Meshes and Computational Geometry
Master 2 Informatique ID3D – Université Lyon 1

TP2 and 3 – Iterators on a triangulated surface data structure, Differential properties

In this project, the triangular mesh data structure will be used to store triangulations of the surface of an object (vertices corresponding to 3D points) as well as triangulations of points in 2D. The aim of this session is to add a set of iterators to navigate on a triangulation.

Before proceeding, add to your work the routine of loading a mesh stored in an OFF in the triangulated mesh data structure by implementing the face adjacency identification algorithm.

Iterators

Complete your `mesh` module with a set of iterator classes to visit the faces and the vertices of a `Mesh`:

- `Iterator_on_faces`: Iterator that can be used to visit the faces of a `Mesh` triangulation, in the same way as STL iterators can be used to visit the elements of an STL container. The order in which the faces are visited is not important. Do not forget to provide the additional member functions `faces_begin()` and `faces_past_the_end()` in the `Mesh` data structure. These should return some `Iterator_on_faces` values.

  In your implementation, the `Iterator_on_faces` will contain an access to the associated `Mesh` triangulation (and therefore to its internal face container), as well as the current position that can be materialized by an index. The class will also have an overload of the `++ operator` to shift the current position to the next one, and an overload of the `* operator` returning a reference on the current face.

- `Iterator_on_vertices`: Iterator that can be used to visit the vertices of a `Mesh` triangulation. As previously, your `Mesh` data structure should provide the member functions `vertices_begin()` and `vertices_past_the_end()` returning some values of `Iterator_on_vertices`.

- `Circulator_on_faces`: Special topological iterator used to iterate counter-clockwise on the set of incident faces at a given vertex. In the `Mesh` data structure, you should therefore provide a member function `incident_faces(Sommet & v)` returning a `Circulator_on_faces` positioned on one of the incident faces at vertex `v`. In your implementation, the iterator will contain an access to the associated triangulation, an access to the vertex around which it turns, as well as an access to the current face which can be materialized by an index. The class will also have a `++ operator` overload in order to switch from the current face to the next one, and a `* operator` overload returning a reference on the current face.

- `Circulator_on_vertices`: Iterator used to iterate, counter-clockwise, on all the vertices that are adjacent to a vertex. The class `Mesh` should contain some functions `adjacent_vertices(Sommet & v)` returning a `Circulator_on_vertices` positioned on one of the vertices adjacent to vertex `v`.

- Test all your iterator classes in a small test program:

```
Mesh titi;
titi.loadOFF(''cube.off'');
Iterator_on_vertices its;
Circulator_on_faces cf ;
for (its=titi.vertices_begin();
     titi !=titi.vertices_past_the_end();
     ++its)
{
  Circulator_on_faces cfbegin
         =titi.incident_faces(*its) ;
  int cmpt=0 ;
  for (cf=cfbegin,++cf;
       cf!=cfbegin;
       cf++)
     cmpt++ ;
  std ::cout<< ''valence of the vertex ''
            << cmpt <<std ::endl ;
}
```

Normal and average curvature estimation at a vertex

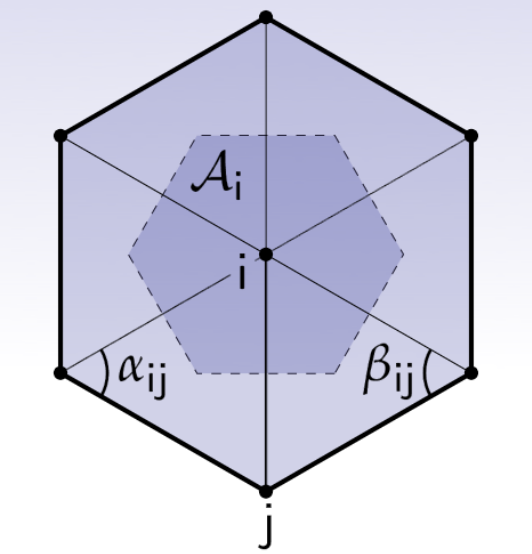
- Etant donnée une surface plongée dans $\mathbb{R}^3$ dont on ne connait qu'une approximation sous la forme d'un maillage triangulé, nous avons vu en cours comment obtenir une estimation du Laplacien d'une fonction scalaire u connue en chacun des sommets du maillage (u peut par exemple être donnée sous la forme d'un tableau où les valeurs de u sont indexées dans le même ordre que les sommets auxquels ils correspondent). On utilise pour cela la formule dite de la cotangente.

$$(Lu)_i = \frac{1}{2A_i} \sum_j (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i)$$

- Ce Laplacien est utile pour résoudre les équations de diffusion de la chaleur sur une surface en reflétant certaines caractéristiques géométriques intrinsèques de cette surface, indépendantes de sa paramétrisation.

- Estimation de la normale et de la courbure moyenne

- On calcule le Laplacien de la fonction u correspondant à la coordonnée x (resp. y puis z) de chaque sommet **s**.
- Le Laplacien $\Delta\mathbf{s}$ est le vecteur (Δx, Δy, Δz).
- $\Delta\mathbf{s}$ correspond en tout point à $2H\mathbf{n}$, où H est la courbure moyenne à la surface en **s** et **n** son vecteur normal.



- Utilisez vos `itérateur`s de sommets et vos `circulateur`s de faces et de sommets pour évaluer la normale et la courbure moyenne à chaque sommet en utilisant l'approximation ci-dessus.
  - Concernant A_i, vous pourrez dans un premier temps prendre le tiers de l'aire des triangles adjacents au sommet s_i.
  - Vous pourrez si vous le préférez améliorer l'approximation en considérant le barycentre des faces pour les découper en éléments d'aire plus pertinents (comme illustré ci-dessus).
- Réalisez un affichage du maillage avec une coloration dépendante de la courbure (vous pouvez par exemple exprimer vos couleurs dans le repère HSV, https://fr.wikipedia.org/wiki/Teinte_Saturation_Valeur).