Change Management Approach

In order to make sure changes are made in an ordered and controlled way, we ensured that all change decisions were made and approved as a group decision. This required frequent meetings to ensure we all had something to work on but prevents unnecessary changes being made by a single person without the group's knowledge and potentially causing errors in documentation or code from mismanagement. This plan fits well with our existing agile team management plan as an agile approach works best with short tasks and frequent meetings to review the current progress.

Code Updates

Our strategy for code changes focused on a list of all missing features.

1. Each member to goes through and familiarise themself with the existing code and documentation for the project. This allowed us to each identify areas we were unsure on.
2. Meet and go through each of the requirements in turn to see if it is currently being met by the existing game. In each case where it isn't, we recorded it. Based off this list and anything brought up by a group member, that was decided to be sufficiently important to the game, we compiled a list of all features that needed to be added to the game.
3. At each meeting we assigned items from the list to members to implement before next meeting. This allowed us to each be aware of what we are supposed to be working on and prevents unnecessary features being added. Across development, a few extra changes that needed to be made were found and added to the list.

Changes made within the code should be clearly marked using comments to make it clear what parts we have added. All changes should also be recorded by the person who made them in a document to keep track of all changes.

Documentation Updates

The way we approached documentation was initially looking through all the existing documents left by the previous team relating to requirements, architecture and testing and discussed what we wanted to keep and what we needed to adapt to fit our aims for this part of the assessment as a group. Additionally, we decided that changes to how we would fulfill requirements and to the implementation were to be documented - always ensuring that we kept within our initial parameters for how much we wanted to change certain elements. At the end of each sprint, we collectively discussed the changes we had made within the week and whether they differed too much from what we had been given originally. Additionally the overall changes to the documentation made and listed here, have all been checked in a final group meeting to see whether they fell within our plans.

Testing Report

During the process of considering what group's project to undertake, one of the things we considered were their testing report and methods. We wanted it to be at a standard that was both acceptable to us, and at a level at which we can easily pick it up and expand on it. As the previous group adopted black box testing, white box testing, and Unit testing, we felt as if their methods and approaches were satisfactory. Additionally, they stated that the way they conducted testing was done specifically with Scrum methodology in mind, which appealed to us as this is what we have used in the past. We decided to adopt their practice of carrying out tests as soon as a feature is finished during sprints. Though we have reduced the number of unit tests we wrote because, as stated by 'Undecided' previously, the way their code is structured makes it difficult for unit tests to be constructed given that graphical elements are very difficult to test. Unit tests are still utilised as we thought they would be useful to implement prior to coding the minigame which is possible because the main functionality of the minigame is entirely distinct from how it is displayed on the screen.

We used both white box testing and black box testing extensively throughout the assessment. White box testing was used, as we went through all the requirements not yet met, and we made a test for each one of them to see whether we could get everything to work correctly. We continued to take advantage of the incremental approach used by the previous group, carrying out tests during the development as we progressed through it, while also testing the product at the end of its development to see if the initially met requirements were still met and if we have met the further requirements set by the brief, which is what we have done previously as our system testing and acceptance testing. During this process, we also focused on the two failed white box tests and fixed the bug found previously by 'Undecided'.

Additionally, throughout the project we used a member of the team who wasn't doing any coding to play through the game at the end of each sprint and asked them to note down any bugs or problems within the game as black box testing. These were all solved in the subsequent sprint, and since we didn't have any remaining in the end we didn't think it necessary to carry out additional formal black box testing.

We also continued to use the methods we used on our own product and have applied it into our development, such as System testing and Acceptance testing, as previously stated. Integration testing was also very successful in helping us iron out any issues when putting different units together, and so it was vital in the process of expanding and adding to the existing code. This was once again done using a mixture of black box and white box testing using the top-down approach as we were comfortable with it and knew how effective it can be. Overall, our testing was very successful as we passed every white box test (16/16) [1] and every unit test (5/5) [2] we created.

Method and Plans Changes

A comprehensive documentation of changes made to the previous groups Method and Planning document can be found here [3]. As we were undertaking a new project, we decided to rethink the methods that we selected to use by altering the past teams primary constraints. The changed constraints were: 'short project of about 4 weeks', 'the requirements would not change' and 'team members had experience with this kind of project, but not with all of the previous team's software/tools'.

As a result of this, we considered new models for the project. We considered the Waterfall model due to the fact that the requirements would not change, but we decided that this would not be effective due to the possibility of the customer asking us to alter the ways we had implemented the requirements - such as asking us to change what the final boss is. Instead, we used the Agile model that the past team did, not only to make it easier to follow on from their practices but also due to the past experience we have working as an Agile group. As a group, we have also found it effective to meet 2 rather than 3 times due to 3 meetings per week often not giving members enough time to complete their assigned tasks - hence we decided to have 2 meetings per week, on a Monday and a Friday.

For assessment 4 on the other hand, the requirements will change. Despite differing constraints, our Agile model is already well suited to this and so we feel confident that this approach will carry through well to the next assessment. As well as this, we felt satisfied with the Gantt chart provided by ‘Undecided’ for assessment 4 [4]

The tools that were used by ‘Undecided’ were mostly what we had used as a group in the past, therefore by using the same software tools meant that we did not have to learn how to use any new programs. As a result we continued to use Facebook Messenger for communication, Google Drive for document sharing and production, Github for version control. As well as this, using the well-made artefacts from the previous group resulted in less time and effort having to be put into changing plans such as the Gantt Chart [5] which was effectively and clearly displayed on the team’s website and Tiled Map Editor which was used to create the in-game map. Although we did not have experience with Tiled Map Editor, we found it easy to use to add additional colleges and departments as we had the previous team’s map artefact, and so we did not have to use alternative software. Despite this, we were not familiar with Trello which the past team had used for their Scrum approach. We questioned whether it would be more effective to use Github since we had more experience using it to implement the Scrum model. We decided as a group that utilising Github would be most effective as we would have most of our work, such as code version control, in one place rather than having to change between tools.

In regards to team organisation, we found that having too many roles put unnecessary pressure on members of the group - so we removed/altered some of the roles. Rather than all 6 members of the group developing software, such as ‘Undecided’ did, we decided it would be best to only have 3 developers due to project having very little left to implemented. Instead, we decided on a Meeting Chair which allowed discussion in meetings to flow more easily and be directed in ways which would benefit our project. We removed the role of Client Interface as we all felt comfortable talking to and emailing the client. We also removed the Audio Designer as we felt like this was such a minimal part of the project that it did not require a role and instead we would assign audio to a member of the group who has less work to do towards the end of the project. We added a Report Editor who would check over all of the documentation in order to remove any errors as well as add additional comments which allows further groups to more easily understand our decisions and methodology.

[1] Testing Table
[2] MiniGameTest.java

[3] Changed Methods and Planning
[4] Assessment 4 Gannt Chart
[5] Assessment 3 Gannt Chart