Group 3 - Space Complexity

Members:
Kristy Mok, Ben Chalk, Corina Rivero Montiel, Euan Faller, Joel Cutler, Luke Callen, Sam Ade Fowodu

Method selection and planning

4a)

For the project, the team selected the Agile framework which focuses on collaboration, adaptation and continuous improvement. More specifically, we selected Scrum where the project development is divided into sprints for respective tasks.

Scrum allows regular iterations which is helpful for our project that involves different phases (requirements gathering, implementation, architecture, etc). It also gives us the flexibility to adapt to new challenges that may arise during the project, whilst providing continuous feedback since it breaks the project down into weekly tasks where we later evaluate our progress. This makes it simpler to manage as a relatively small team, as well as enhancing transparency and equitable workload distribution.

In each meeting, we report what work members have finished, currently working on and will continue working on for the week. In addition, at the end of each sprint, we review what has been completed and what not, what needs priority and what to do differently for the next sprint. If members have any problems and delays in their work, their sprint time and deadline will be recalibrated and adjusted accordingly.

Development/collaboration tools we used for our project:

- Google Drive/Docs: used for recording meetings and practical session minutes. The minutes detailed the progress of the projects and everyone's respective tasks. The shared drive organised and gathered all of our important documents in one space, allowing for a collaborative environment which also enabled all team members to contribute to the project on a real-time basis.
- Plant UML(Tasks Management): it was used to create the Gantt chart that contains a clear view of the different tasks assigned, their start and finish dates, and each team's member contribution. The visual format made it easier to track everyone's task(s) progress and clearly outline their priorities through dependencies of different members' tasks during the project development.
- Github(Version Control): it was chosen for source code management. It is where the code is stored and the tool facilitates collaboration of all team members. It helped ensure that the code was constantly up to date.
- IntelliJ IDE: it was the IDE chosen to develop the code. Its features like code completion, debugging and refactoring tools improved productivity and allowed us to focus on building the game more efficiently.
- Java 17: we used it to develop the core components of the game.
- LibGDX: it was our chosen game development framework. It provides the necessary tools for building 2D games. It also offers high flexibility for managing graphics, input handling, and game physics, making it suitable for developing our game.
- Discord: it was our chosen platform for communication, it facilitated smooth communication between the team members during the development of the project.

Alternatives considered:

- Notion (Tasks Management): It helped us plan collaboratively since everyone in the team could access and modify the files. There, we kept track of priority tasks and deadlines. It gave us a simplified visual representation of our sprints and made tracking easier. However, after consulting it with our practical professor, Notion was

not providing an ideal scalable Gantt chart and could not effectively display task dependencies. Hence, PlantUML was recommended to give a clearer graphical depiction of each member's contribution alongside the deliverable tasks and their dependency on others.

- JMonkey: We considered it as a library for game development but ultimately, we chose LibGDX since it has more extensive documentation and is better suited for 2D games.

4b)

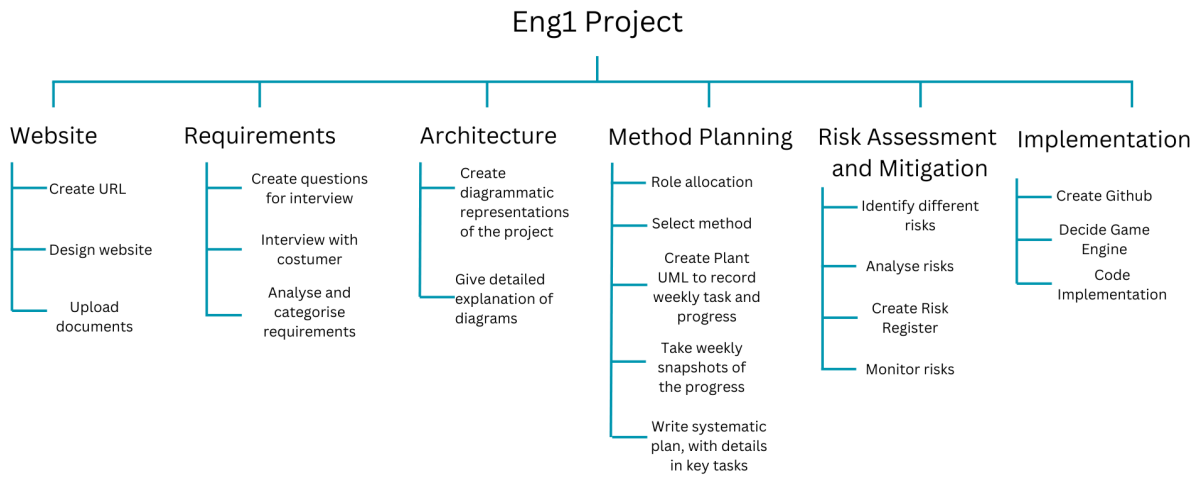
In the project's first week, we discussed allocating different roles and responsibilities. To begin with, we used various frameworks such as the York Strengths and Myers Briggs Type Indicator to better grasp our individual strengths and weaknesses. We then had an open discussion where everyone took turns to express their abilities and what they felt most comfortable working on, for example, some felt more inclined to work on the code implementation section and others on either project management or eliciting requirements. Alongside this process, we considered the mark contribution for the corresponding sections, hence negotiation for certain roles was conducted to ensure everyone had roughly the same mark contribution average. All members agreed unanimously on their final distinct roles.

Throughout the allocation process, we kept in mind the allocation of at least two people for each deliverable section to ensure we had a low bus factor to mitigate risks across the project development. In particular, three members undertook the implementation section as it was the most critical aspect of the overall project development. In addition, within each section, we avoided joint leadership and assigned the same leader of the deliverable section to oversee its following tasks. This was to maintain constant work style and quality throughout and reduce overlapping and confusion in each member's work process.

Overall, we felt that this was an appropriate approach to the team and project as it allowed members to work on deliverables they either set to challenge themselves or which aligned with their personal strengths and interests. In general, allowing our members to choose their dedicated deliverables increased their overall motivation towards the project.

4c)

Work Breakdown



<u>Phase 1: Initial Setup (Friday 27th September):</u>

Tasks:

- Team setup: assign roles and responsibilities
- Create a shared drive and GitHub repository, and choose communication platforms
- Select method planning: Agile and Scrum

<u>Phase 2: Website Setup and Requirements Gathering (September 30 - October 14):</u>

Tasks:

- Set up the website to store key documents and updates
- Brainstorm requirements and conduct stakeholder interviews
- Analyse and categorise gathered requirements
- Document the requirements and prepare reference material

Dependencies:

- Website and shared documents must be set up before documenting requirements

Priority: High

<u>Phase 3: Method Planning and Risk Assessment (October 3 - October 17):</u>

- Record each sprint and update tasks weekly on PlantUML
- Write the risk management plan and mitigate identified risks

Dependencies:

- Completion of requirements gathering before working on the architectural aspect

Priority: Medium to High

<u>Phase 4: Architecture Design and Diagram Representation (October 10 - October 24):</u>

Tasks:

- Begin diagrammatic representation of game architecture
- Finalise and explain detailed diagrams

Dependencies:

- Analysis of requirements informs the design process of the implementation aspect

Priority: High

Phase 5: Core Game Implementation and Testing (October 18 - November 7):
Tasks:

- Decide the game engine (libGDX) and begin coding base features like the grid and camera
- Begin implementation of UI elements (eg., menu, clock, progress bar) and core gameplay features

Dependencies:

- Architecture must be in place before core features are developed

Priority: High

Phase 6: Risk Review and Final Submission (November 8 - November 10):

- Review and finalise risk management, and update the register
- Package the code and submit final documents and game assets

Dependencies:

- All core features and implementation tasks must be completed

Priority: High