UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

ENG 1 Team Assessment
Group 18

METHOD SELECTION + PLAN

Group Members:

ROB ANDERSON

MORGAN ELKINS

ZIAD ISMAILI ALAOUI

SAM KNIGHT

SCOTT REDSHAW

GEORGE TONNERO

4.a - Software engineering method

We chose to use scrum as our development method of choice. This is because it best suited our team as it is relatively small and addresses the problem of the changing conditions when developing under time constraints. We split each week into short development sprints in which some of the requirements are implemented into the program and then at the end of the week we review what we have done and discuss what to implement next week. This allows us to easily keep track of what has been completed, what is currently being implemented, check what needs to be done and adjust our plan if any unexpected challenges occur. This is kept track of using Gant charts and other development tools, which are updated at each scrum meeting to reflect how the project is going and allow for the team to easily view our current progress.

Before deciding on scrum, we considered alternate development methods.

- The first of these was traditional plan driven methods which is when an overall project roadmap is used to guide the project through its life cycle. The advantage of this would have been that it is initially easy to plan and organise the work which needs doing as we can set a start date and deadline for each of the requirements. However, this leads to its biggest downfall, which is that when something goes wrong, or an unforeseen change occurs it is very difficult to adjust it to incorporate this without redoing most of the plan again. However, we did end up incorporating this method when doing the overall plan and write up/documentation for the advantages stated above.

- We also discussed using extreme programming (XP) which has the advantage of high usability due to constant user involvement. However, it requires two people to work on one project (paired programming) which has a high cost and isn't good for a changing environment which our project is and doesn't allow the group to share the load of the content equally.

Development and collaboration tools

For the project we have used a multitude of tools to help with developing both the code and the overall project. The use of the tools in conjunction with each other ensures good team organisation and increased productivity.

Collaboration tools:

- One of these is the use of the communication software discord which has lots of useful features. One of these are the voice channels which allow us to have virtual team and scrum meetings, which is useful when we can't meet in person or want to have a quick discussion easily. Another feature are the text channels which allow us to send useful information to each other such as

links, images and useful messages to help consolidate knowledge within the group and make sure everyone is up to date with the progress of the project. We also discussed using Zoom for communication but we decided it would be better to have both chat and meetings in the same place, like we can with discord, as it makes it easier to send information during meetings.

- Another tool is Jira which is used for issue and project tracking. This allows us to organise and set sprints where everyone on the team can contribute and see what needs to be done, who needs to do it and when it needs to be done by. This increases our productivity as it allows us to focus on setting the tasks and can use the software to sort out the timeframes and organisation.

- Lastly we use Google Drive as storage and file sharing to easily give everyone in the group the resources necessary to complete their task and look at other people. It also allows for simultaneous text editing using Google Docs which is useful when collaborating on the documentation and also filling in logbooks to track progress within the group.

Development tools:

- The Java Framework we chose to use is Libgdx which is a game development framework based on openGL and has library functions for movement, sounds, collisions (using box2D) and more. We chose this because during our research this seemed to have good documentation and be relatively simple to use whilst still being powerful enough to create our intended game.
    - We also discussed using LWJGL however it was a bit too low level for our project and it would lead to programming unnecessary things, like 2D physics, into the game where we could instead use ready built libraries like in Libgdx which would streamline our development.

- The IDE we chose to use is IntelliJ IDEA studio which is a feature filled IDE specifically designed for Java development. Some of the useful features we chose the IDE for was good integration with gradle (an application for running/compiling the game), auto-generating JavaDocs, git integration and it being easy to use by everyone on the team.
    - We also discussed using Eclipse, but we found that it was difficult to get it to work with Gradle and it's older so has fewer features which made it unappealing for our team to use.

- The version control we chose to use Git and use GitHub to host it. We chose Git as it was the one that we were most familiar with and had the advantage over other VCSs of branching capabilities. We also chose it because we could use it in conjunction with GitHub to be able to easily collaborate on code, comment on changes others have made and avoid merge conflicts though pull requests.

4.b - Team’s approach to team organisation:

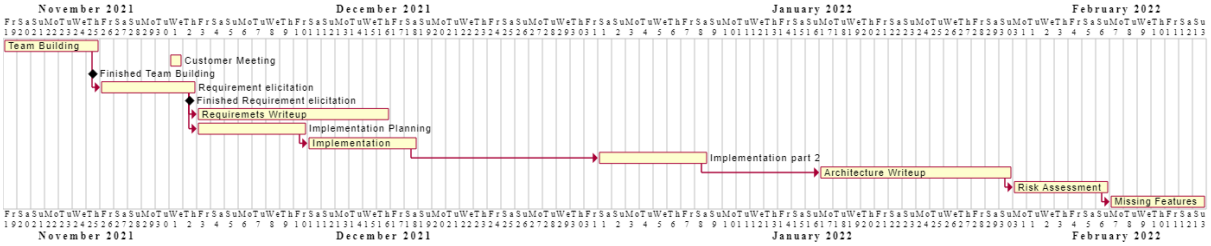
Prior to the implementation phase, all team members worked collaboratively to think up all the features that the game would have.
The division of labour permitted a more effective advancement of the project. Certain features of the game were distributed to specific members of the group to be worked on individually. Nevertheless, a mutual supervision was carried out in order to reduce the bus factor as much as possible. Each member of the team could contribute to another’s part, and contrariwise.

The fruitfulness of this approach stemmed from the ability of each member to dedicate their attention to intricate features without being distracted by other tasks or duties. A collective review was often organised to comment on each other’s work and point out potential ameliorations. The evolution of the project correlates with the adequacy of the approach the team chose. In this case, it proved to be efficient and rewarding.

4.c - Project's timeline:

The planning changed throughout the completion of this project due to external circumstances, personal reasons, and technical encounters.

The first projection of the timeline started off, on the 22nd of November, as follows:



(A bigger image is available at `https://eng1-group18.github.io/ENG1-GROUP18.io/`)

The changes only affected the way we arranged the tasks, but not the plan per se.

Team Building - 19.11.21 to 25.11.21

Getting to know each other and familiarise ourselves with the tasks in hand was a primordial step in this project. That gave us the opportunity to discuss what each person's strengths and weaknesses are, what they mostly enjoy doing and what they want to take up in the project.

Customer Meetings - 26.11.21 to 02.12.21

Before working on anything, it is paramount to understand what the client's expectations are. We worked over potential questions that could help us in understanding what the client desires.
The meeting outcomes gave us a clearer understanding of the client's vision of what his game would look like, what its features and functions must be, etc…
That led us to work over the requirements.

Requirements - 03.12.21 to 16.12.21

That was the core of the planning. What must the client see? What must the client not see? What ought to be featured? We compiled a list of requirements and ranked them in order to prioritise those which matter the most, according to our constraints.

Implementation - 11.12.21 to 18.12.21

This phase overlaps with the Requirements phase, as we could already start implementing bits we knew were necessary while working out other requirements that would be implemented later.
LibGDX was our framework of choice, powered by Java. We split the team out into two parts, those who were confident with programming and willed to implement, and those who preferred to continue to write up.

Architecture Write Up - 17.01.22 to 30.01.22

So as to implement the trickiest parts of the game, beyond the basics, we needed to design a concrete representation of the architecture of the game. We needed to have a universal and clear idea as to what the relationships between different objects in the game are, how they are meant to interact, and so forth.
This phase tremendously helps for further development and implementations, to circumvent potential confusions and oversights.

Risk Assessment - **31.01.22 to 06.02.22**

After having implemented the fundamental parts of the game, it is crucial to determine what potential issues users may encounter while using the product. Assessing which risks could conceivably arise would help us understand which sections need to be reviewed and fixed.

Missing Features - 07.02.22 to 13.02.22

The ongoing implementation may lack features that were expected by the client. Acknowledging which features are missing and planning their insertion is what this phase is dedicated to.

Changes

The planning was amended at multiple junctures for various reasons. On the 29th of November, due to the time that the Requirements phase took, which was longer than expected, we had to shift the implementation rightwards in the timeline.

We had to acknowledge the deadline of the first assessment, therefore we pulled the last phases (Risk Assessment and Missing Features) backwards in order to complete the project before the due date.

On the 17th of January, we extended the time we allowed for the last phases to ensure their correct completion.