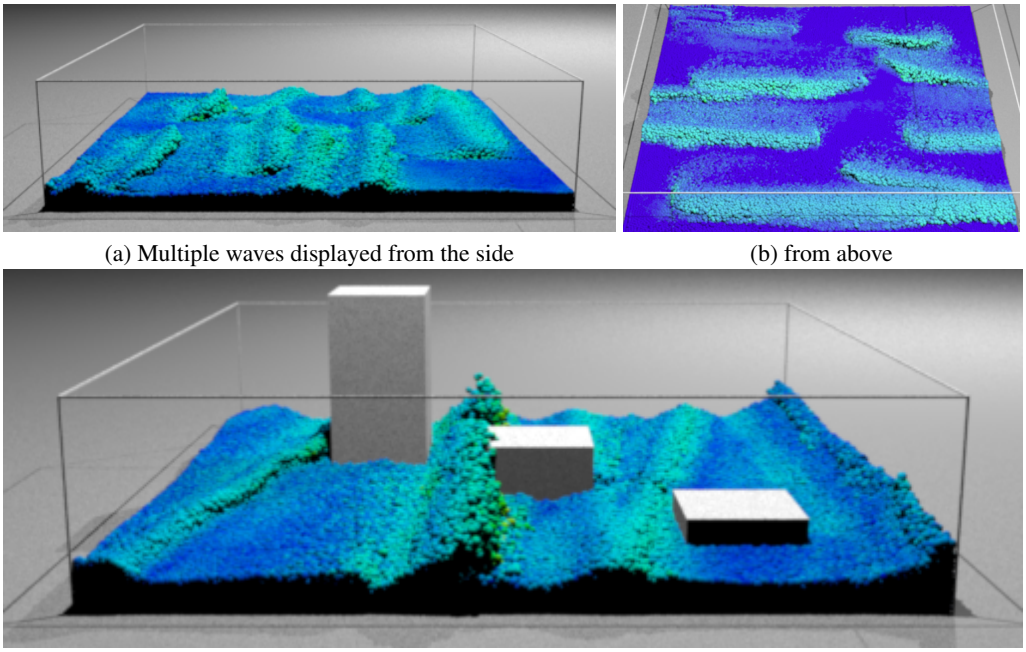
Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2015)
F. Jaillet, G. Zachmann, and F. Zara (Editors)

A New Force Model for Controllable Breaking Waves

Mathias Brousset [1] Emmanuelle Darles [1] Daniel Meneveaux [1] Pierre Poulin [2] Benoît Crespin [3]

[1] XLIM, UMR CNRS 7252, Université de Poitiers [2] Dep. I.R.O., Université de Montréal [3] XLIM, UMR CNRS 7252, Université de Limoges



(a) Multiple waves displayed from the side (b) from above

(c) Waves interacting with static blocks

Figure 1: Breaking waves obtained with our model. An ocean scene with multiple waves is displayed (a) from the side and (b) from above; (c) and breaking waves and backflow waves colliding with blocks. The waves combine naturally, while each of them is modeled with its specific parameters (height, width, speed, orientation, crest slope, breaking time).

Abstract

This paper presents a new method for controlling swells and breaking waves using fluid solvers. With conventional approaches that generate waves by pushing particles with oscillating planes, the resulting waves cannot be controlled easily, and breaking waves are even more difficult to obtain in practice. Instead, we propose to use a new wave model that physically describes the behavior of wave forces. We show that mapping those forces to particles produces various types of waves that can be controlled by the user with only a few parameters. Our method is based on a 2D representation that describes wave speed, width, and height. It handles many swell and wave configurations, with various breaking situations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.3.8 [Computer Graphics]: Application—J.2 [Computer Applications]: Physical Sciences and Engineering—Earth and atmospheric sciences

DOI: 10.2312/vriphys.20151334

1. Introduction

Over the past decade, fluid simulation has spawned major interest in the computer graphics community. Motions of many fluid types (e.g., water, fire, smoke) have been simulated successfully with physically based methods, mostly derived from the Navier-Stokes equations. Whether based on Eulerian or Lagrangian solutions, these methods can deliver accuracy and realism, but they often suffer from long computation times, large memory requirements, and very little control during computations.

Waves over a large body of water capture an essential part of our vision of any coastline, but initiating conditions in existing solvers in order to generate the expected waves is a daunting task. However, creating realistic water simulations without these solvers is without doubt as difficult. We seek to provide controls to more easily create different waves in a physically based solver, thus benefiting from its power, yet providing more artistic freedom.

Waves are examples of turbulent fluid phenomena that have often been used as illustrative demonstrations. In real fluid demonstrations, waves are usually produced with a mechanical oscillating panel. In physically based methods, they are therefore generated mostly with the help of a similarly oscillating plane [BT07, SR09, MM13], and thus the fine control of waves remains difficult to achieve in practice.

In fact, few papers address how to simulate physically complex natural phenomena, while still offering artificial or artistic control with intuitive user interactions or parameterizations. One rare exception for waves comes from Mihalef et al. [MMS04], who control waves using a library of 2D breaking waves. In their system, the user selects a set of 2D slices to produce a 3D animation of a wave in an Eulerian solver. Since modelling and controlling the fluid depend on a library of 2D slices and their physical properties, the manipulation cannot be accomplished during the simulation, which can strongly interfere with the artist's expression. Moreover, as the slices store velocity vector fields, the artist also loses much design freedom. In another work worth mentioning, Radovitzky and Ortiz [RO98] develop a 2D finite Laitone solution that represents a breaking wave using hyperbolic periodic functions, following the shape of plunging waves. Unfortunately, their model requires several parameters, and controlling the generated waves during simulation is impossible.

From the game industry angle, we should point out that some solutions offer high-level controls for the shape of a single wave, in the form of Bezier curve attractors. They can produce the general shape of a wave, but do not handle breaking waves, nor do they integrate well in a physical fluid simulator for the important secondary fluid effects.

Our proposed representation accounts for high-level user control, enabling the propagation of different configurations of waves, as shown in Figure 1. While we have applied our model to a smoothed particle hydrodynamics (SPH) solver, it should also extend naturally to an Eulerian solver. More specifically, our main contributions are:

- a simple and physically based wave model that can represent choppy, linear, or breaking waves with ease of implementation, i.e., as a straightforward additional body force;
- the ability to easily control the shape and behavior of generated waves for artistic expression, thanks to its few parameters;
- an interactive user control of waves offered through a simple yet intuitive graphical user interface (GUI).

The paper is organized as follows. Section 2 reviews existing methods for wave simulation in computer graphics. Section 3 introduces our wave model, together with its parameters to control swells and breaking waves. Section 4 describes our implementation and presents various results. Finally, Section 5 gives our conclusions and future work.

2. Related Work

Over the past two decades, computer generated ocean waves have drawn the attention of several authors [FR86, Tes99, MMS04, DCGG11, NSB13]. They can be classified in two main families: on the one hand, procedural or spectral methods that characterize water height according to time; on the other hand, simulation methods that aim at solving Navier-Stokes (NS) equations in 2D or 3D.

2.1. Procedural and Spectral Wave Simulation

Procedural or spectral methods are often used to model waves for oceans of infinite depth, where the water surface is mostly represented as a heightfield. These methods are common because they are very fast to compute and visually appealing, although they do not correspond to physically based models. Some models [FR86, Pea86] use parametric equations to derive a surface to approximate swells and breaking waves.

Tessendorf [Tes99] animates a heightfield using fast Fourier transforms (FFTs), to which Bruneton et al. [BNH10] add levels of detail to more realistically render the ocean surface in real time. Because tuning parameters for these methods is usually more difficult, Thon and Ghazanfarpour [TG02] propose to use real-world measurements as heightfields, and to add Perlin noise to reduce repetitive visual artifacts. The main limitation of these methods is that they cannot represent breaking waves, since a heightfield has only one height value z for each horizontal position (x, y). In addition, these methods are difficult to tune for handling interactions with solid objects.

2.2. Navier-Stokes-based Wave Simulation

A large body of work in computer graphics has focused on computational fluid dynamics (CFD) for liquids, based

on Navier-Stokes partial differential equations (NSE). Numerical solutions usually make use of an Eulerian description with finite differences to approximate a solution [HW65, FM96], with particles and a level-set to track the fluid surface and reduce compressibility [Sta99, FF01, FSJ01, EMF02]. A Lagrangian approach considers the fluid as a set of particles and computes interaction between them to approximate a solution [MCG03, BT07, SR09, PTB*03, IOS*14].

From a library of 2D simulated wave slices, Mihalef et al. [MMS04] generate 3D waves. Their method consists in combining a series of 2D slices to model the new wave at given times; velocities between slices are linearly interpolated. In their framework, each slice corresponds to a snapshot of a Eulerian 2D simulation with its associated vector field. Starting from an initial 3D geometry which can be shaped as a swell or a breaking wave, the targeted wave is eventually obtained during the simulation.

With shallow water equations, based on a simplified version of NSE, Thurey et al. [TMFSG07] simulate breaking waves at interactive framerates with a 2D plane. The wave height and propagation speed can be controlled, and breaking waves are completed using an additional mesh. However, interactions between water volumes and solid objects cannot be physically handled.

Based on oceanography studies, the velocity field of a 2D NSE simulation can be initialized by a combination of hyperbolic functions that represent a solitary breaking wave [RO98]. The idea is to combine horizontal and vertical descriptions of a nonlinear wave [BGH*04], using hyperbolic secant and hyperbolic tangent functions, i.e., $y = H\,\mathrm{sech}(x-\omega t)\tanh(x-\omega t)$, where H is the wave height, and ω denotes the pulsation, encoding the propagation speed in the x direction. Unfortunately, wave control remains subtle, and the resulting waves are only valid in shallow water. In addition, the resulting waves only break when the ground rises, which limits their applicability to the relief of an ocean floor. Moreover, these waves propagate only along the 2D plane corresponding to the propagation direction. Finally, because their interaction requires to solve the Korteweg-de Vrie partial differential equation [KdV95], they remain too costly for interactive applications.

Darles et al. [DCG11] extend this model for 3D breaking waves by adding a procedural force to a multiscale SPH solver. This method produces various types of plunging and surging breaking waves of varying height, but it depends highly on the fluid depth, and breaking is controlled by the ground geometry only.

This paper proposes a new method that reduces many of these limitations, allowing an artist to create several configurations of breaking waves with few intuitive parameters. Our model can be employed to produce propagating and interacting waves in various directions. It can also generate and control swells, independent of the depth. It corresponds to an extension and a simplification of the model proposed by Darles et al. [DCG11], based on highly controllable additional forces that can be easily combined in a 3D fluid simulation system.

2.3. Governing Equations and SPH Method

We use the Lagrangian form of the NS equations, which describes the flow of a fluid represented as particles. The Lagrangian form of the momentum NS equation stands as follows:

$$\frac{d\vec{v}_i}{dt} = -\frac{1}{\rho_i}\nabla p_i + \mu_i \nabla^2 \vec{v}_i + \frac{\vec{F}_i^{ext}}{\rho_i} \quad (1)$$

$$\frac{d\vec{x}_i}{dt} = \vec{v}_i \quad (2)$$

where for a given particle i, $\vec{x}_i$ corresponds to the position, $\vec{v}_i$ the velocity, ρ_i the particle density, p_i the pressure, μ_i the dynamic viscosity, and $\vec{F}_i^{ext}$ external forces. Equation (1) states, in the Lagrangian case, that the acceleration of a particle i at each time step depends on a sum of internal forces (pressure and viscosity) and external forces (gravity, and in our case, wave forces as will be described in Section 3).

Based on this formulation, smoothed particle hydrodynamics (SPH) have been widely used in computational fluid dynamics for numerically approximating the differential terms of Equation (1). Each quantity A_i for a given particle i is interpolated using its j neighbors. The basic SPH formulation [Mon92, Mon05] is:

$$A_i = \sum_j \frac{m_j}{\rho_j} A_j W(\vec{x}_i - \vec{x}_j, h) \quad (3)$$

where m_j represents the mass of particle j, h is the maximal interaction distance between two particles, and W is a smooth interpolation distance function. Symmetric derivative formulations can be used to approximate gradient and Laplacian of a field A, as well as pressure gradient and velocity Laplacian, to approximate a solution of Equation (1) [IOS*14].

3. A New Generic Wave Model

Our wave model simplifies and generalizes the soliton representation proposed by Darles et al. [DCG11]. It relies on the addition of new forces to fluid particles, with parameters that control swells and breaking waves in a plane with varying height, speed, breaking time, and breaking duration. We also show how this model can be extended to various configurations, including orientation changes, crest variations, and waves crossing each other.

Figure 2: Four steps of a breaking wave: (a) shoaling, (b) propagation, (c) steepening, and (d) breaking.

3.1. Our 2D Nonlinear Wave Forces

From swells to breaking waves, four animation steps can be distinguished [Kel48] (see Figure 2): (a) the fluid rises and a wave comes out; (b) the wave propagates and the corresponding swell may vary according to certain conditions, such as water depth or wind; (c) steepening begins before breaking, due for example to a higher ocean floor or stream variations; and (d) the wave breaks since particles on the crests have a higher speed than the others.

During the swell and propagation of waves, a periodical motion of the surface can be observed, in which water particles follow an elliptical path. According to the Airy theory in finite depth [WJ10], the vertical speed of a particle increases as the particle gets closer to the water surface. More precisely, when the wave moves forward, each particle is subject to a vertical force that raises it, and to a horizontal force that slightly pushes it in the wave direction. When the wave passes the particle, the vertical force is released and the water backflow pushes the particle backward while it goes down.

Our model approaches these effects using external forces within a fluid simulator. It relies on a horizontal component, providing the wave motion, and a vertical component, driving the wave elevation. The wave propagation is given by a user-defined wave speed ω; the shoaling step consists in initializing all the horizontal and vertical forces at $t = 0$. We define the wave force acting on a particle i with the following formulation, using hyperbolic secant and tangent functions:

$$F_{x_i} = -\sqrt{g\,d}\,\mathrm{sech}(A\,d_{z_i})\tanh(A\,d_{z_i})\lambda_x \quad (4)$$

$$F_{z_i} = \begin{cases} H\,\mathrm{sech}^2(A(x_i-\omega t-x_{shoal}))\lambda_z & \text{if } x_i < x_{break} \\ 0 \quad \text{otherwise} \end{cases} \quad (5)$$

where $A = \sqrt{3H/4d^3}$; H is the user-defined wave height in meters, t is the current time step, ω is the propagation velocity of the wave, $d_{z_i} = z_i - z_i^0$ is the relative depth of particle i, z_i^0 is the initial depth of the particle, and λ_x and λ_z are two user-defined parameters that control the shape of breaking waves; $\sqrt{g\,d}$ is an attenuation term proportional to gravity [RO98], and ωt (Equation (5)) defines the wave position.

Figure 3 shows the horizontal force (Equation (4)) applied to a given particle; the x coordinate in the plot represents the particle depth d_z. These curves reach their maximum when the particle is close to the wave crest. Figure 4 shows the vertical force (Equation (5)). This function reaches its maximum when the particle is near the shoaling point, and it is canceled when the particle reaches the breaking point.

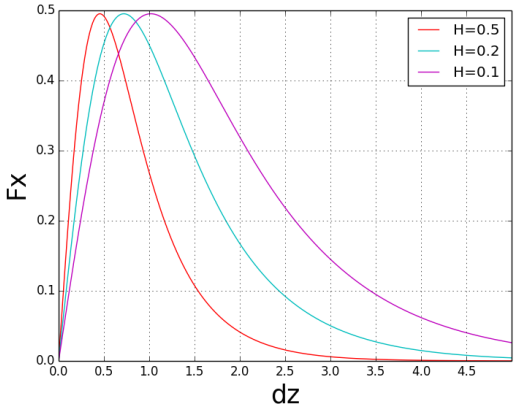


Figure 3: Representation of the horizontal component of the wave force F_x acting on a particle according to its height d_z for three values of H.

Each step of the wave's life can be distinguished in our formulation:

1. Shoaling
The shoaling point x_{shoal} locates the *origin* of a wave; the forces are initialized at $t = 0$, and begin at position x_{shoal}. The vertical force is relatively large in order to raise the water volume (blue part of the curve in Figure 4 (top)).

2. Swell propagation
Horizontally, when the wave reaches a particle at position x, fluid particles are pushed, providing a horizontal elliptic motion described in the Airy theory [BGH*04], thanks to Equation (4). The closer a particle is to the water surface, the stronger is the force applied to the particle. The particle's backward motion is due to the SPH simulation backflow after the wave moves away. Vertically, the force increases progressively to generate the particle ascent (vertical elliptic motion), provided in our model by the combination of hyperbolic tangent and secant that linearly decreases according to the particle's depth (Equation (5)).

3. Steepening
Steepening corresponds to an increase of the vertical component of the wave force when the horizontal coordinate particle position x is close to x_{shoal}. When the horizontal component of the force becomes larger for crest particles, the wave begins to steepen; the breaking stage can be initiated at any moment (green part of the curve in Figure 4 (top)).

4. Breaking
Breaking is due to the force discontinuity in our model: the crest's horizontal speed is larger than the water particles located below, thus resulting in their fall. This discontinuity is modeled by canceling the vertical force (red part of the curve in Figure 4 (top)).

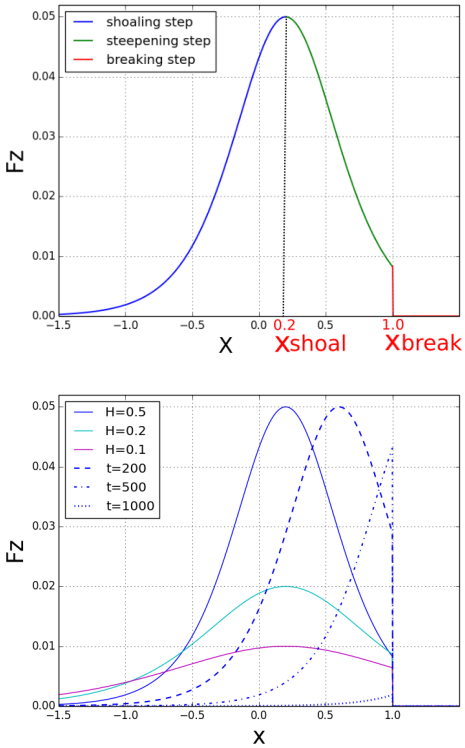


Figure 4: Top: Representation of the vertical component of the wave force F_z acting on a particle with $H = 0.5$, $\omega = 0.002$, $t = 0$, and $\lambda_z = 0.1$. Bottom: Representation of the vertical component of the wave force F_z for three values of H at three times t.

Discussion

The curves in Figure 3 illustrate the horizontal force component for three values of H. At low values, the horizontal wave force reaches a maximum in a larger interval of particle depths d_z, horizontally pushing deep fluid particles (i.e., not only those located near the surface), and thus propagating swell waves. At larger values of H, the horizontal wave force reaches a maximum in a smaller interval of d_z, pushing particles located near the surface and producing breaking waves.

The curves of the plot in Figure 4 (bottom) show the evolution of F_z for different values of H and t. Firstly, the vertical wave force magnitude varies according to the value of H: with low values, the magnitude is low as well as its variation between steepening and breaking steps. In this configuration, particles are not located high enough to simulate a breaking wave and their motion is mainly driven by the horizontal component of the wave's force. The simulation of breaking waves requires sufficiently large values of wave height H. In addition, the lowering of the vertical wave force (which models the shoaling and steepening steps) varies according to time. The larger the value of t, the shorter the decreasing part of the curve (steepening). For low values of t, this phase allows wave particles to rise and to produce a breaking wave.

With this representation, the horizontal component of the force is not correlated to the vertical component. It is thus possible to produce low-amplitude waves corresponding to swells as well as breaking waves, as will be shown in the next sections. The expression related to the vertical force component guarantees a linear growth of the wave, that can be controlled over space and time, using parameters H, ω, λ_x, and λ_z.

3.2. Control Parameters

This section describes how the parameters of our model can be used to control local motion of waves, as well as their global shape. They include local parameters such as individual wave height H, pulsation ω, shoaling and breaking points x_{shoal} and x_{break}. Scaling forces using λ_x and λ_z (considered as additional global parameters) produce different types of waves.

Wave Height (H)

The wave height affects the amplitude of a wave, and thus its shape, derived from Equation (4). For instance, swell is obtained with $H < 0.2$, while larger values of H produce breaking waves, as illustrated in Figures 5 and 6.

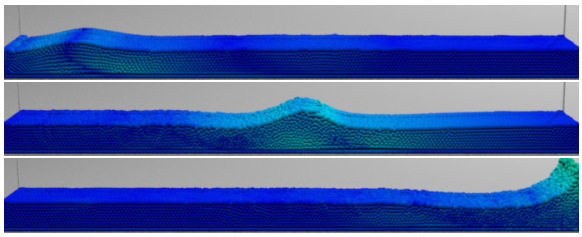


Figure 5: The shoaling and propagation of a simple swell before it interacts with the wall on the right ($H = 0.22$, $\omega = 0.002$, $d = 0.2$, $\lambda_x = 1.0$, $\lambda_z = 32.5$).

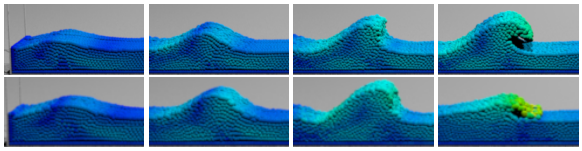


Figure 6: Top: the four typical steps of breaking waves: shoaling, swell, steepening, and breaking ($H = 0.24$, $\omega = 0.002$, $d = 0.2$, $\lambda_x = 1.0$, $\lambda_z = 32.5$). Bottom: breaking wave obtained using a linearly decreasing value of H down to 0.0.

Wave Pulsation (ω)
Wave pulsation controls the propagation speed of a wave and the steepening duration (see Figure 7). The longer this phase, the higher the particles in the breaking wave. With this parameter, it is possible to control the shape and the type of the simulated wave. For instance, low values ($\omega < 0.002$) produce plunging breaking waves, intermediate values ($0.002 < \omega < 0.003$) produce surging breaking waves (i.e., plunging without pipes), and larger values of ω result in swells or choppy waves, mainly present in deep waters. Combining waves with different values of ω generate nonuniform motions with crossing waves, that can also be observed on beaches.

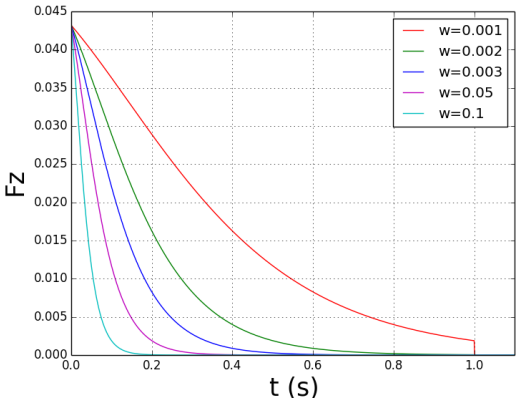


Figure 7: Duration of the steepening step of the vertical component wave force for five values of ω evaluated on a particle located at $x = 0$ with $H = 0.5$, $x_{shoal} = 0.2$, and $x_{break} = 1.0$.

Scaling Wave Force (λ_x and λ_z)
These parameters control globally the duration of a wave phase, acting independently in the x and z directions. Thus shoaling and breaking can be affected with a change of λ_z over time, $\lambda_z \gg \lambda_x$ emphasizing breaking. Similarly, propagation speed and direction are affected with a change of λ_x over time.

Shoaling and Breaking Points (x_{shoal} and x_{break})
These two points let the user define the position for the starting of the swell (x_{shoal}) and the position where the wave breaks (x_{break}).

3.3. Extension to 3D Waves

The wave model described in the previous sections is defined for a propagation along the x-axis, z corresponding to the wave height. In this section, we provide some examples of derivations for a more general propagation in the xy plane, providing a wide variety of configurations.

Wave Orientation
With a given propagation direction θ in the xy plane, the corresponding rotation can be directly applied to the horizontal force component F_x and to the corresponding particles, providing waves propagating in any direction.

Crest Variations
Another example of wave tuning consists in giving several values of H so that the crest height varies. With our GUI, the user draws a polyline that is mapped onto H values during the simulation, as shown in Figure 8.

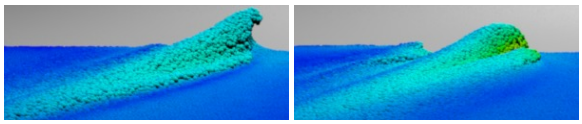


Figure 8: Rotating particles to propagate the wave along the desired direction. The breaking wave has been given a 15-degree propagation angle, with an oblique crest, and $H_{max} = 0.4$, $\omega = 0.002$, $d = 0.25$, $\lambda_x = 1.0$, and $\lambda_z = 32.5$.

Interactions between Waves
As stated earlier in Section 2, interactions between several waves require solving the KdV equation [KdV95], which cannot be accomplished within an interactive context. We propose here a simple alternative, which provides simulation results very close to observed phenomena.

The user can generate as many waves as desired, each wave j with its own propagation parameters (H_j, ω_j, θ_j, λ_{x_j}, λ_{y_j}). When two waves overlap, summing wave forces does not correspond to actual wave crossing in the general case. Instead, we propose to use the maximum wave function for each particle i, i.e., $F_{x_i} = \max(F^1_{x_i}, ..., F^N_{x_i})$ and $F_{z_i} = \max(F^1_{z_i}, ..., F^N_{z_i})$ for N simulated waves. We have produced two facing waves using moving walls, and compared their traversing each other with our wave model (Figure 9). We can observe similar visual behaviors.

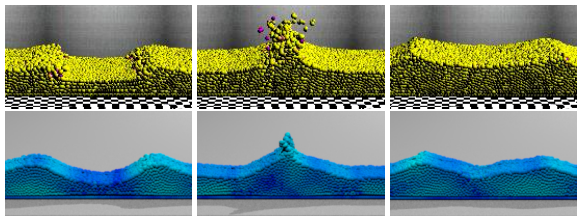


Figure 9: Top: Two facing waves produced using a moving wall. Bottom: Same configuration using our model.

Generation and Control of Waves
The user creates a new instance of a wave in our simulation system with only a few initial settings: a starting location and time t, and a propagation direction. As the wave rises

from time t, its parameters can be modified (even interactively during the simulation) through editable polylines acting as time-dependent scaling factors. For instance, a polyline increasing from 0 to 1 during the shoaling stage produces a smoothly rising wave. This process is illustrated in Figure 10. Each wave has its own set of parameters and can be replayed as many times as needed. Once the user is satisfied with the result, the corresponding parameters and their polylines can be exported for later use. The simulations illustrated in this paper have been produced with this system.

4. Results

This section illustrates results produced with our wave model, including shoaling, propagation, steepening, and breaking. With our system, several scenarii can lead to interacting waves: cross sea (waves meeting at oblique angles), waves meeting from opposite directions, and a wave overtaking another one. We demonstrate its efficiency within a SPH simulation system, with several wave configurations and we discuss the performance of our system.

4.1. Parameter Values for Different Waves

Figure 10 illustrates our GUI, used to control each parameter (value and editable curve). For instance, the user can control the wave height during shoaling progress, with a smoothly increasing curve. Each parameter can be controlled the same way with our user interface, including swell speed or crest inclination, for instance.

With our model, waves are controlled in various ways for providing several shapes, speeds, or breaking shapes, from swells to collapsing waves.

Symmetrically Crossing Waves
Our model can represent many interacting crossing waves with visual plausibility. Figure 11 shows an example of two symmetrical waves, which is a common phenomenon in deep waters. As we can see from this sequence, the interaction of two waves that propagate in opposite directions creates a typical shock wave; the two waves continue afterward to propagate with their own distinct parameters. With our model, this phenomenon can be represented simply by acting on the λ_x parameter and using a positive and a negative value to represent opposite propagation directions in our wave interaction system.

Obliquely Crossing Waves
Figure 12 shows an example with oblique waves corresponding to a cross sea, a rare phenomenon observable in deep waters. This effect can be simulated using the wave orientation parameter θ with two different orientations. The interaction of multiple oblique waves produces the result illustrated in Figure 13.

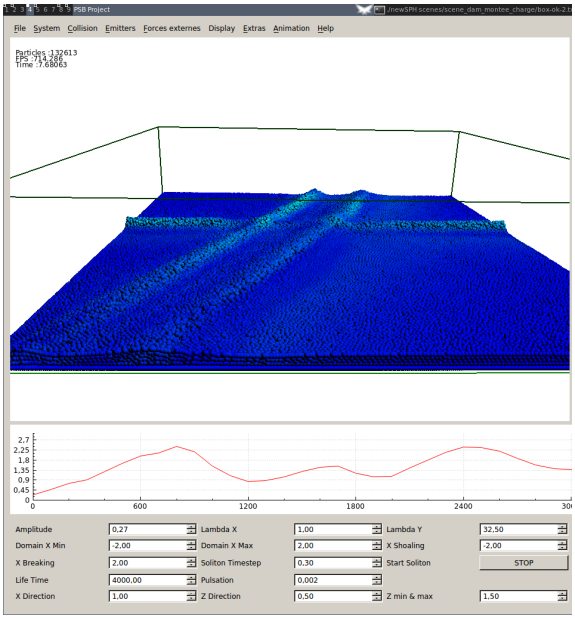


Figure 10: Screenshot of our OpenGL interface with its complete graphical user interface. The user can draw and edit one curve for each model parameter to control the wave behavior over time. Curves and values can even be changed during simulation.

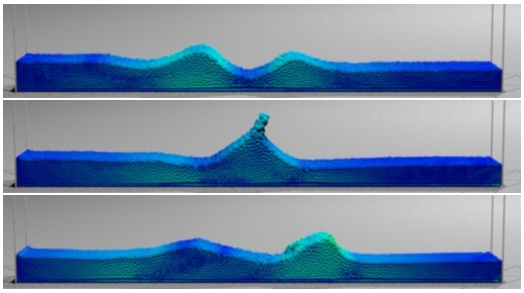


Figure 11: Two waves crossing each other with the same opposite speed $\omega = 0.002$. The wave originating from the left is higher ($H = 0.27$) than the one from the right ($H = 0.23$).

Overtaking Waves
The sequence illustrated in Figure 14 corresponds to a wave that overtakes another one, i.e., with higher speed propagation ω. The interaction of these two waves produces a temporary unique wave, separated later on into the two original waves that continue to propagate according to their respective speeds, as can be observed with real waves.

Multiple Waves
We have combined various types of waves, including different speeds, heights, and crests, and interacting with objects (interactions are naturally handled by the SPH solver).

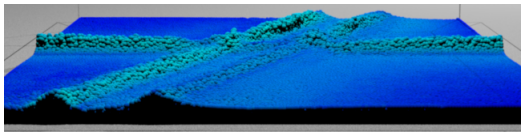


Figure 12: An oriented wave obliquely crossing with two others. The two parallel waves have been given a 45-degree propagation direction, with $H = 0.26$, $d = 0.25$, and $\omega = 0.002$.

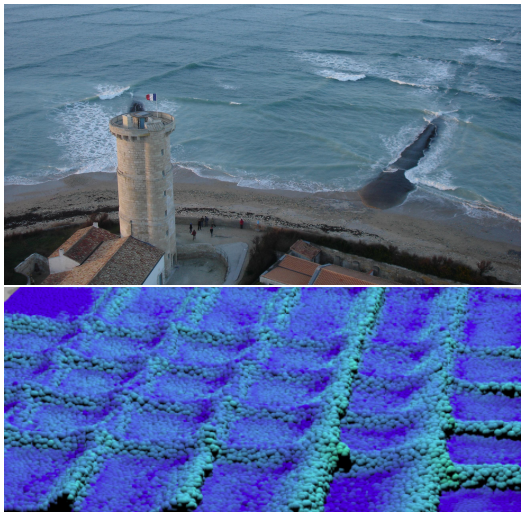


Figure 13: Top: Behavior of wave crossings in reality, image courtesy of Michel Griffon. Bottom: Simulation using our model with a similar behavior, with $H = 0.23$, $\omega = 0.002$, $d = 0.25$, $\lambda_x = 1.0$, and $\lambda_z = 32.5$.

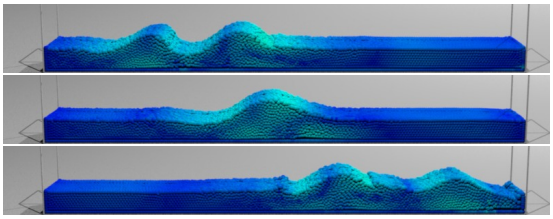


Figure 14: A faster and smaller wave ($H = 0.23$, $\omega = 0.003$) overtakes a slower and higher one ($\omega = 0.002$, $H = 0.27$), with $d = 0.25$ for both waves.

Figure 1 illustrates some of the configurations that we have defined; Figure 15 shows two frames of the simulation without blocks; Figure 16 uses another configuration of several waves, with additional interactions with blocks.

A video showing all of our results is available at https://vimeo.com/133911694.

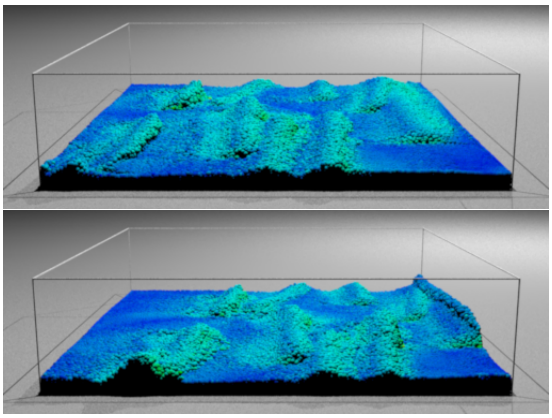


Figure 15: Two frames of multiple waves and interactions with limited width and several directions.

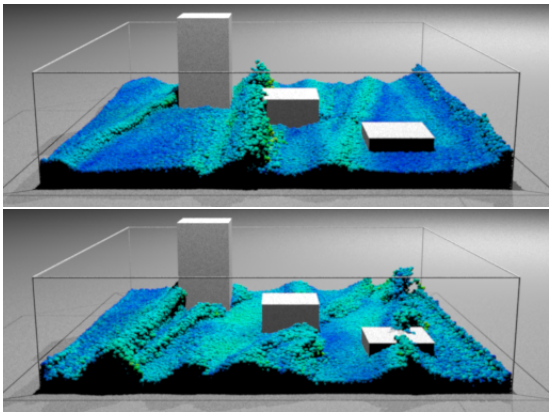


Figure 16: Two frames of multiple waves and interactions with static blocks.

4.2. Performance and Discussion

The results provided in this paper have been produced on an Intel Xeon E2620 2.4GHz processor with 16GB of RAM. SPH computations run on the GPU (GTX Titan) using CUDA 6.5, including computations of wave forces. Particle neighbors are identified using a uniform grid, as described by Ihmsen et al. [IABT11]. The interactive simulation system uses 10 iterations per rendered frame ($\Delta t = 0.001$ sec.), as shown in Table 2; particle density and pressure are handled using the formulation introduced by Ihmsen et al. [IOS*14].

Step	Swell Figure 5	Plunging Figure 6	Opposite Figure 11	Cross sea Figure 12
Neighborhood search	47.89%	48.79%	48.97%	61.48%
Internal forces evaluation	41.41%	40.54%	40.58%	31.26%
Wave forces evaluation	0.94%	0.99%	1.7%	0.86%
Integration	0.82%	0.8%	0.83%	0.49%
Memory management	8.94%	8.88%	7.92%	5.91%

Table 1: Proportion of computation times per CUDA kernel for all our examples.

Scene	Particles	Time per frame
Short width (Figure 6)	30k	0.013s
Long width (Figure 12)	100k	0.022s
Large depth (Figure 8)	255k	0.076s

Table 2: Average computation time per frame for our scenes.

Our wave model is implemented as an additional body force within a fluid solver. The simulation is displayed interactively with OpenGL, and the Mitsuba renderer [Wen10] was used for offline rendering of the particles in all figures provided in the paper. The particles are colored according to their velocity to highlight the application of our external force. As detailed in Tables 1 and 2, the computational cost corresponding to the additional forces of our model is about 1% of the entire computation time, which keeps our system interactive even with less powerful computers and GPUs. In fact, the overall system performance essentially depends on the computations of fluid dynamics, and on the number of particles, rather than on our external force model.

Our model produces various types of ocean waves, given the adequate bounds for each parameter. Note that parameters are correlated; for instance, changing the wave speed ω also influences its height.

$H > 0.5$ produces unrealistic results because the resulting vertical force is too large and particles explode. Parameter ω is responsible for the propagation speed and the type of the wave: swell, choppy, plunging or surging breaking waves. When $\omega > 0.1$, forces result in unrealistic waves that travel too fast.

5. Conclusion and Future Work

Our simple wave force model produces a wide variety of phenomena. It makes it possible to generate swell, choppy, or breaking waves that can interact with each other. We have shown that with the adequate control of each parameter, many different scenarios can be modeled, for example, a swell wave that suddenly turns into a breaking wave, or different waves that interact with each others. The list of these effects is not exhaustive and the creative possibilities are rich. As our model is defined as an external force, it may be used in any type of SPH solver (i.e., WCSPH [BT07], PCI-SPH [SR09], IISPH [ICS*14]), and also in 3D Eulerian [EMF02] or hybrid solvers [CIPT14].

Validating crossing wavesas well as other wave phenomena remain difficult because of the lack of measured data with real fluids. In this paper we have shown examples that visually assess our simple model, but given more real data, or specific observations, we would very much like to better validate our model.

In the future, we aim at reducing some limitations concerning parameter control. For instance, wave speed affects the wave shape. It could be interesting to propose mapping functions that would control some higher-level parameters, such as speed, wave height, or breaking duration. We would also like to add fine-scale details, such as foam and sprays, and use our generic wave model to localize areas that generate these effects.

Acknowledgements

The work was financed by the Poitou-Charentes region and the MIRES federation. Mathias Brousset and Pierre Poulin were also supported in part by GRAND.

References

[BGH*04] BALDWIN D., GÖKTAŞ Ü., HEREMAN W., HONG L., MARTINO R., MILLER J.: Symbolic computation of exact solutions expressible in hyperbolic and elliptic functions for nonlinear pdes. *Journal of Symbolic Computation 37*, 6 (2004), 669–705. 3, 4

[BNH10] BRUNETON E., NEYRET F., HOLZSCHUCH N.: Real-time realistic ocean lighting using seamless transitions from geometry to brdf. *Computer Graphics Forum 29*, 2 (2010), 487–496. 2

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *SIGGRAPH/Eurographics SCA* (2007), pp. 209–217. 2, 3, 9

[CIPT14] CORNELIS J., IHMSEN M., PEER A., TESCHNER M.: Iisph-flip for incompressible fluids. *Computer Graphics Forum 33*, 2 (2014), 255–262. 9

[DCG11] DARLES E., CRESPIN B., GHAZANFARPOUR D.: A particle-based method for large-scale breaking wave simulation. *Machine Graphics & Vision 20*, 1 (2011), 3–25. 3

[DCGG11] DARLES E., CRESPIN B., GHAZANFARPOUR D., GONZATO J.-C.: A survey of ocean simulation and rendering techniques in computer graphics. *Computer Graphics Forum 30*, 1 (2011), 43–60. 2

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. In *SIGGRAPH* (2002), pp. 736–744. 3, 9

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *SIGGRAPH* (2001), pp. 23–30. 3

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483. 3

[FR86] FOURNIER A., REEVES W.-T.: A simple model of ocean waves. In *SIGGRAPH* (1986), pp. 75–84. 2

[FSJ01] FEDKIW R., STAM J., JENSEN H.: Visual simulation of smoke. In *SIGGRAPH* (2001), pp. 15–22. 3

[HW65] HARLOW F., WELCH J.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids 8*, 12 (1965), 2182–2189. 3

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A parallel sph implementation on multi-core cpus. *Computer Graphics Forum 30*, 1 (2011), 99–112. 8

[ICS*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 426–435. 9

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: Sph fluids in computer graphics. In *Eurographics 2014 - State of the Art Reports* (2014). 3, 8

[KdV95] KORTEWEG D. J., DE VRIES G.: On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. *Philosophical Magazine Series 5 39*, 240 (1895), 422–443. 3, 6

[Kel48] KELLER J.: The solitary wave and periodic waves in shallow water. *Communications on Pure and Applied Mathematics 1*, 4 (1948), 323–339. 4

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *SIGGRAPH/Eurographics SCA* (2003), pp. 154–159. 3

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 104. 2

[MMS04] MIHALEF V., METAXAS D., SUSSMAN M.: Animation and control of breaking waves. In *SIGGRAPH/Eurographics SCA* (2004), pp. 315–324. 2, 3

[Mon92] MONAGHAN J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30* (1992), 543. 3

[Mon05] MONAGHAN J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics 68*, 8 (2005), 1703–1759. 3

[NSB13] NIELSEN M. B., SÖDERSTRÖM A., BRIDSON R.: Synthesizing waves from animated height fields. *ACM Trans. Graph. 32*, 1 (2013), 1–9. 2

[Pea86] PEACHEY D. R.: Modeling waves and surf. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (1986), SIGGRAPH '86, ACM, pp. 65–74. 2

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R.: Particle-based simulation of fluids. *Computer Graphics Forum 22*, 3 (2003), 401–410. 3

[RO98] RADOVITZKY R., ORTIZ O.: Lagrangian finite element analysis of newtonian fluid flows. *International Journal for Numerical Methods in Engineering 43*, 4 (1998), 607–619. 2, 3, 4

[SR09] SOLENTHALER B., R. P.: Predictive-corrective incompressible sph. In *Proceedings of the 36th Annual Conference on Computer Graphics and Interactive Techniques* (2009), SIGGRAPH '09, ACM, pp. 40:1–40:6. 2, 3, 9

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH* (1999), pp. 121–128. 3

[Tes99] TESSENDORF J.: Simulating ocean water. In *SIGGRAPH Course Notes* (1999). 2

[TG02] THON S., GHAZANFARPOUR D.: Ocean waves synthesis and animation using real world information. In *Computers and Graphics* (2002), vol. 26, pp. 99–108. 2

[TMFSG07] THUREY N., MÜLLER-FISCHER M., SCHIRM S., GROSS M.: Real-time breakingwaves for shallow water simulations. In *Pacific Conference on Computer Graphics and Applications* (2007), pp. 39–46. 3

[Wen10] WENZEL J.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 9

[WJ10] WIEGEL R., JOHNSON J.: Elements of wave theory. *Coastal Engineering Proceedings 1*, 1 (2010), 2. 4