Method Selection and Planning

Group 4 - Cohort 2

Theo Coleman

Matias Duplock

Mohamed Eljak

Katie Schilling

Juliet Urquhart

Nora Wu

Software Engineering Methods and Collaboration Tools

We used plan-driven methods to develop our project, as the product brief meant that our requirements were unlikely to change in a major way, even after the client interview, and so it would be most effective to have a clearly defined path for our project development. Plan-driven methods work best in stable environments, and as our requirements were outlined and identified from the very beginning of our project, this was the most suitable engineering approach for us to take.

By structuring our client interview around clarifying areas of the brief, we were then able to centre our plan around the stakeholders of our project, as the requirements and tasks would be elicited directly from the product brief and the results of our client interview.

We considered using an agile methodology to develop our project in case requirements or features of the game were to change (or need to be changed) during the development process. However, we decided that this risk wasn't enough of an instability to justify the use of agile methods, as this would also mean our documentation would be more difficult to complete during the development process and we would require regular and constant feedback from the client, which wasn't a necessary feature of our project. Ultimately, we decided agile methods would not be relevant or useful to our development.

We utilised a number of collaborative tools throughout our project development. Our approach was to initially work independently, but on a shared/collaborative site, and then to meet regularly to discuss the work we'd done and implement our efforts into one.

Shared Drive

The first collaboration tool we set up was a shared Google drive to hold all of the key documents relating to the project. This included planning documents (on which we brainstormed ideas as a group), the client questions and answers from our interview, the assessment deliverables and a Gantt chart created by Katie to track the progress of the project and manage identified tasks.

Shared documents were particularly useful, as we would work on these collaboratively during group meetings to compile our ideas. We could then refer back to these documents later to develop and add more detail (individually and in our own time), and also to draw from these plans to use in other areas of the project development.

Katie created the documents for each of the assessment deliverables with just the layout and criteria from each section in the documents. Once we had assigned each group member ownership to a deliverable, they could then work straight onto these shared documents, so that everyone in the group could see the progress of their section. This allowed for constant and immediate feedback and discussion on the project documentation. It was an effective system, as it meant we did not have to submit our work to each other and wait for feedback (or vice versa, request someone's work so that we could give them feedback). Instead, all of the work and its progress was immediately available to every member of the group, as we all had access to the shared drive, which meant that feedback and collaboration was possible at every point throughout the development process.

Gantt Chart

To outline our plan-driven approach to development, Katie created a Gantt chart, which was accessible by the whole team on the shared drive. As detailed later in this document and on our project website, the chart identified our key tasks (divided into Planning, Game Development and Documentation sections), and assigned owners and dates to each of the tasks. It also displayed the timeline of the project, allowing everyone in the team to visualise

the deadlines for each task. It made the dependencies of the project clear, which allowed us to effectively prioritise our work and carry out project development in the most efficient manner possible. The Gantt chart also allowed us to adjust or add to the tasks we had outlined, allowing for some flexibility and the continuous development of our project plan.

GitHub

Theo programmed the foundation code for the simulator, which he then shared with the team through a shared GitHub repository created by Mohamed. We held a group meeting in which Theo presented his core/foundation code to us. He had identified areas of the code that required development and outlined tasks of any requirements he hadn't yet implemented into the program. From this list, we each took on tasks to individually work on before our next meeting. Using the repository, we each cloned Theo's code and created our own branches to start working on our assigned areas of the code.

During the subsequent meetings, we shared our individual branches with the group, and helped one another to complete and submit these assigned tasks to the overall code. Theo then combined the collaborations so that they were all part of one programme project.

Approach to Team Organisation

We recognised early on that we all had different strengths that were relevant to the different areas of the project. However, we also knew that it was important for the workload to be divided equally between us. Our approach was to identify and outline our key tasks for the project - this systematic plan can be seen later in this document - and the key tasks were then categorised. We then each volunteered ourselves to manage, lead and/or complete certain tasks. For the tasks that required more work, such as the larger documentation sections, another group member would be assigned to this task. This was a useful approach, as even if the group member wasn't hugely confident with that task, they had another group member who was confident to work alongside. It also meant that the confident group member could have some support with the workload, making the tasks easier to manage for everyone.

The categories the tasks came under were 'Planning', 'Game Development' and 'Documentation'. The planning tasks were mostly initial, collaborative activities that we completed during group meetings. As the other two categories were identifying integral areas of the project, we decided to assign group members that would oversee and manage all of the tasks under that category. This meant that there would be someone keeping track and checking updates and changes made by other group members, ensuring that everyone was kept on task and working to deadlines, rather than everyone simply relying on each other to keep up to speed. Theo volunteered to oversee the 'Game Development' tasks, and Katie volunteered to manage the 'Documentation' tasks.

As the project required a good amount of development and write-up, dividing the tasks between us to work independently (or in pairs) and then later come together to combine our work was the most efficient approach to project organisation. The way we outlined the tasks also ensured that each task didn't depend too strongly on another, or on multiple other tasks. The progress of our project would be more efficient if we could complete our assigned tasks concurrently. Therefore, the category owners established the foundation of the tasks so that the other group members could then all complete their tasks at the same time, ensuring we completed a lot of the workload between us in good time.

For example, Theo programmed the base and foundation of the code for the game, and then assigned each of us areas of the game to work on. We then went away and individually programmed our sections by creating new branches in the GitHub repository created by Mohamed. As category leader, Theo kept track of people's progress and then combined our work into one program. Katie created the documents for each of the written deliverables by adding the title page and the criteria for the section to each document. Again, we worked independently (or in pairs, where relevant), on our sections, and Katie checked in on the progress and correctly formatted the sections of the documentation where necessary.

Systematic Plan for the Project

Programming Tasks

Task	Start date	Finish date	Owner(s)	Prioritisation	Dependency
Foundation Code	11/10	14/10	Theo	High	n/a
Building on the foundation	14/10	17/10	Matias	High	Foundation Code
Widget programming	14/10	21/10	Juliet	Medium	Foundation Code
Programming user variables	14/10	09/11	Theo	Medium	Foundation Code
Buildings menu	14/10	15/10	Nora	Medium	Foundation Code
Placing buildings	14/10	17/10	Katie	Medium	Foundation Code
Timer	14/10	21/10	Mohamed	Medium	Foundation Code
Music and sound effects	14/10	21/10	Mohamed	Medium	Foundation Code
Building Interaction	04/11	09/11	Matias	Medium	Placing buildings
Pause Screen	04/11	05/11	Nora	Medium	Building on the foundation
Settings and Load Screens	04/11	07/11	Theo	Medium	Building on the foundation
Adjusting Buildings Menu	04/11	05/11	Katie	Medium	Buildings menu

Documentation Tasks

Task	Start date	Finish date	Owner	Prioritisation	Dependency
Website	14/10	10/11	Mohamed	Low	Deliverables
Requirements	14/10	10/11	Katie and Juliet	Medium	Client Interview
Architecture	14/10	09/11	Juliet and Nora	Medium	Game Development

Method selection and planning	14/10	04/11	Katie	Medium	n/a
Risk assessment and mitigation	14/10	10/11	Mohamed	Medium	Game Development
Implementation	14/10	01/11	Nora	Medium/High	Game Development

Teaching Weeks

Week 2

Week 2 was the first week of our project development, and so we focused purely on the planning aspects. We read through the product brief, and discussed and made notes on key requirements and tasks. From these notes, we created our client interview questions in order to clarify and elicit actual requirements from the user. We booked a slot for the interview, which we would be carrying out the following week.

Week 3

Our interview session was at the beginning of this week, and so we were able to outline a clear plan from this point onwards. From the results of the interview and from reviewing the brief, we listed and categorised key tasks that we identified. We then also had a group member overseeing each of the categories. Theo began working on the foundation of the code for the game and from this, assigned additional tasks to each group member to work on programming independently, which became tasks within our project plan.

Week 4

During week 4, we focused on programming our assigned tasks, and discussed our work in the group meetings. Members who had smaller areas of code to work on, or who were not as confident in this area of development, were assigned larger roles in the ‘Documentation’ category, and began working on their tasks within this section.

Week 5

Ahead of consolidation week, we started to focus more on the documentation section of our project development. We set the working deadline to after consolidation week to ensure that each group member had something to work on and make efficient progress with during the week. This plan ensured that we would return to teaching at a good stage of our project development, ready to start bringing all of our work properly together.

Consolidation Week

We took the opportunity of Consolidation Week to work individually on our assigned tasks, particularly our areas of the documentation. We maintained contact on our project group chat, updating each other on our progress throughout the week, outlining what still needed to be done and what we should address in our first meeting when teaching resumed.

Week 6

During the final working week before the project hand-in, we worked closely on finishing off our tasks. We caught up on the progress we each made over Consolidation Week and then outlined which tasks needed finalising over the week. We divided the final few features of the code between us, setting a tight deadline to complete these tasks, so the game development could be finished promptly. We also ensured that each section of the documentation was close to completion and everyone was aware of what needed to be finalised, with the agreed goal of having everything completed ahead of the hand-in deadline.