INFSI 350 – Informatique Graphique 3D et Réalité Virtuelle
Travaux Pratique

Modélisation et Traitement Géométrique

Base de code
La base de code pour ce TP est disponible ici :
http://www.telecom-paristech.fr/~boubek/ens/rv/tp/infsi350_tp_mod.zip
Télécharger et décompresser l'archive.

Dans le repertoire infsi350_tp_mod, le fichier main.cpp contient l'application C++/OpenGL/Glut de base au TP.

Note : tous les traitements implémentés seront appliqués au maillage courant à l'aide de touches claviers, afin de pouvoir enchainer plusieurs traitements à la suite. Ceci peut se faire en modifiant la fonction « void keybord (unsigned char, int, int). »

I. Structure Simple de Maillage 3D

Le fichier *gmini.cpp* contient une class maillage miniale avec une classe Sommet (*Vertex*) et *Triangle* minimales également. Le fichier *gmini.cpp* contient également une fonction « *draw ()* », dessinant le maillage stocké dans l'instance globale *mesh*.

I.a Observer le code.

Notez que quelques méthodes simples sont fournis, notamment la méthode de chargement de maillage depuis une fichier OFF (loadOFF) et une méthode simple de calcul des normales (recompute normales).

Le format OFF est un format ASCII simple de maillage. Exemple :

```
OFF
3 1 0
1.0 1.0 0.0
1.0 -1.0 0.0
0.   0.0 0.0
0 1 2
```

Ici le modèle OFF contient 3 sommets, indexés par un triangle. Un fichier sphere.off est chargé par défaut. D'autres fichiers OFF sont contenus dans l'archive (repertoire models).
./gmini fichier.off pour les afficher.
Vous pouvez créer vos propres OFF avec blender, en vous assurant que toutes les faces sont converties en triangles.

I.b Créer une méthode void *makeCube ()* pour générer le maillage d'un cube unitaire. Remplacer le chargement du modèle OFF par la génération d'un cube. Faire de même pour une méthode void *makeSphere (unsigned int resU, unsigned int resV). Penser à rajouter une méthode void clean ().*

I.c La fonction de calcul des normales par sommet utilise une pondération uniforme par face incidente. Remplacez-la par une pondération par les angles des faces formés aux sommets. Comparez les deux versions sur le modèle *monkey.off*.

II. Lissage

On se propose d'implémenter un opérateur de lissage de maillage sous le forme d'un filtrage laplacien. On pourra tester sur le modèle *max_50k.off*.

II.a Implémenter une méthode void *smooth ()* effectuant un filtrage laplacien du maillage. Cette méthode modifiera la position de chaque point en les déplaçant le long du vecteur laplacien (topologique).

Principe :

1. Calculer le barycentre du 1-voisinage de chaque sommet
2. Déplacer chaque sommet vers son barycentre associé
3. Recalculer les normales

II.b Ajouter une paramètre de control *alpha*, compris entre 0 et 1 permettant de moduler le filtrage (1 = déplacement complet). Tester en associant les touches '1', '2' et '3' à l'application d'un lissage avec *alpha* égal à 0.1, 0.5 et 1.0. *Penser à rajouter un raccourci clavier permettant de recharger l'objet d'origine.*

Bonus. *Créer une classe HalfEdgeMesh équipée de demi-arêtes et implémenter le lissage laplacien via l'opérateur de Laplace-Beltrami à poids cotangents.*

III. Simplification

III.a Implémenter un opérateur de simplification de maillage *void simplifyMesh (unsigned int r)* qui applique la simplification partitionnement spatial en grille du maillage. On considérera simplement le représentant moyen par cellule.

Principe :

1. Calculer un cube englobant C le maillage M. L'élargir légèrement.
2. Créer un grille uniforme G de resolution r à l'interieur de C.
3. Pour chaque sommet du maillage, ajouter sa position et sa normale au sommet représentant de la cellule de G qu'il intersecte. Normaliser tous les représentants à la fin.
4. Pour chaque triangle du maillage, ré-indexer ses trois sommets sur les sommets représentants de leurs cellules respective si les trois cellules sont différentes ; éliminer le triangle sinon.
5. Le maillage simplifié est formé des représentants non nuls de la grille et de la liste des triangles ré-indexés. Penser à recalculer les normales avant affichage.

III.b Tester à divers resolutions en associant les touches '4', '5' et '6' aux simplification à résolution 64x64x64, 32x32x32 et 16x16x16.

Bonus. Implémenter une méthode s*implifyAdaptiveMesh* (unsigned int n) basée sur un octree (limité à n sommets par feuilles) en lieu et place de la grille uniforme (touche '9').

IV. Subdivision

IV.a Implémenter la subdivision de Loop sous forme d'une méthode v*oid subdivideLoop ()* de la

classe *Mesh*. Pour ce faire, utiliser la stratégie par table de hashage vue en cours et décrite dans l'article « A factored Approach to Subdivision Surfaces », Warren and Schaefer, 2004 (http://faculty.cs.tamu.edu/schaefer/research/tutorial.pdf). Associer la touche '8' à l'application d'une étape de subdivision.

IV.b Tester sur le modèle *double-torus.off*. Qu'observez-vous après suffisamment de passes de subdivision.