UPDATES

NP Studios – Team 8

Team Members

Lucy Ivatt, Jordan Spooner, Alasdair Pilmore-Bedford, Matthew Gilmore,
Bruno Davies, Cassandra Lillystone

Updated Requirements Justification

The first requirement we changed was **UR_seeHUD**. When implementing we came to the decision that we would like the entire map to be seen on the screen at once. This ruled out the need for having a mini-map as the player is already able to see the entire map, therefore we removed the mini-map aspect from **UR_seeHUD**.

In addition, we changed the game save system (**FR_save_quit)**. We decided that we are only going to save progress at the end of each level, meaning the saved content will only be which levels the user has completed successfully. This means the user doesn't have to repeat levels when they open the game up another time, making it more convenient for them. If the user clicks to quit the game entirely when they are mid-level, their progress through that level and things such as their health and water level will not be saved, so when they reopen the game they will need to start that level again with reset attributes. Each level is only approximately 1-2 minutes long, so we felt that our original idea of saving was over the top and unnecessary.

A slight change we made to **UR_instruct_engines** was to specify that in this sprint we are only going to be implementing 2 fire engines, which they can select to move around and attack. We are also only implementing 3 enemy fortresses (3 levels) which the user can attack at this stage of development. We have done this as this is what the brief asks for.

We have added the requirement of music to be played during the menu state (**UR_music**). This is just to make the game more intriguing and interesting (**UR_interest**) when the user first opens it up. We thought that it is good for our target audience, as if the game is in the menu state on Open Days, waiting for people to come over and play, the music may actually entice people to come over and grab their attention more. We added a functional requirement **FR_play_music** to match this user requirement.

We also added the requirement **UR_collisions** which explains that the user cannot operate the fire trucks to go through other obstacles (buildings). This is to make the game a little harder for the user, as for them to just go through things would make it too easy for them and a little boring. However, they can go through other fire trucks because they are not technically obstacles as the user controls them. We have also added a functional requirement **FR_deny_collisions** to match this user requirement.

For the **NFR_error_ prone** requirement, we realise this is quite vague as it is hard to test for every possible glitch, however we have tried our best to test both automatically and manually for any possible glitch to then eliminate them if we come across any.

Finally, we acted on the feedback we received from the original statement of requirements. We made various requirements less vague, such as **FR_enemies_die**, **FR_end_game**, **UR_victory_screen** and **FR_open_minigame**. We also added more fit criteria on non-functional requirements to make them easier to assess, as well as changing a couple of non-functional requirements to be functional.

View out updated statement of requirements:

https://npstudios.github.io/files/Updated_Statement_of_Requirements.pdf

Summary of Tools and Method Changes

Starting the second assessment, we thought it would be beneficial to review the tools we were using now that is was time to start implementation. Originally, we had decided to use Trello for our organisation of tasks as well as for our scrum backlog. Upon planning the backlog, we discovered a tool called GitKraken Glo Boards. This is a tool similar to Trello however it is integrated with GitHub's issue tracking. We investigated it and noticed that the Glo Boards had a nearly identical concept and interface as Trello, but with the addition that any lists and cards created would be added and synced with the issues board on GitHub. We also noticed we could assign GitHub users to the tasks. Due to these features we decided to use this for our implementation rather than Trello. We created lists for Backlog, In Progress and Completed, as we had originally done on Trello. We are continuing to use Trello for the general organisation of documentation, as the lists were already created in great detail and we had made use of the feature of TeamGantt.

We decided to use a game engine as this would reduce the learning curve as not many of us have experience of game coding. In addition, it will save us time that we can instead use to improve the finer details of the game. As a group we came to the decision to use the libGDX game engine. This is because it has an extensive range of modules we can import, its well documented, and it allows us to export for Windows and Linux.

We first started by creating a new repository on GitHub which would act as a test repository. We decided to use this to follow a libGDX tutorial together so that we could learn the basics before beginning the actual game. We thought this would be a good idea as we all learnt not only how to develop a game in Java using libGDX but also how to use git effectively before starting the main project. We didn't want any errors caused by our learning process to negatively affect the game we had created so far.

Another slight change to our method is that we have all decided to use IntelliJ as our IDE. Originally we were each going to use our preferred IDE, whether they were the same or different to each other, however as we were setting the project up we found it was easier for us all to use the same one so that we could all help each other out if we ran into issues. IntelliJ was the easiest one for us to set the project up in as it was fairly intuitive and simple to use. As well as this, libGDX also provided specific documentation using this IDE to set up and export the projects. Additionally, it has a lot of efficient tools built in, such as generating constructors, getters and setters, which makes our lives easier as programmers. It also allows us to automatically generate Javadoc HTML pages which will be useful for any other groups who may inherit our projects in assessment 3.

Previously, we said that we had a minimum of two group meetings per week. Due to the larger workload in the implementation stage of the project, and the fact that we can't have face to face meetings over the Christmas period, we have decided to increase that number to three. Three meetings a week, minimum, means that we can get a lot of the implementation done effectively as a team while we can still meet up in person. This will be a harder task when we are all at home for Christmas as we can communicate more effectively in person. As well as this, we will all have individual plans for the festivities. We don't have a large amount of time between coming back in January and the deadline for the assessment, so we wanted to get a very large percentage of the programming done before we all leave because of this.

View our updated plans here: https://npstudios.github.io/files/Updated_Plans.pdf

Explanation and Justification of Risk Assessment Changes

As the project progressed, we made updates to severity based on what risks actually occur ed. Going through the project, we kept track of any risks which occurred by logging them in our risk tracker (**https://npstudios.github.io/files/Updated_Risk_Tracker.pdf**). This has allowed us to see how often risks seem to occur, what their impact has been and whether the mitigation we originally planned was suitable enough.

The most occurring risk was **R1_Absence**. We originally had the likelihood as medium however as it has occurred quite often enough to now be considered as having a high likelihood. The severity remained the same because it has only tended to be a single team member absent at any one time. This meant that the rest of us could pick up the missed work, resulting in low impact on the project. Changing the likelihood to high means that we are more prepared for people to miss group meetings occasionally, meaning we can adjust schedules and plan more meetings if absence leads to delay in implementation.

We also changed the likelihood rating for **R7_Skill**, as well as it's severity. Originally, they were both set to medium, however we have changed them both to low. This is because the risk has only occurred once in the assessment two sprint, which shows it isn't as likely as we thought. We also felt as though there is a low chance of it occurring again as we have successfully set up the project and are deep into implementation so the likelihood of new challenges arising is unlikely - we all understand the fundamental processes such as using git, IntelliJ and the game engine libGDX. When the risk did occur, we managed to deal with it quite quickly due to having six group members. We can work as a team to help overcome any issues related to the skills required for the project; we all have our own strengths so we can use them to help others who may possess that skill as a weakness, meaning less negative impact on the product. This is why we changed the severity rating from medium to low.

The third risk we updated was **R14_Length** by changing the severity rating to low. This is because we realised, we could organise more group meetings and do more individual work to make sure we finish the tasks that take longer. This does not negatively impact the project as there are enough of us to spread the workload.

The mitigation for two of the risks has changed as well. For **R8_Instances**, we originally said that we would use previous versions from GitHub if this risk occurred. We are maintaining this plan, but only for major issues. When we encountered some merge conflicts, we managed to resolve them by using online resources to teach ourselves. We realised this method was enough for smaller issues. Retrieving previous versions is only a method for when a major issue occurs which can't be solved.

For **R14_Length**, we will also keep the original mitigation plan but also added that we will do more work individually. If we make sure that we complete more work in our own time, it will make up the time taken when tasks have overrun the time we originally allowed. Group meetings are limited to when everyone is available, however individual work can be done whenever is suitable for you.

View our updated risk table here:

https://npstudios.github.io/files/Updated_Risk_Register.pdf