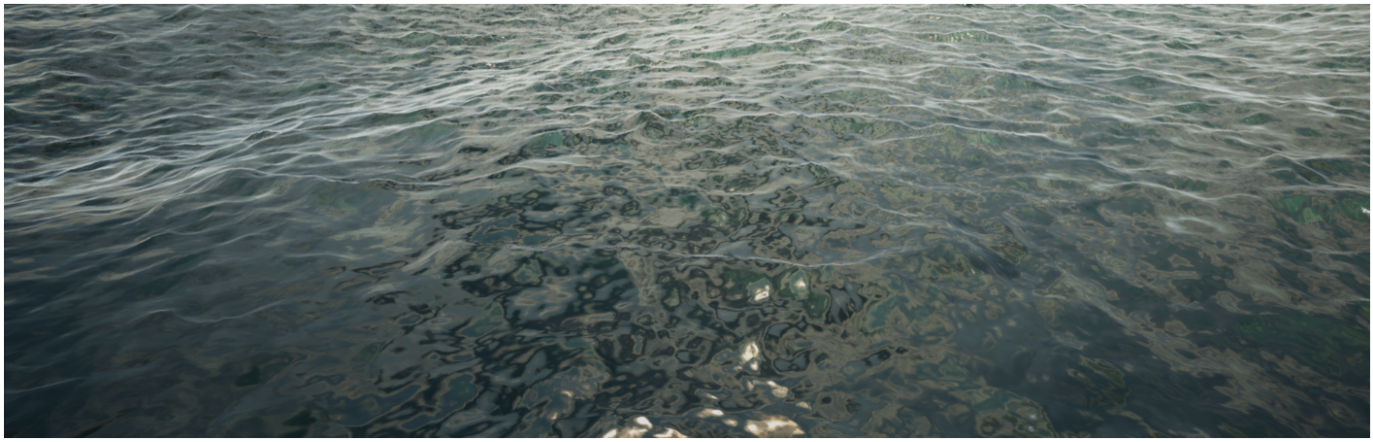
Water shading and geometric simulation in Computer Graphics

Laura Nunes Rodrigues
pg50542@alunos.uminho.pt



***Abstract*—Water is a fascinating element that plays a significant role in our natural environment. In computer graphics, accurately rendering water surfaces is essential for creating realistic virtual environments. Water shading and geometric simulation techniques have emerged as powerful tools to achieve visually convincing representations of water bodies. This essay explores the concepts, challenges, and advancements in water shading and geometric simulation in the field of computer graphics.**

***Index Terms*—water, shading, waves, simulation, reflection, refraction, foam, bubbles, caustics**

I. Introduction

Water has captivated human fascination for centuries. In the realm of computer graphics, achieving realistic and captivating water surfaces is a challenging yet essential goal. As stated in [1], "the surface of the sea is, by its very nature, a highly complex problem for computer graphics: it is a dynamic system, which excludes pre-computation". Water shading and geometric simulation techniques have emerged as powerful tools to create virtual water bodies, enabling the creation of immersive and visually convincing environments. These techniques encompass a range of principles and algorithms that try to accurately depict the appearance and dynamic behavior of water.

Water shading in computer graphics revolves around capturing the intricate interplay of light and water surfaces [1]. Reflections, refractions, transparency and surface roughness are some key elements that contribute to the visual fidelity of water rendering.

Geometric simulation, on the other hand, focuses on the dynamic behavior of water. Water is a dynamic element in nature, exhibiting a wide range of movements, from gentle ripples to powerful waves and turbulent flows. Geometric simulation techniques aim to model and simulate these dynamic behaviors by incorporating principles of fluid dynamics. By considering physical forces such as gravity, wind, and interactions with objects, these techniques enable the recreation of realistic water motion. The goal is to achieve visually compelling animations and interactions, seamlessly blending virtual water with the surrounding environment.

In addition to surface appearance and dynamic behavior, water shading techniques often encompass the simulation of secondary effects associated with water bodies. These effects, including foam [2], spray, splashes, and ripples [3], contribute to the overall realism of water rendering.

As proposed by [3], the following paper shall discuss key elements such as Ocean Radiosity, Surface Wave Optics, Water Volume Effects, Waves and Motion on the surface. The first three referring to water shading and some of its aspects such as reflections, refraction, etc. And the following regarding geometric simulation aspects such as wave creation and movement.

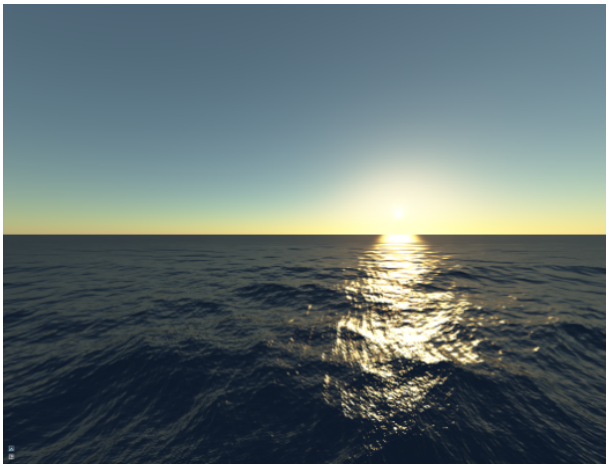


Fig. 1. Results obtained by [1]

II. Ocean Radiosity

Radiosity can be described as "a method of rendering based on an detailed analysis of light reflections off diffuse surfaces" [4], which means it can be used to accurately trace the flow of light on the water surface.

In [2], when trying to create a realistic ocean simulation, Carreiro highlights water **translucency**, "light that goes through it isn't blocked as easily as light through an opaque surface". The duality between light transmission and reflection is what causes our eyes to pick different tones and details during different times of the day when looking at the ocean. This is important to understand because only then can we aim toward the visual needs to accomplish a water shader with satisfactory fidelity.

The key components of the **ocean surface**'s illumination are listed in [3] as:

"In addition to specular reflection of direct sunlight and skylight from the surface, some fraction of the incident light is transmitted through the surface. Ultimately, a fraction of the transmitted light is scattered by the water volume back up through the interface and into the air."

By adding these components we can obtain, in a simplified version, the total light intensity or radiance at the surface:

$$L_{\text{ABOVE}} = rL_S + rL_A + t_U L_U \tag{1}$$

In this formula presented by [3], L_S is the direct light from the sun, L_A is the diffuse atmospheric skylight and L_U is the upwelling light (the light just below the surface that is transmitted through the surface into the air).

It should be noted that here are, although, a lot of factors that also affect this formula, such as reflectivity and transmissivity.

When it comes to **under the surface** radiosity, [3] also simplified the calculations by simply summing L_D (the "direct" light from the sun that penetrates into the water), L_I (the "indirect" light from the atmosphere that penetrates into the water), L_{SS} (the single-scattered light from both the sun and the atmosphere) and L_M (the multiply-scattered light).

$$L_{\text{BELOW}} = tL_D + tL_I + L_{\text{SS}} + L_M \tag{2}$$

On bigger waves, the reflection or transmission of light in-between surfaces can also be considered.

III. Surface Wave Optics

Water surfaces reflect their surroundings, including the sky, nearby objects and the overall environment. Accurately simulating reflections is essential for achieving a convincing water shading effect.

According to [3], the ocean surface is a near perfect specular reflector, with well-understood reflectivity and transmissivity functions.

When light rays hit the water surface, they are usually split into two components: a transmitted ray continuing underwater at a refracted angle and a reflected ray. Both of these have diminished intensity.

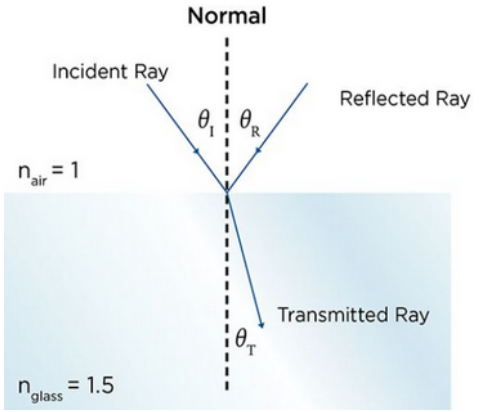


Fig. 2. Light ray behaviour on the surface [2]

When it comes to Ray Reflection, Specular and Diffuse Reflection should be considered. According to [5], specular reflection happens on "smooth surfaces such as mirrors or a calm body of water". When in rougher surfaces, usually diffuse reflection takes place.

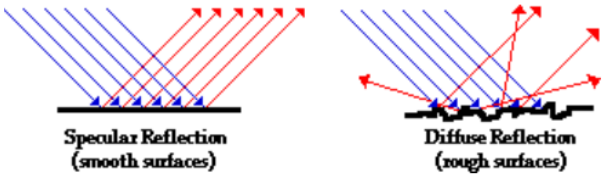


Fig. 3. Light reflection [5]

To simplify, many of the articles that were analysed only took into account specular reflection and ignored roughening effects.

A. Specular Reflection

As explained in physics, in a perfectly smooth surface, when a specular reflection occurs the incident ray and the reflected ray have the same angle with respect to the surface's normal, as shown in figure 2 ($\theta_I = \theta_R$).

B. Transmission or Refraction

When light passes air and goes through the surface of water, it slows down, causing it to change direction slightly. This change of direction is called refraction. As light enters a more dense substance (higher refractive index), the refracted ray goes more towards the surface's normal.

Refraction is the bending of light rays which can cause the distortion of viewed objects beneath the water. Accurately simulating refraction is crucial for achieving the transparent and distorted appearance of water. Refraction algorithms consider the refractive index of water and calculate the distortion of light as it passes through the water surface.

When creating water shaders though, solutions found also take into consideration these components to get realistic light rendering:

C. Fresnel Reflectivity and Transmissivity

In [3] is stated that the reflectivity R and transmissivity T are "related by the constraint that no light is lost at the interface". From this is taken the following equation:

$$R + T = 1 \tag{3}$$

"The Fresnel effect is what causes the amount of light reflected to depend on the viewing angle" [6].

This effect works so that light reflects best at a lower angle of incidence on a water surface. And the more the surface reflects light, the less we are able to see through it.

Because of this another aspect arrises: **Water color** (as mentioned by [2]).

D. Water color

Water should be darker close to the camera and a little brighter farther from it. This way, the Fresnel effect will work better, also creating a realistic refraction effect. How light refracts below the surface is also the reason why objects in shallow water can be seen as distorted.

E. Transparency

Water is typically transparent to some degree, allowing visibility of objects or structures submerged beneath the surface. Water shading algorithms incorporate transparency to accurately depict the visual properties of water.

F. Surface Roughness

Water surfaces exhibit varying degrees of roughness or smoothness, resulting in different levels of reflection and refraction. The surface roughness can be influenced by factors such as wind, waves, interactions with other objects, etc. Although, as mentioned previously, this isn't something that is always depicted, it is nevertheless an important element that could be considered.

G. Subsurface Scattering

A subsurface scattering function can be used to simulate the diffusion of direct sunlight inside the water waves when facing the sun. This way one can simulate the result of a given directional light by changing its direction and even choose its tint.

This subject will be further explained in section IV.

IV. Water Volume Effects

Now focusing on the optical behavior of the water volume below the surface, there are many aspects of importance.

Following the structure used in [3], underwater shading can be divided into two categories: light behaviour and underwater effects such as caustics and sunbeams.

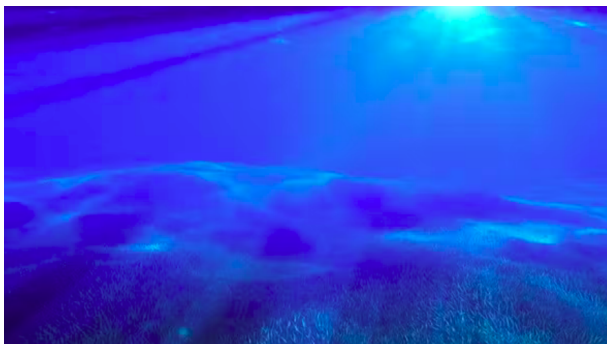


Fig. 4. Underwater rendering of light [7]

A. Light behaviour

As mentioned in [3], the main causes for light scattering and absorption in the ocean are: water molecules, living and dead organic matter and non-organic matter. "In most oceans around the world, away from the shore lines, absorption is a fairly even mixture of water molecules and organic matter. Scattering is dominated by organic matter however."

To be able to replicate light absorption and scattering, these should be considered [3]:

- **absorption coefficient:** the rate of absorption of light with distance
- **scattering coefficient:** the rate of scattering with length
- **extinction coefficient:** the sum of the two previous ones
- **diffuse extinction coefficient:** the rate of loss of intensity of light with distance after taking into account both absorption and scattering processes

- **bulk reflectivity:** is ”intended to allow us to ignore the details of what is going on, treat the volume as a Lambertian reflector, and compute a value for bulk reflectivity”.

B. Underwater effects

In [8], **Caustics** are defined as a light pattern that are a result from light rays reflecting or refracting from a curved surface and hence focusing only in certain areas of the receiving surface.

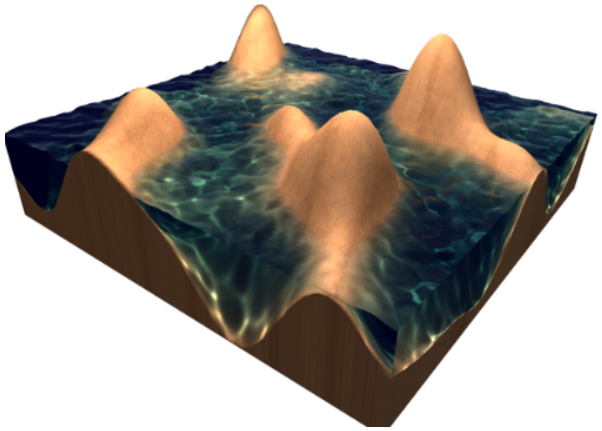


Fig. 5. Rendering of caustics by [9]

Sunbeams, also known as God-rays, are a result of direct sunlight passing into the water volume. Part of this light is scattered in other directions and a fraction arrives at the camera which is what creates the effect of God-rays.

Other effects mentioned in [10] are fog and particles.

Water can contain small floating **particles**, such as sand, small bits of seaweed, etc. The presence of these is what usually causes a reduction in visibility.

While trying to replicate the lack of visibility that happens underwater, specially as one gets deeper into the ocean, the previous authors added **fog** to their simulation. In addiction, for underwater objects, when looking at them from under in the direction of the water surface, their textures were darken, in order to mimic over-exposed areas.

V. Waves

The key element to create virtually accurate water scenarios are waves. An example of one could be a scenario that combines together waves of different scales, ranging from the kilometric to the millimetric. Unfortunately, another issue could arise because processing all of this information would result in a storage expensive program.

Therefore, the biggest difficulty is to get high quality rendering with minimum memory usage and computation cost.

Fortunately, there are already multiple known algorithms.

Mentioned in multiple articles ([3], [1], [10], [2], [11], [12], etc), **Gerstner Waves** is on of the most recurrent methods. Another well known method is **Fast Fourier Transforms (FFTs)**, which is more complex than the previous one but produces wave height fields that are more realistic. As mentioned in [3], these waves, called **linear waves** or **gravity waves** are a fairly realistic representation of typical waves on the ocean when the weather is not too stormy.

In [2] the author mentions the use of Gravity waves, ”which are caused by the pressure of gravity” and says they are a simple swaying effect of the tessellated geometry normals that create a background noise for the actual bigger waves that the author created and that are known as **Trochoidal waves** (or Gerstner waves).

According to [13] gravity waves are present ”on the surface of the ocean” and are primarily generated by winds and gravity. The authors also state that, ”although the ocean wave includes internal waves, tides, edge waves and others, it is clear that we should display at least the surface gravity waves on the computer screen, to finally represent the ocean”. Similarly to [2], the authors state that ”simple sinusoidal or trochoidal expressions can approximate a simple ocean wave”.

A. Gerstner Waves

Because of its simplicity compared to multiple other methods, in [3] is considered to be a good starting point: ”the mathematics is relatively light compared to FFTs, several important oceanographic concepts can be introduced and they give us a chance to discuss wave animation”.

Gerstner waves were first found as an approximate solution to the fluid dynamic equations (that are also be used for multiple fluid simulation like [14] and [15]).

To simply put it, the Gerstner method describes the motion of a point on the fluid surface. As said in [3], points on the surface of the water go, approximately, through a circular motion as a wave passes by.

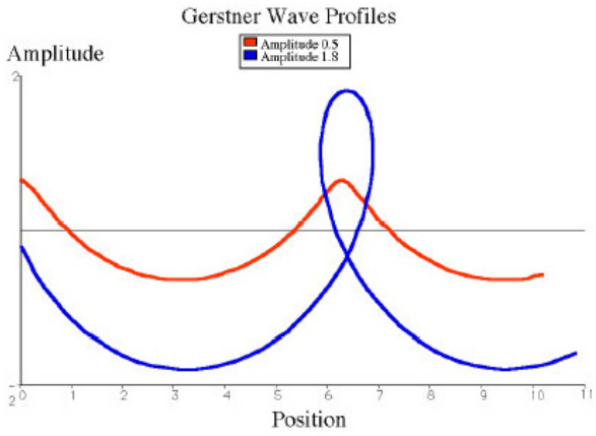


Fig. 6. Profiles of two single-mode Gerstner waves in [2]

B. Fast Fourier Transforms (FFTs)

Fast Fourier Transforms or FFTs uses statistical models based on observations of the real sea to create realistic looking waves. As mentioned in [16], this method has been used multiple times commercially, for example, in movies like *Titanic* and *Waterworld* (world renowned movies that at the time of their creation represented big breakthroughs in graphic computing).

In the statistical models, the wave height is considered a random variable of horizontal position and time, $h(x, t)$. Besides that [3] also mentions that these models are "based on the ability to decompose the wave height field as a sum of sine and cosine waves". For this matter FFTs come in handy as they a fast method for evaluating these sums.

On a more technical level, for computer graphics purposes, the slope vector of the wave-height field is also needed in order to find the surface's normal, angles of incidence and other aspects. "One way to compute the slope is though a finite difference between FFT grid points, separated horizontally by a 2D vector Δx" [3].

To get an idea of the importance of the size of the grid numbers and how the can result in different outcomes, here are some examples presented by [3]:

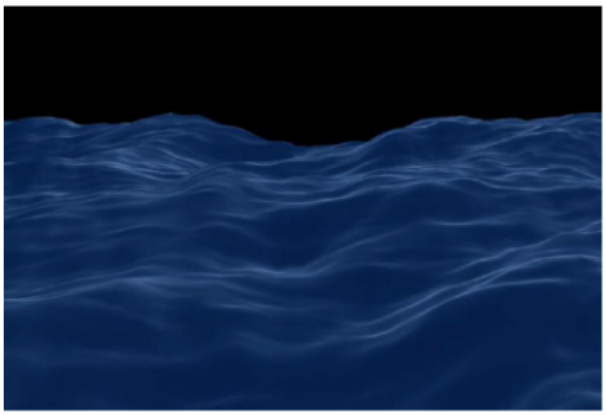


Fig. 7. Rendering with a low number of waves

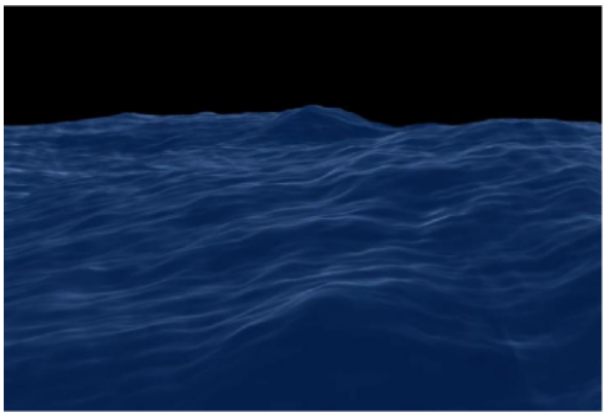


Fig. 8. Rendering with a reasonably good number of waves

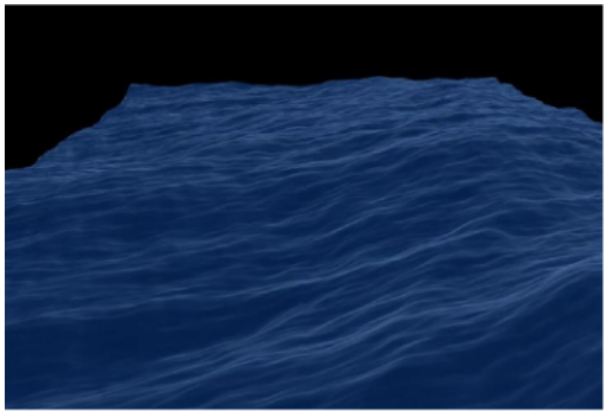


Fig. 9. Rendering with a high number of waves

VI. Motion on the surface

As previously mentioned, water is dynamic, with its movement and behavior influenced by physical forces such as gravity, wind or interactions with objects. Simulating the dynamic behavior of water is crucial for achieving realistic water shading. Computational fluid dynamics techniques, such as Navier-Stokes equations (which we take into consideration a bit further since [3] considered them for Bernoulli's equation) or particle-based simulations, are employed to model the movement, waves, splashes, and other dynamic effects in water.

To mimic object motion on the ocean surface there are many mathematical equations and physics laws that have helped study this and can aid on getting a relatively accurate result.

A. Bernoulli's Equation

Bernoulli's principle states that an increase in the speed of a fluid occurs simultaneously with a decrease in static pressure or the fluid's potential energy [17].

Bernoulli's equation may be used due to its capability on simulating a variety of surface dynamics effects, including wave breaking in shoaling shallow water. [3] presents the final equation as follows:

$$\frac{\delta\phi}{\delta t} + \frac{1}{2}(\nabla\phi)^2 = -p - U \tag{4}$$

$$\nabla^2\phi(x, t) = 0 \tag{5}$$

Where ϕ is the potential velocity function.

B. Linearization

To reduce Bernoulli's equation, [3] went even further as to applying two restrictions: "linearize the equations of motion, and limit evaluation of the equations to just points on the surface itself, ignoring the volume below the surface".

The final equations for surface motion that were presented are the following:

$$\frac{\delta\phi(x_\perp, t)}{\delta t} = -gh(x_\perp, t) \tag{6}$$

$$\frac{\delta h(x_\perp, t)}{\delta t} = \sqrt{-\nabla_\perp^2}\phi(x_\perp, t) \tag{7}$$

Which is then transformed into:

$$\frac{\delta^4 h(x_\perp, t)}{\delta t^4} = g^2 \nabla_\perp^2 h(x_\perp, t) \tag{8}$$

C. Dispersion

Because of the equation 8, the author in [3] had to take into consideration the Dispersion Relation that is in fact present in natural ocean waves and can be measured experimentally.

Considering w as temporal frequency of surface height movements, k as the spatial extent of the propagating wave, the author presents:

$$w = \pm\sqrt{gk} \tag{9}$$

VII. Water extra effects

To enhance the realism of water shading, secondary effects associated with water bodies can be simulated. These effects may include foam, spray, ripples or bubbles.

A. Foam

On [2] the author states: "**Foam** is important for interaction and at Abyssal our main goal was to make sure that any object in contact with the ocean would generate it, especially when currents are strong."

And on [16] is stated "Probably the best way to render this would be to use particle system, but this would be quite costly for our purposes". For this matter this author opted by rendering foam as if it was a a texture on top of the water surface.

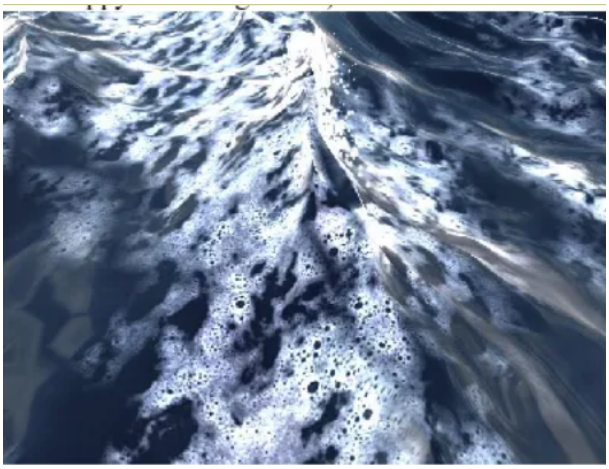


Fig. 10. Water foam rendered at [16]

Besides that, to give a more realistic effect they measured the amount of foam an interaction would create and increased it accordingly.

As a final touch, when a foam-producing point was detected, "we shouldn't set its foam amount immediately to maximum – the vertex is likely to spawn foam the next few frames as well, and increasing the foam amount slowly gives a better visual result".

Another approach is described in [2] where **Distance Fields** data is used. Here the author created a force field for every object and generated foam every time those fields overlapped with the ocean surface.

Other aspect that was taken into consideration was that foam was only generated in the direction of the ocean's current.

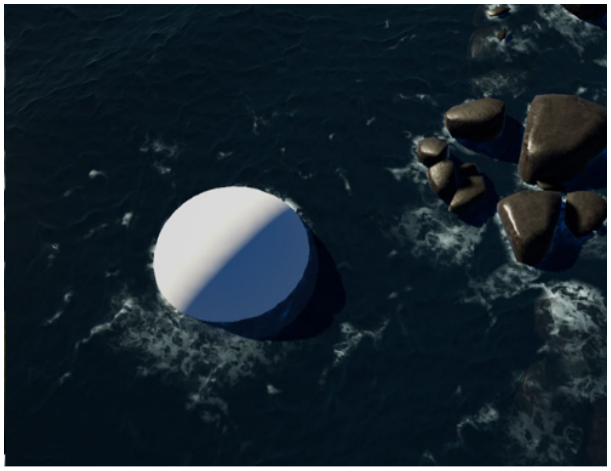


Fig. 11. Water foam rendered at [2]

B. Spray

When water collides against objects it creates **Spray**. The same can happen when bigger waves collide against each other. For this purpose [16] used a particle system with simple Newtonian dynamics to recreate it:

"Each particle is given an initial velocity taken directly from the water-surface's velocity, at the spawning position, with added turbulence. It's then updated according to gravity, wind and other global forces thereafter."

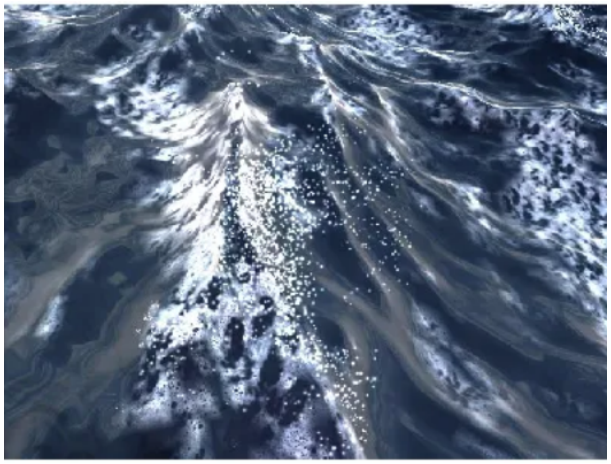


Fig. 12. Water spray rendered at [16]

C. Bubbles

Similarly to section VII-B, **Bubbles** can be created using a particle system. In [16] the author opted for moving the bubbles "on a sinus path around the buoyancy vector up to the surface were they are killed".

D. Ripples

Capillary waves, also commonly denominated by ripples, are a result of the immediate reaction to the interaction on an object with water.

"Ripples follow behind any object that moves fast enough to leave a trail and diverge if they collide with other ripples until their energy disperses completely." - [2].

The author's solution was to locate all the objects that were on the ocean's surface and at the same time near the camera. Then those objects' specific texture maps were injected into the water shader.

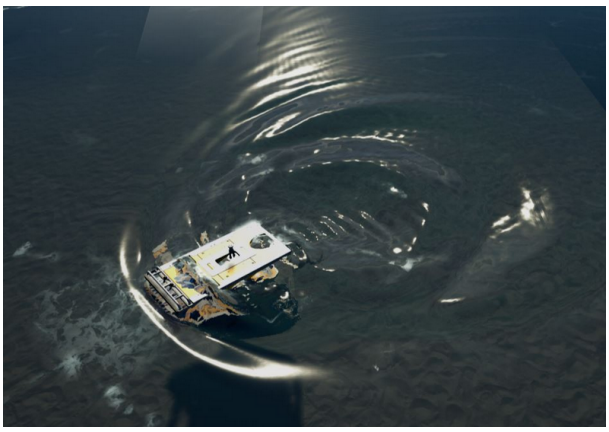


Fig. 13. Water ripple rendered at [2]

This method, however, quickly became too heavy to process.

A successful implementation would be presented in [18]. Here the wave surface was sampled by a quad mesh aligned with iso-parameter lines for reducing normal noise.

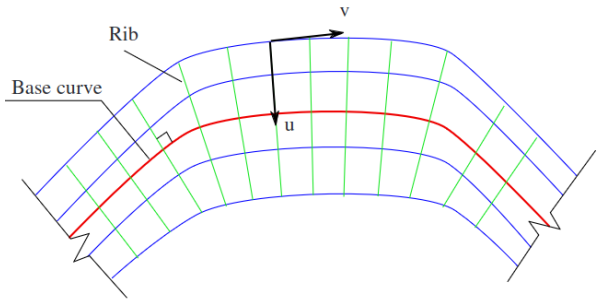


Fig. 14. Water ripple theory by [18]

It should be noted that, for ripple collisions, a dedicated mesh patch was also generated. When it comes to the capillary ripples fading effect, ripples would progressively fade with distance d.

$$\delta = \begin{cases} 0, & d \leq d_{\text{near}} \\ \frac{d - d_{\text{near}}}{d_{\text{far}} - d_{\text{near}}}, & d_{\text{near}} < d < d_{\text{far}} \\ 1, & d \geq d_{\text{far}} \end{cases} \tag{10}$$

VIII. Challenges and Advancements

Water shading and geometric simulation present significant challenges in terms of computational complexity and achieving real-time performance. Simulating the complex behavior of water in real-time applications requires efficient algorithms and optimizations. Researchers and developers have proposed various advancements to address these challenges. These include level-of-detail techniques to adapt the simulation resolution based on the distance from the viewer, simplified models for real-time simulations and parallel computing approaches to harness the power of modern graphics hardware.

Throughout this article were mentioned multiple techniques and algorithms that were proposed for solving many of these complex behaviours. However there is always space for improvement since some of them also have some downsides, that were even mentioned in the previous articles.

From section V-B, about **FFTs**, the approach mentioned by [3] has several limitations that make it unworkable for really interactive applications. The author explains that, because the wave height is computed everywhere in one FFT computation, computing wave height in limited areas of the surface grid is simply not possible without compromising previous calculations.

"This makes it very hard to customize the wave propagation problem from one location to another. So if you want to put some arbitrarily-shaped object in the water [and] move it around some user-constructed path", this might not be the best method to do so. Another problem mentioned is that "if you want to have a shallow bottom, with a sloping beach or underwater sea mound or some other variability in the depth of the water, the FFT method again is a challenge use."

Another occurring problem mentioned in [18] is related to **Ripples** which were addressed in section VII-D. The authors state: "The filtered geometric information should be accounted for in the illumination model, which is a general and tough problem."

For [1], a problem associated with their **BRDF** based algorithm is that it only works for deep water waves and does not work for coasts and shores. Also, the authors mention that they "do not handle white-caps" (waves blown by the wind that are white at their tops) "which appear for winds above 25 $km.h^{-1}$".

And lastly, some limitations of the simulation proposed by [10]:

- the difference between the simulated particle motion and the physical particle motion from the video, which "could help to improve the sense of presence".

- automatic detection of the water surface to dictate the lighting parameters in real-time, since the authors used light detection and the ambient colour of the background, which updates per frame, to update the underwater lighting effects

On a more positive note however, [19] mentions: "An interesting development is the increased use of ray tracing methods in current real time applications. Even if the presented GPU ray tracing method is again restricted to certain simplifications upcoming approaches may circumvent these restrictions resulting in either dropping some restrictions, qualitative better renderings or both."

Through research, some relatively recent breakthroughs involve:

- Real-Time Water Rendering: some progress has been made in real-time water rendering, particularly in achieving high-quality reflections and refraction. Advancements in techniques such as screen space reflections, ray tracing, and advanced shading models have allowed for more realistic and interactive water surfaces in real-time applications and games.
- Fluid Simulation: Computational fluid dynamics (CFD) methods for simulating fluid behavior, including water, have seen advancements in terms of accuracy and efficiency. Researchers have developed improved numerical methods, such as the Lattice Boltzmann method (LBM), for simulating complex water motion.
- GPU Acceleration: Utilizing the power of GPUs for water shading and simulation has been a major focus point for researchers. GPU-based techniques have enabled faster computations, allowing for more detailed and visually appealing water rendering. As GPUs evolve, so does the ability to process highly complex algorithms.

IX. Conclusion

Water shading and geometric simulation techniques have revolutionized the way computer graphics render water bodies. By accurately capturing the surface appearance and dynamic behavior of water, these techniques enable the creation of realistic virtual environments that immerse users in lifelike water simulations. Ongoing advancements continue to push the boundaries of visual fidelity and real-time performance, enhancing the overall quality of water rendering in computer graphics. As technology progresses, we can look forward to even more realistic and captivating water simulations in the future.

The field of water shading and geometric simulation in computer graphics is continuously evolving. Researchers and developers constantly strive to refine algorithms, improve visual fidelity and optimize computational efficiency. As technology progresses, we can anticipate even more stunning and realistic water simulations that push the boundaries of virtual environments, further blurring the line between the virtual and the real.

References

[1] E. Bruneton, F. Neyret, and N. Holzschuch, "Real-time realistic ocean lighting using seamless transitions from geometry to brdf," *Computer Graphics Forum*, pp. 487–496, Jan 2010. [Online]. Available: https://inria.hal.science/inria-00443630/en

[2] R. Carreiro, "A look through the water's surface," Sep 2020. [Online]. Available: https://abyssal.eu/a-look-through-the-waters-surface/

[3] j. Tessendorf, "Simulating ocean water," *SIGGRAPH 2004 Course Notes 31*, 2004. [Online]. Available: https://people.computing.clemson.edu/ jtessen/reports/papers_files/coursenotes2004.pdf

[4] A. Martin, "Radiosity." [Online]. Available: https://web.cs.wpi.edu/ matt/courses/cs563/talks/radiosity.html

[5] the Physics Classroom, "Specular vs. diffuse reflection." [Online]. Available: https://www.physicsclassroom.com/class/refln/Lesson-1/Specular-vs-Diffuse-Reflection

[6] T. Andersson, "Sky reflections and fresnel," May 2018. [Online]. Available: https://unitywatershader.wordpress.com/2018/05/14/sky-reflections-and-fresnel/

[7] "Underwater lighting." [Online]. Available: https://elements.envato.com/underwater-lighting-KGXENFX

[8] J. Guardado, "Chapter 2. rendering water caustics," 2007. [Online]. Available: https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-2-rendering-water-caustics

[9] L. Baboud and X. Décoret, "Realistic water volumes in real-time," *Eurographics Workshop on Natural Phenomena*, 2006. [Online]. Available: https://artis.inrialpes.fr/Publications/2006/BD06a/water06.pdf

[10] S. Thompson, A. Chalmers, and T. Rhee, "Real-time mixed reality rendering for underwater 360° videos." [Online]. Available: https://www.wgtn.ac.nz/__data/assets/pdf_file/0011/1771868/ismar2019-paper-real-time-mixed-reality-rendering-for-underwater-360-videos.pdf

[11] M. Finch, "Chapter 1. effective water simulation from physical models," 2007. [Online]. Available: https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-1-effective-water-simulation-physical-models

[12] K. Li and L. Wu, "Water simulating in computer graphics," *Reports from MSI*, Sep 2007. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:205412/FULLTEXT01.pdf

[13] N. Lee and K. W. Baek, Nakhoon Ryu, "Real-time simulation of surface gravity ocean waves based on the tma spectrum," *International Conference on Computational Science*, 2007. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-72586-2_16

[14] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications: Proceedings of the 2003 acm siggraph/eurographics symposium on computer animation," *ACM Conferences*, Jul 2003. [Online]. Available: https://dl.acm.org/doi/10.5555/846276.846298

[15] J. Stam, "Real-time fluid dynamics for games." [Online]. Available: http://graphics.cs.cmu.edu/nsp/course/15-464/Fall09/papers/StamFluidforGames.pdf

[16] L. S. Jensen, "Deep water animation and rendering," Sep 2001. [Online]. Available: https://www.gamedeveloper.com/programming/deep-water-animation-and-rendering

[17] G. K. Batchelor, *An introduction to fluid dynamics*. Cambridge University Press, 2000.

[18] Q. Yu, N. Praizelin, F. Rochet, and F. Neyret, "Featured-based vector simulation of water waves," 01 2009.

[19] B. Fleck, "Real-time rendering of water in computer graphics," *186.162 Seminar WS*, 2008. [Online]. Available: https://www.cg.tuwien.ac.at/courses/Seminar/WS2008/arbeit$_f leck.pdf$