Real-time Generation of Plausible Surface Waves

Kristian Yrjölä and Thomas Larsson
Department of Computer Science and Electronics
Mälardalen University

Abstract

We present a fast and flexible algorithm for the simulation and rendering of ocean waves. The method is designed to support efficient view frustum culling and various simple wave effects such as choppy waves, capillary waves, wave refraction, round waves, and wave-land interaction, which makes the model suitable in, e.g., computer games. The waves are numerically robust, and the execution time of the generated waves can be controlled dynamically. Finally, experimental results illustrate the interactive performance and the visual quality of the generated waves.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: water, waves, animation, culling, graphics, simulation

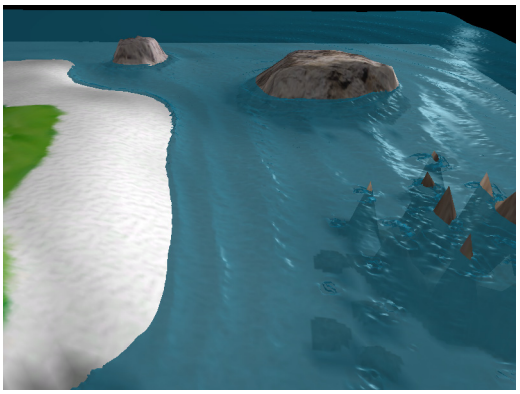

Figure 1: *An image of waves generated with the proposed method.*

1 Introduction

Simulation of realistic water waves is a computationally challenging problem [Semtner 2000; Hinsinger et al. 2002; Iglesias 2004]. There are many different types of waves, e.g., the tides, seismic tsunami waves, internal waves (waves below the water surface), object interaction waves, gravitational waves, and capillary waves [Peachey 1986]. The latter two are also called wind waves, and it is the generation of plausible wind waves in real-time that is in focus here.

Some previously proposed wave generating methods for computer graphics applications have been based on the summation of sine waves, such as Gerstner's and/or Biesel's wave models [Fournier and Reeves 1986; Cieutat et al. 2001; Hinsinger et al. 2002]. Additional details to the water surface can be added by using noise functions [Thon et al. 2000]. Wave refraction can be simulated with wave tracing [Ts'o and Barsky 1987; Gonzato and Sac 1997].

Other approaches rely on fast Fourier transforms [Tessendorf 2004; Hu et al. 2004], and solving the Navier Stokes equations for physically-based simulation of fluids [Kass 1991; Stam 1999; Foster and Fedkiw 2001; Mihalef et al. 2004]. Realistic simulation of fluids, however, can be extremely time consuming, and therefore inappropriate for interactive computer graphics applications. Nevertheless, Stam has proposed a fast and useful algorithm that solves the Navier–Stokes equations and produces complex fluid-like flows in real-time [Stam 1999; Stam 2003].

Several other methods are based on a combination of summing sine waves, spectral approaches, and other techniques [Thon et al. 2000; He et al. 2005]. Some approaches also use the modern programmability features of GPUs [Schneider and Westermann 2001; Isidoro et al. 2002; Kryachko 2005; Baboud and Décoret 2006]. We refer to the survey by Iglesias [2004] for a broader presentation of related work than the one given here.

Our solution is based on oriented Wave Train Boxes (WTBs), which are responsible for generating waves over spatially limited sea areas. The basic shapes of the waves are computed using Gerstner's equations [Tessendorf 2004], but by using several extensions and tracing wave rays inside the WTBs in the wind direction, simple forms of some natural wave effects are achieved, e.g., wave–land collisions, wave refraction, capillary waves, and round waves. In Figure 1, an example image of the generated waves is shown. The computational complexity of our approach is linear in the number of water surface points.

The main contribution of this paper is a simple and flexible wave model for the fast generation of plausible water surfaces with the following advantages: (1) The generated waves are frame-independent, which means the water surface is re-calculated from its rest position each frame. This is both memory friendly and numerically robust. (2) Wave trains are spatially limited, and mutually independent, which means that View Frustum Culling (VFC) can be used to speed-up the calculations. (3) The complexity of the water surfaces can be interactively controlled, since the computations are arranged in layers with the most basic features applied first. Furthermore, the water surface is generated as a combination of an arbitrary number of independently processed simple wave trains, so the most important can be selected according to a given time budget. (4) The model supports a simple and fast method for affecting the waves near land (wave refraction). (5) Land obstacles are detected, and in the water area behind them, the waves are canceled out. (6) The method produces plausible waves at interactive rates, see benchmarks in Section 3.

2 Wave model

A water surface is represented by a uniformly sampled height field which we call the water grid. For the generation of the waves, oriented WTBs are associated with the water grid. A WTB is used to handle the creation and simulation of a restricted rectangular area of waves. Each WTB handles a set with one or more wave trains traveling in roughly the same direction.

To create waves, a wind is associated with each WTB. Wind parameters are wind speed, direction vector, wind origin, life length, and area defined by fetch and width. When a wind starts, the size of the WTB grows during a build-up phase up to a maximum. Some properties of the wind directly define parts of the WTB, such as direction, width and origin point. The other properties help to build-up, sustain and in the end kill the WTB.

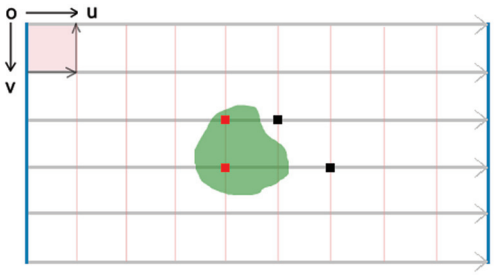

Figure 2: *A WTB with six wave rays. The internal coordinate system has its origin in the lower left corner when looking along u. Action points are registered around obstacles in order to change wave train parameters at those specific points.*

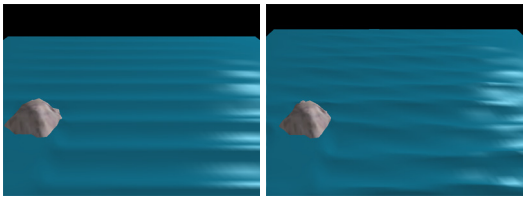

Figure 3: *Images generated from a WTB with a single wave train (left), and a WTB with three wave trains (right). As expected, more wave trains give a more irregular surface.*

There are one or more wave trains per WTB, each with its own direction, amplitude, wave length, and phase. The more wave trains used, the more complex waves are generated. Example waves resulting from a WTB with a single wave train and another with three wave trains are shown in Figure 3.

A WTB also has a set of wave rays in order to gain control over the wave train parameters locally. With this local control, the height of a wave can be increased as it approaches land, and waves behind an obstacle can be damped. However, when there are no land obstacles inside a WTB, it is possible to iterate over the water grid points in any order since each point's location is then calculated independently.

In Figure 2, the most important structural elements of a WTB are shown. The direction vectors u and v define the internal coordinate system of the WTB, with the origin always in the lower left corner looking along u. The lengths of these vectors are the actual uniform lengths between the water grid points. Thus, the WTB has as many wave rays as it spans over grid points in width. If something obstructs the surface, action points are registered as shown in Figure 2. An action point indicates a change of the wave train parameters will take place at that specific point.

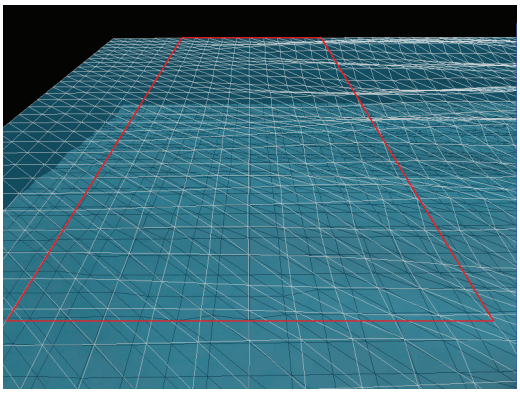

Figure 4: *Smoothening area along the side of a WTB.*

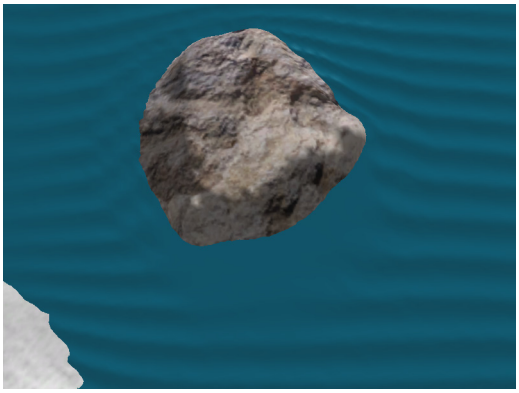

Figure 5: *Dampening of waves due to wave ray land collisions.*

The actual computation of the wave shape is based on Gerstner's wave model, which was suggested more than two hundred years ago in oceanography (see e.g. [Tessendorf 2004; Fournier and Reeves 1986]). This model is chosen because its inherent linear time complexity in the number of computed water surface points, but still, the computed wave shapes look plausible.

Waves are created by specifying a wind speed, h. The maximum wave height, H, of fully developed seas can then be approximated by

$$H = \alpha \frac{h^2}{g} \tag{1}$$

where g is the gravitational constant, and α is a dimensionless constant. This simple equation is the result of an early hypothesis suggesting a direct relation between wind speed and wave height. A reasonable approximation for fully developed wave heights can be computed by using $\alpha = 0.21$ [Resio et al. 1999][1].

The animation of the water grid points is handled by Gerstner's parametric wave equations. The goal is to make a point $\mathbf{p}_0 = (x_0, z_0)$ on the rest surface of the water (where $y_0 = 0$) to travel in a stationary circular orbit. However, by displacing the point at time t to

$$\mathbf{p} = \mathbf{p}_0 - \frac{\mathbf{k}}{k} A \sin(\mathbf{k} \cdot \mathbf{p}_0 - wt) \tag{2}$$

$$y = A \cos(\mathbf{k} \cdot \mathbf{p}_0 - wt) \tag{3}$$

the surface is perturbed into repeated trochoids, thereby also supporting the generation of so called choppy waves, depending on the relationship between k and A, where A is the amplitude. The wave vector, $\mathbf{k} = (k_x, k_z)$ is the horizontal direction in which the wave travels. The magnitude of $\mathbf{k}$ is related to the wave length λ, and is given by $k = 2\pi/\lambda$. The frequency w is related to k and the water depth D at $\mathbf{p}_0$ according to

$$w = \sqrt{gk \tanh(kD)} \tag{4}$$

This equation makes the waves slow down as the water depth decreases. However, when the water depth is large enough, we can

[1] Fournier and Reeves [1986] suggest another simple equation for determining the wave height given only the wind speed.

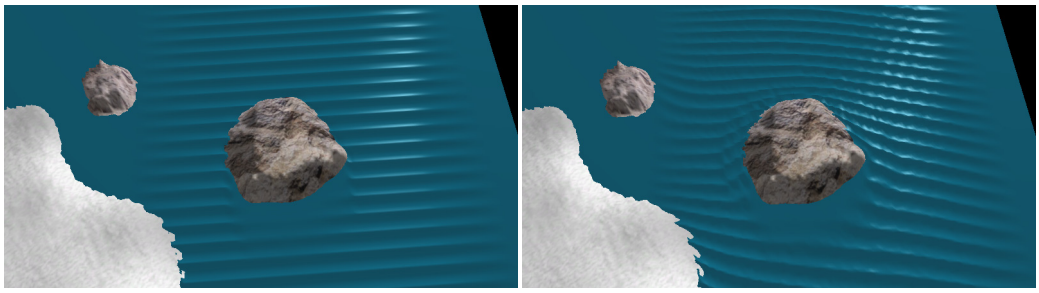

Figure 6: *Waves with no phase shift (left) and waves with a phase shift as a function of water depth (right).*

ignore the depth and simply use

$$w = \sqrt{gk} \tag{5}$$

Note that the equations 2 and 3 only create a wave with a very unrealistic regular shape. Therefore, to create a plausible wave shape, a set of N such sine waves are used and added together according to

$$\mathbf{p} = \mathbf{p}_0 - \sum_{i=0}^{N-1} \frac{\mathbf{k}_i}{k_i} A_i \sin(\mathbf{k}_i \cdot \mathbf{p}_0 - w_i t + \Phi_i) \tag{6}$$

$$y = \sum_{i=0}^{N-1} A_i \cos(\mathbf{k}_i \cdot \mathbf{p}_0 - w_i t + \Phi_i) \tag{7}$$

As can be seen, each sine wave has its own wave vector, k_i, amplitude, A_i, frequency w_i, and a phase shift Φ_i.

To avoid waves that are abruptly ended at the sides of the WTBs, a way to make the waves gradually disappearing into the water is needed. Therefore, each WTB has a smoothening area along its sides where the wave height is decreasing linearly, which is illustrated in Figure 4.

When the height of the water grid points inside the WTB is updated, the internal coordinate system is traversed along the wave rays. Since the rays are arbitrarily oriented, we have to detect which water grid points are closest. Some precautions must be taken to avoid neighboring rays updating the same grid points, or that no rays will access some grid points, resulting in visual artifacts. Therefore, for every internal grid point, the small quadrant area behind to the left is scanned for actual surface grid points, marked as a pink square in Figure 2. Because the internal grid of the WTB has the same distance between points as the water grid, albeit oriented, there can only be three possible outcomes for such a scan: zero, one or two points inside. The height is summed up for all wave trains at these points, and stored. If there is an action point, the parameters of the wave trains are adjusted. If the wave is approaching land and the depth is decreasing, the amplitude is increased slightly, but set to zero as land peaks above the surface. After the obstacle, if it is water behind it (from the wave direction point of view), the amplitude is increased again gradually to the full amplitude.

By computing the position of the water grid points along the wind direction, land collisions becomes trivial to detect, and the handling of waves hitting a land obstacle becomes extremely efficient. The land collision points can even be pre-computed as long as the WTBs are stationary. In Figure 5, this way of detecting land collisions and setting the wave amplitude to zero afterwards is shown. In nature, however, the broken waves would pass along the sides of the obstacle, grow along the crest and perhaps meet again depending on the size of the obstacle. This phenomenon is called diffraction.

Finally, note that some visual artifacts may arise during the wave animation near land when the amplitude is too high or the slope is low (like on a beach) because of land peeking through the bottom of the waves. One possible solution to this could be to adjust the parameters of the waves to make the trough of the wave stay above land. For example, the water level can be raised somewhat to simulate run-up, and decreasing the amplitude instead of increasing it can also help. Some visual artifacts may also arise when polygons of the water surface are coplanar with land polygons due to z-fighting. Increasing the depth buffer resolution can alleviate these artifacts, but can never remove the problem completely.

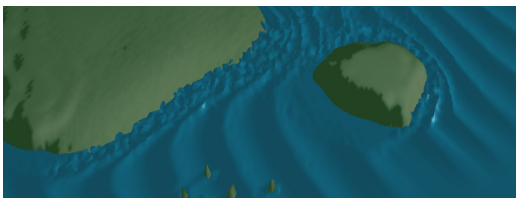

Figure 7: *Example of waves starting to queue up towards land.*

2.1 Wave effects

While updating the parts of the water grid covered by a WTB, several effects adding more details and variations can be applied. For example, a simple form of wave refraction can be achieved simply by shifting the phase of the waves locally depending on the water depth. In Figure 6, it is shown how the resulting waves look with and without this type of refraction. In contrast to this, the usual way to achieve wave refraction for Gerstner waves is to change the wave frequency according to Equation 4. However, when the frequency is changed, waves may easily start to queue up towards land as the animation proceeds according to Figure 7. This is because the speed is reduced to almost zero. Our approach keeps the propagation speed, and focuses on bending the waves.

In reality, there are also a great number of water waves that make up irregular patterns of waves with different heights. To model a seemingly irregular height variation, we can simply change the height value y along a wave in the v-direction of the WTBs by modifying the amplitude according to the formula

$$A' = A - a \sin(t + u) \sin(vf) \tag{8}$$

where a is scale factor, and f determines how often a higher wave will appear in the v-direction. Of course, some randomness can also be added to Equation 8. However, the resulting waves look more chaotic even without it, as can be seen in Figure 8.

There are also capillary waves giving rise to fine grained details on the water surface. Therefore, we add more details to the water

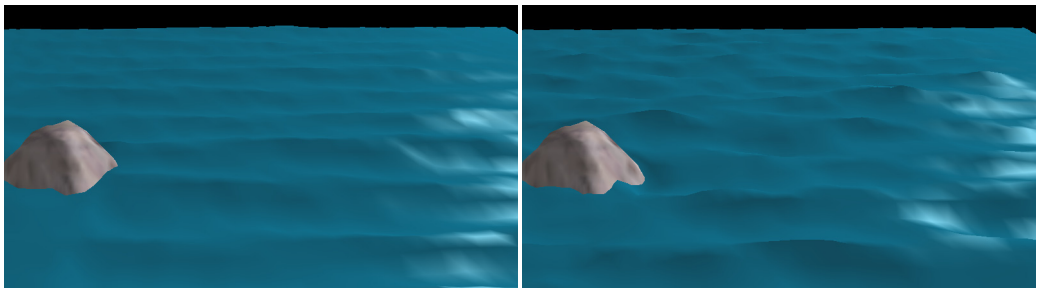

Figure 8: *Waves without the sideways height variation effect (left), and with it (right).*

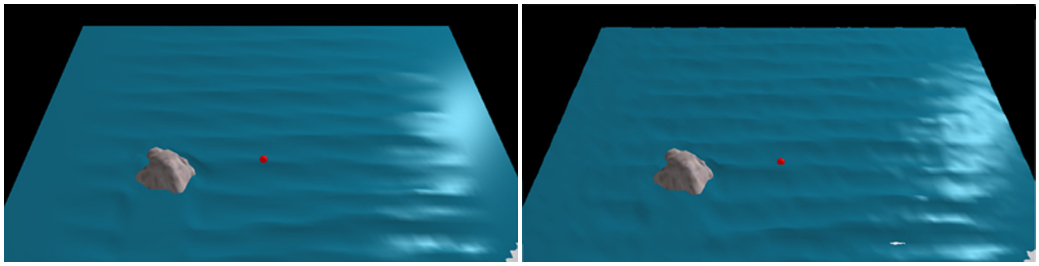

Figure 9: *A water surface without noise (left) and with noise (right).*

surface by applying a noise function adjusting the height of each water grid points slightly using the equation

$$y' = y + b\sin(rt) \tag{9}$$

where b is the noise amplitude, and $r \in [0, 1]$ is a random value. The effect of this noise function is shown in Figure 9. To avoid a completely flat water surface in areas where no wave trains are applied, the noise function must be applied directly to all visible water grid points, independently of where the WTBs are located.

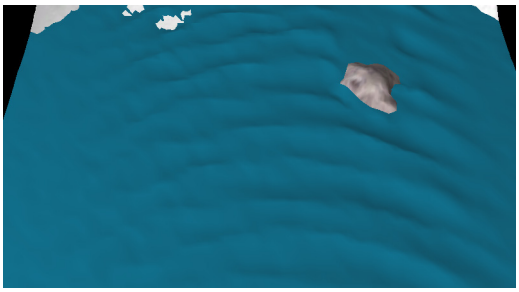

Figure 10: *Round waves resulting from waves spreading outwards from the water–wind contact area.*

Another effect that may arise can be referred to as round waves. When a wind blows over a surface with a limited width, the arising waves tend to spread outwards from the contact area. This effect can easily be incorporated into our model by a function shifting the phase of the waves in the v-direction of the WTB. For example, using the formula

$$\Phi = \frac{(v - 0.5V)^2}{V} \tag{10}$$

where V is the width of the WTB, the phase shift becomes symmetric around the center of the WTB giving rise to the round waves depicted in Figure 10.

2.2 View frustum culling

The concept of WTBs is well-suited for view frustum culling. Each WTB is a container for wave trains, and it defines a limited rectangular area where these wave trains affect the water grid points. For each WTB, we create an enclosing volume, an Oriented Bounding Box (OBB). The dimensions of the OBB are defined by the rectangular area of the WTB together with the maximum wave height, which is given by the sum of all the maximum heights of all involved wave trains.

The actual visibility check is made by extracting the planes of the view frustum each frame from the combined view and projection transformation matrices $\mathbf{A} = \mathbf{MP}$, which is done very efficiently by adding columns of $\mathbf{A}$ [Gribb and Hartmann 2001]. Then an OBB–plane overlap test is executed for each plane of the frustum, which can be computed very efficiently [Akenine-Möller and Haines 2002].

However, we need to take precaution to avoid potential false culling of overlapping WTBs. When two or more WTBs overlap, the maximum wave height inside the overlap area is the sum of the maximum wave height for each involved WTB. Therefore, when we determine the height of the OBB, we take all overlapping WTBs into account. Note that this computation can be done as a pre-process, if the WTBs are placed at fixed locations in the virtual environment.

2.3 Interactive control

To be able to interactively control the frame rate, our approach admits generating the waves using a computationally layered approach, where the most fundamental and important features are applied first. We start with conservative view frustum culling of the WTBs. Then, for each remaining WTB, the positions of the affected water grid points are computed, as they are passed by the wave rays. Note that the actual computations of the positions of the water grid points are made with the currently enabled wave trains and the possible effects either turned on or off. Finally, the noise effect may be added to the water grid.

If time permits, it would also be possible to make an additional pass over all water grid points that fall inside any of the WTBs to add other special effects, such as foam or spray, by considering the roughness of the water around each water grid point. It would also be possible to add new WTBs on the fly. However, then the computation of overlapping WTBs, and land collision points, must also be done on the fly, which otherwise are pre-computations.

2.4 Computational complexity

The performance of the algorithm is adaptive to the actual number of water grid points computed per frame. Since a single water surface point is processed in constant time, the worst case for the computational complexity of the algorithm is given by $O(kn)$, where k is the number of WTBs, and n is the number of water grid points.

In the best case, the algorithm is in $O(k)$, which occurs when all WTBs, and thus all wave trains, are completely outside the view frustum, and when the (global) noise effect is turned off. Therefore, from a scene design point of view, it makes sense to define the size of neighboring WTBs with efficient VFC in mind.

3 Results

Our wave generation method has been implemented in C++ and we have evaluated the performance by using some benchmark scenarios. The test equipment was an AMD64 3000+ 2.0 GHz CPU, with 1 GB RAM and graphics hardware ATI9800 pro with 128 MB RAM. No optical effects were used in the lighting calculation except OpenGL's own fixed-function pipeline with one light source. All animation calculations are done on the CPU, and VFC is only done for WTBs and not for the land and water geometries themselves. Some captured animations are available showing our wave generation approach in action[2].

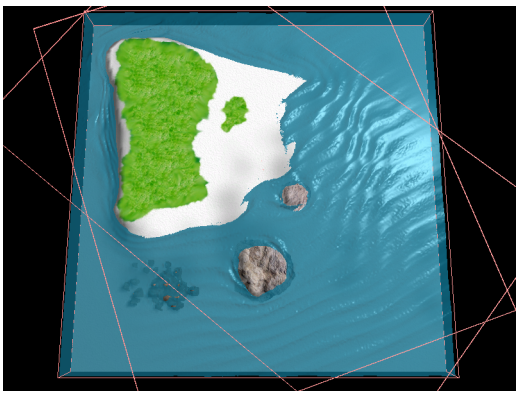

Figure 11: *Bird-eye view over the WTBs in the first scenario.*

LOD	64x64	128x128	256x256	512x512
WC	3.20 ms	12.2 ms	49.9 ms	199 ms
WR	0.80 ms	2.80 ms	10.8 ms	44.0 ms
FPS	>75	42	13	4

Table 1: *Wave Calculation (WC) and Water Rendering (WR) times in the first scenario.*

In the first scenario, there are two WTBs, both of them covering most of the water surface, see Figure 11. One of the WTBs includes one wave train, and the other two wave trains. The effects

[2] See http://www.idt.mdh.se/personal/tla/waves/

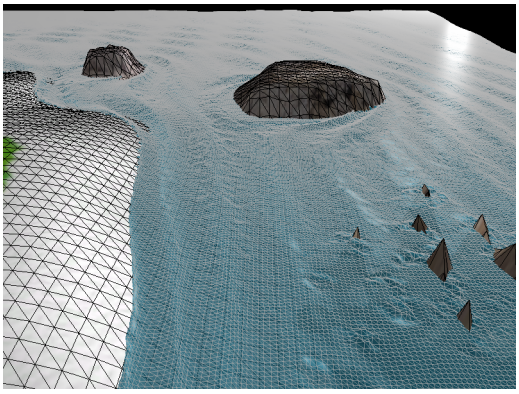

Figure 12: *Wireframe rendering of the waves in the first scenario with water grid LOD 512x512.*

used were basic Gerstner waves and depth-dependent wave refraction. The measurements were repeated for four different levels-of-detail (LODs) of the water grid, see Figure 12 for an example. The execution times are presented in Table 1. As can be seen, the time to generate the waves increases linearly with the number of grid points as expected.

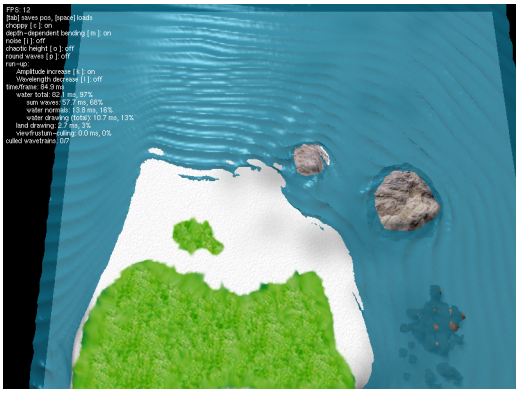

Figure 13: *Scenario 2 with seven WTBs inside the view frustum.*

In the second scenario, seven WTBs, with one wave train in each, were included to be able to examine how VFC affected the execution time. A fixed water grid of size 256x256 was used. Again, basic Gerstner waves and depth-dependent wave refraction were used. When all seven WTBs were visible (see Figure 13) the water calculation time was 57.7 ms. With none of the WTBs visible, the water calculation time decreased to 0.6 ms. In cases where one or more WTBs were culled, a speed-up of 3–10 ms per WTB was observed depending on their sizes. When the water grid size was increased to 512x512 in the same scenario, the water calculation time was 232.2 ms with all WTBs inside the view frustum, and the gain per culled WTB was between 20–40 ms.

Effect	256x256	512x512
Depth-dependent refr.	1.2 ms	6 ms
Noise	3 ms	13 ms
Irregular height	9.5 ms	39 ms
Round waves	0.5 ms	1.4 ms

Table 2: *The execution times of the tested wave effects for two LODs of the water grid.*

The second scenario was also used to measure the computation time for various parts of the algorithm. The measured parts were depth-dependent refraction, noise, irregular height, and round waves. The results from this experiment, with all seven WTBs inside the view frustum, are shown in Table 2.

4 Conclusion and future work

Even though realistic physically-based simulation of ocean waves is too complex for real-time graphics application, it is clear that simpler solutions, with a plausible look and feel, can be incorporated with success in, e.g., computer games or virtual reality applications. The key features which make plausible wind waves possible in our approach are the linear time complexity, flexible and independent wave effects, and VFC of WTBs. We expect that an implementation of our approach based on, e.g., the SIMD SSE instruction set and multiple core CPUs, would improve the performance by an order of magnitude making the computation of water surfaces of 1024x1024 grid points feasible in real-time. Some computations can also be moved to the GPU for further improvements.

Possible future work includes examining various approaches to add plausible foam, spray and breaking waves in an additional pass over the visible WTBs. Working with dynamically moving WTBs would also be interesting, e.g., to achieve waves generated by boats in motion. Another possibility would be to further accelerate the wave generation process by improving the VFC. For example, this includes examining ways to efficiently clip WTBs only partly inside the frustum.

References

AKENINE-MÖLLER, T., AND HAINES, E. 2002. *Real-Time Rendering (2nd Edition)*. A K Peters, Ltd.

BABOUD, L., AND DÉCORET, X. 2006. Realistic water volumes in real-time. In *Eurographics Workshop on Natural Phenomena*, Eurographics.

CIEUTAT, J. M., GONZATO, J. C., AND GUITTON, P. 2001. A new efficient wave model for maritime training simulator. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*, IEEE Computer Society, Washington, DC, USA, 202.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 23–30.

FOURNIER, A., AND REEVES, W. T. 1986. A simple model of ocean waves. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 75–84.

GONZATO, J.-C., AND SAC, B. L. 1997. A phenomenological model of coastal scenes based on physical considerations. In *8th Eurographics Workshop on Computer Animation and Simulation*, Springer-Verlag, Berlin, Heidelberg, Germany, 137–148.

GRIBB, G., AND HARTMANN, K., 2001. Fast extraction of viewing frustum planes from the world-view-projection matrix.

HE, H., LIU, H., ZENG, F., AND YANG, G. 2005. A way to real-time ocean wave simulation. In *Proceedings of the Computer Graphics, Imaging and Vision: New Trends (CGIV'05)*, IEEE Computer Society, Washington, DC, USA, 409–415.

HINSINGER, D., NEYRET, F., AND CANI, M.-P. 2002. Interactive animation of ocean waves. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 161–166.

HU, Y., VELHO, L., TONG, X., GUO, B., AND SHUM, H. 2004. Realistic, real-time rendering of ocean waves. *Computer Animation and Virtual Worlds. Special Issue on Game Technologies.*.

IGLESIAS, A. 2004. Computer graphics for water modeling and rendering: a survey. *Future Generation Computer Systems 20*, 8, 1355–1374.

ISIDORO, J., VLACHOS, A., AND BRENNAN, C. 2002. Rendering ocean water. In *ShaderX: Vertex and Pixel Shader Tips and Tricks*. Wordware Publishing.

KASS, M. 1991. Height-field fluids for computer graphics. In *WSC '91: Proceedings of the 23rd conference on Winter simulation*, IEEE Computer Society, Washington, DC, USA, 1194–1198.

KRYACHKO, Y. 2005. Using vertex texture displacement for realistic water rendering. In *GPU Gems 2, Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, ch. 18, 283–294.

MIHALEF, V., METAXAS, D., AND SUSSMAN, M. 2004. Animation and control of breaking waves. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 315–324.

PEACHEY, D. R. 1986. Modeling waves and surf. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 65–74.

RESIO, D. T., SWAIL, V. R., JENSEN, R. E., AND CARDONE, V. J. 1999. Wind speed scaling in fully developed seas. *Journal Of Physical Oceanography 29*, 1801–1811.

SCHNEIDER, J., AND WESTERMANN, R. 2001. Towards real-time visual simulation of water surfaces. In *VMV*, 211–218.

SEMTNER, A. 2000. Ocean and climate modeling. *Communications of the ACM 43*, 4, 80–89.

STAM, J. 1999. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128.

STAM, J. 2003. Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference*.

TESSENDORF, J. 2004. Simulating ocean water. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, ACM Press, New York, NY, USA.

THON, S., DISCHLER, J.-M., AND GHAZANFARPOUR, D. 2000. Ocean waves synthesis using a spectrum-based turbulence function. In *CGI '00: Proceedings of the International Conference on Computer Graphics*, IEEE Computer Society, Washington, DC, USA, 65.

TS'O, P. Y., AND BARSKY, B. A. 1987. Modeling and rendering waves: wave-tracing using beta-splines and reflective and refractive texture mapping. *ACM Trans. Graph. 6*, 3, 191–214.