SE 3XA3: Test Report
Snake 2.o

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

December 5, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
2018-12-04	1.0	Andy worked on 5 - how Intergrated & System testing helped the process
2018-12-05	1.0	Usman worked on Functional Requirements & tracing the requirements to the test
Date 2	1.1	Notes

1 Functional Requirements Evaluation

Through the strategy of dynamic testing, the main testing for the requirements was done. Most of the functional requirements were met, and the errors that were found during the execution were fixed. The project was demonstrated to different peers, and with the help of their review, the project was molded to achieve all the functional requirements described in SRS.
During the testing, we found the food usually appears within the snake's body which violated one our functional requirements, so to resolve the issue we used boundary conditions to limit the appearance of the food within the gameplay screen. After making the snake and food display on the screen, we found a bug that the snake's body is not aligned with the food most of the time, we changed the code, so the game is divided into rows and columns with the block size equal to the size of the snake and food. Splitting the screen in grids made the food to reappear within a grid, and the snake could easily eat it making our further implementation easy.

2 Nonfunctional Requirements Evaluation

2.1 Usability

2.2 Performance

2.3 etc.

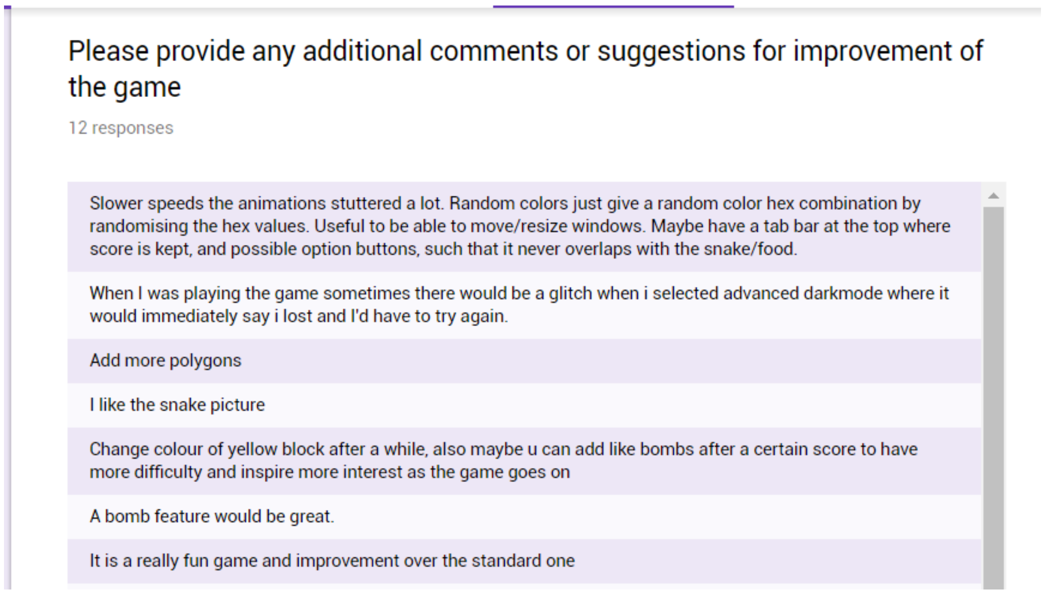
3 Comparison to Existing Implementation

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



Please provide any additional comments or suggestions for improvement of the game

12 responses

Slower speeds the animations stuttered a lot. Random colors just give a random color hex combination by randomising the hex values. Useful to be able to move/resize windows. Maybe have a tab bar at the top where score is kept, and possible option buttons, such that it never overlaps with the snake/food.

When I was playing the game sometimes there would be a glitch when i selected advanced darkmode where it would immediately say i lost and I'd have to try again.

Add more polygons

I like the snake picture

Change colour of yellow block after a while, also maybe u can add like bombs after a certain score to have more difficulty and inspire more interest as the game goes on

A bomb feature would be great.

It is a really fun game and improvement over the standard one

Figure 1: Peer Feedback & Comments

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 acheived. Beyond that, system and intergrated 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 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

The main testing for this program was done through dynamic testing whcih has been discussed in the requirements. The validation for the testing of the product was done by peer review, surveys and self-testing. Boundary cases and groups of test cases were used in dynamic testing to visualize the output and fix it.

7 Trace to Requirements

To meet the functional and non-functional requirements for the program, the requirements were divided into groups and modular were created for each group. A module for the high score part was made in which the requirements for displaying the highest score, displaying the highest score and storing it was done.
Moreover, a theme module was made to meet the requirements regarding the selection of the theme. The user could select two types of themes from the main menu and then the gameplay would have a background of that color, with different themes the color of the snake changes.
To focus on the major requirements, most of the requirements were accomplished in the Gameplay and Interface module. Gameplay module was responsible for all the code in the backend. Requirements controlling the snake, altering the speed for the snake, checking boundary conditions for each level in the game was done in this module.
The interface module is more focused on the frontend, it visualizes the backend program to a user-interface which increases usability and allow the user to communicate with the program easily.ted for each group. A module for the highscore part

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