Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 8, n. 1, 2021.

Trabalho apresentado no XL CNMAC, Evento Virtual - Co-organizado pela Universidade do Mato Grosso do Sul (UFMS).

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Uma comparação de métodos indiretos para geração de quadrangulações

Jair de Melo Freitas[1]
Universidade Federal do Cariri, Juazeiro do Norte, CE
Vicente Helano F. Batista Sobrinho[2]
Universidade Federal do Cariri, Juazeiro do Norte, CE

Malhas poligonais são frequentemente utilizadas para a representação da superfície de objetos geométricos e possuem aplicações importantes em diversas áreas como a computação gráfica, engenharias e física computacional. Tal representação se baseia na ideia de decompor a superfície de um objeto complexo em um conjunto de polígonos simples, passando a representar sua forma por uma coleção de vértices, arestas e faces [2]. As faces mais utilizadas são triangulares (malhas triangulares) ou quadrilaterais (malhas quadrangulares). Malhas triangulares são bastante utilizadas em aplicações de computação gráfica [5], enquanto malhas quadrangulares apresentam vantagens em soluções numéricas pelo método dos elementos finitos [3].

Dentre os métodos de geração automática de malhas quadrangulares, os chamados métodos indiretos [1], que utilizam uma malha triangular como ponto de partida, destacam-se devido à facilidade de implementação e ao custo computacional reduzido. Neste trabalho foram avaliados experimentalmente dez métodos indiretos. Os algoritmos foram implementados na linguagem de programação C++, utilizando estruturas de dados e funções da biblioteca *Visualization and Computer Graphics Library* (VCG)[3], versão 1.0.1. Três destes algoritmos fazem parte da VCG e os demais foram implementados pelos autores deste trabalho. O código desenvolvido e as malhas sobre as quais foram realizados os testes estão disponíveis em `https://github.com/Jairjua/quadrangulations`.

Os algoritmos analisados consistem de uma primeira etapa de agrupamento de triângulos adjacentes e da aplicação de uma segunda etapa de subdivisão, já implementada na VCG e utilizada por todos os algoritmos em questão, produzindo uma malha final composta somente por quadriláteros. A etapa de subdivisão consiste em inserir um novo vértice, no baricentro de cada face triangular ou no ponto médio de uma das diagonais interiores de cada face quadrangular, e conectá-lo a novos vértices inseridos no ponto médio de cada aresta da própria face. Por conveniência, eles foram agrupados e numerados como a seguir:

- Algoritmo de Subdivisão Simples (SS): realiza somente uma etapa de subdivisão, aqui considerado apenas para efeito de comparação.
- Algoritmos Gulosos por Aresta (GA-1, GA-2 e GA-3): realizam agrupamentos antes da subdivisão. Uma fila de prioridade de máximo é criada para as arestas e então tenta-se agrupar os triângulos que compartilham cada aresta dessa fila. Cada aresta é associada a uma medida, diferente em cada algoritmo, conforme estratégia já bem utilizada [4].

[1]ja-ir@outlook.com
[2]vicente.sobrinho@ufca.edu.br
[3]http://vcg.isti.cnr.it/vcglib

- Algoritmos Gulosos por Face (GF-1 e GF-2): realizam agrupamentos antes da subdivisão com o auxílio de uma fila de prioridade, mas agora definida para as faces triangulares iniciais e usando uma medida de qualidade de triângulo. Tenta-se agrupar a face da vez com alguma face triangular adjacente, dando prioridade maior àquela com a maior aresta em comum.
- Outras Heurísticas de Construção Incremental (HCI-1, HCI-2, HCI-3 e HCI-4): realizam agrupamentos antes da subdivisão e, com exceção do algoritmo HCI-4, já estavam implementados na VCG. Consistem em percorrer todas as faces da malha triangular inicial e tentar agrupar a face da vez àquela adjacente que produza o melhor quadrilátero segundo uma medida de qualidade. Estes algoritmos se distinguem pela quantidade de travessias realizadas.

Todos os algoritmos foram aplicados em três malhas com características topológicas distintas: botijo (40.016 faces, 20.000 vértices, genus 5), filigree (5.000 faces, 2.372 vértices, genus 65) e gargoyle (194.256 faces, 97.130 vértices, genus 0). A partir dos dados coletados, observou-se que o algoritmo SS, o qual não utiliza agrupamento de faces, produziu-se uma malha com qualidade inferior e com maior número de faces e vértices inseridos. Também foi percebido o surgimento de padrões em torno dos vértices da malha original.

Dentre os algoritmos que realizaram uma etapa de agrupamento, o algoritmo HCI-1 foi o que mais realizou agrupamentos em todas as malhas e o que apresentou menor tempo de execução. Por outro lado, todos os demais apresentaram um aumento na média e no desvio padrão da qualidade das faces com relação ao algoritmo SS, com exceção do HCI-1, que permaneceu com a mesma média na malha botijo, além de apresentar a pior média nas outras malhas.

A qualidade dos vértices, representada aqui pela quantidade de vértices com valência 4, conhecidos como *vértices regulares*, apresentou um aumento significativo após a realização dos agrupamentos. Isso se deve ao fato de que o esquema de subdivisão insere vértices não regulares de valência 3 apenas nos triângulos isolados, enquanto que vértices regulares são inseridos nas faces quadrangulares, que são maioria nos algoritmos que realizam agrupamento. Todos os algoritmos que realizaram agrupamento apresentaram aproximadamente a mesma quantidade de vértices regulares na malha final, para todas as malhas consideradas neste trabalho.

Ao final, pôde-se concluir que a utilização de algoritmos de agrupamento antes de realizar a subdivisão produz malhas de melhor qualidade em termos tanto de valência dos vértices, quanto da qualidade das faces quadrangulares. Dentre os algoritmos analisados, de modo geral, o HCI-4 pode ser considerado como a melhor alternativa, uma vez que foi o segundo mais rápido e produziu malhas com qualidades das faces e dos vértices próximas dos melhores valores obtidos.

Referências

[1] Bommes, D., Lévy, B., Pietroni, N., Puppo, E., Silva, C., Tarini, M. and Zorin, D. Quad-mesh generation and processing: A survey, *Computer Graphics Forum*, 32:51-76, 2013. DOI:10.1111/cgf.12014.

[2] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P. and Levy, B. *Polygon Mesh Processing*. A K Peters/CRC Press, 2010.

[3] D'Azevedo, E. F. Are bilinear quadrilaterals better than linear triangles?, *SIAM Journal on Scientific Computing*, 22:198-217, 2000. DOI:10.1137/S106482759630406X.

[4] Velho, L. Quadrilateral meshing using 4-8 clustering, *Proceedings CILANCE 2000 Symposium on Mesh Generation and Self-Adaptivity*, 61-64, 2000.

[5] Yuksel, C., Lefebvre, S. and Tarini, M. Rethinking texture mapping, *Computer Graphics Forum*, 38:535-551, 2019. DOI:10.1111/cgf.13656.