Sujet 1 : Fractionnement de maillage

Elias MUNOZ
Léo KAMMERLOCHER
Thomas FOUCOUR

Janvier 2021

Contents

1 Introduction

1.1 Problème

Le but de ce projet est de réaliser un fractionnement de maillage sur un objet 3D pour créer une animation dans laquelle il se casse.

Le projet est disponible sur ce repertoire github

1.2 Solutions existantes

Il existe plusieurs approche pour pouvoir réaliser un fractionnement de maillage qui consiste à faire :

- Une grille de voxels à l'intérieur de l'objet 3D et découper notre objet selon ces voxels.
- Un diagramme de Voronoï 3D.
- Une triangulation de Delaunay 3D en utilisant des tétraèdres et en les basculant.

1.3 Notre approche

Nous avons choisi la méthode du diagramme de Voronoï pour ce projet. Le but sera de générer des points à l'intérieur de notre objet, des graines, qui vont être utilisées pour créer les cellules de Voronoï.

Pour cela nous utilisons un algorithme de Voronoï itératif.

2 Méthode

2.1 Ressource utilisées

Pour ce projet nous avons récupéré le starter des TPs d'animation et rendu que nous avons modifié pour permettre le chargement et l'affichage d'un ou plusieurs objets.

Pour le diagramme de Voronoï, beaucoup de méthodes plus ou moins complexes sont utilisées (dans le cas d'un algorithme recusif utilisant des octrees par exemple). Nous avons cependant opté pour un algorithme naïf, plus simple mais avec un temps de calcul plus élevé.

Pour cette méthode nous avons eu besoin d'effectuer des opérations sur des plans. Pour cela nous avons nous-même implémenté les fonctions nécessaire à l'aide des cours de Fondamentaux de l'Informatique Graphique.

Pour la physique nous avons utilisé le SDK bullet3 qui à permis de donner une enveloppe physique aux objets et de calculer les interactions entre enveloppe physiques.

2.2 Détails de l'implémentation

Nous avons séparé notre code en trois grandes parties :

- La gestion de la géométrie
- La gestion de l'affichage.
- La gestion de la physique et la génération de points.

1. Pour la gestion de la géométrie nous avons créé une classe **Geometry** qui regroupe toute les fonctions effectuant les calculs et opérations dont nous avons besoin. C'est à dire, des fonctions permettant de récupérer des plans, des droite ou encore de gérer les intersections et de faire un changement de base. Nous avons ensuite créé une classe **Voronoï** permettant de générer notre diagramme de Voronoï.

2. Pour le projet, notre programme doit pouvoir afficher plusieurs Objets ainsi que leur boites englobantes. Pour cela, nous avons implémenté une classe **Mesh** disposant de trois vbos, une permettant l'affichage des faces, un autre pour l'affichage des arêtes et le dernier pour l'affichage de sommets. Les informations sur la couleur, taille et position des faces, arêtes et sommets sont obtenus en parcourant le MyMesh associé à la classe **Mesh** et sont par la suite charger dans les vbos. Le programme permet de changer entre deux modes de vue (normal ou wireframe). Dans le mode wireframe, seul la boite englobante et les arêtes des objets s'afficheront.

3. Pour générer les graines nous avons implémenté deux méthodes dans la classe **SeedGenerator**, dans lesquelles nous choisissons une répartition régulière ou aléatoire des graines.
   Pour la gestion de la physique des différents objets crées nous avons créé une classe **Physics** qui contient les méthodes pour initialiser et terminer un monde dynamique, ainsi que la création des différents objets dynamiques ou non.

2.3 Expérimentation

3 Résultat

Nous avons animé un cube qui tombe sur un sol avec un autre objet qui n'a rien à voir avec le diagramme de Voronoï. Nous n'avons pas réussi à récupérer la bonne droite d'intersection entre deux plans. Nous avons le bon vecteur mais pas le bon point et nous ne comprenons toujours pas pourquoi ça ne marche pas.
Les points que nous récupérons à la fin de la fonction "compute_voronoi" ne sont donc pas les bons et ne permettent pas de générer les cassures du cube.

4 Conclusion

4.1 Discussion

Grâce a ce projet nous avons pu aborder l'implémentation des diagramme de Voronoï dans un espace 3D. Pour ce faire nous avons dû apprendre des méthodes pour récupérer les intersections de droites et de plans. Nous avons aussi dû chercher des méthodes pour pouvoir réaliser l'animation des objets.

4.2 Améliorations possibles

Beaucoup d'améliorations peuvent être apporté aux programmes. Une fonction permettant de générer des points d'une grille est implémenté dans le programme, on pourrai donc fractionner le cube en plusieurs cubes de plus petites tailles ou en différents objets. On peut aussi procéder a une coupe de l'objet en générant les graines sur une même ligne et en appliquant l'algorithme de Voronoi entre ces points.

La complexité de notre algorithme peut être amélioré en utilisant un algorithme diviser pour regner de voronoi avec l'utilisation d'octree.

4.3 Travail d'équipe

Ce projet a été réalisé en équipe en suivant une méthode agile simplifiée. Nous avions un à deux rendez-vous par semaine pour parler de l'état d'avancement du projet et des taches restées en suspend mais aussi pour créer et répartir de nouvelles tâches. Chaque collaborateur travaillait sur une des trois parties principales du projet (Géométrie, Affichage et Ajout de la physique aux objet et de la génération de graines). Un collaborateur pouvait venir en aide sur une autre partie lorsqu'il n'avait plus de tâches à son actif ou qu'il avait trouvé une meilleur implémentation pour une fonction.

Nous avons utilisé Github pour travailler sur notre projet. Chaque collaborateur travaillait sur sa branche et une fois les tâche effectués il la fusionnait avec la branche develop Comme chaque partie possède ces propres classes, nous avons eu très peu de conflits à gérer.

Pour la création et répartition de tâches, nous avons utilisé Zenhub.