Module	SEPR
Year	2019/20
Assessment No.	3
Team	Mozzarella Bytes
Members	Kathryn Dale, Emilien Bevierre, Ravinder Dosanjh, Elizabeth Hodges, Daniel Benison, Callum Marsden

Method selection and planning

(N.B. Changes made to this document for Assessment 3 are in bold)

Development and collaboration tools

The team chose to use the agile method SCRUM; this method fits into the framework of the software engineering project that will be running over the course of 2nd year. By using the SCRUM method, the team benefits from its advantages such as lowering the risks by keeping a transparency for product owner and stakeholders and the team. There is a flexibility that allows adaptability for customer changes and requirements which other methods do not take into consideration as they use a higher amount of documentation. With less documentation that comes with SCRUM changes are easier to handle as clarification through documentation. **We chose to keep using this approach as we continued with the development of the project as it is an approach that we thoroughly researched during Assessment 1 and were already implementing.**

The assessment is split into 4 smaller assessments each with different requirements; SCRUM fits well with the software engineering project with this in consideration as the flexibility of SCRUM is beneficial to maintain the adaptability of the team.

Agile methods are based on a manifesto with four principles: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan. With these four principles, the characteristics for agile methods are incremental, cooperative, straightforward and adaptive.

By doing weekly sprints we developed an incremental approach, that kept the team and project open to new requirements from the customer, therefore remaining adaptive in our approach. By using this method, we created a straightforward approach with clear documentation of how the project will proceed.

By using the SCRUM method, the team will perform weekly sprints and therefore maintain a deadline one week before hand-in; with daily stand-ups over Facebook Messenger; over the holidays the weekly review sessions will also take place over Messenger as a meet-up is not possible. **As the duration of Assessment 3 doesn’t cover any holiday periods, it is unnecessary for us to have to implement it. However, we kept their use of weekly sprints and regular stand-ups over Facebook Messenger – although our meetings have been planned for every two days to account for the busy and varied timetables of different team members.**

For development tools we will be writing the application in Java whilst using the libGDX open-source framework [1]. Using libGDX as our framework will allow us to reuse code between platforms, significantly decreasing the amount of new code that has to be written, increasing the speed at which it can be ported. **Due to our prior experience with libgdx, we continued with its use for the development of the project. Furthermore, by sticking with the pre existing framework, we will reduce the number of changes that we will need to make in the refactoring stage of development.**

LibGDX is specifically good for 2D games, which is the format our game will be. It functions with programming languages Java, C and C++, with the requirement that we have to use Java for our game [2]. Using libGDX allows for Java tools to be reused as the framework is Java based. Furthermore, on its own Java has a slow performance[3], however libGDX uses Java Native Interface (JNI) which uses a native platform that enables Java to run on a virtual machine. This means that Java can be used in combination with languages as C and C++, which can handle different methods better than Java and which are closer to assembly code. As another tool within libGDX we will use the 2D particle editor which allows us to make 2D effects for our game, such as the water effect from the fire engines; with the 2D editor it allows for velocity, duration changes and in which direction the water is going[4].

There are two big build systems: Gradle and Maven. Between these two we chose Gradle as it has a faster build time and more modular customisation. With the incremental builds that Gradle use, it checks whether tasks are updated, if it is the task is not executed which gives a shorter build time. One of the downsides to Maven a goal can only be attached to another goal, this restriction is on Gradle. As we chose to work with LibGDX it is easier to use Gradle as it requires less plugins to make it compile [5]. **Again, we also used Gradle due to it's fast, incremental builds.**

Tiled[6] is an open-source general purpose editor we are using to make a map of York. We chose it as it produces a Tiled Map in a format that is accepted by Libgdx. Tiled provides an easy-to-learn interface and allows the map to be separated into layers. Layers can later be used to define which areas of the map should be collided with, this is useful as it eases the implementation for the game.

For the creation of assets, we are using Aseprite [10]. Aseprite is a pixel art and animation creation software specifically built for game development. It has many features that allow for the quick development of assets which will be useful for creating new art as well as modifying open-source art for the game. We also used MagicaVoxel [11] for the creation of the firetruck assets. MagicaVoxel is a open-source 3d Voxel editor that allows for the creation of 3D models. As a team we decided we wanted to have a fake 3D fire truck, this is accomplished by creating a Voxel model which is then sliced into multiple layers. In the game it is reconstructed with a slight offset between layers to create the effect of having a 3D model in a 2D environment. **For the sake of continuity, we plan to use MagicaVoxel to create any new firetruck assets, as well as any aliens for patrols or other key features needed for the game. However, we**

also plan to use Affinity photo and Cinema4D for the creation of other assets as these are software's that our team members are significantly more familiar with.

The codebase has been setup so that it can be written in Eclipse, Intellij and Visual Studio Code. We chose to make it compatible with all 3 programs in order to make it as accessible as possible for our team and teams that inherit our code later on. **Intellij was our chosen environment to develop the project. This is different from the environment used by the original team (Eclipse) however our team members are more familiar with it.**

One of the tools used for collaboration between the team is GitHub. GitHub is a code sharing platform that uses the git version control system, improving the ease of multiple people working on the same codebase between weekly sprints, furthermore we are also using GitHub to manage the tasks that need to be done. **We also used GitHub to share our code between group members.** This also provides a history that we can use to check project progress against the Gantt chart to stay within the schedule. We will also be using Google Drive for storing and collaborating on documentation as well as report taking. Lastly, Facebook Messenger will be our main means of communication for daily updates on progress and will allow us to solve simple problems with no meet-up face-to-face necessary. **Again, as the new team taking on the project, we used Facebook Messenger to discuss questions we had about the original code with other team members. This allowed team members to study the code in their own time.** Using several platforms for different aspects of the project will allow more organisation and structure between code, documentation and daily updates. We decided on these tools because team members were familiar with them before the start of the project.

Team organisation

The team members have been allocated different roles in the group based on Belbin team roles which will help create a high-performing team. The different roles are based on the strengths and weaknesses of each team member, which will result in a more efficient group [7]. We **followed the example set by the last group and trialled using Belbin team roles. It is a concept that we hadn't come across before, but after researching it we felt that this would be a very effective way to implement our SCRUM.**

- **Emilien Bevierre** - Specialist
- **Ravinder Dosanjh** - Completer finisher
- **Daniel Benison** - Monitor evaluator and specialist
- **Callum Marsden** - Resource investigator
- **Elizabeth Hodges** - Implementer
- **Kathryn Dale** - Team worker and co-ordinator

These roles were allocated based on the roles that we had assigned each other in our original assessment, however we also had a group discussion and looked at each team member's past contributions to decide which role would best suit them from the Belbin team model.

As previous Project Owner, Kathryn was best placed to take on the role of team worker and co-ordinator, as she already had experience in delegating tasks and motivating the team.

Following from his previous role as Development Lead, Daniel was a good fit to become the Monitor Evaluator and Specialist. Having the greatest experience in coding games and using Java for other projects, he can provide specialist support to other team members struggling to overcome obstacles in their code.

Emilien is also a specialist due to his experience in creating graphics, and so took a lead in the design of assets and the overall visuals of the game.

In past assessments, Callum was responsible for coordinating with the client. For this and the next assessment this role has been expanded to Resource Investigator, allowing him to research different resources that we could use, as well as communicate with the client and gather feedback from the target group that'll be playing the game.

Elizabeth's calm and focused demeanour makes her a suitable fit for the role of Implementer. Her organisational skills and coding experience allow her to find the best ways to implement the ideas from other team members in the most effective manner.

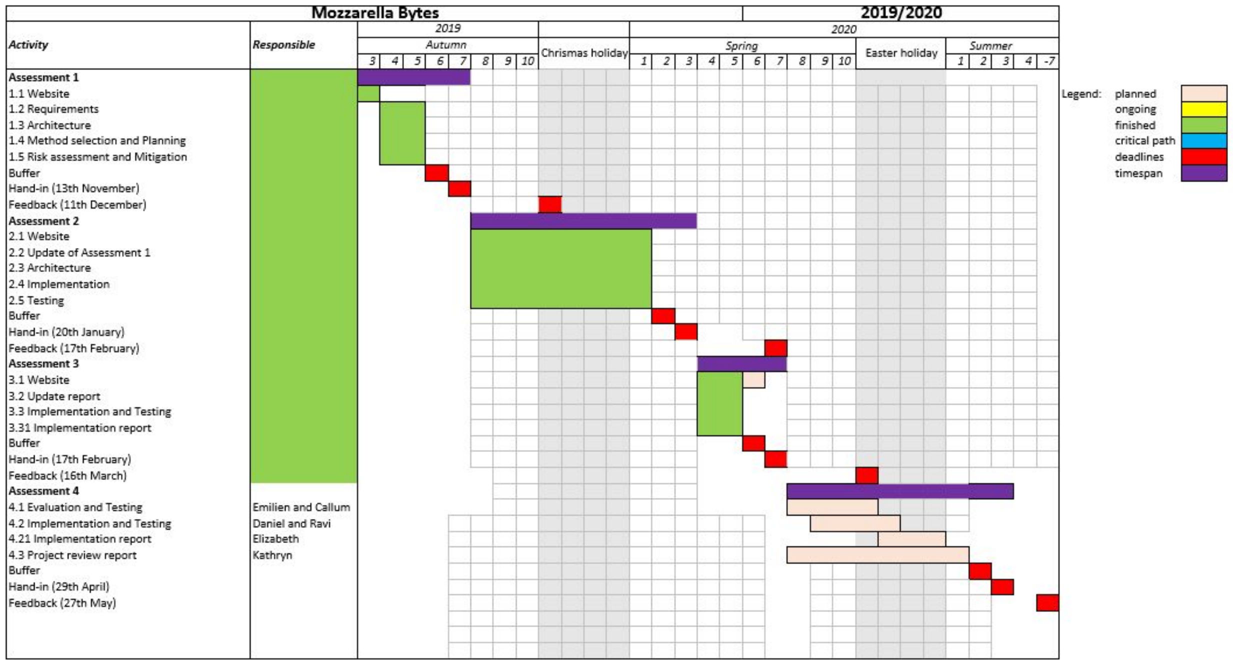
Due to his high levels of focus and ability to spot errors that others can't, Ravi has been assigned the role of Completer Finisher.

These roles have been delegated by the team as a whole and who is most confident having which role in the team; the roles will be maintained throughout the project unless it is seen unfit or the team no longer has the need for that role.

Plan for rest of SEPR (dependencies and critical path)

The most common task dependency is finish-to-start which means the predecessor task needs to be finished before the next task, the successor, can begin [9]. Using this approach, assessment one needs to be completed before assessment two can begin. This includes the updating of the assessment briefs and documentation. In regard to dependency for each individual assessment tasks will run alongside each other as to maintain the plan for finishing a week before the hand-in deadline; however, some tasks do depend on the previous task such as implementation and the implementation report are dependent on each other, as the report cannot be done before the implementation is done.

The future for the project is shown in figure below:



The figure shows the plan for the future of the SEPR project with all the tasks required for each assessment and the duration of each task within the assessment. We aim to finish all our assessments a week before each hand-in date, which is what the buffer represents. By having a buffer, we leave ourselves extra time to redo and go through all of our tasks as a group, as we split up tasks to individual or smaller groups, so we are more efficient in our workload. The key tasks for all of the assessment is laid out on the figure, along which the earliest starting dates and latest finishing dates. **We kept the buffer as we realised that this could be a very useful method of staying ahead of our work load. However, we changed the duration of some of the tasks and the responsibilities to better suit our team.**

For assessment 4, we have used the parts of the project brief and the deliverable requirements to state the tasks on the Gantt chart, and assigned them based on the roles allocated from the Belbin team working model.

References

[1] "libgdx", Libgdx.badlogicgames.com, 2019. [Online]. Available: https://libgdx.badlogicgames.com/. [Accessed: 22- Oct- 2019].

[2]"How to choose an Android game engine: libGDX vs Unity - Dario Penic", Dario Penic, 2019. [Online]. Available: http://dariopenic.com/how-to-choose-an-android-game-engine-libgdx-vs-unity. [Accessed: 24- Oct- 2019].

[3]J. Cook, "LibGDX Game Development By Example", Google Books, 2019. [Online]. Available: https://books.google.co.uk/books?hl=en&lr=&id=uxt1CgAAQBAJ&oi=fnd&pg=PP1&dq=why+use+libgdx+&ots=6L0SGsk3U-&sig=Uf3O5xC6LIbXzkkjvz10ORIKZj0&redir_esc=y#v=onepage&q&f=false. [Accessed: 25- Oct- 2019].

[4]"libgdx/libgdx", GitHub, 2019. [Online]. Available: https://github.com/libgdx/libgdx/wiki/2D-Particle-Editor. [Accessed: 27- Oct- 2019].

[5] A. Springfellow, "Gradle vs Maven", DZone, 2017. [Online]. Available: https://dzone.com/articles/gradle-vs-maven [Accessed: 13- Dec- 2019].

[6]" tiled", Tiled, 2019. [Online]. Available: https://doc.mapeditor.org/en/stable / [Accessed: 13- Dec- 2019].

[7]"The Nine Belbin Team Roles", Belbin.com, 2019. [Online]. Available: https://www.belbin.com/about/belbin-team-roles/. [Accessed: 29- Oct- 2019].

[9]"Understanding Task Dependencies in Project Management", Projectinsight.net, 2019. [Online]. Available: https://www.projectinsight.net/project-management-basics/task-dependencies. [Accessed: 01- Nov- 2019].

[10] " Aseprite", Aseprite, 2019. [Online]. Available: https://www.aseprite.org / [Accessed: 13- Dec- 2019].

[11] ” MagicaVoxel”, MagicaVoxel, 2019. [Online]. Available: https://ephtracy.github.io/ [Accessed: 13- Dec- 2019].