Throughout the progression of the project, the team structure and management progressed with it. There were many changes to the team structure involving role switching. As the project progressed we found that group members were gravitating towards their own preferred roles naturally and we decided that this is how we would get the most out of our group members when it came to motivation.

What we found was, the group naturally split into 2 distinct subgroups **[1]**, a development team and a testing and documentation team. Obviously, group members held their roles and contributed within their subgroups however it made sense to us to let members work on the parts of the project which they felt most confident in.
Regular group reviews ensured that, as the group naturally split into 2 further subteams, it prevented the code development subteam progressing without informing the testing and documentation subteam.

Due to us deciding on a very flat hierarchy from the start, this allowed group members to change jobs easily as each role had reasonably equal responsibility and contribution to the project. The flexibility of the group meant that group members could settle into their preferred concrete roles by late on in the assessment and we saw this occurring a lot during assessment 3. We did, however, ensure that we always kept a soft leader **[2]**. The soft leader did not set tasks, as this was a decision made by the whole group during the group meetings, but they instead ensured regular communication over the messaging application as well as ensuring that the team, as a whole, was on schedule to complete each iteration of the assessment. They also scheduled group meetings and found an appropriate time for all group members to attend. The soft leader ensured all risks were identified early when adding or modifying requirements and implementing them into the game. This helped mitigate any potential time costly risks.

One benefit of the group naturally splitting into 2 logical subteams (development team and testing/documentation team) was that it made completion of tasks more efficient. The reasons for this were, when a change to the code was made and a new feature added, it did not require explanation to all 5 other group members. Because the development team was made up of only 2 group members, whenever a new feature was added it was usually understood by the other developer as a lot of the code changes were done collaboratively. This was also useful as in-person meetings began to happen less and less the further into the assessment iterations we got due to other studies and activities interfering, meaning it would have been difficult to explain a new implementation of a feature to all members of the group. Another benefit of having a smaller development subteam was, there were far fewer clashes on the git repository which needed resolving as the likelihood of 2 group members working on the code simultaneously without knowing the other group members was working on the code was decreased massively. This meant that during assessment 3 and 4 we had barely and clashes to resolve when committing changes to the git repository. In assessment 2 we found that this process was very time costly. The splitting up of the group also allowed a more parallel way of structuring the work on the project because, as one team was developing the code, the other team could be working on documentation and creating tests. This was much more concurrent than when we had all 6 group members cross-collaborating on separate documents and iterations of the code causing confusion. We found that the simpler method favoured the progression of the project in the later assessments so this group layout became concrete.

An issue which arose during assessment 2 and continued throughout the rest of the assessment lifecycle was the gradual decline of contribution from a group member which, by assessment 3, had become comparable to permanent loss of a group member as there was no contribution to the project from this individual. This required us to use the risk mitigation which we had defined in assessment 1 and redistribute the work to the other members of the group. This allowed us to ensure the progression of the project was not delayed very much and it kept the whole team on schedule. The loss of contribution from the group member was dealt with efficiently by the group member who played the lead role as they took on the tasks which were not getting any contribution. This mitigated the risk efficiently and did not affect the completion of other tasks from the other group members.

By the end of the project, the software engineering development method which we used was the agile scrum methodology **[3]**. By the end of the project, the software engineering tools which we used were: LibGdx **[4]**, IntelliJ **[5]**, Google Drive **[6]**, Tiled **[7]**, Atom **[8]**, Facebook Messenger **[9]**, Gmail **[10]**, GitHub **[11]**,

Throughout the progression of the project, we kept the scrum methodology which we selected in assessment 1. We initially chose the scrum methodology as it was much more flexible than XP. It was very useful because, as the project aged, our group roles naturally changed, along with the tasks each group member had to complete, due to a range of different factors. The regular reviews and group reflection on each sprint allowed us all to keep up to date with the progression of the project and the tasks which needed completing regardless of individual group member's contribution to a specific task. We did find that, in assessment 2 and the current assessment (4), group meetings and reviews have been occurring less frequently. This is largely due to the assessment deadlines being situated near exams in January and May. In assessment 2 the assessment ran largely over the Christmas vacation making it difficult to organise a time which suited every group members. We did, however, try to keep group meetings and reviews to a less frequent once a week (excluding during the Christmas vacation period), occasionally 2 if not all group members were present for one of the meetings in the week. We assured that in each meeting we had, there was at least one member of each subteam present so the information from the meeting could be reported back to the respective subteams. Because of the exams timetabled in January and May, we did find ourselves overcompensating and getting ahead of the predicted schedule, made on the Gantt chart **[12],** during assessments 2 and 4.

We found a big challenge was the addition of new requirements and dealing with how they might affect previously implemented requirements, as well as swapping projects with another group and the potential risks this created. We worked around this by regular group meetings and constant updates to documentation.

Throughout the project, the only tool which we changed was out team organisation tool Monday.com **[13]**. We justified this change and discussed the relevant risks in **[14]**. Excluding this, we did not change any tools we used. We did, however, add in some new tools such as Atom for collaborative programming. We also had to use Tiled **[7]** when we switched to another groups project as their game was dependent on this software. We decided instead of removing the use of Tilemaps in the code, it was more efficient to learn how to use the Tiled **[7]** software and how Tilemaps work with Java and LibGdx. We discussed and justified this decision in a group meeting and evidenced it previously in the assessment. By keeping the same team management and communication tools, it meant we didn't have to spend time learning new tools in this area, allowing us to prioritise actual progression of the code and documentation.

As contribution lacked in some areas the Google Drive **[6]**, changelog feature allowed us to see which areas the contribution was lacking allowing us to communicate with certain group members and make sure everyone was clear on what needed to be completed. We found it very useful as group members could not lie about the completion of a task. The tool, as well as all the documents being stored in one place, allowed every group member to be able to access and monitor changes and contribution. This kept us on schedule and allowed us to identify and mitigate the necessary risks very early on in the assessment iterations.

Due to Git **[11]**, showing all commits to our group's repository, it was also used for a similar reason to the one discussed above. It also showed useful when avoiding clashes when modifying code simultaneously. Any clashes that did occur took significant time to resolve, however, due to the Git repository keeping track of all changes made, as well as storing a description of the modifications in the code in the respective commit (defined by the developer who made the commit), this allowed us as a group to understand what caused the clashes and the location of the clashes in the code. Throughout the project, we used Drive documents and sheets to keep checklists on the jobs which needed to be completed as well as changelogs to keep track of any documentation changes. Initially, contribution to the change logs and viewing of the checklists was strong, however, further into the assessments and with exam season approaching, it was clear that motivation and dedication began to lack. To ensure that all group members knew what tasks were left to complete, the progress was constantly discussed in the group meetings so that even the group members who were not regularly checking the checklists and contributing to the changelogs still kept up to date with the progress of the project and were aware of what tasks remained for each assessment iteration.

Bibliography

[1] Warwick Organizational Staff; Organizational Studies: Modes of management
https://books.google.com.au/books?hl=en&lr=&id=zYuPgt4AJysC&oi=fnd&pg=PA262&dq=Managing+teams&ots=rw-ryxZyv1&sig=HUqyrDyRMimxus2lObIQnpm4OLA#v=onepage&q=Managing%20teams&f=false

[2] Training Industry, Soft Leadership Training [Online] Available:
https://trainingindustry.com/articles/leadership/soft-leadership-training/

[3] L. Rising; N.S. Janoff, The Scrum software development process for small teams - IEEE software: Volume: 17 Issue: 4
https://ieeexplore.ieee.org/abstract/document/854065

[4] LibGdx, [Online] Available:
https://libgdx.badlogicgames.com/

[5] JetBrains IntelliJ, [Online] Available:
https://www.jetbrains.com/idea/

[6] Google Drive, Google [Online] Available:
https://www.google.com/drive/

[7] Tiled, [Online] Available:
https://www.mapeditor.org/

[8] Atom, [Online] Available:
https://atom.io/

[9] Facebook Messenger, Facebook [Online] Available:
https://www.messenger.com/

[10] Gmail, [Online] Available:
https://mail.google.com/

[11] Github, [Online] Available:
https://github.com/

[12] Gantt Chart Limewire:
http://limewire.me/docs/assessment1/Gantt%20Chart%201.pdf

[13] Monday, [Online] Available:
https://monday.com/

[14] Updates to methods used - update report assessment 2 Limewire:
http://limewire.me/docs/assessment2/Updates2.pdf