Computer Games Laboratory - WS 18/19

Team Plot Twist

Alexander Müller

Kagan Batuker

Jan-Philipp Fahlbusch

Domenik Popfinger

Contents

1. Formal Game Proposal **3**
1.1. Game Description 3
1.1.1. Game Idea 3
1.1.1.1. Relation to Course Theme 3
1.1.1.2. The Team 4
1.1.1.3. Actions 4
1.1.1.4. Power-Ups 4
1.1.2. Concept Art and Sketches 5
1.2. Technical Achievement 9
1.2.1. Steamworks P2P networking 10
1.2.2. Water simulation 10
1.2.3. Weather simulation 10
1.3. “Big Idea” Bullseye 11
1.4. Development Schedule 11
1.5. Assessment 14

2. Game Prototype **15**
2.1. Overview 15
2.2. Rules and Gameplay 15
2.2.1. The Prototype Components 15
2.2.2. Boat Tile Crew Positions 16
2.2.3. Crew Variations 17
2.2.4. Start Positions 18
2.2.5. Actions per Turn 18
2.2.5.1. Reposition Crew 18
2.2.5.2. Move Boat 19
2.2.5.3. Attack 19
2.2.5.4. Moving Pieces Towards Center 21
2.2.6. Special Cases 21
2.2.6.1. Two Players on Same Field 21
2.2.6.2. Player on Center Ring 22
2.2.6.3. Attacking on Same Side 22
2.2.7. Players Get Eliminated 23
2.3. Experience 23
2.4. Learnings from Prototype 23
2.5. Game Revisions 24

3. Interim Report **25**
3.1. Scripting and Gameplay 25
3.1.1. Ship Movement 25
3.1.2. Cannon Usage 26

3.1.3. Crew Movement 27
3.2. Modeling 28
3.3. Rendering 29
3.3.1. Water simulation 29
3.3.2. Cloth 30
3.4. Networking 30
3.5. Sounds 30

4. Alpha Release 31
4.1. Scripting and Gameplay 31
4.1.1. Ship Movement 31
4.1.2. Cannon Usage 31
4.1.3. Crew Movement 32
4.1.4. Other Scripting and Gameplay 32
4.2. Modeling and Animation 32
4.2.1. Modeling Tasks 32
4.2.2. Animation Tasks 33
4.3. Rendering 33
4.3.1. Water Simulation 33
4.3.2. Cloth 34
4.4. Networking 34
4.5. Sound 34
4.6. UI 34
4.6.1. Tutorial 34
4.6.2. Player HUD 35

5. Playtesting 36
5.1 Testing Process 36
General Questions 36
Formal Elements 36
Procedures, Rules, Interface and Controls 37
Specific Question on Gameplay 37
End of Session 37
5.2 Testing Results 38
5.3 Evaluation 38

6. Public Presentation and Conclusion 40

Project Structure Document

1. Formal Game Proposal

1.1. Game Description

1.1.1. Game Idea

Fatal Tides is a three dimensional multiplayer battle arena game, taking place on sailboats caught in a huge maelstrom. It features exciting naval battles fought for survival, as the vortex is created by a hungry kraken, looking for its next meal, and only the last ship is spared by the monster.
The multiplayer is fully realised over Steam and steamworks and will feature thrilling battles between two and four teams fighting for survival. Each team, with up to two players, pilots a sailboat caught in the current and tries to attack the enemy vessels, in hopes that they will get destroyed by the kraken. A team wins as soon as all other ships were eaten by the monster. To achieve this, players have a selected amount of actions at their disposal, to distract and hinder the other boats from escaping the current and hopefully sending them to the depth, before they can do the same with you.
To create more variety and create more replayability, our game features power-ups, which can be acquired during the sea battle, as well as unlocking a variety of ships and characters with unique abilities.
The longer the battle in the maelstrom lasts, the hungrier the kraken will get and it will worsen the current, drawing the ships faster into the center. Also a storm will slowly form overhead, sending lightning into the maelstrom, which can hit your ship and light it on fire. You then have to extinguish the flames or go down in a burning wreck. The last stage will include the kraken helping with its huge tentacle arms, which will fling from the depth of the maelstrom to the boats overhead, trying to hit them and damage the vessels.
Also, to give people a tutorial and learning curve for our game, it will feature a small set of story inspired Coop scenarios with different ship layouts and crew members. These will teach the basic mechanics of the game and especially in the later scenarios, will feature certain puzzle elements, requiring the players to figure out the most effective strategy to survive the maelstrom. For example you could find yourself on a huge ship, which has already lost a great deal of its crew and you and your friends have to defeat the enemy ships. These scenarios are all played against an AI, which controls the other boats.
Another important aspect of our game, is the visual representation of the water, weather and wind. We will simulate the water flow of the maelstrom, the clouds forming overhead and the wind moving the cloth from the sails.

1.1.1.1. Relation to Course Theme

Our game takes the theme “Twister”, interpreted as a tornado, and puts it into the water, creating a huge maelstrom. This vortex is literally the centerpiece of our game, as the core gameplay revolves around two to four ships caught in the deadly current, battling for

survival. The maelstrom is caused by a huge kraken in the center, craving after human flesh and trying to navigate the ships directly into its deadly fangs.

1.1.1.2. The Team

Each team in our game has the same structure. It consists of up to two human players, taking the roles of the captain and the first mate, and a few NPC crew members, which can be given orders by the players. Players and the crew can roam freely on deck and can carry out predefined actions on certain parts of the ship.
The commands to the crew members can be given with simple commands from a button press on your controller and directing it with the joystick towards the crew members that should follow it. Available commands are firing the cannons, repairing the ship, adjusting the sail to allow for more maneuverability, fishing power up items from the sea and boosting the moral for the rest of crew, by playing the war drums.

1.1.1.3. Actions

As with any historic naval battle, the primary weapon of choice is the cannon. This medieval war instrument can be aimed at the other ships (the camera perspective will change, allowing for a clean view towards the targets) with the joysticks, showing and approximated trajectory curve, which highlights the targeted impact location. Based on a few random and fixed variables, your shot will land near the targeted area, which will always leave a chance to miss the enemy ship entirely.
Of course these attacks will leave the ships damaged and might, if the cannoneer is lucky or very skilled, even hit crew members or the captain and send them overboard. To not leave the ship unattended for long, these lost souls will return to the deck after a few second to enact revenge. Also with building materials stored below deck, your crew or yourself can fix the broken parts of your ship. This is important, as damage to the ship's hull or masts causes a decrease in maneuverability and the health bar of the ship and fixing them will restore a certain amount back.
The main task for the human player (if only one is on the boat) is to command the crew to their posts and steer the vessel. The current of the maelstrom will always provide a certain base speed of the ship, but through smart maneuvering closer to the center of the maelstrom and back towards the outer edge, the ship can gain and lose speed. This is important to maybe move away from the enemies target path and position oneself in a more promising position. If a boat has two human players stationed on it, both players can alternate with steering, while also helping the crew shoot at the enemy or repair the ship.
Other actions are adjusting the sails of the boat, improving or decreasing the maneuverability, fishing crates from the sea for power-ups, building materials and ammunition. Also the moral of the crew can be boosted by playing the war drums. This results in better aiming and higher repair speeds of the crew mates and players.

1.1.1.4. Power-Ups

To allow for a more diverse player experience, the ammunition and building materials on the boat are limited. New materials have to be fished from the sea, where crates will be floating. These creates can contain normal cannon balls, wood planks and cloth for repairing, but can also contain special equipment that can either enhance your capabilities or have a negative

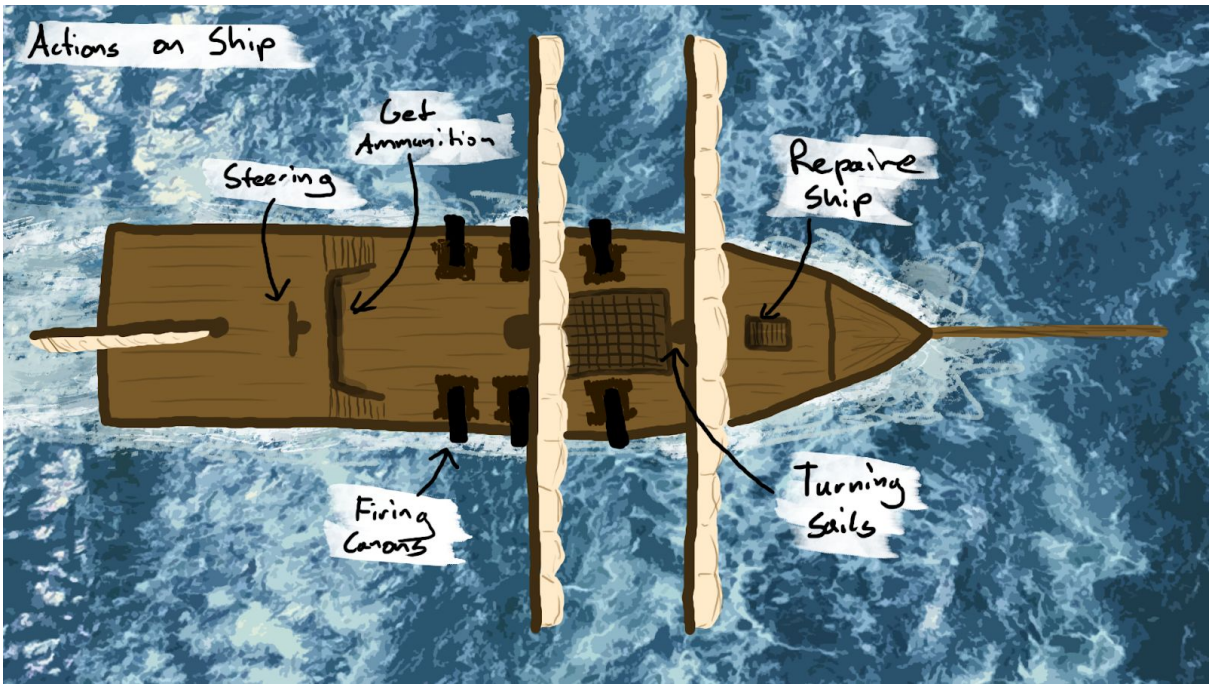
effect.
All consumables will spawn randomly on the map. When the ship goes through the power-up, it will be activated instantly. Other materials (cannon balls, planks etc.) will be picked up the same way. These consumables can give positive as well as negative attributes, depending on the context. Power-ups vary from speeding or slowing the ship, freezing or inverting the current, making the ship invulnerable or making it unaffected by the current.

1.1.2. Concept Art and Sketches

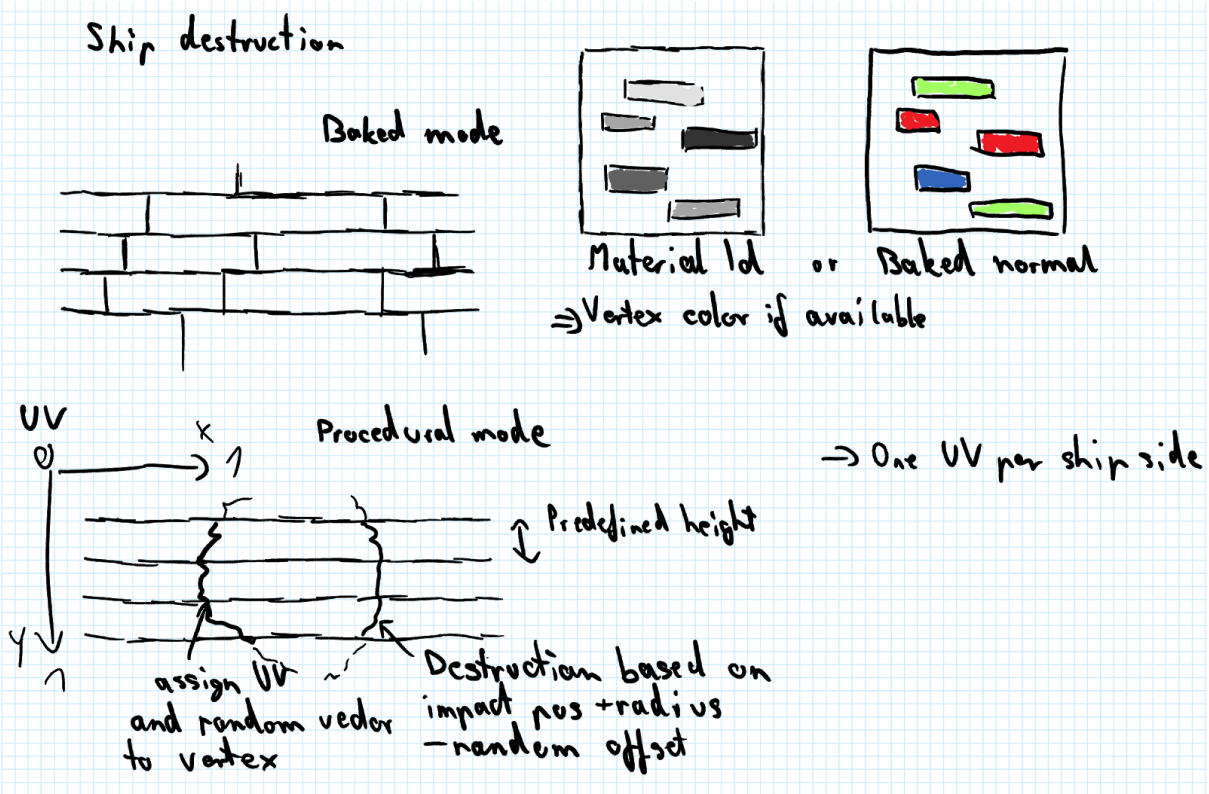
Concept art that conveys the general feel of the game. The ships are caught in the maelstrom and fire at each other with cannons, trying to sink the other enemy ship. Also in the background, a storm is slowly building up, ready to also release his wrath on the ships.

Concept art showing the action on the ship. The ship is already deep into the maelstrom, leaning heavily towards the center. The captain, controlled by the player, is steering the boat towards the outer banks of the current, while commanding the crew to adjust the sails (guy climbing up the masts).

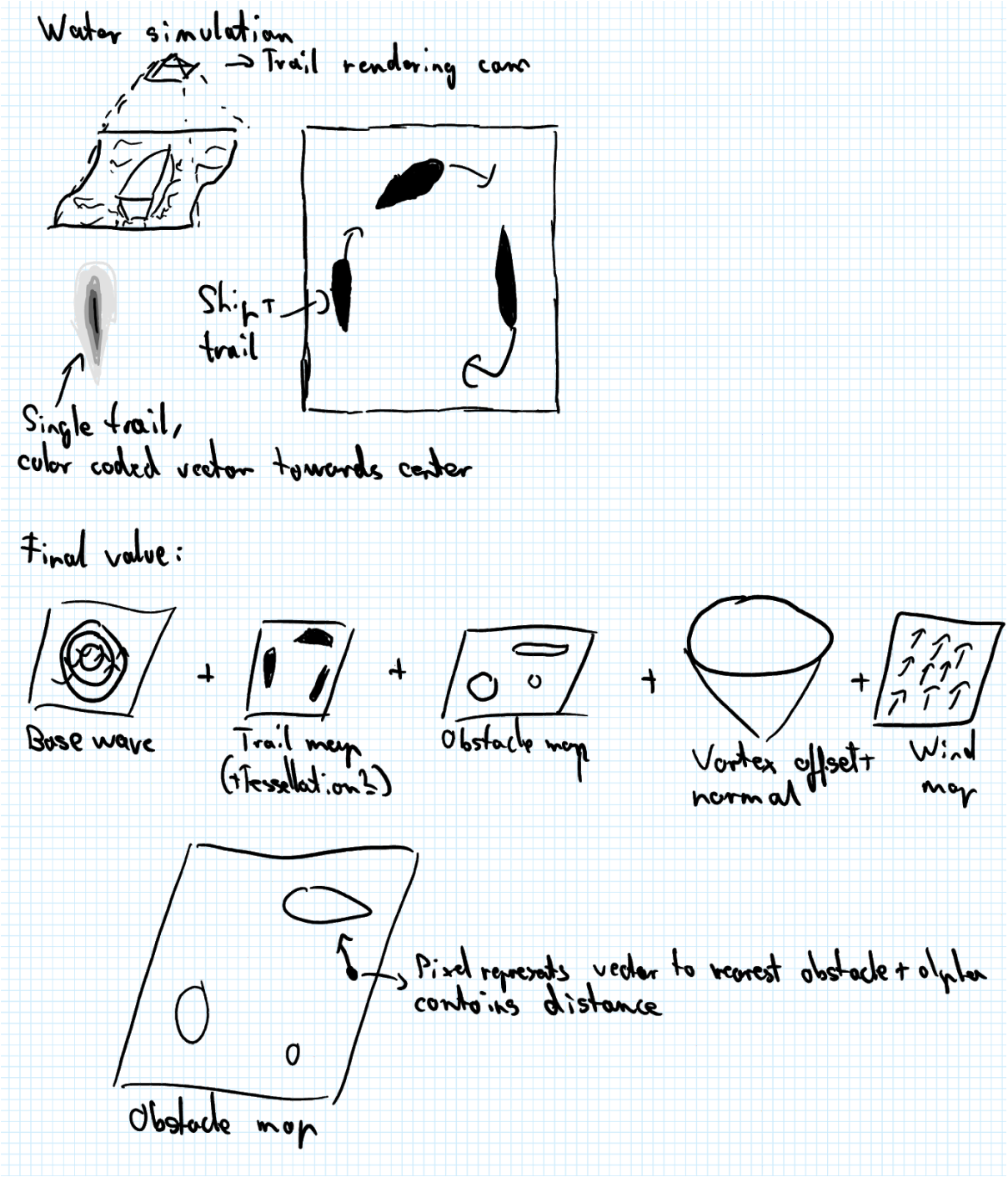
Concept art showing a overhead view of the naval battle. The two ships are firing cannon balls at each other, while caught in the current. The maelstrom slowly pulls them towards his deadly center



Sketch of the actions on the ship. As the players and the crew can move freely around the deck, this sketch shows a few action that can be performed. The steering of the vessel can be done on the higher deck, while below the storage for ammunition and building materials lies. Crew members will automatically go get the ammunition and building materials causing the ship feel alive with action. Cannons are stationary on the deck and can be fired into both directions (also one direction might not be as effective during battle). For repairing the hull of the ship, crewmates will disappear below deck with building materials. The sails can be repaired by climbing the mast. The sails can be turned at the mast.



Technical sketches to the ship destruction. The sketches show two different implementation methods for having destruction show on the ship. One is the baked mode, where the destruction will be predefined and the second is the procedural mode, which will base the destruction on impact position and radius.



Technical sketches to the water simulation. These sketches show different components of the water rendering maps. It consist of the base wave, trail map, obstacle map, vortex offset and normal and the wind map.

1.2. Technical Achievement

There are 2 main technical achievements in this project, Steamworks support p2p networking and a complete water and weather simulation for the level running on the GPU.

1.2.1. Steamworks P2P networking

https://partner.steamgames.com/doc/home
https://steamworks.github.io/
In addition to interfaces for accessing different kinds of user data from Steam, the steamworks API provides a p2p networking interface. By providing optional relay servers, the interface assures a connection between host and client without any NAT issues. After setting up a lobby containing all the players, network packets can be send to other players in it using the corresponding SteamId. The Unity plugin TNet will be used for object serialization, remote procedure calls (RPCs) and remote creation calls (RCCs). The existing udp/tcp backend of TNet will be replaced by the Steamworks API.
In order to access Steamworks C++ API, the Steamworks.NET wrapper will be used.
The Steamworks API enables additional, optional features like online leaderboards, cloud saves and community services to be added to the game.

1.2.2. Water simulation

https://docs.unity3d.com/2018.1/Documentation/ScriptReference/Experimental.Rendering.AsyncGPUReadback.Request.html
A compute shader supported simulation of the water will be used to create unique water behaviour every game. To get data like ship positions back to the CPU code, Unity's async GPU readback functionality will be used. A GPU based solution allows for a more complex wave simulation, as well as proper interaction between ships and water. Tessellation will be used to improve the visual quality around the ships.

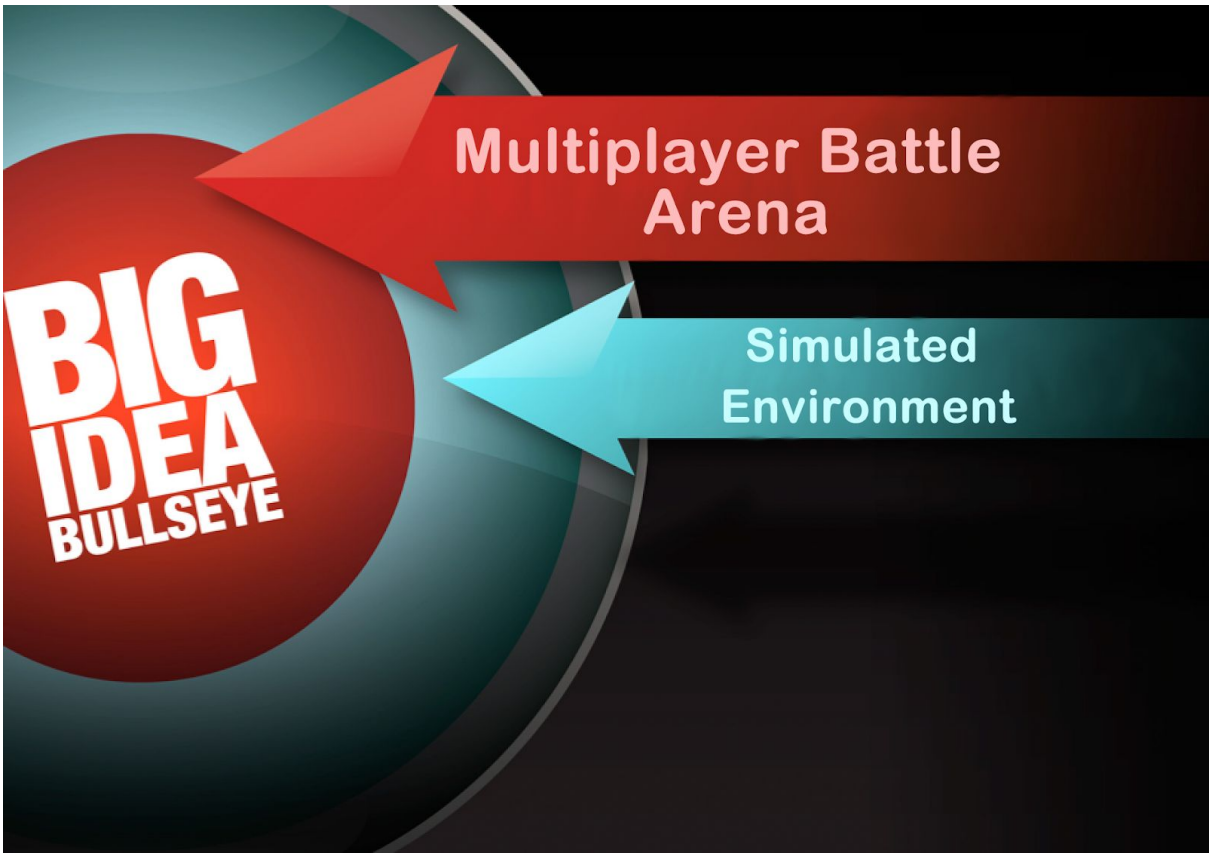
Concept art for the final vortex structure. It shows how the structures of the waves could look in the final game representation.

1.2.3. Weather simulation

The core of the weather simulation is the GPU based wind simulation. The wind affects the water simulation, the gameplay behaviour of the ships (as well as the visual behaviour of the sails) and visual indicators on the map like trees on islands. A PhysX based cloth system in combination with simple C# logic will be used to enable proper interaction with the sails.

Advanced weather affects like volumetric clouds might be added to the game if there is enough time left.

1.3. “Big Idea” Bullseye



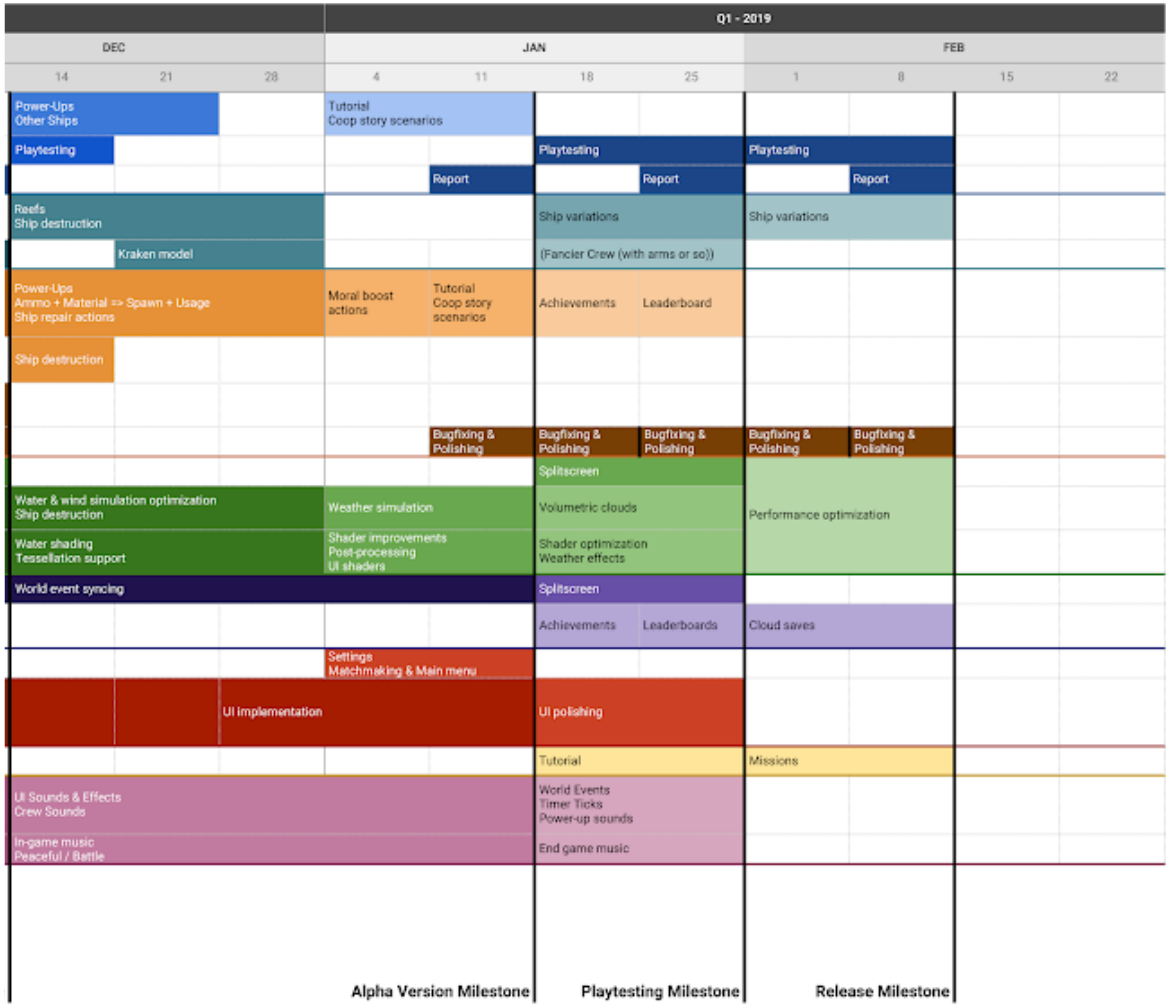
1.4. Development Schedule

For high resolution image, see the Wiki page of our project.
https://wiki.tum.de/display/gameslab1819/Plot+Twist

PHASE		DEV	DETAILS	Q4 - 2018									
				OKT				NOV					
	PROJECT WEEK:			5	12	19	26	2	9	16	23	30	7
1	Game Design		- Components			Game Concept Game Idea Technical Achievements			Controls Basic Global Events Game Duration				
		Everyone	- Playtest						Paper Prototype				
		Everyone	- Report					Report		Report			Report
2	Modelling	Domenik, Jan	- Static objects						Basic Ship-Model (untextured)		Ship Textures		
		Domenik	- Dynamic objects								Simple Crew		
3	Scripting	Domenik, Jan, Kagan	- Gameplay						Simple ship movement Cannons (autoaim) Player movement		Ship Movement with "physical" influence Cannon usage (Camera + Effect on other ships)		
		Jan, Kagan	- Systems						Game Input System		Crew AI System		
		Alex, Jan	- Tools						Add basic debug tool Add rewired input Setup CI		Simulation debug tools Advanced networking debug tools		
		Everyone	- Bugfixing & Polishing										Bugfixing & Polishing
4	Rendering	Alex	- Render backend						Setup SRP		Lighting backend		
		Alex	- Tech art						AsyncGPUReadback functionality		Vortex simulation Flow map rendering	Trail rendering	Cloth rendering Wind simulation
		Alex	- Shaders								Ship shader Sail shader		
5	Networking	Alex	- Gameplay								Ship syncing Crew syncing		
		Alex	- Steam						Add Steamworks.NET Update AppID Add Serialization backend				
6	UI	Kagan	- Menus						Start Game / End Game				
		Alex, Jan, Kagan	- In-Game						Ingame Debug UI		Core UI design		
7	Level Design	Everyone	- Level design										
8	Sound	Kagan	- Effects						Cannon fire Hull damage Ship Movement		Reloading Repairing Environment Sinking		
		Kagan	- Music								Main Theme In-game Music		
							Game Idea Milestone			Prototype Milestone			Interims Demo Milestone

Funktional Minimum
Low Target
Desirable Target
High Target
Extras

PHASE		DEV	DETAILS
	PROJECT WEEK:		
1	Game Design		- Components
		Everyone	- Playtest
		Everyone	- Report
2	Modelling	Domenik, Jan	- Static objects
		Domenik	- Dynamic objects
3	Scripting	Domenik, Jan, Kagan	- Gameplay
		Jan, Kagan	- Systems
		Alex, Jan	- Tools
		Everyone	- Bugfixing & Polishing
4	Rendering	Alex	- Render backend
		Alex	- Tech art
		Alex	- Shaders
5	Networking	Alex	- Gameplay
		Alex	- Steam
6	UI	Kagan	- Menus
		Alex, Jan, Kagan	- In-Game
7	Level Design	Everyone	- Level design
8	Sound	Kagan	- Effects
		Kagan	- Music



1.5. Assessment

Fatal Tides will feature a smooth multiplayer experience with fancy, simulated water and weather effects.
The gameplay consists of navigating your ship through a giant maelstrom, while trying to sink your foes and surviving their attacks. To be successful at this task, you will have to delegate your crew according to the situation and your tactic.
Since the the gameplay is fast and action-packed battle arena, people who like a replayable and fun multiplayer experience with their friends will also like Fatal Tides.

2. Game Prototype

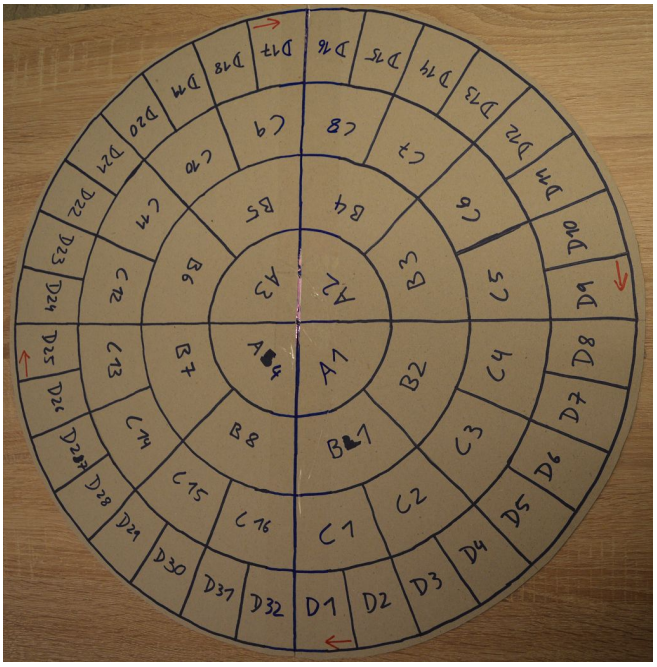
2.1. Overview

To visualize our game, we created a paper prototype that covers the main aspects of our game: Multiplayer battles supported by interesting maelstrom mechanics. The paper prototype is a four person board-game where each player commands his ship, assigns orders to their crew and tries to avoid getting consumed by the kraken nesting in the center of the vortex.

2.2. Rules and Gameplay

2.2.1. The Prototype Components

The board game consists of four parts: A map of the maelstrom, four paper boats, four ship cards and 11 figures for each ship card, representing the crew members.

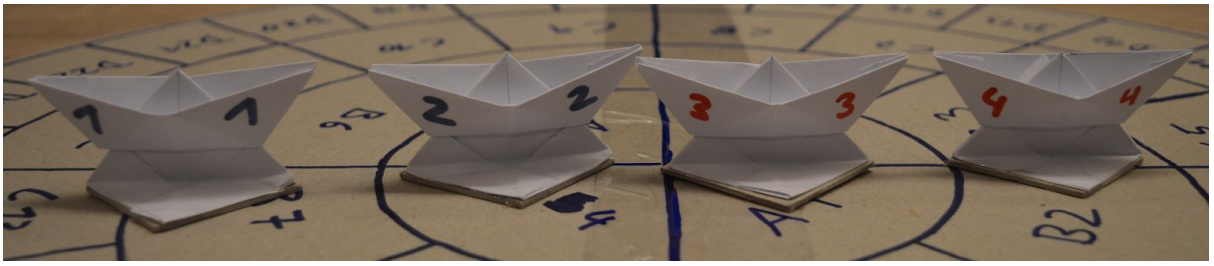


Map

The map is a circular cardboard divided into four rings. Players will use their paper boats to simulate the ship. Each paper boat can be placed on a region marked in the map. The number of regions double on each outer ring. The rings are named from A to D while the regions in each ring takes a number as a suffix. Each additional outer ring has twice as many fields as the preceding inner ring.

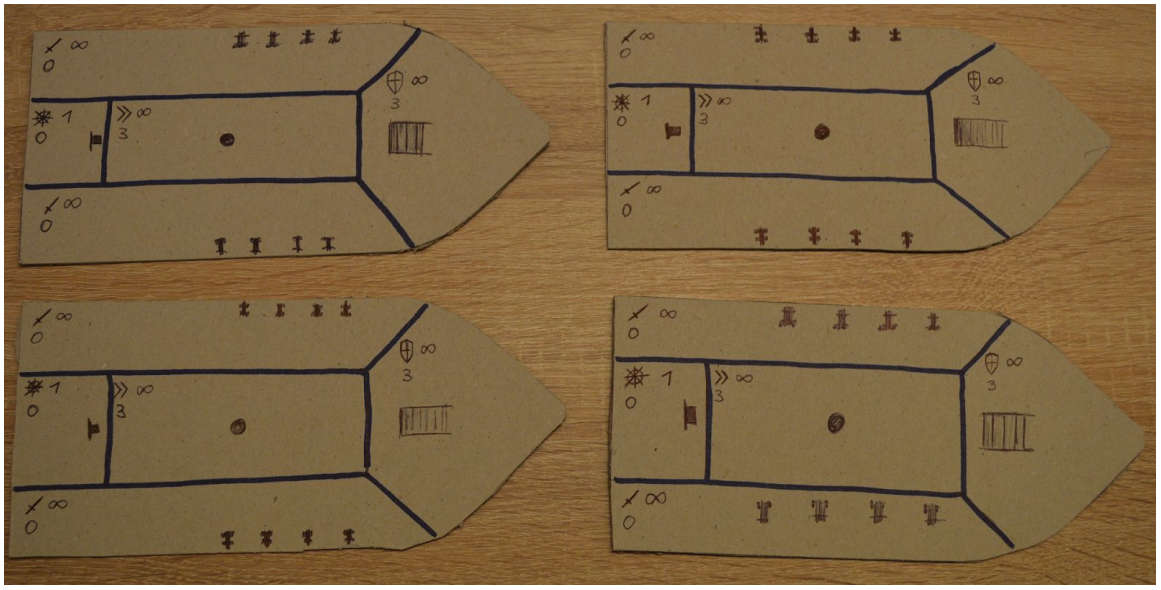
Paper Boats

Each player has his own paper boat that is numbered. These boats will represent the actual ships and interactions with other players are realized through them. Your ship has **25 hitpoints** and if you lose them all, you are defeated and is eliminated from the game.



Ship Cards and Figures

As it will be on our game, the players have 11 figures at their disposal. These are divided into 10 crew members and 1 captain. These figures can be placed on the ship cards to assign them to different tasks. The captain counts as three crew members when put on another task aside from steering. Many different combinations are possible and bring a variety to the game. Assignment is done via placing crew members on different parts of the ship card. A 12-sided dice and a 6-sided dice will also be used for combat and kraken mechanics.

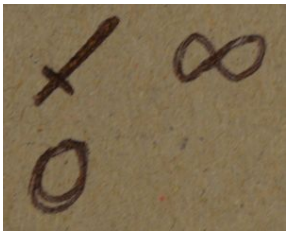


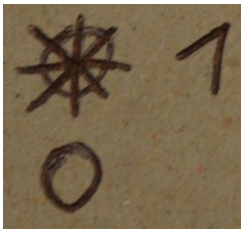
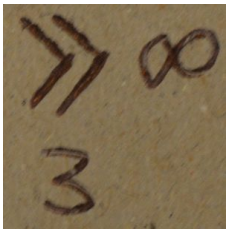
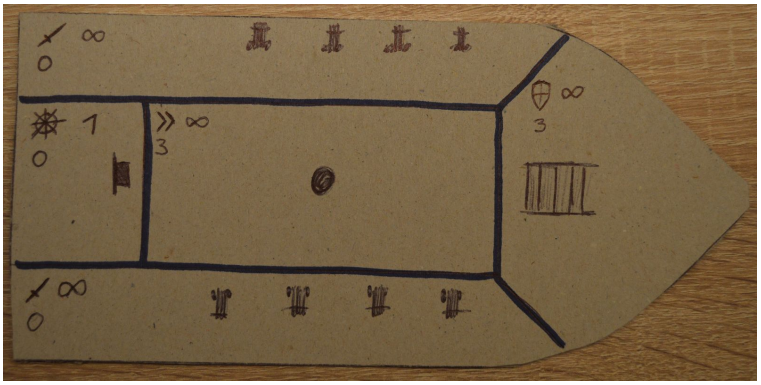
2.2.2. Boat Tile Crew Positions

The player can distribute the crew on 4 different abilities:

- Offense: Symbolized with a sword. Placement is on the sides of the ship, where the cannons are placed. Your offense points indicate how much damage you can deal with your cannons. The base value is zero and the number of units that can be placed here is unlimited.
- Defense: Symbolized with a shield. Placement is on the bow(front) of the ship, which has a stair leading to the basement. There the crew makes repairs and prevents the ship from taking additional damage. A base value of 3 is present on every ship. Your defense value indicated how much you can shield yourself from enemies' cannon barrage. The number of units that can be placed here is unlimited.

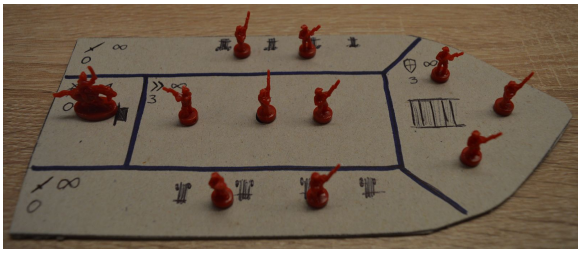
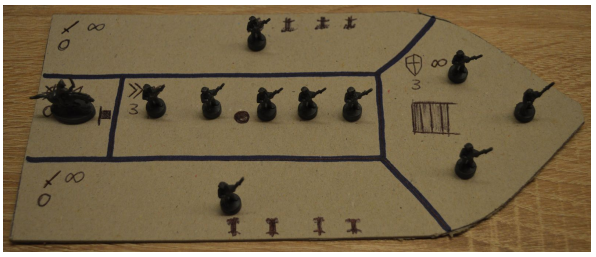
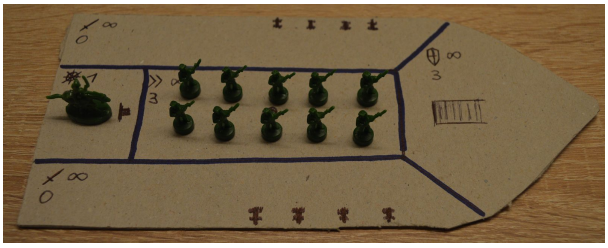
- Steering: Symbolized with a ship wheel. Placement is on the back of the ship. Only one person (the captain) is needed here. Steering allows your ship to navigate to an outer ring. The base value is zero.
- Movement: Symbolized with two angle brackets, implying motion. Placement is on the center of the ship so that the crew can adjust the sails. Each ship has a base value of 3. Your movement points determine how much you can traverse the map in a single round. The number of units that can be placed here is unlimited.



2.2.3. Crew Variations

Your game strategy may change on each round depending on the situation, so the player will usually shift the crew members to other parts of the ship and adapt to different situations in the maelstrom. Some of the possible assignments are presented below:

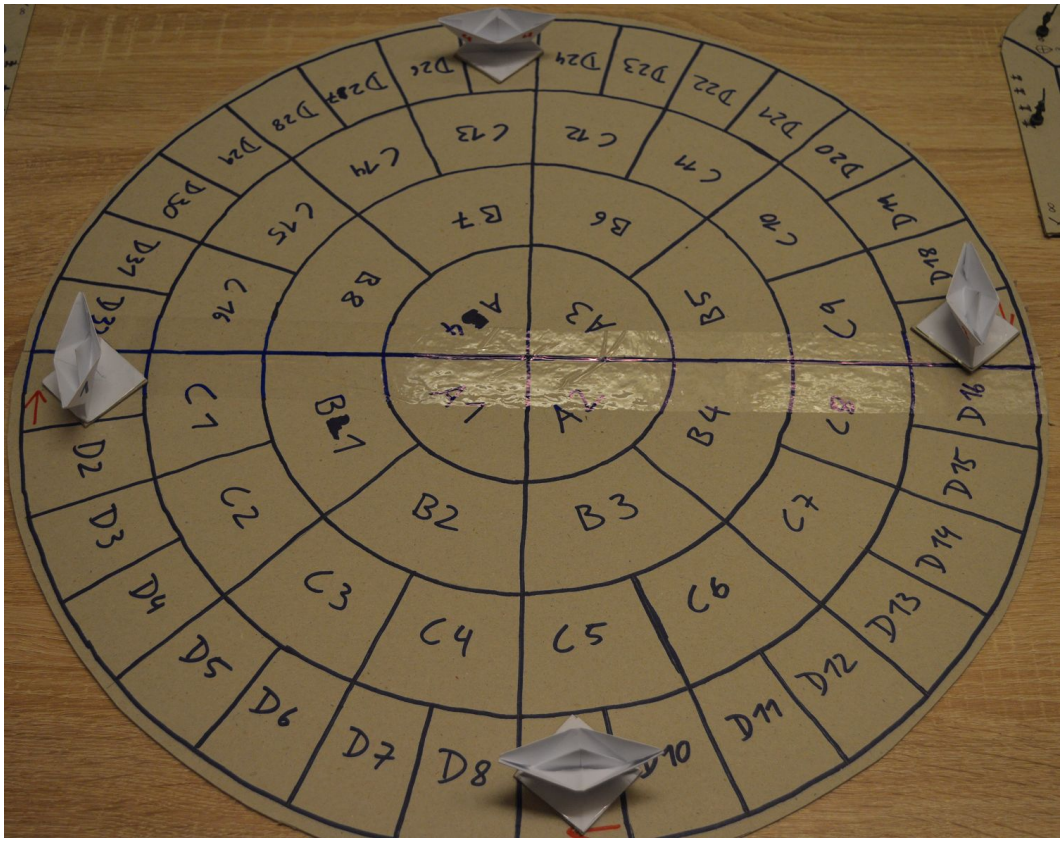





- Top left: A standard placement. 4 men on offense, 3 on defense, 3 on movement and the captain on steering.
- Top right: Similar to top right but 2 men are shifted from offense to movement, enabling one additional outer ring traversal.

- Bottom left: All crew members are placed on movement to be able to move rapidly, either out of harm's way or to reposition for a deadly attack on the next round.
- Bottom right: All crew members apart from one is positioned on the cannons. With this, the player has an offense value of 12, a defense value of 4 but no movement and steering: An all-out attack strategy.

2.2.4. Start Positions

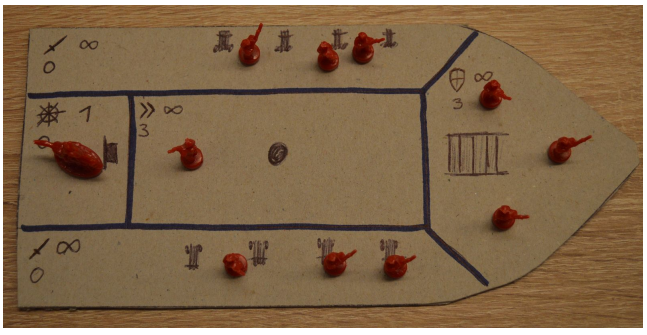
Each player starts at the outer ring and as far as they can from others. Below: Players start on D1, D9, D17 and D25 (marked by a red arrow).



2.2.5. Actions per Turn

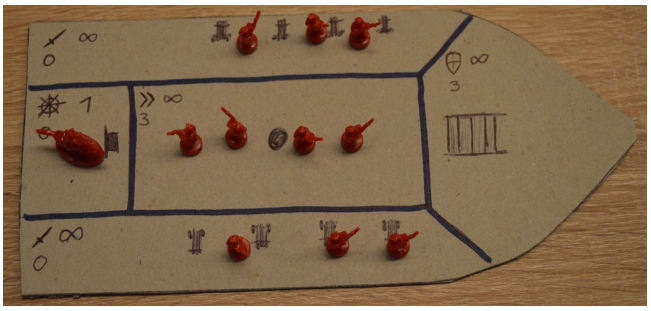
The game consists of rounds, while rounds consist of four turns. The game continues until one ship is remaining. In the first three turns players can participate while on the last step the kraken makes its move.

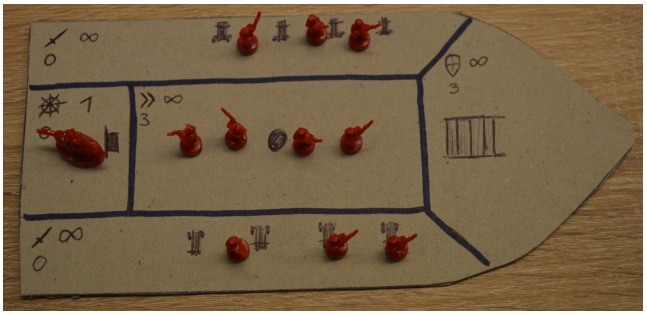
2.2.5.1. Reposition Crew

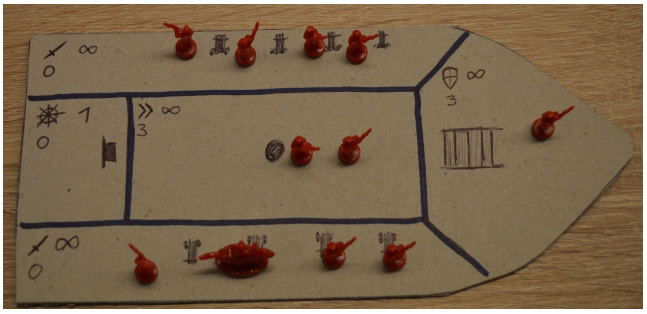


The players assign their crew to new positions in the first turn. The assignments are not revealed to players until turn three so that their actions are not influenced. The captain counts as

three members while he is not on steering duty.





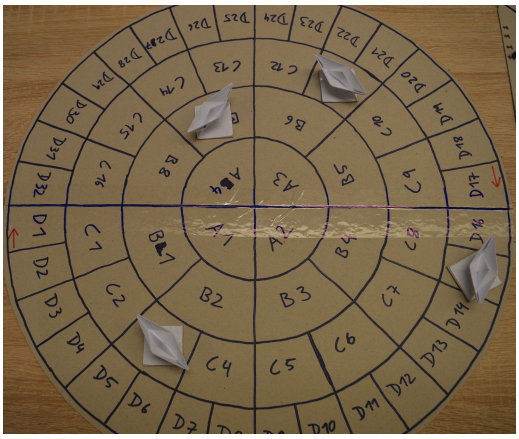


2.2.5.2. Move Boat

In this turn players move their ships according to their movement points. The movement is in the direction of the maelstrom (clockwise) and players cannot move against the current. Moving to a region within the same ring and moving into an inner ring costs one movement while moving to an outer ring costs two movement points. Keep in mind that your captain needs to be steering in order to move to an outer ring. If you have five people on movement, you can travel two outer rings. If you have 10 people you can travel three outer rings.
Since the current is also moving the ship forward, every ship gains a base 3 movement speed. Since the current is faster in the center, moving through the inner rings is also faster for the ships and this is also realized in our map.

Calculating where to go

Every player calculates his movement and decides on which field he wants to go. This decision is stored somewhere (on paper). After everyone is ready, the everyone shows their trajectory to others and moves their ship.

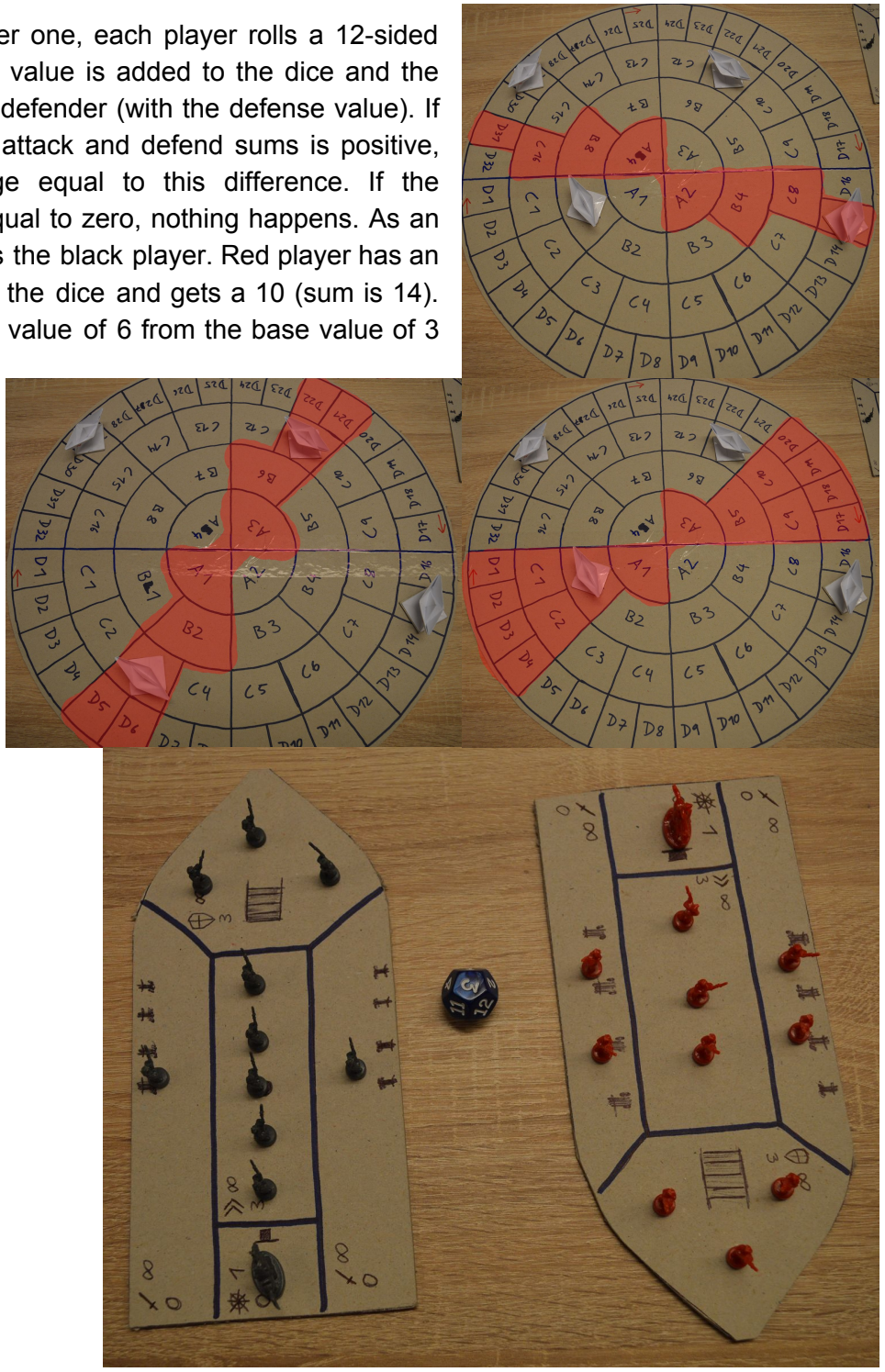


2.2.5.3. Attack

After the movement turn comes the battle turn. If two ships face each other on the maelstrom, they can fire at each other. If there are multiple ships in range, the player chooses which one to attack. The firing width of the ship depends on the ring the ship is in,

as can be seen on the images below: A player can find more targets, if his ship is closer to the center.

When a ship fires at another one, each player rolls a 12-sided dice. The attacker's offense value is added to the dice and the same things is done for the defender (with the defense value). If the difference between the attack and defend sums is positive, the defender takes damage equal to this difference. If the difference is less than or equal to zero, nothing happens. As an example: Red player attacks the black player. Red player has an offense value of 4. He rolls the dice and gets a 10 (sum is 14). Black player has a defense value of 6 from the base value of 3 and 3 men assigned. He rolls a 7. The sum is 13. The difference is 14-13 = 1, so the red player inflicts 1 damage to the black player.

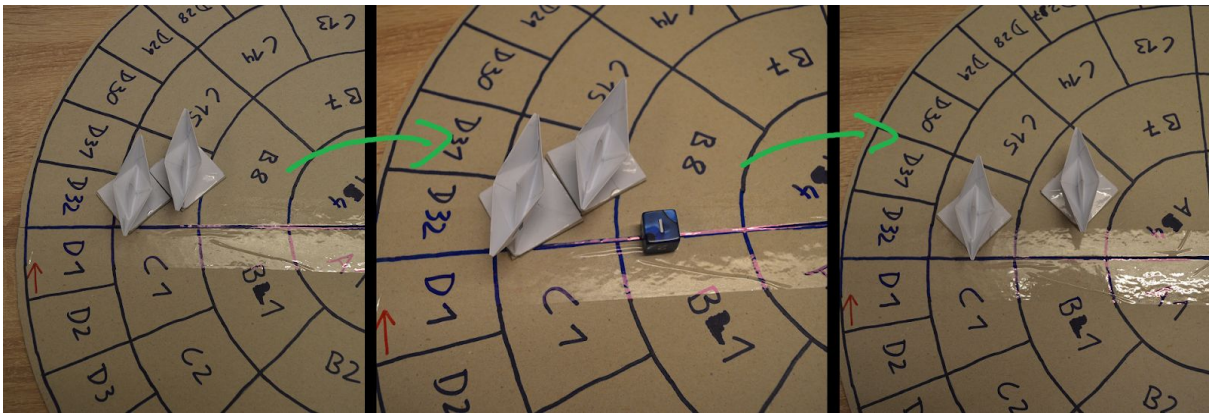


2.2.5.4. Moving Pieces Towards Center

After the battle is done, the maelstrom pulls every player to the center. Every ship moves one ring inward. If a ship is in the center of the maelstrom the kraken comes hunting for him (see 2.2.6.2).

2.2.6. Special Cases

2.2.6.1. Two Players on Same Field

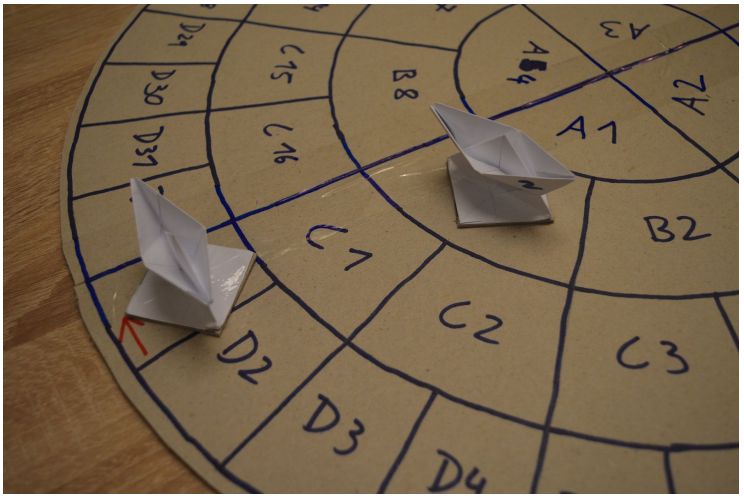


If two player land on the same field after a movement turn or at the end, when the maelstrom pulls the boats towards the center, each player rolls a 6-sided dice and the loser is rammed to the inner ring.

2.2.6.2. Player on Center Ring

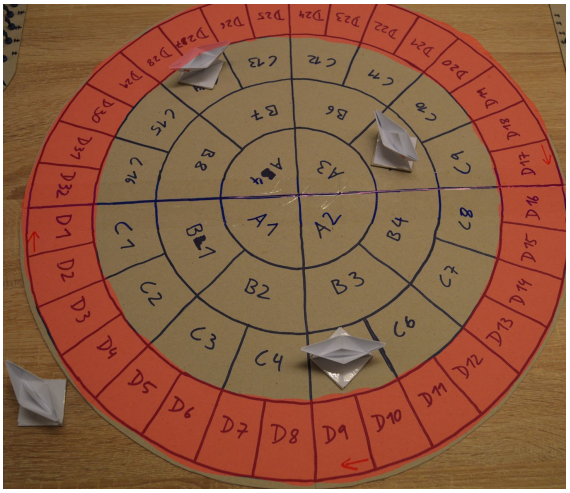
At the last turn, if any ship is on the center, the kraken tries to sink it. A 6-sided dice is rolled and the ship takes damage equal to the roll.

2.2.6.3. Attacking on Same Side



If the ships are on the same quadrant on the battle phase, the attacker's offense value is doubled. This is because the ships are near each other so that cannons won't miss their target as easily. Being on the same quadrant is risky but a player can also inflict serious damage to an assailant.

2.2.7. Players Get Eliminated



When a player gets eliminated, the maelstrom gets stronger. When starting a new round, the most outer ring will be inaccessible. With every ship sunk, the map gets smaller so the game length is exponentially reduced, preventing boredom.

2.3. Experience

When we played our prototype we realized that it got really exciting and competitive. This is due to the nature of multiplayer battles. We also came up with the rules that support our core ideas like giving each ship a base movement speed, making traversal faster in inner rings or doing more damage when ships are near each other. We also needed to adjust some numbers for offense, defense, hit points etc. which also took some time to figure out.

2.4. Learnings from Prototype

We think that we are on the right track since our gameplay experience was really fun and immersive. Prototyping enabled us to see where to focus on our project and merge additional details to provide a good playing experience.

2.5. Game Revisions

We are thinking about doing some simplification for our early version. Complex details will be handled if we have enough time for them as we will first focus on making a stable game with the core and fun elements in it. The importance of balancing has also shown itself: Poor balancing can lead to frustrating experiences and we will need to fine-tune some values to make the game more enjoyable.More time will be reserved for playtesting to find the optimal balance.

3. Interim Report

During our first weeks of development, we made good progress and finished almost all our level two tasks and started working on our level three tasks. We had some minor setbacks and underestimated the challenge in creating the movement for the ship and the simulation for the vortex.
We can currently move with up to four players, over the steam network, across a dynamically generated maelstrom, while assigning the crew to different posts on the ship. The cannon usage is already implemented but not merged into our master branch yet, which will happen in the coming days.
Despite our setbacks, we are still optimistic that our implementation goals of all layers are achievable. Also we are still on track and will most likely implement all our task from layer three and most of layer four, which should give our game a good polished feel at the end. Also, we only had some minor adjustments in gameplay concerning controls for the ship and crew, based on our earlier vision.

3.1. Scripting and Gameplay

In the gameplay and scripting section, we worked on all the basic components of our game and got these up and running. The movement of the ship, cannon usage and crew movement are the core and basis of our gameplay. All these systems need to run smoothly for us, so that we can move on to the higher level gameplay and scripting elements, such as smart crew movement, sea battle and power-ups.

3.1.1. Ship Movement

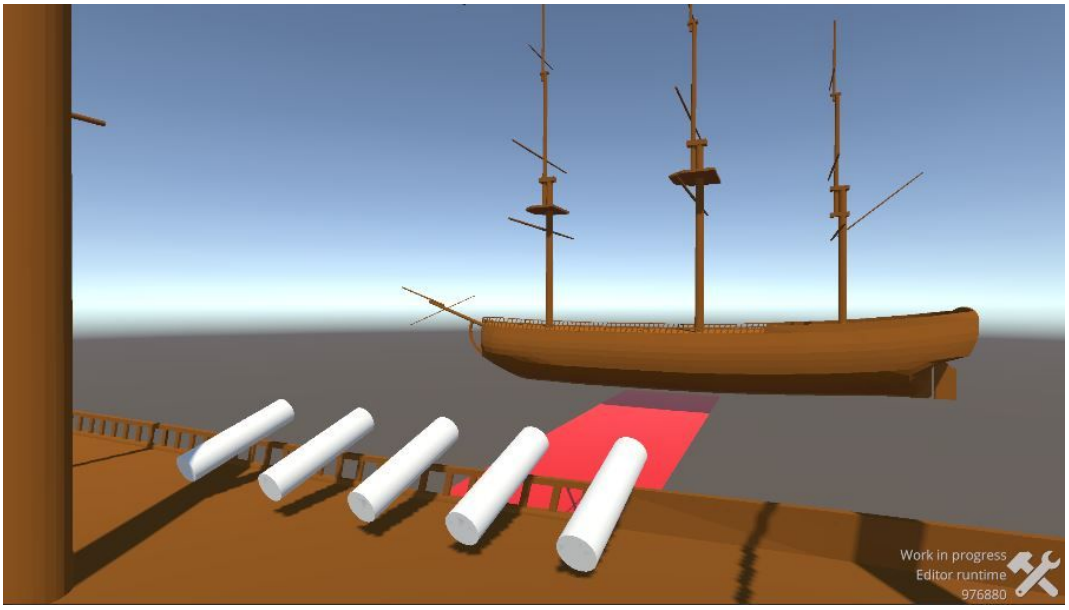
The ship movement was very hard to get right, as there are a lot of moving and rotating components that interact with each other. The movement went through three major design and scripting phases: basic movement in a very simple maelstrom, physics based movement and movement depending on the maelstrom heightmap.
For the basic movement, we build a simple placeholder maelstrom from planes at a 45 degree angle, so we could test our early movement scripts concepts. We then went ahead and moved the boat along the Z-axis of its transform, while rotating it around the Y-axis to simulate the rotation of the maelstrom. Also one can rotate the boat to the left and right to influence the steering. The boat then moved based on its collider forward on the simple maelstrom. Here we had some trouble finding the right rotation axes, as these were constantly changing through all the different rotations and movement affecting the ship.
The next step, basing the movement on physics, was easier, as we only had to replace the transforms with impulses affecting the ship. The hard part here, was finding the right force, as these ships have a quite large mass, a large impulse was needed.
The last step was the most complicated one. As the basic maelstrom simulation heightmap was finished, the movement could be adjusted to this input. For this, we have to calculate the orientation of the ship, based on the height sample at four sample points. These four points are located at the left, right, front and back side of the ship. With this we created two axes along which we could orient the rotation of the ship, left to right and front to back. With

simple tangential calculations we got our angles and the mean height location of the ship, at which we can position and orient it.
This was the easier part, the more complex job was to apply these orientations and positions toward the moving ship. After a lot of experiments and research on calculations with quaternions, we finally found a way to apply the correct numbers to our ship. We first apply the rotation around the Y-axis on how the user wants to rotate his boat. Afterwards the rotation around the Z-axis follows, tilting the boat into the maelstrom. Lastly we added the rotation around the X-axis, making the boat sit perfectly on the water.
Also the camera rotation around the boat was added, so the surrounding of the maelstrom can be watched. The movement of the boat is now up and running and only needs some further tweaking to make it feel just right. The screenshot below shows the boat in action on the vortex.

3.1.2. Cannon Usage

Firing cannons and shooting each other is an essential part of the game which needed an early implementation. This core gameplay mechanic was easy to implement but hard to nail the details on: The cannon fires through a button trigger -in the PC this is the space bar- and a cannonball is instantiated at the tip of the cannon. A impulse force is added to the cannon, whose power can be adjusted through the script. The user can also rotate the cannons to any direction with of course some restrictions. The cannons are on both sides of the ship and the user can switch between these cannons and fire from the side of the ship that he/she faces. There is a separate camera on the ship that rotates to each side to clearly see the cannon targets. Every ship has a health value right now and if the ship gets hit by a cannon, the collider is triggered and the ship takes damage. The damage is synced through the network.
One hardship we encountered while implementing was one of the features we wanted but we did not get it to work until this deadline: Calculating cannon trajectory. The idea is to give the user a visual feedback on the cannon's trajectory so the user has an idea where to shoot at. Unfortunately the trajectory calculation and the real trajectory of the cannon do not match

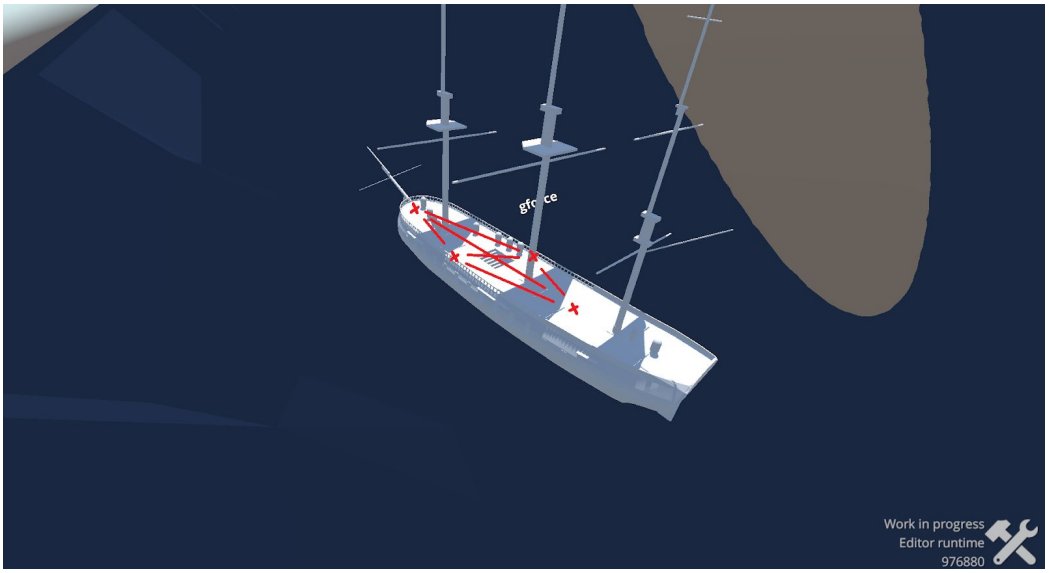
at the moment. We will fix this by reviewing the physics calculation implemented or find another way to do it.



3.1.3. Crew Movement

We also went ahead and implemented a simple crew movement to four action points on the boat (see screenshot below; X marks an action point and a line marks the movement path). Here the most challenging part was to keep the crew on the boat and stop them from falling off. The movement is very simple and does not account for any objects or people in the way, which will be the next developing step. The only thing that is implemented, is that the crew will always move on the ship and not float above the deck.

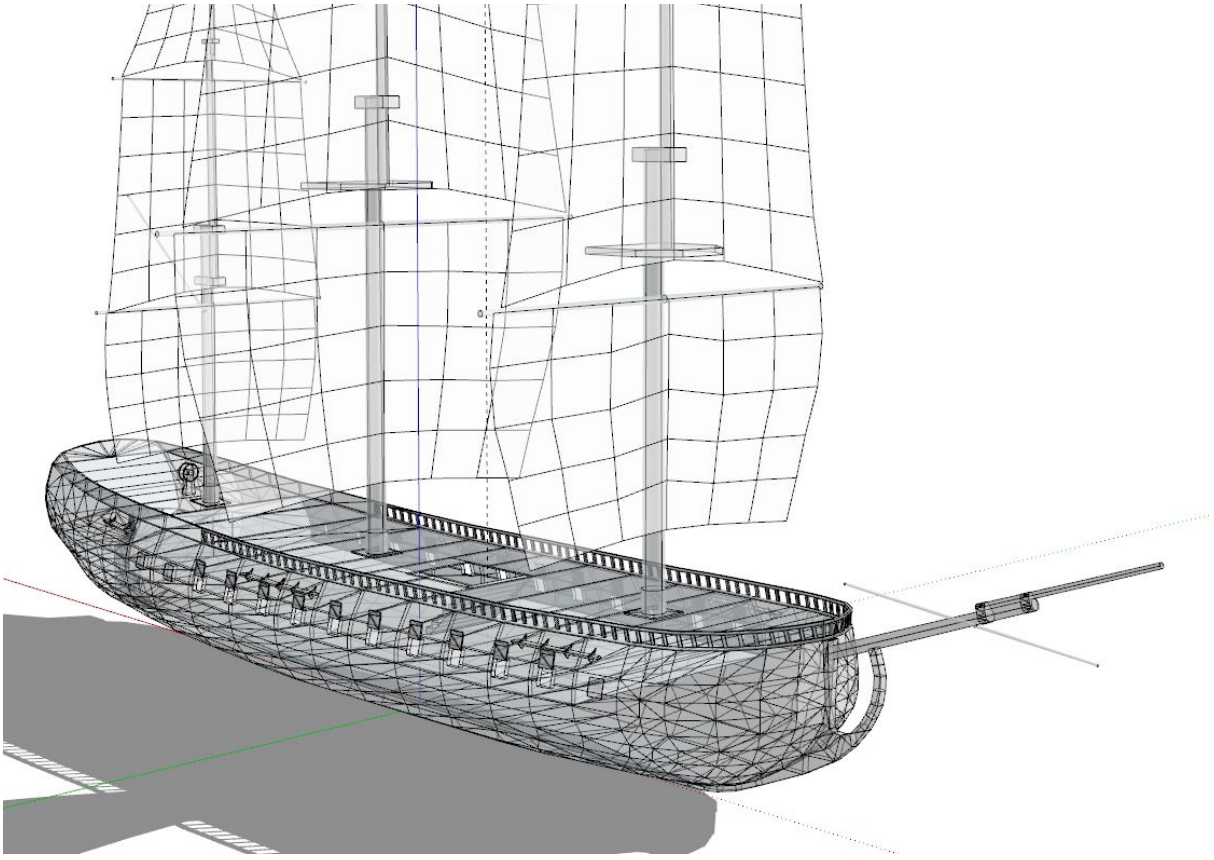
Here, we first had the captain also implemented as a moving object, which the player could steer and interact with. However, after first tests, we found that steering alone was already a tough job and that when the player was walking on the deck, he had no mind for steering, causing the boat to slowly drift towards the center. With this, we decided to permanently place the captain at the steering wheel of the boat.



3.2. Modeling

As for the 3D model of the ship, we had an additional requirement besides overall quality. We needed it to be a clean model with a modular structure so that we could add destruction-effects later on. Therefore most freely available 3D models found online didn't suffice and we had to create our own model.
Since we wanted an iconic looking ship as our base ship, we used a real ship as a reference. The re-built Frigate "Hermione" was the perfect fit as a classic ship from the 18th century with a modern-time reconstructed version for high-quality source material.

In its current state, the model features the most basic components: Moveable masts, sails, a rudder, a steering wheel and a lower deck with cannons.



As seen in normal content creation pipelines we also created additional "High-Poly" meshes of several components to create normal- and bump-maps later in our development schedule.

Currently, the only dynamic objects onboard are the sails. We used the unity cloth physics to animate them and created a custom shader for double-sided face-rendering the reduce clipping and visual errors.
During later development, we plan to use the modular masts to rotate the Sails according to the wind and rotate the steering wheel and the rudder according to the player input.

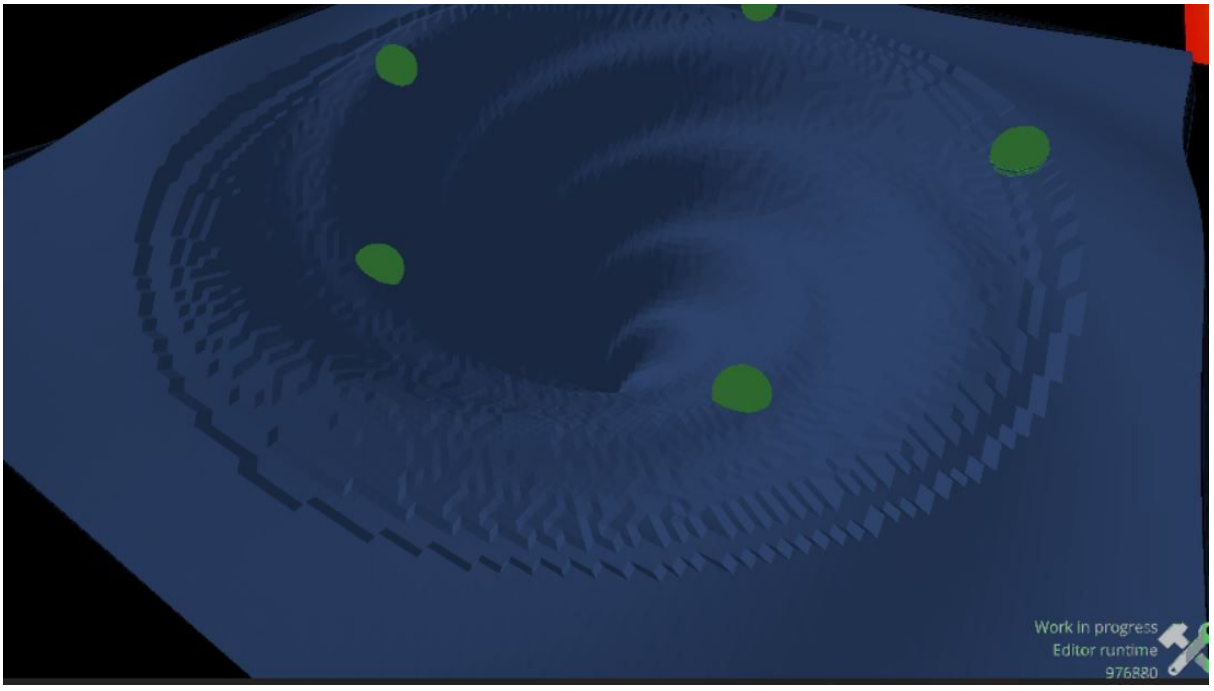
3.3. Rendering

3.3.1. Water simulation

The water simulation is the core component of the game. It is the first component that is being updated every frame. This happens in multiple steps:

- The water depth is generated based on the elapsed time. Changes on the generator function allow us to change the basic shape of the vortex easily, i.e. we switched from a linear function to an exponential one to closer approximate an actual vortex structure. The result of this stage is currently being written into a simple greyscale texture.
- Next up the waves are being generated. They are modelled based on depth, angle and distance to the center. This creates the typical vortex waves you would expect. They are being written to a texture as well.
- The final map being generated for now is the flow map, which indicates the direction and strength of the flow at any position based on the wave and angle of the vortex. This map is currently unused but might affect the ship controls later.

The depth map is being read back to the CPU, so the ship controllers can access it later. Thus, the only value affecting the ships height now is the depth map. Once the pipeline is being optimized the final map will contain the wave height and flow values (in separate channels) so they can be accessed by the gameplay scripts. The whole process happens synchronously now and takes about 2.5 ms for a 1024x1024 texture. The performance impact should be fine; however, the whole process can be done asynchronously which add some latency but doesn't impact performance as much.



The current simulation output. The sphere height is determined by a C# script sampling the GPU results. The whole vortex is flat shaded for easier debugging.

3.3.2. Cloth

For the sails, a simple PBR shader with disabled culling is used. In the future, the underlying BRDF will be exchanged for one supporting more advanced lighting models to more accurately simulate the interaction of the cloth with the light sources.

3.4. Networking

For the networking, a serialization plugin called “TNet” is used. However, since this project is steamworks based, the whole networking backend needed to be written. It is build on top of a lobby/matchmaking system. Currently the editor handles all the setup for easier debugging, however you’ll be able to do the lobby setup manually through the main menu in the final game.

Thus far, the ships, the crew members and some global simulation variables are being synced with client authority. However, there are still some open issues that need to be fixed in the future, including some movement interpolation, probably client side movement prediction and server side collision.

3.5. Sounds

Sound effects for firing the cannon, taking hull damage from a cannonball and ship moving in the maelstrom are added and implemented. In order to create a more immersive experience, these core sound effects are a must as they give audio feedback to the most important gameplay mechanics in game. As of now cannon and ship sounds are static meanwhile taking hull damage can fire a random sound from an array. THe sounds were mostly taken from the internet (no copyright infringement) or edited from various sounds using digital audio editing software.

4. Alpha Release

Following the weeks after the interims report, we were hard at work at finished all of our desired targets (we moved some back to high, as we shifted a few focus points of our alpha release) and most of our high targets. With this, all desired functionalities of our game are currently working, leaving only work for the next in the aspects of fine tuning the gameplay and controls, graphical polishing, adding more sounds and models, as well as bug fixing, which arise during playtesting.
In addition, we released our first alpha build of our game on steam, enabling to give out steam keys to our selected play testers over the next weeks. Also, we can easily update our game with the newest fixes and adjustments, and all our play testers will receive the update automatically.

4.1. Scripting and Gameplay

All components we wanted in our game are up and running. Some still have some minor issues and most still need fine tuning. We completed the ship movement, cannon usage and crew movement. On top of this, we added crates for ammunition and repair material refill and boost triggered by the assignment from the crews.

4.1.1. Ship Movement

The ship movement received some minor tweaks and was adjusted to fit to the vector fields of the maelstrom. The boat will now rotate around the center of the vortex and will slowly drift towards the middle. We also disabled the ability to turn the boat around in the maelstrom and sail into the wrong direction. The vessel will now continue into the turning direction of the vortex, only at a slower speed and backwards.
In addition we tweaked the parameters for the movement and made them available via a config file and debug console command, for easier adjustment in the playtesting phase. The movement still needs adjustment to feel just right, but we are getting there.

4.1.2. Cannon Usage

The cannon behaviour script was refined to render the trajectories for each cannon accurately. At first, we tried an unorthodox approach to cannon behaviour to reduce the load on the network. Colliders were added at runtime to each trajectory line. When a player fired the cannons, any collision between these colliders and an enemy ship at that time was checked. If colliding, the firing player broadcasted that he hit the enemy, also taking away health points. We tried this so that cannonballs would not be synced over the network, aiming to reduce lag. However, after tryouts, we saw that it was too unrealistic and could hinder the fun and immersion. Our second approach syncs the cannonballs and each player checks if they get hit. If that is the case, they broadcast this event to the network.UI is also adjusted to handle this event for everyone' clarity. Cannons on the top of the deck are also removed and moved to the lower deck. Camera positions also had to be adjusted according to that. We will look into how our implementations will affect player lag and immersion in playtesting and make fixes according to our results.

4.1.3. Crew Movement

The crew movement we had to redo large portions of it, as we adjusted them to be able to move on a Unity NavMesh. This bore some challenges with it, as Unity does not allow for moving NavMeshes. We bypassed this by creating a reference boat in the scene, on which we simulate the crew moving on board the ship over the NavMesh. Afterwards, the position of the crew is synced to the player's boat in the vortex. This allows for easy crew movement on the boat, with easy target assignment and animation for the crew.
On top of this, we added boost and station reassignment for the crew. The boost is calculated per crew member assigned to a certain station (Cannon, Repair, Sail and Fishing) and boosts the effect of each station. The boosts are more crew members manning Cannons, faster repair time for Repair, faster movement speed in the direction the boat is facing for Sail and more resources obtained from crates, when fished from the sea, for Fishing. Also the stations of Cannon, Repair and Fishing need at least one crew member to function at all.
The stations reassignment allows to choose a crew member from one station and assign him to another. This allows for more control over the crew. This whole process is aided by a HUD element for the player and will be explained in more detail under the UI section.

4.1.4. Other Scripting and Gameplay

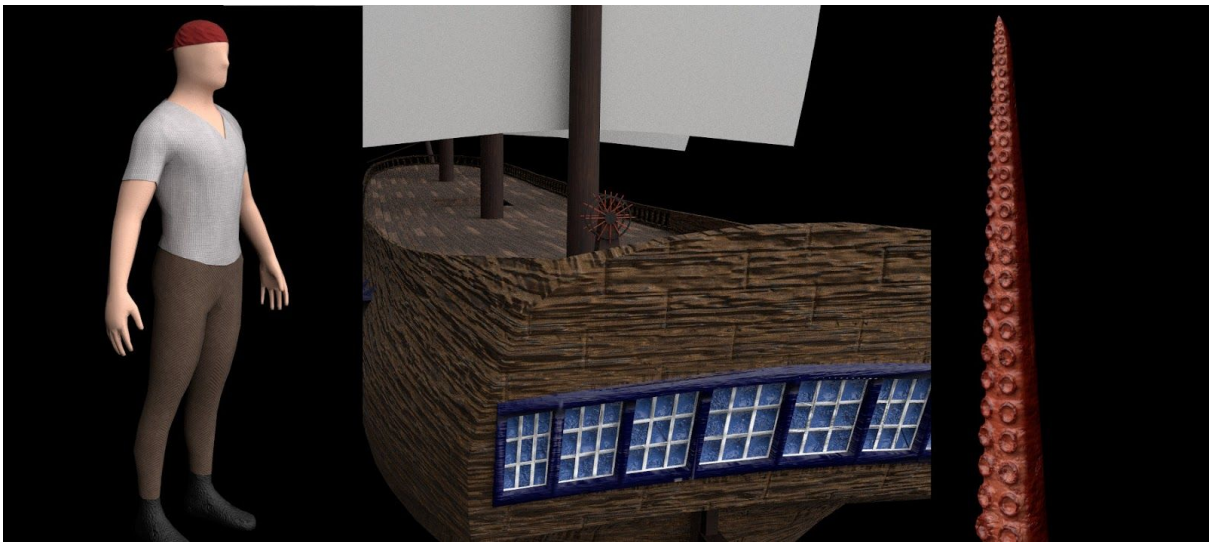
Another task we implemented are the crates. These provide needed ammunition and repair materials, which are consumed by the crew. These crates spawn below the water surface and slowly surface in the vortex and move around the center point of the maelstrom. These can be collected by the player, by colliding into them.

4.2. Modeling and Animation

In preparation for the alpha milestone we needed all basic models to be finished, which includes a mesh with good or acceptable details, textures and rigging for animated models. We managed to achieve all targets we had for this point in our timeline, with varying levels of polish.

4.2.1. Modeling Tasks

Currently we have three more complex meshes. The ship, a tentacle and a human crew member.
The mesh of the ship underwent little changes during the last month since most functional parts were already done for the interim report, but we finished the texturing of the model.
The tentacle is a completely new mesh which was created to include the advertised kraken into our game. It is already fully textured and rigged.
As for the crew of our ship, we needed a low-poly, but realistic human mesh with a simple humanoid rig for full-body animations. We created it from ground up to ensure a clean mesh which won't need a rework later on.
In addition to the complex meshes we also created a simple, textured crate to use as model and placeholder for the swimming collectible items.



4.2.2. Animation Tasks

The two organic meshes needed animations to be of use for us.
The animations created for the tentacle are currently very basic and consist of an surfacing, an attack and an diving animation plus a unused idle state.
Animating the crew would have been a much bigger challenge compared to the tentacle, based on the complexity of the rig. Therefore we used a free motion capture library to get realistic animations with useable quality. These animations are the only directly imported assets in the 3D art segment of our project.

4.3. Rendering

The rendering is done when it comes to the technical aspects of it, e.g. the water simulation. However, there is still plenty of room for polishing, like post processing, more advanced shading features, clouds, etc.

4.3.1. Water Simulation

The GPU based water simulation is the core of the game. It's the backend of the visual appearance of the water, as well as the gameplay (through GPU resource readback). In order to achieve the final result as seen in the live demo, the following steps are necessary:

1. The height of the vortex is calculated based on the world position of each point
2. The wave height is generated for each point
3. Based on the derivative of the height function, the normals are calculated
4. The height and normals are packed into a texture for the rendering and read back to CPU memory for the gameplay
5. Flow is calculated as a vector based on the normals
6. Flow intensity is calculated based on the normals and height

7. Flow direction and intensity are packed into a second texture (currently unused, will be added as external force to the ships in the future)

4.3.2. Cloth

The sails use Unity's cloth component for their simulation (which is internally based on PhysX). The external acceleration is based on the movement direction of the ship. Therefore the wind always appears to be blowing from behind the ship, which is not accurate in our vortex, but looks better in the final game. The lighting is currently a simple diffuse calculation with disabled back face culling which results in a smooth overall appearance.

4.4. Networking

The networking features for the game are all implemented by now. This includes a lobby system, the deterministic water simulation and syncing all gameplay interactions like the ship's movement, the crew members, the canons and the crates.
However, there is still quite a lot of room for improvements. I.e. the movement systems are not interpolated yet, and some movement prediction might be a good option for our game. The networking system will be monitored closely during the playtests, in order to detect possible lag or other issues.

4.5. Sound

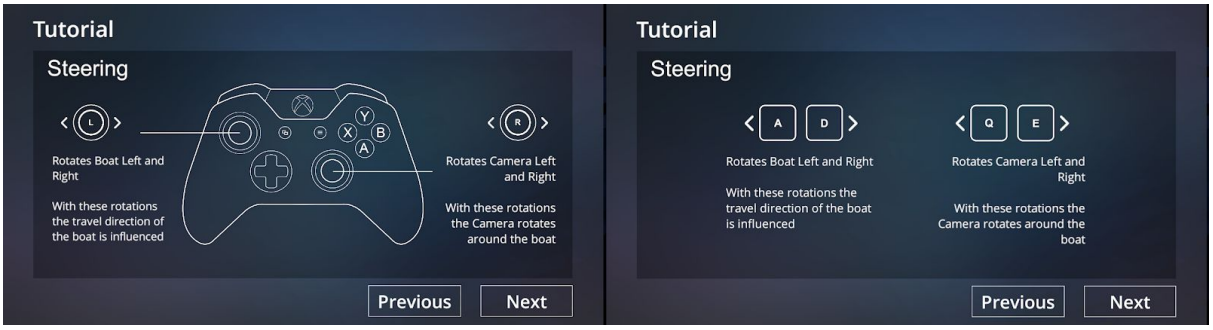
Basic sound effects like cannon firing sounds were already implemented. To increase the immersion, more sound effects are added into the game like ship sailing across the water, crashing wood when a cannon hits a ship and when a player collects one of the crates. Additionally, an early version of the game theme is also composed, which plays throughout the gameplay.

4.6. UI

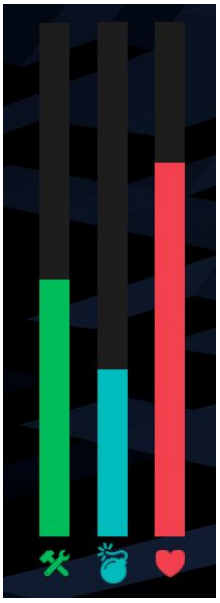
The UI is one of the more important features we added, as with this the navigation around our game and retrieving important informations from the game are achieved. We added a simple Lobby screen for the beginning of the game, allowing the players to wait until everyone joined their game. Also a tutorial screen was added, to explain new players the basic concepts of our game. Lastly we added a player HUD screen for the most important informations concerning the boats.

4.6.1. Tutorial

The tutorial screen automatically adjusts to the used input format (mouse and keyboard or controller) and shows the respective key mappings. This can also switch mid tutorial, if the user decided to use a different input format. The screens show the mapping of the ship movement, the crew assignment and the cannon usage. For the crew assignment, we explain the concept of the assignment and how one can reassign a crew member from different stations. The cannon screen focuses on the concept of the cannons, explaining how they work.



The above screenshot shows the different tutorial screens for either controller users (left) or keyboard and mouse users (right)



4.6.2. Player HUD

The HUD shows the crew assignment and the health, ammunition and repair material bar. The three bars depicted show the current status of the players health (red), ammunition (turquoise) and repair material (green) and are displayed at the right side of the screen. They are also accompanied by small symbols.

The crew assignment HUD is located on the lower left corner and shows how many crew members are at each station. Each station is placed at the same position they can be accessed via the D-Pad on the Controller and are represented by small symbols (Cannon for cannon station, band aid for repair station, speedometer for sail station and chest for fishing station). If a crew member from a station will be reassigned, the corresponding symbol will turn orange, to indicate the player from which station a crew member will be taken for the assignment on the new station.

We also added a HUD over the other player's boat, to show how much health they have and what their steam name is, so the player can distinguish between the players. Also, when someone takes damage, it will be shown on every player's screen as a floating text. This text contains information on how much damage was received and who hit whom.

5. Playtesting

For our playtesting phase, we released a development build on steam and distributed keys to willing participants. These were mostly friends from team members and are all more or less into gaming. With the steam release, we had a very simple method for distributing our game to our participants, without having to meet with them in person.

5.1 Testing Process

For playtesting we mostly recruited friends from our group members and let them test the game on their own hardware over steam. As we have a multiplayer game, we also had to play against our friends during playtesting. One session contained between two to four people (of which one was a member of our developing group).
We briefly told them what we expected from them (testing the game, talking about what they were experiencing or trying to achieve during gameplay and afterwards giving feedback) and then let them start the game. The first screens they saw, were the tutorial screens, giving them a brief overview of the controls and gameplay concepts. Afterwards they were let loose on our maelstrom and battled each other for the victory. During two to three playing sessions (depending how fast all other participants were sunk), our playtesting participants had to test the crew assignment, boat controls, collecting crates and cannon shooting (as all of these are basic gameplay elements).
Afterwards, we gave our participants a questionnaire (created with google forms) asking them questions about different aspects of our game. These were categorized into “General Questions”, “Formal Elements”, “Procedures, Rules, Interface and Controls”, “Specific Questions on Gameplay” and “End of Session”. Here we asked the participants all kinds of questions, which would ensure us helpful feedback on our game “Fatal Tides” and help us improve our gameplay experience.

5.2 Questions

These were the questions we asked:

General Questions

- What was the first impression of the game?
- Did that impression change as you played?
- If yes, please describe how the experience changed for you.
- Was there anything you found frustrating?
- If yes, which things triggered a frustrating experience for you?
- Did the game drag at any point?
- If yes, please describe the parts where you experienced drag.
- How long did the game feel?
- What was the most exciting thing about the game?
- Were there particular aspects that you found satisfying?
- If yes, which parts were satisfying for you?

Formal Elements

- Describe the objective of the game.
- Was the objective clear at all times?
- If no, when and where did the objectives feel unclear?
- What was your strategy for winning?
- Did you find any loopholes in the system?
- If yes, what were they?
- What elements do you think could be improved?
- What elements did you like about the game?

Procedures, Rules, Interface and Controls

- What Input device did you use?
- How easy were the procedures and rules to understand?
- How did the controls feel?
- Did the controls make sense?
- If no, which parts of the controls did not make any sense to you?
- Did anything feel clunky or awkward?
- If yes, what felt clunky or awkward to you?
- How helpful did you find the tutorial?
- How easy did you retrieve information from the HUD (head up display) during gameplay?
- Was there anything about the interface you would change?
- If yes, what parts would you change?
- Are there any controls or interface features you would like to see added?
- If yes, what would these be?

Specific Question on Gameplay

- How intuitive did the controls for the boat feel?
- How intuitive did the controls for the cannons feel?
- How easy did you find the assignment of crew members?
- How easy did you find hitting other players?
- What would you think would be the ideal numbers of player in the maelstrom battling each other?
- How easy did you find to collect crates?
- How did you find the difficulty of the game?
- How stable was your network experience?
- Any additional feedback for the gameplay you would like to give us:

End of Session

- Overall, how would you describe this game's appeal?
- Would you purchase this game?
- If no, why would you not buy this game?
- What elements of the game attracted you?
- What was missing from the game?

- If you could change just one thing, what would it be?
- Who do you think is the target audience of the game?
- If you were to give this game as a gift, who would you give it to?
- Any additional feedback you would like to give us (can be anything you would like to let us know):

5.3 Testing Results

We had a total of eight participants playtesting our game and got very helpful feedback for our game and gameplay experience.

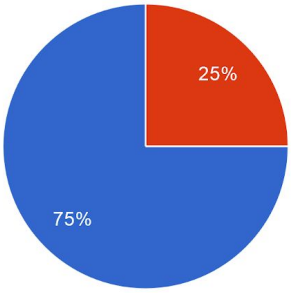
5.3.1 Summary

Overall, the concept and theme of our game were received very well and were the most exciting and satisfying variables of the experience. However, all our participants had a high frustration rate with the current condition of our game (as presented during playtesting). All participants agreed that more players add to the fun and that four is the best optimum. On average our playtesters found the game too long, the difficulty at about the right level and they found “Fatal Tides” neither appealing or unappealing.
The tutorial and interface was to most very helpful and intuitive. However, the goal of the game was not always clear to all participants and four people mentioned improvements for the HUD (Head up Display).
On our controls in the game we got the most feedback from our testers. Here the majority thought that the overall controls made sense, but had a feeling that aiming with the cannons and controlling the ship and the feel of movement still needed work. The crew assignment on the other hand was for the majority already very good. We also had testers, who tested both controller and keyboard, and we received better feedback on the feel with the controller than the keyboard. The overall opinion on the camera controls was that more fine tuning with potential addition of vertical movement and zoom is still needed. On average, playtesters found the difficulty of hitting other players way to hard and als collecting the crates proved to be harder for most players.
On networking, we got the result that it still has problems and was causing the most frustration with players. Many testers experienced unstable connections causing other players to lag over the world and increasing the difficulty of hitting enemies dramatically. This has also shown itself when a tester was geographically far away from the host.
Most players from our playtesting group found the visuals of the boat, tentacle, crates and crew members already appealing. However, many wanted better visuals for the water.
On the audio aspect, we also mainly got positive feedback, although many thought that it was too loud. One participant also wanted a more action loaded soundtrack for the fighting in the maelstrom.
Our users found the target audience for our game very broad and think that everybody could play and like “Fatal Tides”. Most tester would gift this game to friends and other close people, so they could play it together with them. All of them thought that our game definitely needs more polishing and was received as unfinished but great potential. Half of the game testers would buy our game if it spend a few more months in development.

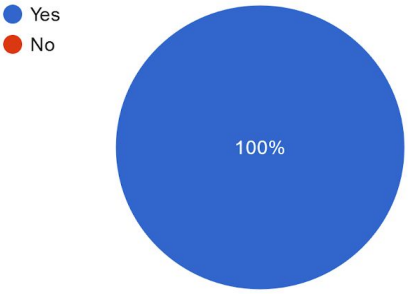
5.3.2 Graphs

Was the objective clear at all times?

8 Antworten

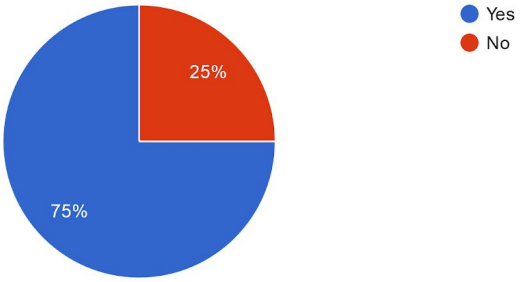


anything you found frustrating?



Were there particular aspects that you found satisfying?

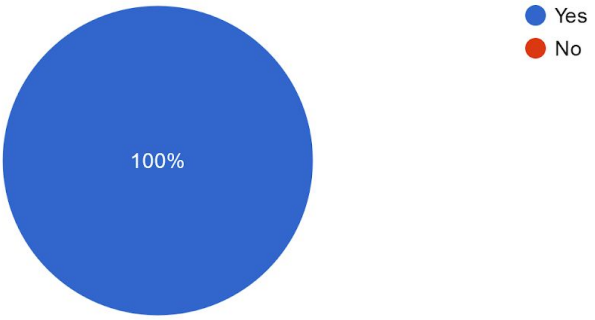
8 Antworten

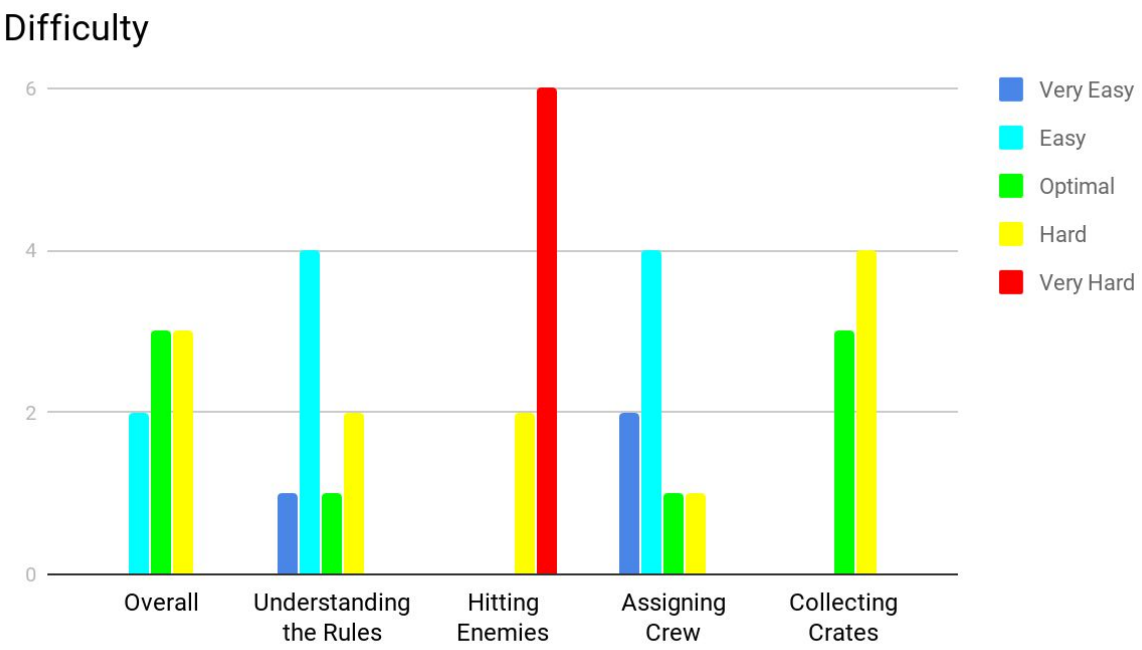


Some Answers to our "General Questions

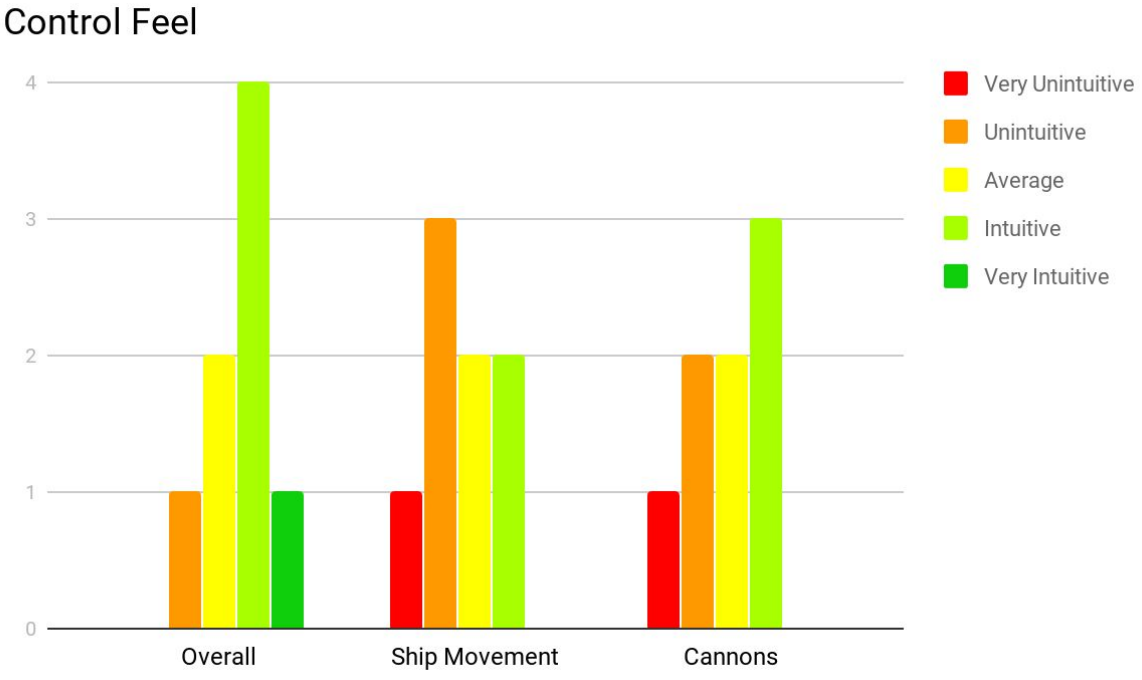
Did anything feel clunky or awkward?

8 Antworten





The perception of the difficulty



Visible discrepancy between overall control "logic" and actual feel of the systems

5.4 Evaluation

After doing the playtests and collecting the results, we concluded that the overall dynamics and core gameplay elements were implemented very well. The overwhelming majority of the testers could clearly identify the goals and the mechanics of the game. Since this was an alpha release, investigating how well the core elements works was the first priority and it can be said that the idea and development of the game was very well received.
On the other hand, feedback also made it clear that there are some major problems in the game that needs polishing and reworking, like networking or camera flexibility.
Looking back on the feedback, testers have shown us a fresh look into our game and pointed out some features that we overlooked that needed polishing. Most of the criticism was constructive and were on subjects that we can improve on or we are planning to improve on.
The changes that will be made based on playtesting are as follows: To flesh out the game we will add start and end screens and a lobby. Camera controls for both movement and cannon mode will be made more flexible and more user friendly. Cannon controls will also be refined. The gameplay variables like ship health, cannon damage, speed etc. will be evaluated and tested again and again intensively. We will focus on optimizing our game in terms of networking to reduce lag on each participant. Adding more visual effects and feedback (animations, particles, interface) and improving water shading are next on the list. We also plan to add power-ups and more quality-of-life improvements such as better sound effects and an UI that fits more into the game theme. The rest of the time will consist of polishing the game and getting it ready for the final release.

6. Public Presentation and Conclusion