Method and Planning

Cohort 1 Group 1:
Ileri Adegeye
Maida Ahmed
Holly Ainsworth
Anthony Colin Jones
Luca Gilligan
Alex Joyner
Minhaj Zaidi

Software Engineering Methods

Our software engineering methods were based on an Agile approach. This involved us meeting each week to review our work and plan assignments for everyone for the next week. We found that this approach suited us because it gave us flexibility in our work, with us being able to continue older tasks or move on to new ones as we needed. In-between meetings we each had a week-long sprint in which we would attempt to get as much of work for the week done, individually or as part of a group.

The only additional parts to the process we had to incorporate was analysis of other teams' work and documentation to choose the appropriate project that aligned well with our planning structure. This allows us to expand on the work the original group did, as it is better to work with well documented code and use a similar planning structure.

In addition, we maintained a flexible relationship with our customer where our game requirements were specific enough to meet the customer's needs but still gave us room to make our own design choices. Also, we were able to approach the customer whenever we had any questions. This suited us because it meant that we were always able to make sure that the customer was satisfied with our work.

We also had the advantage of testing the product with users to ensure that we were meeting our requirements in a satisfactory manner. This gave us an external stakeholder perspective that allowed the implementation team to tweak any features that needed to be modified and gave us the opportunity to adjust our plans anytime any features needed changing by suggestion of the end user.

We initially started with implementing the features we were required to develop. This is because we had a clearer understanding of the requirements compared to the initial development. We also conducted a risk assessment so we were aware of all the potential risks that could occur during game development and how they could be avoided.
We made sure to leave comments and docstrings throughout our code so anyone who read the code would be able to understand what it does and in the instance of us inheriting this code we could keep track of what was an original feature and what was a new addition by the current team.

Source Control

For source control, we decided to use Git and GitHub. GitHub is a free platform that allows clean collaboration on code and project files, allowing users to work on separate branches of the codebase in parallel using separate branches. It is quick and easy to learn, aligning with our development time.
It keeps track of each person's changes using blame, helping us maintain an equitable contribution from each person. GitHub also provides a supported desktop application (GUI). This would provide a clean visualisation of our current branch and history, and integrate more cleanly with the service than a third-party GUI.

We were able to fork the original team's project as our team is familiar with github and has previously used it. We were also able to access previous issues and contributions in the codebase, making it easier to keep track of who did what and what features were implemented in tandem.

Collaborative Software

To collaborate on our work, we developed our deliverables on a shared Google Drive, to allow all of us to access each of our deliverables at any time we wanted and make any changes that we needed, helped by the fact that Google Drive is free to use.

Coding tasks issues were all put in github issues.
To communicate with each other throughout the project we created a centralised WhatsApp group chat.

Summary

To support our Agile approach, these tools worked well because they allowed us to collaborate quickly and adapt our work as plans changed each week. GitHub’s branching and issues helped us manage tasks during our weekly sprints and let us work without disrupting others. Google Drive supported the frequent updates to documents that Agile development naturally produces, while WhatsApp gave us a fast and flexible way to resolve questions between meetings. These tools matched our workflow and helped us keep progress throughout the project.

Team organisation

Throughout the project we would assign one more task to each team member each week. Then at the Tuesday practical sessions we would review our progress and in a separate meeting on the Friday, we would identify what needed doing in the week and assign tasks to each person.
We did not want everyone to be working on implementing the game at the same time as that could easily lead to errors through contradicting code, in addition to the game development becoming hard to organise. Also, some group members had prior experience working on our chosen game engine libGDX, while other group members had never used it before.
In addition, it would not be ideal for everyone in the group to be working on the documentation at once, as this could lead to many people in the group overriding each other's work and the documentation becoming disorganised.
Therefore we split into 2 groups. The programming group would develop the game code while the Documentation team would continue to work on the deliverables.
This allowed each group member to focus on one area of the project that their skill set was the most suited to. This would also make both the documentation and coding sections of the project easier to manage as there were less people doing each.

To make sure that the documentation was as accurate as it could be, there was some crossover, with some Documentation team members doing some coding and some coding group members doing some documentation work to make sure both groups fully understood the work of the other.
In addition this was beneficial to the coding team because they often had a lot of tasks to complete so any help the documentation team could give them was welcome, while the coding team doing some documentation was helpful for the documentation team because it allowed them to make sure their documentation was as accurate as possible.

Project plan

Throughout the project we would decide individual tasks for each person in our group meetings each week that would contribute to our larger goal of completing all the Assessment Two deliverables to a high standard.

Initial goals

Our initial goals were:

1) Fork the previous team's previous collaborative software and website.
2) Evaluate new requirements..
3) Finalise our additional requirements.
4) Start developing the game and continue working on other deliverables.
5) Keep updating the previous teams deliverables.
6) Have our game ready to submit for Assessment 2 and have deliverables completed

Plan evolution

Gantt chart showing overall group progress and workflow:

https://eng1-c1g1.github.io/assets/images/final_gantt.png

As the project progressed, we soon realised that there were too many tasks for us to complete as one group and we began splitting into sub-groups and creating plans for each individual subgroup to complete.
While in the beginning of the project our plans were quite linear, with us all doing one task which fed into the next one, as the project progressed we split into groups more and more often to get more done.
It became apparent that giving an individual a clear programming implementation to do directly related to the requirements gave better results than giving a group of individuals vague instructions.
Our task allocation grew near the end for deliverables and less for implementation, as implementation was executed earlier so the team didn’t have to do a late development sprint near the end of the project.