SE 3XA3: Test Report
Mini-Arcade

Andrew Hum	William Lei	Arshan Khan	Jame Tran
huma3	leim5	khana172	tranj52
400138826	400125240	400145605	400144141

Contents

List of Tables

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

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:
Results:

1.2 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

1.3 Pong Functional Tests

Test Name:
Results:

1.4 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

3.2 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.

3.3 Pong Testing

3.4 Flappy Testing

4 Automated Testing

4.1 Launcher Testing

Description of tests:

Results:

4.2 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.

4.3 Pong Testing

Description of tests:

Results:

4.4 Flappy Testing

Description of tests:

Results:

5 System Tests

5.1 Launcher Testing

Test Name -
Initial State -
Input -
Expected Output -

5.2 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

5.3 Pong Testing

Test Name -
Initial State -

Input -
Expected Output -

5.4 Flappy Testing

Test Name -
Initial State -
Input -
Expected Output -

5.5 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.

5.6 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-9	FR13
FR-N-12	FR10
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

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

Table 4: Trace Between Automated Tests and Requirements

7 Trace to Modules

7.1 Functional Requirements Traceability Matrix

Test	Modules
FT-1	-
FT-2	-
FT-3	-
FT-4	-
FT-5	-
FT-6	-
FT-7	-
FT-8	-
FT-9	-

Table 5: Trace Between Functional Tests and Modules

7.2 Non-Functional Requirements Traceability Matrix

Test	**Modules**
NFT-1	-
NFT-2	-
NFT-3	-
NFT-4	-
NFT-5	-
NFT-6	-
NFT-7	-
NFT-8	-
NFT-9	-

Table 6: Trace Between Non-Functional Tests and Modules

7.3 Automated Testing Traceability Matrix

Test	**Modules**
AT-1	-
AT-2	-
AT-3	-
AT-4	-
AT-5	-
AT-6	-
AT-7	-
AT-8	-
AT-9	-

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.