RAPPORT DE PROJET

Proposé à :

Sébastien Morgenthaler

Par :

Ashkan MOTAMEDIFAR

Mehdi JAFARI ZADEH

Hiver 2023

TABLE DES MATIERES

Introduction 4
1. Transformation d'un maillage en graphe dual : 5
1.1. Calcul des Centroids dans le Graphe Dual 5
1.2. Formation des Arêtes dans le Graphe Dual 5
2. Tri des arêtes du maillage pour trouver les facettes adjacentes 6
2.1. Tri des arêtes à l'aide du tri par sélection 6
2.2. Tri des arêtes à l'aide du tri par tas (Heap Sort) 7
2.3. Tri des arêtes à l'aide d'un arbre AVL 8
2.4. Tri des arêtes à l'aide d'une table de hachage 10
3. Coloration du graphe dual 12

Introduction

Dans le domaine de la modélisation géométrique et de la visualisation de données, la transformation d'un maillage en un graphe dual représente une avancée technique significative. Cette étude présente une méthodologie innovante pour transformer un maillage en un graphe dual. En calculant le centroïde de chaque facette, l'étude propose une méthode systématique pour identifier ces sommets cruciaux. Ce processus est essentiel pour la construction d'un graphe dual précis et fonctionnel, permettant une meilleure compréhension et manipulation des structures de données complexes.

Le document progresse ensuite en détaillant la méthodologie de connexion de ces centroïdes. En reliant les centroïdes des facettes voisines, il crée les arêtes du graphe dual, une étape fondamentale pour visualiser les relations entre différentes parties du maillage. L'étude évalue et compare divers algorithmes pour le tri des arêtes, tels que le tri par sélection, le tri par tas, l'utilisation de l'arbre AVL, et le hachage, soulignant leurs efficacités respectives dans l'identification des facettes adjacentes. Cette analyse approfondie offre un aperçu précieux des techniques optimales pour structurer et analyser les données de maillage.

Enfin, le document se conclut sur l'application de l'algorithme de Dijkstra pour la coloration du graphe dual. Cette étape, cruciale pour la distinction visuelle des différentes régions du graphe, permet d'évaluer la distance la plus courte à partir d'un nœud de départ et de colorer les autres nœuds en fonction de leur distance par rapport à ce point. Cette méthode offre non seulement une clarté visuelle accrue mais aussi une compréhension approfondie des relations spatiales au sein du graphe. En somme, cette recherche apporte une contribution notable au domaine de la modélisation géométrique, offrant des méthodes innovantes et efficaces pour la transformation de maillages en graphes duaux.

1. Transformation d'un maillage en graphe dual :

Pour transformer un maillage (mesh) donné en graphe dual (dual graph), deux étapes principales sont effectuées.

1.1. Calcul des Centroids dans le Graphe Dual

La première étape consiste à trouver les sommets du graphe dual. Pour ce faire, il faut d'abord extraire les facettes du maillage initial et trouver le point central de chaque facette. Pour trouver le point central (centroid), la moyenne des valeurs X, Y, Z, de chaque sommet d'une facette doit être calculée.

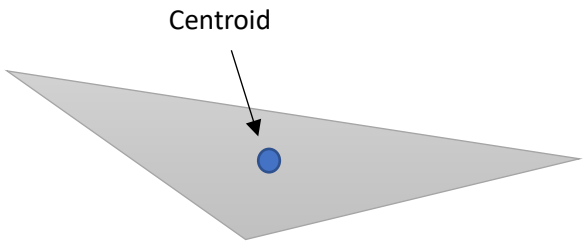


Figure 1. Méthode de création d'un point central

1.2. Formation des Arêtes dans le Graphe Dual

La deuxième étape est de connecter les points centraux des facettes du maillage dans le graphe dual, ou plutôt de trouver les arêtes du graphe dual. Pour ce faire, parmi les points centraux trouvés, il est nécessaire de localiser les points sur les facettes voisines et de les connecter ensemble.

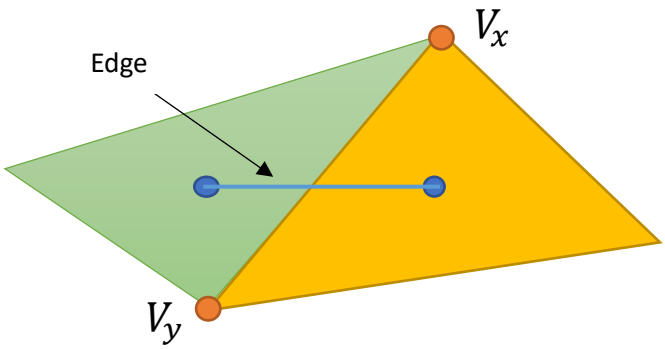


Figure 2. Création d'une arête entre les points centraux des facettes du maillage pour créer un graphe dual

2. Tri des arêtes du maillage pour trouver les facettes adjacentes

Lorsque deux facettes sur un maillage sont voisines, elles partagent deux sommets parmi les trois disponibles, créant ainsi une arête commune entre ces deux sommets. Par conséquent, en trouvant des arêtes communes entre les facettes qui ont deux sommets identiques et le même numéro de facette, on peut identifier les facettes voisines et connecter leurs points centraux. Dans l'implémentation réalisée, un tableau nommé "Facets" est utilisé, qui contient un nombre de lignes égal au nombre de facettes du maillage et trois colonnes pour stocker les trois sommets de chaque facette. De plus, un autre tableau à trois colonnes nommé "MeshEdges" est utilisé pour stocker les arêtes du maillage, où chaque arête est enregistrée sous forme de triplet composé du premier sommet, du second sommet et du numéro de la facette. Dans la première étape de l'algorithme mis en œuvre, les facettes et les arêtes du maillage présentes dans le fichier d'entrée sont lues par la fonction "Read" et placées dans les tableaux "Facets" et "MeshEdges". Pour chacune des facettes du maillage, six arêtes au lieu de trois sont stockées dans le tableau "MeshEdges", car les arêtes du maillage sont non orientées, nécessitant ainsi de stocker les deux directions de chaque arête dans le tableau pour les opérations ultérieures. L'objectif principal du tri des arêtes dans l'implémentation réalisée est de trouver des facettes adjacentes ayant des arêtes communes, afin de tracer une arête dans le graphe dual entre deux facettes adjacentes.

2.1. Tri des arêtes à l'aide du tri par sélection

Dans le tri par sélection, qui est l'un des types d'algorithme de tri basé sur la comparaison, on commence par trouver le plus petit élément d'un tableau donné et on l'échange avec le premier élément du tableau. Ensuite, on trouve le deuxième plus petit élément du tableau et on l'échange avec le deuxième élément du tableau. Cette méthode est répétée pour les $n-1$ premiers nombres du tableau. En fait, à chaque étape, le tableau est divisé en deux parties : la première partie du tableau est déjà triée, tandis que les autres éléments n'ont pas encore été triés.

Dans l'implémentation réalisée, nous utilisons également la comparaison de deux arêtes pour trouver la plus petite arête. Une arête est plus petite que l'arête suivante si elle a un numéro de sommet initial plus petit que celui de l'arête suivante, ou, si les numéros de sommet initial de deux arêtes sont égaux, si elle a un numéro de deuxième sommet plus petit que l'arête suivante.

L'implémentation de l'algorithme de tri par sélection est simple, mais dans le cas moyen et le pire des cas, elle a une complexité temporelle de l'ordre de $O(n^2)$, ce qui n'est pas idéal par rapport aux autres algorithmes de tri.

Pour trier les arêtes dans la transformation d'un maillage en graphe dual en utilisant le tri par sélection, le temps consommé pour des graphes avec différents nombres d'arêtes est illustré dans la Figure 3. Le temps consommé est uniquement lié aux étapes de tri des arêtes et à la transformation du maillage en graphe, et ne prend pas en compte le temps lié à la lecture du fichier d'entrée et à l'écriture du fichier de sortie.

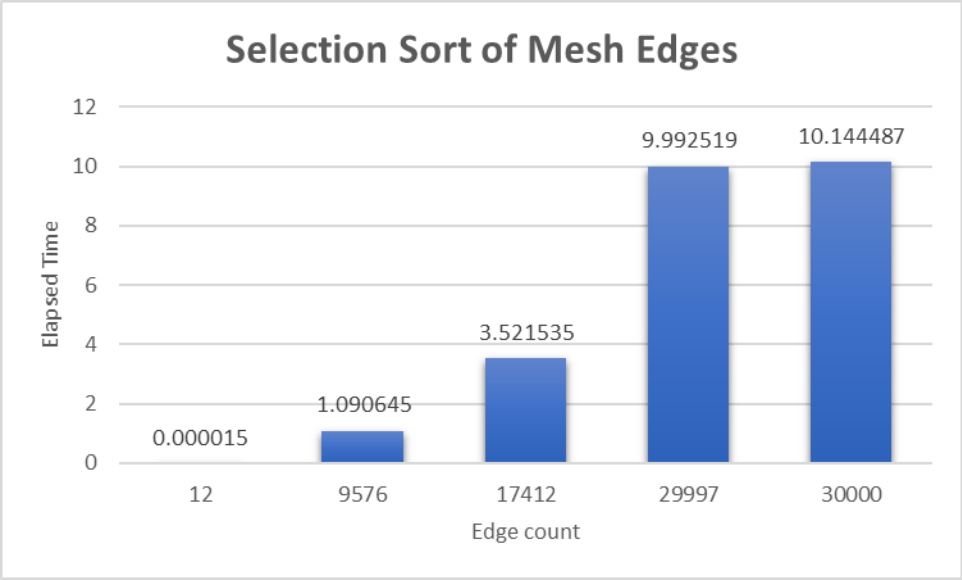


Figure 3. Résultats de l'utilisation du tri par sélection pour des maillages avec différents nombres d'arêtes

2.2. Tri des arêtes à l'aide du tri par tas (Heap Sort)

L'algorithme de tri par tas est similaire au tri par sélection en ce sens qu'à chaque étape, le plus grand élément du tableau est sélectionné et placé à sa position définitive. Dans cette méthode de tri, un arbre de tas maximum ou minimum est d'abord construit à partir de l'ensemble du tableau donné. Ensuite, le plus grand élément de cet arbre est retiré et placé à la fin du tableau trié. Après avoir retiré et traité le plus grand élément, un autre arbre de tas maximum est construit à partir des nombres restants pour trouver le deuxième plus grand nombre. Ce nombre a placé une position avant la fin du tableau. Cette opération est répétée jusqu'à ce qu'il ne reste plus aucun nombre dans le tas et que le tableau soit entièrement trié.

Pour trier n éléments à l'aide d'un arbre de tas, n opérations de suppression du nœud racine de l'arbre de tas sont nécessaires. Le coût de l'opération de suppression du nœud racine du tas est de l'ordre de $O(\log n)$. Le temps nécessaire pour construire l'arbre de tas doit également être pris en compte, car dans un cas normal, les éléments du tableau donné ne sont pas triés. Par conséquent, la complexité temporelle du tri par tas dans le pire des cas est de l'ordre de $O(n \log n)$.

Dans l'implémentation réalisée pour le tri des arêtes du maillage à l'aide du tas, deux fonctions ont été utilisées : HeapCreate pour construire le tas et HeapSortEdges pour placer les éléments dans le tas et récupérer les arêtes triées. La Figure 4 montre le temps nécessaire pour trier les arêtes pour des maillages avec différents nombres d'arêtes.

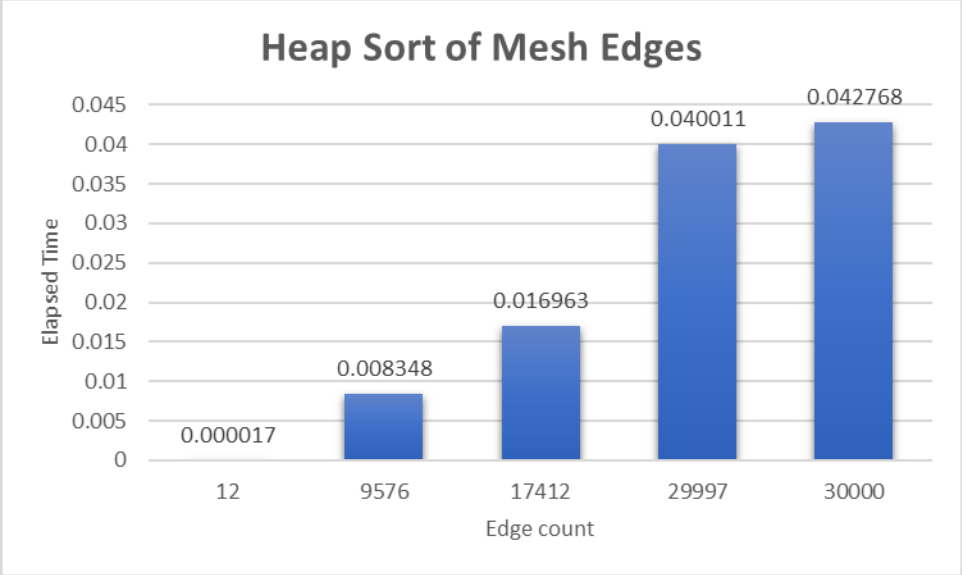


Figure 4. Résultats de l'utilisation du tri par tas pour des maillages avec différents nombres d'arêtes

Comme le montre la Figure 4, le temps consommé par l'algorithme de tri par tas est beaucoup moins élevé que celui du tri par sélection. Par exemple, pour un maillage ayant environ 30 000 arêtes, l'algorithme de tri par tas effectue l'opération de transformation en environ 0.04 seconde, alors qu'avec l'algorithme de tri par sélection pour la même transformation, plus de 10 secondes sont nécessaires.

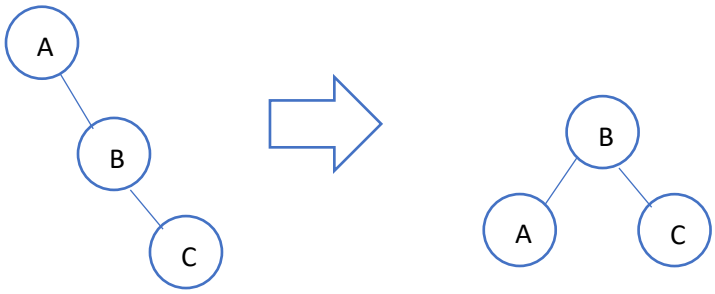
2.3. Tri des arêtes à l'aide d'un arbre AVL

Un arbre AVL est un type d'arbre de recherche binaire (BST) qui maintient l'équilibre des hauteurs de ses branches. Dans un arbre AVL, les sous-arbres gauche et droit de chaque

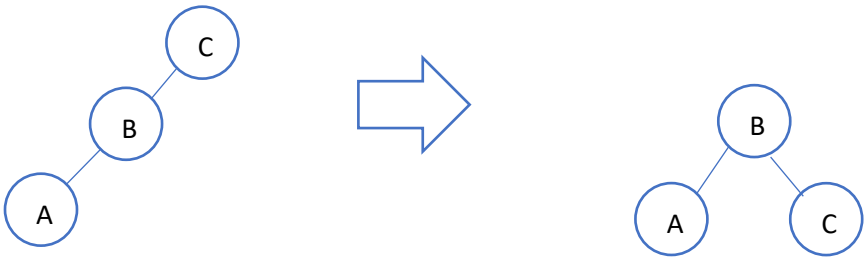
nœud sont régulièrement vérifiés pour s'assurer que la différence de hauteur ne dépasse pas une unité. Le facteur d'équilibre (Balance Factor) est utilisé pour évaluer cette hauteur.

Balance Factor = height of left subtree - height of right subtree

Lorsque l'équilibre d'un arbre AVL est perturbé, des rotations sont effectuées sur les nœuds pour rééquilibrer l'arbre. Il existe deux types de rotations définies pour l'arbre AVL : la rotation simple et la rotation double. La rotation simple elle-même est divisée en deux types : rotation vers la gauche et rotation vers la droite. Si un arbre devient déséquilibré, par exemple, lorsqu'un nœud est inséré du côté droit du sous-arbre droit, une simple rotation vers la gauche, similaire à celle illustrée dans la Figure 5, sera nécessaire. Dans cette figure, le nœud A est déséquilibré et est équilibré en effectuant une rotation vers la gauche sous le sous-arbre gauche du nœud B.



L'autre type de rotation simple est la rotation vers la droite, qui est nécessaire lorsqu'un nœud est ajouté du côté gauche du sous-arbre gauche. La Figure 6 illustre cette rotation effectuée sur le nœud B.



Les rotations doubles sont une combinaison de rotations droite et gauche, et vice versa, qui contribuent à l'équilibrage de l'arbre AVL. La complexité temporelle des opérations d'insertion, de suppression et de recherche dans un arbre AVL est de l'ordre de $O(\log n)$. Dans l'implémentation réalisée pour la transformation d'un maillage en graphe dual, l'arbre AVL a été utilisé comme troisième stratégie pour trier les arêtes du maillage, et les résultats temporels de la transformation à l'aide de l'arbre AVL sont présentés dans la Figure 5.

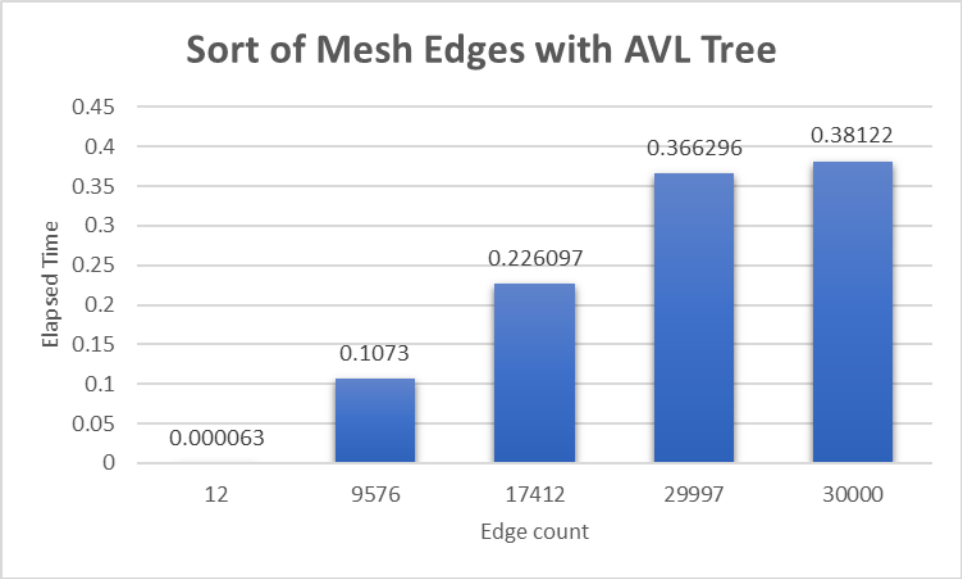


Figure 5. Résultats de l'utilisation d'un arbre AVL pour le tri des arêtes sur des maillages avec différents nombres d'arêtes

Les résultats présentés dans la Figure 5 indiquent que l'utilisation d'un arbre AVL nécessite beaucoup moins de temps que le tri par sélection et peut effectuer l'opération de transformation en environ 0.37 seconde pour des maillages d'environ 30 000 arêtes. D'autre part, le temps nécessaire à la construction et à la traversée de l'arbre AVL est plus élevé comparé au tri par tas.

2.4. Tri des arêtes à l'aide d'une table de hachage

Une table de hachage est une structure de données utilisée pour stocker des paires clé-valeur. Ces tables offrent une grande vitesse et efficacité et sont implémentées dans la plupart des langages de programmation. Le but de créer des tables de hachage est de stocker des données avec des outils allant au-delà des indices d'un tableau, ce qui permet de créer des codes flexibles pour la gestion des données. Pour implémenter une table de hachage, une fonction de hachage est utilisée, qui agit comme un traducteur. La

fonction principale de la fonction de hachage, également appelée fonction de hashage, est de convertir les mots en indices de tableau pour simplifier l'utilisation des tableaux par les humains. Une fonction de hachage doit avoir certaines caractéristiques, la première étant la rapidité de la fonction en termes de complexité temporelle. La deuxième caractéristique est la distribution normale des indices de sortie et l'évitement de la concentration des sorties de la fonction de hachage sur des indices spécifiques. La troisième caractéristique est le déterminisme de la sortie, c'est-à-dire que si l'entrée est la même, la sortie de la fonction de hachage doit également être la même.

Des nombres premiers sont utilisés pour créer des fonctions de hachage et uniformiser la sortie de ces fonctions, prévenant ainsi les collisions. Une collision dans les fonctions de hachage se produit lorsque des entrées différentes produisent la même sortie. Il existe deux stratégies connues pour résoudre les collisions dans les fonctions de hachage, qui sont le chaînage séparé (Separate chaining) et le sondage linéaire (linear probing).

Dans la méthode du chaînage séparé (Separate chaining), un tableau ou une liste chaînée supplémentaire est utilisé pour stocker plusieurs valeurs à un seul indice. Par contre, la stratégie du sondage linéaire (linear probing) consiste à trouver le premier emplacement vide suivant dans le tableau lorsqu'une collision se produit et à y insérer la paire clé-valeur.

Avec une implémentation optimisée de la fonction de hachage, il est possible d'atteindre une complexité temporelle de l'ordre de $O(1)$, indépendamment de la taille des données d'entrée n, permettant ainsi de stocker, de mettre à jour ou de récupérer des données très rapidement, et d'augmenter considérablement la vitesse d'enregistrement et de recherche des données.

Dans l'implémentation réalisée pour la transformation d'un maillage en graphe dual, une table de hachage a également été utilisée pour stocker et récupérer de manière ordonnée les arêtes du maillage. La méthode de stockage consiste à utiliser le numéro du premier sommet de l'arête, accompagné d'un indice, pour générer la clé de hachage. L'opération de récupération commence également à partir du premier sommet, et les clés pertinentes sont générées avec leurs indices pour que les arêtes récupérées soient ordonnées en fonction du numéro de leur premier sommet. Les résultats de l'implémentation de la table de hachage pour le tri des arêtes et la transformation du maillage en graphe dual sont présentés dans la Figure 6. Ces résultats montrent une meilleure performance de cet algorithme par rapport au tri par sélection et à l'arbre AVL, bien que la structure de l'arbre de tas ait réalisé l'opération de transformation dans un temps plus court.

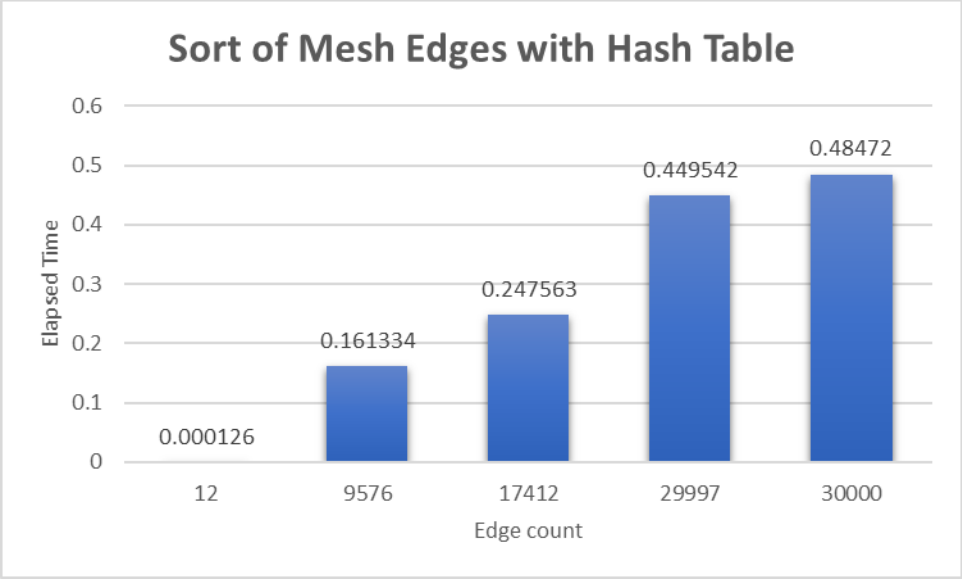


Figure 6. Résultats de l'utilisation d'une table de hachage pour le tri des arêtes sur des maillages avec différents nombres d'arêtes

3. Coloration du graphe dual

Pour colorer le graphe dual transformé à partir d'un maillage, la première étape consiste à déterminer un nœud de départ, puis à calculer la distance de chaque nœud par rapport à ce nœud de départ. Une fois cela établi, on peut attribuer une couleur spécifique au nœud de départ et changer la couleur utilisée pour colorer les autres nœuds en fonction de leur distance par rapport au nœud de départ. L'algorithme de Dijkstra a été utilisé pour déterminer les plus courtes distances entre les nœuds. L'algorithme de Dijkstra trace un arbre des chemins les plus courts allant du nœud de départ aux autres nœuds du graphe.

La méthode de fonctionnement de l'algorithme de Dijkstra est la suivante : après avoir choisi un des sommets comme point de départ, les nœuds voisins de ce sommet sont examinés. Le nœud voisin qui est accessible depuis le nœud de départ avec le poids le plus faible est sélectionné comme le nœud suivant. L'algorithme s'arrête finalement dès qu'il trouve le chemin le plus court du point de départ à la destination. Pour trouver les chemins les plus courts entre les autres nœuds restants, les nœuds actuellement sélectionnés sont également utilisés comme nœuds intermédiaires.

Dans l'implémentation réalisée pour transformer un maillage en graphe dual, la distance entre le nœud de départ et les autres nœuds sur le graphe est conservée dans un tableau nommé "Distance". De plus, dans cette implémentation, un tableau appelé "IncludeSet" est utilisé pour conserver les nœuds qui ont été sélectionnés jusqu'à présent, dont le

chemin le plus court par rapport au nœud de départ a été calculé. Une fonction nommée "MinDistance" est également utilisée dans l'implémentation, qui trouve la plus courte distance entre les nœuds non sélectionnés pour le calcul de la distance par rapport au point de départ.

Dans un graphe non orienté comme le graphe dual transformé à partir d'un maillage, chaque arête du graphe est parcourue exactement deux fois, et dans un graphe orienté, elle est parcourue exactement une fois. La mise en œuvre la plus simple de l'algorithme de Dijkstra utilise un tableau ou une liste chaînée, et dans notre programme, un tableau a été utilisé. Ainsi, dans ce cas, la complexité temporelle de l'algorithme est $O\ (|V^2| + |E|)$ où V est le nombre de sommets et E est le nombre d'arêtes du graphe. Au lieu de l'algorithme de Dijkstra, on peut également utiliser l'algorithme de Floyd, qui a une complexité temporelle plus élevée que l'algorithme de Dijkstra. Un exemple de coloration réalisée pour le graphe dual par l'implémentation effectuée est visible dans la Figure 7.

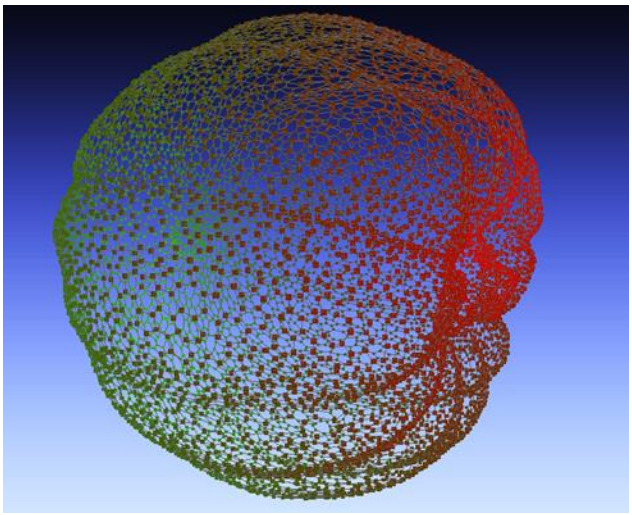


Figure 7. Graphe dual coloré après transformation à partir d'un maillage en utilisant l'algorithme de Dijkstra