Real-Time Rendering of Ocean Water With Dynamic Skybox

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.	Apr 4	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.	Apr 4	
Huge update implementing dynamic	Apr 6	

skybox (load uv-sphere, paint and load custom sky-gradient, load 2 base skyboxes (clouds + stars), grayscale the clouds and setup UI params, draw stars, skysphere + sun, clouds and have everything blend in perfectly)		
Add time of day animation (moves sun, blends sky colours, changes base water colour)	Apr 6	
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)	Apr 7	This may turn out not to be very simple due to visual artifacts likely appearing if done incorrectly.
Implement Gerstner Waves for more complex swell modeling and animation.	Apr 7	This will need some tweaking of user settings to look better.

Dynamically scale the projected grid frustum, based on Gerstner settings to ensure the edges of the screen always touch water.	Apr 10	I already have this hard-coded, so this should be easy
Update the UI to include the ability to add/remove Gerstner waves with settings + add missing animation properties.	Apr 10	
Now that I have the generation of my dynamic skybox setup, I need to fix the global reflections that are still sampling from the default skybox. To do this, I just need to render to texture 6 times and then the sampling code is the exact same.	Apr 10	
Phong specular sunlight (of varying colour) and refine fresnel.	Apr 10	
Implement local reflections with terrain islands. Include UI slider for water surface distortion amount.	Apr 11	
Implement refractions	Apr 11	

Measure any impact that scaling the projected grid resolution has on runtime performance	Apr 11	
Implement sea foam, play with exaggerated foam for shearing along shorelines.	Apr 11	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 12	
Optimize	Apr 13	It may prove difficult to balance visual fidelity and the scale of the simulation
Add shadow mapping	<OPTIONAL>	
Add a boat with camera that you can move around in	<OPTIONAL>	
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.

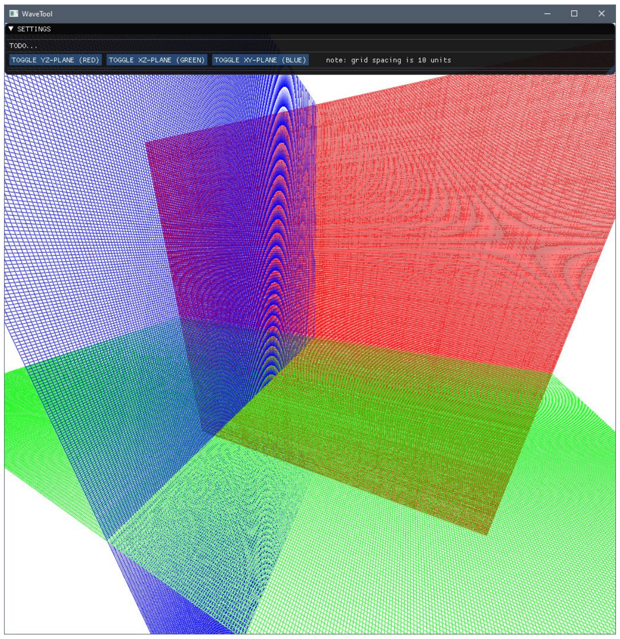
- ***SUMMARY:***

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.

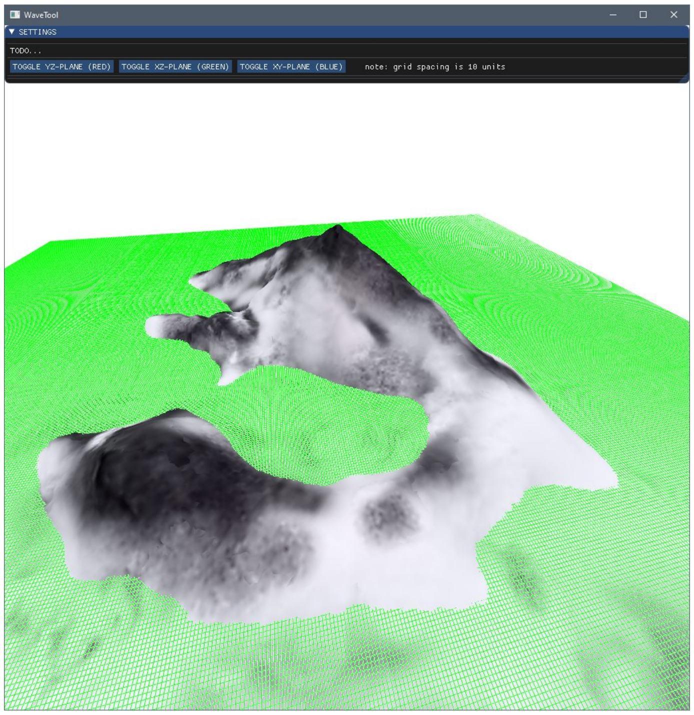
First off, my application can render the 3 Cartesian world planes for debugging purposes (each is either solid R, G or B), but the XZ-plane (green) is particularly useful at this stage for visualizing the water displacement that will be described below shortly.



TODO...

note: grid spacing is 10 units

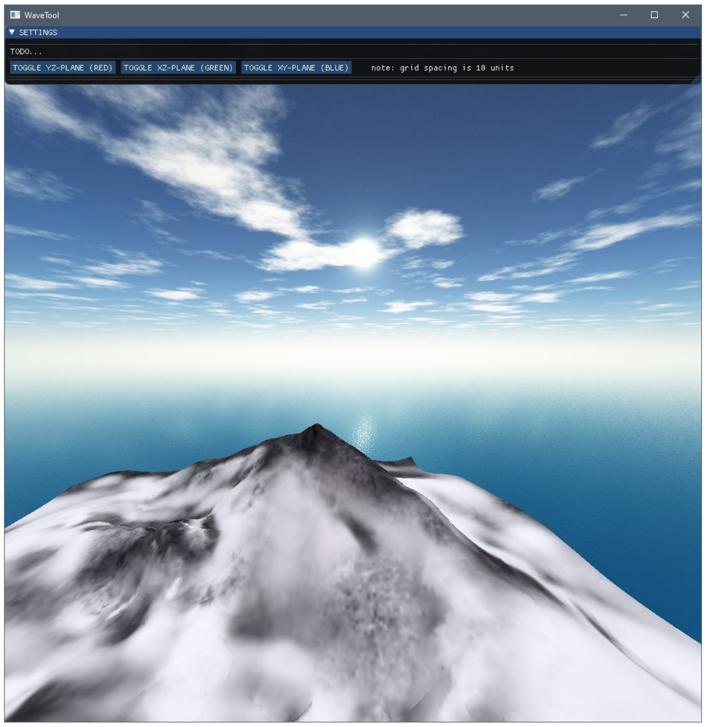
Second, a Mount Everest mesh gets loaded in to serve as some terrain (notably ocean floor) which I will use in the future to test local reflections and refractions.



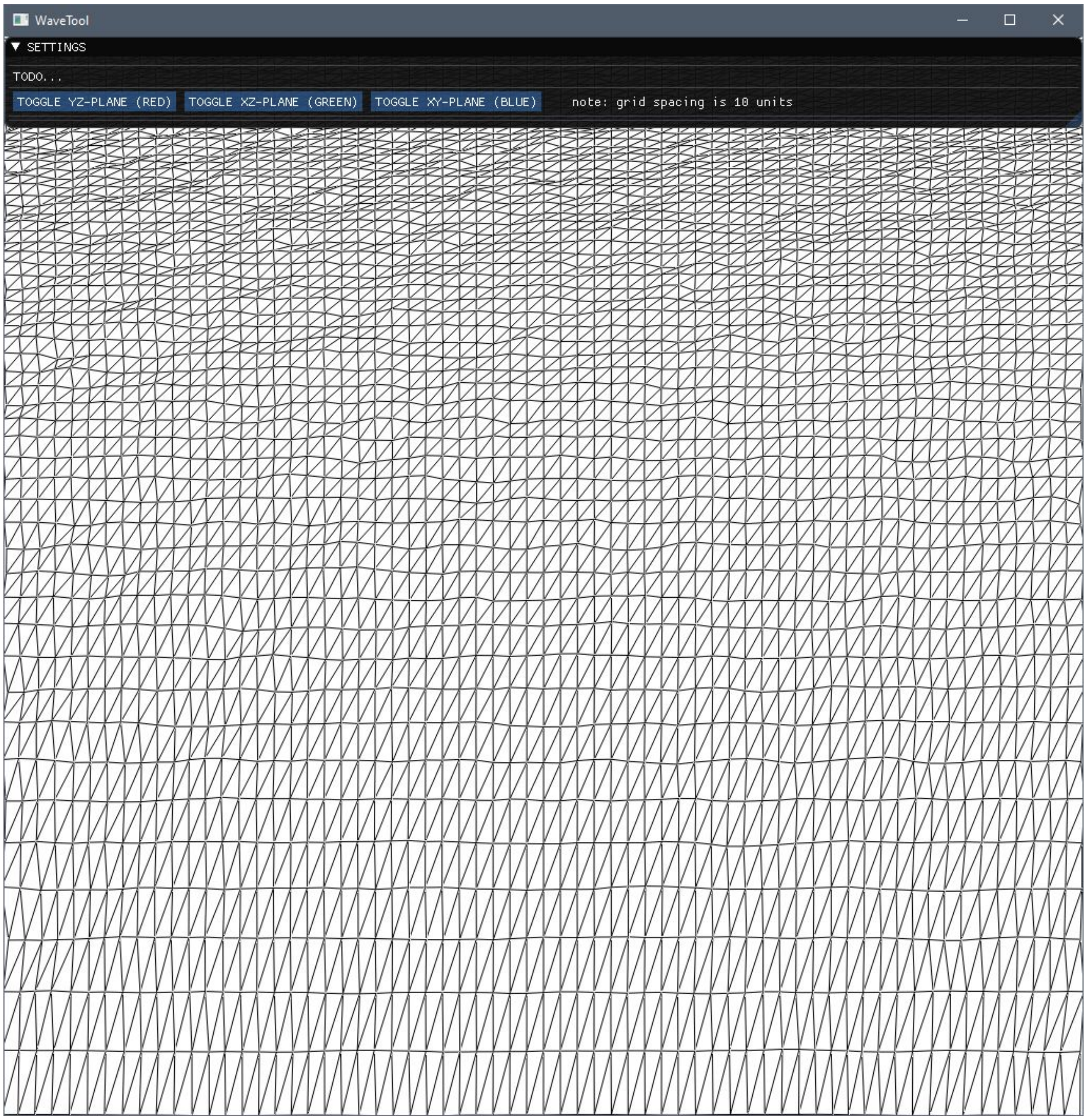
TODO...

note: grid spacing is 10 units

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.

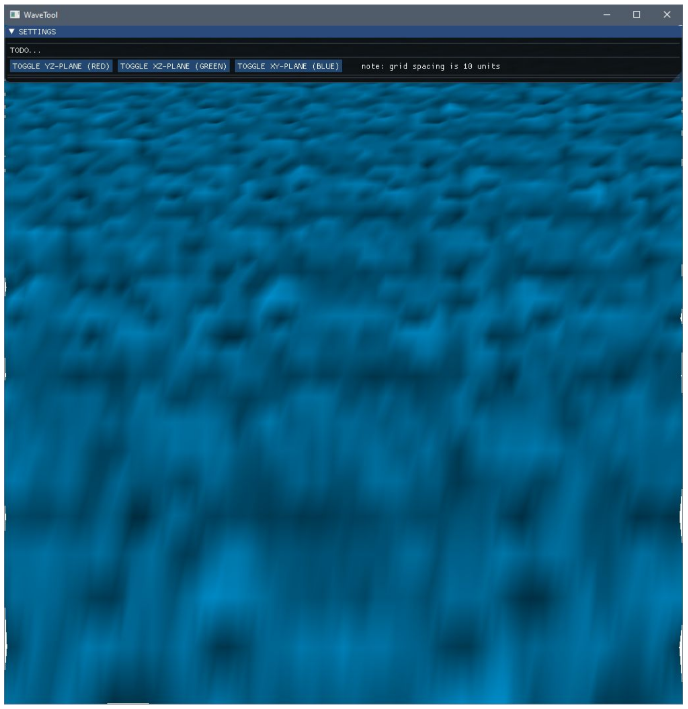


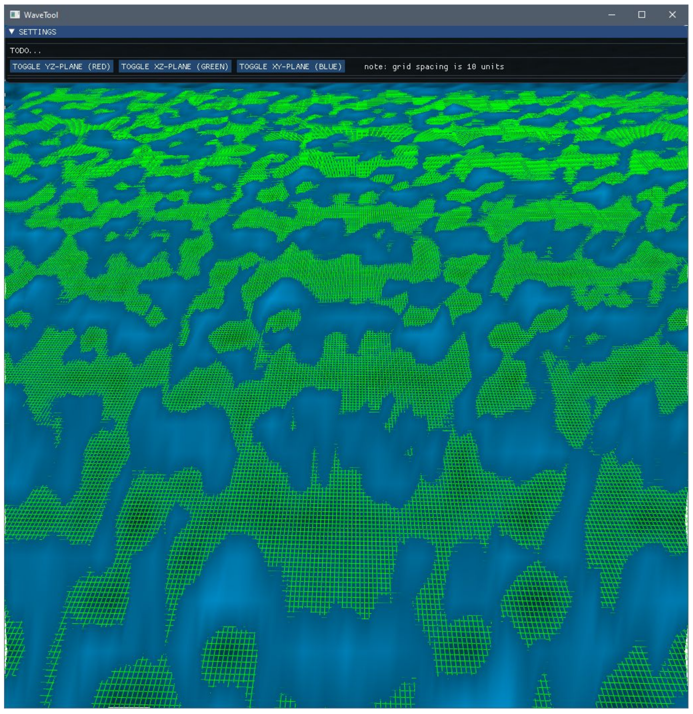
Next, I moved onto implementing the water grid using the projected grid method. This has been a little tricky to implement and currently I am able to render the grid (as seen below), but only when the camera frustum can entirely project onto the XZ-plane. This currently means that I disable rendering 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

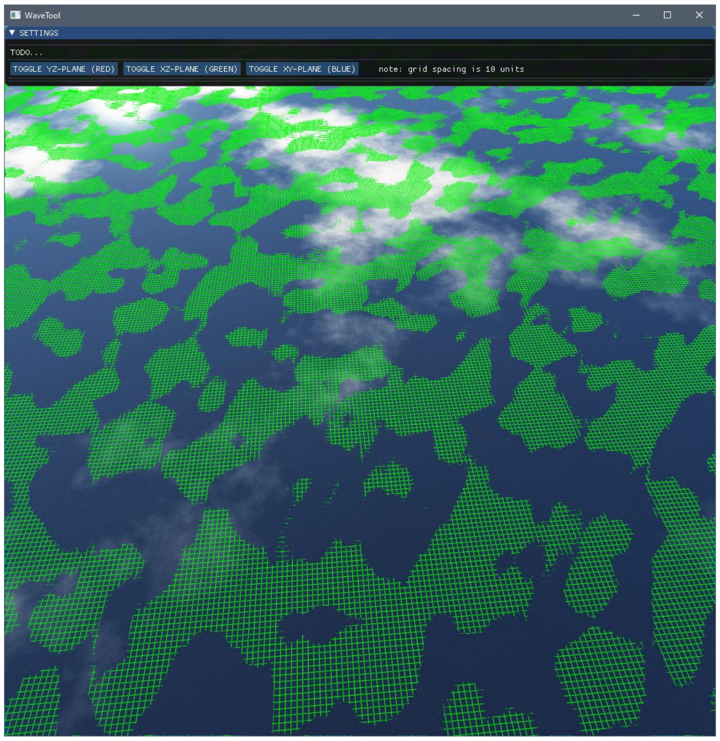
Here is the water heightmap rendered (with the noise texture also used as a multiplier for a demonstration blue colour). There are gaps on the sides which can later be fixed by scaling up the grid slightly.



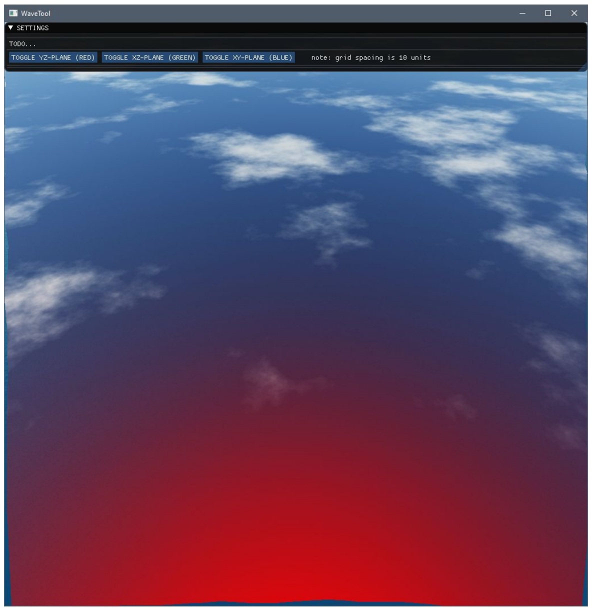


NOTE: the green XZ-plane is useful to show the water heights

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.



- ***MORE RESULTS:***

I finished implementing the projected grid concept, with some slight bugs to fix. In report #1, there was the restriction that the grid only showed up when the camera was pointing downwards. Now, the camera is seperate from the projector and is free to look anywhere (even below the water). The displaceable volume is set up to have an amplitude capable of holding all the Gerstner Waves and some extra sine wave displacement. There is also some hard-coded scaling of the sampling frustum size to ensure there are no gaps on the sides of the screen with no water.

Per-vertex normals are now computed across the water grid. Thus, the surface is no longer a flat mirror with no character. At the moment, the global reflection colour component is incorrect since it's being sampled from the base skybox. I decided that a good skybox is necessary to set the atmosphere for the scene, so I set up systems to dynamically generate one based on time of day, along with UI parameter tweaking. I was searching around for good free license-friendly skyboxes and I came across this tool that generates space skyboxes. It can be found at the following link.
https://github.com/wwwtyro/space-3d

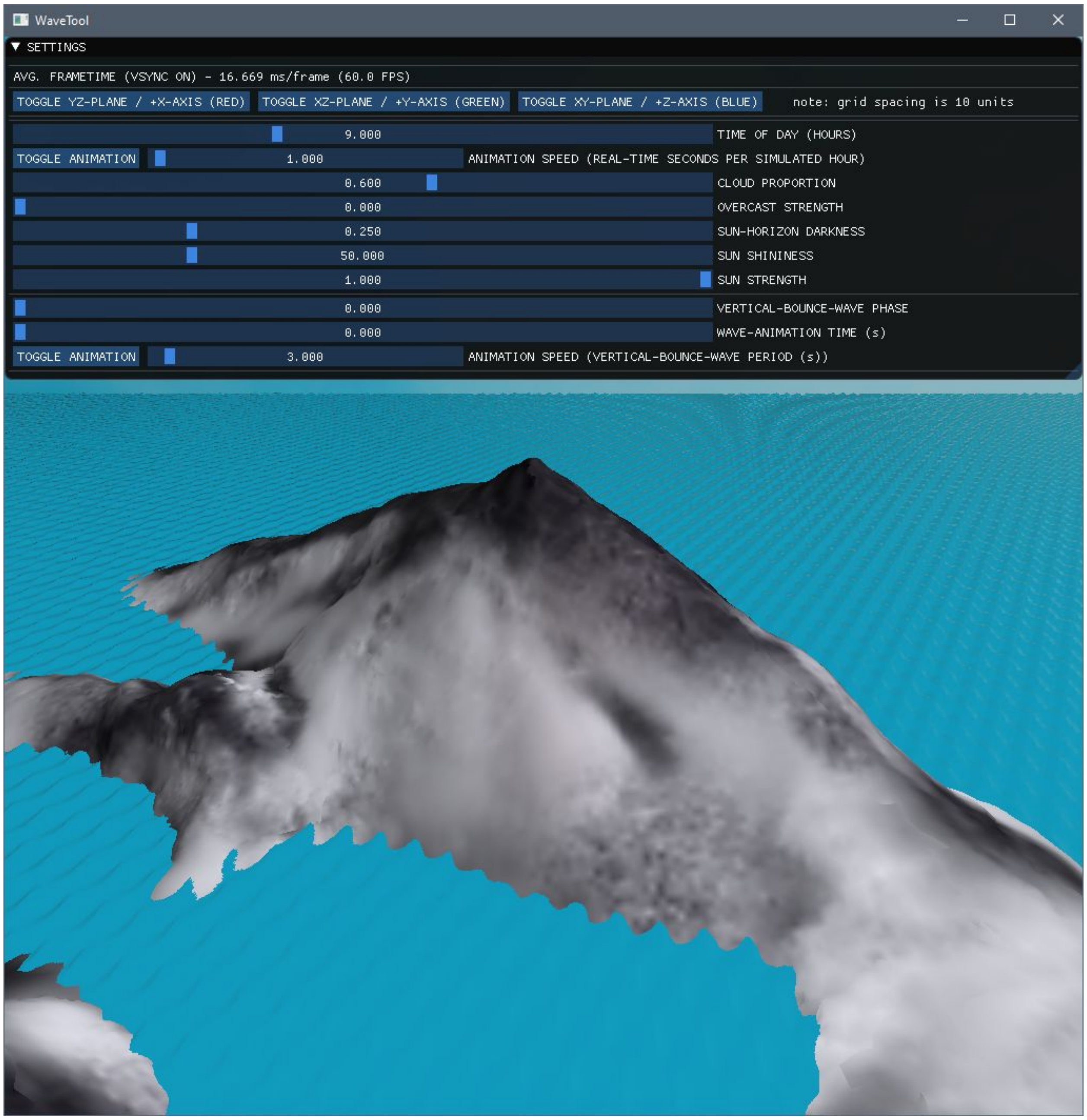
I exported two skyboxes of the same seed. The first I denote as the "cloud skybox" that has only nebulae. The second is the "star skybox" with only stars. My dynamic sky has three main parts that get alpha blended in layers. The star skybox is first rendered fully opaque.

On top of that, I next draw on the "skysphere" which geometrically is the inside of a unit radius uv-sphere I exported from Blender. I painted a 1D texture gradient of sky colours in Inkscape, that I sample by time of day to set the colours of the "peak" of the atmosphere (near the sun) and the "sun-horizon" (atmosphere orthogonal to sun direction). These two colours are then interpolated to shade the whole sphere. I also add on phong shading which results in drawing a nice sun. There are several issues here though. Notably, due to the geometry of the uv-sphere banding, the sun changes size drastically with altitude changes. It also appears more diamond-shaped sometimes rather than circular, which could be due to the resolution of the sphere. I also vary the skysphere alpha based on time of day in order to reveal the background stars at night time.

Finally, the third layer is the cloud skybox. From the imported nebulae skybox, I first apply a grayscale calculation to it to convert it into intensity patches. I have a UI parameter that can be set to specify the proportion of clouds to draw (intensities above this threshold get drawn with proper shading). There is also an overcast strength parameter that increases the grayness of both the clouds and the intermediate sky.

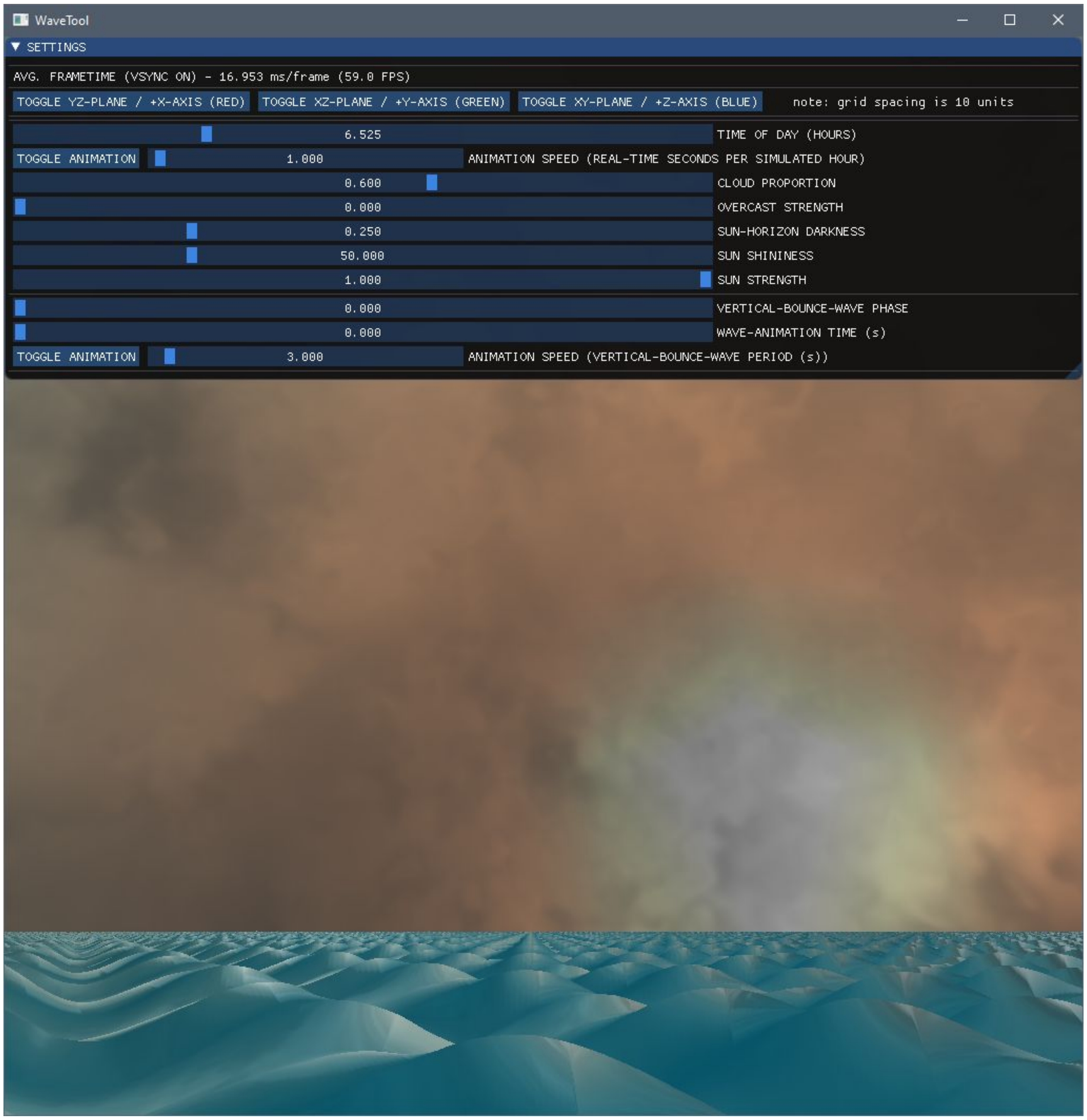
Putting everything together, you can simulate an animation of the sun moving midnight to midnight (rising in the east), along with the water colour changing as well.

I'm currently in the process of refining the wave animation system. I implemented Gerstner Waves to act as the foundational waves. The existing heightmap now acts as a small-scale displacement map to give more texture to the surface. There is also a sine wave that bounces the whole surface up and down to add some more visual character.


WaveTool
▼ SETTINGS
TOGGLE YZ-PLANE / +X-AXIS (RED)
TOGGLE XZ-PLANE / +Y-AXIS (GREEN)
TOGGLE XY-PLANE / +Z-AXIS (BLUE)
TOGGLE ANIMATION
TOGGLE ANIMATION

```
AVG. FRAMETIME (VSYNC ON) - 16.669 ms/frame (60.0 FPS)
note: grid spacing is 10 units
9.000    TIME OF DAY (HOURS)
1.000    ANIMATION SPEED (REAL-TIME SECONDS PER SIMULATED HOUR)
0.600    CLOUD PROPORTION
0.000    OVERCAST STRENGTH
0.250    SUN-HORIZON DARKNESS
50.000   SUN SHININESS
1.000    SUN STRENGTH
0.000    VERTICAL-BOUNCE-WAVE PHASE
0.000    WAVE-ANIMATION TIME (s)
3.000    ANIMATION SPEED (VERTICAL-BOUNCE-WAVE PERIOD (s))
```

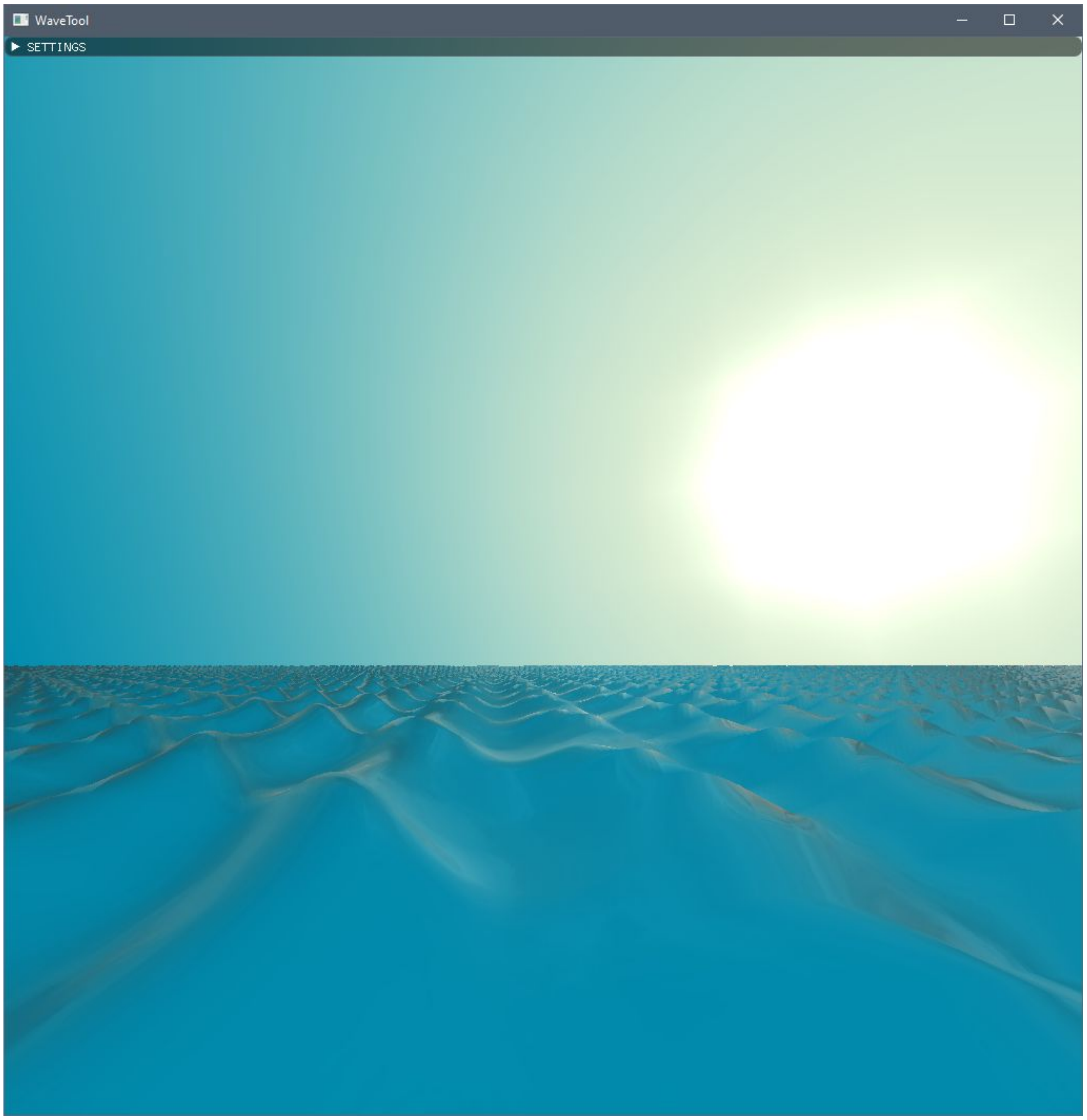
UI WINDOW



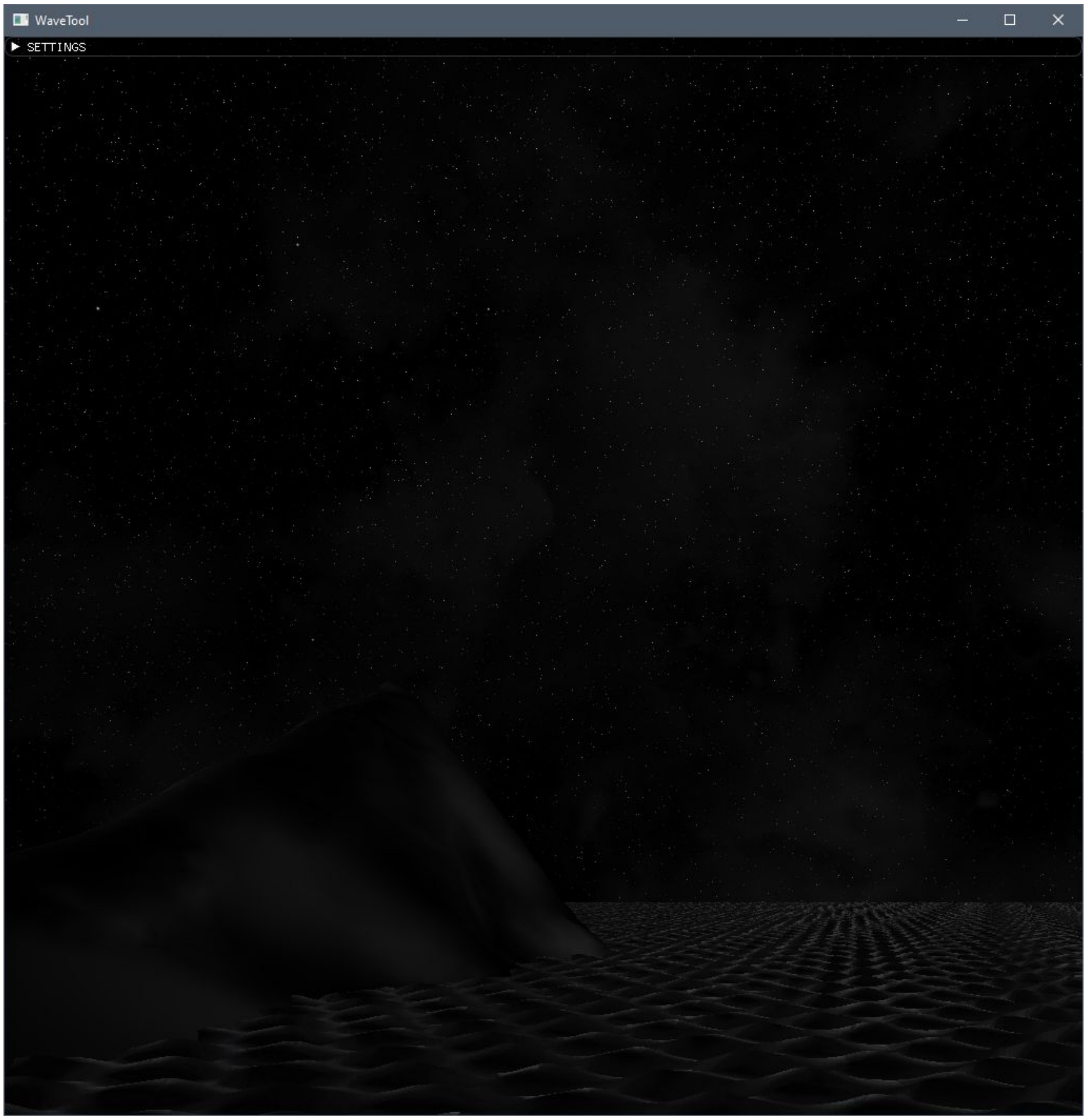
```
AVG. FRAMETIME (VSYNC ON) - 16.953 ms/frame (59.0 FPS)
TOGGLE YZ-PLANE / +X-AXIS (RED)   TOGGLE XZ-PLANE / +Y-AXIS (GREEN)   TOGGLE XY-PLANE / +Z-AXIS (BLUE)    note: grid spacing is 10 units

                  6.525                  TIME OF DAY (HOURS)
TOGGLE ANIMATION  1.000                  ANIMATION SPEED (REAL-TIME SECONDS PER SIMULATED HOUR)
                  0.600                  CLOUD PROPORTION
                  0.000                  OVERCAST STRENGTH
                  0.250                  SUN-HORIZON DARKNESS
                  50.000                 SUN SHININESS
                  1.000                  SUN STRENGTH
                  0.000                  VERTICAL-BOUNCE-WAVE PHASE
                  0.000                  WAVE-ANIMATION TIME (s)
TOGGLE ANIMATION  3.000                  ANIMATION SPEED (VERTICAL-BOUNCE-WAVE PERIOD (s))
```

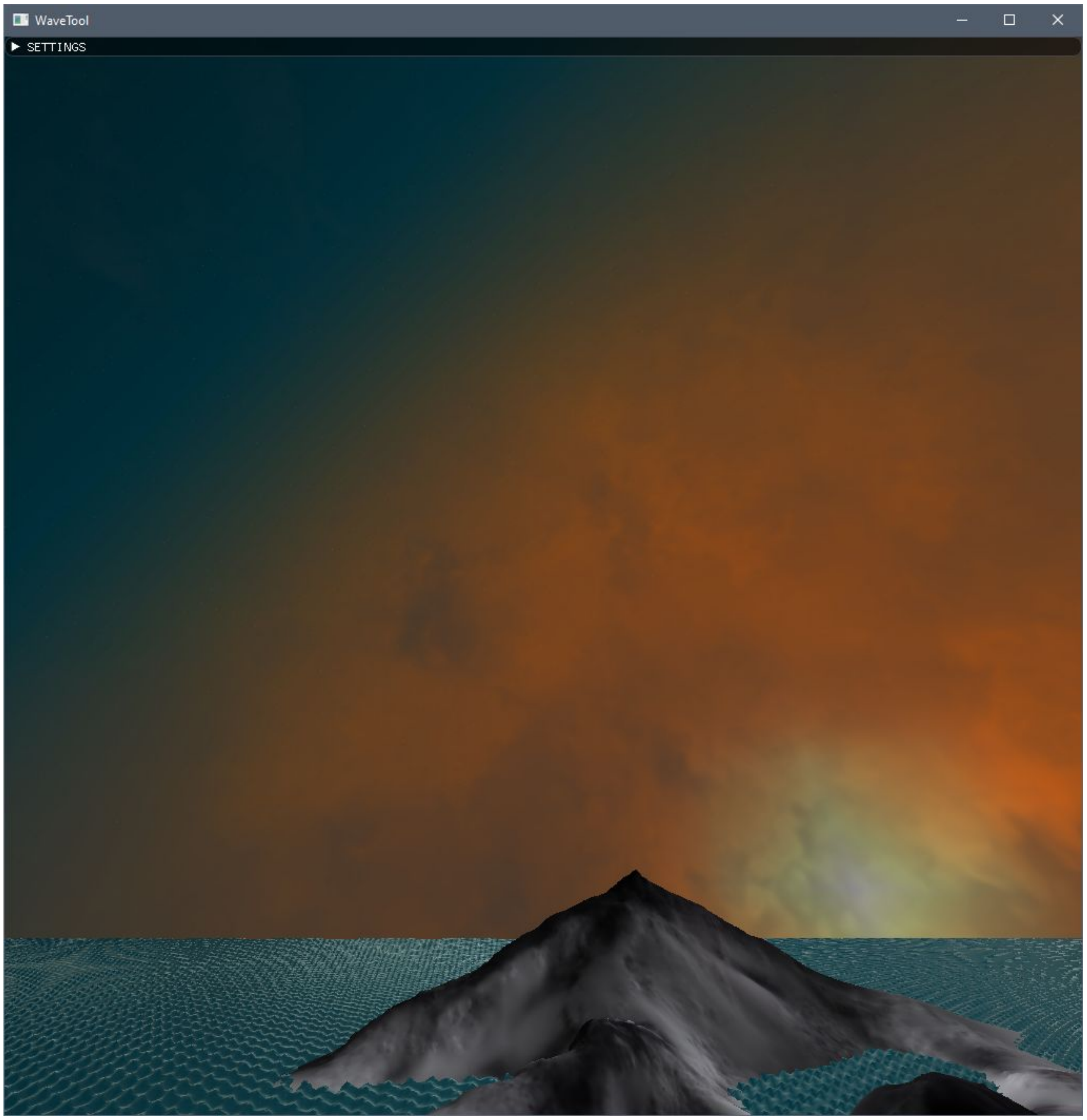
SUNRISE



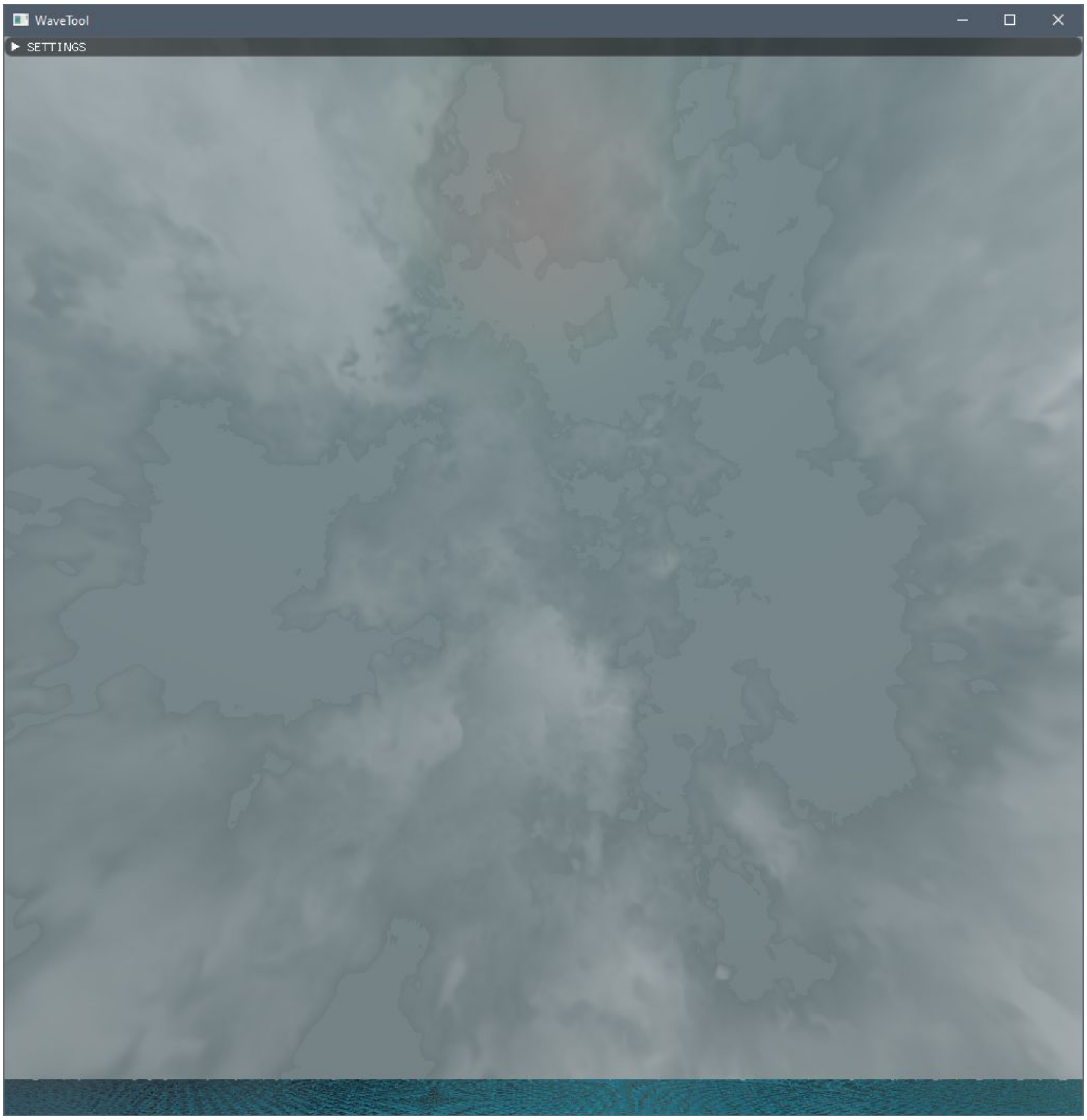
SUN RISING WITH CLEAR SKY



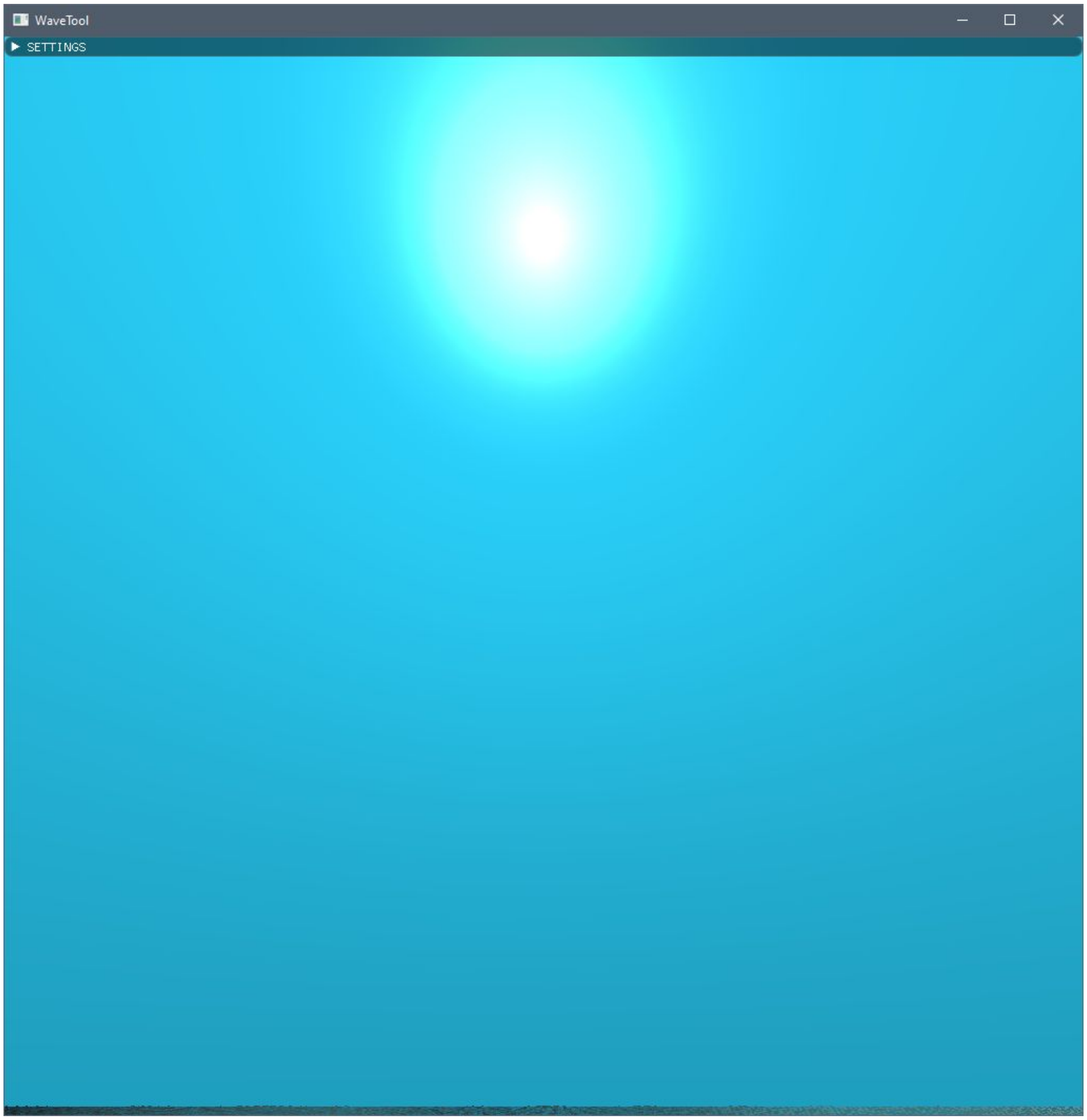
STARS AT MIDNIGHT



SUNRISE FROM AFAR



AN OVERCAST SKY AT NOON WITH MEDIUM CLOUD PROPORTION



AN CLEAR SKY AT NOON (SUN HIGHLIGHT)

- ***<u>REFERENCES:</u>***

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

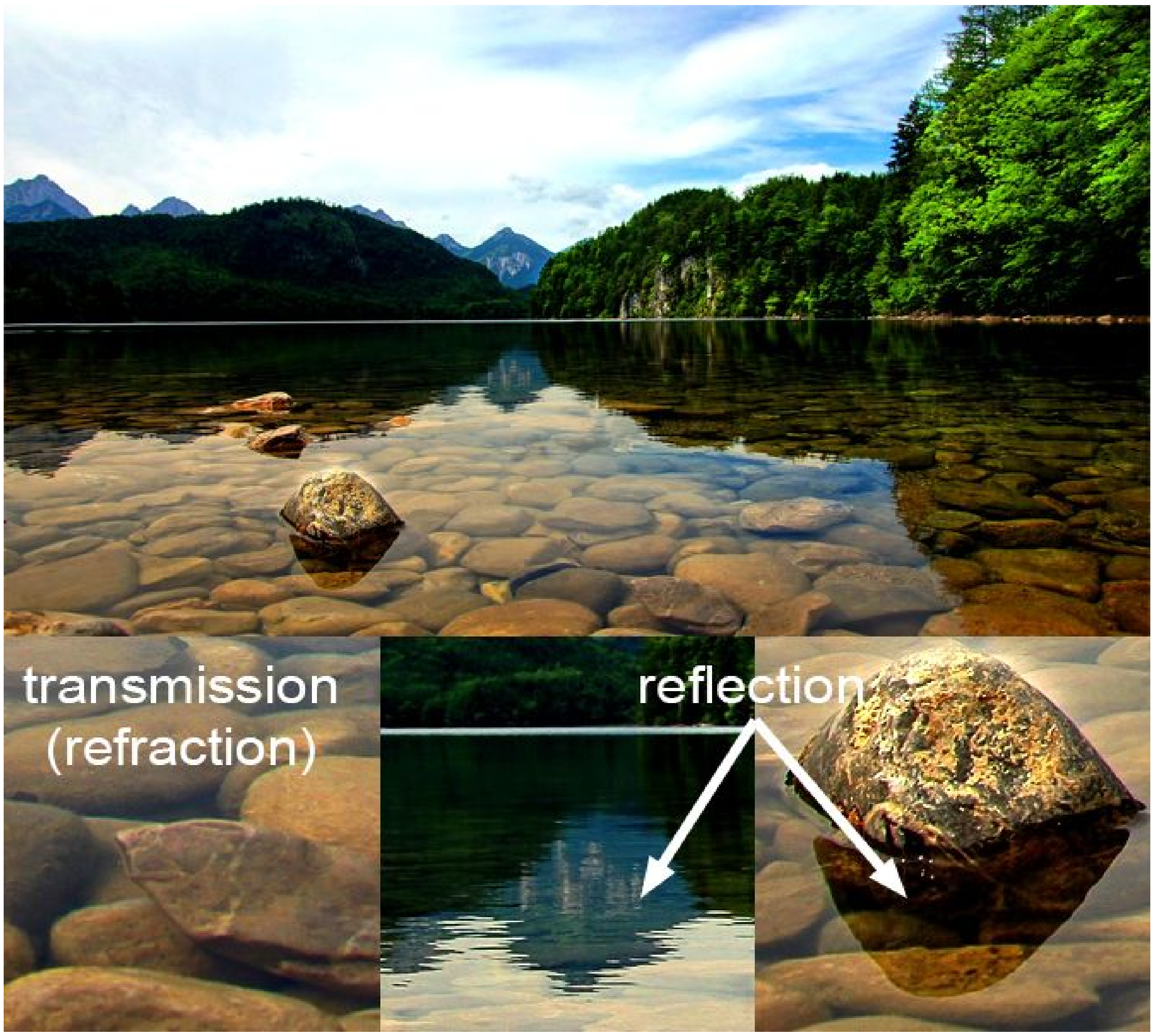
Mark Finch, Cyan Worlds, “Chapter 1. Effective Water Simulation from Physical Models”.
Nvidia GPU Gems. 5th Printing, September, 2007.
https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-1-effective-water-simulation-physical-models

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

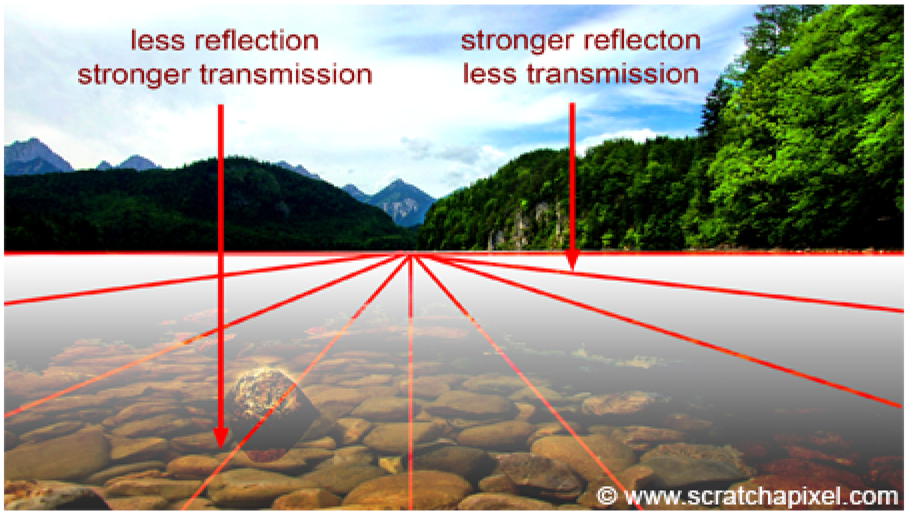
Shunli Wang, Fengju Kang, “Automatic Optimization for Large-Scale Real-Time Coastal Water Simulation”. Mathematical Problems in Engineering. 2016.
https://www.hindawi.com/journals/mpe/2016/9034649/

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

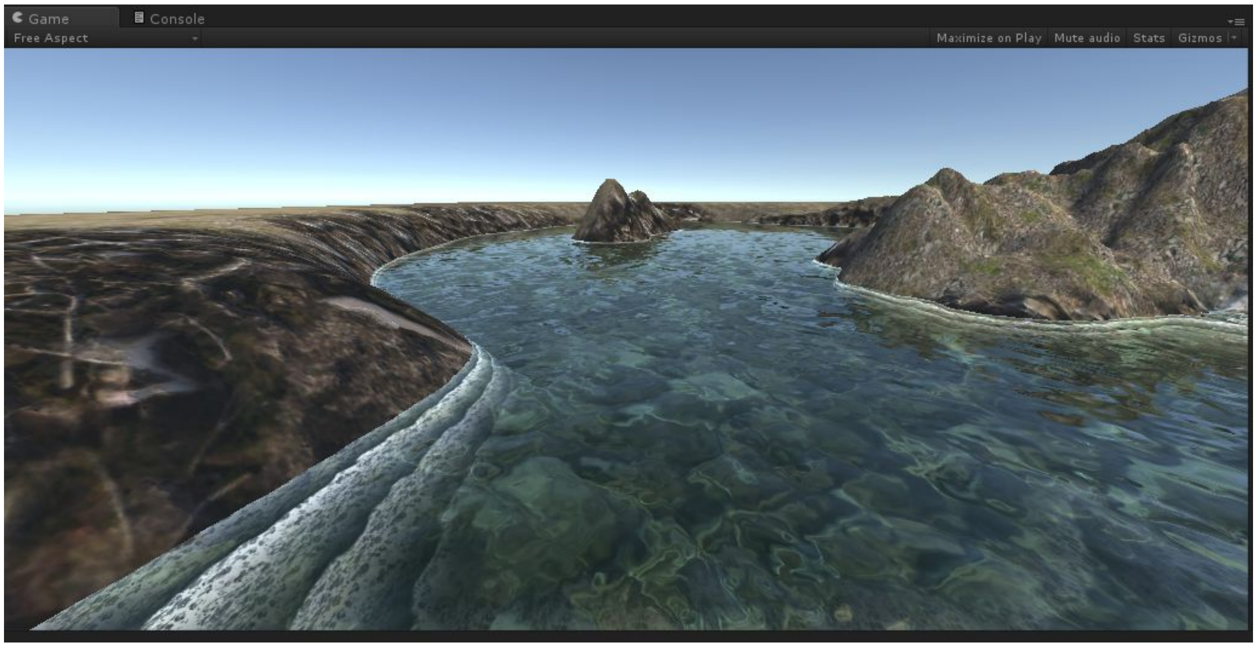
- ***IMAGES:***



reflection/refraction



Fresnel reflectance



An example of shoreline white water / foam

Sunlight beam and small island features

Foam