SE 3XA3: Test Plan
PineSweeper

Team 7; PineApple
Prince Sandhu - sandhps2
Varun Rathore - rathorvs
Vishesh Gulatee - gulatev

December 8, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
Date 22 Oct.	1.0	Initial Document Template
Date 22 Oct.	1.1	General Information Draft
Date 23 Oct.	1.2	Unit Testing Draft
Date 23 Oct.	1.3	System Test Draft
Date 25 Oct.	1.4	Plan Draft
Date 25 Oct.	1.5	Proof of Concept Testing Draft
Date 26 Oct.	1.6	Existing Implementation Draft
Date 29 Oct.	2.0	General Information Completed
Date 29 Oct.	2.1	Unit Testing Completed
Date 29 Oct.	2.2	System Test Completed
Date 30 Oct.	2.3	Plan Completed
Date 30 Oct.	2.4	Proof of Concept Testing Completed
Date 31 Oct.	2.5	Existing Implementation Completed
Date 31 Oct.	2.6	Appendix Completed
Date 31 Oct	3.0	LaTeX document completed
Date 05 Dec.	4.0	Revision 1

1 General Information

1.1 Purpose

The document highlights the testing procedures to be used in the validation and verification phase of the Software Development cycle of team *PineApple*'s project, *PineSweeper*. The purpose of this document is to categorize the test cases identified by members of *PineApple* based on a type (structural, functional, unit etc), which is defined by the properties, scope and desired outcome of the test case. The test cases are conceived based on the requirements highlighted in the SRS document, the proof of concept model, and the team's envision of the finished product. As such, it precedes the Design Specification and Implementation phases of the development cycle. As the project progresses, any and all revisions made will be documented.

1.2 Scope

The *PineSweeper* application, is a redeveloped video game. The application will be designed using the MVC architecture (Model-View-Controller), hence the theoretic scope of testing will cover each of the components of the MVC pattern. Testing the model will encompass testing the algorithms responsible for setting up the game, gameplay and logic of the game. Testing the view and control will encompass testing the visual representation of the game and registering user response respectively, which are key aspects of the game's user interface.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
MVC	Model-View-Controller Architecture
UI	User Interface
GUI	Graphical User Interface

Table 3: **Table of Definitions**

Term	**Definition**
Blackbox Testing	Tests the behaviour of the program, without being concerned by the source code of the program.
Whitebox Testing	Tests the structure of the program
Function Test	Input/Output blackbox test
Structural Test	Code-based whitebox test
Unit Test	A small series of test for individual components of the complete system
Dynamic Test	Code to be executed while testing
Static Test	Code inspection
Manual Test	Hand-written test cases
Automated Test	Test handled by automation frameworks
System Test	Blackbox testing of the integrated finished product
Stress Test	Testing limits of a system

1.4 Overview of Document

The *PineSweeper* project is a re-implementation of an open source imitation of MineSweeper, called *JSweeper*. Based on requirements documented and reference implementation, the team has categorized and test cases based on a type (structural, functional, unit etc), which is defined by the properties, scope and desired outcome of the test case.

2 Plan

2.1 Software Description

PineSweeper is a game that?s main objective is to avoid and identify all the pineapples hidden under the tiles by analyzing numbers which represent the distances to the mines in its proximity. The software is implemented by the Java language in the *Eclipse* IDE.

2.2 Test Team

The test team consists Varun Rathore, Vishesh Gulatee, and Prince Sandhu. A significant allocation of tests are to be conducted by Varun as he holds the title of being the lead tester for the project. Furthermore, *PineApple* has decided that each member of the team is not to test any code that he has written because one is less likely to discover bugs in their program if they test their own code.

2.3 Automated Testing Approach

The automated testing approach that is used is the automated JUNIT framework as this approach allows us to break the testing units into smaller parts.

2.4 Testing Tools

PineApple has collectively decided that the JUNIT framework will be utilized in order to analyze the code for testing purposes.

2.5 Testing Schedule

PineApple's Gantt Chart

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Graphical User Interface Testing

1. FR-GUI-Test-1

   Name: Select from one of three game difficulties: beginner, intermediate, and advanced.

   Type: Manual, Functional, Dynamic

   Initial State: The main menu or when the game is in progress (timer is running).

   Input: The beginner difficulty level is selected.

   Output: New game with beginner difficulty and corresponding dimensions BDIMENSIONS and corresponding number of mines BMINES.

   How test will be performed: Manually using the graphical user interface, the user is to left-click on the button using the in-game menu which specifies the beginner difficulty level, with the correct output being the corresponding board dimensions with the corresponding number of mines.

2. FR-GUI-Test-2

   Name: Select from one of three game difficulties: beginner, intermediate, and advanced.

   Type: Manual, Functional, Dynamic

   Initial State: The main menu or when the game is in progress (timer is running).

   Input: The intermediate difficulty level is selected.

Output: New game with intermediate difficulty and corresponding dimensions IDIMENSIONS and corresponding number of mines IMINES.

How test will be performed: Manually using the graphical user interface, the user is to left-click on the button using the in-game menu which specifies the intermediate difficulty level, with the correct output being the corresponding board dimensions with the corresponding number of mines.

3. FR-GUI-Test-3

   Name: Select from one of three game difficulties: beginner, intermediate, and advanced.

   Type: Manual, Functional, Dynamic

   Initial State: The main menu or when the game is in progress (timer is running).

   Input: The advanced difficulty level is selected.

   Output: New game with advanced difficulty and corresponding dimensions ADIMENSIONS and corresponding number of mines AMINES.

   How test will be performed: Manually using the graphical user interface, the user is to left-click on the button using the in-game menu which specifies the advanced difficulty level, with the correct output being the corresponding board dimensions with the corresponding number of mines.

4. FR-GUI-Test-4

   Name: A new game, with the same difficulty as the current game, must be started if the new game/reset icon is selected. Type: Manual, Functional, Dynamic

   Initial State: The game is in progress (timer is running) or the game has terminated (a hidden mine has been clicked upon).

   Input: New game/reset icon is selected.

   Output: New game is started, with the same difficulty level as the game that is presently running. The mines shall be in different locations than the previous.How test will be performed: Manually using the graphical user interface, the user is to left-click on the new game/reset icon, with the correct output being a new game with the same dimensions as the game that was being played (or completed) previously.

5. FR-GUI-Test-5

   Name: Contents of a covered cell must be displayed when clicked upon.

   Type: Manual, Functional, Dynamic

   Initial State: The game is in progress (timer is running).

   Input: Tile/Button (which contains either: a mine, a number, or a blank) is clicked upon in the PineSweeper Grid.

   Output: Tile/Button disappears and reveals one of the three aforementioned elements hidden beneath it.

   How test will be performed: Manually using the graphical user interface, the user is to left-click on one of the numerous tiles/buttons that lie within the PineSweeper grid, with the tile/button disappearing once clicked.

6. FR-GUI-Test-6

   Name: If a blank cell is selected, the product shall display all eight bordering cells; the Ripple Effect occurs for each neighbouring blank cell

   Type: Manual, Functional, Dynamic

   Initial State: The game is in progress (timer is running).

   Input: Hidden blank cell is clicked upon.

   Output: All eight bordering cells are displayed and the same process follows, recursively, for each of the neighbouring blank cells.

   How test will be performed: Manually using the graphical user interface, the user is to left-click on a tile which uncovers a blank cell. Consequently, all eight of the cell?s borders will be displayed, and the process continues recursively, until either a number, or the edge of the grid has been encountered.

7. FR-GUI-Test-7

   Name: Covered cells must be able to be flagged by the user.

   Type: Manual, Functional, Dynamic

   Initial State: The game is in progress (timer is running).

   Input: Right mouse click on a covered cell.

   Output: The same cell remains covered, marked with a flag icon.

   How test will be performed: Manually using the graphical user interface, the user is to right-click on a tile, with the output being the same cell marked with a flag icon.

8. FR-GUI-Test-8

   Name: User prompted, the game can be paused and resumed. Type: Manual, Functional, Dynamic

   Initial State: The game is in progress (timer is running).

   Input: User clicks on anything outside the game window.

   Output: The in-game timer pauses.

   How test will be performed:The user is to left-click or right-click on anything outside the game window and consequently, the in-game timer pauses.

9. FR-GUI-Test-9

   Name: User prompted, the game can be paused and resumed.

   Type: Manual, Functional, Dynamic

   Initial State: The game is in progress, but paused (timer is not running).

   Input: User clicks within the game window.

   Output: The in-game timer resumes.

   How test will be performed: The user is to left-click or right-click on anything within the game window and consequently, the in-game timer resumes.

3.1.2 Game Structure Testing

1. FR-GS-Test-1

   Name: The game must terminate if a hidden pineapple is selected.

   Type: Structural, Automated, Dynamic

   Initial State: The game is in progress (timer is running).

   Input: Hidden pineapple is clicked upon.

   Output: Game terminates, the timer stops, and the user is unable to click on any other tiles. The locations of all of the hidden mines are revealed. The user is only able to reset the game, change the difficulty using the in-game menu, or close the application.

   How test will be performed: User clicks on a tile/button which contains a pineapple. Immediately after, check that any additional actions cannot be performed with the same game.

2. FR-GS-Test-2

   Name: The game terminates if all cells that do not contain pineapples are uncovered.

   Type: Structural, Automated, Dynamic

   Initial State: The game is in progress (timer is running) with all but one tile containing a hidden pineapple.

   Input: User clicks on the one remaining tile which does not contain a pineapple.

   Output: The game terminates (timer stops). The user must only be able to reset the game, change the difficulty using the in-game menu, or close the application.

How test will be performed: User clicks on a tile/button which does not contain a pineapple. Immediately after, the system checks that any additional actions cannot be performed with the same game.

3. FR-GS-Test-3

   Name: The game remains in progress if a tile uncovers a blank or a number. Type: Structural, Automated, Dynamic

   Initial State: The game has just commenced (timer is running).

   Input: User clicks on a tile which does not cover a hidden pineapple.

   Output: The tile which is clicked on disappears and the game is still in progress (timer running).

   How test will be performed: User clicks on a tile/button which does not contain a pineapple. The system verifies that a hidden pineapple has not been uncovered and hence, the game remains in its current state.

3.2 Tests for Nonfunctional Requirements

3.2.1 Usability Testing

1. NFR-Use-Test-1

   Title: The game must run under Windows, MacOS, and Linux. Type: Manual, Structural, Static

   Initial State: Game has not yet been executed.

   Input/Condition: Game has been launched.

   Output/Result: The game functions as expected.

   How test will be performed: The program is executed on the specified operating system and shall be checked for the expected behaviour.

3.2.2 Performance Testing

1. NFR-Perf-Test-1

   Title: The game responds immediately, by human perception, to user input. Type: Manual, Structural, Dynamic

   Initial State: The game is in progress (timer is running).

   Input: Click on an in-game button which produces a response from the game.

   Output: The game responds correctly to the user input in under RESPONSETIME.

   How test will be performed: The user clicks an option which produces a response from the PineSweeper game.

3.2.3 Security Testing

1. NFR-Sec-Test-1

   Title: The game does not respond to invalid inputs. Type: Manual, Structural, Dynamic

   Initial State: The game is in progress (timer is running).

   Input: Give invalid mouse inputs or keyboard inputs to the PineSweeper Game. Output: The game state does not change.

   How test will be performed: Attempt to click on an uncovered grid tile (which displays either a number or a blank), or give keyboard inputs to the game.

4 Tests for Proof of Concept

4.1 Game Testing

1. PC-Game -Test1
   Title: The game displays numbers that indicate the number of mines bordering the tile.

   Type: Functional, Dynamic, Manual

   Initial State: The game is launched with default settings.

   Input: Left-click using cursor.

   Output: Reveal item hidden beneath.

   How test will be performed: The user will left-click on a tile, which reveals a number that corresponds to and correctly indicates the number of nearby mines.

2. PC-Game -Test2
   Please refer to FR-GS-Test 1.

3. PC-Game -Test3
   Please refer to FR-GS-Test 2.

4. PC-Game -Test4
   Please refer to FR-GS-Test 3.

4.2 GUI Testing

1. PC-GUI-Test1
   Title: The application can be quit at any time. Type: Functional, Dynamic, Manual

   Initial State: The game is launched with default settings.

   Input: Left-click the close window button.

   Output: Game window closes.

   How test will be performed: Manually left-click on the close window button with the cursor and examine that the window closes.

2. PC-GUI-Test2
   Title: Clicking on the check box (designed for testing purposes) uncovers all tiles on the *PineSweeper* grid.

   Type: Functional, Dynamic, Manual

   Initial State: The game is launched with default settings.

   Input: Left-click the check box.

   Output: All tiles are removed.

   How test will be performed: Manually left-click on the check box button with the cursor, with the output being all tiles removed from the *PineSweeper* grid.

3. PC-GUI-Test3
   Title: Cells that contain a pineapple must display the number "-1".
   Type: Functional, Dynamic, Manual

   Initial State: The game is launched with default settings.

   Input: Left-cursor click on a tile which is supposed to contain a hidden pineapple.

   Output: Reveal "-1", which represents the pineapple.

   How test will be performed: Manually left-click on one of the tiles, with the output being a "-1", indicating a hidden pineapple.

5 Comparison to Existing Implementation

1. EI-Test1
   Please refer to FRS-GS-Test 1.

2. EI-Test2
   Please refer to FRS-GS-Test 2.

3. EI-Test3
   Please refer to FRS-GS-Test 3.

4. EI-Test4
   Title: The existing implementation should have different sizes (small, medium, and large) for its GUI.

   Type: Functional, Dynamic, Manual

   Initial State: The game is launched with default settings.

   Input: Click on the small, medium, and large sizes using the in-game menu.

   Output: Game GUI size changes, corresponding to the dimensions clicked upon.

   How test will be performed: Using the graphical user interface, the user is to left-click upon the various grid sizes provided in the in-game menu, with the corresponding output being the change in dimensions of the game board.

6 Unit Testing Plan

PineSweeper has collectively decided to use the automated JUNIT framework. Due to the face that the application follows MVC architecture, *PineApple* is able to break down and segregate classes based on which component of the architecture they belong to.

6.1 Unit testing of internal functions

Most of the internal testing will be comprised of testing the classes that the Model comprises of which, where in the methods and data structures in the class are manipulated by methods of classes that comprise Controller. The model classes include helper class that set up components of the *PineSweeper* grid and a main class that represents the application?s gameplay logic. The main model class comprises of methods, such that each method represents a different game response to a user input. This user input is facilitated into the game via the controller classes. Hence unit test cases will be designed to invoke the different game responses to user input. To that end test cases will not only contain proper inputs, expected by the user, but also inputs that raise exception. No external libraries need to be imported.

6.2 Unit testing of output files

While the Controller classes are responsible for deriving user inputs, the View classes are responsible for creating desired output. The classes used in View to create the application's GUI, are able to provide users with a sophisticated presentation of the game?s response (computed in the model classes) to the user input. As such the unit test cases to test the view classes will be modified versions of the test cases used in testing of internal methods. Such test cases shy away from testing the translation of code to output, instead focusing on testing the end result. Hence these test cases follow the black box model of testing. The test cases will focus on evaluating the desired graphical output and graphical preferences of the user. In addition, since the game also allows users to change visual themes (from the ones provided), test cases will be created so as to test if user preferences can be modified and saved.

7 Appendix

7.1 Symbolic Parameters

Table 4: **Table of Definitions**

Term	**Definition**
RESPONSETIME	0.25 seconds
BDIMENSIONS	9 Tiles x 9 Tiles
BMINES	10 Mines
IDIMENSIONS	16 Tiles x 16 Tiles
IMINES	40 Mines
ADIMENSIONS	24 Tiles x 24 Tiles
AMINES	99 Mines

7.2 Usability Survey Questions

Sample Survey to be handed out to test users.

User Survey

Please complete the following survey using a scale from 1-10, with 10 being the best, as you play through the game.

Entertainment:

User Friendliness:

Game Difficulty:

Overall Appearance:

Additional Comments: