

# SE 3XA3: Test Report

## Mini-Arcade

Andrew Hum  
huma3  
400138826

William Lei  
leim5  
400125240

Arshan Khan  
khana172  
400145605

Jame Tran  
tranj52  
400144141

# Contents

<b>1</b>	<b>Functional Qualities Evaluation</b>	<b>3</b>
1.1	Launcher Functional Tests . . . . .	3
1.2	Scoreboard Functional Tests . . . . .	3
1.3	Maze Functional Tests . . . . .	3
1.4	Pong Functional Tests . . . . .	4
1.5	Flappy Functional Tests . . . . .	5
<b>2</b>	<b>Nonfunctional Qualities Evaluation</b>	<b>5</b>
2.1	Usability . . . . .	5
2.1.1	GUI Testing . . . . .	5
2.1.2	Game-play Testing . . . . .	6
2.1.3	Running the Product . . . . .	6
2.2	Performance and Robustness . . . . .	6
2.2.1	Game-play Testing . . . . .	6
<b>3</b>	<b>Changes Due to Testing</b>	<b>7</b>
3.1	Launcher Testing . . . . .	7
3.2	Scoreboard Testing . . . . .	7
3.3	Maze Testing . . . . .	7
3.4	Pong Testing . . . . .	7
3.5	Flappy Testing . . . . .	7
<b>4</b>	<b>Automated Testing</b>	<b>7</b>
4.1	Launcher Testing . . . . .	7
4.2	Scoreboard Testing . . . . .	7
4.3	Maze Testing . . . . .	7
4.4	Pong Testing . . . . .	8
4.5	Flappy Testing . . . . .	8
<b>5</b>	<b>System Tests</b>	<b>8</b>
5.1	Launcher Testing . . . . .	8
5.2	Scoreboard Testing . . . . .	9
5.3	Maze Testing . . . . .	9
5.4	Pong Testing . . . . .	11
5.5	Flappy Testing . . . . .	14
5.6	Usability Testing . . . . .	14
5.7	Performance and Robustness Testing . . . . .	15
<b>6</b>	<b>Trace to Requirements</b>	<b>16</b>
6.1	Functional Requirements Traceability Matrix . . . . .	16
6.2	Non-Functional Requirements Traceability Matrix . . . . .	17
6.3	Automated Testing Traceability Matrix . . . . .	18

<b>7</b>	<b>Trace to Modules</b>	<b>19</b>
7.1	Functional Requirements Traceability Matrix . . . . .	19
7.2	Non-Functional Requirements Traceability Matrix . . . . .	20
7.3	Automated Testing Traceability Matrix . . . . .	21
<b>8</b>	<b>Code Coverage Metrics</b>	<b>22</b>

## List of Tables

<b>1</b>	<b>Revision History</b> . . . . .	<b>2</b>
2	Trace Between Tests and Functional Requirements . . . . .	16
3	Trace Between Tests and Non-Functional Requirements . . . . .	17
4	Trace Between Automated Tests and Requirements . . . . .	18
5	Trace Between Functional Tests and Modules . . . . .	19
6	Trace Between Non-Functional Tests and Modules . . . . .	20
7	Trace Between Automated Tests and Modules . . . . .	21

## List of Figures

Table 1: **Revision History**

Date	Version	Notes
4/2/2020	1.0	Andrew - Formatted Document & Section 1 & Section 8
4/3/2020	1.1	Andrew - Section 2, 3, 4, 5, 6, & 7
4/3/2020	1.2	William - Section 1-7
4/5/2020	1.3	Arshan - Pong related sections

# 1 Functional Qualities Evaluation

**Description of Tests:** These tests are used to demonstrate that the requirements of the product are fulfilled based upon the functional requirements from the Software Requirements Specification. These tests will demonstrate basic functionality of the game and its correctness.

## 1.1 Launcher Functional Tests

Test Name: FR-N-1

Results: The user is able to open up scoreboard.

Test Name: FR-N-2

Results: The user is able to open up Maze.

Test Name: FR-N-3

Results: The user is able to open up Flappy.

Test Name: FR-N-4

Results: The user is able to open up Pong.

Test Name: FR-N-5

Results: The user is able close the application.

## 1.2 Scoreboard Functional Tests

Test Name: FR-N-6

Results: The user is able to open up scoreboard for Maze.

Test Name: FR-N-7

Results: The user is able to open up scoreboard for Pong.

Test Name: FR-N-8

Results: The user is able to open up scoreboard for Flappy.

## 1.3 Maze Functional Tests

Test Name: FR-N-9

Results: The user is able to select How To Play

Test Name: FR-N-12

Results: The user is able to return to the launcher

Test Name: FR-MGM-1

Results: The user is able to select Play

Test Name: FR-MGM-2

Results: The user is able to return to the Main Menu screen

Test Name: FR-MGM-3

Results: The user is able to generate a maze by difficulty

Test Name: FR-MGM-4

Results: The user is able to move the player

Test Name: FR-MGM-5

Results: The user is able to view the Victory screen

Test Name: FR-MGM-7

Results: The user is able to return to the Main Menu screen

## **1.4 Pong Functional Tests**

Test Name: FR-N-14

Results: The user is able to return to the launcher

Test Name: FR-MGP-1

Results: The user is able to select different difficulty levels and maximum scores (visual feedback)

Test Name: FR-MGP-2

Results: The main menu transitions to the game without error

Test Name: FR-MGP-4

Results: The user is able to move the player's paddle up and down

Test Name: FR-MGP-5

Results: The user is able to gain a point by scoring on the AI, or vice versa

Test Name: FR-MGP-6

Results: The game screen transitions to the end game screen without error

Test Name: FR-MGP-7

Results: The user is able to return to the main menu from the end game screen

Test Name: FR-MGP-8

Results: The user is able to return to the Launcher from the end game screen

Test Name: FR-MGP-9

Results: The user is able to suspend gameplay by pausing the game

Test Name: FR-MGP-10

Results: The user is able to unsuspend the gameplay and resume playing

Test Name: FR-MGP-11

Results: The user is able to return to the main menu screen from the pause screen

Test Name: FR-MGP-12

Results: The user is able to return to the Launcher screen from the pause screen

## **1.5 Flappy Functional Tests**

Test Name:

Results:

# **2 Nonfunctional Qualities Evaluation**

## **2.1 Usability**

### **2.1.1 GUI Testing**

Description of Tests: Usability of the Graphical User Interface (GUI) was tested by the group members that did not work on the given screen, and by family members of each of the development team. Due to current events, usability testing was limited, however, we feel that our tests still proved that the system is usable to an acceptable extent.

Test Name: NFT-2

Results: From people outside of the development team, they stated that the GUI was visually appealing and easy to navigate through looks.

Test Name: NFT-3

Results: Sample users stated that the GUI was non-intrusive and didn't inhibit their experience during gameplay.

### **2.1.2 Game-play Testing**

Description of Tests: Users are to play the game with no prior explanation or instructions, and will be judged on how they handle the games and are able to navigate through them.

Test Name: NFT-4

Results: Sample users who had no instructions given to them were able to successfully play and complete the games with ease.

### **2.1.3 Running the Product**

Description of Tests: These tests will focus on being able to run the product with it not previously installed on the computer.

Test Name: NFT-7

Results: The game is able to be easily installed and run with little to no instructions.

## **2.2 Performance and Robustness**

### **2.2.1 Game-play Testing**

Description of Tests: The development team, as well as non-development users, will test the games and launcher through playing/running the software to judge the performance and robustness.

Test Name: NFT-1

Results: The average FPS of the game is above 30

Test Name: NFT-5

Results: The launcher and all games are opened, when requested, in under 30 seconds.

Test Name: NFT-6

Results: Users (both development and non-development) play each of the games and state that there is no noticeable input lag.

## **3 Changes Due to Testing**

### **3.1 Launcher Testing**

During development, many non-formal tests were run to ensure the correctness of the implementation. But after completing the development, there were no changes made due to the result from the tests defined in this document.

### **3.2 Scoreboard Testing**

During development, many non-formal tests were run to ensure the correctness of the implementation. But after completing the development, there were no changes made due to the result from the tests defined in this document.

### **3.3 Maze Testing**

Through code development, many non-formal test methods were run to ensure correctness. However, once the program was completed, there were no changes made due to the formal testing methods and cases defined in this document.

### **3.4 Pong Testing**

Throughout code development, many non-formal test methods were run to ensure continuity with the requirements. However, once the program was completed, there were no major changes made due to the formal testing methods and cases defined in this document.

### **3.5 Flappy Testing**

## **4 Automated Testing**

### **4.1 Launcher Testing**

All tests for Launcher is done manually because they are GUI related.

### **4.2 Scoreboard Testing**

All tests for Scoreboard is done manually because they are GUI or file IO related.

### **4.3 Maze Testing**

Description of tests: Given that games are difficult to test using automated testing, there are a limited amount of automated tests that were able to be run. These tests are located in the Maze folder under the file name testMaze.py. These tests are used to test for correct



object creation and that the functions within the objects work correctly.

Test Names: AMT1, AMT2, AMT3, AMT4, AMT5 Results: These 5 test cases were all executed successfully and passed. This shows that the entity objects used in the maze program work correctly and are providing the correct data.

## 4.4 Pong Testing

Description of tests: Since automated testing is limited for game applications, the automated tests were limited to checking instantiation and calculation functions. The functions affecting visual aspects are better for manual testing

Results: In the pong folder, the file testPong.py contains the test cases for all the Pong modules. These tests will be called ATP1, ATP2, ATP3, ATP4, ATP5, ATP6, ATP7, ATP8, ATP9, ATP10. All of these tests passed successfully which shows that all objects are instantiated and calculations are implemented correctly.

## 4.5 Flappy Testing

Description of tests:

Results:

# 5 System Tests

## 5.1 Launcher Testing

**Test Name** - FR-N-1

**Initial State** - Main Screen

**Input** - User clicks on Scoreboard

**Expected Output** - Scoreboard opens and is displayed on the screen.

**Test Name** - FR-N-2

**Initial State** - Main Screen

**Input** - User clicks on Maze

**Expected Output** - The mini-game Maze opens and is displayed on the screen.

**Test Name** - FR-N-3

**Initial State** - Main Screen

**Input** - User clicks on Flappy

**Expected Output** - The mini-game Flappy opens and is displayed on the screen.

**Test Name** - FR-N-4

**Initial State** - Main Screen

**Input** - User clicks on Pong

**Expected Output** - The mini-game Pong opens and is displayed on the screen.

**Test Name** - FR-N-5

**Initial State** - Main Screen

**Input** - User clicks on close button

**Expected Output** - The software will be terminated.

## 5.2 Scoreboard Testing

**Test Name** - FR-N-6

**Initial State** - Scoreboard Screen

**Input** - User clicks on Maze

**Expected Output** - The scoreboard screen will display the scoreboard for Maze.

**Test Name** - FR-N-7

**Initial State** - Scoreboard Screen

**Input** - User clicks on Pong

**Expected Output** - The scoreboard screen will display the scoreboard for Pong.

**Test Name** - FR-N-8

**Initial State** - Scoreboard Screen

**Input** - User clicks on Flappy

**Expected Output** - The scoreboard screen will display the scoreboard for Flappy.

## 5.3 Maze Testing

**Test Name** - FR-N-9

**Initial State** - Main Menu of Maze

**Input** - Mouse click on the 'How To Play' button

**Expected Output** - The 'How To Play' screen that shows the instructions of the game.

**Test Name** - FR-N-12

**Initial State** - Main Menu of Maze

**Input** - Mouse click on the 'Home' button

**Expected Output** - Returns the user back to the Launcher screen

**Test Name** - FR-MGM-1

**Initial State** - Main Menu of Maze

**Input** - Mouse click on the 'Play' button

**Expected Output** - Brings the user to the 'Difficulty Selection' screen

**Test Name** - FR-MGM-2

**Initial State** - Pause Screen / End Screen

**Input** - Mouse click on the 'Menu' button

**Expected Output** - Brings the user back to the Main Menu screen of the Maze

**Test Name** - FR-MGM-3

**Initial State** - Difficulty Screen of Maze

**Input** - A difficulty selection via mouse click

**Expected Output** - A randomly generated maze, with the size defined by the selected difficulty

**Test Name** - FR-MGM-4

**Initial State** - Playing the Maze

**Input** - Keyboard input of either 'WASD' or the arrow keys

**Expected Output** - The player object moves according to the keyboard input

**Test Name** - FR-MGM-5

**Initial State** - Playing the Maze

**Input** - Reaches the goal via keyboard input

**Expected Output** - The victory screen is displayed with the user's current score and the overall high score.

**Test Name** - FR-MGM-7

**Initial State** - Victory Screen of Maze

**Input** - Mouse click on the 'Continue' button

**Expected Output** - Brings the user back to the main menu screen of Maze

**Test Name** - AMT1

**Initial State** - No Cell created

**Input** - Create a cell

**Expected Output** - A Cell object is successfully created

**Test Name** - AMT2

**Initial State** - Cell created

**Input** - Cell with id 0 in a 5x5 maze

**Expected Output** - Cell has walls connecting it to Cell 1 and Cell 6

**Test Name** - AMT3  
**Initial State** - Cell created  
**Input** - Cell with id 6 in a 5x5 maze  
**Expected Output** - 4 surrounding walls

**Test Name** - AMT4  
**Initial State** - Cell created  
**Input** - Newly generated Cell  
**Expected Output** - The current cell has not been visited

**Test Name** - AMT5  
**Initial State** - No Player Created  
**Input** - Create a Player  
**Expected Output** - A Player object is successfully created

## 5.4 Pong Testing

**Test Name** - FR-N-14  
**Initial State** - Return to the Launcher  
**Input** - Mouse click on the Quit Game button on the Main Menu  
**Expected Output** - Display the Launcher screen

**Test Name** - FR-MGP-1  
**Initial State** - Pong Main Menu screen  
**Input** - Mouse click on the Difficulty and Maximum Score buttons  
**Expected Output** - Visual indication of selected difficulty and maximum score on the screen

**Test Name** - FR-MGP-2  
**Initial State** - Pong Main Menu screen  
**Input** - Mouse click on the 'Begin' button  
**Expected Output** - Transition to the Game screen and the game begins

**Test Name** - FR-MGP-4  
**Initial State** - Pong Game screen  
**Input** - Keyboard input: up or down arrow key  
**Expected Output** - The paddle moves up or down with respect to the button pressed

**Test Name** - FR-MGP-5  
**Initial State** - Pong Game screen  
**Input** - Keyboard input: up or down arrow key enough times such that the ball passes the AI's or the player's paddle

**Expected Output** - An increase in the AI's or the player's score by one

**Test Name** - FR-MGP-6

**Initial State** - Pong Game screen

**Input** - The user or AI scores a point so that their total score is equal to the predefined maximum score

**Expected Output** - Transition from the Game screen to the End Game screen

**Test Name** - FR-MGP-7

**Initial State** - Pong End Game screen

**Input** - Mouse click on the 'New Game' button on the End Game screen

**Expected Output** - Transition to the Main Menu screen

**Test Name** - FR-MGP-8

**Initial State** - Pong End Game screen

**Input** - Mouse click on the 'Quit Game' button on the End Game screen

**Expected Output** - Transition to the Launcher screen

**Test Name** - FR-MGP-9

**Initial State** - Pong Game screen

**Input** - Keyboard input: 'ESC' key

**Expected Output** - Suspend gameplay and transition to the Game Pause screen

**Test Name** - FR-MGP-10

**Initial State** - Pong Game Pause screen

**Input** - Mouse click on 'Resume' button or Keyboard input: 'ESC' key

**Expected Output** - Unsuspend gameplay and transition back to Game screen

**Test Name** - FR-MGP-11

**Initial State** - Pong Game Pause screen

**Input** - Mouse click on 'New Game' button

**Expected Output** - Transition to the Main Menu screen from the Game Pause screen

**Test Name** - FR-MGP-12

**Initial State** - Pong Game Pause Screen

**Input** - Mouse click on the 'Quit Game' button

**Expected Output** - Transition to the Launcher screen from the Game Pause screen

**Test Name** - ATP1

**Initial State** - No Paddles created

**Input** - Create two paddles of arbitrary size and location

**Expected Output** - Two instantiated Paddle objects

**Test Name** - ATP2

**Initial State** - Existing Paddle

**Input** - Check attributes of a paddle: coordinates, size, color, etc

**Expected Output** - All attributes are at their expected location

**Test Name** - ATP3

**Initial State** - Existing Paddle

**Input** - Draw paddle to screen

**Expected Output** - Object still exists and no error message is shown

**Test Name** - ATP4

**Initial State** - No existing Ball

**Input** - Create a Ball object

**Expected Output** - Ball object is instantiated

**Test Name** - ATP5

**Initial State** - Existing Ball

**Input** - Check attributes of a ball: coordinates, color, size, and location

**Expected Output** - All attributes are at their expected values

**Test Name** - ATP6

**Initial State** - Existing Ball

**Input** - Draw ball to screen

**Expected Output** - Object still exists and no error message is shown

**Test Name** - ATP7

**Initial State** - None

**Input** - Enter a new difficulty

**Expected Output** - The new difficulty is equivalent to the desired value

**Test Name** - ATP8

**Initial State** - None

**Input** - Enter a new maximum score

**Expected Output** - The new maximum score is equivalent to the desired value

**Test Name** - ATP9

**Initial State** - None

**Input** - Display text to the screen

**Expected Output** - Text is shown on the screen, somewhat briefly

**Test Name** - ATP10

**Initial State** - None

**Input** - Three arbitrary values for the playerScore, aiScore, and maxScore into the calculateScore function

**Expected Output** - Actual final score matches our expected final score

## 5.5 Flappy Testing

**Test Name** -

**Initial State** -

**Input** -

**Expected Output** -

## 5.6 Usability Testing

**Test Name** - NFT-2

**Initial State** - Program is launched on default settings and is at the launcher screen

**Input** - User plays game for 5 minutes

**Expected Output** - Users provide feedback stating the game is visually appealing and the GUI enhances usability.

**Test Name** - NFT-3

**Initial State** - Program is launched on default settings and is at the launcher screen

**Input** - User plays the game for 5 minutes

**Expected Output** - Users will state the the GUI is non-intrusive and does not take away from the usability and game-play.

**Test Name** - NFT-4

**Initial State** - Program is launcher on default settings and is at the launcher screen

**Input** - Users play the game for 5 minutes

**Expected Output** - Users state that the overall product is intuitive and easy to learn with minimal to no instructions.

**Test Name** - NFT-7

**Initial State** - Mini-Arcade is currently not installed

**Input** - User will attempt to install and run the program

**Expected Output** - User will be able to install and run the program with little-to-no difficulty.

## 5.7 Performance and Robustness Testing

**Test Name** - NFT-1

**Initial State** - Program is launched on default settings and is at the launcher screen

**Input** - User will play the game for 5 minutes

**Expected Output** - User will report minimal-to-no frame drops and pygame will display an average of 30 or greater frames per second

**Test Name** - NFT-5

**Initial State** - Program is launched on default settings

**Input** - User will be asked to launch one of the available games

**Expected Output** - The requested actions will take no longer than 30 seconds to complete

**Test Name** - NFT-6

**Initial State** - Program is launched on default settings and is currently playing one of the available games

**Input** - User will be asked to provide the program with keyboard input

**Expected Output** - They will report minimal-to-no input lag and the game will update within a quarter second of the user input.



## 6 Trace to Requirements

### 6.1 Functional Requirements Traceability Matrix

Test	Req.
FR-N-1	FR4
FR-N-2	FR1, FR2
FR-N-3	FR1, FR2
FR-N-4	FR1, FR2
FR-N-5	FR3
FR-N-6	FR5
FR-N-7	FR5
FR-N-8	FR5
FR-N-9	FR13
FR-N-12	FR10
FR-N-14	FR30
FR-MGM-1	FR7
FR-MGM-2	FR13
FR-MGM-3	FR9
FR-MGM-4	FR12
FR-MGM-5	FR8, FR11, FR13
FR-MGM-7	FR13, FR31, FR3
FR-MGP-1	FP1, FP2
FR-MGP-2	FR25
FR-MGP-4	FR25
FR-MGP-5	FR26
FR-MGP-6	FR28
FR-MGP-7	FR29
FR-MGP-8	FR30
FR-MGP-9	FP3
FR-MGP-10	FP3
FR-MGP-11	FR30
FR-MGP-12	FR30

Table 2: Trace Between Tests and Functional Requirements

## 6.2 Non-Functional Requirements Traceability Matrix

Test	Req.
NFT-1	NFR1, NFR8, NFR9, NFR10, NFR11, NFR12
NFT-2	NFR2, NFR8, NFR9, NFR10, NFR11, NFR12
NFT-3	NFR3, NFR8, NFR9, NFR10, NFR11, NFR12
NFT-4	NFR4, NFR8, NFR9, NFR10, NFR11, NFR12
NFT-5	NFR5, NFR8, NFR9, NFR10, NFR11, NFR12
NFT-6	NFR6, NFR8, NFR9, NFR10, NFR11, NFR12
NFT-7	NFR7, NFR8, NFR9, NFR10, NFR11, NFR12

Table 3: Trace Between Tests and Non-Functional Requirements

### 6.3 Automated Testing Traceability Matrix

Test	Req.
AMT1	FR9
AMT2	FR9
AMT3	FR9
AMT4	FR9
AMT5	FR11, FR12
ATP1	FR25
ATP2	FR25
ATP3	FR25
ATP4	FR26, FR28
ATP5	FR26, FR28
ATP6	FR26, FR28
ATP7	FP1
ATP8	FP2
ATP9	FP1, FP2, FR29, FR30
ATP10	FR28

Table 4: Trace Between Automated Tests and Requirements

## 7 Trace to Modules

### 7.1 Functional Requirements Traceability Matrix

Test	Modules
FR-N-1	M2, M3
FR-N-2	M2
FR-N-3	M2
FR-N-4	M2
FR-N-5	M2
FR-N-6	M3
FR-N-7	M3
FR-N-8	M3
FR-N-9	M8
FR-N-12	M2, M8
FR-N-14	M13
FR-MGM-1	M8
FR-MGM-2	M4, M6, M8
FR-MGM-3	M4, M5, M6, M8
FR-MGM-4	M6, M7
FR-MGM-5	M5, M7, M8
FR-MGM-7	M6, M8
FR-MGP-1	M13
FR-MGP-2	M13
FR-MGP-4	M11, M12
FR-MGP-5	M9, M10, M11, M12
FR-MGP-6	M9, M10, M11, M12
FR-MGP-7	M11, M13
FR-MGP-8	M13
FR-MGP-9	M11, M13
FR-MGP-10	M11, M13
FR-MGP-11	M13
FR-MGP-12	M13

Table 5: Trace Between Functional Tests and Modules

## 7.2 Non-Functional Requirements Traceability Matrix

Test	Modules
NFT-1	M4 - M13
NFT-2	M4 - M13
NFT-3	M4 - M13
NFT-5	M4 - M13
NFT-6	M4 - M13
NFT-7	M4 - M13

Table 6: Trace Between Non-Functional Tests and Modules

### 7.3 Automated Testing Traceability Matrix

Test	Modules
AMT1	M4, M6
AMT2	M4, M6
AMT3	M4, M6
AMT4	M4, M6
AMT5	M7
ATP1	M12
ATP2	M12
ATP3	M11, M12
ATP4	M9
ATP5	M9, M10
ATP6	M9, M11
ATP7	M9, M11, M13
ATP8	M9, M11, M13
ATP9	M11, M13
ATP10	M10, M13

Table 7: Trace Between Automated Tests and Modules

## 8 Code Coverage Metrics

With the tests mentioned throughout this document, our group has produced an approximate 85% code coverage. This is evident through the traceability matrices above, as they show that every module has been covered, with some being covered multiple times.