Modélisation 3D à Partir de Fichiers Lidar

Antoine Blancy

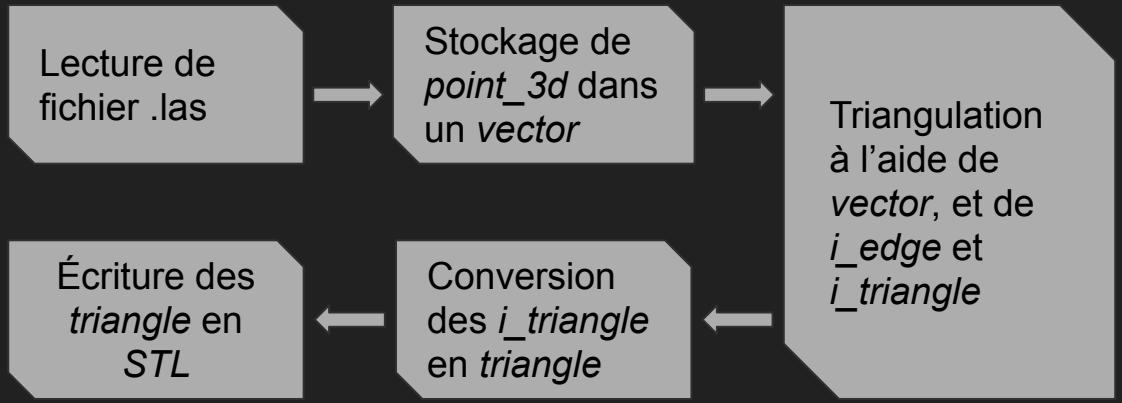
Le contexte

Les notions de base

- Lidar → Fichiers .las
- Stéréolithographie→FIchiers .stl
- En passant par des vecteurs

Les structures et types différents

- *error_code*
- *point_2d, point_3d*
- *triangles*
- *i_triangle, i_edge*
- *vector (plusieurs types)*
  - *point_2d, point_3d, i_edge, i_triangle*
- *header*
- *point_record_1*
- *stl*



Les types géométriques

Pour la gestion des triangles et des points

- *point_2d, point_3d*
  - *{double x, y, (z)}*
- *Triangles*
  - Trois *point_3d*

Les types *header* et *point_record_1*

Les fichiers Lidar

- *header*: L'en-tête du fichier .las
  - Contient des informations sur le fichier, notamment, le nombre de points
- *point_record_1*: Les points eux même

Les types *i_triangle* et *i_edge*

Pour le traitement de triangles et d'arêtes

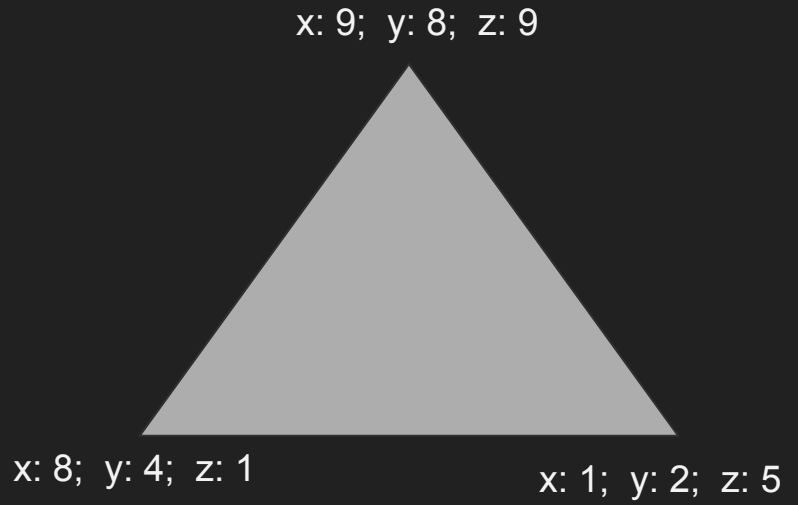
- *i_triangle*: Les indices qui “pointent” vers les sommets d’un triangle
  - *{int p1, p2, p3}*

- *i_edge*: Les indices qui “pointent” vers les extrémités d’une arête
  - *{int p1, p2}*

Le principe des *i_edge* et *i_triangle*

p1: 7; p2: 2; p3: 4
p1: 1; p2: 6; p3: 3
...
...
...
...
...
...

0	x: 3; y: 8; z: 9
1	x: 2; y: 9; z: 6
2	x: 1; y: 2; z: 5
3	x: 6; y: 4; z: 7
4	x: 8; y: 4; z: 1
5	x: 6; y: 1; z: 6
6	x: 3; y: 7; z: 6
7	x: 9; y: 8; z: 9



Le type *vector*

- On a plusieurs types de *vector* différents
  - *vec_i_triangle, vec_i_edge, vec_point_2d, vec_point_3d*

- *Le but est de stocker dans une sorte de liste et n’utiliser que des fonction pour récupérer le contenu*

La fonction vector_get()

```
error_code vector_get(vec_point_2d *v, int index, type *element)
{
    //Gestion d'erreur

    *element = v->content[index];
    return ok;
}
```

Fonction pour l’affichage d’erreurs

```
void error_message_displayer(error_code e, char* taskname);
```

- La fonction imprime simplement le code d’erreur du premier argument, suivi du *taskname*

- Il suffit d’appeler cette fonction, et bien documenter la tâche dans le taskname pour savoir où exactement est le problème

Exemple d’utilisation de la fonction

```
error_message_displayer(vec_i_triangle_get(&triangles, i,
&tmp), "Getting tmp_triangle from triangles in main");
```

Étape finale

Passer de points à triangles

- En appliquant l’algorithme de Bowyer-Watson, on a la triangulation de Delaunay
  - Une triangulation qui permet d’ éviter les triangles à angle très petits

Conclusion

- Un projet qui montre une implémentation pratique des listes/vecteurs

- Un projet avec beaucoup de librairies et beaucoup de pièces interdépendantes