SE 3XA3: Test Plan
Mini-Arcade

Team #104
Andrew Hum, 400138826
Arshan Khan, 400145605
Jame Tran, 400144141
William Lei, 400125240

February 27, 2020

Contents

1 General Information **1**
- 1.1 Purpose . . . 1
- 1.2 Scope . . . 1
- 1.3 Acronyms, Abbreviations, and Symbols . . . 1
- 1.4 Overview of Document . . . 1

2 Plan **1**
- 2.1 Software Description . . . 1
- 2.2 Test Team . . . 1
- 2.3 Automated Testing Approach . . . 3
- 2.4 Testing Tools . . . 3
- 2.5 Testing Schedule . . . 3

3 System Test Description **3**
- 3.1 Tests for Functional Requirements . . . 3
  - 3.1.1 General Navigation . . . 3
  - 3.1.2 Mini-Game - Maze . . . 7
  - 3.1.3 Mini-Game - Flappy . . . 9
  - 3.1.4 Mini-Game - Pong . . . 11
- 3.2 Tests for Nonfunctional Requirements . . . 13
  - 3.2.1 Area of Testing1 . . . 13
  - 3.2.2 Area of Testing2 . . . 14
- 3.3 Traceability Between Test Cases and Requirements . . . 14

4 Tests for Proof of Concept **14**
- 4.1 Area of Testing1 . . . 14
- 4.2 Area of Testing2 . . . 15

5 Unit Testing Plan **15**
- 5.1 Unit testing of internal functions . . . 15
- 5.2 Unit testing of output files . . . 15

6 Appendix **16**
- 6.1 Symbolic Parameters . . . 16
- 6.2 Usability Survey Questions? . . . 16

List of Tables

1 Revision History 2
2 Table of Abbreviations 2
3 Table of Definitions 2

List of Figures

1 General Information

1.1 Purpose

The purpose of testing our project is to verify that it meets the requirements outlined in the 'Software Requirements Specification' and ensure that it is implemented correctly.

1.2 Scope

The test plan develops a baseline for testing the functionality and correctness of Mini-Arcade. Its core objective is to verify that the games run correctly and efficiently all with a single click utilizing the launcher. The test plan documents will highlight what is to be tested of our project, testing methods and what resources we will use to test our software.

1.3 Acronyms, Abbreviations, and Symbols

1.4 Overview of Document

This document will outline a detailed testing plan with the tools that will be utilized and the approximated schedule of testing. It will also give in-depth test cases and the method of testing for the functional requirements, non-functional requirements, the proof of concept tests and the unit-testing plan.

2 Plan

2.1 Software Description

The software is a launcher for a selection of games for the user to play. These games are updated from their original versions to be more visually pleasing and challenging.

2.2 Test Team

The test team is composed of all team members: Andrew Hum, Arshan Khan, Jame Tran, and William Lei.

Table 1: **Revision History**

Date	Version	Notes
2/24/2020	1.0	Andrew and Arshan divided the project into workable parts for group members and began the rough draft of sections 1, 2, 5
2/26/2020	1.1	Andrew completed sections 1 and 5
2/27/2020	1.2	Andrew revised sections 1 and 5 for grammatical errors
2/27/2020	1.3	William completed section 3.1

Table 2: **Table of Abbreviations**

Abbreviation	Definition
FDM	Functional, Dynamic and Manual Testing

Table 3: **Table of Definitions**

Term	Definition
Functional Testing	Testing derived from the functional requirements of the software.
Dynamic Testing	Testing through executing test cases during runtime.
Manual Testing	Testing conducted by providing manual inputs and people checking for outputs.

2.3 Automated Testing Approach

The tests will be automated by pytest because it is very popular and allows "assert rewriting".

2.4 Testing Tools

2.5 Testing Schedule

See Gantt Chart at the following url:
../../ProjectSchedule/3XA3-ProjSched.gan.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 General Navigation

1. FR-N-1
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on Leaderboard
   Output: Leaderboard opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

2. FR-N-2
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on Maze
   Output: The mini-game Maze opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

3. FR-N-3
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on Flappy
   Output: The mini-game Flappy opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

4. FR-N-4
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on Pong
   Output: The mini-game Pong opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

5. FR-N-5
   Type: FDM
   Initial State: Main Screen
   Input: User clicks on close button
   Output: The software will be terminated.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

6. FR-N-6
   Type: FDM
   Initial State: Leaderboard Screen
   Input: User clicks on Maze
   Output: The leaderboard screen will display the leaderboard for Maze.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

7. FR-N-7
   Type: FDM
   Initial State: Leaderboard Screen
   Input: User clicks on Flappy
   Output: The leaderboard screen will display the leaderboard for Flappy.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

8. FR-N-8
   Type: FDM
   Initial State: Maze - Menu Screen
   Input: User clicks on Help
   Output: The screen will display the instructions for how to play the mini-game.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

9. FR-N-9
   Type: FDM
   Initial State: Flappy - Menu Screen
   Input: User clicks on Help
   Output: The screen will display the instructions for how to play the mini-game.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

10. FR-N-10
    Type: FDM
    Initial State: Pong - Menu Screen
    Input: User clicks on Help
    Output: The screen will display the instructions for how to play the mini-game.
    How test will be performed: The application will be opened and the

user will manually provides inputs to the software and observes for the output of the software on the screen.

11. FR-N-11
    Type: FDM
    Initial State: Maze - Menu Screen
    Input: User clicks on Leaderboard
    Output: The leaderboard of the mini-game opens and is displayed on the screen.
    How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

12. FR-N-12
    Type: FDM
    Initial State: Flappy - Menu Screen
    Input: User clicks on Leaderboard
    Output: The leaderboard of the mini-game opens and is displayed on the screen.
    How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

13. FR-N-13
    Type: FDM
    Initial State: Maze - Menu Screen
    Input: User clicks on Back
    Output: The Main Screen opens and is displayed on the screen.
    How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

14. FR-N-14
    Type: FDM
    Initial State: Flappy - Menu Screen

Input: User clicks on Back
Output: The Main Screen opens and is displayed on the screen.
How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

15. FR-N-15
    Type: FDM
    Initial State: Pong - Menu Screen
    Input: User clicks on Back
    Output: The Main Screen opens and is displayed on the screen.
    How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

3.1.2 Mini-Game - Maze

1. FR-MGM-1
   Type: FDM
   Initial State: Maze - Menu Screen
   Input: User clicks on a difficulty level
   Output: A maze will displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

2. FR-MGM-2
   Type: FDM
   Initial State: Maze - Game Screen
   Input: User clicks on home
   Output: Menu screen of Maze will displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

3. FR-MGM-3
   Type: FDM
   Initial State: Maze - Menu Screen
   Input: User clicks on a specific difficulty level, then clicks home, and repeats this for 5 times in total
   Output: A maze will displayed on the screen every time the user clicks a difficulty level, and there should be no patterns for when a specific maze will be displayed.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

4. FR-MGM-4
   Type: FDM
   Initial State: Maze - Game Screen
   Input: User clicks a movement key
   Output: The object will move according to the key-movement mapping and the movement will be displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

5. FR-MGM-5
   Type: FDM
   Initial State: Maze - Game Screen
   Input: Object reaches end of maze through a movement
   Output: A score (base on time elapsed) along with high score will be displayed on the end game screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

6. FR-MGM-6
   Type: FDM
   Initial State: Maze - End Game Screen
   Input: User clicks on Next

Output: A maze will displayed on the screen.
How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

7. FR-MGM-7
   Type: FDM
   Initial State: Maze - End Game Screen
   Input: User clicks on Return
   Output: The Menu Screen opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

3.1.3 Mini-Game - Flappy

1. FR-MGF-1
   Type: FDM
   Initial State: Flappy - Menu Screen
   Input: User clicks on start
   Output: The game will be initialized/started and the game screen will be opened and displayed
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

2. FR-MGF-2
   Type: FDM
   Initial State: Flappy - Game Screen
   Input: User controlling the character to make sure it will not collide with any object
   Output: Their will be randomly generated objects approaching toward the character, and their speed and amount generated will be increased as time elapses.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the

output of the software on the screen.

3. FR-MGF-3
   Type: FDM
   Initial State: Flappy - Game Screen
   Input: User clicks space key for 5 times separated by a short period of time
   Output: The character will move up a constant amount every time the space key is being clicked.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

4. FR-MGF-4
   Type: FDM
   Initial State: Flappy - Game Screen
   Input: User controls the character to collide with an object
   Output: A score (base on time elapsed) along with high score will be displayed on the end game screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

5. FR-MGF-5
   Type: FDM
   Initial State: Flappy - End Game Screen
   Input: User clicks on Restart
   Output: The game will be initialized/started and the game screen will be opened and displayed.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

6. FR-MGF-6
   Type: FDM

Initial State: Flappy - End Game Screen
Input: User clicks on Return
Output: The Menu Screen opens and is displayed on the screen.
How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

3.1.4 Mini-Game - Pong

1. FR-MGP-1
   Type: FDM
   Initial State: Pong - Menu Screen
   Input: User clicks on Single Player
   Output: The game screen will be opened and displayed and will request the user to input a max score.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

2. FR-MGP-2
   Type: FDM
   Initial State: Pong - Menu Screen
   Input: User clicks on Multiplayer
   Output: The game screen will be opened and displayed and will request the user to input a max score.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

3. FR-MGF-3
   Type: FDM
   Initial State: Pong - Game Screen (Both single and multiplayer, requesting max score input)
   Input: User inputs a integer between 1 to 10.
   Output: The game will be initialized or started. How test will be performed: The application will be opened and the user will manually

provides inputs to the software and observes for the output of the software on the screen.

4. FR-MGP-4
   Type: FDM
   Initial State: Pong - Game Screen (Both single and multiplayer)
   Input: User clicks a movement key
   Output: The corresponding paddle will move according to the key-movement mapping and the movement will be displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

5. FR-MGP-5
   Type: FDM
   Initial State: Pong - Game Screen (Both single and multiplayer)
   Input: User control the paddle to hit the ball until the ball reaches the boundary on either side (and did not hit a paddle)
   Output: The score of the opposite will be increased by 1 and the change will be displayed on the game screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

6. FR-MGF-6
   Type: FDM
   Initial State: Flappy - Game Screen (Both single and multiplayer)
   Input: User control the paddle to hit the ball until either side reaches the max score
   Output: The score between the two player will be displayed on the end game screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

7. FR-MGP-7
   Type: FDM
   Initial State: Pong - End Game Screen
   Input: User clicks on Restart
   Output: The game will be initialized/started and the game screen will be opened and displayed.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

8. FR-MGP-8
   Type: FDM
   Initial State: Pong - End Game Screen
   Input: User clicks on Return
   Output: The Menu Screen opens and is displayed on the screen.
   How test will be performed: The application will be opened and the user will manually provides inputs to the software and observes for the output of the software on the screen.

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

Type: Functional, Dynamic, Manual, Static etc.

Initial State:

Input:

Output:

How test will be performed:

3.2.2 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2 Area of Testing2

...

5 Unit Testing Plan

The pytest library will be used for the unit testing of our project.

5.1 Unit testing of internal functions

To efficiently use unit-testing for our project, we will use hard-coded, expected, and unexpected, inputs for individual functions and methods. These functions and methods will then provide output, and we will verify that the resulting output is correct or that the program handles the unexpected input correctly. For example, telling the game that the game was won, and the expected output should be the end-game screen. As games are more difficult to completely test with unit tests, we will only test the functions that can be tested by providing an expected and unexpected output with input values relating to a current state or completed event. To cover a wide range of scenarios, the input variables will test both expected output, and reaction to incorrect/unexpected input values. There will be no need for stubs or drivers to test our project. To ensure high-quality coverage, we will be using testing coverage metrics. Our goal is to cover a minimum of 60% of the project with unit tests alone, derived by the total lines of code in the project divided by the number of lines covered by the test cases.

5.2 Unit testing of output files

In-depth testing of the output files using unit testing will be not applicable for our project, and any unit tests to test output files would prove to be not useful and ineffective in both coverage and effective use of time.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions?

This is a section that would be appropriate for some teams. A possible set of questions to ask beta testers would include:

- What game did you play first?
- What did you think of that game?
- How long did you play?
- Would you play again?
- Did you play any other games?
- Was it easy to get started?