SE 3XA3: Software Requirements Specification
Mini-Arcade

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

April 7, 2020

Contents

List of Tables

- Table 1: Revision History
- Table 2: Work Partitioning Events
- Table 3: Work Partitioning Event Summaries

List of Figures

- Figure 1: System Use Case Diagram

Table 1: **Revision History**

Date	**Version**	**Notes**
2/3/2020	1.0	Project Drivers and Project Issues
2/8/2020	1.1	Completed Functional Requirements
2/8/2020	1.2	Completed Non Functional Requirements
2/9/2020	1.3	Revision 0
4/1/2020	1.4	Edit section 1, 2, and 4
4/4/2020	1.5	Edit Section 3
4/4/2020	1.6	Edit Section 3

This document highlights the Functional and Non-Functional Requirements for Mini-Arcade. This document will be used as a reference for future documents and during software development.

1 Project Drivers

1.1 The Purpose of the Project

The purpose of our project is to provide players with a seamless, satisfying experience when playing the mini games. Thus, we are focusing on making visual, functional, and difficulty enhancements to a pre-existing set of mini games. The mini games that will be recreated are Maze, Pong, and Flappy.

1.2 The Stakeholders

1.2.1 The Client

The clients for the product are Dr. Asghar Bokhari and the SFWRENG 3XA3 Teaching Assistants. As clients, they will administer the development of the product as well as offer assistance and clarification wherever possible. They will also determine the degree to which the product delivers on the requirements.

1.2.2 The Customers

The intended customers of the product are any individuals with the ability to download the product online. ~~This includes a child, up to a senior citizen. The product is created to entertain its users and give them a joyful experience.~~ There is no target demographic for this game, for it is available for anyone who is interested in playing some mini games.

1.2.3 Other Stakeholders

All other stakeholders include all current developers as well as any future developers who wish to further improve upon the open-source project themselves.

1.3 Mandated Constraints

The system is to be built under the following constraints

- The product must run successfully on at least Windows and MacOS.
- The product must be able to be run by running Launcher.py in the command line
- The product must not require any specialized hardware which the common user may not have.

- The product must be free of cost to all users.
- The product must be sufficiently complete by March 28, 2020.
- The product must contain at least ~~one~~ two playable mini-games.
- Development cost of the product must not exceed $0.
- The mini-game(s) must have a visual and/or difficulty improvement to be considered "updated".

1.4 Naming Conventions and Terminology

Mini-Game - a simple game that can be played for a short or long amount of time.

1.5 Relevant Facts and Assumptions

There will be approximately 2000 lines of code for the entire program. 5 main modules: Launcher, Scoreboard, Maze, Pong, Flappy, with 4 sub-modules per game

It is assumed that the pygame and thorpy libraries are sufficient to update the mini games. Also, the mini games will be either visually enhanced or with new difficulty levels; both will be true only if time permits the developers to achieve this. Other assumptions are that the user will have a working computer, understand English and is familiar with the use of a computer.

2 Functional Requirements

2.1 The Scope of the Work and the Product

The Current Situation:

Currently no launcher. To run the games you need to install the code from using Ubuntu, and running each game individually.

- To launch a game you must first install the python freegames library using a command line
- You then must launch each game indvidually that you want to play.

The three games we are planning to modify are Maze, Flappy, and Pong. These games are very simple with minimal functionality and poor performance.

- Maze: a minimal 2-dimensional maze with one preset maze that is generated that can be navigated through single mouse clicks with little-to-no difficulty.
- Flappy: a simplistic game where the user utilizes the mouse-click to prevent their character from colliding with randomly generated objects, and loses when the collide

- Pong: a very simple game that resembles the classic game Pong. It is a two-player game that has contains minimal physics aspect as well as poor graphics and poor performance.

2.1.1 The Context of the Work

- This mini game launcher is developed for the course, SFWRENG 3XA3, professor Dr. Asghar Bokahri and the teaching assistants assigned to our group, Thien Tandinh and Oluwaseum Owojaiye.
- Before development, we will need to create several documents to effectively plan out the project. These documents include: the Test Plan, the Development Plan, the Module Guide, and the Module Interface Specification.
- We will need to re-develop and add additional functionality to the Maze, Pong, and Flappy game
- We will also develop a launcher that contains a scoreboard, and acts as a central hub for the user to access and easily run any of the developed mini-games.

2.1.2 Work Partitioning

Table 2: Work Partitioning Events

Event Number	Event Name	Input	Output
1	Selecting/Launching a Game	Mouse/Keyboard	Selected Game
2	Accessing the Scoreboards	Mouse/Keyboard	Display Scoreboard
3	Playing the Maze	Mouse/Keyboard	Final Score
4	Playing Flappy	Keyboard	Final Score
5	Playing Pong	Mouse/Keyboard	Final Score

Table 3: Work Partitioning Event Summaries

Event Number	Summary
1	Based on the user's input, launch the specified game
2	If the user chooses to access the scoreboard, a list of the high scores will be listed and will have the option to be sorted by each game.
3	The user will select the difficulty of the maze and navigate their object through the maze using the keyboard. Once completed the system will provide the user with their final score which is the time it took to complete the maze.
4	The user will start a new game and use the space-bar to avoid obstacles. The system will provide their final score once they collide with an obstacle based on the time elapsed.
5	The user will use the keyboard to play against an opponent controlled by the system to try to score against them. They will play till either they or their opponent reaches a desired score. The system will then display the final score and record their score.

2.1.3 Individual Product Use Cases

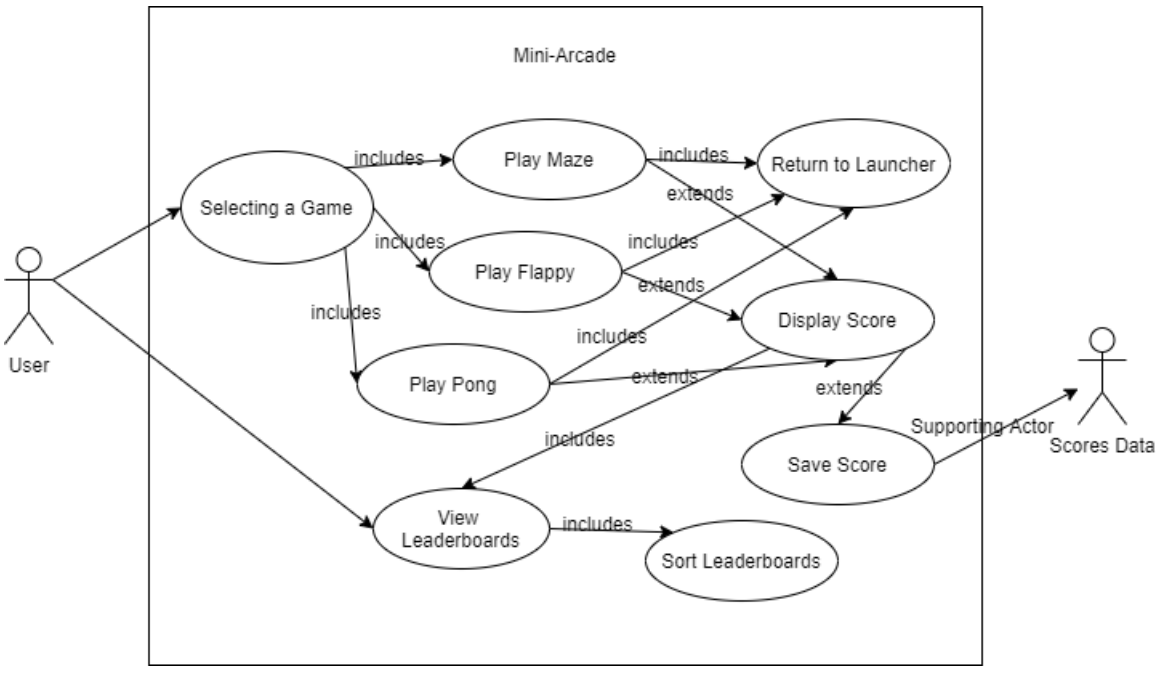

Figure 1: System Use Case Diagram

2.2 Functional Requirements

BE1. The user wants to play a game

- FR1. The system shall provide the user with several different mini-game options
- FC1. The launcher will display the button for launching Maze, Flappy, and Pong
- FR2. The system shall launch the desired mini-game in the same window
- FC2. The software will display the mini-game in the same window
- FR3. The system shall provide an option to close the application
- FC3. The software will display the button for closing the application

BE2. The user wants to view their scores

- FR4. The system shall provide the user with the ability to view the scoreboards from the launcher
- FC4. The launcher will display the button for launching scoreboard
- FR5. The system shall provide the user with the ability to sort their scores based on the game
- FC5. The scoreboard will display 3 buttons used to toggle the scoreboard between Maze, Flappy, and Pong
- ~~FR6. The system shall provide the user the ability to view the current game's scoreboards from the game home screen~~
- ~~FC6. In the menu screen of each game, a button that will launch the scoreboard for that game will be displayed.~~

BE3. The user wants to play Maze

- FR7. The system shall provide the user with different difficulty levels
- FC7. The game will have an easy, medium, and hard difficulty as options
- FR8. The system shall record the user's current score based on the time elapsed
- FC8. A time elapsed counter will be presented during the maze execution
- FR9. The system shall randomly generate a maze based on the selected difficulty
- FC9. Each time a difficulty is chosen, a different maze will appear of the selected size
- FR10. The system shall provide a home button to ~~exit the game at any time~~ return to the launcher from the main menu screen

- FC10. Pressing the home button will return to the user to the launcher
- FR11. The system shall display the user's current score and high score once they complete the maze
- FC11. Once the maze is completed, the times will appear at the end screen.
- FR12. The user shall be able to control the objects movement with the keyboard
- FC12. Keys will result in the player's character moving in the desired direction
- FR13. The system shall provide an option to proceed to another maze once completed
- FC13. Button to return to the main menu to start again upon completion
- ~~FR14. The system shall provide an option to return to the launcher~~

BE4. The user wants to play Flappy

- FR15. The system shall provide the user with the option to start a new game
- FR16. The user shall use the space-bar to control their character
- FR17. The system shall randomly generate objects to interfere with the character
- FR18. The system shall increase speed as the time elapsed increases
- FR19. The system shall generate more objects as the time elapsed increases
- FR20. The system shall provide the user with the final time elapsed once the game is over
- ~~FR21. The system shall provide the option to view the score boards for Flappy~~
- FR22. The system shall provide the user the option to play again
- FR23. The system shall provide the user with the option to return to the launcher

BE5. The user wants to play Pong

- FP1. The user shall be able to choose from three difficulty levels
- FP2. The user shall be able to choose the maximum score of the game
- ~~FR24. The system shall provide the user with the option to play alone or with a friend~~
- FR25. The ~~user(s)~~ user shall use the ~~keyboard~~ up and down arrow keys to navigate their paddle
- FR26. The system shall add a point to the side that scores

- ~~FR27. The system shall require a final score to play up to~~
- FR28. The system shall end the game once the final score is reached by either side
- FP3. The system shall allow the user to pause the game during playtime
- FR29. The system shall provide the option to play again
- FR30. The system shall provide the option to return to the main menu or launcher at any time

BE6. The user is done playing

- FR31. The system shall provide an option to quit the current game, and close the application
- FC31. A button that will allow the user to quit the game

3 Non-functional Requirements

3.1 Look and Feel Requirements

NFR1

- DESCRIPTION: The games should run at a suitable, fast, stable frame rate
- RATIONALE: In order for the game's performance feels responsive and the animations feel "smooth", a high frame rate is needed
- FIT CRITERION: The game should run at a consistent 30 fps or more
- PRIORITY: HIGH

NFR2

- DESCRIPTION: The overall theme of our arcade app should be visually appealing
- RATIONALE: The colours of the games should capture the user's attention to keep their interest
- FIT CRITERION: The colours used by our games should abide by the color theory, to ensure important areas of the game stands out. Colours used for the player character should be complementary to the background and the obstacles, in order for the player to quickly find the player model. The score should also be complementary to the background for the player to view the score quickly.
- PRIORITY: HIGH

3.2 Usability and Humanity Requirements

NFR3

- DESCRIPTION: The UI should be non intrusive and easy to use
- RATIONALE: In a fast paced, competitive game the UI should not interfere with the user's ability to play the game as much as possible
- FIT CRITERION: Ensure that 70% or more of the screen is dedicated to the game.
- PRIORITY: HIGH

NFR4

- DESCRIPTION: The game-play should be intuitive and easy to learn. Instructions will be quickly provided in the game manual or alternatively on the menu screen.
- RATIONALE: In order to attract an large, diverse user-base, the games should be very easy to understand with minimal instructions
- FIT CRITERION: The game should use word-minimal tutorials, have easy to understand game-play and appropriate visual effects. Visual effects should be highly visible, and indicate in-game events.
- PRIORITY: MEDIUM

3.3 Performance Requirements

NFR5

- DESCRIPTION: The game should load quickly and effortlessly
- RATIONALE: In order to ensure a pleasant user experience, the app should keep any "non-gaming" time to a minimum
- FIT CRITERION: Ensure that the game is properly optimized to keep loading times under 20 seconds
- PRIORITY: MEDIUM

NFR6

- DESCRIPTION: The game should have minimal input lag
- RATIONALE: To avoid latency/discrepancies between what the user sees and what the user is doing, input lag should be minimized as much as possible
- FIT CRITERION: Ensure that the game is updated within a quarter of a second of user input
- PRIORITY: HIGH

3.4 Operational and Environmental Requirements

NFR7

- DESCRIPTION: The arcade app should be easily accessible to the end user on any OS and machine
- RATIONALE: Many end users will have a wide variety of computers with different processors and operating systems
- FIT CRITERION: The arcade app needs to be able to run on a wide variety of end user computers, running Windows 7 or Windows 10.
- PRIORITY: HIGH

3.5 Maintainability and Support Requirements

NFR8

- DESCRIPTION: Maintenance should be kept to a minimum
- RATIONALE: Maintenance of the app will be kept to a minimum, in order to not interfere with the user or to split the user-base between different software versions
- FIT CRITERION: The game arcade app should be released with commonly occurring issues and problems resolved
- PRIORITY: LOW

3.6 Security Requirements

NFR9

- DESCRIPTION: Games app should not allow players to modify any in game data, attributes or properties that are not intended to be tampered with
- RATIONALE: Cheating should be prohibited in a competitive setting
- FIT CRITERION: An experienced player should know the result of any actions in the game
- PRIORITY: LOW

3.7 Cultural Requirements

NFR10

- DESCRIPTION: The product will not purposefully be offensive to any religious or ethnic group
- RATIONALE: Being offensive to any racial or religious groups will limit the player base
- FIT CRITERION: The content of the game will not include any material that may be offensive to any religious or ethnic groups
- PRIORITY: HIGH

3.8 Legal Requirements

NFR11

- DESCRIPTION: The product must not break ~~the law~~ any international laws
- RATIONALE: ~~Mini-Arcade must not break the law~~ This ensures the player-base and accessibility in many countries is maximized
- FIT CRITERION: Do not break any laws
- PRIORITY: HIGH

3.9 Health and Safety Requirements

NFR12

- DESCRIPTION: The product must not cause any adverse health effects
- RATIONALE: Playing games on a screen can cause various injuries, such as repetitive strain injuries or eye strain which we should take care to avoid
- FIT CRITERION: ~~We will have popups to alert the user that they should take a break after an extended period of play (1 hour or more)~~ The average length of a game will be kept under ten minutes
- PRIORITY: MEDIUM

4 Project Issues

4.1 Open Issues

The following describes issues that has been raised , but have not yet solved:

1. The current program is still in the form of python scripts, which make it relatively hard for beginners to run the game (due to the need to installing 2 external libraries). Therefore, it would be easier for access if it is compiled into a standalone executable. But this require the use of external libraries and also there will be problems with reading and writing files in standalone executable.

4.2 Off-the-Shelf Solutions

As some of the mini-games are based on commonly spread ideas, there is a decent amount of similar games out there ~~, such as pac-man, flappy bird, pong, etc~~.

4.3 New Problems

Our project provides an all-in-one package to some of the common mini-games, with a launcher providing access to the mini-games without the need of internet connection on computers.

4.4 Tasks

Task	**Timeline**
Launcher Model Implementation and Partial Graphic Library Adaptation	February 13^{th}
Adopt New Graphic Library	February 23^{th}
Launcher Development and Game Remake	March 15^{th}
Testing and Revision	March 29^{th}

Task 1: Basic model for launcher and partially modify the mini-game to adapt with new graphic library (at least modify the main menu).
Task 2: Finish the adaptation to the new graphic library.
Task 3: Finish development of the launcher and remake the mini-games to provide more functionality to each mini-game.
Task 4: Test the launcher and mini-game, fix bugs found during testing and optimizes the mini-games.

4.5 Migration to the New Product

Not applicable.

4.6 Risks

Risks are inherent in the development of any product. The following information highlights possible risks for this project:

- Complexity of Graphic Library
- Over-estimated Project Size
- Difficult Testing
- Bad Programming Practices
- Improper Git Usage

4.7 Costs

There is no monetary cost to this project as all resources are free of charge. The original code this project is based on is a open-source project, and all technologies required in the development of the project such as graphic libraries and programming tools are all free, resulting in no monetary cost.
The labour cost required for this project is relatively high, as each team member will require to dedicate around 5 to 7 hours minimum per week in the development of this project.

4.8 User Documentation and Training

Inside the project, there will be a README.MD file inside the repository which contains the information With each mini-game having different controls and playing method, there will be a help section integrated into each mini-game containing help information.

4.9 Waiting Room

As this project only re-made part of the mini-games in the original open source project, the future plan will be remaking all mini-games in the original open source project and integrate them into the launcher.

4.10 Ideas for Solutions

The following are purposed solution to some of the issues stated in section 4.1:

- Pyinstaller is a library that can compile python script into a standalone executable. The use of library is still pending due to the need of further investigation of the library to determine if it meets all of our need.

4.11 Symbolic Parameters

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