Software Testing Report

Group 26 - Spice Traders (prior team 22)

James McNair
Dan Wade
Alice Cui
Charlie Crosley
Robert Murphy
Marc Perales Salomo

a) Testing summary

For automated testing as a group we decided that using JUnit was the best approach, this is due to the fact that it is the most commonly used unit testing library for Java and will most likely be the most reliable and have the most documentation available online.

Along with JUnit we decided to use Mockito, this library will be used to mock portions of both LibGDX and portions of our game so each component can be isolated and tested individually.

To structure our testing, we agreed to dedicate each testing class to a specific function, for example a testing class will perform component tests on a specific component or will perform an integration test to ensure components are working well together.

We intend to use automated testing for the following test types:

- Unit tests
- Component tests
- Integration tests
- System tests

With the higher bullet points having more tests as the more broad the test, the longer it takes to write and ensure the credibility of the tests.

Along with automated testing, we are going to use manual testing for everything that cannot be automatically tested. This will involve manually going through and testing certain features of the product, for example usability testing and ensuring the product runs well on different platforms.

We intend to use both white and black box testing to ensure that the product works as an entire system, and that each unit of the system operates individually of each other.

- Our testing classes can be found here: https://github.com/booksaw/spice_traders/tree/master/core/src/test/java/com/mygdx/pirategame
- The page of our website with further information about testing can be found here: https://booksaw.github.io/spice_traders/testing/

b) Testing report

Early on we realised that it would be impossible to cover our entire game with testing due to both timeframe and many LibGDX methods throwing errors when run in headless mode. This led us to decide to target the important modules within the codebase for testing, and focus on thoroughly testing critical components of the game. The important modules were narrowed down to the sections of the project that would be most noticeable if something went wrong with them. This lead us to focus our testing on the following modules:

Pathfinding

This is as any errors with pathfinding could cause large amounts of lag in game

- In this module there is 98% code coverage

Powerups

An integral part of combat is collecting power ups so it is important they work as expected

- In this module there is 87% code coverage

Map

If there are any issues with the map it will result in the game being unplayable as players would not be able to navigate around the world.

- In this module there is 64% code coverage
- The missing 36% is LibGDX API calls which cannot be tested using JUnit.

The testing module of our game contains 136 tests all of which are passing both locally and on GitHub Actions (see Continuous Integration for details). These tests provide a code coverage of 44% with the remainder of our game being either non-critical components, API calls which cannot be tested or rendering code for example the main menu. Additional coverage is achieved by performing a series of manual tests which are described below so the entire game is tested by at least a single test.
To ensure that we tested all functional and non-functional requirements we created a traceability matrix to track our progress. This table is too large to put in this document so it can be found here:
https://booksaw.github.io/spice_traders/testing/Tracability_Matrix/index.html. This document helped us track where tests were missing so we could focus our efforts on the missing areas. This led us to reach a 97% requirements coverage by unit tests and manual tests combined where the missing requirement was skipped as it is known to not be implemented.

Along with unit testing, we performed a series of manual tests to ensure that any requirements that cannot be tested by unit tests are still tested. The table showing evidence of these tests can be found on the website as it is too large to fit on this document. https://booksaw.github.io/spice_traders/testing/**.** However a stripped down version is listed below containing all manual tests and their result. But this table does not contain screenshots of evidence.

TestID	Description	Requirement	Passed
MT_1	Give the game to a new player and test that they are able to complete the game on all 3 difficulties, starting with the easiest.	NFR_OPERATABILITY - The game should be easy to understand.	Yes
MT_2	Show each menu to an native english speaker and ensure all prompts and buttons are understood without further context of the game.	NFR_USABILITY - All on-screen information is comprehensible and in plain English.	Yes
MT_3	Apply a colour blind filter to the game using an external program and ensure the game can be completed from start to finish.	NFR_ACCESSIBILITY - The system should be operable by players with impaired vision.	Yes
MT_4	Install the game on a new computer and test that it is able to run without errors	NFR_INTEGRATABILITY - The system can draw upon a list of predetermined assets.	Yes
MT_5	Run the game on both windows and mac computers and test the game can be completed on both.	NFR_RESILIENCE - Running the game on different hardware that meets the minimum requirements should not result in errors.	Yes
MT_6	Run the game on a University computer and ensure the game can be completed without any errors, crashes or noticeable lag.	NFR_RESILIENCE - Running the game on different hardware that meets the minimum requirements should not result in errors.	Yes
MT_7	Run the game 10 separate times, each time inspecting the map by moving around it to check if there were any overlapping tiles.	NFR_PRECISION_CONSTRAINT - Entities shall not occupy the same position on the map.	No

MT_8	Run the game, click new game and ensure all difficulty options are selectable and the game loads the correct difficulty level.	FR_DIFFICULTY - Users should be able to select from easy, medium and hard.	Y
MT_9	Multiple times during gameplay save the game and load the save, ensuring that the game is in the same state as before the load.	FR_SAVE - A save file should be generated as a .xml file.	Yes
MT_10	Test that when launching the game the user is presented with the main menu, and ensure that the game can be started from the main menu by pressing “new game”	FR_MAINMENU_START - Main menu allows player to start the game	Yes
MT_11	Explore the default level paying attention to the health bar, coins and points and ensure they all respond when coins are collected, ships are defeated and damage is taken.	FR_UI_STATISTICS - UI displays health, coins, experience, etc.	Yes
MT_12	Press the escape key and the pause button during gameplay and ensure the pause menu is displayed.	FR_PAUSE_HALT - Pause menu stops gameticks (pauses the game)	Yes
MT_13	Resume the game by pressing the escape key after pausing it and test that the game resumes from the state it was in before the game was paused.	FR_PAUSE_RESUME - Leaving pause menu resumes game from when it was paused	Yes
MT_14	Use the W, A, S and D keys to move the player's ship up, left, down and right respectively and test if the ship moves in the expected direction.	FR_PLAYER_MOVEMENT - Movement is done using WASD or the arrow keys	Yes
MT_15	In the options menu toggle the music and sound effects and ensure they toggle appropriately in game.	FR_UI_SOUNDTOGGLE - Music and sound effects should be toggleable.	Yes
MT_16	From the main menu, press the exit button and test that the game closes.	FR_MAINMENU_EXIT - Main menu allows player to exit/close the game	Yes

In manual testing the test *MT_7* failed as two power ups were found to have spawned at the same location, for this test to not fail additional checks need to be implemented to stop powerups spawning on top of each other. Along with this we did not test the requirement NFR_TIMING as it is not implemented (see implementation deliverable for details).