

# SE 3XA3: Test Report

## Snake 2.o

Team #30, VUA30  
Usman Irfan - irfanm7  
Andy Hameed - hameea1  
Vaibhav Chadha - chadhav

December 4, 2018

# Contents

- 1 Functional Requirements Evaluation 1
- 2 Nonfunctional Requirements Evaluation 1
  - 2.1 Usability . . . . . 1
  - 2.2 Performance . . . . . 1
  - 2.3 etc. . . . . 1
- 3 Comparison to Existing Implementation 1
- 4 Unit Testing 1
- 5 Changes Due to Testing 1
- 6 Automated Testing 2
- 7 Trace to Requirements 2
- 8 Trace to Modules 2
- 9 Code Coverage Metrics 3

# List of Tables

- 1 Revision History . . . . . i

# List of Figures

- 1 Peer Feedback & Comments . . . . . 2

Table 1: **Revision History**

Date	Version	Notes
2018-12-04	1.0	Andy worked on 5 - how Intergrated & System testing helped the process
Date 2	1.1	Notes

This document ...

## **1 Functional Requirements Evaluation**

## **2 Nonfunctional Requirements Evaluation**

### **2.1 Usability**

### **2.2 Performance**

### **2.3 etc.**

## **3 Comparison to Existing Implementation**

This section will not be appropriate for every project.

## **4 Unit Testing**

## **5 Changes Due to Testing**

Through integrated and system testing, which encompassed the majority of the testing done on the software, the user interface as well as bugs and errors in the gameplay were modified to fix erroneous properties of the software. By continuously executing the game, it was easy to estimate changes in object coordinates within the interface. For example, the menu buttons were arranged through trial and error by testing the software continuously until the desired look was achieved. Beyond that, system and integrated testing confirmed that all modules were working correctly and any change in one of the modules did not affect the function of other modules through dependency relations.

Similarly, the gameplay was tested and verified by the developers of the software as well as peers and classmates to ensure proper functioning. Through feedback received in the google survey, errors and modifications were made: For example, one user suggested an excitement element to be added to the game and a maze feature was added to the advanced difficulty

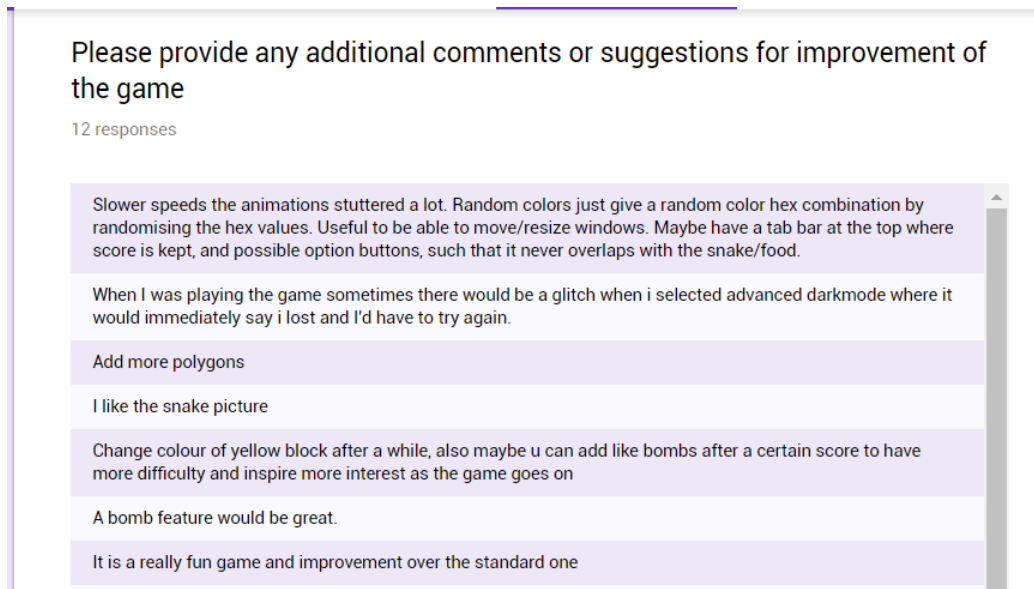


Figure 1: Peer Feedback & Comments

gameplay mode to accomodate for that. As seen in Figure 1, feedback received from peers included both functional and non-functional properties and aided in the software revision process.

## 6 Automated Testing

## 7 Trace to Requirements

## 8 Trace to Modules

Integrated testing can visibly be traced back to the modules created. The main interface uses the Interface module along with GUI module for interface text and buttons. It is connected to the highscore module and theme module through the highscore and difficulty level buttons respectively. It also connected to the help module through the Help button. If any of these buttons is clicked and an error is released, the error can be traced back with ease depending on the button that was clicked prior to the malfunction.

This same pattern is applied in the theme module, between the regular,

dark and random modes. Each button corresponds to color and setup parameters that reflect the chosen theme. If a specific them is not working, it can be traced back in the theme module through the commented blocks of code corresponding to each respective theme.

In the gameplay module, testing can be traced back based on user action and system response. Through the use of commenting it is visible to identify particular functionality such as snake direction change, detection of barrier collisions, snake body collisions, collision with a food block and so on. The traceability of malfunctioning parts within the actual game can be traced back within the gameplay module, which encompasses all functionality under which the snake game operates.

## **9 Code Coverage Metrics**