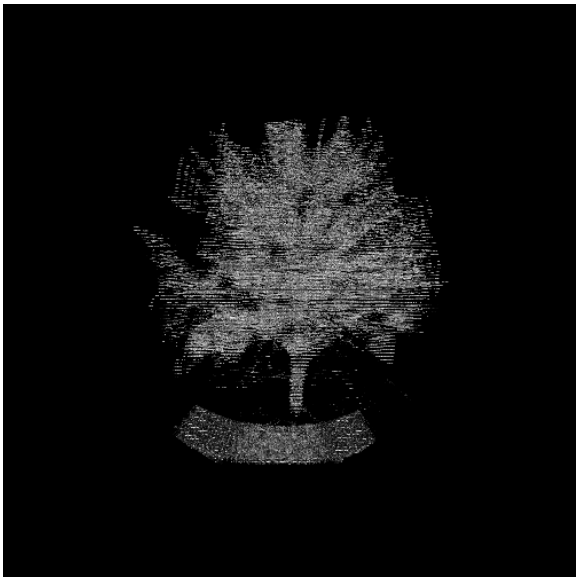
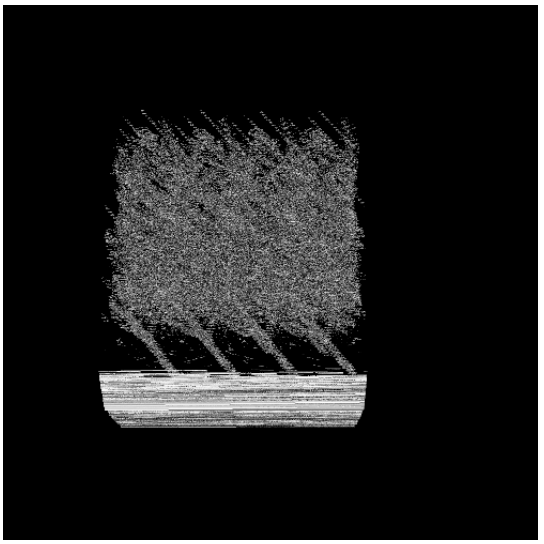
Pierre ZACHARY
2183251
Rendu Projet VTK

Voici mes images pour le projet vtk :

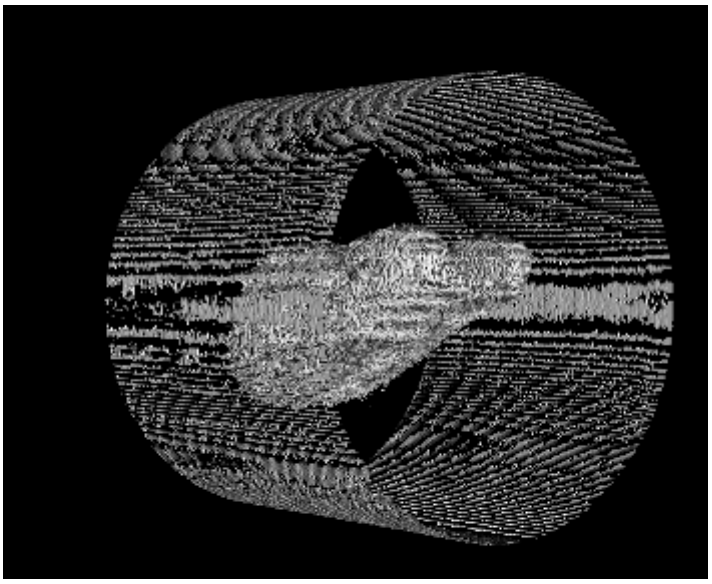
Mystere 1 (tirelire)

Mystere 2 (arbres ?)

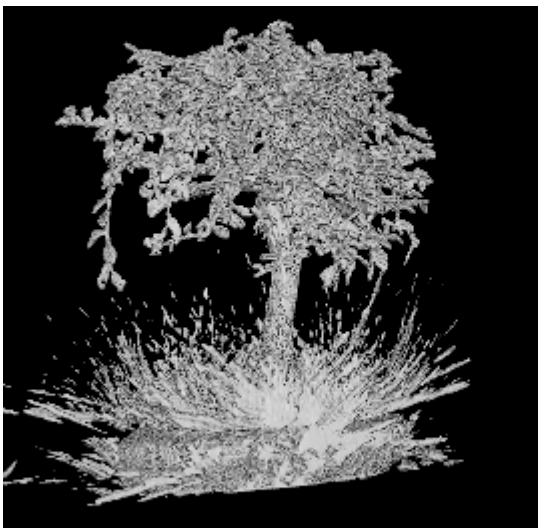




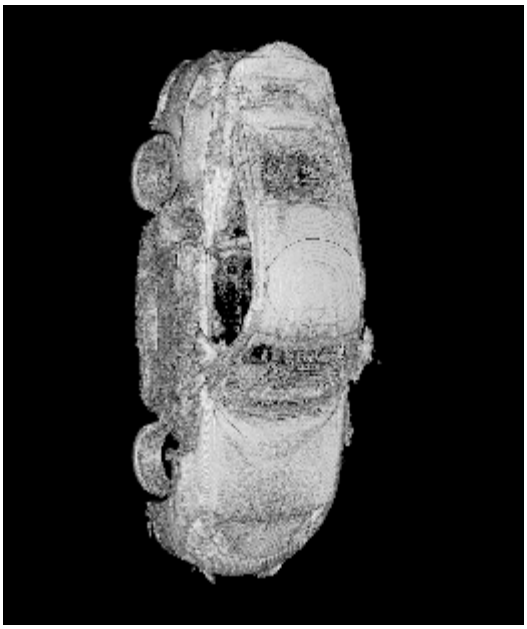
Mystere 4 (dent)



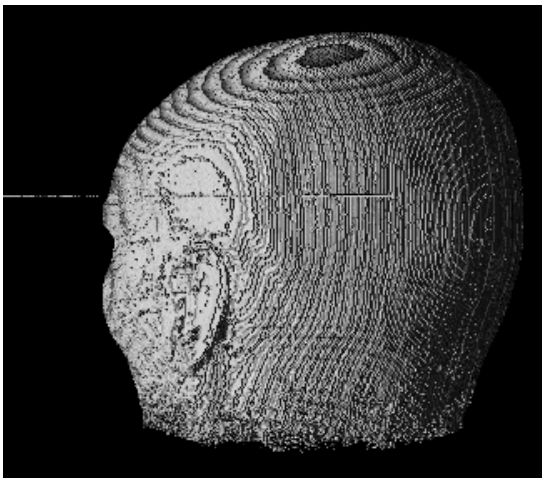
Mystere 5 (arbre seul)



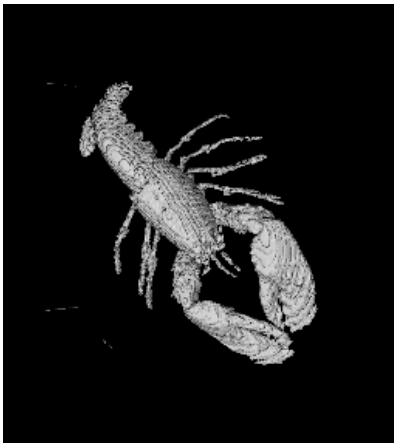
Mystere 6 (voiture)



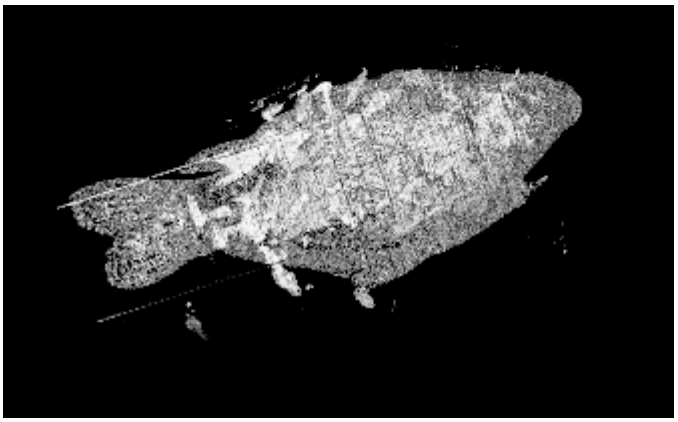
Mystere 8 (tete)



Mystere 10 (homard)



Mystere 11 (poisson)



Je n’ai pas eu le temps de trouver les bonnes valeurs de coupes pour le mystere 9…

Explication du code :

Pour optimiser un maximum le code, j'ai repris les principes de l'out of core et du tp7 avec le parallélisme. Dans les grandes lignes, voici comment fonctionne mon code :

- On répartit les Z entre chaque processus
- Chaque processus va charger une partie de ses Z dans le fichier raw (il divise ses Z par numPasses)
- On fait une boucle pour effectuer plusieurs azimuth et/ou modifier les valeurs de coupe à chercher
- Chaque processus fait le rendu de ses Z en plusieurs fois : il charge les données pour une partie de ses Z, fait le rendu et garde le rendu finale pour ses Z via un zbuffer local
- Une fois que tous les processus ont fait le rendu de leurs Z, on fait un MPI Gather sur root pour récupérer tous les rgba et zbuffer et calculer l'image finale

Cette méthode permet de charger les images en utilisant tous les coeurs et sans avoir besoin d'autant de ram que le fichier raw fait d'octets. L'inconvénient étant qu'il y a plusieurs read du fichier Raw par processus, pour générer chaque rgba locale, avec d'effectuer le rgba finale.

MPI_File_read permettrait d'optimiser cela (optimiser pour le read sur plusieurs processus), mais je ne l'ai pas utiliser pour conserver la fonction ReadGrid.

J'ai passé beaucoup de temps à installer vtk sur windows, mais la façon de lire les fichier n'est pas la même avec windows et donc je n'ai pas pû générer mes images avec.

J'ai fait beaucoup de recherche par rapport aux classes vtk qui exploitent MPI, il y a notamment un renderer qui a l'air de faire le boulot :
https://vtk.org/doc/nightly/html/classvtkCompositeRenderManager.html
Cette classe permet de répartir le rendu entre les différents processus et obtenir le résultat sur la render window de root.

Il y a aussi cette classe : https://vtk.org/doc/nightly/html/classvtkMPIImageReader.html qui est optimisé pour le read de données en parallel, cela implique aussi de changer la vtkRectilinearGrid en vtkImageData, ce qui doit être faisable à priori car ImageData possède un vector de scalar.

Je n'ai pas utilisé ces deux classes dans mon rendu pour respecter les méthodes vu en td mais je tenais à le préciser vis à vis du temps de recherche que ça m'a pris.