Change Report

Formal Approaches to Change Management

Following Assessment 2, we had to decide, as a group, which other group's SEPR project we would have to pick up to carry into Assessment 3 to complete. So that we could easily decide which group's project we should pick up, we decided to evaluate each group's project based on several criteria, such as: code readability, how well written their documentation was, and the likelihood that we would be able to implement their requirements in the short time we had for Assessment 3.During the presentations given by the rest of the cohort, we had 3 groups that we thought would be good options, which we then reduced to our chosen project, Undecided, in a team meeting.

Following on from this meeting, we went through their project and compared it to their requirement specification in order to create a list of exactly what we had to implement in order to meet their specification.As modifying another group's code could lead to very damaging side effects, we decided to modify their code as little as possible, instead focusing on implementing our own additions around their existing code. However, to be able to do this, we had to understand how their code worked. To do this, we spent a few hours going through their previously written code and documentation in order to get a good understanding of how the game fully functions.

After reading Somerville's Software Engineering [1], we decided to follow a simplified version of a Change Request Form, which we have called a Change Log[2], which would allow us to accurately keep track of any changes that were making to the game. Our version was greatly simplified due to our smaller team size and much less formal style of development.
In this Change Log, we would submit a change request, which would include a description, which documents/classes it impacted, a reference to the requirement it was for and a detailed explanation, as well as an ID for traceability. Once a change has been implemented, two members of our team would go through the game, making sure that none of the core functionality had been affected by its implementation.This method allows every member of the group to see what is being changed in the game and who is working on it, in order to prevent multiple people from working on the same change at the same time, reducing the risk of conflicting code and wasted time.

When a change is suggested, we discuss its merits as a team, deciding whether it allows us to move towards meeting the requirement specification and if it doesn't, it is rejected, in order to make us more efficient. Once a change has been implemented, a member of our team would go through the game, making sure that none of the core functionality had been affected by its implementation, as well as using the testing methods previously used by Undecided. Having IDs in our change request form has enabled us to have both backward and forward traceability so that we could match requirements to our changes and vice versa, to ensure that we were developing in the right area for our game to reach the preset requirement specification.

Sea of Geese

Testing Report

Our team decided to use a different approach to testing than the previous team. We decided to use a method following the approach our group used in assessment 2. We use Unit testing, and Black box testing, in which black box is split into Integration, System and Acceptance Testing. Undecided use grey box, merging both white and black box, but we believe white box isn't necessarily efficient testing for this particular project, primarily because we care mostly about the component output than specific internals.

More so, we put a heavier weight on black-box testing, as in the context of the project, black box testing allows you to test what the user is more likely to care about, the components being correctly displayed and functional.

Black Box Testing : https://davidjnorman.github.io/SEPR/assessment_3/BlackBoxTesting.pdf
Initial Test Run: 7/18 fails
Final Test run: 2/18 fails

Unit Testing: https://davidjnorman.github.io/SEPR/assessment_3/UnitTesting.pdf
Initial Test Run: 6/9 fails
Final Test run: 4/9 fails

Our black box testing was vast, and I'd say there isn't much change in the concepts of the type of area that should be tested when compared to what Undecided used for their white box tests, as testing generality and function in game is similar regardless of the testing type. Any that passed are detailed in the document and each have a given solution written in.

Our Unit Testing covered: The minigame, Weather system and Sea monster. The errors in the black box testing were mostly fixed, other than 1.17, which is a LibGDX framework problem with collision, and 1.18, a temporary, non-critical error given its nature. Errors not fixed were all down to the nature of the code using random methods, which aside from 2.09 could not be tested and pass without altering code, going against the purpose of the testing.

The SailingScreen and CombatScreen classes have received minor changes to make them more modular, and thus, easier to Unit test. These changes are very small and don't make any difference to the execution of the code, simply to make it easier to both understand for anyone that looks at our code in the future and also to test what we have implemented since taking over from Undecided.

A Traceability matrix is provided below, covers most implemented areas, though we have not included requirements we did not attempt as the implementation would not be in place to justify a lack of testing, we have included some non requirement tests in their stead.
https://davidjnorman.github.io/SEPR/assessment_3/TracabilityMatrix.pdf

Below are my test scripts, we did have to comment out any libGDX sections of code for the code to function, however functionality of code remains.
https://davidjnorman.github.io/SEPR/assessment_3/Assessment3-TestScripts.zip

Sea of Geese

Methods and Plans

The updated plan may be found on our website, with updated portions highlighted, including our plan for Assessment 4 [3].

At the beginning of Assessment 3, we re-evaluated our method for software development that we had used in the previous assessment, whilst also comparing it to Undecided's methods. This time, instead of splitting into three groups of two or keeping as one whole group like Undecided, we split into two groups of three. The idea behind this was, it would mean that there would not only be more accountability for everyone to do the work they were on, but would also mean that we would get each section completed much faster, allowing us to make more meaningful progress. In these groups, two people were only developers and the third was focused on ensuring that changes were correctly submitted and tracked, as well as some development. Having someone in each group be in charge of the change report meant that we had much more clarity in what had been changed to our code between each person and there was much fewer instances of people working on the same part of the code at the same time without realizing. We also felt that being in just one group would mean that work is done less efficiently, with people working alone on larger sections.
As the change report also included which sections were being modified at the time, it allowed each group to work on their own section of the project, making it much easier to merge individual branches together after some work had been completed.

We continued to using Github as the version control which Undecided did too, as it allowed each group to work in their own branch whilst also allowing us to track what had been implemented in each commit, making it very easy for not only the developers in each team to know what had been worked on, but also for the change report manager to be able to easily see what had been modified since last time.We decided to have less whole team meetings in this assessment than last time, as we found it difficult to find a time where all 6 members were free. Instead, having us split into two teams, allowed us to set up meetings more often for the two development teams, whilst having the main meeting for the whole team in our SEPR practical, in order to catch up with what the other team has been working on. This allowed us to have our meetings be more productive as not only could we ensure everyone in each development team could make it, but we could also only talk about the section that was relevant to our group before we discussed it with the other team in our Friday practicals. We discussed following Undecided's method of 3 meetings a week, but felt that it took too much time away from development and that the increase in meetings wouldn't correlate to better or more efficient development. We kept with Undecided's communication tool of Facebook Messenger as we already used it, whilst using Discord for any team meetings that we needed to do whilst not at campus.When we first received Undecided's project, we looked through their game and requirements and made a list of what we would have to implement in order to be able to achieve their requirements [4]. We used this list a lot during our development, as it allowed us to mark what we were currently working on and then mark something as completed once it had been fully implemented and tested, allowing us to ensure that everyone in the group had an easy way of finding out not only what the other team had been working on, but also what they had left to implement. This made it easier for each time to decide what they would work on at each time, as we didn't have to go back and forth finding out what had been previously done since the last meeting.

References

[1] I. Sommerville, *Software Engineering*, Pearson, 10th ed., 2016
[2] https://davidjnorman.github.io/SEPR/assessment_3/ChangeLog.pdf
[3] https://davidjnorman.github.io/SEPR/assessment_3/Plan3.pdf
[4] https://davidjnorman.github.io/SEPR/assessment_3/ReqToImp.pdf
[5] https://davidjnorman.github.io/SEPR/assessment_3/BlackBoxTesting.pdf
[6] https://davidjnorman.github.io/SEPR/assessment_3/UnitTesting.pdf
[7] https://davidjnorman.github.io/SEPR/assessment_3/TracabilityMatrix.pdf
[8] https://davidjnorman.github.io/SEPR/assessment_3/Assessment3-TestScripts.zip