Informatique graphique et réalité virtuelle
Projet-Approximaton variationnelle de forme[1]

Jie CHEN, Yushan SUN

1.Introduction

Variational Shape Approximation(VSA) est une méthode de simplification de surface par partitionnement. Il a le résultat précis et optimal pour les maillages, mais parce qu'il utilise un algorithme itératif il est plus lent et la convergence n'est pas garantie. Il inclut trois étapes principales pour choisir les pointes plus utiles, ils sont 'Initial Seeding', 'Distortion-minimizing Flooding' pour faire une Partitionnement Géométrique et 'Proxy Fitting' pour renouveler les proxies. On a bien fini cette partie de partitionnement. De plus, dans la partie de 'Remeshing', il y a quatre étapes, 'Anchor Vertices', 'Edge Extraction', 'Triangulation' et 'Polygons'. Même si nous avons des problème dans 'Triangulation', nous avons presque fini. Tous les fonction que nous utilisons sont métrique L2,1.

2. Partitionnement Géométrique

Traitement principale:

$\{\text{findNeighbours}()\}_{\text{proxy}}$->$\{\text{partitioning}()\}_{\text{first seed}}$ ->$\{\text{proxyFitting}()$-> $\text{partitioning}()\}_{\text{itération}}$

2.1 Chercher des proxies et Distorsion minimisant inondations

Pour commencer plus simple, on choisit les triangles comme grains par hasard. On considère des triangles comme un proxy, on commence d'agrandir les régions des triangles. Comme l'algorithme de Lloyd, on voudrait que le triangle qui va être ajoute Distorsion minimisant inondations à des régions est plus <proche> de proxy, c'est à dire qu'il a une erreur de distorsion plus petite, la formule est dans l'annexe B. Du coup pour chacune triangle de grain Ti, on insère ses trois adjacent triangles Tj dans une queue prioritaire globalement, avec une priorité égale à son erreur de distorsion. Et on ajoute une étiquette supplémentaire indiquant l'étiquette i de la procuration qu'ils sont testés contre (un triangle peut donc apparaître jusqu'à trois fois dans la file d'attente, avec différentes étiquettes et les priorités). Ensuite on procède les triangles éjecté par la file d'attente avec moins de distorsion jusqu'à ce que la file d'attente prioritaire soit vide. Pour chaque triangle sauté hors de la file d'attente, on vérifie son assignation de proxy: si elle a déjà été attribué à un proxy, on ne fait rien et aller à le triangle suivant dans la file d'attente; autrement, on assigne à la région de le proxy indiqué par la balise, et appuyez sur le (jusqu'à deux) sans étiquette incidents triangles dans la file d'attente avec la même étiquette. Quand la file d'attente prioritaire a été vidée, chaque triangle a été attribué à une procuration: par conséquent, nous avons une nouvelle partition.

On a modifié la classe Triangle et a ajouté 'lable', 'tag', 'err' et ses 'neighbours'. On a créé une classe Proxy qui inclut l'infrolation de Xi, Ni, des ' adjacentProxy' etc. Pour le gérer facilement, on a créé une chaîne de erreur queue 'errque'. Chaque fois, on insère le triangle avec une erreur dans cette queue et pop éjecter le triangle avec plus petite erreur. Finalement, on a des proxies qui inclut l'information de ses triangles, de ses vertex,

2.2 Proxy fitting

Une fois que on a trouvé une nouvelle partition R sur la surface S, on souhaite maintenant mettre à jour les procurations respectives Pi = (Xi, Ni) dans l'ordre pour eux d'être les meilleurs représentants de leur mise à jour nouvellement associés région Ri. Cette minimisation se fait facilement en utilisant les équations données dans l'annexe B.

3. Remesh

Traitement principale:
vertexClass()->edgeExtraction()->findMoreAnchor()->triangulation()->insertTriangle()

3.1 Calculer des anchor vertex

Depuis les procurations peuvent être considérés comme approximatifs faces du maillage final, nous devons mettre les sommets à l'intersection de les procurations. Ainsi, nous créons un sommet de l'ancre à chaque sommet d'origine où au moins trois régions se rejoignent. Afin de tenir compte de toutes les régions, nous vérifions ensuite si chaque limite de région a au moins trois sommets d'ancrage qui lui sont attachés; sinon, nous ajoutons simplement ancrage sommets en conséquence comme il garantira la présence d'au moins un polygone par région.

On a faire un classification pour stocker les ‘edge vertex’ et ‘anchor vertex’ par ses nombre de voisins proxies. Et on les distribue pour chaque proxy, en fin chaque proxy a l’information de ses ‘anchor vertex’ et ‘edge vertex’.

3.2 Extraction Edge

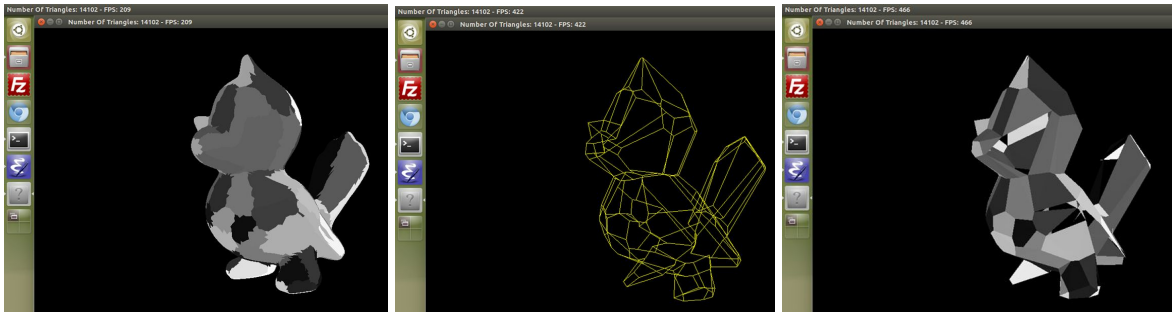
Pour avoir une bonne approximation du contour de chaque proxy, on va chercher plus de ‘anchor vertex’. Pour le faire, il faut savoir chaque bord d’un proxy. Il faut créer un class ‘anchorPair’ qui inclut l’inforation de deux ‘anchor vertex’ d’un bord et les ‘edge vertex’ entre de ce bord. On a donc créer un vecteur dans chaque proxy pour stocker chaque bord ou chaque ‘anchorPair’. Finalement, on calcule selon un algorithme simple de subdivision corde longueur récursive pour ajouter nouveau ‘anchor vertex’ et renouvele le ‘anchorPair’ dans chaque proxy.

3.3 Triangulation

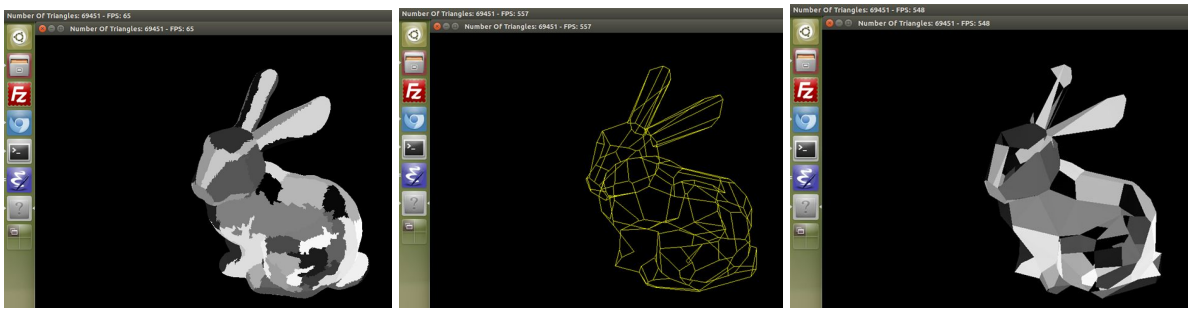
Avec les sommets et les arêtes ancrage définis, nous ont déjà un maillage polygonal. Toutefois, lorsque le nombre des proxies est assez petit, les polygones n’ont aucune garantie d'être presque plat ou convexe. Ainsi, nous avons besoin de trianguler ce graphique initial afin d'être en mesure d'appeler vraiment un maillage. Cela se fait à travers un "Discrète" Contrainte triangulation de Delaunay (CDT) de faire le traiter robuste pour toute sorte de rapprochement extrême: en effet, nous créera triangles Delaunay-comme dans chaque région, tout en limitant les bords à base d'ancrage existants pour faire partie de la finale triangulation.

Dans cette étape, on les suivi et on trouve que le résultat des triangles finales ne sont pas assez, c’est-à-dire, il manque quelque triangle, et j‘ai trouve cet algorithme il peut pas garantir que tous les ‘anchor vertex’ et tous les bord sont utilisé dans la triangulation, j’ai donc essayé de compléter les triangle selon l’information des ‘bord’ et des ‘anchor vertex’ qui ne sont pas encore utilisé, mais je n’ai pas réussi, je donc dessine tous les bord.

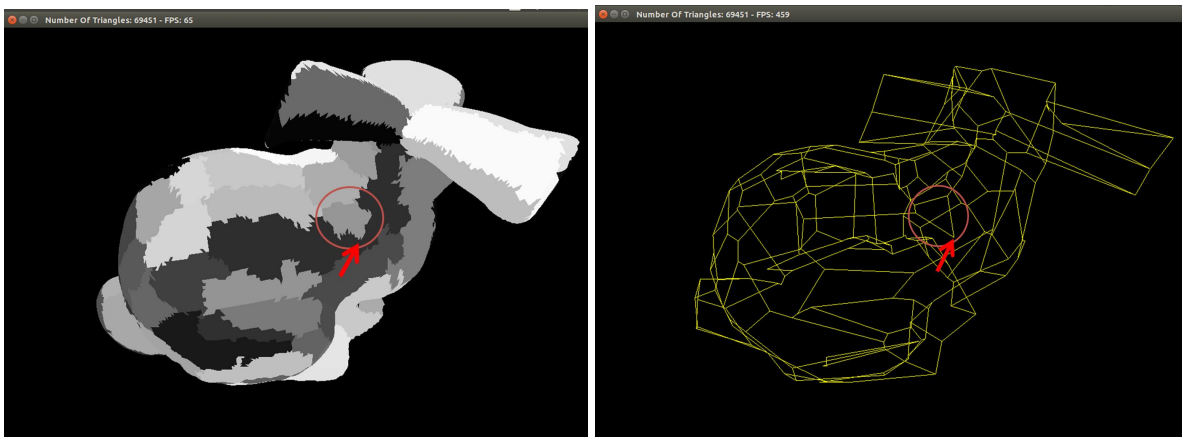
4. Résultat



(a) Partitionnement (b) Bords des proxies (c) Remesh
(tweety.off, 7053 vertex; 14102 triangles; 100 proxies; 10 itération; 18 seconds)



(d) Partitionnement (e) Bords des proxies (f) Remesh
(bunny.off, 35947 vertex; 69451 triangles; 100 proxies; 10 itération; environ 5 minutes)



(g) Partitionnement (h) Bords des proxies avec ‘edge extration’

Le premier rang de résultat de ‘tweety’ et le deuxième hang de ‘bunny’, on peut voir que le portrait principal est bien exprimé mais sans ‘Extraction Edge’. En revanche, le troisième rang de résultat ‘bunny’ utilisé ‘Extraction Edge’, il conserve plus d’information de contour de proxy.

5. Conclusion

Le partitionnement variationnel conserve bien le contour de propriété géométrique d’objet, mais il est lentement pour le faire. Il est donc utilisé quand’il n’existe pas trop de triangles.

Référence

[1] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. ACM Transactions on Graphics (Proc. Siggraph) 23, 3 (2004), 905–914.