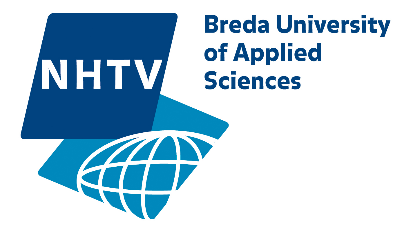
****

Final Product Description

Team Name : (As in perforce depot name)

Team Members:

Student Name: Arthur Kuylaars Student Number: 170108

Student Name: Erico Wiggers Student Number: 151850

Student Name: Joey Jacobs Student Number: 171427

Important Links. (if any) [Swarm links only]

Guidelines document: https://swarm1.nhtv.nl/l/buup59

Description of Game.

The final product should be a playable game which demonstrates key game features listed in this outline.

We are going to make a real time strategy game where the player commands a fleet of ships. We’re going to expand on existing game concepts from Battleships. It is a top down 3D peer-to-peer multiplayer game where two players can match up against each other. The goal is to sink the enemy fleet. Controlling the fleet will be in real time, instead of turn-based. There are multiple ship types with different advantages and disadvantages.



Framework Features still to complete

* Sprites
* Smoke particles
* Water particles
* Pixel particles
* Water shader
* Raknet
* A\* star
* Heap structure

Outline distinct enhancements to the framework

Joey

* 2D UI graphics system
* Particle system
* Water

**Arthur Kuylaars 170108**

**Raknet networking system**

Implement a peer to peer network system that can run on the Wi-Fi connection of the Raspberry pi. The system needs to handle serialization of data that can be send via the network, also a convention needs to be put in place in what format data should be send.

There are only 2 players during a game session. Each player sends its own data to the other and they both sync their game state at a set interval.

**Ship behaviour**

Our game uses battleships as units that can be controlled, these ships need to move in a realistic manner which means they can have varying speeds and turning speeds. There needs to be some ship behaviour in place to make sure the ships move realistically when tasking it to move into a position.

**State Manager (Menu, Gameplay)**

The game needs a state manager which handles at which state the game runs, the game needs to be able to assign a winning player and a losing player. Also, there must be the ability to disconnect from any sessions which results in the enemy player winning. The game must handle various states and must make sure the other peer it’s connected to is kept up to date about this.

**Erico Wiggers 151850**

**Fog of war**

Implementing fog of war is not the fog of war which requires a shader but rather a fog of war which is calculated by ray casting. This will simply make the ship have two contact points. One on the front and one on the back. Those are used for ray casting between other ships to calculate if the enemy ships should be rendered or not.

A link who uses this system is from world of tanks which is called a spotting mechanic or visible system within their game. A link to the video explaining the spotting mechanic can be found here, skip to **3.18** in the video: [**https://www.youtube.com/watch?v=RMF82fz4kDk**](https://www.youtube.com/watch?v=RMF82fz4kDk)

**Path finding**

Within our game it should also be possible to move around multiple or individual ships. For this we are going to implement a pathfinding system using the A\* star algorithm. There can be done quit a bit of optimization with it but we will start with the basics first. <https://www.youtube.com/watch?v=-L-WgKMFuhE>

**Unit control system**

Our game requires also a unit control system which let’s the player select and move around their fleet. The system should detect the amount of ships selected and calculate with use of the pathfinding for the right path to move along. [**https://www.youtube.com/watch?v=XXOEWsxLnAo**](https://www.youtube.com/watch?v=XXOEWsxLnAo)

**Joey Jacobs 171427**  
  
**Water System**  
  
Since the water plays an important part of what we’re going to make, I wanted to make some fine looking water. I want to have a system that is capable of rendering water *with* wave animations. This should be done using the GPU. Depending on the capabilities of the Pi, I want to give it some specular lighting as well. Things I might have to consider are realism and performance issues.

**GPU Particle System**In block A I made a particle system that ran on the CPU. But this should also be possible on the GPU. The reason I’m going for the GPU is because a lot of particles can be rendered in parallel, thus it should provide better performance on the Pi. The types of particles I want to support are; water splashes, smoke and single pixel particle effects.

**(Motion) Blur**I want to make some simple motion blur / blur effects. Motion blur when moving the camera, and normal blur that occurs as an effect when something impacts the ship. Right now I have no clue how it works, but I’m thinking of using the framebuffer for this.

Project Planning / Deliverables:

**We make use of Trello as a leading planning and schedule. We have put in some labels which indicates what the priority of the task is. Those can be a must have, should have and could have ranging from highest priority on the left to lowest on the right. This means a must have label indicates that a task has a high or crucial priority. The other label indicates in which week we want to have features implemented.**

**https://trello.com/b/S8PCC7Uv/battleships-block-c**

Assistance you might require

* Water Shader
* Multithreading pathfinding(not a must)
* Networking using Raknet