Simulation and Control of Breaking Waves using an External Force Model

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

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

Abstract
This paper presents an extension of a paper initially presented at the conference VRIPHYS 2015 [1]. It describes a new method for intuitively managing swells and breaking waves within fluid solvers, based on an external force controlled with only a few parameters. Generating waves with conventional approaches requires pushing particles with oscillating planes. As the resulting waves are only handled by the fluid simulator, they cannot be controlled easily; breaking waves are also difficult to produce in practice. Instead, we propose to use a new wave model that physically describes the behavior of “wave forces” with parameters that explicitly affect wave speed, height, and width. We also propose to map each parameter with user-defined curves. As shown in the results, these forces applied to a smoothed-particle hydrodynamics (SPH) system provide a wide range of effects, such as swells with varying speed and height, breaking waves, curved wave fronts, overtaking waves, and curved wave paths. We show how it is possible to handle crossing waves with our model. Each effect demonstrated in our results only requires a few easy-to-implement operations.

Keywords: Physics-based animation, Fluids, SPH, User control

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 fluid 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 fluid 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. They are usually produced with an oscillating panel for both real fluid demonstrations and physically based simulation methods [2, 3, 4]. In such a configuration, a fine control of waves remains difficult to achieve in practice.

Only a few papers address how to simulate such 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. [5], who controls waves using a library of 2D breaking waves. In their system, a 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 an artistic expression. Moreover, as the slices store velocity vector fields, the artist also loses much design freedom.

In another work worth mentioning, Radovitzky and Ortiz [6] 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 simula-
tion is impossible.

From the game industry perspective, 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 important secondary fluid effects.

This paper proposes a representation based on an external “wave force” that can be controlled by the user, relying on intuitive parameters such as wave speed, height, width, and orientation. In this extended version of our original paper [1], we develop a 2D control of waves for creating curved wave fronts and wave paths. We have applied our external force model to a smoothed-particle hydrodynamics (SPH) solver. Some results obtained with our approach are shown in Figure 1. More specifically, our main contributions are:

- a new wave model corresponding to an external force applied to fluid particles, based on physical observations; it enables a user to control choppy, linear, or breaking waves with ease of implementation;
- a 2D vertical control that can be used to interactively modulate the wave profile, speed, height, and width during its propagation;
- a 2D horizontal control that extends our paper [1] for controlling wave fronts and wave paths on the water surface while creating crossing and overtaking waves.

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, as well as curves affecting the propagation of wave fronts. 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 [7, 8, 5, 9, 10]. 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 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 [7, 11] use parametric equations to derive a surface to approximate waves and breaking waves. Tessendorf [8] animates a heightfield using fast Fourier transforms (FFTs), to which Bruneton et al. [12] 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 Ghazanfarpoor [13] 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, with particles and a level set to track the fluid surface and reduce compressibility. A Lagrangian approach considers the fluid as a set of particles and computes interaction between them to approximate a solution.

From a library of 2D simulated wave slices, Mihalef et al. [5] 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 an 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. [25] simulate breaking waves at interactive frame rates 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 [6]. The idea is to combine horizontal and vertical descriptions of a nonlinear wave [26], using hyperbolic secant and hyperbolic tangent functions, i.e., \( y = H \sec h(x - \omega t) \tanh(x - \omega t) \), where \( H \) is the wave height, and \( \omega \) 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 Vries (KdV) partial differential equation [27], they remain too costly for interactive applications.

Darles et al. [28] 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. [28], 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:

\[
\frac{d\vec{v}_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \mu_i \nabla^2 \vec{v}_i + \frac{\vec{F}^{\text{ext}}_i}{\rho_i} \tag{1}
\]

\[
\frac{d\vec{x}_i}{dt} = \vec{v}_i, \tag{2}
\]

where for a given particle \(i\), \(\vec{x}_i\) corresponds to the position, \(\vec{v}_i\) the velocity, \(\rho_i\) the particle density, \(p_i\) the pressure, \(\mu_i\) the dynamic viscosity, and \(\vec{F}^{\text{ext}}_i\) 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 [29, 30] is:

\[
A_i = \sum_j m_j \rho_j W(\vec{x}_i - \vec{x}_j, h) \tag{3}
\]

where \(m_j\) represents the mass of particle \(j\), \(h\) is the maximal interaction distance between two particles, and \(W\) is a smooth interpolation 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) [22].

3. A New Generic Wave Model

Our wave model simplifies and generalizes the soliton representation proposed by Darles et al. [28]. It corresponds to an external wave force that controls swells and breaking waves in a profile plane with varying height, speed, breaking time, and breaking duration. In this extended version of our original paper [1], we provide additional controls of a wave over the water surface, including wave fronts and wave paths, and show that the model can also handle crossing and overtaking waves.

3.1. Our 2D Nonlinear Wave Forces

From swells to breaking waves, four animation steps can be distinguished (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 [32], 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 that provides the wave motion, and a vertical component that drives the wave elevation. The wave propagation is given by a user-defined wave speed \(\omega\); 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:

\[
F_x = -\sqrt{2d} \sech(A d_c) \tanh(A d_c) \lambda_x, \tag{4}
\]

\[
F_z = \begin{cases} 
H \sech^2(A(x_i - \omega t - x_{\text{shoal}})) \lambda_z & \text{if } x_i < x_{\text{break}} \\
0 & \text{otherwise},
\end{cases} \tag{5}
\]

where \(A = \sqrt{3H/4d^2}; H\) is the user-defined wave height in meters, \(t\) is the current time step, \(\omega\) is the propagation velocity of the wave, \(d_c = z_i - z_{\text{shoal}}\) is the relative depth of particle \(i\), \(z_{\text{shoal}}\) is the initial depth of the particle, and \(\lambda_x\) and \(\lambda_z\) are two user-defined parameters that control the shape of breaking waves; \(\sqrt{2d}\) is an attenuation term proportional to gravity [6], \(d\) denotes the width of the wave, and \(\omega t\) (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 \(d_c\). 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.

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

1. Shoaling.

The shoaling point \(x_{\text{shoal}}\) locates the origin of a wave; the forces are initialized at \(t = 0\), and begin at position \(x_{\text{shoal}}\). 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 \[26\], 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 \(5\)).

Steepening corresponds to an increase of the vertical component of the wave force when the horizontal coordinate particle position \( x \) is close to \( x_{\text{shoal}} \). 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)).

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)).

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 \( d \), 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 \( d \), pushing particles located near the surface and producing breaking waves.

The curves of the plot in Figure 4 (top) 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 \( \tau \), the shorter the decreasing part of the curve (steepening). For low values of \( \tau \), 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 \( H \), \( \omega \), \( \lambda_s \), and \( \lambda_c \).

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 \( \omega \), shoaling and breaking points \( x_{\text{shoal}} \) and \( x_{\text{break}} \). Scaling forces using \( \lambda_s \) and \( \lambda_c \) (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, swells are obtained with \( H < 0.2 \), while larger values of \( H \) produce breaking waves, as illustrated in Figures 3 and 6.
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.

**Generation and Control of Waves.**

A single wave is an instance of our force model, with initial settings: starting location $x$ and time $t$, and a propagation direction. As the wave rises from time $t$, its parameter values can be modified (even interactively during the simulation) through editable curves acting as time-dependent scaling factors. For instance, a Bezier curve increasing from 0.0 to 1.0 during the shoaling stage produces a smoothly rising wave. This process is illustrated in Figure [12]. Each wave has its own set of parameter values and can be played as many times as needed.

**Wave Orientation.**

With a given propagation direction $\theta$ 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 way of tuning a wave consists in giving several values of $H$ so that the crest height varies. With our GUI, the user draws a curve which is mapped onto $H$ values during the simulation, as shown in Figure [8].

**Wave Fronts.**

In the real world, wave fronts do not always appear as straight line segments over the water surface. Water streams, obstacles, and seabed geometry can contribute to the curving of propagated wave fronts. An artist might thus want to give an arbitrarily curved shape to wave fronts on the water surface (in the $xy$ plane, as shown in Figure [9]). In our system, curves traced in the top-view profile (based on Bezier curves) are used to control a wave front. Each curve is used as a translation function $T$ applied to the $x_i$ parameter in our external wave force, so that the wave front can be accordingly shaped:

$$ F_x = F_{x_i + T_x}, $$

$$ F_z = \begin{cases} H \sech^2 \left( A \left( x_i + T_z - x_{\text{shoal}} \right) - \omega t - x_{\text{shoal}} \right) \lambda_z & \text{if } x_i < x_{\text{break}} \\ 0 & \text{otherwise} \end{cases} $$

An example of a wave produced with this method is shown in Figure [13]
Wave Paths.

Instead of relying on a simple translation for creating a wave front, we can also generalize the approach of defining a curved path for a curved wave front to the one of interpolating both wave fronts and wave directions. One can interpret this operation as specifying Bezier patches in 2D, where two wave fronts correspond to curves at \( u = 0 \) and \( u = 1 \), associated with time \( t_1 \) and \( t_2 \) respectively, and each pair of points for a given parametric value \( 0 \leq u \leq 1 \) are connected along a \( v \) path (Figure 10). As long as there are no self-intersections in this parametric patch, each point will have a unique parametric value in time (along \( u \)) and orientation (along \( v \)) to generate the forces in time. Complex wave fronts and wave paths can then be generalized to sets of connected Bezier patches. An example of such a curved propagation is shown in Figure 14.

\[
P_i(x_i, y_i) \text{ corresponds to } (u_i, v_i)
F_x = F_u \times \vec{V}_x
F_y = F_u \times \vec{V}_y
F_z \text{ does not change}
\]

Interactions between Waves.

As stated earlier in Section 2, interactions between several waves require solving the KdV equation [27], 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 \( (H_j, \omega_j, \Theta_j, \lambda_x^j, \lambda_y^j) \). When two waves overlap, summing up 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., \( F_{x_i} = \max(F_{x_i}^1, ..., F_{x_i}^N) \) and \( F_{z_i} = \max(F_{z_i}^1, ..., F_{z_i}^N) \) for \( N \) simulated waves. We have produced two facing waves using oscillating walls, and compared their traversal of each other with our wave model (Figure 11). We can indeed observe similar visual behaviors.

4. Results

This section illustrates results produced with our wave model, including shoaling, propagation, steepening, and breaking. With our system, several scenarios 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. A video containing all the frames of our animated results is available at https://vimeo.com/149543784.

Figure 12: 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. The red curve corresponds in this case to the speed of waves over time.
4.1. Parameter Values for Different Waves

Figure 12 shows 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. Once the user is satisfied with the result, the corresponding parameters and curves can be exported for later use. The simulations illustrated in this paper have been produced with this system.

Wave Fronts.

Figure 13 shows two examples of a curve that defines a wave front, applied on the water surface, and the resulting wave with linear propagation. The translation associated with the curve is directly applied to our external wave force, which makes the wave front correspond to the traced curve. Figure 19 shows three frames from an animation of two wave fronts that cross each other, illustrating that our wave force model behaves naturally in a more complex situation.

Figure 14: A wave propagates (top left, top right, bottom left, bottom right) along a user-traced curved path.

4.2. Management of Multiple Waves

Our model can represent many interacting crossing waves with visual plausibility.

Symmetrically Crossing Waves.

Figure 15 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 \( \lambda \) parameter and using a positive and a negative value to represent opposite propagation directions in our wave interaction system.

Figure 15: 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 16 corresponds to a wave that overtakes another one, i.e., with higher speed propagation \( \omega \). 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.

Figure 16: A wave propagates (top left, top right, bottom left, bottom right) along a user-traced curved path.
Figure 16: 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.

Obliquely Crossing Waves.
Figure 17 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 \( \theta \) with two different orientations. The interaction of multiple oblique waves produces the result illustrated in Figure 18.

Figure 17: An oriented wave obliquely crossing two others. The two parallel waves have been given a 45-degree propagation direction, with \( H = 0.26, d = 0.25, \) and \( \omega = 0.002 \).

Curved Wave Fronts Crossing.
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 18 illustrates some of the configurations that we have defined; Figure 19 shows two frames of the simulation without blocks; Figure 20 uses another configuration of several waves, with additional interactions with blocks.

Figure 18: 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 19: Two waves propagate along curved wave fronts and cross each other with different curved shapes.

Figure 20: Two frames from an animation of multiple short waves from different directions interacting with each other.

4.3. 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, including computations of wave forces. Particle neighbors are identified using a uniform grid, as described by Ihmsen et al. [33]. 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. [22].
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 [34] 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.

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Swell</th>
<th>Plunging</th>
<th>Opposite</th>
<th>Cross sea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighbourhood search</td>
<td>11.39%</td>
<td>48.79%</td>
<td>48.97%</td>
<td>61.48%</td>
</tr>
<tr>
<td>Internal forces evaluation</td>
<td>41.41%</td>
<td>40.54%</td>
<td>40.58%</td>
<td>31.26%</td>
</tr>
<tr>
<td>Wave forces evaluation</td>
<td>0.94%</td>
<td>0.99%</td>
<td>1.7%</td>
<td>0.86%</td>
</tr>
<tr>
<td>Integration</td>
<td>0.82%</td>
<td>0.8%</td>
<td>0.83%</td>
<td>0.49%</td>
</tr>
<tr>
<td>Memory management</td>
<td>8.94%</td>
<td>8.88%</td>
<td>7.92%</td>
<td>5.91%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Scene</th>
<th>Particles</th>
<th>Time per frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short width (Figure 6)</td>
<td>30k</td>
<td>0.013s</td>
</tr>
<tr>
<td>Long width (Figure 17)</td>
<td>100k</td>
<td>0.022s</td>
</tr>
<tr>
<td>Large depth (Figure 5)</td>
<td>255k</td>
<td>0.076s</td>
</tr>
</tbody>
</table>

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 $\omega$ also influences its height.

Too high amplitudes produce unrealistic results because the resulting vertical force is too large and particles explode. Parameter $\omega$ 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 wave force model produces a wide variety of phenomena, making it possible to generate swell, choppy, or breaking waves that can mutually interact. We show 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 other. We also show several ways to control the wave shape, including the inclination of its crest and its wavefront, based on 2D curves. The direction of the wave over time can also be manually defined, thanks to a curve which provides its direction. 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 [2], PIC-SPH [3], IISPH [23]), and also in 3D Eulerian [19] or hybrid solvers [55].

Crossing waves as well as other wave phenomena remain difficult to validate in practice, because of the lack of measured data for real fluids. In this paper we have shown examples that assess visually our simple model, but given 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 wave speed, height, or breaking duration. A relationship with Bezier 2D patches and “natural” wave front and wave propagation would also help the artist in specifying more realistic waves.

## 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


