Project Freedom – Making an Ocean Shader

By Robert Kilkenny and Manu Kolehmainen

The basis of this project was to create a working simulation that allows a user to observe an ocean shader in an empty scene to display different properties of shader matrices. The equations used in this project will be sourced from research papers and games with competent ocean graphics (specifically using resources from the games *Sea of Thieves* and *Atlas* and information from NVIDIA). To accomplish this, we needed to utilize the GPU of computers to handle many similar computations in parallel and optimize the computations we assign to the model to ensure that the simulation’s performance does not falter.

Before the simulation could be started, we had to make sure OpenGL was able to render shaders in the application by transforming the water surface’s vertices into the basis of the camera by using a series of transformation matrices[[1]](#footnote-1).

The first type of wave simulation we used for the ocean was a Sum of Sines approximation: . While a Gerstner wave (also known as a Trochoidal wave)[[2]](#footnote-2) could have also been used, these waves are complex and since we wanted to offer the ability to alter values in the simulation. It was likely the user could insert values to break the simulation and make unnatural waves, for Gerstner waves can curl easily if only changing the amplitude of the function. Also, this function has an easily computed partial derivative, which will be used for expressing the normal of the mesh which is required for lighting and reflections.

However, this simulation is prone to creating patterns in the ocean shader and can be awkward to apply to the mesh due to limitation in how we can tile it, so we then decided to also use the JONSWAP spectrum, which used real-life data to model ocean currents in different conditions.[[3]](#footnote-3) While this complicated calculating the normals of each vertex, using the Fast Fourier transform we could calculate the waves by their frequency[[4]](#footnote-4) to generate more waves for less obvious tiling in the simulation.

*JONSWAP Spectrum[[5]](#footnote-5)*

By adding smaller features in conjunction with these wave simulations, such as specular reflections with a skybox, we could make the waves look much more realistic akin to how the waves behave in AAA games. By determining the reflection of the skybox by the angle of the vertex to the image and the vertex to the camera, we can create an approximate (specifically a Schlick’s approximation) for the strength of the reflection (expressed by Fresnel equations) and make reflections appear realistic by becoming stronger as the angle the camera views it at decreases.[[6]](#footnote-6)

1. <https://learnopengl.com/Getting-started/Coordinate-Systems> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Trochoidal_wave> [↑](#footnote-ref-2)
3. <https://wikiwaves.org/Ocean-Wave_Spectra> [↑](#footnote-ref-3)
4. <https://en.wikipedia.org/wiki/Fast_Fourier_transform> [↑](#footnote-ref-4)
5. <https://wikiwaves.org/Ocean-Wave_Spectra> [↑](#footnote-ref-5)
6. <https://en.wikipedia.org/wiki/Schlick%27s_approximation> [↑](#footnote-ref-6)