Simulation du problème à N corps

Le code source fourni dans ce répertoire simule un système de N particules avec une masse, qui exercent des forces de gravitation les unes sur les autres.

Pour compiler:

Se mettre dans le répertoire qui contient ce fichier.
Taper:

```
./build.py
```

Par défaut, ./build.py utilise gcc/g++ comme compilateurs C/C++. Si vous ne pouvez pas utiliser gcc ou g++, vous définissez les variables d'environnement CC et CXX avec les compilateurs dont vous disposez.

Par exemple, pour utiliser clang et clang++:

```
export CC=clang
export CXX=clang++
./build.py
```

Si tout s'est bien passé : les fichiers ncorps_OpenMP, ncorps_Serial, ncorps_TBB, ncorps_THREADS sont créés dans le répertoire install/nom du compilateur C/release/double/

Pour exécuter la version séquentielle de référence (si les compilateurs gcc et g++ sont utilisés) :

Taper :

```
./install/gcc/release/double/ncorps_Serial [-n <particules>] [-it
<itérations>] [-g]
```

où

- <particules> est un entier positif (nombre de particules, par défaut 4096)
- <itérations> est aussi un entier positif (nombre d'itérations en temps, par défaut 300)
- -g est une option qui affiche une fenêtre graphique du nuage de particules (si les librairies OpenGL sont installées sur la machine)

Pour exécuter une des versions parallélisées :

Taper (pour la version OpenMP):

```
./install/gcc/release/double/ncorps_OpenMP [-n <particules>] [-it <itérations>] [-g] [-threads <threads>]
```

où

- <particules> est un entier positif (nombre de particules, par défaut 4096)
- <itérations> est aussi un entier positif (nombre d'itérations en temps, par défaut 300)
- -g est une option qui affiche une fenêtre graphique du nuage de particules (si les librairies OpenGL sont installées sur la machine)
- <threads> est le nombre de threads à utiliser (par défaut 1)

Même chose pour les versions utilisant TBB ou std::thread.

Mesure des performances (profiling)

Compiler le code en mode "profile":

```
./build.py -m profile
```

Exécuter le code

```
./install/gcc/profile/double/ncorps_Serial
```

qui crée un fichier `gmon.out`

Pour obtenir les mesures de performance, il faut taper la commande:

```
gprof ./install/gcc/profile/double/ncorps_Serial > profile.out
```

Examiner le fichier `profile.out` pour voir quelle(s) fonction(s) du code prennent le plus de temps à s'exécuter et qu'il est intéressant de paralléliser en premier.

Exercice

Parallélisation OpenMP

Modifier le fichier `src/CPU/openmp/systemCPU_OpenMP.cxx` en ajoutant la(les) pragma(s) OpenMP en fonction des résultats du profiling.

Recompiler et exécuter en utilisant différents nombres de threads, par exemple:

```
./install/gcc/release/double/ncorps_Serial
./install/gcc/release/double/ncorps_OpenMP -threads 1
./install/gcc/release/double/ncorps_OpenMP -threads 2
./install/gcc/release/double/ncorps_OpenMP -threads 3
./install/gcc/release/double/ncorps_OpenMP -threads 6
```

Calculer les speed-up par rapport à la version de référence séquentielle

Parallélisation TBB

Modifier le fichier `src/CPU/tbb/systemCPU_TBB.cxx` en utilisant les fonctions de la librairie TBB.

Faire les mêmes tests que dans la version OpenMP