Ocean wave simulator in Unity

Student Name: Hussen
Email:hmah@kth.se

1. Introduction

The project chosen to be presented is called Simple Ocean Wave Simulation Using Unity and C#. The main objective of this is to simulate movements of ocean waves using sinusoidal functions [1]. This method will help replicate the dynamics of the water surface that can be modified by adjusting the parameters like wave height, frequency, and speed. This method aids in producing a dynamic water surface that can be controlled. The point of the simulation is to focus on visual impact and its simplicity. The simulation is a great way to learn about how to model mesh deformation, real-world data animated on a computer screen fluid nature of water.

Simulating water is very complex because we have to take into account water's fluidity, reflecting and refractive qualities, and interaction with light, making it a challenging natural phenomenon to copy from the real world. Professional simulations often use expensive computational techniques such a custom shaders or fluid dynamics. This project will simplify the task using mathematical functions and real-time manipulation. The project aims to show that even just a basic implementation can produce a visually convincing and interesting effect. Aesthetics is important in graphics as perceived realism often has more impact than true physical accuracy. With emphasis on interactive parameters, users can now change the behavior of waves and see in real time how that affects in real time.

2. Background

Recreating water has been a long challenge in the field of computer graphics. From a simple 2D wave to fully simulated 3D surface dynamics. For interactive applications, such as games, developers often use approximations that are good enough rather than completely accurate physics. This project is using procedural animation to simulate waves on the plane. The method is based on sinusoidal displacement. Every direction of the vertex of the mesh is changed in its y-axis direction by a sine wave. The result is waves moving across the surface of water. This function is time-dependent, which means it produces continuous and smooth movement that imitates oceanic waves. Such animation is lightweight so that it will run well even on low-end devices well. The mathematical formula involves the use of the sine function with phase shifts based on position and time[1].

```
 vertex.y = Mathf.Sin(Time.time * waveSpeed + vertex.x * waveFrequence + vertex.z *
waveFrequence) * waveHeights;;
```

This will make sure that waves will travel diagonally across the surface and be controlled through exposed public parameters in the Unity Spectator. Wave behavior can then be customized to produce choppy, calm, or fast-moving surfaces. This figure shows a simple ocean surface moving like a wave

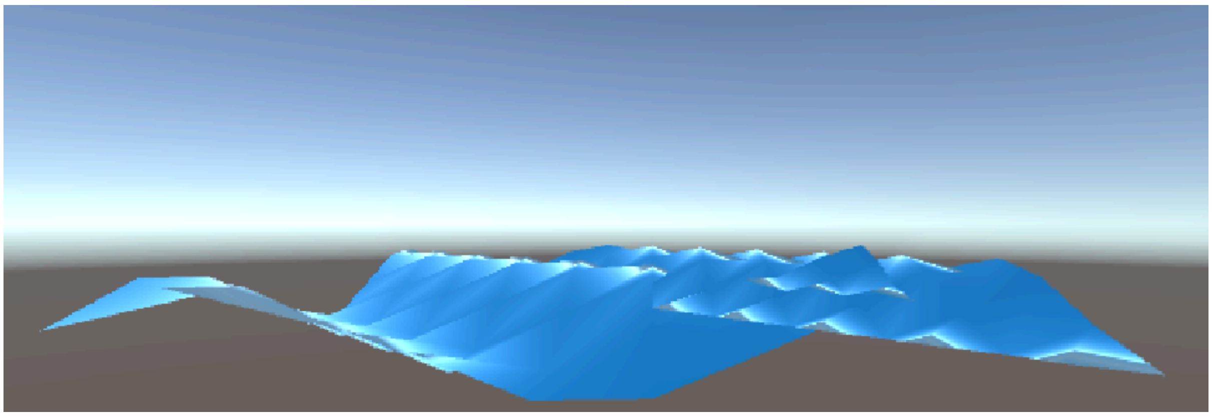
Figure 1: The waves of water.

3. Implementation

Tools used

1. Unity
2. C# scripting

The plane is created as an ocean surface, which is applied with custom material with transparency and reflectivity[2]. Add ambient lightning for realism. Materials were used to create through Unity shaders, changed the rendering mode was changed to transparent[3]. The smoothness and metallic values have been increased to almost max. Additionally, we have applied a normal map that adds detailed surface waves.

4. Result

The goal of this project was achieved, creating a controllable, animated water surface. The result is satisfying and appealing and can be used in simple games or simulations. The variable enabled real-time alterations to wave speed, height, and frequency, which created a procedural animation principle and provided feedback. By combining a mixture of maps, normal maps, and skybox into the batch, the water took on a more natural appearance. A smooth and continuous surface. The waves may be made by a simple math formula, but they are convincingly displayed in a natural motion. The high-resolution mesh also enhanced the curvature of waves. Different wave height and speed settings were tested for evaluation. Slow, ocean surface where generated by a lower setting, while stormy conditions were replicated by a high-frequency setting. Achieving common-sense level patterns requires attention to lightning and the color balance of each pixel.

5. Reflection

This project provided me with valuable insight on how to use Unity's mesh manipulation system and real-time images. Have learned how vertex resolution affects visual fidelity, meaning how lighting materials and setting up the surroundings affect how realistic 3d scenes look. One thing that I learned is how crucial it is to balance speed and visual detail when optimizing mesh density. Additionally learned that simple math, like sine wave, can be used to facilitate complex visual simulations. In the future, the work could continue with new implementation techniques that could be like adding water caustics, dynamic lightning, GPU shaders, and the ability to create interactive objects that could interact with water, for example space or a boat. Overall, the project was both stimulating and effective in demonstrating basic graphic design ideas.

6. User study Idea

To evaluate the realism and functionality of this simulation, a user study can be implemented in which participants observe various wave configurations and provide feedback on how realistic they are. For example.

Case 1: Ocean is calm(low frequency,low height)

Case 2: Ocean level is moderate(matched setting)

Case 3: Stormy sea (fast and frequent waves).

Participants who took part in it would respond to questions such as:

- “Which scenario appears most realistic?”
- “Which option would you prefer for use in a video game or film?”

This way could help tune and refine parameters and emphasize visual elements that users perceive as credible. This kind of evaluation is of how people see things as important.

6. Reference

1.Jensen,H.W & Christensen, P.H.(1998). *Efficient simulation of light transport in a scene with participating media using photon maps.*

2. Unity documentation:https://docs.unity3d.com.

3. Unity Manual: Mesh and Materials

4. Unity Asset Store (Skyboxes and materials).