Method Selection and Planning

Assessment 1:
Team 14: Bass2

Katie Maison
Saud Kidwai
Jacob Poulton
Cody Spinks
Felix Rizzo French
Joachim Jones

Assessment 2:
Team 6: Team siKz

Ryan Bulman
Frederick Clarke
Jack Ellis
Yuhao Hu
Thomas Nicholson
James Pursglove

4.(a)

Methodology we have chosen is agile.
The agile methodology allows the team to create many sub-optimal prototypes, test them, suggest improvements and finally, implement them, in a short space of time. This makes idea generation easier and the overall end product a lot more polished as each section or prototype has been well-tested and refined. This method allows multiple members, working independently, to complete their tasks and have their work, either accepted, or changed to match the description. Once we develop each of the prototypes, they are merged onto the main branch and compiled as the main program. This, we feel is the best way to complete our project to the best standard.

For **Communication** we have chosen Discord [1] because:

- It is a resource that each team member has experience with already.
- It has multiple ways to communicate so all can be kept on the same platform. For example; instant messaging for when we are doing work outside of meetings. It is also easy to go into calls if we want to do a quick meeting if we decide it is needed without having to send a link like with Zoom for example.
- This is important for our chosen methodology because the approach means we are working together on most aspects of the project, communicating with each other changes that might need to be made to each of our prototypes and deciding when to merge each prototype to the main branch. This is especially important if any of our prototypes requires features from other prototypes from any other of the team members.
- This term we've had the luxury of having in person meetings, so we took advantage of this to the full, and met each team member to discuss any changes or any features that we needed implementing, we each found this incredibly useful.

For **storing our documentation**, we have decided to use Google Drive and Google Docs [4]

- This is because of the ability to work on the same document concurrently, which is important as this mimics the essence of an in person meeting and lets every team member be on track and up to date with everything that's going on.
- Furthermore, the fact that Google Drive stores all files on the cloud and has a version control system mitigates the risk of losing files due to corruption or otherwise in Risk 12.
- An alternative to this would be one drive or dropbox, however google drive is the system that each team member already uses and has been provided by the university so it was easy to adopt.

For our **organisation system** we could've used something like Trello, but we found that a google doc suggesting what each of the team members will be doing was more effective than each trying to learn how to use trello concurrently and having to make a new card every time we wanted to add a new feature. We then discussed what we have been doing in our meetings and made it very clear what we still needed to do and are currently working on. This worked very well, allowing each of the desired features to be implemented in the required timeframe with minimal disruptions.

For our **version control system,** we have chosen GitHub as it allows us to collaborate with one another efficiently and as it is the industry standard, we decided that using GitHub would enable us to gain relevant experience from this project which we can then use in the future for our personal projects or when we work within the industry.

For our **development environment** we chose **VSCode [5]** . Based on our team members' experience, we found that there were quite a few advantages of using VSCode, such as the keyboard shortcuts and integrated Version Control support such as Git. Using the shortcuts available and the ability to automatically find potential bugs and fix them, transform code, generate code,etc.,we will be able to increase our productivity and focus our time and energy towards further development of the game and the project as a whole. It made it a lot easier knowing that each member of the team had quite a lot of prior experience using this software, which meant we could really focus on the problem at hand rather than getting used to a new software package

4.(b)

In terms of **team organisation**, the team's approach is:

- Instead of having a 'team leader', we split each section, dividing the project evenly to allow each member to have an equal opportunity (taking into account each member's strengths). We communicated well and made sure that any person that needed help with anything, was seen to and their problem would be resolved. This sped up the development cycle as no-one was waiting on the 'one' person to make decisions, we each made a decision, implemented it, and if the rest of the group didn't like it, we'd give constructive criticism and help them make the changes.
- Every member of the team in charge of sub-parts of the project (based on their strengths) this is because:
    - We want each member to feel the same level of importance and significance
    - Each member gains experience of leading meetings and discussions, hence avoiding any conflicts within the team and giving each member a sense of "taking-charge" of the project. .
- As our project progressed, we carried on with our agile approach for the remainder of our time. Once we had limited time left, with quite a few smaller finishing tasks to fulfil, we each benefited from the approach, as it allowed us to complete our prototypes quickly and efficiently, eventually, completing the project with time to spare.
- During this process we ensured that each member of the team would be up-to-date with the tasks being performed, which also ensured continuity between implementation and documentation..
    - This will help to keep the project on track and for example, if any of the members are ill/drop-out, then at least one of the members in the team will be able to continue from where they left off, hence it wouldn't be much of an issue as compared to if no one had any idea of what that person was working on. This helps to mitigate risk 4 and 7.

We **planned** our project in gantt charts.

The first shows the overall process and each of the following breaks the sections down into smaller tasks.

- We did this so that everybody is aware of the order in which we should do things and so that meetings can have a clear aim each week.
- As can be seen in the first overview Gannt chart however, we have chosen to add a review task that is constantly active.
    - This means that we plan to have a team member reviewing the work done while the rest of the team progresses in the project to ensure that the project isn't rushed or delayed, causing an estimation risk which is caused by technical debt as described in Risk 003 and an overall delay as described in Risk 004

4.(c)

For our **weekly plan** we used the idea of ‘weeks’ as the weeks we met up and worked as a group at least once. We did this because there were difficulties in meeting up and working as a group over the duration of the Easter break due to clashing schedules and upcoming exams.

Week 1+2:

- Week 1 + 2 was a lot easier than the first assessment, as we all knew each other pretty well by then. We knew what our strengths were and how to delegate each task to each person.
- We had a discussion about the other team's project and how we could potentially add features to it, generating more and more ideas as we went along.

Week 3:

- Week 3, we started looking properly at the code we’ve been given, reading the documentation and trying to make sense in our heads which bits we were going to change and which bits we were going to remove.
- We figured out the structure of the project pretty quickly and started writing very simple test code such as drawing some text to the screen to see if we could add some labels to the screen. This was an important step because this made it a lot easier to build upon later on.

Week 5:

- We thought about how to delegate tasks to the members of the group. The tasks were split up into deliverable tasks and implementation tasks.
- The implementation tasks were split up amongst the people who wanted to and were most comfortable with programming.
- We distributed the tasks evenly allowing each team member to have an opportunity to get some experience with the development cycle.

Week 7:

- We had a few weeks of implementation, where each team member got on with their designated task.
- We have to take into account that we had other commitments in this time, for example revising for the upcoming exams after the Easter break and general studies of other subjects.
- This week we found out how each person was doing, where they were up to, whether they needed any help or needed any explanation of which features needed to be implemented next.

Week 8 +9:

- This is where we finalised our project.
- Each member fed back into the group via many in person and online meetings, allowing them to get feedback on code they had written.

- We also used these meetings to discuss any new features or changes that needed to be made, for example, we decided to add a ‘golden barrel’ which would give the player gold on collision.
- This is where all of the prototypes/branches were merged into the main branch, allowing us to see how the finished product would look and see if we needed to make any changes.
- We then made these changes, fixed any bugs and finished the implementation
- After this, we made sure that all of the deliverables were up to date and completely finished.

Bibliography

[1]“Discord | Your Place to Talk and Hang Out,” *Discord*. https://discord.com/. (accessed Feb. 01, 2022).

[2]“Why use Google Docs vs. Microsoft Word,” *PaperStreet*, Aug. 28, 2019. https://www.paperstreet.com/blog/why-use-google-docs-vs-microsoft-word/. (accessed Feb. 01, 2022).

[3]GitHub, “GitHub,” *GitHub*, 2018. https://github.com/.

[4]“Google Docs,” *Google.com*, 2019. https://docs.google.com.

[5]VSCode, “Visual Studio Code,” 2022. https://code.visualstudio.com/.

[10] Braude, E. J., & Bernstein, M. E. (2016). Software engineering: modern approaches. Wave-land Press

[11] B.-A. Andrei, “A STUDY ON USING WATERFALL AND AGILE METHODS IN SOFTWARE PROJECT MANAGEMENT,” 2019. [Online]. Available: http://www.rebe.rau.ro/RePEc/rau/jisomg/SU19/JISOM-SU19-A12.pdf.