Module: ENG1/ASSESSMENT2

Title: Software Testing Report

1. Charles Stubbs
2. Annabelle Partis
3. Kieran Ashton
4. Yu Li
5. George Tassou
6. Alex Shore

a)

The testing process is essential to ensure that Auber is bug free and working as we (the developers) intended. Our testing approach for Auber mainly consists of automated tests written within the project file in a test folder. Due to time constraints, we agreed as a team that manual testing was not as efficient as unit testing and decided to focus predominantly on unit tests with a few manual tests for code instances that were hard to tests via other means. We are focusing our testing around pre-existing frameworks that would work with our implementation would be a more efficient way of finding bugs within the game.

In order to test our Auber code, we decided to use Unit Testing as our main method, this incorporates testing frameworks such as Mockito and JUnit to implement and execute the testing code in a developer friendly manner, allowing us to write it in a limited amount of time and ensure the tests are working.

The main advantages of using Unit Testing for our project is the fact that it is fast to execute as well as easy to write as well as being easy to control and maintain, this is particularly useful in our project as the time constraints are tight and we must meet deadlines to keep the stakeholders satisfied.

To enable us to run the tests successfully we have used a skeleton from Tom Grill: https://github.com/TomGrill/gdx-testing . It is licensed “under the Apache 2 License” - https://github.com/TomGrill/gdx-testing/blob/master/README.md - meaning we are allowed to use it in most instances without the risk of infringement.

The use of testing frameworks enable us to write the test cases in a manner that can be easily executed, due to Auber being created using a Java-based framework in LibGDX, the JUnit framework was the obvious choice. JUnit is a very readable framework that is easy to interpret, this helps when it comes to working in a group as one member can write the code and it can be interpreted easily by the other members of the group.

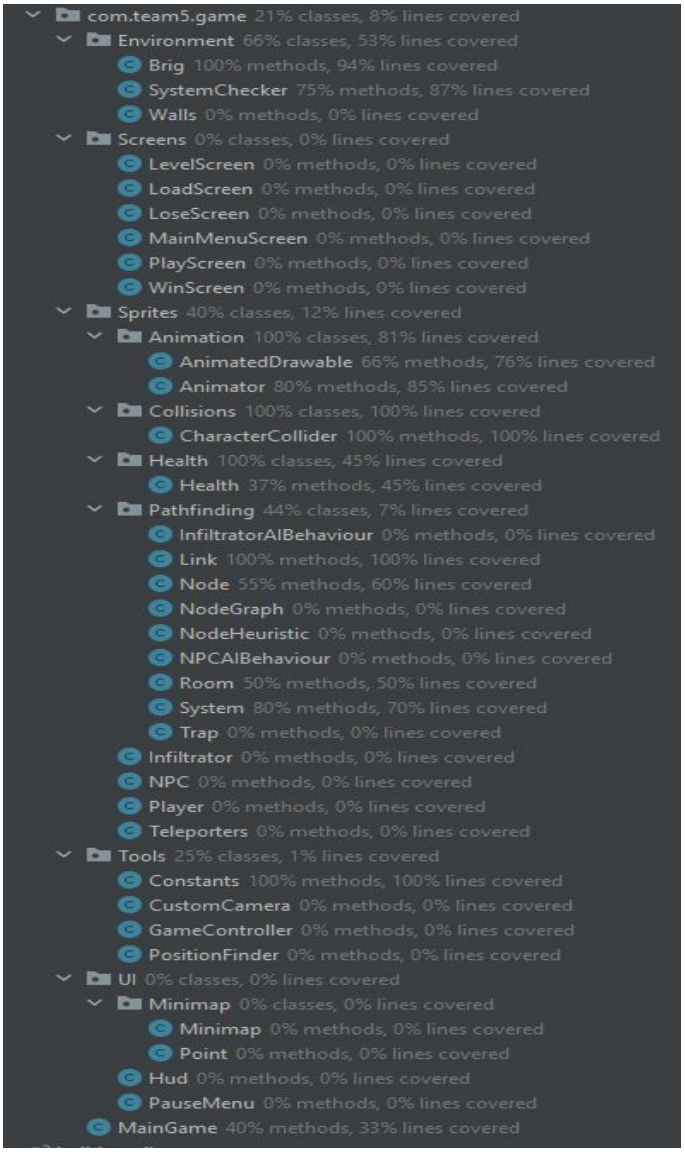
As well as using Unit testing in conjunction with the JUnit framework, we have also decided to use Mocking. Mocking involves creating a mock environment to test isolated portions of the code without the need to run the entire system for each test. We have implemented Mocking using the Mockito framework. Due to our Auber implementation having a large variety of classes, mocking is applicable as we can isolate specific classes and run the tests in isolation from the rest of the code, allowing for tests to be run faster and more efficiently and therefore speeding up the testing process significantly.

Finally, we also conducted manual scenario tests. These involved having a goal, for example teleporting to a particular room and checking the outcome with our hypothesis to ensure it was accurate and correct compared to the requirements and brief we have been given. Whilst we believe unit tests are the most efficient use of our time, we understand that the scenarios provide user perspective to the testing-driven development of the project. So all systems that would not be testable through normal means have attempted to be tested using manual testing.

b)

Automated Testing:

For the automated testing of our Auber project, we implemented JUnit code as well as Mockito, evidence of this can be found below:



Element	Class, %	Method, %	Line, %
com.team5.game	21% (12/56)	14% (38/269)	8% (132/1606)

The screenshots above show the percentage of the code covered by the automated testing process.

Furthermore, the table detailing all automated tests that have been done can be found on the website (https://stubbs774.github.io/runtimeerrors-two/website/testing/testingTable.html).

Manual Testing:

During the testing process, we realised some tests would not be possible using automation by writing code, because of this we have decided to also use Manual testing to increase the test coverage and minimise the chance of bugs occuring within the game. All tests were done by playing the game with the sound muted apart from the mute button test, evidence in provided in the video links found in the table:

testID	Test	Pass /Fail	Evidence	Tester
063	Clicking on an NPC causes loss of health.	Pass	https://drive.google.com/file/d/1idgNwpxPnFk12npOJYE1h75ZQHuZtDse/view?usp=sharing	Charlie Stubbs
064	Traps work as intended.	Pass	https://drive.google.com/file/d/16353mXafdum-OwiC69kma3NOwVTQ4m9w/view?usp=sharing	Annabelle Partis
065	Check all NPCs have spawned.	Pass	https://drive.google.com/file/d/1idgNwpxPnFk12npOJYE1h75ZQHuZtDse/view?usp=sharing	Charlie Stubbs
066	Mouse inputs work as intended on the menu.	Pass	https://drive.google.com/file/d/1yf8xheOP3mai9VvZRixDZRw0xNFp460D/view?usp=sharing	George Tassou
067	Players cannot walk through walls.	Pass	https://drive.google.com/file/d/1seJKqukH29FVIdrHtkjybGy6_2kMe2R2/view?usp=sharing	Yu Li
068	Menu screen loads as intended.	Pass	https://drive.google.com/file/d/1yf8xheOP3mai9VvZRixDZRw0xNFp460D/view?usp=sharing	George Tassou
069	Level screen loads as intended.	Pass	https://drive.google.com/file/d/1yf8xheOP3mai9VvZRixDZRw0xNFp460D/view?usp=sharing	George Tassou
070	Play screen loads as intended.	Pass	https://drive.google.com/file/d/1yf8xheOP3mai9VvZRixDZRw0xNFp460D/view?usp=sharing	George Tassou
071	Check that the correct animation exists (e.g. player spawns in)	Pass	https://drive.google.com/file/d/1seJKqukH29FVIdrHtkjybGy6_2kMe2R2/view?usp=sharing	Yu Li
072	Check that the teleport map works.	Pass	https://drive.google.com/file/d/1K6yADtpMP_Zu1chNXJ-WtR8FwQVLrbNq/view?usp=sharing	Charlie Stubbs

073	Check that the teleporter works.	Pass	https://drive.google.com/file/d/1K6yADtpMP_Zu1chNXJ-WtR8FwQVLrbNq/view?usp=sharing	Charlie Stubbs
074	Test that the sprint ability works as intended.	Pass	https://drive.google.com/file/d/1R9MHU1jR3YCGkQ6aerCqrqUGADdhpETb/view?usp=sharing	Annabelle Partis
075	Test that the mute button stops all sound.	Pass	https://drive.google.com/file/d/1Hj7-aINXj22qaEWiA9BdnrlFJi9Q_Z0B/view?usp=sharing	Kieran Ashton
076	Tests that the wrist teleport ability works as intended.	Pass	https://drive.google.com/file/d/1BIMgpfniEtAFi_1GCBCczFJ1klwd79xw/view?usp=sharing	Kieran Ashton
077	Tests the enemy slowdown ability.	Pass	https://drive.google.com/file/d/14ZUNWmpjYexNFOQ2649gCGlLKoZFZCCX/view?usp=sharing	Annabelle Partis

As evidenced in the test tables as well as the screenshots from our coding terminal, the code and the game passed every test that we have either implemented or done manually. This shows the code is of a high quality and provides a large amount of clarity to us that the game does not have any obvious bugs while playing.