Project Proposal: Boat and Water Simulation

Author: Christopher Noland
Email: ccnoland1@ualr.edu
Professor: Dr. Basu

---

1. Objective:

The goal of this project is to create a Unity simulation that features a dynamic water mesh and enables players to interact in a multiplayer environment. The simulation will allow players to join a server-client relationship and compete to collect points by capturing water buoys within a time limit. Key features of the project will include:

- Water mesh manipulation for realistic water physics and effects.
- Multiplayer support using Unity's **Netcode for GameObjects**.
- Performance optimization via **BurstCompile** for handling heavy calculations efficiently.

By completing this project, I aim to develop a deeper understanding of game networking, water physics, and performance optimization in Unity.

---

2. Rationale:

The motivation for selecting this project is twofold:

1. **Nostalgia and Inspiration:** Many of my favorite childhood video games involved ships, boats, or water-based environments where the dynamics of water played a key role in the gameplay. This nostalgic connection to water-based games inspired me to create a simulation that emphasizes water physics and interaction.
2. **Challenge and Learning Opportunity:** The complexity of combining networked multiplayer gameplay with water simulation provides a unique challenge. The project is also an excellent opportunity to enhance my skills in Unity, particularly in areas where I have limited experience (e.g., **Netcode for GameObjects** and **BurstCompile**). This project strikes a balance between achievable goals and technical challenges that will allow me to grow as a developer.

---

3. Approach:

The development of the project will proceed incrementally, with each component building on top of the previous one. The plan is to:

1. **Establish Project Baseline:** Create the initial game environment, basic water mesh, and movement mechanics for the player-controlled boat.
2. **Add Networking with Netcode for GameObjects:** Implement multiplayer functionality, allowing players to connect and interact within the game world.
3. **Add Buoys and Point System:** Introduce water buoys in the game world that players can capture to earn points.
4. **Add Game UI and Timer:** Implement a user interface (UI) to display the score and countdown timer for the game's time limit.
5. **Implement Point Tracking Across Clients:** Ensure that the points players collect are synchronized across all clients in the multiplayer session.
6. **Polish and Prepare for Presentation:** Finalize the game mechanics, optimize performance, fix bugs, and prepare the project for presentation.

Throughout the development, I will rely on Unity documentation, video tutorials, and online forums to assist with technical challenges, particularly in scripting and networking, as my experience with C# is still developing.

---

4. Timeline:

The following is a proposed timeline for the completion of the project:

Task	Start Date	End Date
Create Project Baseline	September 30th	October 9th
Add Network for GameObjects	October 10th	October 19th
Add Buoys and Point System	October 20th	October 28th
Add Game UI and Timer	October 29th	November 5th
Add Point Tracking Across Clients	November 6th	November 13th
Polish and Prepare for Presentation	November 14th	December 1st

---

5. Possible Issues:

There are several potential challenges that may arise during the development of this project:

1. **Networking Issues (Netcode Implementation):** Since I have no prior experience with Unity's **Netcode for GameObjects**, synchronizing player positions, buoy states, and

score data across clients could be difficult. Ensuring smooth communication between the server and client may require careful attention to detail and extensive testing.
2. **Water Mesh and Physics Integration:** Achieving realistic water behavior and physics may prove to be a challenge. Water simulation requires managing wave dynamics, buoyancy, and interactions with the player's boat, which could introduce performance bottlenecks.
3. **BurstCompile Integration:** The **BurstCompile** package is designed to optimize performance by compiling C# code to highly efficient native code. However, as this is an area I have little experience with, I may encounter challenges in integrating it effectively into the project. Specifically, ensuring that BurstCompile optimizes the necessary computations without causing compatibility issues could require in-depth research.
4. **Multiplayer Synchronization:** Keeping track of scores, player positions, and the state of the game (buoy collection, timer) across different clients in a seamless way could present synchronization issues, especially under network latency or when dealing with multiple players.

---

6. Future Enhancements (To be considered after project completion):

Once the core functionality has been implemented, additional features could be added to enhance the project:

- **Advanced Water Physics:** Implementing more complex water effects like waves, tides, and varying water textures based on the environment.
- **AI Opponents:** Adding AI-controlled boats to increase the difficulty of the game when fewer players are present.
- **Enhanced Multiplayer Features:** Introducing team-based gameplay, chat functionality, and leaderboards.
- **Cross-Platform Support:** Ensuring that players on different platforms (e.g., PC, console) can play together.

---

7. Code/Development:

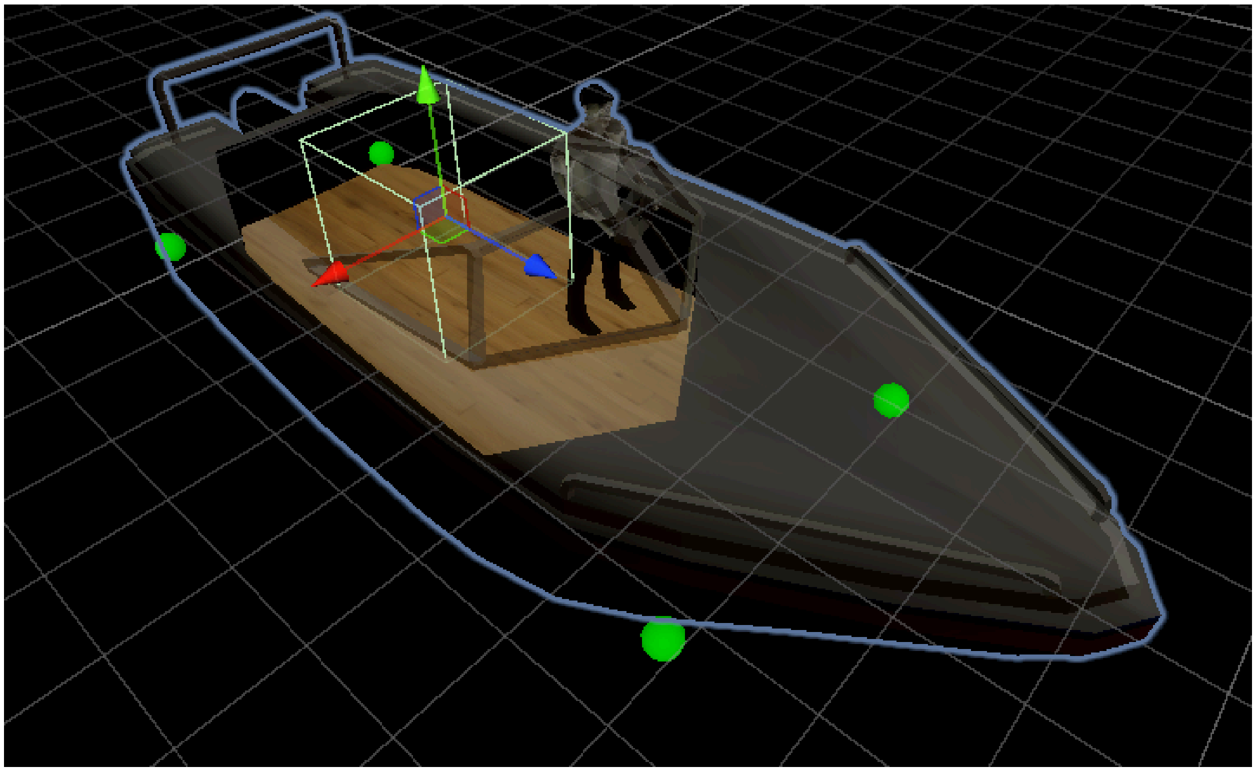
This is some pieces of the code that

7.1 Mesh Generator:

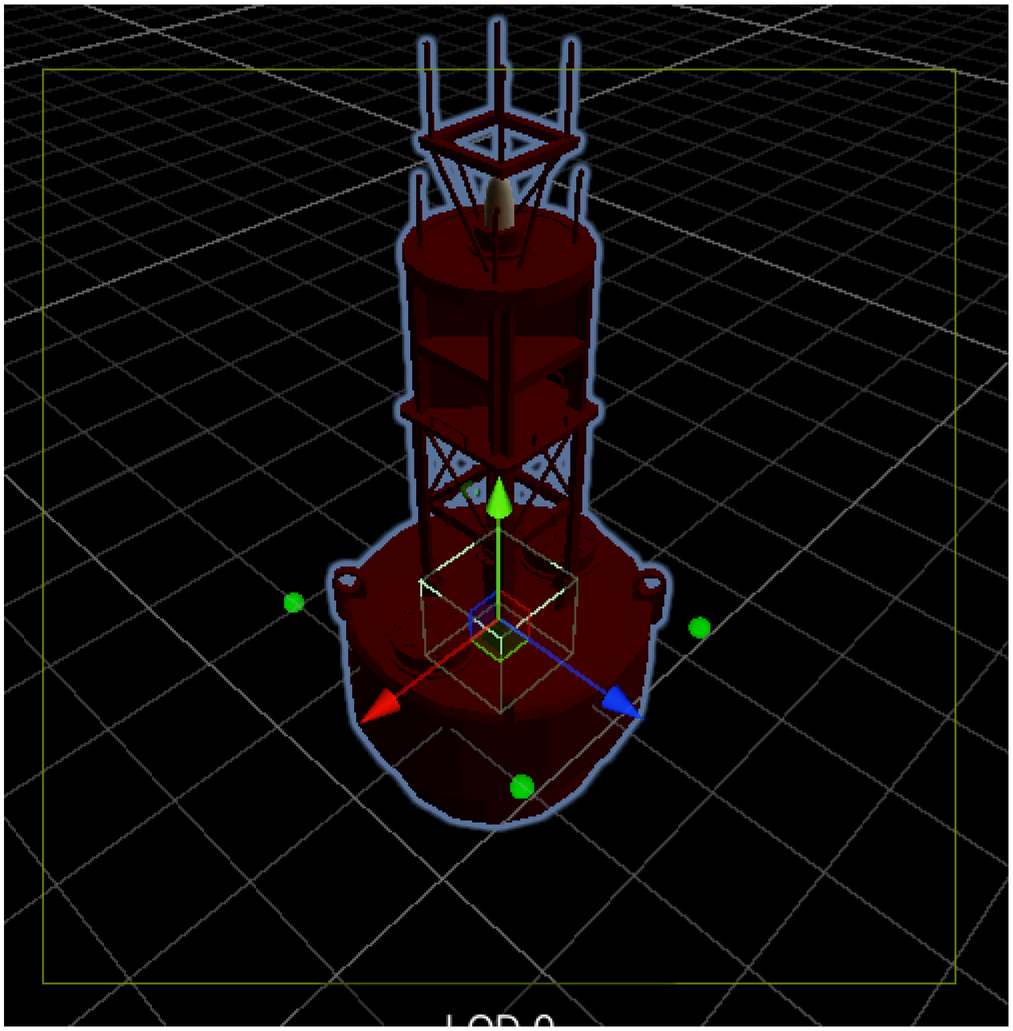
```
var tries = new int[Mesh.vertices.Length * 6];

for (int x = 0; x < Dimensions; x++)
{
    for (int z = 0; z < Dimensions; z++)
    {
        tries[index(x, z) * 6 + 0] = index(x, z);
        tries[index(x, z) * 6 + 1] = index(x + 1, z + 1);
        tries[index(x, z) * 6 + 2] = index(x + 1, z);
        tries[index(x, z) * 6 + 3] = index(x, z);
        tries[index(x, z) * 6 + 4] = index(x, z + 1);
        tries[index(x, z) * 6 + 5] = index(x + 1, z + 1);
    }
}
```

This shows a function for generating the vertices for the wave mesh which will be the baseplate of the project. This is used in conjunction with a function to generate the triangles based off of the vertices which will give the mesh its signature wave pattern when combined with Sin functions.



In order to create the buoyancy aspect of the player I had to add different float points, which are shown in the above image by the green spheres, which are used in a script by first getting the location of those points and comparing it to the height of the wave function. If the height of the float points are lower than the waves, it is brought up and if it is higher than gravity brings it down.



Another development snippet is the buoy that I implemented to use as a scores system so that whenever players ran into it then their score would be increased, then the buoy would despawn and respawn at another position. The flotation works similarly to how to boat works.

---

8. Conclusion:

The Boat and Water Simulation project is an exciting challenge that combines multiplayer functionality, water physics, and performance optimization. By completing this project, I aim to gain a deeper understanding of networking in Unity, as well as improving my skills in game development and programming. With careful planning and iterative development, I expect to create a fun and functional simulation that can be used as a foundation for future game projects.
