Methods and Planning

Software Engineering Methods & Tools

Development Process

We are using an agile-scrum [0] approach for our software engineering project, as it allows our team to carry out work in a flexible manner, while providing the flexibility to evaluate our progress and plan for the next phase of work as we go along. Details of how we use this methodology to run our meetings and plan tasks are found in the team organisation section.

One of the key tools needed for any software engineering project is a version control system. This is a tool that will allow us to keep track of code changes, and collaborate on the same piece of code. We're using **GitHub**[1] for our code repository, as it's a popular tool that our team members are familiar with.

While developing, we plan to use git's branch and pull request functionality to help us manage multiple developers on the same project. To ensure our code quality is high, we ensure that a different developer performs a code review for every pull request, and that appropriate unit tests have been added and successfully pass before approving the branch to be merged.

We are using test driven development for our acceptance tests, as these are being written before development starts to ensure our code meets the requirements. While development is occurring, unit tests will be added to our code using **JUnit**[2]. This is so that we maintain a high level of test coverage, and have confidence that our code is working as intended. It will help with ensuring regressions do not occur. We are using **CircleCI**[3], a continuous integration server, to run all the unit tests every time a commit is made to GitHub, this will ensure that nobody accidentally breaks the codebase. For more details, please see the testing report.

To manage our development tasks, we plan to use the **GitHub Projects** feature of our code repository. This lets us manage our tasks and code issues with a Kanban board[4], and allows for items to be assigned to team members so everybody knows who's doing what. GitHub also allows us to link issues to the project, and these are useful because team members can collaborate on design decisions and keep track of what's happening in the code. More about our team organisation is discussed later.

Development Tooling

To develop the project, we decided, after much deliberation, to use **Java**[5] and **IntelliJ IDEA**[6] as our programming language and IDE. This is because Java is a language accessible to all members of the team, as well as the entirety of our cohort. We did give serious consideration to other programming languages such as C#[7] with Unity[8] however few in our team have any experience in the language which means it may take longer and cause problems later on in the process.

We needed a library that provides useful game APIs to help speed up our development process. We did some research and found 4 options of libraries we could use: Swing[9], Mini2Dx[10], LibGDX[11], LWJGL[12]. We quickly realised LWJGL and Swing weren't right for our project. In the end we decided to use the **LibGDX** Java game development framework. We chose LibGDX over Mini2DX because LibGDX has support for the IDEs that we intend to use, and it provides features that simplify several game development steps.

Design Processes

We will need to do some designing whilst the development stages are progressing. The main tool we will be using to design the game assets is **Piskel**[13], an online tool that allows us to easily create game assets. We expect designs to start as paper prototypes that will be developed further as the project goes on.

As we'd have to produce a map for our game (see requirement 4.1.1), we were pleased to discover that LibGDX has support for tiled maps. There is useful software called **Tiled** [14] that allows us to create a high quality 2D map and export it for use with the framework. This will save us massive amounts of time compared to alternative solutions, and this can be used to improve gameplay and ensure that there are few bugs.

Collaboration Tools

In addition to using GitHub projects as mentioned above, we also need some communication and collaboration tools to help us keep the project on track.

For the first few weeks we were using **Facebook Messenger**[15] for team communication, but it became apparent that it wasn't up to the job as we kept losing important pieces of information. We decided to switch to **Slack** [16] as it allows us to have separate chat channels for different aspects of the project, letting us keep all necessary information separated. Since separate discussions are split into separate channels, we are able to find important information more easily for future reference. The team engages in daily communication on Slack, we use this to consistently report our progress, ask questions on ambiguous issues and organise future in-person meetings.

We are using integration between our development tools and Slack to help us keep track of progress in one clear feed, in particular the **GitHub** integration and the **CircleCI** integration. This helps with our development method, as it ensures all team members are kept up to date with the status of the repository and test results.

For online meetings we are using **Join.me** [17] Join.me is a free tool with voice, text and video chat as well as a screen sharing feature which allows a presenter to share their screens with other team members. Join.me allows efficient and effective team meetings to take place even whilst all members are in separate locations. We made extensive use of Join.me over Christmas holiday period, where we used it to hold a full team meeting, using our scrum method as described later.

For document storage and collaboration, we are using **Google Drive**[18] with a shared team folder. This contains all of the documents and other files we've been working on. We are making use of **Google Docs, Sheets & Forms**[18] as they allow us to collaboratively work on documents at the same time, as well as providing mechanisms that allow us to review and comments on our documents.

Team Organisation

Roles

We decided early on to have a team leader role. The team leader is responsible for ensuring everybody has a task to be working on, and for making sure progress is being made. They are also responsible for producing meeting plans, and answering queries of any team member.

In Assessment 1, we voted to decide a group leader - Brooke Hatton won as we felt he was a natural leader. For Assessment 2, we decided that Jason Mashinchi would take over as group leader, as he had experience in a software development team. We plan to alternate group leaders throughout the assessments, to give the other member a break.

Every member of the team is assigned tasks to do at the end of every meeting for the sprint ahead. We decide who does what based on who wants to take on the task, or based on previous experience if it's relevant to a task.

Meetings

The team aims to have a meeting at least once a week, during which our team leader acts as our scrum master. We may not strictly follow the agile-scrum rules, but we've found that our approach works well for us.

Our team leader prepares for every meeting by producing a brief plan of what needs to happen during the meeting to ensure that we are making progress. Having a plan means that we can stay on track within the meeting and don't go off track and don't make progress. We tend to keep meetings high-level so that we don't waste time on implementation details that can be decided without the entire group present.

Meetings are scheduled to happen multiple times a week depending on availability of team members. Our meetings always signify the start of a new sprint. We typically start with each member going through what they've been working on in the previous sprint, and they raise any problems or questions they may have. This allows us to catch issues, concerns or blockers that have arisen early on in the meeting, so we can take them into account when planning the next sprint.

We then proceed to discuss what needs to be done during the next sprint, and assign tasks to each individual in the team. This is great because it means everybody has something to be working on for the next week, and ensures that we are making sufficient progress in the project.

During Sprints

Everybody is assigned a task to be working on during sprints, and they are responsible for ensuring their bit of work gets done. Every team member uses the kanban board containing issues on GitHub to keep the rest of the team up to date on where their task is at, and they make sure that their appropriate issues/tasks are assigned to themselves.

If any problems arise during the week, team members use their assigned GitHub issues to discuss anything related to their task, or we communicate through Slack when we need to discuss something more general. Splitting up the communication methods in this manner allows us to find discussion when necessary, as often we make design decisions within GitHub that can be referenced later.

Systematic Plan

Assessment 2

The focus of the team will fully switch to the second assessment on Wednesday 9th November, once the first assessment has been submitted. We plan on completing the second assessment by Tuesday Spring Week 2 giving us a week to account for any unexpected developments or fixing issues that happen to arise. It also gives us time to analyse, criticise and improve our own work thus improving the quality of our code and enhancing the functionality and efficiency. This will also ensure that we have very high quality documentation, once all group members are happy with the readability of our code we plan on requesting that students from other teams look at sections of our code to test readability. There will also be a 2 week rest from SEPR to account for time spent studying for exams and also the exam week itself.

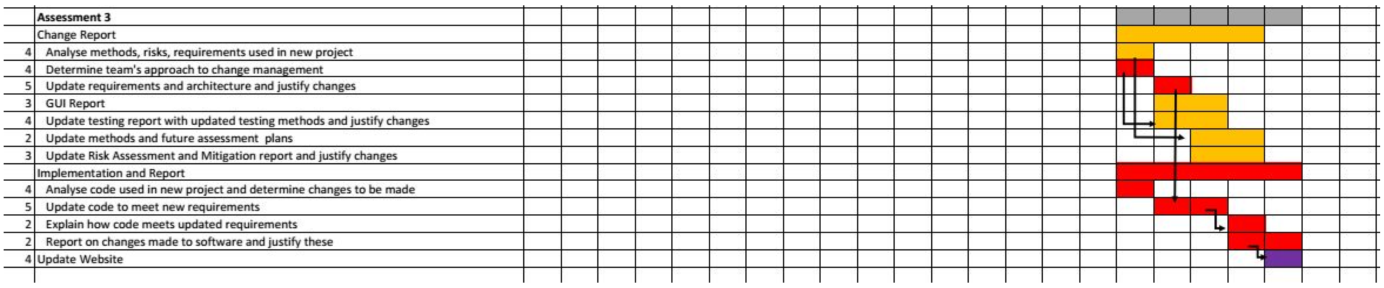
Assessment 3

After the completion of Assessment 2 on Tuesday 24th January we will work as a team to decide the project to take on for Assessment 3. This will be done within one week, after which we shall decide upon areas that need to be worked upon over the next assessment, and delegate tasks accordingly to team members. Documentation is started in the first week with everyone focusing on making sure reports and code documentation are up to date. within the next week all documents will be near to completion and the code will have started to be updated. Again, we plan on completing the work a week before the deadline to give us time to fix any issues found.

A Gantt chart [19] containing the schedule for key tasks from all the assessments can be found in the appendix. This chart includes priorities, task dependencies and a critical path.

Gantt Chart

Below is an extract of our Gantt chart. The full version can be seen on our website under Appendix B (http://lihq.me/Downloads/Assessment1/AppendixB.pdf).



Appendix: Task Assignment Summary

Assessment 1:

- Ben Jarvis was assigned the requirements document
- Benjamin Grahamslaw was assigned the risks and mitigations report
- Brooke Hatton and Jason Mashinchi were assigned the Architecture report
- Joseph Shufflebotham and Vishal Soomaney were assigned the Methods report

Assessment 2:

- Benjamin Grahamslaw was assigned the GUI report & user manual
- Brooke Hatton, Jason Mashinchi & Joe Shufflebotham were responsible for the implementation of the game, testing and the design choices made.
    - Other team members contributed towards implementing smaller features
- Jason Mashinchi was assigned the testing report
- Brooke Hatton & Vishal Soomaney were assigned the architecture report
- Ben Jarvis & Benjamin Grahamslaw were assigned the document updates
- Joe Shufflebotham & Vishal Soomaney were assigned the implementation report

Bibliography

[0] Waterfall to Agile: Flipping the Switch - Bhushan Gupta [Online] Available: http://www.uploads.pnsqc.org/2012/papers/t-21_Gupta_paper.pdf [Accessed 25/10/2016]

[1] Github [Online] www.github.com [Accessed 8/11/2016]

[2] JUnit [Online] http://junit.org [Accessed 22/01/2017]

[3] CircleCI[Online] https://circleci.com/ [Accessed 22/01/2017]

[4] Kanban board [Online] www.github.com [Accessed 8/11/2016]

[5] Java [Online] www.oracle.com [Accessed 8/11/2016]

[6] IntelliJ [Online] www.jetbrains.com [Accessed 8/11/2016]

[7] C# [Online] www.msdn.microsoft.com [Accessed 8/11/2016]

[8] Unity [Online] www.unity.com [Accessed 8/11/2016]

[9] Swing [Online] www.oracle.com [Accessed 8/11/2016]

[10] Mini2DX [Online] www.mini2dx.org [Accessed 8/11/2016]

[11] libGDX [Online] www.libgdx.badlogicgames.com [Accessed 8/11/2016]

[12] LWJGL [Online] www.lwjgl.org [Accessed 8/11/2016]

[13] Piskel [Online] www.piskelapp.com [Accessed 8/11/2016]

[14] Tiled [Online] www.mapeditor.org [Accessed 8/11/2016]

[15] Facebook Messenger [Online] https://en-gb.**messenger**.com/ [Accessed 23/01/2017]

[16] Slack [Online] www.slack.com [Accessed 8/11/2016]

[17] Join.me [Online] www.join.me [Accessed 8/11/2016]

[18] Google Drive [Online] https://www.**google**.com/**drive**/ [Accessed 23/01/2017]

[19] Gantt Chart [Online] http://lihq.me/Downloads/Assessment1/AppendixB.pdf