Universidade Federal de Santa Maria
Departamento de Eletrônica e Computação
Prof. Cesar Tadeu Pozzer
Disciplina: Computação Gráfica Avançada
E-mail: pozzer@inf.ufsm.br
Data: 19/08/2017

LOD – Level of Detail

1. Introdução

Programas gráficos, em tempo real, têm como característica marcante o uso de algoritmos que otimizam ao máximo o pipeline de renderização, de forma a garantir alta qualidade gráfica, com uma razoável taxa de frames por segundo (FPS), como o uso da menor resolução geométrica possível (número de polígonos).

Dependendo do tipo de cena a ser visualizada, diferentes técnicas de otimização são aplicadas. Para interiores, como prédios, algoritmos de BSP, portais e PVS desempenham um papel importante na eliminação de regiões da cena que não são visíveis, garantindo deste modo uma redução de dados a serem processados pela placa gráfica.

Aqui é apresentado um outro conceito de otimização, que pode ser usado em cenários abertos, como terrenos, e em objetos dispostos em qualquer tipo de cena. Este conceito é conhecido como LOD (*Level of Detail*), e se aplica a toda técnica que altera a complexidade geométrica de um modelo em favor de ganho de desempenho no pipeline gráfico, procurando ao máximo manter a geometria "visual" do objeto.

2. LOD

Existem diversos algoritmos que exploram as propriedades de LOD, que têm como característica alterar a resolução geométrica de objetos em função de parâmetros como, por exemplo, a distância da câmera e propriedades geométricas do objeto.

Considerando-se o parâmetro câmera, sabe-se que à medida que objetos se distanciam, devido a projeção em perspectiva, tornam-se menores e consequentemente menos visíveis (representados por um número menor de pixels na tela). Logo, mesmo representados por um número menor de polígonos em relação a resolução geométrica máxima (que deve ser usada quanto o objeto está muito próximo da câmera), continuam com o mesmo aspecto visual. Cabe ressaltar que aplicações como jogos, não necessitam alta fidelidade geométrica como uma aplicação de visualização científica, por exemplo. **O que interessa apenas é o aspecto visual obtido**.

Quanto ao parâmetro de propriedade geométrica, verifica-se que regiões menos planares, necessitam maior quantidade de polígonos para terem correta representação em relação a regiões planas. Isso pode tanto ser aplicado sobre terrenos, como objetos e personagens. Como exemplo, pode-se comparar o grau de refinamento da face com o corpo de um personagem em um jogo. Dependendo do grau de realismo, pode-se ter uma quantidade muito maior de polígonos na face para expressar emoções do que em todo o resto do corpo. Entretanto, se o personagem estiver longe da câmera, pode-se reduzir esta complexidade sem perder em realismo.

Basicamente existem dois tipos de LOD: os discretos e os contínuos. Estes dois tipos de LOD são discutidos nas próximas seções.

2.1. LOD Discreto

A estratégia mais simples de se implementar LOD é gerar uma seqüência de modelos pré-calculados, variando crescentemente na quantidade de polígonos (Fig. 1). Durante a renderização, necessita-se apenas

decidir qual modelo será usado, em função da distância da câmera. A medida que o objeto se afasta, troca-se o modelo corrente por outro com menos detalhes.

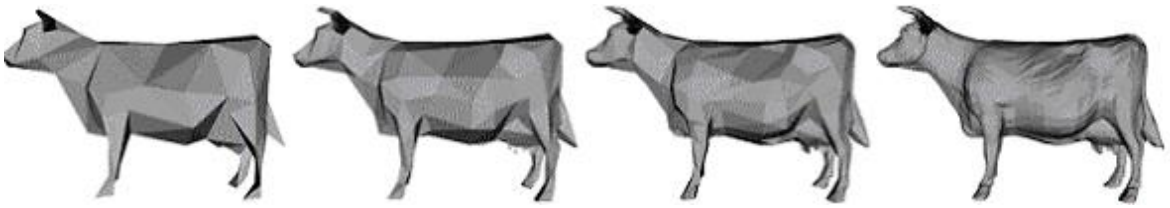


Figura 1: Modelos com 248, 532, 994 e 5.804 faces, respectivamente [1].

Cada modelo pode ser previamente gerado de forma manual, ou por algum método de LOD contínuo (Seção 2.2), de forma a maximizar a resolução com um menor número de polígonos possível.

Esta técnica tem como principal vantagem a eficiência, visto que necessita apenas definir, em função da distância da câmera, qual modelo será apresentado. Entretanto, a variação geométrica brusca na troca dos modelos pode ser facilmente perceptível, e produz um efeito chamado de *popping*. Uma possível solução é definir um grande número de modelos, com transições suaves, porém penaliza-se o modelo pela grande quantidade de memória necessária. Outra solução é definir um único modelo, e a medida que é distanciado, sofrer um aumento transparência, até o ponto onde não é mais enviado ao pipeline de renderização.

2.2. LOD Continuo

Esta estratégia faz a redução gradativa (simplificação), em tempo real, do número de polígonos de um objeto, segundo um critério geométrico ou perceptual, usado para definir a resolução em cada região da malha do objeto. Os critérios são caracterizados por uma métrica a ser aplicada na função de adaptação. O critério geométrico se refere a resolução entre a malha a ser desenhada e a imagem projetada. O perceptual leva em consideração a relevância que a informação tem para o olho humano, que se fixa em altas freqüências da imagem, localizadas na silhueta.

Pode-se definir estes dois critérios de forma mais detalhada [9]:

- A resolução deve variar com a distância e ângulo de projeção dos polígonos;
- Polígonos cuja normal é perpendicular ao vetor de visualização devem ser refinados, para ressaltar a silhueta;
- O centro da imagem deve ser mais refinado, pois é a região com maior probabilidade de estar sendo visualizada pelo observador;
- Se o observador estiver se movendo muito rápido, pode-se baixar a resolução.

Existem várias formas de fazer a simplificação de malhas. Nesta seção é apresentada a técnica de contração de vértices, como mostrado na Fig. 2. Uma solução é mover o vértice v_1 sobre v_2. Outra é mover v_2 sobre v_1, ou mover ambos para uma posição intermediária ou até mesmo fora da aresta que os une. A determinação de qual ação tomar não é trivial. [2] usa uma equação quádrica para encontrar a solução ótima, que tende a gerar malhas com maior qualidade.

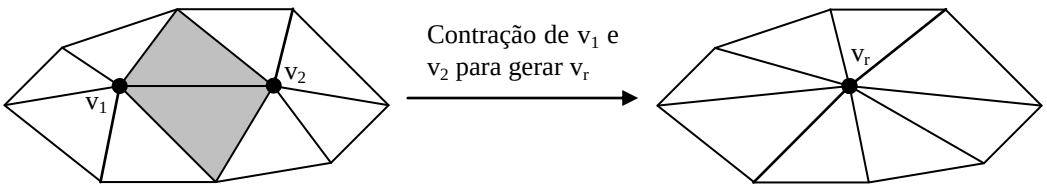


Figura 2: Contração (*Edge colapse*) dos vértices v_1 e v_2 resulta na geração de v_r. Ocorre também a remoção de dois triângulos e 3 arestas.

Este método avalia para cada vértice uma função de custo. O vértice que tiver o menor custo é o primeiro a ser contraído, quando uma simplificação for necessária. A cada contração de vértices, deve-se recalcular os custos dos vértices vizinhos. Cada triângulo da malha tem uma equação de plano associada, e um conjunto

de vizinhos. A função de custo, para mover um vértice, é a soma das distâncias ao quadrado entre cada um dos planos com a nova posição. Na Fig. 3 pode-se facilmente verificar a funcionalidade desta métrica de erro. Movendo-se o vértice V sobre C, não ocorre o deslocamento de nenhum plano. Entretanto, movendo-se C sobre V, ocorre um deslocamento do plano direito do cubo, que compartilha o vértice C, produzindo um custo maior que zero. Este custo foi resultado da deformação do objeto, que não ocorreu ao mover V sobre C. Um formalismo mais matemático sobre técnicas de LOD adaptativas pode ser visto em [3].

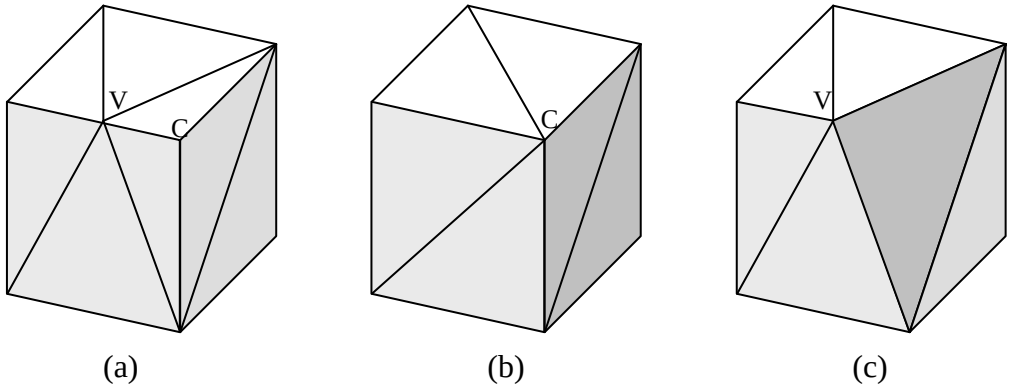


Figura 3: Vértice adequado para remoção [4]

3. Terrenos

A geração de terrenos, sem otimização de rendering, é uma tarefa relativamente simples, pois consiste em gerar uma malha de triângulos, a partir de um mapa de alturas (*height field*), que geralmente é obtido a partir de uma imagem em tons de cinza, onde o valor de cada pixel, que pode variar entre 0 e 255, é aplicado na coordenada y de cada vértice do terreno (Fig. 4).

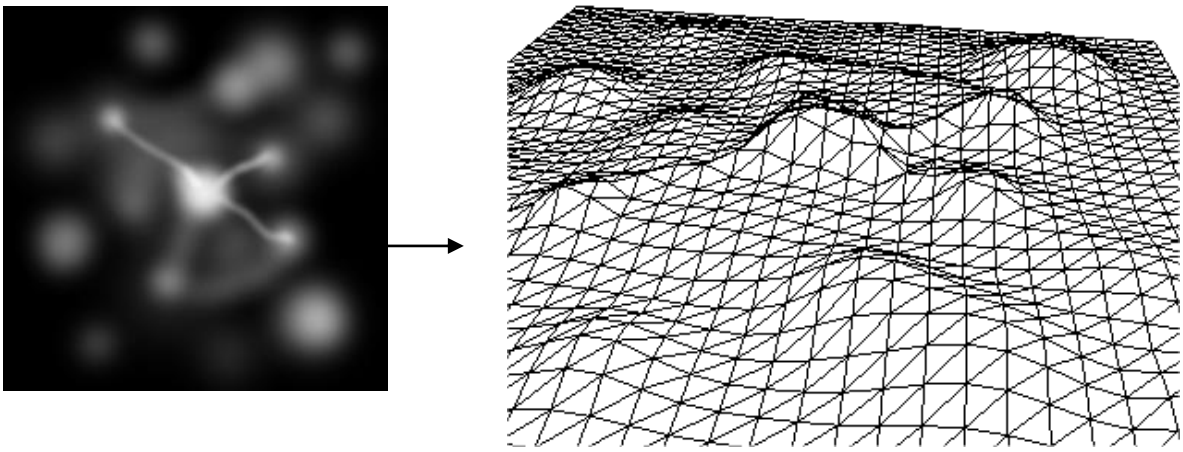


Figura 4: Geração de um terreno a partir de uma imagem em tons de cinza.

Esta estratégia produz representações regulares, onde todos os vértices estão alinhados nos eixos x e z. Isso pode não acontecer em situações reais, caso a obtenção dos dados for resultado de uma amostragem adaptativa, onde regiões mais acidentadas possuam um maior número de amostras (ver Seção 3.2.3).

Apesar de simples, a representação regular não é uma estratégia eficiente no que se refere a renderização, principalmente quando o terreno é extenso ou tem alta resolução. A cada frame, todos os vértices do terreno devem ser processados, independente de estarem visíveis ou não ao observador. Para contornar este problema, de imediato surgem duas estratégias, que geralmente são aplicadas seqüencialmente sobre o terreno: realizar culling, para determinar as regiões visíveis, e sobre elas, aplicar algum algoritmo de LOD.

3.1. Culling de Terrenos

Pela distribuição geográfica dos vértices do terreno ser retangular e planar (não volumétrica, como em um interior), geralmente dispostas no plano xz em linhas e colunas, a técnica de repartição do espaço por

quadtrees [5][6] é simples e fornece um meio fácil para a realização de culling. Esta técnica consiste em repartir o espaço recursivamente em quatro quadrantes, até que as regiões assumam um tamanho mínimo definido (Fig. 5).

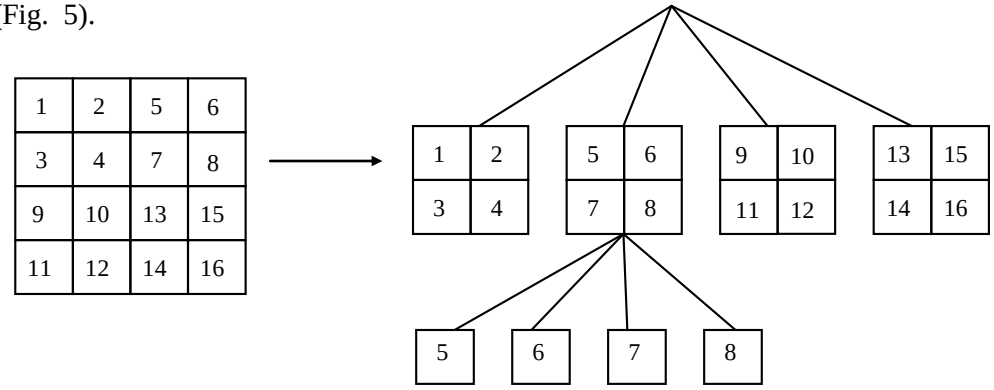


Figura 5: a) Mapa de um terreno com 16 regiões, b) Quadtree gerada em dois níveis.

Culling é um processo para determinar se um objeto, ou parte dele, não é visível para então não renderizá-lo. Os métodos padrões de culling comparam o volume envolvente do objeto com o frustum de visão. Se estiver fora, o objeto não é visível e é descartado.

Para o caso de terrenos, cada quadrante da quadtree é recursivamente testado com o frustum. Caso o quadrante esteja completamente dentro, a recursão é finalizada, o quadrante é renderizado e um novo quadrante é testado. Caso parte do quadrante estiver dentro, a busca segue para cada uma das folhas do nó testado. O frustum de visão é determinado em função das características da câmera, como abertura, posição e direção.

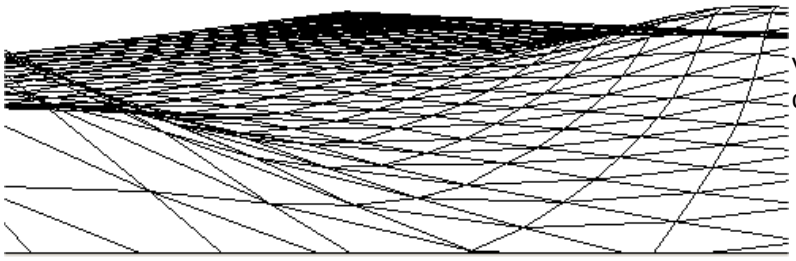
Na Fig. 6 é apresentado como o culling é realizado. Tendo-se a posição e direção da câmera, para este exemplo, define-se dois planos que delimitam o frustum de visão no plano horizontal xz, desconsiderando-se as alturas do terreno [10]. Caso um quadrante da quadtree estiver fora da área de frustum, será eliminada da renderização.

3.2. LOD de Terrenos

Diversas técnica de LOD podem ser aplicadas sobre terrenos. Neste trabalho são apresentadas três abordagens de LOD adaptativo, que fazem uso de quadtree para representar os dados e logo podem representar diferentes regiões com diferentes resoluções, e ao mesmo tempo facilitando o processo de culling.

3.2.1. Quadtrees com Resolução Variável por Quadrante

Uma forma simples e barata de realizar LOD em terrenos é calcular a distância da câmera a cada quadrante da quadtree que define o terreno. Esta distância é usada para determinar em qual resolução o quadrante será renderizado. Caso a distância for maior que um limite, pode-se também realizar culling. Na Fig. 7 é apresentado o resultado da aplicação deste método [10]. Observa-se nitidamente as regiões onde ocorrem as transições de resolução.



Visão a partir da câmera.

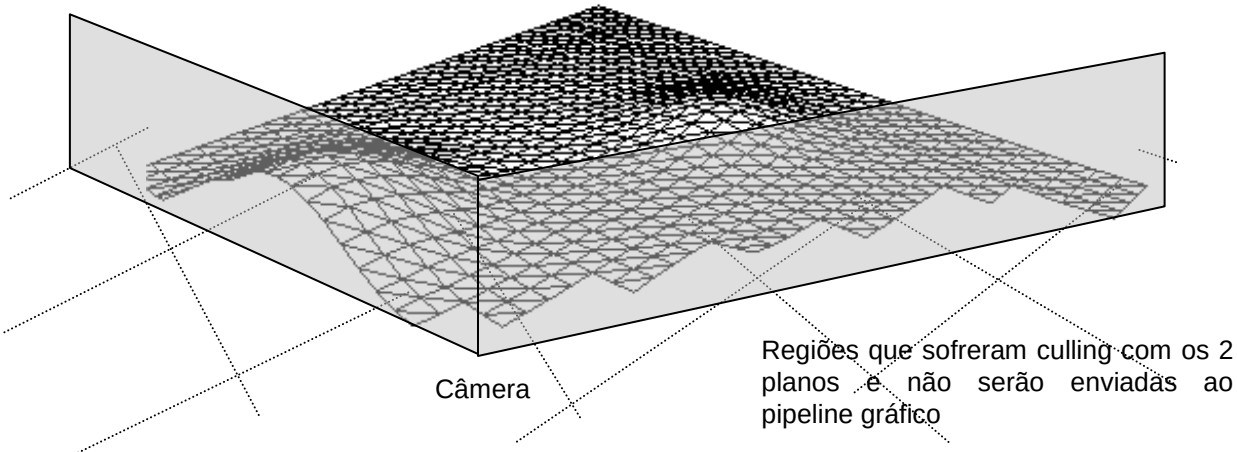


Figura 6: Culling: a) Região qualquer do terreno. b) Posicionamento da câmera que gerou a), definição dos planos do frustum e cálculo das regiões visíveis (quadrantes da quadtree dentro do frustum).

Esta técnica exige cuidados especiais na junção de quadrantes com resoluções diferentes, pois a presença de T-vértices [4] pode levar ao aparecimento de falhas (fendas), que podem ser facilmente visíveis. Estas falhas ocorrem quando faces vizinhas não compartilham os mesmos vértices. A Fig. 7(d) evidencia este problema e em (c) é apresentada uma solução. Maiores detalhes do problema podem ser vistos em [7].

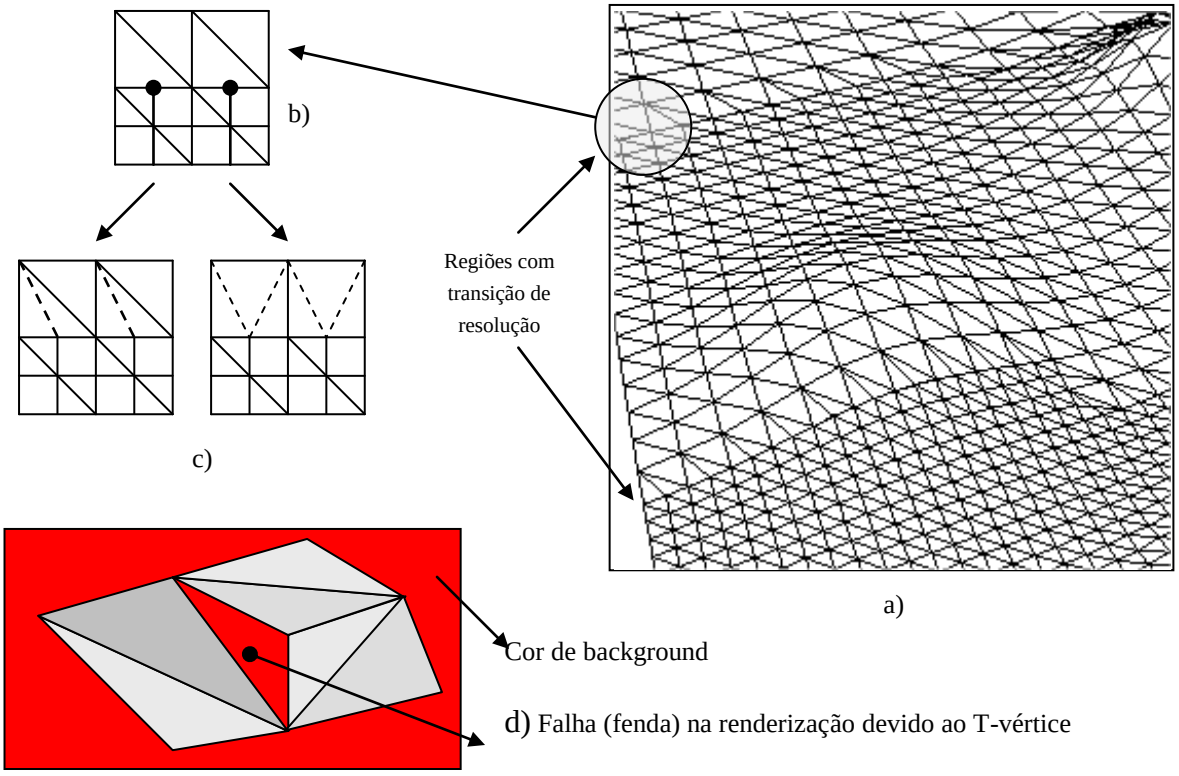


Figura 7: LOD adaptativo em relação a distância.

3.2.2. Quadtrees com Resolução Variável por Região

Nesta estratégia [8], define-se como elemento básico uma estrutura de 9 vértices, com 4 quadrantes (Fig. 8 (a)), que podem ser subdivididos de forma recursiva até a obtenção da resolução desejada. Dos 9 vértices, 5 estão sempre habilitados, e os demais devem ser verificados, segundo atributos dos vértices do terreno, além da posição da câmera. Na Fig. 9 (b-f) são apresentadas algumas configurações de vértices.

A cada passo do algoritmo, deve-se determinar quais vértices serão renderizados para então definir os triângulos que incluem estes vértices.

Quando um quadrante é dividido (Fig. 9 (g)), deve-se habilitar o vértice central e verificar quais vértices e arestas devem ser habilitados, bem como possíveis propagações em outros quadrantes, para evitar a formação de T-vértices, como mostrado na Fig. 9 (h).

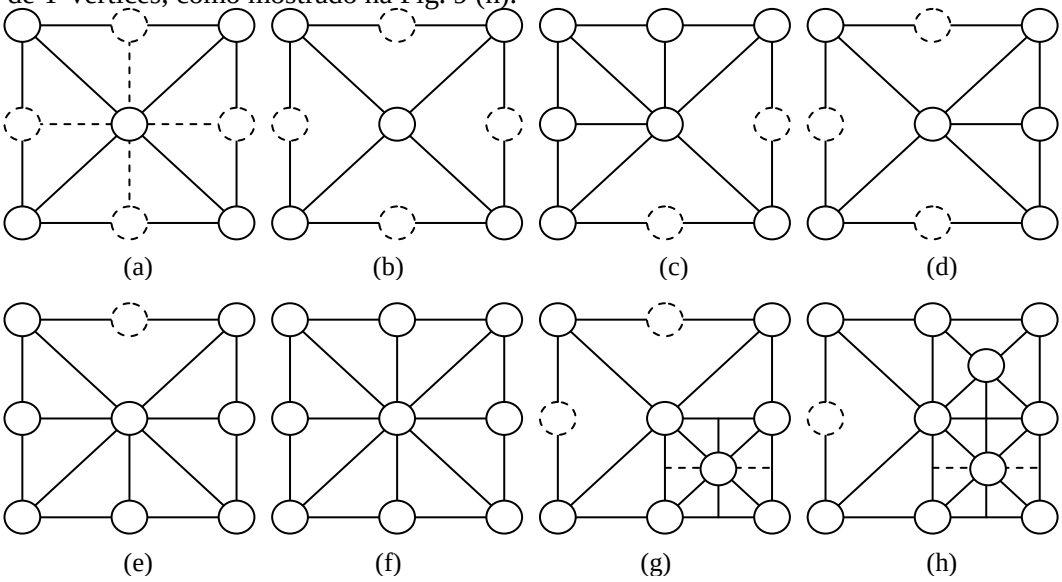
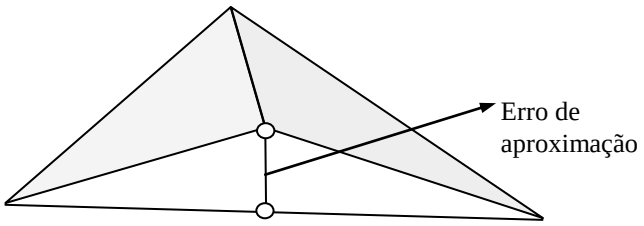


Figura 9: Vértices e arestas pontilhados não estão habilitados. a) Configuração inicial. g-f) Algumas possíveis divisões dos quadrantes. g) Divisão do quadrante inferior direito. h) Propagação no quadrante superior para evitar a formação de T-vértice.

O trabalho mais cuidadoso está na definição de quais vértices serão habilitados. A técnica adotada é a de erro de interpolação de vértices, que é a diferença de altura entre a posição correta do vértice com a altura do vértice do triângulo que aproxima o vértice quando este estiver desabilitado (Fig. 10). Vértices com maior erro devem ser habilitados preferencialmente em relação a vértices com menor erro. Outro variável que deve ser levada em consideração é a distância do vértice com a câmera.



3.2.3. QuadLod

A técnica de quadlod [9] é baseada em uma malha irregular, decomposta em blocos regulares (Fig. 11), organizada por meio de uma quadtree. Um bloco regular é uma região retangular, cujo interior é sempre uma malha irregular. A fronteira de um bloco regular é composta por 4 poligonais retas, cada uma composta por arestas da malha irregular. Dois blocos regulares vizinhos têm fronteiras comuns, para evitar a formação de T-vértices. Todo terreno pode ser visto como um único bloco regular, que em um processo recursivo de subdivisão, gera diversos blocos regulares em diferentes níveis hierárquicos. É usado o processo de LOD contínuo para fazer a remoção de arestas dentro de uma malha irregular. A estrutura regular permite uma rápida eliminação de regiões do terreno fora do campo de visão (culling).

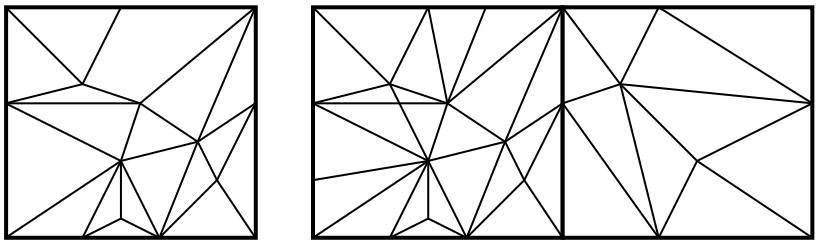


Figura 11: a) Bloco regular com interior irregular. b) Junção de dois blocos regulares, com interiores em diferentes níveis de resolução, mas que compartilham vértices vizinhos

3.2.4 Técnicas Atuais de LOD para terrenos

As técnicas mais atuais de LOD para terrenos são: Geometry Clipmapping [11], CDLOD [12] e ROAM [13]. Para maiores detalhes de implementação de LOD em GPU, consulte [14, 15].

3.2.4. Conclusões

Comparando-se as três técnicas de LOD para terrenos apresentadas, ambas têm vantagens e desvantagens. Sem dúvida a maior vantagem da primeira técnica se refere a velocidade de processamento, pois considera apenas a distância como parâmetro de resolução. A principal desvantagem, que é uma das principais vantagens da segunda e terceira técnicas, se refere ao fato de não levar em consideração aspectos geométricos do terreno. Caso uma região com muito detalhes estiver um pouco afastada, poderá perder detalhes perceptíveis, mesmo quando vistas de longe, e ainda produzirá o efeito de popping quando ocorrer a transição do nível de resolução. Outra vantagem da segunda técnica é que, mesmo regiões próximas, caso forem muito planas (erro de interpolação baixo), podem ser representadas por poucos polígonos.

A principal característica da técnica de quadlod é que tem as vantagens da representação regular, bem como das irregulares.

4. LOD AI

O Lod de malhas tem como principal característica fazer a redução da quantidade de dados enviado para a placa gráfica. Entretanto, para isso, um considerável gasto adicional de CPU faz-se necessário.

Para reduzir o consumo de CPU pode-se fazer uso de Lod sob um novo paradígma que está surgindo no desenvolvimento de jogos chamado Lod Ai. Quando a quantidade de agentes em um jogo é muito elevado, pode-se usar a mesma estratégia aplicada a polígonos em IA, ou seja, o que não é visível ou pouco visível pode receber menos atenção quando comparado com os dados sendo processados mais próximos do observador.

Lod Ai se refere ao controle de agentes (autônomos ou não) que estão fora do campo de visão da perspectiva do jogador, de modo que o comportamento dos agentes permanece realista com um baixo consumo de recursos.

5. Referências Bibliográficas

[1] Garland M., Heckbert P. S. Surface simplification using quadric error metrics. Proceedings of the 24th annual conference on Computer graphics and interactive techniques, p.209-216, August 1997.

[2] Garland M., Heckbert P. S. Simplifying surfaces with color and texture using quadric error metrics. IEEE visualization 98, pp. 263-269, july 1998.

[3] Eberly, David. 3D game engine design, a practical approach to real-time computer graphics. Ed. Morgan Kaufmann, 2001.

[4] Moller,T. and Haines,E. Real-Time Rendering, A.K.Peters Ltd, Natick, MA, 1999.

[5] Watt A. 3D computer graphics. Ed. Addison-Esley, 1993.

[6] Hearn D., Baker M. P. Computer graphics, c version. Ed. Prentice Hall, 1997.

[7] Ulrich T. Rendering massive terrains using chunked level of detail control. Available at http://tulrich.com/geekstuff/chunklod.html#downloads. April 2002.

[8] Ulrich T. Continuous lod terrain meshing using adaptative Quadtrees. Available at: http://www.gamasutra.com/features/20000228/ulrich_pfv.htm. February 2000.

[9] Toledo R. P. R. Quadlod, uma estrutura para a visualização interativa de terrenos. Dissertação de Mestrado, Puc-Rio. Abril de 2000.

[10] Poyart E. Visualizador de terrenos desenvolvido como trabalho acadêmico na PUC-Rio. 2002 (comunicação pessoal).

[11] Frank Losasso, Hugues Hoppe. Geometry clipmaps: Terrain rendering using nested regular grids. ACM Trans. Graphics (SIGGRAPH), 23(3), 2004.

[12] Strugar, Filip. "Continuous distancedependent level of detail for rendering heightmaps." Journal of graphics, GPU, and game tools 14, no. 4 (2009): 5774.

[13] Duchaineau, M., Wolinsky, M., Sigeti, D. E., Miller, M. C., Aldrich, C., and MineevWeinstein, M. B. (1997). ROAMing terrain: real-time optimally adapting meshes. In Proceedings of the 8th Conference on Visualization'97, pages 81–88. IEEE Computer Society Press.

[14] Alex Thomas Almeida Frasson, Handling Massive Terrains Based On Digital Elevation Data, UFSM, 2015, TCC.

[15] Alex T. Almeida Frasson, Tiago Augusto Engel, Cesar Tadeu Pozzer, Improving Terrain Visualization Through Procedural Generation and Hardware Tessellation, Sbgames 2016.