UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Master informatique CHPS

HPC Center ROMEO Centre de Calcul Regional

Rapport de Projet CHPS0802

Programmation GPU

Sujet : Implémentation d'un calcul de Raytracing avec CUDA

Professeur : François ALIN

Élève : William MARTINEZ

Sommaire :

Sujet : Implémentation d'un calcul de Raytracing avec CUDA **1**
Sommaire : **2**
Introduction **3**
Implémentation du Raytracing **3**
Classe et relation 3
Choix d'architecture 3
Intégration CUDA **3**
Le Choix droit dans le mur 3
Implémentation finale 4
Etat du Projet **4**
Partie Réussite 4
Bugs Présent 4
Partie non-implémenté 5
Conclusion **5**

Introduction

Le projet qui nous a été donné a pour objectif la mise en œuvre d'un moteur de raytracing (simplifié) en utilisant CUDA, et la programmation parallèle sur GPU.

Le raytracing est une technique de rendu réaliste qui simule la manière dont la lumière interagit avec les objets dans une scène simulée en 3D. Bien que cette méthode soit réputée pour sa qualité visuelle, elle est également très coûteuse en calcul. L'utilisation de CUDA permet d'exploiter la puissance de calcul parallèle des GPU pour accélérer considérablement le processus de rendu. Son utilisation est tout indiqué car la plupart des calculs ne demande pas une grande complexité mais beaucoup de répétitions. Exactement le cas où les GPU brillent le plus.

Ce rapport présente les choix techniques, les étapes de développement et les performances obtenues lors de l'implémentation de ce projet.

Implémentation du Raytracing

L'implémentation d'un Raytracing peut se faire de nombreuses manières possible, et dans de nombreux langages possible. Ici le choix du langage est imposé (C++,CUDA), mais malgré cela il reste encore de nombreuses façons de faire.

Pour le cas de mon implémentation j'ai choisi d'essayer d'utiliser un maximum la POO, n'ayant pas eu la meilleur relation avec C++ je voulais faire en sorte que ce projet me permettra de me remettre à jour au niveaux de mes compétences dans ce langage et de devenir plus à l'aise. (choix qui me fera grandement défaut plus tard voir partie Intégration CUDA)

Classe et relation

Pour ce qui est de l'implémentation du Raytracing, la première approche à été d'essayer de plus ressembler au possible à ce qui nous a été présenté en séance de TP et d'essayer d'en dévier le moins possible.
Pour le détail différents classe et leurs relations entre elles vous pouvez vous référer au schéma UML ci-dessous :

Choix d'architecture

Dans cette partie je vais énumérer les différentes classes disponibles dans la partie séquentielle de mon algorithme de Raytracing et expliquer leur fonctionnement et leur but :

coucou

Intégration CUDA

Lorsque la partie séquentielle de mon projet fut stable et commença à donner de bon résultat, ce fut le moment où je commençais à regarder plus en détail la façon dont je voulais utiliser CUDA pour augmenter les performances de mon algorithme. Je ne partis de rien puisqu'au début du projet lors de ma partie de planification je voyais déjà assez

clairement où était le potentiel gain de vitesse de l'utilisation de cuda. La parallélisation du calcul d'intersection entre les objets et tous les rayons de la Vue. Pour une image de 2048*2048 pixel cela fait 4 194 304 opérations toutes identiques et sur le même rythme, parfait pour l'utilisation du GPU. Mais je n'avais pas encore percuté qu'au début de mon implémentation j'avais fait un choix qui allait me faire prendre plusieurs heures de retard.

Le Choix droit dans le mur

Lorsque j'ai commencé à coder ce projet, comme je l'ai expliqué dans la partie implémentation du Raytracing j'ai voulu utiliser un maximum de POO. Cela ne m'a pas dérangé avant que très tardivement dans l'avancée du projet, lors du moment où j'ai commencé à attaquer la partie cuda. Une question venna me perturbé "L'utilisation dynamique de méthode fonctionne en CUDA ?" et en me renseignant sur le sujet sur la possibilité qui s'offrait je mesurais petit à petit l'ampleur de l'erreur que j'ai réalisée.

Deux choix s'offraient à moi : recommencer entièrement le code en utilisant des structures et en transformant toutes les méthodes de classe et les héritage en code compatible CUDA. Ou bien décider de rester avec ce que j'avais actuellement et tenter de faire un traducteur polymorphisme cpu -> cuda gpu. Sortant d'une longue période de débogage, j'avais peur de casser ce que j'avais réussi à obtenir jusque là, j'opta donc pour la seconde solution.

Encore maintenant je ne suis pas sûr que ce soit la meilleure solution possible à ce moment. Mais je pense que vouloir préserver un "minimum viable product" est toujours important et je pense donc avoir choisi la solution avec laquelle j'était le plus confiant.

J'ai donc passé de nombreuses heures à essayer de trouver des solutions pour essayer d'utiliser un semblant de polymorphisme en CUDA. Ce qui ne fut pas très concluant vu que CUDA n'est pas vraiment adapté pour ce genre de logique..

Implémentation finale

Dans cette partie je vais présenter le détails de mon implémentation de CUDA dans ce projet, et essayer d'expliquer pourquoi ces choix ont été fait :

coucou

Etat du Projet

Actuellement le projet peut être compilé à l'aide d'un CMakeLists.txt et des fichiers qu'il génère. Les éléments qui sont dans ma scène peuvent être vu dans le résultats "**test_img.ppm"** et semble plutôt être juste et réel dans la mesure du possible (voir partie bug ci-dessous). La partie Cuda à été écrite mais n'a pu être suffisamment travaillé pour permettre d'être fonctionnelle, mais l'entièreté du code est présente il ne reste plus qu'une dernière couche de débogage pour finir le tout. Dans cette partie nous allons passer en revu les parties réussites et raté du projet pour faciliter la navigation dans ce projet.

Partie Réussite

Pour ce qui est des éléments réussit comme dit précédemment, il est possible de :

-Créer des sphères et des plan au choix et les intégrer dans le monde
-Pouvoir régler la lumière ambiante du monde pour exagérer les contrastes
-Setup les couleurs des différents éléments visible dans le monde
-Créer une lumière qui se projette dans le monde pour influencer les couleurs
-Choisir la résolution de notre image à la création de la Vue
-Enregistrer ses résultats en .ppm

Et le reste (hormis la partie google test) est à quelque heure tout au plus d'être entièrement fonctionnel.

Bugs Présent

Deux bug majeurs sont présent dans l'état actuel de ce projet :

- Bug d'intersection des objets :

Selon les divers tests que j'ai fait, semble que tous les objets que je veux représenter dans ma scène se retrouvent à l'affichage exactement à l'opposé d'où il devrait être. J'ai essayé de regarder séparément chacun des objets et ma matrice et tout semble bien être instancié avec les bonnes valeurs qui permettent leur bonne représentation théorique. Sur papier si j'ai bien compris tous les éléments du raytracing les position inverse de ce que vous pouvez trouver dans le fichier main_test.cpp devrait être celle qui ont un affichage.

Ma théorie sur la provenance de ce bug, est l'existence d'une faute dans le calcul d'intersection de mes rayon un signe négatif en trop ou en moins de ce qu'il faudrait avoir pour que le tout fonctionne normalement.

Temps estimé passé sur le bug : 4h (arrêt car trop de perte de temps et fonctionnel en inversant les positions)

- Bug de fonctionnement de CUDA :

Suite à mes nombreuses bataille avec cuda sur son installation sur mon PC comme décrit dans la partie CUDA de ce rapport

Temps estimé passé sur le bug : 10h (arrêt car manque de temps, date de rendu jour même et rapport toujours pas fini)

Partie non-implémenté

Au total par rapport à mes prévisions de mes capacités et du temps réservé à la résolution de ce projet 2 élément n'ont pas eu le temps d'être implémenté :

- Opacité des éléments :

Lors de ma planification de ce projet je voulais essayer d'intégrer aux différents calculs de couleur un calcul qui me permet d'intégré l'opacité d'un matriel.La classe Matériel qui représente au final uniquement une couleur s'appelle comme cela pour cette raison.

Le projet était de pouvoir garder en mémoire tous les et renvoyer par l'intersection avec les objets de ma scène et de les classer dans l'ordre contraire en excluant les

éléments à -1. Et avec cette liste la parcourir pour permettre de calculer l'ajout de la couleur noir + couleur materiel a * opacité a + couleur matériel b * opacité b, … En remettant le tout X dès qu'une opacité de 1 est détectée.

- Les Test Google :

Au vu de à quel point le projet à changé au fur à mesure de mon avancée (nombreux renommage et changement de fonctionnalité des méthodes) j'avais décidé de remettre la partie google test à la fin du projet pour pouvoir éviter de devoir changer cela à chaque petite modification. Malheureusement comme à chaque fois je me suis fait rattraper par le temps et la partie CUDA pris beaucoup plus de temps que prévu et empiétant sur le temps alloué à cette partie. Une erreur de planification basique dans les projets : Oublier d'allouer du temps à l'imprévu (pour ma part 1 jour de week end était réservé = 6h de boulot mais cela ne fut pas suffisant).

Conclusion

Même si ce projet de Raytracing n'a pas été complètement un succès pour mon implémentation il m'a permis de renouer grandement avec le C++ et pouvoir me rendre beaucoup plus à l'aise avec lui. Il m'a aussi permis de solidifier grandement mes compétences en CUDA apprise pendant ce cours (même si elle semble encore à désirer au vu du résultat final). En terme d'algorithmique pur je pense avoir récupérer les éléments que je voulais comprendre le plus. Étant amateur de jeux vidéo j'ai de nombreuses fois entendu parler du Raytracing à tout va, et ayant passé plus de 40h à essayer de l'implémenter je pense en avoir désormais une vision bien plus claire et précise.

Toute la partie tentative d'optimisation m'a fait énormément réfléchir sur les différents problèmes qui se portent sur ce sujet. Il est clairement visible que même avec toute la bonne volonté et compétence du monde, pour implémenter le plus efficacement le Raytracing l'utilisation du GPU est un élément obligatoire. Les exemples présent dans notre classe ne comportent que seulement une dizaines d'éléments et pourtant cela peut suffir à mettre à mal la plupart des PC présent pour dépasser les 10 FPS en 720p.

Pour conclure je dirais que le projet m'a été bénéfique sur le plan d'apprentissage et de familiarisation malgré sa réussite partielle.