R6.05 - Modelisation géométrique
TP2 - Manipulation de maillages triangulaires

Rémi Synave

Contexte de travail

Pour réaliser ce TP, vous utiliserez le langage que vous voulez parmi les quatre suivants : python (procédural ou objet), C, C++ ou java. Lisez attentivement le sujet en entier et identifiez bien les notions qui seront à utiliser pour faire votre choix.

Dans le reste du document, les instructions sont données pour un développement objet mais vous pouvez parfaitement faire un développement en langage procédural (C ou python).

Il est conseillé de travailler sous linux avec un éditeur de code (VSC, sublime text ou autre) et de compiler/exécuter avec un terminal (celui intégré à VSC compris).

Rappel de cours - Le maillage triangulaire

Un modèle numérique est la représentation numérique d'un objet. Il existe 4 familles de modèles numériques : les modèles volumiques continus, les modèles volumiques discrets, les modèles surfaciques continus et les modèles surfaciques discrets.
Ces représentations sont utiles dans des domaines aussi variés que : les jeux vidéos, le cinéma, l'architecture, la médecine, etc.

Le maillage triangulaire est une représentation surfacique discrète d'un objet :
— Surfacique car seule la surface de l'objet est modélisée.
— Discrète car elle est approximée par un ensemble fini d'éléments.

Comme son nom l'indique, le maillage triangulaire représente la surface d'un objet grâce à des triangles. La surface peut être couverte d'une texture pour un rendu réaliste de l'objet. Bien qu'il soit possible de créer beaucoup de petits triangles pour représenter la surface le plus fidèlement possible, celle-ci reste approximée. Seules les surfaces planes (assez rare dans les objets réels) sont modélisables de manière exacte. Dans ce TP, nous ne nous intéresserons qu'à la géométrie de l'objet.

Les triangles sont formés par 3 sommets eux mêmes caractérisés par une position `(x,y,z)` dans l'espace. Il existe de nombreuses méthodes pour stocker les informations d'un maillage triangulaire. Nous nous intéresserons ici à la structure de données suivante :

- Des sommets stockés dans un tableau. Chaque sommet est caractérisé par ses 3 coordonnées (x,y,z). Il est aussi identifié par un indice (le numéro de la case du tableau). De plus, chaque sommet aura une liste de sommets voisins. Les voisins sont les sommets directement accessibles en suivant une arête en partant du sommet de départ. Ces voisins seront identifiés par leur indice dans le tableau.
- Des faces stockées dans un tableau. Chaque face est caractérisée par les 3 indices des sommets formant le triangle. Chaque face est également identifée par un indice (le numéro de la case du tableau).

Les formats de fichiers

Les maillages triangulaires peuvent être stockés dans des fichiers pour lesquels il existe de nombreux formats. Nous nous intéresserons ici à deux formats de fichiers relativement simples et qui sont supportés par un grand nombre de logiciel de modélisation : `ply` et `stl`. Ces deux formats de fichiers sont compatibles avec `blender`.

Ces deux formats sont tous les deux disponibles sous forme ascii (lisible avec un éditeur de texte classique) et binaire (non lisible avec un éditeur de texte classique). Nous nous concentrerons sur les versions ascii.

Le format de fichier `PLY`

Ce format de fichier propose de stocker des maillages (triangulaires ou non) et tout un ensemble d'autres informations comme la couleur par exemple. Nous nous limiterons aux informations géométriques : la position des sommets et la liste des faces. Nous faisons l'hypothèse que tous les fichiers `PLY` sont en format ascii et structurés suivant les trois sections suivantes : 1 - en-tête, 2 - liste des positions des sommets, 3 - liste des faces.

Exemple d'un cube stocké dans un fichier ply ascii :

```
ply
format ascii 1.0
comment zipper output
element vertex 8
property float x
property float y
property float z
element face 12
```

```
property list uchar int vertex_indices
end_header
-1 -1 -1
1 -1 -1
-1 1 -1
1 1 -1
-1 -1 1
1 -1 1
-1 1 1
1 1 1
3 3 1 0
3 2 3 0
3 3 7 1
3 5 1 7
3 6 5 7
3 5 6 4
3 4 6 2
3 4 2 0
3 6 7 3
3 3 2 6
3 1 5 0
3 4 0 5
```

Ce qui se trouve entre `ply` et `end_header` est l'en-tête. Cet en-tête donne des informations importantes. Deux élements nous intéressent : le `8` suivant le `element vertex` correspond au nombre de sommets et le `12` qui suit le `element face` qui donne le nombre de faces. Ici, notre cube est donc modélisé grâces à 8 sommets et 12 faces (6 faces carrées découpées en deux triangles).

Les 8 lignes suivantes correspondent à la position X Y Z des sommets. La première ligne de la liste de sommet correspondra au sommet d'indice 0.

Les 12 lignes suivantes donnent la composition de chaque face. Le format `PLY` permet de stocker des maillages triangulaires ou non. Le premier nombre de chaque ligne correspond au nombre de sommets contenus dans la face. Dans notre cas, ce sera toujours 3 pour correspondre à un maillage triangulaire. Les trois nombres suivants correspondent à l'indice de chaque sommet composant la face.

Le format de fichier `STL`

C'est le format de fichier standard et lisible par tous les logiciels. Bien qu'il soit très utilisé, ce format de fichier est pourri car il introduit beaucoup de redondances d'information et permet difficilement de gérer le voisinage.

Il est également disponible sous format ascii et binaire et ne permet, cette fois, de ne stocker que des maillages triangulaires. Il demande également de stocker les normales à

chaque face. Cette information est obligatoire mais peut être remplie de 0. Nous nous limiterons encore au format ascii.

La structure d'un fichier au format stl est la suivante :

```
solid nomDuSolide
facet normal Nx Ny Nz
    outer loop
        vertex v1x v1y v1z
        vertex v2x v2y v2z
        vertex v3x v3y v3z
    endloop
endfacet
...
facet normal Nx Ny Nz
    outer loop
        vertex v1x v1y v1z
        vertex v2x v2y v2z
        vertex v3x v3y v3z
    endloop
endfacet
endsolid nomDuSolide
```

Le fichier commence par une ligne contenant `solid` suivi d'un nom de solide.

Ensuite, le fichier est complété par un ensemble de groupes de 7 lignes.
Ce groupe de 7 lignes correspond à une face du maillage. La première ligne est l'information de normale de la face. Si l'information de normale n'est pas disponible ou non nécessaire, il est possible d'y mettre :
`facet normal 0 0 0`
La ligne suivante sera toujours `outer loop` pour indiquer le début de la description d'une face par ses trois sommets. Les trois sommets seront ensuite définis par leurs trois coordonnées X Y Z précédées du mot-clé `vertex`.
La description de la face se termine par les deux lignes `endloop` et `endfacet`.
Ce groupe de lignes est répété autant de fois qu'il y a de faces.

Finalement, le fichier se termine par `endsolid` suivi du nom de solide choisi en début de fichier.

TODO

Seul les deux premiers items doivent être traités en priorité. Tous les autres peuvent être traités dans l'ordre que vous souhaitez. Suite au développement d'une méthode, écrivez un test qui sauvera le résultat du traitement dans un fichier et vérifiez le résultat grâce à Blender.

1. Écrire une classe `Sommet`, une classe `Face` et une classe `Maillage`. Les attributs des classes `Sommet` et `Face` sont donnés dans le dernier paragraphe de la section **Rappel de cours**. La classe `Maillage` doit contenir un constructeur qui initialise l'objet grâce aux données contenues dans un fichier `PLY`.
   Les données de voisinage sont construites lors de la lecture des faces. Si une face est composée des sommets `s1`, `s2` et `s3`, alors, on ajoute `s2` et `s3` à la liste des voisins de `s1` **à condition que ceux-ci n'y soient pas déjà présents**. Idem avec `s1` et `s3` pour `s2`. Idem avec `s1` et `s2` pour `s3`.
2. Écrire une méthode permettant de sauver le maillage dans un fichier au format `PLY`.
3. Écrire une méthode qui calcule la surface de l'objet. La surface de l'objet est la somme des aires des triangles le composant.
   Il est conseillé d'écrire une classe `Vecteur3D` dont l'un des constructeurs prendra en paramètre deux objet `Sommet` et d'y ajouter une méthode pour calculer sa norme (sa taille) ainsi que le produit vectoriel par un autre vecteur. Vous pourrez ainsi facilement calculer l'aire d'un triangle en vous référant à `https://fr.wikipedia.org/wiki/Aire_d%27un_triangle#À_partir_des_coordonnées_des_sommets`
4. Écrire une méthode permettant d'inverser les normales des faces du maillage triangulaire. Il s'agit ici de parcourir toutes les faces et d'inverser deux indices de sommets sur les trois.
5. Écrire une méthode permettant de centrer le maillage. Pour ce faire, parcourez tous les sommets pour trouver quels sont les minimum et maximum suivant les coordonnées X, Y et Z. Il vous faudra ensuite translater tous les sommets pour que le minimum soit égale à l'opposé du maximum. N'hésitez pas à ajouter des méthodes intermédiaires comme une méthode `translater` dans la classe `Sommet`.
6. Écrire une méthode permettant de faire une homotéthie du maillage.
7. Écrire une méthode pour bruiter le maillage. Cette méthode devra prendre en paramètre un coefficient pour contrôler la force du bruitage.
8. Écrire une méthode permettant de sauvegarder le maillage dans un fichier au format `STL` puisau format `COLLADA` donc d'extension `dae`.

9. Pour les plus rapides : écrire une méthode permettant de vérifier que toutes les arêtes partagent exactement deux faces.
10. Pour ceux qui vont encore plus vite : écrire une méthode permettant de faire la subdivision de Loop. Nous ferons l'hypothèse que tous les maillages triangulaires n'ont pas de bords et qu'il n'y a donc que des points intérieurs au maillage à traiter.