Thèse

Pour l'obtention du grade de

Docteur de l'université de poitiers

Faculté des sciences Fondamentales et Appliquées
(Diplôme national – arrêté du 25 mai 2016)

École Doctorale **: Sciences et Ingénierie pour l'Information, Mathématiques**
Secteur de Recherche **: Informatique**

Présentée par

Mathias Brousset

****

Simulation et Rendu de Vagues Déferlantes

****

Thèse soutenue le 7 Décembre 2017 devant le jury composé de :

Daniel MENEVEAUX, Professeur, université de Poitiers ..... *Directeur de Thèse*
Pierre POULIN, Professeur, université de Montréal ..... *Encadrant*
Emmanuelle DARLES, Maître de conférence, université de Poitiers ..... *Encadrante*
Céline LOSCOS, Professeure, université de Reims Champagne-Ardenne ..... *Examinatrice*
Marie-Paule CANI, Professeure, Ecole Polytechnique ..... *Rapporteur*
Christophe RENAUD, Professeur, université du Littoral Côte d'Opale ..... *Rapporteur*

Il serait ubuesque de commencer cet humble travail de thèse en omettant de remercier ceux qui l'ont rendu possible. Pour ne pas déroger à l'usage, mais aussi parce que c'est sincère, voici mes remerciements exhaustifs.

Je tiens à exprimer ma reconnaissance envers Daniel, mon directeur de thèse, pour son aide ainsi que ses encouragements à répétition, qui ont finalement permis à cette thèse d'arriver à son terme. Je souhaite également remercier Pierre pour son aide précieuse et les discussions hautement intéressantes, fussent-elles scientifiques ou mélomanes, mais aussi pour son accueil au LIGUM. Ce fût un plaisir. Je remercie également Mickaël, pour les discussions toujours très enrichissantes et pour son aide technique qui m'a été hautement précieuse, mais aussi pour avoir accepté d'encadrer la dernière partie de ma thèse. Je tiens également à remercier Emmanuelle et Benoît pour leur aide enrichissante. Travailler avec vous tous a été une superbe expérience.

Je souhaite remercier ma famille (Alain, Bernadette, Mélissa, Gladys), dont le soutien tenant parfois de l'abnégation, m'a permis d'arriver ici. Mon grand père Roland, qui a joué le rôle de catalyseur dans mon intérêt pour l'informatique, est sans doute au sommet de la chaîne de causalité dans l'accomplissement de cette thèse, et pour ça je le remercie.

Je souhaite également remercier et saluer mes amis : Jérôme et Aurélie, Romain et Camille, Mickaël B. pour les références musicales, les projets et autres excursions, Antonio-Spierdalaj pour ma désormais haute maîtrise de la langue russe. Je tiens aussi à remercier les doctorants du laboratoire XLim, notamment Valentin.

Résumé

Depuis plusieurs décennies, la communauté informatique graphique s'intéresse à la simulation physique du mouvement et du rendu des fluides, qui nécessitent d'approcher numériquement des systèmes complexes d'équations aux dérivées partielles, coûteux en temps de calcul. Ces deux domaines trouvent entre autres des applications dans le domaine vidéoludique, qui requiert des performances pouvant offrir des résultats en temps interactif, et dans la simulation d'écoulements réalistes et complexes pour les effets spéciaux, nécessitant des temps de calcul et d'espace mémoire beaucoup plus considérables. Les modèles de la dynamique des fluides permettent de simuler des écoulements complexes, tout en offrant à l'artiste la possibilité d'interagir avec la simulation. Toutefois, contrôler la dynamique et l'apparence des vagues reste difficile. Cette thèse porte d'une part sur le contrôle du mouvement des vagues océaniques dans un contexte d'animation basée sur les équations de Navier-Stokes, et sur leur visualisation réaliste. Nos deux contributions principales sont : (i) Un modèle de forces externes pour contrôler le mouvement des vagues, avec leur hauteur, leur point de déferlement et leur vitesse. Une extension du modèle pour représenter l'interaction entre plusieurs vagues et des vagues tournantes est également proposée. (ii) Une méthodologie pour visualiser les vagues, à l'aide d'une méthode de rendu réaliste, en s'appuyant sur des données optiques des constituants océaniques pour contrôler l'apparence du fluide considéré comme milieu participant. La simulation et le contrôle de la dynamique des vagues sont mis en œuvre dans un simulateur basé sur la méthode SPH (*Smoothed Particle Hydrodynamics*). Afin d'obtenir des performances interactives, nous avons développé un moteur de simulation SPH tirant parti des technologies GPGPU. Pour la visualisation, nous utilisons un moteur de rendu existant pour traiter la visualisation physico-réaliste des milieux participants. Utilisées conjointement, les deux contributions permettent de simuler et contrôler la dynamique d'un front de mer ainsi que son apparence, sur la base de ses paramètres physiques.

Mots-clés : *simulation, contrôle, rendu, synthèse d'images, vagues déferlantes, océan, GPU, GPGPU, SPH*

Abstract

Physics-based animation and photorealistic rendering of fluids are two research fields that have been widely addressed by the computer graphics research community. Both have applications in the video-entertainment industry and are used in simulations of natural disasters, which require high computing performance in order to provide interactive time results. This thesis first focuses on simulating breaking waves on modern computer architectures and then on rendering them in the case of oceanic environments. The first part of this thesis deals with physics-based animation of breaking waves, and describes a simple model to generate and control such waves. Current methods only enable to simulate the effects but not the causes of water waves. The implementation of our method takes advantage of GPGPU technologies because of its massively parallel nature, in order to achieve interactive performances. Besides, the method was designed to provide user-control of the physical phenomena, which enables to control in real time all the physical parameters of the generated waves, in order to achieve the desired results. The second part of this thesis deals with the optical properties of water in oceanic environments and describes a model that enables realistic rendering of oceanic scenes. It provides user-control of oceanic constituants to tune the appearance of oceanic participating media.

Keywords : *Simulation of Breaking Waves, SPH, GPGPU, Breaking Waves Rendering*

Table des matières

Introduction et objectifs

1

Représenter des phénomènes du monde tangible dans des applications virtuelles représente un ensemble de défis dont l'essor est apparu il y a déjà quatre décennies. Faire converger le réalisme des applications numériques avec la réalité permettrait de simuler ces phénomènes avec des coûts et des risques (matériels, humains) réduits. L'accroissement du besoin de simulation de ces expériences, qu'elles soient à visée ludique, scientifique ou industrielle, s'est fait naturellement avec l'explosion de l'accès au monde virtuel. Le feu, les explosions ou encore les raz de marée sont des phénomènes physiques particulièrement dangereux du fait de l'énergie qu'ils libèrent. Parvenir à les animer et à les afficher de manière réaliste est donc devenu incontournable, avec notamment des applications dans le secteur des effets spéciaux, des jeux vidéo, ou encore des simulations de catastrophes naturelles. Pour imiter le mouvement des milieux continus (fluides, gaz, plasma) et des milieux non continus (solides indéformables), il est d'abord nécessaire de comprendre les notions physiques, parfois absconses, que leur existence fait intervenir. De nombreux scientifiques ont contribué à améliorer notre compréhension du mouvement des fluides, décrit par les équations de Navier-Stokes. De même, la compréhension de l'interaction entre la lumière et la matière, est représentée par l'équation du transfert radiatif. Traduire ces deux modèles physiques en algorithmes permettant de les visualiser sur ordinateur est un problème connu auquel s'attelle une partie de la communauté scientifique. Enrichir les phénomènes reproduits selon le desiderata d'un artiste est également un besoin croissant, particulièrement dans le processus de conception des films d'animation. Cette thèse vise à apporter quelques humbles pierres à l'édifice, dans le cas du contrôle et de la simulation réaliste de vagues déferlantes d'une part, et pour leur affichage 3D photoréaliste d'autre part.

Pour simuler la dynamique d'un liquide en informatique, deux familles de méthodes prévalent. La première consiste à animer une surface de manière procédurale, en utilisant des cartes de hauteur animées [Tes99, TG02, NSB13]. Ces modèles tentent généralement d'imiter le comportement périodique des océans, en reproduisant l'effet, mais pas les causes. Ces méthodes offrent une complexité algorithmique performante, mais reposent avant tout sur des observations d'océanographes et ne permettent pas de représenter certains phénomènes à fortes turbulences tels que les tourbillons ou les vagues déferlantes. La seconde famille de méthodes vise à approcher numériquement une solution aux équations de la mécanique des fluides [MCG03, MM13, FM96]. Ces méthodes décrivent le mouvement du liquide en fonction du temps, et considèrent le domaine simulé comme un volume et non plus une surface. Les méthodes basées sur la mécanique des fluides sont plus génériques et permettent de simuler un grand nombre de matériaux et de phénomènes. Cependant, les temps de calcul nécessaires sont nettement supérieurs aux méthodes procédurales, y compris avec des architectures parallèles performantes. En effet, elles nécessitent de simuler un grand nombre de particules de fluide dépendantes les unes des autres, et il est difficile de les simuler de manière interactive. Ces méthodes offrent en revanche un niveau de réalisme important.

Bien qu'il soit désormais possible de simuler le mouvement de fluides sur ordinateur, générer et contrôler avec précision des phénomènes propres aux grands domaines d'eau tels que les mers reste un défi de taille. Il existe de nombreuses variations de mouvement observables, telles que la houle, la mer croisée ou encore les vagues déferlantes. Les méthodes actuelles basées sur la mécanique des fluides permettent de simuler tout type de vagues, mais font abstraction des notions physiques sous-jacentes à leur formation et à leur propagation,

puisqu'elles impliquent de modifier la géométrie de l'environnement dans lequel le fluide est simulé. En outre, contrôler le mouvement du fluide grâce à l'interaction de l'utilisateur constitue un problème difficile. Il existe actuellement très peu de méthodes permettant de contrôler le mouvement et l'apparence d'une vague dans le cadre d'une animation physique. Une exception remarquable de Mihalef et al. [MMS04] permet de construire l'apparence et le comportement d'une vague 3D à l'aide d'une librairie de profils de vagues 2D. Même si leur solution produit des résultats convaincants, la méthode ne permet pas de contrôler l'apparence du fluide durant la phase de simulation. Arriver au résultat désiré par essais et erreurs peut devenir compliqué. Pour traiter ce problème de manière générale, nous introduisons un modèle de forces externes décrivant physiquement la propagation d'une vague. Notre modèle permet de simuler les étapes de formation, de propagation, de déferlement et de dissipation des ondes mécaniques que constituent les vagues grâce à un ensemble de paramètres physiques simples, tels que la vitesse, l'orientation et l'amplitude. Notre méthode a également pour but de contrôler l'ensemble des paramètres, simplement en les considérant comme fonctions du temps au travers d'une interface intuitive. Un de ses grands avantages est sa rapidité d'exécution, car elle donne la possibilité de modifier les paramètres en cours de simulation. Elle est de plus parallélisée sur carte graphique et permet donc d'obtenir des résultats de manière interactive, avec plusieurs dizaines d'images par seconde.

Un autre défi en informatique graphique est de reproduire et contrôler l'apparence des liquides et leur interaction avec la lumière (partie visible du spectre électromagnétique). Nous souhaitons à plus forte raison reproduire cette interaction lumière/matière avec les milieux et les phénomènes océaniques. Actuellement, peu de méthodes permettent de réaliser un rendu réaliste et plausible des milieux océaniques, à l'exception notable de la méthode de Premoze et Ashikhmin [PA00]. Afficher des vagues sur un écran de manière réaliste est difficile, du fait du nombre de paramètres jouant un rôle dans l'apparence de l'eau de mer telle que nous pouvons la percevoir. Dans la communauté océanographique, deux catégories de propriétés existent pour identifier les propriétés optiques de l'eau : les IOP (de l'anglais *Inherent Optical Properties*, les propriétés optiques inhérentes au milieu), qui ne dépendent que du milieu lui-même (la salinité de l'eau, sa teneur en différents matériaux biologiques), et les AOP (de l'anglais *Apparent Optical Properties*, les propriétés optiques apparentes), qui font intervenir les paramètres externes au milieu, tels que la position du soleil ou des nuages. Notre objectif est d'apporter un maximum de données physiques pour traiter l'ensemble du problème. Nous souhaitons également faire intervenir l'apparence de l'écume, des embruns, des bulles et du sable dans nos rendus. Nous proposons une méthodologie plausible permettant de faire varier l'apparence de l'eau en fonction de ses IOP, faisant eux-mêmes varier la capacité de l'eau d'absorber la lumière ou de la diffuser (son coefficient d'absorption, *absorption coefficient* en anglais) ou à la diffuser (son coefficient de diffusion, *scattering coefficient*). Afin d'y parvenir, un ensemble de paramètres issus de la biologie marine sont utilisés pour déterminer les coefficients de diffusion et d'absorption du milieu. La méthode permet également de représenter une partie du volume d'eau chargé en particules de sable, conformément aux particules de sable traitées grâce à la contribution précédente. Le modèle permet aussi de rendre de manière physique les particules d'écume, de bulles et d'embruns grâce à des intuitions simples, et toujours avec la possibilité de contrôler leur apparence en faisant varier les paramètres physiques sous-jacents.

Le travail présenté dans cette thèse a donné lieu à des communications dans des journaux et des conférences :

- A New Wave Force Model for Controllable Breaking Waves, Vriphys 2015, Lyon ;
- Un nouveau modèle pour la génération et le contrôle de vagues déferlantes, journées de l'afig 2015, Lyon ;
- Simulation and Control of Breaking Waves Using an External Force Model, Computers & Graphics, June 2016 ;

Les résultats des travaux réalisés lors de cette thèse peuvent être visionnés aux adresses internet suivantes :

- vriphys : https://vimeo.com/133911694 ;
- C&G : https://vimeo.com/149543784 ;

La suite de cette thèse est organisée en quatre parties. Le chapitre 2 propose un tour d'horizon de l'état de l'art dans les domaines de la simulation basée physique et la visualisation de fluides. Le chapitre 3 présente notre modèle de contrôle de vagues déferlantes. Nous y détaillons également le système de parallélisation sur carte graphique utilisé pour simuler des fluides. Le chapitre 4 présente notre méthodologie pour la simulation d'éclairage des milieux participants des vagues déferlantes. La méthode utilise des paramètres issus de travaux d'opticiens et océanographes. Elle permet aussi de représenter les particules océaniques de type sable, écume, bulles, embruns. Le chapitre 5 fait le bilan de cette thèse et propose des perspectives pour les travaux futurs.

Simulation et rendu de fluides 2

Sommaire

2.1 Introduction

Dans ce chapitre, nous abordons en premier lieu les bases théoriques nécessaires à la compréhension des mécanismes sous-jacents à la propagation des vagues océaniques. Nous y définissons la notion d'onde, et subséquemment les modèles de houle à base de fonctions périodiques et de spectres permettant de formaliser de manière procédurale le comportement de la surface de la mer en eaux profondes. Ces modèles, d'abord introduits par des océanographes sur la base d'observations empiriques, ont connu et continuent de connaître un grand succès en informatique graphique car la démarche d'implémentation qu'ils requièrent est relativement aisée.

Par la suite, nous présenterons une description physique de la dynamique des milieux continus notamment par une introduction au système d'équations de Navier-Stokes qui permet de généraliser la description d'un écoulement de fluide en fonction de ses caractéristiques intrinsèques. Nous décrirons également un ensemble de méthodes utilisant cette description physique. La considération volumique de cette description physique permet de pallier les limitations d'une représentation purement surfacique pour la simulation de certaines déformations. Les méthodes de simulation numérique permettant d'approcher des solutions à ce système d'équations suscitent un large intérêt en informatique car elles permettent de représenter les turbulences d'écoulements avec un niveau de granularité très fin, et tirent parti des paradigmes d'implémentation GPGPU (*General Purpose Computing on Graphics Processing Units*) notamment à l'aide de patrons de conception de type *Gather* ou *Scatter*. Après avoir détaillé les descriptions physiques d'écoulements de fluides, nous décrirons des méthodes de simulation numérique, et en particulier la méthode SPH (*Smoothed Particle Hydrodynamics*) que nous avons utilisée dans le cadre des travaux et publications réalisés au cours de cette thèse.

Les modèles de la physique numérique possèdent l'inconvénient majeur de disposer d'une géométrie intrinsèque différente de la réalité, car elles requièrent une discrétisation du volume de liquide dont elles simulent l'écoulement. À titre d'exemple, un volume de liquide représenté par une simulation SPH est un nuage de points et non pas un volume fermé par une surface continue comme dans le monde tangible. De nombreuses méthodes pallient cette limitation d'ordre géométrique afin d'approcher la surface du liquide par une fonction implicite ou une représentation explicite telle qu'un maillage triangulaire. Nous décrivons dans la suite de cet état de l'art un ensemble de méthodes permettant de convertir la géométrie du modèle physique en une géométrie permettant d'approcher au mieux la surface de l'eau.

Après avoir pris en compte ces considérations dynamiques et géométriques, nous posons les bases de la théorie des milieux participants, dont l'objectif est de décrire l'interaction entre le rayonnement lumineux et la matière. Nous décrivons les éléments à prendre en compte pour avoir une description plausible de cette interaction entre la lumière et la matière. Nous y décrivons également les méthodes principales permettant de faire des calculs de simulation d'éclairage tenant compte de ces interactions.

Les différentes sections de cet état de l'art, au-delà des mises en oppositions des différents paradigmes et théories, constituent le fil conducteur des travaux de cette thèse, durant laquelle nous avons mis en œuvre une chaîne de traitements permettant de contrôler tous les aspects d'une simulation de fluide, de l'animation au rendu.

2.2 Phénoménologie et physique

L'étude des mouvements de l'eau s'est historiquement faite à deux échelles. La première, à l'échelle macroscopique, tente de formaliser le comportement de la surface de l'océan dans son ensemble, sur la base d'observations océanographiques. Il s'agit donc d'une approche phénoménologique de l'étude des mouvements océaniques, qui traite en grande partie des

eaux profondes. La seconde, à une echelle plus petite, tente de répondre de manière physique aux problèmes d'écoulements de fluides propres à des milieux peu profonds, et considère la plupart du temps le fluide comme un volume, contrairement aux travaux des océanographes.

Cette partie de la thèse propose un tour d'horizon sur les bases théoriques utilisées en informatique graphique pour la simulation numérique de fluides en synthèse d'images.

2.2.1 Observations océanographiques

Les vagues sont des ondes mécaniques. Dans le sens général du terme, les ondes sont des perturbations (mécaniques, électromagnétiques) altérant les propriétés du milieu qu'elles traversent à un instant donné. Les perturbations que les vagues créent dépendent de leurs propriétés (figure 2.1) :

- la hauteur H, ou amplitude ;
- la longueur d'onde λ.

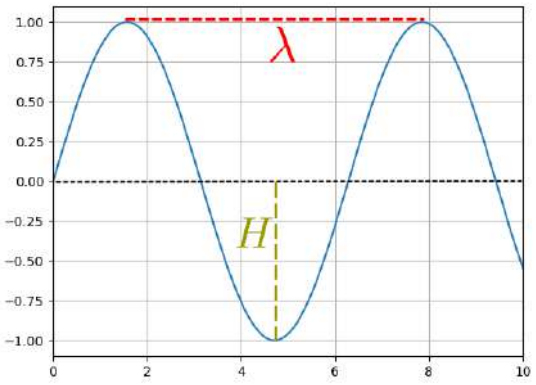
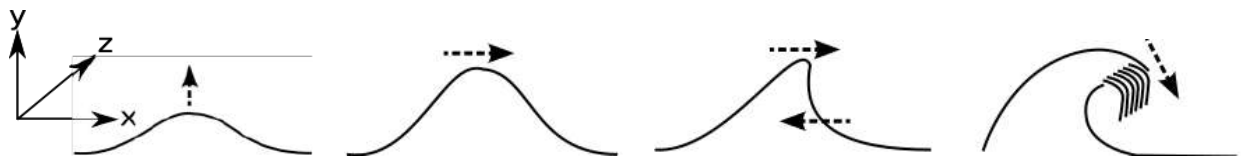


FIGURE 2.1 – *Une onde est une perturbation caractérisée par une amplitude ainsi qu'une longueur d'onde.*

Dans le cas de la mécanique des fluides, les ondes produisent des déformations visibles sur la surface de l'eau. Elles sont la plupart du temps produites sous l'effet du vent, ou le passage de véhicules marins. Lorsqu'elles se rapprochent de la plage, les vagues peuvent éventuellement déferler, selon leur amplitude, et sous l'effet de l'augmentation de la hauteur du sol au niveau de la plage. Les étapes correspondant à la formation d'une vague jusqu'à son déferlement sont décrites sur la figure 2.2 :



(a) formation (b) propagation (c) transformation (d) déferlement

FIGURE 2.2 – *Les quatre étapes de la vie d'une vague.*

Sous l'effet du vent par exemple, le volume d'eau est déformé, donnant lieu à la formation de la vague, première étape de son cycle de vie (étape a). La vague se propage ensuite et se renforce, toujours sous l'effet du vent, et se rapproche des eaux côtières. C'est l'étape de propagation (étape b). Lorsque l'eau de rapproche de la plage, le volume d'eau situé en profondeur est ralenti à cause du sol qui s'élève, et le volume d'eau constitué par la vague garde sa vitesse de propagation. Cette différence de vitesse entre les deux volumes produit une transformation de

la vague, proche du déferlement (étape c). Lorsqu'elle atteint les eaux peu profondes, la vague finit par déferler et atteint finalement la plage (étape d).

Beaucoup d'océanographes ont tenté de formaliser, de façon mathématique, le comportement global de la surface d'un océan à l'aide d'équations paramétriques. Ces modèles sont notamment valides pour les eaux profondes car ils permettent de représenter le comportement périodique de la houle en particulier.

Au 19è siècle, Gerstner propose un modèle mathématique pour représenter des vagues de houle se propageant sur la surface de l'océan, en faisant l'hypothèse d'une profondeur infinie (critère se basant sur la théorie de la houle d'Airy), et d'un fluide incompressible [Gero9]. Ce modèle se traduit par un ensemble de deux fonctions F_x et F_y, qui permet de déterminer la position d'une particule de position (x, y) au cours du temps.

$$F_x(x, y, t) = x_0 + \frac{e^{ky}}{k}\sin(k(x + ct)) \tag{2.1}$$

$$F_y(x, y, t) = y_0 - \frac{e^{ky}}{k}\cos(k(x + ct)). \tag{2.2}$$

Les différents paramètres de cette équation sont :

- $k = \frac{2\pi}{\lambda}$, le nombre d'onde, avec λ la longueur d'onde ;
- c, la vitesse de la vague, pour une propagation sur l'axe x.
- x_0, le point d'origine pour les abscisses ;
- y_0, le point d'origine pour les ordonnées ;
- t, le temps courant ;

De la même manière, Stokes [Sto47] définit un modèle de houle (dit houle de Stokes de premier ordre) sur la base de fonctions périodiques, intégrant également une amplitude et une direction de propagation.

Ce modèle permet de représenter les vagues comme des ondes trochoïdales, et est utilisé en informatique graphique, notamment avec des approches à base de transformées de Fourier rapides (nous le décrivons en section 2.3).

Par opposition à ces modèles analytiques, Pierson et Moskowitz [PM64] ont proposé un modèle spectral en utilisant des mesures de la hauteur du niveau de l'eau :

$$F_{PM}(f) = \frac{ag^2}{(2\pi)^4 f^5} e^{-\frac{5}{4}(\frac{fm}{f})^4}. \tag{2.3}$$

Les différents paramètres de cette équation sont :

- F_{PM}, la fonction permettant de déterminer un spectre pour une fréquence donnée ;
- f, la fréquence de la vague, en Hertz ;
- a, la constante de Phillips [PAoo] ;
- g, l'accélération gravitationnelle ;
- f_m, le pic de fréquence, fonction de la vitesse du vent U_{10} définie par $\frac{0.13g}{U_{10}}$

La méthode JONSWAP [Has73] est une extension du spectre de Pierson et Moskowitz, permettant de tenir compte d'un vecteur de propagation :

$$F_J(f, \theta) = F_{PM} D(f, \theta). \tag{2.4}$$

Les paramètres de cette équation sont :

- f, la fréquence de la vague ;
- θ, l'angle de propagation obtenu par une direction du vent $\vec{u}$;
- D, la fonction permettant d'obtenir une propagation directionnelle [PAoo].

Pour extraire un champ de hauteur de ces méthodes spectrales, Premoze et Ashikhmin [PAoo] utilisent une image générée avec un bruit gaussien. Le champ de hauteur dans le domaine spatial est alors obtenu en utilisant une transformée de Fourier inverse. Ils utilisent ensuite la géométrie obtenue dans un contexte de rendu réaliste.

2.2.2 Mécanique des fluides

En physique, la mécanique des fluides est une branche de la mécanique des milieux continus. Cette dernière étudie les déformations d'objets solides ainsi que les écoulements de liquides, dont les propriétés sont des fonctions continues à l'échelle de l'observateur, dans l'espace (x, y, z) et au fil du temps t. La mécanique des fluides s'intéresse à l'étude des fluides au repos (statique des fluides), ainsi qu'au mouvement des fluides (dynamique des fluides).

Dans le but d'augmenter le réalisme des applications, la communauté a rapidement cherché à tirer parti des modèles de la dynamique des fluides et des outils de la dynamique des fluides numérique (*Computational Fluid Dynamics* (CFD) en anglais).

Cette section décrit les équations de la mécanique des fluides régissant le mouvement des fluides.

Équations de Navier-Stokes

Les équations de Navier-Stokes sont des équations aux dérivées partielles qui n'admettent à ce jour pas de solution analytique (et sont à ce titre parties de l'un des sept problèmes du prix Millénaire). Elles permettent d'estimer la quantité de mouvement d'un fluide ainsi que la variation de volume :

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = \vec{g} + \mu \nabla \cdot \nabla \vec{u} - \frac{1}{\rho} \nabla p \tag{2.5}$$

$$\nabla \cdot \vec{u} = 0. \tag{2.6}$$

La première équation du système s'appelle l'équation du bilan de la quantité de mouvement. Elle est en quelque sorte l'adaptation de la seconde loi de Newton (décrivant le vecteur de forces d'un objet résultant de son accélération et de sa masse, $F = m \cdot a$) pour les fluides. Le mouvement d'un fluide ne dépend pas uniquement, contrairement à un solide, des forces externes qui s'exercent sur ce dernier ainsi que de son accélération. Il faut tenir compte des forces internes au fluide pour en calculer le mouvement. Cette équation permet d'en tenir compte. La partie droite de cette équation correspond à une somme de forces, avec :

- $\vec{u}$ est un champ vectoriel correspondant à la vélocité du fluide ;
- $\vec{g}$, les forces externes dues à la gravitation ; il s'agit d'un vecteur de dimension trois ; la valeur communément utilisée, $(0, -9.81, 0)$, correspond à l'accélération due à la gravité terrestre ;
- μ correspond au coefficient de viscosité dynamique ; plus cette force est élevée (proportionnellement à son coefficient), plus le fluide est résistant aux déformations et à l'écoulement ;
- $\nabla \cdot \nabla$ correspond à l'opérateur laplacien ; il permet entre autres de connaître la différence entre la valeur d'une fonction et sa moyenne ; utilisé conjointement avec le coefficient de viscosité, il permet donc d'affecter l'accélération du fluide dépendamment de l'accélération moyenne du fluide ;

- ρ correspond à la masse volumique du fluide ; pour l'eau, elle est de $998.29kg \cdot m^{-3}$;
- ∇p correspond au gradient de pression ; le gradient est un opérateur différentiel qui étend la notion de coefficient directeur en n dimensions ; il permet de savoir dans quelle direction le fluide exerce une pression et dans quelle mesure ; intuitivement, l'eau d'une piscine exerce une pression élevée en direction du sol, et n'en exerce aucune en direction du ciel.

La seconde équation s'appelle l'équation de conservation de la masse. Dans le cas de fluides peu compressibles (tels que l'eau), elle s'appelle aussi l'équation d'incompressibilité. Elle fut initialement exprimée par l'équation suivante :

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \vec{u} = 0, \tag{2.7}$$

qui indique que la variation de densité dans un volume de fluide donné correspond à la quantité de matière entrant dans le volume. Pour un fluide incompressible, on considère qu'il n'y a pas de variation de densité, c'est pourquoi l'équation 2.5 est utilisée pour la simulation de fluides incompressibles.

Il est à noter qu'il n'existe pas de fluide complètement incompressible dans la nature, et l'eau ne fait pas exception. Toutefois, il est correct, dans l'approche numérique des équations de Navier-Stokes, de considérer les fluides dont le nombre de Mach (rapport de la vitesse de l'écoulement du fluide à la vitesse du son dans ce fluide) est inférieur à 0.3 comme incompressibles, car les variations de masse volumique du fluide sont alors négligeables. Nous décrivons les méthodes permettant d'approcher ce système d'équations en section 2.4. Coûteux en temps de calcul de par sa nature volumique, l'utilisation des équations de Saint-Venant sont parfois préférées pour certains types d'écoulements. Nous les décrivons dans la section suivante.

Équations de Saint-Venant

Les équations de Saint-Venant (*Shallow Water Equations* (SWE) en anglais) sont des équations dérivées des équations de Navier-Stokes 2.5. Elle comprennent donc une équation du bilan de quantité du mouvement ainsi qu'une équation du bilan de conservation de la masse :

$$\frac{Dh}{Dt} = -h\nabla \cdot \vec{u} \tag{2.8}$$

$$\frac{D\vec{u}}{Dt} = -g\nabla(h + H) + \vec{F_{ext}}. \tag{2.9}$$

Les différents termes de ce système d'équations sont :

- h, la hauteur de la surface de l'eau ;
- H, la hauteur du sol sous la surface de l'eau ;
- $\vec{u}$ représente la vélocité 2D du fluide sur le plan horizontal (x, z) ;
- g, la gravité ;
- F_{ext} représente les accélérations dues aux forces externes.

Ce système d'équations est notamment utilisé pour représenter des phénomènes liés aux écoulements en milieux peu profonds tels que les écoulements de rivières et de canaux, et côtiers. Il fait l'hypothèse d'un écoulement non visqueux en deux dimensions [LvdP02].

Les équations de Saint-Venant représentent la variation du niveau d'eau y en considérant le fluide comme un ensemble de colonnes d'eau statiques sur le plan horizontal (x, z) :

$$\frac{\partial \vec{u}_x}{\partial y} = \frac{\partial \vec{u}_z}{\partial y} = 0. \tag{2.10}$$

2.3 Animation procédurale

FIGURE 2.3 – *Des exemples de simulations de la surface de l'océan utilisant des méthodes procédurales [Tes99, HNC02a, DB12, FR86].*

Pour des océans de grande profondeur et des grandes étendues d'eau, les méthodes procédurales ou spectrales sont souvent utilisées. Le principe est de considérer la surface de l'eau à l'aide d'un champ de hauteur, déterminé suivant des fonctions périodiques. Leur efficacité en termes de réalisme visuel et de temps de calcul les rend très populaires, bien qu'elles ne reposent pas directement sur les équations de la mécanique des fluides.

Peachey [Pea86] propose de modéliser la surface de l'océan avec des quadrilatères, et de les animer en utilisant le modèle de vagues de Stokes pour produire un effet de houle.

Fournier et Reeves [FR86] animent des vagues déferlantes en représentant la surface de l'océan. Pour y parvenir, ils proposent d'utiliser des fonctions trochoïdes de la forme suivante :

$$x = x_0 + r \cdot \sin(\kappa x_0 - \omega t) \tag{2.11}$$
$$z = z_0 + r \cdot \cos(\kappa x_0 - \omega t) \tag{2.12}$$

dans lesquelles les paramètres r et κ permettent de contrôler la forme de la trochoïde. Ils proposent également d'introduire un ensemble d'effets tels que la profondeur ou le vent, permettant aux trochoïdes de s'approcher de l'apparence d'une vague déferlante. Leur modèle géométrique (une surface) ne permet toutefois pas d'obtenir un véritable déferlement. Hinsinger et al [HNC02b] utilisent un modèle de houle conjointement avec une représentation géométrique adaptative en fonction de la distance de l'observateur, pour afficher la surface d'un océan virtuellement infini avec des performances temps-réel.

Tessendorf [Tes99] anime un champ de hauteur en utilisant des transformées de Fourier rapides (FFT), auxquelles Bruneton et al. [BNH10] ajoutent du niveau de détail afin d'effectuer un rendu réaliste de l'océan en temps interactif. L'ajustement des paramètres pour ces méthodes étant difficile, Thon et Ghazanfarpour [TG02] proposent d'utiliser des mesures réelles comme

paramètres, et les auteurs ajoutent également un bruit de Perlin pour obtenir certains effets visuels. La principale limitation de ces méthodes est qu'elles ne peuvent pas représenter de vagues déferlantes car chaque position horizontale (x, z) ne peut comporter qu'une seule hauteur y. De plus, elles restent malgré tout difficiles à paramétrer pour une interaction avec des objets solides.

Il existe également certaines approches qui ajoutent des effets issus des simulations 3D dans des simulations purement surfaciques. Notamment, Thürey et al. [TMFSG07] détectent et marquent la phase de transformation (i.e. étape c de la figure 2.2) d'une vague par un segment de ligne droite, et génèrent un *patch* permettant de modifier la géométrie d'une simulation par carte de hauteur animée, afin d'afficher une vague déferlante. O'Brien et Hodgins [OH95] utilisent un système de particules pour représenter les *splashes* occasionnés par la chute d'un objet solide dans un liquide représenté par un champ de hauteur. Toutefois, la cohérence visuelle entre la dynamique de la surface et des *splashes* reste difficile à obtenir en pratique.

Le principal avantage des cartes de hauteur animées est qu'elles sont extrêmement rapides à calculer : l'absence de dépendance entre les données fait qu'elles sont facilement parallélisables. Elles sont donc de ce fait idéales pour représenter un vaste domaine océanique avec de hautes performances.

En revanche, la nature $2D$ de leur modèle géométrique ne permet pas de simuler certains phénomènes tels que des vagues déferlantes : en effet, pour chaque paire de coordonnée $(x; z)$, le modèle permet au plus une coordonnée y. Certains travaux ont toutefois essayé d'intégrer ces effets 3D dans des modèles 2D. La cohérence physique de ces modèles est toutefois limitée, les effets du déferlement n'ayant entre autres pas d'incidence sur le comportement de la carte de hauteur. Elles ne sont donc adaptées qu'à la simulation de domaines avec une turbulence limitée. Ces modèles sont de plus uniquement basés sur des observations. Bien que plausibles, ils ne reposent pas sur la mécanique des fluides ne peuvent donc représenter qu'un nombre limité de phénomènes.

Pour pallier toutes ces limitations, la communauté informatique graphique s'est intéressée à utiliser les modèles de représentation issus de la mécanique des fluides détaillés ci-après.

2.4 Simulation basée sur la mécanique des fluides

Les méthodes basées sur les équations issues de la mécanique des fluides ont connu un succès florissant depuis les années 1990. La communauté s'est particulièrement penchée sur deux familles de méthodes : les méthodes eulériennes, cherchant à estimer le mouvement pour un ensemble de positions fixes, et les méthodes lagrangiennes, représentant le fluide sous formes de particules, et cherchant à estimer leurs positions à chaque pas de temps. Ces méthodes possèdent chacune des avantages et inconvénients complémentaires. C'est pourquoi la communauté s'est également intéressée à développer des méthodes hybrides dans le but de tirer parti des avantages des deux familles de méthodes.

2.4.1 Simulation eulérienne

Les méthodes eulériennes discrétisent l'espace simulé à l'aide d'une grille. Dans les années 1960, Harlow et Welch introduisent le concept de grille MAC (*Marker And Cell*). Dans la terminologie anglophone, ce type de grille est qualifié de *staggered* (décalée en français) car l'estimation du champ de pression est calculée à une position différente du champ vélocité, comme illustré sur la figure 2.4. La grille MAC est utilisée pour calculer les opérateurs différentiels du système d'équations 2.5 en procédant par différences centrales. Foster et Metaxas [FM96] ont introduit ce concept en informatique graphique. Ils proposent également d'utiliser soit un champ de hauteur, soit des particules sans masse pour identifier la position

du fluide, dans l'optique de visualiser la surface. Stam [Sta99] introduit le concept d'advection semi-Lagrangienne, en utilisant une grille pour approcher les équations de Navier-Stokes, mais en utilisant des particules pour calculer l'advection du fluide. Foster et Fedkiw [FF01] utilisent une méthode hybride comprenant des courbes de niveau et des particules pour suivre le mouvement du fluide. De nombreuses méthodes utilisent également des grilles adaptatives. Losasso et al. [LGF04] utilisent un arbre octal afin de pouvoir représenter certains phénomènes avec un haut niveau de détail. Chentanez et Müller [CM11] utilisent, dans un contexte d'implémentation sur GPU, une structure de type *tall cell grid* comprenant des cellules de haute taille en profondeur, et des cellules cubiques fines pour la surface qu'ils définissent comme région d'intérêt. Feldman et al. [FOK05] optent pour une discrétisation par tétraèdres dans un contexte de simulation de gaz et d'interaction avec des objets solides. Il existe également de nombreuses méthodes pour représenter la surface du fluide dans une simulation eulérienne. Foster et Metaxas [FM96] utilisent un champ scalaire représenté géométriquement par un champ de hauteur, ou également en utilisant un ensemble de particules sans masse. Enright et al. [EMF02] améliorent la qualité de la surface définie par une fonction implicite et un *levelset* en plaçant des particules des deux côtés de celui-ci, permettant d'éviter des problèmes de perte de volume. Bargteil et al. [BGOS06] représentent la surface du liquide par un maillage triangulé. Ils l'advectent et le déforment à chaque pas de simulation pour le faire correspondre à la surface du liquide.

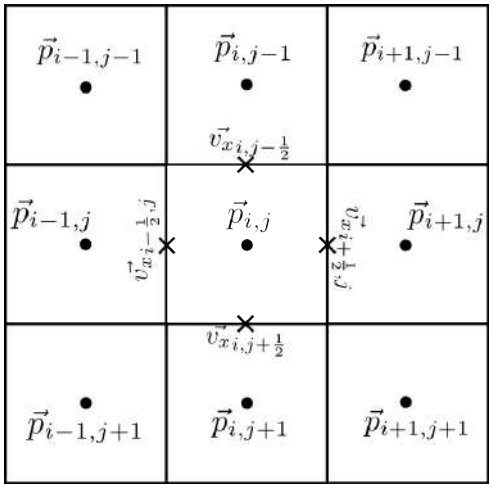


FIGURE 2.4 – *L'approche eulérienne considère le domaine de simulation comme un ensemble de points fixes, à l'aide d'une grille MAC (Marker And Cell).*

FIGURE 2.5 – *Des exemples de simulations de fluide eulériennes [CM11, FM96].*

Une boucle de simulation eulérienne est donnée dans l'algorithme 1. Nous décrivons dans les paragraphes suivants les différentes étapes de cet algorithme.

Algorithme 1 : Algorithme d'une simulation eulérienne.

```
while anime do
    Calculer l'advection du fluide
    Calculer la pression du fluide
    Identifier la surface du fluide
```

Advection du fluide

L'advection correspond tant bien au transport du fluide dans la grille, qu'au transport de ses propriétés. À ce titre, le champ de vélocité se déplace de la même manière que le liquide physique lui même. L'objectif *in fine*, qui est de calculer le gradient de pression et le divergent de vélocité des équations de Navier-Stokes, requiert de connaître la dérivée de champs scalaires et vectoriels (i.e. la pression et la vélocité). Les différences centrales (l'équation 2.13 donne l'exemple pour une simulation 3D) sont généralement utilisées à dessein.

$$\begin{aligned} \frac{\partial u}{\partial x_i} &\approx \frac{u_{i+1/2} - u_{i-1/2}}{\Delta x} \\ \frac{\partial v}{\partial y_i} &\approx \frac{v_{i+1/2} - v_{i-1/2}}{\Delta y} \\ \frac{\partial w}{\partial z_i} &\approx \frac{w_{i+1/2} - w_{i-1/2}}{\Delta z}. \end{aligned} \tag{2.13}$$

Calcul de pression

Aussi appelée projection de pression, cette étape vise à s'assurer du caractère incompressible de la simulation, plus concrètement que le divergent de vélocité reste à zéro : $\nabla \cdot \vec{u} = 0$. Elle est en général une étape critique de toute simulation eulérienne dans la mesure où elle requiert de résoudre numériquement un système linéaire coûteux. Pour y parvenir, la méthode du gradient conjugué préconditionné est souvent utilisée pour résoudre le système linéaire [FF01, Bri08]. Elle consiste en une méthode itérative permettant de converger vers une solution en un nombre limité d'itérations. L'ouvrage de Bridson [Bri08] détaille de manière exhaustive les différentes approches pour paramétrer le préconditionneur.

Identification de la surface du fluide

Identifier la surface d'un volume de fluide représenté par un ensemble de surfaces de niveau ϕ (*levelset* en anglais) consiste à conserver les cases de la grille pour lesquelles $\phi(\vec{x}) = 0$. *A contrario*, une valeur inférieure à 0 signifie que le point est compris dans le volume, et une valeur supérieure à 0, à l'extérieur. Une fonction distance signée est généralement utilisée pour cet effet. Les valeurs de cette fonction sont calculées pour chaque case de la grille, puis stockées. Une interpolation est ensuite réalisée pour déterminer la valeur de cette fonction pour un point $\vec{x}$ quelconque [Bri08].

Conclusion

La difficulté de capturer certains phénomènes de petite échelle (inférieure à Δx) tels que des éclaboussures est la principale limite fondamentale des méthodes eulériennes. Certaines méthodes utilisent des méthodes purement lagrangiennes conjointement à la simulation eulérienne

pour représenter des effets à haute résolution. Losasso et al. [LTKF08] utilisent des particules *SPH* (méthode décrite en section 2.4.2) dans une simulation eulérienne afin de représenter des éclaboussures dans des simulations turbulentes. D'autre part, une limite pratique majeure est la difficulté d'obtenir des performances temps-réel : les méthodes eulériennes nécessitent de calculer l'équation de continuité, et d'utiliser une grille avec une haute résolution pour des simulations avec un niveau de détail approprié.

2.4.2 Simulation lagrangienne : *Smoothed Particle Hydrodynamics*

Initialement développée pour répondre à des besoins de simulation numérique dans le domaine de l'astrophysique [Mon92, Mon05] (simulation de fissions d'étoiles, formations d'amas stellaires, etc.), la méthode lagrangienne *Smoothed Particle Hydrodynamics* (SPH) a depuis suscité un intérêt de la communauté informatique graphique (parmi les autres méthodes utilisant des points matériels, e.g. FLIP) pour la simulation de fluides (cette méthode n'est donc pas spécifiquement dédiée à la résolution d'un système d'équations particulier, comme discuté dans la section 2.4.2.2) et un grand nombre de méthodes dérivées ont vu le jour, les unes essayant par exemple de traiter les inconvénients physiques spécifiques à la simulation de certains matériaux, les autres se penchant sur l'optimisation de l'implémentation sur les architectures des GPU modernes, en tirant parti des technologies GPGPU (*General Purpose computing on Graphics Processor Units*) telles que OpenCL et CUDA.

Cette section définit les principes d'une simulation lagrangienne, puis fait un tour d'horizon des méthodes de l'état de l'art (et plus particulièrement pour les SPH).

2.4.2.1 Dérivée matérielle

Les équations de Navier-Stokes ont été pensées de telle manière à résoudre des problèmes d'écoulement de fluide dans un contexte eulérien. Dans ces conditions, approcher numériquement une solution aux équations de Navier-Stokes requiert une grille eulérienne comme référentiel.

L'approche lagrangienne des fluides est l'une des deux méthodes d'approche numérique d'écoulement de fluides. Plutôt que de représenter le volume d'un fluide à l'aide d'une grille, le fluide est directement représenté à l'aide de points matériels (des particules). Il faut toutefois adapter les équations de Navier-Stokes afin de passer d'une description eulérienne à une description lagrangienne : la dérivée particulaire est l'outil mathématique permettant de connecter ces deux points de vue.

La dérivée matérielle est un opérateur différentiel, dont la formule est

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + v \cdot \nabla. \tag{2.14}$$

En utilisant la dérivée matérielle afin d'approcher les équations de Navier-Stokes (équations 2.7 et 2.5) dans un contexte lagrangien (des points matériels qui se déplacent), nous obtenons le système d'équations suivant :

$$\frac{D\rho}{Dt} = 0 \tag{2.15}$$

$$\frac{D\vec{u}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\vec{u} + \frac{\vec{f_{ext}}}{\rho}. \tag{2.16}$$

Ce système fait à nouveau l'hypothèse d'une variation de densité nulle (le fluide est incompressible).

Un des atouts de la description lagrangienne est que les particules ont une masse : le nombre de particules étant fini, de même que leur masse, on considère que la masse totale du fluide ne varie pas au cours du temps. Cette hypothèse permet d'écarter le calcul de l'équation de bilan de la masse. Bien que cela soit un atout des méthodes lagrangiennes, nous verrons dans les sections suivantes que c'est aussi une de leurs principales limites, que la communauté tente de corriger.

2.4.2.2 Principe

Plutôt que de représenter le fluide sous forme d'une grille et de calculer les vélocités pour une liste de points connus (comme décrit dans la section 2.4.1), les méthodes lagrangiennes considèrent le fluide sous forme d'un ensemble de points matériels, dont on essaie de connaître les variations de positions au fil du temps.

En raisonnant par somme de contributions pondérées pour chaque point matériel (voir la figure 2.6), de la manière suivante :

$$A_i(\vec{x_i}) = \int A(\vec{x_j}) W(||\vec{x_i} - \vec{x_j}||, h) d\vec{x_j}, \tag{2.17}$$

avec les paramètres suivants :

- A_i : la grandeur (scalaire ou vectorielle) que l'on cherche à estimer (par exemple une densité ou un gradient) ;
- $\vec{x_i}$: le point matériel dont on étudie les propriétés ;
- $\vec{x_j}$: le point voisin courant au point étudié ;
- W : une fonction de lissage (*smoothing kernel* en anglais) appropriée au domaine d'application. Dans le cadre de la simulation de fluide, il s'agit la plupart du temps d'une fonction polynomiale approchant une gaussienne, telle que décrite dans la section 2.4.2.4 ;
- h représente le support de la fonction.

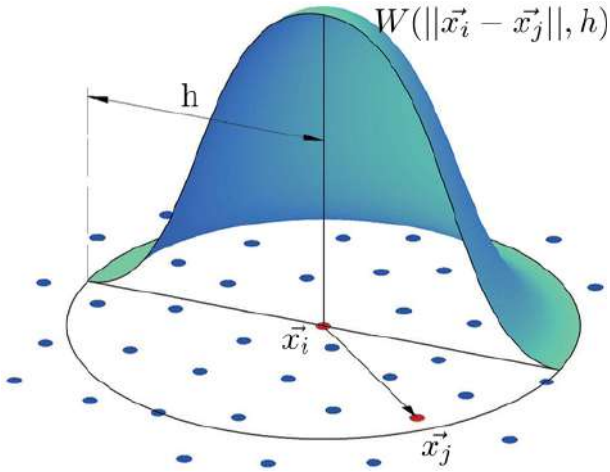


FIGURE 2.6 – *Plus les particules sont loin de la particule centrale, moins elles contribuent à son mouvement. Celles se situant au-delà d'une distance h ont une contribution nulle.*

Le but est donc d'intégrer la fonction à évaluer sur tout le domaine afin de déterminer les propriétés d'une particule de fluide utilisant les particules voisines, soit en simulation numérique :

$$A_i(\vec{x_i}) = \sum_j m_j \frac{A_j}{\rho_j} W(||\vec{x_i} - \vec{x_j}||, h). \tag{2.18}$$

Les termes supplémentaires de cette formule sont :

- m_j : la masse de la particule voisine ;
- A_j : la valeur du champ scalaire ou vectoriel de la particule voisine ;
- ρ_j : la densité de la particule voisine.

Les particules voisines jouent un rôle essentiel dans l'estimation des termes différentiels et des champs scalaires des équations de Navier-Stokes. Il est donc nécessaire d'effectuer au préalable un calcul de voisinage afin de connaître, pour chaque particule, quelles sont les particules dont la distance est comprise par le support de la fonction de lissage utilisée.

2.4.2.3 Calcul de voisinage

Le calcul de voisinage est l'étape critique sur laquelle reposent toutes les performances de la simulation, et forme le goulot d'étranglement des performances de toute simulation SPH. Un calcul de voisinage peu performant occasionne un temps de calcul élevé et un long temps d'accès aux particules voisines, alors qu'un calcul de voisinage efficace permettra aisément d'obtenir des performances interactives. L'algorithme venant tout de suite à l'esprit pour déterminer le voisinage de toutes les particules est celui de la force brute (algorithme 2) :

Algorithme 2 : Calcul de voisinage par force brute.

```
foreach particule i do
    foreach particule j do
        if i != j et ||x_i − x_j|| < h then
            Ajouter la particule j aux voisins de la particule i
```

Bien que cette solution permette rapidement d'obtenir un premier prototype pour réaliser des simulations, il va sans dire que l'algorithme est chronophage et inefficace (complexité en $O(n^2)$).

Une partie de la communauté informatique graphique s'est intéressée au développement de calculs de voisinages plus performants de par leur nature algorithmique d'une part, et d'autre part grâce à leur parallélisation sur carte graphique. Ces algorithmes sont détaillés dans les paragraphes ci-après.

a) Méthode par grille régulière

Les grilles régulières sont les structures accélératrices offrant les meilleures performances pour la simulation de particules SPH [IABT11, HKK07]. Leur construction peut être réalisée en $O(n)$ (n représentant le nombre de particules simulées), et l'accès à une particule en $O(1)$, la structure correspondant concrètement à un tableau. Pour utiliser une structure de grille régulière, on détermine en premier lieu une taille de cellule d. La taille optimale correspond à la valeur du support des fonctions de lissage. On évite ainsi de tester trop de particules dans le cas de cellules trop grandes, ainsi que de tester trop de cellules dans le cas de cellules trop petites.

La première étape consiste à déterminer l'indice en trois dimensions de la cellule dans laquelle se trouve chaque particule. Cela se calcule facilement avec une grille d'origine $\vec{o}$, et de taille $\vec{s}$:

$$\vec{c} = \left(\lfloor \frac{x - \vec{o}_x}{d} \rfloor, \lfloor \frac{y - \vec{o}_y}{d} \rfloor, \lfloor \frac{z - \vec{o}_z}{d} \rfloor \right). \tag{2.19}$$

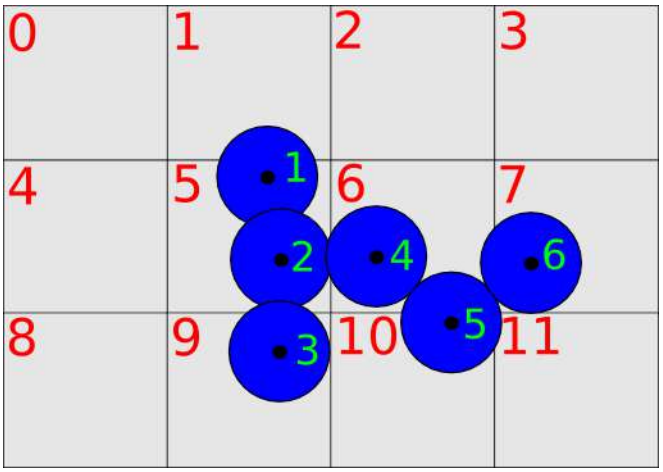


FIGURE 2.7 – *Grille régulière*

FIGURE 2.8 – *Représentation 2D du calcul de voisinage par grille régulière avec tri par index. Les indices en vert sont les indices des particules. Les indices en rouge sont les indices de cellules.*

indice de tableau	liste non triée	liste triée	début de cellule	fin de cellule
0	9 : 3	5 : 1	-	-
1	7 : 6	5 : 2	-	-
2	10 : 5	6 : 4	-	-
3	5 : 2	7 : 6	-	-
4	6 : 4	9 : 3	-	-
5	5 : 1	10 : 5	0	1
6	-	-	2	2
7	-	-	3	3
8	-	-	-	-
9	-	-	4	4
10	-	-	5	5
11	-	-	-	-

FIGURE 2.9 – *Liste d'indices triée. Les couleurs font référence aux indices des particules et des cellules de la figure 2.8.*

Une fois l'indice en trois dimensions déterminé, il est aisé d'en déduire un indice de tableau $1D$:

$$c = k + l \cdot K + m \cdot K \cdot L. \tag{2.20}$$

Les cellules ne stockent pas les particules, mais simplement une référence vers la première ainsi que la dernière particule présentes dans la cellule. Les particules sont par la suite triées en fonction de leur indice en une dimension de cellule, permettant d'améliorer grandement la cohérence des accès dans la mémoire vive. Des méthodes de tri des particules et de leurs propriétés sont accessibles dans la littérature. Le tri *radix* est souvent utilisé car il est facilement parallélisable. Le tri des tableaux de propriétés peut être fait en une seule passe sur GPU à l'aide d'un itérateur de type *zip*. La librairie *thrust* permet de le faire en utilisant le paradigme GPGPU. La figure 2.8 et le tableau 2.9 illustrent le fonctionnement de l'algorithme. Lorsque l'on souhaite accéder aux voisins, il suffit alors simplement d'itérer sur les indices des cellules voisines.

Cette structure de données offre de très bonnes performances, notamment lorsque parallélisée sur carte graphique, pour un domaine fini. Elle a donc été utilisée pour les travaux réalisés et décrits dans le chapitre 3.

Algorithme 3 : Calcul de voisinage basé sur une grille régulière

Calculer l'indice de cellule des particules
Trier les particules et leurs propriétés selon les indices de cellule
Sauvegarder la référence des première et dernière particules de chaque cellule

La principale limite de l'algorithme 3 est que le domaine de la grille est fini. Dans le cas de simulations instables, les particules peuvent être éjectées du domaine de la grille. Dans ces conditions, les accès mémoire deviennent incohérents et la simulation explose.

b) Méthode par table de hachage spatial

Tout comme les méthodes eulériennes, la principale limite des grilles régulières en SPH est que le domaine de simulation du fluide se limite au domaine de la grille et le fluide ne peut pas exister en dehors. Il est donc impossible d'effectuer certaines simulations dont le domaine doit varier. L'exemple le plus probant est celui d'un aquarium qui subit une cassure et dont le fluide s'échappe.

La méthode du calcul de voisinage par table de hachage spatial [IABT11] (*spatial hashmap* en anglais) permet de contourner en partie ce problème. Elle permet de représenter un environnement virtuellement infini. Une fonction de hachage souvent utilisée pour faire correspondre une position de l'espace à une adresse de tableau est présentée en équation 2.21.

$$f(\vec{x}) = \left((\lfloor \frac{x}{d} \rfloor \cdot p1) \text{ xor } (\lfloor \frac{y}{d} \rfloor \cdot p2) \text{ xor } (\lfloor \frac{z}{d} \rfloor \cdot p3) \right) \% m. \tag{2.21}$$

Les paramètres de cette formule sont :

- m, la taille de la table de hachage ;
- x, y, z, la position de la particule ;
- d, la taille de chaque dimension d'une cellule ;
- $p1 = 73856093$;
- $p2 = 19349663$;
- $p3 = 83492791$.

Les nombres $p1, p2$ et $p3$ sont de grands nombres premiers, pour réduire les collisions. Toutefois, pour diminuer le risque de ces collisions, il faut prévoir une valeur de m importante, et une taille d suffisamment faible. Pour accéder aux voisins d'une particules, il suffit en pratique de calculer l'indice des 27 cellules voisines à la particule courante. Ces indices permettent de alors de faire le lien avec l'adresse mémoire des particules voisines. Si cette méthode permet en théorie de simuler un domaine infini, la liste de particules reste toutefois limitée. De plus, cette méthode alloue beaucoup de mémoire inutilement puisque la table de hachage reste en général faiblement remplie [IABT11].

2.4.2.4 Estimation de grandeurs physiques

Conformément à l'équation du bilan de mouvement présentée en section 2.2.2, il est nécessaire d'approcher plusieurs termes scalaires et vectoriels pour approcher une solution numérique de l'équation :

- la densité ρ afin de connaître la masse volumique locale du fluide ;
- le laplacien $\nabla \cdot \nabla \vec{u}$ de vélocité afin de connaître la vélocité moyenne du voisinage de la particule ;
- la pression p, déduite de la densité grâce à une équation d'état (initialement la loi des gaz parfaits) ;
- le gradient de pression ∇p afin de connaître dans quelle direction s'exercent localement les forces de pression.

a) Calcul de densité

Le calcul de densité correspond à l'interpolation d'un champ scalaire permettant d'approcher la masse volumique du fluide à la position d'une particule fluide. Il permet d'approcher par la suite le champ de pression d'une particule ainsi que son gradient.

Le calcul de densité est autosuffisant car seule la masse des particules, constante, est nécessaire pour l'interpoler. La formule initiale pour approcher une grandeur (équation 2.18) peut de ce fait être simplifiée par :

$$\rho_i(\vec{x}_i) = \sum_j m_j W(||\vec{x}_i - \vec{x}_j||, h). \tag{2.22}$$

Une fonction de lissage très couramment utilisée par la communauté [MCG03] pour estimer ce terme est une gaussienne (la courbe bleue de la figure 2.10 illustre cette fonction avec $h = 1$) :

$$W_{poly}(\vec{x}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - ||\vec{x}||^2)^3, & \text{si } \vec{x} \leq h \\ 0, & \text{sinon.} \end{cases} \tag{2.23}$$

Les équations 2.24 et 2.25 peuvent être utilisées pour estimer le gradient et le laplacien du noyau 2.23 (courbe rouge et courbe verte sur la figure 2.10).

$$\nabla W_{poly}(\vec{x}, h) = -\frac{945}{32\pi h^9} \vec{x} (h^2 - ||\vec{x}||^2)^2 \tag{2.24}$$

$$\nabla^2 W_{poly}(\vec{x}, h) = -\frac{945}{32\pi h^9} (h^2 - ||\vec{x}||^2)(3h^2 - 7||\vec{x}||^2). \tag{2.25}$$

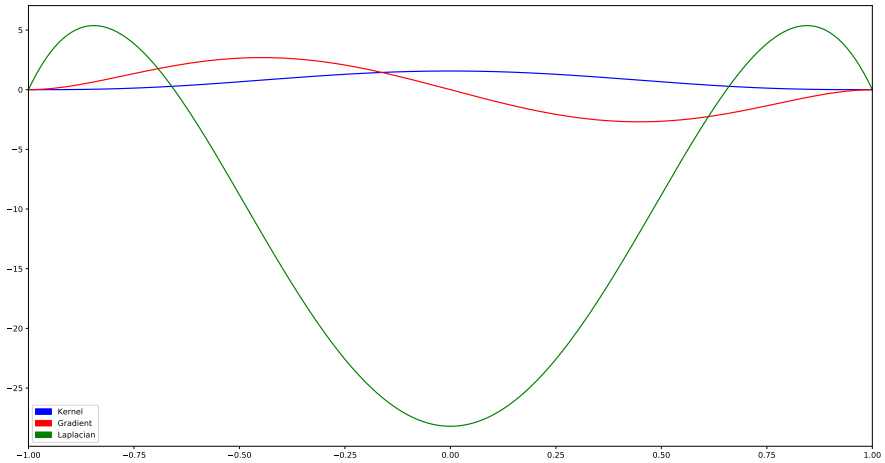


FIGURE 2.10 – *La fonction de lissage souvent utilisée pour le calcul de densité.*

b) *Calcul de pression*

Le calcul de pression au niveau d'une particule est nécessaire pour estimer ensuite un gradient de pression, permettant lui-même d'orienter les particules dans la direction vers laquelle s'exercent en moyenne les forces de pression. Les forces de pression seront inversement proportionnelles à la distance entre les particules : plus elles sont proches, plus elles se repoussent fortement, et elles se rapprocheront (voir la figure 2.11) au contraire plus facilement si elles sont distantes (toujours avec le seuil de la distance support). Une première formulation proposée par la communauté [DC96, MCG03] est la loi des gaz parfaits,

$$p_i = k(\rho_i - \rho_0), \tag{2.26}$$

dont les termes sont :

- p_i, la pression de la particule courante ;
- k, une constante permettant d'ajuster le potentiel de compressibilité du fluide, plus il est elevé, et plus les particules auront tendance à se repousser ;
- ρ_i, la densité de la particule courante ;
- ρ_0, la densité que l'on souhaite approcher, soit $998.29kg \cdot m^{-3}$ pour de l'eau.

Il s'agit d'un point extrêmement sensible de la méthode SPH : une valeur de k trop faible induit des forces de répulsion très faibles entraînant une compressibilité trop élevée pour de l'eau, bien que permettant une stabilité de simulation très bonne et un pas d'intégration élevé (de l'ordre de 10^{-2}). Au contraire, un nombre k élevé permet de pallier les problèmes de compressibilité, au lourd prix d'un pas de simulation très faible (de l'ordre de 10^{-4}). Il s'agit d'un point essentiel de la méthode SPH, et de nombreuses méthodes tentent de permettre un faible pas de temps avec une faible compressibilité ($\leq 0.1\%$). Un tour d'horizon de ces méthodes est présenté en section 2.4.2.9.

c) *Calcul des forces de pression*

Pour approcher les forces de pression, il est peu courant d'utiliser des gaussiennes, présentées dans l'expression 2.23. La fonction de lissage en pic est généralement préférée ([MCG03])

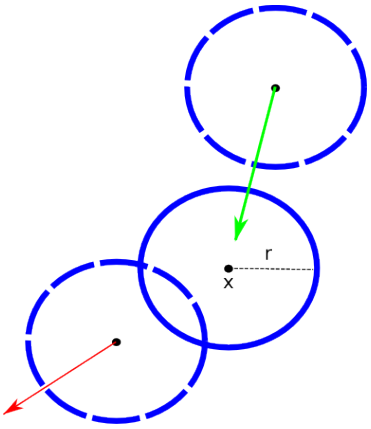


FIGURE 2.11 – *Lorsqu'une particule se rapproche de la particule courante, celle-ci est repoussée dans la direction opposée sous l'influence des forces de pression (particule en bas à gauche). A contrario, elle s'en rapprochera lorsqu'elles sont plus éloignées (particule en haut à droite).*

$$W_{spiky}(\vec{x},h) = \frac{15}{\pi h^6}\begin{cases}(h-||\vec{x}||)^3, & \text{si } 0 \leq ||\vec{x}|| \leq h\\ 0, & \text{sinon.}\end{cases} \tag{2.27}$$

Le gradient de cette fonction (figure 2.13) est défini par :

$$\nabla W_{spiky}(\vec{x},h) = -\frac{45}{\pi h^6}\frac{\vec{x}}{||\vec{x}||}(h-||\vec{x}||)^2. \tag{2.28}$$

Elle est illustrée sur la figure 2.13. La formule SPH pour approcher le gradient de pression présent dans les équations de Navier-Stokes est directement déduite de la formule 2.18,

$$F^i_{pressure} = -\sum_j m_j \frac{p_j}{\rho_j}\nabla W(\vec{x}_i - \vec{x}_j, h). \tag{2.29}$$

L'expression 2.29 ne permet pas d'assurer le principe des actions réciproques (troisième loi de Newton) lorsqu'elle est utilisée pour le calcul de forces internes (pression, viscosité). Ce principe est exprimé par la formule :

$$F_{A\to B} = -F_{B\to A}, \tag{2.30}$$

et indique que les forces exercées par un premier corps, sur un deuxième, doivent correspondre exactement aux forces subies par ce deuxième corps. Pour corriger ce problème, l'utilisation de la formule 2.31 permet de mieux approcher la troisième loi de Newton.

$$F^i_{pressure} = -\frac{m_i}{\rho_i}\nabla p_i = -\frac{m_i}{\rho_i}\cdot\rho_i\sum_j m_j\left(\frac{p_i}{\rho_i^2}+\frac{p_j}{\rho_j^2}\right)\nabla W(\vec{x}_i-\vec{x}_j,h). \tag{2.31}$$

Bien que cette formulation permette d'améliorer l'approche du principe des actions réciproques, elle repose malgré tout sur une estimation de densité et de pression sur la base d'une équation d'état, qui nécessite un faible pas de temps pour assurer un niveau de convergence acceptable [SP09]. Des approches plus récentes ont été proposées par la communauté et permettent d'améliorer grandement la dynamique des particules grâce à des schémas d'estimation du champ de pression plus précis. Nous détaillons ces méthodes dans la section 2.4.2.9.

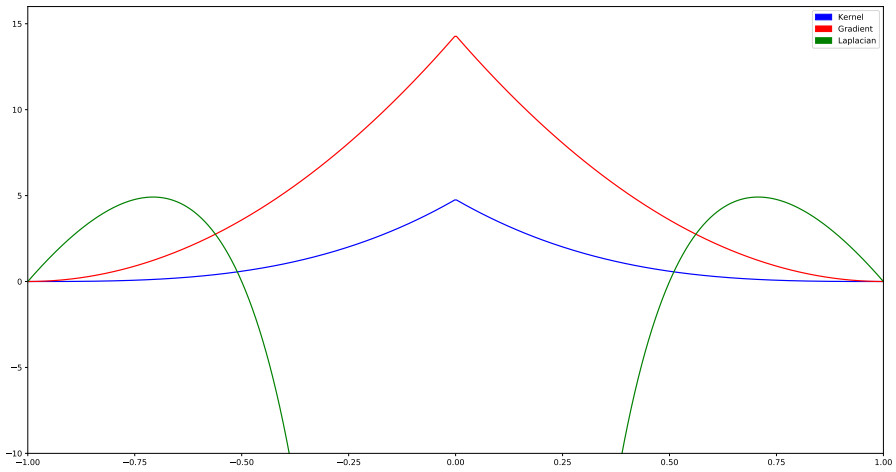


FIGURE 2.12 – *La fonction de lissage en pic, son gradient et son laplacien, utilisés pour l'estimation des forces de pression.*

d) Calcul des forces de viscosité

Le coefficient de viscosité de l'eau est très faible. Dans certains simulateurs les forces de viscosité sont donc négligées. Dans les simulations SPH, une viscosité artificielle est souvent requise pour assurer la stabilité de la simulation. Les forces de viscosité, appliquées à une particule, lui permettent d'avoir un mouvement cohérent par rapport à celui des particules voisines. Les forces de viscosité consistent donc à pénaliser le mouvement d'une particule, s'il s'éloigne de la quantité de mouvement voisin. L'opérateur différentiel laplacien permet de mesurer à quel point un champ vectoriel, en un point donné de l'espace, diffère de ce même champ par rapport à d'autres points de l'espace. Comme pour le calcul des forces de pression, des formulations permettant de mieux approcher le principe des actions réciproques font consensus [MCG03, IOS+14]. L'équation 2.32 permet d'approcher les forces de viscosité en tenant compte de ce principe. Également, le noyau gaussien n'est ici pas recommandé. À la place, le noyau W_{visc} (équation 2.33, dont le laplacien est décrit en équation 2.34) est proposé dans la littérature.

$$F^i_{viscosity} = m_i \nu \nabla^2 \vec{u}_i = m_i \nu \cdot 2 \sum_j \frac{m_j}{\rho_j} \vec{u}_{ij} \frac{\vec{x_{ij}} \cdot \nabla W_{ij}}{\vec{x_{ij}} \cdot \vec{x_{ij}} + 0.01h^2} \tag{2.32}$$

$$W_{visc}(\vec{x}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{\vec{x}^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2\vec{x}} - 1, & \text{si } 0 \leq \vec{x} \leq h \\ 0, & \text{sinon.} \end{cases} \tag{2.33}$$

$$\nabla^2 W_{visc}(h, \vec{x}) = \frac{45}{\pi h^6}(h - \vec{x}). \tag{2.34}$$

Dans nos expériences la formulation exprimée par l'équation 2.32 offre des résultats plausibles pour simuler la dynamique de l'eau ou d'un liquide homogène. Toutefois, elle ne permet pas de représenter le comportement d'écoulements multi-composants, c'est à dire qui comportent plusieurs liquides. Une partie de la communauté s'intéresse aux problèmes de l'estimation du champ viscosité avec de tels écoulements. Nous ne décrivons pas ces méthodes car elles dépassent la portée de cette thèse. Nous redirigeons le lecteur vers l'état de l'art de Ihmsen et al. [IOS+14].

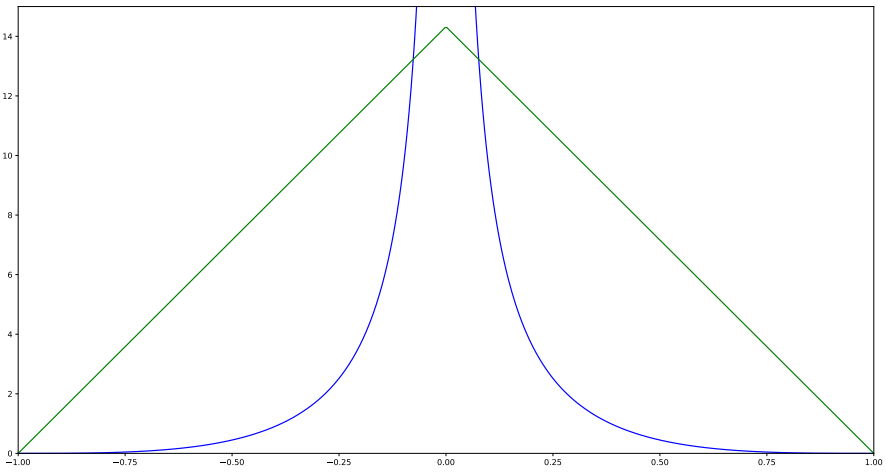


FIGURE 2.13 – *La fonction de lissage et son laplacien, utilisés pour l'estimation des forces de viscosité.*

e) *Calcul des forces de tension de surface*

Plusieurs articles proposent d'approcher les forces de tension de surface s'exerçant à l'interface des liquides. Ces forces s'expliquent grâce aux liaisons des molécules d'eau dont la cohésion est importante à l'interface du fluide. Dans la réalité, ces forces permettent par exemple à des matériaux plus denses que l'eau de flotter. Dans la simulation de liquides basée sur des SPH, elles permettent notamment de réduire la courbure et lisser l'interface du fluide. Plusieurs articles suggèrent d'identifier les particules de surface en utilisant un attribut drapeau [MCG03, BT07] :

$$C_s(\vec{x}) = \sum_j m_j \frac{1}{\rho_j} W(||\vec{x}_i - \vec{x}_j||, h). \tag{2.35}$$

Pour appliquer des forces aux particules d'interface, plusieurs méthodes sont proposées. Müller et al. [MCG03] proposent d'appliquer des forces sur les particules d'interface, pointant vers l'intérieur du liquide, le long de la normale à la surface.

La normale à la surface du liquide au niveau d'une particule d'interface peut être déterminée en calculant le gradient du drapeau. Seules les particules ayant une normale de longueur $||\vec{n}|| > \sigma$ sont considérées comme particules d'interface, afin d'éviter les erreurs numériques de normalisation. Dans nos expérimentations, nous avons déterminé qu'une valeur de $\sigma = 15$ donne de bons résultats pour la plupart des scénarios. L'identification des particules d'interface est également utilisée dans le processus de reconstruction de surface [SCN+16]. Cette méthode est très simple à mettre en place mais donne des forces discontinues car le gradient du drapeau manque de précision et peut fortement varier d'une particule à l'autre.

D'autres approches [BT07, AAT13] permettent d'exercer des forces de cohésion entre les particules, comme c'est le cas entre les molécules d'eau. Ces approches ont pour avantage de mieux conserver le mouvement car les forces exercées sont symétriques, contrairement aux méthodes basées sur l'attribut drapeau. De plus, elles ne dépendent pas de cet attribut qui s'est révélé manquer de précision.

Les forces de tension de surface sont la plupart du temps implémentées dans des systèmes s'intéressant à des effets observables à l'échelle microscopique (e.g. simulation de gouttes d'eau). Les effets dus aux forces de tension de surface sont souvent imperceptibles dans les systèmes à grande échelle. Elles n'y sont donc pas implémentées.

2.4.2.5 Intégration numérique

L'intégration numérique est l'étape de la simulation permettant d'advecter les particules de liquide en fonction des forces internes et externes précédemment calculées. Simuler un fluide en temps-réel consiste à produire une seconde de simulation en une seconde. Pour un pas de temps de 0.01 seconde, il faut donc faire 100 itérations de la boucle de simulation par seconde. Dans la pratique, ces performances sont difficles à atteindre. Un pas de temps aussi élevé occasionne la plupart du temps des instabilités numériques, et des problèmes de collision (si la particule se déplace trop vite, elle peut traverser un maillage ou une particule sans l'intersecter). Simuler le mouvement d'un liquide à 100 images par seconde implique finalement de traiter un faible volume.

Dans la littérature, le pas d'intégration est parfois calculé avec la condition CFL (*Courant-Friedrich-Levy*) :

$$\Delta t = \lambda \frac{h}{||\vec{v_{max}}||}. \tag{2.36}$$

Les paramètres de cette équation sont :

- h, le support des fonctions de lissage ;
- $\vec{v_{max}}$, la vélocité maximale parmi toutes les particules.
- Le paramètre λ permet de restreindre la quantité de déplacement par itération. Avec $\lambda = 1$, la particule ayant la plus forte vélocité aura une longueur de mouvement étant au plus égale à son diamètre.

Une fois le pas de temps déterminé, la quantité de mouvement et le changement de position sont appliqués aux particules grâce à l'intégration numérique. La méthode semi-implicite d'Euler est communément utilisée [IOS+14] :

$$\vec{u}_{t+\Delta t} = \vec{u}_t + \Delta t \vec{F}_t / m \tag{2.37}$$

$$\vec{x}_{t+\Delta t} = \vec{x}_i + \Delta t \vec{u}_{t+\Delta t}. \tag{2.38}$$

D'autres contributions utilisent également d'autres schémas d'intégration, tels que le schéma Leap-Frog ou Verlet [Kelo6]. Certaines approches adaptent l'intégration numérique des particules en fonction de régions d'intérêt. Goswami et Pajarola [GP11] proposent d'utiliser une heuristique du mouvement du liquide. Lorsque le mouvement est globalement rapide, le pas de temps peut être calculé avec la condition CFL. Lorsque le mouvement est plus lent, la stabilité de la simulation peut être assurée avec un pas de temps plus long, proportionnellement aux forces et aux vélocités des particules. Bien que ce type d'approche ait un coût (il faut réaliser une recherche de vélocité maximale dans une liste de taille potentiellement très importante, coûteuse sur des architectures GPGPU [IOS+14]), elles permettent tout de même de réduire le nombre d'itérations du fait d'un pas de temps calculé plus grand.

D'autres contributions proposent également de réaliser une intégration numérique seulement sur les particules dont le déplacement dépasse un certain seuil [GP11, MFRC13]. Dans ces approches, les magnitudes des champs de vélocité et le drapeau sont utilisés pour déterminer l'activité d'une particule. Si ces magnitudes dépassent un certain seuil, elles sont marquées comme étant actives. Les procédures d'intégration numérique ne traitent alors plus que les particules marquées comme telles. Ces approches peuvent notamment s'avérer très efficaces pour simuler le mouvement des liquides dont l'écoulement est fortement hétérogène d'une région à l'autre. Pour des simulations globalement turbulentes, leur intérêt est toutefois limité car peu de particules sont inactives.

2.4.2.6 Détection et réponse aux collisions

La détection des collisions et leur réponse sont deux étapes critiques de la simulation de particules SPH car elles doivent permettre d'assurer la cohérence de la simulation d'un point de vue géométrique (les particules ne doivent pas passer au travers des objets), mais aussi physique (les estimations des champs scalaires et vectoriels doivent rester correctes). Pour la simulation de particules, deux approches sont en général envisagées : les approches représentant les bords à l'aide de maillages triangulés, et celles discrétisant toute la scène de manière uniforme, avec des sphères. Lors des travaux de cette thèse, les deux approches ont été mises en œuvre et chacune comporte des avantages et des limites, que nous décrivons dans cette section.

a) Collisions avec des primitives géométriques

Cette partie traite des intersections entre une sphère et un maillage triangulé.

Les collisions entre des particules de fluide et des maillages triangulés sont généralement composées de deux étapes :

- la détection de la collision : il y a une intersection entre les deux primitives géométriques ;
- la réponse à la collision : c'est à dire la procédure qui modifie le comportement de la particule suite à la collision.

L'ouvrage de Ericson [Eri04] fournit une réponse très performante à ce problème. Il calcule premièrement le point le plus proche du triangle par rapport au centre de la sphère. Il considère ensuite qu'il y a une collision lorsque la distance entre ce point et le centre de la sphère est inférieure au rayon de la sphère.

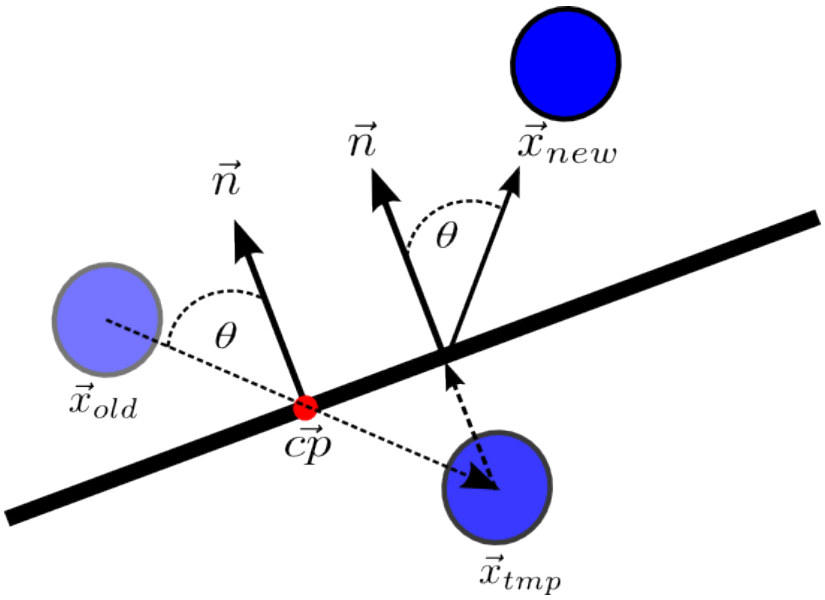


FIGURE 2.14 – *Détection et réponse à une collision entre une particule et un triangle. $\vec{x_{old}}$ représente la position de la particule avant l'intersection avec le maillage. $\vec{x_{tmp}}$ représente sa position temporaire, lorsque la collision est détectée. Une fois la collision traitée, la particule est placée à la position $\vec{x_{new}}$.*

Plusieurs approches pour traiter la réponse à une collision avec un maillage existent. Kelager [Kel06] propose de traiter ce problème avec une approche par impulsion et projection (figure 2.14) :

$$\vec{u}_i = \vec{u}_i - (1 + c_R \frac{d}{\Delta t ||\vec{u}_i||})(\vec{u}_i \cdot \vec{n})\vec{n}. \tag{2.39}$$

Les différents paramètres de cette équation sont :

- c_R, le coefficient de restitution de la surface touchée ; une valeur de 1 occasionne des rebonds parfaits, alors qu'une valeur de 0 absorbe toute l'énergie cinétique de la particule. $c_R \approx 0.5$ donne en général de bons résultats ;
- d correspond à la profondeur de pénétration de la particule dans la surface ; elle est calculée grâce au point de contact $\vec{cp}$;
- $\vec{n}$ correspond à la normale à la surface touchée.

Dans le cadre de nos expérimentations, cette méthode (que nous avons utilisée dans le cadre des travaux présentés au chapitre 3) fournit de bons résultats, mais comporte toutefois un inconvénient rhédibitoire. Lorsque le pas de temps choisi est élevé (typiquement $\Delta t = 0.01$), les collisions ne sont pas toujours détectées car une particule peut avoir un déplacement important et passe alors au travers de la surface, sans qu'il n'y ait de détection de collision. Pour assurer la stabilité de la simulation, il est donc essentiel d'utiliser la condition CFL pour calculer un pas de temps suffisamment court. La plupart des implémentations actuelles semblent notamment utiliser plutôt un échantillonnage des bords avec des sphères pour cette raison.

b) *Échantillonnage des bords avec des sphères*

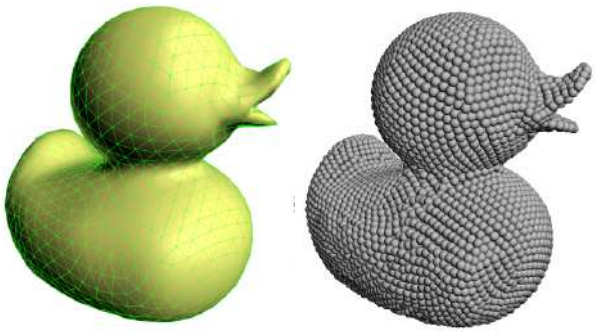


FIGURE 2.15 – *Echantillonnage d'un maillage triangulé avec des sphères. Issu de [ACAT13].*

Les détections de collisions entre des sphères et des primitives géométriques conduisant systématiquement à réduire fortement le pas de temps de la simulation, la communauté s'est davantage intéressée à mettre en œuvre des approches permettant d'échantillonner les bords du domaine de simulation avec des sphères (figure 2.15), dans le but de calculer des forces de répulsion entre les particules de bord et les particules de liquide.

Akinci et al. [AIA+12] proposent une méthode de calcul de forces d'adhésion et de friction entre le liquide et des objets solides. Akinci et al. [ACAT13] fournissent une réponse aux problématiques d'échantillonnage et de répulsion qui semble faire consensus au sein de la communauté informatique graphique [IOS+14].

En premier lieu et en amont de la phase de simulation, leur méthode échantillonne les maillages triangulés en discrétisant d'abord les sommets, puis les arêtes, et enfin les faces (figure 2.16). Chaque sommet est représenté par une sphère dont le rayon est identique à celui des particules de fluide. Des particules sont ensuite uniformément placées le long de chaque arête de chaque triangle. Les arêtes éventuellement redondantes souvent présentes dans des formats de type *obj* peuvent facilement être écartées en utilisant une structure

garantissant l'unicité, par exemple un ensemble. Les faces sont ensuite échantillonnées grâce à un remplissage par balayage (*scan-line*).

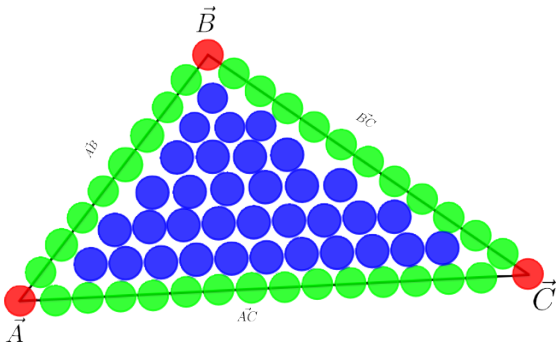


FIGURE 2.16 – *Discrétisation du triangle avec des sphères. Dans un premier temps les sommets sont traités, puis les arêtes, puis l'intérieur de la face en utilisant un algorithme de remplissage par balayage.*

Un potentiel de répulsion est ensuite attribué à chaque particule de bord, dépendamment de son nombre de particules de bord voisines (équation 2.40).

$$V_{b_i} = \frac{1}{\sum_j W_{ik}(||\vec{x}_i - \vec{x}_j||, h)}. \tag{2.40}$$

Les estimations de densité et de pression des particules de fluide doivent également être influencées par les particules de bord [AIA+12] :

$$\rho_i = \underbrace{\sum_j m_i W(||\vec{x}_i - \vec{x}_j||, h)}_{\text{Contribution des particules fluides}} + \underbrace{\sum_k (\rho_0 V_{b_k}) W(||\vec{x}_i - \vec{x}_j||, h)}_{\text{Contribution des particules de bord}}. \tag{2.41}$$

Enfin, des forces d'adhésion des particules fluides sur le solide peuvent être approchées en utilisant la formule suivante [AIA+12] :

$$F^a_{f_i \leftarrow b_k} = \beta \rho_0 V_{b_i} (\vec{x}_i - \vec{x}_k) W(||\vec{x}_i - \vec{x}_k||, h). \tag{2.42}$$

Le paramètre β est le coefficient d'adhésion du matériau de bord. Akinci et al. [AAT13] proposent d'utiliser le noyau décrit en équation 2.43 afin d'améliorer l'estimation des forces d'adhésion d'une particule.

$$W(r,h) = \frac{0.007}{h^{3.25}} \begin{cases} \sqrt[4]{-\frac{4r^2}{h} + 6r - 2h}, & \text{si } 2r > h \text{ ou } r \leq h \\ 0, & \text{sinon.} \end{cases} \tag{2.43}$$

La modification de l'estimation du champ de pression tenant compte des particules de bord permet de repousser les particules de fluide des objets de collision. L'atout majeur de cette méthode est qu'elle permet d'unifier toute la simulation, utilisant uniquement le formalisme SPH, pour les forces internes et ses interactions avec les conditions aux bords. Par ailleurs, le fait d'échantillonner les bords avec des particules évite l'utilisation d'une seconde structure accélératrice car celles-ci peuvent être contenues dans la même grille que les particules de liquide, simplifiant l'architecture du simulateur développé. Pour des solides indéformables et immuables, l'échantillonnage et le calcul du potentiel de répulsion peuvent être calculés une seule fois en amont du processus de simulation. Les solides déformables (et donc en déplacement) doivent en revanche être ré-échantillonnés à chaque pas de simulation pour éviter les problèmes de sous-échantillonnage dus aux déformations, ce qui implique un coût de calcul non négligeable.

2.4.2.7 Algorithme d'une simulation SPH.

L'algorithme d'une simulation SPH avec équation d'état (souvent abrégé par SESPH pour *State Equation* SPH) suit généralement quatre étapes, décrites dans l'algorithme 4. Au début de chaque itération, le voisinage est mis à jour préférablement avec la méthode par grille régulière décrite à la section 2.4.2.3. Cela permet d'estimer ensuite la densité (équation 2.22) et la pression au niveau d'une particule en utilisant l'équation de Tait (équation 2.46). Ces deux champs permettent par la suite d'estimer les forces internes (pression, viscosité et tension de surface). L'attraction gravitationnelle s'exerçant de manière identique sur chaque particule, le formalisme SPH n'est pas nécessaire et celle-ci est le plus souvent approchée par $F_{grav}(\vec{x}_i) = \vec{g} \cdot m_i$. La somme des forces appliquées à une particule permet de déterminer l'accélération et subséquemment de mettre à jour la vélocité et la position de chaque particule. Nous illustrons une séquence d'une simulation SPH obtenue avec notre implémentation à la figure 2.17.

Algorithme 4 : Boucle de simulation SPH

```
Mettre à jour le voisinage
foreach particule do
    Calculer sa densité
    Calculer sa pression
foreach particule do
    Calculer les forces de pression
    Calculer les forces de viscosité
    Calculer les forces externes
    (Calculer les forces de tension de surface)
foreach particule do
    Calculer la vélocité de la particule
    Mettre à jour la position de la particule
```

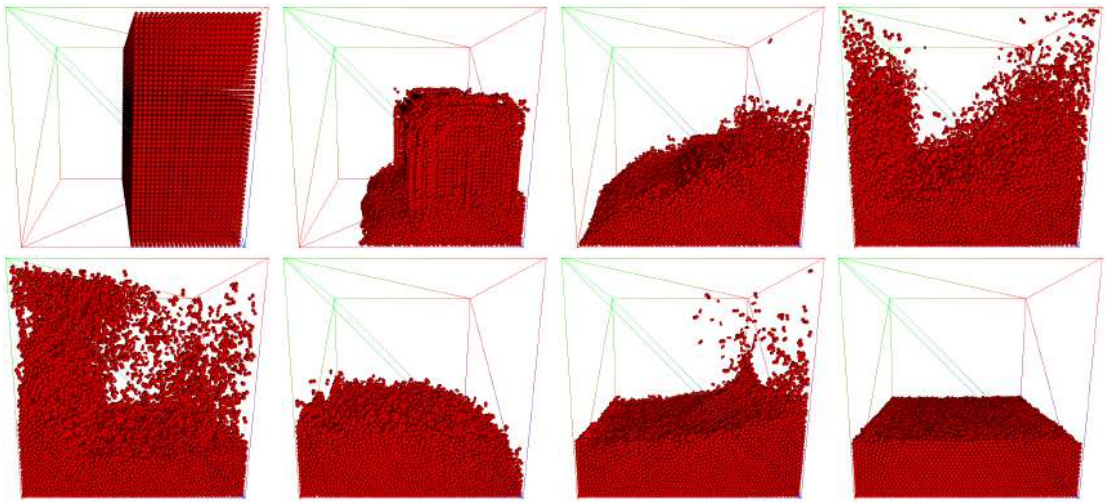


FIGURE 2.17 – *Plusieurs étapes lors de la simulation de la chute d'un cube d'eau dans un cube.*

2.4.2.8 Modèles surfaciques

Dans le but d'améliorer les performances des simulateurs, certains auteurs se sont intéressés au développement de modèles permettant de simuler uniquement le mouvement de la surface

du fluide en approchant numériquement une solution aux équations de Saint-Venant (décrites en section 2.2.2). Lee et Han [LH10] ont d'abord proposé un modèle basé sur les SPH pour approcher les équations de Saint-Venant en ignorant la géométrie du sol, et en proposant un ensemble de noyaux 2D pour estimer les champs scalaires et vectoriels. Bucher et al. [BBC+11] ont étendu ce modèle en intégrant la hauteur du sol dans l'équation de bilan du mouvement, et en proposant un modèle d'interaction fluide/solide. Nous illustrons ces méthodes en figure 2.18.

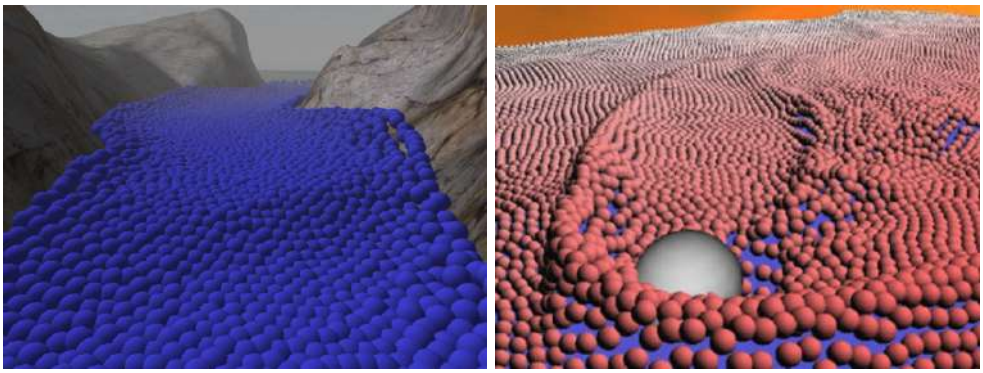


FIGURE 2.18 – *Des exemples de simulation de SPH surfaciques [LH10, BBC+11].*

Le principal atout de ces deux méthodes est de réduire le nombre de particules, car seule la surface est modélisée, ce qui suffit en général largement à couvrir les besoins physiques nécessaires dans le cadre de simulations peu turbulentes, ou des écoulements laminaires. La méthode SPH étant facilement parallélisable sur GPU, ces méthodes surfaciques sont d'autant plus adaptées à un contexte de simulation temps réel. D'après les tests que nous avons réalisés, la principale limite qui demeure dans la méthode SWSPH (*Shallow Water* SPH) est son instabilité pour des écoulements turbulents et à forte vélocité.

2.4.2.9 Dérivation aux modèles peu compressibles

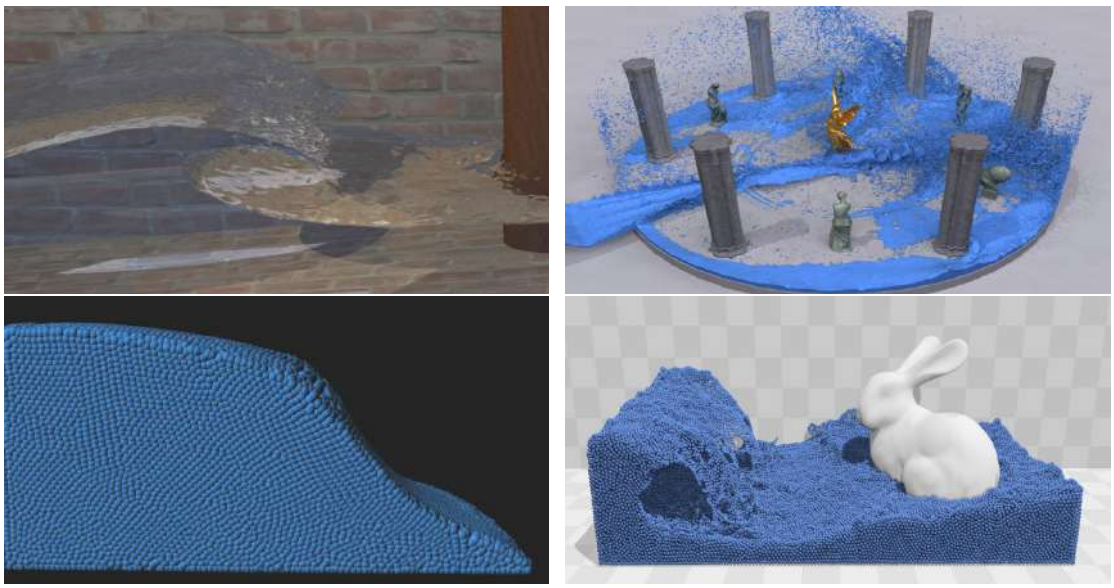


FIGURE 2.19 – *Des exemples de simulations peu compressibles [SP09, BK15, BT07, MM13].*

La principale faiblesse de la méthode SPH pour animer un fluide réside dans le calcul de pression, et donc par effet de bord dans l'approche du gradient de pression de l'équation de bilan de la quantité de mouvement. Dans une simulation SPH, chaque particule devrait comporter en moyenne environ 30 à 40 particules voisines (excepté pour certaines régions de haute turbulence tels que les éclaboussures, où le nombre de particules voisines est moins important). L'utilisation de l'équation des gaz parfaits pour calculer la pression d'une particule ne permet pas de respecter cette contrainte. Bien qu'il soit possible de la paramétrer afin d'influencer le potentiel de compressibilié du fluide, le résultat est soit instable pour des pas de temps raisonnables (condition nécessaire pour une simulation interactive), soit peu réaliste pour des fluides peu compressibles (densités et pressions trop élevées, le fluide est comprimé tel un gaz).

De nombreux articles de la communauté (nous les illustrons en partie en figure 2.19) tentent de lever cette limite, avec pour but de simuler de l'eau en respectant au mieux la condition d'incompressibilité (la variation de densité est presque nulle) et avec le pas de temps le plus élevé possible. Les solutions proposées peuvent être classifiées selon trois familles :

- les méthodes basées sur une équation d'état, les moins réalistes, mais permissives sur le pas de temps ;
- les méthodes approchant une équation de Poisson, offrant de très bons résultats au prix d'un pas de temps très court ;
- les modèles prédictifs/correctifs, offrant à ce jour le meilleur compromis.

Cette section fait un tour d'horizon de ces trois approches.

a) Équation d'état

Les simulations utilisant une équation d'état de manière non itérative sont les méthodes les plus simples pour faire des simulations SPH. Plusieurs équations d'état ont été proposées dans la bibliographie. La première est l'équation des gaz parfaits (équation 2.44).

$$p_i = \rho_i - \rho_0. \tag{2.44}$$

L'équation des gaz parfaits permet d'utiliser un pas de temps de simulation élevé (de l'ordre de 10^{-2}), au détriment d'une forte compressibilité du fluide.

Desbrun et Cani [DC96] introduisent une version modifiée de cette équation (équation 2.45), en ajoutant une constante k permettant de contrôler le potentiel d'incompressibilité du fluide.

$$p_i = k \cdot (\rho_i - \rho_0). \tag{2.45}$$

Dans cette seconde équation, l'incompressibilité, mais aussi la stabilité de la simulation, sont proportionnelles à la constante k. Ainsi, une plus grande valeur de k nécessite un pas de temps de simulation plus court, et inversement.

Becker et Teschner [BT07] proposent d'utiliser l'équation de Tait (équation 2.46) pour estimer les forces de pression s'exerçant sur une particule, conjointement à l'utilisation d'un modèle de force pour modéliser les effets de tension de surface.

$$p_i = B\left(\left(\frac{\rho}{\rho_0}\right)^7 - 1\right). \tag{2.46}$$

Dans cette équation, B est une constante permettant d'ajuster la compressibilité du liquide. Pour les simulateurs utilisant une équation d'état afin d'approcher la pression d'une particule, c'est cette dernière formulation qui semble prévaloir dans la bibliographie [IOS+14].

b) *Équation de Poisson*

Plusieurs contributions basées sur les SPH proposent d'estimer les forces de pression grâce à une projection de Poisson, qui est originellement une technique employée dans les systèmes d'inspiration eulérienne. Premoze et al. [PTB+03] introduisent la projection de Poisson aux systèmes particulaires au début des années 2000. Ils approchent une solution à une équation de Poisson sur la base d'un système de particules mettant en œuvre la technique de l'estimation des forces en deux passes. L'objectif est d'estimer dans un premier temps les forces externes et internes exceptées celles de pression pour calculer des positions intermédiaires, puis de calculer les forces de pression sur la base de ces positions intermédiaires. Elle est particulièrement utilisée dans les schémas de type prédiction et correction (voir paragraphe suivant). Ihmsen et al. [ICS+14] proposent un système itératif, les SPH implicites et incompressibles, approchant un système linéaire résolu par gradient conjugué afin de corriger progressivement les erreurs d'estimation de densité, jusqu'à parvenir à une moyenne inférieure à un seuil utilisateur. Alternativement, ils proposent d'utiliser la méthode de Jacobi (méthode itérative par inversion de matrice) pour résoudre le système. Goswami et al. [GEF15] pallient la lenteur du système (qui résulte du calcul du système linéaire) en proposant une implémentation sur GPU, et tirant parti de la mémoire des textures sous CUDA. Plus récemment, Cornelis et al. [CIPT14b] proposent une méthode hybride (appelée IISPH-FLIP) pour tirer parti de la conservation de la masse de la méthode IISPH avec les performances de la méthode FLIP. Losasso et al. [LTKF08] simulent des eaux turbulentes avec une simulation eulérienne pour représenter le volume d'eau, et avec des SPH pour représenter les éclaboussures. He et al. [HLL+12] proposent une méthode de résolution locale et itérative d'équation de Poisson dans le contexte de simulation de liquides visqueux.

Les méthodes basées sur la résolution d'une équation de Poisson produisent des résultats réalistes car elles permettent d'estimer avec précision les forces de pression qui s'exercent sur les particules, ce qui permet d'améliorer nettement certains effets turbulents (tels que les éclaboussures) difficiles à obtenir avec des SPH utilisant une équation d'état. En revanche, ces systèmes nécessitent des pas de temps assez faibles (dans nos expérimentations, un pas de temps de 0.001 produit des résultats réalistes et stables) et il est de ce fait difficile d'obtenir des simulations interactives avec ces méthodes.

c) *Modèles predictifs correctifs*

Les modèles utilisant une équation d'état nécessitent d'utiliser un pas de temps très court lorsque le liquide simulé est peu compressible, afin d'éviter les erreurs d'estimation de la densité des particules, et des forces de pression qui en résultent. Pour corriger cette limite, la communauté s'est largement intéressée au développement de modèles prédictifs/correctifs. Leur objectif est d'estimer d'abord toutes les forces exceptées celles de pression, puis de calculer des forces de pression en plusieurs itérations, en mettant à jour les positions des particules à chaque itération et en ré-estimant les densités afin de minimiser l'erreur commise à la première estimation.

La première contribution mettant en œuvre ce paradigme, de Solenthaler et Pajarola [SP09], a permis de réduire l'erreur d'estimation de densité à 0.1%. Le schéma itératif qu'ils proposent est basé sur une prédiction de l'erreur commise, conduisant le solveur à calculer des forces de pression jusqu'à ce que celle-ci soit inférieure à un seuil utilisateur ou à un nombre plafond d'itérations. Les calculs itératifs ont pour paramètres des positions et vélocités intermédiaires, recalculées à chaque itération.

Macklin et Müller [MM13] développent un modèle inspiré de la simulation des solides [MHHR07] en intégrant un ensemble de contraintes indépendantes d'un calcul de forces de pression. Ces contraintes sont résolues en utilisant la méthode de Jacobi. Bender et Koschier

[BK15] proposent une méthode corrigeant de manière itérative la vélocité des particules, en plus des forces de pression.

Ces méthodes revendiquent une perte de volume inférieure à 0.01%. Elles ont pour principale limite un temps de calcul bien plus important pour chaque itération de la simulation, puisqu'elles requièrent plusieurs itérations pour estimer un champ de pression convenable. En revanche, cette limite est également leur point fort car la correction de l'erreur rend ces méthodes bien plus permissives quant au choix du pas de temps. Dans nos expériences, une simulation SPH utilisant une équation d'état devient instable dès lors que le pas de temps est supérieur à 0.001 (1000 images par seconde sont requises pour obtenir une simulation temps réel), à cause des erreurs accumulées lors de l'estimation de densité. À titre de comparaison Macklin et Müller [MM13] revendiquent une stabilité de simulation en utilisant un pas de temps de 0.016 (62 images par seconde sont requises pour produire une simulation temps-réel).

2.4.2.10 Modèles adaptatifs

Simuler un grand nombre de particules est extrêmement coûteux en temps de calcul. Ce coût s'explique notamment par la necessité de calculer puis d'accéder plusieurs fois au voisinage de chaque particule. En outre, ce coût est très élevé lorsqu'il y a plusieurs millions de particules à traiter.

Une partie de la communauté s'est intéressée à réduire le nombre de particules tout en maintenant le volume total du fluide. Une autre partie s'est interessée à ne calculer un mouvement de particules que lorsque cela est utile. Nous décrivons ici les méthodes d'adaptativité géométrique, et quelques éléments d'adaptativité temporelle sont donnés en section 2.4.2.5, toujours dans le contexte de la simulation de liquides. Récemment, Manteaux et al. [MWN+16] publient une étude exhaustive sur les méthodes physiques adaptatives, et non cloisonnée à l'animation de fluide.

a) Adaptativité géométrique

Dans une simulation de liquide, on peut la plupart du temps distinguer des zones turbulentes et des zones plus calmes. Avec une simulation SPH, ces deux types de régions sont pourtant échantillonnées de la même manière, la taille des particules étant constante. Il peut donc paraître discutable que les temps de calcul soient identiques quelle que soit l'activité dynamique du liquide, ou quelque soit l'intérêt visuel de la zone simulée. La communauté s'est rapidement intéressée à échantillonner les particules simulées en fonction de leur intérêt global. Des méthodes basées sur des critères géométriques [DC99, APKG07, ZSP08], physiques [YW+09, OK12], basées sur la position de l'observateur [BG11] ou sur le choix de l'utilisateur [HS13] (qui peut spécifier manuellement une région d'intérêt) ont été élaborées.

Adams et al. [APKG07] utilisent une représentation hiérarchique des particules. Ils proposent de déterminer la taille d'une particule en fonction de sa distance par rapport à l'axe médian du volume. De cette manière, les particules proches de toute surface du fluide seront plus fines que celles plus proches de l'axe médian. Ils proposent de subdiviser les particules en deux, positionnées de manière symétrique autour de la particule supprimée. Elles sont aussi placées de telle manière qu'aucune des deux particules ne soit plus proche que la particule subdivisée, par rapport aux autres particules. Deux particules de même taille et distantes de la surface peuvent être fusionnées en position $\vec{x}_i = \frac{\vec{x}_j + \vec{x}_k}{2}$, uniquement si elles sont suffisamment éloignées des autres afin d'éviter des problèmes d'estimation du champ de densité. Des particules de tailles différentes sont considérées comme étant voisines lorsque $||\vec{x}_i - \vec{x}_j|| \leq \max(r_i, r_j)$, r correspondant au rayon d'une particule. Zhang et al. [ZSP08] proposent de subdiviser selon la distance à la surface, en identifiant les particules de surface

les plus proches avec l'attribut drapeau. Ils proposent de subdiviser une particule en quatre, placées aux sommets du tétraèdre centré à la position de la particule subdivisée. Ces solutions ne permettent toutefois pas d'assurer un niveau de détail dans certaines régions turbulentes, le critère d'échantillonnage étant purement géométrique. Yan et al. [YW+09] proposent d'enrichir ces conditions d'échantillonnage, en tenant compte de l'énergie des particules. Cette solution permet en outre d'améliorer le niveau de détail dans les régions turbulentes mais éloignées de la surface.

Solenthaler et Gross [BG11] proposent de faire deux simulations simultanées (l'une avec une haute résolution, et l'autre avec une faible résolution), et d'animer et afficher les parties nécessaires en fonction de la position de l'observateur. Orthmann et Kolb [OK12] ont développé une méthode représentant le fluide en sous-volumes (*blend-sets*), chacun étant représenté à la fois par un ensemble de particules de haute résolution et de faible résolution. Ils adaptent le formalisme SPH pour une interpolation entre des *blend-sets* de différentes résolutions, et utilisent un pas de temps adaptatif pour limiter les erreurs d'estimation de pression lors des fusions de particules. Leur modèle s'adapte par ailleurs à un modèle de simulation peu compressible.

Les méthodes adaptatives sont utilisées pour améliorer les performances globales du système en réduisant le nombre de particules, et permettent d'effectuer des simulations avec niveau de détail. En revanche, les opérations de fusion ou de subdivision qu'elles impliquent peuvent résulter en des problèmes d'erreurs d'estimation du champ de pression.

2.4.3 Simulation par des approches hybrides

En simulation numérique, nous avons pu voir que les paradigmes lagrangien et eulérien, ainsi que les méthodes procédurales ont des avantages et des limites. Les systèmes lagrangiens ont une masse constante et représentent très bien des effets turbulents, mais des estimations de densité et de forces de pression sont coûteuses en temps de calcul, malgré des performances intéressantes. Les systèmes eulériens permettent d'assurer une simulation sans divergence, mais pâtissent d'une lenteur plus importante et animent difficilement certains effets à petite échelle. Les méthodes procédurales, extrêmement rapides, ne reposent pas sur la physique et ne profitent pas pleinement de la représentation en 3D. Certaines méthodes tentent de concilier deux, voire parfois trois de ces familles de méthodes.

Chentanez et al. [CMK14] représentent une région d'intérêt par des particules au niveau de la surface du liquide, et par un système eulérien plus en profondeur. La hauteur de la surface pouvant varier, ils proposent également une méthode de transition d'un système vers l'autre. Les régions de plus faible intérêt sont représentées par un champ de hauteur gouverné par les équations de Saint-Venant. Raveendran et al. [RWT11] développent un modèle s'inspirant de la méthode FLIP [BR86]. Ils utilisent une grille eulérienne pour réaliser une projection de Poisson, dont ils transfèrent les forces de pression résultantes aux particules comprises dans les cellules respectives. Conjointement, ils appliquent une correction de densité aux particules. Cornelis et al. [CIPT14a] utilisent un modèle d'inspiration similaire en transférant le résultat de la projection de pression réalisée selon la méthode FLIP, dans les particules SPH des cellules correspondantes. Une méthode analogue pour la simulation de bulles a été proposée par Lee et al. [LHK09].

Les systèmes utilisant une approche hybride permettent de représenter chaque région du fluide avec un système adéquat (e.g. une carte de hauteur animée pour les eaux profondes, et des systèmes physiques dans les région à forte turbulence), ce qui permet d'améliorer les performances globales du système. En revanche, la démarche d'implémentation requise est souvent complexe, et l'interaction entre les modèles pose souvent des problèmes de nature physique (e.g. lorsqu'un objet flotte sur une zone de transition) car les systèmes sont hétérogènes

par essence et ne représentent pas les mêmes phénomènes. L'interaction entre ces modèles ne peut pas être décrite par un modèle physique.

2.4.4 Animation et contrôle de fluide

L'animation de fluide en synthèse d'image est la plupart du temps, au vu de la littérature existante, un processus qui exclut l'intervention de l'utilisateur sur la phase de simulation. Il existe toutefois, dans le cas de la simulation de fluide, quelques contributions qui proposent d'injecter une possibilité de contrôle dans des systèmes basés physique. Dans cette section nous présentons ces quelques méthodes.

L'approche proposée par Mihalef et al. [MMS04] aborde ce problème en utilisant une bibliothèque de vagues déferlantes pré-existantes en deux dimensions. Avec ce système, l'utilisateur sélectionne un ensemble de profils pour contrôler les vagues dans le temps et dans l'espace en utilisant une simulation eulérienne. La modélisation et le contrôle de la vague dépendent d'un ensemble de profils 2D et de leurs propriétés physiques, mais aucune interaction n'est possible pendant la simulation et le résultat obtenu est difficile à prévoir par un artiste. McNamara et al. [MTPS04] utilisent une approche par vecteurs de contrôle pour influencer les champs de force s'exerçant sur le liquide.

Rasmussen et al. [REN+04] définissent un ensemble de types de particules de contrôle, qui permettent de maîtriser localement une propriété du liquide qui leur est propre (e.g. particule de contrôle de la viscosité). Thürey et al. [TKPR06] proposent d'utiliser de telles particules dans des systèmes eulériens et SPH pour modifier localement la vélocité et les propriétés physiques du fluide, permettant à celui-ci de prendre la forme d'un maillage triangulé. Raveendran et al. [RTWT12] définissent des forces d'attraction en fonction de la distance à un maillage triangulé animé. Plus récemment, Manteaux et al. [MVW+16b] ont développé une méthode permettant de copier/coller des éléments d'intérêt d'une simulation précalculée dans une simulation finale, sans simulation intermédiaire.

Les méthodes de contrôle de fluide proposées dans la littérature permettent de modifier localement la dynamique du fluide. Toutefois, ces modèles ne permettent pas de contrôler la dynamique d'une vague selon des paramètres physiques (i.e. amplitude, vélocité), et ils ne permettent pas non plus de la contrôler durant la phase de simulation. La méthode que nous décrivons au chapitre 3 permet de lever cette limite.

2.5 Rendu de phénomènes océaniques en informatique graphique

La géométrie des modèles physiques utilisés en simulation numérique ne correspond en général pas à la géométrie observée dans la réalité. Souvent simulé à l'aide d'un nuage de points, un fluide est à notre échelle, un ensemble de volumes fermés par une surface continue. De nombreuses méthodes ont été développées au cours de ces trois dernières décennies pour approcher une surface plausible à partir d'un nuage de points. L'essor de la simulation de liquides a notamment permis une augmentation significative de contributions dans ce sens. Pour visualiser la surface d'un liquide représenté par des points mobiles, deux paradigmes sont généralement mis en opposition : d'une part les méthodes reconstruisant une surface triangulée, souvent délicates à mettre en œuvre avec des performances interactives, et d'autre part les méthodes utilisant l'architecture de la carte graphique pour afficher une primitive géométrique par particule, donnant l'impression d'un volume fermé lorsque leur densité est suffisante.

Ces dernières sont souvent préférées pour des applications avec des contraintes temps réel. Les méthodes basées sur des polygonisations sont préférées pour des applications basées sur de la simulation d'éclairage réaliste *a posteriori*, pour des rendus de haute qualité, car elles offrent un niveau de détail souvent bien supérieur, au prix d'un coût de calcul ne permettant pas des performances interactives.

Le rendu de liquides utilisant une méthode de simulation d'éclairage approche la plupart du temps les lois optiques de l'interaction entre la lumière et la matière, qui souvent sont incompatibles avec un affichage en temps interactif. Pour approcher au mieux la réalité, les méthodes basées sur un lancer de rayon et une intégration de Monte-Carlo sont nécessaires malgré leur aspect chronophage.

Après un tour d'horizon des méthodes de visualisation et d'extraction de surface, nous présentons les bases théoriques fondamentales de l'interaction entre la lumière et la matière, puis nous présentons les méthodes majeures permettant de calculer une image réaliste dans le respect de ces concepts.

2.5.1 Représentation géométrique du domaine océanique simulé

2.5.1.1 *Point Splatting*

Macklin et Müller [MM13] identifient les particules de surface à l'aide du drapeau et visualisent ces dernières avec la technique du *point splatting*. Ils améliorent la qualité des *splats* en affichant des ellipsoïdes pour chaque particule, avec un ensemble de noyaux anisotropes introduits par Yu et Turk [YT13]. Ils ajoutent également un effet de transparence du volume d'eau en utilisant une méthode de filtrage dans l'espace écran décrite ci-dessous.

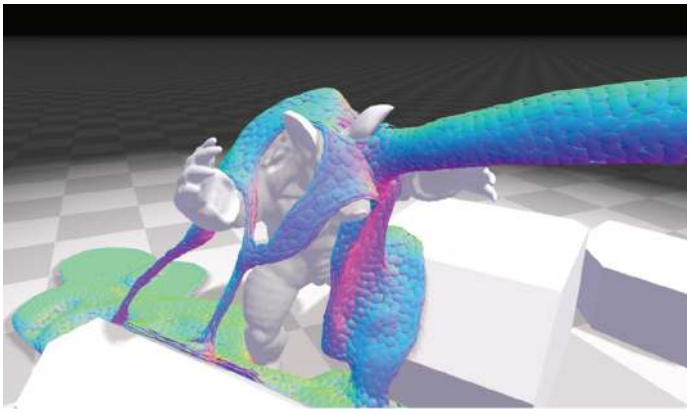


FIGURE 2.20 – *Exemple de rendu utilisant la technique du Ellipsoid Point Splatting [MM13].*

2.5.1.2 *Screen Space Filtering*

Van der Laan et al. [vdLGS09] proposent la méthode du *Screen Space Filtering*. Elle permet de visualiser la surface du fluide, sans reconstruction à base de polygones tels que les *Marching cubes* [LC87], en utilisant directement un *fragment shader* exploitant directement la liste des particules. Considérant les particules comme des sphères, ils proposent de visualiser directement le tampon de profondeur de la carte graphique en utilisant un filtre gaussien afin de limiter les discontinuités. La figure 2.21 illustre le rendu d'une animation basée sur les SPH, utilisant cette méthode.

FIGURE 2.21 – *Exemples de rendus utilisant la technique du Screen Space Filtering. Issu de [MM13].*

2.5.1.3 *Marching Cubes*

À la fin des années 1980, Lorensen et Cline [LC87] proposent la méthode des *marching cubes* (adaptation 3D des *marching squares*), pour construire le maillage triangulaire de l'isosurface d'un champ scalaire dans une grille 3D. Dans des simulations SPH, le champ drapeau [MCG03] des particules est utilisé comme paramètre d'isosurface. Cette méthode est éprouvée et fournit des résultats de bonne qualité. Toutefois, obtenir un résultat avec un niveau de détail acceptable requiert d'utiliser une grille avec des cellules fines, occasionnant des coûts de calcul parfois prohibitifs. Certaines méthodes pallient cette limitation en visualisant directement l'isosurface avec des *metaballs* [ZSP08].

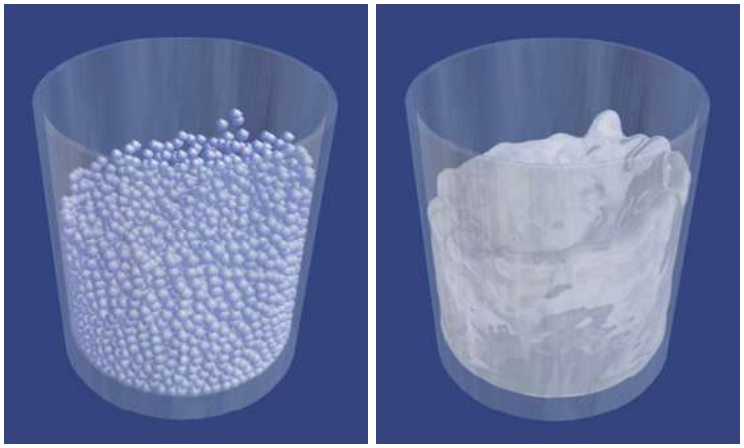


FIGURE 2.22 – *À gauche, la visualisation des particules. À droite, la surface reconstruite à l'aide de la méthode des Marching cubes. Issu de [MCG03].*

2.5.1.4 *OpenVDB*

Museth et al. [MLJ+13] proposent une structure de données hierarchique permettant entre autres d'approcher la surface d'un nuage de points en utilisant des surfaces de niveau (*levelset* en anglais, une fonction $f(x, y, z) = k$). Ils proposent également une implémentation de cette structure dans la librairie OpenVDB. Utilisée conjointement aux méthodes de filtrage de surface implémentées dans la librairie, elle permet d'approcher la surface d'un liquide avec un niveau de détail contrôlable. Parallélisée sur le processeur central (CPU), elle ne permet toutefois pas d'atteindre des performances interactives. Le pendant GPGPU de cette méthode, GVDB, a été développé au moment de l'écriture de cette thèse, et permettrait une reconstruction de surface interactive avec un niveau de granularité fin. La figure 2.23 illustre un exemple de reconstruction de surface avec OpenVDB.

FIGURE 2.23 – *Exemple de reconstruction de surface avec la librairie OpenVDB. Issu de [MLJ+13].*

2.5.2 Représentation optique : simulation d'éclairage

Dans cette section, nous posons les bases théoriques nécessaires à la compréhension des mécanismes utilisés dans les moteurs de simulation d'éclairage physico-réalistes. Dans un premier temps, nous détaillons l'équation du rendu présentée par Kajiya [Kaj86], que les moteurs d'éclairage modernes tentent d'approcher. Nous présentons ensuite les principales techniques de simulation d'éclairage, puis nous présentons les principes fondamentaux du transport de la lumière dans des milieux participants. Enfin, nous traitons les problématiques du transport de lumière dans les milieux semi-transparents avec une méthode probabiliste dite de Monte-Carlo.

2.5.2.1 Équation de rendu

L'équation de rendu (ou de luminance) fut introduite par Kajiya en 1986 [Kaj86]. Elle permet de calculer la luminance sortante en tout point d'une scène :

$$L_s(\vec{p}, \vec{\omega}_o) = L_e(\vec{p}, \vec{\omega}_o) + \int_\Omega f(\vec{p}, \vec{\omega}_o, \vec{\omega}_i) L(\vec{p}, \vec{\omega}_i) |\cos\theta_i| d\vec{\omega}_i. \tag{2.47}$$

Les différents termes de cette équation correspondent à :

- L_s, la luminance en sortie d'une surface ;
- L_e, la luminance émise par la surface (source lumineuse) ;

- $\vec{p}$, un point d'une surface de la scène ;
- $\vec{\omega_i}$, la direction du rayon lumineux incident ;
- $\vec{\omega_o}$, la direction de réflexion du rayon lumineux ;
- f_r, la BRDF (pour *Bidirectionnal Reflection Distribution Function*) ; la BRDF est une fonction permettant de mettre en correspondance le rayonnement incident avec la luminance réfléchie, c'est à dire la capacité et la manière d'une surface à réfléchir la lumière ; afin de connaître la luminance émise par une surface, une intégration sur toutes les directions incidentes possibles doit être effectuée.

C'est en général l'équation 2.47 que les moteurs de rendu basés physique approchent en utilisant les méthodes basées sur un lancer de rayon. Nous décrivons le problème de l'éclairage global dans la section suivante.

2.5.2.2 Simulation d'éclairage

Le problème du calcul d'éclairement global a pour but d'estimer une solution de l'équation de rendu introduite par Kajiya [Kaj86], et regroupe un ensemble d'algorithmes qui permettent de calculer, sur la base de cette équation, la quantité de lumière en entrée et en sortie d'une surface donnée en tenant compte de l'éclairage direct et de l'éclairage indirect, i.e. les rayons lumineux ayant effectué plusieurs rebonds avant d'intersecter la surface étudiée. Aujourd'hui, les algorithmes principaux permettant de traiter le problème de l'illumination globale sont :

- le tracé de chemin, avec de nombreuses variantes (caches d'éclairement, tracé de chemin bidirectionnel, volumétrique) ;
- le tracé de photons.

Le tracé de chemin, également introduit par Kajiya [Kaj86], consiste à choisir aléatoirement le chemin parcouru par un rayon depuis l'observateur, pour chaque pixel. Le rayon est initialement lancé de manière déterministe, puis aléatoire lorsque celui-ci intersecte une surface. L'inconvénient principal de cette méthode est le bruit obtenu lors du rendu, lequel peut toutefois être réduit en augmentant le nombre maximal de rayons. Un nombre de rayons important implique en revanche des temps de calcul importants. Pour réduire le bruit occasionné par la variance obtenue dans le tirage aléatoire des chemins, Lafortune et Willems, ainsi que Veach et al. [LW93, VG95] proposent une méthode de tracé de chemin bidirectionnel. Cette méthode consiste à lancer des chemins depuis la position de l'observateur et depuis les sources lumineuses. Ils définissent ensuite des rayons d'ombre qui contribuent à l'apparence du pixel en sortie. Un rayon d'ombre est une interconnexion entre un point de contact du chemin partant de la caméra, et un point de contact d'un chemin partant d'une source. Lafortune et Willems [LW96] étendent aussi la technique du tracé de chemin pour les volumes semi-transparents. Il s'agit du tracé de chemin volumique, qui permet notamment d'apporter une réponse aux problèmes d'illumination globale dans un contexte d'éclairage dans des milieux participants. Nous reviendrons à cette problématique à partir de la section 2.5.2.4.

Le tracé de photon est une méthode de simulation d'éclairage introduite par Jensen [Jen96]. Cette méthode se compose de deux étapes. La première, le tracé de photons, consiste à émettre et faire circuler un grand nombre de photons depuis des sources lumineuses dans la scène afin de constituer une carte de photons. Les photons se voient attribuer une chance de survie à chaque intersection. À chaque intersection, un photon peut être absorbé, réfracté ou réfléchi. Cette possibilité est calculée avec un échantillonnage de Monte-Carlo, que nous décrivons en section 2.5.2.3. Une fois le tracé de photon effectué, l'étape de rendu est traitée. Le terme d'éclairage indirect de l'équation du rendu est calculé en fonction de la concentration de photons autour du point étudié, déterminée par une estimation de densité. Le point fort du tracé de photons réside dans sa capacité à représenter des caustiques (des motifs visibles sous

l'eau, dus à la concentration de rayons lumineux), qui est notamment une des limites des méthodes de tracé de chemin.

La figure 2.24 illustre des rendus obtenus avec de telles méthodes.

FIGURE 2.24 – *Exemples de rendus obtenus avec les méthodes du tracé de chemin (à gauche [Kaj86]) et du tracé de photons (à droite [Jen96]).*

2.5.2.3 Échantillonnage de Monte-Carlo

L'échantillonnage de Monte-Carlo est une méthode probabiliste initialement mise au point par Metropolis dans les années 1940. Elle est aujourd'hui massivement utilisée dans la production des images de synthèse réalistes. Cette méthode est ainsi particulièrement utilisée pour simuler le transport de la lumière dans des milieux participants. Elle fait intervenir plusieurs variables aléatoires (e.g. la distance parcourue par un photon dans le milieu, son absorption, sa diffusion). Dans cette section, nous notons une variable aléatoire X.

Une fonction de densité de probabilité (PDF pour *Probability Density Function*) permet d'estimer la probilité qu'une variable ait la valeur x en paramètre. Une PDF a pour contrainte d'être normalisée, c'est à dire que son intégrale doit valoir 1. Cette contrainte signifie qu'une variable a une probabilité pour toute valeur comprise dans le support de la fonction.

La méthode de Monte-Carlo fait également intervenir la notion de fonction de répartition (CDF, pour *Cumulative Distribution Function*), qui est une fonction monotone croissante de 0 à 1. Une telle fonction, que nous notons $P_X(x)$ donne la probabilité que la variable X ait une valeur inférieure à x. Ces deux fonctions sont mises en relation grâce à l'intégrale suivante :

$$P_X(x) = \int_{x1}^{x} p_X(x')dx'. \tag{2.48}$$

Ces fonctions de distributions sont utilisées afin de déterminer aléatoirement le trajet de la lumière dans les moteurs de simulation d'éclairage, comme solution à l'équation de rendu. Nous décrivons leur utilisation dans un contexte de solution à l'équation du transfert radiatif dans les sections ci-après, dans lesquelles nous parlons de fonction de phase afin de décrire la trajectoire probabiliste d'un rayon de lumière ou d'un photon.

2.5.2.4 Milieux participants

Les milieux participants sont des volumes semi-transparents. Dans la réalité, lorsqu'un photon entre en contact avec un tel milieu, plusieurs cas de figure peuvent survenir :

- la lumière est absorbée par le milieu ; ceci peut donner lieu à une production d'énergie (e.g. chaleur). Le *coefficient d'absorption* est la propriété de tout milieu participant indiquant sa capacité à absorber le rayonnement électromagnétique ;
- la lumière peut être diffusée par le milieu ; à chaque fois qu'il est diffusé, le photon peut perdre une partie de l'énergie qu'il transporte, jusqu'à être finalement absorbé ; le coefficient de diffusion décrit la capacité du milieu participant à diffuser ce rayonnement ; ce photon peut être renvoyé vers sa direction de provenance : on parle alors de rétrodiffusion (*backscattering* en anglais). Il peut également être diffusé "vers l'avant" par rapport à sa direction de provenance : on parlera plutôt de prodiffusion (*forward scattering* en anglais).

Ces propriétés sont dépendantes du milieu participant et elle font partie des IOP (*Inherent Optical Properties*). Les IOP sont l'ensemble des propriétés optiques inhérentes au milieu, contribuant à l'apparence finale de celui-ci.

2.5.2.5 Équation du transfert radiatif

L'équation du transfert radiatif (ETR, équation 2.49) est un modèle mathématique qui décrit les interactions d'ordre électromagnétique entre la lumière et les matériaux. Elle permet en outre de modéliser la variation de luminance pour tout point et toute direction donnés :

$$\frac{dL(\vec{x},\vec{\omega})}{dx} = \alpha(\vec{x})L_e(\vec{x},\vec{\omega}) + \sigma(\vec{x})L_i(\vec{x},\vec{\omega}) - (\sigma(\vec{x}) + \alpha(\vec{x}))L(\vec{x},\vec{\omega}). \tag{2.49}$$

Les termes de cette équation sont :

- $\vec{x}$, le point étudié dans le milieu ;
- $\vec{\omega}$, la direction étudiée au point $\vec{x}$;
- $L_e(\vec{x},\vec{\omega})$, la luminance émise au point $\vec{x}$, dans la direction $\vec{\omega}$;
- $L_i(\vec{x},\vec{\omega})$, la luminance incidente au point $\vec{x}$, dans la direction $\vec{\omega}$;
- $\alpha(\vec{x})$, le coefficient d'absorption du milieu au point $\vec{x}$ (une *IOP*) ;
- $\sigma(\vec{x})$, le coefficient de diffusion du milieu au point $\vec{x}$ (une *IOP*).

En informatique graphique, les moteurs de rendu (e.g. Mitsuba [Wen10]) approchent la version intégrée de l'ETR, qui se présente sous la forme d'une double intégrale (équation 2.50). L'objectif est d'intégrer cette équation sur l'ensemble des directions pour collecter les flux incidents au point $\vec{x}$, et le long d'une direction $\vec{\omega}$.

$$\begin{aligned} L(x,\vec{\omega}) &= \int_{x_e}^{x_s} \tau(x_e,x')\alpha(x')L_e(x',\vec{\omega})dx' \\ &+ \int_{x_e}^{x_s} \tau(x_e,x')\sigma(x') \int_{\Omega} \rho(x',\vec{\omega'},\vec{\omega})L(x',\vec{\omega'})d\vec{\omega'}dx' \\ &+ \tau(x_e,x_s)L(x_0,\vec{\omega}) \end{aligned} \tag{2.50}$$

Les paramètres additionnels de cette seconde équation sont :
- x_e, le point d'entrée du rayon dans le milieu ;
- x_s, le point de sortie du rayon dans le milieu ;
- x_0, le point le plus proche de la surface du milieu ;
- x', le point à l'intérieur du milieu.

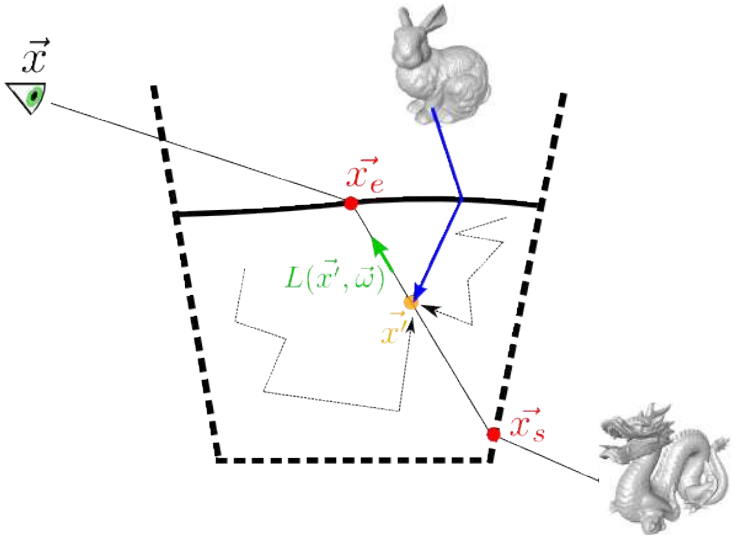


FIGURE 2.25 – *Illustration du milieu participant dans un verre d'eau.*

2.5.2.6 Fonction de phase

Les milieux participants ont, en fonction de leur constitution, une propriété de diffusion du rayonnement lumineux incident en un point qui leur est propre, essentiellement fonction de l'angle du rayon incident et de la longueur d'onde du rayon ($P(\theta, \lambda)$). Il s'agit de la fonction de phase (*scattering phase function*).

Une fonction de phase isotrope diffuse la même quantité d'énergie dans toutes les directions. Dans la réalité, les milieux participants peuvent diffuser le rayonnement incident de deux façons :

- le rayon est globalement diffusé vers l'avant, par rapport à la source lumineuse (prodiffusion) ;
- le rayon est globalement renvoyé vers la source (rétrodiffusion).

Ces deux cas de figure constituent les formes de diffusion anisotrope, par opposition antinomique à la diffusion isotrope (cf. figure 2.26).

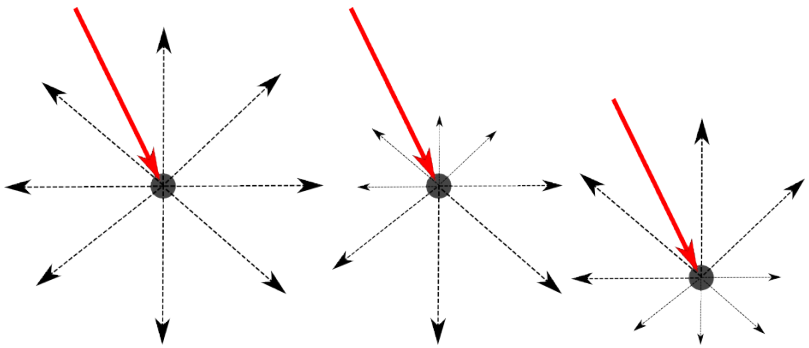


FIGURE 2.26 – *À gauche, le rayon incident en rouge est diffusé dans toutes les directions à égale intensité, de manière isotrope. Au milieu, la majeure partie de l'énergie incidente est diffusée vers l'avant. À droite, la majeure partie de cette énergie est diffusée vers l'arrière.*

2.6 Conclusion

Dans cet état de l'art sur l'animation et la visualisation de liquide en informatique graphique, nous avons décrit un ensemble de modèles issus d'études océanographiques pour formaliser le comportement de la surface de l'océan dans des eaux profondes. Ces modèles de houle et ces modèles spectraux sont largement utilisés en informatique graphique. Nous avons aussi mis en avant leurs limites, dont la principale est qu'ils ne permettent pas de représenter certaines déformations du volume d'eau comme des vagues déferlantes, de par leur nature 2D. Nous y avons également décrit un ensemble de méthodes qui approchent numériquement des solutions aux équations de Navier-Stokes dans le cas de fluides incompressibles, permettant de représenter un volume d'eau et non pas une surface en mouvement. Nous avons particulièrement mis en avant les modèles utilisant le paradigme lagrangien (par opposition au paradigme eulérien), et notamment la méthode *Smoothed Particle Hydrodynamics*. Ces méthodes sont particulièrement appréciées de la communauté informatique graphique de par leur nature facilement parallélisable et des simplifications physiques qu'elles apportent. Ces simplifications physiques sont également leurs limites, largement palliées par des contributions en particulier sur les problèmes de compressibilité. Cet ensemble de points forts nous a en particulier conduits à développer un solveur SPH, parallélisé sur carte graphique, permettant de simuler un nombre important de particules en temps interactif, en utilisant les méthodes de l'état de l'art. Nous utilisons ces développements dans le cadre de notre méthode de simulation de vagues déferlantes, décrite en chapitre 3, qui reste un problème ouvert et plus particulièrement dans le contexte de l'art numérique tel que la conception des films d'animation.

Dans la seconde partie de cet état de l'art, nous avons d'une part fait un tour d'horizon des méthodes permettant d'extraire ou de visualiser (ou les deux à la fois) la surface des particules simulées. Certaines d'entre elles utilisent directement l'architecture de la carte graphique pour approcher une surface, révélant parfois des défauts (aspect rugueux ou volumes ouverts), et les autres, plus longues à calculer mais plus efficaces, parcourent un champ scalaire du liquide pour calculer une isosurface, approchée par un maillage triangulé. La suite de nos travaux utilise à ce titre la librairie OpenVDB [MLJ+13] pour approcher cette surface en utilisant la parallélisation sur CPU, et offre la possibilité de filtrer le résultat pour en améliorer la qualité. Enfin, nous avons également posé la problématique de la visualisation physico-réaliste de la surface du liquide faisant particulièrement intervenir les problèmes d'interaction entre le rayonnement électromagnétique et la matière, et la théorie des milieux participants. Ces phénomènes physiques sont notamment approchés par certaines méthodes de simulation d'éclairage telles que le tracé de chemin. La visualisation de liquide considérant l'eau comme un milieu participant reste également un problème ouvert, et peu de contributions (à l'exception notable des travaux de Premoze et Ashikhmin [PA00]) permettent à ce jour de tenir compte de l'ensemble des paramètres de l'équation du transfert radiatif dans le cas de l'eau de mer. Nous développerons ainsi le chapitre 4 autour de cette problématique en proposant un cheminement permettant de tenir compte d'un maximum de cet ensemble de paramètres. Ces travaux sont intégrés dans l'outil Mitsuba [Wen10], proposant en outre des méthodes d'éclairage global dans le contexte des milieux participatifs.

Simulation de vagues déferlantes 3

Sommaire

3.1 Introduction

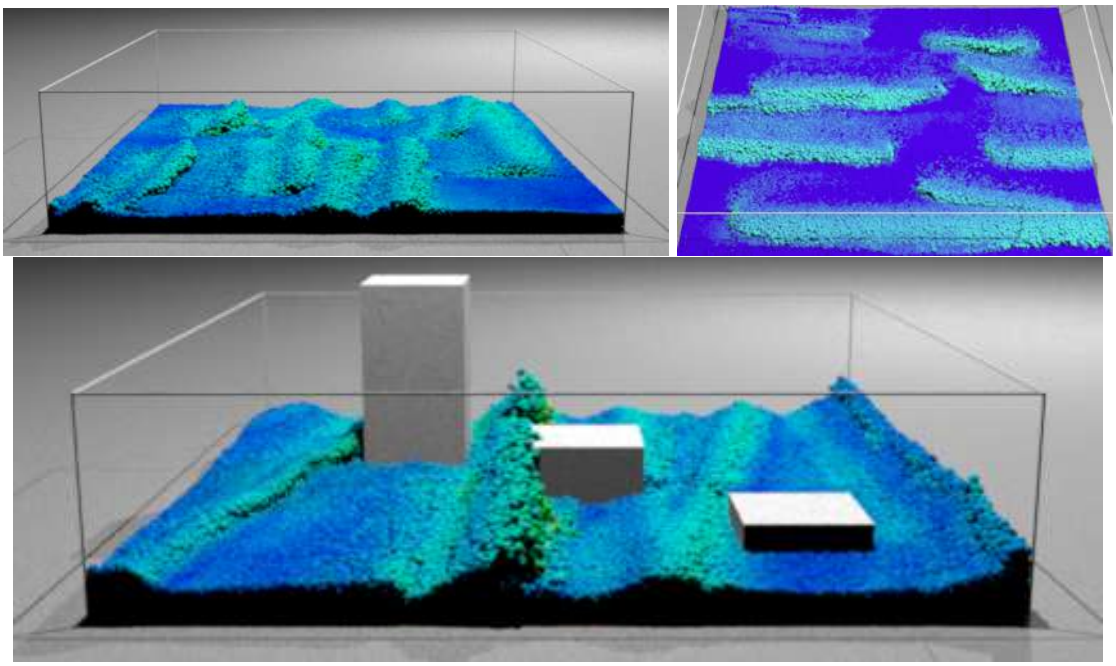


FIGURE 3.1 – *Notre modèle est capable de simuler un grand nombre de vagues, définies individuellement.*

Les travaux de ce chapitre ont été motivés par une double perspective. D'une part, l'animation et le contrôle de vagues dans le cadre de systèmes basés sur la physique est un problème ayant fait l'objet d'assez peu de contributions dans la littérature. Pour générer de telles perturbations, les auteurs ont la plupart du temps recours à des techniques telles que le déplacement d'un plan solide permettant de générer une vague manuellement. D'autre part et surtout, le contrôle de la dynamique du fluide, bien qu'étant un sujet largement abordé par la communauté informatique graphique [MVW+16a, MMS04, RTWT12, TKPR06], reste un domaine particulièrement exploratoire dans le cas des vagues océaniques.

Notre objectif est de proposer une méthode permettant de simuler la dynamique de plusieurs types de vagues (y compris déferlantes, comme nous l'illustrons sur la figure 3.1) dans les systèmes physiques basés sur les équations de Navier-Stokes. Nous souhaitons que cette méthode soit générique, c'est à dire qu'elle ne dépende pas du système physique utilisé et qu'elle puisse être intégrée dans un solveur eulérien ou lagrangien. Par ailleurs, nous souhaitons que le modèle soit contrôlable et qu'un artiste puisse influer sur le comportement dynamique (e.g. leur vitesse, leur amplitude, leur déferlement) des vagues au cours du temps.

Nous décrivons en premier lieu un modèle de force externe permettant de décrire la propagation d'une vague océanique, que nous qualifierons désormais de *soliton* puisqu'il décrit la dynamique d'une seule vague. Bien que ce type de modèle ait déjà été utilisé dans la littérature [DCG11], nous montrons comment utiliser un modèle 2D de représentation des vagues comportant peu de paramètres, dans l'optique de fournir un outil de contrôle pratique pour un concepteur d'effets spéciaux. Nous présentons également l'interface implémentée et utilisée lors de la réalisation des résultats.

Nous décrivons par la suite une série d'extensions qui permettent un contrôle bien plus large du modèle pour obtenir des phénomènes océaniques en trois dimensions, tels que des crêtes de vagues courbes, ou des propagations de vagues non linéaires. Nous présentons également une solution simple permettant une interaction aisée et pourtant visuellement plausible entre plusieurs solitons.

Notre objectif est de contrôler à la fois le mouvement océanique et l'apparence visuelle

(simulation dynamique et simulation d'éclairage). Nous souhaitons intégrer un système de particules marines permettant de représenter la dynamique de particules d'écumes, des bulles, des embruns et du sable dans les milieux océaniques, dans le but de prendre en compte un maximum de paramètres ayant une influence sur l'apparence des vagues. Notre système étend une méthode de la littérature [IAAT12] dont nous proposons d'une part une version parallélisée sur carte graphique avec CUDA, et d'autre part une version étendue pour la simulation du mélange de sable dans l'eau, tel que c'est le cas dans la réalité dans les zones peu profondes, ou de forte vorticité. Nous illustrons toutes les étapes que nous souhaitons mettre en œuvre dans une chaîne de traitement sur la figure 3.2.

Nous discutons par la suite des performances de nos outils de simulation, toujours interactives, y compris avec un nombre de particules important.

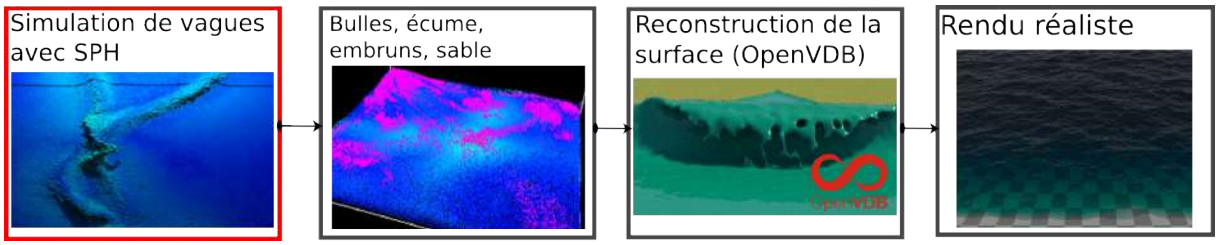


FIGURE 3.2 – *La première contribution de ce chapitre se place à la base de notre chaîne de traitements pour la représentation de vagues en informatique graphique : leur animation.*

3.2 Modèle de l'onde solitaire : le soliton

Darles et al. [DCG11] proposent de mettre en œuvre la propagation d'une vague dans un simulateur *SPH* adaptatif en utilisant une force décrivant la propagation d'une onde solitaire, nommément un soliton. Un soliton est une onde se propageant sans déformation, observable dans la réalité par exemple pendant les mascarets (ondes de marée montante provoquant une élévation du niveau de l'eau d'un fleuve). Les solitons ont la propriété d'être des solutions exactes à l'équation de Korteweg et de Vries (équation 3.1),

$$\frac{\partial \phi}{\partial t} \cdot \frac{\partial^3 \phi}{\partial x} \cdot 6\phi \cdot \frac{\partial \phi}{\partial x} = 0. \tag{3.1}$$

Il s'agit d'un modèle physique de vagues en eaux peu profondes, particulièrement adapté à la simulation de vagues déferlantes. Darles et al. [DCG11] reprennent la formulation du soliton donnée par Radovitzky et Ortiz [RO98] détaillée en équation 3.4, en ajoutant une composante z. Radovitzky et Ortiz ont utilisé cette formulation 2D dans un contexte de simulation par méthode des éléments finis, représentant un fluide compressible par un maillage triangulaire 2D.

$$\vec{F}_x = \sqrt{gd}\frac{H}{d}\text{sech}^2(A(x - \omega t + \phi)) \tag{3.2}$$

$$\vec{F}_y = \sqrt{3gd}\frac{H}{d}^{\frac{3}{2}}\frac{y}{d}\text{sech}^2(A(x - \omega t + \phi)\tanh(x - \omega t + \phi)) \tag{3.3}$$

$$\vec{F}_z = \sqrt{gd}\frac{H}{d}\text{sech}^2(A(z\sin\theta - \omega t + \phi)). \tag{3.4}$$

Les paramètres de cette équation sont :

- l'amplitude A, caractérisée par $\sqrt{\frac{3H}{4d^3}}$;
- H, la hauteur de la surface ;
- g, l'accélération gravitationnelle ;
- ω, la vitesse de propagation de la vague ;
- d, la largeur de l'onde ;
- sech, la sécante hyperbolique, définie par $\frac{1}{\cosh(x)}$.

Bien que ce modèle permette de simuler plusieurs types de vagues, il peut être compliqué à contrôler pour un artiste car il nécessite de faire varier plusieurs paramètres sur les trois composantes du modèle. Notre objectif est de montrer comment il est possible de contrôler l'ensemble des phénomènes avec un modèle de soliton plus simple en 2D, et d'exploiter le contrôle afin de générer des effets dans toutes les directions. Dans les sections suivantes, nous en proposons une forme simplifiée en 2D, pour laquelle nous substituons une partie du modèle de force par un contrôle direct de la force par le biais de courbes de contrôle. Dans les sections suivantes nous décrivons d'abord ce modèle, puis nous introduisons une méthode simple pour le contrôler de manière graphique. Nous introduisons par ailleurs plusieurs extensions pour contrôler la forme et le comportement de la vague toujours par le biais d'une interface simple. Notre méthode permet d'influer sur la forme de la crête d'une vague, sur sa direction de propagation ainsi que sur ses paramètres physiques tels que l'amplitude et la vitesse.

3.3 Modèle de soliton simplifié 2D

A l'instar de Darles et al. [DCG11], notre approche repose sur l'ajout de forces externes aux particules de fluide, grâce à plusieurs paramètres contrôlant la houle et les vagues déferlantes, tels que la hauteur, la vitesse, le moment du déferlement et la durée du déferlement. Dans les sections suivantes, nous décrivons notre méthode de contrôle des points de levée et de déferlement de la vague, son amplitude, sa largeur et sa vitesse. Nous montrons également comment le modèle peut être étendu pour modifier l'orientation de la vague, pour faire varier sa crête et pour simuler le croisement de vagues. Nous décrivons également une méthode simple pour des propagations non linéaires de vagues dans des carreaux de Bézier.

Durant la phase de houle d'une vague, un mouvement périodique à la surface peut être observé. Les particules du fluide suivent alors un mouvement elliptique. D'après la théorie d'Airy en profondeur finie [WJ10], la vitesse verticale des particules augmente lorsque la particule se rapproche de la surface de l'eau. Plus précisément, lorsque la vague avance, chaque particule est assujettie à une accélération verticale et une accélération horizontale, qui donnent le déplacement des particules dans la direction de la vague. Lorsque la particule n'est plus dans le domaine de l'onde, elle finit par retomber et les forces résiduelles de reflux provoquent une accélération dans le sens inverse.

Notre modèle permet d'approcher ces effets en utilisant des forces externes dans un système de simulation de fluide. Elles reposent sur une composante horizontale permettant le déplacement de la vague, et sur une composante verticale permettant son élévation. La vitesse de propagation de la vague est décrite par une pulsation ω fournie par l'utilisateur ; la phase de levée consiste à initialiser les forces horizontale et verticale au temps $t = 0$. Les forces appliquées sur une particule i sont définies de la manière suivante [RO98] :

$$F_{x_i} = -\sqrt{gd} \cdot \operatorname{sech}(Ad_{y_i})\lambda_x \tag{3.5}$$

$$F_{y_i} = \begin{cases} H \operatorname{sech}^2(A(x_i - \omega t - x_{shoal}))\lambda_y, & \text{si } x_i < x_{break}, \\ 0, & \text{sinon.} \end{cases} \tag{3.6}$$

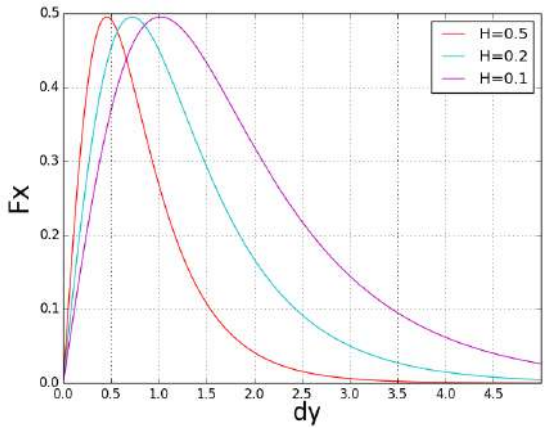


FIGURE 3.3 – *Représentation de la composante horizontale de notre modèle de force F_x, agissant sur une particule en fonction de sa profondeur d_y pour différentes valeurs de H.*

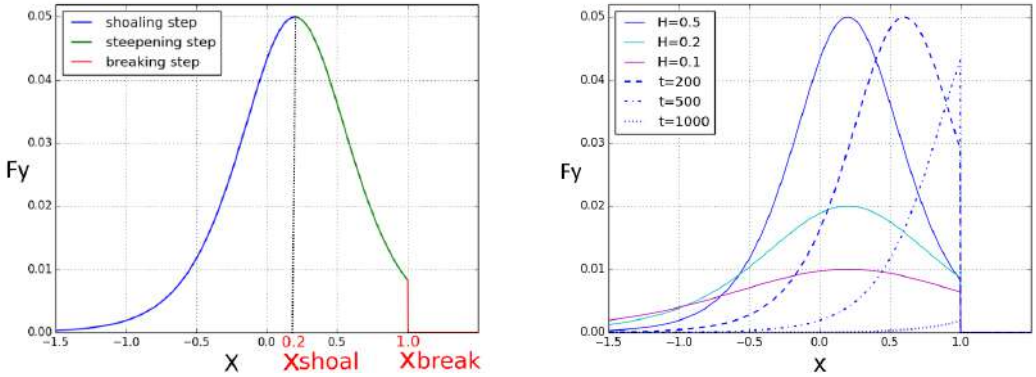


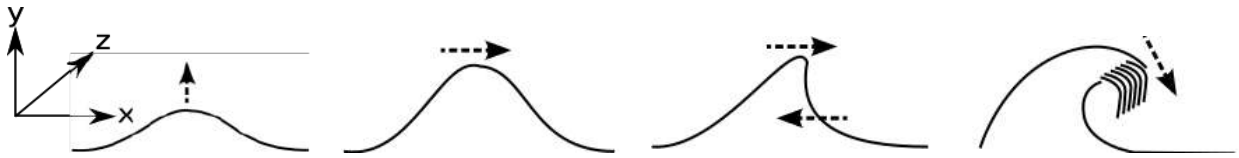
FIGURE 3.4 – *À gauche : représentation de la composante verticale F_y de la force agissant sur une particule avec $H = 0.5$, $\omega = 0.002$, $t = 0$, et $\lambda_z = 0.1$. À droite : représentation de la composante verticale de force F_z pour différentes valeurs de H au cours du temps.*

Dans ce système, H est la hauteur de la vague définie par l'utilisateur, t est l'instant courant, ω est la vitesse de propagation de la vague, $d_{y_i} = y_i - y_i^0$ est la profondeur relative de la particule i, y_i^0 est la profondeur initiale de la particule, et λ_x et λ_y sont deux paramètres utilisateurs permettant de contrôler la forme de la vague. $\sqrt{g\,d}$ est un terme d'atténuation proportionnel à la gravité [RO98], et $\omega\,t$ (équation 3.6) détermine la position de la vague.

La figure 3.3 montre la force horizontale (équation 3.5) appliquée à une particule quelconque ; l'axe des abscisses représente la profondeur de la particule d_y. Ces courbes atteignent leur maximum lorsque la particule est proche de la crête de la vague. La figure 3.4 montre la force verticale (équation 3.6). Cette fonction atteint son maximum lorsque la particule est proche du point de levée et est annulée lorsque la particule atteint le point de déferlement.

Le principal point fort de ce modèle est qu'il comporte relativement peu de paramètres et de surcroît ils sont intuitifs, ce qui répond partiellement à notre problématique.

Nous rappelons les quatre étapes correspondant à la formation d'une vague jusqu'à son déferlement en figure 3.5. Ces quatre étapes peuvent être simulées par notre modèle. Les paragraphes suivants décrivent leur fonctionnement avec notre système.



(a) formation (b) propagation (c) transformation (d) déferlement

FIGURE 3.5 – *Les quatre étapes de la vie d'une vague.*

a) Formation de la vague

Le point de levée x_{shoal} représente l'*origine* d'une vague ; les forces sont initialisées à $t = 0$, et débutent à la position x_{shoal}. La force verticale est relativement grande dans le but d'obtenir une élévation du volume d'eau (partie bleue de la courbe sur la figure 3.4, à gauche)

b) Propagation de l'onde mécanique

Horizontalement, lorsqu'une vague atteint une particule de position x, cette dernière subit une accélération de manière à obtenir un mouvement elliptique décrit par la théorie d'Airy [BGH+04], grâce à l'équation 3.6. Plus une particule est proche de la surface, plus cette force sera importante. Le mouvement de retour subi par une particule est dû au reflux occasionné par la simulation SPH lorsque le domaine de l'onde dépasse la particule. Verticalement, la force augmente progressivement de manière à provoquer la montée de la particule (mouvement elliptique vertical), fourni par notre modèle grâce à l'utilisation de fonctions hyperboliques, diminuant de manière linéaire en fonction de la profondeur de la particule (équation 3.6).

c) Formation du déferlement

La formation du déferlement correspond à une augmentation de la composante verticale des forces de la vague lorsque la position horizontale x de la particule est proche du point x_{shoal}. Lorsque la composante horizontale de la force devient plus grande au niveau des particules de la crête, la vague commence à prendre sa forme déferlante. Le déferlement peut alors avoir lieu (partie verte de la première courbe sur la figure 3.4).

d) Déferlement

Le déferlement est dû à la discontinuité de la force dans notre modèle : la vitesse horizontale de la crête est supérieure à celle des particules situées en-dessous, ce qui leur permet de retomber. Cette discontinuité est modélisée par une annulation (ou une atténuation) de la force verticale (partie rouge de la courbe sur la figure 3.4 , à gauche). Dans nos expériences, une annulation de la force tend à produire un effet de déferlement glissant. A contrario, une atténuation tend généralement à produire un effet de déferlement plongeant.

3.3.1 Discussion

Les courbes de la figure 3.3 illustrent la composante horizontale de la force pour trois valeurs de H. Pour des valeurs faibles de H, la force atteint une valeur maximale sur une hauteur importante du volume d'eau, permettant de faire avancer la vague. Pour des valeurs plus grandes de H, la hauteur de l'eau est plus petite et plutôt à la surface, produisant ainsi une différence de vitesse qui conduit au déferlement.

Les courbes de la figure 3.4 montrent l'évolution de F_y pour différentes valeurs de H et de t. Premièrement, la force verticale varie en fonction de H : avec de faibles valeurs, sa magnitude est faible de même que les variations durant les phases de formation et de déferlement. Dans cette configuration, la houle produite ne permet pas d'obtenir un déferlement et le mouvement des particules résulte principalement de la composante horizontale de la force. La simulation de vagues déferlantes nécessite des valeurs de H adaptées. De plus, la diminution de la force verticale (qui décrit les phases de levée et de début de déferlement) varie au cours du temps. Plus la valeur de t augmente, plus la part décroissante de la courbe correspondant au début du déferlement se réduit. Pour de faibles valeurs de t, cette phase permet aux particules de se lever et de produire un déferlement.

Avec cette représentation, la composante horizontale de la force n'est pas corrélée à la composante verticale. Il est donc possible de produire des vagues de faible amplitude, ou des vagues déferlantes comme cela est présenté dans les prochaines sections. La formule de la composante verticale garantit une croissance linéaire de la vague, qui peut être contrôlée au cours du temps et dans l'espace, en faisant varier les paramètres H, ω, λ_x et λ_y.

3.4 Contrôle des paramètres d'une vague

Cette section décrit comment les paramètres de notre modèle peuvent être utilisés pour contrôler localement ou globalement le comportement et la forme de la vague. Cela inclut le contrôle des paramètres de hauteur H, de pulsation ω, de point de levée x_{shoal} et de point de déferlement x_{break}. Modifier les paramètres λ_x et λ_y (considérés comme des paramètres globaux additionnels) permet d'obtenir différents types de vagues.

Hauteur

Le paramètre H affecte l'amplitude de la vague (sa hauteur) et sa forme, dérivée de l'équation 3.5. Par exemple, de la houle peut être obtenue avec $H < 0.2$, alors que des valeurs plus grandes peuvent permettre d'obtenir des vagues déferlantes, que nous illustrons sur les figures 3.6 et 3.7.

Pulsation de la vague

La pulsation de la vague contrôle sa vitesse de propagation et la durée de la formation du déferlement (voir la figure 3.8). Plus cette phase est longue, plus les particules de la crête seront

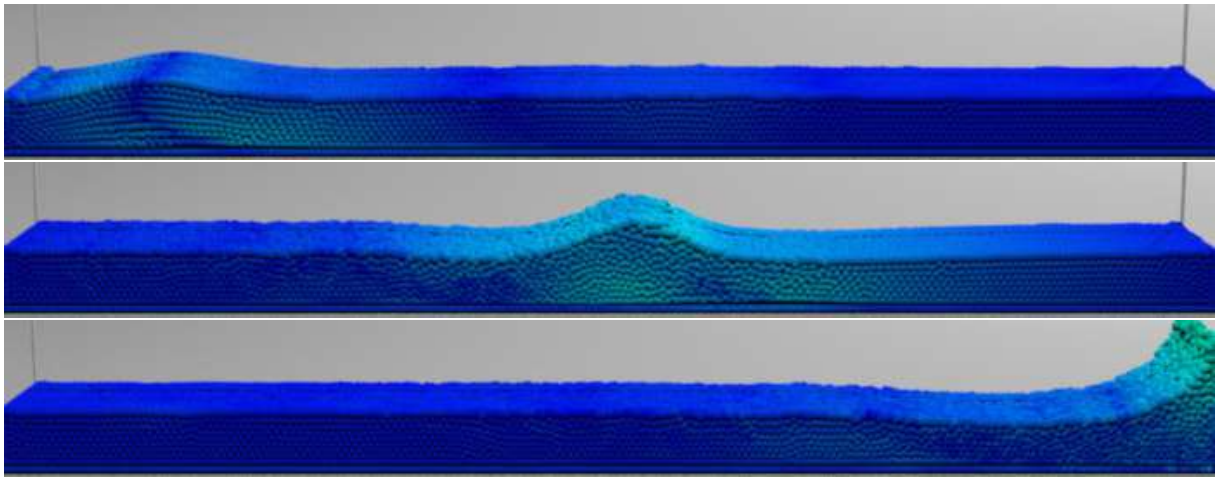


FIGURE 3.6 – *La levée et la propagation d'une houle simple avant qu'elle n'interagisse avec le bord de la région. ($H = 0.22$, $\omega = 0.002$, $d = 0.2$, $\lambda_x = 1.0$, $\lambda_z = 32.5$). La couleur d'une particule illustre sa vélocité (bleu foncé : faible vélocité).*

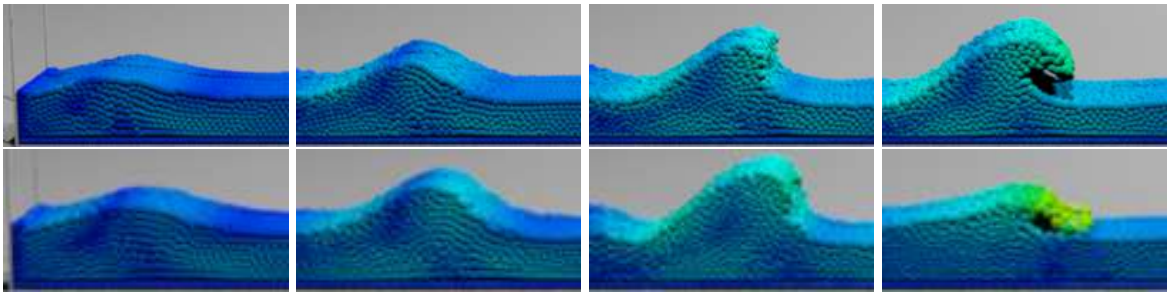


FIGURE 3.7 – *En haut : les quatre phases typiques d'une vague déferlante, qui sont la levée, la propagation, la formation du déferlement, et le déferlement ($H = 0.24$, $\omega = 0.002$, $d = 0.2$, $\lambda_x = 1.0$, $\lambda_z = 32.5$). En bas : vague déferlante obtenue en utilisant une valeur linéairement décroissante de H jusqu'à 0.*

hautes. Grâce à ce paramètre, il est possible de contrôler la forme et le type de la vague. Par exemple de petites valeurs ($\omega < 0.002$) permettent de produire des vagues plongeantes, des valeurs intermédiaires ($0.002 < \omega < 0.003$) donnent des vagues glissantes, et des valeurs plus grandes permettent d'obtenir de la houle, qui se produit principalement en eaux profondes.

Combiner des vagues avec différentes valeurs de ω permet de représenter des croisements de vagues non uniformes que l'on peut observer sur certaines côtes.

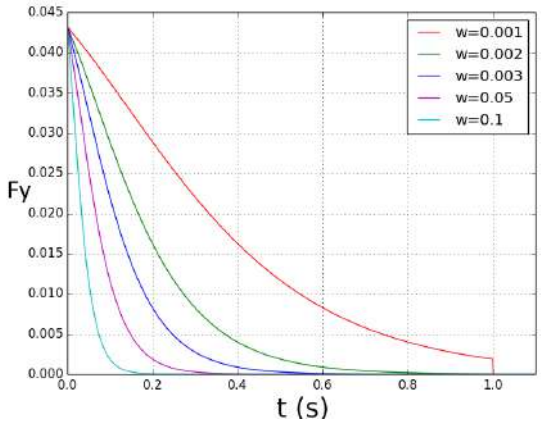


FIGURE 3.8 – *La durée de la phase de début de déferlement pour la composante verticale de la force avec des valeurs différentes de ω, évaluée sur une particule située en $x = 0$ avec $H = 0.5$, $x_{shoal} = 0.2$, and $x_{break} = 1.0$.*

Mise à l'échelle de la force des vagues

Les paramètres λ_x et λ_y permettent de contrôler globalement le comportement de la vague, en agissant indépendamment dans les directions x et y. Ainsi, la levée et le déferlement peuvent être affectés par une modification de λ_y au cours du temps, le choix d'une valeur $\lambda_y \geq \lambda_x$ garantissant le déferlement. De façon similaire la vitesse de propagation et la direction sont affectées par une modification de λ_x au cours du temps.

Points de levée et de déferlement

Les points x_{shoal} et x_{break} permettent à l'utilisateur de définir le point de démarrage de la vague, et l'endroit de la scène où celle-ci déferle. Le point de levée permet de spécifier la position en abscisse à laquelle l'onde mécanique démarre sa propagation, indépendamment du fait que du fluide soit placé au même endroit ou non. De la même manière, le point de déferlement permet de spécifier une abscisse à laquelle l'onde cesse d'exister.

3.5 Extension du modèle à la 3D

Le modèle décrit précédemment est défini pour une propagation de la houle en 1D, le long des abscisses, le paramètre y correspondant à la hauteur. Pour parvenir à un contrôle aussi complet que possible, un certain nombre d'extensions s'avèrent nécessaires. Les besoins sont multiples (e.g. crête de vague oblique, propagation non linéaire) et nous souhaitons contrôler un certain nombre de paramètres supplémentaires de la vague qui ne sont pas décrits dans le modèle mathématique :

- un paramètre d'orientation de la vague, pour contrôler sa direction de propagation ;
- une fonction permettant de contrôler la levée de la vague en fonction de sa position dans l'espace, de manière à pouvoir animer une vague dont la hauteur de la crête n'est pas nécessairement linéaire ;
- une fonction permettant de contrôler la vitesse de la vague, toujours en fonction de sa position dans le domaine ; cela permet d'obtenir une avancée de la vague qui n'est pas forcément uniforme ;
- une fonction permettant de faire interagir plusieurs vagues pour obtenir des résultats plus complexes et avec une plus forte turbulence ;
- une fonction permettant de faire varier l'orientation de la vague au fil du temps, ce qui permet d'obtenir des propagations de vagues suivant une forme précise.

Nous décrivons les méthodes mises en œuvre pour contrôler l'ensemble de ces effets dans les paragraphes qui suivent.

Orientation des vagues

Notre modèle de soliton impose une propagation sur l'axe des abscisses due à sa nature $2D$. Nous souhaitons passer outre cette limitation afin de pouvoir contrôler le sens de propagation d'un soliton. Pour y parvenir, nous définissons un vecteur directeur $\vec{v_{dir}}$ tracé par l'utilisateur dans notre interface, qui représente la direction de la vague. Nous calculons la rotation correspondant à ce vecteur sur les particules du domaine simulé et obtenons pour chaque particule une position $\hat{\vec{x}_i}$. La force du soliton est ainsi directement appliquée sur ces positions, ce qui permet d'obtenir le nouveau sens de propagation. Nous décrivons la méthode mise en œuvre dans l'algorithme 5.

Algorithme 5 : Algorithme permettant de contrôler le sens de propagation d'un soliton.

```
//on suppose que l'utilisateur a défini un vecteur directeur v_dir
//spécifiant la direction de la vague
Calculer la matrice de rotation M à partir de v_dir
Routine CUDA Orientation d'une vague
foreach particule du fluide i do
    Appliquer la matrice de rotation M sur la position x_i
    Stocker le résultat dans x̂_i
Appliquer la force du soliton sur x̂_i
```

Dans notre implémentation, chaque soliton dispose de son propre vecteur directeur. Le point fort de notre méthode est que chaque soliton dispose de ses propres paramètres, ce qui maximise le contrôle possible de l'utilisateur. Ici par exemple, chaque soliton peut avoir une direction de propagation différente permettant d'obtenir des cas de figure complexes lorsque plusieurs vagues sont simulées en même temps. Nous illustrons cet effet sur la figure 3.9.

Variation des crêtes de vagues

Nous souhaitons simuler des vagues dont l'amplitude est variable et fonction de la position dans le domaine. L'objectif est de pouvoir faire varier la hauteur de la vague en différents endroits. Cet effet est notamment observable dans la nature lorsque la profondeur du sol sous-marin n'est pas homogène, ce qui peut par exemple occasionner un déferlement en certains endroits seulement. Nous voulons ici avoir une valeur de H associée à chaque valeur de z dans l'espace afin de contrôler cet effet. Pour cela, l'utilisateur définit une courbe dans

l'interface graphique. Les abscisses décrivent la valeur de z et les ordonnées décrivent la valeur de H pour la valeur de z correspondante. Dans notre implémentation, les particules soumises au soliton ont pour valeur de H celle définie par la courbe. Ainsi, notre interface permet à un utilisateur de simuler une vague dont la hauteur de la crête est tracée de manière arbitraire. La ligne brisée permettant de définir l'amplitude peut par ailleurs être modifiée pendant la simulation. Nous décrivons la méthode mise en œuvre dans l'algorithme 6, avec pour exemple une ligne brisée définie par des points.

Algorithme 6 : Algorithme permettant de contrôler localement la hauteur de la vague.

Routine CUDA Interpolation de l'amplitude
//Nous supposons que l'utilisateur a tracé une ligne brisée $H = f(z)$
foreach *particule du fluide i* **do**
  Trouver les deux points $p1$ et $p2$ de la ligne brisée avec les valeurs de y les plus proches de $\vec{x_{y_i}}$.
  Calculer la valeur de H par interpolation linéaire entre $p1$ et $p2$.
Calculer la force sur soliton sur la particule $\vec{x_i}$ avec l'amplitude interpolée

Comme pour le contrôle de l'orientation des vagues, chacune des vagues définies dispose d'une courbe de contrôle qui lui est propre. De cette manière, chaque vague peut être contrôlée de manière totalement indépendante. La figure 3.9 illustre une vague ayant une crête oblique.

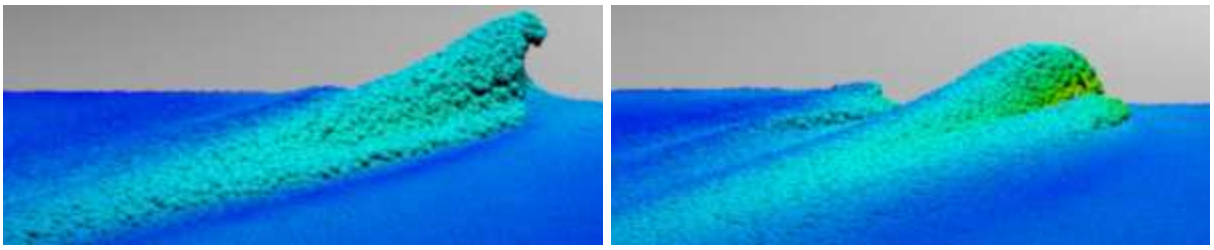


FIGURE 3.9 – *Rotation des particules pour propager la vague dans la direction voulue. La vague déferlante a un angle de propagation de* 15 *degrés, avec une crête oblique, et* $H_{max} = 0.4$, $\omega = 0.002$, $d = 0.25$, $\lambda_x = 1.0$, *et* $\lambda_y = 32.5$. *Les deux images illustrent l'avancement oblique du déferlement d'une vague.*

Interaction entre vagues

Nous souhaitons pouvoir simuler plusieurs solitons à la fois pour élaborer des scènes plus complexes. De base, le modèle mathématique ne représente pas l'interaction entre plusieurs soliton puisqu'il ne décrit la propagation que d'une seule onde indépendamment des autres. Dans ce contexte, simuler plusieurs solitons à la fois conduit la plupart du temps à des instabilités de simulation ou a des explosions numériques.

Les interactions entre plusieurs vagues nécessitent de calculer une solution pour les équations de Korteweg-de Vries [KdV95], ce que nous cherchons à éviter dans un contexte de simulation interactive. Nous proposons en revanche une alternative simple, permettant d'obtenir des résultats plausibles par rapport au phénomène observé.

L'utilisateur peut générer autant de vagues qu'il le souhaite, chacune avec ses propres paramètres de propagation (H_j, ω_j, θ_j, λ_{x_j}, λ_{y_j}). Lorsque deux vagues se croisent, calculer la somme des forces ne correspond pas au croisement de deux vagues. En revanche, nous proposons d'utiliser la magnitude maximale de la force résultante pour chaque particule i, i.e., $F_{x_i} = \max(F^1_{x_i}, ..., F^N_{x_i})$ et $F_{y_i} = \max(F^1_{y_i}, ..., F^N_{y_i})$ pour N vagues simulées. Cette méthode permet d'assurer la stabilité de la simulation car elle permet d'approcher plusieurs solitons

dans les régions du fluide où ils se croisent. Nous décrivons la méthode mise en œuvre dans l'algorithme 7

Algorithme 7 : Algorithme permettant l'interaction entre plusieurs solitons.

```
Routine CUDA Interaction des solitons
foreach particule du fluide i do
    foreach chaque soliton j do
        Estimer les forces du soliton j à la position i
    Appliquer le soliton d'amplitude de soliton maximale pour la position i
```

Cette méthode permet d'obtenir un comportement visuellement identique à celui obtenu avec plusieurs vagues générées artificiellement avec des murs en mouvement. Les deux méthodes sont comparées sur la figure 3.10.

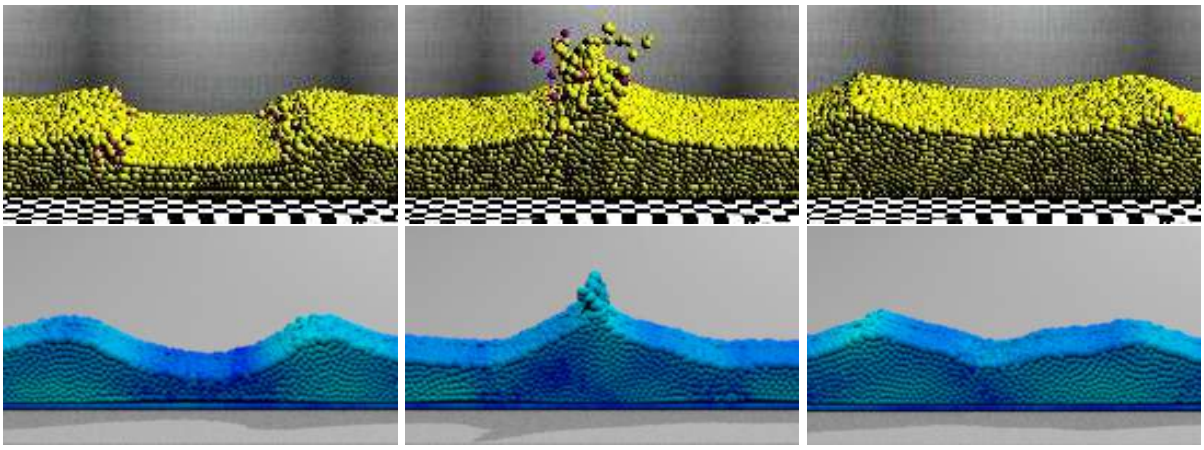


FIGURE 3.10 – *En haut : deux vagues opposées générées avec des murs en mouvement. En bas : la même configuration avec notre modèle.*

Génération et contrôle de vagues

Dans notre système, l'utilisateur crée l'instance d'une nouvelle vague avec peu de paramètres initiaux : un point de départ, un temps t et une direction de propagation. Lorsque la vague apparaît au temps t, ses paramètres peuvent être modifiés (même de manière interactive pendant la simulation) grâce à plusieurs courbes de contrôle modifiables utilisées comme des facteurs de paramètres, qui sont fonctions du temps. Par exemple, une courbe variant de 0 à 1 lors de la phase de levée de la vague permet de produire une levée progressive. Ce processus est illustré sur la figure 3.15. Chaque vague dispose de son propre jeu de paramètres et peut être rejouée autant de fois que l'utilisateur le souhaite, avec l'avantage de faibles coûts de calculs. Une fois qu'il est satisfait du résultat, les paramètres ainsi que leur courbe peuvent être exportés pour un usage ultérieur. Les simulations illustrées dans ce chapitre ont été produites de cette manière.

Fronts de vagues courbes

Dans la réalité, les fronts de vagues n'apparaissent pas toujours au niveau de la surface de l'eau comme des segments de lignes droites. Les flux maritimes, les obstacles ou la géométrie du sol sous-marin sont autant de paramètres qui peuvent contribuer à la courbure des fronts de vagues se propageant. Un artiste peut par conséquent vouloir modéliser un front de vague

en propagation selon une forme arbitraire, en le traçant par exemple sur la surface de l'eau (sur le plan $x; y$, comme illustré sur la figure 3.11). Dans ce système, nous traçons des courbes sur la vue de haut (en utilisant des courbes de Bézier) afin de contrôler la forme d'un front de vague.

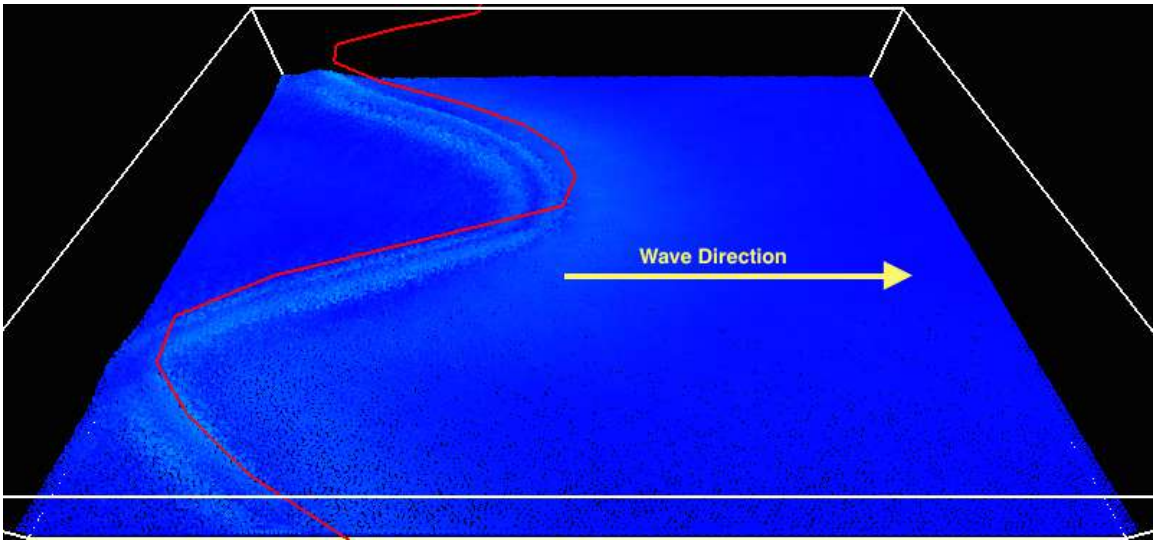


FIGURE 3.11 – *La courbe rouge, tracée par l'utilisateur, permet de modéliser le front de la vague pendant la simulation.*

Notre interface graphique propose également un moyen de tracer le profil de chaque front de vague, en utilisant un ensemble de courbes d'interpolation (e.g. des courbes de Bézier 2D ou des courbes splines). Les abscisses représentent la largeur du domaine, et les ordonnées représentent la position à laquelle la force du soliton sera appliquée. Chaque courbe est utilisée comme fonction de translation T, et appliquée au paramètre $\vec{x_i}$ de notre modèle de forces externes. Il est donc possible de représenter le front d'une vague selon le système d'équations suivant :

$$F_{x_i} = F_{(x_i + T_{z_i})} \tag{3.7}$$

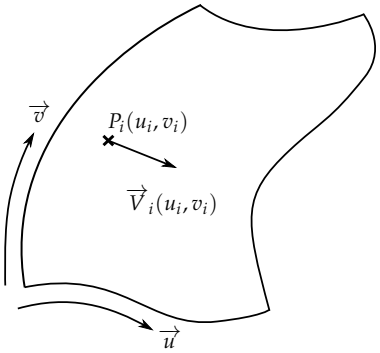
$$F_{z_i} = \begin{cases} H \text{ sech}^2(A((x_i + T_{z_i}) - \omega t - x_{shoal}))\lambda_z, & \text{si } x_i < x_{break}, \\ 0, & \text{sinon.} \end{cases} \tag{3.8}$$

La figure 3.16 illustre un front de vague modélisé avec cette méthode.

a) Chemins de vagues

La fonction de translation peut également être généralisée pour créer un chemin de vague ainsi qu'un front de vague courbe, en interpolant à la fois la forme du front de vague et sa direction. Pour y parvenir, nous représentons le domaine d'une vague à l'aide d'un carreau de Bézier 2D. Deux fronts d'une vague y sont représentés, et correspondent aux courbes en positions $u = 0$ et $u = 1$, qui sont respectivement associées au temps t_1 et t_2. Chaque paire de points pour une valeur paramétrique donnée ($0 \leq u \leq 1$), est représentée dans le carreau le long du chemin v, comme illustré sur la figure 3.12. Tant que le carreau ne comporte aucune auto-intersection, chaque point aura une seule valeur dans le temps (le long de l'axe u) et dans son orientation (le long de l'axe v) pour générer la force. La figure 3.17 illustre un exemple de simulation mettant en œuvre notre méthode, utilisant une propagation courbe de la vague.

Pour conclure, un des points forts de notre modèle est sa formalisation sur seulement deux dimensions. Il comporte ainsi peu de paramètres. Notre approche a consisté en quelque sorte à



$P_i(x_i, y_i)$ correspond à (u_i, v_i)
$F_{x_i} = F_{u_i} \times \overrightarrow{V_x}$
$F_{z_i} = F_{u_i} \times \overrightarrow{V_z}$
F_{y_i} ne change pas

FIGURE 3.12 – *Le chemin de vague est contrôlé par un carreau de Bézier, défini dans le plan $x;z$. Une particule P_i ayant pour coordonnées (x_i, y_i, z_i) dans le système de coordonnées, est associée aux coordonnées (u_i, v_i) à l'intérieur du carreau de Bézier ; V est le vecteur tangent interpolé dans le carreau, qui permet de donner la direction de la force du soliton.*

substituer des éléments d'interface graphique permettant de contrôler directement un champ de forces à la complexité du contrôle d'un modèle purement mathématique. En outre, bien que les essais et erreurs soient toujours nécessaires, il est relativement intuitif de contrôler le mouvement de la vague et d'arriver au résultat désiré dans notre système interactif, et dans lequel les paramètres peuvent être contrôlés en temps réel.

D'autre part, les effets mis en œuvre et décrits dans cette section sont indépendants les uns des autres (interaction entre vague, vague oblique, fronts courbés, chemin de vague) et peuvent être combinés dans une même animation aisément grâce à notre méthode d'interaction entre plusieurs vagues, et car chaque vague dispose de son propre jeu de paramètres qui lui est propre.

3.6 Apparition et dynamique de particules marines

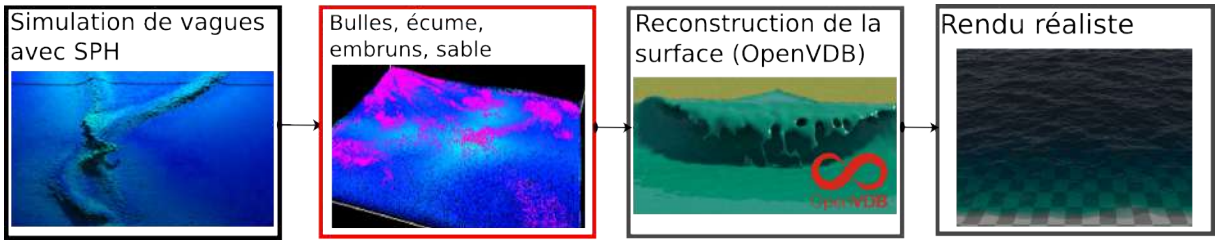


FIGURE 3.13 – *Ajouter les particules marines intervient dans notre chaîne de traitements comme post-traitement sur chacune des trames simulées, durant la phase de simulation.*

Nous souhaitons simuler des phénomènes physiques liés aux vagues déferlantes tels que la présence d'embruns, de bulles, d'écume, ou encore la présence de sable dans l'eau au niveau de la plage. Notre objectif est d'augmenter le nombre d'effets pris en compte par notre simulateur pour effectuer une simulation d'éclairage la plus exhaustive possible. Il existe très peu de contributions pour la simulation de tels matériaux dans la littérature. C'est d'autant plus le cas dans le cadre de la simulation de liquides basée sur des SPH. Toutefois, une exception remarquable a été proposée par Ihmsen et al. [IAAT12]. Afin d'améliorer l'apparence de nos simulations et en vue d'effectuer des rendus réalistes tenant compte du maximum de matériaux marins, nous avons développé cette méthode dans une version parallélisée sur GPU avec CUDA. Cette section décrit le fonctionnement de la méthode ainsi que son adaptation au paradigme GPGPU. Nous y détaillons également comment y intégrer les effets de mélange de sable avec l'eau dans les zones à haute vorticité.

3.6.1 Génération et mouvement des particules marines

En vue d'augmenter les aspects du rendu de notre chaîne de traitements, nous souhaitons enrichir les aspects dynamiques de la simulation de vagues en représentant le mouvement des bulles, de l'écume, des embruns et du sable résultant de leur passage. Pour y parvenir, deux étapes sont généralement à considérer : i) la formation de ces particules, afin de déterminer quels sont les critères physiques qui permettent l'apparition de telles particules dans nos simulations ; ii) la dynamique de ces particules, pour simuler de manière plausible le mouvement de ces particules, dépendamment de celui des vagues. Dans cette section, nous reprenons les travaux de Ihmsen et al. [IAAT12] qui répondent à ces deux questions dans un contexte de simulation SPH sur CPU, pour simuler des embruns, de l'écume et des bulles. Nous étendons ces travaux en proposant une parallélisation GPU de leur méthode. Nous étendons aussi ces travaux tenant compte de la formation et du mouvement de particules de sable dans l'eau, pouvant résulter du passage d'une vague au niveau d'une plage par exemple.

Ihmsen et al. [IAAT12] proposent une méthode pour déterminer si une particule SPH génère ou non des particules marines. La méthode consiste en trois potentiels, dépendant des propriétés géométriques et physiques de la particule et/ou de son voisinage. Ces trois potentiels sont :

- I_{ta}, le potentiel d'air piégé dans le fluide ;
- I_{wc}, le potentiel de crête de vague ;
- I_k, le potentiel d'énergie cinétique.

Pour calculer le potentiel d'air piégé ils proposent de mettre en corrélation la vélocité relative des particules avec leur capacité à générer des particules marines :

$$v_i^{diff} = \sum_j ||\vec{v}_{ij}|| \cdot (1 - \hat{\vec{v}}_{ij} \cdot \hat{\vec{x}}_{ij}) W(\vec{x}_{ij}, h) \tag{3.9}$$

avec le noyau SPH suivant

$$W(\vec{x}_{ij}, h) = \begin{cases} 1 - \frac{\vec{x}_{ij}}{h}, & \text{si } ||\vec{x}_{ij}|| \leq h \\ 0, & \text{sinon.} \end{cases} \tag{3.10}$$

Les vecteurs notés $\hat{\vec{x}}$ sont des vecteurs normalisés. Le potentiel de crête de vague est calculé en identifiant les zones concaves et convexes du fluide, que l'on connaît en estimant la courbure de la surface au niveau de la particule courante :

$$I_{wc} = \sum_j (1 - \hat{\vec{n}}_i \cdot \hat{\vec{n}}_j) W(\vec{x}_{ij}, h). \tag{3.11}$$

Enfin, le potentiel d'énergie cinétique est déterminé en mettant en relation la masse d'une particule avec sa vélocité :

$$I_k = \frac{m_i}{2} ||\vec{v}_i||^2. \tag{3.12}$$

Ces trois potentiels sont alors normalisés à l'aide d'une fonction générique :

$$\Phi(I, \tau_{min}, \tau_{max}) = \frac{\min(I, \tau_{max}) - \min(I, \tau_{min})}{\tau_{max} - \tau_{min}}. \tag{3.13}$$

Les paramètres de cette fonction sont :

- I, la valeur du potentiel étudié, parmi les trois décrits précédemment ;
- τ^{min}, le seuil plancher du potentiel courant ;
- τ^{max}, le seuil plafond du potentiel courant.

Un nombre de particules diffuses à générer est déterminé en mettant en relation les trois potentiels normalisés :

$$n_d = I_k(k_{ta} I_{ta} + k_{wc} I_{wc})\Delta t. \tag{3.14}$$

Les particules sont alors uniformément distribuées dans un cylindre dont les bases sont centrées en $\vec{x}(t)$ et $\vec{x}(t + \Delta t)$, et dont le rayon est identique à celui des particules de fluide. Elles sont placées grâce à une fonction de distribution uniforme, une vélocité initiale leur est attribuée, dépendamment de leur nature (bulles, embruns, écume), mais aussi dépendamment des particules voisines du fluide.

Conformément à la contribution d'Ihmsen et al. [IAAT12], nous calculons la vélocité voisine moyenne d'une particule du fluide en utilisant la formule suivante :

$$\tilde{\vec{v_f}} = \frac{\sum v_f \rho_d}{\sum_f \rho_d}. \tag{3.15}$$

Les paramètres de cette formule sont :

- ρ_d, la densité du voisinage du fluide, estimée à la position de la particule diffuse ;
- $\vec{v_f}$, la vélocité d'une particule voisine du fluide.

À chaque itération, la nature d'une particule diffuse est mise à jour sur la base du nombre de particules voisines du fluide. Les particules ayant le plus de voisines ($\geq \kappa_{bubble}$) sont identifiées comme étant des bulles, les particules en ayant le moins ($< \kappa_{spray}$) comme des particules d'embruns. Un intervalle intermédiaire ($\kappa_{spray} \leq n \leq \kappa_{bubble}$) permet d'identifier les particules d'écume. Toutes ces valeurs sont des constantes utilisateur et, dans nos expériences, les valeurs suivantes nous ont permis de produire des résultats plausibles :

- $\kappa_{spray} = 6$, valeur bien inférieure au nombre de particules voisines du fluide pour une particule de surface, c'est une valeur observable notamment au niveau des éclaboussures ;
- $\kappa_{bubble} = 30$, correspondant en moyenne au nombre de voisins d'une particule du fluide à l'intérieur d'un volume de liquide ;
- Les valeurs intermédiaires correspondent généralement au nombre de particules voisines du fluide observé à la surface du liquide, ce qui permet à ce titre de placer des particules d'écume plausiblement à la surface de l'eau.

3.6.2 Génération et simulation de particules de sable

Dans le but de simuler le plus d'effets possible avec notre méthodologie de rendu réaliste (décrite en chapitre 4), nous souhaitons simuler le mélange de sable dans l'eau dû aux phénomènes de flux et reflux au niveau de la plage, et dans les zones plus profondes avec une forte vorticité résultant du déferlement d'une vague.

Nous proposons d'ajouter à la méthode d'Ihmsen et al. [IAAT12], deux critères permettant de calculer le potentiel d'une particule du fluide à générer des particules de sable. Avec notre implémentation, une particule fluide génére du sable si sa vorticité dépasse un seuil κ, et que sa distance au sol est inférieure à un seuil donné. Nous utilisons le formalisme SPH pour estimer le rotationnel d'une particule fluide, introduit par Monaghan [Mono5] :

$$\nabla \times \vec{v_i} = \sum_j \vec{v_{ij}} \times \nabla W(\vec{x_i} - \vec{x_j}, h). \tag{3.16}$$

Nos expériences ont montré qu'une distance au sol inférieure à h (le rayon d'interaction de la simulation) ainsi qu'une amplitude de rotationnel de 0.01 donnent des résultats plausibles

car seules les particules en contact direct avec le sol sous-marin ont le potentiel de générer du sable.

Lorsque le sable se mélange à l'eau, il est naturel que son mouvement soit dépendant du mouvement alentour de l'eau, mais aussi dépendant de la pesanteur dans l'eau. Nous proposons de combiner ces deux paramètres de la façon suivante :

$$\vec{v_{sand}} = \vec{v_{sand}} + \Delta t \left(\frac{\vec{g_w} + \tilde{\vec{v}_f}}{\Delta t} \right). \tag{3.17}$$

Les paramètres de cette équation sont :

- $\tilde{\vec{v}_f}$, correspond à la vélocité moyenne du voisinage, telle que décrite dans l'équation 3.15 ;
- $\vec{g_w}$, correspond à l'accélération gravitationnelle dans l'eau.

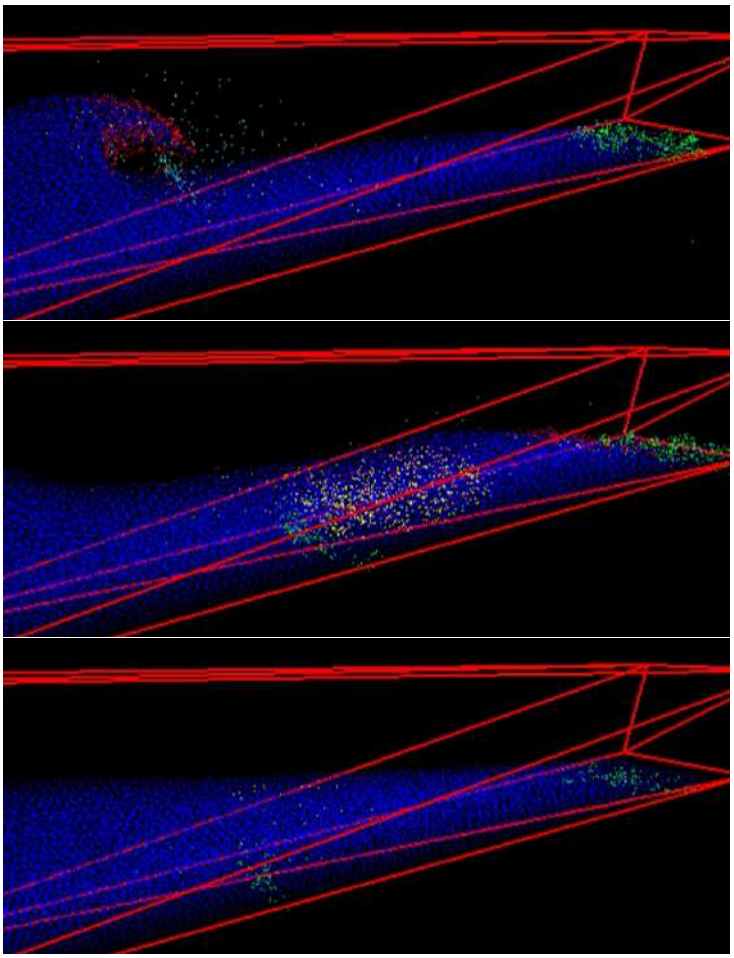


FIGURE 3.14 – *Le passage de la vague déferlante occasionne un vortex faisant remonter des particules de sable (en vert) qui suivent le mouvement de l'eau, mais aussi celui de la gravité. Au niveau de la plage, du sable se mélange constamment à l'eau et suit le mouvement du flux et du reflux.*

3.6.3 Implémentation sur GPU

Nous avons implémenté cette extension de la méthode d'Ihmsen et al. [IAAT12] sur carte graphique en utilisant des routines CUDA, à l'instar du reste de notre solveur SPH. Une première routine permet d'estimer l'ensemble des potentiels d'une particule fluide en parallèle.

Il permet également de distribuer uniformément un nombre variable de particules diffuses à l'intérieur du cylindre, conformément à la méthode décrite en section 3.6.1. Un nombre de particules diffuses maximum η est défini et commun à chaque particule, permettant une gestion statique de celles-ci dans des listes pré-allouées. L'utilisation de ces listes implique en amont une allocation unique d'espace mémoire parfois inutilisé, c'est pourquoi le choix de la valeur de η est important pour obtenir une empreinte mémoire adaptée.

Notre deuxième routine CUDA a pour effet d'advecter les particules diffuses en itérant dans les listes, qui nécessite d'abord un calcul du voisinage fluide (en réutilisant les routines dédiées à cet effet) afin de classifier chaque particule diffuse d'une part, et d'estimer la vélocité moyenne et voisine de chaque particule diffuse ensuite. L'algorithme 8 résume l'implémentation du modèle. La figure 3.14 illustre une séquence dans laquelle ces particules marines sont simulées.

Algorithme 8 : Algorithme de la méthode [IAAT12], programmé sur carte graphique.

Kernel CUDA Généreration des particules diffuses
foreach *particule du fluide i* **do**
  Calculer I_k, I_{ta}, I_{wc}
  Calculer n_d
  Distribuer n_d particules diffuses dans le cylindre compris entre $\vec{x}_i(t)$ et $\vec{x}_i(t+\Delta t)$
  Attribuer une durée de vie aux particules diffuses
Mettre à jour le voisinage fluide des particules diffuses
Kernel CUDA Advection des particules diffuses
foreach *particule diffuse i* **do**
  Classifier la particule
  Calculer $\tilde{\vec{v}}_i(t+\Delta t)$
  Calculer $\vec{x}_i(t+\Delta t)$
  Mettre à jour sa durée de vie

Pour conclure, le modèle d'Ihmsen et al. [IAAT12] ainsi que notre amélioration pour la simulation de sable comportent quelques limites physiques et des limites d'implémentation sur lesquelles plusieurs contributions pourraient être apportées. Premièrement, les performances du simulateur diminuent drastiquement (tableau 3.1), eu égard au fait que la dynamique des particules diffuses tient compte de la dynamique des particules voisines du fluide, ce qui requiert un deuxième calcul de voisinage entre les particules diffuses et les particules du fluide. Un second point d'amélioration concerne l'utilisation de listes pré-allouées dans notre implémentation : leur utilisation résulte en une allocation de mémoire très importante, dont l'utilisation n'est parfois que partielle dans le cas de simulations peu turbulentes, puisque peu de particules diffuses seront générées, les potentiels restant faibles. La principale limite physique de ce système est qu'il n'existe pas de forces internes associées aux particules marines. Une perspective intéressante pour notre implémentation pourrait être d'ajouter des forces de viscosité aux particules d'écume, ou encore de modéliser les effets de tension de surface au niveau des bulles ainsi que la dynamique des particules du fluide sous l'effet de leur flottabilité. L'implémentation de ces travaux nous sert notamment dans le cadre de notre méthodologie de rendu hors ligne de milieux océaniques, décrite au le chapitre 4.

Simulation	# Particules	Temps par trame
Sans particules diffuses	$\approx$ 100k	22ms
Avec particules diffuses	$\approx$ 100k	58ms

TABLE 3.1 – *Temps de calcul moyen d'une trame avec et sans simulation de particules diffuses.*

3.7 Mise en œuvre et résultats

Cette section illustre notre logiciel ainsi que les résultats obtenus avec notre modèle de vagues, comprenant les phases de levée, de propagation, de formation de la déferlante et le déferlement, sans utilisation de la méthode d'Ihmsen et al. [IAAT12] et de notre contribution à leur modèle. En utilisant le système développé, plusieurs scenarii peuvent conduire à des interactions entre les vagues : la mer croisée (des vagues dont les directions de propagation sont obliques), des vagues aux directions opposées, ainsi que des vagues se dépassant l'une avec l'autre. Cette section présente par ailleurs l'efficacité du système de vagues, notamment dans son implémentation au sein d'un système SPH parallélisé sur carte graphique, et dont les performances sont interactives.

Ces résultats sont présentés dans une vidéo accessible à l'adresse https://vimeo.com/149543784.

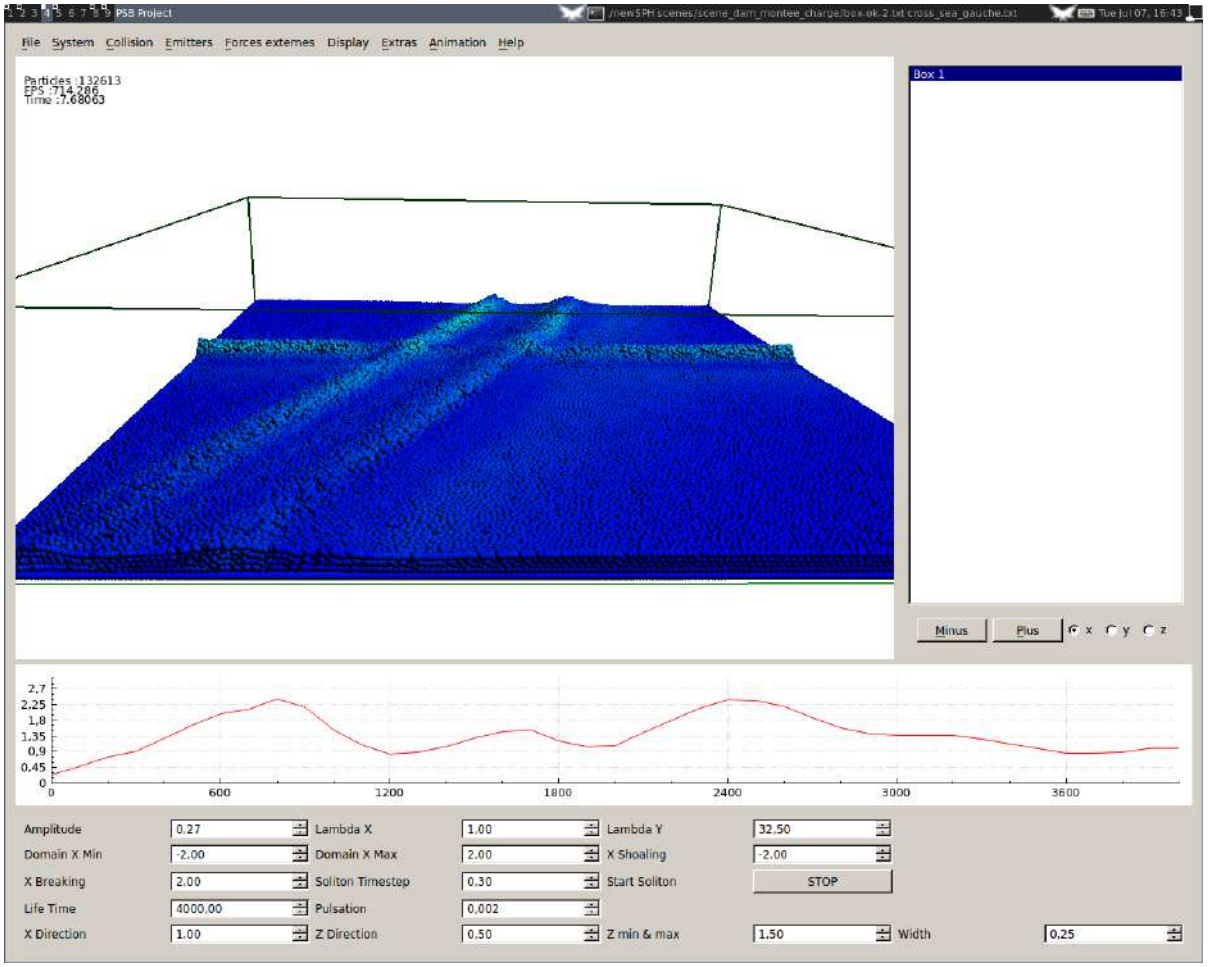


FIGURE 3.15 – *Capture d'écran de la visualisation OpenGL et de l'interface graphique. L'utilisateur peut tracer et éditer une courbe pour chaque paramètre du modèle afin de contrôler le comportement de la vague en fonction du temps. Les courbes et les valeurs peuvent être modifiées en cours de simulation. La courbe rouge correspond, dans le cas présent, à la vitesse de la vague au fil du temps.*

Le moteur de simulation est écrit en C++, et tire parti de routines parallélisantes exécutées sur carte graphique à l'aide de la technologie CUDA. L'interface de contrôle des paramètre utilise le framework Qt.

3.7.1 Valeurs des paramètres pour différentes vagues

La figure 3.15 présente notre interface graphique permettant de contrôler tous les paramètres d'une vague, par l'intermédiaire d'une valeur, et d'une courbe éditable permettant de contrôler cette valeur au fil du temps. Par exemple, l'artiste peut contrôler la hauteur de la vague pendant sa phase de levée, en traçant une courbe croissante. Tous les paramètres peuvent être contrôlés de la même manière dans l'interface, tels que la vitesse de propagation, ou l'inclinaison de la crête par exemple. Lorsque l'artiste est satisfait du résultat obtenu, les paramètres, tout comme l'animation, peuvent être exportés pour un usage ultérieur ou pour réaliser un rendu réaliste de l'animation. Les résultats illustrés dans cette section ont été obtenus avec ce système et en utilisant une méthode de lancer de rayon du moteur de rendu *Mitsuba* [Wen10].

Fronts de vagues

La figure 3.16 illustre deux exemples de courbes définissant un front de vague, que nous avons appliquées à la surface du liquide, dont on peut observer la propagation linéaire de l'onde. La fonction de translation associée à la courbe définissant le front de vague est directement appliquée au modèle de forces externes, permettant de faire correspondre le front de vague à la courbe tracée. La figure 3.22 illustre trois images d'une même animation de deux fronts de vague se croisant, montrant que notre modèle se comporte naturellement dans des situations plus complexes.

Chemins de vagues

La figure 3.17 illustre une séquence de quatre images dans laquelle un front de vague se propage le long de deux courbes définies par l'utilisateur, spécifiées par un carreau de Bézier. Ce carreau définit l'orientation de la force externe, ainsi que la translation correspondante à la surface de l'eau.

Avec notre modèle, les vagues sont contrôlées selon plusieurs techniques et paramètres. Il permet de simuler des vagues avec des formes et des vitesses variées, comme par exemple des vagues déferlantes ou des vagues de houle, pouvant interagir les unes avec les autres.

3.7.2 Gestion et interaction de vagues multiples

Le modèle permet d'animer un grand nombre de croisements de vagues, tout en maintenant une plausibilité visuelle.

Croisement de vagues symétriques

La figure 3.18 montre un exemple de deux vagues symétriques, phénomène commun en eaux profondes. Comme nous pouvons l'observer sur la séquence, l'interaction de deux vagues se propageant dans des directions opposées crée une éclaboussure typique. Après coup, les deux vagues poursuivent leur trajectoire en maintenant leurs paramètres respectifs. En utilisant cette méthode, le phénomène peut être reproduit aisément en générant deux vagues, chacune ayant pour paramètre un vecteur directeur opposé l'un avec l'autre.

Dépassement de vagues

La séquence illustrée dans la figure 3.19 correspond à une vague dépassant l'autre, celle de gauche ayant une vitesse de propagation ω supérieure à la seconde. L'interaction entre ces deux vagues résulte temporairement en une unique vague, séparée peu après avec les deux vagues originales poursuivant leur propagation selon leur vitesse respective.

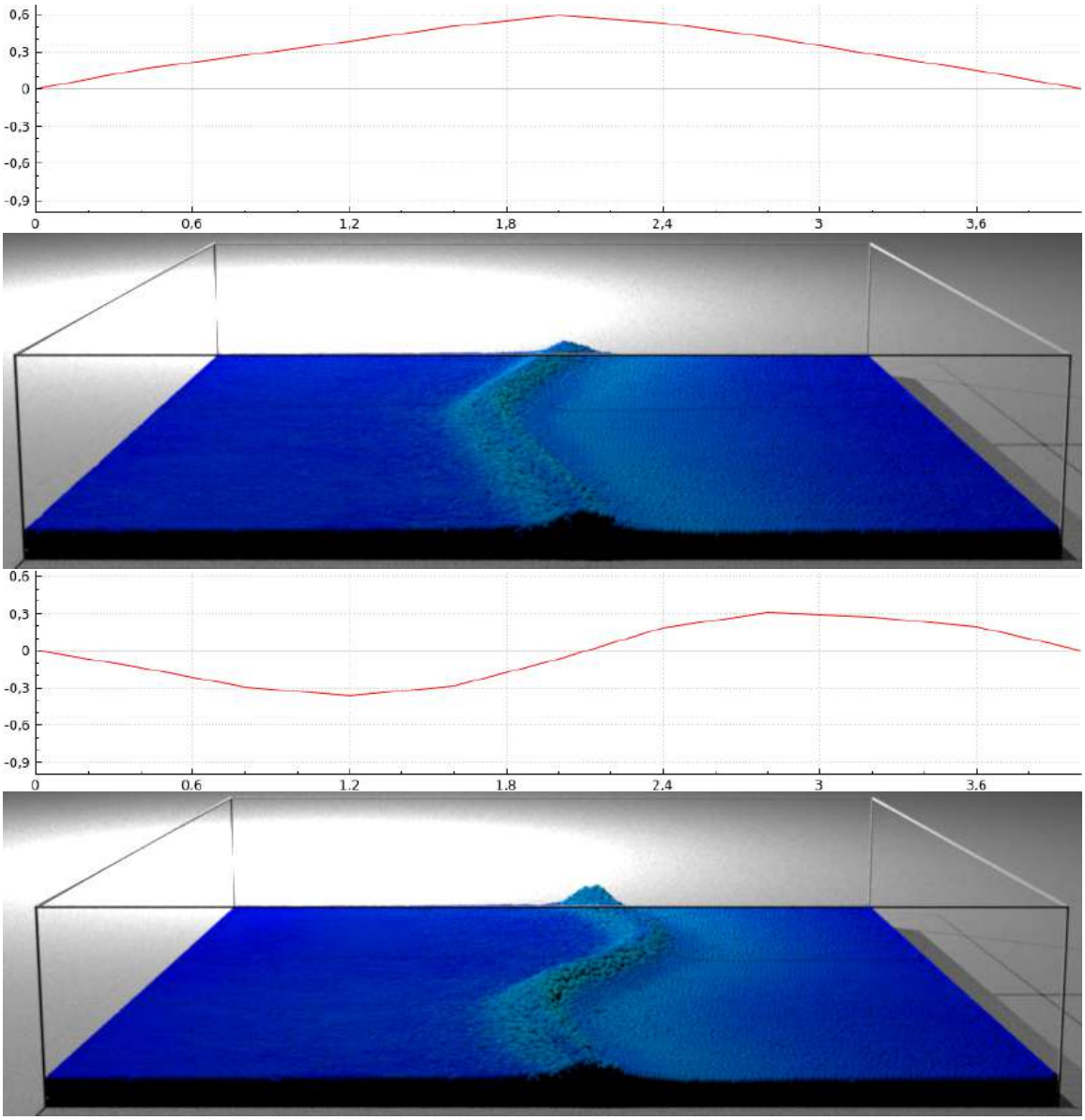


FIGURE 3.16 – *Deux fronts de vague courbes, chacun étant défini par sa courbe respective, et se propageant de manière linéaire le long de sa direction.*

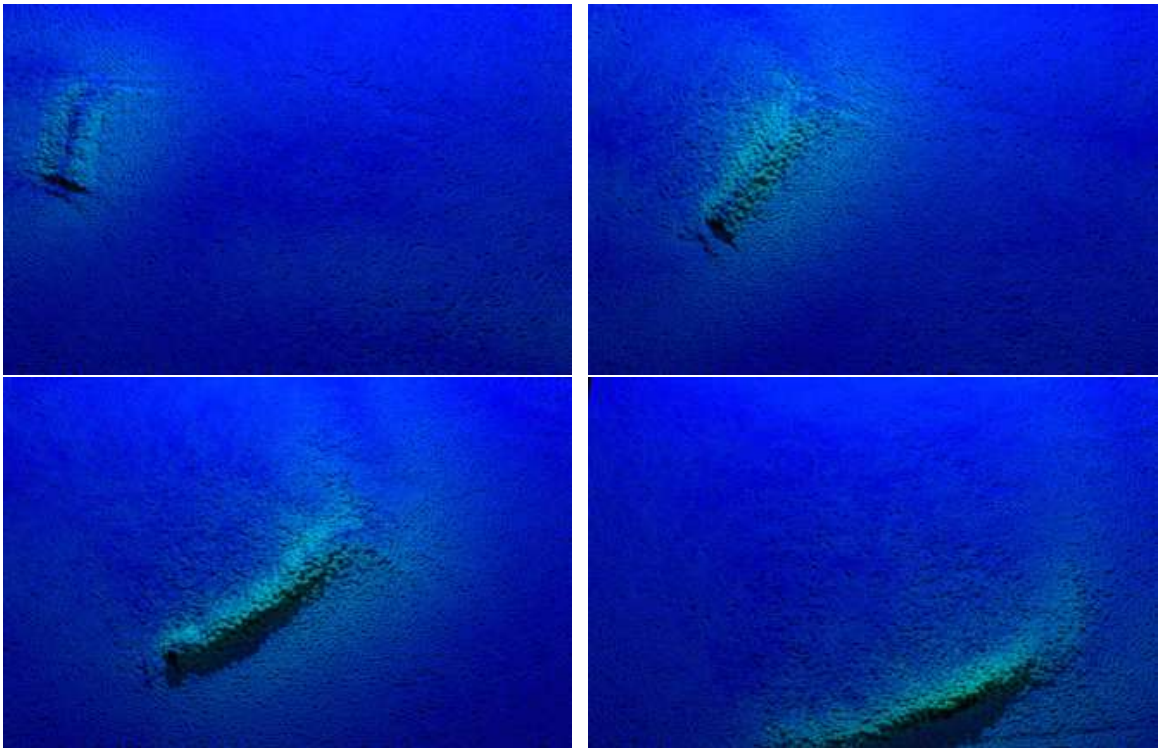


FIGURE 3.17 – *Propagation d'une vague le long d'un chemin spécifié par l'utilisateur.*

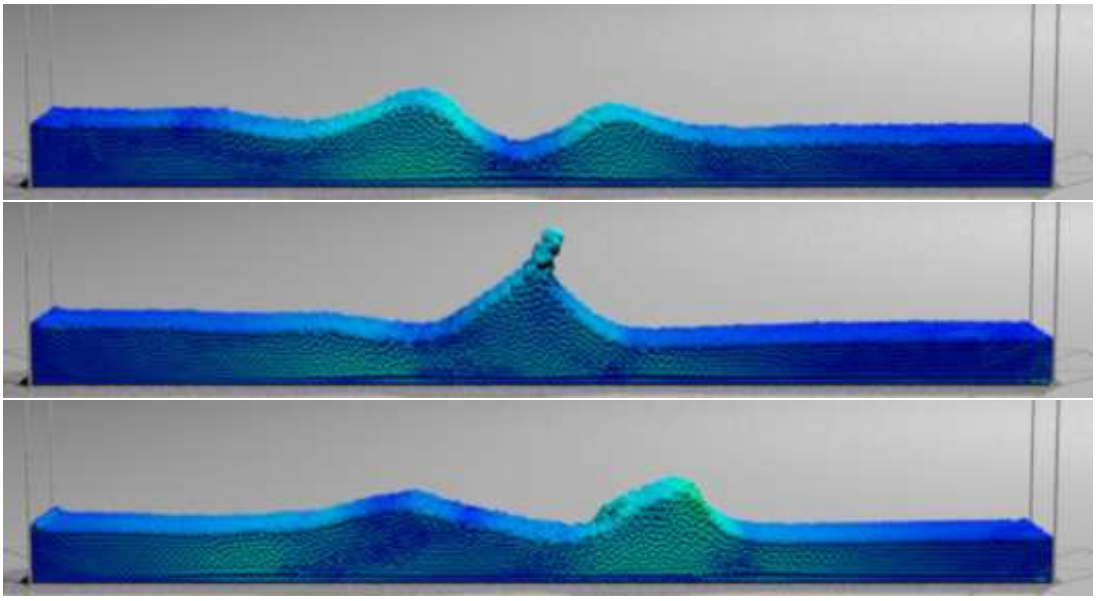


FIGURE 3.18 – *Deux vagues se croisant avec les mêmes paramètres, mais dans des directions opposées, et une vitesse* $\omega = 0.002$. *La vague provenant de la gauche a une hauteur plus importante* ($H = 0.27$) *que celle provenant de la droite* ($H = 0.23$).

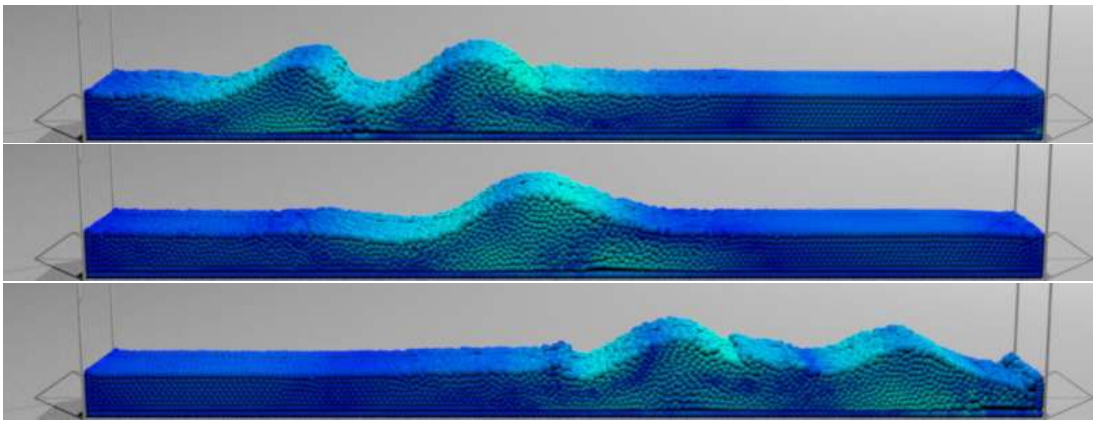


FIGURE 3.19 – *Une vague ($H = 0.23$, $\omega = 0.003$) en dépasse une plus lente et plus haute ($\omega = 0.002$, $H = 0.27$).*

Croisement oblique de vagues

La figure 3.20 montre un exemple de croisement oblique de plusieurs vagues, comme c'est le cas dans la réalité dans les situations de mer croisée. Lorsque le vent forme un train de vagues alors en propagation, il peut survenir un changement de direction du vent formant alors un nouveau train de vagues avec une orientation différente, ce qui a pour effet d'un croisement entre deux trains de vagues. Nous simulons cet effet en utilisant le paramètre d'orientation θ du modèle pour produire deux trains de vagues ayant chacun leur orientation. L'interaction oblique entre les vagues produit un résultat comparable à la réalité. Nous illustrons cet effet sur la figure 3.21.

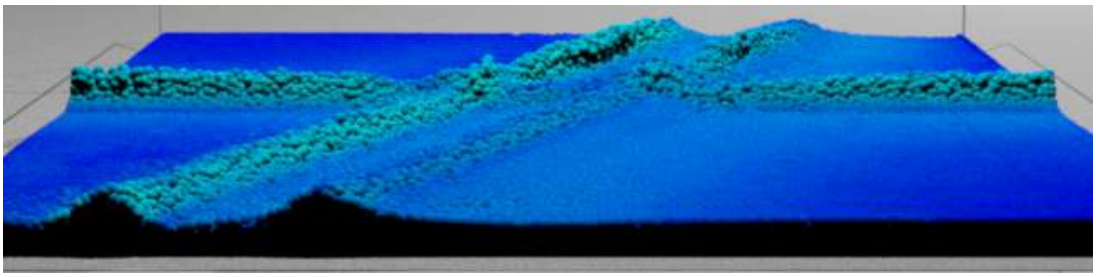


FIGURE 3.20 – *Une vague orientée croisant deux autres vagues de manière oblique. Les deux vagues parallèles ont un angle de propagation de* 45 *degrés, avec $H = 0.26$, $d = 0.25$ et $\omega = 0.002$.*

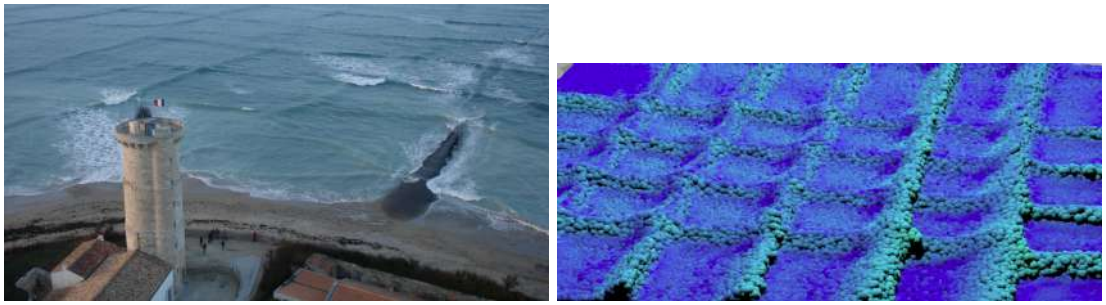


FIGURE 3.21 – *A gauche : le comportement de la mer croisée dans la réalité, courtoisie de Michel Griffon. À droite : simulation du même phénomène avec notre simulateur, avec $H = 0.23$, $\omega = 0.002$, $d = 0.25$, $\lambda_x = 1.0$ et $\lambda_z = 32.5$.*

Croisement de vagues courbes

Nous avons combiné de nombreux types de vagues, avec des vitesses, des hauteurs et des crêtes différentes. Nos simulations permettent également une interaction avec des objets solides (dont l'interaction est simulée naturellement, à l'aide du solveur SPH). La figure 3.1 illustre certaines des configurations que nous avons définies. La figure 3.23 illustre deux images d'une simulation sans obstacle. La figure 3.24 utilise une autre configuration avec plusieurs vagues, et en plus des obstacles avec lesquels les vagues interagissent.

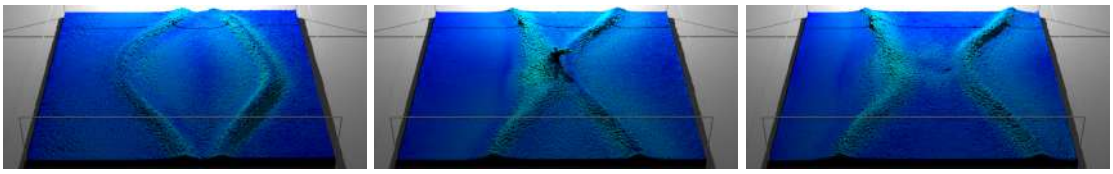


FIGURE 3.22 – *Deux vagues se croisant, ayant chacune une forme de crête différente.*

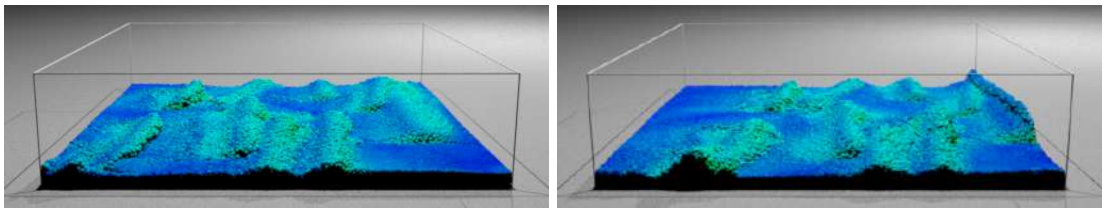


FIGURE 3.23 – *Deux images d'une animation de vagues multiples, avec différentes interactions.*

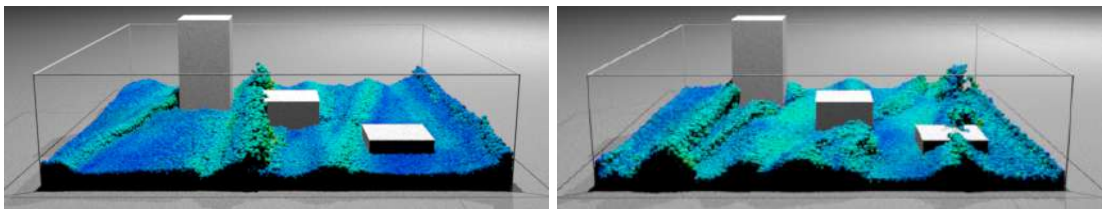


FIGURE 3.24 – *Deux images d'une animation avec des vagues multiples, interagissant avec des blocs statiques.*

3.8 Discussion

Tous nos résultats ont été générés avec un calculateur dont la configuration est la suivante :

- un ordinateur comportant un processeur Intel Xeon E2620 2.4GHz ;
- 16GB de RAM ;
- les estimations SPH sont exécutées sur une carte graphique Nvidia GTX Titan, à l'aide de routines CUDA, y compris pour les calculs de forces du modèle de vagues.

Les voisinages sont calculés à l'aide d'une grille uniforme [IABT11]. La simulation, interactive, réalise 10 itérations pour chaque image rendue ($\Delta t = 0.001$ *sec*), ce que nous montrons au tableau 3.3. Les estimations de densité et de pression sont calculées à l'aide de la méthode décrite par Ihmsen et al. [IOS+14].

Le modèle de vagues est implémenté comme une force externe additionnelle dans notre simulateur SPH. La simulation est affichée de manière interactive avec OpenGL, et en utilisant le moteur de rendu *Mitsuba* [Wen10] pour réaliser un rendu *offline*.

Les particules sont affichées avec des couleurs en fonction de leur vélocité pour illustrer l'application de la force externe : les particules en foncé sont presque immobiles et les particules

en bleu clair ont une vélocité importante. Nous pouvons constater sur les tableaux 3.2 et 3.3 que le coût de calcul par itération de notre modèle s'élève à environ 1% du temps de calcul global. Cela permet de conserver une simulation interactive et l'essentiel du temps de simulation est davantage consacré aux estimations SPH, plutôt que le modèle de vagues lui-même.

Étape	Houle Figure 3.6	Déferlement plongeant Figure 3.7	Vagues opposées Figure 3.18	Mer croisée Figure 3.20
Voisinage	47.89%	48.79%	48.97%	61.48%
Forces internes	41.41%	40.54%	40.58%	31.26%
Forces de vagues	0.94%	0.99%	1.7%	0.86%
Intégration	0.82%	0.8%	0.83%	0.49%
Gestion mémoire	8.94%	8.88%	7.92%	5.91%

TABLE 3.2 – *Proportion des temps de calcul par routine CUDA pour nos exemples, pour notre premier simulateur.*

Scène	Particules	Temps par trame
figure 3.7	30k	0.013s
figure 3.20	100k	0.022s
figure 3.9	255k	0.076s

TABLE 3.3 – *Temps de calcul moyen d'une trame sur les scènes testées, avec le premier solveur.*

Récemment, nous avons également mis en œuvre la méthode IISPH (décrite en section 2.4.2.9), de manière à profiter au maximum des différents types de mémoire de la carte graphique. Cette version optimisée tant sur les performances que sur la physique nous a permis d'améliorer grandement la qualité de la dynamique du liquide simulé, augmentant substantiellement les éclaboussures par exemple. Nous obtenons des performances du même ordre malgré un nombre de particules bien plus élevé du fait de l'utilisation de particules de bords, contrairement à la première implémentation qui utilisait des maillages triangulés. Les performances de cette seconde implémentation sont présentées de manière analogue aux tableaux 3.4 et 3.5. Nous observons que le temps passé à mettre le voisinage à jour est très inférieur. La proportion de temps dédié au calcul de forces internes est nettement supérieur et s'explique par le caractère itératif de la méthode, qui requiert au minimum deux itérations pour la technique du gradient conjugué préconditionné. Ces itérations multiplient le nombre d'accès en mémoire globale ou en mémoire des textures et nuisent fatalement aux performances globales du simulateur.

Étape	Faible épaisseur figure 3.7	Épaisseur importante figure 3.18	Profondeur importante figure 3.20
Voisinage	18.27%	18.3%	18.30%
Forces internes	58.56%	58.01%	59.61%
Forces de vagues	0.81%	0.87%	0.92%
Intégration	1.39%	1.73%	1.27%
Gestion mémoire	20.97%	21.09%	19.60%

TABLE 3.4 – *Proportion des temps de calcul par routine CUDA pour nos exemples, pour notre second simulateur. Le temps passé à estimer le voisinage est nettement inférieur et permet de concentrer les ressources sur les calculs de la dynamique.*

Notre modèle produit de nombreux types de vagues océaniques, lorsque les bons paramètres sont fournis en entrée. Toutefois, ces paramètres sont corrélés : par exemple, modifier la vitesse de propagation ω peut influer sur sa hauteur. De plus, de trop hautes amplitudes produisent des vagues peu réalistes car la composante verticale de la force résultante est trop importante, et les particules peuvent alors être éjectées car les forces ne sont plus adaptées.

Scène	Particules	Temps par trame
Faible épaisseur (figure 3.7)	30k	0.0013s
Épaisseur importante (figure 3.20)	100k	0.0037s
Profondeur importante (figure 3.9)	255k	0.0077s

TABLE 3.5 – *Temps de calcul moyen d'une trame sur les scènes testées, avec le second solveur. Les performances sont jusquà 10 fois supérieurs par rapport à la première implémentation.*

De la même manière, une vitesse de propagation ω trop importante donne des résultats peu réalistes car l'onde mécanique se propage trop rapidement.

3.9 Conclusion

Ce chapitre présente nos contributions pour l'animation et le contrôle de vagues dans le contexte d'une simulation basée physique lagrangienne et utilisant la méthode SPH. Les méthodes actuelles consistent généralement à modifier les bords pour entraîner une poussée du fluide afin de produire l'effet d'une vague déferlante. Cela nous a conduits à proposer une méthode plus générale et paramétrable. Dans la continuité des travaux de Darles et al. [DCG11], nous avons développé un modèle utilisant la théorie des solitons pour produire des vagues dans notre solveur SPH. La formulation 2D que nous utilisons est simplifiée et permet de réduire le nombre de paramètres à contrôler. Avec notre simulateur, les paramètres sont contrôlés intuitivement à l'aide d'une interface représentant chaque paramètre comme une fonction du temps. Malgré sa nature 2D et le faible nombre de paramètres du modèle, nous avons mis en place plusieurs extensions, indépendantes du modèle physique, permettant d'augmenter le nombre d'effets possibles, tels que des fronts de vagues courbes à l'aide de fonctions de translation, ou encore des propagations non linéaires à l'aide de carreaux de Bézier. La force principale de notre contribution, au-delà de la simplicité de mise en œuvre et du grand nombre de possibilités, est son coût de calcul très faible : dans les résultats obtenus, le temps dédié au calcul des forces des vagues est d'environ 1% du temps de calcul global. Ce point fort nous a permis de produire nos résultats avec des performances interactives. Les dernières optimisations mises en œuvre ont permis d'augmenter les performances des simulations jusqu'à un facteur 10. Par ailleurs, afin d'améliorer la plausibilité des effets simulés ainsi que dans l'optique d'effectuer des rendus réalistes mettant en œuvre le plus possible d'effets des milieux océaniques, nous avons implémenté une version parallélisée sur CUDA et enrichie d'une méthode de simulation d'écume [IAAT12]. Notre extension permet de conserver des performances interactives, et de simuler les effets de mélanges d'eau et de sable dans les zones de haute vorticité et de faible profondeur.

Les travaux mis en œuvre dans ce chapitre ont été présentés dans les publications suivantes :

- *A New Force Model for Controllable Breaking Waves*, VRIPHYS 2015 [BDM+15b] ;
- *Un nouveau modèle pour la génération et le contrôle de vagues déferlantes*, AFIG 2015 [BDM+15a] ;
- *Simulation and Control of Breaking Waves Using an External Force Model*, Computers & Graphics, Juin 2016 [BDM+16].

Visualisation Physico-Réaliste de Vagues Déferlantes

4

Sommaire

4.1 Introduction

Les travaux présentés dans ce chapitre ont pour objectif d'établir une connexion entre les modèles et les données de la communauté d'océanographie, permettant de décrire les propriétés optiques de l'eau, ainsi que leur interaction avec le rayonnement électromagnétique, plus spécifiquement le spectre visible dans le but d'effectuer des rendus réalistes de configurations océaniques et de leurs vagues. Darles et al. [DCGG11] proposent un état de l'art sur les techniques de rendu de l'océan en informatique graphique. À ce jour, peu de contributions, à l'exception des travaux de Premoze et Ashikhmin [PA00], caractérisent l'apparence d'un volume d'eau en le considérant comme un milieu participant, et en tenant compte des propriétés physiques du volume et de ses constituants.

Dans un premier temps, nous traitons la question de l'apparence théorique de l'eau pure, que nous faisons varier en fonction de ses température et salinité. Nous étudions ensuite son apparence dans le cas océanique, en couvrant une large partie de l'ensemble des paramètres ayant une influence sur cette apparence dans la réalité. Les réponses à cette question seront traitées de manière spectrale. En effet, discrétiser l'interaction entre la lumière et la matière sur seulement trois canaux (RGB) amène fatalement à ignorer tout un ensemble des effets de cette interaction.

Nous décrivons par la suite une formulation permettant d'unifier les propriétés possibles (la température, les différentes concentrations) d'un volume d'eau tout en respectant certaines lois optiques, validant ainsi notre méthodologie. Nous introduisons également une fonction de phase jusque là très peu utilisée en informatique graphique pour traiter de façon réaliste le parcours de la lumière dans un volume d'eau.

L'ensemble de ces propriétés sont modulables selon le desiderata de l'utilisateur. L'objectif est de permettre un contrôle aisé des paramètres pour une mise en œuvre dans un moteur de rendu spectral permettant de simuler l'interaction entre le volume d'eau et la matière. Nous décrivons également une méthodologie simple permettant d'afficher de manière visuellement convaincante les matériaux océaniques souvent visibles dans la réalité tels que l'écume, les embruns, les bulles et le sable.

Nous présentons également un ensemble de résultats, illustrant en premier lieu l'apparence de l'eau dans des configurations de laboratoire, puis dans des configurations extérieures plausibles, pour des eaux calmes et des eaux turbulentes simulées à l'aide de notre solveur SPH.

4.2 Optique de l'eau pure

Contrairement à l'intuition générale, l'eau pure n'est pas incolore. Elle correspond à un milieu homogène ayant la caractéristique d'absorber complètement et de manière très rapide les longueurs d'onde importantes. Le rouge est absorbé à une profondeur d'environ quatre mètres. Le vert est quant à lui absorbé complètement après vingt mètres et le bleu à plus de quarante mètres. La figure 4.1 montre le coefficient spectral d'absorption (*spectral absorption coefficient* en anglais) de l'eau pure dont la température est de 20 degrés Celsius. Ces mesures ont été réalisées par Pope et Fry [PF97].

La capacité de l'eau pure à absorber les longueurs d'onde varie légèrement en fonction de sa température, car une variation de température fait varier le comportement des liaisons *O-H* des molécules d'eau, qui absorbent alors la lumière différemment. La figure 4.2 montre que la capacité de l'eau pure à absorber les longueurs d'onde les plus longues (le rouge), augmente avec la température. La figure illustre également le spectre d'absorption de l'eau pure. Röttgers et al. [RDMS10] proposent un modèle linéaire permettant de connaître la

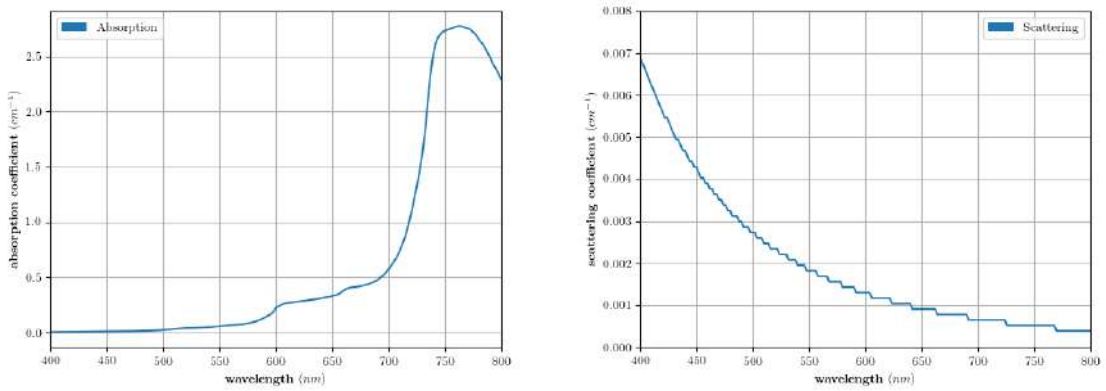


FIGURE 4.1 – *L'eau pure à 20 degrés Celsius absorbe très fortement les longueurs d'onde plus grandes (le rouge et le vert), et beaucoup plus faiblement les longueurs d'onde courtes (le bleu).*

variation du coefficient d'absorption d'une longueur d'onde donnée pour une température en degrés Celsius donnée :

$$a_w(T, \lambda) = a_w(T_0, \lambda) + (T - T_0)\Psi_T(\lambda). \tag{4.1}$$

Les différents paramètres de cette fonction sont :

- T, la température ;
- λ, la longueur d'onde ;
- T_0, la température de référence, soit 20 degrés Celsius ;
- $\Psi_T(\lambda)$, le coefficient de variance du coefficient d'absorption pour une longueur donnée, constant et obtenu par des mesures présentées par Mobley [Mob] et Röttgers et al. [RMU14].

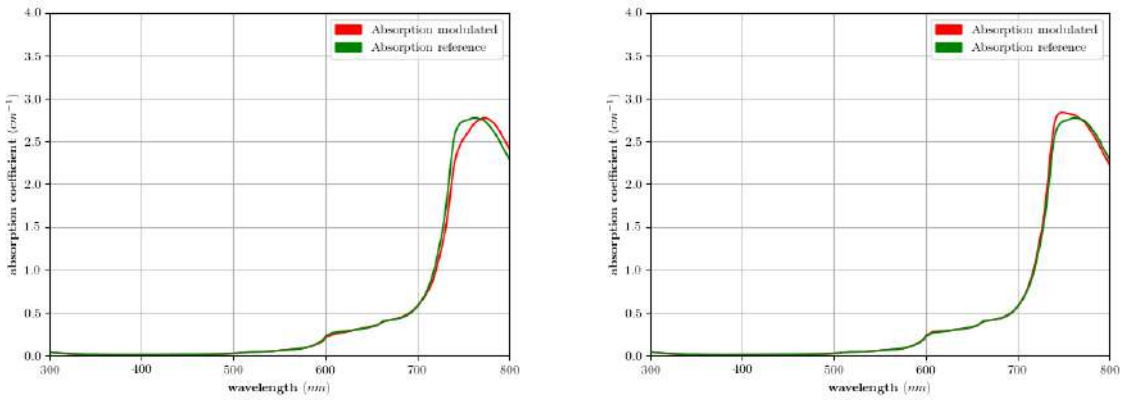


FIGURE 4.2 – *Absorption de l'eau pure à 0 degré et 30 degrés. La courbe verte est la courbe de référence (20 degrés), et la courbe rouge représente le spectre modulé.*

La variation du coefficient de diffusion spectral de l'eau pure est en revanche considéré comme négligeable et nous ne modélisons donc pas cet effet, que nous illustrons sur la figure 4.3. Bien que nous ne modélisions pas l'effet de la température sur la diffusion, la courbe modulée (i.e la courbe calculée avec des variations de paramètres) par notre implémentation diffère de la courbe de référence, qui considère une salinité de 38.4 grammes de sel par litre

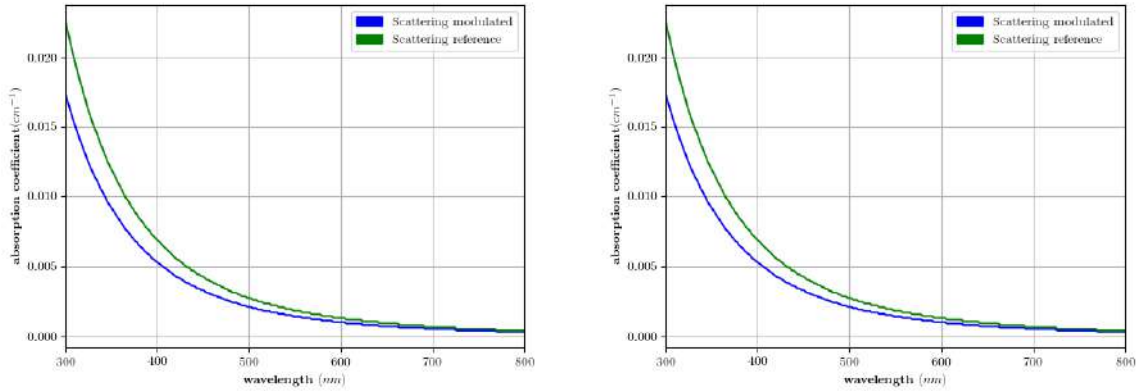


FIGURE 4.3 – *Diffusion de l'eau pure à 0 degré et 30 degrés. La courbe verte représente la donnée de reférence (20 degrés), et la courbe bleue représente le spectre de diffusion modulé.*

d'eau. Les trois courbes de la figure 4.2 n'illustrent que les variations de température pour une salinité nulle. Nous détaillons cet aspect en section 4.3.

Nous faisons varier la température d'un volume d'eau pure en figure 4.4, de la même manière qu'avec les spectres d'absorption de l'eau pure illustrés en figure 4.2. Nous pouvons constater que la variation de l'apparence de l'eau pure en fonction de la température est très difficilement perceptible. Pour visualiser cet effet nous illustrons en figure 4.5 les différences absolues (auxquelles nous avons appliqué un facteur de 50) entre les deux premières, puis entre les deux dernières images de la séquence illustrée en figure 4.4. Malgré le facteur appliqué, les différences sont très faiblement perceptibles et laissent apparaître le bruit de la simulation Monte-Carlo.

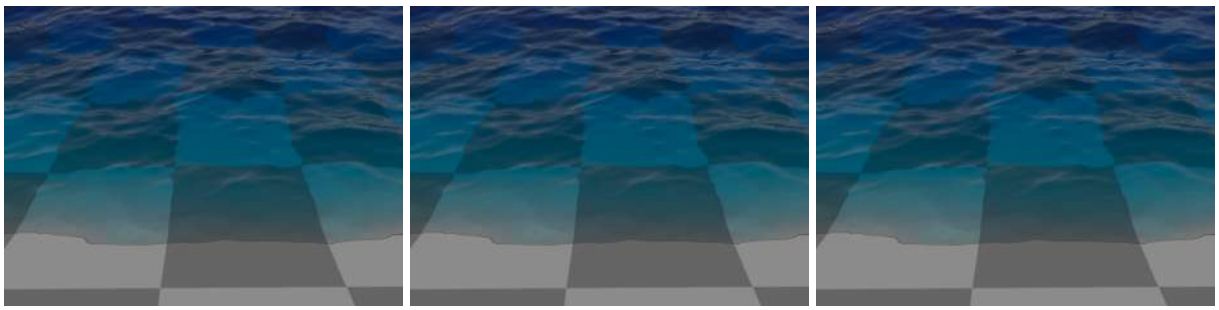


FIGURE 4.4 – *Rendus de l'eau pure avec des températures de* 10, 20 *et* 30 *degrés Celsius.*

4.3 Optique de l'eau salée

De la même manière que le modèle permettant d'approcher le coefficient d'absorption spectral de l'eau pure pour une température donnée, Röttgers et al. [RMU14] proposent un modèle similaire permettant de faire varier le coefficient d'absorption de l'eau salée à une longueur d'onde donnée, pour une salinité donnée exprimée en mg/L :

$$\alpha_w(S,\lambda) = \alpha_w(S_0,\lambda) + (S - S_0)\Psi_S(\lambda). \tag{4.2}$$

Le paramètre S représente la salinité de l'eau, et S_0 représente la salinité de référence, dont la valeur est de $38.4 mg/L$, correspondant à la salinité moyenne de l'eau de mer [Mor68]. À la manière de Ψ_T, $\Psi_S(\lambda)$ correspond au coefficient de variation du spectre d'absorption de l'eau due à la salinité. Les influences de la température et du sel étant indépendantes quant à

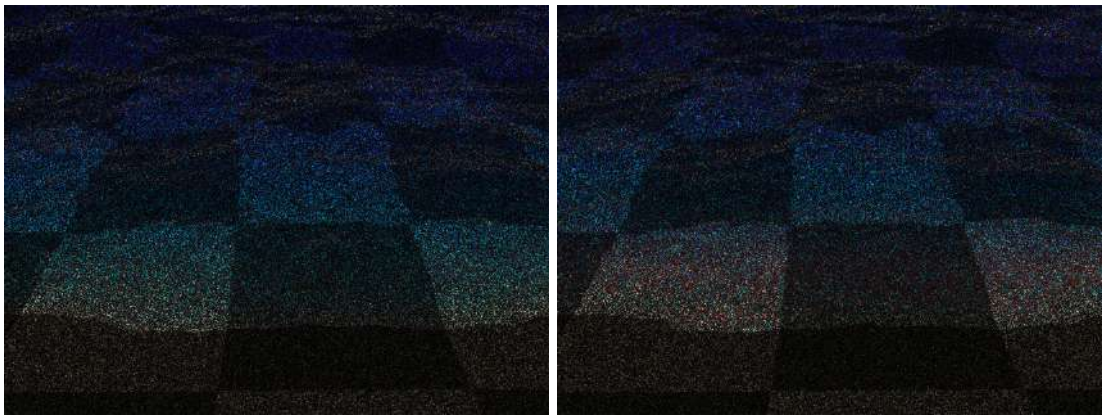


FIGURE 4.5 – *La différence absolue entre les rendus obtenus pour des températures de 10 degrés et 20 degrés, et la différence entre les rendus avec des températures de 10 et 30 degrés.*

l'évaluation du coefficient d'absorption de l'eau salée, Röttgers et al. [RMU14] proposent de regrouper les équations 4.1 et 4.2 avec l'équation suivante :

$$\alpha_w(T,S,\lambda) = \alpha_w(T_0,S_0,\lambda) + (T-T_0)\Psi_T(\lambda) + (S-S_0)\Psi_S(\lambda). \tag{4.3}$$

Le coefficient de diffusion de l'eau est lui aussi dépendant de la température et de la salinité de l'eau. Toutefois, contrairement au coefficient d'absorption, il est beaucoup plus difficile à modéliser. Zhang et al. [ZHH09] proposent le modèle décrit en équation 4.4, basé sur des équations aux dérivées partielles :

$$\beta(90,\lambda,T,S) = \beta_w(90,\lambda,T) + \beta_s(90,\lambda,S) \tag{4.4}$$

$$\beta_w(90,\lambda,T) = \frac{\pi^2}{2\lambda^4}(\rho\frac{\partial n^2}{\partial\rho})^2_T kT_k\beta_T f(\delta) \tag{4.5}$$

$$\beta_s(90,\lambda,S) = \frac{\pi^2}{2\lambda^4 N_A}(\frac{\partial n^2}{\partial S})^2\frac{M_0}{\rho}\frac{S}{-\frac{\partial(\ln(a_0))}{\partial S}}f(\delta). \tag{4.6}$$

Les différents paramètres de ce système d'équations sont :

- ρ, la densité de l'eau ;
- N_A, la constante d'Avogadro ;
- M_0, la masse moléculaire de l'eau ;
- a_0, l'activité dynamique de l'eau, représentée sous forme de nombre réel ;
- $f(\delta)$, le facteur de Cabbanes ;
- δ, le facteur de dépolarisation de l'eau ;
- k, la constante de Boltzmann ;
- T_k, La température en degrés Kelvin ;
- β_T, La compressibilité isotherme.

Ce système considère notamment que la variation du potentiel de diffusion de l'eau pure est fonction des fluctuations de densité dues à l'évolution de la température (équation 4.5), ainsi que des variations de salinité. De manière analogue, l'équation 4.6 ne modélise que les variations de concentration de sel dans l'eau pour calculer le coefficient de diffusion. Enfin, le coefficient de diffusion est calculé en considérant la salinité et la température comme deux paramètres indépendants (équation 4.4).

Pour simplifier cette approche, nous considérons que le milieu est incompressible, de manière analogue à l'hypothèse faite en dynamique des fluides (comme nous l'avons détaillé

en section 2.2.2), et que la salinité est constante dans tout le milieu (le milieu est parfaitement homogène). Boss et Pegau [BP01] ont proposé un modèle permettant d'approcher le coefficient de diffusion de l'eau salée avec une fonction linéaire (équation 4.7), dont nous avons choisi l'utilisation, conformément à ces considérations, et afin de simplifier la mise en œuvre de notre contribution :

$$\beta = \beta_{pw} \cdot \left(1 + \frac{0.3}{S_0} S\right). \tag{4.7}$$

Dans cette équation, $beta_{pw}$ correspond au coefficient de diffusion de l'eau pure, et S_0 correspond à la salinité de référence (i.e. 38.4 grammes par litre d'eau).

De manière similaire aux variations de température, l'impact de la salinité sur l'apparence de l'eau demeure subtil, comme le montre la figure 4.6. Une salinité exagérée ($275 g/L$, correspondant à la salinité de la mer morte), permet toutefois de mettre l'emphase sur cet impact qui devient nettement plus perceptible, comme le montre le résultat de droite à la figure 4.7.

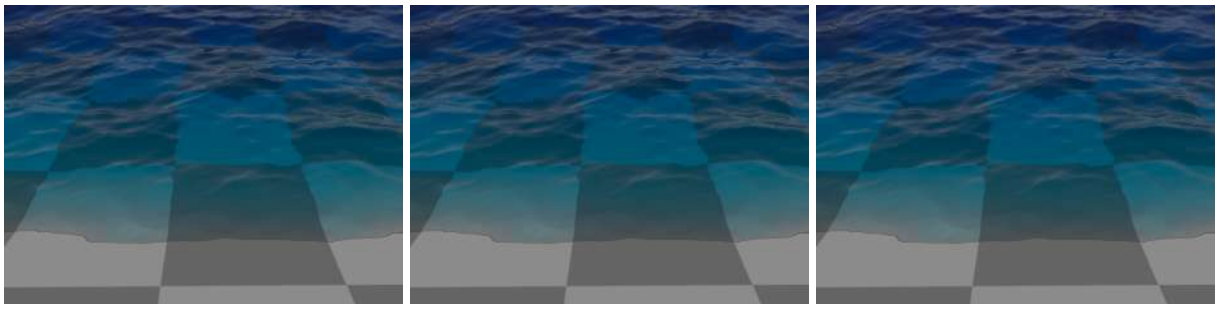


FIGURE 4.6 – *Des résultats avec une concentration de* 0, 38.4 *et* 275 *grammes de sel par litre d'eau, pour une température fixée à* 20 *degrés Celsius.*

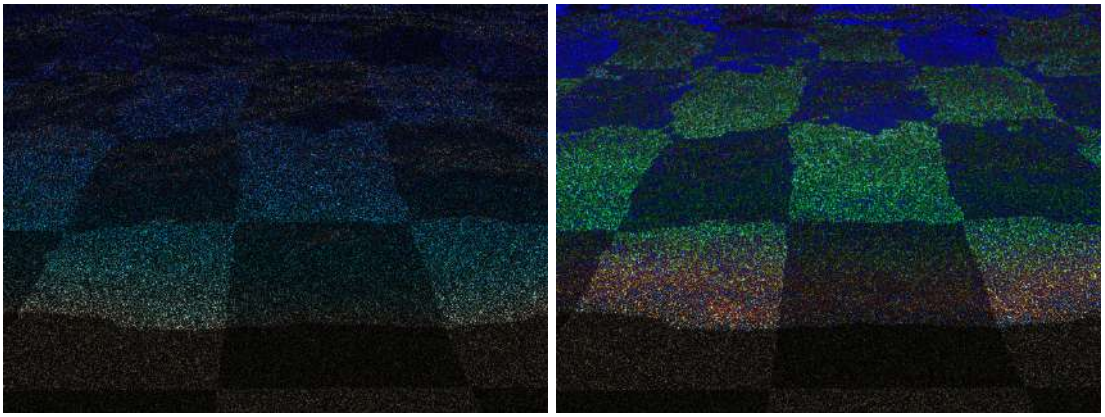


FIGURE 4.7 – *La différence absolue entre le rendu obtenu pour une salinité de* $10 g/L$ *et* $20 g/L$*, et la différence absolue entre le rendu avec des salinités de 10 et* $275 g/L$.

Malgré leur subtilité dans l'apparence de l'eau, nous choisissons de modéliser les effets de l'eau et du sel afin d'approcher aux mieux les modèles de l'optique en océanographie.

4.4 Optique des constituants océaniques

En plus de modéliser l'apparence de l'eau de mer sur la base de sa température et de sa salinité, nous souhaitons également introduire les variations de son apparence dues à la présence de pigments tels que la chlorophylle. Les pigments sont des substances chimiques produites par les entités végétales (planctons, algues) et animales (zooplanctons) dont la

présence dans l'eau en fait varier le coefficient d'absorption. L'influence de ces pigments sur la diffusion de la lumière, lorsqu'ils sont dilués dans l'eau, est très faible [Mob] et nous ne les prenons pas en compte dans les modèles car ils sont visuellement très négligeables pour le rendu.

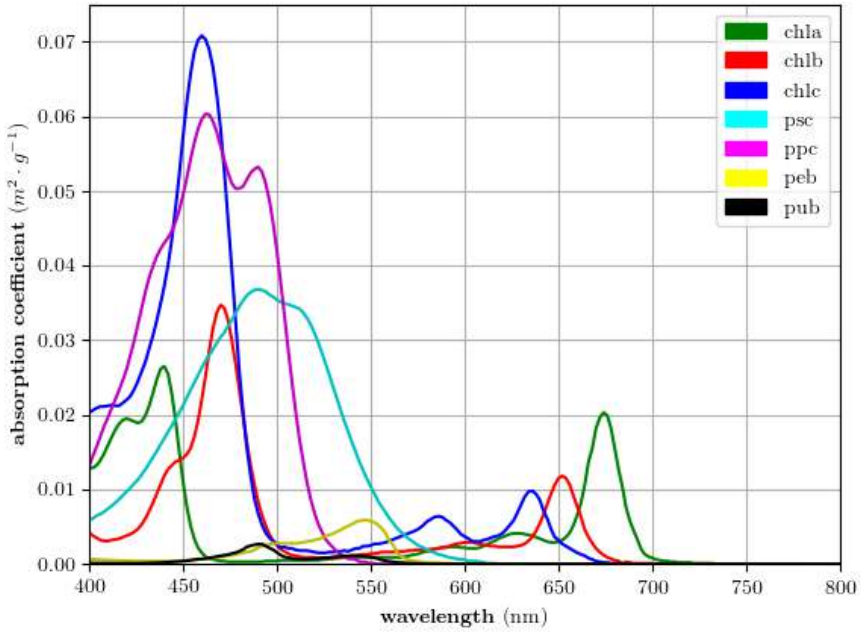


FIGURE 4.8 – *Les coefficients d'absorption spectraux des principaux pigments trouvés dans l'océan.*

Il n'existe à notre connaissance pas de modèle mathématique permettant d'approcher le coefficient d'absorption de pigments océaniques en fonction de leur concentration. A contrario, il existe des contributions dans la communauté océanographique ayant réalisé des mesures sur le coefficient d'absorption des principaux pigments. Bidigare et al. [BOMK90] proposent une étude regroupant l'absorbance des principaux pigments océaniques (figure 4.8) :

- CHLA : chlorophylle de type A, qui est le principal type de chlorophylle présent dans l'eau de mer ; son spectre d'absorption présente notamment des pics, au niveau du bleu et du rouge, expliquant notamment la teinte verdâtre des eaux chargées en planctons ;
- CHLB : chlorophylle de type B, dont le spectre d'absorption est globalement similaire mais décalé de celui de la chlorophylle A, lui attribuant ainsi une couleur jaune ;
- CHLC : chlorophylle de type C, plus rare que les deux précédentes, mais produite dans la réalité par certaines algues ; sur terre et de façon cumulée, les concentrations de chlorophylle sont en général comprises entre 0 et $60 mg/m^3$ (tous types cumulés) selon les données publiées sur le site de la NASA [Nas17] ;
- PSC : carotenoïdes photosynthétiques ;
- PPC : carotenoïdes photoprotecteurs : le spectre d'absorption des caroténoïdes leur permet d'avoir une couleur orange observée notamment dans les bassins du parc national de Yellowstone ;
- PEB : phycoérythrobiline ;
- PUB : phycourobiline.

Les spectres d'absorption sont donnés en m^2/g^{-1}. Combinés à une concentration donnée en paramètre, ils permettent d'obtenir un spectre d'absorption en $mg \cdot m^3$.

Il est aisé d'obtenir le coefficient d'absorption d'une eau ayant une constitution quelconque. La loi de Beer-Lambert, présentée en équation 4.8, permet de mettre en relation les propriétés optiques de l'eau avec sa constitution.

$$I(\lambda, X) = I_0(\lambda) \cdot e^{-\alpha X}. \tag{4.8}$$

Les différents paramètres de cette équation correspondent à :

- $I_0(\lambda)$, l'intensité du rayonnement incident, de longueur d'onde λ ;
- I, l'intensité du rayonnement sortant ;
- X, la longueur du trajet du rayon ;
- α, le coefficient d'absorption du milieu.

Cette loi énonce une propriété d'additivité (équation 4.9) extrêmement utile puisqu'elle énonce que le coefficient d'absorption d'un milieu donné correspond à la somme des coefficients d'absorption des constituants qui le composent :

$$\alpha = \sum_{i=1}^{n} \alpha_i(\epsilon_{\lambda,i}, l, C). \tag{4.9}$$

Les termes de cette équation sont :

- α, l'absorbance ;
- C, la concentration en $\text{mol} \cdot L^{-1}$;
- ϵ, l'absorbance molaire ;
- l, la longueur du trajet optique dans la solution ;

4.5 Fonctions de phase

Les fonctions de phase permettent de décrire la distribution angulaire de la lumière en fonction de la direction du rayon lumineux incident. Il existe plusieurs fonctions de phase :

1) La fonction de phase de Henyey-Greenstein (équation 4.10).

$$b(g, \theta) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}} \tag{4.10}$$

Le paramètre g, compris dans l'intervalle $[-1; 1]$, permet de faire varier la capacité du milieu participant à diffuser la lumière. Une valeur de 0 signifie que le milieu est isotrope, c'est à dire que la lumière incidente est diffusée de manière égale dans toutes les directions. A contrario, une valeur tendant vers -1 met l'emphase sur le potentiel de rétrodiffusion (*backscattering*) du milieu. Une valeur tendant vers $+1$ correspond à un milieu diffusant la lumière vers l'avant (*forward scattering*). La fonction de Henyey-Greenstein est en fait une approche gaussienne dont le paramètre g permet d'en contrôler le pic.

2) La fonction de phase de Fournier-Forand [FJ99] (équation 4.11) ressemblant davantage (figure 4.10) à une fonction delta de Dirac.

$$b(\theta) = \frac{1}{4\pi} \frac{1}{(1-\delta)^2 \delta^\nu} ([\nu(1-\delta) - (1-\delta^\nu)] + \frac{4}{u^2}[\delta(1-\delta^\nu) - \nu(1-\delta)]) \tag{4.11}$$

$$\nu = \frac{3-\mu}{2} \tag{4.12}$$

$$\delta = \frac{u^2}{3(n-1)^2} \tag{4.13}$$

$$u = 2\sin(\theta/2). \tag{4.14}$$

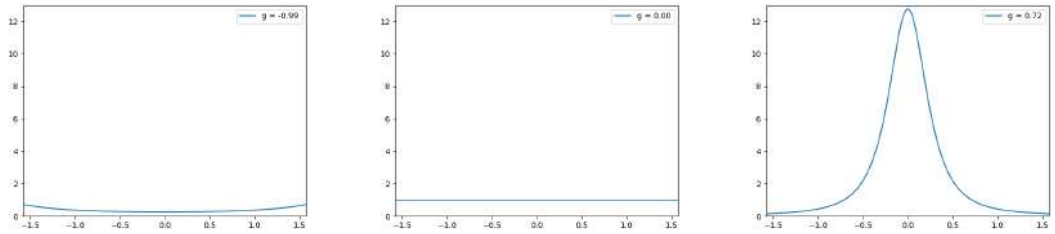


FIGURE 4.9 – *La fonction de phase de Henyey-Greenstein avec différentes valeurs de g. Une valeur de* 0 *revient à utiliser une fonction isotrope. Une valeur de* $+1$ *produit un effet de prodiffusion. Une valeur de* -1 *produit un effet de rétrodiffusion.*

Le paramètre μ permet de faire varier le pic de la fonction. Nous proposons d'introduire cette fonction dans le contexte de l'informatique graphique. Elle semble faire consensus dans la communauté océanographique car elle approche mieux les mesures de référence (du coefficient de diffusion de l'eau) que la fonction d'Henyey-Greenstein [Mob, MSBo2].

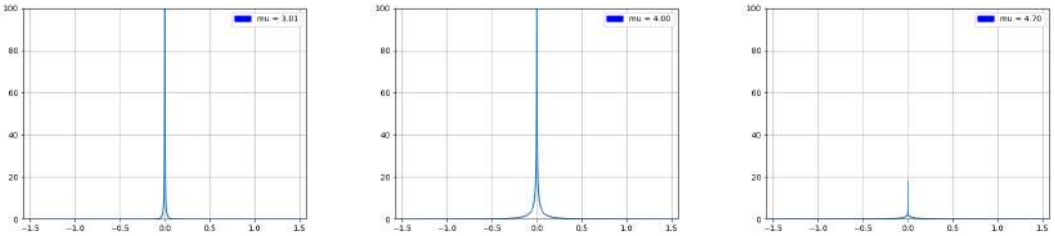


FIGURE 4.10 – *La fonction de phase de Fournier-Forand avec différentes valeurs de* μ.

Les auteurs définissent également une fonction de répartition pour déterminer la distribution angulaire de la fonction de phase $\omega(\theta)$, définie en équation 4.15. Nous souhaitons obtenir l'inverse de cette fonction afin de réaliser un échantillonnage d'importance dans le volume d'eau.

$$\omega(\theta) = \frac{1}{(1-\delta)\delta^{\nu}}\left((1-\delta^{\nu+1}) - \frac{u^2}{4}(1-\delta^{\nu})\right). \tag{4.15}$$

Indéterminée lorsque $\theta = 0$, cette fonction n'est pas directement inversible. Pour pouvoir réaliser un échantillonnage d'importance sur la fonction de phase de Fournier-Forand, nous approchons la réciproque de la fonction de répartition par l'expression polynomiale suivante :

$$\frac{ax^5 + bx^4 + cx^3 + dx^2 + ex}{f + gx^5 + hx^4 + ix^3 + jx^2 + kx}. \tag{4.16}$$

Cette expression comporte des indéterminations lorsque $f \leq 0$ et il faut donc choisir une valeur positive et supérieure à 0. Nous proposons un tableau de coefficients (tableau 4.1) pour plusieurs valeurs de μ. Pour des valeurs de μ intermédiaires aux valeurs échantillonnées dans le tableau (e.g., $\mu = 3.18$), nous réalisons une interpolation linéaire de la valeur de θ en sortie des deux expressions polynomiales aux valeurs de μ les plus proches (3.10 et 3.25 si $\mu = 3.18$).

Le tableau 4.2 montre que nos interpolations sont exactes pour plusieurs valeurs de μ.

4.6 Reconstruction de surface

Nous souhaitons réaliser une simulation d'éclairage du liquide représenté par notre simulateur SPH. Cette opération n'est pas directement possible car nous simulons un ensemble

	$\mu = 3.1$	$\mu = 3.25$	$\mu = 3.5$	$\mu = 4$	$\mu = 4.5$	$\mu = 5$
a	-0.00154049	-0.0186513	-97.788	-0.950314	2.6839	76.7885
b	0.0211677	0.0822799	322.649	2.02072	-3.33357	-147.309
c	-0.0469381	-0.117278	-354.458	0.0427001	-1.91619	37.596
d	0.0364888	0.0620757	131.737	-2.35195	2.49813	32.7075
e	-0.0091779	-0.00842619	-2.12982	1.23884	0.0699701	0.401825
f	0.585563	0.110025	34.3385	0.41686	0.00291362	0.00414597
g	-0.603657	-0.109605	-6.4177	0.169559	-0.391801	10.105
h	2.96962	0.514295	0.0550929	-1.25414	2.91245	13.7573
i	-5.8694	-0.987982	14.3421	2.47526	-4.38233	-64.3274
j	5.83013	0.981439	31.0258	-1.45134	1.39295	37.6433
k	-2.91225	-0.508173	-73.3437	-0.356204	0.466536	2.87703

TABLE 4.1 – *Le tableau de coefficients pour l'expression polynomiale décrite en équation 4.16.*

de particules discrètes (i.e. des sphères). Notre objectif ici est donc d'obtenir un modèle géométrique conforme à celui attendu pour une utilisation dans un moteur de rendu, tel qu'un maillage polygonisé. Ce besoin est souvent nécessaire dans les simulations basées physique, qui utilisent souvent des discrétisations à base de points.

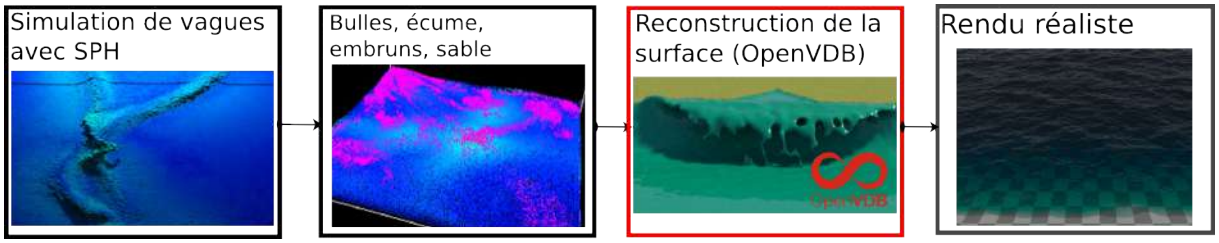


FIGURE 4.11 – *Dans notre chaîne de traitements, après l'étape de simulation de vagues et de particules océaniques, la surface du liquide est reconstruite pour chaque trame en utilisant la librairie OpenVDB.*

Nous utilisons la librairie OpenVDB [MLJ+13] pour réaliser une reconstruction de la surface du liquide. Pour ce faire, la librairie prend en entrée le nuage de points de chaque trame de simulation SPH que nous exportons au format binaire. Le nuage de points est alors converti en *levelset*. Le module de polygonisation de *levelset* d'OpenVDB est alors utilisé, qui permet de trianguler ce dernier pour la valeur 0. La polygonisation résultante pouvant être grossière, nous filtrons le *levelset* résultant avant la polygonisation grâce à un filtre gaussien. Le résultat peut être amélioré davantage en utilisant les mécanismes d'érosion et de dilatation des voxels de la structure VDB. Nous illustrons en figure 4.12 les trois étapes de reconstruction de surface, tenant compte de ces améliorations.

4.7 Visualisation de particules marines

Nous souhaitons mettre en œuvre une méthodologie pour la visualisation des particules générées et simulées, telle que nous l'avons décrite en section 3.6. Dans cette section nous décrivons la méthode mise en œuvre pour visualiser les particules marines dont la dynamique est simulée en amont de notre chaîne de traitement, nommément les bulles, embruns, l'écume et le sable.

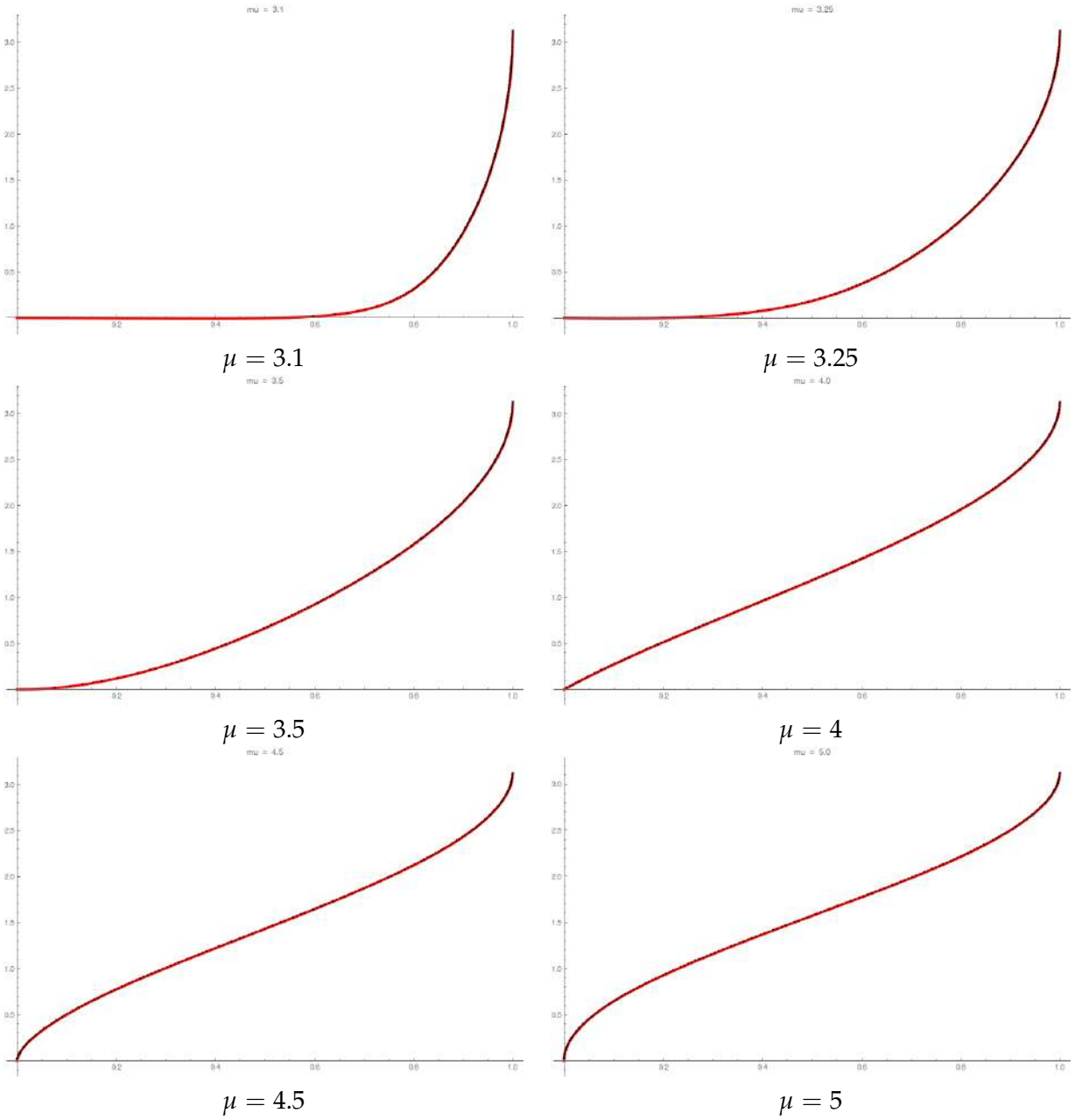


TABLE 4.2 – *Pour différentes valeurs de μ, notre approche polynomiale (en rouge) correspond bien à la CDF discrétisée (en noir).*

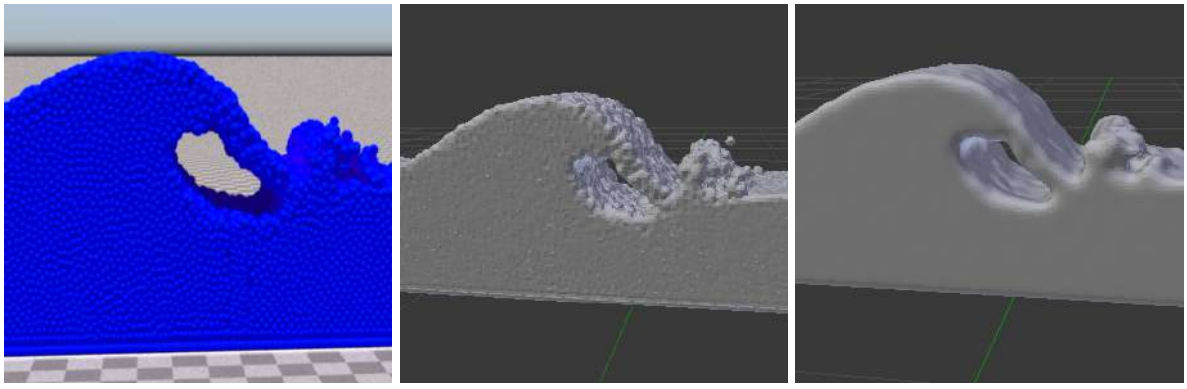


FIGURE 4.12 – *La visualisation du nuage de points à gauche, sa polygonisation au milieu, puis l'amélioration du résultat à droite après filtrage.*

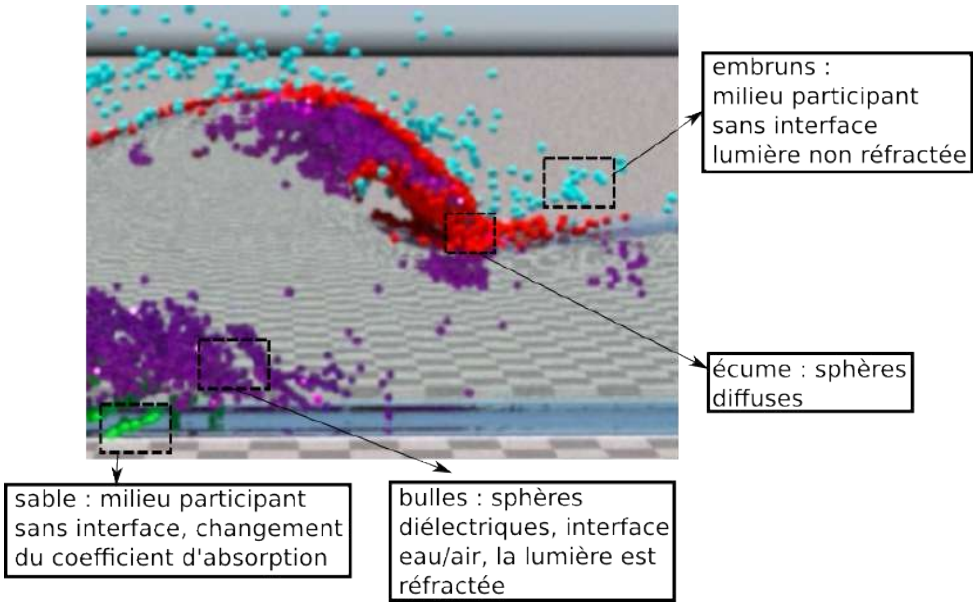


FIGURE 4.13 – *L'interaction entre le rayonnement lumineux et nos différents milieux participants.*

Bulles

Les bulles sont des sphères d'air présentes dans l'eau. La plupart du temps, elles sont générées sous l'effet des vagues déferlantes et du vent qui conduisent le volume d'eau à assimiler un certain volume d'air. Les bulles évoluent dans le volume d'eau sous les effets de la flottabilité et de diffusion de la matière. Leur apparence est caractérisée par une interface, due au passage du rayonnement lumineux d'un milieu participant, que nous considérons homogène (le volume d'eau), à un milieu sans interaction (i.e. le volume d'air inclus dans la bulle). Nous proposons d'afficher chaque particule de bulle individuellement sous forme de sphère, en considérant l'interface entre le volume d'eau et le volume d'air comme une interface diélectrique permettant de modéliser la réfraction du rayon lumineux la traversant par un indice statique ($\eta = 1.33333$), permettant de rester en cohérence avec la visualisation du volume d'eau réalisée en amont.

Embruns

Les embruns sont des particules fines (i.e. des aérosols) générées sous l'effet de la prise au vent de l'eau au niveau des crêtes de vagues, comme nous l'avons détaillé et implémenté en section 3.6. Selon leur densité, leurs propriétés optiques sont similaires à la brume ou au brouillard, et sont des milieux participants sans interface (un rayon lumineux ne change pas de trajectoire lorsqu'il intersecte le milieu). Pour approcher les effets de l'interaction entre la lumière et les embruns, nous définissons les particules d'embruns comme un ensemble de sphères de rayon faible, avec des coefficients de diffusion et d'absorption identiques à ceux de la brume.

Écume

L'écume de mer correspond à de la mousse opaque et blanchâtre dont l'origine peut être de nature éolienne, ou due à la turbulence d'un liquide au niveau de sa surface. Elle est donc typiquement observée au niveau des eaux peu profondes dans le cas de houle ou de vague déferlantes, et lorsqu'une vague rentre en contact avec un massif rocheux. Pour obtenir des effets visuellement plausibles il est par consquent incontournable de représenter les effets

de l'écume sur l'apparence de la surface de l'eau de mer. Pour approcher cet effet, nous modélisons les particules d'écume comme un ensemble de sphères diffuses permettant de reproduire l'aspect blanc et mat observable dans la nature.

Sable

Le sable est constitué de plusieurs matériaux diélectriques (e.g. quartz, limonite) à l'état granulaire, avec une forte capacité à diffuser la lumière. Simuler l'interaction entre la lumière et chaque grain de sable s'avère complexe [MPH+15, MPG+16]. Nous contournons cette limite en considérant chaque particule de sable, précédemment simulée par notre méthode, comme étant une région d'eau chargée en sable. Ainsi, nous ne simulons pas l'interaction entre la lumière et chaque grain de sable, et nous ignorons les effets de diffusion. De manière analogue à Mobley [Mob] pour les planctons, nous faisons l'hypothèse que le sable dilué dans l'eau a exclusivement une influence sur le potentiel d'absorption du volume. Chaque particule simulée est considérée comme un milieu participant sans interface dont le spectre d'absorption est identique au volume d'eau qui le contient. Nous ajoutons toutefois à ce spectre un deuxième spectre d'absorption correspondant à celui du sable, issu des travaux de Premoze et Ashikhmin [PA00].

Nous représentons l'apparence des particules marines par une approche purement empirique, résultant de nos observations. Cette méthodologie a pour atout sa simplicité dans la mesure où elle est directement intégrable dans tous les moteurs de rendu. Toutefois, notre approche ne se base aucunement sur des modèles physiques, ou d'optique en océanographie. Il existe assez peu de données disponibles qui permettent de caractériser l'apparence de l'ensemble de ces particules. Certains modèles physiques permettent toutefois de les approcher en partie [Mob].

4.8 Mise en œuvre dans Mitsuba

Les résultats sont produits avec le moteur de rendu Mitsuba [Wen10], qui permet d'échantillonner les spectres d'absorption et de diffusion sur trente longueurs d'onde. Bien que le temps de rendu augmente de manière linéaire avec le nombre de longueurs d'onde échantillonnées, la qualité du résultat obtenu de cette manière permet de mieux représenter les effets des spectres d'absorption et de diffusion, comme nous l'illustrons à la figure 4.14.

Nous avons implémenté la fonction de phase Fournier-Forand, telle que décrite par Fournier et Jonasz [FJ99], dans le module *phase* de Mitsuba, ainsi que notre fonction de répartition approchant la réciproque de la CDF 4.16 liée à la fonction de Fournier-Forand.

Notre simulateur physique traite en premier lieu les aspects dynamiques (la dynamique des particules fluides, ainsi que celles des particules marines et du sable) avec une implémentation entièrement parallélisée sur GPU, avec des performances interactives, comme nous l'avons détaillé en section 3.8. Ce processus communique ensuite avec notre processus de reconstruction de surface par l'intermédiaire d'un exporteur XML selon le schéma XML de Mitsuba. Le dernier processus de notre chaîne de traitement permet d'intégrer les spectres d'absorption et de diffusion du milieu, calculés en fonction de la constitution saisie par l'utilisateur. Cette constitution est caractérisée par une température, une salinité, ainsi que des concentrations pour chaque type de pigment, comme nous l'avons détaillé en section 4.4. Ces deux dernières étapes ne sont actuellement pas parallélisées sur carte graphique. La partie visualisation de notre chaîne de traitement ne peut donc se faire avec des performances interactives. Nous

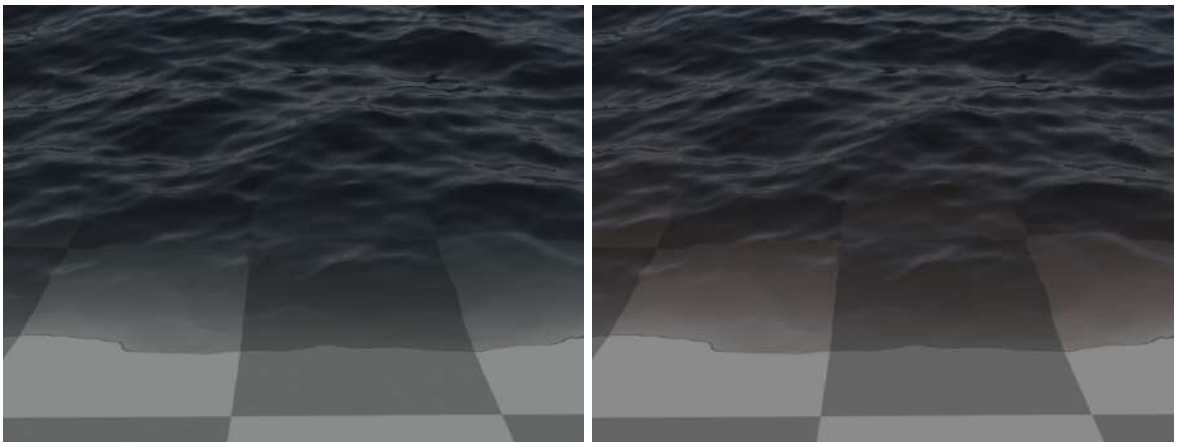


FIGURE 4.14 – *Comparaison entre un rendu RGB à gauche et un rendu spectral à droite. Nous pouvons constater que le rendu RGB résulte en un sous-échantillonnage des spectres d'absorption et de diffusion, manquant certains effets visuels. Dans cet exemple l'eau comporte une température de 20 degrés Celsius et une concentration de* $10 mg/L$ *de PSC.*

abordons toutefois au chapitre 5 des perspectives pour une mise en application interactive de ces travaux.

4.9 Résultats et discussion

Dans cette section, nous présentons un ensemble de résultats mettant en œuvre pour chaque pigment, son spectre d'absorption avec des concentrations plausibles. Ces résultats sont des cas de laboratoire dans la mesure où les IOP et les AOP utilisées ne correspondent pas à des configurations plausibles, et ont été réalisés pour étudier les constituants un à un. Nous présentons par la suite plusieurs cas de figure réalistes, i.e. avec une composition plausible de l'eau de mer, ainsi que des conditions d'éclairage et une apparence du domaine plausible. Ces résultats ont été générés d'une part avec un modèle procédural (figure 4.22), et d'autre part avec un solveur physique de type IISPH (figure 4.25) pour lequel les vagues ont été produites à l'aide de notre méthode présentée en chapitre 3. Dans le cas du rendu de simulations physiques, nous présentons également des rendus de particules marines de type embruns, écume, bulles et sable.

4.9.1 Présence de pigments

Nous contrôlons ici l'apparence de l'eau en faisant varier la concentration d'un seul pigment à la fois. Conformément aux mesures de spectres d'absorption de Bidigare et al. [BOMK90], faire varier la concentration des différents types de chlorophylle (figures 4.15, 4.16 et 4.17) influe largement sur l'apparence verte de l'eau. On constate également que la présence de caroténoïdes (psc et ppc, figures 4.18 et 4.19) donne une teinte rouge voire orange à l'eau. Les différents types de phycobiline (figures 4.20 et 4.21) dont nous avons pu récupérer les spectres d'absorption mettent l'emphase sur sa teinte bleutée.

4.9.2 Particules marines

Nous avons choisi de représenter les particules d'embruns par un milieu participant sans interface, dont les coefficients d'absorption et de diffusion correspondent à ceux de la brume ou du brouillard, en fonction de la densité de particules d'embruns. En effet, représenter ces particules par un milieu participant comportant une interface serait peu réaliste dans la mesure

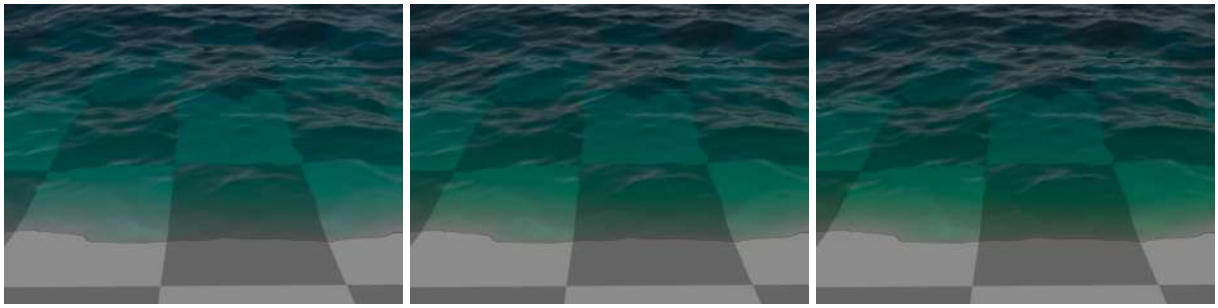


FIGURE 4.15 – *Des résultats avec une concentration de* 10, 20 *et* 30 *mg de CHLA par litre d'eau.*

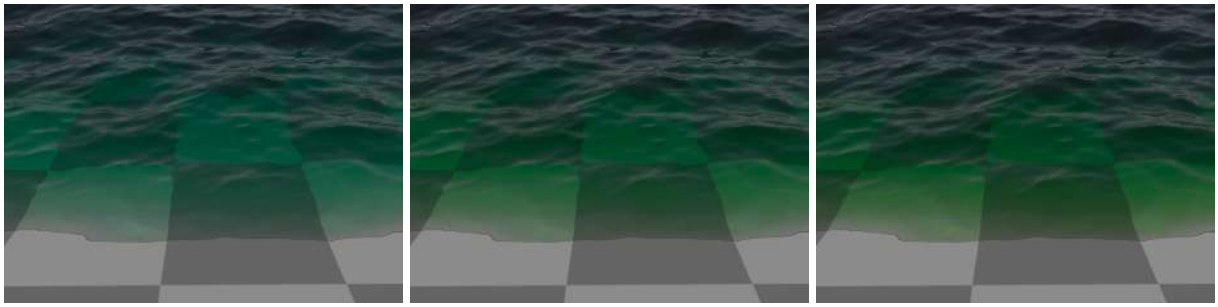


FIGURE 4.16 – *Des résultats avec une concentration de* 10, 20 *et* 30 *mg de CHLB par litre d'eau.*

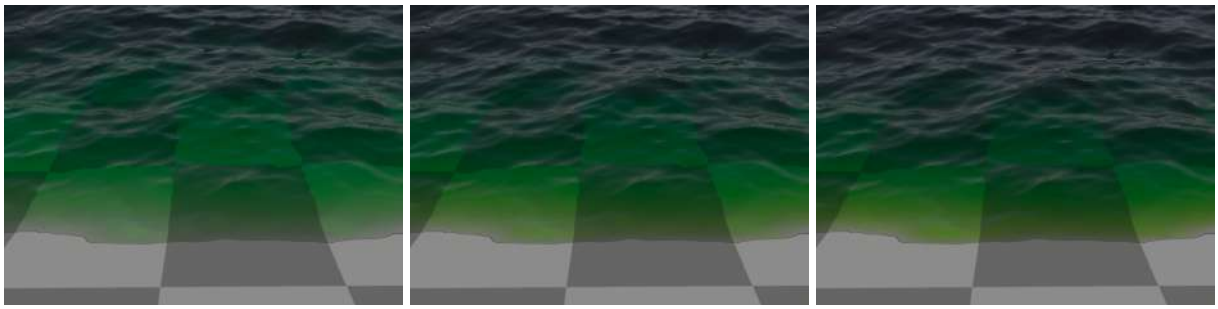


FIGURE 4.17 – *Des résultats avec une concentration de* 10, 20 *et* 30 *mg de CHLC par litre d'eau.*

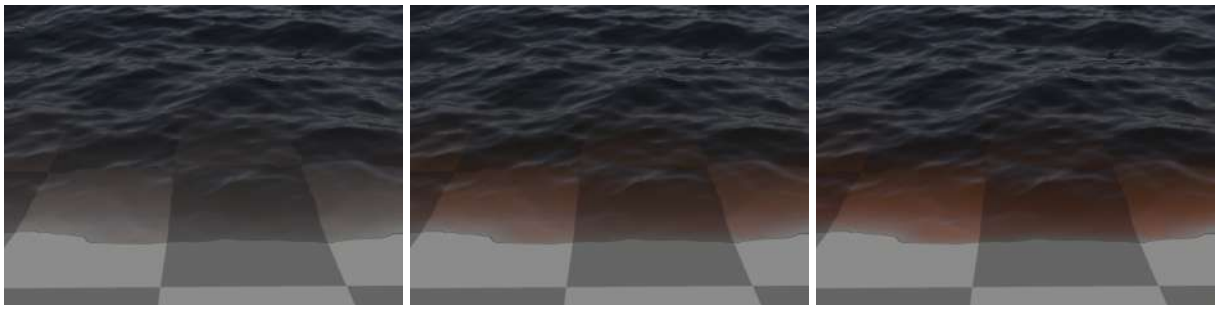


FIGURE 4.18 – *Des résultats avec une concentration de* 10, 20 *et* 30 *mg de PSC par litre d'eau.*

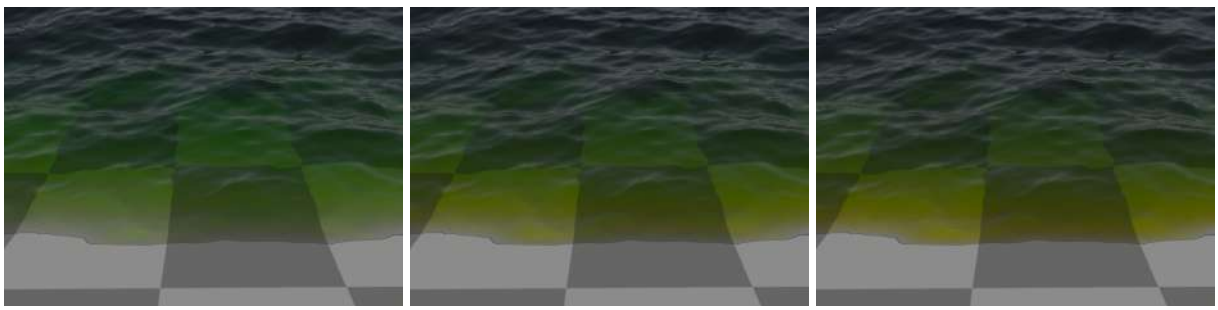


FIGURE 4.19 – *Des résultats avec une concentration de* 10, 20 *et* 30 *mg de PPC par litre d'eau.*

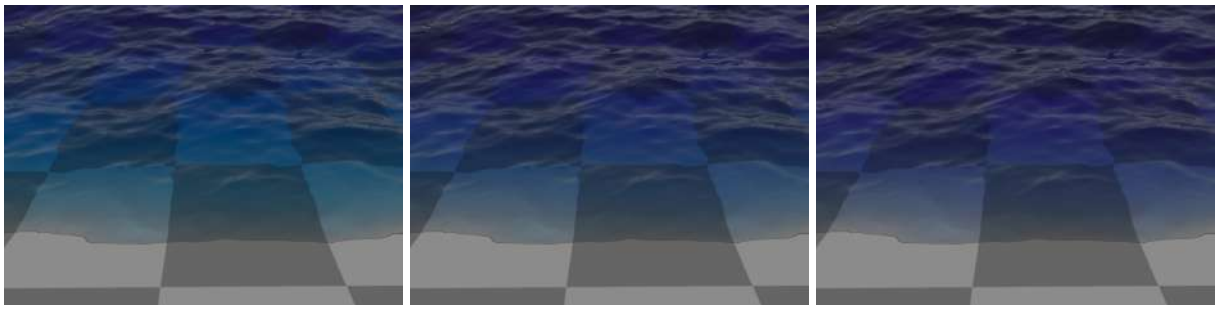


FIGURE 4.20 – *Des résultats avec une concentration de* 10, 20 *et* 30 *mg de PEB par litre d'eau.*

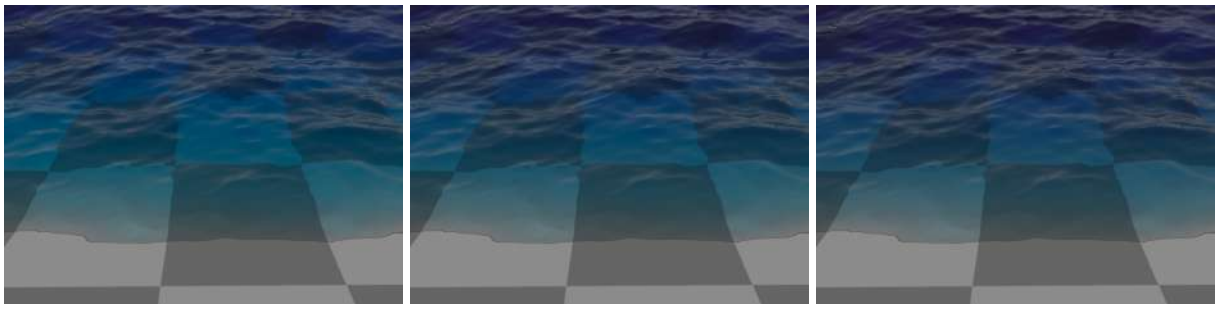


FIGURE 4.21 – *Des résultats avec une concentration de* 10, 20 *et* 30 *mg de PUB par litre d'eau.*

ou des taches spéculaires seraient visibles, ce qui n'est généralement pas observable dans la nature.

4.10 Conclusion

Les travaux présentés dans ce chapitre constituent une méthodologie permettant de reproduire l'apparence de l'eau de mer pour des moteurs de simulation d'éclairage réaliste. Nous avons en premier lieu détaillé les caractéristiques optiques de l'eau pure, puis celles des constituants océaniques les plus courants. Nos travaux introduisent également la fonction de phase de Fournier-Forand pour les moteurs de rendu en informatique graphique.

D'autre part, nous avons développé un outil permettant de calculer les coefficients d'absorption et de diffusion pour une constitution donnée par l'artiste et dans le respect de la loi optique de Beer-Lambert. Les paramètres couvrent une large partie des constituants océaniques courants tels que le sel, la chlorophylle ou les caroténoïdes.

Notre étude, ainsi que les travaux qui en découlent, ont été intégrés dans le moteur de rendu Mitsuba [Wen10], compilé de sorte à effectuer des rendus sur trente longueurs d'onde pour améliorer l'impact des données spectrales sur l'apparence du rendu final. Les résultats obtenus mettent en scène un grand nombre de configurations, telles que des océans générés à l'aide d'un modèle de simulation procédurale, ou des volumes d'eau obtenus grâce à notre implémentation GPGPU de la méthode SPH et de la méthode de simulations de particules marines. Le rendu des particules marines a été effectué selon une méthodologie simple mais plausible, augmentant le spectre d'effets possibles.

FIGURE 4.22 – *Résultat obtenu en utilisant un modèle d'animation procédurale, avec une concentration de 60mg de chlorophylles par litre d'eau.*

FIGURE 4.23 – *Résultat obtenu en utilisant un modèle d'animation procédurale, avec une concentration de 10mg de chlorophylles par litre d'eau.*

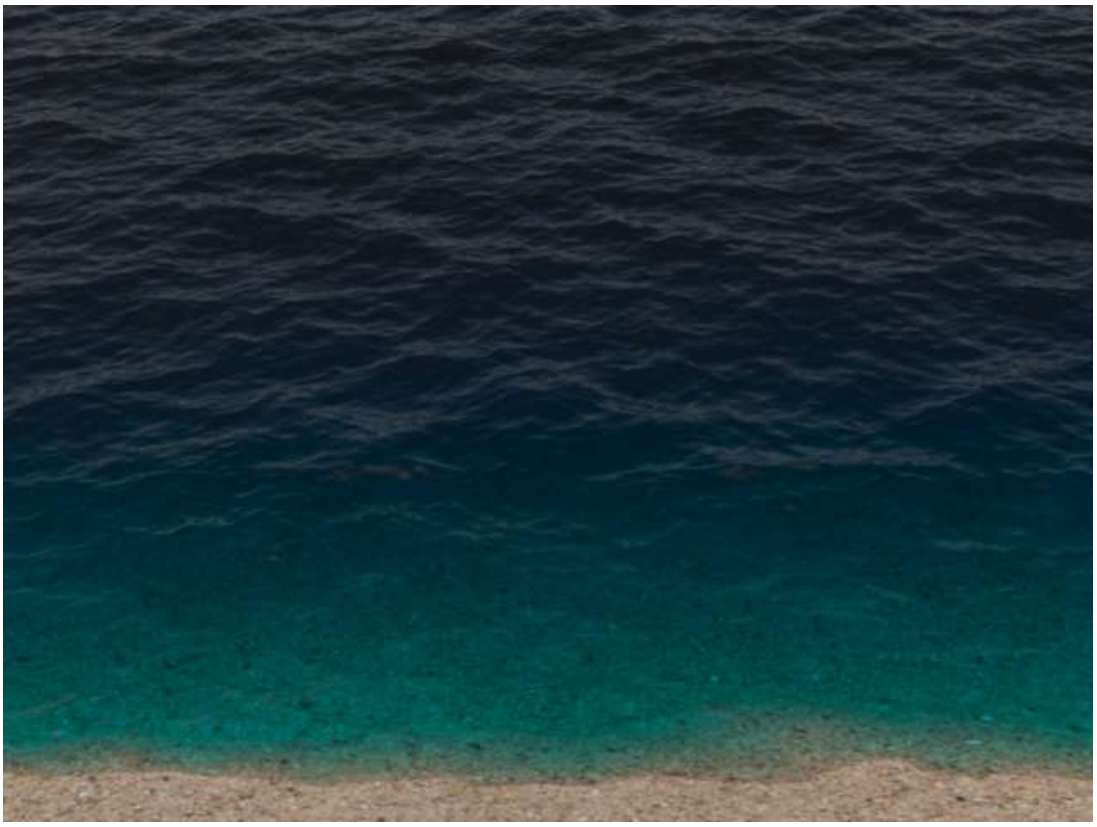


FIGURE 4.24 – *Résultat obtenu en utilisant un modèle d'animation procédurale, avec une concentration de 10mg de cyanobactéries par litre d'eau.*

FIGURE 4.25 – *Résultat obtenu en utilisant notre simulateur IISPH.*

Conclusion et perspectives 5

Sommaire

5.1 Rappel de la problématique

Les travaux de cette thèse ont été développés sur la base d'une double problématique. D'une part, la simulation de la dynamique de phénomènes complexes et turbulents dans les milieux océaniques, principalement caractérisés par des vagues déferlantes. D'autre part l'apparence sur ordinateur de ces phénomènes sur la base d'un ensemble de propriétés physiques et optiques. Également, nos travaux ont été réalisés avec pour objectif de proposer des méthodes avec une facilité de contrôle par l'utilisateur ou l'artiste.

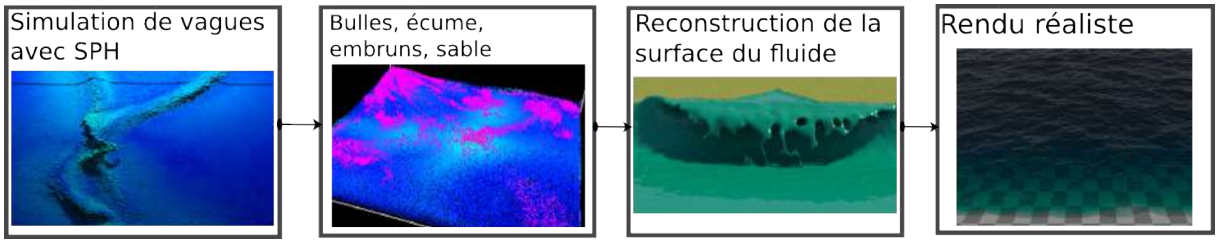


FIGURE 5.1 – *La chaîne de traitements de nos travaux.*

5.2 Rappel des objectifs

Cette double problématique nous a conduits à l'élaboration d'une chaîne de traitements (que nous rappelons ici sur la figure 5.1) complet de simulation et de rendu comportant une nouvelle méthode de simulation de vague, ainsi qu'une méthodologie complète pour leur rendu réaliste. L'élaboration de ce système constituait l'objectif majeur de ces travaux de thèse et, pour parvenir à le développer, il était en premier lieu nécessaire de bien comprendre les mécanismes en jeu. Ainsi, l'étude de la physique des vagues fut nécessaire afin d'en reproduire au mieux la dynamique dans le contexte d'une simulation numérique, mais également l'étude de leurs propriétés optiques et de leurs interactions avec la lumière afin de parvenir à une simulation d'éclairage réaliste. Par ailleurs, un objectif clé était de faire de cette chaîne de traitement un outil contrôlable par un utilisateur ou un artiste dans ses aspects dynamiques et optiques.

5.3 Bilan des travaux réalisés

La phase d'implémentation des modèles et méthodologies proposés a principalement compris deux étapes : l'implémentation des modèles permettant la simulation de la dynamique d'un environnement océanique, puis la conception d'une méthodologie permettant de reproduire son apparence grâce aux méthodes d'éclairage global. Nous faisons dans les deux sections suivantes un bilan des contributions réalisées dans ces deux domaines.

5.3.1 Simulation de fluides

La première étape de nos travaux a été caractérisée par les traitements liés à l'animation de vagues. La toute première étape a consisté au développement d'un simulateur SPH parallélisé sous CUDA. Nous avons enrichi l'implémentation de ces travaux existants par l'introduction de notre modèle de forces externes. Ce dernier est une simplification 2D d'un modèle de soliton présenté par Radovitzky et Ortiz [RO98] et Darles et al [DCG11]. Il comporte peu de paramètres et demeure de ce fait facile à contrôler. Nous avons enrichi notre modèle pour pouvoir représenter une grande quantité d'effets possibles, tels que des interactions d'un

nombre quelconque de vagues, des crêtes et des fronts de vagues tracés par l'utilisateur, et des propagations courbes définies dans des carreaux de Bézier. Cet ensemble d'extensions permet de lever la limite inhérente à la nature 2D de notre modèle, tout en conservant une facilité de contrôle certaine. Un autre avantage majeur de notre méthode est son coût de calcul quasi-nul qui permet de conserver l'interactivité d'une simulation SPH sur GPU. Ces travaux ont été publiés et présentés aux journées de l'AFIG [BDM+15a], à *VRIPHYS* [BDM+15b] et *Computers & Graphics* [BDM+16]. Nous avons ensuite implémenté une version parallélisée sur GPU des travaux d'Ihmsen et al. [IAAT12] pour enrichir notre visualisation, permettant de représenter les effets d'écume, des bulles et d'embruns. Nous avons également enrichi cette méthode pour simuler les effets de mélange de sable dans les régions océaniques appropriées.

5.3.2 Rendu de fluides

La seconde étape de nos travaux a consisté au développement d'une méthodologie permettant de compléter notre chaîne de traitement, notamment dans les aspects visualisation et rendu réaliste qu'elle nécessitait. La première étape fut d'intégrer les travaux de Museth et al. [MLJ+13] à notre simulateur SPH afin de pouvoir extraire une surface du liquide de bonne qualité. À la suite de cette première étape, nous avons réalisé une étude sur les propriétés optiques de l'eau pure et de l'eau de mer. Nous avons pu extraire une grande quantité de données et de modèles de la communauté d'optique et d'océanographie. Nous avons ensuite intégré ces données et ces modèles dans le logiciel Mitsuba [Wen10], afin de réaliser un rendu réaliste de scènes océaniques, tirant parti à la fois de données concrètes et de modèles d'éclairage global physiquement réalistes. En outre nous avons utilisé une version compilée permettant de faire des rendus sur trente longueurs d'onde pour améliorer significativement l'impact de nos données spectrales sur la qualité du rendu final. L'objectif de ces travaux était notamment de réaliser un pont entre deux disciplines complémentaires, nommément l'informatique graphique et l'optique en océanographie. Nous avons pu réaliser des résultats permettant de mettre en œuvre l'ensemble de notre chaîne de traitement. Ces travaux sont présentés dans un article qui est actuellement en cours de soumission.

5.4 Travaux futurs

Tant bien sur les aspects animation que rendu, les travaux que nous avons publiés et mis en œuvre présentent un grand nombre de perspectives de recherche, qu'elles soient de natures fondamentale ou technique. Nous détaillons ces perspectives dans cette section.

5.4.1 Simulation de fluides

Notre modèle de soliton pourait être amélioré dans le sens où la corrélation entre les paramètres est sa limite majeure. Il serait intéressant de mettre en œuvre un modèle de plus haut niveau utilisant celui-ci dans le but de ne pas manipuler directement les paramètres du modèle. Contrôler le modèle via une interface utilisant par exemple une sémantique permettant d'ajuster les paramètres automatiquement permettrait de faciliter le travail de l'artiste.

Cette approche permet de simuler un grand nombre de phénomènes plausibles, y compris des scénarios non réalistes, selon le desideratum de l'artiste. Toutefois, dans le cas de scénarios plausibles tels que la mer croisée, valider notre contribution est peu aisé du fait du peu de données disponibles pour comparer nos résultats à une source de référence.

Nous pourrions également améliorer davantage notre système de simulation de particules marines en intégrant un système de forces internes à ces particules. Actuellement, seules les particules du fluide ont une influence sur leur dynamique, et il serait par conséquent

intéressant d'ajouter des effets liés par exemple à la viscosité de l'écume. Les performances de ce système pourraient être améliorées. Actuellement, il a été développé comme une méthode de post-traitement à chaque trame de simulation SPH. Il serait en outre plus efficace de l'intégrer directement à la simulation SPH car cela permettrait d'uniformiser notre simulation et de n'utiliser qu'une seule structure accélératrice, et un seul calcul de voisinage. À contrario, le code sous-jacent d'une telle uniformisation s'en verrait quelque peu complexifié.

Ajouter du niveau de détail dans la simulation, en combinant un modèle surfacique utilisant des *Shallow Water SPH* [BBC+11] pour représenter les eaux profondes et intermédiaires (dans lesquelles les vagues se propagent), et des SPH classiques pour les eaux peu profondes (dans lesquelles les vagues déferlent) a fait l'objet d'une première implémentation lors du commencement des travaux. Les résultats obtenus n'ont pas permis de mettre en avant des simulations stables. Cette perspective est hautement intéressante car elle permettrait de réduire dramatiquement le nombre de particules à simuler, et par conséquent d'obtenir de meilleures performances. La mise en œuvre actuelle utilise une zone tampon de la simulation dans laquelle une particule SWSPH comporte des particules voisines SPH, et inversement. Les particules de chaque type contribuent à l'estimation de densité, de forces de pression et de viscosité de chaque type de particule voisine. Une partie exploratoire nécessaire consisterait à concevoir une fonction de lissage permettant d'améliorer l'estimation des champs SPH dans la zone tampon.

5.4.2 Rendu de fluides

Un grand nombre de perspectives à nos travaux portant sur le rendu de fluides est envisageable. Ils concernent avant tout la qualité et la quantité de données utilisées pour effectuer un rendu réaliste d'une scène océanique.

Dans les résultats faisant intervenir notre simulateur SPH, nous avons utilisé la librairie OpenVDB pour reconstruire la surface du liquide. Cette librairie est parallélisée sur CPU, et obtenir la surface d'une seule trame coûte généralement plusieurs secondes de calcul, y compris pour des simulations comprenant peu de particules. Une version parallélisée sur GPU avec CUDA, GVDB, a été développée au moment de la rédaction de cette thèse. L'intégrer à nos travaux serait un atout majeur car elle permettrait d'augmenter dramatiquement les performances générales de tout notre chaîne de traitements.

Nous avons utilisé le moteur de rendu Mitsuba [Wen10], parallèle sur CPU, pour effectuer nos simulations d'éclairage. Une image nécessite parfois plus d'une heure à calculer. Ajoutée au temps requis pour reconstruire une surface d'une simulation SPH, il en résulte que les performances globales de notre chaîne de traitements ne permettent pas d'assurer une utilisation interactive. Développer un moteur de tracé de chemin en utilisant l'API massivement parallèle NVidia Optix permettrait d'outrepasser cette limitation et, conjointement à une utilisation de la librairie GVDB, nous permettrait de proposer un outil complet de simulation et de rendu physiques en temps interactif.

Afin de simplifier en premier lieu le calcul du coefficient de diffusion spectral de l'eau pure, nous avons utilisé un modèle linéaire publié initialement par Boss et Pegau [BP01]. Une perspective à explorer consiste en l'implémentation du modèle à base de dérivées partielles publié par Zhang et al. [ZHH09].

Les données sur l'absorption de la lumière par les nombreux pigments contenus dans l'eau de mer sont extrêmement rares. Les données utilisées pour nos travaux, bien que couvrant la plupart des pigments communs, sont incomplètes. Réaliser des mesures exhaustives serait indéniablement avantageux pour améliorer la qualité des images produites, et serait très facilement intégrable au travail réalisé jusque là.

L'indice de réfraction de l'eau n'est pas constant, mais il est fonction de la longueur d'onde

du rayon en entrée du milieu. Des mesures caractérisant un spectre de réfraction existent pour de l'eau pure, mais les données pour d'autres constituants océaniques sont rares.

Nous utilisons actuellement des données proposées par Premoze et Ashikhmin [PAoo] pour représenter la présence de sable dans l'eau, et modifions le coefficient d'absorption de l'eau en fonction d'une concentration spécifiée par l'utilisateur. Une amélioration intéressante serait de représenter individuellement des grains de sable dans des zones d'intérêt, en utilisant les spectres de diffusion de certains matériaux granuleux présents sur les plages, tels que l'hématite. Cette perspective permettrait de représenter et contrôler l'apparence de plusieurs types de sable et de plages.

Actuellement, nous considérons le volume d'eau comme un milieu homogène, i.e. dont les spectres d'absorption et de diffusion restent les mêmes dans tout le milieu participant. Une perspective majeure à cette contribution serait de représenter le volume d'eau par un milieu participant hétérogène, dont les concentrations varient par exemple en fonction des caractéristiques physiques locales du volume. Un lien évident entre la simulation SPH permettant d'interpoler localement une propriété quelconque, permettrait de faire un pont entre la simulation basée physique d'un volume d'eau et ses propriétés optiques inhérentes.

Bibliographie

[AAT13] Nadir Akinci, Gizem Akinci, and Matthias Teschner. Versatile surface tension and adhesion for sph fluids. volume 32, pages 182 :1–182 :8, 2013. (Cité pages 24 et 28.)

[ACAT13] N. Akinci, J. Cornelis, G. Akinci, and M. Teschner. Coupling elastic solids with smoothed particle hydrodynamics fluids. In *Computer Animation and Virtual Worlds*, volume 24, pages 195–203, 2013. (Cité pages 27 et 102.)

[AIA+12] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Transactions on Graphics (TOG)*, 31(4) :62, 2012. (Cité pages 27 et 28.)

[APKG07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. volume 26, 2007. (Cité page 33.)

[BBC+11] Peter Bucher, Solenthaler Barbara, Nuttapong Chentanez, Matthias Müller, and Markus Gross. Sph based shallow water simulation. In *VRIPhys*, 2011. (Cité pages 30, 92 et 102.)

[BDM+15a] M. Brousset, E. Darles, D. Meneveaux, P. Pierre, and B. Crespin. Un nouveau modèle pour la génération et le contrôle de vagues déferlantes. In *Journées de l'AFIG*, 2015. (Cité pages 70 et 91.)

[BDM+15b] Mathias Brousset, Emmanuelle Darles, Daniel Meneveaux, Pierre Poulin, and Benoît Crespin. A new force model for controllable breaking waves. In *VRIPhys*, 2015. (Cité pages 70 et 91.)

[BDM+16] M. Brousset, E. Darles, D. Meneveaux, P. Poulin, and B. Crespin. Simulation and control of breaking waves using an external force model. volume 57, pages 102–111, 2016. (Cité pages 70 et 91.)

[BG11] Solenthaler Barbara and Markus Gross. Two-scale particle simulation. volume 30, pages 81 :1–81 :8, 2011. (Cité pages 33 et 34.)

[BGH+04] D. Baldwin, Ü. Göktas, W. Hereman, L. Hong, R.S. Martino, and J.C. Miller. Symbolic computation of exact solutions expressible in hyperbolic and elliptic functions for nonlinear PDEs. volume 37, pages 669–705, 2004. (Cité page 50.)

[BGOS06] Adam W. Bargteil, Tolga G. Goktekin, James F. O'Brien, and John A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph.*, 25(1) :19–38, 2006. (Cité page 13.)

[BK15] Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics. In *Symp. on Computer Animation (SCA)*, pages 147–155, 2015. (Cité pages 30, 33 et 102.)

[BNH10] E. Bruneton, F. Neyret, and N. Holzschuch. Real-time realistic ocean lighting using seamless transitions from geometry to BRDF. volume 29, pages 487–496, 2010. (Cité page 11.)

[BOMK90] Robert R. Bidigare, Michael E. Ondrusek, John H. Morrow, and Dale A. Kiefer. In-vivo absorption properties of algal pigments. volume 1302, pages 290–302, 1990. (Cité pages 77 et 84.)

[BP01] Emmanuel Boss and W Scott Pegau. Relationship of light scattering at an angle in the backward direction to the backscattering coefficient. *Applied Optics*, 40(30) :5503–5507, 2001. (Cité pages 76 et 92.)

[BR86] JU Brackbill and HM Ruppel. Flip : A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2) :314–343, 1986. (Cité page 34.)

[Bri08] Robert Bridson. *Fluid Simulation for Computer Graphics*. 2008. (Cité page 14.)

[BT07] M. Becker and M. Teschner. Weakly compressible SPH for free surface flows. In *Symp. on Computer Animation (SCA)*, pages 209–217, 2007. (Cité pages 24, 30, 31 et 102.)

[CIPT14a] J. Cornelis, M. Ihmsen, A. Peer, and M.s Teschner. IISPH-FLIP for incompressible fluids. volume 33, pages 255–262, 2014. (Cité page 34.)

[CIPT14b] Jens Cornelis, Markus Ihmsen, Andreas Peer, and Matthias Teschner. Iisph-flip for incompressible fluids. *Computer Graphics Forum*, 33(2) :255–262, 2014. (Cité page 32.)

[CM11] N. Chentanez and M. Müller. Real-time eulerian water simulation using a restricted tall cell grid. In *ACM TOG, proceedings of SIGGRAPH*, pages 82 :1–82 :10, 2011. (Cité pages 13 et 102.)

[CMK14] Nuttapong Chentanez, Matthias Müller, and Tae-Yong Kim. Coupling 3d eulerian, heightfield and particle methods for interactive simulation of large scale liquid phenomena. In *Symp. on Computer Animation (SCA)*, pages 1–10, 2014. (Cité page 34.)

[DB12] Jonathan Dupuy and Eric Bruneton. Real-time Animation and Rendering of Ocean Whitecaps. In *SIGGRAPH Asia*, page Article No. 15, 2012. (Cité pages 11 et 102.)

[DC96] M. Desbrun and M-P. Cani. Smoothed particles : A new paradigm for animating highly deformable bodies. In *Eurographics*, 1996. (Cité pages 21 et 31.)

[DC99] Mathieu Desbrun and Marie-Paule Cani. Space-Time Adaptive Simulation of Highly Deformable Substances. Research Report RR-3829, INRIA, 1999. (Cité page 33.)

[DCG11] Emmanuelle Darles, Benoit Crespin, and Djamchid Ghazanfarpour. A particle-based method for large-scale breaking wave simulation. *Machine Graphics & Vision International Journal*, 20(1) :3–25, 2011. (Cité pages 46, 47, 48, 70 et 90.)

[DCGG11] Emmanuelle Darles, Benoît Crespin, Djamchid Ghazanfarpour, and Jean-Christophe Gonzato. A survey of ocean simulation and rendering techniques in computer graphics. In *Computer Graphics Forum*, volume 30, pages 43–60. Wiley Online Library, 2011. (Cité page 72.)

[EMF02] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In *ACM TOG, proceedings of SIGGRAPH*, pages 736–744, 2002. (Cité page 13.)

[Eri04] Christer Ericson. *Real-Time Collision Detection*. Boca Raton, FL, USA, 2004. (Cité page 26.)

[FF01] N. Foster and R. Fedkiw. Practical animation of liquids. In *SIGGRAPH*, pages 23–30, 2001. (Cité pages 13 et 14.)

[FJ99] Georges R Fournier and Miroslaw Jonasz. Computer-based underwater imaging analysis. In *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pages 62–70. International Society for Optics and Photonics, 1999. (Cité pages 78 et 83.)

[FM96] N. Foster and D. Metaxas. Realistic animation of liquids. volume 58, pages 471–483, 1996. (Cité pages 2, 12, 13 et 102.)

[FOK05] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. In *ACM TOG, proceedings of SIGGRAPH*, pages 904–909, 2005. (Cité page 13.)

[FR86] A. Fournier and W.-T. Reeves. A simple model of ocean waves. In *SIGGRAPH*, pages 75–84, 1986. (Cité pages 11 et 102.)

[GEF15] Prashant Goswami, André Eliasson, and Pontus Franzén. Implicit Incompressible SPH on the GPU. In *VRIPhys*, 2015. (Cité page 32.)

[Ger09] Franz Gerstner. Theorie der wellen. *Annalen der Physik*, 32(8) :412–445, 1809. (Cité page 8.)

[GP11] Prashant Goswami and Renato Pajarola. Time Adaptive Approximate SPH. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2011)*, 2011. (Cité page 25.)

[Has73] Klaus Hasselmann. Measurements of wind wave growth and swell decay during the joint north sea wave project (jonswap). *Dtsch. Hydrogr. Z.*, 8 :95, 1973. (Cité page 8.)

[HKK07] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on gpus. In *Computer Graphics International*, pages 63–70, 2007. (Cité page 17.)

[HLL+12] Xiaowei He, Ning Liu, Sheng Li, Hongan Wang, and Guoping Wang. Local Poisson SPH For Viscous Incompressible Fluids. *Computer Graphics Forum*, 2012. (Cité page 32.)

[HNC02a] D. Hinsinger, F. Neyret, and M-P. Cani. Interactive animation of ocean waves. In *Symp. on Computer Animation (SCA)*, pages 161–166, 2002. (Cité pages 11 et 102.)

[HNC02b] Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. Interactive animation of ocean waves. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 161–166. ACM, 2002. (Cité page 11.)

[HS13] Christopher Jon Horvath and Barbara Solenthaler. Mass Preserving Multi-Scale SPH. Technical report, Emeryville, CA, 2013. (Cité page 33.)

[IAAT12] M. Ihmsen, N. Akinci, G. Akinci, and M. Teschner. Unified spray, foam and air bubbles for particle-based fluids. In *The Visual Computer*, volume 28, pages 669–677, 2012. (Cité pages 47, 58, 59, 60, 61, 62, 63, 70, 91 et 107.)

[IABT11] M. Ihmsen, N. Akinci, M. Becker, and M. Teschner. A parallel SPH implementation on multi-core CPUs. volume 30, pages 99–112, 2011. (Cité pages 17, 19, 20 et 68.)

[ICS+14] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner. Implicit incompressible SPH. volume 20, pages 426–435, 2014. (Cité page 32.)

[IOS+14] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner. SPH fluids in computer graphics. In *Eurographics*, 2014. (Cité pages 23, 25, 27, 31 et 68.)

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. *Rendering techniques*, 96 :21–30, 1996. (Cité pages 39, 40 et 103.)

[Kaj86] James T. Kajiya. The rendering equation. In *ACM TOG, proceedings of SIGGRAPH*, pages 143–150, 1986. (Cité pages 38, 39, 40 et 103.)

[KdV95] D. J. Korteweg and G. de Vries. On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. volume 39, pages 422–443, 1895. (Cité page 55.)

[Kelo6] Micky Kelager. Lagrangian fluid dynamics using smoothed particle hydrodynamics. 2006. (Cité pages 25 et 26.)

[LC87] W. Lorensen and H. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *SIGGRAPH*, pages 163–169, 1987. (Cité page 37.)

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. In *ACM TOG, proceedings of SIGGRAPH*, pages 457–462, 2004. (Cité page 13.)

[LH10] H. Lee and S. Han. Solving the shallow water equations using 2d sph particles forÂ interactive applications. 26(6) :865–872, 2010. (Cité pages 30 et 102.)

[LHK09] Ho-Young Lee, Jeong-Mo Hong, and Chang-Hun Kim. Interchangeable sph and level set method in multiphase fluids. *The Visual Computer*, 25(5) :713–718, 2009. (Cité page 34.)

[LTKF08] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4) :797–804, 2008. (Cité pages 15 et 32.)

[LvdP02] Anita T. Layton and Michiel van de Panne. A numerically efficient and stable algorithm for animating water waves. 18(1) :41–53, 2002. (Cité page 10.)

[LW93] Eric P Lafortune and Yves D Willems. Bi-directional path tracing. 1993. (Cité page 39.)

[LW96] Eric P Lafortune and Yves D Willems. Rendering participating media with bidirectional path tracing. In *Rendering Techniques 96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 91–100, 1996. (Cité page 39.)

[MCG03] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Symp. on Computer Animation (SCA)*, pages 154–159, 2003. (Cité pages 2, 20, 21, 23, 24, 37 et 103.)

[MFRC13] Pierre-Luc Manteaux, Francois Faure, Stéphane Redon, and Marie-Paule Cani. Exploring the Use of Adaptively Restrained Particles for Graphics Simulations. In *VRIPhys*, 2013. (Cité page 25.)

[MHHR07] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. volume 18, pages 109–118, 2007. (Cité page 32.)

[MLJ+13] Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. Openvdb : An open-source data structure and toolkit for high-resolution volumes. In *ACM TOG, proceedings of SIGGRAPH*, pages 19 :1–19 :1, 2013. (Cité pages 38, 43, 80, 91 et 103.)

[MM13] Miles Macklin and Matthias Müller. Position based fluids. In *ACM TOG, proceedings of SIGGRAPH*, volume 32, pages 104 :1–12, 2013. (Cité pages 2, 30, 32, 33, 36, 37 et 102.)

[MMS04] V. Mihalef, D. Metaxas, and M. Sussman. Animation and control of breaking waves. In *Symp. on Computer Animation (SCA)*, pages 315–324, 2004. (Cité pages 3, 35 et 46.)

[Mob] Curtis Mobley. *Ocean Optics Book*. (Cité pages 73, 77, 79 et 83.)

[Mon92] J.J. Monaghan. Smoothed particle hydrodynamics. volume 30, page 543, 1992. (Cité page 15.)

[Mon05] J.J. Monaghan. Smoothed particle hydrodynamics. volume 68, pages 1703–1759, 2005. (Cité pages 15 et 60.)

[Mor68] A. Morel. Note au sujet des constantes de diffusion de la lumiere pour l'eau et l'eau de mer optiquement pures. *Cahiers Oceanographiques*, (20) :157–162, 1968. (Cité page 74.)

[MPG+16] Thomas Müller, Marios Papas, Markus Gross, Wojciech Jarosz, and Jan NovÃ¡k. Efficient rendering of heterogeneous polydisperse granular media. *ACM TOG, proceedings of SIGGRAPH Asia*, 35(6) :168 :1–168 :14, 2016. (Cité page 83.)

[MPH+15] Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. Multi-scale modeling and rendering of granular materials. *ACM TOG, proceedings of SIGGRAPH)*, 34(4), 2015. (Cité page 83.)

[MSB02] Curtis D Mobley, Lydia K Sundman, and Emmanuel Boss. Phase function effects on oceanic light fields. *Applied optics*, 41(6) :1035–1050, 2002. (Cité page 79.)

[MTPS04] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. In *ACM Transactions On Graphics (TOG)*, volume 23, pages 449–456. ACM, 2004. (Cité page 35.)

[MVW+16a] P-L. Manteaux, U. Vimont, C. Wojtan, D. Rohmer, and M-P. Cani. Space-time sculpting of liquid animation. In *Proceedings of the 9th International Conference on Motion in Games*, pages 61–71, 2016. (Cité page 46.)

[MVW+16b] Pierre-Luc Manteaux, Ulysse Vimont, Chris Wojtan, Damien Rohmer, and Marie-Paule Cani. Space-time sculpting of liquid animation. In *Proceedings of the 9th International Conference on Motion in Games*, pages 61–71, 2016. (Cité page 35.)

[MWN+16] P.-L. Manteaux, C. Wojtan, R. Narain, S. Redon, F. Faure, and M.-P. Cani. Adaptive physically based models in computer graphics. *Computer Graphics Forum*, 2016. (Cité page 33.)

[Nas17] Nasa, 2017. https ://neo.sci.gsfc.nasa.gov/. (Cité page 77.)

[NSB13] M. B. Nielsen, A. Söderström, and R. Bridson. Synthesizing waves from animated height fields. In *ACM TOG, proceedings of SIGGRAPH*, pages 2 :1–9, 2013. (Cité page 2.)

[OH95] James F O'brien and Jessica K Hodgins. Dynamic simulation of splashing fluids. In *Computer Animation'95., Proceedings.*, pages 198–205. IEEE, 1995. (Cité page 12.)

[OK12] Jens Orthmann and Andreas Kolb. Temporal blending for adaptive sph. volume 31, pages 2436–2449, 2012. (Cité pages 33 et 34.)

[PA00] Simon Premoze and Michael Ashikhmin. Rendering natural waters. In *Pacific Conference on Computer Graphics and Applications*, pages 23–434, 2000. (Cité pages 3, 8, 9, 43, 72, 83 et 93.)

[Pea86] Darwyn R. Peachey. Modeling waves and surf. In *SIGGRAPH*, pages 65–74, 1986. (Cité page 11.)

[PF97] Robin M Pope and Edward S Fry. Absorption spectrum (380–700 nm) of pure water. ii. integrating cavity measurements. *Applied optics*, 36(33) :8710–8723, 1997. (Cité page 72.)

[PM64] W. Pierson and L. Moskowitz. A proposed spectral form for fully developed wind seas based on the similarity theory of s. a. kitaigorodskii. volume 69, pages 5181–5190, 1964. (Cité page 8.)

[PTB+03] S. Premoze, T. Tasdizen, J. Bigler, A.E. Lefohn, and R.T. Whitaker. Particle-based simulation of fluids. volume 22, pages 401–410, 2003. (Cité page 32.)

[RDMS10] Rüdiger Röttgers, Roland Doerffer, David McKee, and Wolfgang Schonfëld. Pure water spectral absorption, scattering, and real part of refractive index model. 2010. (Cité page 72.)

[REN+04] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *Symp. on Computer Animation (SCA)*, pages 193–202, 2004. (Cité page 35.)

[RMU14] Rüdiger Röttgers, David McKee, and Christian Utschig. Temperature and salinity correction coefficients for light absorption by water in the visible to infrared spectral region. volume 22, pages 25093–25108, 2014. (Cité pages 73, 74 et 75.)

[RO98] R. Radovitzky and O. Ortiz. Lagrangian finite element analysis of newtonian fluid flows. *International Journal for Numerical Methods in Engineering*, 43(4) :607–619, 1998. (Cité pages 47, 48, 50 et 90.)

[RTWT12] Karthik Raveendran, Nils Thuerey, Chris Wojtan, and Greg Turk. Controlling liquids using meshes. In *Symp. on Computer Animation (SCA)*, pages 255–264, 2012. (Cité pages 35 et 46.)

[RWT11] Karthik Raveendran, Chris Wojtan, and Greg Turk. hybrid smoothed particle hydrodynamics. In *Symp. on Computer Animation (SCA)*, pages 33–42, 2011. (Cité page 34.)

[SCN+16] Marcos Sandim, Douglas Cedrim, Luis Gustavo Nonato, Paulo Pagliosa, and Afonso Paiva. Boundary Detection in Particle-based Fluids. volume 35, pages 215–224, 2016. (Cité page 24.)

[SP09] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible SPH. In *ACM TOG, proceedings of SIGGRAPH*, pages 40 :1–6, 2009. (Cité pages 22, 30, 32 et 102.)

[Sta99] J. Stam. Stable fluids. In *SIGGRAPH*, pages 121–128, 1999. (Cité page 13.)

[Sto47] George G Stokes. On the theory of oscillatory waves. *Trans Cambridge Philos Soc*, 8 :441–473, 1847. (Cité page 8.)

[Tes99] J. Tessendorf. Simulating ocean water. In *SIGGRAPH Course Notes*, 1999. (Cité pages 2, 11 et 102.)

[TG02] S. Thon and D. Ghazanfarpour. Ocean waves synthesis and animation using real world information. volume 26, pages 99–108, 2002. (Cité pages 2 et 11.)

[TKPR06] N. Thürey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. In *Symp. on Computer Animation (SCA)*, pages 7–12, 2006. (Cité pages 35 et 46.)

[TMFSG07] N. Thurey, M. Müller-Fischer, S. Schirm, and M. Gross. Real-time breakingwaves for shallow water simulations. In *Pacific Conference on Computer Graphics and Applications*, pages 39–46, 2007. (Cité page 12.)

[vdLGS09] Wladimir J. van der Laan, Simon Green, and Miguel Sainz. Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, pages 91–98, New York, NY, USA, 2009. (Cité page 37.)

[VG95] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques (Proceedings Fifth Eurographics Workshop on Rendering)*, pages 145–167, 1995. (Cité page 39.)

[Wen10] J. Wenzel. Mitsuba renderer, 2010. http ://www.mitsuba-renderer.org. (Cité pages 41, 43, 64, 68, 83, 86, 91 et 92.)

[WJ10] R. Wiegel and J. Johnson. Elements of wave theory. volume 1, page 2, 2010. (Cité page 48.)

[YT13] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. volume 32, pages 5 :1–5 :12, New York, NY, USA, February 2013. (Cité page 36.)

[YW+09] He Yan, Zhangye Wang, et al. Real-time fluid simulation with adaptive sph. 2009. (Cité pages 33 et 34.)

[ZHH09] Xiaodong Zhang, Lianbo Hu, and Ming-Xia He. Scattering by pure seawater : Effect of salinity. volume 17, pages 5698–5710, 2009. (Cité pages 75 et 92.)

[ZSP08] Yanci Zhang, Barbara Solenthaler, and Renato Pajarola. Adaptive sampling and rendering of fluids on the gpu. In *Eurographics*, pages 137–146, 2008. (Cité pages 33 et 37.)

Liste des figures

Liste des tableaux

Liste des Algorithmes

Titre : Simulation et Rendu de Vagues Déferlantes

Résumé : Depuis plusieurs décennies, la communauté informatique graphique s'intéresse à la simulation physique du mouvement et du rendu des fluides, qui nécessitent d'approcher numériquement des systèmes complexes d'équations aux dérivées partielles, coûteux en temps de calcul. Ces deux domaines trouvent entre autres des applications dans le domaine vidéoludique, qui requiert des performances pouvant offrir des résultats en temps interactif, et dans la simulation d'écoulements réalistes et complexes pour les effets spéciaux, nécessitant des temps de calcul et d'espace mémoire beaucoup plus considérables. Les modèles de la dynamique des fluides permettent de simuler des écoulements complexes, tout en offrant à l'artiste la possibilité d'interagir avec la simulation. Toutefois, contrôler la dynamique et l'apparence des vagues reste difficile. Cette thèse porte d'une part sur le contrôle du mouvement des vagues océaniques dans un contexte d'animation basée sur les équations de Navier-Stokes, et sur leur visualisation réaliste. Nos deux contributions principales sont : (i) Un modèle de forces externes pour contrôler le mouvement des vagues, avec leur hauteur, leur point de déferlement et leur vitesse. Une extension du modèle pour représenter l'interaction entre plusieurs vagues et des vagues tournantes est également proposée. (ii) Une méthodologie pour visualiser les vagues, à l'aide d'une méthode de rendu réaliste, en s'appuyant sur des données optiques des constituants océaniques pour contrôler l'apparence du fluide considéré comme milieu participant. La simulation et le contrôle de la dynamique des vagues sont mis en œuvre dans un simulateur basé sur la méthode SPH (*Smoothed Particle Hydrodynamics*). Afin d'obtenir des performances interactives, nous avons développé un moteur de simulation SPH tirant parti des technologies GPGPU. Pour la visualisation, nous utilisons un moteur de rendu existant pour traiter la visualisation physico-réaliste des milieux participants. Utilisées conjointement, les deux contributions permettent de simuler et contrôler la dynamique d'un front de mer ainsi que son apparence, sur la base de ses paramètres physiques.

Mots-clés : *simulation, contrôle, rendu, synthèse d'images, vagues déferlantes, océan, GPU, GPGPU, SPH*

Title : Breaking Waves Simulation and Rendering

Abstract : Physics-based animation and photorealistic rendering of fluids are two research fields that have been widely addressed by the computer graphics research community. Both have applications in the video-entertainment industry and are used in simulations of natural disasters, which require high computing performance in order to provide interactive time results. This thesis first focuses on simulating breaking waves on modern computer architectures and then on rendering them in the case of oceanic environments. The first part of this thesis deals with physics-based animation of breaking waves, and describes a simple model to generate and control such waves. Current methods only enable to simulate the effects but not the causes of water waves. The implementation of our method takes advantage of GPGPU technologies because of its massively parallel nature, in order to achieve interactive performances. Besides, the method was designed to provide user-control of the physical phenomena, which enables to control in real time all the physical parameters of the generated waves, in order to achieve the desired results. The second part of this thesis deals with the optical properties of water in oceanic environments and describes a model that enables realistic rendering of oceanic scenes.

It provides user-control of oceanic constituants to tune the appearance of oceanic participating media.

Keywords : *Simulation of Breaking Waves, SPH, GPGPU, Breaking Waves Rendering*