Evaluation

For this assessment we were assigned a new set of requirements, and to ensure that these were all implemented fully we added additional tests to our requirements testing document [1]. We broke down the requirements given into smaller requirements that when all met would fulfil the new requirements. This made it easier to incrementally test our game as it meant we would not have to wait until the end of implementation to test requirements. This played well into the test driven development approach our team has been following for this project.

An important consideration in our evaluation was to ensure no previously existing requirements had been broken during the addition of our new requirements. To do this we went through the existing requirements tests and made sure that all of these were still passing. This was particularly important for many of the functional requirements as when adding lots of new functionality it is easy to accidentally break or change existing code.

As this testing was to be the last as were were creating a final product we took extra care doing our requirement testing. We made sure that each of the requirements was checked by one team member and done again by another. If they did not get the same result we would test the requirement again with another team member. This extra care meant that we could be confident that the final product met the brief. We also wanted to make sure that we had not strayed too far from the original brief, so a couple of team member made sure to go through our requirements tests and cross reference them with assessment documents to make sure they matched up. Again as this was the last assessment we wanted to be certain that the product we were delivering met the brief.

As we tested we updated our requirements testing table, noting whether the test had passed or failed and making a comment if anything had changed significantly from the last assessment. Green highlighted text showed new comments and Red highlights showed text that is no longer relevant.

Testing-

We continued to use a combination of black box and J-unit testing to make sure our code was of appropriate quality. However before doing any of these test we had to decide what we believed to be appropriate quality code. We decided that for our code to be of an appropriate quality it would have to meet a list of criteria. Though our research[2][3] we found there are many different metrics for measuring the quality of code and decided to follow some laid out by ISO 9126-1[3]. These are functionality, reliability, usability, efficiency, maintainability.

Our group has used a combination of black box test and J-unit test to ensure proper functionality of our code. We continuously used these through our development, as per our test driven approach. J-unit test were used to test whether specific functions and classes functioned in a way that satisfied the requirements. We also used black box test for this same purpose. A combination of these two method ensured full test coverage of our functional requirements, all of these tests have passed so we are confident the functionality of our code is of appropriate quality. We found black box test particularly useful for the new requirements given as we found it difficult to encapsulate these requirements into unit test. The black box tests allowed us to easily test functionality such as requirement f11[4] , it is very obvious for a tester to know if they turn into a zombie after death and if the game in continuing. While we did not use j-unit test for the new requirements we continued to use the existing ones to ensure that none of the previous requirements were broke whilst implementing the new features.

To test the reliability of our code we made sure each of the team went though and performed each of the black box test individually. The purpose of this was to increase our confidence that the code reliably worked for different users. We would log failures and the conditions surrounding them to help debugging issues. For our final product non of the team members reported any failures when performing the test so we are confident the reliability of the code is of appropriate quality.

As this is the final product we did not rigorously test the metrics of usability and maintainability to the extent of other metrics as nobody would inherit this code to use. However this did not mean we compromised on these two metrics. As we were only adding to inherited code and only had limited time we could not significantly change the architecture of the code. While this is the case the code we added was done in a modular and readable way. We are confident that the usability and maintainability of the code is of appropriate quality.

One of the requirements for the final product is that is can be run on any university computer, we decided that this would be a good efficiency measure for the product. As our product satisfies this requirement we are confident our code's efficiency is of appropriate standard.

Given more time we would have liked to have been more rigorous with our testing. For instance we would have liked to design more unit test that cover more edge cases and have more people play test our product. However as I have described above we believe that the tests we have devised have sufficiently covered the main errors and most important functionality of our product. All errors that were flagged have been fixed and we believe that all of our testing has shown the product code and gameplay meets all requirements and is of a high standard.

Testing result-

Out of our black box test [5] only one of them failed this was 9.15. We decided that this test was too specific and did not accurately reflect what our requirements was asking of us. For this reason we designed a new test that more accurately reflected what the requirements were asking. This resulted in test 9.16 which passed. The rest of our Black box test passed on all tries. We ran all of our inherited all unit tests to make sure existing functionality still worked, these all passed.

[1] - Shaun of the devs , ‘RequirementsTesting4’ [Online], April 2019, Available: https://lloydbanner.github.io/SEPR-Team-7/RequirementsTesting4.pdf

[2] - Test institute, ‘What is software quality assurance’ , 2019, Available: https://www.test-institute.org/What_is_Software_Quality_Assurance.php

[3]- ‘Software quality ISO standards’ [Online], 2008-2009, Available: http://www.arisa.se/compendium/node6.html

[4]- Shaun of the devs , ‘Requirements4’ [Online], April 2019, Available: https://lloydbanner.github.io/SEPR-Team-7/Req4.pdf

[5]- Shaun of the devs , ‘BlackBoxTestingEvidence’ [Online], April 2019, Available: https://lloydbanner.github.io/SEPR-Team-7/BlackBoxTestingEvidence4.pdf