Method Selection and Planning

Engineering Methods

We decided to go with an agile development method as it allows software to be released in iterations, this means that it can be more efficient and easier to see what's happening within the code. Making it easier to fix bugs, or places where the code is broken, and doesn't work. This means that changes and updates will be less likely to impact the deadlines, meaning that it's easier to keep the project on track.

To ensure we followed the agile development process, we made sure not to solely rely on processes and tools, and instead on the development of it ourselves. We made sure to implement continuous integration whilst constantly reviewing what we were creating and comparing it to the original requirements, meaning that we could stay on track and evaluate whether we are heading in the right direction. To measure our progress we kept looking at the final product compared to where we were currently at. By stepping back and looking at what was already done, we can see how far along the project we are clearly, giving a better idea of what we want and need to achieve.

A technique in agile programming is pair programming, where two people work on programming together. The first person drives the programming, by writing the code whilst the observer looks over the code that is being written. This way, two different types of brain usage is taking place: one more proactive, thinking directly about what to write, whilst the other is looking at the code with a bigger picture, seeing if it is going to work and is going in the right direction. Before the programming starts, the pair programming can plan and discuss the code they are going to write. By having two different minds, ideas can be relayed between the two people programming meaning overall, potentially a better idea can be developed. We found this method really useful, it increased our productivity and production time, meaning we were able to cover more in the time we were given, meaning we could mitigate the risk of not meeting our customers deadline.

We planned bi-weekly meetings during implementation to discuss what we had achieved since the last meeting so that we had a better understanding of the direction we were heading, and what we might need to adjust in the plan to stay on track. By taking time to look at what we individually and collectively have achieved, can give us a better look at what we are creating and whether it fits the requirements. Additionally, we discussed whether our requirements were still suitable after working on developing the game, as sometimes it makes it clear that some requirements may be unnecessary or over-complicated. This way we can look at the final target and make sure we are constantly heading towards it.

We used GitHub projects to plan what we were going to do and keep track of what has already been done. This included using a dashboard with cards which we turned into issues. Then we assigned people to the task so that we could be clear as to who was responsible for each task. To the issues, we added ‘labels’, assigning a different reason to each task. This meant it was easier to know what to work on and when. For example, some issues had the label ‘bug’ which reminded us to fix a bug, whilst another was ‘required feature’ or ‘useful feature’. Using a drag and drop method to change the status of the cards means that we could easily track what was already being implemented and what has already been implemented. This way, in our bi-weekly meetings we could review what has been done, and add any new cards to the dashboard to be completed. GitHub projects worked well with our agile development process as we are able to easily track what is happening and what needs to be done. The tickets can easily be changed to fit new deadlines or targets.

We considered using Jira to plan and organise our team because it's a really great tool for keeping people on track and updating what needs to be done and what has already been completed. Most of us hadn't had experience with Jira before so it would take a while to set up and learn how to navigate and use it. Additionally, as we were creating our project on GitHub and using it as our main source of storing and updating code, for us, it made sense that we were to store our project organisation on there too. This way it was in one place so there wasn't too much to navigate through and we could directly see the changes to the code all in the same place.

A collaboration tool we used for team planning and discussion is Mural which is a whiteboard online where you brainstorm ideas by dragging in boxes similar to post-it notes. We used this when planning the user requirements, as mentioned in the requirements document, and when discussing what we wanted our game to look like and how it was going to function. The tool is useful as you can collaborate with each other at the same time and see the changes live. There are functions where you can start a poll and there are anonymous votes on different boxes, meaning that it's easier for the whole group to express their opinions equally.

To make sure the team were collaborating together whilst we weren't together in person, we created a discord group where we could discuss progress. We messaged daily, outlining what we had accomplished that day and our plan for the next day, depending on what we had completed that day. This way, we could see who had done what and know the progress we were making. For a team of size 6, this was a really effective method of collaborative working, because we had certain people doing more coding whilst some others were more designated to the task planning and documentation.

Organisation Methods

Initially we had a group discussion at what people's strengths and weaknesses were when approaching an engineering project. This meant we could divide the team into team members who were going to work on the creation of the game as they had more experience in programming and the other team members could do the documentation side of the project. These teams were divided even further into people who were better at specific tasks and hence could work together to produce a high quality output.

From here, we set out a rough plan of separate things we needed to cover and loosely assigned people to them. We had meetings regularly at the beginning of the project in order to discuss where everyone was up to in their assigned tasks. We also used the group chat daily so everyone was able to ask any questions they may have in between meetings and tasks could be redelegated if a team member was struggling. The group chat was also used to keep everyone aware of the progress each team member made that day. Tasks were again reviewed in the weekly meetings to see if anyone could perhaps use some help, or had finished the assigned task. The tasks were then split into smaller sub sections so they could be delegated to the relevant team members.

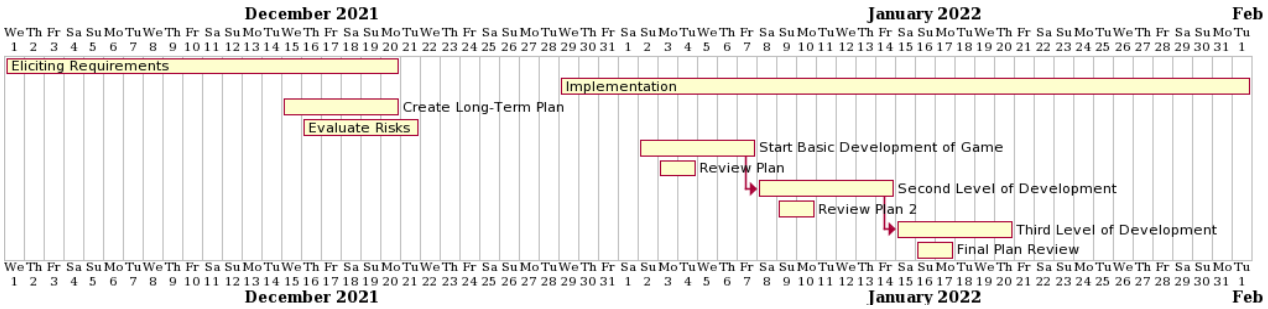
The main tasks for the creation of the game were micro-planned using the GitHub issues feature which meant all team members could easily keep track of what was yet to do, what was in progress and what had been completed.

This approach worked for our team as we were able to regularly check in with each other and ensure everyone is on track to completing their tasks and if they were struggling they could be given the opportunity to work with the rest of the team to solve any issues they were having. All team members were also able to easily keep track of the progress of the project using the GitHub and also the discussions in the groupchat.

This approach was appropriate for the project as there were many tasks that required completing so by constantly reviewing the tasks we were able to ensure all the tasks were completed in a timely manner and we did not leave anything out that was required for the project. As the deadline came closer we held more frequent meetings to ensure everyone was on the same page and all tasks were getting completed and so we could discuss final decisions of the project such as aesthetics.

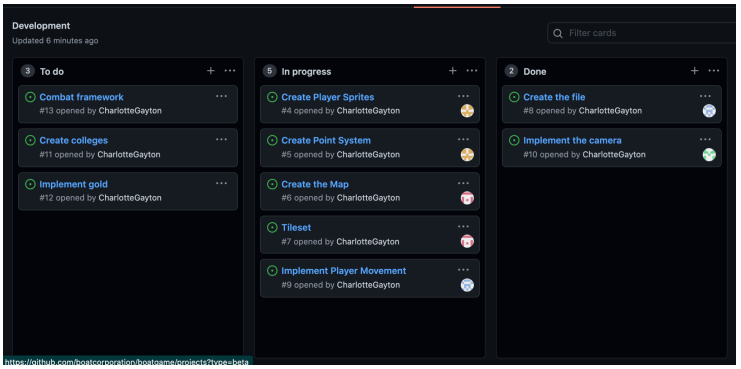
Give a systematic plan for the project. Your plan should lay out the key tasks, their starting and finishing dates, as well as task priorities and dependencies. Provide weekly snapshots of the plan on your team's website and discuss how the plan evolved throughout the duration of the project (5 marks, ≤ 2 *pages*).

- Gantt Chart



Micro-planning

We have used GitHub Issues in order to track, record and assign tasks to team members. In the project we created and stored the different tasks that needed to be completed and assigned them to specific members of the team. This is useful as we are able to keep track of what we need to get done, and what has already been completed.



<u>There were 5 key stages to our planning process:</u>

Stage 1 - Fully understanding the requirements of the customer.

To fully understand what the customer wanted we had a Q&A session with the customer, before which we created a summary of our understanding of their requirements. We then used this summary to write any questions that we were unsure of. After our meeting with the customer we were then able to write a more detailed and fully comprehensive summary of exactly what the customer needed for the product.

Stage 2 - Software requirements

The purpose of this stage of our planning process was to convert the customer's summary into key points that we could use to track our progress and have an easy to access list of all the features we needed to implement. This can be seen in our requirements document where all of the requirements are listed.

Stage 3 - Designing the product

This stage was a sketch of what we wanted the product to look like after understanding the customers requirements.

Stage 4 - Development

Stage 4 was the actual development of the product itself using LibGDX. This stage is where we actually implemented all the features we had discussed previously.

Stage 5 - Quality check

The final stage of our planning was the quality check of the product in which all team members played the game multiple times, trying different things such as winning, losing, playing for a short & long period of time etc, to ensure there were no bugs at any point in the game and all requirements were checked against to ensure they were all fulfilled.

Initial plan was created using Mural, in which we discussed all the features that were required for this project and brainstormed all the relevant planning points required. We were then able to create the Gantt Chart to lay out a rough timeline of the project. From there this allowed us to create our task tracker tables, and add in the required deadlines for each feature.

Our plan evolved throughout the project as certain factors changed. We kept track of our tasks using a variety of methods. We had the initial gantt chart which showed rough guidelines of the whole project. We then used a custom task tracker to enable us to individually track each task that needed to be completed and the progress of that specific task. As each week progressed we went back to the plan and updated the progress of each task using colour co-ordination(Red for not started, orange for in progress and green for complete), if the task was incomplete either due to it not being started or it being in progress, a new deadline was assigned to it and then the task was again reviewed on the next deadline date until it was completed. This method of planning worked well for us as it was easy for everyone to understand and see which tasks are yet to be completed. The use of github issues was also efficient as we were able to assign even more in depth tasks to the relevant team members and the same progress levels of; to do, in progress and done were used.

Game Development Plan

Tasks	Initial deadline	Completed?	New deadline	Completed?	New deadline	Completed?
Engine set up	14 January					
Basic Navigation	14 January		15 January			
Boat-like movement	14 January		15 January			
Colleges	17 January					
Combat framework (including capturing)	21 January		23 January		24 January	
All artwork	21 January					
Combat with colleges	24 January		26 January		27 January	
Point system through combat and sailing	24 January		27 January			
Capture of colleges	27 January					
Game fully working and all features implemented without bugs	29 January		31 January			

Documentation Plan

Tasks	Initial deadline	Completed?	New deadline	Completed?	New deadline	Completed?
Requirements						
Part A	14 January					
Part B	14 January					
Architecture						
Part A	29 January		31 January			
Part B	29 January		31 January			
Method selection and planning						
Part A	21 January					
Part B	21 January					
Part C	1 February					
Risk assessment and mitigaiton						
Part A	21 January		24 January		27 January	
Part B	21 January		24 January		27 January	
Implementation						
Part A	1 February					
Part B	1 February					
Website						
Complete	1 February					