TD GPGPU n°4

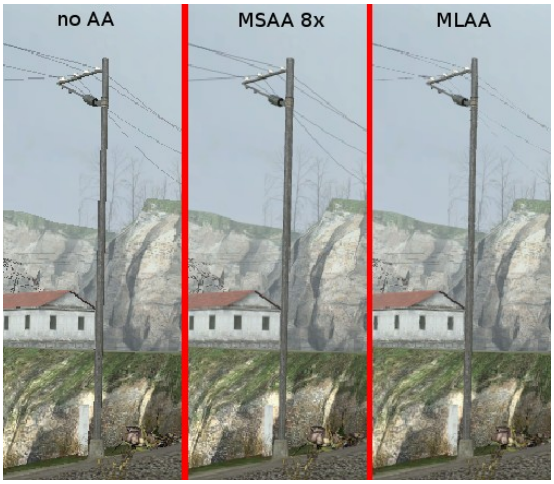
Cours de GPGPU
MII 2 option SIS / IMAC 3ème année

Antialiasing avec Cuda

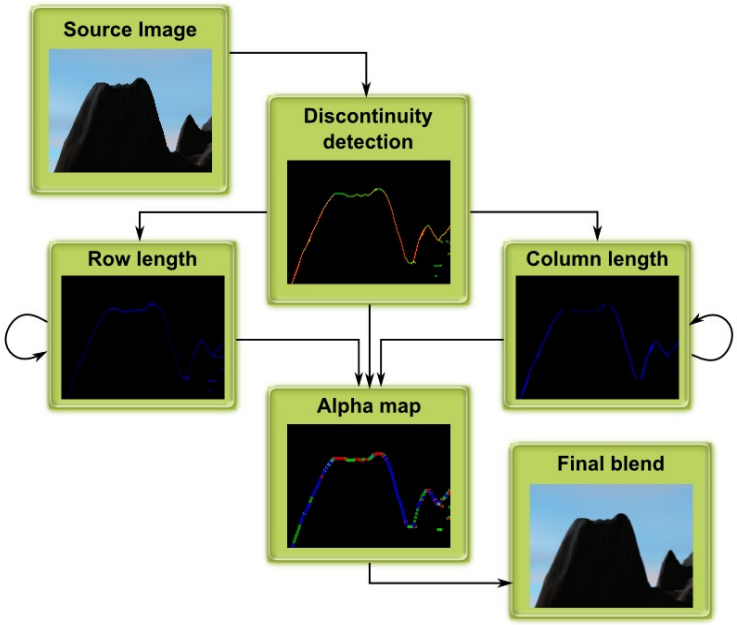
Les derniers TD seront consacrés à la réalisation d'un cas concret programmé avec Cuda : l'antialiasing en espace image

---

L'aliasing est un phénomène de crénelage dû à l'échantillonnage discret de la projection d'une scène 3D. Pour le rectifier, on utilise souvent un suréchantillonage par pixel (MSAA) permettant de calculer le poids du recouvrement d'un triangle sur un pixel. Malheureusement cette technique a un coût certain, notamment lorsque la scène est lourde (bcps de triangles). Certains jeux sur console ont même dû abandonner la possibilité de faire de l'antialiasing. Dans le cas de scènes lourdes, il est donc intéressant de faire de l'antialiasing en espace image, avec une complexité dépendante du nombre de pixels et non du nombre de triangles. Cet algorithme est appelé MLAA pour MorphoLogical AntiAliasing.



Voici un exemple schématique d'implémentation du MLAA :



Vous trouverez également des précisions dans le chapitre 3 de ce document :
http://igm.univ-mlv.fr/~biri/mlaa-gpu/TMLAA.pdf
et dans cette présentation :
http://igm.univ-mlv.fr/~biri/mlaa-gpu/prez_siggraph_mlaagpu.pdf

Préparation : chargement et sauvegarde d'un fichier image (10 pts)

Remettez la main sur vos TD précédents pour pouvoir charger et sauvegarder une image couleur (format ppm ou autre) en ayant passé l'image au GPU : chargement de l'image en CPU, puis dans Cuda, recopie simple par un kernel, retour au CPU et sauvegarde dans un fichier.

Étape 1 : Conversion en Lab (30 pts)

Cette étape vise à transformer l'image représentée en mode RVB en une représentation de couleur proche de la vision humaine, la représentation Lab. En effet, les « distances » calculées sur ces couleurs sont cohérentes avec les différences de perception que nous en avons.

1. Trouvez sur Internet les formules permettant de passer d'une représentation de couleur RVB en Lab. Quel type d'opération parallèle faudra t'il faire pour réaliser cette tâche : map, reduce, scan ?
2. Implémentez les dans un kernel Cuda pour générer l'image en Lab...

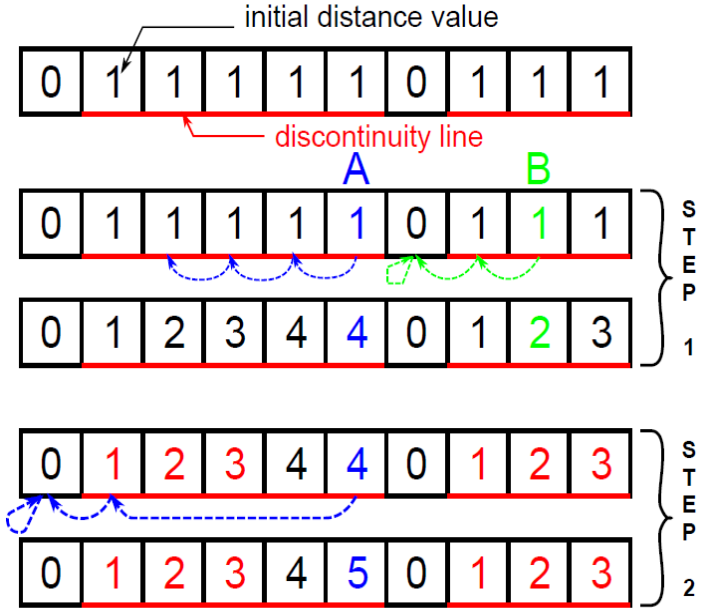
Étape 2 : Calcul de la discontinuité (30 pts)

Dans cette étape, nous calculons dans deux images résultats la présence ou non de discontinuité entre un pixel et son voisin de droite et de bas. Cette présence de discontinuité se calcule en comparant la distance entre les couleurs des deux pixels avec un seuil préfixé : le seuil de discontinuité (par défaut prenez la valeur 1.3). La distance entre 2 couleurs se calcule exactement comme la distance entre 2 position.

1. Faites un kernel calculant la présence ou non d'une discontinuité entre un pixel et son voisin de droite et de bas.
2. Faire en sorte de combiner dans un même kernel l'étape 1 et l'étape 2. En effet, on n'a pas besoin de l'image en représentation Lab par la suite.

Étape 3 : Calcul des distances le long des segments de discontinuité (100 pts)

Cette étape a pour objectif de calculer, pour tout pixel étant sur un segment de discontinuité, la distance qui le sépare de l'extrémité de ce segment. En GLSL, la technique employée est appelée *recursive doubling* et est illustrée sur ce schéma :



Dans votre cas, en Cuda, vous utiliserez la méthode que vous voudrez pour calculer les 4 distances (en nombre de pixels) suivantes :

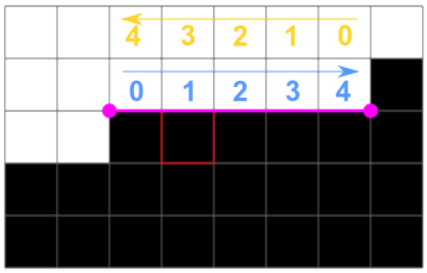
- Distance vers l'extrémité gauche pour un pixel sur une ligne de discontinuité horizontale.
- Distance vers l'extrémité droite pour un pixel sur une ligne de discontinuité horizontale.
- Distance vers l'extrémité haute pour un pixel sur une ligne de discontinuité verticale.
- Distance vers l'extrémité basse pour un pixel sur une ligne de discontinuité verticale.

Cette distance s'exprime en nombre de pixel vis à vis du bord. Le pixel à l'extrémité du segment a ainsi une distance de 1.

1. A quel type de famille d'algorithme parallèle ce calcul vous fait penser : reduce, scan, map... ?
2. En utilisant le plus efficacement possible la mémoire « shared » réalisez ce calcul dans un kernel. Le calcul doit renvoyer quatre images contenant les 4 distances pour tout pixel, voire une image contenant 4 octets pour les quatre distances (comme une image RGBA en somme).

Étape 4 : Calcul des poids de mélange (100 pts)

Une fois les distances calculées, on peut déduire, pour chaque pixel sur une ligne de discontinuité, sa position relative p ainsi que la taille T du segment. La position relative est le minimum des deux distances aux extrémités du segment. La taille du segment est la somme des distances moins 1.



attention dans l'exemple ci-dessus les distances sont calculées à partir de 0

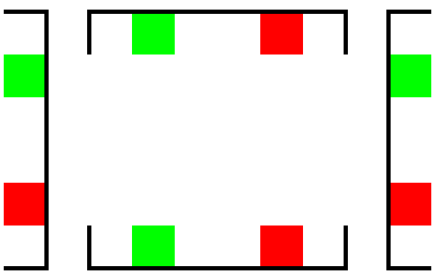
Ainsi pour un segment de discontinuité horizontal, on a :

$$p = min(d_{droite}, d_{gauche})$$
$$T = d_{droite} + d_{gauche} - 1$$

ou d_{droite} et d_{gauche} sont les distances aux extrémités gauche et droite.
Le coefficient de mélange pour le pixel avec son voisin peut alors être calculé grâce à la formule suivante :

$$w = \frac{1}{2}\left(1 - \frac{2p+1}{T}\right)$$

C'est ici qu'une complexité apparaît : ce poids w doit être calculé pour un pixel qui doit se mélanger avec son voisin mais quel voisin ? Pour le savoir, il faut vérifier que le pixel appartient bien à un pixel situé dans une forme en L et laquelle, donc vérifier que le pixel appartient à une des configuration suivante (vert ou rouge) :



Faites un kernel qui calcule, pour chaque pixel, son poids de mélange avec ses 4 voisins. Dans le cas où aucun mélange ne doit être effectué avec un voisin, mettre le poids à 0.

Étape 5 : Mélange final (50 pts)

Faites un kernel où chaque pixel se mélange, quoiqu'il arrive avec ses quatre voisins. L'ordre importe peu (il est très rare qu'un pixel se mélange avec 2 ou plus de voisin).

La totale (variable)

Réalisez des tests sur plusieurs résolutions d'images. Donnez les temps de calcul en ms pour chaque étape et pour l'ensemble du processus.

Le nombre de point offert sur cette partie dépend du nombre de ms T que votre algorithme mettra sur mon ordinateur par rapport à un temps planché P et un objectif O (qui est dépassable).
La formule du nombre de point est :

$$Si\ T > P : 0\ pts$$
$$Si\ O \leqslant T \leqslant P : 100\,\frac{T-P}{O-P}$$
$$Si\ T < 0 : 100\,\frac{O}{T}$$