Accelerating and enhancing rendering of realistic ocean scenes

Emmanuelle Darles, Benoît Crespin, Djamchid Ghazanfarpour

Accelerating and enhancing rendering of realistic ocean scenes

E. Darles, B. Crespin, D. Ghazanfarpour
XLIM – Université de Limoges
83 r d'Isle, 87000 Limoges, France
[emmanuelle.darles|benoit.crespin|djamchid.ghazanfarpour]@xlim.fr

ABSTRACT

Modeling and rendering realistic ocean scenes has been investigated by many researchers in the past few years, providing different models of ocean waves, fine-scale details, and light-water interactions. This paper describes a practical data structure for accelerating ray-marching for a well-known wave model defined as a sum of trochoids waves. This data structure exhibits nice properties that allow to take into account spatial and temporal coherence as well as reducing aliasing effects. In addition to this, it offers a unified framework in which fine-scale details, such as scattering phenomena due to spray, foam or particles suspended in the water, can be added easily. Finally, we present some benefits of the model, which can handle physically-based light-water interactions and glare effects.

Keywords

Physically based modeling, ocean scenes, graphics data structures, ray-marching, sphere tracing

1. INTRODUCTION

Realistic ocean scenes are more and more present in computer-generated films, motion pictures and electronic games, but creating such scenes is a difficult task because a lot of modeling and rendering aspects need to be addressed. Furthermore, as the ocean surface is usually very large, raytracing is computationally intensive. The most important aspect in this context is the surface model used to represent realistic waves; it should allow realistic waves shape and motion at the lowest possible cost.

Remaining issues are light-water interactions and complex phenomena occurring at the surface, which usually depend on atmospheric conditions:

- foam, appearing through advection and turbulent diffusion phenomena during the simulation, modifies radiance at the surface
- sprays, *i.e.* small droplets created by foam interacting with wind, involve attenuation and diffusion of light above the surface
- second-order scattering: the attenuation and diffusion of light within the water body, due to particles in suspension

Taking these phenomena in consideration greatly enhances the realism of ocean scenes as long as they rely on physical parameters and are approximated with sufficient accuracy. In a raytracing environment we also wish to reduce computations and memory requirements.

This paper addresses all these issues, by proposing a new method based on a set of spheres that combines an efficient surface model and physically-based complex phenomena, integrated into a unique data structure. After reviewing different surface models and complex phenomena proposed in the literature in section 2, we choose a parametric model based on real measurements that gives realistic results and present an efficient method for computing ray-surface intersections based on sphere tracing in section 3. This approach is then extended to include complex phenomena as described above, using the same simple and efficient data structure (section 4). Finally, results are presented in section 5 along with considerations about implementation and performances of our approach.

2. RELATED WORK

Modeling and rendering realistic ocean scenes has been investigated by many researchers in the past few

years. We focus mainly on papers dealing with scenes depicting the ocean far from the shore, i.e. where breaking waves do not appear. Real-time methods are also briefly reviewed although our work is more oriented towards ray-tracing. A more general state-of-the-art report can be found in [Igl04].

2.1 Ocean surface models

Models that attempt to represent the ocean surface accurately can usually be classified in two different approaches: parametric or spectral. Parametric approaches [FR86, Pea86, TB87, CGG01] represent the ocean surface as a sum of periodic functions which describe waves as a circular motion of water particles. This procedural model is very efficient but does not yield realistic results. On the other hand, spectral methods [MWM87,Tes01] are based on oceanographic measures, synthesized by spectral analysis and hence represent the ocean surface as a height field computed from a sum of sinusoids of various amplitudes and phases; small-scale waves and ripples are modeled directly by adding noise perturbation. This approach ensures high realism, but is not easily controllable.

To overcome these problems, hybrid, procedural models were proposed [PA01, TG02]. The parameters can be obtained automatically from real oceanographic spectra, which allows realistic animations of the ocean surface. Another advantage of the model is its implicit definition, which is well-adapted to ray-tracing. Real-time representations of hybrid models can also be obtained by generating an optimized surface mesh only where necessary, thus reducing both the sampling and the number of waves on the fly [HNC02, Mit05, HVT+06]. But real-time methods can hardly handle complex phenomena reviewed in the next section.

2.2 Light-water interactions and complex phenomena

The model proposed in [PA01] includes the apparition of foam on waves crests, which locally modifies the optical properties of the ocean. The surface is discretized and rendered by a Monte Carlo path tracer, which also takes into account the turbidity of the ocean, *i.e.* the opacity of the surface depending on particles suspended in the water. This parameter allows to determine the amount of light scattered towards the surface, and provides realistic simulations of several types of water. Second order scattering is also taken into account in [IDT03] by computing radiance at sampled points on a set of horizontal slices within the water volume using volume rendering graphics hardware.

Complex phenomena, such as foam, spray or splashes are usually modeled and rendered using particle systems [JG01,TFK+03,HW04]. In [TRS06], animation of drops and small-scale interactions such as ripples or splashes are obtained by combining a two-dimensional water surface simulation and a three-dimensional fluid simulation along with a physically based model.

As a conclusion, among these methods only one [PA01] seems able to provide realistic ocean scenes, by combining an efficient surface model and physically-based complex phenomena. Our work follows the same goal, by integrating these different aspects into a unique data structure.

3. RAY-WAVES INTERSECTIONS

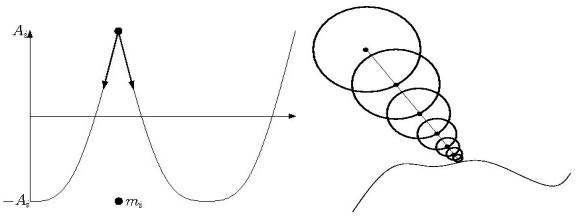

Figure 1. (a) Trochoid example (b) Sphere tracing

Our implementation uses the ocean surface model of [TG02], where the reader can find more details. The model is based on a summation of elementary waves called *trochoids* (see Fig. 1a), which yields realistic waves shapes varying from a smooth profile in the case of calm ocean, to a sharpened crested shape in case of an agitated ocean surface due to greater wind activity. Each trochoid is defined as a displacement function applied to a reference plane, parameterized by its amplitude A_i, its frequency f_i, its wavelength λ_i, its direction of propagation in the plane θ_i and its phase ϕ_i. Several methods are proposed in [TG02] to extract correct parameters from real measurements. All the pictures in this paper were obtained using Pierson-Moskowitz spectrum; high-frequency trochoids were amplified to represent capillary waves created by the wind.

Realistic ocean waves are obtained using 50 different trochoids or more. The counterpart is that a lot of evaluations of trigonometric functions (or access to a precomputed, lookup table storing trochoids) are needed to compute the displacement of a single point; this is particularly critical when rendering ocean scenes using ray marching, since it involves computing many intersections between rays originating from the camera and the ocean surface.

3.1 Original sphere tracing formulation

Consider the zero-set of an implicit function $F: R^3 \rightarrow R$, *i.e.* the set of points $p \in R^3$ that satisfy

$F(p) = 0$. Brute-force ray marching attempts to find the intersection between a ray and such a surface using a fixed, constant increment Δ. Let p be a point along the ray such that $F(p) > 0$, and let $p' = p + \Delta$ be another point along the ray such that $F(p') < 0$; in that case, the ray crosses the surface somewhere between p and p'. Otherwise the process continues for p' and $p'' = p' + \Delta$. The choice of Δ is critical: if it is too small, the whole process requires a huge amount of evaluations of the implicit function, if it is too large intersections can be missed. A solution to this problem is the *sphere tracing* method [Har96, HW96], applicable for implicit surfaces that exhibit a Lipschitz bound L such that $|F(p)-F(q)| \leq L\|p-q\|$, $\forall\ p,q \in R^3$. In that case, the brute-force algorithm can be enhanced by computing Δ at each evaluation step by: $\Delta = F(p) / L$. This guarantees that the next point along the ray $p' = p + \Delta$ satisfies $F(p) \geq 0$, making it possible to converge very quickly to the intersection (see Fig. 1b). The term "sphere" refers to the fact that Δ represents the radius of a sphere guaranteed not to intersect the zero-set of the implicit surface.

Suppose that F is defined as a sum of N functions:

$$F(p) = \sum_{i=1}^{N} n_i(p) \qquad (1)$$

Unenhanced sphere tracing in that case would use the global Lipschitz bound L of function F; but this would also require to evaluate each individual term at every step along the ray. The solution is to maximize L by computing an "efficient" bound L_e such that $L_e < L$ without evaluating each term; this is summarized in Algorithm 1.

Require: components sorted by decreasing $n_i(p)/L_i$
Require: minimum m_i of each component
$L_e = 0$; $den_e = 0$; $num_e = m_0 + m_1 + \ldots$
for all component i in the sorted list
  num_e += $n_i(p) - m_i$; den_e += L_i;
  if $(L_e > (num_e / den_e))$ then **return** L_e;
  else $L_e = num_e / den_e$;

Algorithm 1: computation of L_e for a point p

It can be noted that the sorting step still seems to require the evaluation $n_i(p)$ of each component; actually an approximated value based on the previous iteration is used instead. Experiments using this method show speedups by at least a factor 10 compared to brute-force raytracing, for fractal surfaces defined by about 10 individual components.

3.2 Sphere tracing the ocean surface

In the case of our ocean surface model, and assuming that the equation defining the reference plane is $y=0$, the displacement n_i at a point $p:(x, y, z)$ at time t is defined as:

$$n_i(p) = ((y - A_t)/N) - A_i \; trochoid(k_i(x * \cos\theta_i + z * \sin\theta_i) - \omega_i * t + \varphi_i) \qquad (2)$$

where N is the total number of waves, At is a global amplitude term, $\omega_i = 2\pi f_i$, $k_i = 2\pi / \lambda_i$ and *trochoid* describes the motion of a water particle. Thus, the implicit function has the same form as in Eq. 1.

In order to apply Algorithm 1, we still need to address several issues:

- ***choice of a minimum m_i for each component***: since each component is bounded, its lowest value occurs when $(y-A_t)/N=-A_i$ and the trochoid reaches its maximum value. In that case, the global minimum is given by $m_i=-2A_i$ (see Fig. 1a).
- ***choice of a Lipschitz bound L_i for each component***: in the case of a displacement function applied to a reference plane (such as a trochoid), we can simply use its maximum derivative. For our model, this maximum slope is obtained when the term $trochoid(k_i(x\ cos\theta_i + z\ sin\theta_i) - w_i t + \phi_i) = 1$
- ***antialiasing***: as in [TG02], a simple antialiasing technique consists in progressively reducing the amplitudes of high-frequency components depending on the projection of the corresponding pixels on the screen (see Fig. 2). In our case, unnecessary components are neglected during the computation of the global efficient bound L_e, thus reducing the overall computation cost.

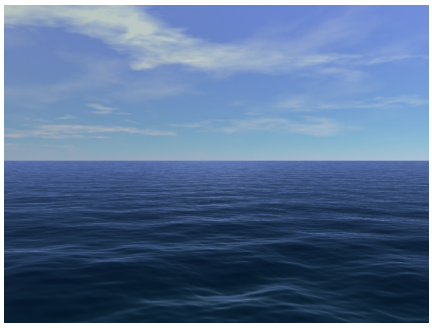

Figure 2. Simple antialiased scene

3.3 Coherence and data structure

As noted in [HW96], *spatial coherence* can be exploited since the evaluation $n_i(p)$ is bounded:

$$n_i(p') - L_i\|p - p'\| \leq n_i(p) \leq n_i(p') + L_i\|p - p'\|$$

where L_i is the Lipschitz bound and p' is a neighboring point. We can now estimate the value of $n_i(p)$, provided that $n_i(p')$ was evaluated before. This estimation is used to perform the prerequisite sorting stage of Algorithm 1, thus avoiding numerous function evaluations; as a counterpart, the global Lipschitz bound L_e may not be a maximum depending on the distance $\|p-p'\|$. In our implementation, we use estimates if $\|p-p'\|$ is lower

than a maximum distance, which is larger than the distance between successive points along the ray.
Image coherence can also be exploited directly by the sphere tracing algorithm since, for a given frame, a sphere computed along a ray is guaranteed not to contain an intersection with the surface. Thus, if a ray intersects a sphere computed for a previous ray, then it may progress safely to the next intersection with this sphere. Otherwise, the sphere tracing is performed as usual (see Fig. 3a).

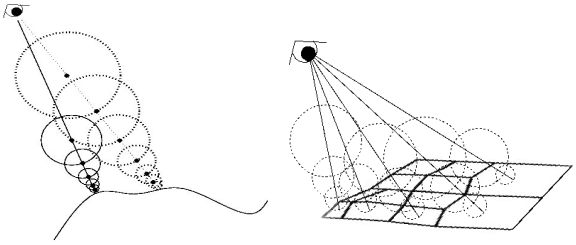

Figure 3. (a) Image coherence (b) Quadtree structure

It is noticeable that exploiting image coherence requires storing successive spheres along rays. Our data structure is a quadtree, aligned with the reference plane, which stores successive spheres along rays indexed by their intersection with the surface (see Fig. 3b). Any accelerating structure could be employed, but a quadtree seems a natural choice since fluid surfaces usually have regular shapes (especially ocean waves) which adapt well to axis-aligned partitioning. Following [HW96], image coherence is thus implemented by searching for previously computed spheres in a breadth-first manner, recursively investigating the neighbors of a ray at the same level in the quadtree. It is also worth noticing, as shown on Fig. 3b, that this data structure is well-adapted to antialiasing: subdivision simply depends on the distance to the camera.

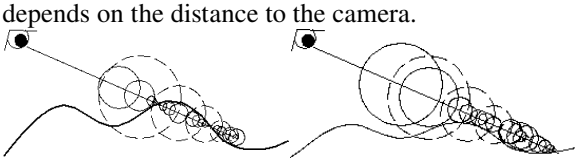

Figure 4. Starting from spheres traced at frame *t* (dotted), intersection at frame *t+1* can be closer (left) or further (right)

Our data structure is also useful to exploit *temporal coherence* between two successive frames if the camera does not move. In this case, spheres stored for a single ray at frame *t*, as well as the ray-surface intersection point p_t, can be re-used for the same ray at frame *t+1* (see Fig. 4):
1. Existing spheres from frame *t* are marked as *old*;
2. Sphere tracing is performed backwards (*i.e.* toward the camera) by re-computing the radius of existing spheres and marking them as *new*, until the maximal amplitude of the surface is reached. New spheres are also added to fill the gaps that may result;
3. The algorithm stops if the last intersection found is closer to the camera than p_t (Fig. 4, left);
4. Otherwise, sphere tracing is performed forwards starting from p_t (Fig. 4, right).
This strategy can also be adapted for image coherence: if a sphere intersecting a ray was marked as 'old', its radius is simply recomputed and marked as 'new'.

3. LIGHT-WATER INTERACTIONS AND COMPLEX PHENOMENA

4.1 Foam

We are interested in rendering foam which appears in ocean scenes far from the shore, also called passive foam, due to advection and turbulent diffusion of bubbles near the surface under determined conditions (*i.e.* wind speed greater than 13km/h). Fig. 5 shows a real picture of this phenomenon.

Figure 5. Real ocean scene with passive foam

We do not address in this paper active foam due to breaking waves, which has shorter lifetime and surface coverage, since our surface model does not represent this type of waves.
One approach to foam rendering in computer graphics makes use of particle systems [HW04] to simulate life and death of bubbles at the surface. Realistic results are obtained, but this solution can't be applied in the case of large ocean scenes. Another solution is to compute the amount of foam at a point on the surface according to different criteria: height variation between its neighbors [JG01], or amplitude of waves propagating around it [JBS03], which yields more foam in high-frequency areas. Unfortunately foam does not move in a realistic way between successive frames of an animation. Therefore, as in [PA01], we prefer another approach which considers different atmospheric conditions occurring in the process of foam apparition based on oceanographic results [MM86].

4.1.1 Position

The amount of foam s covering the surface at a given point depends on wind speed U_{10} measured at 10 meters above the surface, and the difference between water temperature T_w and air temperature T_a:

$$s = 1.59 * 10^{-5} * U_{10}^{2.55} * e^{0.086*(T_w - T_a)} \quad (3)$$

This empirical formulation is used in oceanographic research; but experiments show that it does not take into account the apparition of foam in regions more exposed to wind, especially on waves crests, nor in perturbed regions. Therefore, we propose to extend Eq. 3 in order to include the height of the considered point, and its local slope, thus obtaining the amount of foam s':

$$s' = s * \left(\frac{y - y_{min}}{y - y_{max}}\right)^{n} * \|\vec{N}\| \quad (4)$$

where y_{min} (resp. y_{max}) is the minimal (resp. maximal) height and $\vec{N}$ is the (non normalized) normal vector at the considered point. Parameter n allows the user to control the spread of foam on waves crests; based on empirical experiments, our most visually convincing results were obtained with $n \in [6, 9]$. To avoid noticeable foam patterns repetitions at the surface, a noise function is added, filtered according to the distance to the viewer in the same way as the antialiasing technique described in section 3.2.

4.1.2 Radiance

Here again, an empirical formula for the radiance L_{foam} is suggested by oceanographers [Koe84]:

$$L_{foam} = s' * f_{ef} * R_e$$

where R_e is the radiance of pure white foam. The efficiency factor f_{ef} is originally defined as $0.4 +- 0.2$; to account for the foam's age, we modify it into $0.4 - 0.2.t.10^{-2}$ (truncated to 0 if negative) where t is the time parameter. Total radiance at the surface L_{tot} is then:

$$L_{tot} = L(0) + (1 - k_r) L_r(0)$$

$$L(0) = L_{sky} + L_{sun} \cos\theta_{sun} + L_{foam}$$

where $L(0)$ is the radiance on the surface computed by taking account the luminance of the sky L_{sky} (resp. sun L_{sun}), k_r is the Fresnel reflexion coefficient and $L_r(0)$ is the refracted radiance (see section 4.3).

To take into account the motion of foam during an animation, we use a texture in image space to store positions and ages of foam on the surface. For each new frame, a preprocessing step consists in incrementing age and displacing positions using formulas similar to Eq. 2. Then, for each pixel, the distances between its projected point on the surface and the neighboring points from the previous foam texture are considered:

1. if the distance is lower than a given threshold, the corresponding texel is used
2. if all the distances are greater than the threshold, a new texel is created in the foam texture using Eq. 4.

These distances are computed as soon as an intersection point between the corresponding ray and the ocean surface is found, thus the integration of this computation in our sphere tracing algorithm is straightforward.

Figure 6. Foam obtained with U_{10} = 13 km/h

Foam obtained with this method is shown on Fig. 6 for $U_{10} = 13$ km/h, $T_a = 20°C$ and $T_w = 12°C$.

4.2 Sprays

Oceanic spray can be seen as a gaseous exchange between the ocean and the atmosphere. It is mainly due to bubbles bursting at the surface, which release drops of diameter approximately one magnitude lower than bubbles'. Depending on their size, these drops can either go back in the water or evaporate. In the latter case, organic or gaseous molecules carried by the drops are left in the atmosphere, and thus form oceanic sprays.

In computer graphics, most existing methods simulate sprays using a particle system generated according to the surface height [JG01, HW04] or surface variations [JBS03]. But these methods do not take into account the different atmospheric conditions, and do not simulate the path of light going through sprays (as a participating medium). On the other hand, participating media in computer graphics is the subject of numerous papers (see [PCPS97] for a survey). A method is proposed in [PSB99] to characterize light attenuation and diffusion due to different atmospheric molecules in daylight. We propose to extend this work to the case of oceanic sprays.

Size distribution of sprays of radius r can be empirically computed as [MM86]:

$$\frac{dF(r)}{dr} = 1.373 U_{10}^{3.41} r^{-3} 10^{1.19} (1 + 0.0057 r^{1.05}) e^{-B^2} \quad (5)$$

$$B = 0.380 \log(r) / 0.65$$

To take into account different atmospheric conditions, we modify this equation into:

$$\frac{dF(r)'}{dr} = \frac{dF(r)}{dr} * s' \qquad (6)$$

where s' is the amount of foam computed from Eq. 4 at the orthogonal projection on the surface.

Furthermore, the vertical profile of sprays is described by [SF79]:

$$\frac{dF(r)}{dr} = \frac{dF(r,0)}{dr} e^{\frac{-z}{H_{max}}}$$

where $dF(r,0)$ is the size distribution at the ocean surface computed from Eq. 6, and H_{max} is the maximal height where sprays disappear. We now have a complete description of sprays distribution above the surface.

According to Beer-Lambert's law, the intensity of a light path going through a participating medium is diffused and attenuated. In the case of sprays, this attenuation is due to the sprays' optical thickness (AOD), defined by [PSB99]:

$$\tau_{aod} = \int_0^{z_{max}} k_e(z)dz \quad , \quad k_e = \int_{r_{min}}^{r_{max}} k(r)\frac{dF(r)}{dr}$$

$$k(r) = Q_e \pi r^2$$

where z_{max} is the maximal height above which sprays are negligible. Parameter k_e is called the extinction coefficient, and $k(r)$ is the efficient section of a spray of radius r. Parameter Q_e is called the attenuation efficiency (Q_e ~= 2 for oceanic sprays). The attenuation is then defined as:

$$L'(0) = L(0)e^{-\tau_{aod}}$$

Figure 7. Light attenuated by sprays

During sphere tracing, every time a new step is taken along a ray, the corresponding point is projected onto the surface and the amount of foam is computed, yielding to an attenuation coefficient defined for each sphere below z_{max} (since our algorithm makes use of spatial coherence as described in section 3.3, some spheres can be reused). The diffuse radiance for a given ray is obtained by integration of the diffuse radiance in all M spheres along this ray:

$$L(0) = L_{foam} + (L_{sky} + L_{sun}\cos\theta_{sun}) \sum_{i=1}^{M} e^{-k_e(z)\Delta z}$$

where Δz is the vertical distance between two successive spheres.

The influence of sprays on the scene is shown on Fig. 7 for $U_{10} = 18.5$ km/h, $T_a = 20°C$ and $Tw = 12°C$, and is noticeable only because light is attenuated.

4.3 Second order scattering

As in the previous section for the atmosphere, light is also greatly attenuated when going through water (a very small amount of light reaches 100m under the surface). This phenomenon is called second order (or sub-surface) scattering. Light diffusion in that case is mainly due to particles in suspension in the water volume, which can either be organic (phytoplancton) or formed of water molecules of different densities, and characterize the water's turbidity. Different water types (resp. A, B or C) correspond to different turbidities (resp. clear, turbid or very turbid).

Due to its complexity, second order scattering can only be simulated using hierarchical data structures: illumination volumes [IDT02], octrees [JB02] or horizontal planes [IDT03] were proposed. But these methods do not take particles in suspension into account. Another approach [PA01, CS04] consists in computing the diffuse radiance according to the type of water and its concentration of phytoplancton particles; we propose to simplify and adapt this idea to our data structure.

Following [PA01], the directional radiance $L(0,\theta,\varphi)$ in direction (θ,φ) re-emitted to the surface is defined as:

$$L(0,\theta,\varphi) = L(z,\theta,\varphi)e^{-cR} + L_{df}(z)$$
$$L_{df}(z) = L_{df}(0)(1 - e^{(-c - K_d\cos\Theta)R}) \qquad (8)$$

where $L(z,\theta,\varphi)$ is the radiance at the bottom of the sea, $L_{df}(z)$ is the total diffuse radiance, c is the extinction coefficient of water, K_d is the diffusion coefficient, and $R = z/(\cos\theta)$.

$L_{df}(0)$ is the diffuse radiance at the surface:

$$L_{df}(0) = (0.33 * b_b * E_d(0))/a\pi \qquad (9)$$

where b_b is the backscattering coefficient, inversely proportional to the phytoplancton concentration C, a is the absorption coefficient, and $E_d(0)$ represents the *downwelling irradiance* at the surface, which depends on atmospheric properties (interested readers should refer to [PA01] for more details).

We replace the term $E_d(0)$ in Eq. 9 by the diffuse radiance $L(0)$ computed by Eq. 7:

$$L_{df}(0) = (0.33 * b_b * L(0))/a\pi$$

In that case, the ray is pointing downwards; the bottom of the sea is modeled as a set of trochoids, which allows to apply the sphere tracing algorithm to compute its intersection with a refracted ray. As in the previous section, a diffuse radiance is computed for each sphere by simplifying Eq. 8:

$$L_{df}(z) = L_{df}(0)(1 - e^{(-c - K_d)z}) \qquad (10)$$

with z the vertical distance between the surface and the center of the sphere. Finally, the total diffuse radiance for a refracted ray is obtained by integrating the diffuse radiance of all M spheres along the ray:

$$L_r(0) = (Z_{bottom}) e^{-cZ_{bottom}} + \sum_{i=1}^{M} L_{df_i}(z) \Delta l \qquad (11)$$

where Δl is the distance along the ray and L_{dfi} is the diffuse radiance of each sphere computed by Eq. 10. In this case again, spatial coherence allows us to reuse diffuse radiance for previously computed spheres. Fig. 8 shows images obtained for clear and turbid waters and constant phytoplancton C = 1.0mg/m^3.

Figure 8. Clear and turbid water

5. RESULTS

5.1 Implementation

Our method is implemented as a plugin for Autodesk's Maya 6.5. In order to obtain more realistic results, a post-processing step is added for each frame to include glaring which is particularly frequent in ocean scenes, due to high contrasts between specular neighboring surfaces.

Figure 9. Glaring effect

Our implementation is an extension of a method proposed in [Tho01]: the most specular points on the surface are considered, and a "glare texture" is mapped on these points, whose size is determined according to the distance from the viewer. A minimal distance between two specular points is also introduced to avoid a concentration of glare points within a region, which would yield unrealistic results. An example is shown on Fig. 9, where all complex phenomena treated by our algorithm are also enabled (foam, sprays and light scattering).

5.2 Performances

Performances of our method were evaluated on different ocean surfaces defined by an increasing number of trochoids waves; the other parameters were constant. As expected, our method guarantees a significant speedup compared to brute-force ray-marching and unenhanced sphere tracing to compute intersections between a ray and the surface (see Table 1), increasing with the number of trochoids.

N	W. ray tracing	W. sphere tracing
30	45%	26%
40	49%	32%
50	50%	34%

Table 1. Speedup compared to brute-force ray-tracing (middle) and unenhanced sphere tracing (right) for 30, 40 and 50 trochoids (N)

Spatial coherence was evaluated on a single 640x480 image; temporal coherence was evaluated on a single 640x480 frame of an animation (see videos attached to the paper). Both comparisons were also significant (see Table 2). The total rendering time for a single image listed on the same table shows an interesting result: our method gives better results for a higher number of trochoids. This better convergence is due to our maximization approach, which tends to neglect most of the small amplitude waves with high frequencies.

N	SC	TC	Time
30	21%	24%	4:05mn
40	23%	30%	3:38mn
50	27%	39%	2:50mn

Table 2. Speedup obtained with spatial coherence (SC) and temporal coherence (TC)

Ray-surface intersections	62.5%
Sprays	18.3%
Second order scattering	12.4%
Glaring effects	3.7%
Foam	3.1%

Table 3. Repartition of different steps

Table 3 shows the computations repartition between the different parts of the algorithm. Unsurprisingly, most of the time is spent to find intersections between rays and the surface, whereas foam is computed very quickly since most of this part is performed using textures.

6. CONCLUSION

This paper presents an efficient method to render realistic ocean scenes. A sphere-based data structure allows to accelerate ray-surface intersections computations by heavily relying on spatial and temporal coherence which are well-adapted to ocean scenes. To increase the realism of generated scenes, we also propose new formulations to integrate physically-based phenomena such as second order scattering, foam and sprays into our data structure, without significantly reducing performances. The limitations of our method are mainly due to the surface model, which can not represent breaking waves and physically-based foam and spray induced by this phenomenon; we are currently studying how to integrate this type of waves.

7. REFERENCES

[CGG01] Cieutat., J.M., Gonzato, J.C, Guitton, P., A new efficient wave modeling for maritime training simulator, SCCG, 2001.

[CS04] Cerezo, E., Seron, F.J, Rendering natural waters taking fluorescence into account, Comp. Animation and Virtual Worlds, 15(5), 2004.

[FR86] Fournier, A., Reeves, W.T, A simple model of ocean waves, SIGGRAPH, 1986.

[Har96] Hart, J.C, Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces, The Visual Computer, 12(10), 1996.

[HNC02] Hinsinger, D., Neyret, F., Cani, M.P, Interactive animation of ocean waves, SCA, 2002.

[HVT+06] Hu, Y., Velho, L., Tong, X., Guo, B., Shum, H., Realistic real-time rendering of ocean waves, Computer Animation and Virtual Worlds, 17(1), 2006.

[HW96] Hart, J.C, Worley, S.P., Hyper-rendering of hyper-textured surfaces, Implicit Surfaces, 1996.

[HW04] Holmberg, N., Wunsche, B.C, Efficient modeling and rendering of turbulence water over natural terrain, GRAPHITE, 2004.

[IDT02] Iwasaki, K., Dobashi, Y., Nishita, T., An efficient method for rendering underwater optical effect using graphics hardware, Vol. Graphics, 2002.

[IDT03] Iwasaki, K., Dobashi, Y., Nishita, T., A volume rendering approach for sea surfaces taking into account second order scattering using scattering maps, Vol. Graphics, 2003.

[Igl04] Iglesias, A., Computer graphics for water modeling and rendering: a survey, Future Generation Computer Systems, 20(8), 2004.

[JB02] Jensen, H.W., Buhler, J., A rapid hierarchical rendering technique for translucent material, SIGGRAPH, 2002.

[JBS03] S. Jeschke, H. Birkholz, H. Shumann. A procedural model for interactive animation of breaking ocean waves. Computer Graphics, 2003.

[JG01] Jensen, L.S., Goliàs, R., Deep-water animation and rendering, http://www.gamasutra.com, 2001.

[Koe84] Koepke, P., Effective reflectance of ocean withecaps, Appl. Optics, 23(18), 1984.

[Mit05] Mitchell, J.L., Real-time synthesis and rendering of ocean water, Technical report, ATI, 2005.

[MM86] Monahan, E., MacNiocaill, G., Oceanic Witecaps: Their Role in Air Sea Exchanges Processes, D. Reidel, 1986.

[PA01] Premoze, S., Ashikhmin, M., Rendering natural waters, Computer Graph. Forum, 20(4), 2001.

[PCPS97] Perez-Cazorla, F., Pueyo, X., Sillion, F., Global illumination techniques for the simulation of participating media, EG Workshop on Rendering, 1997.

[Pea86] Peachey, D.R., Modeling waves and surf, SIGGRAPH, 1986.

[PSB99] Preetham, A., Shirley, P., Smiths, B., A practical analytic model for daylight, SIGGRAPH, 1999.

[SF79] Schettle, E., Fenn, R., Models for the aerosols for the lower atmosphere and the effects of humidity variations on their optical properties, AFGL-TR-79-0214 676, Envir. Research Papers, 1979.

[TB87] Ts'o, P.Y., Barsky, B.A., Modeling and rendering waves: wave-tracing beta-splines and reflective and refractive texture mapping, ACM Trans. Graph., 6(3), 1987.

[Tes01] Tessendorf, J., Simulating ocean waters, SIGGRAPH Course Notes, 2002.

[TFK+03] Takahashi, T., Fujji, H., Kunimatsu, A., Hiwada, K., Saito, T., Tanaka, K., Ueki, H., Realistic animation of fluid with splash and foam, Eurographics, 2003.

[TG02] Thon, S., Ghazanfarpour, D., Ocean waves synthesis and animation using real world information, Computer and Graphics, 26(1), 2002.

[Tho01] Thon, S., Representation de l'eau en synthèse d'images, PhD thesis, Université de Limoges, 2001.

[TRS06] Thürey, N., Rüde, U., Stamminger, M., Animation of open water phenomena, SCA, 2006.