Team Fantasy

Design report

Timothy Runhaar
t.p.runhaar@student.utwente.nl
s2509148

Tim van de Wetering
t.vandewetering@student.utwente.nl
s2858460

Wouter v.d. Boer
w.a.a.vandenboer@student.utwente.nl
s2186969

Abdel Bel Makhfi
a.a.belmakhfi@student.utwente.nl
s2334585

Yifan Sun
y.sun-3@student.utwente.nl
s2594544

Tycho Dubbeling
t.b.dubbeling@student.utwente.nl
s2795825

November 12, 2023

Supervised by Y.D.C. Barrios Fleitas
-
University of Twente
Bsc. Technical Computer Science
Design project Q1 2023 - 2024

Contents

1 terms and definitions

Sidescroller is a 2D game where the player-controlled object is viewed from the side and primarily moves over the screen horizontally.

Action game is a combat-focused game where all components of the combat flow in real-time, as opposed to turn-based games.

Roguelike is a sub-genre of video games where a loss condition will reset all player progress. Often characterized by randomized obstacles/levels and rewards.

Co-op is a multiplayer game where players work together toward a common goal.

2 Acknowledgements

We would like to express our gratitude to Y.D.C. Barrios Fleitas (Yeray), who served as both our supervisor and client. Y.D.C. Barrios Fleitas let the team extensively explore ideas and approaches, while holding it to high standards. This blend of accountability, constructive criticism and encouragement helped the members of this project group successfully complete an ambitious project in the short time of one module.

3 Introduction

This report describes the system design of Team Fantasy, a game specifically designed for researching team effectiveness. Sections describing the process, system requirements and user guides are also included. Team Fantasy was designed and developed during the BSc. Technical Computer Science Design project module at the University of Twente during quartile 1 of the 2023 - 2024 academic year. Y.D.C. Barrios Fleitas served as both the client and supervisor for the team.

4 Domain Analysis

4.1 Introduction to the domain

The domain of our system revolves around the creation of a specialized game designed to assess and evaluate team performance. In contrast to many traditional games, which often prioritize individual enjoyment and entertainment, our focus is distinct. We have set out to develop a game that places an emphasis on the significance of teamwork, making it the only way to beat the game. In traditional gaming experiences, players may not necessarily rely heavily on their

teammates to achieve victory, as the primary goal often centres around individual performance. In our case, the game's core design philosophy is centred on creating and measuring effective collaboration within a team. Success in this game is linked to the ability of players to coordinate their efforts, communicate to each other, and adapt to evolving challenges as a cohesive unit. One of our primary goals in crafting this game was to ensure inclusivity and accessibility for everyone. We wanted to create an environment where anyone, regardless of their level of gaming expertise, could engage in the gameplay. This commitment to inclusivity not only levels the playing field but also minimizes the influence of prior game knowledge on the data we collect for evaluation later on.

Our game is designed to be approachable, providing a low barrier to entry. This means that both experienced gamers and those who have never engaged with video games before can seamlessly pick it up and play together. The intuitive design and gameplay mechanics ensure that the learning curve is gentle, thus allowing for a broad and diverse player base.

4.2 Client, users and interested parties

The versatility of our game is a core feature that sets it apart. It serves as a versatile tool for evaluating teamwork performance in a wide range of settings, extending its applicability beyond recreational gaming. Whether you're in a professional workspace, an educational institution, amongst friends, or within your family, this game provides a unique platform to assess your collaborative abilities as a team in a simple and effective way.

Upon the completion of a game with a group of five players, a lot of valuable data is generated. This data holds significant potential for researchers, allowing them to draw conclusions and conduct in-depth analyses. By studying the gameplay patterns, strategies, and interactions among players, researchers can gain valuable insights into teamwork dynamics, ultimately contributing to a better understanding of how teams function and how they can be optimized for success.

To ensure immediate feedback and self-assessment, our game is equipped with a feature that presents users with graphs of their team's performance. These post-game statistics offer a visual representation of their strengths and areas for improvement. Users can reflect on their own contributions and learn from the data presented to make informed adjustments for future games.

While self-reflection is an important aspect, the primary objective of our game is to facilitate research and analysis by experts in the field. The game is designed to capture a comprehensive dataset that researchers can use to tailor their inquiries, hypotheses, and analyses according to their specific needs. This approach empowers researchers to leverage the game as a robust tool for studying teamwork dynamics across a variety of scenarios in the game.

4.3 Existing Solutions

It's worth acknowledging that several popular games like League of Legends and Overwatch already have successfully incorporated teamwork as a central aspect of their gameplay, involving teams of five players. These games also provide users with data that could potentially be used for analysis. However, there are distinct differences that set our game apart, making it an ideal choice for researchers and team performance evaluators.

In games such as League of Legends and Overwatch, while data is available, it is often limited to a certain set of metrics. The way these games are structured and how they evolve in the future is also predominantly under the control of their developers. This means that researchers have little influence over how the gameplay can be adapted to enhance data collection or improve data reliability. Additionally, researchers often lack the autonomy to determine which specific data points are recorded and accessible for analysis.

This inherent lack of control can be a significant hindrance to researchers seeking to tailor their analyses to specific research questions or objectives. It can limit the depth of insights that can be drawn from the data, as researchers are bound by the constraints and directions set by the game developers.

This limitation enhances the significance of having a game created specifically to serve the goal of comprehensive team performance evaluation. In our game, we've crafted an environment where researchers could actively participate in shaping the game dynamics and data collection process. This flexibility allows for the optimization of gameplay to improve data accuracy and relevance to the research objectives.

Furthermore, researchers have the unique opportunity to dictate which data points they would like to record, granting them a level of control and customization that is seldom available in existing games. Our game's design prioritizes the empowerment of researchers, enabling them to collect and analyze data in ways that align perfectly with their research needs and objectives.

This is the most important reason why we've embarked on the creation of this game. We aim to provide researchers with an ideal platform that not only offers the benefits of team-focused gameplay but also empowers them with the ability to shape and fine-tune the research process itself. The result is a powerful tool that can revolutionize the study of teamwork dynamics in various ways.

5 System Specification

Much thought was put into selecting the right tools. This was a difficult task as most team members had no experience developing video games. Tools had to be picked such that all members could make valuable contributions and so a real, valuable, product could be created within the 10 weeks. The considered options fell into two categories; a fully fledged game engine or a programming language accompanied with the necessary libraries. Languages such as C++, Python and Java were considered. Java seemed to be the language most common to all

(used extensively during the Software Systems module). For any possible game engine, we felt it was important to pick one on which there are abundant online resources such as documentation and tutorials. The team felt Unity, Godot and Unreal Engine ticked these boxes. Eventually, Godot was picked as it looked to be relatively easy to learn and had good 2D game development support. Team Fantasy is to be structured in such a way that inter-role dynamics have a big impact on the experience and success of a group of players. Teamwork needs to be essential to a group of players' success. Some inspirations for the design of the combat and movement were the flash (engine) games "epic war 5" and "Siegius".

5.1 Game Design

Team Fantasy has a number of requirements to it that will be elaborated below. In this section we wish to give an overview of what was chosen an why. This section was originally written for the project proposal, which was created in order to properly discuss, with our client, the product we intended to build. Significant edits have of course been made.

Team Fantasy, aside from the product requirements, also needed to fit within the scope of a single Design Project (10 weeks). This means that there had to be maximum complexity to the first implementations of features that are essential to the product. Brainstorming possible video game types the team came up with a 2D setting, in a co-op multiplayer. The latter was essential given the fact that team effectiveness has to play a key-role in the gameplay. The 2D setting was chosen simply as a result of the compact schedule and lack of experience of most/all team members. The general setting was then eventually specified to become "Roguelike". The roguelike genre takes inspiration from the game Rogue, which game was released in 1980. The team of 5 characters is in a procedurally generated world, where the team is to move collectively towards the right side of the playing area (sidescroller), encountering more enemies. Team Fantasy is an action game, opposed to a turn based game. A turn based setting was considered thoroughly, but not used because, after discussion with our client, the pressure a shortage of time may create while having to actively control something (a character for instance), was deemed more likely to create a team working environment with pressure and stress. Allowing the team to be tested more deeply. Different levels, (often referred to as stages within Team Fantasy) mean different levels of difficulty, changed by the number of enemies being spawned for instance. Team Fantasy is to be played with 5 players and thus 5 different characters were developed. It was essential that the team was dependant on each other. Such that there was a true reliance for team dynamics, all though some players are given the option for "selfish decisions" which may pay off for the entire team. These are to create moments of conflict between players. Below we will give a more specific description of the gameplay mechanics and the different characters.

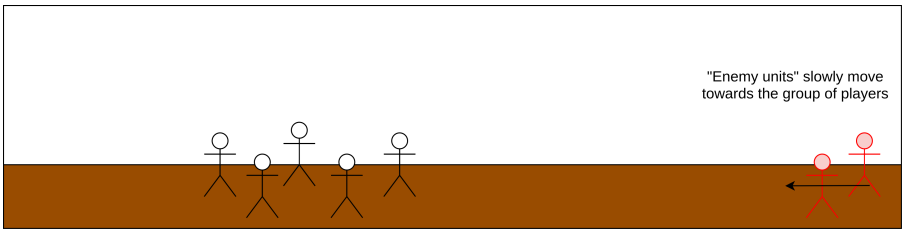

Figure 1: Movement of characters (first draft)

Figure 2: Movement of characters (final version)

5.1.1 Overview

Players will be moving on a 2D plane. Their objective will be to survive getting attacked by a wave of enemies which will attempt to defeat the players. When the players survive a wave they will be rewarded with options for treasure. Players will not be able to move through enemies and vice versa. The 2D plane of movement restricts the enemies as well, meaning that particularly large (and potentially powerful) enemies will be unable to be able to pass the player(s) at the front line. Unless given players are beaten. We believe such a design will play into the class system for this game, as players are reliant on the abilities of their peers, the different characters players may choose will be discussed next.

One of the reasons for the game being 2D is for the ease of development. The game being 2D requires taking fewer dimensions of movement into account as compared to a 3D game. Furthermore, there are many free assets available for sprite-based games, which can potentially be supplemented with our own artwork. A 3D game would require learning 3D modelling to be able to add our own artwork.

5.1.2 Characters

At the start of each game, players will need to choose their characters. For effective gameplay, this will need to become a team activity where the right choices need to be made collectively.

Guardian The guardian will be able to take much more damage than most other classes. The guardian will deal low damage per second himself but protect other classes from getting overwhelmed by enemies. The guardian will get overwhelmed by enemies if not supported properly by other classes and can only attack enemies that are physically close to him.

Figure 3: Guardian character sprite

Archer The archer will have relatively weak defences, meaning the archer will have to be defended by the guardian. The archer will have relatively quick attacks that can be used at long range, though individual attacks will do middling damage. This should make the archer well-suited for quickly reducing the amount of low defence/health enemies present. The archer will also be able to quickly pick off small enemies that manage to slip past the guardian, preventing said enemies from doing much damage in the backline.

Figure 4: Archer character sprite

Sorcerer The sorcerer will be the class with the weakest defence of all proposed classes. The sorcerer will be a slow attacker that will either do significant amounts of damage with each hit or damage an entire area. The sorcerer relies on protection by other classes during their cooldown period between attacks. The sorcerer should be specialised in taking down strong enemies. The attacks will be long/medium-range.

Figure 5: Sorcerer character sprite

Thief The thief will get an option before the start of every level/wave; the thief may either fight together with the rest of the team or separate from his team and fight his own wave of enemies. If the thief survives their (smaller) wave of enemies, the entire team wins. If the thief loses, but the rest of the team survives their wave, the entire team still wins. This would obviously be harder for the team as they are short on one player. If both the thief and the rest of the team lose, it counts as a full loss. As such, splitting off from the main party has the trade-off between getting two chances (the thief wave and the main wave of enemies) or fighting the main wave with the full force of the team. The thief will have less defence than the guardian, but better defences than the archer and the sorcerer. Their damage properties will be similar to that of the archer, but the range of the thief will be rather short (like the guardian). The thief will move faster than all other classes. Stretch goal: allow the thief a broader range of movement options.

Figure 6: Thief character sprite

Soldier The soldier will be a class with weaker defence than the guardian, but stronger defence than any other class. Its strength will primarily be stunning/disabling enemies through attacks. This means that the soldier will render enemies unable to attack or move, preventing them from running past the frontline defences. The soldier will do middling damage per second and will have some limited area-based abilities/attacks. Its main role is protecting the guardian from some damage and keeping groups of enemies from overwhelming the players.

Figure 7: Soldier character sprite

5.1.3 Leader and Items

At the start of a game, the players will vote for a player to be the leader of the team. After the players survive a wave of enemies, the leader will get a choice between three items. The leader will get to assign the item he picked to one of the players. These items will have an effect on the player they're assigned to, with some effects fitting some classes better than others. [If the thief defeats their wave of enemies before the main wave has been beaten, the thief gets to pick and assign an item]. These items will have positive effects on the players. As such, every wave will become gradually more difficult to compensate. We believe this dynamic to be interesting, as the leader is democratically elected, yet has the only tangible say in who gets what item. We expect this to create a certain responsibility for the leader even while the choice is solely in the hands of the leader after they have been elected earlier.

There will be multiple enemy types to prevent there being obvious optimal item choices to pick. All classes will have a "basic attack" which they will be able to use frequently and abilities that are on a cool-down. The abilities will either be stronger than basic attacks or have a special effect that makes it stand out from normal attacks.

6 Planning

Much needs to happen during the span of 10 weeks. During this period it is extremely important to stay on track as time is limited. We will define a number of deliverables in order to structure our planning. Scrum/Agile sprints were created given these deliverables as end goals for each one week sprint respectively.

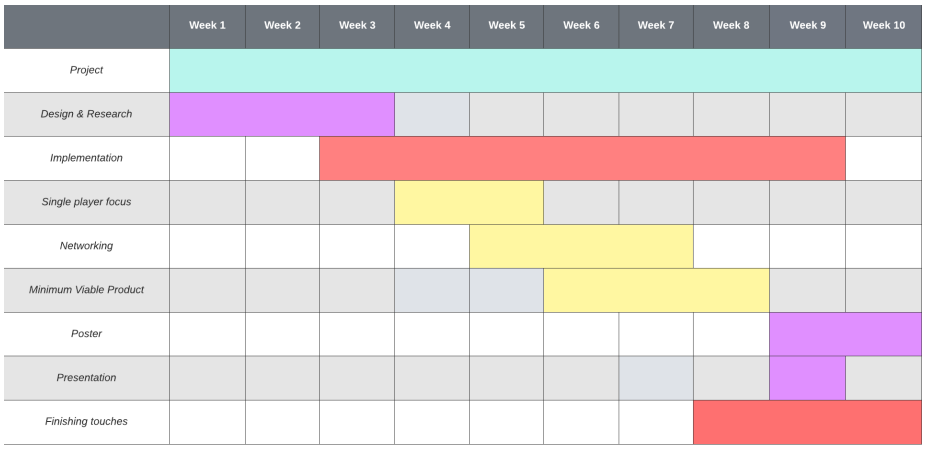

Figure 8: Planning Gantt chart

6.1 Deliverables

Please note that we aim to update the design report daily according to new developments and changes the team made.

When each deliverable is completed, the team checks what additions still need to be made in order to describe the result and processes (of the respective deliverable) accordingly. During week 9 an extra meeting is to be held with the supervisor in order to get feedback on the poster and rehearse the presentation. The team hopes to learn about how team effectiveness might be evaluated by the client. We aim to focus the logging of in-game events on the identified important factors.

Week 3 Project proposal.

Week 3 Preliminary software design report.

Week 4 First single-player level with one class and only basic attacks, important individual events are logged.

Week 5 A health system is implemented where attacks affect health.

Week 5 A naive enemy is implemented that can damage players and be damaged by players.

Week 6 Basic networking is completed; two players can walk around in the world together, and important statistics on teamwork between players are logged.

Week 7 All classes are built with their abilities working as envisioned.

Week 7 A level can be played by 5 players (all of different classes).

Week 8 item rewards are given out at the end of each round and affect player(s) statistics/abilities accordingly, player choices are logged.

Week 8 Players can, using the game's interface, connect to each other, select their classes and start the level(s).

Week 9 A preliminary poster design is developed.

Week 9 A preliminary poster presentation is prepared and rehearsed.

Week 9 A preliminary formal presentation is prepared and rehearsed.

Week 10 final work is done on the report, putting all sections together and completing a spelling check.

7 Project organization

Communication is done through Discord and WhatsApp. All code and documents are stored on or indexed on a git project stored on gitlab.utwente.nl. The decision was made to use GitLab issues for issue and to-do tracking. A meeting is held with the supervisor/client at least once a week, possibly more if needed. These meetings are preferably offline, especially if there is much to be discussed. A group contract is signed so that all team members know what is expected. This group contract specifies what exactly should happen if any team-working problems arise. Group meetings without the supervisor/client are held twice a week at a minimum. At least one meeting is held physically (offline) each week.

8 Requirements analysis

8.1 MoSCoW & user stories

MoSCoW stands for Must have, Should have, Could have and Won't have. The MoSCoW methodology is often used to prioritize requirements, such that one can easily see where priorities shoudl be put. All the above requirements are accompanied with user stories.

Must have

Player movement, interaction, and level progression.
: As a player, I want to be able to interact with my character, so that I feel empowered in influencing the outcome of the game.

Data logging on in-game statistics(like task completion time, role utilization, and error rates), relevant for team dynamic analysis.
: As a researcher, I want to receive data about playthroughs of the game, so I can use it for interpreting team dynamics.

different character classes for players to pick from.

As a researcher, I want all players to have different roles in their team so that the players will only succeed when they work well as a team. As a player, I want to be able to select from a variety of character classes at the start of each game, so I can choose a role that suits my preferred play style.

The game is playable and engaging with no major bugs.

As a player, I want the game to be engaging, so that I can enjoy myself while playing the game.

Archer ranged attacks

As an Archer, I want to have ranged attacks that can quickly eliminate low-health enemies, so that I can support my team from a distance.

Guardian high defense

As a Guardian, I want to absorb enemy attacks to protect my teammates.

Should have

Play over network

As a team, we want to play the game over the network so that we can join the game remotely.

First wave of enemy units (first "level")

As a researcher, I want the players to face an obstacle that requires teamwork to overcome so that I can research the dynamics of the team of players.

Real-time multiplayer

As a researcher, I want the game to be in real-time so that the game simulates the pressure of time on the players.

Basic Sound and Graphics

As a player, I want the game to have graphics so that the game does not look visually monotone.

As a player, I want the game to have sound so that the game is more engaging.

Post-game surveys or questionnaires to gather player feedback on team dynamics

As a researcher, I would like the game to give a questionnaire to the players so that I can gather data for my research.

Item rewards.

As a researcher, I would like the game to give rewards that can be allocated to a player, so that a conflict situation may arise within the team of players.

As a player, I want to be rewarded for defeating an obstacle, so that I feel satisfaction for overcoming said obstacle.

multiple waves.

As a player, I want to be able to continue playing after finishing a single stage, so that I can challenge myself to get as far as possible.

As a researcher, I want the game to be continued after the first stage so that I can measure the difference in performance between teams that beat the first stage.

enemy variety

As a player, I want to see a variety of enemy types in each wave to keep the gameplay interesting and prevent obvious optimal item choices.

leader assigns items

As the leader of the team, I want to have the responsibility of choosing and assigning items to my teammates after each wave, considering each player's class and abilities.

high damage Sorcerer

As a Sorcerer, I want to deal significant area damage with my attacks, so that I can make up for my low defence.

tutorial/manual

As a player, I want to easily understand how the game works, including how classes, abilities, and items work, so I can strategize effectively with my team.

Could have

special movement options for thieves.

As a thief, I would like to be able to move my character around in multiple ways, so that I can approach the obstacle in multiple ways and avoid being attacked.

local co-op.

As a player, I would like to be able to play the game with other people on the same device, so that it takes less effort to play with other people.

Advanced Graphics and Animations

As a player, I would like the game to look visually appealing so that it is nice to look at.

Soldier stun

As a Soldier, I want to be able to stun and disable enemies, preventing them from overwhelming our defences and helping to protect the Guardian.

Thief splitting from team

As a Thief, I want to have the choice to separate from my team and take on a smaller wave of enemies, with the chance to secure victory for the whole team if I succeed.

gameplay feedback
: As a player, I want the game to provide clear feedback on my performance and progress, so I can track how well my team is doing and improve our strategies.

Won't have

in-game economy outside of item rewards.
: As a player, I would like the game to have an economy outside of items, so that I can improve my character without being given a reward.

Character customization (skins etc)
: As a player, I would like to be able to customize my character's visual appearance, so that the visual design of my character can look more appealing to me personally.

Complex Storyline
: As a player, I would like the game to have a complex storyline so that I can keep engaged with the game in other ways than gameplay.

8.2 Minimum Viable Product

During week 6 we found that a clear definition of a Minimum Viable Product (MVP) would help set our goals more clearly. The plan was to finish the MVP at the start of week 8 so that we could get important feedback on it before our supervisor went away for a week. The above MoSCoW section (8.1) already laid out what we "should have" and thus helped us more formally define what our MVP should entail.

The Minimum Viable Product for this project is a 2D video game designed to help researchers evaluate the team's performance. To be specific, five players should have the ability to join the game over the network. Players must be able to select unique characters and then engage in interactive gameplay within a dynamic game world and fight together against enemies. At the beginning of the game, all players should have an opportunity to vote leader. When all players are defeated, the game must end and redirect to the main menu. The game should capture and log relevant data during gameplay.

While our team encountered challenges in meeting the initial deadline for the Minimum Viable Product (MVP), we demonstrated resilience by rescheduling and successfully presenting it to our supervisor a few days later. The implemented features included all character classes, and players could progress through an unlimited number of stages, each increasing in difficulty until everyone in the team died. However, the planned features of items and the voting system were not integrated into the MVP at this stage.

During the evaluation, our supervisor provided constructive feedback. Positive remarks were given for the selection screen, gameplay experience, and the character/attack animations. However, an area identified for enhancement was

the player camera aspect. Specifically, there was a desire for a more refined and zoomed-in focus on the players.

Despite the initial setback, the feedback from our supervisor has provided valuable insights and helped us prioritize certain aspects of the game that needed more work for the time that we had left for the project.

9 Risk Analysis

For the development of Team Fantasy, we deem it essential to evaluate potential risks. The SWOT analysis provides a framework to identify the project's Strengths, Weaknesses, Opportunities, and Threats.

9.1 Strengths

- **Interdisciplinary Team:** The project benefits from a diverse team with members possessing different skills and perspectives.
- **Clear Project Vision:** The game's concept is well-defined, with a focus on team effectiveness research, which aligns with the project's goals.
- **Supervision and Support:** Having Y.D.C. Barrios Fleitas as a supervisor and client provides valuable guidance throughout the project.
- **Agile Approach:** The adoption of Scrum/Agile methodologies ensures adaptability and incremental progress.

9.2 Weaknesses

- **Limited Game Development Experience:** Most team members lack prior experience in game development, which may lead to challenges during the learning curve.
- **Networking Complexity:** Implementing multiplayer networking is a known challenge, and the team anticipates potential difficulties.
- **Time Constraints:** The project timeline of 10 weeks may limit the depth of gameplay and features that can be implemented. It's important to mention that, out of these 10 weeks, only 6 were planned for the actual programming and implementation of the game. The remaining weeks were primarily for choosing a suitable game engine, creating a project proposal and working on the poster/presentation in the final week.

9.3 Opportunities

- **Research Impact:** Team Fantasy offers an opportunity to contribute to team effectiveness research, potentially leading to valuable insights.

- **Learning Experience:** Overcoming the challenges of game development and networking presents a valuable learning opportunity for team members.
- **Collaboration:** Collaborating with a supervisor and team members can foster teamwork and communication skills.

9.4 Threats

- **Networking Issues:** Networking complexities pose a significant threat and may lead to delays or limited multiplayer functionality.
- **Unforeseen Technical Challenges:** The team may encounter unforeseen technical issues that could disrupt the project's progress.
- **Scope Creep:** Expanding the game's scope beyond the available resources and time frame could negatively impact project completion.

This SWOT analysis displays the initial assessment of the team. Adjustments to this analysis have obviously been made as more was learnt about our tools, the difficulties we had and the hurdles we got past quickly. More regarding this process can be found under the Evaluation section.

10 System Design

Our goal when designing the game was to have people who have never played games, be able to easily understand how our game works. They should be able to start the game, connect with their team and play without any problems.

10.1 Menus

We the game starts, it will first show a couple of menus. So we had to properly design them to be easily understandable.

10.1.1 The Main Menu

The main menu is the first menu you see when you start the game. With options to host a game, join a game, and quit the application. Which was done to be as clear as possible. If "Host Game" or "Join Game" is clicked, the lobby menu will be shown.

10.1.2 The Lobby Menu

There are 2 lobby menus, one for when "Host Game" is clicked, and the other for when "Join Game" is clicked. These 2 menus are almost identical, the difference is that when you host a game, you have a "Host" button. When joining a game, you have a "connect" button. In both cases, you can choose your own player

name, and put in the desired IP address. For hosting, this will be the IP that the clients need to put in to join your game. Your own (local) IPv4 address, will be automatically filled in. This was done to make starting a game easy on your local network. When you host, you have an extra button below the display of joined clients to start the game, which then will show the character selection menu (10.1.3).

10.1.3 The Character Selection Menu

In the character selection menu, you can select your character and lock it in. When you lock it in, the character will be greyed out for the other players. At the top left of the screen, you can click on the available characters to select one and view a description. When you lock it in, everything will be greyed out, so you have to make the right choice the first time. When everybody has locked in a character, the game will start.

10.1.4 The Menu Flow

The flow of the menus is described above (10.1). Below is an illustration of said flow.

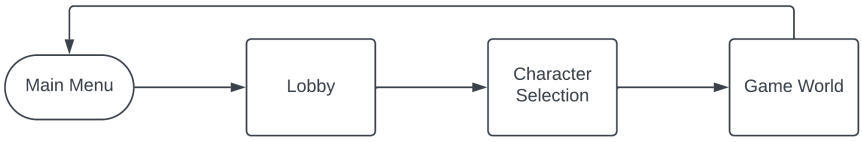

Figure 9: Flow of menu's/UI

10.2 Gameplay

An important aspect of our game's design was what player classes were available, what they were able to do and how they interact with the world.

10.2.1 The Guardian Class

The Guardian was designed to be able to take a lot of damage from enemies without being defeated, while not being able to do much damage to the enemies themselves. This design was chosen to force the ranged classes to cooperate with the Guardian if they wished to beat the enemies. The notable defensive capabilities of the Guardian allow them to survive the attention of a group of enemies for much longer than other classes, allowing the Guardian to distract enemies from the ranged classes, which can do the required damage to take out the enemies before the Guardian gets taken out.

10.2.2 The Soldier Class

The Soldier can stun enemies with a close-range attack, stopping enemies from moving or dealing damage. The Soldier does not deal much damage and cannot take much damage from enemies without dying. This design was chosen to force the Soldier to need to work together with other classes, while still vastly reducing the damage their allies will take. The special characteristic of the Soldier allows them to notably impact the game, without being able to defeat the enemies without support.

10.2.3 The Archer Class

The Archer has little defence but is capable of inflicting high amounts of damage to the enemies from a long distance. The Archer was given long range so that it could stay out of range of enemies and hide behind the Guardian or the Soldier while attacking. This allows the Archer to avoid dying. The reason they can deal high amounts of damage is to actually provide the damage needed to take out enemies before the Guardian gets overwhelmed. The Archer was given little defence because it vastly increased the reliance on the Guardian and Soldier classes.

10.2.4 The Sorcerer Class

The Sorcerer has little defence but is capable of inflicting high amounts of damage to the enemies from a long distance. The Sorcerer needs a significant amount of time to charge up their attack, especially compared to the Archer, but their attack will inflict a massive amount of damage. This long charge time forces the frontline (Guardian and Soldier) to not yield too much ground to the enemies, as the Sorcerer cannot reposition without cancelling their attack. All considerations for the Archer class also apply to the Sorcerer.

10.2.5 The Thief Class

The Thief has little defence, but high speed and deals a lot of damage at close range. The lack of defence makes the Thief rely on either the Guardian taking the attention of the enemies or the Soldier stunning the enemies. When the Guardian has the attention of the enemies, the Thief can use their speed to quickly run around the enemies and attack them from behind. When the Soldier stuns the enemies, it is safe for the Thief to run in and start attacking the enemies. In both cases, the Thief can make up for the lack of damage of the defensive classes. The low defence and range of the Thief prevents them from attacking the enemies without the support from the Guardian or Soldier.

10.2.6 The Game Flow

The flow of the gameplay is described above (10.2). Below is an illustration of said flow.

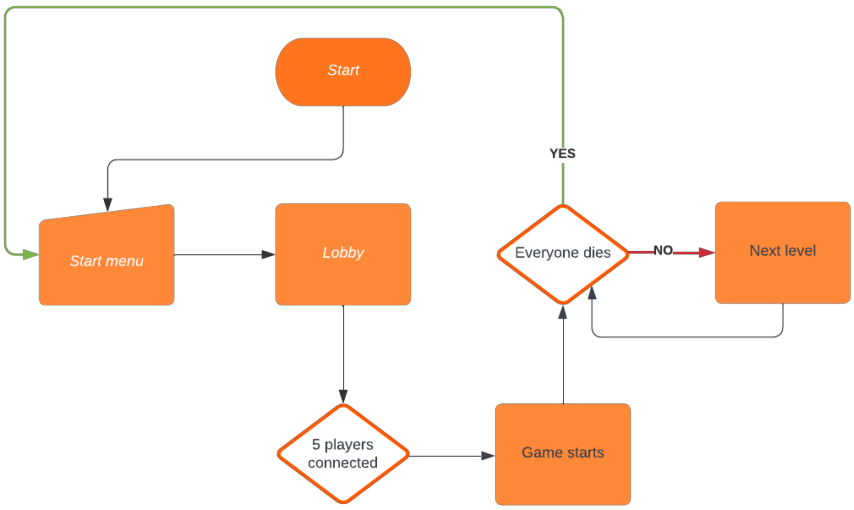

Figure 10: Flowchart of the game

10.3 System Architecture

10.3.1 Game Engine

During the planning phase, the question of which toolset to use came up. We had a couple of options to choose from:

- Godot
- Unity
- No game engine, just using an API like OpenGL or Vulkan manually
- Middleware which handles the boilerplate code for us. This lies between writing everything using raw API calls and using a game engine. A candidate for this was something like bgfx

We first ruled out the option of manually using an API. We figured that this would use up too much time and did not provide us any benefits compared to using middleware due to this being a 2D game. Since we are making a 2D game, which is a lot easier on the CPU and GPU, the performance benefits gained from using the API ourselves were not needed.

We faced a challenging decision during the selection process for the game engine. Initially, we considered Unity but eventually ruled it out for a specific reason. Given our team's lack of prior experience with any game engines, we recognized that the time constraint was a significant hurdle. None of us had prior knowledge of Unity or Godot, and the learning curve was a concern.

Godot emerged as an attractive option because it's entirely free and appeared to be more approachable for beginners. However, this choice was not without

its own set of challenges. We had to weigh the advantage of using a game engine with pre-made tools, which typically speeds up development, against the potential delays caused by learning a new engine. We were unsure if, within our tight time frame, the time invested in learning Godot would ultimately pay off in increased development speed, or if it might be more efficient to use simpler middleware that abstracted some of the lower-level API handling.

In the end we we chose Godot because we felt like the time learning the engine would pay off in the end. We then used the newest stable version there was at the time, which was Godot 4.1.

10.3.2 Scripting Language

Another significant aspect of our engine selection process revolved around scripting languages. We had the choice between two scripting languages, C# and GDScript, and each had its own set of pros and cons.

C# was an appealing option due to its wide usage, especially in other game engines like Unity. However, for us, C# has a crucial limitation. It was not integrated into the main Godot editor, which meant seting up an external editor. Furthermore, GDScript is considered the main language of the Godot engine. This meant that, at the time, there were fewer tutorials, documentation, and community resources available compared to GDScript. Additionally, We thought C# would be less likely to have the full feature set and functionalities within the Godot ecosystem, compared to GDScript. We also deemed C# as a language which would take longer for us to learn than GDScript.

On the other hand, GDScript, while not as performant as C#, was a more mature language within the Godot ecosystem. It offered a wealth of tutorials and resources, making it easier for us to learn and troubleshoot. The fact that GDScript had a Python-like syntax, which some of our team members were familiar with, was a significant advantage.

Given our time constraints and the need to quickly ramp up our development efforts, we decided to prioritize the ease and speed of learning. For this reason, we ultimately chose GDScript. This choice not only aligned with our team's familiarity with Python-like languages but also ensured that we had access to a wealth of resources that would help us overcome challenges during development.

10.3.3 General System Architecture

The game consists mainly out of Godot scenes. We have multiple types of scenes we use:

- **UI scenes:** The main menu, lobby, et cetera.
- **Level scenes:** The *world*, where the characters exist.
- **Entity scenes:** These are scenes that describe the behaviour and properties of entities, like the player and enemies.

Every `Player` instance, gets a `PlayerClass`. This `PlayerClass` is one of the following: `Guardian`, `Soldier`, `Archer`, `Sorcerer` or `Thief`. These classes all implement their own logic, like animations and the spawning of projectiles.

Enemies are implemented in a similar manner, but it does differ. Since enemies do not need classes like players do, they are simply implemented as a hierarchy. We have a base `Enemy` class. The base class serves as the foundation from which all the enemy types are derived, with each enemy type having its own class.

The `GameRun` script handles the game logic, like enemy spawning and handling when all players get defeated. `GameRun` also handles logging, and collecting the data needed for logging.

Almost the entire logic can be traced back to the autoloaded `LobbyManager` script. An autoloaded script is a script that gets initialized on startup and is accessible in other scripts, like a singleton. The `LobbyManager` script stores most of the networking state, which will be explained in-depth in 10.3.4.

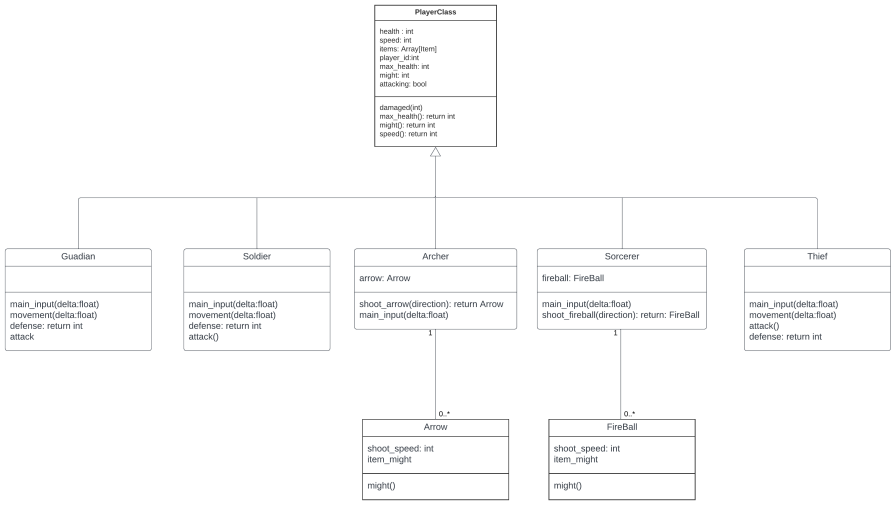

Figure 11: PlayerClass class diagram

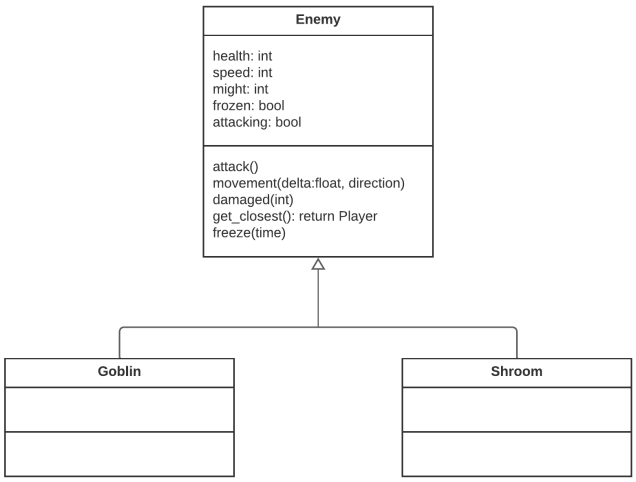

Figure 12: Enemy class diagram

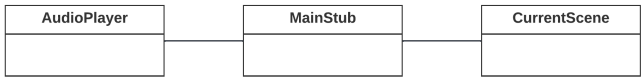

Figure 13: Top of the scene tree

10.3.4 Networking System Architecture

Team Fantasy uses a client-server model for multiplayer, where one of the players is also the server. We use Godot's `MultiplayerSynchronizer`, which manages the real-time synchronization of locations, actions, and other crucial game data, to guarantee that every participant has consistent game states. We use Godot's `MultiplayerSpawner` to synchronize spawning projectiles like arrows and fireballs, which transmits client actions to the server and synchronizes these projectiles across all clients. The general network connection is maintained using Godot's `ENetMultiplayerPeer`, which gets attached to each player.

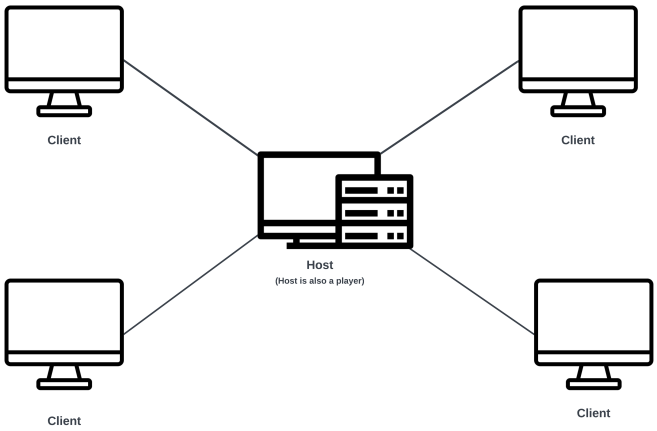

Figure 14: Illustration of networking strategy

10.3.5 Networking Protocol

All the state changes (like loading a new scene) are done via *Remote Procedure Calls (RPC)*. An RPC is a function call from over the network. This means that a client can cause an event on another system. RPCs are automatically reliable in Godot, but the `MultiplayerSynchronizer` is not. Luckily Godot has an option to also make it reliable, which of course comes at a performance cost. This, however, was not a concern for the type of game we were making, so that made it easy for us to choose UDP over TCP.

Godot itself is also more made for UDP, it does have TCP support, but Godot has a solution developed specifically to combat unreliability, so this was an easy choice. Especially seeing how most of Godot automatically uses UDP anyway, so it would have taken extra time for all network parts to get it up and running.

10.4 Logging

A JSON text file is created for each "stage". This is saved in the same folder as where the game is located. The JSON contains important game events, such as who is chosen as leader, what item is given to which player, what the time of death is of each player (if the given player died at all). The location data of players is also logged, such that this may be used for more complex grading, such as the proximity between players during the stage. A reference Python program to evaluate and visualize the data is included. This script uses Pandas and Matplotlib to create graphs and charts representing the data. These figures are saved as images and a .pdf file is created containing graphs deemed interested accompanied with explanatory text. In order to "visualize" the stage logs, simply move the log file into the "logs to be visualized" folder and run

the script. This process is elaborated on more in the user guide section (16). Example outputs of the data visualization code accompany this document.

11 Testing

Developing the testing strategy took much effort. The tools that we used were very different from what we are used to. Godot allows for tremendous freedom using Godot Scripts. However, not all development was done through these more traditional programming tools. Unit and integration tests are extremely useful to test functionality created using Godot Scripts but do not cover anything else. The testing strategy needs to contain strategies that fill in the gaps of automated tests.

11.1 Testing Strategy

Due to the time constraints of our game project, we were compelled to make strategic decisions regarding our testing approach. While we would have liked to implement thorough unit testing and acknowledge the value of tools like GUT for automated tests, the short time window for our project makes it really hard to implement it effectively.

We prioritized manual testing and concentrated our efforts on testing the critical aspects of the game, the game-play loop and essential functionalities (home/character lock-in screens), as advised by our supervisor. This approach allowed us to avoid spending valuable time on extensive unit tests while focusing more on refining the game itself, working on the design report, preparing for the poster and creating a good presentation.

It is important to note that within our manual testing framework, we systematically and separately assessed the performance of characters, enemies, and the intricate mechanics surrounding items. This approach aimed to ensure the proper functioning of individual elements before combining them into the overall gameplay loop. This testing process was iteratively performed multiple times to guarantee the robustness and reliability of each component. By adopting this comprehensive testing strategy, we were able to identify and rectify issues at an early stage, ensuring that the final product aligns with our quality standards without losing too much time to reach our other goals when creating the game.

We would like to make clear that, in a scenario where the game project is picked up by other developers and they would not have the same time constraints as us, we highly recommend adding and expanding the use of automated unit testing. This would enhance the game's long-term maintainability, and quality and ensure a smoother development process for any future modifications. Automated unit testing can be a powerful tool for safeguarding and extending the game's functionality in a more systematic manner.

11.2 GUT - Godot Unit Test

GUT is the library used for automated unit tests. In the `test/` directory one finds all unit and integration tests created for Team Fantasy.

12 Evaluation

In this section, we hope to evaluate what form our final product took and how this may be used in its intended purpose - evaluation of team effectiveness. We hope to talk about the results, without considering the specific limitations of the project, but rather solely look at what purpose Team Fantasy may serve in the future.

12.1 Result

Team Fantasy is a complete video game that focuses on evaluating team effectiveness. Players can connect multiple computers over the network and interact in a virtual world over the network. Enemies are spawned, attacked and can be attacked. Data is logged on game events and on the "special" aspects of our game - e.g. the leader who is democratically elected and later has the power to make decisions all by himself like choosing items for their team members. This data can be visualized and presented later. This however is untested and there is no guarantee that the data which is output by Team Fantasy represents the real world. Team Fantasy, in its current form, is a well-polished video game with team evaluation features that are not researched.

Because the game was purposefully crafted with the evaluation of team performance as its central focus, the inherent design lends itself to seamless extensions. This flexibility is a deliberate feature that enables customization based on the specific needs of researchers in the future.

The extensibility of the game lies not only in the extension of team performance evaluation but also in the integration of additional variables and metrics. Researchers have the capability to change the game to their unique requirements easily, adding new parameters, challenges, or aspects that align with the goals of their study. This adaptability ensures that Team Fantasy serves as a dynamic and evolving tool, capable of accommodating a wide spectrum of research objectives.

Beyond team performance metrics, the game can readily incorporate diverse data-logging features. Whether it be tracking individual player actions, strategical decision-making processes, or the nuanced dynamics of team communication, the extensibility of Team Fantasy provides researchers with a lot of choices to design and implement data collection mechanisms that align precisely with their research needs.

13 Conclusion

Team Fantasy was a project that taught its creators much about game development, specifically in combination with team effectiveness evaluation. The product clearly was created by an inexperienced team that made significant improvements over the course of its development. Exploring team performance through video games is a field that needs to be developed further, such that polished products may be developed and evaluated on their performance. How well can team effectiveness be evaluated in a video game and does this correlate to team effectiveness outside of the video game?

As the field of research evolves, the game remains a flexible and powerful resource, ready to accommodate new insights and methodologies, thereby contributing to the ongoing advancement of knowledge in the realm of team dynamics and evaluation.

14 Future Work

Currently, our game features two distinct enemy types: the goblin and the mushroom. The goblin, though relatively easy to defeat, poses a threat with its fast attacks and substantial damage output. On the other hand, the mushroom, though more challenging to overcome, strikes at a slower pace. These variations in enemy characteristics demand diverse approaches from the team, emphasizing adaptability and cooperative tactics.

To further add to the game experience, you could actively explore the introduction of new enemy types. These additions could include enemies with ranged attacks, requiring the team to refine their positioning and defensive strategies. Additionally, enemies with stunning abilities could introduce an element of unpredictability, necessitating quick thinking and communication among team members.

This ongoing development of diverse enemy types aims to enhance the strategic depth of our game, ensuring that players are constantly engaged and challenged. This adaptability in gameplay mechanics also opens doors for future updates and expansions, keeping the gaming experience fresh and inviting for both novice and experienced players alike. It also gives the players multiple ways to express themselves and work with each other to beat the game.

Another thing that could be added if this game gets picked up by other developers, is the introduction of unique abilities for hero characters, supplementing their basic attacks. This addition would not only add a new layer of complexity to the gameplay but also diversify the ways in which players can contribute to their team's success, thereby offering a more nuanced evaluation of teamwork performance.

These hero abilities could span a range of functionalities, including but not limited to dealing additional damage, providing shields for defence, or offering healing to sustain the team through challenges. Incorporating such abilities not only opens the door to facing more formidable enemies but also introduces new

dimensions to player interactions, fostering a deeper level of teamwork.

However, this potential enhancement to the game comes with its own set of challenges. The original design of the game did not account for abilities such as shields and healing. Therefore, the introduction of these features would necessitate a comprehensive rebalancing of the game. Factors such as map design, enemy characteristics, and spawn rates would need to be reconsidered to accommodate the strategic impact of hero abilities.

This undertaking aligns with the need to continually refine and elevate the gaming experience to evaluate the team's performance in more interesting ways. It ensures that as we introduce new elements, the game remains not only challenging but also finely tuned for optimal enjoyment and teamwork. Moreover, the introduction of hero abilities opens up exciting possibilities for future updates, allowing us to regularly inject fresh and engaging content into the game.

Looking ahead, one area we are considering for refinement in our game is the synchronization of enemy spawns. Currently, the game employs a randomization mechanism for enemy spawning, with the assumption that all clients will experience identical spawns. However, as the game progresses, this reliance on determinism can lead to synchronization issues among players.

To address this, we recognize the importance of implementing a more robust synchronization system that ensures consistency across all clients. The current approach, while functional in the early stages, may result in desynchronization as the game unfolds. This becomes particularly apparent in later stages when precise coordination and timing are crucial for success.

Our vision for the future involves a strategic shift away from relying solely on determinism. Instead, if some developers pick this game up in the future, they should aim to implement a synchronization mechanism that takes into account the specific spawning locations of enemies, ensuring a more coherent and synchronized experience for all players and the game itself. This adjustment is pivotal for maintaining the integrity of the cooperative gameplay, especially as the game progresses and the problem becomes more apparent.

In the event that other developers this project and find themselves with more time, we strongly advocate for the implementation and expansion of automated unit testing. This will help the game's long-term maintainability and quality, offering a foundation for a smoother development process when implementing future modifications. Automated unit testing serves as a strong tool, systematically ensuring the reliability of existing functionalities while facilitating the seamless extension of the game's capabilities. By embracing this practice, developers can fortify the game against potential issues and streamline the overall development workflow for increased success.

15 Reflection

15.1 Process

In this section, we cover all the things that happened within both the team and the code in chronological order. We hope to give a good overview of what went well, what mistakes were made and especially how these changed our future decisions.

15.1.1 Week 1-2

The first week was used for the team to get to know each other and our supervisor. During weeks one and two the main activities were meetings in which we figured out (partially with our supervisor/client) what was to be made and what the possibilities were. Everyone individually explored the tools that we might want to use and tried to build very basic games with these tools. This helped us find preferences in tools. Godot was eventually chosen.

15.1.2 Week 3

At the start of week 3, a more rigorous structure was set out. A preliminary plan was created and firm goals were set for each following meeting. The team was split up into two groups; Wouter, Timothy and Abdel worked on the project proposal and first sections of this design report. Tim, Yifan and Tycho started programming, working towards a working single-player environment and exploring what would be needed for a working multiplayer framework, as we were anticipating possible problems with networking (9.4). The project proposal contained a preliminary:

- game design
- planning
- project organisation
- risk analysis
- requirement analysis

This was presented to our supervisor/client. The general idea was defined well by the original project description provided by Y.D.C. Barrios Fleitas and had already been discussed thoroughly during the previous two weeks. As a result, little had to be changed in the game design. Y.D.C. Barrios Fleitas stressed the importance of "moments of conflict" for the players, where hard team decisions have to be made. This would allow more interesting research on the behaviour of the players. Data collection was also discussed, where we would want both data collection on individual and team performance. This was Incorporated into the planning (6). The team also received much useful feedback regarding the planning, risk and requirement analysis. It was suggested that

the planning (6) be changed such that there would be more time for finishing the poster, preparing and rehearsing the presentation and finishing the report. The idea of a Gantt chart was mentioned and Y.D.C. Barrios Fleitas made the suggestion to use the SWOT methodology in order to structure our risk analysis.

Peer review session 1 The first peer review session took place during week 3, where other Design project teams presented their projects, findings and whatever they wanted feedback on. After these presentations teams get matched up with another team in order to get feedback from an "external" group of peers. We made the decision to present our project proposal so that we might be able to make more improvements to it. The peer group took the time to go through our entire project proposal document and provided us, in our eyes, with very useful points of improvement. Questions were noted that the group had after reading the introduction, as this was simply not detailed enough for someone who is not familiar with the project concept. We also had a discussion on testing. The group rightly pointed out our lack of depth in our testing strategy in the project proposal and was curious how we would approach this. This peer review helped kick-start our search for an effective testing strategy.

15.1.3 Week 4

At the start of week 4 a similar task division was kept. Wouter, Timothy and Abdel worked to incorporate the feedback given on the project proposal into this design report. On Tuesday most work that could be done on the report was completed and so the entire team was put to work on the software. A first single-player scene was created, with the soldier class being able to walk around and attack. A health system was immediately implemented at this point and a first naive enemy was completed ahead of schedule. Collision, damage on attacks and a "death event" was fully working. The team experienced a gap where Tim, Yifan and Tycho had already built significantly more skill in developing the project using Godot compared to the other group that had occupied itself with other tasks. One-on-one meetings were held in order to catch everyone up to a similar programming level.

15.1.4 Week 5

Development was continued this week, but the entire operation became quite uncoordinated. The team found that it wasn't always clear who was working on what and how this was going. We decided that a more structured approach was needed. Team meetings were from now on to be held more frequently and have a more structured flow. Each meeting now started with every team member presenting what they were doing and how this was going. We found this approach to provide a good basis for discussion, where, the conversation after these more formal "presentations" focused automatically on the problems we needed to solve at the given moment. The planning and design were laid out quite clearly already and thus only the meetings with the supervisor were

used as true reflection moments where we could discuss a possible different approach. During this week more classes were developed and a world was created with a parallax background in which a player could move from side to side. In hindsight this week was especially important because of the adjustment in our team working approach, which contributed greatly towards the progress we made in later weeks. Networking progressed slowly. Most of the time was spent on searching for solutions that worked properly with the engine. It had to be reliable and fast enough for a real-time game. Users should also be able to have an easy interface for hosting and joining games.

Peer review session 2 The second peer review differed much from the first for us. During the first peer review, we had set out a plan but had not made significant developmental steps. This was different during this second peer review session. The team opted to devote much presenting time to a demo where we showed what had been built already as we hoped to get feedback on this. Feedback on the demo was great to have, as this confirmed that the visual aspects looked good. In hindsight, we did not get much feedback on how to develop team fantasy, this is not odd as we cannot expect there to be others who have poured much energy into game development. We hope to use the next peer review session as a sort of test where we let our peers play the game so they can provide feedback, not as other developers, but as potential users. Our peers again, rightly, put attention on the testing aspect of our game. It seems like they were also wondering what the best approach would be. This is very valuable as they helped brainstorm approaches. Well-defined manual tests were suggested by multiple peers. This is something we took with us for the testing strategy.

15.1.5 Week 6

Week 6 is the week where we are scheduled to complete the networking software such that two players, on different devices, can interact in a world together. At the start of the week, connections could be instantiated and characters could be chosen in the menu. A game however could not be played yet; this had to be, by the schedule, completed this week. Tim did a lot of work here. In hindsight, we should however have assigned someone to help him, as this was a major task. We did not complete the goal on Friday so extra work was put in over the weekend in order to stay on schedule. Progress was in the meantime being made on the different characters. On Friday we had a meeting with our supervisor where we focused on the fact that networking seemed to need (slightly) more time than needed. Y.D.C. Barrios Fleitas suggested that we aim to have a working MVP by next week, as this is realistic by the planned schedule (virtually all aspects of a realistic MVP were to be finished by the end of week 7 anyway). The team had not yet defined what exactly an MVP should contain as our goals were laid out in smaller "deliverables" which focused on individual aspects of the final product. In hindsight, the MVP definition would have helped much if there had been significant delays in development. We believe a clear definition

of what the MVP entails would have allowed us to easily shift extra focus to what is important for a working product (the MVP) and let go of things that are only "nice to haves". The requirement analysis using the MoSCoW methodology (8.1) already helped with this, but the MVP could have made this more concrete and thus is something we, in hindsight, should have defined early on.

15.1.6 Week 7

During this week more work was done to get a fully working product that met a large portion of the originally identified requirements. The MVP was completed early during the week. The entire team was now pulled into the inner workings of networking so that more variable values could be transmitted over the network using the same structure. The MVP as defined above (8.2) was now completed and discussed with the team's supervisor. Feedback was given on the experience and visuals. The decision was made to make proper use of AI tools to create artwork. The logging was also finally starting to be implemented. A course was laid out on how we were to have the different components interact. A text file containing JSON formatted data is created by Godot. While a Python program (to be written in a later week) would parse and visualize this data.

Peer review session 3 During this peer review presentation, we wanted to spend much time on the demo, as we hoped to get some last feedback on the general experience of the application/game. It was interesting to see how these peer review sessions changed the further we got into the project. Little technical discussion could be held between the teams as teams were specialising in their own programming language and tools. It was however very useful to receive feedback from a bit more of a user's perspective, even though these are all computer science students who at least conceptually understand how things work. Big takeaways were on the division of classes, where the archer and sorcerer fulfilled similar roles. We later made the sorcerer's attacks much stronger but also much slower. The Soldier was also said to be the right size, while all other characters were too small. This was a known issue which still needed some consideration, as we felt the camera might simply need to be zoomed in.

15.1.7 week 8

Some bugs were still being identified over the course of this week related to networking. Godot's high-level multiplayer tools utilise UDP with ENet on top. ENet features can be turned on and off as desired. This means one does not need to use the full feature set, similar to TCP. The bugs identified through play testing were caused by the fact that variables were not always synced; packets were sometimes lost. Letting ENet check if packets were received properly and, if not, re-transmitting them, resolved the issue. A second type of enemy was introduced, which was larger and stronger yet much slower.

15.1.8 week 9

The Formal Methods and Tools colloquium was scheduled for Friday week 9. This meant that priority had to be put here rather than on the poster presentation (week 10 Thursday). The colloquium presentation was rehearsed with the supervisor where extensive feedback was provided. Obviously, the presentation concerned some team members more than others, as the presentation was held only by a subgroup (Timothy and Wouter). The rest had the freedom to concern themselves with other factors such as the report. The report had been worked on throughout the entire project, and a general outline was completed on what needed to be written. The team made major steps during this week towards a polished report. During the same meeting in which the team received feedback, we also discussed a preliminary poster.

Peer review session 4 The fourth and last peer review session of the module did not encompass any presentations but rather a matching of teams, where each team got grouped with another, and feedback was to be given on the posters. We received some constructive feedback from the other team, as they suggested we focus on making sure the poster was understandable to everyone and not just the people who already knew what our project was on.

FMT Colloquium Our team was the first team to present, we, together with the supervisor, decided it was important to extensively demo the game, and make the presentation (and slides) engaging while going in-depth on the technical details. The presentation went smoothly and so did the demo. Tough questions were asked on design choices and the applicability of our product. The rehearsal session with Y.D.C. Barrios Fleitas improved the presentation tremendously as he made us consider the things that took the "flow" out of the presentation (e.g., the long start-up time for the demo).

Figure 15: Team Fantasy poster

15.1.9 week 10

Week 10 was solely focused on the poster presentation and completing the final deliverable, with a major emphasis on this design report. The poster presentation was set up mostly well, with 5 laptops constantly ready to play a demo.

We brought a separate router in order to not have any problems with things like port forwarding over eduroam. Some groups came with very large posters, in hindsight, the team should have looked to spend money on a big print (and discussed this with the supervisor) and thus had, in the poster department, as much of a professional look as other groups. That said, the poster presentation was built up from the technical FMT colloquium presentation as we had already a clear flow on how to present our concept. The big difference obviously was that during the poster presentation, the goal is to get as much attention from the listener as possible and not focus on technical aspects unless specifically questioned on. This we believe went well.

15.2 Contributions

A balance was constantly sought in the extent to which tasks were to be divided effectively. The project's components were highly specific and made with tools that were unfamiliar to all team members. We valued creating a technical foundation in the tools we used for each team member such that all discussions and technical decisions could be followed by the entire team. It was on the other hand very important to divide tasks as there was much to be built given the time frame. Below one finds a list of all contributions made by each team member.

Wouter van den Boer Scrum master and documentation supervisor. Also worked to improve some of the player classes, data collection, data visualization and multiplayer on multiple devices.

Tim van de Wetering Various documentation implementation of networking and handling of game states of peers in the game.

Timothy Runhaar Various documentation and implementation of player classes and interactions, specifically focusing on the Archer's special interactions.

Abdel Bel Makhfi Various documentation and implementation of player classes and interactions, specifically focusing on the Sorcerer's special interactions.

Yifan Sun Various documentation and implementation of player classes and interactions. Implemented much of the UI, such as the lobby screen, menu and character selection screen.

Tycho Dubbeling Outlined the coding framework, building the first implementations and setting the structure for all other code. Major contributions to player classes, enemy behaviour and networking interactions.

15.3 Team evaluation

The team identified both strengths and points of improvement.

15.3.1 Strengths

Effective communication Even though there were some hick-ups, the team identified all struggles regarding team coordination and communication. The approach of short but frequent meetings that was applied after having identified coordination problems early on, served the team very well.

Division of tasks Division of tasks was made whenever the team entered a new stage in the project. This was done effectively and clearly.

Integration of feedback The team seemed to pursue a general "coachable" attitude, where we constantly looked for feedback in order to improve our game. This is evident from the constant feedback seeking, from peers and our supervisor.

15.3.2 Points of improvement

Testing strategy The testing strategy should have first of all been laid out earlier during the project. Secondly, the testing strategy should have had a "time shortage" contingency version. As it was clear from the start that we might have a shortage in time and thus it was realistic that little time was left for the evaluation and testing of our work. This should have been anticipated in advance and the planning should have been adjusted accordingly.

Equality of contribution Not all team members were able to contribute as much as others. This was identified and discussed, but we believe not rapid enough. The task division worked well, but it should have been every ones task to monitor the (lack of) developments in other fields within the project. This would have helped us identify points of concern earlier.

16 User guide

The game is packaged to be run on Windows devices using a binary. We provide a zip folder "TeamFantasyGame" containing two folders, one called "Logging" and one called "Windows"; an explanation about the former will come later. The "Windows" folder contains a `.pck` file and an executable file used to launch the game (Binary executables can be created for Linux and MacOS).

Figure 16: Main Screen after launch

From here you can either choose to host the game or to join a game depending on what is agreed with the rest of the players (the person hosting does not have any advantages in the game).

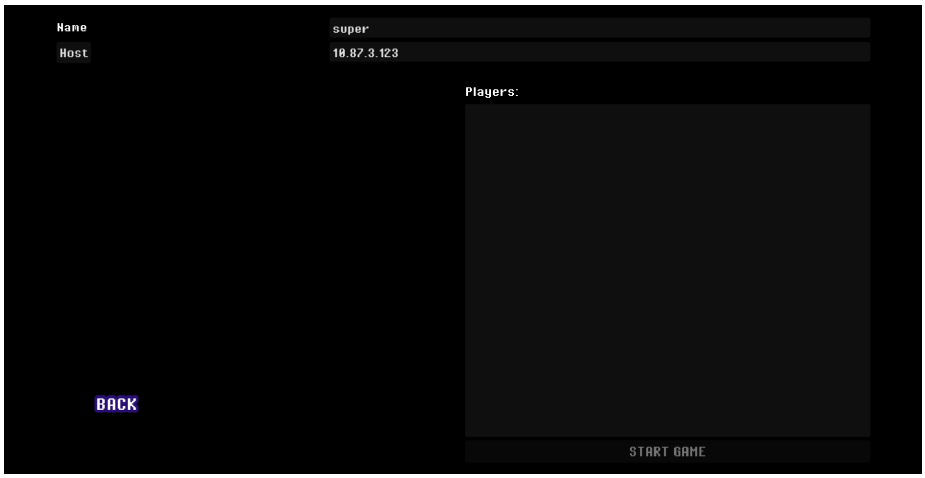

Figure 17: Host Menu

Hosting requires you to set your (public) IP address in the GUI and have port 8080 opened. This port may need to be forwarded in the NAT.

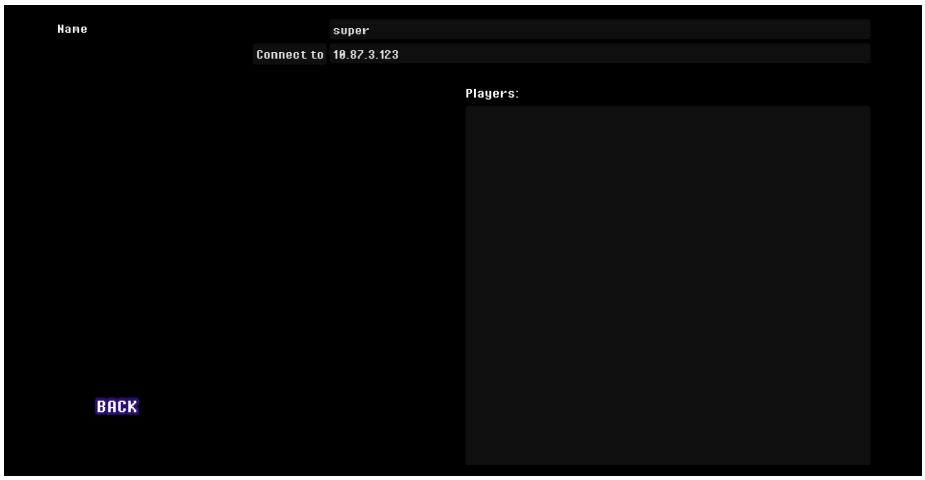

Figure 18: Client Menu

Whereas to join a game you simply input the host's (public) IP address.

Once all clients have connected, the host can launch the game, bringing all players to the character selection screen.

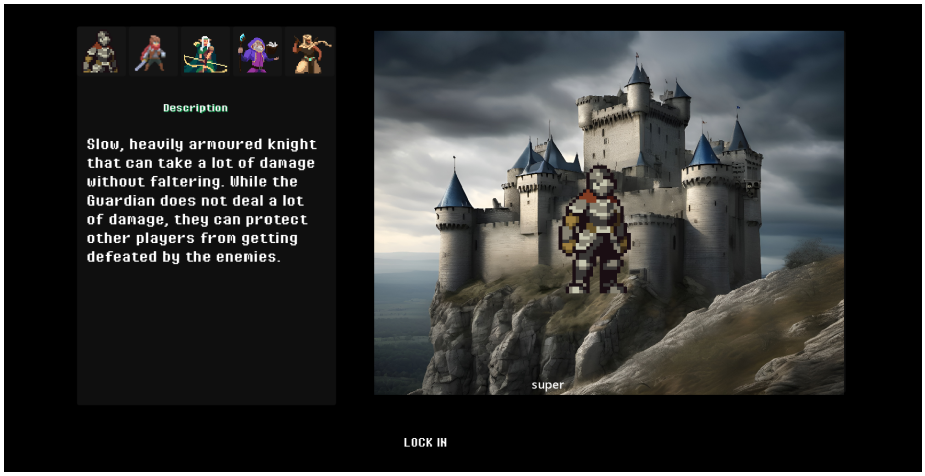

Figure 19: Character Selection Menu

At this stage, players have to lock in their desired characters based on a first-come first-serve policy. Once everyone has selected a character, they are led to the start of the first round.

Figure 20: Start of the first round

The first round is different than the others, as at the start of the game players have 10 seconds to vote for a leader[see implications of this role in section...]. Players can move around the environment using the keyboard keys WASD and attack using the left mouse button. The designated leader also gets to select which player receives the random item[see benefits of Items in section...] given after each successful round. Once all members of the team have been defeated the game ends, and all players are sent back to the main menu.

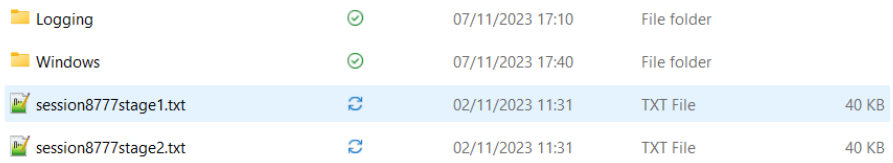

Figure 21: .txt logs generated in Main folder

In the main folder "TeamFantasyGame", you will be able to see a "session[gameID]stage[roundNB].txt" file containing all the data logged in each round. To visualize this data, first copy the `.txt` log file into "Logging/logs_to_be_visualized".

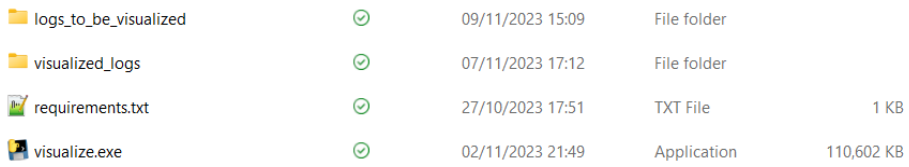

Figure 22: Logging Folder

In the "Logging" folder you will see a "visualize.exe" file to be run once all the files you want to have visualized are correctly placed. Once the executable has completed its run-time, proceed to the "Logging/visualized_logs" folder. In this folder will be present a PDF file containing all the plots for each log file you placed in "logs_to_be_visualized". Additionally, if you want to access the generated plots individually you can do so in the "Logging/visualized_logs/images" folder wherein the plots are categorized per gameID and round.

A Artwork credits

- **Guardian:**
  Artist: Namatnieks
  Portfolio: aamatniekss
- **Soldier:**
  Artist: rvros
- **Monsters and Sorcerer:**
  Artist: Luiz Melo
  Twitter: @LuizGdeMelo
- **Archer and Thief:**
  Artist: Chierit
  Twitter: @chierit7
- **World Background:**
  Artist: Zapwo
  Twitter: @wo_zap

The AI generated wallpapers were created using the img2go AI art generator.