SE 3XA3: Test Plan
Flight Shooting Game

Team 15, Cloud 10
Yijun Chen and cheny161
Zefeng Wang and wangz217
Tianxing Li and lit20

December 5, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
2018-10-26	0.0	Version0 done
2018-12-05	1.0	Version1 done

1 General Information

1.1 Purpose

~~The function of our program project is to run a 2D retro flight-shooting game.~~ The purpose of our project is to implement a 2D retro flight-shooting game.This game will be able to interpret user inputs, display the pixel-style user interface and play sounds. ~~The testing for the software will include the PyUnit as our automated unit testing. For structural testing, our group will implement PyUnit to test collision detection, user input recognition, and path coverage.~~The testing for the software will include the Unittest as our automated testing. For structural testing, our group will implement Unittest to test collision detection, user input recognition, and path coverage. Functional testing will consist of the minimum set of test cases which cover all programming statements. ~~Framing resizing, sound output and graphics displaying will be tested.~~ Mutation testing will also be used to design new test cases and evaluate the quality of the existing software tests.

1.2 Scope

Testing will provide independent information about the quality of ~~our~~ the game, risk of ~~failure to the users~~ software failure such as crashing, getting stuck, and will ~~allow our group~~ assist us in making ~~make~~ sure ~~that the full requirements of our game are being met.~~ the completion of meeting the requirements. It performs root cause analysis which also helps in maintaining efficiency. Before releasing the software, all of our tests must be thoroughly completed and passed.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Definitions**

ACRONYM/ ABBREVIATION	**INTENDED MEANING**
PyUnit	It is an easy way to create unit testing programs and UnitTests with Python
The program	Any individual that utilizes the product after completion
PoC	Proof of Concept
SRS	Software Requirements Specification
GUI	Graphical User Interface
~~apk~~	~~Android Application Package~~
Pyxel	A retro game engine for Python
~~Kivy~~	~~An Android app engine for Python~~
Pyxel	The most important library that the game is relied on.
MIT license	A permissive free software license
Structure Testing	Testing derived from the internal structure of the software.
Functional Testing	Testing derived from a description of how the program functions.
Dynamic Testing	Testing which includes having test cases run during execution.
Static Testing	Testing that does not involve program execution.
Manual Testing	Testing conducted by people.
Automated Testing	Testing that is run automatically by software.

1.4 Overview of Document

The most frequent testing method is manual test because the main output of the software is the graphic and audio information. Unittest implemented through PyUnit is used ~~as the main testing method in our software to test that each unit of our software is performed as designed~~. Structure testing, functioning testing, and mutation testing act as means to arrange testing activities. Since this is an open source redeveloped project, our game is similar to the original Flight Shooting Game. Studying their testing systems will drastically reduce the time needed to build our test cases. This document will present what is going to be tested of the software.

2 Plan

2.1 Software Description

The software is a retro style shooting game ~~operated on~~ implemented by Python. The user can play the game through the basic keyboard operation.

2.2 Test Team

The individuals responsible for testing are Yijun Chen, Tianxing Li, Zefeng Wang.

2.3 Tools Used for Testing

The tool that will be utilized for this project is PyUnit. It will be used to automate the unit testing.

2.4 Testing Schedule

Task	**Team Member**	**Date**
Player status	YC	October 28th 2018
Bullets	TL	November 2nd 2018
Collision	ZW	November 4th 2018
Gift status	YC	November 5th 2018
Sound	TL	November 7st 2018
Graphics	ZW	November 9th 2018
Usability	YC	November 11th 2018
Performance	TL	November 15th 2018

Table 3: Testing Assignemts

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 UserInput

Player Control

- FS-PC-1

  Type: Functional, Dynamic, Manual
  Initial State: ~~A graph of~~ Game is running and player plane stays still at the bottom of the window.
  Input: “W” or “UP” button. (Press once)
  Output: The player plane moves up for certain distance in range of 1/5 of the window vertically.
  How test will be performed: Press “W” or “UP” button once, player plane moves upwards for a short distance described before. If it moves in other direction or stays still, then there is an error.

- ~~FS-PC-2~~

  ~~Type: Functional, Dynamic, Manual~~
  ~~Initial State: A graph of player plane stays still at the top of the window.~~
  ~~Input: “S” or “DOWN” button. (Press once)~~
  ~~Output: The player plane moves down for certain distance in range of 1/5 of the window vertically.~~
  ~~How test will be performed: Press “S” or “DOWN” button once, player plane moves upwards for a short distance described before. If it moves, then there is an error. If it moves in other direction or stays still, then there is an error.~~

- FS-PC-2

  Type: Functional, Dynamic, Manual
  Initial State: A graph of player plane stays still at the bottom of the window.
  Input: “W” or “UP” button. (Hold)
  Output: The player plane keeps moving up until it reach the top of the

screen.
How test will be performed: Hold “W” or “UP” button, player plane keeps moving up until it reaches the top of the window. If it moves in other direction or stays still or it does not stop when hit the top, then there is an error.

- FS-PC-3..4

  Do the similar test for “DOWN” direction with key “S” or “DOWN”.

- FS-PC-5..6

  Do the similar test for “LEFT” direction with key “A” or “LEFT”.

- FS-PC-7..8

  Do the similar test for “RIGHT” direction with key “D” or “RIGHT”.

- FS-PC-9..10

  Do the similar test for “UPLEFT” direction with (key “W” and “A”) or (“UP” and “LEFT”).

- FS-PC-11..12

  Do the similar test for “UPRIGHT” direction with (key “W” and “D”) or (“UP” and “RIGHT”).

- FS-PC-13..14

  Do the similar test for “DOWNLEFT” direction with (key “S” and “A”) or (“DOWN” and “LEFT”).

- FS-PC-15..16

  Do the similar test for “DOWNLEFT” direction with (key “S” and “D”) or (“DOWN” and “RIGHT”).

- FS-PC-17

  Type: Functional, Dynamic, Manual
  Initial State: Game running and play is alive with full “energy” and certain enemy planes on the field.
  Input: “U button. (Press once)
  Output: All the enemy planes should be destroyed instantly.
  How test will be performed: When “energy” is full, player press “U” once and all enemy planes not including the boss alive are destroyed instantly. If any stay alive, there is an error.

Main Control

1. FS-MC-1

   Type: Functional, Dynamic, Manual
   Initial State: The game is on the start page.
   Input: “SPACE” key. (Press once)
   Output: The game enters the game page and the game starts.
   How test will be performed: When it is on the start page, press “SPACE” once, the game starts immediately. If there is any other behaviours, there is an error.

2. FS-MC-2

   Type: Functional, Dynamic, Manual
   Initial State: The game is on the death page.
   Input: “Q” key. (Press once)
   Output: The game quits immediately.
   How test will be performed: When the game is on death page, press “Q” once to quit the game. The windows should ~~vanish~~ be closed.

3. FS-MC-3

   Type: Functional, Dynamic, Manual
   Initial State: The game is on the death page.

Input: “R” key. (Press once)
Output: The game restarts immediately.
How test will be performed: When the game is on death page, press “R” once to restart the game. A new game should start.

3.1.2 Other functional requirement tests

1. FS-R-1
   Type: Functional, Dynamic, Manual
   Initial State: The game on the start page.
   Input: Play one round.
   Output: Modification on high mark.
   How test will be performed: After one round, if the new score is higher than the original ~~mark~~ score, the high ~~mark~~ score is set to the new ~~mark~~ score, else it remains the same. If the game does not behave in this way, there is an error.

2. FS-R-2
   Type: Functional, Dynamic, Manual
   Initial State: The game is running and player is alive.
   Input: None.
   Output: The player plane is shooting bullets automatically.
   How test will be performed: Start a game and the player is shooting bullets automatically.

3. FS-R-3
   Type: Functional, Dynamic, Manual
   Initial State: The game is running and player is alive.
   Input: Some operations to make the player’s plane collide other enemy objects.
   Output: Loss of life point.
   How test will be performed: Let the player’s plane collide with other enemy objects to see if the life point decreases.

4. FS-R-4
   Type: Functional, Dynamic, Manual
   Initial State: The game is running and player plane is alive.
   Input: Some operations making the player plane collide with gift object.
   Output: The player plane gains some benefit.

How test will be performed: If player's plane collides with a "+" sign gift, the life point of the player plane increases. If player's plane collides with a "R" sign gift, the fire mode of the player's bullet becomes double instead of single.

3.2 Tests for Nonfunctional Requirements

Look and feel

NF-L-1

Type: Structural, Static, Manual
Initial State: main page
Input/Condition: press space to start
Output/Result: three kinds of enemy: enemy jet, head, boss
How test will be performed: enemy jet(with two different graphs), head, boss have all be seen by the player.~~ and then the player will be asked whether they can till the relationship between the difficulty level and three types of enemy.~~

Usability

NF-U-1

Type: Structural, Static, Manual
Initial State: main page
Input/Condition: controls and operations are not introduced to the player
Output/Result: player's feedback
How test will be performed: the controls and operations will not be introduced to the players and after they have been playing for a while, ask them for confusion or difficulties they have encountered.

Performance

NF-P-1

Type: Structural, Static, Manual
Initial State: main page
Input/Condition: controls and operations are introduced to the player
Output/Result: player's feedback
How test will be performed: the controls and operations will be introduced to the players and after they have been playing for a while, ask them whether they have felt any latency or slow response speed.

Operational and Environmental

NF-O-1

Type: Structural, Static, Manual
Initial State: main page
Input/Condition: controls and operations are introduced to the player
Output/Result: player's feedback
How test will be performed: the controls and operations will be introduced to the players and after they have been playing for a while, ask them ~~whether they have encountered any thing that would violate their culture or religious.~~what is the level of difficulty of the operations and what kind of the overall experience do they have.

4 Tests for Proof of Concept

As for shooting game program, the proof of concept should be focusing on testing the key question: whether the game is fun. This would request an approximation of ~~80~~ 50% code, ~~5~~ 20% of art work, image, 20% of sound effect to be precise have been done.

4.1 Willingness to Play

1. FC-1

   Type: Functional, Dynamic, Manual
   Initial State: main page
   Input/Condition: controls and operations are introduced to the player
   Output/Result: player's feedback
   How test will be performed: the controls and operations will be introduced to the player and after they have been playing for a while, ask them what they like and dislike about the game.

5 Comparison to Existing Implementation

There are two tests that compare the program to the Existing Implementation of the program. Please refer to:

- test SS-2 in Tests for Nonfunctional Requirements - Usability

- test SS-3 in Tests for Nonfunctional Requirements - Performance

6 Unit Testing Plan

The Python Unittest framework will be used to do unit testing for this project.

6.1 Unit testing of internal functions

In order to create unit tests for internal functions of the program, the functions which can return value(s) in each class will be tested independently while functions that do not return, but just mutate values will also be tested with the help of some other return functions to monitor the mutated values. The classes will be initialized independently with input values for their constructors. Some functions may need extra arguments which will also be included in input values. A series of unit tests can be conducted with expected output and actual output values of functions are given. This project will not need any stubs or drivers to be imported in terms of testing. Coverage metrics will be involved to determine how much of code in program has been covered by the unit test. The test should cover as many cases as possible. However, there are very similar or same functions, such as move(), in different classes and to avoid redundancy, some of them will not be tested. And in this way, our goal coverage will be around 75%.

6.2 Unit testing list

Unit testings are made for each of the following functions.

Table 4: Unit testing overview

Item No.	Function	Input	Output
UF-1	Player.reset(lifepoint)	Integers	
UF-2	SmartBullet.move()		
UF-3	SmartBullet.aim(player)	PlayerObject	
UF-4	SmartBullet.reset(x,y)	Integers	
UF-5	Boss.move()		
UF-6	Boss.show_up()		
UF-7	Boss.check_death(lifepoint, score)	integers	
UF-8	Boss.reset()		
UF-9	Bullet.move()		
UF-10	Bullet.reset()		
UF-11	Enemy.move()		
UF-12	Enemy.reset()		
UF-13	Enemy.check_death()		
UF-14	Enemy.not_collidable()		
UF-15	Gift.move()		

6.3 Unit testing of output files

The only output file will be a text file with a single integer in its content. The expected output value should be strictly equal to actual output value in the unit test.