Over Test :

1. FR-GM-5

   Testing Type: Functional, Dynamic, Manual.

Initial State: Game Page (Either 1 player mode or 2 player mode)

Input: Move the spaceship to hit an asteroid.

Expected Output: The page will show a game over and final scores message when the only player dies in 1 player mode or two players die in two players mode.

Test Result: Passed.

How the test will be performed: The tester needs to control the keyboard to move the spaceship to hit an asteroid. If the original lives are less than 1, the page will show a game over and final scores message.

2 Nonfunctional Requirements Evaluation

2.1 Gaming Quality

2.1.1 Usability

Usability and Humanity Requirement

1. NFR-UH-1

   Testing Type: Non-Functional, Manual.

   Initial State: Game Page and Home Page.

   Input/Condition: Multiple users who play the game.

   Expected Output/Result: User experience of the game's effectiveness, efficiency, and overall satisfaction.

   Test Result: Passed.

   How the test will be performed: Multiple testers will be invited to play the game and provide feedback experience in the game effectiveness, efficiency, and overall satisfaction.

2.1.2 Performance

Performance Requirement

1. NFR-PR-1

   Testing Type: Non-Functional, Dynamic, Manual.

   Initial State: Game Page.

   Input/Condition: The user plays the game (either one player mode or two players mode) on the keyboard.

   Expected Output/Result: The game is operated and displayed properly on the screen.

   Test Result: Passed.

   How the test will be performed: Multiple testers will be invited to play the game in different modes and test whether the game can interact efficiently between the keyboard control and screen.

2.2 System Quality

Operation and environment requirement

1. NFR-OE-1

   Testing Type: Non-Functional, Dynamic, Manual.

   Initial State: Valid browsers

   Input/Condition: The user tries to open the game in different popular browsers such as Firefox, Google Chrome, Microsoft Edge.

   Expected Output/Result: The game should be operated and displayed properly in different browsers.

   Test Result: Passed.

   How the test will be performed: The tester runs the game in different browsers, but the game in different browsers should have the same performance.

2.2.1 Maintainability

Maintainability and Support Requirement

1. NFR-MS-1

   Testing Type: Non-Functional, Dynamic, Manual.

   Initial State: Windows system or macOS system

   Input/Condition: The user runs the game in the Windows system and macOS system separately.

   Expected Output/Result: The game should be operated correctly in both Windows Systems and macOS systems.

   Test Result: Passed.

   How the test will be performed: The tester runs the game in the Windows and macOS systems and the game should have the same performance.

2.2.2 Robustness

Robustness

1. NFR-R-1

   Testing Type: Non-Functional, Dynamic, Manual.

   Initial State: Game Page

   Input/Condition: The user plays the game (either one player mode or two players mode) on the keyboard.

   Expected Output/Result: The game is operated and displayed properly on the screen Without any latency and errors.

   Test Result: Passed.

   How the test will be performed: Multiple testers will be invited to play the game in different modes and test whether there are any latency and errors.

3 Comparison to Existing Implementation

The original source code did not include testing so this section is not applicable.

4 Unit Testing

The team make the decision to use jasmine and vanilla JavaScript to conduct unit testing of only the core functionality of the game. We decided that the majority of our testing would be manual end to end testing would be better since it is a game. End to end testing allows for and entire game scenario flow to be tested ensuring everything works as intended.

5 Changes Due to Testing

5.1 Changes Made in the testing Plan

In the testing plan, our group was planning to use implement some unit test cases to test the individual functions or methods in the JavaScript file. However, we omitted this unit testing plan because unit testing is extremely inefficient and difficult to implement in our Asteroid game. It is extremely difficult to test the individual unit or component of the game because other states of the function may get impacted while doing the unit testing. Therefore, we decide to use Function, Non-Function, Dynamic, Manual testing as our main software testing techniques which aim to focus on testing the functionality, behavior, performance, and quality of the Asteroid War game. More importantly, those test cases can guarantee the user to have a better user experience and play the game without any errors.

After testing our end to end testing scenarios, we found that in the two player mode, if both players died at the same time the game would be stuck in an incorrect state. To fix this, we had to add an additional check for when both players died at the same time. This was because originally there was a possibility that if both players died at the same time, only one player would spawn.Another change that was made from testing were changes to the game's performance. We found that the game would freeze during transitions to certain states so to fix this we utilized the sleep() function to wait

during state transitions.

6 Automated Testing

6.1 Infeasibility of Automated testing in this project

The automated testing technique is not an efficient solution to test this project. The game modules and game files are so large that it is can not easy to develop multiple automation test cases. In this regard, we choose to use Function, Non-Function, Dynamic, Manual testing which tests the functionality, behavior, performance, and quality of the project such as usability, performance, maintainability, and robustness.

7 Trace to Requirements

Table 2: **Requirements Traceability**

Requirement ID	**Test ID(s)**
FR1, FR7, UHR	FR-UI-1, FR-UI-2, FR-UI-3, FR-UI-4, FR-UI-9, NFR-UH-1, NFR-MS-1, NFR-R-1
FR7, FR8	FR-UI-4, FR-UI-5
FR2, FR9	FR-UI-6, FR-GM-2
FR5, FR6	FR-UI-7, FR-UI-8, FR-GM-1, FR-GM-2, FR-GM-3, NFR-PR-1
FR3, FR4, FR6, FR9	FR-GM-4, FR-GM-5

8 Trace to Modules

Table 3: **Module Traceability**

Test IDs	**Modules**
FR-UI-1	M1, M2, M6, M8
FR-UI-2	M2, M4, M6
FR-UI-3	M2, M3, M4, M5, M6
FR-UI-4	M3, M4, M6, M8
FR-UI-5	M2, M3, M4
FR-UI-6	M2, M3, M4
FR-UI-7	M2, M3, M4, M5
FR-UI-8	M2, M3, M4, M5
FR-UI-9	M2, M3, M4, M8
FR-GM-1	M2, M3, M4, M5, M6
FR-GM-2	M2, M3, M5, M6, M7, M9
FR-GM-3	M2, M3, M4, M6, M9
FR-GM-4	M2, M3, M4, M5, M7
FR-GM-5	M2, M3, M4, M5
NFR-UH-1	M2-9
NFR-PR-1	M1-9
NFR-OE-1	M1
NFR-MS-1	M1
NFR-R-1	M1-3

9 Code Coverage Metrics

We decided to go against code coverage metrics to evaluate our project as the majority of our tests were manual end to end tests. Since our tests were not automated and because we utilized end to end tests, we were not able to determine an exact code coverage.