

Rapport CGDI

Corto Cristofoli et Max Royer

3 avril 2025

1 Introduction

Cet article est réalisé dans le cadre de notre cours de CGDI. Il s'agit de la présentation de notre implémentation de l'article de recherche *Curved PN triangle*, par Alex Vlachos, Jorg Peters, Chas Boyd et Jason L. Mitchell.

L'article présente un algorithme de lissage de modèles 3D de basse résolution. Pour cela, l'idée est d'utiliser la méthode des « Curved PN triangles ». Un modèle 3D de basse résolution n'a que peu de polygones et donc une allure nécessairement anguleuse. Pour le rendre plus organique un lissage est nécessaire : il faut rajouter un grand nombre de triangles afin de faire disparaître ces angles. Ce rajout se fait algorithmiquement en suivant des surfaces de bézier triangulaires.

Si l'article décrit l'algorithme de manière théorique, nous nous sommes penchés sur sa compréhension et son implémentation en C++ à l'aide de la bibliothèque **geometry-central**.

2 Principe de l'algorithme

Dans tout le rapport nous considérons qu'un modèle 3D est un ensemble de face triangulaires.

3 Enjeux d'implémentation de *Geometry Central*

4 Indexation des sommets du modèle 3D

On numérote la discrétisation de la face de cette manière pour avoir facilement les nouvelles faces. En effet on remarque que pour un sommet i dans une couche c , on a que ses deux voisins dans la couche $c + 1$ sont les sommets $i + c + 1$ et $i + c + 2$. Cela permet de déduire rapidement les nouvelles faces à ajouter en disant que pour une discrétisation avec un `lod` l , on ajoute pour chaque sommets i dans une couche c avec $c \leq l$ la face $(i, i + c + 1, i + c + 2)$ et la face $(i, i + c + 2, i + 1)$ si $i + 1$ est aussi dans la couche c (*i.e.* $i + 1 < \frac{(c+2)(c+1)}{2}$)

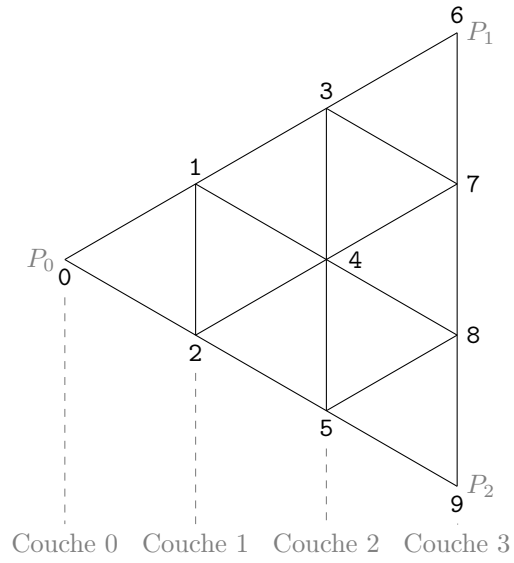


FIGURE 1 – Indexation de la triangulation d’une face pour un lod de 2

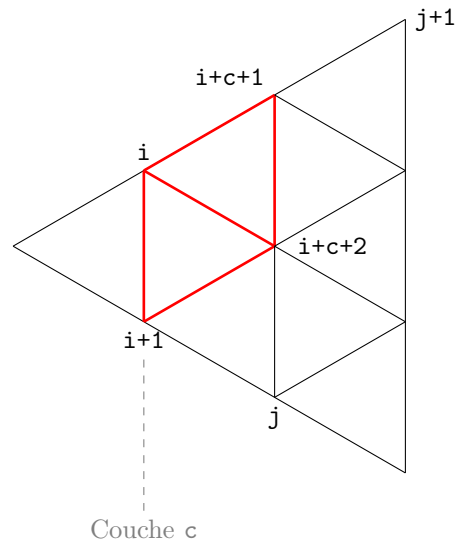


FIGURE 2 – Exemple de faces ajoutées pour un sommet i dans la couche c ainsi que le cas où j et $j+1$ ne sont pas dans la même couche

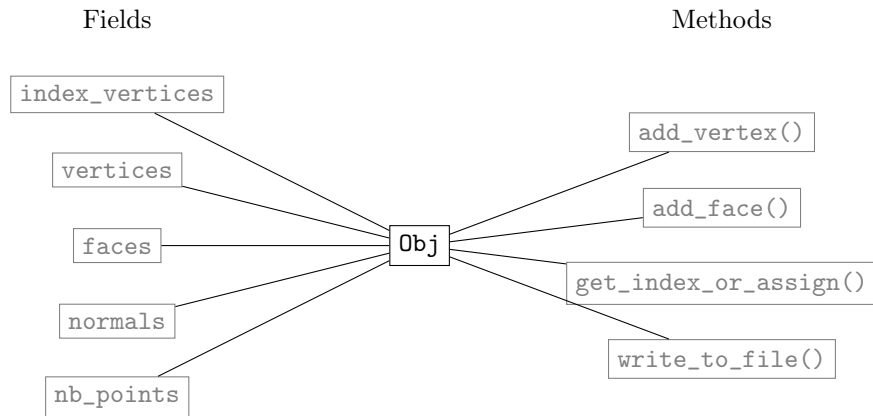


FIGURE 3 – Description de la classe `Obj`

Chaque sommets est alors identifié par ses coordonnées barycentriques en fonction de (P_0, P_1, P_2) , étant donné que ces coordonnées sont rationnelles et sont en fait des multiples de $\frac{1}{10d+1}$ nous avons décidé de garder des coordonnées entières pour identifier les sommets tout en gardant le `lod` lorsqu'il s'agit de calculer la position du sommet dans la tuile de Bézier.

Ainsi un sommet est représenté par un `vector` de `pair(int,int)` de taille 3, tel que pour chaque `pair`, le premier élément est l'indice du point P_i et le second est la coordonnée barycentrique associée au point P_i .

Pour recréer un nouvel objet nous avons créé une petite classe `Obj` ayant les primitives pour construire un tel objet.

Ici on a `Obj.index_vertices : map<vector<pair<int,int>,int>`