Descriptif des programmes

I Introduction

Un paramètre très important lors des simulations de marches aléatoires est la qualité du générateur aléatoire. N'étant pas satisfait du générateur aléatoire du C (rand() fonctionne par congruence linéaire) et après quelques recherches, je suis tombé sur « Mersenne Twister », qui bien initialisé est un excellent générateur.

Simulé des marches demandent de lancer beaucoup de tests pour espérer avoir des valeurs raisonnables. Cette tâche se parallélise aisemment, dès lors que le générateur se parallélise lui aussi. Il suffit de lancer plusieures marches en parallèle. Mon choix de plateforme de calcul s'est arrêté sur le calcul sur carte graphique par CUDA, adapté pour des tâches aussi parallélisable et légère, d'autant plus qu'il existe une implémentation de Mersenne Twister pour CUDA : MTGP.

II Détails d'implémentations

Pour éviter des conflits d'accès, chaque thread fait ses propres statistiques (structure Etat dans le programme), et il suffit juste de les réunir par la suite.

Plusieurs problèmes se posent :

- Sur des machines avec serveur graphique, un calcul CUDA ne doit pas durer plus de cinq secondes, ce qui nécessite donc de savoir interrompre et reprendre une marche aléatoire. Fait.
- Les marches aléatoire de l'exercice 1 n'ont pas toutes la même durée. Des processus seront donc inactifs pendant que d'autres travailleront. Ceci peut-être résolu en construisant un gestionnaire de tâche, chargé d'équilibrer le calcul, pour relever les performances. Non Fait : les tâches sont allouées statiquement.

Exercice2.cu été bien commenté pour expliquer les divers parties du code. Exercice1.cu est relativement similaire.

III Performances

Sur une carte graphique GTX275, j'arrive à une performance brute de 46 milliards de déplacements par seconde pour l'exercice 2, ce qui permet d'avoir un échantillon correcte garantissant la pertinence des calculs.

IV Utilisation

L'interface homme-machine est pour le moins réduite. L'utilisateur entre (à la main) ses données dans le main() à la fin du fichier source, compile et l'exécute. Le programme affiche l'avancement du calcul dans le terminal, tout en ajoutant aux fichiers exercice1.txt ou exercice2.txt les résultats successifs. Des exemples sont donnés dans les codes.

Il suffit de taper make en ayant ajuster au préalable dans le Makefile la localisation de la SDK CUDA

L'affichage à l'écran indique le nombre total de déplacements effectués(tot), la vitesse correspondante(vtot), le nombre de marches calculées(test), la vitesse correpondante(vtest) les résultats et paramètres, le temps écoulé depuis le début de la simulation courante ainsi que l'avancement.

Les données ont été importées dans OpenOffice.org Calc, et les courbes ont été tracées à partir de là.