

Methods and Planning

Group 15

HesHus

Adam Robinson

Tsveta Ivanova

April Jack

Oliver Goodwin

Luke Jackson

Luca Hammond

Chris Rolfe

Software Engineering Methods

General

We are using GitHub as a central repository for our project. This allows us to better manage our artefacts and to track development so work is divided fairly and helps to manage version control meaning that if two people are working on the same piece of code then the changes each person made are merged as far as possible (rather than one person's code being completely overwritten.) This results in better collaboration between team members.

We have devised a weekly system to learn relevant technologies and share tutorials in a shared folder that everyone can use. In each meeting, we outline a goal for the subsequent week, like being able to push and pull from the terminal using GitBash or being able to make a simple game from a tutorial using LibGDX. We adopt this system into a Scrum format which we do during our weekly meets. This allows everyone to be familiar with any new technologies they may need to use, particularly the implementation team.

This method also allows us to shadow each other better - everyone is familiar with the basics of every section of the project so people are able to step in if required in case a team member cannot work for any reason (e.g. they are ill.)

Website

We used GitHub pages to develop our website since it was convenient to do since we were already using GitHub. It provides preset formatting for our website and allows for all members to access and edit it.

Requirements

We made sure to spend a lot of time thinking about our user requirements and made it a key topic to discuss early on, familiarising ourselves with the assessment brief so as to better understand what the client was asking of us. After the initial meeting, we kept in regular contact (roughly once a week) with the client, so we could discuss the accuracy of the elicited user requirements. The requirements team made sure to understand relevant diagrams from the lectures, such as functional and nonfunctional requirement tables. Entries in these tables clearly indicate what the user wants and how the system should be designed around this, helping our architecture team to better design the system.

Risk Assessment

The risk assessment splits risks into three categories (project, product and business), depending on what area of the overall project it affects. It then assigns a likelihood and severity to the risk, as well as any methods that could help to mitigate it. This is a useful thing for the team to refer to when implementing the project, as well as if one of these risks happen.

Architecture

In terms of developing the system architecture, we first researched the basic layout of structural diagrams for games to get an idea of how to structure ours. We used PlantUML to develop these diagrams to indicate the workings and relations of our system components. These diagrams include a class diagram which identifies the key components of the game and their relations to other components (e.g. players and NPCs are classified as objects), a sequence diagram that outlines how the user interacts with the functionalities of the game, and a state diagram that outlines the different states of the avatar (idle, moving, etc). This gives a clear indication to the implementation team on what they need to implement (classes, methods, etc.) We revised and trimmed our class diagrams as we began to implement the project due to libGDX containing prebuild versions of these classes. Eventually we derived a simple version of the classes that we needed to implement ourselves, which greatly simplified what we needed to do and how these classes needed to interact with each other. From this we created equally simple sequence and state diagrams.

Implementation

We chose to use LibGDX, and by extension IntelliJ and Gradle, to develop our game since it provides lots of the basic framework and functionalities we need for our project, helping to reduce the workload and time restrictions that would come from developing it all ourselves. Before starting to write our code, we looked at games that were similar to our project requirements to get an idea of how to implement them. To choose our assets, we looked at Kenney, a platform for open source 2D assets and chose what we thought looked best for our game. It was more convenient than finding individual assets on their own. We used Tiled to create our map of the university and Piskel to create our avatar and buildings. These were used as we considered them to be the simplest to learn whilst still containing all the relevant functionalities, helping us to work most effectively within the time constraints. For collisions and game physics, we considered using Box2D but decided against it, due to the Rectangle engine supplied by LibGDX already containing all the necessary modules. We made sure to iteratively test our game, making several small changes to our project over a few large ones, since it would be easier to rollback changes that broke our build. We also made sure that the entire group tested our game to make sure it could run on different types of desktop (Windows/Linux, etc.) As we developed the project, we messaged our client regularly to enquire about small technical details that came up like movement speed and time/energy costs for activities.

Team Organisation

Our group uses Whatsapp and Slack to communicate. Whatsapp is for general communication and admin across the whole team whereas Slack is used for communication relating to a specific deliverable, as well as online meetings between specific groups. To share documents, we use Google Drive as many of the team were already familiar with it and it is convenient for multiple people working on the same document at once.

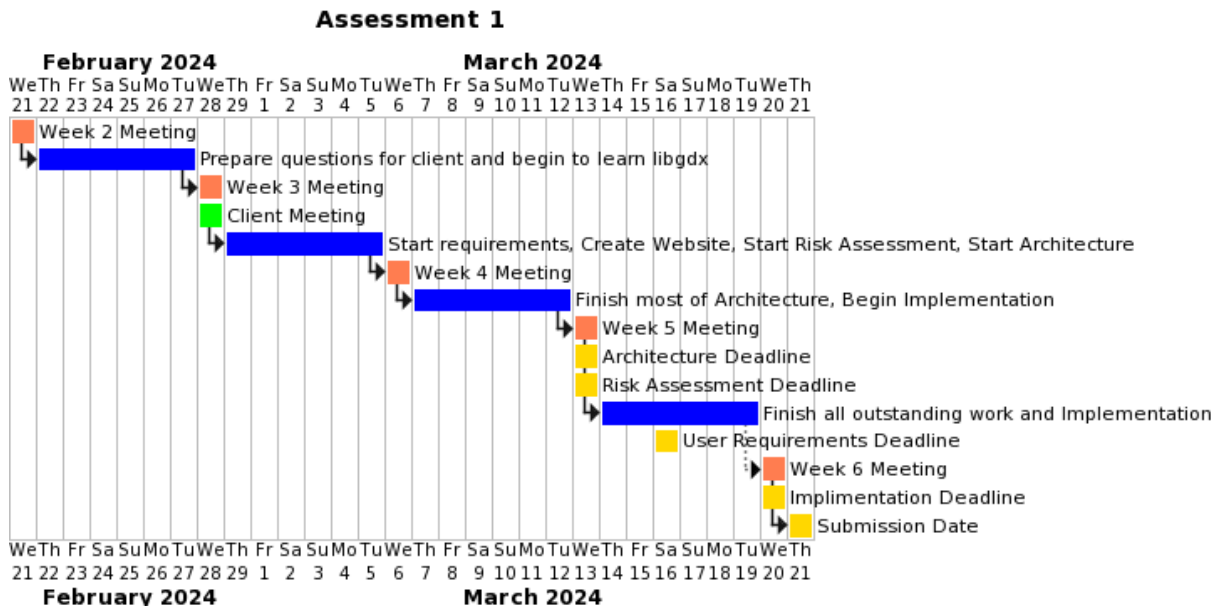
Our group meets weekly during our Wednesday practical session. We first do a scrum to report on the individual processes we have worked on over the last week, what has and hasn't been completed and how we will adjust work in response to the given circumstances. Then we discuss the current state of the project and how to best use our time. We try to apply the previous week's lecture content to our project, as by this point everyone will be familiar with it. At the end of the session we write a summary report indicating what was accomplished and by who. We then complete our scrum by outlining individual tasks for each group member to complete for the following week, and group tasks that the relevant set of people need to complete. We conduct a risk assessment so that we know what can go wrong when doing this, and how to account for this.

Meetings are also held outside of this practical session, typically between people working on a specific deliverable. A summary report is also recorded for every group meeting. We have a large gantt chart which is used to keep track of key dates and deadlines for each deliverable, as well as the project as a whole.

In terms of allocating work, we have tried to allocate an equal amount of marks (~12 marks per person) whilst catering to individual strengths. For example, some members of our group have prior experience with LibGDX and so were deemed more suitable for doing the implementation of the project. We also decided that it was generally a good idea for most of the people doing implementation to also work on architecture as these people would then be more in tune with the structure of the project.

Systematic Plan

- To begin planning the project, we created a gantt chart (see below) containing key dates and information, such as the start and end dates for certain deliverables . The chart was updated week by week, based on new information that would be discussed in meetings.



- At each meeting, we would create a record of the meeting, describing who was at the meeting and when and where it was. It would then detail the key agendas of the meeting, what was discussed and what each member would do in and after the meeting, as well as a report on the progress on the tasks. An example of one of these records can be seen below:

MEETING RECORD			
Date:13/03/2024		Place: CSE/069	
Present: Adam, April, Chris, Luke, Oliver, Tsveta, Luca			
Agenda: <ul style="list-style-type: none">• Discuss the implementation of the project• Work on the outline of team organisation and software engineering• Seek clarification on risk assessment			
Summary of the Discussion: <ul style="list-style-type: none">• Outlined how to best work on the implementation• Spoke to the module leader about questions related to risk assessment			
Reports on Progress to Date:			
Action List			
Action No	Owner	Short description of action	Status:
1.	Luca	Proofread and trimmed outline of software engineering and team organisation	Completed
2.	Implemen	Set a meeting time to discuss further how to approach implementation	Completed
3.	Adam	Updated the requirements table in response to the new user requirements	Completed
4.	Adam	Worked on method report	Ongoing
5.	Implemen	Discussed and compared LibGDX prototypes and how the sub-group can use tiled	Completed
6.	April	Worked on entity-component diagram	Ongoing
7.	Chris		Ongoing

8.	Oli	Worked on and changed parts of the risk assessment based on clarification Worked on logic for the implementation	<i>Ongoing</i>
----	-----	---	----------------

-

SUMMARY OF RESPONSIBILITIES		
Phase of work	Owner	Contributors <i>Give a short account of who did what</i>
Intention	All	Continue to work on given project section and report on progress made

-

- After our weekly meetings, we would also perform a 'scrum', outlining any work to be done in between our weekly meetings, as well as any risks the work might have. An example of a scrum writeup can be seen below:

Week 2 Scrum			
Date: 21/2/24		Place: CSE/069	
Present: Adam, Tsveta, April, Oliver, Luca, Luke			
Completed: <ul style="list-style-type: none">● Researched about the basics of using GitHub			
Outstanding: <ul style="list-style-type: none">● -			
Adjusted: <ul style="list-style-type: none">● -			
Individual Sprints			
Action	Owner/s	Description	Risks

-

Group Sprints			
Action	Owners	Description	Risks
Research technologies for project	All	Research the fundamentals of LibGDX, IntelliJ and Gradle	Struggling with the basics - check the tutorial links Adam put on the Google Drive or message Tsveta/Luke as they have prior experience
Consider questions for client interview	All	Come up with questions to ask in the user interview next week	Multiple people coming up with the same sort of question - write questions in a Google Doc to prevent this

- All meeting records and scrums can be found on the website.