

DAYAMA SEVERIN
SANI KAKA HADJARA

RAPPORT DE PROJET CUDA

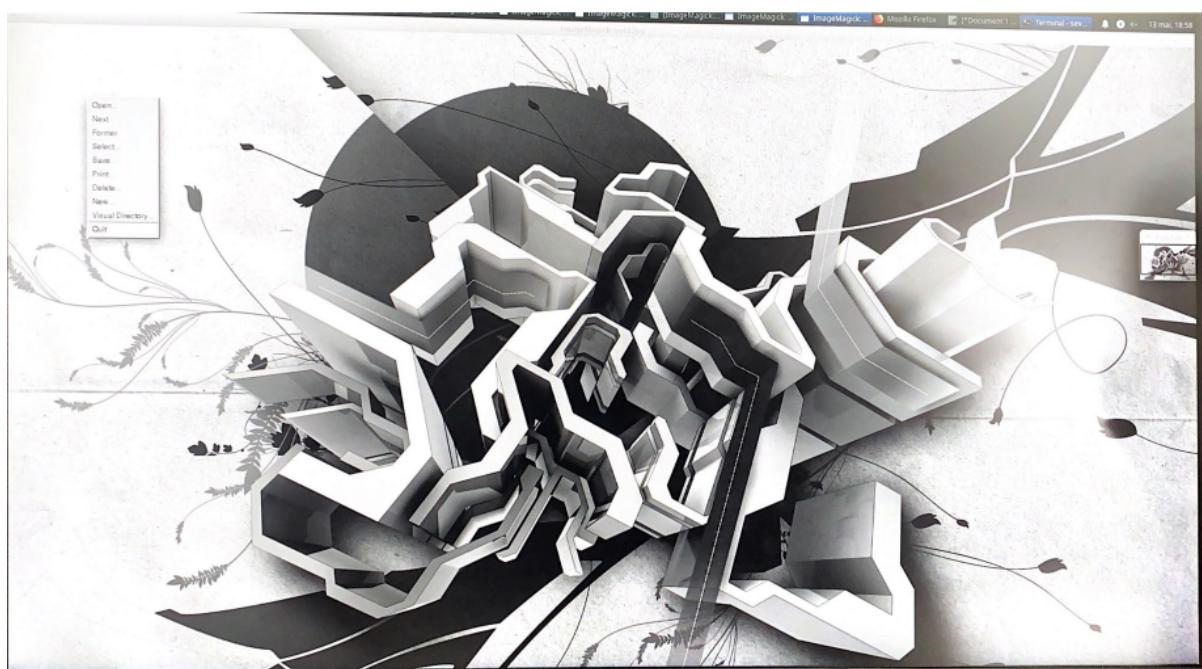
PRÉSENTATION DE L'ALGORITHME :

Les algorithmes de convolution mettent en jeu la multiplication de deux matrices dont l'une est appelée matrice de convolution et présentent donc une complexité arithmétique élevée. La stratégie de programmation massivement parallèle offerte par les GPU permet de réduire considérablement les temps de calculs induits par ces algorithmes.

Notre algorithme se repose sur l'implémentation simple de convolution séparée, comportant trois versions à savoir :

- La version simple testée avec 32 blocs de 4 threads chacun sur une grille dim3 partagée en fonction du nombre de blocs . Ce test nous a donné un temps d'exécution de 0,189568 secondes. La première image ci-dessous est l'image résultante.
- La version de traitement des bordures qui nous permet de traiter les contours de notre image grâce à la méthode **convKernelBorder** qui pour chaque bloc de dimension 3x3 de pixels dans l'image d'origine , récupère pixel par pixel et les fusionne deux à deux pour en créer une autre. Le temps d'exécution de cette version est de 0,21856 secondes.
- La version optimale implémenté directement sur la version séparable qui met en place une mémoire contenant des blocs de pixels, cela nous permet de traiter plus d'un pixel par itération .

Nous définissons les blocs de threads de la même manière que la version simple, par contre la configuration des blocs diffère, les blocs de deux pixels sont superposées afin de ne pas avoir de bande non calculée autour des blocs , par conséquent on crée plus de blocs. Le temps d'exécution de la version optimale est de 0,159586 secondes.



Bordure

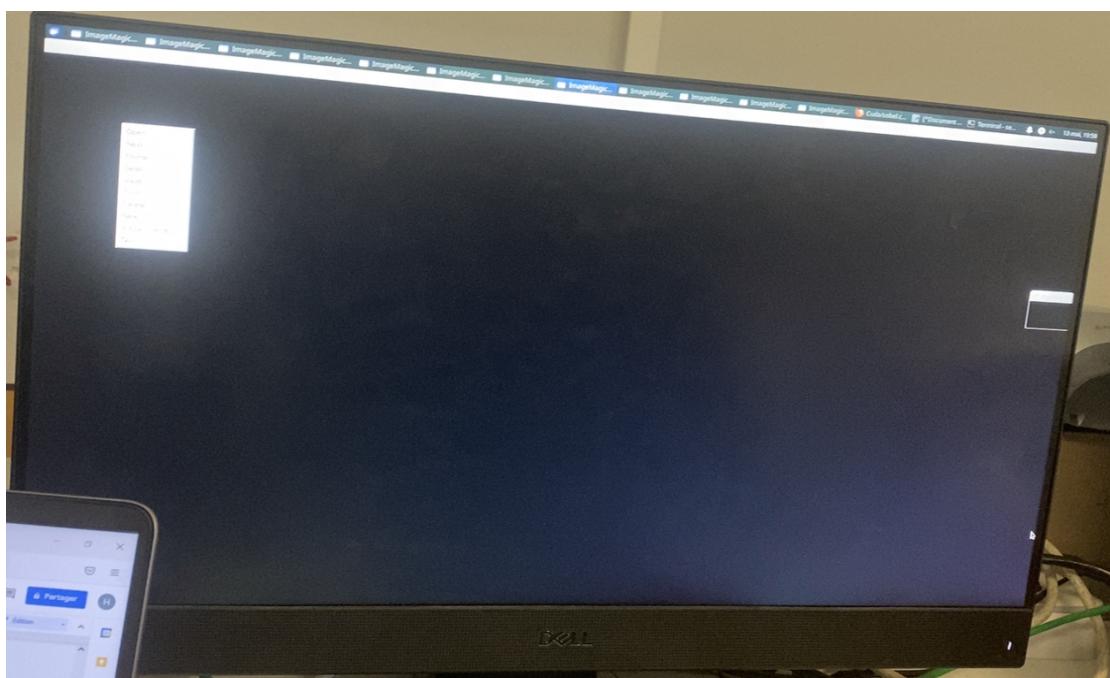


DIFFICULTÉS RENCONTRÉES:

Lors de la réalisation de notre projet nous avons rencontré plusieurs difficultés dont entre autres l'affichage de l'image.

Au début , nous étions confronté à un problème de segmentation de la mémoire qu'on a pu régler en faisant des ajustement au niveau de l'allocation mémoire.

Lorsqu'on exécutait notre programme compilé nous avions l'image résultat qui était créée mais lorsqu'on utilisait la commande **display** pour afficher l'image on avait un écran **noir**.



Ensuite nous avons pu afficher l'image à moitié et triplée , puis nous avons avancé en réglant le problème du triplage .

