Water Caustics and Water Simulation

Linus Jacobson
Lund's University
Faculty of Engineering
Sweden
Email: li7414ja-s@student.lu.se

Arne Stenkrona
Lund's University
Faculty of Engineering
Sweden
Email: ar8058st-s@student.lu.se

***Abstract*—This paper describes an implementation of an approximate simulation of water caustics and water simulation based on an article by Martin Renou [2]. By generating a caustics light map that can later be applied to underwater geometry, caustics light-effect can be generated. The simulated solution worked well but the shadow caused by objects underwater were somewhat immersion breaking as the moving water didn't effect the shadows of said object.**

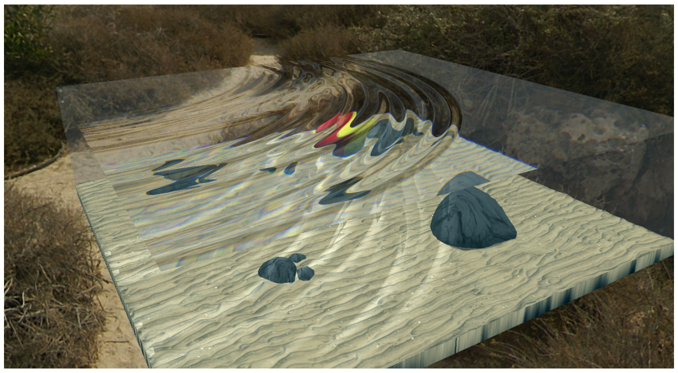

Fig. 1. Render of the scene with caustics.

I. Introduction

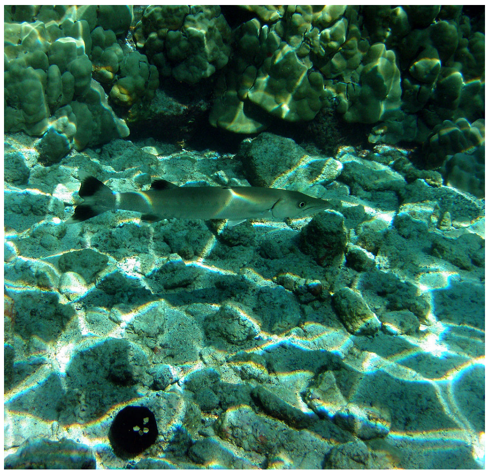

Fig. 2. Water caustics in real life. [5]

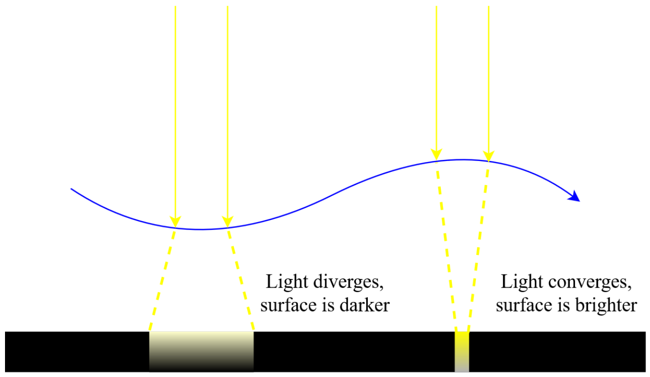
Light diverges, surface is darker
Light converges, surface is brighter

Fig. 3. Water caustics are caused by convex areas of the water surface converging light, and concave areas of the water surface diverging light.

A. Caustics

When light passes through a refractive surface with varying geometry the surface will act as a lens. At convex areas of the surface light will converge and whatever surface lies beyond will receive a concentrated amount of light and will appear brighter. The converse is also true, where a concave area will cause divergent light and darken the surface beyond [1]. This is particularly noticeable in water, as can be seen in figures 2 and 3

B. Simulating Caustics

In order to increase the level of realism in a 3D scene it is useful to reproduce optical phenomena that occur in real life. Simulating accurate water caustics can be tricky, however, especially in real time graphics. Due to the cost of calculating photons leaving a light source, scattering through water, oftentimes good looking but not accurate approximations are used instead [3]. A simple approximation could be sampling from an predefined water caustics animation texture, but that can be immersion breaking as the way the water moves doesn't affect the caustics.

This paper goes through an in-between of a predefined water caustics animation, and real photon simulation; sacrificing accuracy for performance, but maintaining immersion. On top of that it goes through interactive water simulation, creating waves by clicking on the surface. This implementation is based on a methoud outlined by Martin Renou [2].

II. Algorithms and Pipeline

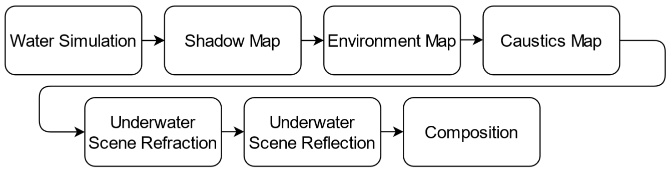

Fig. 4. Overview of the pipeline stages to render simulation.

In figure 4 we can see the pipeline stages. The frame begins by updating the water simulation which is stored in a texture. This is followed by a shadow map pass for all underwater geometry. The environment map and caustic maps are both crucial in order to apply the caustic effect, as we will see. Once the shadow map and caustic map is prepared we render the underwater scene into a refraction texture along with its reflection into a reflection texture. The final composition stage renders the underwater scene again, along with water which uses the underwater refraction and reflection textures to compute its refractive and reflective colours.

A. Water Simulation

Our water simulation is based on the method used by Martin Renou in [2]. We simulate water by using a 2D texture with four 32-bit floating point colour channels (rgba). Each pixel describes the state of water at that point $(u, v),\ u, v \in [0, 1]$ at time $t_k,\ k \in \mathbb{N}$ with the r-channel representing water height $h_{u,v,k}$, g representing water velocity magnitude $w_{u,v,k}$, and b and a representing the x and z component of the surface normal $n_{u,v,k}$. The y component of the normal is only stored conceptually as it is easily reconstructed from its x and z values. Initially we set

$$\begin{aligned} h_{u,v,0} &= 0 \\ v_{u,v,0} &= 0 \\ n_{u,v,0} &= (0, 0, 0) \end{aligned}$$

In each time-step $t_k,\ k > 0$ we update as follows:

$$\begin{aligned} v_{u,v,k} &= \lambda \cdot (v_{u,v,k-1} + 2 \cdot (\text{mean}_4(v_{u,v,k-1}) - h_{u,v,k-1})) \\ h_{u,v,k} &= h_{u,v,k-1} + v_{u,v,k} \end{aligned}$$

where λ is the attenuation and

$$\begin{aligned} \text{mean}_4(v_{u,v,k-1}) = \frac{1}{4}(&v_{u+dx,v,k-1} + v_{u,v+dy,k-1} + \\ &v_{u-dx,v,k-1} + v_{u,v-dy,k-1}) \end{aligned}$$

is the mean of 4 height samples sampled at $(u \pm dx, v \pm dy)$ for some $(dx, dy) \in \mathbb{R}^2$. The normal n_x is set to the normal of the triangle defined by $h_{u,v,k}$, $h_{u+dx,v,k-1}$, and $h_{u,v+dy,k-1}$.

Water droplets are added to the texture upon user interaction by adding

$$\frac{s}{2}(1 - \cos(\pi \max(0, 1.0 - \frac{\text{dist}}{r})))$$

to the height of each pixel, where s is the droplet strength, dist is the distance between the pixel and droplet center, and r is the droplet radius.

B. Shadow Map

Creating shadows using shadow maps generated from the perspective of the light, using an orthogonal perspective matrix, shadows can be emulated. The shadow map is generated by sampling the depth values from the light PoV, resulting in a image with output range of [0,1], where 0 is the z-nearplane and 1 is the z-farplane. This texture is later then passed as a uniform to be sampled in the fragment shader whilst rendering the final scene. Said pixel in the fragment shader is then transformed to world coordinates to later be compared with the shadow map to see if it is in direct sunlight or not, and then shaded accordingly [4].

C. Water Caustics

Just like for the shadow map, a caustics map is generated which then is applied to the subsurface objects that face the light. In order to compute the caustics map, the height and normal map of the water is needed, as well as an environment map of the subsurface area from the light's PoV. The implementation is based on *Real-time rendering of water caustics* by Martin Renou [2]. We've divided this into two steps in our pipeline.

1) Environment map: In order to compute caustics we need to intersect light rays with the underwater scene. For performance, we delegate this to the GPU. We prepare a texture containing the world space coordinates of the scene rendered from the lights point of view. We also store the light space depth position in the alpha channel. The resulting texture is our environment map.

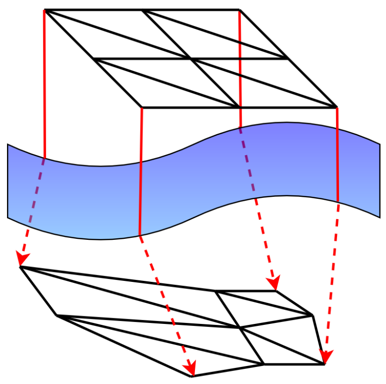

Fig. 5. The vertex shader refracts triangles through the water resulting in some triangles increasing in area and others decreasing in area. The area change tells us how the light intensity is affected by caustics.

2) Caustics map: We simulate a wave-front of light passing throw the water in a vertex shader. We feed in each vertex from the water mesh, compute the refracted light ray from the sun and intersect it with the environment map. This can be done by tracing the ray in increments, comparing the environment map depth value with the rays current distance from the light source in order to determine intersection. This will transform the triangles and either increase or decrease their respective area. An increase in area tells us that light diverged and we should not expect an increase in light from caustics. A decrease in area tells us that the light converged. This is illustrated in figure 5. In a fragment shader we colour in bright values for converging light and dark values for diverging light. The resulting image is a light-space representation of caustic light contribution to the scene. This can be sampled when rendering the underwater scene by computing each fragment's corresponding light space position.

D. *Underwater Scene Refraction*

We render the underwater scene using regular phong lighting. For shadows we sample from the shadow map using each fragments light space position as sample coordinates. Similarly we sample from the caustic map in order to determine caustic light contribution. This render pass is stored in the underwater scene refraction texture.

E. *Underwater Scene Reflection*

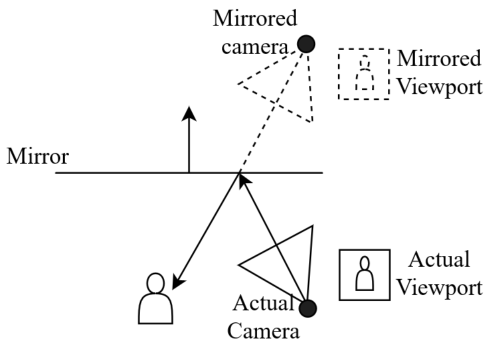

Fig. 6. A mirrored camera reveals what a regular camera sees in a reflection.

In order to render reflections we mirror the camera about the water plane and render the subsurface scene into a texture, as seen in figure 6. This render pass is stored in the underwater scene reflection texture.

F. *Composition*

We begin by once again rendering the underwater scene as we did when generating the underwater scene refraction texture. Next, we render the water surface, computing the amount of refraction and reflection, each of which will contribute to the waters final colour.

In order to determine the refraction colour we will compute refraction vectors which will be used to determine how to sample from the underwater scene refraction texture. The refraction vector is calculated from the light incident vector and the water normal three times with slightly different refractive indices. To sample from the texture, we use the three refraction

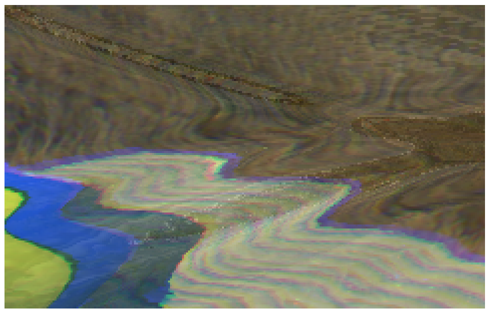

Fig. 7. The blue, yellow, and red edges highlight the chromatic aberration effect.

vectors. We estimate the vector's collision with the scene by assuming that all geometry is of a constant distance from the water surface and transform it to camera clip space and in order to sample the subsurface texture. While the sampling is physically incorrect it is convincing enough. The three colour values are blended together to form the refracted rays colour contribution with a chromatic aberration effect, showcased in figure 7.

To determine the reflective colour we will read from the underwater scene reflection texture. The view-port coordinates are used as sample coordinates, with a heuristic offset based on the simulated water normal in order to provide convincing distortion to the reflection. Again this is incorrect, but convincing enough.

Refraction and reflection colours are blended together to produce the water surface final colour.

III. Result

A. *Caustics and Shadows*

When we add movement to the water, caustics patterns begin to appear on the subsurface geometry. Rings can be seen on the water, which can also be found in the caustics pattern on the water floor (see figure 8).

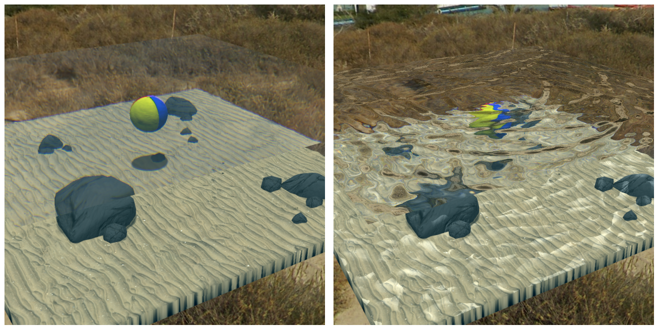

Fig. 8. Left: Waves are flat — Right : Waves move and caustic light effects appear on subsurface geometry.

However, there are issues with rendering the shadow map separately using just a depth map, and it is apparent when there is movement on the water and we watch the shadow generated by the beach ball (see figure 9. It is not effected by the waves.)

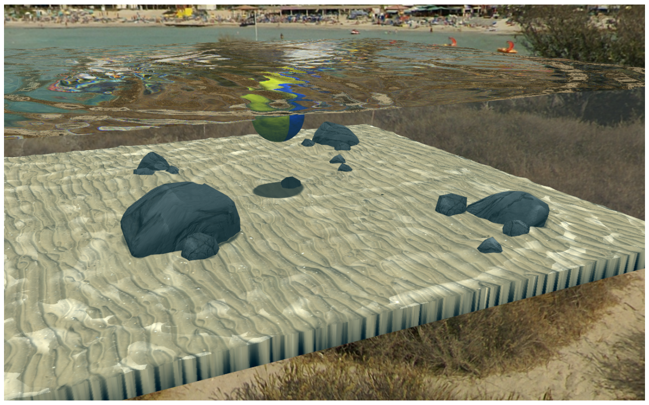

Fig. 9. Shadows rendered under the surface

B. *Reflection and Refraction*

The water rendered from below can be seen in figure 10. Interesting to notice that the default case is total reflection when the water is still. If we add waves we start to see refracted rays (i.e. see the sky). This is due to rendering from a water medium and rendering air, whilst in figure 8 we rendered from air into water.

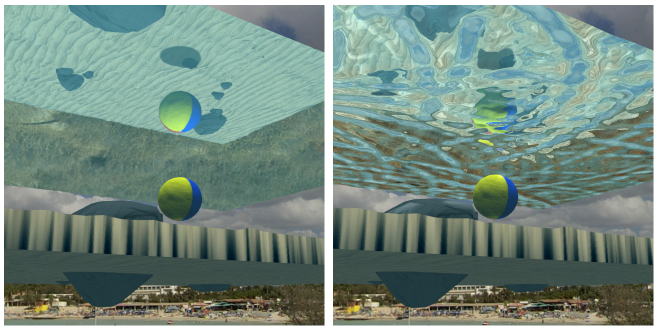

Fig. 10. Left: Waves are flat — Right : Waves move and the reflection is distorted.

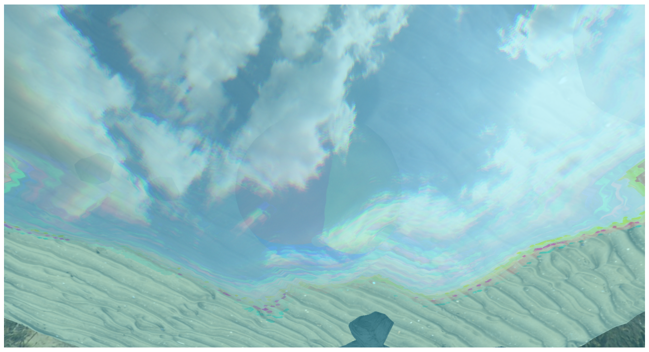

Fig. 11. Shadows rendered under the surface

In figure 11 we can see the refracted sky, and the line of total reflection happening. Notice that there is some reflection in mostly refracted area. You can see the ball being reflected.

IV. Discussion

The result was quite expected, and is quite pleasing to play with. Since the shadows doesn't move underwater, implementing some sort of distortion on the shadow map based on the shape of the water would increase the believably of the simulation. An ray tracer could be developed in parallel to compare an accurate simulation versus the approximation.

The reflection and refraction colours of the water are currently based on a geometry agnostic heuristic. This give particularly noticeable deviation from a physically correct model when geometry intersects the water surface, which is why our scene avoids such cases. The sampling reflection and refraction sampling could be improved by colliding the reflection and refraction rays with the actual geometry, using a similar method to that of the environment map when colliding light rays for caustic computation. Such a method would certainly prove more convincing for the more attentive user.

The simulation could be extended to work in an infinite world, using cascade shadow maps and caustic maps. Although parts of the pipeline already works with dynamic lights, allowing complete freedom would be another improvement.

There is also a lot of room for optimisation, such as calculating the shadow map and the environment map at the same time. And limiting the calculation to the view frustum so we don't spend precision on unseen targets. The underwater scene is also rendered twice, once for the refraction texture and once for the composition. One could simply copy the contents of the refraction texture along with its corresponding depth values into the composition frame buffer and depth buffer to improve frame rate.

References

[1] Pietro Ferraro, *What a caustic!* The Physics Teacher Vol. 34 (1996) p. 572

[2] Martin Renou, QuantStack, *Real-time rendering of water caustics* LINK, sourced 2020-12-16

[3] Juan Guardado (Nvidia) and Daniel Sánchez-Crespo (Universitat Pompeu Fabra/Novarama Technology). *GPU Gems, Chapter 2. Rendering Water Caustics*. LINK, sourced 2020-12-16

[4] Lunds university, *EDAN35 High Performance Computer Graphics. Assignment 2: Deferred Shading and Shadow Maps* LINK, sourced 2020-12-16

[5] Brocken Inaglory, *Caustics made by the surface of water* LINK sourced 2020-12-16