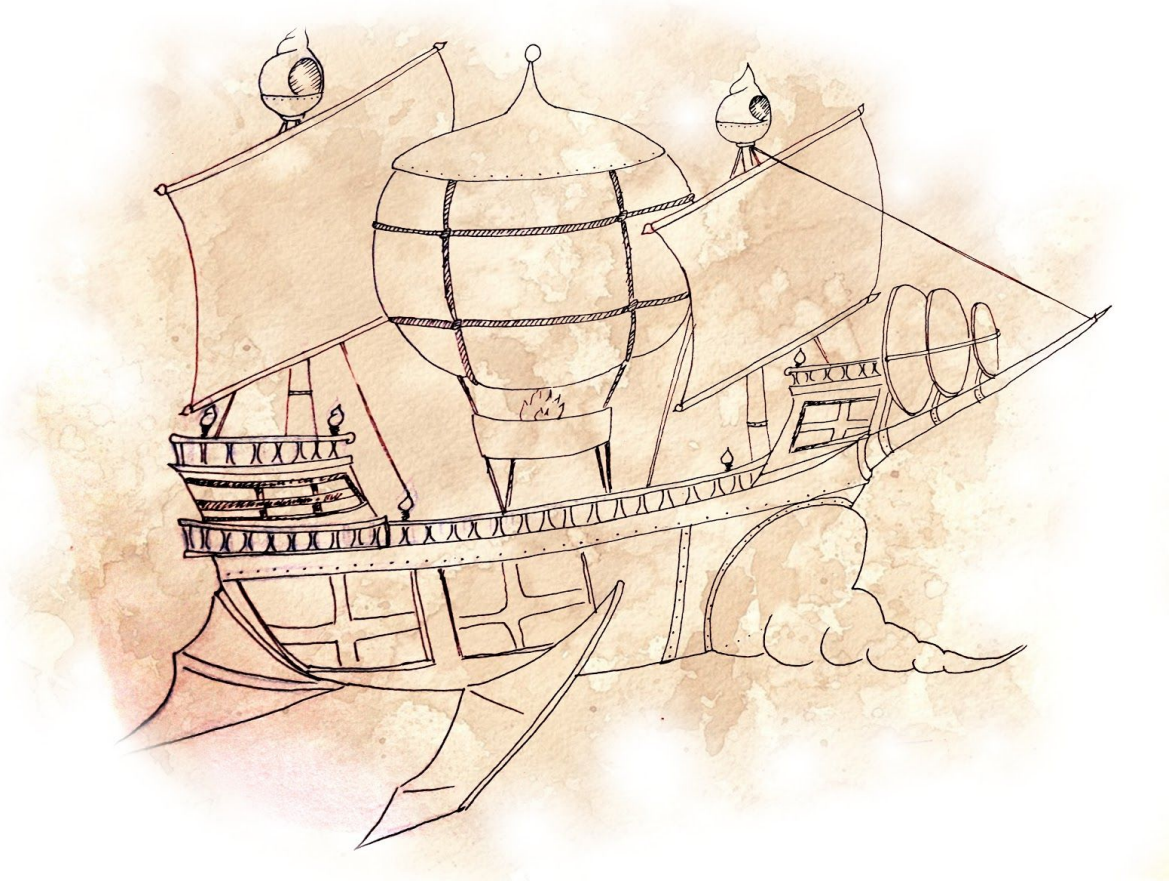
Development diary

for

Master of Tempest



A game made by The Twisted Trash Pandas

Nikita Fetisov
Evgenija Pavlova
Moritz Schöpf
Maximilian Werhahn

Computer Games Laboratory 2018/2019

1. Game proposal stage

1.1. Description

"After the fateful events of the first raging magic storm, many wizards and witches have died and one would believe that the magic would have died with them. But that was not the case. The magic would rise from a dead wizard body to become a fierce tornado, that protects them, now uncovered, the raw magic of its deceased wielder. It is now, that the remaining wizards decided to inherit and capture this magic."

Master of Tempest is a 3D Co-Op game, where communication is the key to win the game and capture a tornado. Both players, playing as the wizard and his apprentice, depend on each other's powers to reach their common goal. Therefore, the wizard depends on the perceptiveness of his apprentice, while the apprentice needs the wizard's insight on the ship to know how to repair it.

1. 2. Gameplay

1.2.a. Moving the Ship

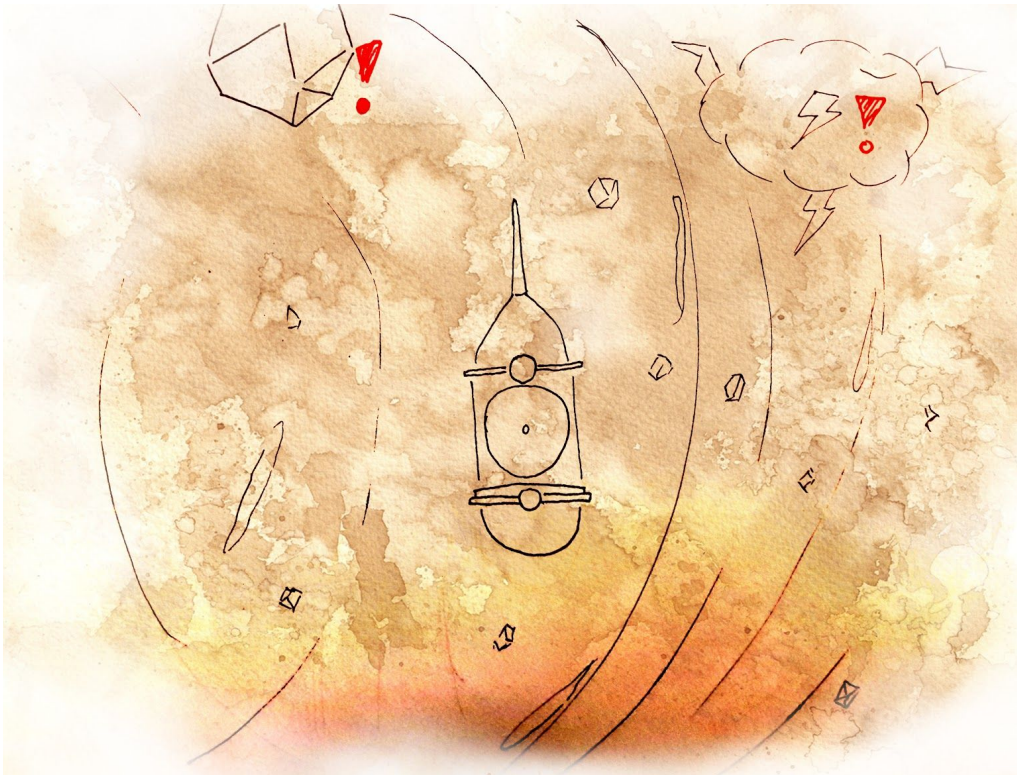
The wizard's main task is to move the ship. To do that he has to recite an incantation by pressing the right key combination. The ship can be moved up, down, left or right. Because the wizard is below deck to move the ship and has a limited view of the tornado (see left picture) , he depends on the correct navigation of his apprentice, who has the overview over the storm (see right picture). The apprentice has different lookouts on the ship to get a good overview so that he can warn the wizard of incoming dangers.

1. 2.b. Repairing the Ship

In front of the wizard is a “hologram” displaying the ship, which shows where the ship has been damaged by the storm. The wizard uses the hologram to tell the apprentice where he needs to go. Not repairing the ship can lead to losing control of the ship and the destruction of it. The apprentice has the power to use his magic (by fulfilling different mini-games) to repair the different ship parts. Although the wizard has also the power and the possibility the repair certain parts of the vehicle, he cannot control the ship for the duration of repairing.

1.2.c. Dangers of the Storm

The tornado contains different perils that the players need to overcome. The “easiest” dangers to survive are flying obstacles that the players need to avoid. Also, there are certain “danger zones” that damage the ship or affect how the ship can be moved, one example



would be a type of blizzard. Affected or destroyed areas have to be repaired either by the apprentice or the wizard.

1.2.d. Actions of the wizard

The wizard has to press different keys in a certain order to select and cast a spell. After pressing the set combination, the wizard can perform some type of mini-game to enhance the duration of said spell. However, muscle memory is rewarded in a way that the effect is more powerful.

1.2.e. Damage to the apprentice

If the apprentice is not careful enough and gets hit by some small projectile he will fall unconscious for a certain amount of time. This can be fatal since incoming damage won't be seen by the wizard. To reduce the time he is unconscious the wizard can cast a spell.

1.3. Technical achievement

The technical achievement, on one hand, consists of a stable network connection between the client and the server. All Objects and player inputs should be synchronized on the server. The apprentice, for example, has to be able to inform the wizard about the exact positions of incoming dangers.
On the other hand, the visual effects, especially the tornado, the underlying simulations and the atmosphere are also a main focus for this game.

1.4. Development schedule

1.4.a. Layered Tasks breakdown

Layered task breakdown is contained in a separate document, see development schedule.

1.4.b. Timeline and milestones

Detailed description of timeline and milestones can be found in separate documents.
Broad overview:

Deadline	**22.11**	**12.12**	**26.12**	**16.01**	**13.2**
State of Game	Game Prototype	Functional Minimum	Low Target	Desirable/High Target	Game finished
Tasks	Physical prototype Game mechanics Game architecture	Techincal (networking, player, environment) Artistic (ship model, tornado, environment)	Technical (player, ship-environment interaction, ship's hologram) Artistic (objects, Wizard's cockpit, ship model, sound, animations, danger zones)	Technical (game menu, balance, difficulty) Artistic (animations, sound, polishing)	Playtesting Adjust features Stable version

1.5. Assessments

The unique selling point of ***Master of Tempest*** is the asymmetric gameplay of the players. Each player has their own responsibilities and has to communicate well with the other player in order to make decisions and reach the goal. This allows for a lot of different strategies for teamplay that the players can experiment with. Especially when the wizard changes the role with the apprentice, the game will feel and play completely different for both of them. It is the perfect game for a casual evening with friends where you can switch out different players and roles. Other games like "Keep Talking Until Nobody Explodes" or "We Were Here" have already proven that this kind of asymmetric gameplay is extremely fun and successful.

1.6. “Bullseye” Idea

The big idea of this game is exactly what is described in the paragraph before: the asymmetric and cooperative gameplay. Both players must cooperate - with the limitations each one was given - to achieve victory in this unique setting.

2. Game Prototype

2.1. Goals

Our main goal of the prototyping stage was to confirm or disprove the hypothesis that asymmetric cooperative gameplay with one player having the information about the environment (*navigator player*) and the other player having the control (*the operator player*) is fun and viable. The proposed gameplay has several aspects, that have to be considered when designing a real-world prototype:

1. Players have to communicate in order to succeed
2. Players have to operate under time pressure
3. The navigator player doesn't have complete information about the environment and has to make decisions on the fly
4. The operator player has to solve a simple puzzle in order to control the ship

We believe the aforementioned aspects to be the core gameplay features of the proposed game.

2.2. Components

2.2.a Map

To represent the map, we use a board. The front side of the board (Fig. 1) has multiple objects (obstacles) glued onto it. The navigator player draws all the information needed from it. The back side (Fig. 2) has a grid and is used by the game masters.

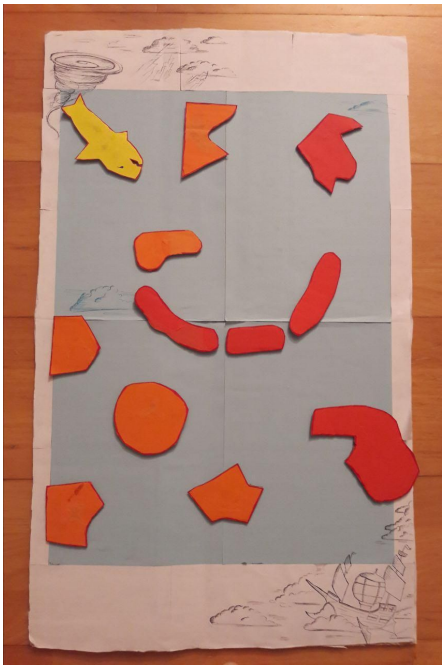


Figure 1 The front side of the map

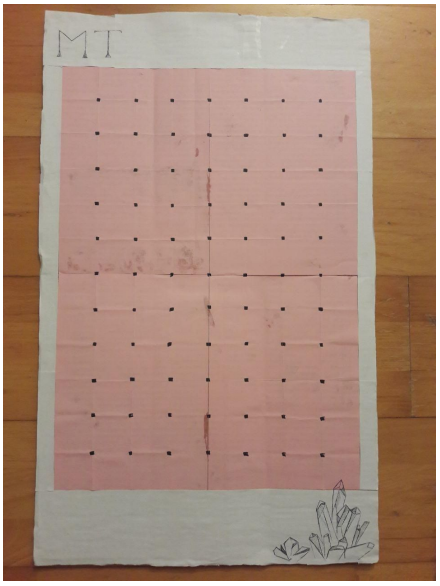


Figure 2 The back side of the map

2.2.b Ship

For the ship representation, we use two magnets. Those magnets are put against each other with the board between them so that when one of the magnets moves, the other one follows.

2.2.c Puzzle

The puzzle element of the gameplay is represented by 4 marked magnets, that the operator player has to arrange on a blackboard (Fig. 3) in a special order to change the direction of the ship's movement.

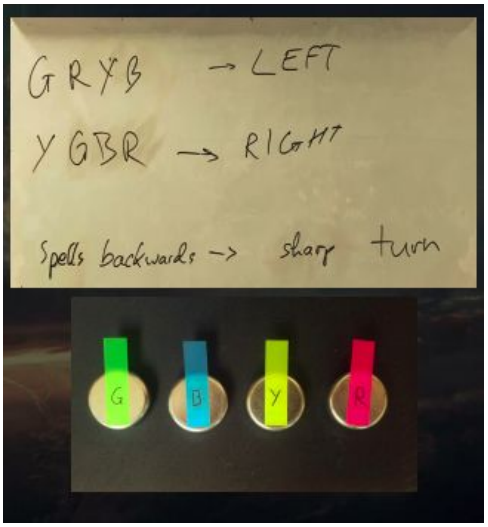


Figure 3 Puzzle

2.2.d Spyglass

We use a spyglass without optical lenses (Fig. 4) that the navigator player has to look through in order to restrict his/her range of vision.

Figure 4 Spyglass

2.2.e Stopwatch

A stopwatch is used by the game masters in order to control the pace of the game.

2.3. Rules

Four people are required to play the game:

1. Navigator player
2. Operator player
3. Timekeeper (game master)
4. Ship mover (game master)

The gameplay happens in discrete steps (turns). Each turn lasts 7 seconds, and at the end of each turn the ship moves one square in the direction it is facing. The game ends when the ship reaches the goal (which is marked with a star on a map), or when the ship hits an obstacle hard enough for the magnets to lose grip.

The task of *the navigator* is to tell the operator in which direction the ship has to be turned so that it doesn't hit an obstacle and moves closer to the destination. To make it more interesting, the navigator player has to restrict his/her vision by looking through the spying glass, imitating "unpredictable changing environment" that is expected to be present in the proposed game. Also, the commands for the turns ("left, right") are processed from the ship mover's perspective, making it more challenging for the navigator to call out the direction changes.

The operator player listens to the directions of the navigator and has to quickly arrange the marked magnets in an order that represents the corresponding direction change. After each turn, he/she has to mix all the magnets back together before arranging them for the new manoeuvre.

Every 7 seconds *the timekeeper* checks the result of the operator's manipulations and tells the ship mover in which direction (if any) the ship needs to be turned. Then *the ship mover* changes the direction the ship is facing accordingly to the timekeeper's input and moves the ship one cell forward on a grid.

This way we manage to fulfil all the requirements listed in the 1st section of the current chapter, as well as to enforce the constant movement on the ship, that captures the feeling of operating something massive with low manoeuvrability.

2.4. Experience playing the prototype

Before coming up with the final rules for the prototype, we tried out several different approaches, but they failed to capture one of the core aspects of the game.
Playing the final prototype was fun for both roles. Switching the roles also proved to be incredibly fun, offering unique experiences both for the navigator and the operator players, and helping to master the game overall. And it was fun to fail too: feeling the time pressure the navigator was giving the wrong commands, and/or the operator was not able to solve the puzzle in time, which caused excitement and desire to try again.
However, the prototype revealed several important issues that should be addressed during the development of the real game.
After a certain amount of playing sessions, the operator player can memorize the required combinations, making the gameplay trivial. This issue should be solved by adding randomization to the controlling process. Also, we plan to give the operator access to the information about the ship's status and special abilities in order to make the gameplay more intense and interesting, as well as encourage two-way conversation between players.
When the maps had straight paths, the gameplay got boring really fast, since both players had nothing to do but wait. This should be solved by creating a dynamic environment, that constantly requires player's attention and actions.
After many tries, the navigator's gameplay was getting trivial and stale. While this would take more time in a video game, since the medium is much more engaging than the board games, we also should add gameplay complexity for the navigator player. We plan to do it in several ways simultaneously: 1) add extra responsibilities to the navigator player as repairing the ship; 2) divide the visible area of the navigator between two outposts, making him/her to constantly change between them in order to have the full information about the environment.

Exploring the prototype ideas was also fruitful in a way we didn't expect: thinking about the ways to represent player's input in real-world brought ideas that could be used in the final version of the game. For example, we consider keeping the "arrange the objects in a specific order" as a gameplay mechanic for the operator player, instead of using the keyboard input for the spellcasting.

2.5. Conclusion

Creating a physical prototype helped us to crystallize the core requirements of the gameplay as well as exposed the potential flaws our game might have if we don't address the weak points properly. We believe the idea to be overall viable and will continue working on the project keeping in mind lessons learned.

Figure 5 Creation process

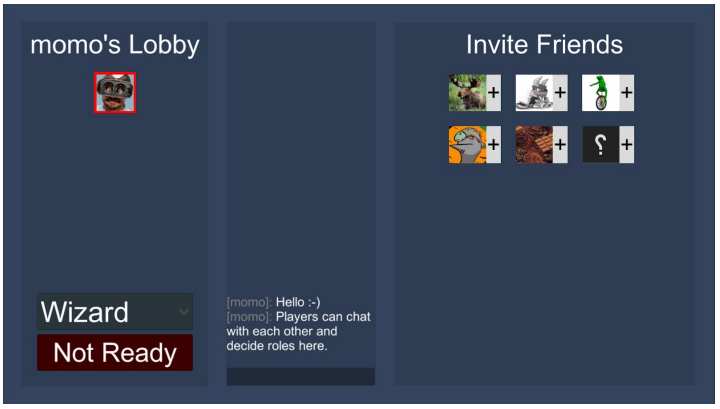
3. Interim report

3.1. Functional Minimum

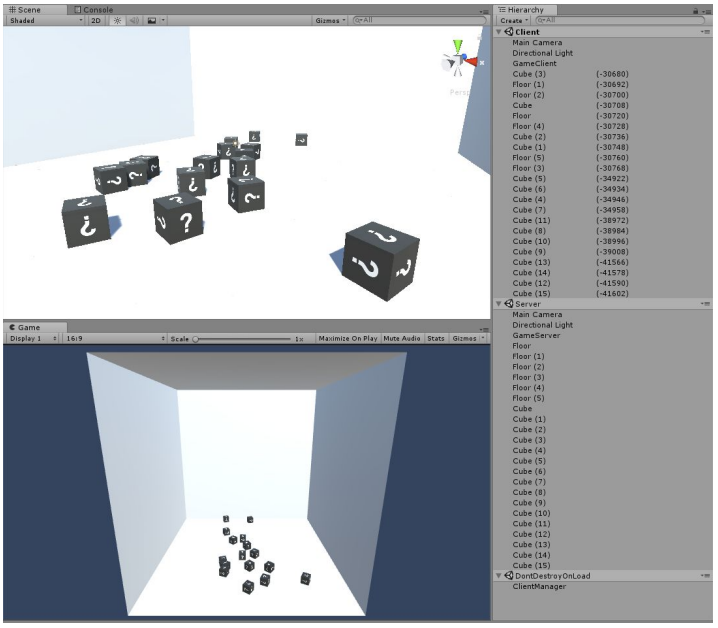
Not all features of this stage are fully implemented.

3.1.1 Networking

All of the features for Networking are implemented for the functional minimum.
The Facepunch.Steamworks API for Unity is used for peer-to-peer networking. This API supports sending TCP/UDP messages between Steam users and other features like lobbies and friend lists.



When the game starts, players can invite their Steam friends into a lobby and chat about which roles they want to take. After everyone in the lobby selected a role and pressed ready, the lobby owner will become the host for the game. Then the host will load the client and the server scene. Other clients will only load the client scene.



The server scene on the host then runs all the gameplay code based on the input from the clients. For example the client has a player character that it wants to move forward. Then the client will send a message to the server to move its character forward. When the server receives the message it will move the character and tell the client its new position. Currently, the networking system supports the synchronization, interpolation and spawning of transforms(position, rotation, scale, parent).

On top of that, custom networking messages can be sent between players. Also, the behaviour of objects on the server can and should be different from the behaviour of this object on the client. Both of this can currently be achieved within one script with separate

functions for these two behaviours. This setup for networking makes it easy to create new networked behaviour and also does not require a different code base for sending and receiving messages. Also debugging is easy because the client and server run inside the same unity project without any external application.

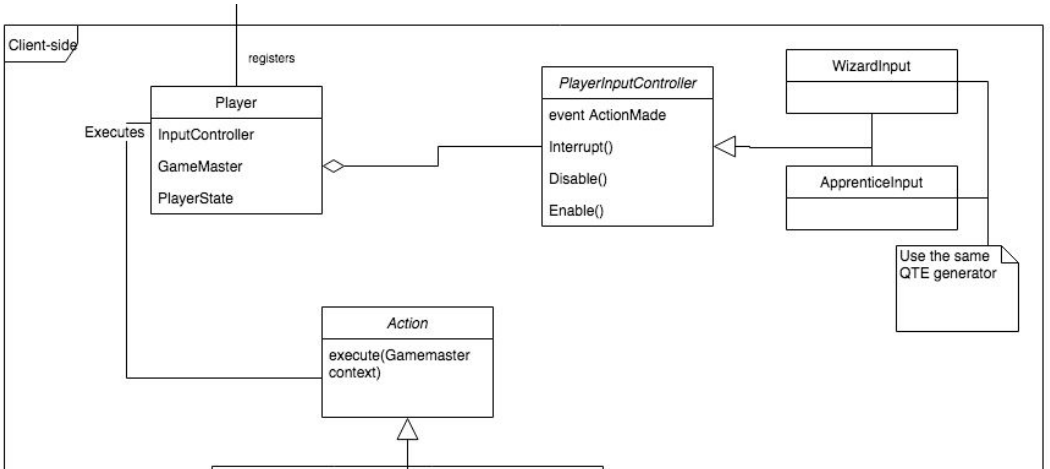
The networking solution still requires some improvements but it is definitely a good foundation for the game to build onto. Further improvements can be made for the performance and consistency of messages as well as the client side interpolation of transforms.

3.1.2 Ship-Environment Interaction

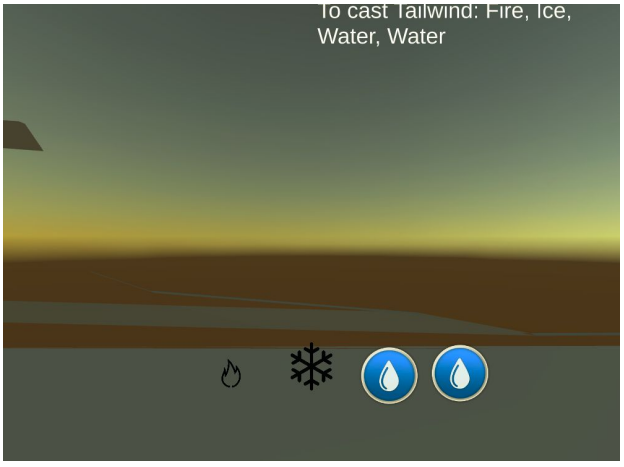
Ship-environment interaction logic is contained in particular environment objects. The Ship only exposes different manipulation components (manipulators) that are used to change the ship's state. When the environment objects register a collision with the Ship, they update the ship's state by calling the manipulator's functions. That can be adding a force to the ship, making damage or applying some status effect. This way all the networking logic that is necessary for the ship-environment interactions is encapsulated in the manipulator components, and at the same time provides the endpoint for the ship's status management that can be used for other types of Ship-X interaction.

3.1.3 Player Controls

Player controls is the point through which players will communicate with the game, so it has to be very carefully tuned and tested. Accounting to the fact, that player controls that we originally have in mind, are subject to change during playtesting and, especially, alpha testing, we wanted to make them as flexible as possible.
The solution we came up with is depicted in the picture. The Player class has a PlayerInputController that determines the player-game interaction. During the startup, the proper input controller for the chosen role is selected and attached to the corresponding player object. This approach allows us to change the particular implementation of the input controller without major issues. Whenever the Player makes a significant action in the game (Wizard might cast a spell, or Apprentice interacts with the ship), the proper Action object is chosen and the event is raised. All action logic is contained within actions, which decouples player interaction from the actual game logic, allowing for easy and independent changes or expansion of both.



The current implementation of player controls presents 3 different roles that player can take on in the lobby. The first one, Wizard, allows the player to look around and toggle a spell-casting mode, during which the player has to sort elements in a sequence that fits one of the spells. After the elements are sorted properly, the player casts a spell by pressing a corresponding keyboard key, and the quick time event (QTE) is started. Each correct input repeats the action that is assigned to the channelled spell, which helps us to achieve a smooth feel to the gameplay.

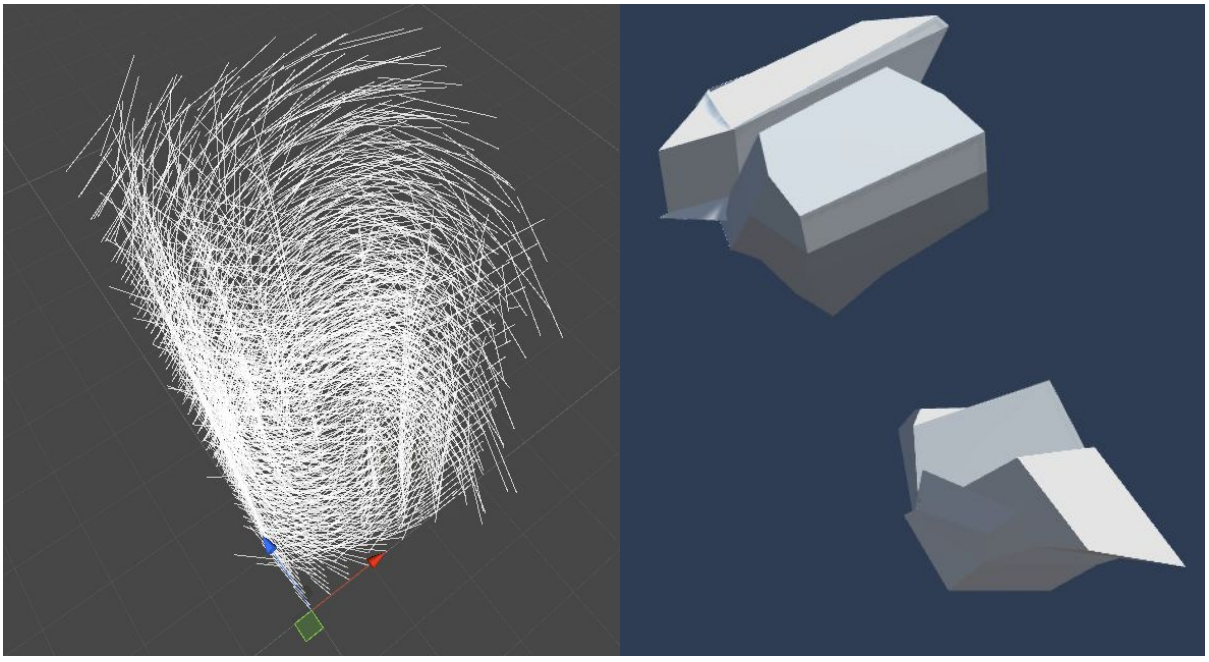


The second role, Apprentice, allows the player to interact with the objects that are in front of the player. Interaction currently starts a QTE too. In the future, we want to experiment with the ways we display the QTE cues to the players.

The third role is a Spectator role, that allows the player to fly around in a free camera mode. The minimal requirements for the player controls are met and will be expanded on in the future versions.

3.1.4 Environment

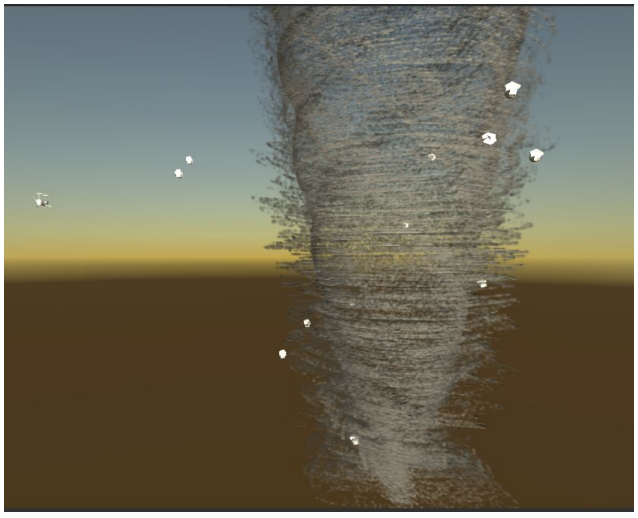
The logic behind spawning new objects is quite simple, so far. The environment spawner can control the initialized environment objects in different ways - by setting the velocity directly or applying various forces. The objects can be of three different types: damaging, supporting and danger zones. Currently, the type, position and target direction of these objects are randomly chosen, but with the position of the ship in mind.

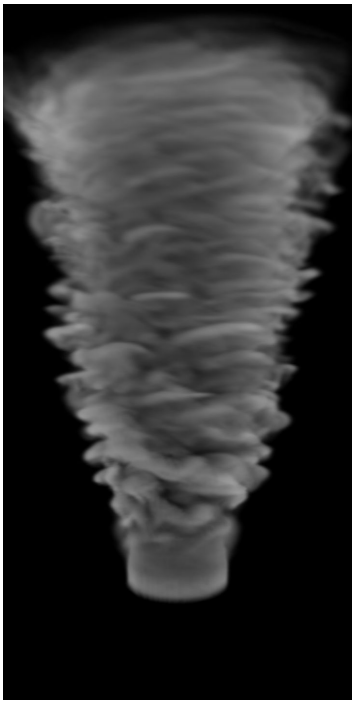


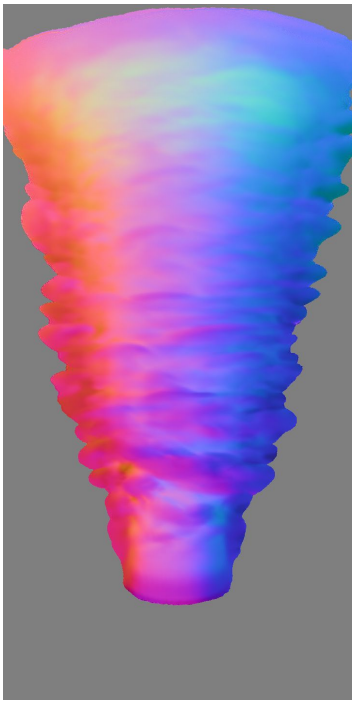
The underlying vector field can either be loaded from a file which was generated by mantaflow beforehand or initialized at the beginning of the scene. Most of the objects will be influenced by the vector field, which also controls the particles of the tornado.
If an object is outside of the set boundaries of the grid, the values will be extrapolated. For the rock three different visualizations are available.

Apart from the visuals, this requirement of the functional minimum is fully complete - objects are spawned in a proper way and target the ship indirectly.

3.1.5 The Tornado



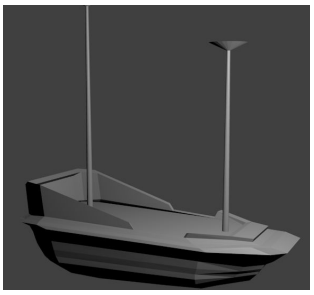




The tornado is currently represented as a particle system. All particles are rendered as transparent objects, which means they first have to be sorted depending on the camera position.
Alternatively or additionally, a simulated and rendered tornado could be used as rotating textures in the centre of the storm. From this diffuse and the corresponding normal texture, which were generated by combining six rendered images lighted differently, it is possible to “fake” the rotating motion. This can be done by rendering the volume from, for example, 72 different angles and then switch the diffuse or normal texture depending on the time and viewing angle.
Overall the tornado fulfils the functional minimum target since it is definitely recognizable as a tornado. However, the other visual effects like clouds or flashes of lightning are missing completely.

3.1.6 The Ship

The ship is being created by an iterative procedure. Starting from a simple model, more details and layers of complexity will be added with each model created.

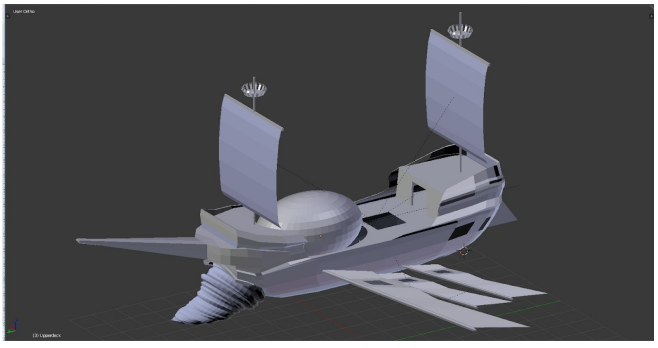


The first model had only vague similarities with an actual ship, but it could be used as a placeholder to implement in-game. It included two masts with outlook points and an accessible upper deck.

For the creation of a second model, the design choices were evaluated, leading to several changes.

The balloon is now placed “lying” on the upper deck. A “hole” in the deck allows it to reach into the room of the lower deck so that the core of the hot-air balloon is accessible below. The whole ship has more length and more sails to give it a more “epic” appeal.

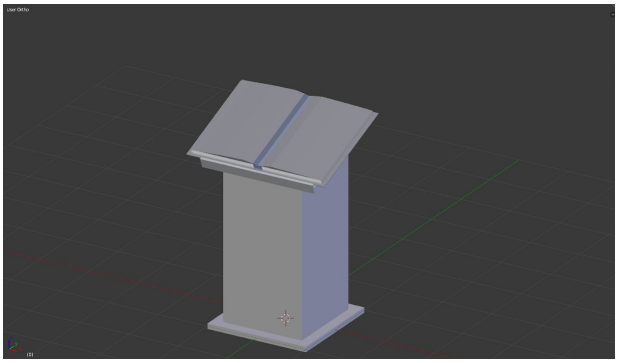
The model is recognizable as a ship and includes design trademarks (like the “crystal-beard” and the ballon/sail arrangement). Also, the model is constructed in a way, that allows destroying/shifting different parts of the ship if the vessel gets damaged. Therefore, the functional minimum is achieved.

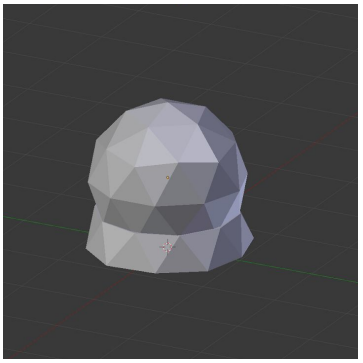


3.2. Low Target

3.2.1 The Ship

Regarding the low target, more assets and details are added to the ship. Those assets function as markers for possible interactions (repairing the ship, looking up spells, etc.) later on. Materials and textures are currently in the editing process and will be added till the next milestone.





4. Alpha release

4.1. Alpha Release - Self-assessment

At this point, we can tell, that we have achieved our low target goal. Additionally, a greater part of the desired target was also achieved. Which parts were achieved and which were not, are described below.

4.2. Networking

All the features of Networking are implemented. Performance and consistency improvements have been made since the Interim Report. There are still some minor issues with the transform updates because the game is played on a fast moving ship with a lot of obstacles flying around but the high target can be regarded as achieved.

4.3. Voice Chat

The big idea of our game is the asymmetric gameplay between two players which requires a lot of communication between them. The fact that this game is played over the internet means that the players would need to find a third party solution in order to talk to each other. This definitely is not a good solution because it requires an additional effort of the players before even starting the game. That's why in-game voice chat was implemented since the Interim Report. It uses push-to-talk with an option to toggle the recording permanently on which makes it easy to use without being distracting during gameplay. On top of that, the voice chat is also used in danger zones during gameplay. In such a zone the voice of both players will be distorted - introducing fun and also a meaningful barrier to the communication.

4.4. Player Controls

Shared player controls such as movement and interactions logic have been improved. The multiplayer nature of the game as well as the fact that the players move inside (on top) of the ship that itself travels with great velocities posed a few challenges and problems that we had to overcome.

4.4.a Wizard

The wizard gameplay was reconsidered. Going away from the quick time event mechanics, and trying to avoid using UI elements and rather give the information to the player through game world objects, we've changed the way spells are cast. Now the wizard has 4 different power sources and an altar with 4 crystals. The energy from the power sources can be drawn by the wizard and placed into the crystals. Depending on the configuration of the crystals, the spell will trigger. The energy placed in the crystals has a lifetime, and the wizard has to renew it if he wants to further support the spell. This makes the wizard's gameplay more dynamic and fun. Another new addition is the wizard's book, that is used to check the configuration for available spells. The wizard holds the book with his hand which also looks like a part of the game world and doesn't break the immersion. Animations for the wizard gameplay were also made, and we can say that the desirable target was reached.

4.4.b Apprentice

The apprentice gameplay was refined since the Interim Report. The apprentice now not only teleports to the crows nests to tell the Wizard what is happening but also has a hammer that can be used to repair damaged parts of the ship. This hammer can be charged to become more effective and it can also be thrown at incoming rocks to damage and destroy them. This results in a gameplay loop where the apprentice constantly has to charge the hammer, talk to the Wizard, move around the ship and teleport to the crows nests. This is challenging and fun while avoiding boredom for the player. Logic, animations and visuals are fully implemented and the desirable target is definitely met for this part of the player controls. For the high target, more special effects can be added and animations could be improved.

4.5. Ship

The ship was separated into multiple parts, each with its own attributes such as destruction value and status. The ship parts were then grouped up into three different areas, where every area can be repaired by the apprentice separately. When a component is hit by an obstacle only this component will be damaged and appears in a different color in the hologram for the wizard. Furthermore, the vertices of the corresponding ship part mesh are displaced into the direction of the impulse of the collision to give slight visual feedback to the apprentice as well. Instead of specifying four repairing areas we decided on three because that way we can cover all parts of the ship. Therefore, the desirable target is fulfilled.

4.6. Environment

Overall, the desirable target regarding the environment logic was reached, but instead of implementing areas which boost the ship in a certain direction we introduced a new type of zone.

4.6.a Danger Zones

To influence the ship directly two different danger zones were implemented. The first one impairs the ship's maneuverability by freezing the entire object, which can not be visible directly, but rather through the particles in this danger zone. These particles follow the rotation of the tornado. The second zone is an "instant kill" zone - all rocks in that zone can be destroyed fully by the apprentice without charging his hammer. However, the ship parts would also be completely destructed if they were hit by any obstacle.

4.6.b Obstacles

The obstacles represent the biggest threat to the players. They are the only game objects which can prevent the ship from reaching the eye of the storm. Currently, there are three different types of rocks which only differ in appearance but underlie the same logical behaviour. They can be destroyed by the apprentice if the health of the obstacles falls below zero. However, if the apprentice miscalculates his damage output and only hits the rock below 25 percent of the starting health, the obstacle will be split up into two new objects.

4.6.c Voice Chat Zones

Because we decided to add ingame voice chat, it only makes sense to increase the difficulty for the player by taking aways or disturbing the central gameplay element - the communication between the two characters. Therefore, we implemented voice chat zones which function as danger zones, but instead of influencing the direct game components such as the ship or rocks, the voices of the players will be either distorted, changed to low-/high-pitch versions or additional echo is being added to them. Overall four different versions are present which can be distinguished by the players through their different distinct and colorful particle systems, similar to the ones of the danger zones.

4.7. Visuals

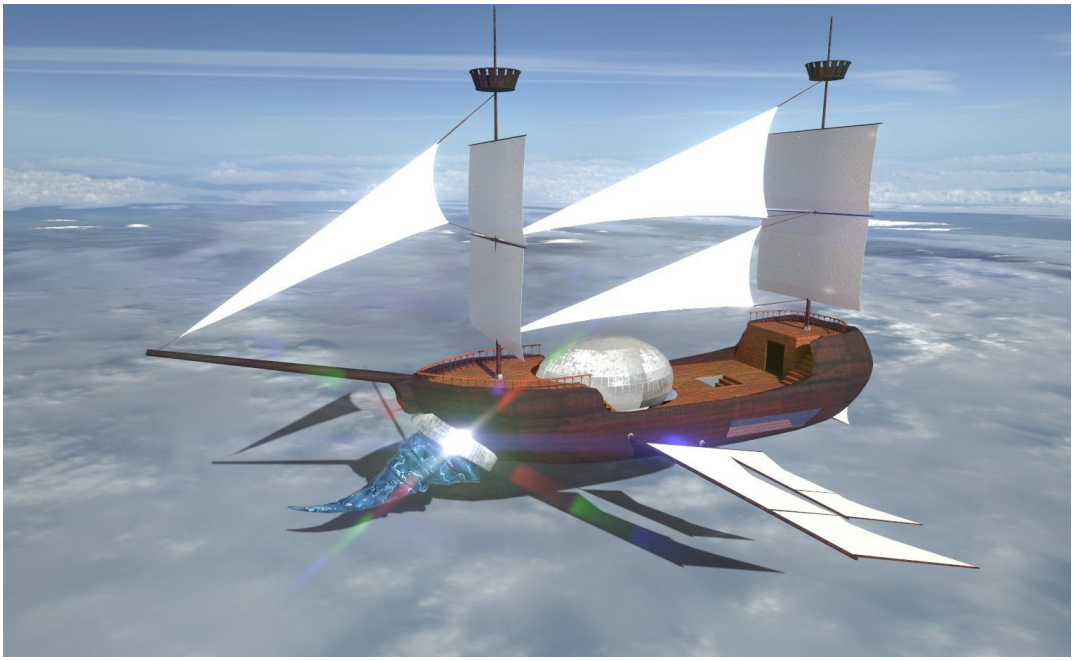
After the Interim Report the visuals of the game gained more importance and were added, edited and changed. Apart from the main look for the game itself, we realized that the visual feedback is important for the gameplay of both players. Through those the players could ascertain certain situations and gain a feeling of control for the game.

After finishing the alpha release version of the game, we can tell that we achieved our low target and a great part of our desired target.

4.7.a Ship

After recreating the ship-model, we can now tell that the ship is looking appealing to the player. The design was changed. More sails were added to give the ship a more “realistic” look. Also, more decks are now available for the apprentice to walk on. Those new spots are now used for the placement of different repair points to make the gameplay more interesting for the apprentice.

Also, we left the balloon's colour at its grey values, to leave ourselves the possibility to implement additional feedback about the current ship movement. This feature is not yet implemented but will be added to the game later on.



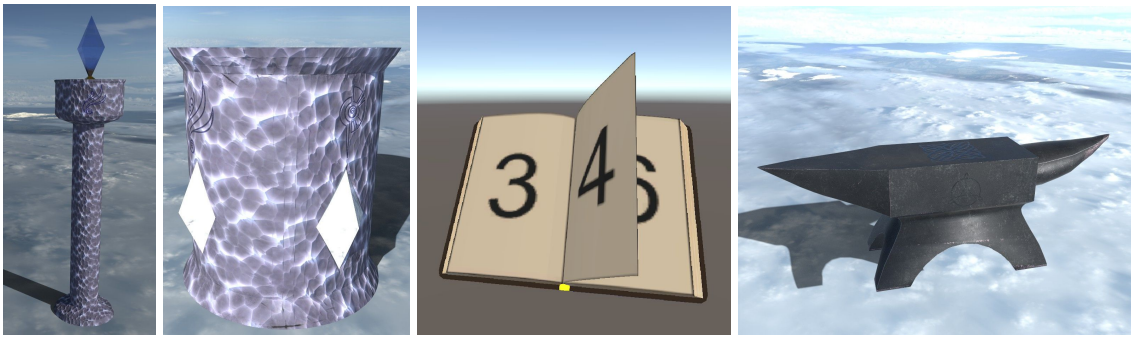
4.7.b Interactibales

Additional objects were added into the game. The most important objects for the wizard was his altar, which he uses to cast spells, and his spellbook, which he uses to look up spells. Also, we needed energy sources as components for the spells.

The altar is circular with different energy points around it so that the wizard needs to move around the altar to cast a spell. As an addition, the hologram of the ship was placed above the altar. The Player can now cast spells and keep an overview of the ships current status.

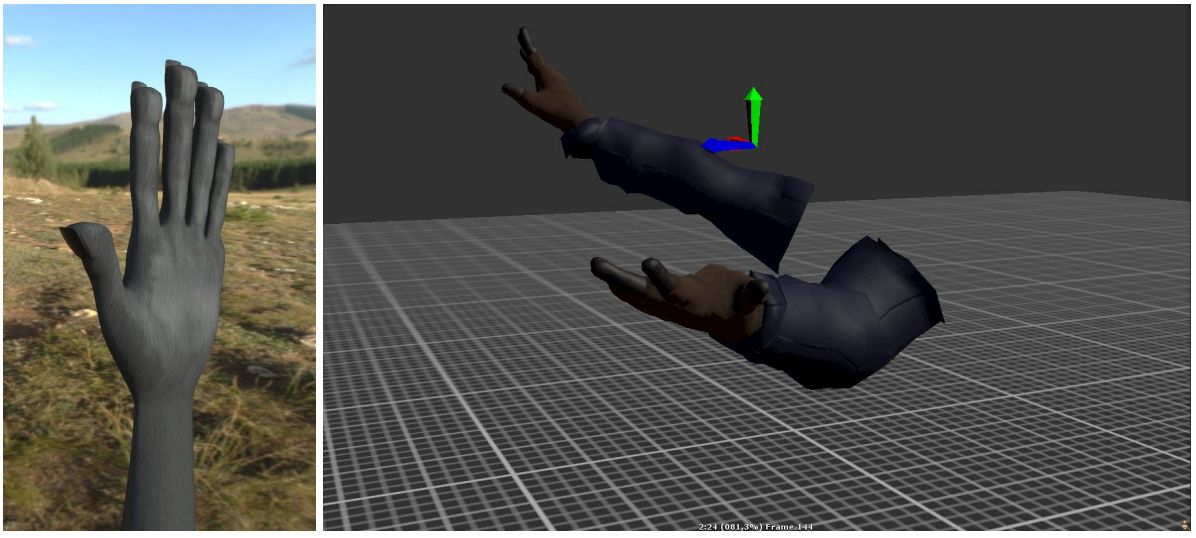
The spellbook can be used with one hand and is quick to be understood because only pictographic symbols were used to describe the spells.

The energy sources are placed around the room and are easily distinguishable from another.

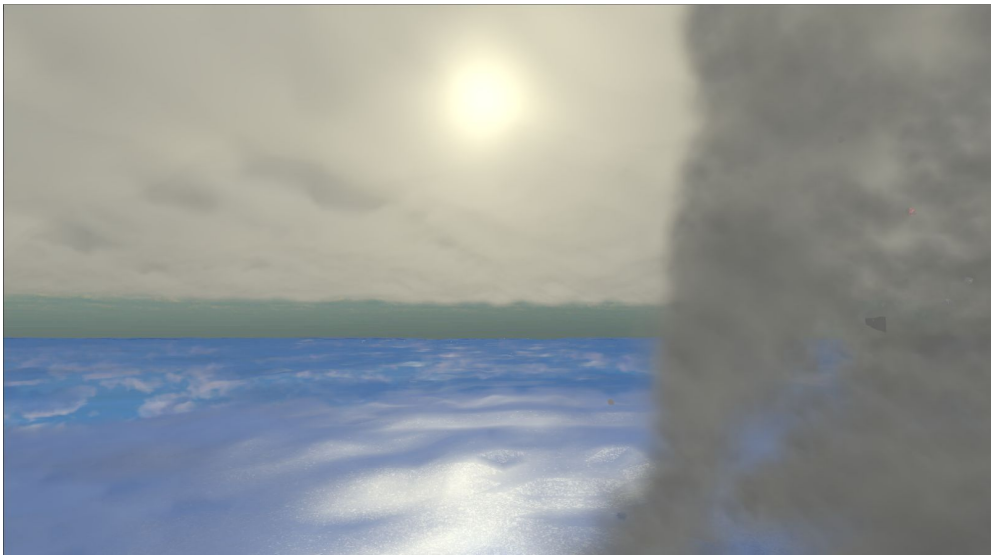


4.7.c Animations

Because of the first-person aspect of our game, only arm models were needed for the characters. For the sake of time, already made models and rigs were used for both player character. Nevertheless, custom textures and animation were made for them and added successfully into the game.



4.7.d Surrounding



The tornado still consists of multiple particles, but now they are increased in size and are rendered in a volumetric way by using lookup tables. Instead of using weather effects the danger-/voice chat zones are more than enough to give additional visual details. To complete the whole world feeling we added strongly reflecting water and a cloudy sky which also consists of volumetric particles. The visual quality of the tornado decreased if looked at from the distance but being inside the storm feels a lot more natural.

4.8. Sound

Apart from background sound for the storm, there is no sound implemented yet. This means that only the functional minimum for sound has been met.

5. Playtesting

5.1. Playtesting results

5.1.a Demographic & Procedure

Over the timespan of one week, we have tested our game with 14 players. Our participants were majorly older than 20 years (92,9%) and male (71.4%).

Most of them (64,3%) stated that they “played nearly every day” and therefore had a better understanding of games in general.

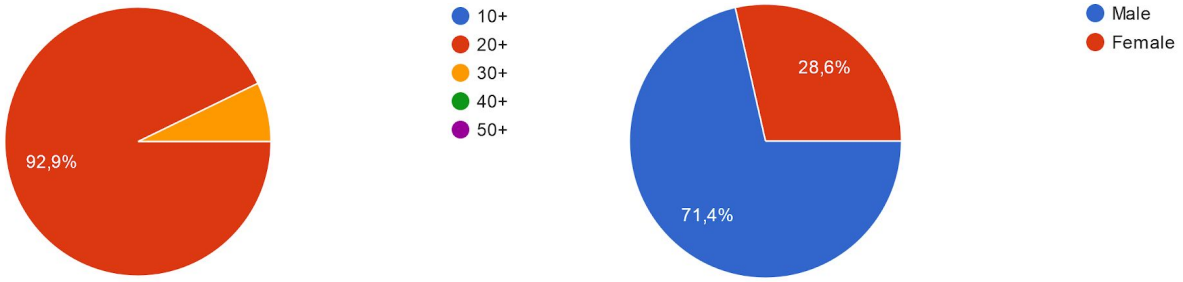


Fig.1: Age and Gender Destribution

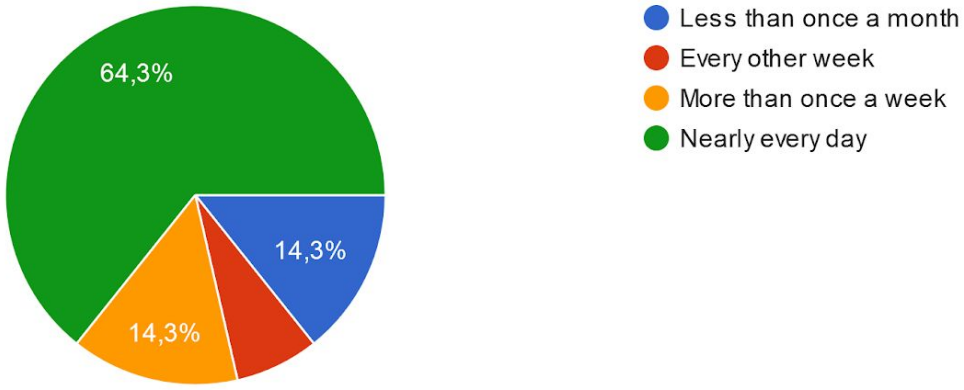


Fig.1.2: Participants general experience in gaming

Because we not only wanted to test our gameplay but also the general performance it would have on different PCs and Laptops, we decided to make small testing sessions with different players over different online communication solutions (like Discord and Skype).

Every session would be performed by two testers and at least one developer/test leader. After first greetings and other formalities, the game would be played by the tester three times. The first time the participants would not receive any explanations and the players themselves should try to figure out how the game works, while loudly speaking about it.

Before the second run, they would receive explanations and other instructions on how the characters and the game works. The participants would then again play the game with the roles they have picked in the first go. Before playing the third and final round, the players would switch roles.

After the playtesting itself, the test leader would have a short interview with both players about the game. Additionally, the players filled out a survey at the end of the session. (To overview the questionnaire in general, click here: https://bit.ly/2RSXS0W)

5.1.b Statistical Results

Overall the players rated the game performance as “average” to above “average”. None of the testers rated the game below the average performance. Additionally, 35,7% of the players did not experience any network lag. Although, 10,1% of the players had network performance which was below the common run.

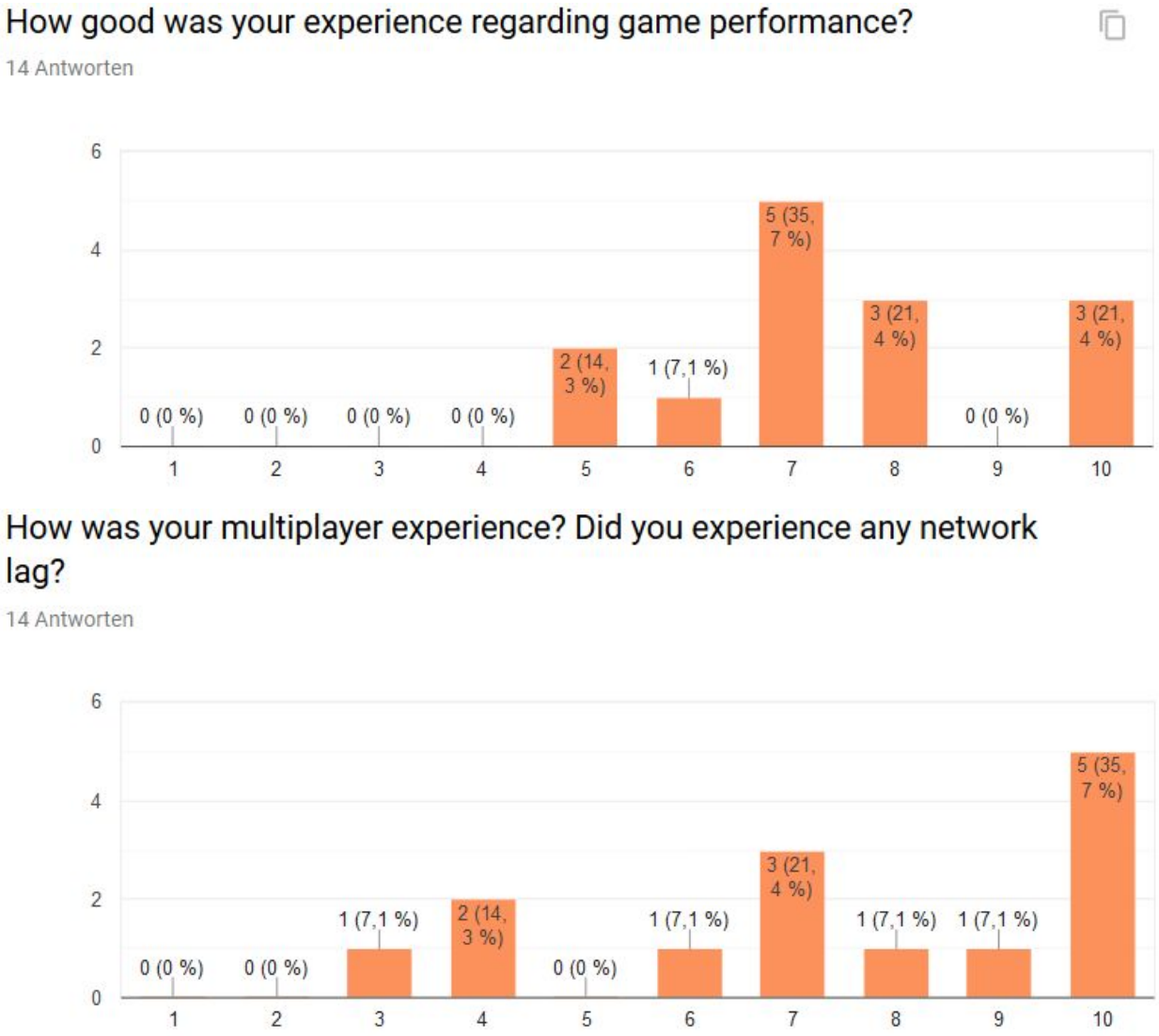


Fig.2: Game and Network Performance

The controls made sense for overall over 71,3% of the players, regarding that only one player rated them as “excellent/natural”. The other 29,7% stated them as below the average.

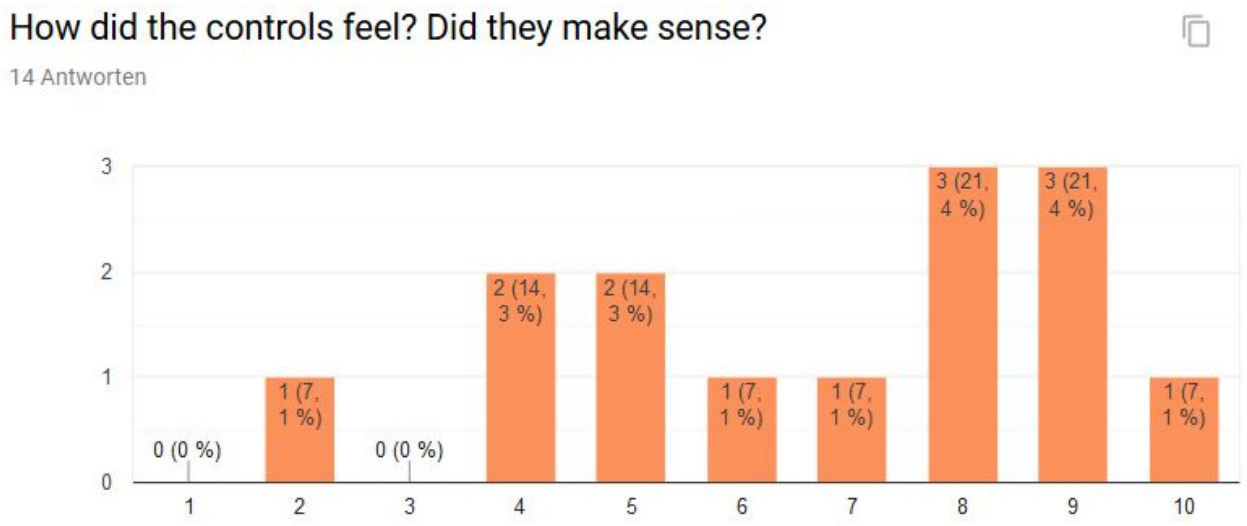
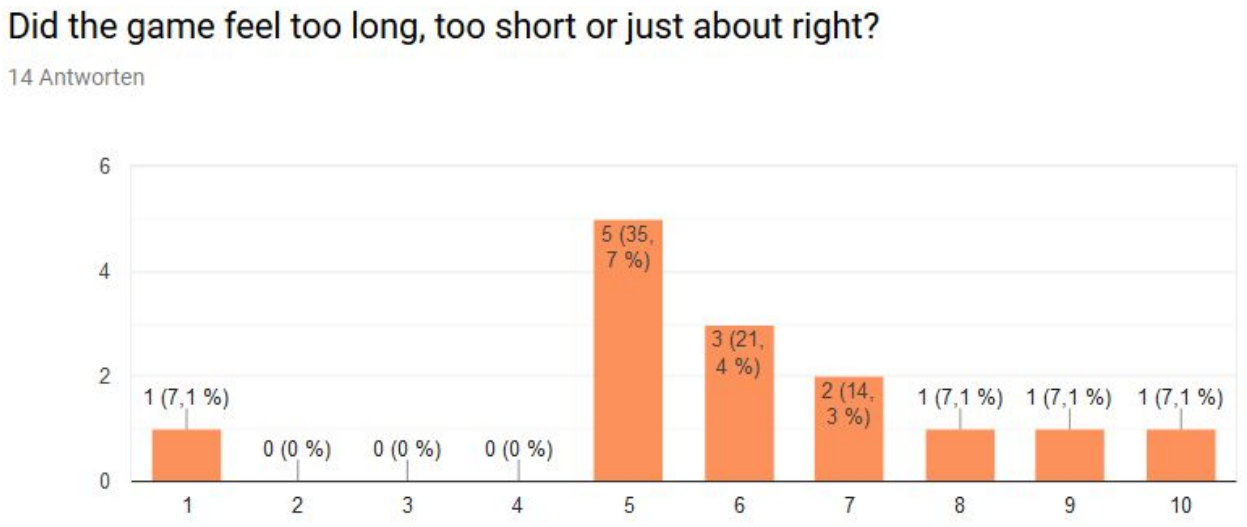


Fig.3: Control Experience

Regarding the current length of the game, the players thought of it as “just about right” (35,7%) and only experienced a medium level of stress (50%). Additionally, the game testers regarded the game as “too easy” (28,6 %).



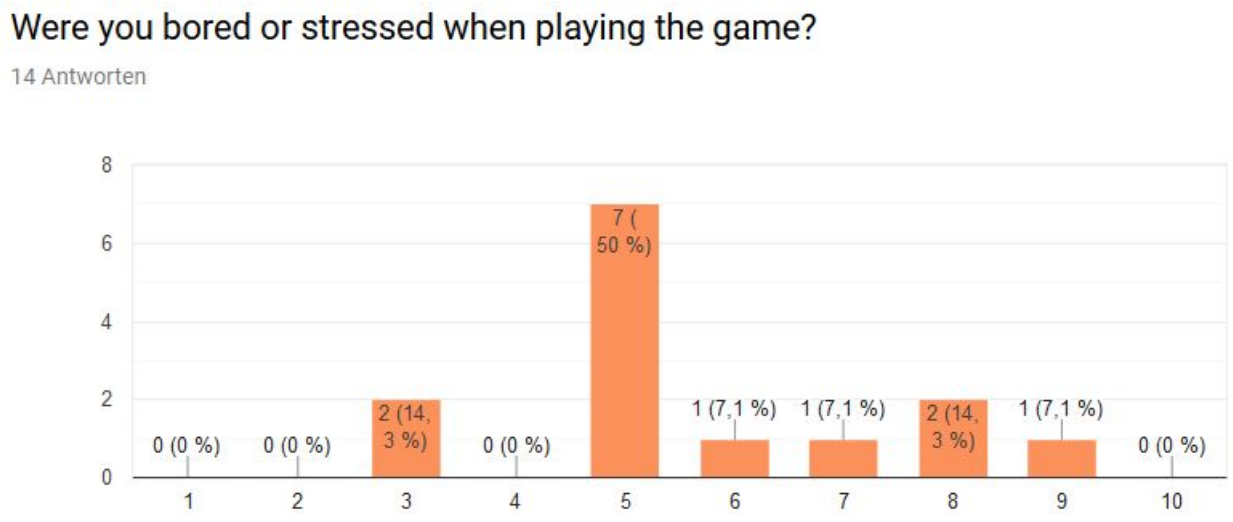


Fig.4: Game Length, Stress Level and Experience of Difficulty

Generally, most of the players had fun playing the game and evaluated it as above average. We can not tell if the players overall felt immersed in the game, because of the even spread of the evaluation.

When asked what the participants liked the most about the game, they would explain that they liked the idea of the game itself, the visuals and the different danger zones (especially the ones that affected the voice chat). On the contrary, they felt that the gameplay and controls of the wizard needed a rework because it took them to long to cast a spell that should react to an immediate situation. Also, many game testers had told us, that they had missed a tutorial or a general explanation about the game and the controls.

57,1 % of the players would consider buying this game at the current status, although only for one euro.

5.2. Evaluation

Feedback from the players gave us a different perspective on the work that we've done. We were aware of some of the problems that the game had at the alpha release state, but with the help from players we learned about other issues, as well as saw the impact of the flaws on the players' impression from the game. For instance, the players felt clueless about the goal of the game, which was hard for us to realise by ourselves since we're so deep in the context of this project. Another major issue for the players was that they lacked feedback on the ship's controls and movement, it was hard for them to tell if they are getting close to

reaching the victory. Another problem, that made the gameplay feel less fun is the reaction window to the incoming dangers. Players felt like it was impossible to make fast decisions and then execute corresponding actions in time. It took too long for the wizard player to change the direction of the ship's movement, and the players chose more "strategic" spells utilization rather than reactionary, which was not our intention.

On the other hand, the players liked the idea of the game a lot and they showed interest in testing the game again after we make improvements to the gameplay. And even with the present problems in the game, all of the testers reported that they had fun playing the game, which is explained by the cooperative and communication-based nature of the game. Also, the players shared their suggestions for the improvement or the new features of the game. These suggestions are extremely valuable, however, the time constraints that we have for this project won't allow us to implement the requested features in the scope of the project.

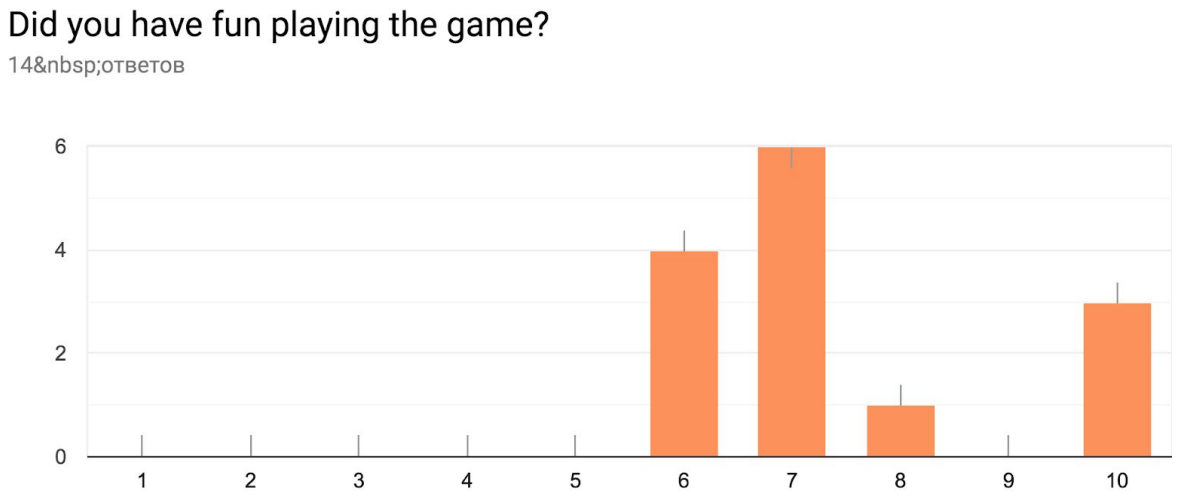


Fig.5: Feedback on fun had

It is clear, that the early feedback holds a great value for the project, and these types of the playtesting sessions with people not involved in the development process should happen as early as possible and on the regular basis, until the development team feels like the game has "converged" to the needs of the players.

5.3. Planned Changes to the Game

After discussing the feedback from the players and discovering the gameplay issues, the list of the high-priority tasks was made. We had to make compromises and leave some of the feature requests unattended because of the time constraints we're facing. Overall, the tasks can be distributed in the 3 following groups:

5.3.a Quality of life improvement

Those are small things that are easy to change, but they have a rather big impact on the feel of the game. Here we plan to add a crosshair, balance the volume for the interaction sounds, communicate the objective to the players, change some of the keybinding and make UI changes based on the received feedback.

5.3.b Gameplay changes

While we cannot make further experiments with the core gameplay, we still can improve the existing one. All the players felt like the wizard controls are too slow, so we plan to improve that by making the spellcasting routine easier. Also we want to make the centre of tornado recognizable, so that the players have no difficulty tracking their progess.

5.3.c Feedback on actions

Another issue that is important to address before the final presentation day, is the lack of the feedback on the ship's movement. Because all movement happens in air in heavily obscured area, and all the objects inside the tornado are affected by its forces, it is difficult for the players to find a point of reference in order to understand how they're moving. We plan to add visual cues that should improve this aspect of the game, as well as make the big obstacles slower.

The task list for the final sprint that was created based on the feedback from the conducted playtesting sessions is depicted in the figure below.

Tasks prioritising, distribution

- Press 1-4 to hit the crystal (no scope azaza) (N)
- Reduce drawing time to .5f (N)
- Add sounds for the wizard gameplay(N)
- Add a crosshair (Mo)
- Q, E to turn pages; R to take out the book (N)
- Update spell pages - extra pages for the mapping (J)
- Make sure that the ship turns to the direction when pulled to the side (N)
- ~~Change the button names (Book/Teleport/Toggle Voicechat) (J)~~
- Turn down the sound (for the crashes for sure)(Max)
- Give the initial objective to the players (Pages for Help) (J)
- Write "Reach the centre "before "F1 for help" (Mo)
- Replace help screen by interactable book in main menu as well (Mo)
- The arrow on the altar could not be seen from the beginning (I think we should leave it black for now) (N)
- Slow down the huge rocks(Max)
- Make big rocks visually different from small rocks (textures/fog/just color) (J)
- Introduction to the players about the game in general (general explanation/wizard/apprentice/dangers & danger zones) (J)
- A pathway from the middle to the front on the right side(Max)
- Increase the distortion on the mesh(Max)
- Increase the interaction collider for the TP zone (N)
- Mention where the player can TP in the HELP (J)
- Fix the wizard's right arm (pinky penetrates the ring finger) (J)
- Win screen needed (zoom out the camera for the sick animation) (Max)
- Improve the visual of the lobby to fit the theme better (Mo+J)
- Mention the toggle of the voice chat (double tab) in the Help screen (J)
- Reset gravity/time scale change after recovering from death (make sure it does) (Max)
- Move mic icon (make it smaller, make it scale with screen size) (Mo)
- Fix voice chat? (Mo)
- Add new decoration items to the ship prefab (+collisions after)(Max)
- Glowing Sphere at the middle (J)

Fig.6: The list of tasks for the last sprint

6. Conclusion

6.1. Summary

6.1.a Final Game

Master of Tempest is an asymmetric, cooperative game in which two players have to fulfil their tasks and communicate in order to achieve their goal - reaching the core of “The Tempest”. However, there are lots of dangers along their way. They have to navigate a flying ship through the tornado by avoiding various dangers such as rocks, parts of huge stone walls and communication- or manoeuvrability-prohibitive areas. Only then, they can avert the imminent apocalypse of this world and truly call themselves masters of The Tempest.

6.1.b Adjustments since Alpha Report

Player Controls

For both players, we added a small dot in the middle of the screen, functioning as a crosshair. That way they can teleport or target interactable objects more accurately.
For the wizard, we adjusted the way the player is able to cast spells. Instead of having to run around the altar to insert the element charge into one of the power sources he can now press the keys 1 to 4 to target the corresponding source - starting from the northern one with the key 1 and then going clockwise. Additionally, the charging time was reduced drastically.
The apprentice's gameplay did not change too much. The only adjustment we made is to lower the repairing speed. In the alpha release the player could repair all ship parts in one repairing area, whereas now, the player only repairs the most damaged part at first.

Ship Movement & Controls

In the alpha release version, it was quite hard to tell in what way the ship was moving or will move after casting a certain spell. Additionally, it was moving with and along the tornado rotation. Instead of having this constant movement around the centre of the tornado, we implemented a more linear type of movement. The ship is now moving forward at a slow speed and can be controlled more accurately by the spells, for example, it can be turned exactly 90 degrees around the local up-vector. Furthermore, we added trails to the ship's crow's nests and vertical sails. That way, when turning or moving vertically, it is clearly visible in which direction you go. Also, the ship slightly rotates around the local z- or x-axis depending on the spell cast.

Magical Crystal

Adding a magical crystal, which is always visible, as the target at the centre of the tornado helped immensely in terms of improving the feeling of movement in relative to the environment.

Environment

We added fog not only to the rocks but also to the ocean and skybox to improve the way the players perceive the distance and size of rocks and dangers. Additionally, the rocks were slowed down greatly so it is easier to avoid them.

Sound

For the overall feeling of our world, we added background music and sound effects for the wizard and apprentice.

6.2. Experience

Overall, we are happy with the final state of the project. The asymmetrical cooperative gameplay feature was brought into life, and we've received enough positive feedback during the demo day and the alpha stage to say that it's a promising idea that people like. Almost all of the initial ideas about the setting and gameplay mechanics have found their way into the final product, except for the few that we had to exclude due to the time constraints. Such were the ideas like having world objects damage the apprentice directly or having other than navigation control spells. Also, it would take much more time to balance the game properly and to adjust the learning curve, to make sure that the game is both fun after playing multiple hours, but is not too challenging for the new players.

The development schedule that was created in the first stage was used more as a guideline and not as an actual schedule. It helped us to assign milestones to the actual points in time so that we were able to see how good (or bad) we're doing on the global scale. Instead of making a schedule at the very beginning of the project and then inevitably failing to stick to it, we chose more agile approach with weekly (more frequent at the times of very intense development) calls, during which we were deciding what features should be implemented in the upcoming days, and then distributing tasks based on our personal preferences and areas of expertise. Our process resembled scrum methodology in some sense, however, it seems like it's virtually impossible to follow a “proper” scrum in a student project, because the number of hours people could spend on the project varied from week to week. The biggest deviation from the schedule for us was bringing the environment and the players’ actions together. Both the environment and the player controls logic took us more time to implement than we initially anticipated, with changing the wizard's gameplay completely after the interim presentation. This led to the actual gameplay runs and tests being very late.

The prototype stage helped us to put our finger on the core gameplay idea, to figure out what makes our game unique and interesting, its strength and weaknesses. It is hard to estimate how much it has helped us, and whether or not we could've reached the same conclusion without investing our time into making the "real world" prototype. However, it might very well be that we could've spent time more efficiently without building the prototype and focusing on eliciting the design decisions instead because most of the time during the prototyping stage was spent thinking about how to make it work with real-world objects and the results were not transferable to the final product.

The playtesting stage was extremely helpful, however, we would've received more useful feedback if we could've postponed it by a week. When we went into the alpha release stage we had certain problems in the game, that we were aware of, and that dragged in too much of the players' attention. But even with a major problem like "players don't feel like they're influencing the game flow" in the way, the critique from the players was extremely valuable and helped us to look at our game under a different angle. It is unfortunate that the time constraints of having to create a project during one semester don't leave enough time for multiple playtesting iterations, as it is extremely satisfactory to watch how the game grows and becomes better based on the potential players' input.

From the organisational point of view inside the team, everything went very smooth. We didn’t have interpersonal conflicts from the very beginning of the project to the end. The personal interests of the team members didn’t overlap too much, the tasks distribution always happened without arguments, and everyone was content with the work they had to do.

6.3. Course personal impressions

Q: Did it [the course] meet your expectations?
Evgenija: Yes, it was exactly what I imagined
Max: Yes, even though I did not necessarily have expectations.
Moritz: Yes. we did what I expected.
Nikita: Yes, it really did provide an experience of building a game from scratch

Q: Are you happy and proud of your game?
Evgenija: Yes, I believe we managed the task at hand as good as we could have in the limitations of time.

Max: Yeah, overall it does feel like a complete game to play for one or two rounds and it is also fun to “role-play” as the corresponding characters.
Moritz: Yes, the game turned out very fun and it doesn’t have many issues considering the complexity of the game.
Nikita: Yes, considering the amount of time we had at our disposal.

Q: Do you feel there wasn't enough time or that the schedule was too compressed?
Evgenija: Winter semesters are always hard because of the lack of time. However, the time between the interim report milestone and the alpha release seemed too short
Max: I think there should’ve been more time for the playtesting phase, so it is actually possible to improve the game multiple times via the gained feedback.
Moritz: There should have been a bit more time before the interim demo.
Nikita: It does seem like the schedule is too compressed: the time dedicated to the active development is very short, and the playtesting stage way too short to properly reflect on the received feedback.

Q: What was the biggest technical difficulty during the project?
Evgenija: Networking and creating the right movement of the ship
Max: Networking and creating an immersive feeling of the environment. (+ Balancing)
Moritz: Networking and the visuals for the environment + balancing
Nikita: Networking and the environment visuals

Q: What was your impression of working with the theme?
Evgenija: It was hard to think outside the box. The theme itself is defined through too clear definitions.
Moritz: A theme is nice and makes sure that we focus on one thing. But it would also be nice to have even tighter requirements.
Nikita: Having a theme helped our new team to start a conversation about the game ideas. Without a common theme from the beginning, it would’ve been harder to decide on the game idea to start actual work.

Q: Do you think the theme enhanced your game, or would you have been happier with total freedom?
Evgenija: Yes, it did. The theme was challenging, but it concentrated our thoughts for the game in one direction. I prefer theme-based projects.
Max: I think it is better to orientate along a certain direction otherwise you might get lost if you have too many ideas. So yes, it enhanced our game.
Moritz: The game is completely based on the theme! Its harder to be creative without any requirements I think.
Nikita: The game is inspired by the theme, so it helped us a lot.

Q: What would you do differently in your next game project?
Evgenija: Invest more time into the game right at the beginning. It would have been more relaxing in the end.

Max: Working on gameplay prototypes first would've been helpful to see which way of casting spells or other controls would be best.
Moritz: Thinking about the game design a lot more.
Nikita: Spend more time on the game design discussions, until everyone agrees and there are no questions left.

Q: What was your greatest success during the project?
Evgenija: The unique gameplay (although it still needs balance); the "look and feel" of the game (the storm looked amazing!); The good working network
Max: The asymmetric, coop gameplay is really fun + visuals + networking.
Moritz: Steam Networking implemented completely by us down to the bytes!
Nikita: The network element is definitely the strongest point of our game: even with not the best connection quality the game plays fine.

Q: Are you happy with the final result of your project?
Evgenija: Yes, although it would have been nice to rethink and retest the game a few times, to polish game and gameplay (eg. gameplay of the wizard)
Max: Yes.
Moritz: Yes
Niktia: Not happy, but the result is good enough all things considered.

Q: Do you consider the project a success?
Evgenija: It was up and running at the demoday (with only few problems appearing), so yes!
Max: Yes, apart from a few more or less game breaking bugs it was fine. People seemed to enjoy it.
Moritz: Sure
Nikita: Yes.

Q: To what extent did you meet your project plan and milestones (not at all, partly, mostly, always)?
Evgenija: The hardest milestones were the alpha release and the interim report. I believe we had a slow start, but we managed at the end.
Max: Probably the desirable target/interim report was not fully achieved. Few things like background visual effects or basic sound were missing. Other than that it went quite smoothly.
Moritz: Most of the milestones were met!
Nikita: The interim report milestone wasn't quite reached, but we've managed to catch up over the Christmas break.

Q: What improvements would you suggest for the course organization?
Evgenija: No suggestions that come to my mind.
Max: Better planning for the separate stages(maybe more time in later stages, especially playtesting).
Nikita: It would be nice to have more time after the playtesting stage to be able to go through 2 iterations and see how the feedback changes. Also, I'm not convinced by the real-world prototype assignment: with the time issue that we already have, it feels like an

impediment. Also, it doesn't make sense to ask students to prepare a schedule with tasks week by week by person for 3 months in advance. It would never work, neither in student projects nor in the industry. This approach might be useful for well-specified and typical projects, but even for those projects, it's not always advisable. As for our situation, we had this assignment before we even went into the "Prototype" stage, and we didn't have the gameplay specified yet. Considering that we also wanted to try things we didn't use before, our time estimates could not have been precise at all. Also, application architecture and application design were not touched in the course. I think it would be better if for the development schedule it was enough to set up milestones with describing functional minimum, desirable, etc. requirements and drop the real-world prototype task, and instead focus more on the game design and application design. It would've been interesting to see how other teams designed their game from the software engineering perspective, not only game design decisions.