Real-Time Rendering of Water Caustics

Will Meinert

Introduction

For my final project, I wanted to simulate water caustics in real time at a reasonable framerate. Because of the complex interactions light rays have with the water and any particles in the water, water caustics are a very computationally expensive effect to simulate. However, they are an essential part of rendering realistic-looking water, so many have found ways to either cheat or render the caustics in a less expensive way. For my project, I focused on rendering caustics in a less computationally expensive way to retain high performance as well as high-quality results, even if it meant not simulating the accurate physics of the scene.

Research

The main source of information that I used for this project was Nvidia's CPU Gems. Chapter 2, written by Juan Guardado and Daniel Sánchez-Crespo, contained a very in-depth and detailed explanation of how to render water caustics using backward raytracing. I also used a blog post written by Martin Renou that explained his method of rendering water caustics using an environment map.

Different Methods

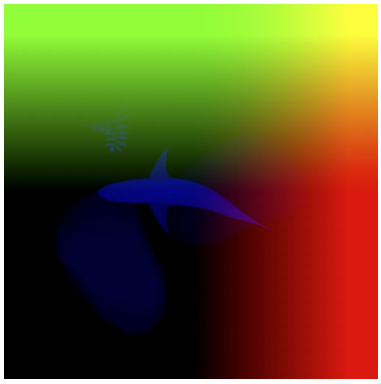
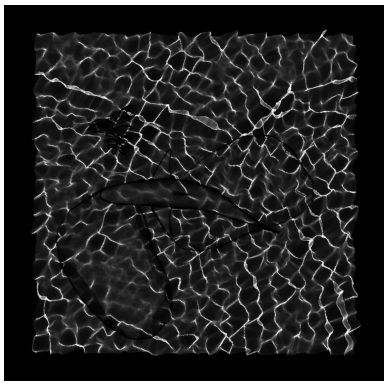
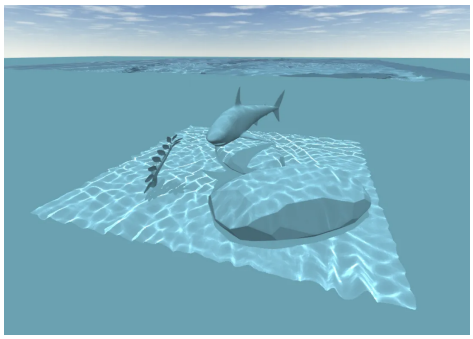
There are a multitude of different ways of rendering water caustics all with different benefits and drawbacks. For my project, I looked at three different methods before deciding which one to fully implement.

Photon Map

The first of the methods that I looked at was photon mapping. This method produces incredibly accurate and high-quality caustics because of its ability to trace the paths of the photons and then be able to model where the concentrated patches of light will form. However, photon mapping is far too expensive and can't be done in real-time very easily.

Environment Map

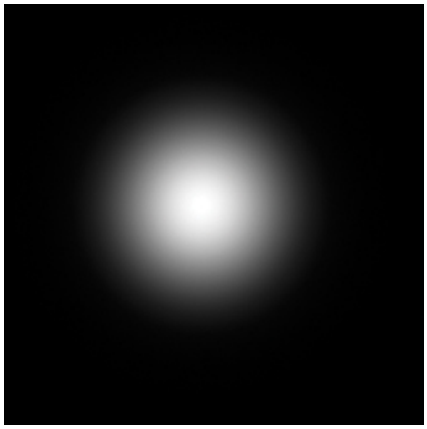
The next method that I looked into was Martin Renou's method of using an environment map. In this method, an environment map is created in the first pass that stores the depth values of the scene in a texture. The second pass then takes those depth values and calculator the caustic intensity and saves that value to a texture which is then applied to the scene.

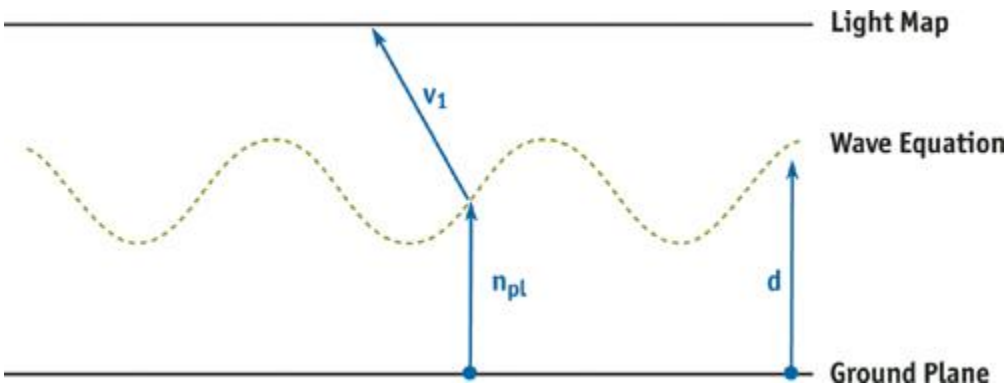




This method produced pretty good quality results and was physically accurate to the waves. However, when looking at other implementations of this method, the framerate was not at the level that I desired.

Backwards Raytracing

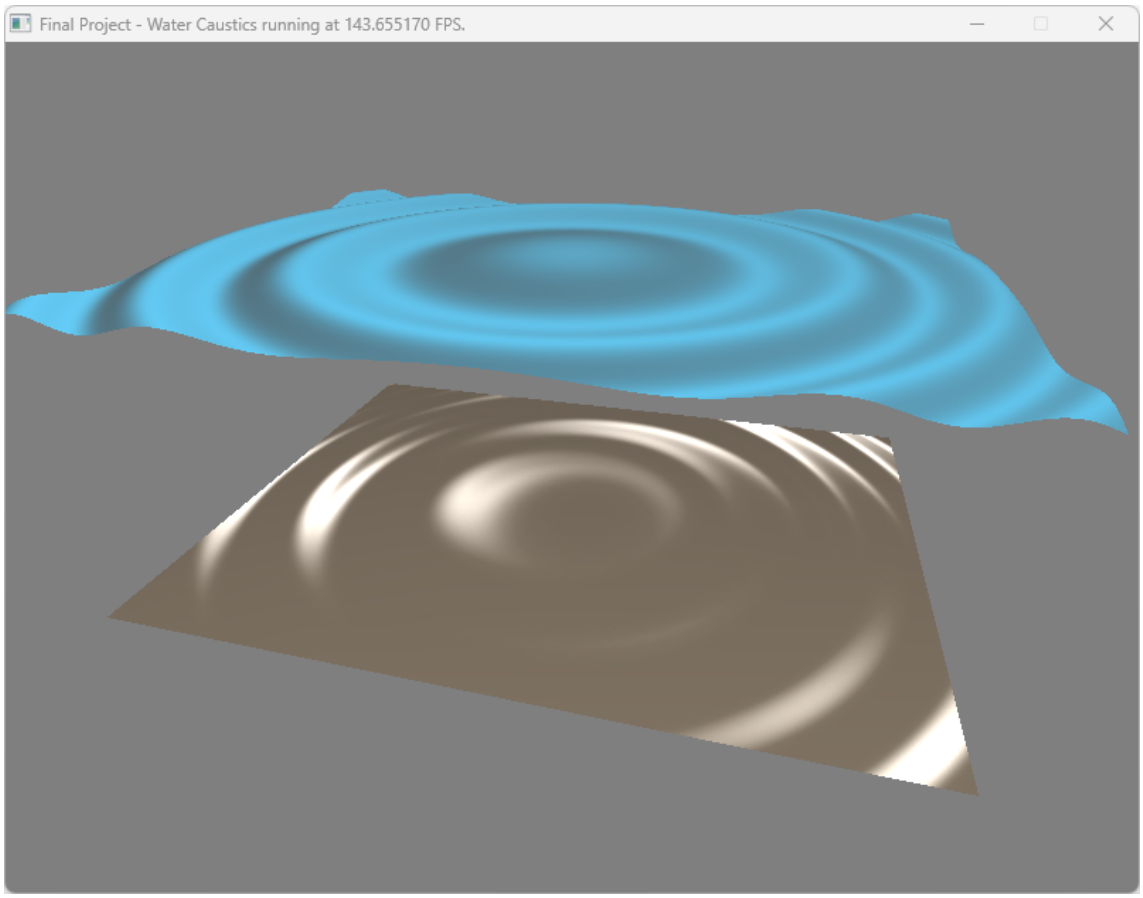
The last method that I looked at came from Nvidia's CPU Gems, Chapter 2. This method sends a vertical ray from every fragment of the ground plane and when the ray collides with the water plane it then calculates the angle of the ray using Snell's law in reverse. This new angle is then used to find where the ray lands on a light map and that coordinate on the light map is then applied to the original ground plane. This method is more of an aesthetics-driven method for rendering underwater caustics as it does not completely accurately simulate the physics of the light rays and focuses more on producing a higher-quality render. This results in this method being much more efficient than the other two methods.





Final Results

Screenshot



Demo Video

https://youtu.be/Tm25yJASxdE

Source Code

https://github.com/whmeinert/CSCI444_FinalProject/tree/main

Conclusion

The final results of my projects achieved what I set out to accomplish, the program renders realistic-looking water caustics in real time while retaining high performance. The method sued to achieve these results was backward raytracing as explained by Chapter 2 of Nvidia's CPU Gems. In the future, I would like to fully implement Martin Renou's method of using environment mapping in the future and see if I could improve the performance without impacting the final result. I would also like to further improve the look of my final result. I could add shadow mapping on the ground so that you could place items into the scene under the water. I could also add screen space reflection on the water as well as add in more realistic waves to improve the overall quality and realism of the scene.

References

"Caustic (optics)." *Wikipedia*, https://en.wikipedia.org/wiki/Caustic_(optics). Accessed 2 May 2023.

"CS148 Project Report." *Project Eugene*, https://www.projecteugene.com/caustics.html. Accessed 2 May 2023.

Fernando, Randima, editor. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-time Graphics*. Addison-Wesley, 2004. Accessed 2 May 2023.

Gérard, Martin, and Alan Wolfe. "Real-time rendering of water caustics | by Martin Renou." *Medium*, 27 August 2020, https://medium.com/@martinRenou/real-time-rendering-of-water-caustics-59cda1d74aa. Accessed 2 May 2023.

Wallace, Evan. "Rendering Realtime Caustics in WebGL | by Evan Wallace." *Medium*, 7 January 2016, https://medium.com/@evanwallace/rendering-realtime-caustics-in-webgl-2a99a29a0b2c. Accessed 2 May 2023.