Assessment 1: Greenfield Development

Method Selection & Planning

Cohort 3 Group 9

Oliver Barden

Connor Burns

Sam Goff

Milap Keshwala

Lewis Mitchell

Jeevan Singh Kang

Harry Thomas

Titas Vaidila

Software engineering methods

Our team followed an iterative and collaborative software engineering approach, where we combined agile-style development cycles with regular communication and documentation to make sure everyone was always up-to-date. To support this we used a range of development tools and collaboration tools, each chosen to fit the project's needs and the team's preferences.

Git

- Use: Version control.
- Justification: Git is widely used in the industry for version control and a majority of our team members are already familiar with Git as they have previously used it in other projects.
- Alternatives: We briefly looked at other forms of version control including Subversion (SVN) and did some research to find it was easier to learn and was more beginner friendly. In the end, we stuck with Git as it is widely used in the industry, better supported, and suited the needs of our projects more closely as Git contains key features like branching which is extremely useful in a project with 8 team members.

GitHub

- Use: To host remote repositories and work collaboratively.
- Justification: GitHub is also widely used in the industry and well supported. Similar to Git, a majority of our team members already knew and/or used GitHub for hosting remote repositories and to work collaboratively. It has a great UI, making it easy for members who are unfamiliar with GitHub to quickly grasp how it works.
- Alternatives: GitLabs which is open-source software but has downsides including bugs and less third-party integrations which we likely need for this project.

libGDX

- Use: Game development framework
- Justification: Open source framework which supports both 2D and 3D games. Since our project is a 2D game, libGDX was a strong fit, further strengthened by its ecosystem. It provides many official and third-party extensions such as Ashley (entity framework) and Box2D (physics engine) which will be key to help develop this project. LibGDX has a large and active community with many resources that will help with learning and debug errors.
- Alternatives: We looked at using LWJGL which LibGDX was built upon. LWJGL would offer lower level access to hardware. We ended up using LibGDX due to it being more widely used, more suited for our project, and the steep learning curve that LWJGL has.

PlantUML

- Use: UML Diagrams
- Justification: Offers a text-based language which is then rendered into an image. The team member who worked on creating UML diagrams preferred this form of creating UML diagrams and so we used the tool which they were most comfortable using.

LucidChart

- Use: Creating CRC cards and Behavioural diagrams
- Justification: Provides an intuitive, drag-and-drop interface which is great for making diagrams collaboratively. It allowed multiple team members to edit in real time which helped the team collaborate efficiently. This tool is extremely easy to use and easy to pick up, which is perfect since most of the team members haven't used this before.
- Alternatives: Draw.io was considered as another free alternative but we felt LucidChart allowed for better real time collaboration and a more intuitive UI.

Visual Studio Code

- Use: Integrated Development Environment (IDE)
- Justification: Has core features such as built-in Git integration which is extremely helpful for the team members who are less familiar with Git. VS Code has Gradle support which is crucial since our Java game engine is libGDX which is built using Gradle. Most team members have previously used this before in past modules, so we felt an IDE that we were all comfortable with would be vital to the success of the project.
- Alternatives: IntelliJ IDEA was considered since there is a large connection between IntelliJ and libGDX, the wiki even mentions how they would recommend IntelliJ as the primary IDE. However, there is no technical or practical link between them, more so a historical one since most libGDX contributors used IntelliJ and it became somewhat a standard.

WhatsApp

- Use: Team communication
- Justification: WhatsApp helps enable instant communication between team members. We would provide updates, schedule meetings, and check up on progress via WhatsApp. This plays a key role in our day-to-day team's work.

Google Drive and Docs

- Use: Miscellaneous planning, shared drive and to write deliverables
- Justification: University of York already uses the google ecosystem which we all had licenses to use. Google Docs also has great real time editing and version history features which helps with collaboration and transparency.

Team organisation

Upon receiving the product brief for this project, we had to come up with ideas such as the hidden events, positive events and negative events. We held a group meeting where everyone pitched ideas for each group of events, and then we voted on which ones were the best. The most voted ideas were then chosen.

You could say we organised our team democratically, meaning we didn't have a group leader, however for certain complex and time consuming tasks we designated a leader who would manage and organise that task to make sure it gets completed on time and with quality. This way of organising our team suits this project since we need to split our work load evenly and we all have around the same amount of experience in working on projects so it doesn't make sense for someone to lead the group.

Throughout the project we have had multiple different tasks being carried out by small subgroups of our team. This is to ensure that progress is being made simultaneously on several different aspects of the project at once. In most cases at least two people were assigned to a specific task to add redundancy - if one team member is unavailable for whatever reason, there will usually be someone else to pick up the slack and continue working.

We made sure that during our meeting everyone pitched in and that we were always aware of the tasks which needed to be completed for that week along with who is responsible for these tasks. We would also discuss if a certain task was incomplete and what steps should be taken to address this.

Project plan

[work-breakdown.png]
For Assessment 1, we have to deliver certain deliverables and an implementation of the product brief. To achieve this, we broke down all the major tasks which needed to be done to achieve this end goal. We understood that not all tasks were equal, for example software selection will likely take a few hours for the team to research potential software we require, while implementation will take a few weeks. This was expected, the tasks at the start will be more about team organisation and planning, while the final stretch will be implementing and executing these plans.

Week 1

During this week, we were introduced to the team. The product brief had not been released yet, and therefore we did some team building exercises and organisation work. This included creating a WhatsApp group chat along with a github organisation which will host the website and the source code. Our work being deemed the best in our cohort greatly boosted team morale for the weeks to come.

Week 2

[week2GanttChart.png]
This week we have received the requirements document for the game and have begun to discuss how we will undertake the project. We were also informed about a client interview we will need to conduct next week. We also created the work breakdown structure this week and structured our gantt chart around this breakdown.

For the majority of the week we focussed on planning, this included reading both the product brief and assessment brief and then drafting up ideas for the project.The website was also set up this week. We also organised a meeting on the 30th Sept, where we discussed software we will be using throughout the project and what work we should be doing and the interview question we will be asking. We agreed to work on the deliverables once we conducted the client interviews. This made sense since we need to form the requirements using the answers from the interview along with the product brief.

Week 3

[week3GanttChart.png]
We were given a date for the interview and so conducted it on the 6th Oct. Following the interview, we transcribed the recording and used it to help form some of the requirements for the projects. We gave ourselves 2 weeks for both the requirements and the risk assessment, simultaneously working on these by splitting the team into sub-groups, as mentioned in team organisation, totalling 2 people working on requirements and another 2 on risk assessment. The rest will work on planning, which has been extended by another week since there were a lot of

similarities between the planning and architecture. As a result, we unknowingly started working on the architecture, yet did not see this as an issue since the majority of the initial architecture was considered planning. These were considered as dependencies since we could not start the core architecture and implementation before getting the requirement and risks down, therefore this was the main priority throughout the week.

Week 4

[week4GanttChart.png]
Surprisingly, we completed both the risk assessment and requirements earlier than expected, likely due to the fact working on this was a priority and we had 2 people working on each task. This meant we had to change up our original plan and bring forward the architecture; certain members were already creating initial class mock-ups in the previous week, so it made sense to start working on the core architecture. The planning also made great progress, we had all the events and other game mechanics decided. We had forgotten the assets that we will be using, and so we added this the gantt chart and made sure we had someone who would be working on creating and managing the assets needed.

Week 5

[week5GanttChart.png]
We decided to start working on the implementation this week as opposed to the following week as we thought 2 weeks likely may not be enough to complete the coding aspect of this project. By the end of the week, most of the architecture design had been completed and just the deliverable remained to be completed. Therefore, we started implementing this design, alongside the architecture deliverable, was now the top priority for us.

Week 6

There were no changes in the plan, everyone was either working on the implementation or the deliverables that had not been yet completed.

Week 7
Once again, no change in plans. The implementation had made a lot of progress, with most team members working on the deliverables as this was the main priority.

The planning gallery containing all the images mentioned in this document can be found here.