Compte Rendu TP C++

Stenzel Cackowski – Clément Gain

I – Réponses aux questions

A – TP1

Question 1

« Dvector x; » libère de la place en mémoire pour un objet de type « Dvector », aucun constructeur n'est appelé, (équivalent du malloc), ensuite l'appelle de « Dvector(2,3); » initialise l'objet « x »en utilisant le constructeur correspondant aux paramètres.

La ligne «Dvector x = Dvector(2,3);» fait directement appel au bon constructeur et il n'y a donc qu'un seul appel au constructeur.

B – TP2

Question 2

Tout d'abord, dans le premier cas, il est techniquement possible de modifier la source car les variables ne sont pas déclarées en const. De plus, dans le premier exemple, les arguments ne sont pas passés par référence, il y a donc 2 objets supplémentaires créés dans l'exemple 1.

C – TP3

Question 2

Voir schéma partie 2.B.1.

Question 3

L'amplitude permet d'avoir des vagues plus ou moins plates, ceci est très facile à voir en changeant la valeur dans le test « main.hxx », ce qui est très intuitif.

D – TP4

Question 2

En observant l'outil gprof, nous avons pu générer un fichier analysis.txt . Nous avons alors pu observer les fonctions qui prennent le plus de temps dans notre programme. Les foncteurs représentent une part significative du temps de l'éxécution totale. Cela s'explique par le très grand nombre d'appels de la fonction. La fonction compute est elle aussi gourmande en temps. On remarque que l'éxécution d'un appel de la fonction représente approximativement 150 microsecondes. Cette durée non négligeable s'explique notamment par le calcul de la fft qui nécessite beaucoup de temps.

II – Compte rendu

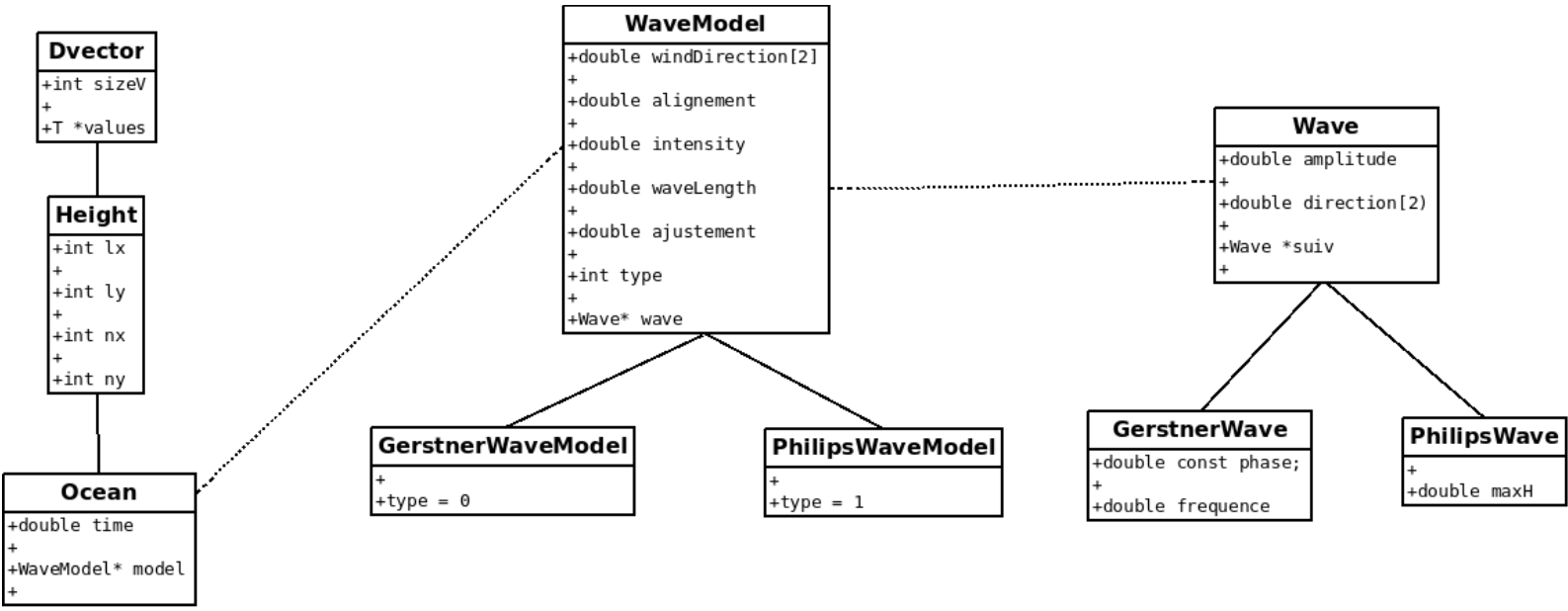
A – Analyse du problème

L'objectif de ce TP était de modéliser et de simuler la surface d'un océan. Pour ce faire, nous allons utiliser 2 modèles de houle, la houle de Gerstner et la houle de Philips. Cette simulation se basera sur un certains nombre d'hypothèses validées par la communauté océanographique. L'objectif n'est pas de produire un modèle complètement fidèle à la réalité mais plutot un modèle réaliste calculable en temps réel.

B – Conception de la solution

1 – Structure du Code

La figure ci-dessous résume la structure du code adoptée pour ce projet. Un trait plein signifie « la classe du bas hérite de celle du haut ». Un trait en pointillé signifie « La classe de gauche est composée d'éléments de la classe de droite ».



a – Dvector

La classe Dvector peut être soit composé de double soit de complex<double>. Ce choix est justifié car dans ce TP nous n'aurons besoin que de vecteur de double et de complex<double>. Pour ce faire, nous avons utilisé une classe générique de vecteur travaillant à la fois sur les complexes et sur les nombres flottants.

La classe Dvector est dotée de tous les opérateurs utiles. Pour les vecteurs travaillant sur des complexes, il est possible d'obtenir le conjugué du vecteur, la transformée de Fourier, la transformée de Fourier inverse. Pour réaliser cette transformée, on utilisera l'algorithme de Cooley-Tukey

b – Ocean

La classe Ocean hérite de la classe Height. Elle contient un attribut double et un WaveModel *. Le time est en double. Le WaveModel

La méthode principale de la classe Ocean est la méthode compute qui permet de donner la hauteur de la houle à un instant t. Cette méthode agit différemment si l'océan est composé de GerstnerWave et de PhilipsWave. Dans tous les cas la méthode boucle sur toutes les Wave du WaveModel et réalise les opérations adéquates. A la fin de chaque compute, on incrémente le temps t afin de voir l'évolution de l'océan dans le temps. Nous avons fait le choix de ne pas incrémenter le temps de la même manière selon si nous sommes dans le cas de Gerstner ou de Philips. En effet, nous avons pu constater grace à nos tests que le rendu visuel était meilleur en augmentant le temps de manière différente selon le cas de Gerstner ou de Philips.

c – WaveModel

En plus des attributs indiqués dans le sujet, nous avons ajouté à notre classe WaveModel un attribut type, type = 0 pour des GerstnerWave et type = 1 pour des PhilipsWave. Cet attribut permet de distinguer le type d'onde notamment dans la méthode compute de Ocean. Notre WaveModel est également constituée d'une liste chaînée de Wave. Dans notre méthode compute, il suffit alors de boucler sur toutes les Wave.

d – Wave

La spécificité de notre classe Wave est qu'elle contient un élément Wave *suiv nous permettant d'avoir une liste chaînée de Wave. Ce choix nous paraissait le plus judicieux pour contenir nos différentes Wave.

2 – Fonctionnement de la simulation

Afin de visualiser notre océan, nous utilisons les fichiers fournis par les professeurs « Visu-Template ». Nous avons adapter notre code afin qu'il s'intègre avec la solution fournie par les professeurs. La méthode compute est donc appelée à plusieurs reprises. A chaque appel de la méthode compute le temps est incrémenté de manière à voir l'évolution de l'océan. A chaque fois on récupère le nouveau champ de hauteur généré par la méthode compute et on le dessine permettant ainsi d'avoir un aspect réaliste de l'océan. A la suite de nos tests, nous avons convenu qu'une taille de 200 par 200 et une discrétisation de 64 points par 64 points permettaient un rendu optimal.

C – Tests

Afin de s'assurer de la robustesse de notre programme, nous avons réalisé des tests régulièrement pour tester les diverses fonctionnalités de notre programme.

D – Limites et pistes d'améliorations

Nous avons consciences que notre programme contient certaines limites dont voici une liste non exhaustives.

Tout d'abord, notre océan ne peut pas être à la fois constitué de GerstnerWave et de PhilipsWave. En effet, la manière dont la méthode compute a été implémenté ne permet pas un couplage des ondes. De plus, certains attributs indiqués dans le sujet ne sont pas utilisés dans nos calculs. Une meilleure compréhension de certains paramètres nous auraient probablement permis de les utiliser dans certaines formules et auraient entrainés un meilleur rendu visuel. Dans notre méthode compute nous ne mettons pas à jour le déplacement horizontal. De plus, certains de nos tests révèlent qu'il existe encore quelques fuites mémoires que nous n'avons pas pu corriger faute de temps.

Enfin le fichier analysis.txt révèle que notre méthode compute est particulièrement couteuse en temps (36 % du temps d'exécution total). L'utilisation du parallélisme notamment de la fft permettrait probablement une diminution du temps d'exécution du programme.

E – Retour d'expérience

Ce projet nous aura permis de découvrir d'une part la programmation en C++, d'autre part d'avoir une première approche des techniques de simulation d'océan. Nous avons notamment pu avoir un aperçu de toute la complexité de ce domaine.

De plus nous avons découvert à travers ce projet la notion de référence et la notion de programmation générique.

Pour conclure, en tant qu'étudiant MMIS, ce projet s'inscrivait particulièrement bien dans notre formation avec un exemple concret de ce que pouvait être la « modélisation mathématiques ».