Université de Limoges
Faculté des Sciences et Techniques

Année Universitaire 2017 - 2018
version du 15 octobre 2018

Rapport de stage
Simulation de la mer

Auteur

Nicot Fabien

Master Informatique ISICG, semestre 4

Laboratoire Xlim à Poitiers, équipe ASALI-IG

Encadrant : Parenthoën Marc

Responsable : Gilet Guillaume

Table des matières

Introduction au rapport de stage

Dans le cadre de la deuxième année de master ISICG j'ai eu l'opportunité de réaliser un stage de 22 semaines au sein du laboratoire Xlim à Poitiers, équipe ASALI-IG.

Le sujet Initial du stage est : "Intrication de modèles procéduraux et de modèles physiques pour la simulation graphique des états de mer", encadré par Marc Parenthoën.

Mon stage a donc porté initialement sur la représentation et la modélisation de la mer en surface à l'aide de plusieurs modèles de représentations différentes : procédurales pour le comportement global de l'eau et une modélisation physique pour les phénomènes ponctuels nécessitant un modèle générateur car potentiellement imprévisibles.

Les modélisations procédurales de l'eau utilisent le plus souvent une carte de hauteurs afin d'animer la surface de l'eau et d'y appliquer les déplacements à l'aide de fonctions d'ondes sinusoïdales, solutions de l'approximation linéaire des équations de Navier-Stokes. Le principal soucis de cette méthode très efficace pour l'animation des mers calmes est qu'elle ne peut représenter que des déplacements du maillage sur la hauteur, ce qui complique la modélisation de vagues cassantes car l'eau qui se détache de la surface ne peut pas être représenté ainsi ; d'autre part, dans le cas des vagues cassantes l'approximation linéaire des équations n'est plus valable.

Le but du stage est de réussir à modéliser graphiquement la mer de manière à ce que son animation soit assez rapide pour permettre une immersion des utilisateurs en temps interactif, tout en représentant les déferlements dont les effets graphiques nécessitent une modélisation précise de leur comportement.

Je vais diviser ce rapport en différentes parties, je vais d'abord présenter la recherche documentaire que j'ai effectuée au sein de mon stage, ce qui m'a permis de mieux définir dans une deuxième courte partie le sujet de mon stage et de me positionner sur ce qui a déjà était fait, en expliquant les raisons des choix faits concernant le sujet du stage. Enfin dans une dernière grosse partie je parlerais du modèle graphique et du développement que j'ai réalisé.

1 Présentation de la recherche documentaire

1.1 Tableau récapitulatif de l'état de l'art

Tableau récapitulatif des papiers lus dans le cadre du stage

Le tableau 1 organise les papiers sur l'animation graphique de la mer que j'ai pu lire avec un tri sur le type d'animation utilisé.

Les différents phénomènes de la mer en surface qu' un marin ou un océanographe peuvent observer visuellement sont décrits dans [PGT03]. Il s'agit principalement des groupes de vagues, des déferlements et de leurs interactions avec les vents, la bathymétrie, les courants et les objets fixes ou flottants. Ce papier se place du point de vue d'un océanographe, et de la manière dont les personnes qui connaissent bien la mer décrivent le mouvement de celui afin de mieux prévoir le comportement de celui-ci.

On trouve dans la littérature trois approches principales pour la modélisation des vagues océaniques : la modélisation procédurale, la modélisation physique et la modélisation phénoménologique. Le tableau 1 organise les papiers selon ces trois catégories d'animation de la mer.

Auteurs	Procédurale		Phénoménologique	Physique	
	Spatiale	Fréquentiel		Eulérien	Lagrangien
[PGT03]			X		
[GM84]			X		
[Dau88]			X		
[MT15]					X
[GM77]					X
[Luc77]					X
[BDM+16]					X
[Bro17]					X
[JW17]			X		
[BKKW17]					X
[FMB+17]					X
[DCGG11]	X	X	X	X	X
[NSB13]	X				
[Pea86]	X				
[FR86]			X		
[T+01]		X			
[OH95]				X	
[WZC+06]					X
[MMS04]				X	
[TDG00]	X	X			
[Bri15]	X			X	X
[BS10]				X	X
[BMF07]	X			X	X
[SS17]	X				X
[BCPR10]			X		
[Phi85]		X			
[PJT04]			X		
[PMT15]			X		
[ADAT13]	X				
[JG01]		X			
[IAAT12]				X	X
[MMCK14]					X
[Kam08]					
[CH17]					
[TMFSG07]	X				
[CI07]	X				

TABLE 1 – Récapitulatif et classement des papiers de recherche lu pendant le stage

1.2 Modélisation procédurale

La modélisation procédurale décrit la surface de mer comme une somme d'ondes sinusoïdales. En effet, l'approximation linéaire eulérienne des équations décrivant la surface de mer (Navier-Stokes) donne le modèle des houles de Airy, qui sont des ondes sinusoïdales, linéairement superposables pour des hauteurs de vagues très petites. Le plus souvent ce type de modélisation est utilisé afin de réaliser le comportement global d'une grande surface de la mer, en admettant que ça n'est pas une modélisation réaliste (en dehors de la description des mer très calmes) pour les mers un petit peu moins calmes, les abstractions réalisées sont un bon compromis entre performances, simplicité d'implémentation et contrôle des vagues. La modélisation procédurale est la méthode la plus rapide est la plus intuitive pour modéliser le mouvement de l'eau, mais cette méthode montre vite ses limites lorsque l'état de mer devient plus dynamique.

La principale différence entre toutes les recherches qui ont été effectuées avec cette approche procédurale est sur le choix des ondes sinusoïdales et la manière d'en faire la somme qui seront utilisées par la méthode.

Dans la manière de générer le mouvement de la mer de manière procédurale, il existe 2 grandes catégories de modélisation, qui utilisent des méthodes très différentes avec des qualités et des défauts complémentaires. Il y a la modélisation spatiale qui utilise des sommes de fonctions sinusoïdales définies par l'environnement de la simulation, par exemple la profondeur de l'eau, et par différentes variables, tandis que la modélisation fréquentielle rationalise les fréquences et hauteurs des vagues telles que décrites par les océanographes pour définir les ondes sinusoïdales de la simulation.

Les deux modélisations sont complémentaires, la modélisation spatiale permet un meilleur contrôle par l'utilisateur de chaque onde, tandis que la modélisation fréquentielle est plus globale, mais intègre les observations des océanographes et semble plus réaliste. Dans la prochaine partie nous verrons plus en détail les spécificités de ces modélisations.

Il y a plusieurs problèmes que l'on va rencontrer avec la modélisation procédurale spatiale et fréquentielle. Tout d'abord le rendu est basé sur des observations rapide de mouvement de l'eau, manque de réalisme dans l'animation dans le cas où la mer est agité, car une grande partie des phénomènes observés dans une mer ne peuvent être modélisés par des ondes sinusoïdales : comme les vagues cassantes, la mousse et les jets d'eau. Les différentes recherches portent aussi sur les manières détournées pour résoudre ces problèmes.

1.2.1 Modélisation spatiale

Comme dit plus haut, la modélisation spatiale utilise une somme d'ondes sinusoïdales afin de représenter la surface. Les deux papiers les plus anciens sur cette méthode que j'ai lu sont celui de Peachey [Pea86] et Fournier [FR86]. Ces papiers datent de 1986, ils sont l'un des premiers papiers traitant de ce sujet (après Max en 1981 [Max81] et Schachter en 1980).

Max [Max81] a défini la modélisation des vagues grâce à une combinaison de fonction d'ondes progressives

$$h(x,t) = Acos(kx - \omega t + \varphi)$$

Avec ψ la hauteur de l'eau en $\mathbf{x}$: position sur le plan horizontal, à l'instant t, A l'amplitude de l'onde, $\mathbf{k}$ le vecteur d'onde (le nombre d'onde et une direction de propagation, inversement proportionnel à la longueur d'onde $\lambda = \frac{2\pi}{k}$), ω est la pulsation de l'onde (inversement proportionnel à la période $P = \frac{2\pi}{\omega}$ et φ le déphasage de l'onde (systématiquement choisi nul dans le papier de Max [Max81])

Max [Max81]) a ensuite rajouté une somme des Nw vagues et de calculer la hauteur en fonction de la hauteur au repos de l'eau.

$$h(x,z,t) = -y_0 + \sum_{i=1}^{N_w} A_i cos(k_{i_x} x + k_{i_z} z - \omega_i t)$$

$$\omega_i = \sqrt{gk}$$

Voici comment calculer le vecteur d'onde avec g la constante de gravité et L la longueur d'onde

Voici les équations les plus basiques et génériques pour générer des ondes, et dans notre cas des vagues. Cette formule permet de lier les paramètres de l'onde et la hauteur de l'eau un instant donné, pour une onde donnée et une position donnée.

Peachey a pris en compte la profondeur de l'eau afin de définir les paramètres de l'onde de la vague, grâce au vecteur d'onde

$$\omega_i = \sqrt{gktanh(hk)}$$

avec d correspondant à la profondeur de l'eau.

Le dernier papier que j'ai lu concernant ce type de modélisation est hybride entre la modélisation spatiale et fréquentielle, je vais ne parler que de la modélisation spatiale, je développerai le reste dans la prochaine partie.

Pour des problèmes d'échelle et de variabilité incompatible avec les ondes sinusoïdales, la représentation graphique des détails de la surface de mer à haute fréquence spatiale passe par une approche statistique du rendu.

Thon et Al [TDG00] modélise les petits détails de sa simulation à l'aide d'un bruit qui est implémenté dans la méthode comme la vague la plus petite de la simulation. Il s'agit des turbulences de la surface. L'impact de ce bruit sur le rendu dépend des paramètres utilisés pour le noise et de la taille qu'il représente sur la surface d'eau.

$$\sum_{i=0}^{n_0-1} \frac{Noise\,(P.m^i)}{m^i}$$

Afin de modéliser la carte de hauteur que nous pouvons utiliser 2 méthodes le bump mapping et le normal mapping. Le premier désigne le déplacement des sommets en fonction de la carte de hauteur grâce à son mappage. Tandis que la normal map permet de modéliser le rendu de la lumière sur la surface des triangles et pas sur les sommets. Ca permet de diminuer la quantité de sommets nécessaire afin de réaliser un rendu convenable de la surface.

Avec la normal map, le déplacement de la surface n'est pas représenté par un déplacement des sommets mais modélise la manière dont la lumière est censée réfléchir le déplacement. Pour les petits effets de rendus cette méthode permet d'avoir des résultats convenables avec un temps de calcul beaucoup plus rapide que pour un déplacement de sommet. Un des problèmes de la normal map est qu'elle doit être précalculée pour le rendu.

1.2.2 Modélisation fréquentielle

On ne peut pas parler de modélisation fréquentielle sans parler de la transformée de Fourier, elle permet de transformer une fonction non périodique en une fonction échantillonnée dans le domaine fréquentiel. Grâce à la transformée de Fourier inverse on peut récupérer les fréquences échantillonnées de la fonction non périodique.

La modélisation fréquentielle utilise la transformée de Fourier inverse afin d'obtenir les fréquences et l'amplitude des différentes ondes qui composent une surface d'eau. Les fréquences de ces ondes sont échantillonné à partir d'images, qui peuvent être tirées de la mer ou bien générer aléatoirement avec un filtrage de certaines zones pour en obtenir des vagues.

Qu'importe la méthode utilisée, cette modélisation permet de générer une animation de la mer crédible et automatique. Cependant l'un des plus gros défauts de la méthode est qu'il est difficile de contrôler son comportement car il est difficile de contrôler sa génération, aussi cette méthode ne peut pas beaucoup évoluer, c'est-à-dire que si la transformée de Fourier a été effectuée sur une prise de vues la simulation ne pourra représenter que l'état de la mer à cet endroit à un instant précis.

Un des papiers pionnier de cette approche est celui de Tessendorf [T$^+$01] qui était utilisé pendant longtemps dans des films tel que Titanic et Waterworld.

Les données utilisées pour la transformée de Fourier viennent d'une prise de vues de l'océan à l'aide d'une caméra panchromatique nommée AROSS. Grâce à cela et à une transformée de Fourier inverse, la méthode récupère les fonctions caractéristiques des fonctions périodiques. Ces fonctions peuvent être utilisées comme pour les modélisations spatiales car la liste des fonctions périodiques obtenues sont dans le domaine spatial.

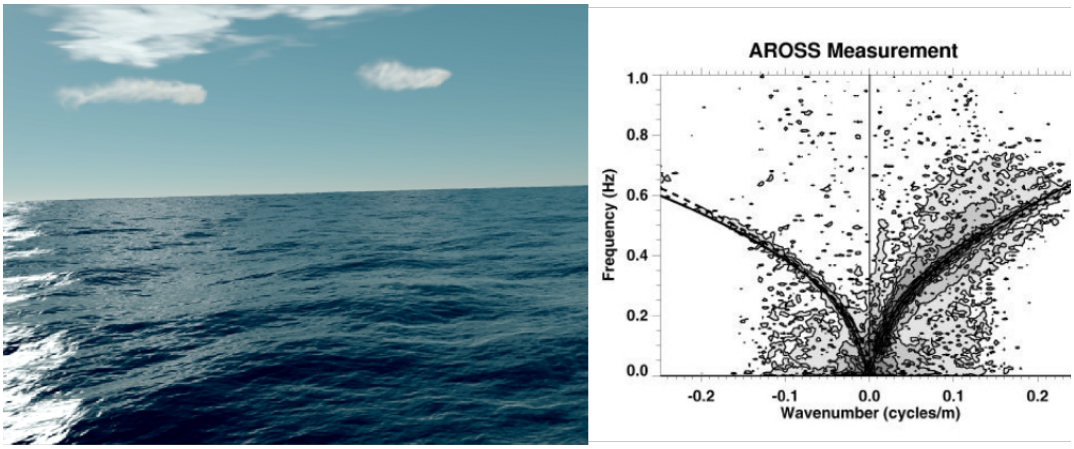

FIGURE 1 – Tessendorf [T+01] Mesure de la surface de la mer à droite et animation de la mer à gauche

Dans le cas de ce papier [T+01], l'auteur utilise la houle de Gerstner que l'on a vue plus tôt afin de modéliser les vagues. L'article est très précis dans les différents phénomènes modélisés comme les vagues abruptes, les perturbations de la surface par d'autres objets (sillages...) mais aussi sur le rendu de l'eau elle-même.

Pour remédier à la difficulté de simuler des états de mer plus agités, Thon et al [TDG00] comporte une partie de modélisation spatiale pour ce qui est des grandes vagues. Pour le comportement global de la surface de l'eau, seules les courtes vagues sont modélisées à l'aide d'une modélisation spectrale.

Les données utilisées en entrée de la modélisation sont générées aléatoirement, on obtient une texture de bruit en entrée de la transformée de Fourier. Le papier utilise un filtre celui du spectre Pierson-Moskowitz qui décrit la distribution de l'énergie ainsi que de la fréquence des vagues dans un océan en équilibre entre les vagues et le vent (on dit que l'état de mer est complètement développé). Il reprend le travail de Mastin 1987 [MWM87] qui utilise déjà cette même méthode cependant, pour des raisons exclusivement graphiques de subjectivité du rendu, il ajoute une partie de sélection des vagues avec une haute fréquence car les plus petites fréquences sont modélisées de manière spatiale. La superposition de trochoïde correspond à la transformée de Fourier inverse.

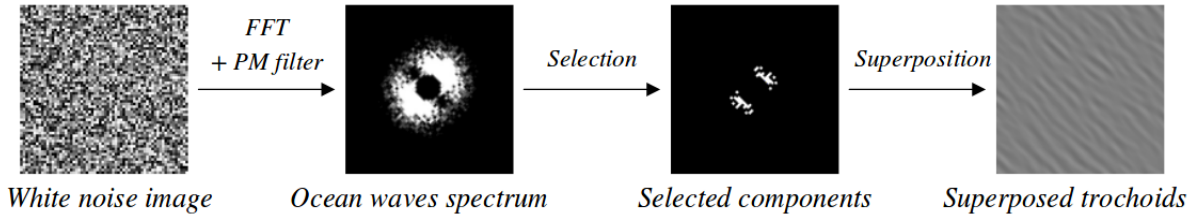

FIGURE 2 – Thon et al [TDG00] Différentes étapes de la méthode

Le papier le plus récent qui utilise la modélisation fréquentielle pour réaliser l'animation de la surface de l'eau est le papier Nielsen [NSB13] qui utilise la modélisation fréquentielle à laquelle il ajoute un déplacement horizontal des vagues. Les paramètres utilisés sont générés grâce à la méthode de Bridson 2008 [Bri15] afin de générer que des fréquences basses pour les déplacements horizontaux.

Réaliser un déplacement horizontal peut créer un problème d'intersection au niveau des vagues. Pour résoudre ce problème, deux méthodes ont été testées, la première est de tester le jacobien en un point qui doit être positif, ce qui signifie que l'orientation est conservée, cette méthode est assez simple à mettre en place mais elle a tendance à ne pas être très robuste et que la surface de l'eau est souvent trop aplatie. La deuxième méthode consiste à réaliser une optimisation explicite des déplacements en mesurant la distance de l'intersection est de vérifier une suite des conditions pour chaque cas particulier qui peut être rencontré dans une détection de collisions (ces algorithmes ne permettent pas à l'heure actuelle une animation en temps interactif).

Ainsi, nous avons vu les modèles procéduraux qui décrivent la mer sont très efficaces pour les états de mer homogènes calmes à modérés, difficiles à utiliser pour un plan d'eau hétérogène et un état de mer dynamique par exemple avec des déferlements. Nous allons voir dans la prochaine section les approches basés physique.

1.3 Modélisation physique

Les approches basées physique pour l'animation des états de mer s'appuient sur la modélisation physique de l'eau. Les équations physiques utilisées sont celles de Navier-Stokes.

C'est grâce aux travaux de d'Henri Navier en 1823 et de George Gabriel Stokes en 1843. Ils apportent une amélioration aux équations d'Euler décrivant la mécanique des fluides. Navier ajoute la notion de viscosité aux équations tandis que Stokes ajoute l'équation de conservation du mouvement.

Les équations décrivant la mécanique des fluides peuvent être utilisées dans de nombreux domaines et dans de nombreux types de fluides. En effet ces équations fonctionnent aussi

pour les gaz. Elles sont donc utilisées aussi pour analyser l'aérodynamisme, mais aussi en astrophysique et en géophysique.

La forme simple des équations de Navier-Stokes est :

$$\frac{\delta \vec{u}}{\delta t} = -(\vec{u}.\Delta)\vec{u} + \mu\Delta^2\vec{U} - \frac{\vec{\Delta}p}{\rho} + \vec{f}$$

Les variables sont :
- $\vec{u}$: la vélocité
- μ : la viscosité
- p : la pression
- ρ : la densité
- $\vec{f}$: les forces externes

Cette formule permet de calculer le changement de vélocité de l'eau, pour cela on peut découper la formule en 4 parties :
- L'advection l'inverse des vitesse entrantes dans la zone
- La viscosité du liquide proportionnel à la vitesse du liquide
- La conservation de la masse en soustrayant la variation de la pression
- Les forces de l'utilisateur comme la gravité

Cette équation est continue et n'admet pas de solution analytique. Pour la résoudre numériquement et l'utiliser dans une simulation informatique, on peut la discrétiser. Pour cela il existe deux grandes familles de méthode : la méthode eulérienne et la méthode lagrangienne. La modélisation eulérienne observe les évolutions du fluide à travers une grille immobile et l'on utilise les dérivées partielles pour exprimer les flux à travers les éléments de la grille, tandis que la modélisation lagrangienne accompagne une particule de fluide et l'on utilise les dérivées totales pour décrire les évolutions de cet élément de fluide au cours du temps.

Pour mieux comprendre voici une métaphore de Bridson [Bri15] "One way to think of the viewpoint is in doing a weather report. In the Lagrangian viewpoint you're in a balloon floating along with the wind, measuring the pressure and temperature and humidity, etc ..., of the air that's flowing alongside you. In the Eulerian viewpoint you're stuck on the ground, measuring the pressure and temperature and humidity, etc ..., of the air that's flowing past. Both measurements can create a graph of how conditions are changing, but the graphs can be completely different they are measuring the rate of change in fundamentally different ways."

1.3.1 Modélisation eulérienne

Comme dit plus haut la modélisation eulérienne résout les équations grâce à des observateurs fixes. Dans la plupart des cas il s'agit d'une grille qui possède plusieurs informations :

— P : Pression de la cellule, stockée dans le centre de la cellule
— U : Vitesse de l'eau de la cellule aux bords (vitesse entrante et sortante)

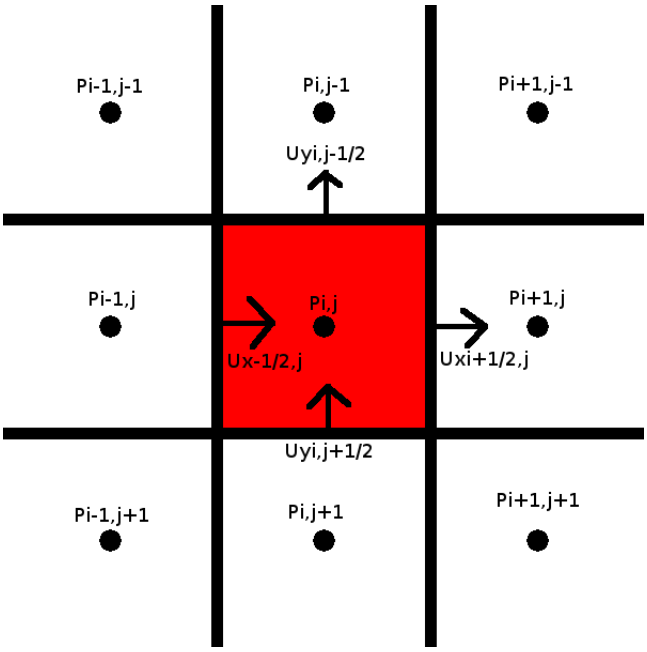

FIGURE 3 – Grille 2D de la modélisation Eulérienne

[OH95] Simulation des effets d'éclaboussures de l'eau en 1995 par O'Brien et al. Pour simuler le mouvement global de l'eau, il utilise des colonnes de hauteur ainsi qu'une surface. Les éclaboussures sont modélisées l'aide de particules.

Modélisation de l'eau en 3 parties :
— L'eau en elle même est représentée par des colonnes
— La surface de l'eau est représentée par maillage de carré
— Les effets discontinus de l'eau (éclaboussures) sont représentés par un système de particules

Les méthodes eulérienne sont très efficaces pour modéliser les phénomènes à grandes échelles de l'eau, par exemple, la propagation d'onde, l'interaction etc ... Cependant pour les effets plus petits comme on peut rencontrer avec des vague cassante la méthode devient beaucoup trop gourmande en ressource car la grille de simulation doit être très précise. c'est pour cela que la modélisation lagrangienne est plus utilisée pour modéliser ces phénomènes.

1.3.2 Modélisation lagrangienne

Contrairement à la méthode eulérienne, la méthode eulérienne modélise le mouvement de l'eau à l'aide de particules qui représentent l'eau modélisée.

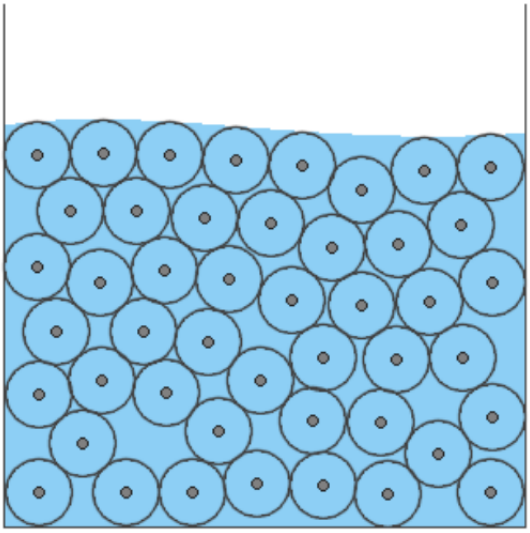

FIGURE 4 – Modélisation lagrangienne en 2D

Comme on peut voir sur cette illustration, le volume de l'eau est représenté par des particules.

Introduit en 1977 par R. A. Gingold et J. J. Monaghan [GM77], le Smoothed Particle Hydrodynamics (SPH) est une méthode permettant de représenter physiquement des fluides. Pour cela les équations de mouvements du système utilisent les équations Newtoniennes appliquées aux particules. Cette méthode a d'abord été utilisée en astrophysique afin de modéliser l'organisation des corps célestes dans l'espace et même l'expansion de l'univers à partir du big bang, mais elle peut bien entendu s'appliquer aux autres fluides.

Les problèmes mathématiques et les solutions trouvées pour résoudre le problème de distribution des particules par R. A. Gingold et J. J. Monaghan sont la distribution de densité, le champ gravitationnel, la distribution générale des particules, le champ magnétique et le courant.

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

Il s'agit de l'équation principale qui régit résout les équations de Navier-Stokes. En effet A_i représente une force d'interaction entre particules (viscosité, tension et pression) pour la particule i. Pour cela on réalise une somme pour les j particule du système de leur force A_j du pas précédent que l'on multiplie par une fonction d'atténuation W.

Le plus souvent les implémentations de cette méthode optimisent le nombre j de particules à calculer en ne prenant que les particules adjacentes. La fonction d'atténuation peut aussi varier en fonction de l'implémentation, mais elle peut ressembler à un kernel

gradient. Un kernel gradient est une grille qui pour chaque case associe une valeur correspondant à une particule voisine, les coefficients sont généré pour diminuer en fonction de la distance avec le centre.

Le deuxième papier que j'ai lu sur le sujet des SPH est la thèse de Mathias Brousset [Bro17]. Il y a deux grandes parties dans cette thèse, le contrôle des fluides à l'aide de forces externes et le rendu de l'eau à l'aide d'un SPH. L'ajout principal du papier de la thèse de Mathias Brousset est sur le contrôle des fluides. En effet l'utilisateur peut dessiner la forme des vagues, des forces externes vont donc être créées pour animer cette vague.

Grâce à cette thèse j'ai remarqué que l'un des champs de recherches sur les SPH le plus actif ces derniers temps est pour contrer une des faiblesses des modélisations physiques, le manque de contrôle des simulations qui sont réalisées. Par exemple comment créer une vague qui ne serait pas liée à un événement physique.

Ainsi ... eulérienne ... lagrangienne ... grandes étendues d'eau ... nous voyons comment les méthodes procédurales ont été combinées aux approches basées physique.

1.4 Interaction entre méthodes procédurales et physiques

Pour les hybridations entre les modélisations procédurales et physiques, il semble qu'il n'y ai qu'un seul papier tiré du film d'animation Vianna [SS17]. Un autre papier [CI07] (publié à la suite du film Surf's Up) pourrait avoir implémenté une méthode physique pour simuler les éclaboussures des vagues cassantes, cependant le papier ne présente pas leur méthode de manière suffisamment détaillée pour affirmer que l'animation des particules est basée physique.

FIGURE 5 – Image du film d'animation Surf's Up [CI07]

Le papier ayant implémenté une réelle intrication entre physique et procédural est le papier tiré du film Vianna [SS17]. Ils ont choisi l'intrication des méthodes car après plusieurs tests ils se sont rendu compte que pour simuler une mer de manière suffisamment précise et convaincante il leur faudrait plusieurs milliards de particules à modéliser. Même pour le budget de Disney, le temps de calcul de cette méthode n'est pas envisageable alors même que le rendu n'est pas réalisé. Grâce à l'intrication des méthodes ils passent plus qu'à quelques millions de particules.

La partie procédurale est réalisée à l'aide de méthodes que l'on a déjà vues précédemment comme celle de Tessendorf [T$^+$01]. La modélisation physique est réalisée à l'aide d'un SPH. La principale avancée technique présentée dans ce papier est le Flux Animated Bundary (FAB). Le FAB est la méthode qui gère l'intrication entre les deux modélisations. Il utilise une zone dans laquelle on souhaite modéliser des détails de la surface d'eau comme les vagues cassantes. A chaque pas de temps une quantité de particules est créés autour de la boîte de simulation, ces particules de SPH sont ajouté au système de simulation. Ensuite la boîte de simulation est déplacée pour suivre l'événement à modéliser. Il supprime ensuite les particules en dehors de la boîte de simulation.

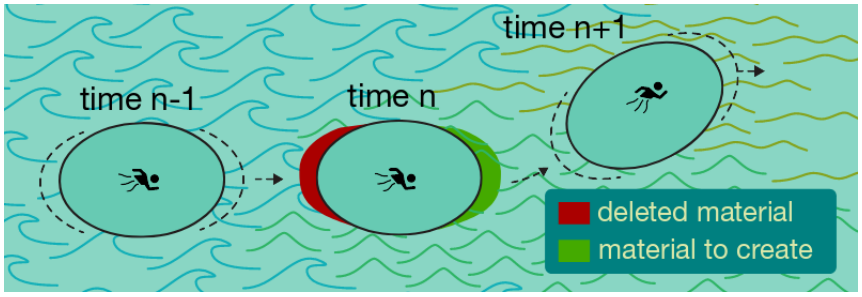

FIGURE 6 – Explication du FAB [SS17]

Conclusion intermédiaire

La modélisation procédurale et la modélisation physique sont les modèles les plus répandus. Ils sont aussi très antagonistes, la modélisation procédurale est rapide avec une animation correcte des surfaces d'eau calme, tant dis que la simulation physique possède meilleur un réalisme dans tous les cas de figure (surtout quand l'eau est très agitée) mais souffre d'un temps de calcul très important.

La combinaison des deux de méthodes permet d'obtenir des bons résultats d'animation tout en baissant le temps de calcul [SS17]. Cependant même si l'on diminue le nombre d'entités des modélisations physiques il est encore impossible d'obtenir un rendu interactif d'une mer. De plus il est aussi très compliqué de faire interagir les deux méthodes.

C'est pour pallier ces problèmes que la modélisation phénoménologique a était créée. Elle est basée sur la représentation et la modélisation des différents effets réels de la mer.

1.5 Modélisation phénoménologique

La modélisation procédurale de la surface de l'eau génère des plans d'eau homogènes. En effet, une onde sinusoïdale est constante sur toute la surface, elle ne varie pas quelle que soit sa position et recouvre toute la surface de l'eau, en observant une surface d'eau on pourrait croire que c'est le cas dans une zone restreinte que les vagues sont régulières et qu'elles correspondent à une sorte d'onde générée par le vent. C'est une vision erronée du mouvement de l'eau, les effets sont plus complexes et plus localisés. Un plan d'eau est très hétérogène et cette complexité n'est pas abordé à l'échelle du plan d'eau par les modèles procéduraux ou physiques actuels.

Prenons l'exemple de la houle, qui est le phénomène le plus visible et qui illustre le mieux le phénomène. Les marins et les océanographes ont remarqué que la houle est organisée en groupes de vagues, avançant au large à une vitesse deux fois moins rapide que les vagues qui constituent chaque groupe. Les vagues naissent à l'arrière du groupe, grossissent en progressant vers le centre du groupe où elles atteignent leur amplitude maximale, puis continuent vers l'avant du groupe en s'atténuant... Un groupe de vagues est limitée aussi en largeur, qui est définie par la longueur de crête de la vague lorsqu'elle passe au centre du groupe. La zone où le groupe présente des vagues d'amplitude non nulle s'appelle l'enveloppe du groupe. Une étude océanographique des groupes de vagues peut être trouvé dans [BCH+12]

Dès 1986, on trouve un modèle graphique de vagues qui s'appuie sur les groupes de vagues [FR86]. Leur enveloppe est caractérisée par une fonction en forme de lingot.

Fournier [FR86] a utilisé la théorie des ondes de Gerstner, c'est à dire une approximation linéaire lagrangienne des équations de Navier-Stokes, et des groupes de vagues pour modéliser la surface de l'eau. La théorie de Gerstner définit le comportement des particules que se trouve sur cette onde, il dit que la trajectoire d'une particule parcourt un cercle durant toute la simulation, les particules sont donc représentées par une fonction sinus et une fonction cosinus pour les x et les y

$$x = x_0 + r * sin(\kappa x_0 - \omega t)$$

$$y = y_0 - r * cos(\kappa y_0 - \omega t)$$

avec κ le nombre d'ondes ω la pulsation de l'onde et r le rayon du cercle de l'onde.

Les particules étant trop coûteuses à mettre en place il décide de passer par un champ de hauteur pour modéliser la surface, il transforma donc ces fonctions pour obtenir une hauteur unique en fonction de la position et du temps à partir de la théorie de Gerstner.

Grâce à différents paramètres issus de l'océanographie, il a pu modifier la trajectoire circulaire des particules de la vague définie par la théorie de Gerstner pour faire des trajectoires ovales et créer des vagues plus abruptes en respectant les modèles océanographiques tenant compte des faibles profondeurs. Grâce à cela il a pu essayer de modéliser des vagues cassantes, pour cela il lui fallut détecter quand est-ce que la vague casse, et comment la modéliser, mais il ne gérait pas explicitement les boucles pouvant apparaître.

Les conditions pour qu'une vague soit cassante ne sont que vitesse de la particule soit plus grande que la vitesse de phase (vitesse de la crête) ou que la pente soit trop élevée. Pour la pente, un angle de 60° avec l'horizontal semble être la bonne limite détecter une vague cassante.

En océanographie, ce sont plutôt les ondelettes de Morlet qui sont utilisées. Les ondelettes ont été introduites en 1984 par Grossmann et Morlet [GM84]. Elle est représentée à l'aide d'une fonction fréquentielle multipliée par une variable d'atténuation qui varie en fonction de la distance depuis le centre. On peut aussi citer le papier de Daubechies en 1988 qui parle aussi des ondelettes.

La modélisation phénoménologique essaie de corriger les faiblesses de réalisme de la modélisation procédurale, en essayant de faire ressembler le plus possible les surfaces d'eau à ce que l'on peut observer. Le but de la modélisation phénoménologique est de rechercher comment sont formé les différents phénomènes modélisant une surface d'eau et comment représenter leurs effets visuels.

On peut se demander comment peut-on modéliser une surface à l'aide uniquement de phénomènes localisés sur la surface de l'eau. Il suffit d'augmenter le nombre de ces phénomènes, c'est la quantité de ces effets avec différentes échelles qui donnent un aspect régulier à l'eau avec un côté bruité de ces effets.

Après le travail pionnier de Fournier qui introduit les groupes de vagues, le premier papier à utiliser cette technique pour les groupes de vague et les déferlements est Parenthoen 2004 [PJT04]. Le but premier de ce papier est de réaliser une simulation en temps interactif, de très grandes étendues d'océan dans un but d'immersion d'un utilisateur dans un système de réalité virtuel, tout en respectant au mieux les connaissances océanographiques.

Pour ce faire la méthode utilise un découpage en plusieurs agents qui auront des impacts différents sur la surface de l'eau. Tout d'abord l'agent le plus important est le groupe qui est représenté à l'aide de la réification d'une ondelette de Morlet en un groupe de vagues.

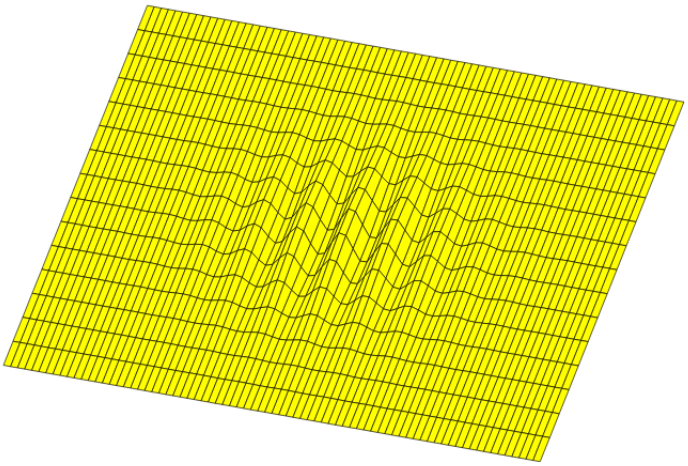

FIGURE 7 – Exemple de groupe de vagues

Il s'agit de l'agent qui va réaliser le mouvement et l'animation des vagues en fonction des donnés et des interactions envoyées par les autres agents. Les paramètres de ces agents sont les mêmes que ceux définis pour la houle :

- Une direction
- Une période
- Une longueur d'onde
- Une hauteur
- Une cambrure
- Un nombre de vague
- Une étendue des crêtes

Il y a quatre autres types d'agents, les vagues cassantes, le vent, le courant et la profondeur. Les vagues cassantes et les groupes de vague réalisent l'animation de la mer mais interagissent avec les 4 autres agents, par exemple le vent peut créer et augmenter l'énergie du groupe de vagues. C'est grâce à une modélisation précise des interactions entre chaque agent que l'on peut obtenir des résultats réalistes d'un point de vue océanographique.

y <- x	Group	Breaking	Winds	Current	Depth
Group	action	wave action	creation energy	wave action transport	wave energy
Breaking	creation action	action	action	transport	indirect via groups
Winds Current Depth	Descriptive agent, no reaction				

TABLE 2 – Interaction entre les différents agents

Ce tableau décrit l'ensemble des interactions où les éléments de chaque colonne impactent ceux de chaque ligne. On peut aussi voir dans la dernière ligne du tableau que les agents de vent, de courant et de profondeur ne sont pas utilisés pour la modélisation mais que pour décrire les interactions sur les groupes et les vagues cassantes.

Le papier suivant que j'ai pu lire sur la modélisation phénoménologique est le papier de Zakaria Belemaalem 2010 [BCPR10]. Il a repris les travaux précédents sur les groupes de vagues et les ondelettes de Morlet.

$$\eta(\overrightarrow{x}, t) = a.exp\left(\frac{-(\overrightarrow{x} - \overrightarrow{x}_c(t))^2}{2\varrho^2}\right).sin\left(\overrightarrow{k}.(\overrightarrow{x} - \overrightarrow{x}_0) + \theta_0 - \omega t\right)$$

Cette formule permet de connaître la hauteur pour une ondelette de Morlet pour un groupe en fonction de la position x et de t la variable de temps. La première partie de la formule sert à l'atténuation en fonction de la distance du centre et la deuxième partie de la formule est la fonction périodique du groupe de vague. xc est la position du centre du groupe de vague, k est le vecteur d'onde, $\overrightarrow{x}_0$ la position initiale, θ_0 la phase du groupe de vague, ω_t la pulsation, ϱ est l'enveloppe étendue et a l'amplitude du groupe de vague.

Pour modéliser la surface de l'eau, il faut pouvoir combiner plusieurs groupes de vagues, pour cela il propose de réaliser un mappage d'un groupe de vagues placé à l'aide d'une distribution de poisson. Pour pouvoir combiner tous les groupes de vagues, il faut juste faire la somme du déplacement sur la hauteur de tous les groupes de vagues.

Il y a 3 variables afin de définir les groupes de vagues qui sont déterminés grâce à des distributions aléatoires :

- La position x du groupe est déterminée grâce à une distribution de Rayleigh
- L'amplitude a est déterminée grâce à une distribution uniforme
- La phase θ_0 est déterminée grâce à une distribution uniforme

Il a aussi créé une méthode afin de détecter, de distribuer et de créer des vagues cassantes sur une surface d'eau en adaptant la méthode de Phillips 98 [Phi85]. La distribution permet de déterminer la moyenne de longueur de vagues cassantes par unité de surface. Trois distributions sont possibles.

Les deux derniers papiers que j'ai pu lire sur la modélisation phénoménologique sont ceux de Jeschke 2017 et 2018 [JW17].

Ils utilisent la notion de paquet d'ondes que l'on a pu voir précédemment pour modéliser la dispersion des vagues suite à une interaction entre un objet et la surface de l'eau. Selon eux, par rapport aux modèles physiques, ils ont réussi à améliorer les différents détails visuels, avec un meilleur coût de calcul et des nouveaux paramètres de contrôle.

La théorie des ondes d'Airy décrit la surface des vagues comme une fonction de hauteur qui varie en fonction du temps. Aussi pour guider les paquets d'ondes ils utilisent des particules qui se déplacent en suivant la direction de leur émission. Il suffit de deux particules, un de chaque côté à l'avant du groupe de vagues. Ces particules servent aussi à interpoler le paquet d'ondes, en effet elles ne servent pas juste à diriger le paquet d'ondes mais aussi à calculer le paquet d'ondes en lui-même.

Les deux particules de chaque paquet d'ondes ont une direction et une vitesse indépendante.

- Subdivision géométrique lorsque 2 particules d'un groupe sont trop éloignés l'une de l'autre ou si leurs directions ont une différence d'angles supérieurs 18°. La subdivision géométrique créée une nouvelle particule entre les deux existante remplacent une particule dans chaque sous-paquets.
- Subdivision dispersive : quand un paquet se disperse on crée deux nouveaux paquets avec la même position et les mêmes paramètres et distribue le spectre des vagues.

Ce papier modélise uniquement les effets des interactions entre les objets et la surface de l'eau. Pour réaliser la propagation de l'onde ils ont décidé de la découper en 4 effets différents qui peuvent être rencontrés :

- Réflection : lors d'une collision entre un paquet d'ondes et un objet, le paquet d'ondes réfléchies en conservant une partie de son énergie.
- Dispersion : les paquets de vagues se propagent avec une certaines vitesse et peuvent rencontrer d'autres vagues qui feront changer leurs comportements.
- Réfraction : changement de la direction d'un paquet de vagues, à cause, par exemple, d'un changement de la hauteur du fond.
- Diffraction : lorsqu'un paquet de vagues longe un objet, certaines vagues en contact avec l'objet (notamment lorsqu'il quitte l'objet) voient sa direction changée afin de continuer à longer l'objet.

La propagation de l'énergie est réalisée par les particules qui guident les paquets d'ondes. L'énergie des paquets de vagues est calculée grâce à 3 paramètres :

- La viscosité diminue l'amplitude du paquet de vagues, l'impact de la viscosité dépend aussi de la longueur des vagues.
- La contamination de la surface du liquide avec par exemple des algues ou des plantes. Ils ont un gros pouvoir d'amortissement sur les vagues.
- Vagues cassantes et non linéarité : la théorie des ondes d'Airy permet de modéliser des vagues avec une faible amplitude. Donc cette méthode ne modélise pas les vagues cassantes.

Subdivision des paquets

- Subdivision géométrique lorsque 2 particules d'un groupe sont trop éloignées, l'une de l'autre ou si leurs directions ont une différence d'angles supérieurs 18°. La subdivision géométrique créer une nouvelle particule entre les deux existante remplace une particule dans chaque sous-paquets.

— Subdivision dispersive : quand un paquet se disperse, on crée deux nouveaux paquets avec la même position et les mêmes paramètres et distribue le spectre des vagues.

1.6 Conclusion de la recherche documentaire

J'ai présenté les trois familles de modélisation de l'eau ainsi que la littérature que j'ai l'occasion de lire pour effectuer mes recherches.

La modélisation procédurale permet de modéliser les surfaces d'eaux calmes en temps interactif. L'animation de la surface d'eau ressembles à une surface d'eau mais ne représente pas le comportement réel des effets observables. De plus ces méthodes ne peuvent pas représenter des petits effets comme la vagues cassante.

La modélisation physique permet de modéliser la majorité des effets observable de l'eau que ça soit les jets d'eau des vagues cassante ou le comportement d'une surface d'eau calme. Le principal problème de cette modélisation et qu'il est actuellement impossible de d'animer un grand volume d'eau en temps interactif.

La modélisation phénoménologique quand elle essaye de réaliser une modélisation des effets de l'eau en temps réel. Elle se base sur l'observation et la modélisation d'un effet pour réaliser une animation la plus réel possibles sur les effets observés. Par exemple les groupes de vagues par Parenthoen [PJT04], la détection de vague cassante par Belemaalem [BCPR10] et la propagation des ondes par Jeschke [JW17].

Le principale manque observé dans cette recherches documentaire se situe au niveau de la modélisation de vagues cassante en temps réel. Cet effet est bien modélisé en grâce au SPH mais ce n'est pas réalisable en temps réel.

2 Positionnement face au sujet

2.1 Détail du sujet

Le sujet intital du stage est le suivant “Intrication de modèles procéduraux et de modèles physiques pour la simulation graphique des états de mer.” Le but de sujet était d'utiliser les points forts et les points faibles de chaque type de modélisations afin d'obtenir un rendu satisfaisant sur tous les points.

la modélisation procédurale devait être utilisée pour définir le comportement global de la surface de l'eau car c'est beaucoup moins coûteux en temps de calcul que la modélisation physique, mais corriger le problème de la représentation des effets d'éclaboussures et de vague cassante non modélisable grâce aux méthodes procédurales à l'aide des méthodes physiques.

Cependant lors de la rédaction du sujet la modélisation procédurale et phénoménologique n'était pas encore dissociée. Du coup le sujet portait plus particulièrement sur l'intrication en les modèles phénoménologiques et physiques.

C'est pour cette raison que nous avons étudié un maximum de méthodes que je vous ai présenté précédemment. Afin de pouvoir choisir la meilleure manière de réaliser l'implémentation pour le stage.

2.2 Dilemme entre rendu interactif et précalculé

L'un des plus gros problèmes que j'ai pu relever dans ma recherche documentaire est la différence de temps de calcul.

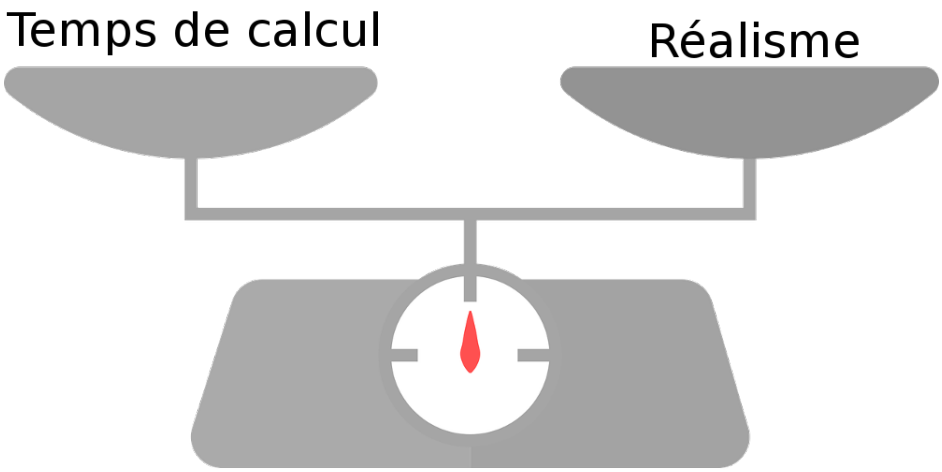

FIGURE 8 – Dilemme entre Temps de calcul et Réalisme

Il faut trouver le bon équilibre entre réalisme et temps de calcul. Le problème est que les deux types de modélisations ont des temps de modélisations trop différents pour être modélisé en même temps. La modélisation procédurale est faite pour modéliser une grande surface d'eau en temps en interactif tandis que la modélisation physique ne permet encore que de modéliser des petites portions d'eau de manière précalculée. L'intrication des deux modélisations aurait forcément conduit à une modélisation précalculée proches des temps que l'on peut observer dans les différents papiers.

La partie qui prend le plus de temps dans la modélisation physique est les interactions entre les particules pour le SPH et les cases de la grille pour la modélisation eulérienne. Les principales interactions qui demandent le plus de temps de calcul sont la force de pression, la viscosité et la tension de surface. Lorsque la différence de temps de calcul est trop importante, l'intrication de deux modèles n'est pas pertinente si on perd la rapidité de l'un et que l'on perd la qualité de rendu de l'autre.

2.3 Choix d'implémentation

Pour l'implémentation nous avons choisi avec mon maître de stage de dévier un peu du sujet initial pour réaliser un programme entièrement en temps interactif. Le moteur de rendu choisi est Unity.

Unity est un moteur de jeu complet ayant une licence gratuite possédant de nombreuses fonctionnalités que nous n'aurons pas à implémenter. C'est aussi un des moteurs les plus connus par sa simplicité et la quantité de fonctionnalité disponible. Il possède aussi la possibilité de modifier le comportement des éléments du jeu grâce à la programmation par script en C sharp. Ce qui est une grande capacité de contrôle et de modification du jeu.

Ce moteur fonctionne avec des objets qui correspondent à des assets que l'on peut instancier dans n'importe quel projet. Il y a plusieurs types d'asset dans Unity :

- Les scènes sont des assets qui permettent d'instancier les autres assets disponibles, il s'agit d'une liste d'objets, dont une caméra fera le rendu. Les objets instanciés peuvent être paramétrés pour la scène, par exemple la position, la rotation, l'échelle etc ...
- Les caméras, se sont elle qui s'occupe de réaliser le rendu de la scène, il peut y en avoir plusieurs dans la scène active en même temps pour certains effets. Seule la caméra principale sera utilisée pour le rendu de la scène finale.
- Les objets, il s'agit d'une liste de composant permettant de définir l'objet. Il y a une très grande liste de composants disponibles
  - Transform permet de définir la position, la rotation et l'échelle
  - Mesh Filter correspond au maillage de l'objet
  - Box Collider permet de définir la forme permettant de détecter la collision de l'objet
  - Mesh Render permet de définir les paramètres de rendu de l'objet

— Le matériau permet de définir l'aspect d'un objet, comment la lumière reflète sur lui, sa couleur ou sa texture, ils peuvent être programmés à l'aide de *shader*.

Un projet réalisés par mon maître de stage effectue déjà l'animation de la surface de l'eau avec des groupes de vagues en temps interactif sur le moteur Unity [PMT15].

Comme précisé dans l'état de l'art, nous avons remarqué qu'il serait impossible de faire du rendu interactif avec une modélisation physique pour les vagues cassantes. C'est pour ça que nous avons décidé d'utiliser un système de particules sans interactions entre particules pour pouvoir rester dans le domaine interactif. Nous utilisons ce compromis car les interactions entre les particules pour les vagues cassante n'impacte pas énormément le comportement des particules.

Le premier travail que j'ai dû réaliser sur le système de particule a été de réaliser un matériau pour le système de particules pas seulement pour une particule, comme les matériaux de base d'unity sont contraints. Le but serait de réaliser un matériau sur lesquels les particules pourraient se déplacer. Il doit aussi être le plus contrôlable possible et prévoir le maximum de cas d'utilisation. Nous allons donc créer notre propre Shader de Sparse texturing.

Le deuxième travail porte sur la détection des collisions entre les particules et les différents objets de la scène. Le module du système de particule par défaut d'Unity possède une méthode pour détecter les collisions entre les différents meshs collider de la scène et les particules du système. Cependant, dans de nombreux cas dont le nôtre, l'animation d'un objet peut être réalisée dans un vertex shader sur la carte graphique. Or, les coordonnés des sommets modifiés dans un vertex shader ne quittent pas la carte graphique, donc les sommets du mesh collider restent fixes, les collisions se font donc sur son état initiale. Une solution possible est de réaliser la détection de collisions sur la carte graphique, c'est cette solution que l'on a choisi d'explorer et d'implémenter.

3 Développement de l'application

3.1 Partie phénoménologique

Analyse du programme existant

Comme vu dans la partie précédante nous avons choisi de reprendre le programme de [PMT15] Parenthoen 2015. Cette application est développée afin de visualiser l'activité du cerveau au travers d'une surface d'eau. Même si cette application a un but artistique, il y a aussi une partie de modélisation de l'eau avec un comportement phénoménologique de l'eau (c'est cette partie que l'on souhaite récupérer), mais aussi lier le comportement de l'eau à l'activité cérébrale de l'utilisateur. Pour cela l'application utilise un casque EEG (Électroencéphalogramme) fonctionnant à l'aide d'un petit programme python.

Le but est quand le l'utilisateur met l'EEG, la simulation de l'eau s'active et la lumière commence à augmenter. C'est la seule étude qui a essayé de traduire les donnés d'un casque EEG dans une représentation des vagues de mer. Dans le papier les longueurs d'onde des vagues changent en fonction de l'EEG dans un certain écart. L'utilisation de l'EEG permet aussi de faire varier la couleur de la lumière, ainsi que le contrôle de la caméra à l'aide d'un mouvement droite gauche pour tourner la caméra. Ce papier utilise une modélisation l'aide de groupes de vagues pour réaliser la modélisation de l'eau.

Ce programme modélise la mer à l'aide de groupes de vagues. C'est un point important car nous disposons d'un autre programme permettant de détecter les vagues cassantes. La prise en main du programme sera donc une partie importante pour la suite afin de mieux appréhender les différents phénomènes.

Travail réalisé sur l'animation phénoménologique

La première partie de mon travail a été d'appréhender le programme déjà existant et de réaliser quelques améliorations afin de s'assurer de la compréhension du fonctionnement du programme. Deux modifications principales ont dû être réalisées, la suppression du contrôle d'un EEG et le passage de la scène en mode journée, car la scène était faite de nuit, très sombre avec des effets difficiles à observer.

Concernant la suppression du contrôle à l'aide d'un EEG, j'ai décidé de garder la fonctionnalité mais de la désactiver par défaut à l'aide d'une simple variable. Puis chaque variables modifiée par l'EEG, que j'ai listé dans les paragraphes précédents, que ça soit pour le contrôle de la caméra ou pour le contrôle des groupes de vagues.

Pour le contrôle j'ai simplement désactivé la modification des variables utilisées pour le contrôle de la caméra si l'on ne souhaite pas utiliser l'EEG, de ce fait, la caméra ne bouge plus que par les flèches du clavier. J'ai aussi développé un nouveau contrôle de caméra plus libre utilisant la souris pour la direction de la vision, les flèches pour le

déplacement en fonction de la direction de la caméra, la touche espace pour monter, la touche shift pour descendre et la touche ctrl pour augmenter la vitesse de déplacement de la caméra. Le but est de faciliter l'observation des phénomènes que l'on va pouvoir ajouter plus tard au programme.

Pour la suppression du contrôle de l'animation de l'eau à l'aide de l'EEG, j'ai localisé toutes les variables précédemment citées pour les instanciés à l'aide de la valeur maximum qui pouvait atteindre, puis j'ai désactivé leurs modifications grâce à l'EGG. J'ai réalisé la même chose pour les variables contrôlant les lumières de la scène.

Ensuite je me suis penché sur le rendu de la scène qui était de nuit, j'ai décidé de la passer jour pour pouvoir mieux distinguer les effets que je pourrais ajouter par la suite. La première que j'ai réalisé était une skybox de jour, que je pourrais utiliser comme cubemap.

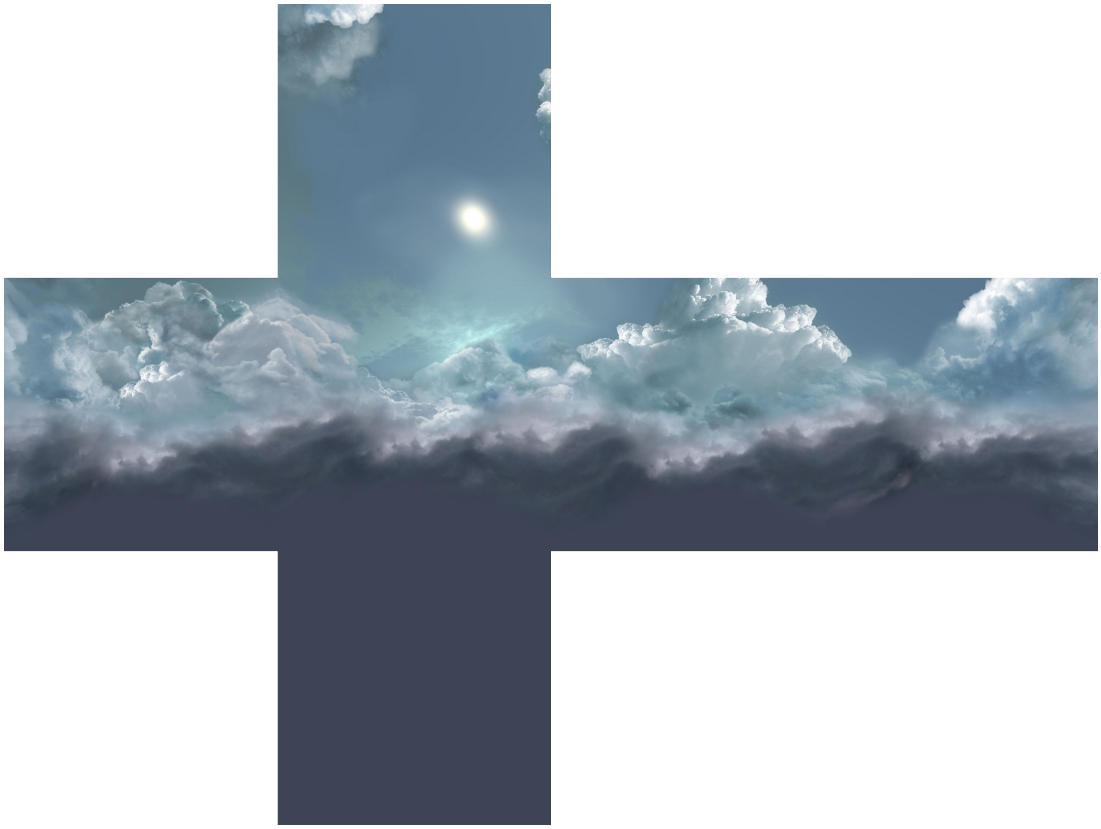

FIGURE 9 – Skybox utilisé pour la scène en mode jour

À cause d'une optimisation d'unity sur le module de skybox, qui oblige les matériaux de la skybox d'être rendu entre la géométrie et le calcul qui pose des soucis avec les objets transparents comme des particules. Du coup j'ai instancié un cube qui est la caméra et j'ai réalisé un shader pour prendre en paramètre la texture, mais aussi pour que le rendu

de la skybox se réalise en premier, qu'il n'écrive pas sur le carte de profondeur et que le rendu ne se fasse qu'à l'intérieur du cube.

En ce qui concerne le matériau de la surface d'eau, il possédait une couleur noire et reflétait que très peu de lumière. La réflexion du matériau ne correspondait pas du tout à ce qu'il devrait être, mais le rendu était quand même correct. Ce problème était compensé avec des lumières beaucoup trop fortes pour une scène de nuit encore plus pour une scène de jour. J'ai donc modifié les lumières et le matériau pour que ça ne soit pas le matériau qui varie en fonction de la nuit ou du jour mais simplement la lumière et la cube map pour les reflets. J'ai aussi augmenté l'impact du coefficient de fresnel du matériau pour mieux marquer les mouvements de l'eau.

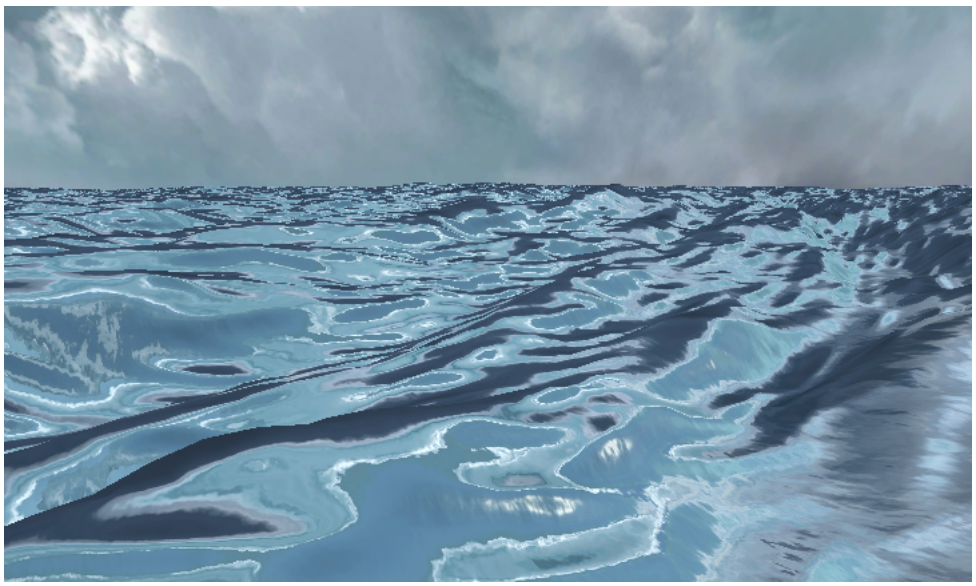

FIGURE 10 – Rendu de la scène en mode jour

3.2 Partie rendu et contrôle du système de particules

3.2.1 Présentation de la méthode

A se moment du stage nous disposions d'un programme Unity réalisant un modélisation de la mer à l'aide de groupes de vagues ainsi qu'un programme en C++ permettant de réaliser la détection des vagues cassantes. Suite au manque de simulation en temps interactif des vagues, nous avons décidé que je devais réaliser un méta-matériau pour des système de particules.

Le méta-matériau doit pouvoir varier en fonction de différents paramètres que l'utilisateur peut transmettre au matériau. Le matériau doit pouvoir aussi bien être modifier par un artiste que par un océanographe qui veut modéliser toutes les différentes étapes d'une vagues cassante. Il faut que l'utilisateur puisse aussi contrôler l'émission des particules du système.

Je vais donc présenter les différents éléments dont nous aurons besoin pour réaliser la simulation, le système de particules de Unity, le fonctionnement des shader Unity et leurs spécificités pour le système de particules et le Sparse texturing. je vais ensuite brièvement présenté ce que l'on souhaite réaliser avant de détailler plus dans la partie suivante.

Le système de particule possède beaucoup de fonctionnalité et de variable permettant de contrôler l'animation et le rendu des particules. Une liste de variables et modules sont disponibles en annexe A.1. Elles peuvent être utilisées afin de modifier l'animation du système de particule. Cependant, pour certaines variables, j'ai dû en détourner l'utilité pour pouvoir implémenter de nouvelles fonctionnalités et économiser de l'espace mémoire, par exemple la variable Color Over Lifetime, dont le gradient est utilisé dans l'un des shaders.

Je vais maintenant détailler le fonctionnement des shaders et leurs spécificités sous Unity. Les deux types de shaders qui nous intéressent sont le vertex shader et le fragment shader (nous n'avons pas utilisé le geometry shader).

Le vertex shader permet de modifier la position des sommets avant de les projeter dans l'espace écran, l'animation de la surface de la mer est fait dans un vertex shader par exemple. C'est aussi par lui qu'arrivent les donnés qui viennent du CPU. Il peut les transmettre au fragment shader.

Le fragment shader quand à lui permet de définir la couleur d'un pixel de l'espace écran appartenant à l'objet auxquels le shader appartient. Les donnés en entrée du fragment shader sont interpolés entre les 3 sommets du triangle dans lequel le pixel appartient et donc défini par le fragment shader correspondant à ce sommet.

Une fonctionnalité unique au système de particule, est la possibilité de modifier les variables vertex stream. Le vertex stream est l'ensemble des variables qui sont envoyées au vertex shader. Cette fonctionnalité a été ajoutée dans la version 2017.2 d'Unity en octobre 2017. Elle permet de passer de nombreuses variables dans les coordonnés de texture, en tout 4 vecteurs de 4 floats soit 16 variables, plus la position, le normal et la couleur, de la particule. Il y a plusieurs variables prédéfinies que l'on peut passer en tant que coordonnés de texture :

- UV, les coordonnés de texture
- AgePercent, le lifetime d'une particule en pourcentage
- Size, la taille de la particule
- Rotation, la rotation de la particule
- Velocity, la vitesse de la particule
- Random, génération de variables aléatoires
- Center, position du centre de la particule

En plus des variables par défaut que Unity nous permet d'utiliser, faire passer les propres variables défini par l'utilisateur aux coordonnés de texture. Il s'agit des Custom Data, ils peuvent être définis dans l'interface d'Unity ou bien être modifié à l'aide d'un script. L'ensemble de ces variables permettront de modifier le comportement des shaders en fonction de l'état des particules et de l'émetteur de particules.

Le sparse texturing est une méthode de plaquage de textures qui diffère de la méthode classique. En effet la méthode classique pour réaliser le plaquage d'une texture est d'utiliser une texture pour l'objet en entier. Le sparse texturing utilise de grandes textures qui sont découpé en plus petite texture qui correspondra à certain sommets. C'est comme si l'objet était découpé en sous objets.

Nous n'allons pas exactement utiliser cette méthode pour le rendu du système de particule, mais nous allons quand même utiliser une grandes texture qui représentera tous les états de la particule dans sa simulation. C'est comme si les particules se déplaçaient sur une texture.

3.2.2 Les matériaux pour le système de particules

Nous avons créé deux matériaux afin de réaliser un rendu de deux manières différentes. Le premier matériau implémente une interpolation entre 3 textures, le deuxième implémente un shader utilisant du sparse texturing. Le matériau d'interpolation est plus simple à comprendre et mettre en place, alors que le matériau de sparse n'est pas limité à 3 points de contrôles.

Partie commune des deux matériaux

Il y a plusieurs parties communes dans les deux shaders de matériaux. Tout d'abord ils sont réalisés en 2 passes de rendues. La première passe est celle qui est commune aux deux matériaux, elle permet d'activer la réception des ombres uniquement pour cette passe. Car la création du carte de profondeur est réalisé grâce au calcul des ombres. De plus on ne peut pas faire recevoir les ombres dans la deuxième passe car la semi-transparence l'en empêche.

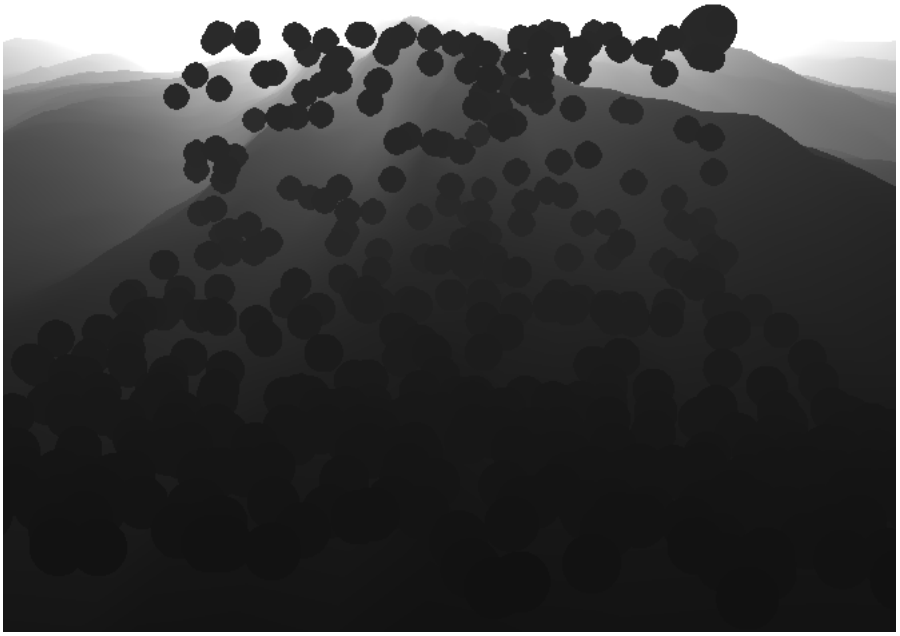

FIGURE 11 – Affichage des particules dans le tampon de profondeur

Le deuxième point commun avec les deux matériaux est l'ajout d'une texture pour définir la transparence de la particule et autres textures pour définir la taille dans le carte de profondeur. L'ajout de cette fonctionnalité est pour l'optique de réaliser un méta matériau pour les systèmes de particules. Cette texture est commune à toutes les particules du système de particules.

```
tex2D(_FadeTex, input.tex.xy).a
```

Code Source 1: Gestion de la transparence

Pour les démonstrations des effets des matériaux je vais réaliser des captures d'écran à l'aide de texture simple permettant de bien mettre en évidence les effets sur lesquels j'ai travaillé. Les captures d'écrans ne sont pas là pour montrer la beauté du rendu, mais de la pertinence et de l'efficacité des effets programmés.

3.2.3 Interpolation entre 3 textures

Le premier matériau que j'ai réalisé est le plus simple des deux à implémenter et à comprendre son fonctionnement. Il réalise l'interpolation d'une à un texture en fonction de la durée de vie de la particule. Pour implémenter cette interpolation nous avons utilisé une variable envoyée par défaut au fragment shader la couleur. J'ai aussi utilisé le module

Color Over Lifetime qui réalise l'interpolation de la couleur en fonction de la durée de vie de la particule. J'ai défini l'interpolation à l'aide d'un gradient que l'on peut définir directement dans l'interface d'Unity.

```
tex2D(_FirstTex,input.tex.xy).rgb*input.col.r+
tex2D(_SecondTex,input.tex.xy).rgb*input.col.g+
tex2D(_ThirdTex,input.tex.xy).rgb*input.col.b,
```

Code Source 2: Affichage de l'interpolation entre les 3 textures

Comme on peut le voir dans ce calcul, la couleur du pixel correspond en un mélange entre trois textures qui varient en fonction d'une composante respective de la couleur qui varie en fonction de la durée de vie de la particule.

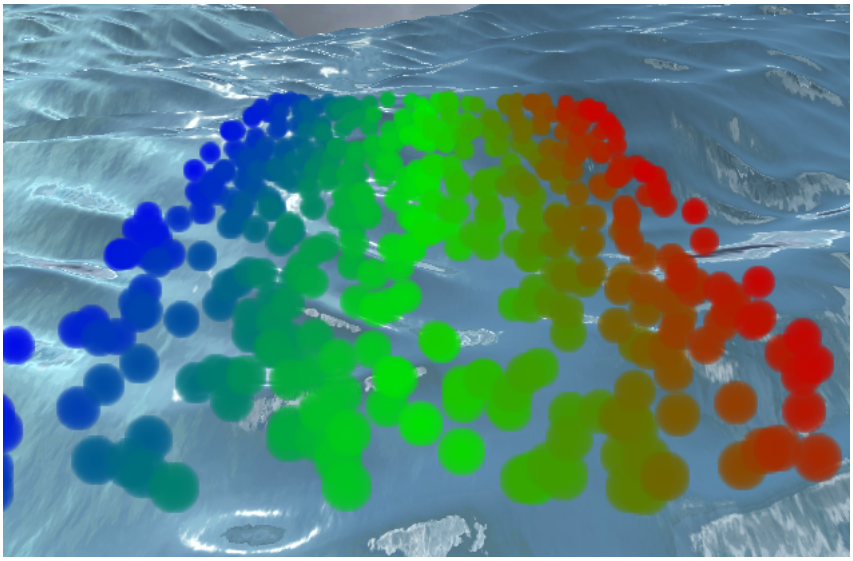

FIGURE 12 – Interpolation entre 3 textures, une rouge, une verte et une bleu

Il s'agit d'un système de particules avec vitesse constante et durées de vie identiques. Il s'agit de l'interpolation en 3 textures une rouge une verte et une bleue dépendant de leurs composantes respectives de la couleur de la particule, qui est définie par un gradient avec lifetime de 0 à rouge, lifetime de 0.5 à vert et lifetime de 1 à bleu. On peut donc voir que les textures affichées correspondent bien au gradient avec pour émetteur de particules la droite de l'image et la fin de vie des particules à gauche.

3.2.4 Sparse texturing

Le principe de ce matériau est qu'il puisse prendre paramètre une méga texture dans la laquelle une particule pourra se déplacer et même varier en fonction de certaines variables. La réalisation de ce matériau est bien plus fastidieuse que le précédent. Car le plaquage de

la texture ne correspond pas aux coordonnés de texture des sommets, de plus ils dépendent aussi des différents paramètres. Je vais d'abord présenter l'idée principale pour réaliser le matériau, puis je parlerai des différentes étapes et différents problèmes que j'ai rencontrés.

Présentation de la méthode

Le but du matériau est de réaliser une projection de la particule sur la texture en fonction de plusieurs paramètres, le premier paramètre que l'on a choisi par défaut est la position d'émission de la particule le deuxième étant la durée de vie de la particule. On a pris l'exemple de la durée de vie mais l'on pourrait prendre n'importe quelle variable comprise entre 0 et 1.

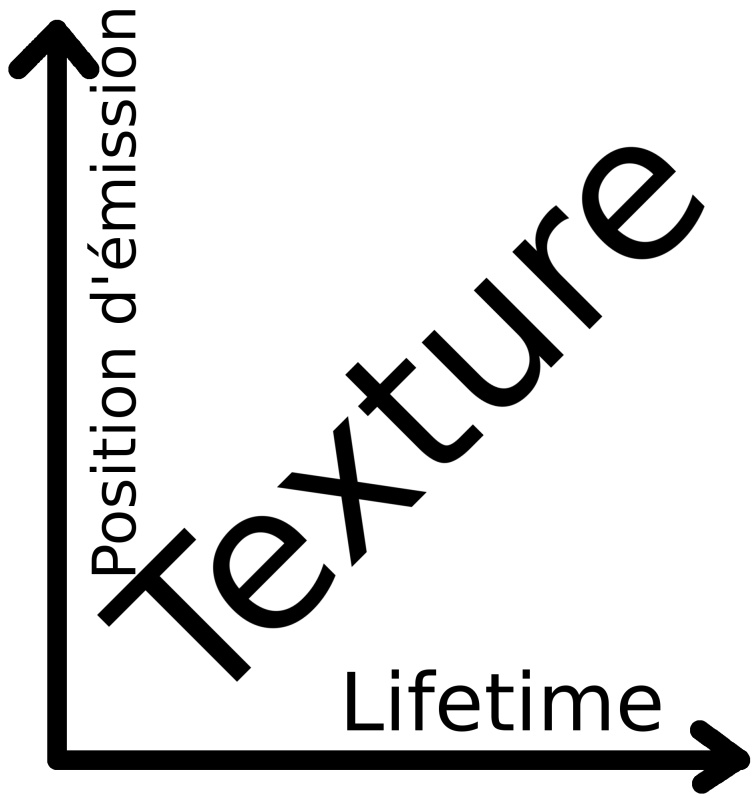

FIGURE 13 – Graphe montrant l'évolution des coordonnés de texture en fonction du temps de vie et de la position

Ce graphe présente l'évolution des coordonnés de textures pour les particules dans notre matériau. J'ai choisi arbitrairement la durée de vie en tant que coordonnée x de la texture et la position grâce à la coordonnée y.

Au début de la programmation, j'avais décidé de calculer directement les coordonnées de texture à l'aide de la position des sommets. Cependant j'ai rapidement remarqué que le fait que la particule est orientée vers la caméra, la projection de vertex en fonction

de la position d'émission déforme la texture de la particule. Pour résoudre le problème j'ai décidé de projeter la particule sur les coordonnés de texture.

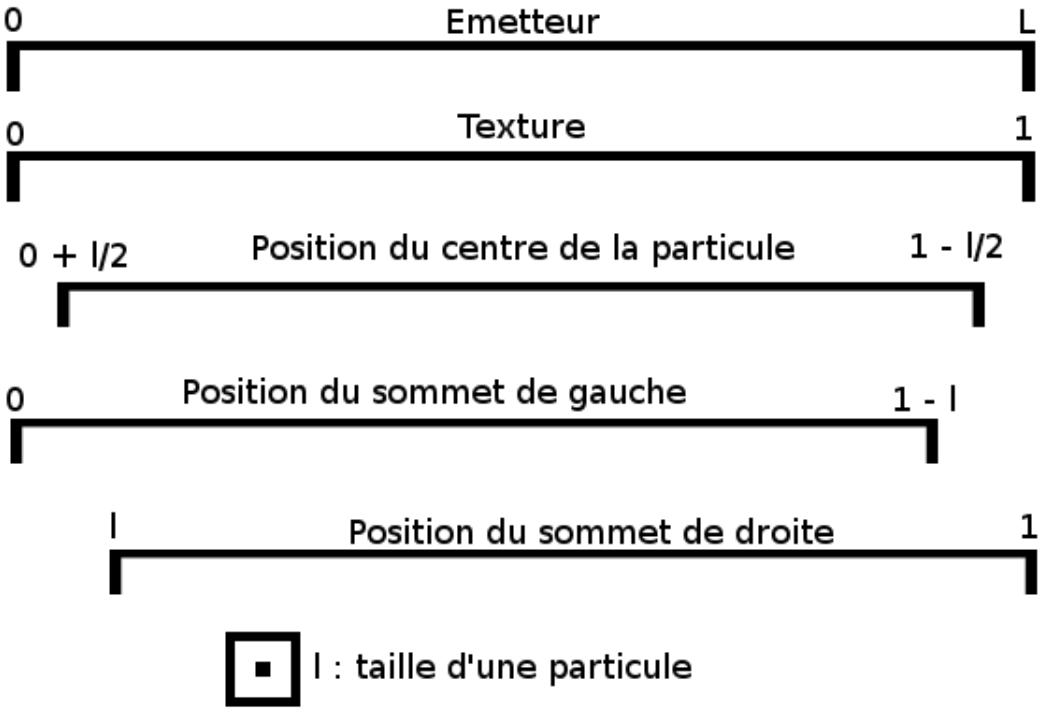

FIGURE 14 – Projection des coordonnés de la particule dans les coordonnés de textures

Grâce au vertex stream je peux récupérer la taille de l'émetteur et la taille d'une particule. Ces donnés me permettent de projeter les particules sur la texture pour calculer les marges à décaler.

$$p_g = p_c/L * (1 - l)$$

$$p_d = p_c/L * (1 - l) + l$$

— p_g position du sommet avec une coordonné de texture de 0 de la particule
— p_d position du sommet avec une coordonné de texture de 1 de la particule
— p_d position du centre de la particule
— L taille de l'émetteur
— l taille de la particule

La dernière information dont j'ai besoin pour calculer la projection est de positionner le vertex par rapport au centre. Au début j'ai essayé de calculer grâce à une différence entre le vertex et le centre de la particule. J'ai eu beaucoup de problèmes à cause du système de billboard de Unity qui change la rotation des axes de la particule. Mais une solution plus simple est plus efficace en temps de calcul est d'utiliser les coordonnées de textures par défaut de la particule qui sont toujours positionnées de la même manière.

Avec cela j'ai réussi à calculer la moitié d'une taille de particule dans la projection. Cette donnée est aussi utilisée pour le calcul de la projection pour limiter la projection du centre de la particule avec une marge d'une demi taille de particule dans les coordonnés de textures.

Calcul de la distance

La variable concernant la distance par rapport à lieu uniquement lors de l'instanciation de la particule sur le CPU, il est passé au shader grâce aux customs data du vertex stream. On ne peut pas utiliser la position de l'émetteur car il se situe en son centre. Pour cela il faut prendre calculer la distance entre la particule et un bord de l'émetteur.

Pour calculer la position du bord il faut d'abord décaler la position du centre de l'émetteur de la moitié de la taille de l'émetteur en considérant qu'il n'y a pas de rotation. On applique la rotation de l'émetteur qu'après. On peut ensuite calculer la distance entre la particule et le point calculé. Il faut ensuite diviser par la taille de l'émetteur pour obtenir une variable comprise entre 0 et 1 comme pour la variable de durée de vie.

3.2.5 Conclusion

Voici le rendu que l'on peut obtenir avec une vitesse d'émission constante, une durée de vie constante et sans gravité et une texture en forme d'arc-en-ciel sphérique pour voir la projection dans les deux coordonnées de la texture.

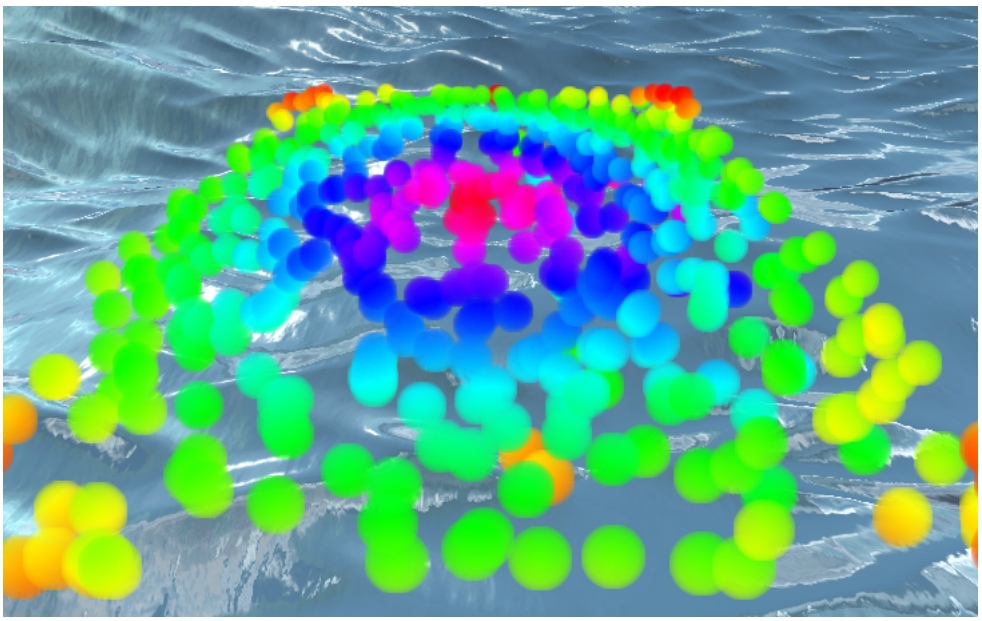

FIGURE 15 – Sparse texturing du système de particules avec une texture d'arc-en-ciel

Voici le rendu avec une grande image bruitée.

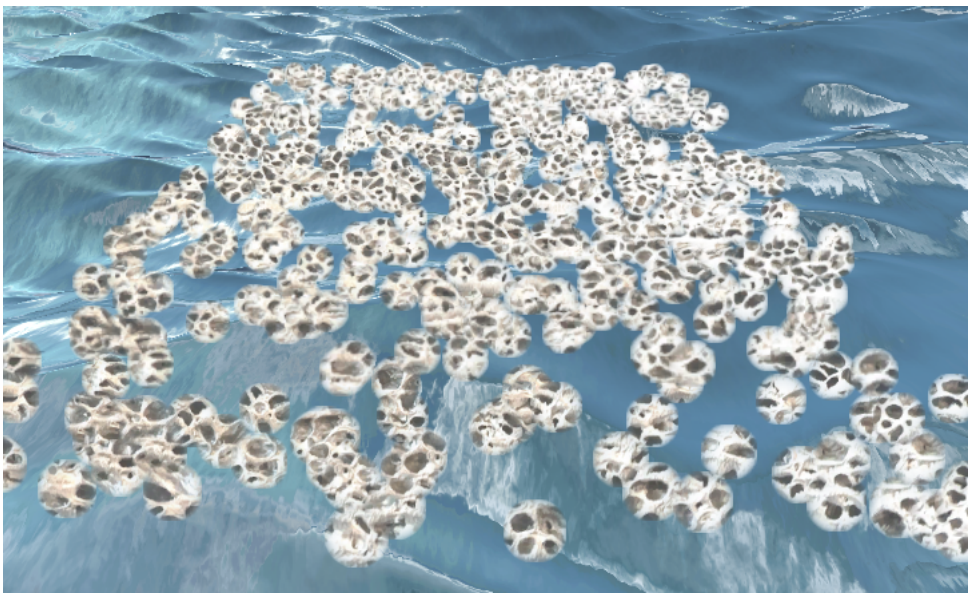

FIGURE 16 – Sparse texturing du système de particules avec une texture bruitée

Les deux matériaux permettent un degré de contrôle différents états des particules du systèmes. L'interpolation entre 3 textures est homogène pour toutes les particules, cependant elle simple à utiliser, à comprendre et à manipuler pour un utilisateur.

La matériau "sparse texturing" quand à lui, ne réalise pas d'interpolation, elle doit être réalisé à la main dans la texture car elle définit toutes les étapes de la vie de la particule. On dispose aussi une différenciation de chaques particule émise grâce à leur position d'émission.

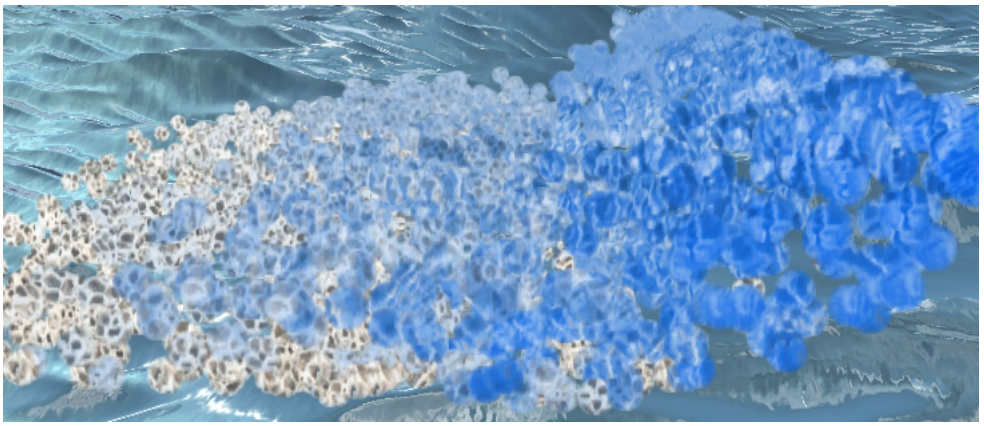

FIGURE 17 – Sparse texturing du système de particules avec une texture plus réaliste avec la gravité

Nous avons pris en exemple la durée de vie d'une particule pour le matériau "sparse texturing", mais on peut utiliser n'importe quelle variable entre 0 et 1. Il est aussi possible d'ajouter des variables, il faudra ajouter une dimension à la texture que l'on souhaite réaliser. Pour le moment on utilise que 2 variables la durée de vie et la position d'émission, nous avons donc besoin que de 2 dimensions à notre texture.

3.3 Détection des collisions

Un des problèmes rencontrés lors des tests que l'on a réalisés avec le système de particules est la détection de collisions avec les objets de la scène. Pour cela Unity nous fournit plusieurs méthodes afin de réaliser cette détection de collisions. Je vais présenter chaque méthode et détailler les points positifs et négatifs de la méthode, pour ensuite en conclure si l'on peut ou pas l'utiliser pour notre simulation.

3.3.1 Les méthodes de collisions pour le système de particule avec Unity

La première méthode, qui est la plus utilisée, est une détection de collision avec tous les objets du monde. La simulation physique réalisée pour cette détection utilise les meshs colliders (maillage de collisions) de chaque objet de la scène. L'avantage de cette méthode est qu'elle peut être généralisée à toute la scène, cependant les meshs colliders sont définis uniquement sur le CPU, alors que notre simulation est réalisée sur GPU, du coup nous ne pouvons pas utiliser cette méthode pour notre simulation.

La deuxième méthode est une détection à l'aide de plans. Cette détection de collision est réalisé par le système de particule. Les plans qui servent à la détection sont créé à la main, c'est-à-dire que l'on peut les positionner au préalable dans l'éditeur d'Unity, ou bien à l'aide d'un script si le système de particule est instancié lors de la simulation. Les plans sont justes défini par une position et une direction, ça veut dire qu'ils sont infinis.

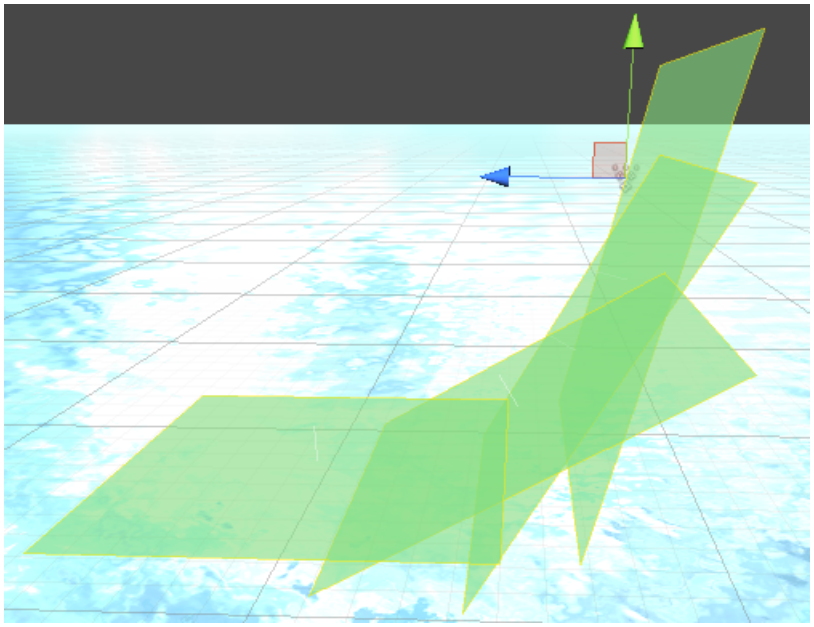

FIGURE 18 – Les plans de collisions pour les particules

Comme on peut le voir sur cette image, l'idée serait de placer les plans pour qu'ils puissent suivre la courbe la vague. Cette méthode fonctionne pas mal pour définir la forme basique de la courbe de la vague, cependant il est impossible de définir les tous les détails de l'eau, ni l'impact de tous les groupes de vagues même en augmentant le nombre de plans. Comme ils sont infinis, on ne peut représenter que des formes convexes. Ce qui n'est pas le ça pour la courbure d'une vague dans notre modélisation de la mer. Donc la plupart des détails ne pourront pas être modélisés pour la collision, un manque de précision qui ferait apparaître des artefacts de collision. Malgré les défauts, on a décidé de garder cette méthode comme option de secours si la méthode que l'on a développée n'est pas suffisante.

3.3.2 Théorie de la méthode implémenté

J'ai eu l'opportunité de lire cette présentation [Cup] réalisée pour la conférence SIGGRAPH 2011. Parmi toutes les méthodes présentées dans cette présentation une a retenu mon attention, la détection de collision à l'aide de la méthode d'occlusion ambiante. En effet l'occlusion ambiante détecte le changement abrupt des normales de deux pixels proches dans l'espace écran avec des profondeurs proches. Même si cela peut s'appliquer aux mêmes objets la méthode arrive à détecter avec la carte de profondeur et de normal des pixels qui correspondent à des objets proches.

La carte de profondeur et la carte des normales sont calculées sur la carte graphique, elles comportent donc les donnés de l'animation de la mer qui est réalisée aussi sur la carte graphique. Pour que cela puisse fonctionner il y a plusieurs conditions à respecter. Tout d'abord il faut pouvoir contrôler l'apparition des particules sur la carte des profondeurs,

une carte avec les particules pour visualiser la carte de profondeur et un autre pour les calculs. Ensuite il faut pouvoir contrôler et réfléchir les particules dont on aura détecté la collision.

Pour la première condition (contrôle de l'apparition des particules sur la carte de profondeur) a déjà été réalisée et présentée dans une partie précédente 11. Pour pouvoir correspondre à la deuxième condition il faut réaliser l'animation des particules sur la carte graphique et pouvoir les contrôler.

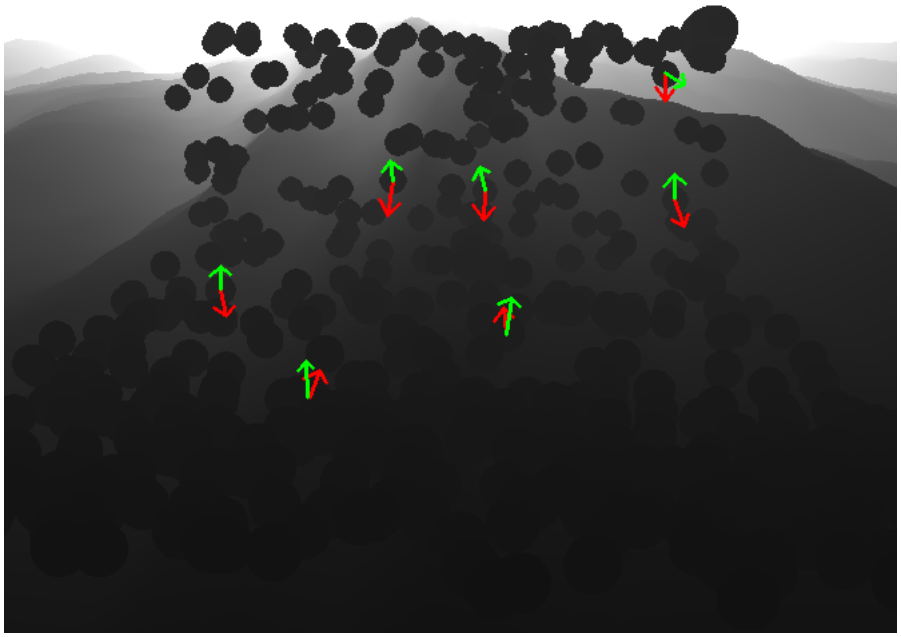

FIGURE 19 – Carte des profondeurs avec normale, flèche rouge : vélocité des particules, flèche verte : normale de la surface

Cette image montre la carte de profondeur avec les particules ainsi que le normal représenté par des flèches, rouge pour la direction de la particule et verte pour la normale de la surface. Pour récupérer la profondeur et la normale sur la carte de profondeurs et la carte de normales, il faut connaître la position de la particule dans l'espace de l'écran. Ensuite on peut récupérer sur les deux cartes les données dont on a besoin.

Il manque encore 2 donnés pour réaliser la détection de collision, la profondeur et la direction de la particule. Pour calculer la direction d'une particule il faut normaliser sa vélocité. Pour la profondeur, il faut calculer le rapport entre la distance entre la particule et la caméra sur la distance avec le plan de clipping lointain.

Grâce à ces donnés il reste plus qu'à réaliser la méthode, d'occlusion ambiante adaptée pour la détection de collisions. Pour cela il faut détecter si la profondeur de la particule et la profondeur du pixel derrière la particule sont proches. Ensuite il faut que la vélocité de la particule et la normale de la scène soient dans des directions opposées.

3.3.3 Implémentation de la méthode

Concernant l'implémentation de la méthode, au moment de la rédaction du rapport, elle n'est pas terminée. Je vais présenter le travail réalisé et les problèmes rencontrés jusqu'à la rédaction du rapport et ce qu'il reste faire jusqu'à la fin du stage.

Le premier point que j'ai essayé de réaliser est de réussir à contrôler les particules sur la carte graphique. Le plus gros problème c'est que je n'ai trouvé aucun moyen de modifier le comportement des particules simulé depuis la carte graphique, car on utilise le système de particule par défaut d'unity. Pour résoudre le problème, il faudrait créer un nouveau système de particules dont l'animation des particules sera calculée sur le GPU dans lequel on pourra faire le test de collision.

Par manque de temps nous avons décidé de ne pas refaire un système de particules en entier mais de réaliser uniquement la méthode, même si on doit passer par le CPU. Le but est de réaliser la méthode est ensuite de l'optimiser. Le contrôle des particules est plus simple à réaliser sur le CPU.

```
int particleCount = ps.particleCount;
ParticleSystem.Particle[] particles = new ParticleSystem.Particle
 ↪  [particleCount];
ps.GetParticles(particles);

for (int i = 0; i < particles.Length; i++) {
if(particles[i].position.y<0){
particles[i].velocity = Vector3.Reflect (particles[i].velocity,
 ↪  Vector3.up) * 0.5f;
}
}
ps.SetParticles(particles, particles.Length);
```

Code Source 3: Gestion de collisions sur le CPU

Il s'agit de la partie du code qui réalise la détection et la réflexion des particules sur le plan horizontal y = 0. Les 3 premières lignes de code permettent de récupérer la liste des particules dans la variable *particles*. Ensuite on parcourt toute la liste de particules grâce à la boucle for, puis c'est dans la condition de l'if que l'on réalise le test de collision, dans cet exemple si la particule est passé sous le plan horizontal de hauteur 0. Quand une

détection a été trouvé on réalise la réfection de la vitesse avec la normale de la surface, à laquelle on applique un amortissement (0.5f dans cet exemple).

Pour récupérer la profondeur et les normals des pixels de la caméra sur le CPU nous avons décidé de passer par des Renders textures. Ces textures permettent de réaliser le rendu dans une image qui sert de texture au lieu d'afficher le rendu sur l'écran de l'utilisateur. Pour cela j'ai créé une nouvelle caméra qui est positionnée au-dessus du système de particule.

Cette nouvelle caméra permet aussi de régler un problème de la méthode, lorsque la particule n'est pas visible par la caméra. Dans ce cas-là, la détection de collision n'aura pas lieu car nous ne pourrons pas récupérer la bonne normale ni la bonne profondeur. C'est pour cela que si l'on arrive bien à positionner une caméra, qui sert juste à la détection de collision, au-dessus que chaque système de particules nous pourrons faire disparaître les zones cachées. De plus nous pouvons utiliser des plans pour réaliser une détection de "secours" sous la surface pour éviter de perdre des particules dans la simulation. De plus si la collision n'est pas visible par la caméra la précision de la simulation n'est pas très importante.

Pour réaliser l'affichage de la carte des normales et de la profondeur, nous pouvons utiliser la texture "uniform sampler2D CameraDepthNormalsTexture ;" dans laquelle sont codées les normales et les profondeurs. Pour décoder les valeurs nous pouvons utiliser la méthode "DecodeDepthNormal". En affichant cette texture dans un shader en post process. Nous avons réussi à récupérer la texture décodée, cependant à cause d'une carte graphique trop ancienne, la texture n'est pas mise à jour lors de l'animation contrairement à "CameraDepthTexture" pour laquelle nous n'avons aucun problème 11.

Pour pallier ce problème nous allons utiliser une fonctionnalité d'Unity qui s'applique à une caméra, les Replaced Shaders. Le fonctionnement de ce type shader est plutôt complexe, je vais donc détailler son fonctionnement étape par étape avant de présenter comment nous allons l'utiliser. Tout d'abord les Replaced Shaders permettent de remplacer les shaders existant par un autre. Pour ce cela il faut un shader qui sera le Replaced Shader, il peut avoir plusieurs subshaders.

```
Shader "ReplacedShaders" {
     SubShader {
         Tags { "RenderType"="Opaque" }
         Pass {
             ...
         }
     }
     SubShader {
         Tags { "RenderType"="SomethingElse" }
         Pass {
             ...
         }
     }
 ...
 }
```

Code Source 4: ReplacedShaders - Source : documentation d'Unity

Dans ce shader on peut voir 2 subshaders, ce sont ces subshaders qui remplaceront les subshaders que l'on souhaite.

```
camera.SetReplacementShader (ReplacedShaders, "RenderType");
```

Code Source 5: Initialisation du shader de remplacement

Grâce à cette fonction on peut attribuer un shader de remplacement à une caméra qui utilisera dans ce cas, la valeur du tag rendertype pour réaliser le remplacement. Dans notre exemple, tous les shaders qui seront exécutés par la caméra et qui auront un subshader ayant un tag "RenderType"="Opaque" auront le subshader remplacé par le subshader de "ReplacedShaders" qui aura le tag "RenderType"="Opaque".

Je souhaite utiliser cette fonctionnalité pour réaliser un sub shader qui affichera les normales de chaque objet qui auront la bonne valeur de tag. Cela permettrait d'outrepasser les limitations matérielles et de mieux contrôler quel élément doit afficher ou non ces normales.

Le Replaced Shader et le test de collision seront réalisés pendant la fin du stage.

Conclusion

Le sujet initiale du stage "Intrication de modèles procéduraux et de modèles physiques pour la simulation graphique des états de mer" a bien évolué. Il y a plusieurs raisons qui nous ont poussé à changer le déroulement du stage. Tout d'abord, d'après des recherches non publiées, les interactions entre particules sont négligeables pour les vagues cassantes dans les modélisations physiques.

Aussi le déroulement du stage a été adapté afin que nos compétences individuelles de mon maître de stage et moi-même soient utilisés au mieux pour se compléter et perdre le moins de temps possible.

Les avancées les plus importantes que nous avons apportées par rapport à l'état de l'art actuelle sont la création de 2 métas-matériaux permettant de réaliser un rendu réaliste des différents états d'une vague cassante. La deuxième avancée est sur la détection de collisions, sur la carte graphique dans l'espace écran, des particules de la vague cassante.

Il reste encore du travail à réaliser par exemple, adapter le programme qui réalise la détection de vagues cassantes pour qu'il puisse fonctionner sur Unity avec la simulation de la surface de l'eau. Avant la fin du stage nous souhaitons finir la détection de collisions des particules afin de valider la méthode théorique.

Lorsque le projet sera finalisé mon maitre de stage souhaite réaliser un article de recherche sur cette méthode, pour ensuite le distribuer gratuitement sur l'asset store d'Unity, car les métas-matériaux et la détection de collisions sont adaptatives (scale, rotate, translate ...) et paramétrables.

A Annexe

A.1 Système de particule d'Unity

Le bouton open editor permet d'ouvrir une fenêtre d'édition des paramètres du système de particule, il ajoute une fonctionnalité, celle de modifier les courbes définissant nos paramètres

- Main module (module principale)
  - Duration : durée d'émission des particules
  - Looping : permet de faire boucler ou non l'émission des particules a la fin de la de la durée d'émission
  - Prewarm : permet de simuler un tour d'émission pour que le système soit déjà fonctionnel lorsque il est instancié
  - Start delay : décale le début de l'émission du système (constante, aléatoire)
  - Start lifetime : la durée de vie d'une particule (constante, courbe, aléatoire)
  - Start speed : vitesse initiale des particules émises (constante, courbe, aléatoire)
  - 3D Start Size : permet de fixer la taille des particules a l'émission dans les 3 axes XYZ (constante, courbe, aléatoire)
  - Start Size : permet de fixer la taille des particules a l'émission (constante, courbe, aléatoire)
  - 3D Start Rotation : permet de fixer la rotation initiale des particules dans les trois axes (constante, courbe, aléatoire)
  - Start Rotation : permet de fixer la rotation initiale des particules (constante, courbe, aléatoire)
  - Random rotation : probabilité qu'une particules ai une rotation initiale inverse (constante)
  - Start Color : couleur de départ de la particule (Color, gradient (dégradée), aléatoire)
  - Gravity modifier : multiplicateur de la force de gravité 0 désactivé 1 activé (constante, courbe, aléatoire)
  - Simulation Space : Local, World et Custom, permet de réaliser la simulation des particules dans un repère précis, par exemple pour le local si le système se déplace, l'ensemble des particules déjà émise se déplace avec, alors que si les particules sont simuler dans l'espaces monde, les particules déjà émise ne bougeront si leur émetteur bouge.
  - Simulation Speed : permet de changer la vitesse de simulation du système (constante)
  - Delta Time : Scaled et Unscaled, unscale fixe le delta temps (ne dépende plus de l'affichage, scaled fait varier le delta de temps en fonction de l'affichage.
  - Scaling Mode : Local, Hierarchy et Shape
  - Play on awake : si vraie lance la simulation à la création du système de particule sinon elle doit être lancé à la main grâce à la fonction play().
  - Emitter Velocity : Transform et Rigid Body calcul de la vitesse émission en utilisant soit la position soit la vitesse
  - Max Particles : Limite le nombre maximum de particules qui peuvent être

généré
  - Auto random seed : permet de générer un aléatoire différent pour chaque simulation
  - Stop action : Que faire quand la simulation est terminé none, disable, destroy et callback.
- Emission Contrôle l'émission des particules
  - Rate Over Time : Le nombre de particules émises par secondes
  - Rate Over Distance : le nombre de particules en fonction de la distance de émetteurs marche que en mouvement
  - Burst : Émission de particules a un moment précis de la simulation. (Time, Count, Cycles, Interval)
- Shape : Permet de définir la forme de l'émission chaque formes d'émission a une liste de variables propres, ils est aussi possible de modifier cette forme (les variables, à l'aide de l'éditeur graphique de la scène.
  - Sphere
  - Hemisphere
  - Cone
  - Donut
  - Box
  - Mesh
  - Mesh Renderer
  - Skinned Mesh Renderer
  - Circle
  - Edge
- Velocity over Lifetime permet de modifier la vitesse sur les 3 axes
- Limit Velocity over Lifetime permet de limiter la vitesse
  - Separate axes permet de contrôler la vitesse sur les trois axes
  - Speed permet de fixer une vitesse maximale
  - Dampen Permet de spécifier la quelle quantité sera limité compris entre 0 et 1 pourcentage
  - Drag permet de freiner la particule, entre 0 et 1, on peut freiner par la vitesse et/ou la taille de la particule
- Inherit Velocity Permet de contrôler la manière dont la vitesse de l'émetteur est transmise aux particules (Initial et Current),on peut appliquer un multiplicateur.
- Force Over Lifetime Permet d'ajouter et de contrôler une force externes à la particule
- Color Over Lifetime Couleur en fonction de la durées de vie (gradient, aléatoire)
- Color by Speed Change la couleur en fonction de la vélocité (gradient, aléatoire)
- Size Over Lifetime Taille de la particule en fonction du temp (gradient, aléatoire variable et gradient)
- Size by Speed Taille de la particule en fonction de la vitesse, on peut choisir les bornes de ces valeurs, pour définir la variation entre ces 2 valeurs on utilise une courbe ou un aléatoire
- Rotation Over Lifetime Permet de réaliser une rotation en fonction de la temps sur

la particule, cette rotation peut être effectuer sur les 3 axes (Constante, courbes et aléatoire)
- Rotation by Speed La rotation semble fixe et de ne pas varier en fonction de la vitesse (A vérifier)
- External Forces Multiplieur des forces induite par les zones de vents qui sont ajouté dans la scène
- Noise Permet de rajouter un peu de comportement chaotique dans le mouvement des particules il y a plein de variables permettant de changer le noise
  - Collision Permet de gérer les collisions des particules
  - Plane Permet de définir les plan sur lesquels les collisions auront lieux
  - World Les collisions seront gérées en fonction des colliders de la scène
- Triggers Permet de réaliser des actions lorsqu'une particule rentre en contact avec un collider passé en paramètre, on peut soit exécuter un script soit utiliser des actions prédéfinis comme supprimer la particule
- Sub Emitters Permet d'instancier des sous système particules en fonction de certains évènements
- Texture Sheet Animation Permet d'animer la particule en faisant varier c'est UV
- Lights Permet de rajouter des lumière aux particules, en leur passant un objet light
- Trails Permet de créer une traînée sur le passages des particules
- Custom Data Permet de définir des valeurs que l'on pourra récupérer à l'aide d'un script
- Renderer Permet de gérer toutes la partie rendu de la particule, le matériaux etc ...

A.2 EEG BCI [PMT15]

L'utilisation du cerveau comme interface suscite de l'intérêt de la recherche de 2006 grâce à l'étude de Birbaumer sur les BCI (Brain Computer Interface). OpenVibe est une suite logicielle qui a réalisé des recherches dans le but de contrôler des mondes virtuels.

Les fréquences de l'EEG sont larges mais seules ceux de 0.5 à 40 hz sont utilisé dans les BCI, ils sont divisés en 5 groupes (voire papier). Chaque groupe de fréquence correspond à un changement d'activité cérébrale. Ces groupes sont limités à des fréquences comprises entre 0.5 à 40 hz à cause des perturbations et du bruit que ces méthodes rencontrent. Fatourechi et al en 2007 ont développé une méthode permettant de filtrer le bruit d'un EEG se pendant cette méthode est très compliqué et pas réalisable en temps réel. Cela explique pourquoi 25% des personnes ne peuvent calibrer correctement un BCI.

Les études précédentes montrent qu'il facile de détecter la différence entre un mouvement vers la droite et un mouvement vers la gauche. Activité dans le lobe frontal gauche pour un mouvement vers la droite et une activité frontale droite pour un mouvement vers la gauche. Ce point est utilisé dans l'application pour faire tourner la caméra. Les bonnes et mauvaises émotions sont aussi simples à détecter, une activité dans la partie préfrontale gauche est signe de sentiments positifs et droite pour négatif.

Lien entre l'EEG et la surface de la mer

Les fréquences sont relevé toutes les secondes et calculé l'énergie électrique grâce à la transformée de Fourier discrète. Les coefficients de la transformée permettent d'obtenir les 6 valeurs, correspondant à l'énergie mesurée par chaque électrode. 4 zones sont mesurées avec 6 fréquences pour chaque donne 24 variables sur lesquels travailler.

Contrôle de la rotation de la caméra

- Tright pour la somme des By des détecteurs de droite
- Tleft pour la somme des By des détecteurs de gauche
- rfast = 1s et rslow = 10s

$$\overline{\mathcal{T}_{right}^{fast}}(t) = \alpha_{fast}\mathcal{T}_{right}(t) + (1-\alpha_{fast})\overline{\mathcal{T}_{right}^{fast}}(t-\delta t)$$
$$\overline{\mathcal{T}_{left}^{fast}}(t) = \alpha_{fast}\mathcal{T}_{left}(t) + (1-\alpha_{fast})\overline{\mathcal{T}_{left}^{fast}}(t-\delta t)$$
$$\overline{\mathcal{T}_{right}^{slow}}(t) = \alpha_{slow}\mathcal{T}_{right}(t) + (1-\alpha_{slow})\overline{\mathcal{T}_{right}^{slow}}(t-\delta t)$$
$$\overline{\mathcal{T}_{left}^{slow}}(t) = \alpha_{slow}\mathcal{T}_{left}(t) + (1-\alpha_{slow})\overline{\mathcal{T}_{left}^{slow}}(t-\delta t)$$

FIGURE 20 – Formules de contre de la caméra à l'aide d'un EEG

Un controler permet de garder la caméra au-dessus de l'eau et l'empêche de passer sous la surface de l'eau, pour cela le programme utilise un test sur 4 points autour de la caméra et mesure leur hauteur par rapport à la surface.

B Codes et Algorithmes

Table des Codes Source

C Illustrations et Images

Table des figures

Références

[ADAT13] Nadir Akinci, Alexander Dippel, Gizem Akinci, and Matthias Teschner. Screen space foam rendering. 2013.

[BCH+12] Hubert Branger, Amin Chabchoub, Norbert Hoffmann, Nail Akhmediev, Olivier Kimmoun, and Christian Kharif. Évolution de l'enveloppe de type soliton pulsé de peregrine d'un groupe de vagues : approches analytique et expérimentale. In *13èmes journées de l'hydrodynamique*, volume 13, pages 1–12, 2012.

[BCPR10] Zakaria Belemaalem, Bertrand Chapron, Marc Parenthoën, and Nicolas Reul. The groupy wave model for simulating dynamical sea surface. In *OCOSS'2010 Conference*, 2010.

[BDM+16] Mathias Brousset, Emmanuelle Darles, Daniel Meneveaux, Pierre Poulin, and Benoît Crespin. Simulation and control of breaking waves using an external force model. *Computers & Graphics*, 57 :102–111, 2016.

[BKKW17] Jan Bender, Dan Koschier, Tassilo Kugelstadt, and Marcel Weiler. A micropolar material model for turbulent sph fluids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, page 4. ACM, 2017.

[BMF07] Robert Bridson and Matthias Müller-Fischer. Fluid simulation : Siggraph 2007 course notes video files associated with this course are available from the citation page. In *ACM SIGGRAPH 2007 courses*, pages 1–81. ACM, 2007.

[Bri15] Robert Bridson. *Fluid simulation for computer graphics*. CRC Press, 2015.

[Bro17] Mathias Brousset. *Simulation et Rendu de Vagues Déferlantes*. PhD thesis, Université de Poitiers, 2017.

[BS10] Colin Braley and Adrian Sandu. Fluid simulation for computer graphics : A tutorial in grid based and particle based methods. *Virginia Tech, Blacksburg*, 2010.

[CH17] Ok-Hue Cho and Sung-dae Hong. Real-time 3d fluid simulation digital art using bci. *Journal of Real-Time Image Processing*, 13(3) :419–429, 2017.

[CI07] Deborah Carlson and Sony Pictures Imageworks. Wave displacement effects for surf's up. In *SIGGRAPH Sketches*, page 91, 2007.

[Cup] Kuba Cupisz. Special effects with depth. `https://blogs.unity3d.com/2011/09/08/special-effects-with-depth-talk-at-siggraph/`. talk at SIGGRAPH 2011.

[Dau88] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7) :909–996, 1988.

[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.

[FMB+17] Yun Raymond Fei, Henrique Teles Maia, Christopher Batty, Changxi Zheng, and Eitan Grinspun. A multi-scale model for simulating liquid-hair interactions. *ACM Transactions on Graphics (TOG)*, 36(4) :56, 2017.

[FR86] Alain Fournier and William T Reeves. A simple model of ocean waves. *ACM Siggraph Computer Graphics*, 20(4) :75–84, 1986.

[GM77] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics : theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3) :375–389, 1977.

[GM84] Alexander Grossmann and Jean Morlet. Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM journal on mathematical analysis*, 15(4) :723–736, 1984.

[IAAT12] Markus Ihmsen, Nadir Akinci, Gizem Akinci, and Matthias Teschner. Unified spray, foam and air bubbles for particle-based fluids. *The Visual Computer*, 28(6-8) :669–677, 2012.

[JG01] Lasse Staff Jensen and Robert Golias. Deep-water animation and rendering. In *Game Developer's Conference (Gamasutra)*, 2001.

[JW17] Stefan Jeschke and Chris Wojtan. Water wave packets. *ACM Transactions on Graphics (TOG)*, 36(4) :103, 2017.

[Kam08] Seung Ihl Kam. Improved mechanistic foam simulation with foam catastrophe theory. *Colloids and Surfaces A : Physicochemical and Engineering Aspects*, 318(1-3) :62–77, 2008.

[Luc77] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82 :1013–1024, 1977.

[Max81] Nelson L Max. Vectorized procedural models for natural terrain : Waves and islands in the sunset. *ACM Siggraph computer graphics*, 15(3) :317–324, 1981.

[MMCK14] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4) :153, 2014.

[MMS04] Viorel Mihalef, Dimitris Metaxas, and Mark Sussman. Animation and control of breaking waves. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 315–324. Eurographics Association, 2004.

[MT15] A Mahdavi and N Talebbeydokhti. A hybrid solid boundary treatment algorithm for smoothed particle hydrodynamics. *Scientia Iranica. Transaction A, Civil Engineering*, 22(4) :1457, 2015.

[MWM87] Gary A Mastin, Peter A Watterberg, and John F Mareda. Fourier synthesis of ocean scenes. *IEEE Computer graphics and Applications*, 7(3) :16–23, 1987.

[NSB13] Michael B Nielsen, Andreas Söderström, and Robert Bridson. Synthesizing waves from animated height fields. *ACM Transactions on Graphics (TOG)*, 32(1) :2, 2013.

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

[Pea86] Darwyn R Peachey. Modeling waves and surf. In *ACM Siggraph Computer Graphics*, volume 20, pages 65–74. ACM, 1986.

[PGT03] M Parenthoën, J Gourrion, and J Tisseau. Les états de mer : un état de l'art. *Actes des 16èmes journées AFIG*, page 129, 2003.

[Phi85] OM Phillips. Spectral and statistical properties of the equilibrium range in wind-generated gravity waves. *Journal of Fluid Mechanics*, 156 :505–531, 1985.

[PJT04] Marc Parenthoën, Thomas Jourdan, and Jacques Tisseau. Ipas : Interactive phenomenological animation of the sea. In *ISOPE 2004*, 2004.

[PMT15] Marc Parenthoen, Fred Murie, and Flavien Thery. The sea is your mirror. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 159–165. ACM, 2015.

[SS17] Alexey Stomakhin and Andrew Selle. Fluxed animated boundary method. *ACM Transactions on Graphics (TOG)*, 36(4) :68, 2017.

[T+01] Jerry Tessendorf et al. Simulating ocean water. *Simulating nature : realistic and interactive techniques. SIGGRAPH*, 1(2) :5, 2001.

[TDG00] Sebastien Thon, J-M Dischler, and Djamchid Ghazanfarpour. Ocean waves synthesis using a spectrum-based turbulence function. In *Computer Graphics International, 2000. Proceedings*, pages 65–72. IEEE, 2000.

[TMFSG07] Nils Thurey, Matthias Muller-Fischer, Simon Schirm, and Markus Gross. Real-time breaking waves for shallow water simulations. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pages 39–46. IEEE, 2007.

[WZC+06] Qiang Wang, Yao Zheng, Chun Chen, Fujimoto Tadahiro, and Norishige Chiba. Efficient rendering of breaking waves using mps method. *Journal of Zhejiang University-SCIENCE A*, 7(6) :1018–1025, 2006.