Evaluation & Testing Report

Following on from Assessment 3

The owners of our adopted project had performed minimal testing, while not covering all of the brief in their Requirements document, with the points system and safe zone being omitted. As a result of this, more tests were implemented in order to properly test and evaluate our product.

Their Requirements and Requirement Testing document were updated to include the Assessment 4 requirements, and include two brief requirements that had been missed. More Black-box tests were added alongside Shaun of the Devs' and some White-box tests were implemented in order to ascertain the quality of our project.

Methods for Evaluation

For evaluating the final product, we used Requirements tests. As the Requirements document was constructed from the brief and had been updated throughout the project timeline, following a peer-review by our team and some additions to cover Assessment 4, it was considered to be comprehensive and if all requirements were met, so too was the brief. Alongside this, Black-Box Tests were completed as implementation-level versions further supporting our confidence that the requirements have been fulfilled.

In order to ensure that the tests continued to certify that code was both providing the correct functionality and was of sufficient quality in light of the change in requirements, we felt it necessary to update our tests by creating more tests while leaving the original tests as-is. The idea behind this was that the previous tests were already certifying that the system met the requirements up until the change in requirements, so our additions could only serve to further constrain and therefore direct the development of our system.

The additions to the requirements tests were simply to add a single new test for each new requirement presented to us; these being to allow the player to be turned into a zombie if hit a certain number of times, the game's story changing once the player becomes a zombie and providing some method through which the player can cure both themselves and also enemy zombies.

With regards to black-box testing, we performed updates in a very similar way to the way we updated the requirements documents. We felt that pre-existing tests would already be certifying the quality and functionality of code up to that point and so adding the appropriate tests to ensure the new requirements were also fulfilled would simply further constrain and direct our creative decisions, improving quality of code and making it easy to add new functionality without risking the pre-existing functionality.

Additionally, we considered a quality game to function correctly with working code, therefore relying on our unit tests, mentioned below.

Requirement Testing Document: link
Black-Box Testing Document: link

Methods for Testing

To test the final product, White-box tests were implemented. The White-box tests were done in the form of unit tests, implemented using the JUnit library for Java. From the beginning, we have felt that the only way we could guarantee that the system was both of appropriate quality and functionality is if the system, in the end, passed all of the tests. The final version of our system has fulfilled this.

With regards to white-box testing, our changes were simply an update on our predecessors' tests in order to make some more difficult to pass: the idea behind this being that this would be a much more accurate identifier as to whether the code was functioning correctly or not. Included in these changes were adding tests to ensure that different kinds of zombies are meaningfully different from one another, updating the player tests such that all of the different kinds of player were meaningfully different from one another (previously only 2 of the 3 were tested) and also adding the tests to ensure that the powerups added during assessment 3 functioned appropriately. We define 'meaningfully different' in this case as the entities having different maximums for health and speed, as from the initial requirements elicitation it was revealed that it was insufficient for zombies and players to only be visually different from one another. All of our unit tests are inside a runnable executable, so the tests are easily recreated.

White-Box Testing Document: link

Testing Results

There are separate files for testing evidence for white-box and black-box tests but both use a tabular format. Each test has an ID to make it easily traceable to requirements, a description to give the reader a better understanding of the test, a section to say whether it passed or failed, and another column for any additional comments. See above for links to all the testing material.

42/42 of our black-box tests passed. We think that our black-box tests are suitable as they cover all parts of the game that we couldn't test in a more controlled environment. Black-box tests by nature can be difficult to reproduce but we think that due to the detail in our Black-Box Testing document, other teams will be able to reproduce our tests and get the same results.

41 of 41 tests passed of our white-box tests passed. We also ran the tests using IntelliJ's built-in code coverage runner. We were happy with our unit testing coverage because what was not tested in this way was covered by black-box tests which were able to cover system and integration tests also.

Requirement Evaluation

The updated version of Shaun of the Devs' Requirements Document can be seen here, updated to include Assessment 4's change in requirements, and 2 additional requirements, F11 and F12, to include the Points System and Safe Location asked for in the original brief. A Requirement Testing Document can be found here.

The following is the new requirements introduced at the start of Assessment 4 with their respective requirement in the Requirement document:

- "It should be possible for a player to turn into a zombie. Modify the game to allow for this, accommodating the new role for the player in the storyline." - F13
- "Add a new secret item to the game, a 'cure', which transforms all zombies in a restricted area into non-zombies." - F14

We have met all of our functional requirements, which are listed below:

- F1 - The game is split into the following six levels: Town, Halifax, Courtyard, Library, Physics and Central Hall.
- F2 - More zombies are spawned in the levels as the player progresses. Eg. more zombies are present in the Physics level than the Town level. More difficult zombies are spawned after the player encounters the safe zone.
- F3 - There are three options for the player character: Nerdy, Sporty and Drama. The Nerdy character has higher health, the Sporty character has higher speed and the Drama character has greater attack damage. Additionally, these characters have unique abilities; Nerdy has a Power Punch for greater damage, Sporty has the ability to Sprint to avoid incoming waves and Drama can fake damage.
- F4 - There are three different types of zombies: Regular, Fast and Flaming. These do not, however, have special abilities or reference a society as this was believed to overcomplicate the game playability and the zombie apocalypse storyline.
- F5 - There is a Geese Shooter mini-game, accessible from the Main Menu/Select Level Screen.
- F6 - There are five different power-ups which are randomly dropped once a wave of zombies is defeated. These power-ups are: Heal, Speed-up, Immunity, INSTA KILL and No Ability Cooldowns. This requirement does not include the secret item.
- F7 - There exists two bosses, one in the Courtyard Level and one in the Central Hall Level. They have much higher health and attack damage than the smaller zombies.
- F8 - The game can be saved and reloaded from the Main Menu/Select Level Screen.
- F9 - All enemies move in the direction of the player character, attacking once they are in range.
- F10 - The player can attack any enemies once they are in range and hits the zombie a distance away from the player. However, there are no weapons other than the power-ups, so the number of hit points the zombie loses does not vary..
- F11 - A points system is implemented, with points being able to be gained from the mini-game and each enemy the player defeats. The score can be seen on the Main Menu/Level Select screen and during the levels.
- F12 - The Library level spawns no zombies, while presenting the player with a large number of points when playing the Library level for the first time.
- F13 - The player turns into a zombie once losing all of their hit points. The Level Select screen changes to many variations of ‘brains’ and upon starting a new level, the enemies are humans.
- F14 - A cure can now appear randomly as one of the power-ups in the game. Once picked up, the player, if a zombie, returns to a human state and all zombies on the level return to a human state. All fighting ceases.

All performance, external interface and non-functional requirements are met, other than non-functional requirement N1.2, which requires there to be a tutorial in the game. Due to the player’s likely familiarity with zombie games and the lower difficulty of the first level, it was decided that this level can be the unofficial tutorial, as players often feel frustrated if a forced tutorial is given.