Change Report

For this assessment, we are required to take over another groups project and continue developing their project. Continuing the previous group's project will mainly consist of changes to the source code as well as altering previous deliverables and adding to this documentation to satisfy any changes made in the code. Due to the nature of our group organisation strategies and agile methodology which we are using, including short frequent sprints, switching projects should not cause too much disruption. Any issues that could potentially arise will hopefully be avoided by frequent communication and group meetings.
Before we started altering another groups code and documentation, we first had to select the group. We had a group meeting in which we downloaded all the groups' projects and assessed them. We discussed how well each group had tested their code to prevent our group spending time fixing unknown bugs due to poor testing previously. We assessed each project based on criteria which we defined prior to playing the games and reviewing code and documentation. These criteria are: **Well structured, modular code** which would be easy to pick up and continue development with ; **Readable code** - documentation of code with comments and Javadoc in non-trivial methods and fields ; **Full and extensive testing strategy** with test outcomes evidenced (black and white box, with appropriate unit tests evidenced) to prevent unnecessary bug fixing ; **Unambiguous requirements** clearly displayed (preferably tabular) and readable ; **What library was used** to write the code and are we familiar with it? ; **Extensive consideration of risks** with severity levels and predicted frequency (likelihood of occurrence) associated with each risk. We split up during the group meeting into teams of 2 and in our pairs we spent time looking at different criteria for all the projects. For example, one pair reviewed the testing methods for all the projects and another pair reviewed how readable and well structured the code was from all the previous projects. Once we had taken note of how well the previous projects satisfied our criteria, we collated our findings in a table: **http://limewire.me/docs//assessment3/ProjectRank.pdf**, which contains a ranking of the projects. Once we had produced the ranking table, we decided to discuss the top two ranked projects to ensure that we all felt that it was the right decision moving forward with whatever project we chose. In the end, we decided that the team "Rear Admirals (AKA Undecided)" would be the best candidate to take over. We felt that choosing this team would limit the possibility of group disruption and we found that they satisfied the criteria well. This is due to the fact they also used LibGdx as did we in the development of our game in assessments 1 and 2 so we would not have to relearn the libraries again. After reading the method selection and planning we found that the group used a scrum methodology with lots of iterations as we have previously used so we were confident this method would continue to work. Now that we had the project we would be taking forward, we needed to agree on a strategy for keeping track of changes in the code and documentation. We decided to have 2 main documents which we would use to keep track of the changes. One is the change management log **http://limewire.me/docs//assessment3/ChangeLog.pdf** which would include all changes made to code and documentation (including risk assessment, requirements, method selection and planning). The document includes the date the change was made as well as who made the change so everyone could transparently see who was changing which parts of the project preventing the need for unnecessary communication after the change was made trying to find out who has changed what in the project. We also noted down how the change impacted the project to prevent clashing or confusion further down the line. Each change has its own ID in this document. All this information allows for traceability of the modifications we made. The second document we used to manage changes was the implementation changelog. Despite all changes to the code being included in the Change Management Log, we felt it beneficial to have a separate document exclusively dedicated to changes in the source code. This makes it more readable for the development team. The most important thing to keep track of is changes to the source code hence the need for a separate document listing the changes made. Altering source code had very high risks associated with the changes because there was potential for previously written methods to break and not function as they were supposed to. Keeping a change management log as well as ensuring use of the GitHub would allow us to restore previous versions if a change made unwanted alterations to other parts of the code and broke the functionality. This prevented large amounts of disruption happening during the development of new features. To test that no catastrophic changes were been made, we used the unit tests that the group published in their assessment 2 documentation to ensure everything was functioning internally as it should. We tried to alter previously written code as little as possible to stop these errors occurring, however, if we needed to change a piece of functionality we made sure it was well described in our implementation change log document. This also allowed us to update the architecture documentation if changes to the original code structure were applied. To change documentation, we used Google Drive as we did in the previous assessments. The reason for using Google Drive in this assessment is due to primarily two reasons. Firstly, the collaborative features meant that all the group members could work on documentation in parallel in real time preventing any clashing. Due to assessment 3 only having 2 large deliverables, we knew that it was inevitable for multiple group members to be working on the same document simultaneously so it made sense to use the Google Drive (google docs, sheets and slides). Another reason for choosing Google Drive to manage changes to documentation was, all the changes are saved and a log of all the changes is displayed as well as the option to restore a document to its previous version therefore increasing traceability of changes alongside our change management log document.

Testing Report

The decision we made as a group was weighted heavily by how well the previous group has tested the source code and how well they have documented and evidenced the tests. We wanted to make sure that whatever code was previously written had been extensively tested with unit testing as well as functionality being tested by black box testing. We wanted to ensure that all areas of the code had been tested. We felt that this would ensure that the code which we would be taking over would be solid and in a good position for development to continue with minimal disruption in the group. To ensure lack of disruption we felt that it would make sense to favour a project which had a similar testing strategy and method of evidencing the testing as we had used in assessment 2. The project we chose did this.

The first thing we did, upon downloading the source code and the unit tests from the previous group, was to use the previously written unit tests to check the program remained functional and the inner structure of the code still passed the appropriate tests. If any tests failed then we would be aware of these issues early on and could work on fixing them before continuing development. Once we could confirm that the previous tests still passed, we could then reuse the tests throughout the development stage whenever we made changes to try and minimise disruption when modifying methods which may cause issues elsewhere in the code. This allowed us to continue development of the source code which was now, to our knowledge, bug-free. This relied on us trusting the tests that the previous group published, which was discussed in detail in a group meeting.

Initially, we wanted to make as little changes to the previously written code as possible as we knew this code was tested and functional, however, as we began adding more features and satisfying new requirements (see updated requirements document), we found that we needed to modify previously written code to implement these features and requirements. This resulted in us both modifying previous unit tests as well as removing and rewriting some of the unit tests for certain methods which had been heavily changed. As the previous group had said there were incompatibilities between JUnit and LibGDX we set out to find a solution that would allow us to use JUnit properly to test methods without worrying about the implementation of LibGDX resources. We found that JUnit does not initialise LibGDX when running unit tests and so methods using LibGDX resources such as Textures would raise errors. To get around this we introduced the mocking framework Mockito; this allowed us to mock LibGDX resources and thus focus just on the logic of the units being tested rather than the implementation of other resources surrounding them.

The changes to the unit tests can be seen in the change management log **http://limewire.me/docs//assessment3/ChangeLog.pdf** with appropriate reasoning to why each of these tests was altered or removed.

We have made no changes to their white box testing document and we have kept all their previous white box tests. We used the white box testing document to test that we were not affecting other areas of previously written code when writing new methods and implementing new functionality. The white box tests which we used to check our code throughout can be seen here: https://taylorwillmott.com/SEPR/Assessment/2/TestingDocs.pdf

We noticed that they haven't included a table displaying their black box tests so we felt that it would be beneficial to create one ourselves. The format we used for black box evidence table is identical to the table evidencing white box tests which the previous group provided us with from assessment 2. We felt that for continuity it made sense to use this layout for our black box testing evidence. This can be shown in the black box testing document here **http://limewire.me/docs//assessment3/BlackBox3.pdf**.

We have extensively written unit tests for every method we implemented that we felt needed testing. We also ensured that the requirements we had implemented were functional through the use of black box testing which was a lot faster to run and a more efficient use of the groups time.

The previous group presented all of their testing evidence using a very similar method to what we did previously in assessment 2. We decided there was no reason to change how the testing evidence was presented. We did, however, notice that the previous team had not given ownership to tests and we feel that this is a key feature when it comes to traceability of testing. This has since been added on any updated or extended tests we have written. We also re-ran all of the previous black box and white box tests ourselves to ensure that we had not severely affected any functionality in previously written areas of the code. We have assigned ownership to the group member(s) who ran the black and white box tests again. Evidence and discussion of both black box and unit tests can be found here: http://limewire.me/docs//assessment3/Tests3.pdf

Testing Material URLs
Black Box Testing - http://limewire.me/docs//assessment3/BlackBox3.pdf
White Box Tests - https://taylorwillmott.com/SEPR/Assessment/2/TestingDocs.pdf
Unit Test Scripts - http://limewire.me/docs//assessment3/unit_tests3.zip
Testing Report - http://limewire.me/docs//assessment3/Tests3.pdf

Methods and Planning

When discussing which group's project we would take forward into assessment 3, one thing which we looked at in detail was the previous group's method selection and planning. To avoid disruption and obtain a project that had been formed in a similar way in which we had previously formed our project in assessments 1 and 2, we wanted to select a group who had chosen similar methods and planning.
We found that team undecided (rear admirals) had taken a scrum approach similar to ours with a very flat hierarchy and frequent sprints and iterations in the development of the source code. This suited our group once the project was taken over as we could continue the frequent sprints which allowed us to find any issues which had arisen, or could potentially arise in the future, very early which meant that we could deal with them very effectively.
We also found that team undecided (rear admirals) had used similar tools to those which we used in assessments 1 and 2. This allowed for an efficient changeover as all of our group members are familiar with the IntelliJ IDE as well as the LibGdx library. The previous team had also used GitHub for their version control and Google Drive for all of their documentation collaboration and organisation. This was useful because we knew all versions of the project were stored and easy to access if for any reason we would need to. We established communication with the previous group in case an event occurred which required us to need a previous iteration of the software or a previously written piece of documentation not available through the group's website (https://taylorwillmott.com/SEPR/).
The previous team had used some different methods and tools in areas, however. One of these which could have caused issues was the use of 'Tiled'. Tiled is a piece of software which allows you to generate maps very effectively and export them as a data structure called a 'TileMap'. The software is very straightforward to pick up and begin to use. Some members of the group were fortunately already equipped with the necessary skills and experience to prevent this new tool disrupting progress in the development of the project. The more difficult task with using this tool was learning about the data structure the map is encoded as and learning how methods were interacting with the Tile Maps. Due to the clear documentation of non-trivial methods written by the previous group, this learning process was very straight forward, despite the potential for disruption. The use of Tiled was discussed and justified by the group in the documentation provided from previous assessments. This gave us no reason to alter the use of this tool and we decided that learning how to use the tool would be less disruptive than changing the source code so it didn't use the software.
There were little changes to be made to the previous group's methods and planning however an updated document can be found on our website here: **http://limewire.me/docs//assessment3/UpdatedPlan3.pdf**. Any changes that were made were to discuss tools that we would be using which the group did not use prior to us taking over the project, as well as discussing how we planned on dealing with the changeover.
Due to the Gantt chart which we used in assessments 1 and 2 been assessment specific not game/group specific, it will remain the same and we will follow it for future assessments as planned previously. This is possible as we designed the Gantt chart to depend on tasks given in the assessment brief document, instead of designing the Gantt chart around specific tasks which only applied to our group/game. This decision was made with the changeover in mind. The Gantt chart can be found here: **http://limewire.me/docs/assessment1/Gantt%20Chart%201.pdf**.

References

- Project ranking table:
  http://limewire.me/docs//assessment3/ProjectRank.pdf
- Change Management Log:
  http://limewire.me/docs//assessment3/ChangeLog.pdf
- White Box Tests:
  https://taylorwillmott.com/SEPR/Assessment/2/TestingDocs.pdf
- Black Box Tests:
  http://limewire.me/docs//assessment3/BlackBox3.pdf.
- Testing Report:
  http://limewire.me/docs//assessment3/Tests3.pdf
- Unit Test Scripts:
  http://limewire.me/docs//assessment3/unit_tests3.zip
- Updated Methods and Planning:
  http://limewire.me/docs//assessment3/UpdatedPlan3.pdf
- Gantt Chart:
  http://limewire.me/docs/assessment1/Gantt%20Chart%201.pdf.