Procedural Water Shading

Implementation Report

Bradley Klemick
Michael Lee
Dhrushit Raval

CSCI 719 Topics in Computer Graphics - Procedural Shading

Spring Semester

Professor Warren R. Carithers

05 / 03 / 2022

Introduction

In this project, we set out to explore how to simulate water in computer graphics using shaders. We wished to first determine how one might render the surface of the water itself, including wave patterns. We also wished to generate caustics in an underwater scene depending on the surface wave. We ended up figuring out how refraction worked and how it created caustics by focusing light rays at one point. We also explored to use GLSL and OSL to model the scenes for simulating caustics. In this report, we summarized all the things we explored and explained the code that generated the results in the last section.

Caustics

As shown in **Figure 1**, when light illuminates a drinking glass, we often see some bright regions in its shadows. These patterns are called caustics, which are caused when light shines through some translucent materials. This phenomenon can also be observed at the bottom of a swimming pool, ponds, etc. Caustics are formed while several light rays get concentrated in one region, making the area way brighter than its surroundings.

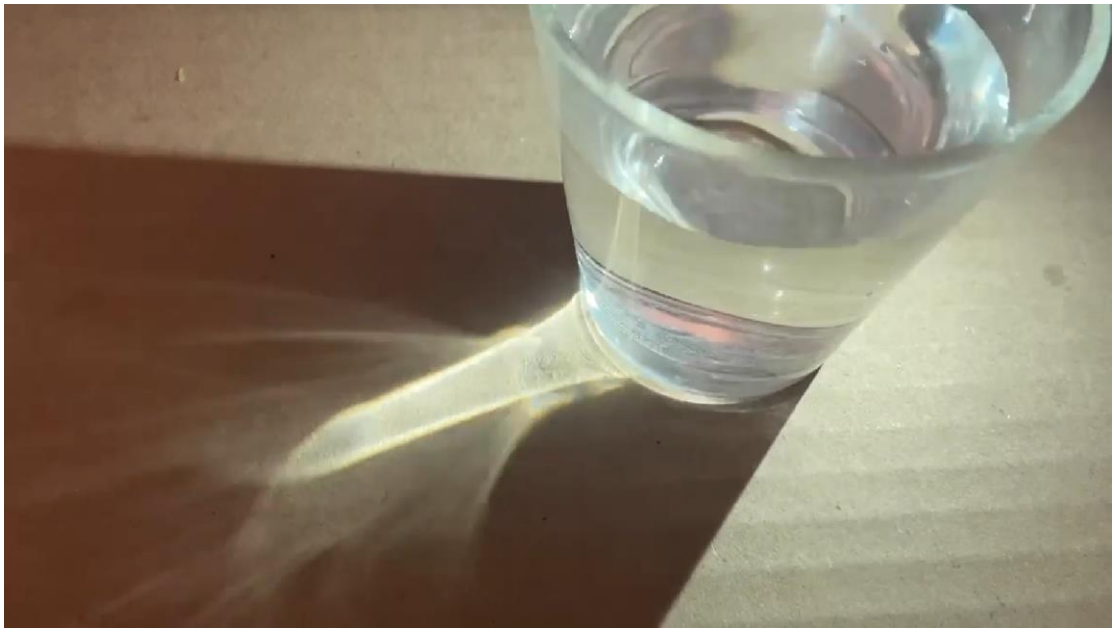

Figure 1: A glass cup of water causing light caustics on a cardboard.

Refraction, in which light travels from one media to another and changes its direction, plays a key role in causing caustics. Since light follows Snell's law (shown in **Figure 2**), how much the light changes its direction depends on the refractive indices of the medium. When light enters a media having higher

index of refraction than the current media's, it tends to bend toward the normal of the interface. On the contrary, when light enters a media having lower index of refraction than the current media's, it tends to bend away from the normal of the interface. Snell's law is mathematically defined as below:

$$\eta_1 \sin\theta_1 = \eta_2 \sin\theta_2$$

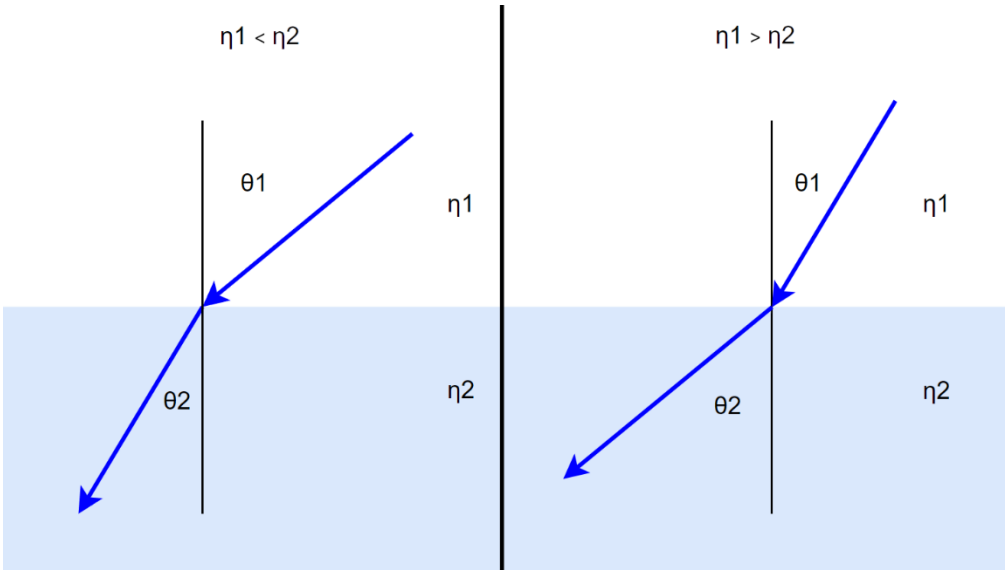

Figure 2: Illustration of Snell's law where η is the index of refraction: left: $\eta_1 < \eta_2$; right: $\eta_1 > \eta_2$.

As shown in **Figure 3**, when multiple light rays go from air (η = 1.0) into water (η = 1.33) having a flat surface, they will all bend towards their own local surface normal and show uniform distribution of light at the bottom of the water without any caustics pattern.

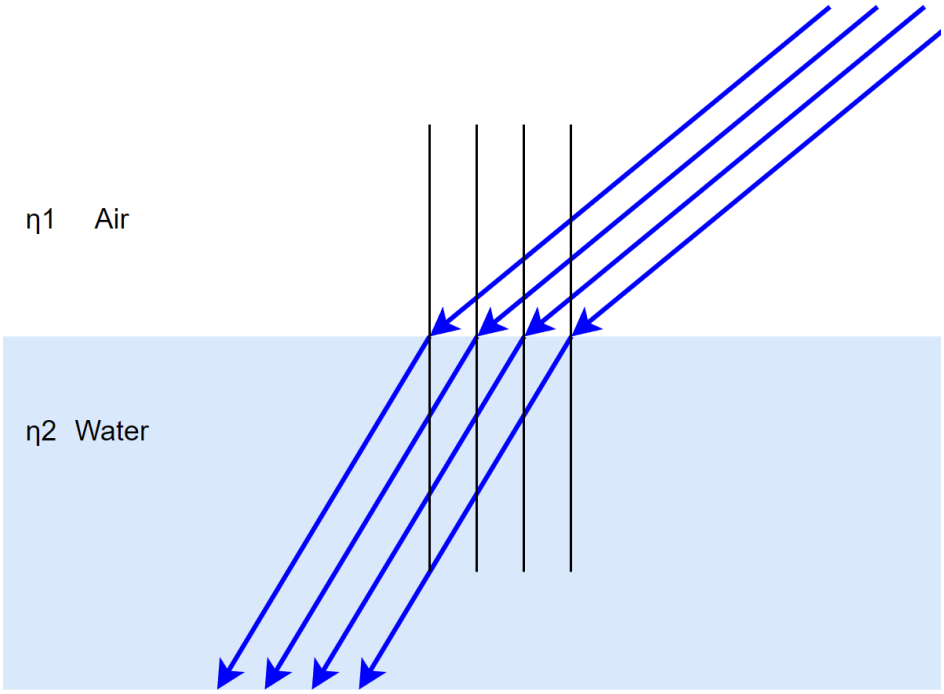

Figure 3: Refractions of multiple light rays by a flat surface of water.

Figure 4, on the other hand, depicts the situation when multiple light rays go from air into water having a curved surface. The light rays will be refracted differently and eventually focus on a small area at the bottom of the water and form a caustics pattern.

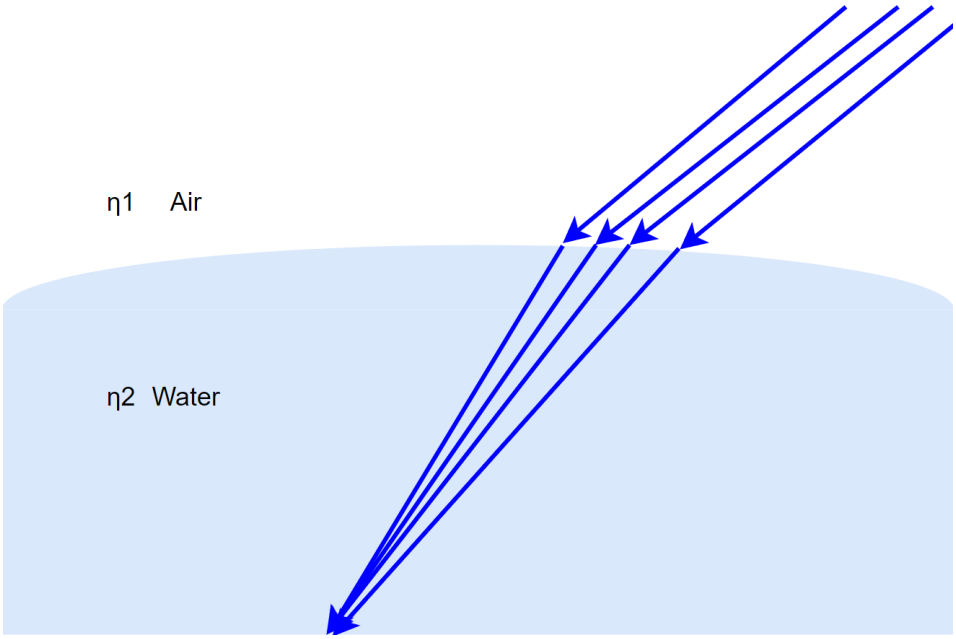

Figure 4: Refractions of multiple light rays by a curved surface of water.

In our project, we assigned some waves, including sin waves and Perlin noise's waves, to the water surface and followed Snell's law to created caustics patterns on the ground underwater.

Water Surface Implementation

In order to create realistic water surface and caustics patterns, we tried several wave functions for simulating the non-uniform surface of the water and kept the best ones:

1. Sum of Sine Waves

We learned this idea from the article on GPU Gems (Finch). Water surface could be treated as a combination of multiple sine waves, which generated some crests and troughs. The sine waves that we used had the following parameters:

• dir: the general direction the waves move forwards with respect to time

• amplitude: the height of the waves above its origin line/plane

• speed: the speed with which the waves travels

• time: the current world time for determining the phase of the wave

```
sinwave = amplitude * sin(PI * ((dot(texCoord, dir) + speed * t) * frequency - 0.5));
```

The sine function is defined as below:

2. Perlin Noise Waves

Another type of waves we tried was generated by 3D Perlin noise with uv values and the water depth as its parameters:

• uv: a vector having the uv coordinates

• depth: the water depth

• frequency: the frequency of the noise

• factor: a factor that stretch the noise in x direction

• amplitude: the height of the waves above its origin line/plane (not applied to the noise function directly)

```
noise("perlin", point(uv[0] * frequency / factor, uv[1] * frequency, depth));
```

Displacement shading was performed in our project in order to deform the water surface. In GLSL, tessellation had to be done before the displacement; in Blender, tessellation modifiers were applied to create subdivisions before the

displacement. Since GLSL and Blender both didn't automatically modify the surface normal according to the displacement, bump mapping was also performed. The procedures of modifying the surface normal were depicted in **Figure 5**. We first found the partial derivatives in tangent and bitangent directions, and then we calculated the tangent (blue) and bitangent (red) vectors. Once we had those vectors, we could get the modified surface normal by taking the cross-production of the bitangent vector and the tangent vector.

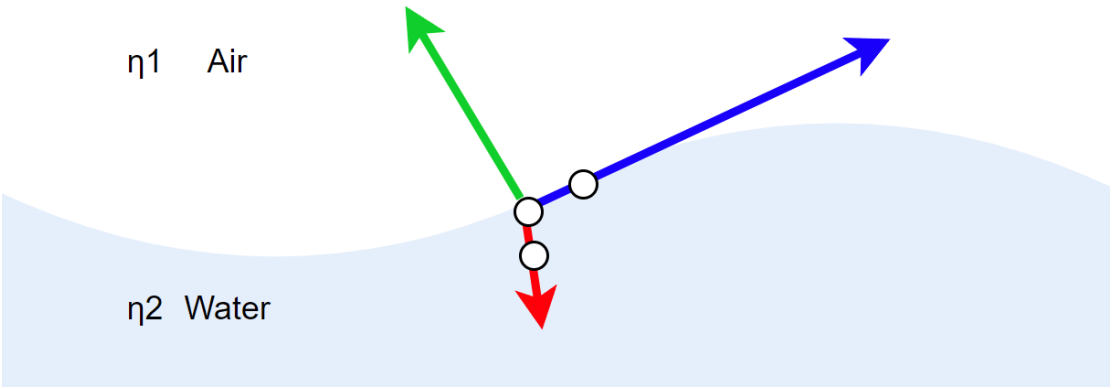

Figure 5: Illustration of finding the modified surface normal. Red arrow: bitangent vector; blue arrow: tangent vector; green arrow: new surface normal.

Caustics Implementation

Caustics calculations were all performed in the fragment shader of the ground, using an idea in Guardado and Sánchez-Crespo's chapter of GPU Gems. As shown in **Figure 6**, we first cast a ray from the ground plane in its normal direction (N_g). After that, we acquire the point on the surface plane and determined its displacement and normal (N_s). And then we calculate the refracted vector (R) and compare it with the light vector (L). Finally, we decide the intensity of caustics pattern based on the dot-production of R and L and raising it to a power. (Details can be seen in our code files.)

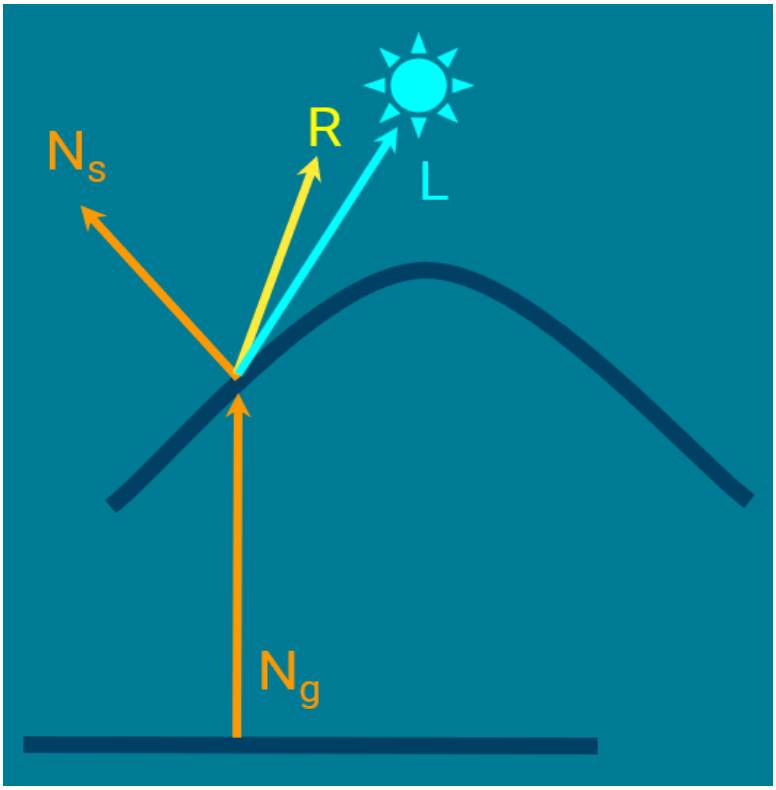

Figure 6: The procedures of calculating caustics intensity.

Results

Figure 7 shows the sum of sine waves result rendered by GLSL. Although the results of sum of sine waves were not bad, they seemed somehow not realistic enough due to its periodic patterns.

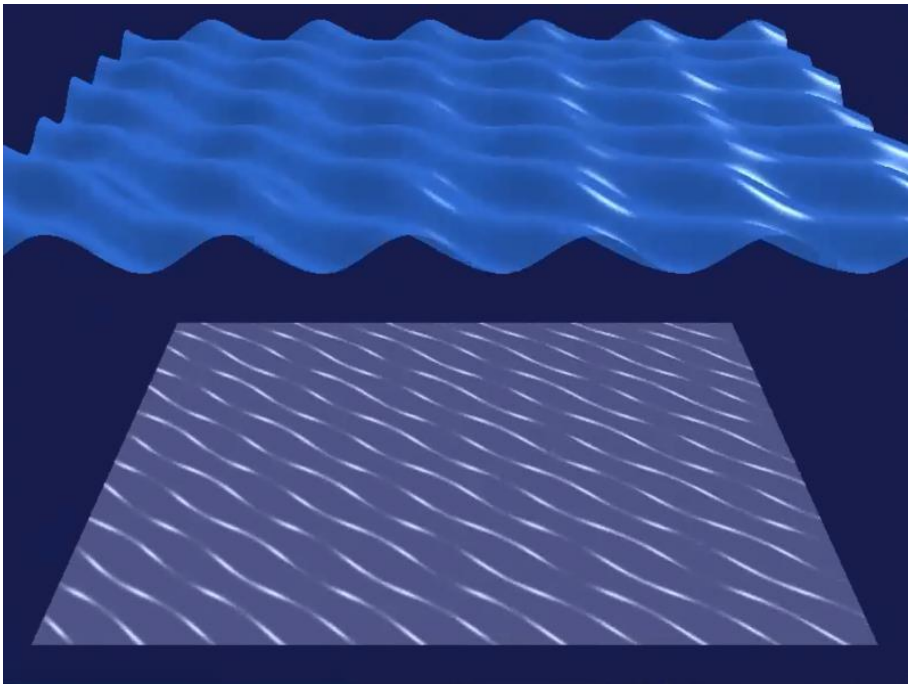

Figure 7: Our result using sum of sine waves as the water surface. (GLSL)

Figure 8 shows the sum of Perlin noise waves result rendered by GLSL. The shape of the water surface is not determined by a single or a few factors. For example, wind blowing, boats passing by, creatures swimming underwater, all these motions can cause disturbances to the water surface and generate some random waves. However, most mathematical waves such as sine waves are not able to encompass this randomness, and therefore, Perlin noises turn out generating the best results we got.

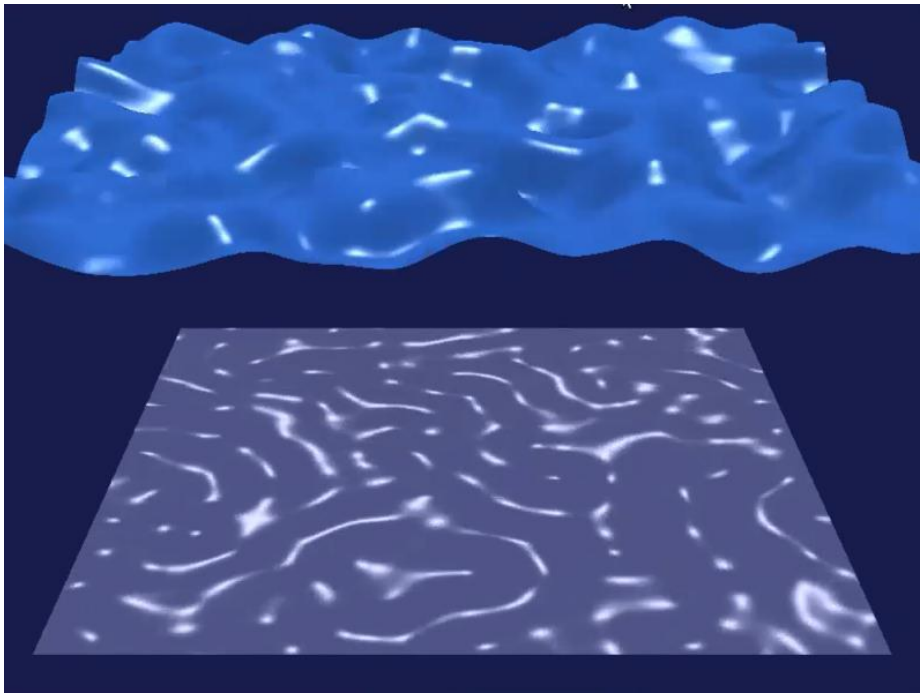

Figure 8: Our result using sum of Perlin waves as the water surface. (GLSL)

Figure 9 and Figure 10 show the 3D Perlin noise waves results rendered by Blender. Frequency and factor of the Perlin noise in **Figure 9** are 7.5 and 1.5 respectively. The result of this combination models a water surface which is similar to the surface of a lake or a river. When we increase the frequency to 150.0 and factor to 9.0 in **Figure 10**, the water surface becomes similar to the surface of sea. These results prove that Perlin noises are robust in modeling water surfaces.

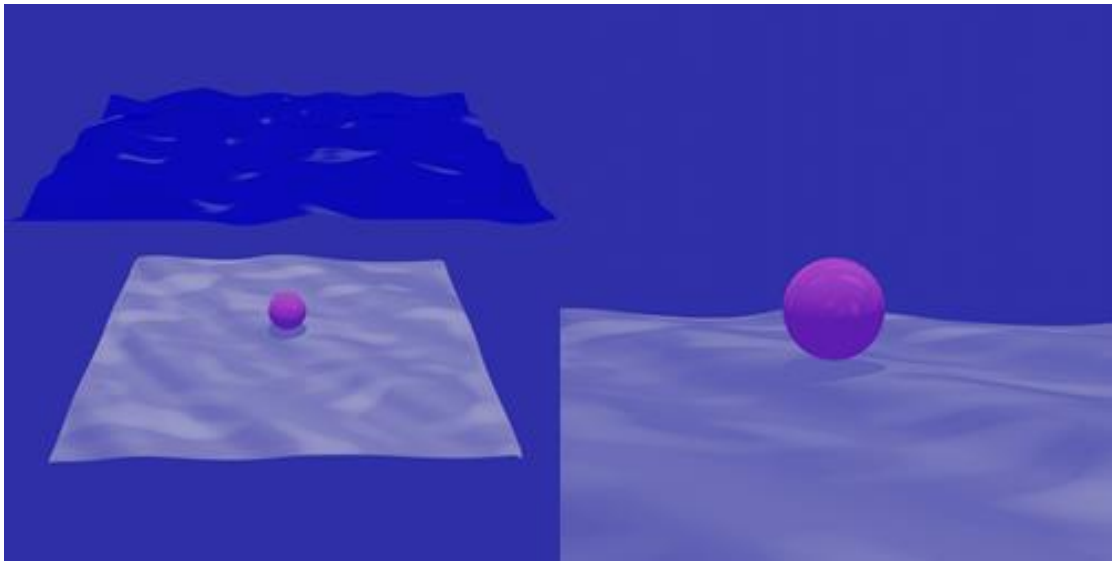

Figure 9: 3D Perlin noise waves results rendered by Blender (frequency = 7.5; amplitude = 12.0; factor = 1.5). Left: surface + ground scene; right: only underwater scene.

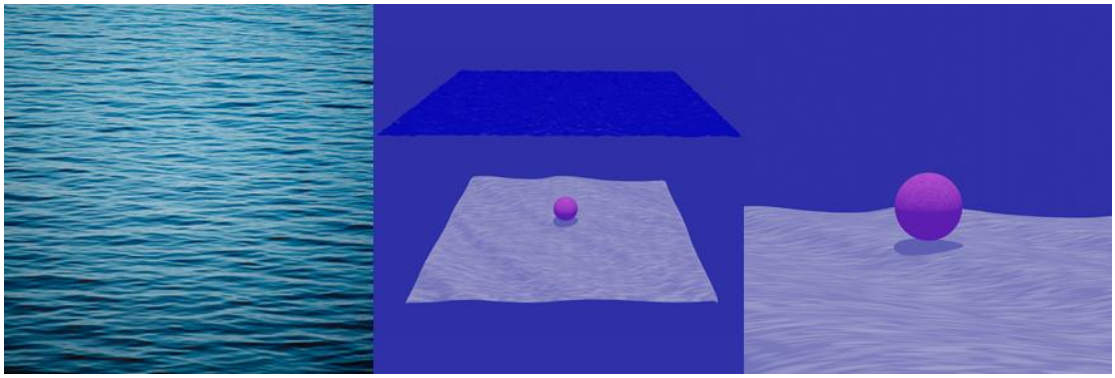

Figure 10: 3D Perlin noise waves results rendered by Blender (frequency = 150.0; amplitude = 6.0; factor = 9.0). Left: sea surface; middle: surface + ground scene; right: only underwater scene.

Future Work

Currently, our water surfaces show only the specular reflections. We can add reflections of some scenes out of the water in the future, for example, a Disney castle as shown in **Figure 11**. Also, we can perform volumetric shading underneath the water instead of just showing a uniform color.

Figure 11: A Disney castle with river reflecting it. (CSCI 711 Final Project - Michael Lee)

Although Perlin noises are robust in modeling water surfaces, they are not encompassing for every case. Therefore, we can try using more wave functions such as Gerstner waves, or potentially, waves that respond to stimuli, such as Renou's real-time implementation.

Finally, it is important to keep in mind the implementation of caustics we utilized is really a heuristic. Our solution starts at a point on the ground and

traces a ray back towards the light. The true, physically correct caustics require tracing rays from the light source, refracting them about the wave surface, and seeing where they concentrate on the floor, as explained by Wallace. If where the light rays are more concentrated should be more intensely illuminated. Photon mapping would implicitly create this more realistic and physically based effect. We could also investigate another approach to approximate this effect more efficiently, such as Wallace's approach, in which he traces rays from a parallel plane through the waves.

References

- Darles, Emmanuelle, et al. "A survey of ocean simulation and rendering techniques in computer graphics." Computer Graphics Forum. Vol. 30. No. 1. Oxford, UK: Blackwell Publishing Ltd, 2011.

- Finch, Mark. “Effective Water Simulation from Physical Models.” *GPU Gems*, Chapter 1. link

- Guardado, Juan and Daniel Sánchez-Crespo. “Rendering Water Caustics.” *GPU Gems*, Chapter 2. link

- Iwasaki, Kei, Yoshinori Dobashi, and Tomoyuki Nishita. "Efficient rendering of optical effects within water using graphics hardware." Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001. IEEE, 2001.

- Kryachko, Yuri. “Using Vertex Texture Displacement for Realistic Water Rendering.” *GPU Gems 2*, Chapter 18: link

- Renou, Martin. “Real-time rendering of water caustics.” *Medium*. link

- Thompson, Stephen, Andrew Chalmers, and Taehyun Rhee. "Real-time mixed reality rendering for underwater 360 videos." 2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). IEEE, 2019.

- Wallace, Evan. “Rendering Realtime Caustics in WebGL”. *Medium*. link