

RAPPORT DE PROJET IN55

SUJET : Modélisation 3d d’objet mathématiques

IOVLEFF Stéphane • CASTELLON Valentin • BENIS Antoine

Responsable d’UV : LAURI Fabrice

Sommaire

Présentation du sujet . . . . . . . . . . . . . . . . . . . . . 3

Conception du projet . . . . . . . . . . . . . . . . . . . . . 3

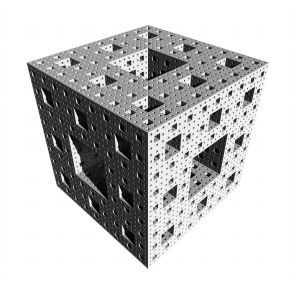
Utilisation d’OpenGL . . . . . . . . . . . . . . . . . . . . . 3

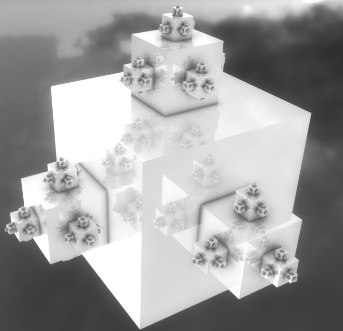
Résultats obtenus . . . . . . . . . . . . . . . . . . . . . . . . ?

Améliorations possibles . . . . . . . . . . . . . . . . . . . ?

Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ?

Présentation du sujet :

Dans le cadre de l’UV IN55, nous avons décidé de choisir le sujet #2 : modélisation 3D d’objet mathématiques. Ce sujet consiste à modéliser des solides platoniciens ou des solides archimédiens pour ensuite créer des fractales à l’aide de ces solides (exemples : Pyramide de Sierpinski, éponge de Menger). Nous avons eu le choix du solide et de la fractale à modéliser.



Nous avons décidé de modéliser une fractale à partir d’un cube. Cette fractale est construite de façon contraire par rapport à l’éponge de Menger puisqu’en effet, au lieu de retirer un cube au centre de chaque face, nous allons en ajouter un.

L’objectif principal sera donc de modéliser cette fractale à l’aide de python et OpenGL.

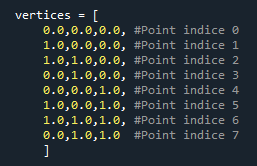
Conception du projet :

**Etape 1 : Le cube**

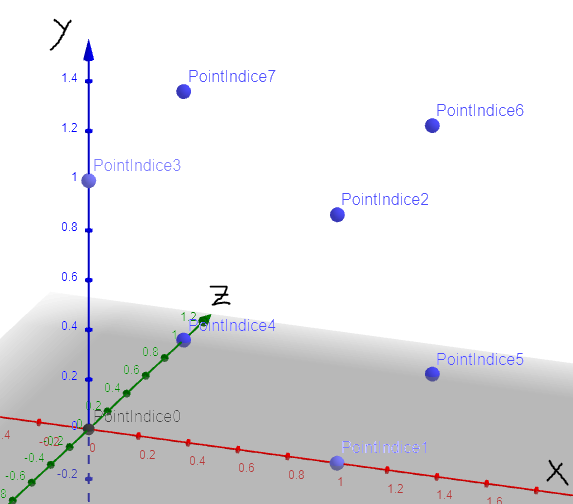
La base de ce projet était la création de cube en 3D. Il fallait donc d’abord mettre en place les outils nécessaires afin de modéliser un cube. Pour cela, il a fallu s’inspirer du premier TD.

Le principe est de renseigner une table des vertices (positions de chacun des 8 sommets dans l’espace). On va également indexer ces points.

Exemple de table des vertices :

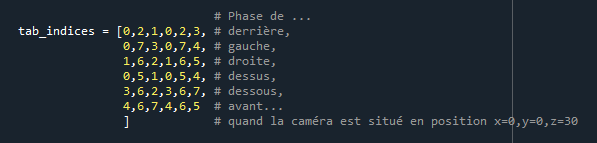


Rendu sur un graphe 3D de chacun des vertices de cette table :

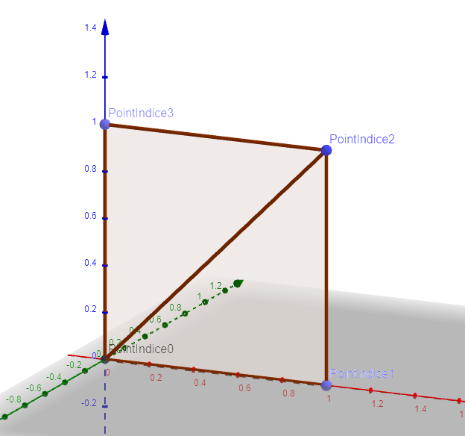


A l’aide de la primitive OpenGL : GL\_TRIANGLES, on va pouvoir être en mesure de relier 3 vertices entre elle pour pouvoir créer des triangles. Ainsi, pour faire une face, on a besoin de 2 triangles rectangle possédant la même hypoténuse.

Pour créer chaque face, on va utiliser une table des indices qui va permettre de relier 3 vertices entres-elles pour former un triangle.

Exemple de la table des indices :

Rendu sur un graphe 3D de la création d’une face :



Le cube étant créé, la première étape est finie.

En guise d’amélioration, au lieu d’avoir des vertices fixes, on a décidé de faire en sorte de pouvoir placer notre cube sur le graphe en renseignant la position d’une vertice. Ainsi, à partir de la vertice de coordonnée choisie et de la taille du cube, on est en mesure de placer les 7 autres vertices pour pouvoir créer notre cube.

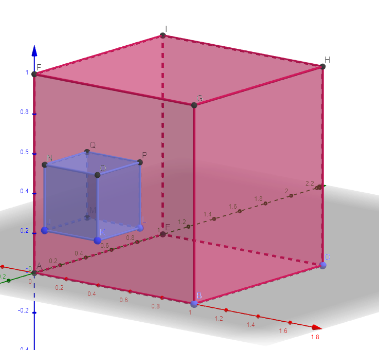
**Etape 2 : Création de la première génération de la fractale**

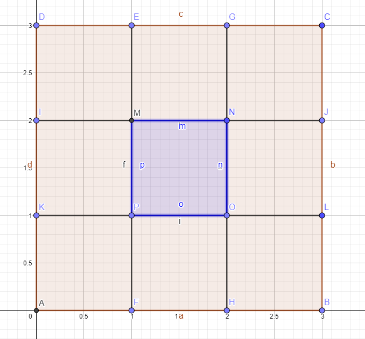
Le cube étant créé, la prochaine étape du projet a été de créer la première étape de notre fractale. Il faut dans un premier temps créer les cubes supplémentaires à rajouter à notre cube initial créer lors de l’étape 1. Puis, dans un second temps, il faut placer tous ces cubes aux bons endroits.

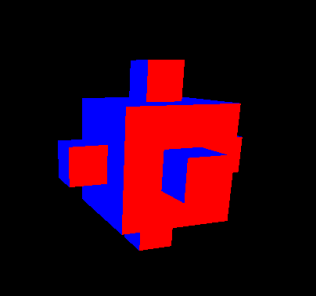
Pour créer les cubes supplémentaires, on rappel la fonction qui créer les cubes (fonction faite lors de la première étape). On veut rajouter un cube chaque face de notre cube initial. On va donc créer 6 cubes.

En ce qui concerne le placement, on place chacun des nouveaux cubes sur le centre des faces du cube initial. On peut réaliser un découpage d’une face d’un cube en 9 cubes de taille 3 fois plus petits que la face initiale. Ainsi, les nouveaux cubes créés sont de taille 3 fois plus petits que le cube de génération précédente.

Découpage plus placement d’un nouveau cube sur une face :





A partir du cube initial, on est donc en mesure de placer les 6 nouveaux cubes en renseignant la position d’une vertice (la taille des nouveaux cubes est déjà connue car elle est 3 fois plus petite que l’ancienne)

Ainsi, on obtient notre génération 1 de notre fractale.

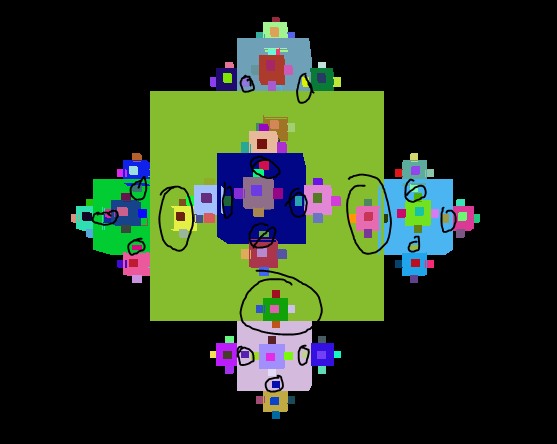
**Etape 3 : Généralisation – création de n génération de la fractale**

Le but maintenant est d’obtenir notre résultat final à savoir notre fractale. On va généraliser le raisonnement sur chaque cube créé lors de l’étape précédente. Le travail va être le même pour chaque génération : lors de la création de la génération n, pour chaque cube créé lors de la génération n-1, on construit 6 cubes que l’on place aux centres de chaque face.

Une fois la généralisation effectuée, la construction de la fractale est terminée.

**Etape 4 : Optimisation**

Certains cubes ne sont jamais visibles puisqu’ils sont à l’intérieur d’un autre. Quand on ne construit pas une face d’un cube, on est en mesure de voir les cubes à l’intérieur du cube de génération précédente.



Les cubes entourés en noir ne sont normalement pas visible et donc construit pour rien.

Afin de réduire l’utilisation de ressource nécessaire à la construction de ces cubes

Pour cela, on va utiliser un flag dans notre algorithme.

Utilisation d’OpenGL :

Pour gérer le coté affichage de nos résultats, on utilise OpenGL, une librairie graphique 3D. Cette libraire va permettre de réaliser tous les tracés, d’afficher les couleurs ou encore de gérer la fenêtre d’affichage et la caméra (point de vue).

Résultats obtenus :

**Avant le lancement du programme :**

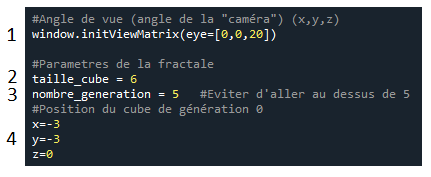
Avant de lancer le programme, il faut définir les différentes caractéristiques de notre future modélisation 3D :

1) La position de la caméra dans l’espace

2) La taille du cube de première génération

3) Le nombre de génération créée

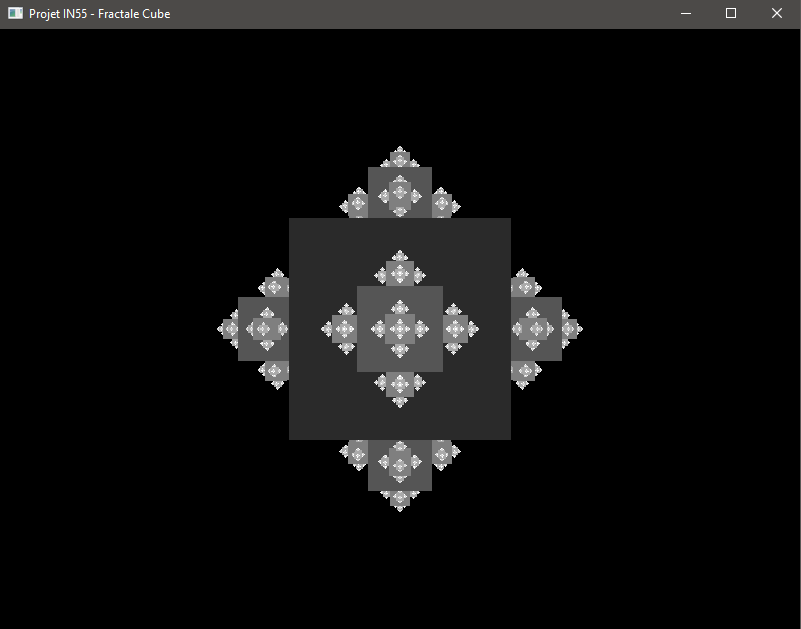
4) La position du cube de première génération dans l’espace



Dans cet exemple, on aura un cube initial de taille 6 unités. La position de la vertice située en haut arrière gauche (quand la caméra est située en position x=y=0 et z>0) est (-3,-3,0). Enfin, on a choisi d’avoir 5 générations de notre fractale.

**Après le lancement du programme :**

Le programme va générer une fenêtre dans lequel va se trouver notre fractale.



Améliorations possibles :

Ajout des ombres/lumières : Pour ajouter un effet plus réaliste à notre

Conclusion :

Annexe :

Algorithme de création de fractale :

On considere les fonctions suivantes :

• creerCube(taille\_cube,x,y,z,couleur) qui permet de créer un cube

• ajouterQueue(e,L) qui permet l'ajout de l'élément e à la fin de la liste L

Données :

• taille\_cube : la taille du cube de génération 0

• m : numéro de la génération à construire (vaut 0 au lancement de l'algorithme)

• n : nombre total de génération à construire

• L : une liste vide

• x, y, z : position d'une vertice du cube de génération 0

• flag : entier servant de flag pour ne pas construire de cube derrière une face

Résultat :

• L : une liste contenant les données de tous les cubes crée (position, taille, ...)

