Sea of Geese

Testing Report

We have evaluated the product through both requirements comparison and testing. We chose to run all of our testing following the methods used by our group in previous tasks as this resulted in a more effective test coverage, which was almost identical methodology to the initial group with the exception of us using static testing too. This was opposed to a likely large and unnecessary amount of testing like the secondary group used, and whilst they opted for peer testing, black box testing, white box testing and JUNIT tests, our testing follows the format of:

1. Static Testing
2. Unit Testing
3. Integration Testing
4. System Testing
5. Alpha Testing

This has proven to have more reliable results in the past, and has overall been a great judge of what has been completed. This also highlighted any flaws we needed to work through. We had all members read through the code, statically testing it to begin with. This gives us a clear way to determine how well it matches the brief and what remains to be done. The code was overall written to a high degree of quality, very cleanly written, well modularised so it was easy to extend and implement new functionality. It was clear that this product would be a good project to continue development on.

Black Box Testing [1]:
https://davidjnorman.github.io/SEPR/assessment_4/Black-Box-TestingEvidence_AHOD4_SeaOfGeese.pdf
Initial Test Run: 3/8 fails
Final Test run: 0/8 fails

Unit Testing [2]:
https://davidjnorman.github.io/SEPR/assessment_4/Unit-TestingEvidence_AHOD4_SeaOfGeese.pdf
Initial Test Run: 1/6 fails
Final Test run: 0/6 fails

Our testing as a whole focused mainly on new features implemented. This was requirements driven and so the updated requirements are linked here [5]. To make the development and testing processes smoother, we have created a traceability matrix [3]. As we didn't alter their original code and only implemented on top of it, I decided it was best to focus on testing our work as recreating their previous tests would be overtesting. Previous group's test coverage is also on there and as can be seen we largely ignored areas already covered.

Our black box testing covered the implemented features fairly well, as this was only a small implementation I focused on key areas. The three initial failures [1], 1.03, 1.04 and 1.08 which were linked to requirements A1.7, A1.8 and A2.7 respectively [5], were all resolved or needed no fix. The first two were not technically guaranteed to spawn as the node map

generation is random, however they still had a significant chance of spawning and thus were passed. Test 1.08 however was an interesting instance where it used its own unique call for battle and so it only required changing a value within the encounter battle call instruction

Unit testing initially seemed like it could only have a minor coverage as most of their screens extended AHODScreen. This meant that it had a super(...) call containing GDX specific commands and thus, seemed impossible. With code adjustments, I was able to Unit test the logical aspects of the code effectively. Receiving accurate results, I managed to cover all new implemented features. The only test that initially failed was the points system, which was corrected with a major code overhaul. The obstacle variants and crew member systems passed flawlessly.

A Traceability matrix is provided here [3], as mentioned above. It covers all new implemented areas based upon the updated requirements as shown in Light Blue. The requirements are compared to the tests accordingly.

The Unit Tests are linked here [4]. They cover the Crew Member system, the Obstacle system and the previously unimplemented Points system. Though it is important to remember that most code needed adjusting so these will not work on all code. The working tests and adjusted code can however be found in the BenTesting branch within our group repository here [8]:
https://github.com/DavidJNorman/SEPR-4/tree/BenTesting

Previous group's testing can be found here, 7Seas of Something Testing Evidence (Assessment 2) [6] and Element of SEPRise Testing Evidence (Assessment 3) can be found here [7]. The combination of testing that these two groups performed covered most areas of code the few failures they had were either fixed with Assessment 3 or removed as requirements in assessment 4, these being B1.3 and B1.8. [5] We didn't feel the need to repeat tests as our work was additive to their code, building on top of it, not altering it.

As for evaluation, the final product meets the brief completely, as can be seen through both the code and by playing the game. This evaluation is done through not only statically testing the code but also through running through the game itself. It has at a bare minimum two modes, both a sailing mode and a combat mode. This is rather unique in comparison to most other projects from assessment 2, using a node map system, and a combat mode, although we have some special extra encountered nodes in addition to this, including some obstacles added within this assessment. There are five colleges and numerous of departments that can be encountered through each level of the node map. The colleges are level bosses and the departments are placed sporadically within the node map used for healing, purchasing buffs and for a minigame.

We added a previously missing points system. Now points are accumulated through winning battles against enemy ships; through battle nodes, encounter nodes and also boss nodes, which give additional points as specified. The boss must be fought to progress thus gaining more points the more you play. Because of the nature of departments, as a shop/minigame medium, there is no method or need to capture them. Gold is gained through succeeding in

battle too, with a larger reward for defeating colleges, but can also be gained through encounter node events.

Through the inherent structure and layout of their progression system, with each college as a final boss and no returning to previous levels we can see that the college capture system is already implicated and therefore did not actually need an implementation. This is in turn, a rather effective way to minimise code, letting the layout do it's work for the game. With this the college node at the end of each level is the only way to proceed, and to get to this boss you need to complete a path through the nodes to get to it. This not only restricts progress as required, but also allows you to experience more of the game and choose your own path.

Gold can be spent through the departments, which allow you to bet on the minigame, repair your ship and purchase upgrades. The minigame is a card matching system that allows you to bet 200 gold for the chance to gain a card you match as a reward.

Finally the two remaining brief requirements put forward in assessment 4, were the addition of a crew member mechanic and a natural obstacle to be added to the lake. The obstacle is a unique node that can be avoided; however should you come across it you must struggle, if you succeed then you can progress, however if you fail then you must accept a random effect such as losing health or a crew member. The crew member system implements three new crew members, Carpenter, Master Gunner and Quartermaster each with a unique effect, Carpenter provides healing upon victory in battle for each one present multiplied by three, Master Gunner reduces the draw cost of cards and Quartermaster increases the gold you receive upon victory in battle, with an additional pieces of gold, three for each one you have.

Overall the product satisfies the requirements put forward and plays well. Both through the enjoyment of the player base and through the evaluation of the pieces that make it up, functioning completely as its own game. The testing revealed the code functioned as expected, some bugs were ironed out and the product is usable, enjoyable and aesthetically pleasing, it fulfils the briefs fully and in this instance as what was required was open to interpretation as opposed to concrete requirements as may be seen in the industry we believe it is far to say that this is a working final product.

The completed product completely meets the final requirements [5], which can be viewed at the following location: https://davidjnorman.github.io/SEPR/assessment_4/Req4.pdf

The non-functional requirements and constraint requirements that don't affect gameplay are few and already achieved, however the many requirements based upon functionality are many with but a few not met. With the completion of our final product the only two requirements unfulfilled, B1.3 and B1.8, are based around an initial idea that movement should require resources. These were no longer necessary requirements as they did not suit the game being developed and thus though displayed are not kept for our project.

Aside from these two there was one more requirement which had been described as fulfilled and tested however we disagreed. This requirement was the points system which had been classed as equivalent to gold/plunder, with the end of our final project this has been implemented. These requirements-testing traceability matrix comparison can be viewed here: [3]. https://davidjnorman.github.io/SEPR/assessment_4/TracabilityMatrix.pdf
As can be seen from this traceability matrix, our product, both through testing from previous groups and the testing we have performed, covers the whole project. Everything implemented and expected in the final version is here, all of the requirements are met within the product.

The only requirements I personally felt were not met have the "Final Product" column cells highlighted in tan. Two of the four are mentioning that though the minigame does provide both 100G and a card, it does not always feel as rewarding as you might hope. The last two tan requirements are that it is based at the University of York, and the reward for defeating enemy ships is unproportional. Whilst this product does contain suitable information to identify it as the university with colleges at the end of each map, this may not always be enough, however this is up to interpretation. The player is rewarded for defeating enemy ships however this does not scale with enemy difficulty, therefore it may feel less rewarding.

Our final product does however, meet all requirements to be implemented, in some form or other. As mentioned above in the evaluation the brief is met completely, and therefore all requirements that have spawned from it are also met. The requirements coverage is technically 100% including previous testing that we didn't feel the need to redo.

REFERENCES:

[1] Black box Testing:
https://davidjnorman.github.io/SEPR/assessment_4/Black-Box-TestingEvidence_AHOD4_SeaOfGeese.pdf
[2] Unit Testing:
https://davidjnorman.github.io/SEPR/assessment_4/Unit-TestingEvidence_AHOD4_SeaOfGeese.pdf
[3] Traceability Matrix:
https://davidjnorman.github.io/SEPR/assessment_4/TracabilityMatrix.pdf
[4] Test Script:
https://davidjnorman.github.io/SEPR/assessment_4/TestScript.zip
[5] Updated Requirements:
https://davidjnorman.github.io/SEPR/assessment_4/Req4.pdf
[6] 7Seas of Something Testing Evidence (Assessment 2):
https://davidjnorman.github.io/SEPR/assessment_4/Testing_Evidence_AHOD2_7Seas.pdf
[7] Element of SEPRise Testing Evidence (Assessment 3):
https://davidjnorman.github.io/SEPR/assessment_4/Testing_Evidence_AHOD3_SEPRise.pdf
[8] Testing Branch:
https://github.com/DavidJNorman/SEPR-4/tree/BenTesting