# SE 3XA3: Test Plan
# **Snake Game Remake**

Team #: 302, Team Name: 404
Student: Shunbo Cui cuis13
Student: Xiangxin Kong kongx9
Student: Shuo Zhang zhans18

April 3, 2020

# Contents



# List of Tables

# List of Figures

Table 1: **Revision History**

| **Date** | **Version** | **Notes** |
|---|---|---|
| Feb 27 | 1.0 | Create the TestPlan file |
| Apr 3 | 2.0 | edit mode test |

# 1 General Information

## 1.1 Purpose

This document is designed to show the detailed test plan for the Snake game. This will include a description of the testing, validation, and verification procedures that will be implemented. All the tests in this document have been created before the final implementation has been completed or testing has actually occurred, so it will be the guide followed during the testing phase of the project.

## 1.2 Scope

The scope of the test plan is to provides a basis for testing the functionality of this improvement of the Snake game. The objective of the Test Plan is to prove all the functional and non functional requirements listed in the SRS document.

## 1.3 Acronyms, Abbreviations, and Symbols

## 1.4 Overview of Document

This Test Plan's main goal is to inform on how Snake game will be tested for correctness in its various objectives and requirements. All tools are stated and their use case is explained. All planned test cases that are used to verify the correctness of the Snake game with regards to its functional and non-functional requirements are also listed.

# 2 Plan

## 2.1 Software Description

Snake game is a recreation of the Java project created by Mohammed Talaat.The player controls a series of dots on a bordered plane. As it move, it leaves a trail behind, resembling a moving snake. The snake has a specific length, so there is a moving tail a fixed number of units away from the head. The length can increase when the head "eat" an item. The player loses when the snake runs into the screen border, a trail or other obstacle, or itself.

## 2.2 Test Team

The test team for this project consists of the following members who are each responsible for writing and executing tests for modules later to be specified:

Shunbo Cui
Xiangxin Kong
Shuo Zhang

## 2.3 Automated Testing Approach

Automated testing will not be used to prove the correctness and validate the snake game project. It has instead been decided to take a strictly manual approach for various reasons including but not limited to: Lack of canvas environment testing framework support. The fact that since it is a video game it is being testing for feel and continuity mostly. As well as user interaction being such an integral part of the project so this should also be represented in how it is tested.

## 2.4 Testing Tools

The project will be tested in Junit framework with eclipse IDE.

## 2.5 Testing Schedule

See Gantt Chart

# 3 System Test Description

## 3.1 Tests for Functional Requirements

### 3.1.1 User Input

1. Test-1

   Type: Functional, Dynamic, Manual

   Initial State: The difficulty is set to normal as default

   Input: The user does not select different difficulty and starts the game

   Output: The snake moves in the normal default speed when the game starts and the items are generated with normal frequency

   How test will be performed: The function that adjust the speed and item generation will run. We will check if the snake covers correct amount of blocks in a fixed time period, and also the new in-game items are randomly generated on time.

2. Test-2

   Type: Functional, Dynamic, Manual

   Initial State: The difficulty is set to normal as default

   Input: The user selects easy difficulty in the title screen and clicks start button

   Output: The snake moves in the slow speed when the game starts and the items are generated with frequency for easy mode

   How test will be performed: The function that adjust the speed and item generation will run. We will check if the snake covers correct amount of blocks in a fixed time period,

and also the new in-game items providing positive effects are randomly generated on time, as well as the trap items removed.

3. Test-3

   Type: Functional, Dynamic, Manual

   Initial State: The difficulty is set to normal as default

   Input: The user selects hard difficulty in the title screen and clicks start button

   Output: The snake moves in the fast speed when the game starts and the items are generated with frequency for hard mode

   How test will be performed: The function that adjust the speed and item generation will run. We will check if the snake covers correct amount of blocks in a fixed time period, and also the amount of in-game items providing positive effects are decreased, as well as the trap items increased.

4. Test-4

   Type: Functional, Dynamic, Manual

   Initial State: The map is set to default

   Input: The user does not select any map in the title screen and clicks start button

   Output: The game will start after loading the default map with no barrier.

   How test will be performed: The function that loads the map will run. We will check if the default map is loaded when the game starts.

5. Test-5

   Type: Functional, Dynamic, Manual

   Initial State: The map is set to default

   Input: The user select a map other than the default one in the title screen and clicks start button

   Output: The game will start after loading the selected map with corresponding terrain and barriers

   How test will be performed: The function that loads the map will run. We will check if the correct terrain and barriers are loaded when the game starts

6. Test-6

   Type: Functional, Dynamic, Manual

   Initial State: Main screen waiting for start

   Input: The user clicks start button

Output: The gaming screen will be shown and a snake with length 2 and default direction to right is generated on random location in the map

How test will be performed: The function that loads the new snake will run. We will check if the snake with correct length and direction are generated when the game starts

7. Test-7

   Type: Functional, Dynamic, Manual

   Initial State: Start state of gaming screen

   Input: The user clicks space key on keyboard

   Output: The snake will start moving to the default direction

   How test will be performed: The function that moves the snake will run. We will check if the snake starts moving with correct direction and speed when space is pressed

8. Test-8

   Type: Functional, Dynamic, Manual

   Initial State: Start state of gaming screen

   Input: The user clicks space key on keyboard

   Output: The starting sound effect will be played

   How test will be performed: The function that plays sound effect will run. We will check if the sound file is output to the audio device successfully

9. Test-9

   Type: Functional, Dynamic, Manual

   Initial State: Start state of gaming screen

   Input: The user clicks space key on keyboard

   Output: A food item will be generated on the random empty location inside the map

   How test will be performed: The function that generates food item will run. We will check if the food item is shown on the random location in the valid area inside the map.

10. Test-10

    Type: Functional, Dynamic, Manual

    Initial State: Gaming state of gaming screen

    Input: The user clicks p key on keyboard

    Output: The game goes to pause state and the pause screen is shown

How test will be performed: The function that pauses the game will run. We will check if the game is paused and the pause screen is shown correctly

11. Test-11

    Type: Functional, Dynamic, Manual

    Initial State: Pause state of gaming screen

    Input: The user clicks p key on keyboard

    Output: The game goes to gaming state again and the pause screen is hided

    How test will be performed: The function that pauses the game will run. We will check if the game is running again and the pause screen is hided successfully

12. Test-12

    Type: Functional, Dynamic, Manual

    Initial State: Gaming state of gaming screen

    Input: The user clicks any direction key on keyboard

    Output: The direction of the snake is fixed to the corresponding direction of pressed key

    How test will be performed: The function that fix the direction will run. We will check if the direction of the snake is changed correctly

13. Test-13

    Type: Functional, Dynamic, Manual

    Initial State: The snake's head is at an empty location

    Input: The snake's head reaches a food item

    Output: The length of the snake is incremented by 1 at the tail

    How test will be performed: The function that grows the snake will run. We will check if the length of the snake is updated correctly when consuming the food

14. Test-14

    Type: Functional, Dynamic, Manual

    Initial State: The snake's head is at an empty location

    Input: The snake's head reaches a food item

    Output: The player's score will be incremented

    How test will be performed: The function that increases the score will run. We will check if the score is updated when consuming the food

15. Test-15

    Type: Functional, Dynamic, Manual

    Initial State: The snake's head is at an empty location

    Input: The snake's head reaches a food item

    Output: The sound effect of scoring will be played

    How test will be performed: The function that plays the sound effect will run. We will check if the sound file is output to the audio device successfully

16. Test-16

    Type: Functional, Dynamic, Manual

    Initial State: The snake's head is at an empty location

    Input: The snake's head reaches a food item

    Output: The food item is removed and a new food item is generated on random empty location

    How test will be performed: The function that generates the food item will run. We will check if the new food item is generated on valid location

17. Test-17

    Type: Functional, Dynamic, Manual

    Initial State: Gaming state of gaming screen

    Input: Fixed time period passes

    Output: The effect item is generated on random empty location

    How test will be performed: The function that generates the effect item will run. We will check if the new effect item is generated on valid location

18. Test-18

    Type: Functional, Dynamic, Manual

    Initial State: Gaming state of gaming screen

    Input: The snake's head reaches an effect item

    Output: The effect item is applied to the snake

    How test will be performed: The function that enables effect will run. We will check if the status of the snake is updated corresponding to the item consumed

19. Test-19

    Type: Functional, Dynamic, Manual

    Initial State: Gaming state of gaming screen

    Input: The snake's head reaches the border or barrier or snake body

    Output: The game over screen is shown

    How test will be performed: The function that shows game over screen will run. We will check if the game is stopped and the game over screen is shown correctly

20. Test-20

    Type: Functional, Dynamic, Manual

    Initial State: Game over screen

    Input: The player clicks restart button

    Output: The starting screen of a new game is shown

    How test will be performed: The function that restarts a game will run. We will check if the game is refreshed and a new game is initialized

21. Test-21

    Type: Functional, Dynamic, Manual

    Initial State: Game over screen

    Input: The player clicks back to title button

    Output: The title screen will be shown

    How test will be performed: The function that shows title screen will run. We will check if the game is stopped and the title screen is shown

22. Test-22

    Type: Functional, Dynamic, Manual

    Initial State: Title screen

    Input: The player clicks enable color blind option button

    Output: The color blind option is enabled

    How test will be performed: The function that enables color blind option will run. We will check if the colors used in the game are adjusted to visible to color-blind people

## 3.2 Tests for Nonfunctional Requirements

### 3.2.1 Usability

1. Test-23

   Type: Structural, Static, Manual

   Initial State: File explorer or command line terminal

   Input: Java command execution or jar file

   Output: The program is opened with Java virtual machine

   How test will be performed: The function that opens the program will be run using command or file on any Java virtual machine. We will check if the program can be successfully opened on Java environment in Linux and Windows system, and the main screen is completely shown.

### 3.2.2 Performance Requirements

1. Test-24

   Type: Structural, Dynamic, Manual

   Initial State: The program is launched and ready to be played

   Input: The player performs operations on the program and ends the game

   Output: The operations get respond from the program under processing time

   How test will be performed: A log of operations will be generated with a module in the program. System clock are used to calculate for the time period needed between receiving the user input and the response from the program.

## 3.3 Traceability Between Test Cases and Requirements

Traceability Between Test Cases and Requirements

| Requirements | Test cases |
|---|---|
| FR3 | Test 1-Test 3 |
| FR4 | Test 4-Test 5 |
| FR6 | Test 6 |
| FR7 | Test 7-Test 8 |
| FR8 | Test 9 |
| FR9 | Test 10 |
| FR10 | Test 11 |
| FR12 | Test 13-15 |
| FR13 | Test 16 |
| FR15 | Test 17 |
| FR16 | Test 18 |
| FR17 | Test 19 |
| FR18 | Test 20-21 |
| FR19 | Test 22 |
| LF1 | Test 23 |
| PR1 | Test 24 |

# 4 Tests for Proof of Concept

The tests for POC will be focused on verifying the functionality of the added feature (like sound and items.) and the new mode.

## 4.1 Area of Testing1

**Test for features**

1. Sound effect

   Type: Functional, Manual, Dynamic

   Initial State: The game is started and the snake is moving in the screen

   Input: The snake interact with the items during the game (hitting the wall, hitting itself, eating food...)

   Output: Different sound effect will be played when snake interacted with different items.

   How test will be performed: Testing by playing the game to find the result that does not match the expected output. Test the audio player method to see whether the method can track to the correct directory of the specific sound effect. Open the sound effect file by clicking it to test whether it is the right sound.

2. item-001

Type: Structural, Dynamic, automated.

Initial State: The normal mode game is started and snake is moving in the screen

Input: The head of snake collide with item-001

Output: the length of snake is cut in half in the next movement. The last-half of array data of snake body is discarded

How test will be performed: turn on item generator to generate item-001. And the tester need to control the snake to collide with item-001. The test will succeed when the arrayList match the expect output arraylist.

3. item-002

   Type: Structural, Dynamic, automated.

   Initial State: The normal mode game is started and snake is moving in the screen and item-002 is generated in the screen

   Input: The head of snake collide with item-002

   Output: The game is terminated.

   How test will be performed: turn on item generator to generate item-002. The tester need to control the snake to collide with item-002. The test will succeed when the **iscollision** method .return true and game terminates.

4. item time

   Type: Functional, Dynamic, automated.

   Initial State: The normal mode game is started and snake is moving in the screen and an item is generated in the screen.

   Input: The head of snake does not collide with item with-in 30 moves.

   Output: item is disappeared.

   How test will be performed: turn on item generator to generate an item. The tester need to control the snake not to collide with item. The test will succeed when **isitem** method return false.

5. item-003

   Type: Functional, Dynamic, automated.

   Initial State: The normal mode game is started and snake is moving in the screen and item-003 is generated in the screen.

   Input: The head of snake does not collide with the item

   Output: The score incremented by 50.

   How test will be performed: turn on item generator to generate item-003. The tester need to control the snake to collide with item. The test will succeed when the score

increment by 50 on the screen and in the variable that stores the score of this game in code.

## 4.2 Area of Testing2

**Test for New Mode**

1. sigh block

   Type: Structural, static, Manual

   Initial State: The game is started and the night mode is selected.

   Input: hit enter to enter the game.

   Output: only the part of map with in 3 steps and special items within 5 steps from snake head are shown in game.

   How test will be performed: check the code too see whether the condition judgment been illustrate properly in code.

2. mode instruction and color

   Type: Functional, dynamic, manual

   Initial State: The game is started and the night mode is selected.

   Input: hit enter to enter the game.

   Output: A description of the mode is shown in screen and background color is changed to black.

   How test will be performed: choose all the combination in main screen with night mode to see whether instruction and black background color show in all the cases.

# 5 Comparison to Existing Implementation

# 6 Unit Testing Plan

## 6.1 Unit testing of internal functions

The whole unit test for the source code will be done be Junit inside the eclipse. Stubs and drivers are not needed in the unit test as the imports should have already been in each class. All possible inputs will be provided in the unit testing to check whether the output and the exception state are matched with expect result. Control-flow testing will be used for the code coverage test. The goal is to cover as more code as possible. At least 60-percent of codes should be covered by the unit test.

## 6.2 Unit testing of output files

A blackbox testing will be set for testing of the output file. In the fist time of playing the normal mode of the snake game, there shall be a file name "Ranking_List.txt" generated in the current directory. If the file already exists, no new file is generated. Inside the file, each line should be in the form "number of rank,score". The number of lines is up to 10 and the score in the file should be right order(rank 1 is the highest score). And it should be updated after each normal mode game(either update the new score or no change)