Method Selection and Planning

Group Number: Team 20
Group Name: **Gourdo Ramsay**

Group Members:
Lauren Waine
Megan Bishop
Tommy George
Bartek Grudzinski
Davron Imamov
Nathan Sweeney

Software Engineering Method Outline and Justification

In order to continue the development of Team 13's game, we decided to continue with our previous software engineering methodology. This was decided because it allowed our team to continue their workflow uninterrupted, not having to redesign and relearn how they were going to construct the implementation.

In Extreme Programming (XP) each team member was responsible for their component of the project: this allowed for each member to control their own workflow. SCRUM meetings add overhead to the development process which requires the members to hold regular meetings. These meetings are beneficial in a larger company which has the manpower for a SCRUM master and will most likely have a far more complicated programming task and more consequential deadlines. To elaborate on the latter, not only can the sprints be dependent on the previous sprints, they can also have dire consequences for the company if not met. We found that by removing the overhead and leaving the programmers to manage their own work, reduced stress and the persistent annoyance from attending meetings. Furthermore, due to the small team size of three programmers, the pair programming from XP supported this style of development heavily. The programmers understood each other's code and could help each other debug the code. They had great understanding and kept a standard which they all achieved within the code. Since each member had their own responsibilities, tasks could be prioritised within a small team because there was no overhead in manpower or man-hours through meetings.

Development and Collaboration Tools

UML: PlantUML

We chose to use PlantUML for our UML architecture diagrams because it was free, easy to update and intuitive since we used it previously. We did not want to learn a new software to speed up the progress of our implementation. PlantUML also had plenty of example code blocks which was very useful to expand our diagram choices.

Task Management: Trello

We chose to use Trello like Team 13 because it was a free software for task management. Our XP methodology worked well with this because it allowed us to quickly note down issues and tasks. Since we did not have SCRUM meetings, this meant the team was informed and could communicate issues in a flexible manner not reliant on timed meetings.

Version Control: Git with GitHub

We decided to use Git for version control and hosted our Git repository on GitHub. We chose GitHub because it allows multiple people to code at the same time and allows changes to be compared to previous versions and restored if needed. We are also able to create and host our website with GitHub which is extremely useful.

Team communication: Discord

To communicate in a rapid environment, we used Discord which allowed for group calls with screen sharing as well brisk file sharing - for ideas. The software used granted an inclusive ecosystem that was free, fast and moldable to our requirements. It supported communication, sharing of files and developing a plan of the project. The software supported multiple OS's and devices: keeping the team informed regardless of location.

Documentation: Google Drive

We used Google Drive to store our documentation of the project as it is accessible and familiar to everyone. Google Drive is also cloud based which we required as we wanted everyone to be able to contribute, even concurrently. Since Google Drive is cloud based there is a very low risk of losing documents as it is all stored remotely.

Java Game Framework: libGDX

We had to continue using this framework since it was chosen by Team 13. Had we decided to rewrite the entire project, we could compromise meeting the deadline or requirements. Therefore, we built upon the existing framework. Furthermore, we already had experience using the framework.

Team Organisation

Our team organisation was a democratic approach through decentralisation with no specific leader. No team member has more authority than the other. Work is to be outlined and volunteered for. Each piece of work has specific requirements that the team came up with along with a priority and start and end date. The advantages of this approach are it allows everyone to use their strongest attributes by picking their work type. It gives a feeling of purposefulness by utilising everyone through equal opportunities. The use of coordinated strict deadlines allows for the project to be completed on time. Most importantly of all, the democratic approach prevents blind faith put onto one person. As the team is inexperienced in game making, a lack of specific leader can ensure that overtime decisions average out in a positive manner. Instead of a volatile system like a specific leader which can fail or complete the project with no inbetween. Compared to the alternatives this decision felt the most fair to each member of the team. A hierarchical team organisation would create power

dynamics that would not be healthy within a University project setting. Since no external variable would drive the team to come together and work with each other unlike an actual professional environment. A matrix team organisation would complicate the ecosystem, being more proficient in a multi-team and multi-project environment. In conclusion, the degree of autonomy and responsibility is well supported by the project for the team. Each team member can support other team members through our chosen software engineering method, XP, which further solidifies our choice.