Method Selection and Planning

Cohort 1 Group 8

Roy Asiku, Tianqi Feng, Andrew Jenkins, Nicholas Lambert, Tom Byron

Software Engineering Methods

When beginning this project, we decided it would be best to all take part in the requirements engineering and the architectural aspects of the project, as for when we begin implementing, it would be best to all have a clear understanding of how our game is supposed to be put together and how different objects would interact with each other. This is best for our group as, being a smaller team, it means that we can delegate the implementation of the game objects throughout our group in order to ensure a smooth and efficient implementation process where each member understands their object's place in the overarching system.

For this project we decided to use the game engine LibGDX along with the Eclipse IDE for our implementation. LibGDX is one of the most popular Java based game engines, with extensive online guides and documentation, which made it ideal for our project as we were unfamiliar with any other engine. It offers a wide variety of features such as font tools, physics engines and graphics tools. We paired this with the Eclipse IDE, again due to its popularity giving us a large amount of accessible online guides to it, and due to its many features. It is already integrated with the Gradle build tool, which is required to run a LibGDX project, and has easy to use git integration. We looked at a few different engines and IDEs before making this decision, such as IitiEngine, but being limited to Java 11 ruled some other engines out. Although we were all familiar with VS Code, it didn't have many online guides with how to use it along with LibGDX, and so would have been much more complicated to set up our project environment.

We chose to use Github and Git as our version control systems, which helped us to manage the codebase of both the website and the game effectively. Git allowed us to track the changes and maintain a comprehensive history of our development process, enabling us to revert to earlier versions when needed, and ensuring reliable backup of our work in case anything wrong happened - for example using commit histories, merging and quashing. GitHub improved our productivity by providing a shared repository for team members to contribute code, review changes and merge updates. By using GitHub's pull requests, we could conduct thorough reviews, ensuring code quality and consistency, before the code from other created branches was merged to the main branch.

For our website, we decided to use github pages. We chose this for two main reasons, the first being free website hosting. This is important because as a small team of students we decided that the cost of hosting the website would be an unnecessary expense and needless effort, it would be best to find a service that handles the hosting for us for free. Secondly, an important aspect of github pages is that it allows for group collaboration on the website, allowing each member to update the html and css accordingly, or push new website resources, such as diagram images or submission PDFs. Also, an important note is that github pages automatically updates the website after a commit, again abstracting the complexity away from us, making it the perfect tool for our project.

Team Organisation

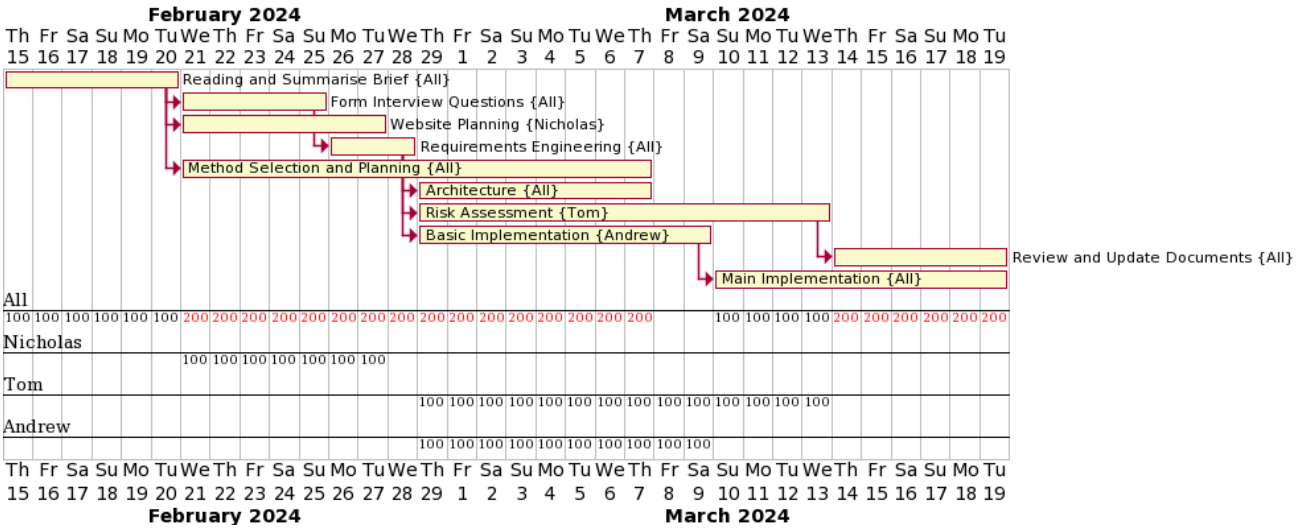
We decided to use discord for our online meetings to provide an easier way to communicate about the current state of our project, and what our next steps should be - this was largely motivated by the fact that our group is geographically spread out and we felt it would be more efficient to meet online. Discord also offers a messaging feature which gives us the ability to communicate even without meeting, and allows the group to work in a more synchronised fashion. Moreover, it offers the ability to ‘screen share’ which means that there are few benefits to meeting in person, since each person can see what the other is doing.

We have scheduled meetings every wednesday, to ensure that the group stays in touch and to build a routine of consistency, which reduces the chances that meetings are missed. We used ‘When2meet’ online which allowed us to synchronise our schedules and find the best time to meet. Our attitude has been to meet, even if there is little to discuss, as this helps reinforce the routine of meeting and working on the project.

We further decided to utilise a Gantt chart to schedule the phases of the development of our project, as being uml based it allows for easy creation and editing of a graphical representation of our projects schedule. It allows us to preempt what is coming within the project and prepare accordingly as a team. Furthermore, it allows us to plan out the sharing of our workload between team members across the development time of our project, and means we know what work we still need to complete - there are no surprises.

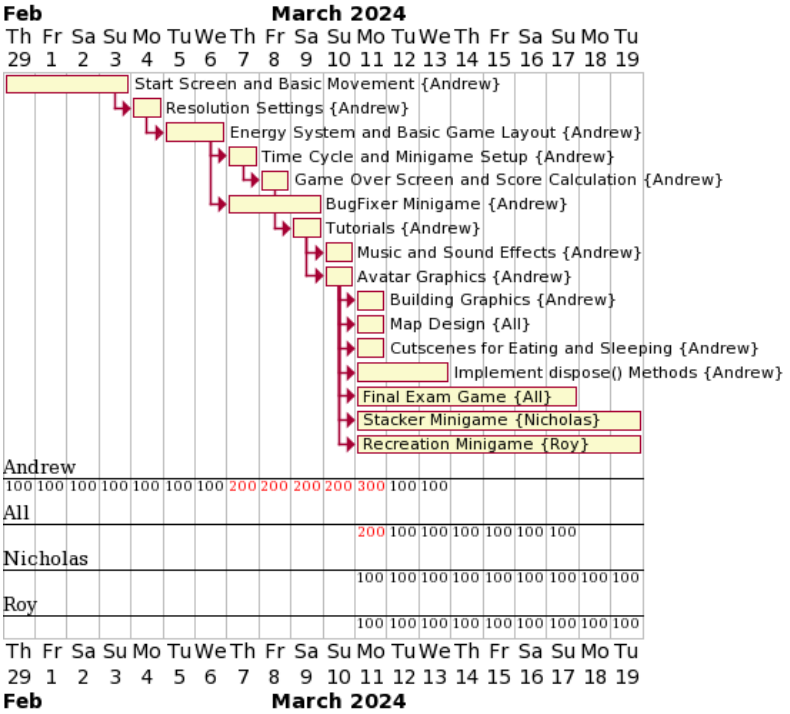
Final Gantt Chart Plan

https://nicholaslambert03.github.io/ENG1Website.io/plan_3.png



Final Gantt Chart Plan - Implementation

https://nicholaslambert03.github.io/ENG1Website.io/plan_implementation.png



Plan Evolution

The evolution of our plan can be viewed on our website:
https://nicholaslambert03.github.io/ENG1Website.io/msap.html

Version 1 (28/02/24)

https://nicholaslambert03.github.io/ENG1Website.io/plan_1.png

The first version of our plan was very rough due to our inability to plan too far into the future without much knowledge of how to do certain things, but we wanted to lay out as much time

Sas possible for implementation as we thought that it would take the longest out of anything. We also structured everything to follow directly from another topic, which didn't really reflect our actual work schedule, as we were already working on method selection and planning as early as this date for instance. This plan was more indicative of when we would draft these documents.

Version 2 (07/03/24)
https://nicholaslambert03.github.io/ENG1Website.io/plan_2.png
Although this plan was somewhat retrospective, it was a reflection of how we would plan and do our work, we had just not formalised it in UML yet. We planned to hopefully complete our first drafts of our architecture, risk assessment and this document around the 7th of March, with the goal of updating them as we continued working on the project and constructing new architecture. We also wanted to have a basic implementation done around this time, which would consist of menu screens, options, a map with basic movement and some placeholder graphics as well as a framework to add new features in the main implementation section. We didn't delegate our tasks too much yet, leaving that to our main implementation down the line, as we wanted to all work together on most aspects of the project. We also added in a review task at the end of our project, as we knew we had to update all of our documents such as architecture to be up to date after we had finalised the game. Additionally, Nicholas would have to continue updating the website with all of our newest revisions, so we allowed some time for that.

Version 3 (11/03/24)
https://nicholaslambert03.github.io/ENG1Website.io/plan_3.png
Our overall plan remained mostly unchanged from the last revision. Our only changes were an extension to basic implementation as we started main implementation a bit later than initially expected, and an extension to risk assessment as we noticed that there are more potential risks that we did not previously mention. We also pulled back our end date to the 19th of March to give us some breathing room before the submission date of the 21st. Our major change was the introduction of the implementation chart. The basic implementation was unfortunately a bit more retrospective as we did not create a formal gantt chart for it previously, but we created a plan for our main implementation and delegated tasks accordingly. Since we started our basic implementation quite early, we had a lot more time to work on minigames as the basic structure of the game was already complete.