Projet AOB IATIC4

Decembre 20, 2018

Un des objectifs de ce projet est d'implémenter un protocole d'expérimentation. Tout résultat dont l'origine n'est pas expliqué sera considéré comme nul.

Installation de la simulation

Contenu de l'archive

1. *demo.html* lance l'application,
2. *WebStart.java* contient le coeur java utilisé pour l'affichage de la simulation (appelle la class FluidSolver),
3. *FluidSolver.java* définit l'objet Fluid Grid et appelle la librairie de calcul en C,
4. *fluid.c* contient les fonctions de simulation,
5. *interface_c_java.swig* décrit l'interface entre les fonctions C et la machine virtuelle Java,
6. *applet.policy* autorise l'utilisation de SWIG dans votre navigateur,
7. *Makefile* contient les règles de compilation.

Dépendances

1. *openjdk* est un kit de développement pour Java, vous trouverez openjdk dans votre gestionnaire de paquets,
2. *SWIG* est un package nécessaire à l'appel de fonctions C depuis le java, c'est un package standard à installer également via votre gestionnaire de paquet,

Configuration

Une fois les paquets installés, il va falloir modifier le chemin des include dans le fichier Makefile. Le plus simple étant d'utiliser la commande *locate* pour obtenir le chemin des fichiers jni.h et jni_md.h et de modifier, selon l'emplacement, la variable *INCLUDE* dans le Makefile.

Lancement de l'application

Quelque soit la solution que vous choisissez pour lancer l'application, il vous faut ajouter le chemin du projet à votre LD_LIBRARY_PATH. Pour cela, mettez vous dans le répertoire du projet et effectuez la commande suivante :

```
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
```

Vous avez deux possibilités pour le lancement de l'application :

1. En ouvrant le fichier demo.html dans votre navigateur. Il vous faudra modifier les règles de sécurité car par défault les appels à des fonctions C sont bloqués. Il faut trouver le fichier java.policy (son emplacement dépend de votre distribution Linux et de la version de java installée) et ajouter la ligne suivante, en modifiant le chemin pour indiquer là où vous avez extrait le projet :

   ```
   grant codeBase "file:/path/to/the/project/*" {
      permission java.security.AllPermission;
   };
   ```

2. En utilisant appletviewer, qui permet de lancer les applets java intégrés dans un document html sans afficher le contenu html. Le lancement d'appletviewer avec les bons paramètres est intégré dans le Makefile, utilisez la commande suivante pour lancer la démo :

   ```
   make run
   ```

Interactions avec l'application

Le programme supporte les intéractions suivantes :

1. Click gauche pour ajouter de la fumée,
2. Maintenir click droit et déplacer pour ajouter de la vélocité,
3. Appuyer sur 'v' pour afficher la grille de vélocité,
4. Appuyer sur 'r' pour réinitialiser l'application,
5. Utiliser '[' et ']' pour modifier la taille de la grille,
6. Utiliser '.' et ',' pour changer le pas.

Optimisation

L'objectif principal de ce projet est d'améliorer les performances de la simulation. Toute amélioration est accepté, néanmoins, il est attendu l'utilisation de différentes techniques vues ou discutées durant les TDs.

Certaines tentatives ne seront pas fructueuses mais si vous expliquez l'intérêt des éléments utilisés et pourquoi leurs applications dans ce cas peut donner de meilleurs résultats, des points seront accordés. En revanche, la simple citation d'un mot clé ainsi qu'un copier-coller de sa page Wikipédia entrainera probablement un retrait de points.

Fonctions effectuant la simulation

La très grande majorité du temps de la simulation est passée dans la fonction *linearSolver* (fluid.c). L'optimisation portera donc sur cette fonction ainsi que toutes celles appellées par celle-ci.

Validité de la simulation

Il est interdit de réduire la précision du calcul de la fonction *linearSolver* en réduisant le nombre d'itérations car l'algorithme utilisé ici est un algorithme itératif, voir *Gauss-Seidel*.

Il est possible de changer l'algorithme utilisé (certains algorithmes permettent l'application d'optimisations différentes) mais c'est une manoeuvre délicate qui ne donnera des points que si complétement mené à bien. Il est donc extrêmement conseillé de tenter cette approche en dernier lieu.

Protocole expérimental

Implémentez un protocole qui vous permettra de mesurer précisément et efficacement une modification des performances de la simulation.

Présentation du protocole

Vous devrez présenter les différents éléments du protocole de mesure et expliquer en quoi ils permettent de gérer les particularités de ce programme.

Temps de référence

Chaque configuration étant différente, vous devrez indiquer la valeur de référence de chaque métrique qui sera utilisée pour mesurer les performances. C'est-à-dire la valeur de chacune ce ces métriques lorsque l'on lance le programme dans son état originel.

Rendu

Le rendu est attendu au plus tard le 14 Février à 23h59. **AUCUN** retard ne sera accepté. Celui-ci sera sous un format archive comprenant un rapport et une version de l'application optimisée.

Rapport

Le rapport devra comprendre la présentation de toutes les optimisations entreprises. Pour chaque optimisation, il est attendu :

1. Une description théorique de son apport (cadre général),
2. Une description technique de son implémentation dans ce programme,
3. Des données justifiant son impact (ou non) sur les performances,
4. D'une analyse expliquant sa conservation ou non dans la version finale (gain/perte, incompatibilité avec d'autres optimisations, etc.).

Code

Le code rendu devra inclure chaque version des fonctions modifiées et les variables nécessaires dans le Makefile pour pouvoir exécuter séparément chaque optimisation que vous avez effectuée. La version qui compilera par défaut sera la version finale (et donc la plus optimisée).

Il est interdit de dupliquer l'ensemble du projet pour chaque optimisation et de renvoyer une archive comprenant plusieurs fois tout les fichiers. Limitez les copies aux fonctions/fichiers que vous modifiez.