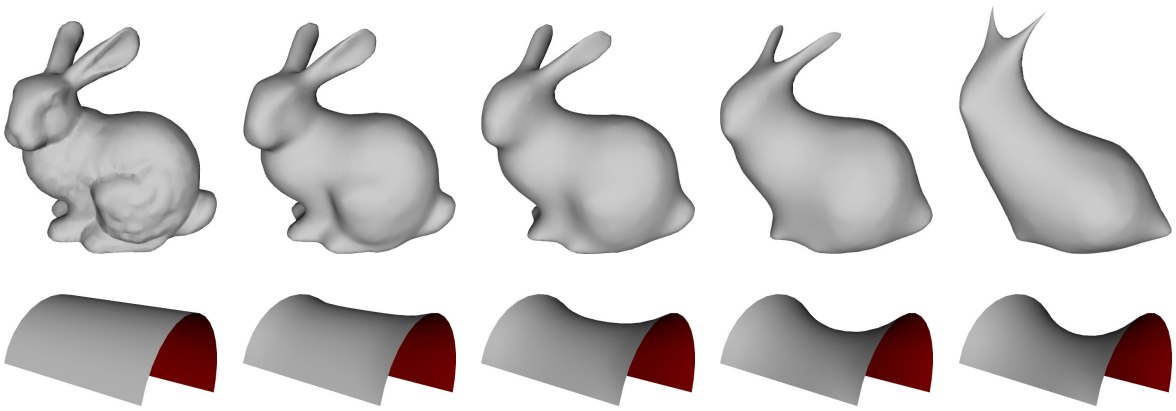
TP 2 : Lissage laplacien



1 Introduction

L'objectif du TP est de manipuler un maillage en utilisant la structure *half edge*. Pour rappel, cette structure facilite l'accès aux voisinages d'un éléments du maillage (sommet, arête, face). Cette facilité d'accès est ici mise en avant par la création d'un filtre de lissage de maillage : le filtre laplacien.

Vous pouvez effectuer ce TP seul ou en binôme. Un petit compte rendu est à rendre une semaine après la séance en salle machine.

2 Base de code

Pour la récupérer la base de code, allez à l'adresse suivante :
https://team.inria.fr/imagine/files/2015/09/TP2_code.zip.

Pour l'utiliser, vous devez tout d'abord créer un lien symbolique vers les libraries externes utilisées par la base de code. Ces librairies sont les mêmes qu'au précédent TP. Si vous n'avez pas le code du précédent TP avec vous, vous pouvez télécharger le dossier `external` ici :
https://team.inria.fr/imagine/files/2015/09/external.zip.

Pour créer le lien symbolique, utilisez les commandes suivante :

```
cd dossier/du/TP/2                    # Acces au dossier du TP
ln -rs emplacement/du/dossier/external    # Creation du lien symbolique
```

Cette manipulation permet de ne pas dupliquer les sources des librairies, et donc d'économiser votre espace personnel. De même, récupérez le dossier `models` qui contient les maillages que vous pouvez utiliser en entrée :
https://team.inria.fr/imagine/files/2015/09/models.zip, et créez un lien symbolique vers ce dossier depuis le dossier du TP.

Ensuite, créez un dossier temporaire depuis lequel vous utiliserez CMake pour générer un Makefile avant de lancez la compilation :

```
mkdir build          # Creation du dossier de build
cd build             # Acces au dossier de build
cmake ..             # Creation du Makefile
make                 # Compilation + edition de liens
./smoothing          # Execution
```

Comme le précédent, ce programme gère la création d'une fenêtre GLFW contenant un contexte OpenGL (3.1). Ce contexte sert à l'affichage d'une scène 3D simple contenant un objet à lisser, et permet de naviguer grâce aux contrôles suivants :

- ↑, ↓, ←, et →pour orienter la caméra
- `PAGE UP` et `PAGE DOWN` pour tourner la caméra
- `Z` et `S` pour zoomer/dézoomer respectivement
- `Q`, `D`, `A` et `E` pour se déplacer dans la scène
- `SHIFT` pour le mode lent

Par ailleurs, la touche `ESPACE` active le lissage du maillage (qui est inactif dans la base de code initialement : à vous de coder !).

2.1 Half edge

La classe `MeshHE` contient un maillage représenté par une structure de *half edge*, et se charge de la transmission des données d'affichage sur la carte graphique. Une instance de `MeshHE` peut ête construite à partir d'un `Mesh` standard (qu'on aura importé à partir d'un fichier OFF par exemple).

Attention cependant : le maillage d'entrée doit être une variété orientable, notion sur laquelle nous reviendrons en cours. Autrement dit (voir figure 1) :

- chaque arête doit être incidente à exactement une face (arête du bord) ou deux faces (autres arêtes)
- chaque sommet doit avoir un voisinage unique ouvert (sommet du bord) ou non (autres sommets)

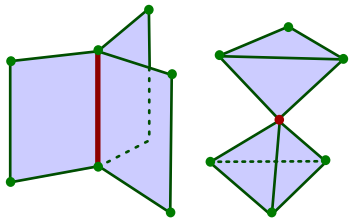


Figure 1: L'arête (à gauche) et le sommet en rouge (à droite) rendent les surfaces auxquelles ils apartiennent non variété.

Prenez connaissance du code de la classe `MeshHE` ainsi que des classes `Vertex`, `Face`, et `HalfEdge`.

3 Consignes

3.1 Parcours de voisinage

Afin de pouvoir calculer le laplacien de la distribution des positions en chaque sommet du maillage, il faut avoir accès à l'ensemble des sommets voisins pour chaque sommet. Ce voisinage est accessible simplement grâce à la structure de half edge. Utilisez cette structure pour implémenter la méthode `MeshHE::GetVertexNeighbors(Vertex* v)`.

Testez le résultat de cette méthode sur un cas simple ; par exemple dans un tetraèdre chaque sommet est le voisin de tout les autres.

Notes :

- Utilisez les attributs `m_next` et `m_twin` des half edges pour parcourir le voisinage d'un sommet.
- Ce voisinage s'appelle aussi le *1-ring* ou *star*.
- Vous pouvez poser l'hypothèse que la maillage n'a pas de bord pour faciliter le parcours.

3.2 Calcul du laplacien

Maintenant que le calcul du voisinage est fonctionnel, vous pouvez implémenter le calcul du laplacien de chaque sommet dans la méthode `MeshHE::Laplacian(Vertex* v)`. Pour rappel, le laplacien d'un sommet p_i s'écrit :

$$\Delta p_i = \frac{1}{\|N(v_i)\|} \sum_{v_j \in N(v_i)} p_j - p_i$$

où $N(v_i)$ est le voisinage du sommet v_i.

Testez le résultat de cette méthode sur un cas simple ; par exemple dans un tetraèdre régulier centré à l'origine, chaque sommet a un laplacien pointant vers l'origine (ie. le laplacien et la position sont colinéaires et opposés).

3.3 Lissage itératif

Enfin, maintenant que le calcul du laplacien est fonctionnel, vous pouvez implémenter le lissage laplacien itératif du maillage dans la méthode `MeshHE::LaplacianSmooth(float lambda, uint nb_iter)`.

Pour rappel, une étape de lissage consiste à déplacer chaque sommet dans la direction de son laplacien. Le paramètre λ contrôle l'intensité de ce déplacement. Attention, pour que le lissage se passe sans encombres, il est nécessaire de calculer le laplacien de chaque sommet avant de procéder à leurs déplacements.

Testez l'effet du lissage sur différents maillages et avec différentes valeurs de λ. Qu'observez vous ?

Notes :

- Afin de compenser la perte de volume induite par le lissage, vous pouvez utiliser la fonction `MeshHE::Normalize()`.
- Les normales du mailage ne sont plus valides après un déplacement des sommets. Vous pouvez les re-calculer avec la fonction `MeshHE::ComputeNormals()`.

3.4 *Bonus :* Préservation des bords

Dans la première partie du TP, vous n'avez considéré que des maillages sans bord. La détection du bord est très aisé avec la structure de half edge. Implémentez les méthodes suivantes :

- `MeshHE::IsAtBorder(Vertex* v)`
- `MeshHE::IsAtBorder(Face* f)`
- `MeshHE::IsAtBorder(HalfEdge* he)`.

Ensuite, effectuez le lissage laplacien d'un maillage à bord, en fixant $\lambda = 0$ pour les sommets du bord. Qu'observez vous ?

Notes :

- Le calcul de la normale d'un sommet dépend de son voisinage. Il se peut que les bords introduisent des effets étranges sur l'ombrage si l'une des fonctions ci dessus est codée de manière non cohérente.

3.5 *Bonus :* Lissage de Taubin

Vous aurez remarqué que les maillages perdent du volume lorsqu'on les lisse avec le filtre de Laplace. Un filtre très similaire permet de limiter cet effet : il s'agit du filtre de Taubin. Ce dernier prend deux paramètre : λ et μ, où $0 < \lambda < -\mu$. Une itération du lissage se divise en deux :

- une itération du filtre de Laplace avec λ pour paramètre
- une itération du filtre de Laplace avec μ pour paramètre

Implémentez ce filtre dans la méthode `MeshHE::TaubinSmooth(float lambda, float mu, uint nb_iter)`.

Testez le sur différents exemples. Qu'observez vous ?

Notes :

- L'auteur préconnise des valeurs de λ et μ proches en valeur absolues : $0,330$ et $-0,331$ par exemple.
- Pour de plus amples informations, voir *Curve and surface smoothing without shrinkage*, Gabriel Taubin, ICCV 1995.

4 Rendu

Le rendu doit contenir les images avant/après des objets que vous avez lissé, avec des explications de votre démarche.

Mettez le tout dans un pdf s'appellant `MS_TP2_NOM1_NOM2.pdf` que vous m'enverrez par mail avec l'objet `[MS][TP2] Nom1 Nom2` en mettant en copie les deux membres du binôme.

5 Contact

Lors de la séance, n'hésitez pas à poser vos questions, pour des problèmes théoriques ou de code. Vous pouvez ensuite poser vos questions par mail : `ulysse.vimont@inria.fr`. Le compte rendu est également à envoyer à cette adresse.