Method Selection and Planning

Part A

We decided to use an Agile methodology for our project. This entails us performing short software development sprints separated by group meetings.

Group meetings are used to review the progress of the previous sprint, as well as to bring up any concerns or bugs found during the sprint. This information along the initial Gantt chart will be used to determine what will be included in the next sprint. The remainder of the meeting will be used to design, plan and designate the tasks for the next sprint.

We used a variety of tools to aid with collaboration. These tools include the following:

- **Discord** - It is a communication platform that we have all used before and so we are all familiar with it. It also supports having different channels so each section of the project can have its own designated location
- **Google Docs** - This tool was used to store all our meeting notes and deliverables so we could all modify them at the same time.
- **Github** - Github will operate as our version control. We intend to stick to a single branch to store our changes and use pull requests to merge changes from our personal copies of the codebase to the main branch.
- **IntelliJ -** This program is an IDE specifically designed for java development. IntelliJ has a direct integration for use with Git and a dependency control system such as Gradle or Maven.

We thought the listed collaboration tools were the best fit for the project as they support both synchronous and asynchronous development and communication which allows us to both work together in project meetings meetings along with working on the project individually between them.

Along with collaboration tools we also decided to use a Game Engine to aid the development so less of our time is spent programming simple components of the program, for example collision detection. Using a game engine will allow us to free up more time for adding content to the game, instead of wasting time writing a framework for the game. Before coming to a decision we looked into several game engines to make sure we made the best choice:

- **LitiEngine -** This engine uses native Java for rendering and its physics engine meaning there is less complexity within as it is not built on top of a rendering engine. LitiEngine was made open source in 2017 and it is still in beta.
- **LibGDX -** This engine uses openGL for rendering and has support for Box2D for the physics within the game. LibGDX was initially released in 2014 leaving a lot of time for more obscure bugs to be found and fix and the documentation to be improved upon.
- **JMonkeyEngine -** This engine uses openGL for rendering and has a comprehensive physics engine. This game engine is packaged with a lot of utility classes to reduce the amount of programming required.

After experimenting with both LitiEngine and LibGDX we came to the conclusion as a group that LibGDX would be the best choice. This is for several reasons:

- LitiEngine is still in beta, and though simple to use even when experimenting with the tutorial code we ran into problems with the software.
- LibGDX is one of the largest Java game engines, this has the advantage of greater availability of support if we encounter problems during development.
- LibGDX is very clearly documented, making it easy to locate the necessary methods and variables to make progress on our game.
- Though not obvious from the surface of the website, JMonkeyEngine is designed solely for 3D games so would not support the style of game the project brief specified

For these reasons LibGDX was the best choice. LibGDX works well as a standalone engine and does not require any additional packages to make development simpler, this has the advantage of meaning we only have to go to a single place to find out information instead of looking through several packages.

Part B

Since the start of the project, we've put an emphasis on communication on our approach to team organisation. This is because we think that in a team project, it is most important that everyone stays informed about every aspect of the project at all times.

We decided to meet twice a week during term time (including the practicals) and at least once a week during the holidays. This schedule allows us to check in on each other's progress regularly and discuss any issues that may have occurred. Typically our meeting starts with a scrum report to check everyone's progress and to make sure we can help each other where appropriate. We continue by analysing what needs to be done for the following meeting according to our current progress and our planning document. We ensure that the workload is evenly distributed, and that everyone knows what they're doing before adjourning the meeting.

During the first practical we engaged in a team discussion to highlight the roles that we might need for the project. Then we assigned our team members with roles, which are in the table below, based on where their strengths lie.

Role	**What the role entails**	**Assignee**
Lead Developer	Support the rest of the team and be the first point of contact for any programming related questions	James McNair
Secretary	Maintain structured meeting notes and ensure nothing is missed during meetings	Dan Wade
Librarian	and be the first point of contact for any enquiries relating to deliverables.	Marc Perales Salomo
Report Editor	Manage quality control for deliverables.	Robert Murphy

The remaining two members (Alice Cui and Charlie Crosley) , though not assigned a specific role, will support all other roles, take an active part in development and research.

This formal role structure maximises our team's potential and facilitates our teamwork. However we understand that ENG1 is a long project, so we ensured that our roles are flexible to swap or change. This flexibility enables our team to accommodate changes during the project with ease.

Part C

In the early stages of the project, before we had a concrete plan, we aimed to prepare ourselves to be ready to start the actual project. This involves making the plan itself, having the requirements document and risk assessment document ready, and learning LibGDX. We started by completing the requirements document from the 19th of November to the 26th, allowing us to gain a better understanding of the tasks involved in the project. This is a high priority task because we need to know what is required of us before we move on to project planning.

The specifics of the project plan are outlined below in the Gantt chart. For a weekly breakdown of the project and to see how the plan evolved throughout the project see the relevant page of our website

The risk analysis of the project is important to do in the early weeks because it enables us to anticipate problems that might arise and prepare tentative solutions for them. Doing this helps us to reduce the chances of major changes in our plan and stabilises it.

Gaining practice on the game engine of our choice is also very important to start early on, because the familiarity with the software will facilitate our later work immensely.

Our plan for the next step of the project revolves around the art design of the game and the implementation of the game's features. The art design isn't as important, however the initiation of the implementation also relies on having the designs ready.

Throughout the project we intend to prioritize the deliverables as they must be completed for a valid submission at the end of the project, ideally we will have a fully functioning game which complete all of the outlined requirements, but if we fall short on some of the requirements we can still submit the project and simply outline what we have not achieved within the game.

Our game is designed in a way where there are a lot of features that can be missed out, for example we intend for each college fight to be unique but if we get behind on the project plan we could skip that idea and make each college fight identical to save on time.

The project plan also involves a stakeholder meeting which we arranged for after we finish the implementation of the combat with colleges. This is a key checkpoint for our project because depending on the feedback we receive, additional tasks could be added to the rest of our plan, which is also why we decided not to assign heavy workloads in the weeks following the meeting.

Gantt chart for our project