Crypto-compression d'objets 3D
CR 2

Master 2 Informatique, IMAGINE
Université de Montpellier
2021 - 2022
GITHUB:https://github.com/EmeryBV/Crypto-compression_of_3D_objects

Équipe :

- Charles Sayamath
- Emery Bourget-Vecchio

UNIVERSITÉ DE MONTPELLIER

FACULTÉ DES SCIENCES MONTPELLIER

Choix de l'algorithme de compression

Au vu des différents algorithmes que nous avons pu présenter dans le compte-rendu n°1, nous avons décidé de nous orienter sur une compression mono-résolution

Codage de la connectivité

Définition

- Sommet libre

  Un sommet libre correspond à un sommet qui n'a pas encore été encodé

- Arête libre

  Une arête libre correspond à une arête qui n'a pas encore été encodée

- Liste active

  La liste active contient les sommets composant un cycle de vertex qui sépare le maillage en deux : la partie intérieure qui comprend les arêtes encodées et la partie extérieure contenant les arêtes non encodées.

- Sommet Focus

  Le sommet focus correspond au sommet sur lequel on va effectuer le parcours de voisins afin d'encoder la valence de ceux-ci.

- Sommet Full

  Un sommet est full lorsque toutes ses arêtes voisines ont été encodées

Pour coder la connectivité, nous avons décidé d'utiliser l'approche "valence-driven". Cette approche est actuellement l'une des meilleures pour coder la connectivité, spécifiquement avec la méthode de Alliez et Desbruns (basée sur l'algorithme de Touma et Gotsman) qui permet d'encoder un sommet avec un maximum de 3.24 bits. Cette borne est la même que la limite théorique possible en énumérant tous les graphes planaires possible.

Notre objectif va être donc d'implémenter dans un premier temps l'algorithme de Touma et Gotsman et de si possible appliquer les modifications de Alliez et Desbruns pour améliorer les performances de l'algorithme.

Algorithme de Touma et Gotsman

Le but de cet algorithme est de partir d'un triangle quelconque du maillage et de itérativement parcourir tous les sommets du maillage afin de les encoder à partir de leur valence.

On obtient à la fin une liste contenant une suite d'instructions qui va nous permettre de reconstruire le maillage originel. Les instructions possibles sont :

- **add**<valence> : correspond à l'encodage de la valence d'un sommet
- **split**<offset> : correspond à l'opération de split d'une liste active
- **merge**<index><offset> : correspond à l'opération de merge de deux listes actives

Dans un premier temps, nous allons initialiser notre liste active en y ajoutant les sommets d'un triangle quelconque. On encode ainsi ces sommets à l'aide de l'instruction **add**<valence>.

On prend un sommet qui deviendra le sommet focus. On parcourt par la suite les voisins du sommet focus dans le sens horaire inverse. On va par la suite avoir trois cas différents :

- ➢ Si l'arête qui sépare le focus du voisin n'a pas été encodée et que le voisin ne l'est pas également, on va alors l'ajouter dans la liste active et l'encoder à l'aide de l'instruction **add**<valence>.
- ➢ Si l'arête n'a pas été encodée mais que le voisin l'a été et qu'il est dans la liste active, l'algorithme effectue une opération de split de la liste active à l'endroit du sommet voisin. On va alors se retrouver avec deux listes actives différentes et on continue l'encodage sur celle qui est la plus grande. L'opération de split est indiquée à l'aide de l'instruction **split**<offset>, offset étant le nombre de vertex de la liste active séparant le sommet focus et le sommet voisin dans le sens horaire.
- ➢ Si l'arête n'a pas été encodée mais que le voisin l'a été et qu'il est dans une liste active secondaire, l'algorithme effectue une opération de merge entre les deux listes actives et continue l'encodage sur celle-ci. L'opération de merge est indiquée à l'aide de l'instruction **merge**<index><offset>, offset étant le nombre de vertex de la liste active séparant le sommet focus et le sommet voisin dans le sens horaire et index l'indice de la liste active à fusionner.

Lorsque toutes les arêtes d'un sommet ont été encodées, le sommet focus est alors retiré. On passe ensuite au sommet suivant dans la liste active, et ainsi de suite jusqu'à arriver à l'encodage complet du maillage.

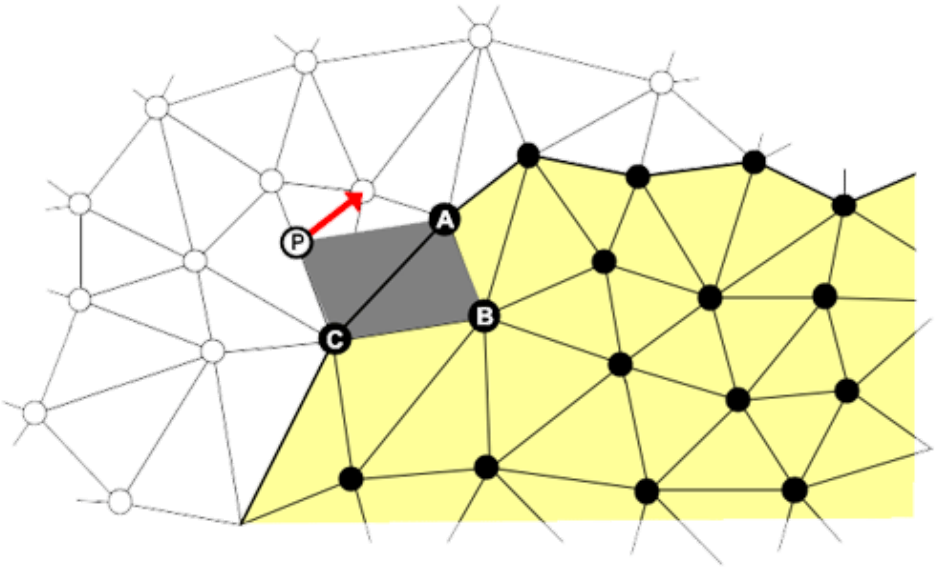
La liste d'instruction va nous permettre de reconstruire notre maillage original. Chaque instruction **add** signifie qu'un vertex doit être créé est relié au précédent de la liste tandis que les instructions **merge** et **split** vont indiquer qu'il faut effectuer ces opérations sur la liste d'instruction.

Codage de la géométrie

Comme décrit dans le compte-rendu n°1, la plupart des méthodes de compression de la géométrie d'un maillage se base sur 2 mécaniques:

Quantification: Les coordonnées des sommets sont généralement des nombres flottants. Si l'on prend deux sommets proches, les premières décimales de leurs coordonnées seront normalement les mêmes, mais à un certain moment, les derniers décimales seront sûrement totalement chaotiques. Ainsi l'entropie pour coder ce style d'information sera très élevé car il sera presque impossible de réaliser de la prédiction. Pour pallier ce problème, la solution est de rendre nulles toutes les décimales trop superflues, c'est la quantification. En appliquant cette quantification, on a donc une compression avec perte, cependant cela n'a pas de grand impact puisque les données retirées ont des valeurs tellement faibles que cela reste négligeable.

Prédiction: Suite à la quantification, nous allons utiliser la prédiction par parallélogramme qui va nous permettre de prédire la position d'un sommet. Chaque nouveau sommet est prédit à l'aide d'un triangle incident afin de former un parallélogramme. La différence entre la véritable position du sommet et la position prédite est alors encodée.



Exemple de codage par prédiction, le vecteur rouge représente la différence entre le sommet prédit et la véritable position du sommet.

Choix de l’algorithme de chiffrage

Pour le chiffrement nous n’avons pas encore défini la méthode que l’on utilisera.

Avancement

Nous avons pour le moment implémenté la structure et les classes nécessaires à l’algorithme.

Nous avons également codé le parser qui va nous permettre d’analyser notre fichier .obj. On va ainsi récupérer les données et les assigner à une classe **Triangle** composé de trois arêtes, **Edge** composé de 2 sommets et **Vertex** composé d’un index, des coordonnées XYZ, de ses voisins triés dans le sens anti-horaire, de l’état encoded qui permet de savoir s’il est déjà encodé et de sa valence.

Nous commençons ainsi l’implémentation de l’algorithme de compression.

Sources:

- Pierre Alliez: Compression de maillages