Introduction à la Synthèse d'Images Réalistes
Modélisation
Rapport de Projet

Yoann SOCHAJ / Benjamin Maurel

Mai 2022

1 Introduction

Le but du projet de reproduire une démonstration javascript traitant des surfaces implicites. Il fallait implémenter cette démonstration, mais avec Processing. Mais nous ne partions pas de rien. Nous partions d'un code processing fonctionnant, mais en 2D.

Pour nous, le sujet se découpait en deux parties. Une partie de subdivision d'un cube en tétraèdre et l'autre l'adaptation du code 2D en 3D.

1.1 Organisation

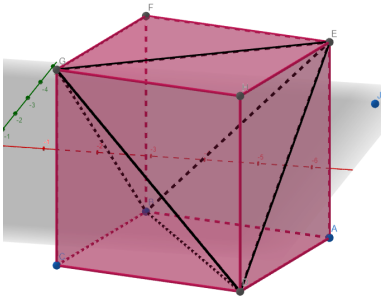
Nous avons travailler sur le sujet ensemble à distance, en utilisant plusieurs outils comme git ou discord. Nous n'avons pas de tache précise à réaliser personnellement, juste quand nous avions le temps, et que nous avions envie, on avait un petit peu en prévenant l'autre personne de l'avancée.

1.2 Guide d'emploi

Nous avons décider d'utiliser PeasyCam, afin de pouvoir bouger la caméra de façon plus agréable. On peut utiliser le clique droit pour zoomer/dezoomer, le clique gauche pour une rotation, et le cliques molettes pour se déplacer. Mais, il y a un problème si vous n'avez pas la library, nous l'avons donc désactiver mais vous avez juste à de-commenter au cas où.

2 Subdivision d'un cube en tétraèdre

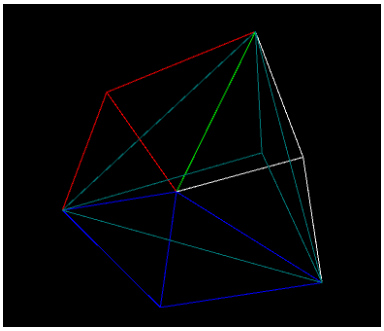
Dans un premier temps, nous avons donc réfléchis voir en combien de tétraèdre subdiviser notre cube. Après pas mal de réflexion et de test sur geogebra nous avons pensé à 6 tétraèdre. Mais après avoir discuté avec d'autres personne de la promo, nous avons trouvé une solution à 5 tétraèdre comme ceci:



On peut y avoir un tétraèdre central, puis les 4 autres sont les coins du cube.

Nous avons donc commencé à vouloir juste afficher 1 cube, avec sa subdivision. Pour plus de lisibilité dans notre code, nous avons choisi de faire une classe CUBE, qui va nous permettre d'avoir les éléments d'un cube, à savoir son point haut gauche, et les tétraèdre qui le compose. Un tétraèdre est une classe qui possède une liste de points.

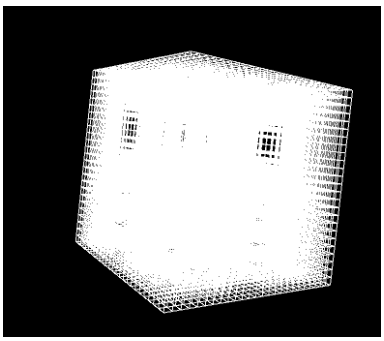
Afin de faire notre cube, il fallait en premier temps retrouvé les autres ponts à l'aide de notre seul point. Pour ce faire, nous avions une variable rez, qui va nous dire la taille des cubes. Et grâce à ceci, on peut retrouver tous les points. Ensuite grâce à notre schéma geogebra, il fallait simplement créer des tétraèdre avec les bons points. Et on arriva à un résultat plutôt convainquant:



Une fois nos cubes bien formés, il fallut adapter le code pour prendre en compte la 3D !

3 Adaptation du code

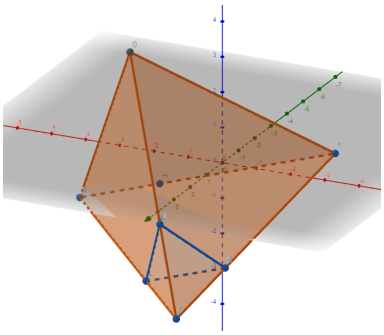
Maintenant que notre subdivision sur un cube marche, il faut adapter la scène 3D pour au lieu de prendre en compte des carrées, prendre en compte des cubes. Nous avons déjà essayé de générer des cubes (sans tétraèdre) afin de savoir si notre espace était bon. Et il l'était :



Ensuite, nous avons adapté le code de l'implicite particules afin de prendre 3 coordonnées, car avant ils prenaient seulement x et z.

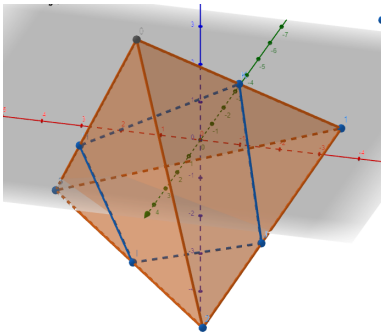
Maintenant que notre boucle qui génère chaque cube fonctionne, il va falloir faire une boucle qui va faire, pour chaque tétraèdre du cube, afficher certains triangles. Nous nous sommes aidé encore une fois de geogebra, afin d'avoir la visualisation de quel point de cube prendre en compte lors de notre création de shape pour faire des triangles.

Dans un premier temps, nous avons donc, comme le sujet nous le demande, décomposer chaque cas en nombre binaire. Exemple sur le cas "0001", il y a un point du tétraèdre qui est à l'extérieur, on doit afficher un triangle sur les 3 arêtes qu'il compose comme ceci:

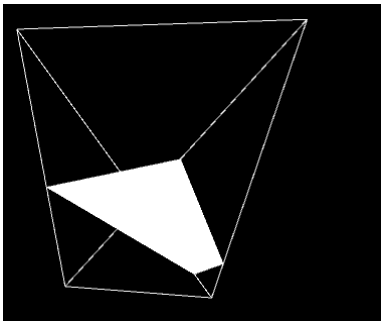


Mais ceci est un cas simple, car il faut "seulement afficher un triangle" en fonction du point le plus proche à la surface calculé.

Mais il y avait un cas plus complexe où on ne pouvait pas se restreindre à un seul triangle. Il fallait afficher une surface comme ceci :

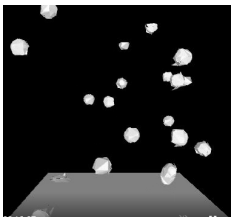


Nous avons donc pensé à décomposer cette surface en deux triangles, et en l'affichant, nous obtenions ce résultat :



Une fois notre variable state, qui va nous dire dans quelle configuration nous sommes, va être opérationnelle, nous aurons plus qu'à faire un switch afin de faire construire la forme de cette situation.

Une fois tout mis en place dans la boucle nous obtenons un résultat comme ceci:



4 Problèmes rencontrés

Le premier problème rencontrer pour nous était, qu'on ne savait pas où commençait, on avait eu pas mal de mal à se lancer vers une bonne direction que ce soit en terme de réflexion ou de coding.

Un second problème fut un triangle qui semblait anormal. Nous avons donc du debuger tout notre programme afin de trouver lequel c'était et de le corriger.

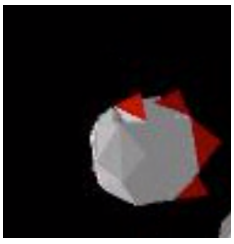


Figure 1: Triangle anormal en rouge

Et un dernier problème, c'est concernant notre résultat final. On y voit encore quelques triangles anormaux et des trous. Nous pensons que c'est dut à la création et surtout l'orientation des tétraèdre dans l'espace 3D générale. Par exemple, deux cubes à côté, ne doivent pas avoir la même orientation de tétraèdre. Mais nous ne sommes pas arrivé à des résultats probants.

5 Conclusion

Ce projet pour nous a était vraiment une bonne expérience. Grâce à celui-ci, nous avons appris à mieux comprendre et utiliser les surfaces implicites. Étant donné que c'est une matière importante de notre cursus, nous sommes ravi d'avoir fait un projet sur la partie modélisation du cours d'ISIR afin d'enrichir nos connaissances.