Projet Modélisation en C++
Simulation de la surface d'un océan

Julien BORDAS & Antoine LADRECH

Février à Mai 2017

Nota Bene

Le package fourni *VisuTemplate* ne fonctionne pas avec notre implémentation.
Notre simulation crée, à chaque pas de temps, un fichier contenant la heightmap à l'instant t, stockée au format lisible par *GNUplot*. Cela nous a permis de créer des GIF animés, que vous pourrez retrouver dans notre archive dans le dossier *./data*.

1 Synthèse des questions des TP

TP1 - Introduction et outils

```
Dvector x;
x = Dvector(3, 1.);
```

Dans cette écriture, l'utilisateur fait appel à l'opérateur =.
Tant que ce dernier ne sera pas surchargé, ce code ne s'exécutera pas !

```
Dvector x = Dvector(3, 1.);
```

Dans cette écriture, l'utilisateur fait appel au constructeur par recopie qui, lui, est défini dans le code.

TP2 - Opérateurs

```
Dvector operator+ (Dvector a, Dvector b);
```

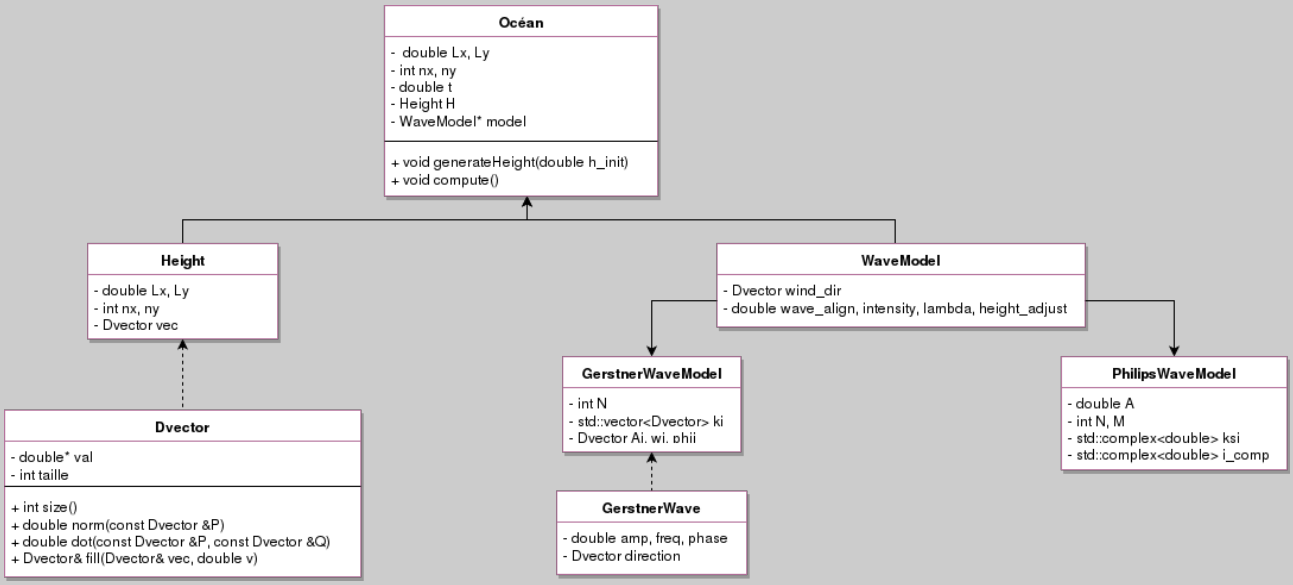
Dans cette écriture, l'opérateur fonctionne par valeurs et crée plus d'objets intermédiaires.

```
Dvector operator+ (Dvector a, Dvector b);
```

Dans cette écriture-ci, l'opérateur fonctionne par références et ne crée pas d'objets intermédiaires : les opérations sont effectuées en place.

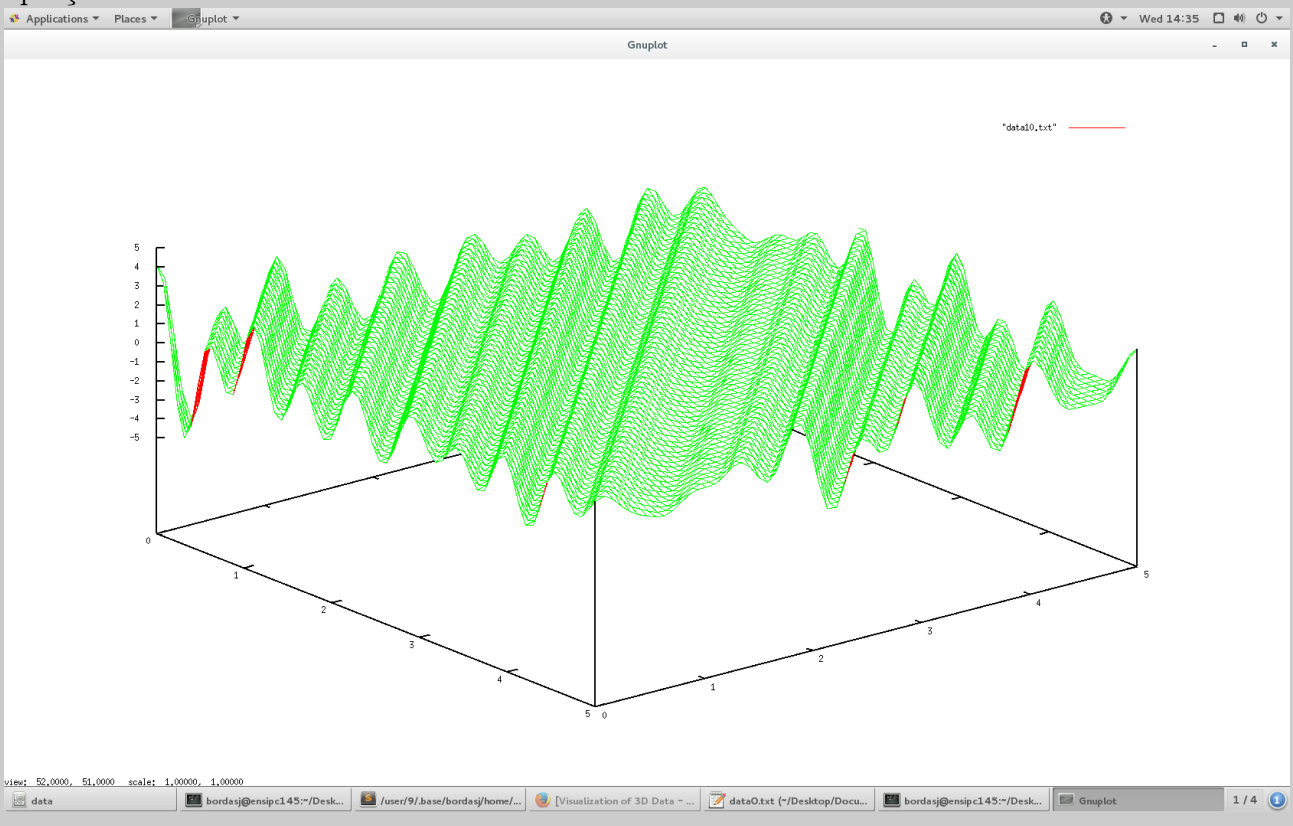
L'opérateur le plus important à implémenter est l'opérateur d'ordre <.
En effet, tous les autres opérateurs de comparaison peuvent dériver de celui-ci, et ainsi permettre de factoriser du code !

TP3 - Héritage



Cette organisation permet de créer plusieurs types de vagues, et même d'en rajouter si nécessaire.

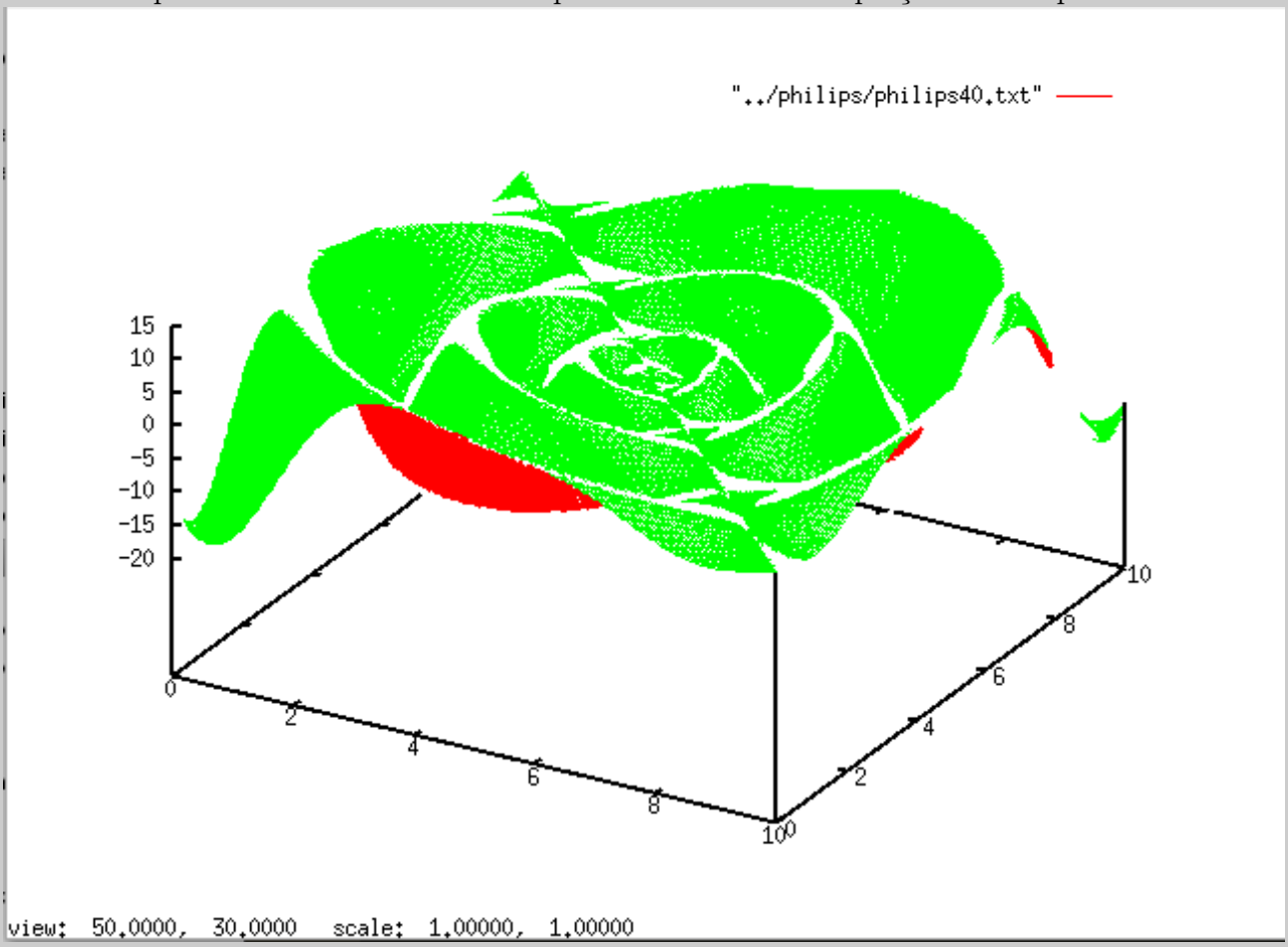
La mise en place du modèle de Gerstner a également été effectuée à partir de ce TP, voici un aperçu de son exécution à un instant t fixé :



TP4 - Programmation générique et FFT

Dans ce TP, nous avons programmé :

- La classe **GenericVectors** : cette classe est un template, elle reprend les fonctions de la classe *Dvector* mais en leur permettant d'être utilisées avec d'autres types que les *doubles*.
- L'**algorithme de la FFT** (et IFFT) 1D qui reprend l'algorithme de Cooley-Turkey ainsi que la version 2D des deux algorithmes de façon naïve : on effectue n FFT sur les lignes puis n sur les colonnes.
- L'implémentation de la classe **PhilipsWaveModel** qui modélise les hauteurs de l'océan en se servant de l'IFFT 2D. Cette méthode à l'avantage de calculer chaque occurrence indépendamment de l'occurrence précédente. Voici un aperçu du champ *htilde* :



2 Synthèse générale et résultats

Ce TP nous a permis d'aborder les notions essentielles de la programmation, telles que la généricité et l'encapsulation.
Ces notions sont d'autant plus importantes qu'elles permettent de programmer proprement et de façon efficace.
Concernant le modèle de Philips, le champs *htilde* semble prendre des valeurs cohérentes mais sa transformée de Fourier inverse est quasi nulle sauf aux extrémités (ou au centre avec un fft

shift). Nous n'avons pas réussi à corriger l'erreur et ne sommes donc pas en mesure d'afficher l'océan par cette méthode.

Influence des paramètres

Modèle de Gerstner

L'entier N représente le nombre de vagues de Gerstner que l'on va additionner afin de créer le modèle.

$$z = A\cos(k.x_0 - \omega t + \phi) \tag{1}$$

L'amplitude (notée A_i ou amp) va avoir un impact sur l'amplitude du cosinus, et donc des vagues.
Le vecteur d'onde (noté k_i ou $direction$) va influer sur la direction de propagation.
La fréquence (notée ω_i ou $freq$) aura un impact sur la vitesse de propagation.
La phase (notée ϕ_i ou $phase$) n'aura pas d'impact (car constante sur l'expérience) si ce n'est de retranscrire l'orientation du vent.

Modèle de Philips

Les paramètres concernant les aspects généraux de l'océan sont les mêmes que pour le modèle de Gerstner.
S'ajoutent deux nombres aléatoires ξ_r et ξ_i qui suivent une loi normale de moyenne 0 et de variance 1. Ils permettent d'altérer de manière aléatoire l'aspect et la progression des vagues.