AMITY UNIVERSITY

UTTAR PRADESH

Amity School of Engineering and Technology

NTCC Internship (ETPT100)

Final Report

14th May 2024 – 1st July 2024

Project Title: Sine Wave Variants For Dynamic Ocean Simulation

Project Area: Graphic Programming

Name: **Mr. Pratyush Soni**

Enrollment No.: **A41105222120**

Semester: 5

Branch: B-Tech CSE (2022-26)

Faculty Guide's Name: **Prof. Garima Panwar** [Assistant Professor]

DECLARATION

I, Pratyush Soni thus affirm that my effort and research went into the "Sine Wave Variants For Dynamic Ocean Simulation" project report. All of the facts, figures, and conclusions in this study are real and have come from reliable sources. All references, quotations, and citations have been properly cited and acknowledged in this study.

The truthfulness and integrity of the information given in this project report are my responsibility. I am aware that any misrepresentation or plagiarism may have serious repercussions and might cause this report to be rejected.

I hereby give **"Amity University, Greater-Noida"** permission to utilize this project report for academic and archival purposes while maintaining ownership of the original work and intellectual property contained therein.

Name and Signature of Student:

Pratyush Soni
Enrolment No.: A41105222120
B.Tech CSE (2022-26)

3rd July 2024

CERTIFICATE

I hereby certify that the work is being presented in the In-house practical training entitled, **“Sine Wave Variants For Dynamic Ocean Simulation”**, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology submitted in Computer Sector and Engineering of Amity University Uttar Pradesh, Greater Noida, is an authentic record of my work carried out under the supervision of **Prof. Garima Panwar.**

Pratyush Soni
B. Tech CSE 5TH SEM
A41105222120

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Prof. Garima Panwar

Amity School of Engineering and
Technology Amity University
Uttar Pradesh,
GREATER NOIDA

ACKNOWLEDGMENT

For the accomplishment of this project, I would like to take this opportunity to express my gratitude to the people who have been a part of this project right from its inception and helped and supported me. It is to them I owe my deepest gratitude.

It gives me immense pleasure to present this project report on "**Sine Wave Variants For Dynamic Ocean Simulation**". I hereby take this opportunity to add a note of thanks to **Prof. Garima Panwar ,** under whom, I have done this project, and also **Prof. (Dr.) Deepshika Bhargava** (HOI), Department of CSE, AUGN for her constant encouragement and guidance.

Sincerely,

Pratyush Soni
Enrolment No.: A41105222120
Computer Science Department, (ASET)

CONTENTS

ABSTRACT

The project “Sine Wave Variants For Dynamic Ocean Simulation” focuses on the dynamic simulation of the ocean using Unity Engine and its rendering pipeline. The main objective of the project was to simulate ocean waves by leveraging the GPU’s parallel computing capabilities by using the custom surface shaders in the Unity Engine. During the course of these 49 days I not only learned the effect of different types of sine waves on ocean’s behavior but also learned a lot about how a rendering pipeline works.

I used Unity Engine to inject our custom shaders in the rendering pipeline to get the results that I desired. The project explored how manipulating the sine wave adding arguments like – time, angle, steepness, amplitude and etc,. adds to the dynamics of the wave. Not only have I made a dynamic ocean but I also worked on light interaction with the normals of the plane according to the nature of the waves, I updated light diffusion and how the camera perceives the light in real-time.

By referring to GPU GEMS I got inspired by its implementation and also I followed instructions laid out by Jasper Flick for using Unity Engine’s Graphic pipeline. Due to their help I successfully completed my project, learned about the GPU efficiency techniques and also how to tinker with the Graphical pipeline of Unity.

The tech stack that I used was – Unity Engine, C# and HLSL. The Unity Engine was used for its Graphical pipeline, C# and HLSL were used for scripting and writing shader codes.

INTRODUCTION

Graphic Programming is a field of computer science where art and engineering work hand in hand, creating things that are not practically possible without shader programming. Have you ever wondered how vast worlds in simulations are rendered without taking a toll on performance, or how huge water bodies are depicted without overheating the entire system?

The answer is the GPU’s great capability for parallel computing, executing several operations in parallel without affecting other processes in the system. This type of implementation and GPU usage is not limited to graphics; the same hardware and methods are used in the AI and ML industries. Complex equations are broken down to run on multiple GPU cores, increasing the model's performance and delivering output more efficiently. This is why GPUs are essential in the AI and ML industry, and why NVIDIA is becoming one of the most valuable companies in the world as they are the leading producers of modern GPUs.

In my project, I mainly explored the architecture of GPUs and how to access it. I investigated how it drastically improves the performance of a 3D or 2D graphic rendering program. I learned about how these programs treat lighting and how different it is from the real world. Further in the document, I will discuss the things I did in more detail, but it’s important to know the benefits of the project. Through the implementation of this project, I learned about how sine waves have different variants and, because of their harmonic nature, are used in many situations such as:

- Day-Night cycles
- Cycling platforms in puzzle-platform games
- Mechanical simulations

My project was simulating ocean waves, and for that, I used multiple sine wave variants. To add realism, I used Gerstner’s Wave, which also has a cosine component.

1.1. Objective

The primary objective of this project is to iterate through different sine wave functions and see how the ocean wave reacts.

1.2. Scope

This is a fairly important part of the movie simulation and game industries and understanding how the fluid should react and putting into a mathematical function that a GPU can compute in real-time is a handy skill to have in the era of GPUs as it is being used everywhere in the industry be it AI and ML or Graphic intensive industries like Fashion, Movie and Games.

This type of ocean implementation will aid the real world simulation to be more efficient and run better on all the machines.

METHODOLOGY

2.1. Technology and Concept Used

The project uses Unity a popular game development platform, specifically the following technology is used:

- ➢ **Unity 3D**

  Unity 3D is a powerful cross-platform game engine used for developing video games and simulations for computers, consoles, mobile devices and websites.

- ➢ **Shader Programming**

  Shaders are scripts written in GLSL or HLSL so that they can run on GPU. It is used to control the rendering of graphics.

  1) **Surface Shaders:** This shader is responsible for how the surface is going to behave.
  2) **Vertex Shaders:** This shader is responsible for how the vertex is going to behave in a certain object

- ➢ **Mathematical Concept for the Wave Simulation**

  1) Sine Wave

  2) Gerstner Wave

- ➢ **Unity Material System**

2.2. Requirement Analysis

Functional Requirements

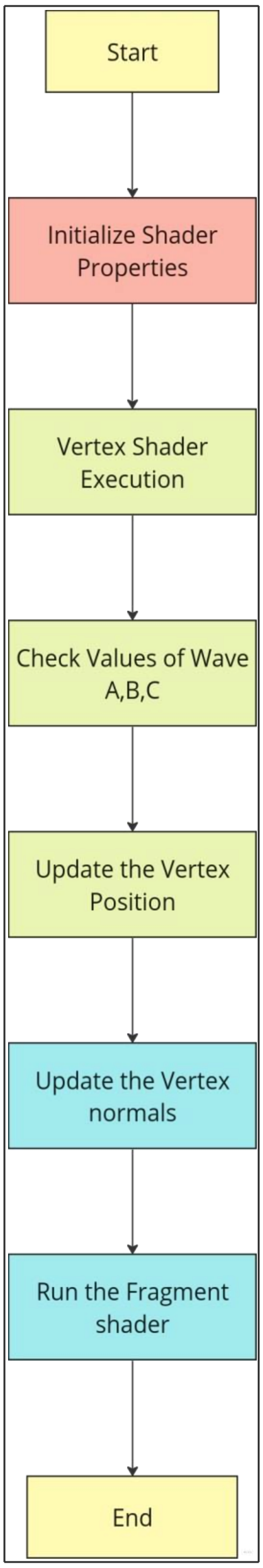
1. **Wave Simulation**
   - Implement mathematical wave patterns (e.g., sine waves, Gerstner waves).
   - Enable directional wave movement.
2. **Shader Development**
   - Create custom surface and vertex shaders for dynamic wave rendering.
   - Real-time adjustments of wave properties.
3. **User Interaction**

- Modify wave parameters (amplitude, wavelength, speed, steepness, direction) through UI.
- Real-time preview in Unity Editor.

4. **Material Management**
    - Develop and manage materials using custom shaders.
    - Apply materials to meshes.
5. **Performance Optimization**
    - Ensure efficient performance with minimal frame rate impact.
    - Optimize shader code for GPU utilization.

OVERVIEW OF THE PROGRAM

3.1. Final Shader's Flow Chart


Start
Initialize Shader Properties
Vertex Shader Execution
Check Values of Wave A,B,C
Update the Vertex Position
Update the Vertex normals
Run the Fragment shader
End

3.2. Wave Using Simple Sine wave

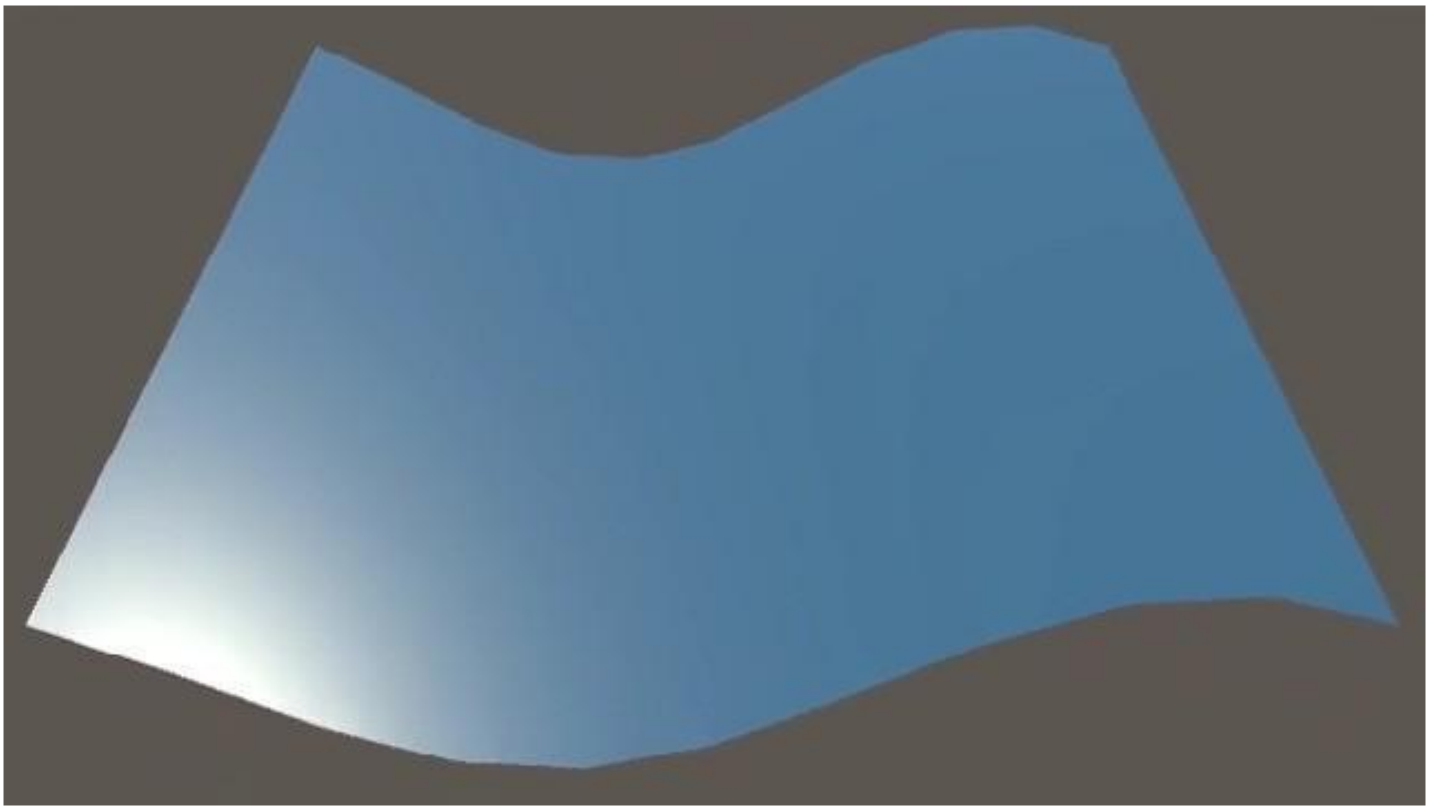
For the first iteration we used the simple sine wave to replicate the ocean wave.
The function was :

$$Py = \sin(kx - kct) = a * \sin(k(x - ct))$$

Where,

Py is all the y component in the vertices of the plane we generated to make the ocean.
k is the wave-number.
c is the speed at which waves will move.
x is the x component of the plane.
a is the amplitude of the wave.

When we use this function we get the following result.



But we can observe the light is not interacting well enough with the shader because the plane itself is not deforming, it's just the material's surface which is not physically present.
Therefore the surface normals are still and do not change hence the lighting doesn't change as well.

To overcome that we will also update the normal of the plane according to the function. For that we need to find the tangent at every point of the plane according to the updated y component position and then find the normal value of it.

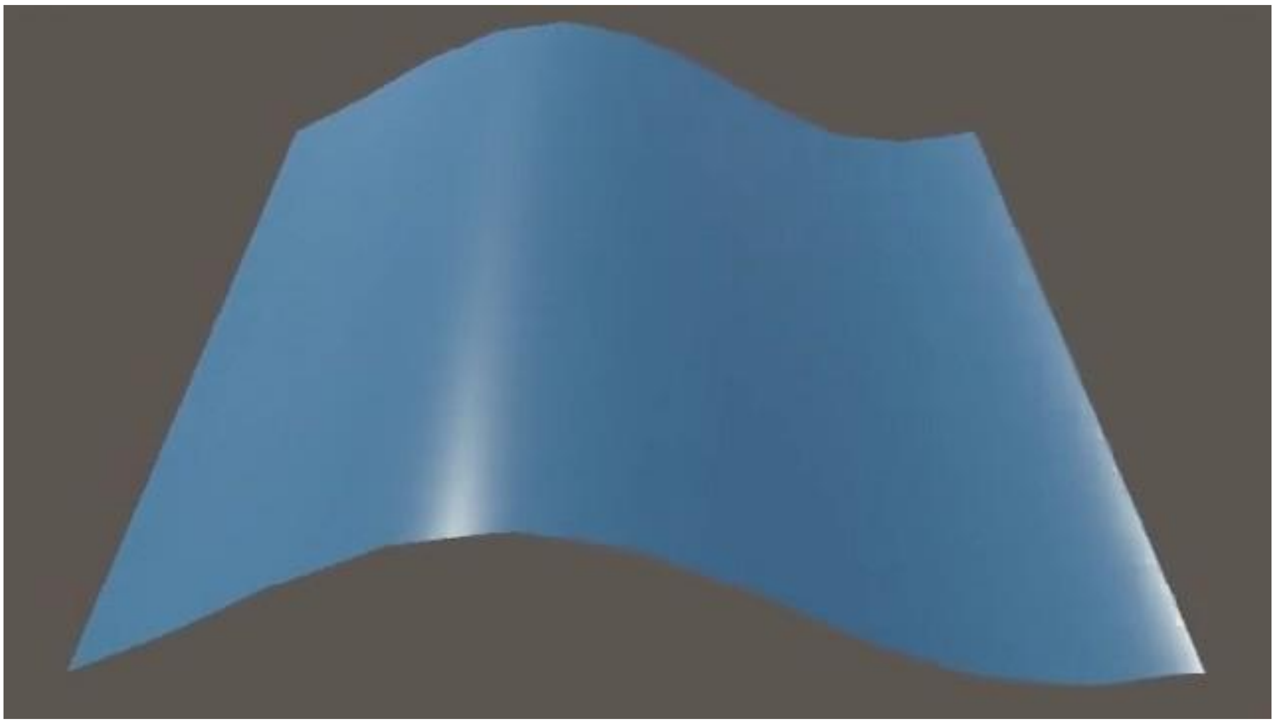
So the function to that will be:

$$T=[1, k*a\cos(f)]$$

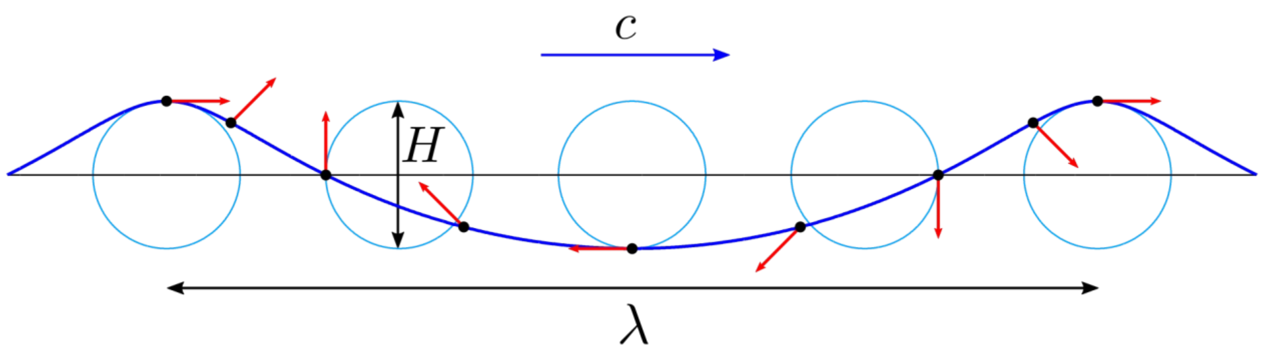
and then find the normal for all the coordinates.

$$N=[-ka\cos f, 1]$$

Now the light is being updated with the wave.



3.3. Ocean Wave using Gerstner Wave.



As we all know real ocean waves don't move in a sine wave format, they move in a specific manner. The similar illusion of movement can be created using Gerstner's Wave equation which says the point of the plane should move in a circular motion rather than a straight up down displacement and neither the speed of the wave is arbitrary nor its amplitude; all of that depends upon the wavelength of ocean waves.

So all the formula used are laid down below:

As we need the circular motion both sine and cosine function is used.

$$f = k\,(D \cdot (x, z) - ct)$$

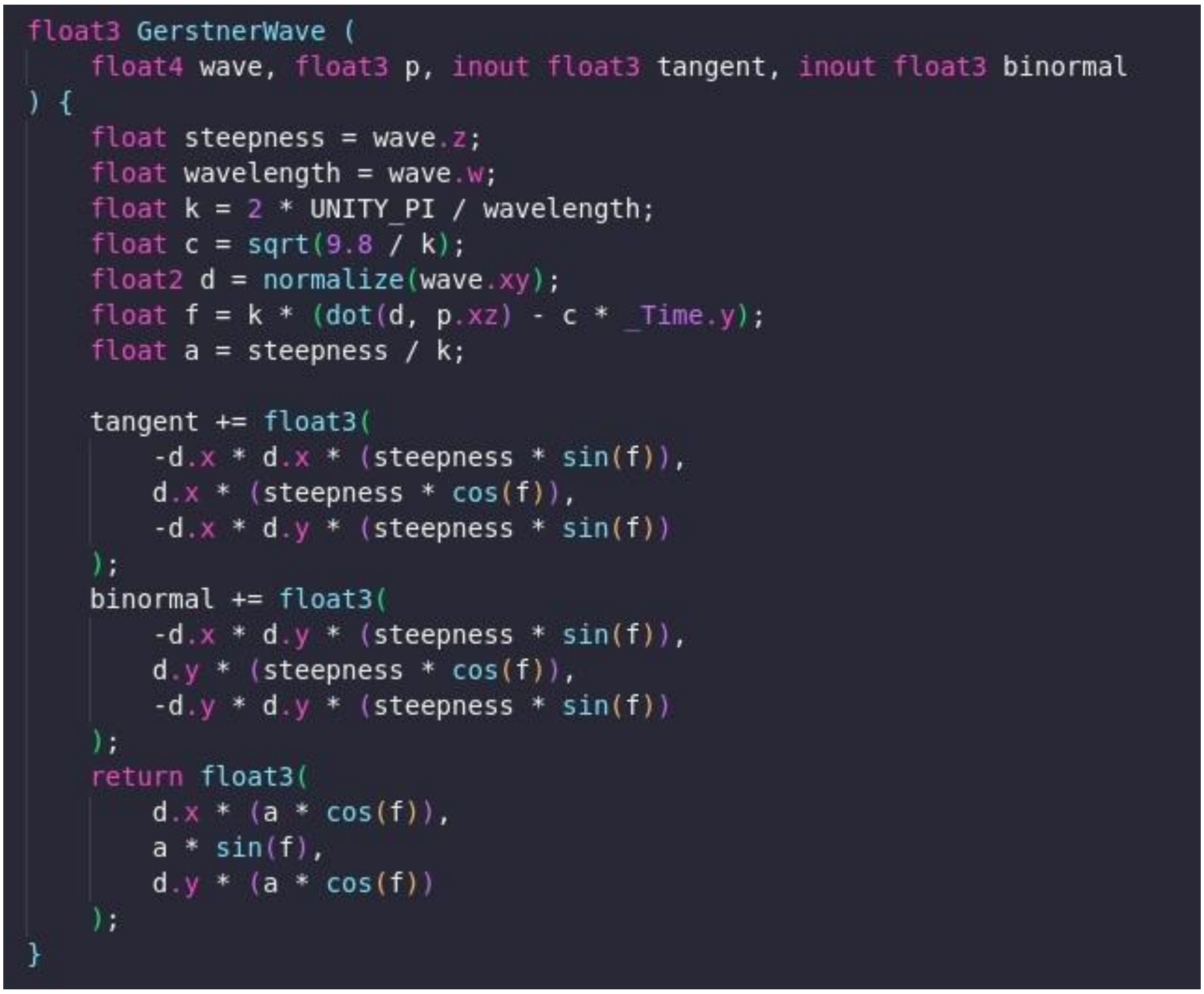
where

D is a vector which is used to give direction of propagation to the wave.

$$P_x = x + \sum_{i=1}^{n} D_{ix} \frac{s_i}{k_i} \cos f_i, \text{ and } f_i = k\left(D_i \cdot \begin{bmatrix} x \\ z \end{bmatrix} - ct\right).$$

This formula is used to add multiple waves and make it look more believable.

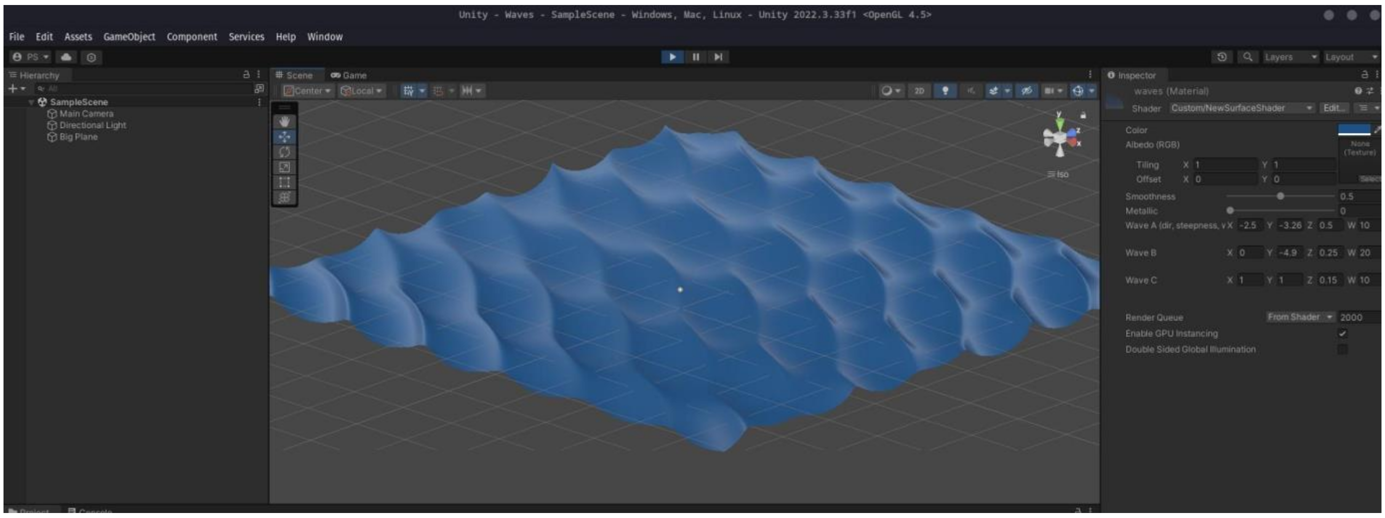
Code:



```
float3 GerstnerWave (
    float4 wave, float3 p, inout float3 tangent, inout float3 binormal
) {
    float steepness = wave.z;
    float wavelength = wave.w;
    float k = 2 * UNITY_PI / wavelength;
    float c = sqrt(9.8 / k);
    float2 d = normalize(wave.xy);
    float f = k * (dot(d, p.xz) - c * _Time.y);
    float a = steepness / k;

    tangent += float3(
        -d.x * d.x * (steepness * sin(f)),
        d.x * (steepness * cos(f)),
        -d.x * d.y * (steepness * sin(f))
    );
    binormal += float3(
        -d.x * d.y * (steepness * sin(f)),
        d.y * (steepness * cos(f)),
        -d.y * d.y * (steepness * sin(f))
    );
    return float3(
        d.x * (a * cos(f)),
        a * sin(f),
        d.y * (a * cos(f))
    );
}
```

Results:


Unity - Waves - SampleScene - Windows, Mac, Linux - Unity 2022.3.33f1 <OpenGL 4.5>
File Edit Assets GameObject Component Services Help Window
Hierarchy
SampleScene
Main Camera
Directional Light
Big Plane
Scene Game
Center Local
Iso
Layers Layout
Inspector

```
waves (Material)
Shader  Custom/NewSurfaceShader  Edit...
Color
Albedo (RGB)                                  None (Texture)
  Tiling   X 1    Y 1
  Offset   X 0    Y 0
Smoothness                                    0.5
Metallic                                      0
Wave A (dir, steepness, v X -2.5  Y -3.26  Z 0.5   W 10
Wave B                    X 0     Y -4.9   Z 0.25  W 20
Wave C                    X 1     Y 1      Z 0.15  W 10
Render Queue               From Shader  2000
Enable GPU Instancing      ✓
Double Sided Global Illumination
```

CONCLUSION AND FUTURE SCOPE

The program is successful in generating dynamic ocean waves and it is also controllable through the Unity’s engine UI in real-time. It fairly runs over 100 FPS and is not resource hungry therefore no further optimization is required.

This project is not only a simple wave program but it also provides the reader knowledge of GPU and Unity Engine’s Graphic pipeline.

REFERENCE

[1]
J. Flick, “Waves,” *catlikecoding.com*, Jul. 25, 2018.
https://catlikecoding.com/unity/tutorials/flow/waves/ (accessed May 20, 2024).
[2]
N. Corp., “Chapter 1. Effective Water Simulation from Physical Models,” *NVIDIA Developer*.
https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-1-effective-water- simulation-physical-models (accessed Jun. 12, 2024).
[3]
U. Technologies, “Unity - Manual: Shaders,” *docs.unity3d.com*.
https://docs.unity3d.com/Manual/Shaders.html (accessed Jun. 01, 2024).