Planning and Method Selection

(Underlined words are additions, There were no removals for our planning and method selection.) Discussions about these changes are in the updates document.

Software Engineering Methods

Software development, particularly in this case of game development, is inherently iterative. Software features which manifest themselves as mechanics and aspects of the game are continuously tweaked until they fit the original requirements or as the requirements are modified as stakeholders dislike the end result. In addition, our development team is small and composed of students. As such, the selected software engineering method must be flexible with regard to the workflow of students in addition to being resilient to shifting requirements. Most engineering methods hold the assumption that full time developers are working on the project, so to accommodate for students some changes will have to be made. Communication will be key as our team architecture is flat and small, so everyone will need to make the most of their abilities; this requires good communication between the team with regards to tasks and the division of labour. In this way, if someone has prior experience with a particular task, or has a niche skill such as designing assets, those skills they bring will be fully utilised during development.

With these considerations in mind it appeared apt to base our software engineering method off of Scrum - an Agile method. Scrum, being an agile method, can deal with changing requirements, and is also very focused on communication between the team, and on doing rather than documenting - unlike plan-driven methods. IBM's rational unified process would be a difficult fit for game development especially with regard to cultivating requirements due to its use of user stories, which are more apt for solutions that solve problems rather than game development.

While Scrum can deal with flexible requirements it isn't suited to flexible workflows that students require. To solve this, sprints will last only a week at most, and daily scrums would be abandoned in favour of a messaging group to discuss tasks to be worked on. While the sprint would last a week, it wouldn't necessarily require work to be done for each day of that given week. It simply lists tasks for each person that are expected to be completed by that week.

At the end of the sprint, the client is consulted about the changes made, and the feedback is considered when the team prepares for the next sprint. Tasks that were completed are marked off, and the remaining tasks, along with incomplete tasks are reconsidered when planning for the next sprint.

In this way our method is flexible enough for everyone to work at their own pace while being able to manage their other commitments, and ensures that requirements shouldn't cause an issue when changed. The weekly meetings with the client means that feedback is constantly received at every point during the development process, and easily implemented. As it was not feasible to contact the client as a team during the holiday period, an increased frequency of meetings took place after this period in order to go over the all sections of the development process and requirements of the project with the client.

Development and Collaboration Tools

We will use Git as a version control system to manage and distribute our code effectively. Git will allow us to easily create multiple versions of the same code and backtrack to previous versions if necessary, for instance if a bug was introduced that we are struggling to find, and ease group work on the project. This will allow every programmer to each work on their own local repository, and then merge the results to the global repository. The branching abilities of git also mean that we can program multiple derivatives of the program (e.g. to develop different new features) at once, whilst also ensuring that code stays updated across the whole repository.

We will use Github as our Git server provider. It is a reliable service that eases our use of Git, and makes managing our repository simpler and therefore less time consuming. Github also comes with a bug and issue tracker attached to each repository which we plan on using extensively during our development process. This will aid our agile development process, allowing us to easily assign tasks to individual developers after each meeting and track everyone's progress towards their weekly goals; as well as recording any issues that need to be solved at a later date before the release of a production ready solution.

We will be using Google Drive/Docs and associated productivity tools to manage, edit and share all non-code files. This service allows every member of the team access to up-to-date files, and also edit collaboratively and concurrently.

Remote communication will take place via a secure instant messaging program known as Discord, a service which also allows easy audio conferencing for remote work and meetings. This will complement the weekly physical meetings that we are planning as part of our agile development process. An additional communication client, Slack, was (and will be) used in order to speed up the time taken for development as it offered someone useful features such as differing rooms and github-bot integration so all members of the project could be kept updated with the development of the project

Further into the project, we will use a variety of image manipulation software to create assets for our project. As we have decided on using a ‘pixel art’ style, the majority of our asset creation will be done using Piskel. [These assets will then be imported into another piece of image manipulation software such as Photoshop or GIMP to add a ‘sepia’ tone to the image.] Tiled, a map editor, was used in order to speed up the process of asset creation for backgrounds in order to complete the project within a reasonable timeframe.

Our project will also include sounds. For this, we have decided to either use sound generation software to create our sounds and in-game music or to use royalty free sounds to save us time and allow us to focus more on implementation. Copyright-free (CC0) assets, both image files as well as sound files were also included as part of the project in order to make the game more visually appealing

We have also used Teamgantt software to help create our Gantt chart. This software allows us to collaboratively add to and edit our Gantt chart throughout the course of the project. It also includes a way for us to assign tasks to different group members and track the completion of separate tasks.

Team Organisation

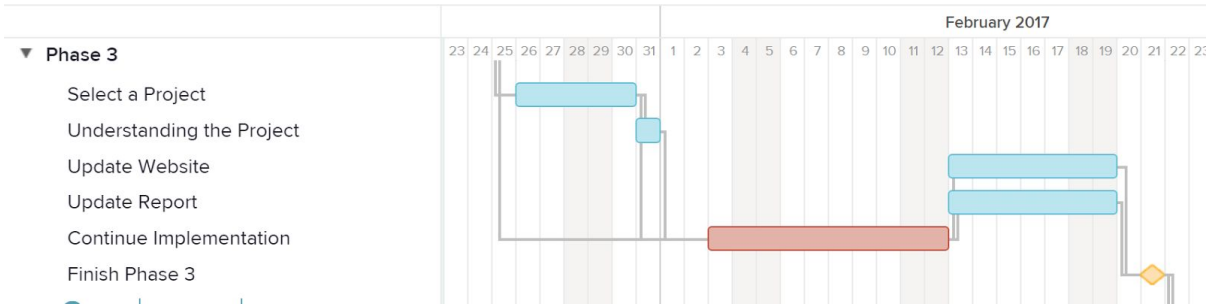
Due to our small team size and limited amount of time to complete the software engineering project, our team's structure consists of a flat team architecture with a single leader to coordinate tasks and organise the team. Team members do not have rigid roles, rather they have fluid roles that aim to make the fullest use of their skills, within the project. The team leader allocates tasks to team members on a short term basis with the consensus of the rest of the team. This will allow members of the team to switch between different roles meaning that people's skills can be put to the best use and team members can learn from others and improve their skills where needed. This structure is in line with the agile development process that we have adopted and plan to keep to throughout the project.

Across Assessment 2, this organisation was modified slightly so there was a subgroup leader for the front end and for the back end to act as a central point of communication for development of these two parts of the project. Other members retained their fluid roles and developed different sections in both parts of the project.

Group meetings are organised using instant messaging and can occur physically or virtually. As outlined in the development and collaboration tools section previous, virtual meetings take place over discord, which is convenient for group meetings that do not require physical interaction or if group members cannot make it to a physical meeting. This allows meetings to be conducted at short notice giving team members a chance to discuss what has been achieved and how we should change our development due to problems that have recently occurred or due to problems that are becoming likely to appear further into the development process.

This team structure is effective for the project because dividing tasks and not having rigid team roles allows for flexibility within the development process. This added flexibility allows team members to spend adequate time on their tasks without interfering with other work they may have from separate modules within the university. In addition, this means that if one part of the development process is accelerating faster than others, resources can be moved and managed efficiently to ensure that all tasks are completed within a reasonable time frame allowing the project to flow smoothly without overworking members of the team.

Development Plan



The full version of the Gantt chart can be seen on the project website.

Phase 2

The following items need to be produced during this phase:

- Website
    - All of the documents produced for this phase
    - Game executable
    - Test Plan and Test Results
    - Game documentation
- Architecture Report
- Implementation
- GUI Report
- Testing Report
- Updated Requirements, Plans and Risk Assessments

Phase 3

The following items need to be produced during this phase:

- Updated Website/Documentation like phase 2
- Change Report
- Implementation and Report

Phase 4

The following items need to be produced during the phase:

- Final Architecture and Traceability Report (Requirements Report)
- Testing and Evaluation
- Implementation and Report
- Project Review