Chef d'Oeuvre

Traitement de la géométrie : décomposition de maillages en variétés topologiques

Spécifications

Laura BARROSO
Martin BOUYRIE
Sébastien EGNER

Encadrant : Nicolas MELLADO
Clients : Nicolas MELLADO, Loïc BARTHES

Université Toulouse III - Paul Sabatier
5 janvier 2021

1 Description

Le but du projet est de créer une application qui permet de modifier n'importe quelle surface **non manifold** en une surface **manifold** tout en garantissant une optimisation adéquate adaptée à la surface **non manifold** d'entrée. La modification apportée est une modification topologique de la surface et ne modifie aucunement la géométrie de la surface d'entrée apportée. C'est une correction topologique. Les algorithmes de modification proviennent de l'article GUEZIEC et al. [2001].

2 Cahier des charges

2.1 Définition du besoin

Le besoin auquel notre application va répondre comme décris précédemment est celui de convertir une surface **non manifold**, fournie par l'utilisateur, en une surface **manifold**, par implémentation des différents algorithmes présentés dans l'article GUEZIEC et al. [2001].

2.2 Exigences fonctionnelles

Notre application devra offrir l'ensemble des fonctionnalités suivantes :

- L'importation des surfaces d'entrée sous différents formats (PLY, OBJ, OFF, etc). Le moteur Radium Engine permet déjà de charger plusieurs formats de fichiers, ce qui devrait nous faciliter la tâche.

- L'exportation des surfaces résultantes, là encore, sous différents formats (OBJ, OFF, PLY, etc).

- Une interface utilisateur qui permettra à celui-ci de spécifier les différents paramètres pour l'application des algorithmes de l'article GUEZIEC et al. [2001] (la surface d'entrée, les options d'exportation, les approches à utiliser, etc). Dans le cas où l'utilisateur ne spécifie pas les approches à utiliser pour le *cutting* ou le *stitching*, notre application doit être en mesure de détecter la meilleure approche à utiliser pour ces deux étapes, selon la surface d'entrée fournie par l'utilisateur. Il n'existe pas à notre connaissance aujourd'hui, un seuil de critère qui nous permettrait de savoir quelle approche est optimale l'une par rapport à l'autre, il est donc envisagé de réaliser des tests sur les différentes approches (une fois qu'elles seront implémentées), afin de pouvoir déterminer à partir d'une surface quelle approche est la plus adaptée pour le *cutting* et le *stitching*.

- La visualisation des surfaces résultantes. Cela permettra à l'utilisateur de visualiser la surface de sortie, avant de l'exporter.

— Une interface Web qui permettrait d'uploader, de convertir et de sauvegarder ses surfaces. Cela pourrait par la suite être intégré directement sur un site et permettre à des internautes d'utiliser notre application directement depuis leur navigateur.

2.3 Priorité des exigences fonctionnelles

Fonction	**Description**	**Priorité**
FP1	Importer des surfaces d'entrée	Forte
FP2	Spécifier les différents paramètres via une interface	Moyenne
FP3	Exporter les surfaces résultantes	Forte
FO1	Visualiser les surfaces résultantes	Faible
FO2	Utiliser une interface Web	Faible

Avec **FP** correspondant à **Fonction Principale** et **FO** correspondant à **Fonction Optionnelle**.

2.4 Validation des fonctionnalités

Fonction	**Objectif à remplir pour la validation**
FP1	L'utilisateur pourra charger ses surfaces à partir des formats de fichiers les plus répandus, tels que PLY, OBJ, ou encore OFF.
FP2	L'utilisateur pourra spécifier les approches à utiliser pour la phase de *cutting* et de *stitching*, l'emplacement et le format du fichier à importer ainsi que celui à exporter.
FP3	L'utilisateur pourra sauvegarder la surface résultante sous les formats de fichiers les plus répandus, tels que PLY, OBJ, ou encore OFF.
FO1	L'utilisateur pourra visualiser la surface résultante dans un petit environnement 3D, sous différents angles et selon différents éclairages.
FO2	L'utilisateur sera en mesure de convertir ses surfaces directement depuis une interface Web, interface qui pourra aisément être intégrée par la suite pour un site Internet.

2.5 Contraintes

L'implémentation de notre application est soumise à plusieurs contraintes :
— Utilisation de Radium Engine
— Code en C++
— Application préférablement cross-platform (Linux, Windows, macOS)
— Utilisation de Gitlab comme gestionnaire de versions

2.6 Délai de rendu

L'application, le descriptif du code ainsi que le manuel d'utilisation sont à rendre pour le 25 février 2021. Le site web accompagnant ce chef d'oeuvre est également à rendre à cette date.

3 Vue générale du système

Il n'y a qu'un seul cas d'utilisation du système : l'utilisateur veut réparer une surface. Pour cela, il doit pouvoir importer son maillage et ensuite choisir les différentes approches pour le cutting et le stitching. Une fois le résultat obtenu, le maillage **manifold** est enregistré.

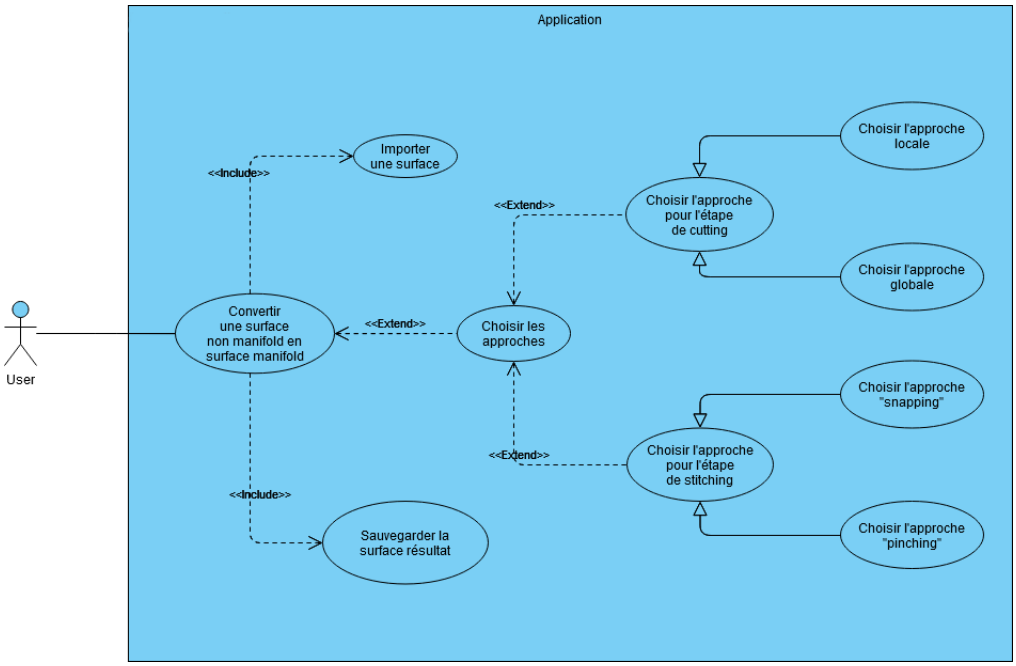

Figure 1 – Diagramme des cas d'utilisations

3.1 Module *cutting*

Ce module représente la première phase de l'algorithme de réparation de maillages et contient les approches locale et globale. Il permet de découper la surface en plusieurs composantes **manifold**. L'utilisateur a la possibilité de choisir l'approche utiliser.

3.2 Module *stitching*

Ce module représente la seconde phase de la réparation de maillages et contient les approches *pinching* et *snapping*. Il permet de reconnecter, quand c'est possible, les différentes composantes **manifold** obtenues pendant le *cutting*. L'utilisateur a la possibilité de choisir l'approche utiliser.

3.3 Module IHM

Ce module permet à l'utilisateur d'intéragir avec l'application. Dans un premier temps, nous prévoyons une interface en ligne de commande et si le temps le permet, nous implémenterons une interface graphique avec la possiblilité de visualiser les modifications effectuées sur la surface.

3.4 Module sérialisation

Ce module permet d'enregistrer les données sur disque. C'est une partie déjà réalisé par Radium pour les formats OBJ et OFF mais pas pour le format PLY. Le format de fichier PLY permet d'ajouter des attributs aux éléments (sommets, faces...) d'un maillage, ce qui nous sera très utile pour tester notre application.

4 Planning et Risques

4.1 Choix

Pour réduire les risques éventuels de temps et de fonctionnalités, nous avons décidé de nous fixer un premier objectif : réaliser une approche du *cutting* (Global) et une approche du *stitching* (Pinching), afin d'avoir un résultat concret et fonctionnel à la fin de ce chef d'oeuvre même si celui ci n'en contiendrait pas éventuellement toutes les méthodes décrites.

4.2 Dépendances

Afin de pouvoir commencer à utiliser la méthode de *cutting*, deux étapes sont nécessaires à son bon déroulement : la suppression des *degenerate faces* pour corriger et vérifier l'état d'entrée de la surface donnée en entrée, ainsi que le marquage des **singularités topologiques** par identification des **arêtes singulières** et **sommets singuliers**. De même, le *stitching* bien qu'indépendant au *cutting* dans son déroulement prend en entrée le résultat du *cutting*, celui-ci est donc indispensable pour tester et confirmer le *stitching* réalisé.

4.3 Matrice des risques

Risque	Probabilité	Impact	Prévention	Solution
Abandon d'un membre du projet	10%	Important	Suivi écrit de nos modifications et implémentations	Reprise des modifications et implémentations par un ou plusieurs membres de l'équipe
Fonctions de prétraitement non fonctionnelles	10%	Important	S'attarder sur les tests de celles-ci	Reprise des fonctions
Prise en main difficile de Radium et OpenMesh	50%	Important	Lire la documentation et faire des tests	Lire la documentation et faire des tests, Demande d'aide à l'encadrant
Retard sur le planning et circonstances inattendues	80%	Moyen	Maintenir le planning, avance continue du projet et bonne répartition des tâches	Organisation prévoyante et réaction rapide, Abandon des fonctionnalités optionnelles
Perte de données	30%	Moyen	Sauvegarder régulièrement les versions, garder des traces	Recommencer le bout de code manquant
Mauvaise répartition des tâches et manque de communication	25%	Moyen	Réunion de groupe récurrentes	Mise en commun et adaptabilité des membres à aider un autre membre sur une tâche

4.4 Planning

Ce projet est réalisé dans le suivi d'un processus itératif.

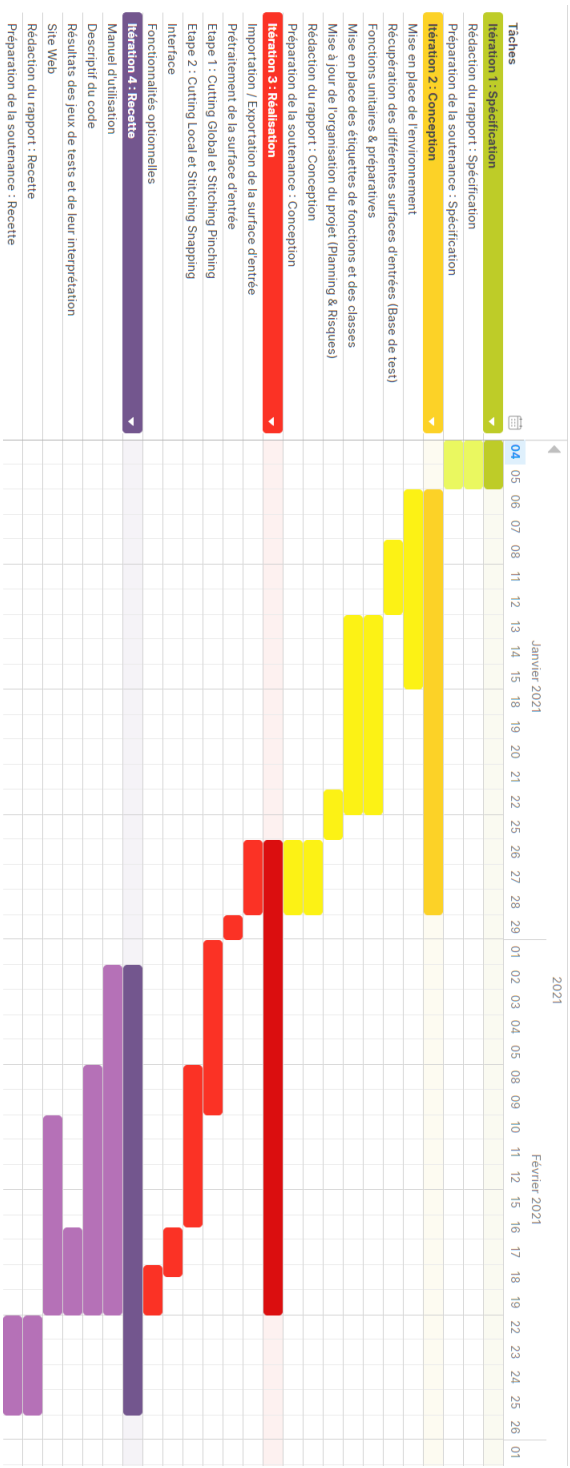

Figure 2 – Diagramme de Gantt

5 Évolutions envisagées

Plusieurs évolutions peuvent être envisagées pour notre chef d'oeuvre :

— L'intégration du système de converion d'une surface **non manifold** en surface **manifold**. En effet, pour l'instant, Radium n'est pas en mesure de charger des modèles **non manifold** (du moins, il n'ajoute pas les éléments qui rendraient ces derniers **non manifold**). On peut imaginer que dans le cas où le modèle que l'on souhaite charger est **non manifold**, Radium puisse utiliser le code développé au cours de ce chef d'oeuvre pour le rendre **manifold** et ainsi le charger correctement

— La réalisation d'un site Web qui utiliserait notre application, afin de donner à la possibilité aux internautes de convertir leurs maillages **non manifold** directement depuis leur navigateur.

Glossaire

arête singulière Une arête est dite *singulière* si au moins trois faces lui sont incidentes. Autrement, on parle d'arête *régulière*. 4, 8

manifold Une surface polygonale est dite *manifold* si tous ses sommets sont réguliers. Dans l'autre cas, on parle d'une surface *non manifold*. 1, 3, 4, 7

non manifold Une surface polygonale dite *non manifold* si au moins un de ses sommets est singulier. Dans l'autre cas, on parle d'une surface *manifold*. 1, 7

singularité topologique Une singularité topologique correspond soit à un **sommet singulier**, soit à une **arête singulière**. 4

sommet singulier Un sommet v est dit *singulier* si $l(v)$ n'est ni *chaîne*, ni un *cycle*. Autrement, on parle de sommet *régulier*. 4, 8

Références

André GUEZIEC, Gabriel TAUBIN, Francis LAZARUS, and Bill HORN. Cutting and stitching : Converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 7(2), 2001.