Programmation générique et FFT

Objectifs

Afin d'améliorer la qualité de représentation de la hauteur de houle, nous allons modifier l'algorithme de simulation de la hauteur pour avoir une représentation spectrale du champs de hauteur.

Les notions abordées dans ce TP sont : la programmation générique, l'héritage et l'encapsulation.

Vous êtes libre de modifier la classe vecteur pour l'adapter aux besoins qui pourront apparaître : vous pouvez ajouter des opérateurs, des attributs ou des méthodes.

Lors de l'implémentation de ce nouvelles classes, vous devrez apporter un soin tout particulier à la gestion de la mémoire et à la documentation de votre code.

Les données des différentes classes doivent, dans la mesure du possible, être stockées de façon contiguë en mémoire.

tabLa gestion d'erreurs doit se faire via des exceptions (utilisez `throw`).

Implémentation

1. Nous allons dans un premier temps implémenter un nouveau modèle de simulation de l'évolution de la hauteur basée sur une représentation spectrale du champ de hauteur. Vous devez donc dérivez une classe `PhilipsWaveModel` de la classe `WaveModel` qui implémentera l'évolution de la hauteur de houle selon le modèle de Philips. Cette classe devra encapsuler le modèle de Philips de la même manière que dans le TP précédent.

Afin d'obtenir une représentation spectrale, nous devons réaliser une transformée de Fourier d'un tableau de donnée, et également la transformée de Fourier inverse. La transformée de Fourier peut-être calculée en utilisant

l'algorithme de CooleyTukey

Algorithme 1 : Transformée de Fourier rapide de Cooley-Tukey

```
1  Function fft(x) :
       Data : x ∈ C^n
       Result : y ∈ C^n
2      if n<=1 then
3          return y=x
4      end
5      even = x[0 :2 :n] // Choisi les éléments d'indices pairs
6      odd = x[1 :2 :n] // Choisi les éléments d'indices impairs
7      // Appel récursif sur chacun des sous tableaux
8      fft(even)
9      fft(odd)
10     for k ← 1 to n/2 do
11         t = odd[k] exp(−2 ∗ π ∗ k/n)
12         x[k] = even[k] + t
13         x[k+N/2] = even[k] - t
14     end
15     return y=x
```

Cet algorithme travaille sur des tableaux ayant pour type de base le complexe.

2. Implémenter une classe générique de vecteur travaillant à la fois sur les complexes et sur les nombres flottants.
3. Implémenter la FFT en utilisant l'algorithme proposé. Afin de simplifier l'implémentation, on se limitera aux cas où n est une puissance de 2.

Pour la transformée de Fourier inverse s'écrit en s'appuyant sur la transformée de Fourier

Algorithme 2 : Transformée de Fourier inverse

```
1  Function ifft(x) :
       Data : x ∈ C^n
       Result : y ∈ C^n
2      if n<=1 then
3          return
4      end
5      x = x̄ // calcul du vecteur des conjugués
6      fft(x)
7      y = x̄/n // calcul du vecteur des conjugués
```

4. Implémenter la transformée de Fourier Inverse en utilisant l'algorithme proposé.
5. Utiliser votre implémentation pour calculer la FFT d'un tableau bidimensionnel stocké sous forme de vecteur.
6. Intégrer la FFT dans le modèle de Philips.

L'ensemble des outils permettant de décrire l'évolution de la houle est maintenant en place. Nous allons les intégrer au sein d'une même classe 0cean. Cette classe devra contenir à minima, les attributs suivant :

1. L_x la longueur du domaine dans la direction x,
2. L_y la longueur du domaine dans la direction y,
3. n_x le nombre de points de discrétisation du domaine dans la direction x,
4. n_y le nombre de points de discrétisation du domaine dans la direction y,
5. t, le temps courant,
6. H, un vecteur contenant la hauteur de la houle.
7. `Model`, un attribut contenant le modèle utilisé,
8. un tableau dynamique `vertices` de nombres en double précision permettant d'interfacer les vecteurs de calculs avec le système de visualisation.

Les méthodes qui devront être implémentés sont, à minima :

1. le constructeur,
2. le destructeur,
3. une méthode d'initialisation de la hauteur `generateHeight`,
4. une méthode `compute` qui permet de calculer pour in instant t la hauteur (et éventuellement le déplacement horizontal) de la houle.
5. une méthode `gl_vertices` qui permet de transformer la houle en un tableau contenant les positions (x, y, h) pour la visualisation.

> **Remarque 1**
>
> La boucle temporelle est réalisée à l'extérieur de la classe `Ocean`, soit dans le `main`, soit dans une classe dédiée à la visualisation `Rendering`.

Analyse

> **Question 1**
>
> Pour chacun des éléments implémentés, définir des tests pertinents.

> **Question 2**
>
> En utilisant l'outil `gprof`, évaluer les performances de votre implémentation.