

SimShip

Edouard Halbert, 2025

SimShip est une simulation de navire naviguant sur un océan infini en temps réel.

Par temps réel, on entend arbitrairement la limite basse de 150 images par secondes en plein écran de 2560 x 1440 (WQHD) sur un ordinateur équipé d'une carte NVIDIA RTX 4070 (micro processeur Intel core I9-13900 K).

Le projet fait suite, des années après, à une approche limitée publiée sur YouTube : ([Basic simulation of ship motion in regular waves](#)). Il s'agissait alors d'un simple démonstrateur centré sur les calculs hydrostatiques. Depuis avril 2012, la vidéo a été vue 133 000 fois.

La simulation fait appel à plusieurs domaines : océan, ciel, soleil, éclairages, nuages, brume, brouillard, caméra, navire, terrain, sons, fumée, interfaces, pilote automatique de navire.

Avant de commencer, le monde de SimShip (la scène) est géo référencé (longitude et latitude). Une unité de 1 dans la simulation vaut 1 mètre. Cela se traduit par le fait que tous les objets de la scène (les vagues, le navire, les bouées, le terrain, etc.) ont le même système de coordonnées. Il n'y a pas de courbure terrestre.

La simulation a été programmée en C++ (norme iso C++20) avec l'éditeur Microsoft Visual Studio Community 2022 qui est gratuit. De nombreuses bibliothèques open source de code ont été utilisées, dont OpenGL (4.3), GLFW, Glad, GLM, Assimp, libigl, ImGui, nanovg, stb, Eigen, FFTW3, Pugixml, Nova, Clipper et OpenAL.

Le but de ce document est d'expliquer les techniques employées dans le code.

OCÉAN

Dans ce domaine de simulation, le papier le plus important est celui de Jerry Tessendorf ([coursenotes2004.pdf](#)). Il s'agit d'un article de 26 pages expliquant la technique employée pour animer un océan en temps réel. Le papier est remarquable mais sa lecture - très importante pour la compréhension de la simulation et des techniques employées - ne permet pas de passer directement au codage.

Pour cela, je me suis largement appuyé sur le blog de Asylum Darth qui offre une vidéo, le code et les explications détaillées de l'implémentation de son océan basé sur le modèle de Jerry Tessendorf. Blog [Ocean rendering](#), vidéo sur YouTube [OpenGL Ocean Rendering](#) et code sur GitHub https://github.com/asylum2010/Asylum_Tutorials.

L'océan est modélisé à partir d'un patch unique. Ensuite 40000 patchs identiques (200 x 200) sont affichés à l'écran pour avoir l'impression d'un grand océan. Ces patchs se déplacent avec la caméra, ce qui donne l'impression d'un océan infini.

Le patch est une grille de 100 m x 100 m (1 hectare) comportant 256 x 256 carrés composés chacun de 2 triangles. Il y a donc 131 072 triangles par patch d'océan (256 x 256 x 2). Chaque point de la

grille a 3 coordonnées x , y , z . Pour se placer dans le référentiel usuel d'OpenGL, y est la hauteur et xz est le plan de l'eau. Sans aucune vague, c'est à dire lorsque l'eau est au repos, y vaut 0. y est positif pour une crête et négatif pour un creux.

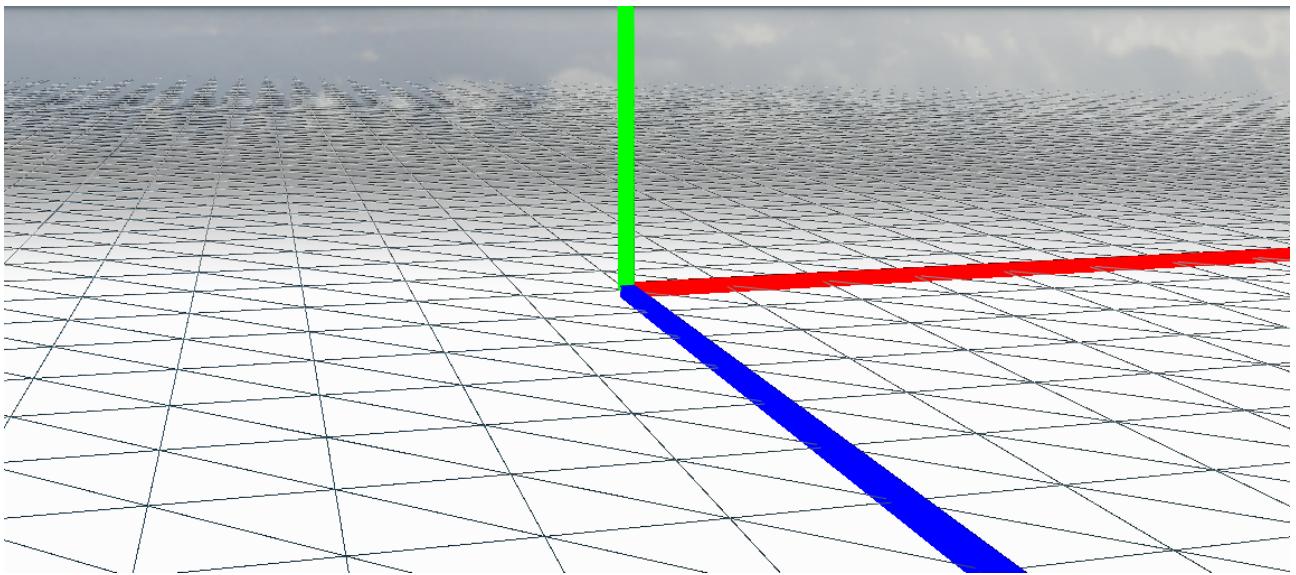


Figure 1: Axe X en rouge, axe Y en vert et axe Z en bleu.

Le modèle de la vague de base est la houle trochoïdale de Gerstner. En dynamique des fluides, la houle trochoïdale est une solution exacte des équations d'Euler qui sont un ensemble d'équations aux dérivées partielles non linéaires qui décrivent le mouvement d'un fluide parfait, c'est-à-dire un fluide idéal sans viscosité ni conductivité thermique. Elles expriment la conservation de la masse, de la quantité de mouvement sous des hypothèses simplifiées. Découverte en 1802 par le baron von Gerstner, ce modèle de vague décrit les ondes de gravité de forme périodique qui se propagent à la surface d'un fluide incompressible de profondeur infinie, en régime permanent. La surface libre de l'écoulement est une cycloïde (ou trochoïde, pour reprendre le terme de Gerstner).

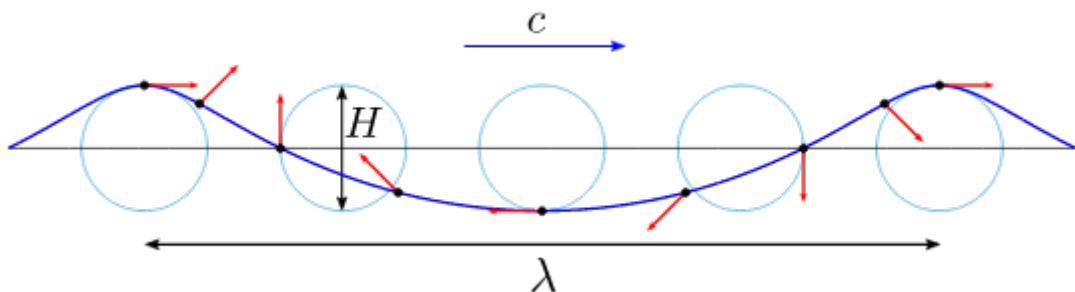


Figure 2: Trochoïde de Gerstner.

Le dessin ci-dessus présente une vague, qu'on va appeler vague de base. Elle est définie par une fréquence (plus ou moins rapide), un décalage dans le temps (la vague commence quelque part sur le trait bleu), et une amplitude (la hauteur entre le bas et le haut du trait bleu). En apparence, la vague de Gerstner est plutôt simple. A la base, c'est une sinusoïde. Pour ajouter de la complexité, on peut sommer des sinusoïdes qui ont différentes fréquences, différentes phases et différentes amplitudes. Pour avoir un aspect réaliste, le patch d'océan est composé de nombreuses vagues de base.

Toutes les vagues de SimShip sont la somme de nombreuses vagues de base dont la fréquence, la phase et l'amplitude sont aléatoires et déterminées au début de la simulation. Ensuite, elles sont mises à jour à chaque image. Ainsi chaque patch comporte 512 x 512 vagues de base définies par leur fréquence, leur phase et leur amplitude.

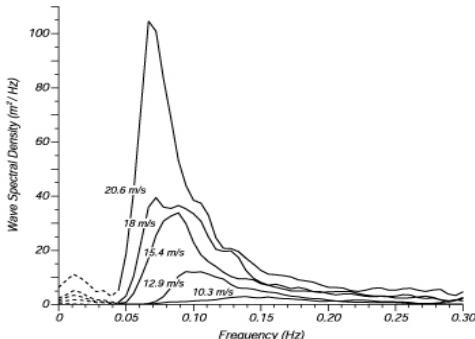


Figure 3: Spectre de vagues.

Les fréquences ne sont pas complètement choisies au hasard. Le spectre de fréquence de ces vagues de base repose sur une analyse des vagues réelles de l'océan. Il existe plusieurs spectres qui diffèrent par les conditions de vent et de fetch (distance sur laquelle le vent souffle sur l'eau). Le spectre retenu est celui de Phillips. C'est celui décrit dans l'article de Jerry Tessendorf. Ce n'est pas nécessairement le plus réaliste mais il est très agréable à regarder.

Après de nombreux essais, il apparaît qu'aucun spectre de vagues ne modélise bien les vagues pour tous les vents allant de 1 kt (calme) à 60 kt (tempête). Sauf à multiplier les patchs. Ainsi certaines simulations évoluées utilisent 4 patchs pour arriver à modéliser le patch, ces 4 patchs ayant des fréquences très différentes. Un patch simule des vagues longues, un autre des vagues plus courtes, etc. Le problème est que la multiplication des patchs diminue le taux d'images par seconde, sachant que la modélisation du bateau réclame aussi du temps de calcul. Pour le moment, SimShip n'embarque qu'un spectre pour un patch.

Le spectre de Phillips a été un peu modifié dans cette simulation pour se concentrer sur les vents de 1 kt à 20 kt, ceux rencontrés le plus fréquemment dans la réalité. Au delà, les vagues sont moins représentatives de ce qu'elles sont dans la réalité.

La constitution du patch à chaque rafraîchissement de l'écran utilise la transformée de Fourier rapide (FFT) qui est un algorithme qui permet de passer efficacement du domaine fréquentiel au domaine spatial (ou temporel). Sans rentrer dans les détails, il s'agit de mettre à jour à chaque image la totalité des 262 144 vagues de base (512 x 512 trochoïdes) dans le domaine des fréquences à l'aide de la transformée de Fourier. La transformée de Fourier inverse (iFFT) permet de reconstruire la surface océanique spatiale (ce que l'on voit) à partir du spectre en fréquence (ce qui évolue à chaque image), ce qui est plus rapide que la sommation individuelle des vagues.

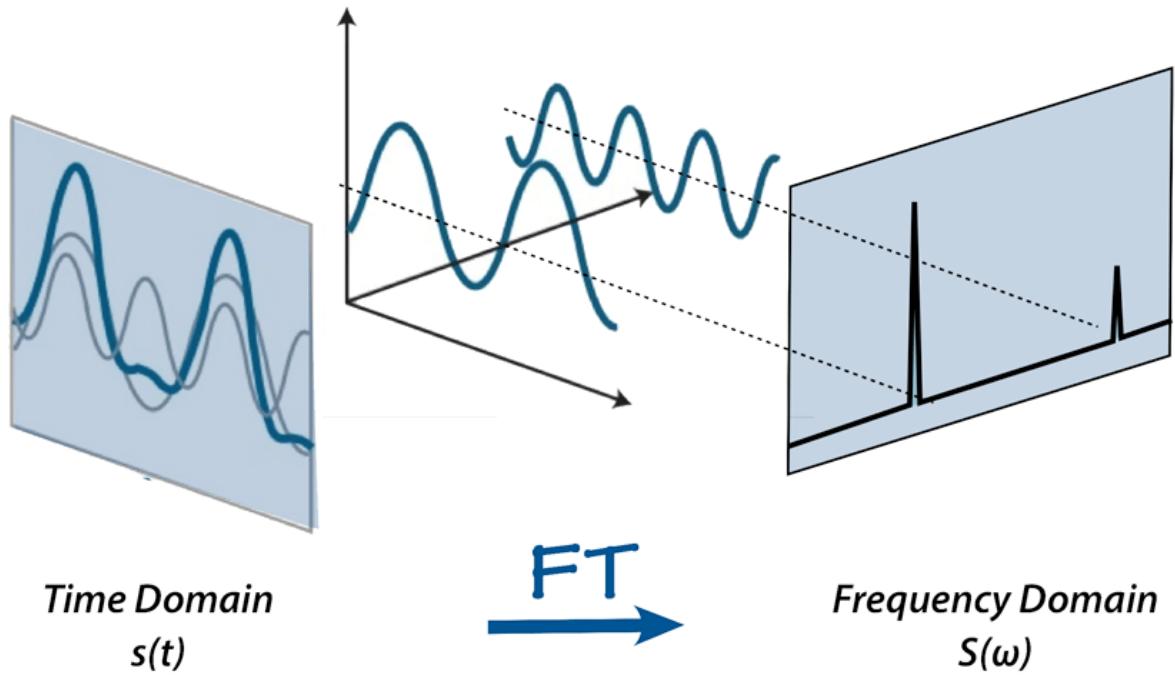


Figure 4: La transformée de Fourier permet de passer d'un domaine à l'autre.

A l'issue de ce processus, on obtient un petit déplacement dx , dy , dz pour chaque point de la grille (256 points x 256 points pour un patch de 100 m x 100 m). Le déplacement en hauteur dy simule les crêtes et les creux des vagues tandis que les déplacements dx et dz simulent le mouvement horizontal de l'eau.

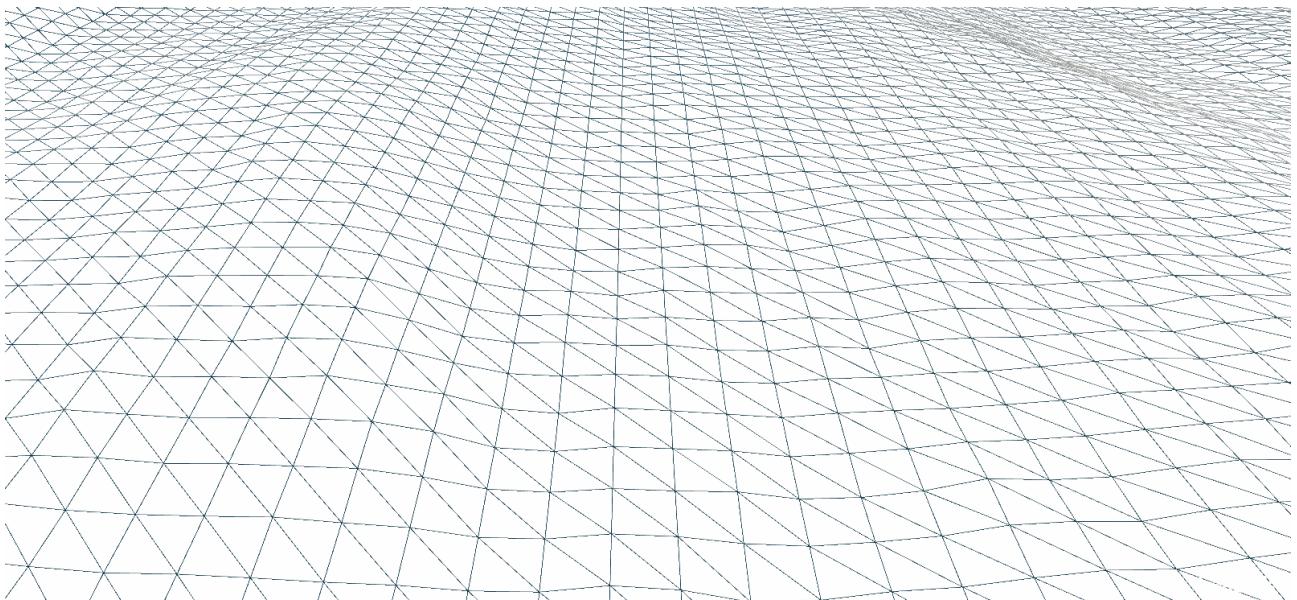


Figure 5: Sans déplacement dx et dz , les lignes sont parallèles.

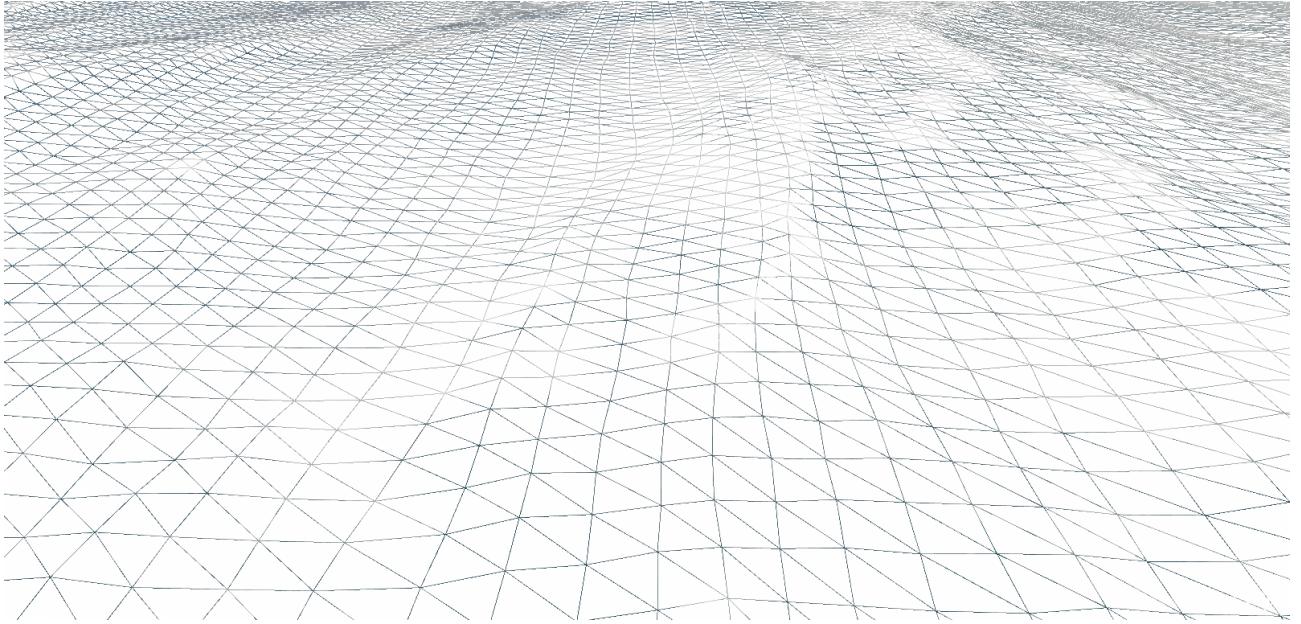


Figure 6: Avec déplacements dx et dz , les lignes ondulent.

A noter que la transformée de Fourier peut être qualifiée de rapide (Fast Fourier Transform) si elle est appliquée sur un nombre fini d'échantillons d'un signal, avec des fréquences discrètes (discontinues). Elle utilise une astuce qui est de ne calculer que la moitié de la grille (avec une diagonale séparant les deux moitiés), l'autre moitié étant déduite en miroir de la première. Ceci a un impact pour le patch de l'océan, à savoir que le bord gauche du patch est égal au bord droit et que le bord avant est égal au bord arrière. Ainsi, quand on place un patch à côté d'un autre, comme il s'agit du même patch et que les bords sont égaux deux à deux, les patchs sont jointifs. Par contre, ils dessinent à l'écran les mêmes vagues qui forment un dessin répétitif, surtout si les vagues sont grosses avec de l'écume. C'est la limite de cette approche.

La FFT effectue de très nombreux calculs mais qui peuvent être effectués en même temps, c'est à dire en parallèle. Pas besoin d'attendre qu'un calcul soit fini avant de faire le suivant. Là où le micro-processeur d'un ordinateur effectue opération par opération, les cartes graphiques modernes possèdent de nombreux petits processeurs qui peuvent faire des opérations en parallèle. Par exemple, une carte graphique NVIDIA RTX 4070 possède 5888 coeurs CUDA qui sont les unités principales de calcul parallélisé pour le traitement des données en virgule flottante. Donc, tous les calculs du patch sont réalisés par la carte graphique. Cela nécessite un langage de programmation particulier, GLSL (OpenGL Shading Language) dans le cas de SimShip. GLSL est un langage de programmation de haut niveau conçu pour écrire des « shaders » dans l'environnement OpenGL. Les shaders sont des petits programmes exécutés directement sur le GPU (Graphics Processing Unit, la carte graphique) et permettent un contrôle précis et performant du rendu graphique : éclairages, ombres, textures, effets visuels avancés, etc. Les Vertex shaders traitent chaque sommet (point) d'un modèle 3D (un mesh) pour le transformer, de l'espace 3D du modèle à l'espace 3D de la scène puis à la surface 2D de l'écran. Les Fragment shaders (ou pixel shaders) exécutent pour chaque fragment (pixel potentiel), le calcul de couleur, de transparence, de texture, de lumière, etc. C'est ce qui donne le look final à chaque pixel de l'image. Enfin il existe des Geometry shaders, des Tessellation shaders et des Compute shaders qui sont des étapes supplémentaires permettant des manipulations plus poussées de la géométrie ou du calcul général, mais moins courants que les vertex et fragment shaders.

Le patch d'océan de SimShip utilise massivement les Compute shaders. On passe les données des trochoïdes à la carte graphique. On lui passe aussi le code de calcul (les Compute shaders), c'est à dire l'algorithme de mise à jour de ces sinusoïdes et la carte graphique calcule en parallèle tout ça. A la fin, on obtient le résultat (les déplacements dx, dy, dz) sur une image où chaque pixel représente ces valeurs. On obtient aussi le gradient entre les points de la grille du patch, ce qui permet de déterminer si de la mousse doit être dessinée à l'endroit où les vagues sont pentues.

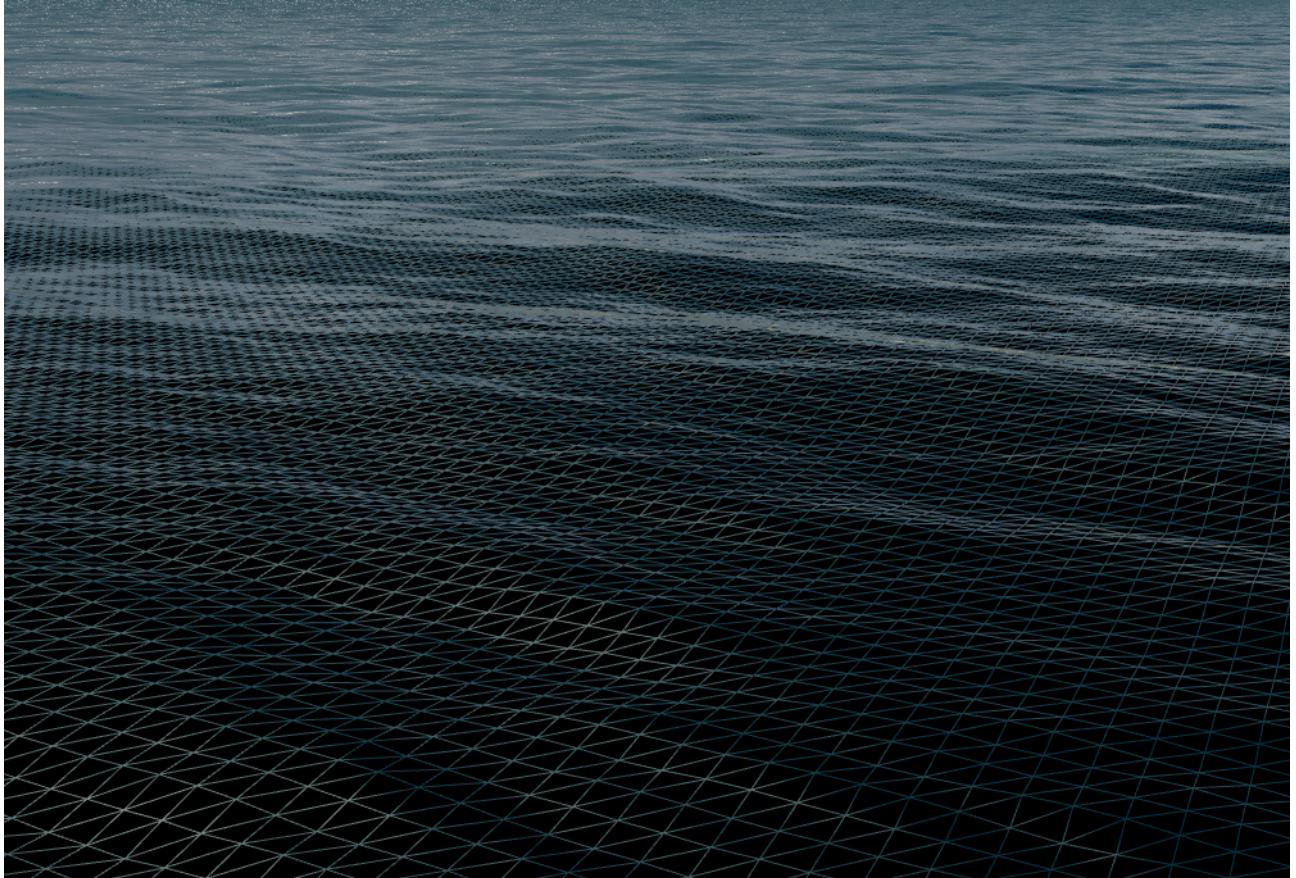


Figure 7: Rendu du patch.

Une fois qu'un patch a été obtenu, 40000 instances de ce patch sont générées par OpenGL et placées autour de la position de la caméra. Seuls les patchs visibles depuis la vue caméra sont dessinés (Frustum culling). Le Frustum culling est une technique d'optimisation en infographie 3D qui consiste à ne pas dessiner les objets situés en dehors du cône de vision de la caméra, appelé le frustum.

Les patchs qui apparaissent à l'écran ont un niveau de détail différent (LOD – level of detail) selon leur distance à la caméra. Le LOD est dit adaptatif car il est mis à jour à chaque image. Les patchs très proches sont dessinés sans perte d'information (donc avec une grille de 256 points x 256 points (représentant 100 m x 100 m) mais ceux très éloignés n'ont que 16 points x 16 points (toujours pour 100 m x 100 m) et entre, il y a des grilles de 128 x 128, 64 x 64 et 32 x 32.

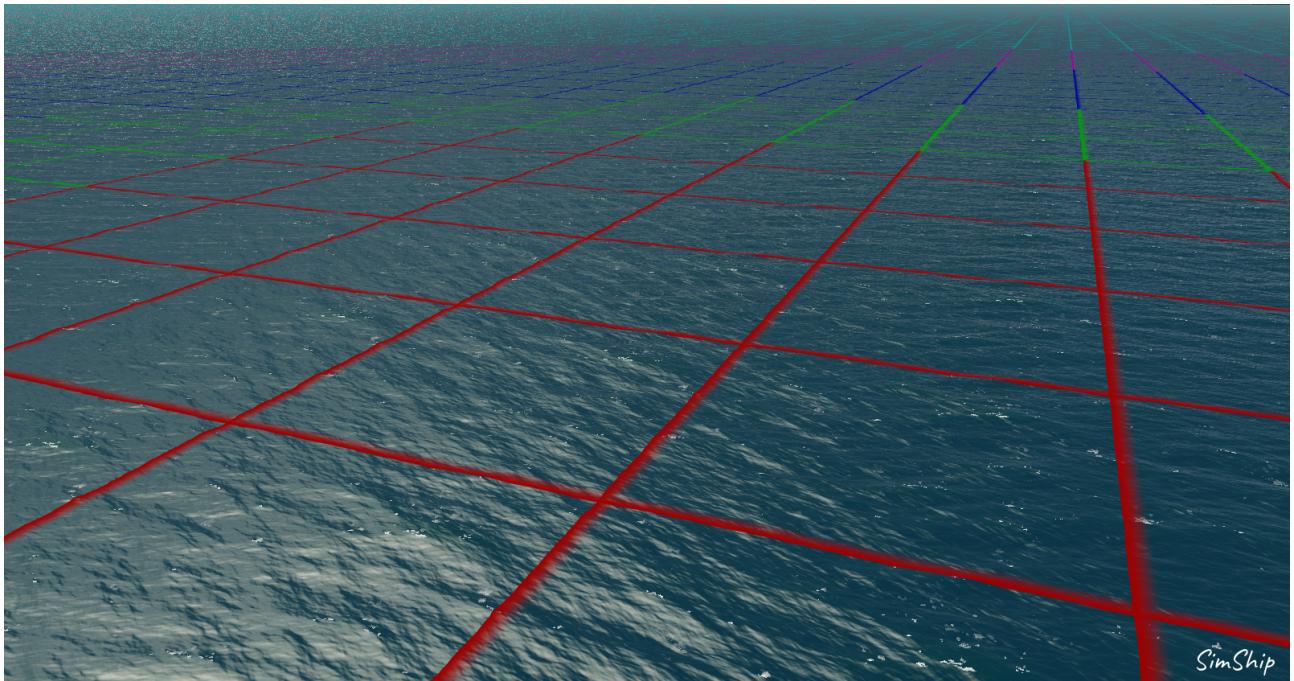


Figure 8: Visualisation des patchs par LOD.

Pour les patchs pour lesquels le navire et son sillage sont dessus, un rendu particulier a lieu pour le sillage (expliqué plus tard).

Voici pour la géométrie de l'océan. On a des points qui se déplacent et qui représentent les vagues. Pour chaque image, 512×512 sinusoïdes sont mises à jour par la carte graphique avec pour résultat les déplacements des points de la grille (dx , dy , dz) et les gradients qui informent sur les vagues qui déferlent. Puis on prend ce patch, et on le duplique avec plus ou moins de définition (LOD) pour avoir un océan visible sur plusieurs dizaines de kilomètres (jusqu'à l'horizon).

Ensuite, il faut éclairer convenablement les vagues pour leur donner l'aspect de l'eau. C'est aussi fait avec les shaders. Les vertex shaders prennent les points et les positionnent correctement à l'écran selon la vue de la caméra. Puis les fragment shaders fabriquent la couleur et la transparence qui sera appliquée à chaque point. Et à la fin, on a un océan animé, à raison de plus de 300 images par secondes, en plein écran. C'est la carte graphique qui fait tout le boulot.

L'éclairage des vagues est complexe. Il dépend de la position du soleil et de sa lumière. L'eau a beaucoup de réflexions, celle du soleil, celle du navire. Il y a aussi de la mousse ajoutée selon le gradient des points vus précédemment pour représenter le déferlement des vagues. Il y a aussi la transparence qui évolue selon l'angle de vision. Plus on regarde verticalement et plus l'eau est transparente. Le modèle physique utilisé pour l'éclairage de l'océan est un modèle BRDF (Bidirectional Reflectance Distribution Function) associé au modèle de Ward et à la réflexion de Fresnel.

Un mot sur la mousse dont le calcul repose sur les jacobiens de la surface de l'eau. Un jacobien est la matrice des dérivées partielles d'une fonction vectorielle qui décrit localement comment cette fonction change. Pour estimer la pente des vagues dans un shader, on applique le jacobien à la fonction de déplacement de la surface océanique : il permet de calculer les dérivées spatiales de la hauteur des vagues (les variations locales de la surface). Ces dérivées fournissent la pente en chaque point, qui sert à déterminer où placer la mousse, par exemple sur les zones à forte inclinaison ou

déferlement. Ainsi, le jacobien transforme l'information de la déformation de surface en un vecteur pente exploitable pour l'apparence dynamique des vagues dans le shader. C'est suffisant pour une représentation simple de l'océan car en se limitant à cette technique, il manque l'eau qui déferle. Ce serait faisable avec des particules.

Enfin, l'océan peut présenter une qualité cinématographique mais être physiquement irréaliste. Il importe donc de vérifier que la période des vagues, leur hauteur significative (celle des 1/3 supérieures) est similaire aux données observées en situation réelle. SimShip permet l'analyse spectrale des vagues. Pour cela, il prend les hauteurs d'eau successives à chaque image au point (0, 0) de la scène (là où on peut faire apparaître un axe rouge-vert-bleu. Avec la suite des hauteurs d'eau, il détermine – encore avec des FFT – la distribution des fréquences et leur densité, à partir desquelles il donne les caractéristiques des vagues.

CIEL

Le ciel est aussi une partie bien plus complexe qu'on l'imagine de prime abord.

Il y a d'abord la source lumineuse représentée par le soleil. Il a une position dans le ciel donnée par l'emplacement de la vue (la caméra), la date et l'heure. Dans SimShip, la position initiale de la caméra est celle de l'île de Houat au sud de la Bretagne en France. Selon la position, la date du jour et de l'heure réelle (solaire), l'azimut (direction horizontale) et l'élévation (angle par rapport à l'horizon) du soleil sont calculés.

Ensuite, ces 2 angles vont donner la couleur de la lumière, orange quand le soleil est bas sur l'horizon et quasi blanche quand le soleil est au zénith.



Figure 9: Exemple de ciel, au petit matin.

Ensuite, le ciel est calculé selon l'ensemble des effets lumineux générés par la diffusion, la réflexion, la réfraction et l'absorption de la lumière solaire dans l'atmosphère terrestre. Le code de calcul est issu de celui de Eric Bruneton ([precomputed atmospheric scattering](#)). Son modèle atmosphérique est une méthode avancée de rendu en temps réel de l'atmosphère terrestre, prenant en compte l'interaction complexe de la lumière avec les constituants de l'atmosphère comme les

molécules d'air, les particules d'aérosols et l'ozone. Ce modèle simule notamment la diffusion multiple de la lumière, intégrant les effets de diffusion de Rayleigh et de Mie, pour restituer de manière réaliste les couleurs du ciel, y compris les variations diurnes (jour, crépuscule) et les effets d'aérosols. Là encore, une fois déterminés les 2 angles (azimut et élévation), c'est la carte graphique qui est sollicitée à l'aide d'un compute shader. Le résultat est une image de la dimension de la fenêtre SimShip qui représente le ciel avec un horizon et le dégradé de couleurs.

Après la position du soleil et le ciel, il faut composer les nuages. Il y a deux manières génériques d'aborder le sujet. L'approche la plus simple et peu coûteuse en temps de calcul consiste à dessiner des motifs aléatoires qui ressemblent à des nuages sur une image puis à appliquer cette image au dessus de la caméra. Cela fonctionne mais les nuages manquent « d'épaisseur » et cela se voit. L'autre technique, bien plus complexe et plus coûteuse en temps de calcul consiste à modéliser des nuages en 3D, appelés nuages volumétriques. Quelques exemples de code peuvent être trouvés sur Internet. Le code source retenu pour SimShip est celui de Federico Vaccaro ([TerrainEngine-OpenGL](#)). Lui-même s'est inspiré de nombreuses sources (voir son site). Ces nuages sont volumétriques, ils se déplacent dans le sens du vent et évoluent d'une manière très réaliste. Sans entrer dans les détails, le calcul fait appel à des images en 3 dimensions (textures 3D) et à la technique du ray-marching qui procède en avançant un rayon de manière itérative par petits segments le long de sa trajectoire. C'est très coûteux en temps de calcul ... mais c'est plutôt réussi. Là encore, c'est la carte graphique qui fait tout le boulot.

Enfin, la brume et le brouillard sont des modèles exponentiels qui correspondent à une extinction lumineuse qui décroît exponentiellement avec la distance parcourue dans la masse d'eau en suspension (gouttelettes microscopiques). A la différence du brouillard, la brume est contenue autour de l'horizon.



Figure 10: Brume de beau temps.

Donc pour résumer cette partie, on calcule la position du soleil, on fabrique l'atmosphère avec ses dégradés de couleur selon l'heure du jour, on applique des nuages volumétriques puis de la brume ou du brouillard éventuellement. Et on applique aussi à l'ensemble une exposition pour simuler la nuit, l'aube et le crépuscule.



Figure 11: Nuages volumétriques.

CAMERA

Avant d'aborder le navire qui est un gros morceau, il fait dire quelques mots sur la caméra. Une bonne simulation nécessite une bonne caméra. C'est elle qui permet de se déplacer dans la scène. Ainsi la caméra offre plusieurs modes : un mode orbital où on tourne autour du navire, un mode libre (first person shoorter) et des vues depuis la passerelle où quelque part sur le pont, selon les navires. Ensuite, les déplacements doivent se faire très progressivement avec la souris ou le clavier. Il y a ainsi plusieurs modes d'interpolation. Il y a aussi la possibilité de zoomer comme si on avait des jumelles. Enfin, la classe Camera fournit l'ensemble des matrices et des vecteurs nécessaires aux différents calculs de la simulation.

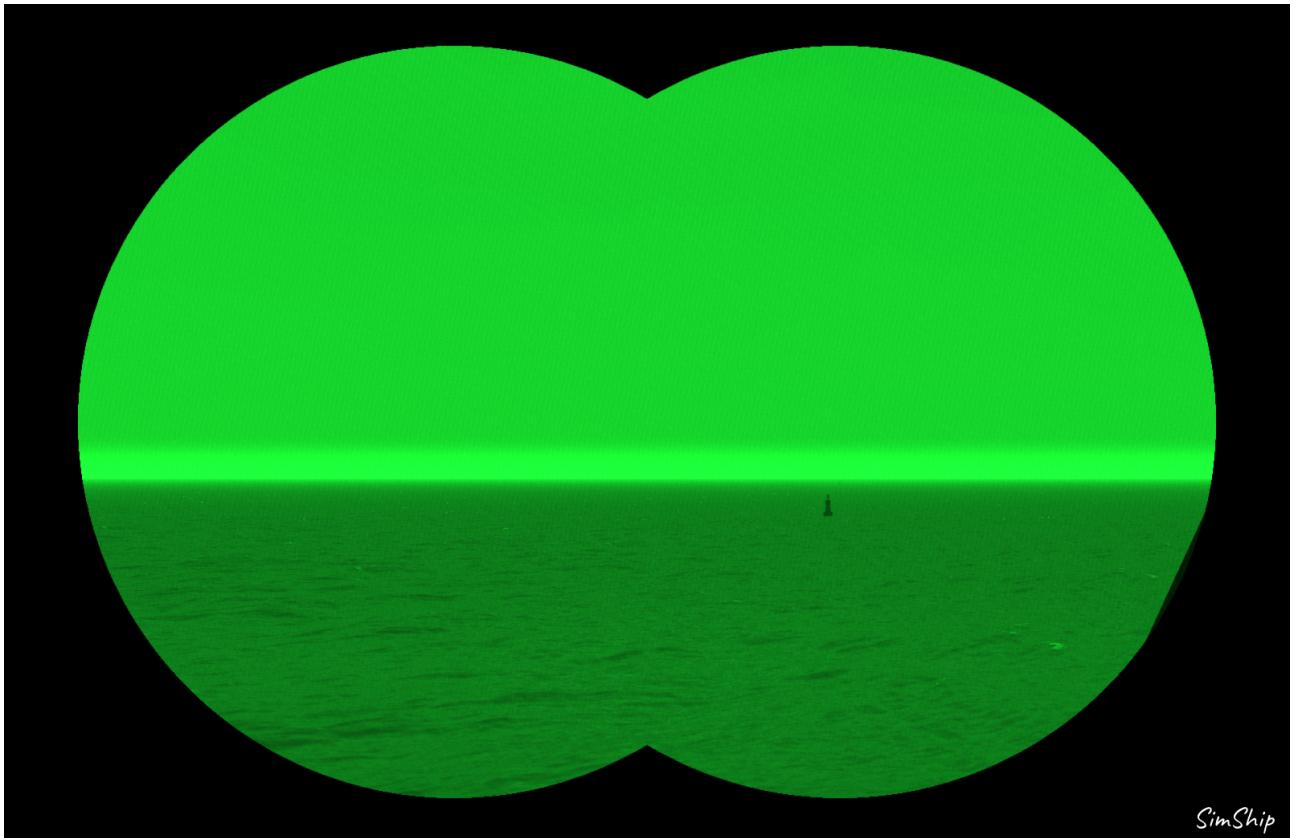


Figure 12: Jumelles en vision nocturne.

NAVIRE

Là encore, c'est une partie assez complexe qui fait appel cette fois au microprocesseur pour les calculs (CPU, Central Processing Unit) et assez peu à la carte graphique.

Le navire doit être vu à l'écran (de préférence joli) et se mouvoir en fonction des vagues (de préférence réaliste). Autant le dire tout de suite, ces 2 qualités sont antagonistes : plus un bateau est beau à l'écran et plus il est détaillé (son modèle 3D est composé de centaines de milliers de faces) alors que pour faire les calculs de flottabilité, le nombre de faces (triangles) est un handicap. Moins il y en a (jusqu'à une certaine limite) et plus les calculs sont rapides. A noter que tout l'affichage à l'écran d'une scène 3D repose sur des triangles à afficher sur lesquels on plaque une image appelée texture. Ainsi, le mot triangle est préféré à face (une face peut avoir 4 sommets ou plus).

Le navire peut avoir des parties animées comme un ou plusieurs radars, un ou plusieurs gouvernails et une ou plusieurs hélices. Ainsi, la simulation doit comporter autant de fichiers 3D que de parties à animer. Exemple, un navire a un radar, un gouvernail et une hélice. Cela fait 4 fichiers : un fichier pour le navire sans son radar, son gouvernail et son hélice, un fichier pour son radar qui tourne, un fichier pour son gouvernail qui pivote et un fichier pour son hélice qui tourne. Mais il y a aussi – et c'est peut-être une des spécificités de SimShip – un fichier uniquement pour la coque (rien que la coque). Ce fichier est utilisé pour calculer sa flottabilité. Avoir un fichier séparé du modèle à afficher est nécessaire car les calculs de flottabilité ne concernent que la coque. Ensuite, il y a un nombre optimal de triangles pour les calculs. Par exemple, si un tanker a un important tronçon de coque au milieu qui ne présente pas d'évolution de la forme, le modèle 3D aura peu de triangles à cet endroit. Avec la puissance des ordinateurs actuels, 2000 à 8000 triangles semblent un bon compromis. Plus de triangles et le mouvement du navire dans les vagues est parfait mais la

simulation est saccadée. Moins de triangles et c'est le mouvement du navire qui paraît saccadé (quand un de ses gros triangles entre ou sort de l'eau).

Il y a donc des modèles 3D à afficher pour chaque navire (fichiers *.glb ou *.gltf) et un modèle 3D (fichier *.obj sans fichier *.mtl de matériau) pour les calculs.

Pour les modèles à afficher, le format glTF a été privilégié (GL Transmission Format) ? C'est un format ouvert et léger destiné au stockage et à l'échange de scènes et modèles 3D. Il utilise le format JSON pour décrire la structure de la scène, les matériaux, les animations et les géométries, facilitant un chargement rapide et une bonne interopérabilité. Le format GLB est la version binaire compacte de glTF, regroupant dans un seul fichier toutes les données JSON, binaires et images, ce qui simplifie la gestion et le téléchargement des modèles 3D.

Au chargement du navire dans la simulation, SimShip ouvre le fichier *.obj du navire afin de déterminer tous les sommets et tous les triangles du navire. Le format obj est un format de fichier texte standard utilisé pour décrire la géométrie des modèles 3D. SimShip utilise la version minimaliste de ce type de fichier. Il stocke uniquement les coordonnées des sommets et les faces définies par les indices des sommets. Aucun autre fichier de définition des textures ou des matériaux, possible avec ce format de fichier, est nécessaire.

A chaque rafraîchissement de l'écran, l'océan est recalculé pour donner les dx, dy, dz de tous les points du patch. Ce patch peut être répété car les bords sont jointifs ce qui signifie que si un navire paraît être sur 2 patchs, il n'est en fait toujours que sur l'unique patch.

A chaque rafraîchissement de l'écran, tous les sommets des triangles du navire sont évalués pour savoir si ils sont sous l'eau ou au dessus. Cela détermine la liste des triangles de la coque du navire qui sont immersés, émergés ou partiellement immersés. Pour chaque triangle immersé (ou partiellement), on détermine la force de pression hydrostatique qu'il subit, connaissant la surface du triangle et la hauteur de la colonne d'eau au dessus de lui. En sommant toutes ces forces hydrostatiques, on détermine la poussée d'Archimède et le centre du volume de carène (le centre de flottaison où s'applique la poussée d'Archimède). Quand on combine la force de gravité connue par la masse du navire et la position du centre de gravité avec la force d'Archimède, on obtient une nouvelle force qui traduit le déplacement du navire verticalement quand on la rapporte à son inertie. De même pour le roulis et le tangage, associés à leurs inerties respectives, transversale et longitudinale.

Une complication importante intervient pour le calcul des hauteurs d'eau au dessus des sommets. Si le patch avait toujours des points fixes, sans déplacements dx et dz, ce serait facile de faire une interpolation car la grille du patch est régulière. Quand on ajoute aux points de la grille les déplacements dx et dz, la grille devient irrégulière. L'interpolation classique est impossible car on se sait pas sous quel triangle interpoler. Il faut alors faire plusieurs itérations pour déterminer quel triangle de l'eau doit interpolé. En général, il faut 5 à 6 itérations par sommet de la coque. Or la coque comporte souvent entre 2000 et 8000 sommets (à faire 5 à 6 fois à chaque image).

Aussi, à noter que l'intersection des triangles du navire avec l'intersection des triangles de la mer présente peu d'intérêt. Cela est très coûteux en temps et apporte peu, même si c'est normalement physiquement juste. L'astuce simplificatrice consiste à affecter un coefficient à la pression hydrostatique d'un triangle partiellement immersé selon le nombre de points immersés. Cette approximation est lissée par le nombre élevé de triangles de la coque.

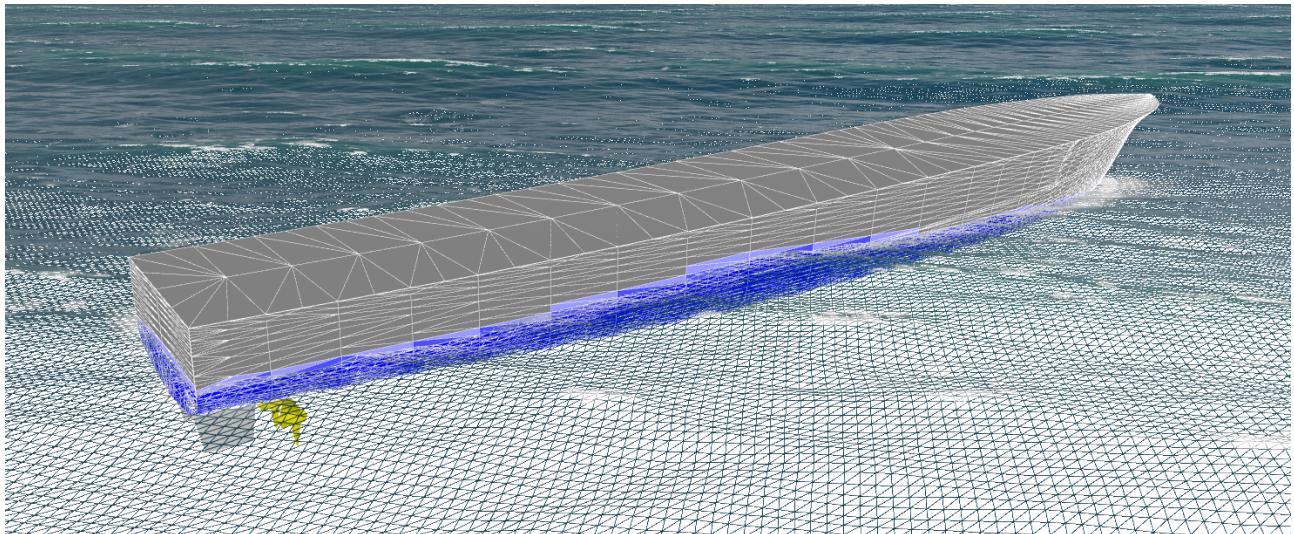


Figure 13: Les triangles colorés en bleu sont immersés.

A ces 2 forces essentielles (Archimède et gravité), sont ajoutées la force produite par l'hélice entraînée par le moteur, la force produite par le propulseur d'étrave le cas échéant, les forces du gouvernail (poussée latérale et résistance à l'avancement), la force de viscosité dépendant de la vitesse du navire et de sa surface mouillée, la force de résistance des vagues qui croît rapidement avec la vitesse du navire et la force centrifuge qui fait incliner le navire dans un virage. La plupart de ces forces ne s'appliquent pas au centre du navire. Ainsi, les navires ont un point pivot situé la plupart du temps vers l'avant du navire quand il avance et vers l'arrière quand il recule. A noter qu'il y a aussi deux autres forces modélisées, liées à l'action du vent, une force de dérive du navire sous le vent et une force de rotation qui amène au bout d'un moment le navire travers au vent.

Au total, toutes ces forces confèrent au navire 6 degrés de liberté, les 3 translations x, y, z (Surge, Sway, Heave) et les 3 rotations autour des axes x, y et z (Roll, Yaw, Pitch).

Enfin, un navire sur l'eau crée des vagues, de l'écume autour de lui et dans son sillage.

Pour les vagues créées par le navire, c'est un domaine connu depuis longtemps. Les vagues de sillages sont appelées les vagues de Kelvin (lord Kelvin) et elles ont été décrites et expliquées par lui en 1887. On peut assez facilement les mettre en équation. Le seul problème de taille est que ces équations ne sont pas solvables pour une simulation en temps réel. Pour avoir des vagues un peu réalistes, il faut compter plusieurs secondes/minutes sur un ordinateur personnel rapide. Il y a quelques exemples de code sur Internet mais ils sont valides sous certaines conditions (comme le nombre de Froude). La solution choisie pour SimShip repose sur l'application d'images pré calculées selon le nombre de Froude (qui est la vitesse du navire rapportée à sa longueur). 100 images sont ainsi calculées en haute définition (pour un temps de calcul avoisinant la demi-heure) pour un nombre de Froude allant de 0,01 (navire très lent) à 1,00 (navire très rapide). La plupart des navires de SimShip ont des nombres de Froude allant de 0.0 à l'arrêt à 0.41 à vitesse maximale. Un seul bateau, une vedette navire de 43 pieds, a une nombre de Froude à 0.95.



Figure 14: Navire militaire rapide avec vagues multiples, latérales et transversales.



Figure 16: $Froude = 0.20$

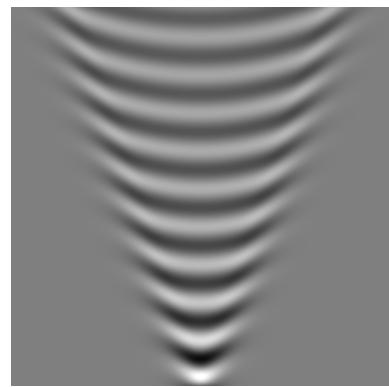


Figure 15: $Froude = 0.41$

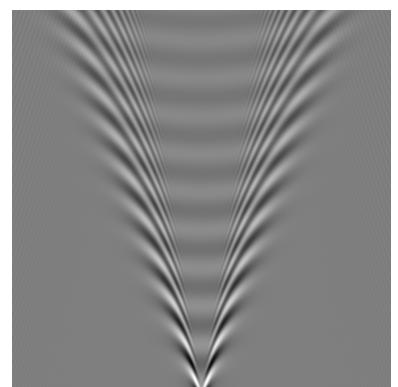


Figure 17: $Froude = 0.95$

L'écume et le sillage sont modélisés de deux façons différentes. Pour l'écume autour du navire, un contour de la coque est créé au chargement du navire dans la simulation. Il s'agit de l'intersection de tous les points de la coque avec l'eau (plan $xz = 0$). Ensuite, les intersections sont ordonnées et un contour fermé est produit, contour qui est un peu élargi d'une manière uniforme (non scalaire) pour dépasser le navire à l'extérieur. Ce contour sert à fabriquer une image de mousse. Cette image est appliquée à la position du navire à chaque image.

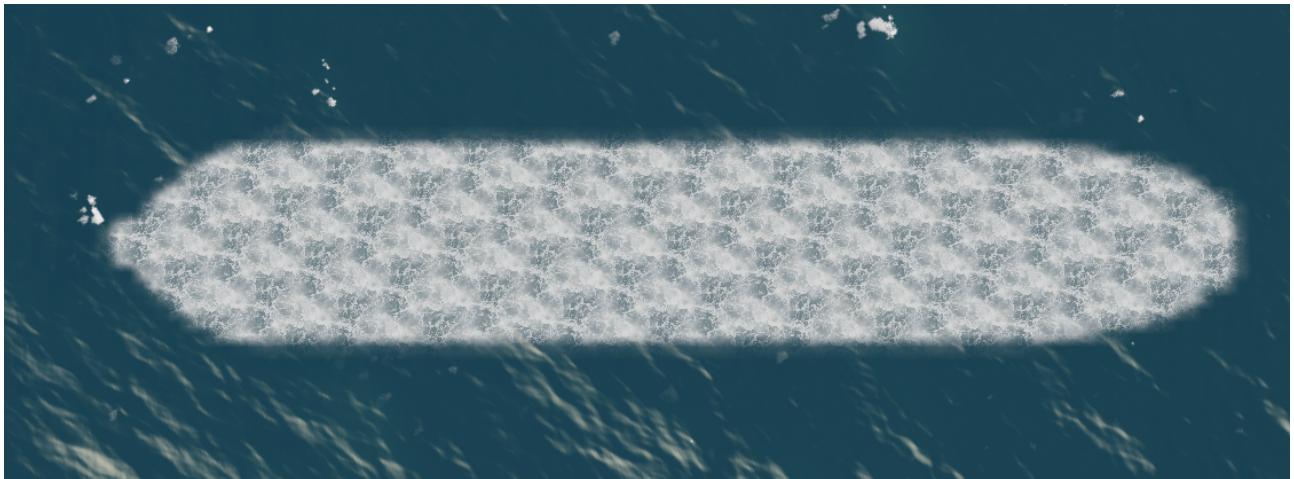


Figure 18: Texture de mousse créée avec le contour de la coque.

Pour la mousse du sillage, la technique retenue est différente. Compte tenu des coûts possibles de calculs, la technique retenue est un peu particulière. Toutes les x images (disons 100), un point – situé vers l’arrière du navire – est ajouté à une liste. Cette liste de points sert à fabriquer un polygone constitué de triangles qui représente vue de dessus le sillage du navire. Ensuite ce polygone sert à constituer une image avec de la mousse, mousse qui sera plus transparente selon l’heure où a été enregistré le point de ce triangle. Plus le point date et plus la mousse disparaît. Un court schéma vaut mieux qu’un long discours :

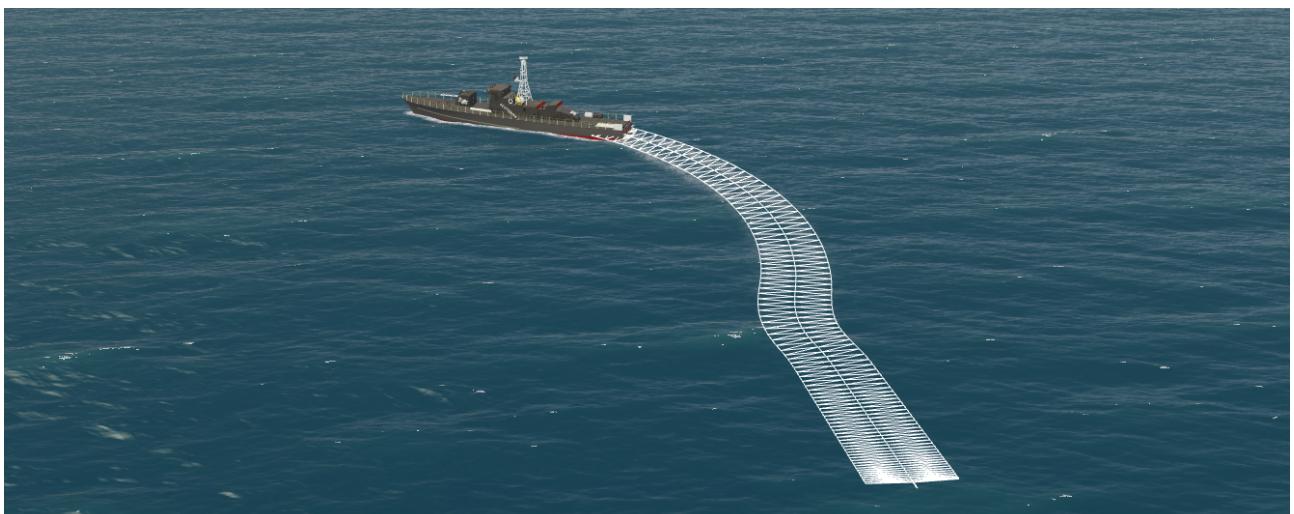


Figure 19: Polygone à partir de points laissés dans le sillage du navire.

Enfin, le navire est éclairé avec la couleur du soleil et selon sa position dans le ciel. Le modèle d’éclairage est un modèle complexe physiquement réaliste. Aux animations des radars, des hélices et des gouvernails s’ajoute la fumée produite par le moteur du navire.

La fumée est un système de particules faisant appel à la carte graphique. Chaque particule de fumée est émise depuis une des cheminées du navire (position, vitesse, durée de vie et couleur). 10000 particules sont calculées à chaque image selon le vent et sa durée de vie et affichées pour simuler la fumée.

L'écume à l'étrave du navire procède du même système de particules. 40000 particules sont émises depuis une ligne bordant l'avant du navire (contour). Leur vitesse dépend du tangage du navire, notamment quand il retombe lourdement dans l'eau et bien sûr de la vitesse du navire.

Last but not least, un mot sur les navires. SimShip en propose 10 à ce jour. Ils représentent un éventail de navires, du petit au gros. Seuls des modèles 3D téléchargeables gratuitement depuis Internet sont proposés (et avec une licence le permettant). Ensuite, pour qu'un modèle soit accepté dans la simulation, il faut réunir plusieurs conditions : 1) que le modèle soit joli, 2) qu'il comporte un nombre raisonnable de triangles, 3) qu'avec le logiciel gratuit Blender, le modèle puisse être travaillé.



Figure 20: Torpilleur, 54 m.



Figure 21: Tanker, 273 m.



Figure 22: Cargo vraquier, 91 m.



Figure 23: Navire de pêche, 21m.



Figure 24: Navire LNG, 284 m.



Figure 25: Navire de pêche, 19 m.



Figure 26: Frégate chinoise type 054A, 135 m.



Figure 27: Navire de soutien, 51 m.



Figure 28: *Grand Banks 43, 13 m.*



Figure 29: *Sous-marin Jang Bogo, 54 m.*

INTERFACE DE CONTRÔLE

L'interface de contrôle regroupe plusieurs éléments :

- les paramètres généraux de la scène, les paramètres du vent (direction et vitesse), l'heure du jour, le ciel, les nuages, l'océan pour lequel on peut par exemple choisir plusieurs couleurs, les paramètres du bateau ;

- le contrôle proprement dit du navire (vitesse, cap sur l'eau et sur le fond, taux de rotation, angles du gouvernail, puissance moteur et propulseur d'étrave, taux de roulis et de tangage) ;
- le contrôle du pilote automatique.

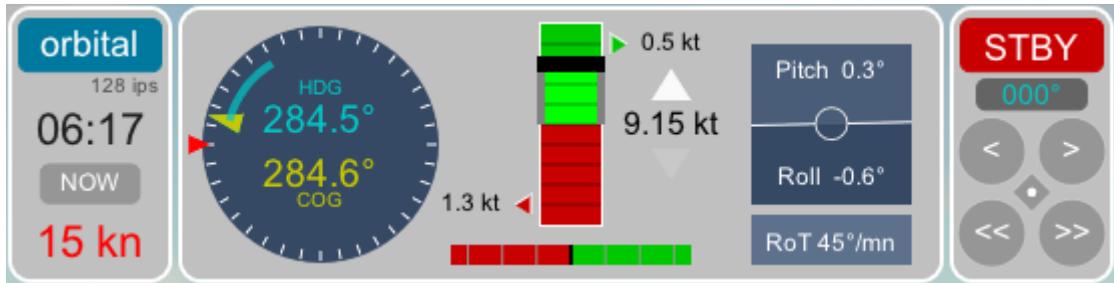


Figure 30: Interface : Système - Navire - Pilote automatique.

PILOTE AUTOMATIQUE

Le navire pivote selon le vent et a une rotation induite par le roulis, de sorte que laisser la navire avec le gouvernail au centre ne fera pas aller le navire tout droit. Il faut compenser ces effets par une légère action sur le gouvernail et cette compensation doit être permanente selon les conditions de mer et de vent. Ainsi, le pilote automatique qui maintient un cap constant (ou quasi constant) à toute sa place à bord des navires.

L'implémentation du pilote utilise un contrôleur PID (Proportionnel-Intégral-Dérivé) pour calculer l'angle de barre optimal. La fonction calcule d'abord l'erreur de cap. La composante proportionnelle (P) réagit directement à l'erreur actuelle. La composante intégrale (I) accumule l'erreur au fil du temps, contribuant ainsi à corriger les erreurs persistantes. La composante dérivée (D) anticipe les changements futurs en fonction de la vitesse de variation de l'erreur. L'angle de barre final est la somme de ces trois composantes, limitée à l'angle maximal autorisé.

A ces 3 paramètres, s'ajoute possiblement (option dans SimShip) un ajustement dynamique qui modifie la valeur de ces 3 paramètres en fonction des conditions.

TERRAIN

Pour ajouter un élément d'ambiance, un terrain a été ajouté. Il s'agit de l'île de Houat situé en Bretagne sud. L'île mesure 3,3 km de long et 1,5 km au plus large.

Un programme séparé de SimShip a été créé afin de constituer un certain nombre de terrains. Ce programme identifie au sein du fichier mondial des contours de côtes ([coastlines-split-4326.zip](#)) le contour de l'île puis sélectionne les bonnes élévations de terrain au sein de fichiers de données de terrain haute définition produit par l'institut géographique national français (IGN) pour réaliser un modèle 3D grâce à la photo satellite elle aussi produite par l'IGN. Le tout forme un fichier de modèle 3D lu par SimShip.

BALISAGE

Le balisage (les bouées) fait lui aussi l'objet d'un programme dédié. Selon une zone, la liste des balises est extraite d'un catalogue national des balises françaises. Le fichier de balises produit est un fichier xml lu par SimShip qui positionne une bouée générique (rouge, verte, cardinale, danger isolé) dans la scène.

SONS

Seuls certains sons sont présents, le son du moteur du navire, le son de son éventuel propulseur d'étrave, le son de la corne de brume du navire et le son des goélands. Tous les sons sont spatialisés. Ils diminuent lorsqu'ils sont éloignés et ils passent d'un côté à l'autre selon la vue.

AFFICHAGE DE LA SCÈNE

Jusqu'à présent, on a expliqué les calculs des positions des vagues, du navire et de tous les objets comme le soleil, les nuages, le terrain, les bouées. On a aussi partiellement expliqué les éclairages des objets. On a donc des triangles composés de sommets affichés à l'écran avec pour chacun un petit bout d'image (texture 2D) et cela compose la scène. En fait, pas tout à fait car l'ordre des opérations d'affichage est important. Un exemple : quand on regarde les vagues depuis la passerelle du navire, c'est à dire à travers des vitres transparentes, il faut dessiner d'abord la mer, puis le navire puis la passerelle car ce qui est en partie transparent est dessiné en dernier. L'océan est en partie transparent ainsi que les vitres du navire, tout le reste est opaque. Certains constituants de la scène ne sont pas des éléments 3D : il s'agit du ciel. Tout est calculé en 3D mais à la fin, la mini scène du ciel est une image qu'on projette à l'écran. Enfin, le brouillard s'applique partout, aux vagues, comme au terrain, aux bouées et au navire. Dans le brouillard, l'avant du navire doit pouvoir être « dans » le brouillard.

En fait, l'ordre des opérations est un sujet en soi. Quand on regarde sous l'eau, c'est à dire que la composante y (hauteur) de la caméra est négative, on ne doit plus afficher le ciel, les nuages, etc. mais on doit changer complètement d'atmosphère avec une visibilité réduite et qui diminue rapidement ainsi que des particules en suspension pour créer une atmosphère.

Ainsi, dans ce flot d'opérations à combiner, tout commence par calculer le mouvement des vagues de l'océan et le mouvement du navire sur ces vagues. Ensuite on fabrique la texture (image) de réflexion du navire sur l'eau. Puis on dessine les objets autres que les vagues et le navire comme le terrain, les bouées, le ciel, les nuages. Puis on dessine soit le navire puis l'océan soit l'océan puis le navire si on est à la passerelle. On ajoute le ou les radars, le ou les gouvernails, la ou les hélices. Quand on dessine les vagues, on les déforme selon les vagues de sillage du navire, on ajoute la mousse autour du navire, la mousse du sillage, la réflexion du navire sur l'eau à laquelle on ajoute un petit effet de flottement comme des rides sur l'eau. On ajoute aussi la mousse sur les vagues qui déferlent. Le fragment shader de l'océan est le plus compliqué des shaders de rendu. Tous ces affichages se font dans ce qu'on appelle un « framebuffer multisample », c'est à dire dans une image hors écran et non crénelée (les traits sont tout fins et adoucis - antialias).

Ensuite, on applique les particules de fumée et les particules de spray à l'avant du navire.

Après qu'on ait obtenu cette image de la scène, on a encore quelques opérations à effectuer dans une phase qu'on appelle de post-processing avant de l'afficher à l'écran. En effet (sans jeu de mots), on ajoute la brume localisée autour de l'horizon ou le brouillard, la partie sous l'eau et la vue quand on a les jumelles (écran noir avec deux ronds laissant voir une scène grossie).

Enfin, on applique l'image à l'écran.

Et on recommence tous les calculs de positionnement et d'éclairage puis on recompose la scène. Et cela toutes les 150 fois par seconde.