Team iPatch

Assessment 2: Software Testing Report

Christian Pardillo Laursen
Filip Makosza
Joseph Leigh
Mingxuan Weng
Oliver Relph

Testing methods and approaches

Throughout the implementation of the testing project, we choose to test the code by using mostly the black box testing and some of the elements of white box testing since unit testing, a subset of white box testing, is not very helpful in game development, where creating test cases is very laborious as events are triggered in all sorts of ways and in many cases handled by the engine, outside of our control. We ran the test on each feature and the majority of class outcomes.

Black box testing

Black box testing is a test that examines the functionality of the code without considering the internal structure. Black box testing can be done from the user's point, and we can simulate the action the user may execute in the game. We can test the function by simply playing the game and check whether the function is implemented or not. After the feature is implemented, a series of black box testing is needed to check if the feature is work as we designed. And the test can imitate the environment observed by the player.

Black box testing is also very useful when the person who is running the test is not a person who wrote the code. Since each team member is playing different roles in the team, not all of us contribute to the internal code, so the other members can run the black box test effectively. But the disadvantage of black box test is obvious which is that we can not accurately imitate every single step of the player so there may be some problems that we can't discover. To deal with this issue, we define the test precisely but may still miss some obscure scenarios.

White box testing

We use the white box testing to test the code structure of the game. It can be done in an early stage without waiting for the GUI to be available. The tester of white box testing, usually a developer who did not write the code, would walk through the code and give feedback to their peer on what they found to be an issue. This was done fairly informally, where the main goal was to prevent architectural issues that might show up in the later stages of development. We would also use it to make sure any old features would not be significantly impacted by new features by looking at characteristics such as the orthogonality of the codebase. This testing would also be employed when the black box testing produced failures if the developer in care of that part of the code was unable to correct the mistake. This testing improves the quality of the code by constantly ensuring that major issues did not pop up before they grew to an unmanageable size.

Testing Analysis

Black box testing

Our black box testing attempts to cover all main features of the game. The main approach is doing functional testing based on the requirements we made to test that specific functions are working well.

Result

Our tests were geared towards testing the deliverables for assessment 2 so by the end we managed to have our program succeed in 9 out of the 10 devised consistently. The one test that failed was the one that tested the pause function, as we failed to implement it for the current version. The tests, then, did not test the whole of the requirements but only the ones required for assessment 2, which should be improved in future versions to keep track of every requirement. Indeed, our codebase still has a few shortcomings despite it succeeding in 9/10 tests. Our tests attempt to imitate what a player would do while playing the game. For example, in test 9, we assumed that the player would press the restore health button once, which would return their health to maximum. In addition But after this they may still press the restore button, which will still make them spend gold but receive nothing, if they are not aware that their healthpool is already full since there is no method to prevent this. So the tests still have a lot of limitations. Later an actual player test may take place to solve this problem.

White box testing

We planned to use unit tests test before we started to implement the function but they were very difficult to implement. So we decided to instead implement these tests as periodic code inspections carried out by other developers to test whether there were any oversights in the programming. This was mostly implemented as inspections that looked at only one artifact to attempt to find issues within it. This was time consuming, but helped avoid issues in architecture that could cause errors later on.

Result

The impact of walkthroughs is difficult to measure, as they were mostly informal and geared towards making sure our code held up to our standards, not necessarily testing functionality. Due to the lack of unit testing there may be some specific and obscure cases in which our code may fail or struggle due to oversights of time complexity and hardware limitations, however we tested our code as extensively as we believed necessary for the purpose of this task.

Linking

The test design and evidence are linked here
https://drive.google.com/drive/folders/1PYDl0uw1bynkSeSN5uiLeJMb0U44Kadh
When the test is failed, it will be labelled Fail and be highlighted by red color; if the test pass, it will be labelled Pass and be highlighted by green color.

The related requirement is based on the updated requirement file.