Real-Time Rendering of Ocean Water

Aaron Hornby
10176084

- ***GOAL:***
    - ***Problem statement:***

I would like to attempt the implementation of real-time rendering of a simulation of ocean water/waves with a particular focus on water foam interaction with shorelines.

    - ***Project goals/benefits:***

The results should provide a well-documented open-source repository that can be studied by other students or others with interest in this topic. The project will primarily benefit hobbyist game developers who wish for an easier entry point into a highly popular feature of many games.

- ***CHALLENGE:***
    - ***Why the problem is hard:***

This project will be a challenge since it incorporates aspects of rendering, modeling and animation. The rendering component will of course be the main focus, and could be used as a way to minimize the modeling effort (e.g. bump/normal/displacement mapping). The animation will probably be a very simple looping mechanism. Water rendering also involves many rendering aspects such as reflections, refractions, fresnel reflectance, foaming, etc.

    - ***Other approaches and their limitations:***

Traditionally, water has been rendered as a texture-mapped quad. This doesn't look very realistic and isn't very extensible to many modern use cases. The referenced approaches (included below), vary from using more realistic techniques for rendering planar water to actually perturbing the surface with displacement maps. There also seems to be a limitation of white water effects as waves shear against shorelines. Also, many implementations don't seem to focus on larger amplitude waves.

- ***APPROACH:***
    - ***My approach:***

I can use an iterative approach to building up the project, through consulting my references. I can start by attempting the simpler planar method, with the surface being static (no animation). Once I have achieved a consistent level of realism, I can then introduce a looping animation. I can then test how well the simulation scales and its impact on the rendering time. Furthermore, I can play around with the displacement

mapping method. Finally, optimization will come into play at some point yet to be determined, but will involve LOD scaling of the underlying grid structure.

- ***Reasonability of success:***

I believe a reasonably photo-realistic real-time simulation of waves is probable mainly due to the large knowledge base surrounding water rendering in general. Of particular usefulness is that of the included shader code in [Johanson 2004]. If I am ever stuck on implementing one feature a certain way, I have at my disposal a plethora of other attempts and techniques to consult.

- ***METHODOLOGY:***
    - ***Projected timeline:***

The task list is as follows (builds mainly upon what is outlined by [Johanson 2004])…

TASK	**DEADLINE**	**CHALLENGE?**
Setup rendering boilerplate.	Feb 22	
Load in terrain model (with a simple texture) that can be used to simulate islands and ocean depth.	Feb 22	
Load in skybox.	Feb 23	
Camera overhaul.	Feb 27	
Roughly implement projected grid, apply height map, apply perfect mirror global reflections with skybox, test fresnel.	Mar 16	
Completely implement the projected grid concept and refine height map displacement controls.	Mar 17	This will have to be adjusted once a better idea of how all components play together.
Compute normals for the water field to test out the global reflections properly.	Mar 17	

Implement a simple looping animation (NOTE: I have an idea written down on paper that uses a simple sine wave animation over the wave height that should work)	Mar 18	This may turn out not to be very simple due to visual artifacts likely appearing if done incorrectly.
Phong specular sunlight (of varying colour) and refine fresnel.	Mar 19	
Implement local reflections with terrain islands. Include UI slider for water surface distortion amount.	Mar 22	
Implement refractions	Mar 23	
Measure any impact that scaling the projected grid resolution has on runtime performance	Apr 6	
Implement sea foam, play with exaggerated foam for shearing along shorelines.	Apr 7	This will be challenging to visually replicate without looking fake (too thin? Too thick? Too predictable?)
Improve the UI with better controls over every parameter.	Apr 11	
Optimize	Apr 13	It may prove difficult to balance visual fidelity

		and the scale of the simulation
Implement Perlin Noise based height field generation	<OPTIONAL>	NOTE: currently i'm just sampling from a texture, but I could generate a new texture each time you run program which would be exportable (along with displaying a seed value).

- ***Fallback plan:***

I have many options and references. For generating a height field, I can either use noise or FFT. I can vary between bump mapping, normal mapping, and displacement mapping.

- ***METRICS:***

Fortunately, this topic has been researched a ton, and thus I should have ample reference material (images, video, simulations, other applications, etc.) to compare my results to.

My application should meet at least the following criteria:

1. Able to run at >30fps (preferably 60fps) on the Linux graphics lab computers.
2. Provide a simple UI to adjust parameters (such as sun position/angle, surface "bumpiness", animation speed, etc.)
3. Suitably realistic for real-time applications, with an ability to scale up or down the simulation.

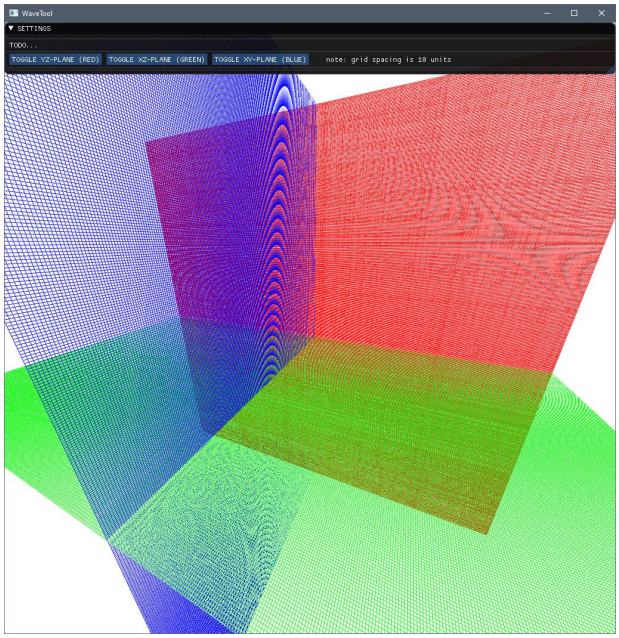
- ***SUMMARY:***

This project will serve as a great learning experience for myself in implementing more advanced rendering techniques. Of particular use will be gaining experience in discovering the challenges with optimizing for real-time applications, while maintaining visual fidelity. I will also develop a better ability to work with researched work and hopefully build on their efforts.

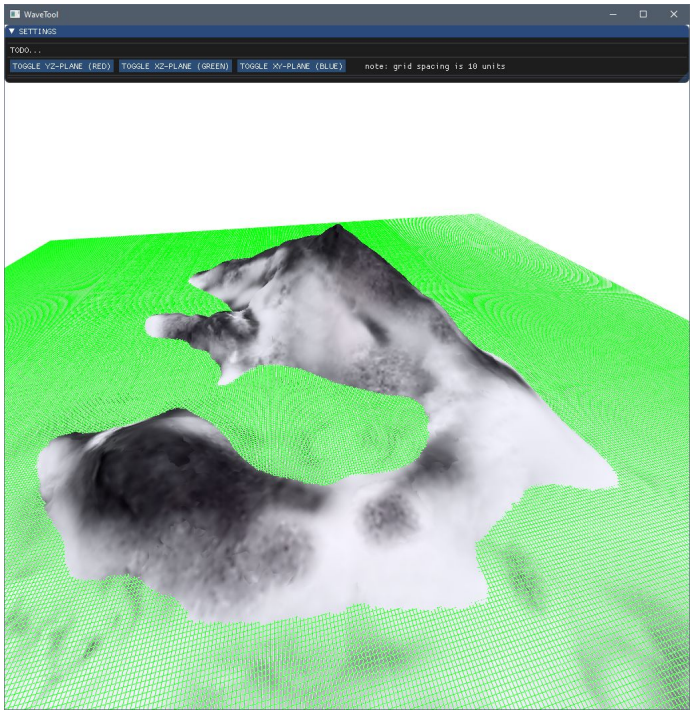
- ***PRELIMINARY RESULTS:***

I should preface this section by saying that progress is a bit behind schedule, but there are some small steps achieved.

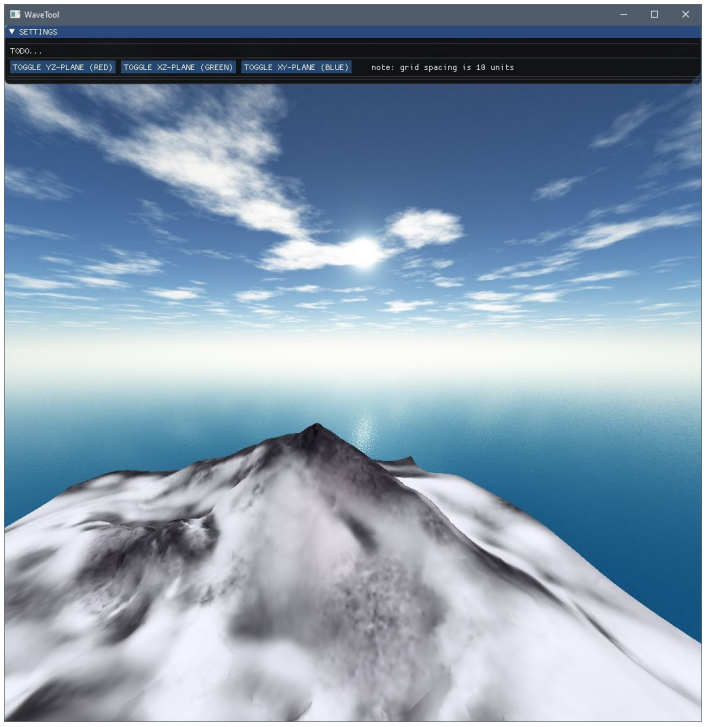
First off, my application can render the 3 Cartesian world planes for debugging purposes (each is either solid R, G or B), but the XZ-plane (green) is particularly useful at this stage for visualizing the water displacement that will be described below shortly.



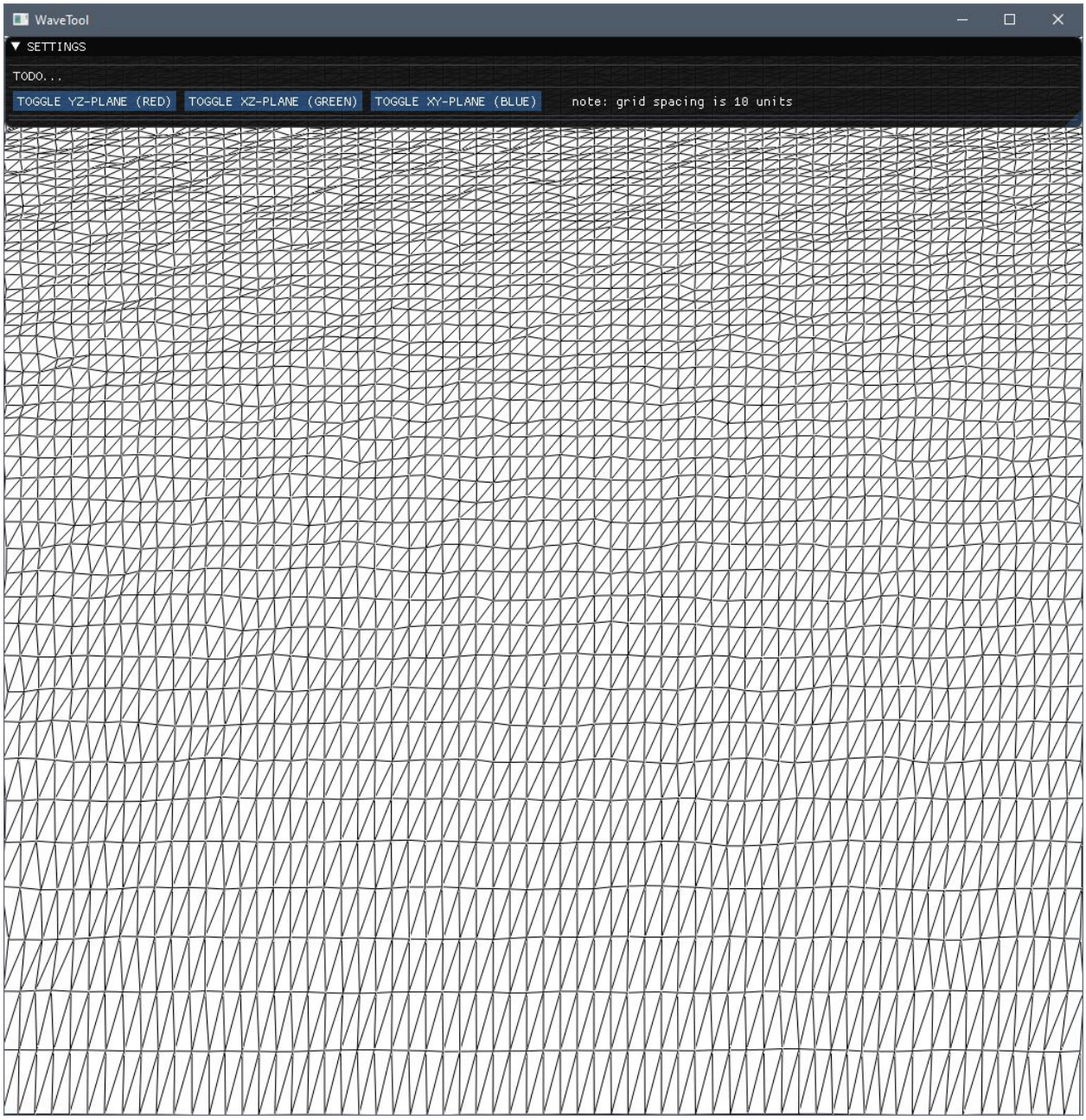
Second, a Mount Everest mesh gets loaded in to serve as some terrain (notably ocean floor) which I will use in the future to test local reflections and refractions.



Next, I implemented a skybox from some cubemap textures I found. This will be used for global reflections (environment mapping) on the water surface. I also reworked the camera system to be free-roaming.

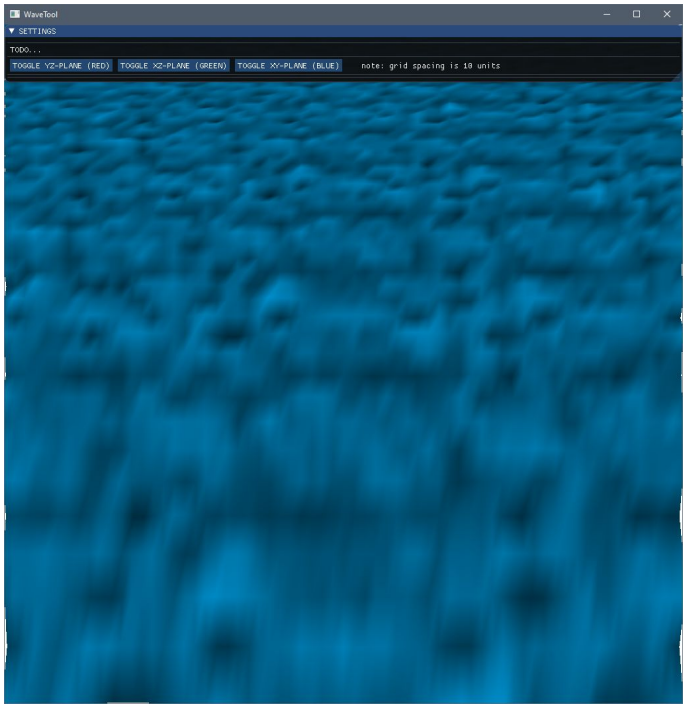


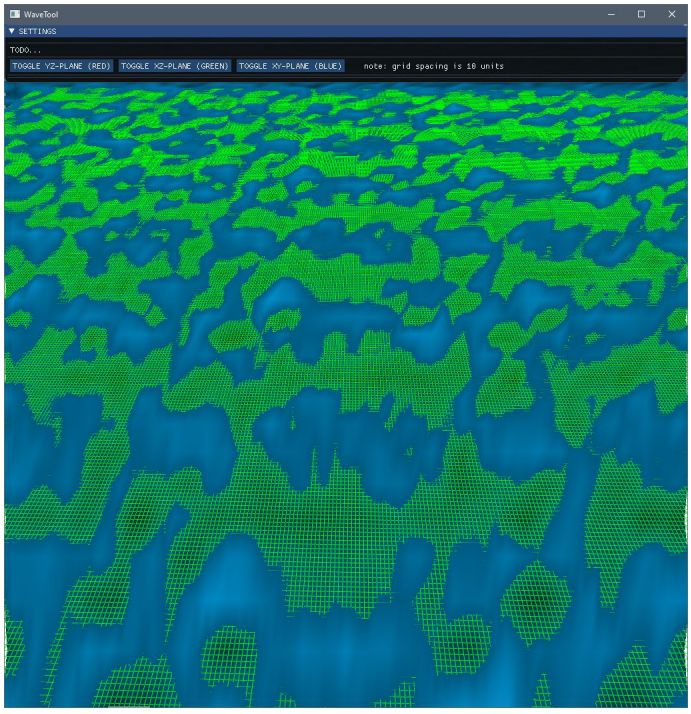
Next, I moved onto implementing the water grid using the projected grid method. This has been a little tricky to implement and currently I am able to render the grid (as seen below), but only when the camera frustum can entirely project onto the XZ-plane. This currently means that I disable rendering of the water grid when you can see above the horizon. The main reason why this is currently a problem is due to the fact that I am treating the camera as the projector. The next step is to implement the projector frustum which will always point towards the XZ-plane and thus the intersection tests will pass and render the water. I also believe the formula I'm using for ray-plane intersections suffers from "high t-values" (where t is the intersection coefficient in the standard ray equation) near the horizon. So I will also try replacing that with another formula I found.



NOTE: the grid is wavy since it is the heightfield rendered

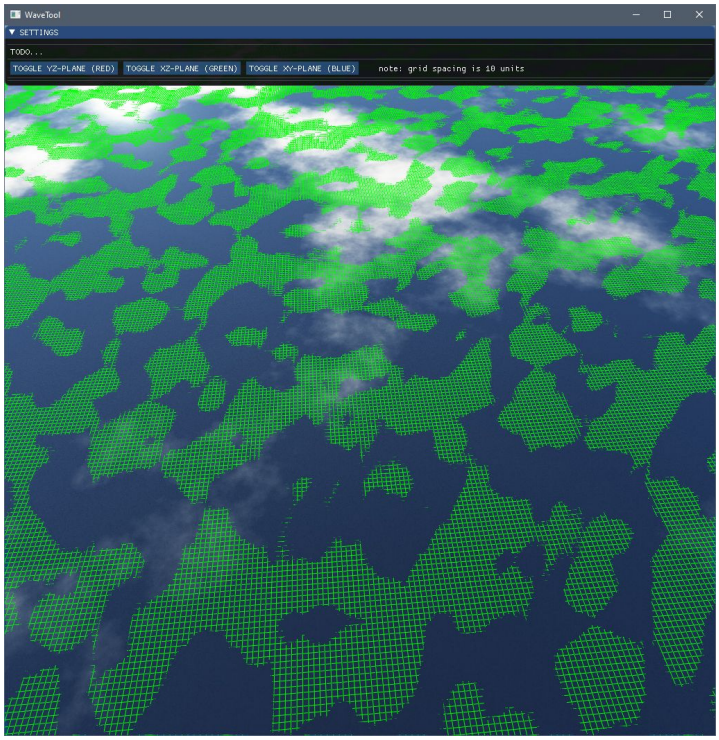
Here is the water heightmap rendered (with the noise texture also used as a multiplier for a demonstration blue colour). There are gaps on the sides which can later be fixed by scaling up the grid slightly.



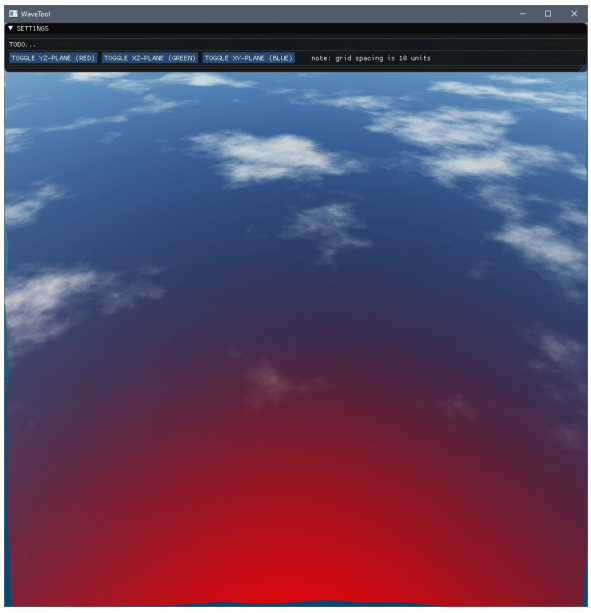


NOTE: the green XZ-plane is useful to show the water heights

Next, I roughly implemented global reflections of the skybox onto the water grid (but, it simply acts as perfect mirror reflection right now as I still need to compute the normals from the heightfield).



Finally, I quickly added Fresnel reflectance (work-in-progress) for LERPing between a test colour (red) and the global reflection colour.



- ***REFERENCES:***

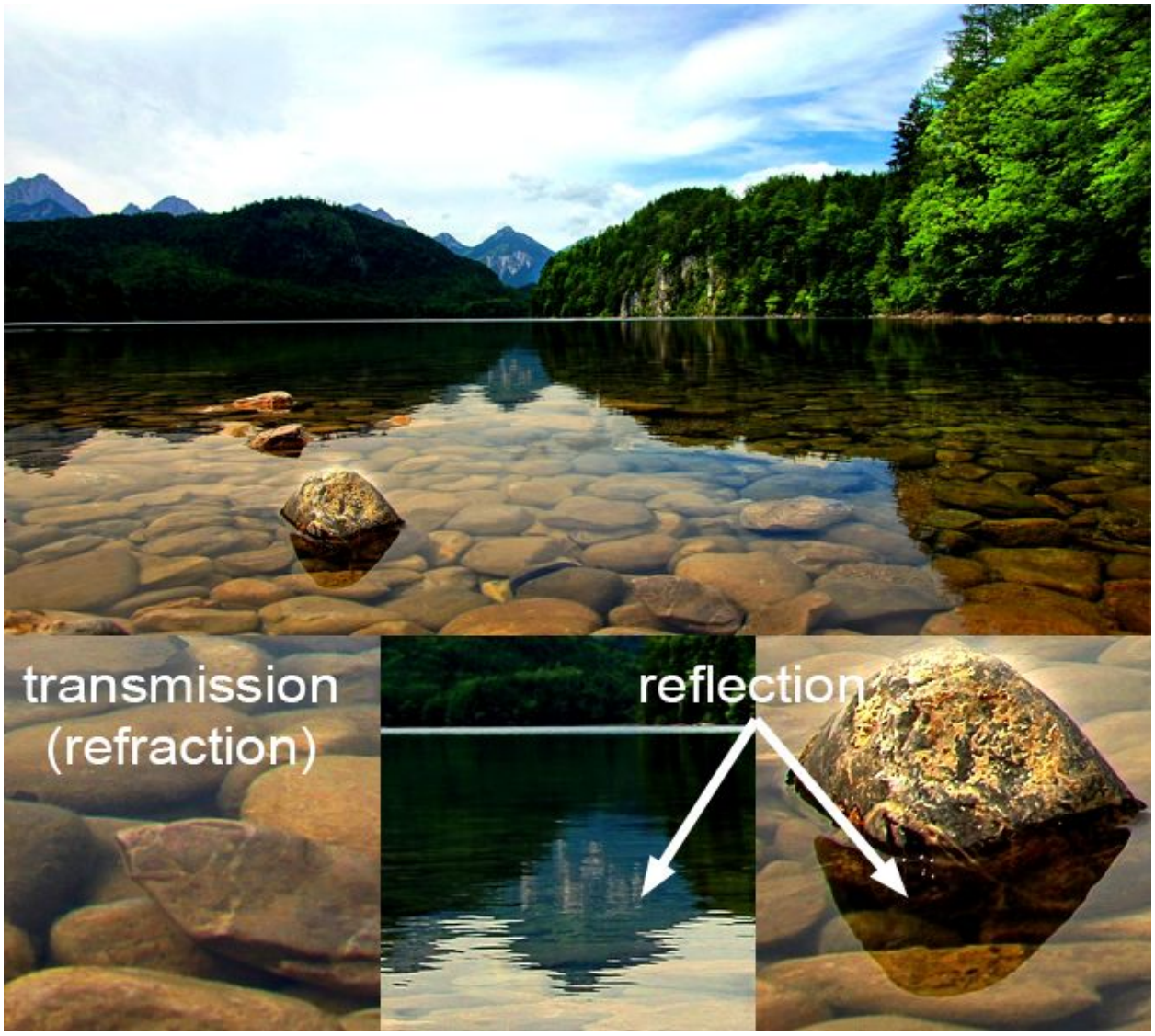
Claes Johanson, “Real-time water rendering - Introducing the projected grid concept”. Lund University. 2004.
https://fileadmin.cs.lth.se/graphics/theses/projects/projgrid/projgrid-hq.pdf

Yaohua Hu, Luiz Velho, Xin Tong, Baining Guo, Harry Shum, “Realistic, Real-Time Rendering of Ocean Waves”. Computer Animation & Virtual Worlds 17(1):59-67. 2006.
http://www.xtong.info/rtwave.pdf

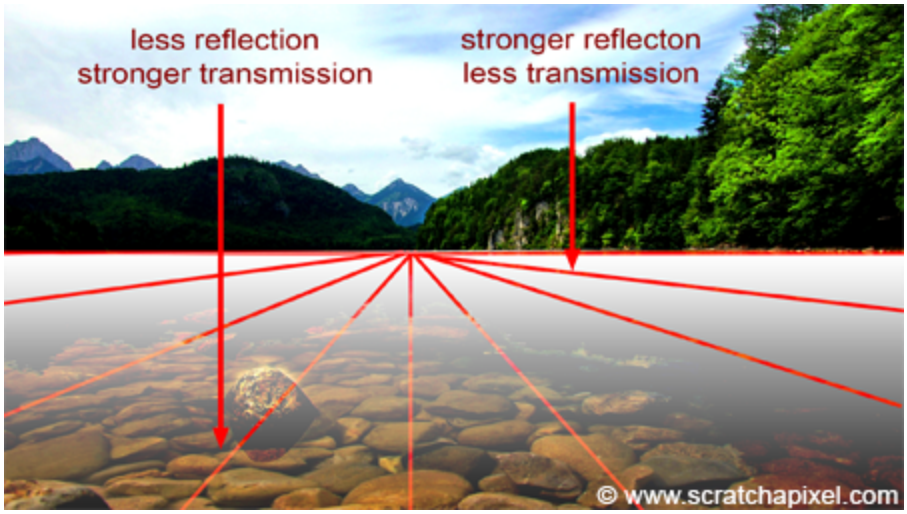
Shunli Wang, Fengju Kang, “Automatic Optimization for Large-Scale Real-Time Coastal Water Simulation”. Mathematical Problems in Engineering. 2016.
https://www.hindawi.com/journals/mpe/2016/9034649/

Robert Golias, Lasse Staff Jensen, “Deep Water Animation and Rendering”. GDCE. 2001.
https://www.gamasutra.com/view/feature/3036/deep_water_animation_and_rendering.php?print=1

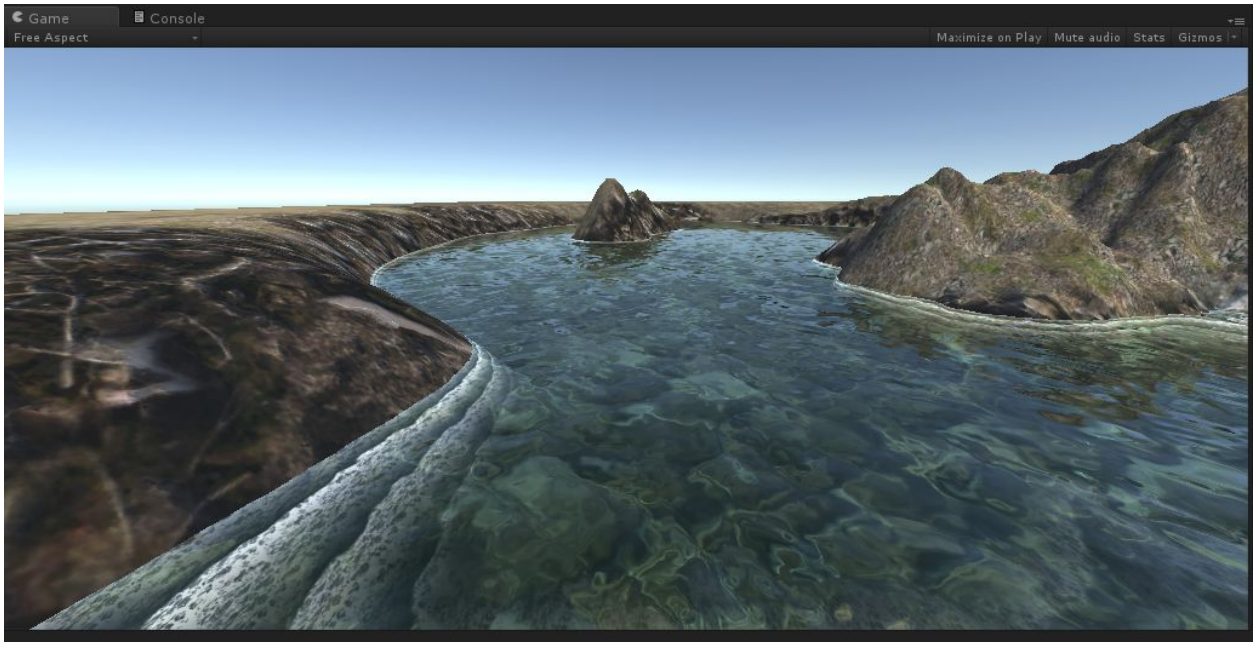
- ***IMAGES:***



reflection/refraction



Fresnel reflectance



An example of shoreline white water / foam

Sunlight beam and small island features

Foam