2
Técnicas e Trabalhos Relacionados

Um bom renderizador de modelos massivos tem que ser capaz de resolver três pontos: reduzir a complexidade da geometria onde ela não for necessária, não renderizar geometrias que estejam ocultas ou fora da pirâmide de visualização e usar pouco processamento durante a renderização da cena. Usar processamento de CPU demais fará o desempenho do renderizador ficar limitado pela capacidade de processamento da máquina, ao invés de se limitado pela capacidade da placa gráfica. Esta seção explica como as idéias de simplificação de modelos e descarte de objetos ocultos têm evoluído para permitir seu uso com modelos massivos. A explicação parte dos algoritmos clássicos de níveis de detalhe voltados para um único objeto e procura mostrar como eles foram estendidos para permitir o tratamento de grandes conjuntos de objetos. Nesta seção também são discutidas formas alternativas de criar representações simplificadas dos objetos do modelo como impostores ou grades de voxels.

Técnicas de simplificação de malhas já são estudadas há muitos anos. Em especial, algoritmos voltados para reduzir a complexidade de um único objeto extremamente tesselado, como os apresentados na figura 2.1, já estão num estágio bem maduro de desenvolvimento. A primeira publicação a tratar do assunto foi Clark (1976), que explica como um mesmo objeto pode ser representado por diferentes níveis de resolução, criados manualmente, quando estiver distante o suficiente da câmera. Schroeder et al. (1992) explica como vértices de um modelo podem ser eliminados, seguindo um critério de erro geométrico, para criar modelos com menor resolução automaticamente. Hoppe (1996) introduz uma estrutura de Progressive Meshes que armazena os diferentes níveis de resolução da malha como uma seqüência de refinamentos a partir de uma malha inicial simplificada. Essa estrutura também pode ser percorrida no sentido contrário, permitindo que a malha seja ajustada

continuamente para ficar com a resolução adequada. Como essa técnica não permite que a resolução do modelo varie de acordo com a direção de onde ele é visualizado, ela é denominada de *view-independent LOD*.

Figura 2.1 - Exemplos de modelos estudados em algoritmos clássicos de níveis de detalhe

Hoppe (1997) propõe uma nova estrutura hierárquica de *Progressive Meshes* que permite que diferentes regiões de um mesmo objeto sejam refinadas independentemente. Assim, regiões do objeto mais próximas à câmera podem ser mais refinadas, enquanto regiões mais distantes podem ser mantidas em resoluções menores. Uma idéia semelhante foi apresentada independentemente por Xia et al, (1997). Essa técnica é denominada de *view-dependent LOD*.

O primeiro problema a impedir que essas técnicas sejam usadas em modelos massivos é que elas são criadas especificamente para trabalhar em cima de objetos individuais. Usá-las em modelos formados por vários objetos, como ocorre em modelos de engenharia, não traz bons resultados. Alguns objetos podem ter características que possam ser eliminadas sem serem notadas quando o objeto for simplificado isoladamente, mas que criam grandes falhas na visualização quando simplificadas em conjunto com outros objetos. Simplificar objetos individualmente também pode dificultar a simplificação do objeto, fazendo com que ele seja representado com mais faces do que seria necessário caso os outros objetos fossem levados em consideração.

Essa característica é contornada por Luebke & Erikson (1997) que sugere que o modelo como um todo seja processado como se fosse um único objeto. Assim, uma única hierarquia de subdivisões/refinamentos é criada para todo o modelo. Essa estrutura é, segundo o autor, semelhante às *Progressive Meshes*,

mas permitindo a operação envolvendo mais de dois vértices a cada passo (o que permite a obtenção de uma estrutura mais compacta) e, principalmente, permitindo a união de vértices em objetos diferentes, para que as geometrias de múltiplos objetos possam ser combinadas em uma única, que represente o conjunto.

Outra solução encontrada foi permitir que os objetos fossem agrupados apenas quando fossem simplificados. O conceito de hierarquia de níveis de detalhe (HLOD) também foi introduzido por Clark (1979), mas nessa publicação tanto a simplificação quanto os agrupamentos de objetos são gerados manualmente. A idéia de agrupar objetos de forma automática, visando preservar as características visuais do conjunto foi introduzida por Erikson & Manocha (1998), que defende o uso níveis de detalhe estáticos para representar objetos ou grupos de objetos do modelo, já que estes podem ser armazenados em *display lists*, muito mais eficientes para serem usadas com as placas 3D atuais do que as geometrias dinâmicas como as *Progressive Meshes*. Outra grande vantagem do agrupamento de objetos, além de permitir a geração de uma simplificação mais fiel ao modelo original, é reduzir a quantidade total de objetos a serem enviados para a placa gráfica.

Erikson et al. (2001) aplica a técnica de HLOD em um visualizador de modelos massivos para cenas estáticas ou cenas dinâmicas com movimentação pouco freqüente de objetos. O modelo usado na visualização é gerado durante uma etapa de pré-processamento que, após gerar diversos níveis de detalhe para cada objeto existente no modelo, agrupa os níveis menos detalhados destes e continua a simplificação em cima dos grupos criados. Ao final do pré-processamento, teremos uma hierarquia em que a raiz contém um único objeto criado de forma a representar o modelo como um todo quando visto de longe e que se subdivide em grupos de objetos que representam o mesmo modelo de forma mais detalhada. As folhas da árvore montada nessa hierarquia irão conter os objetos originais do modelo.

Um dos defeitos dessa forma de geração de HLOD, como se pode ver nas figuras 2.2 e 2.3, é que, apesar de preservar bem a forma do conjunto de objetos, a cor final obtida na simplificação nem sempre representa bem o conjunto de

objetos. Uma boa representação deve ter como cor a média das cores dos objetos visíveis existentes. Apesar desse estudo levar em consideração a área das faces existentes durante o cálculo da geometria simplificada, ele falha ao não conseguir determinar quais faces estão ocultas para deixar de considerá-las no cálculo da cor final da geometria simplificada.

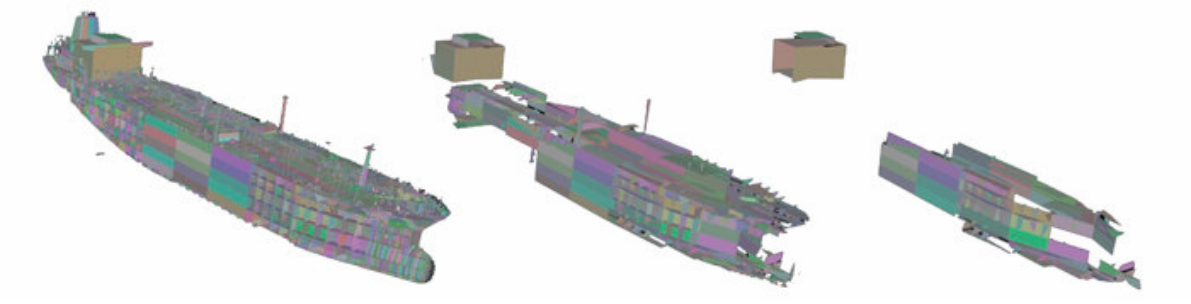

Figura 2.2 - Exemplo de uma técnica tradicional de LOD usada em modelos massivos. Figura extraída da publicação de Erikson et al. (2001)

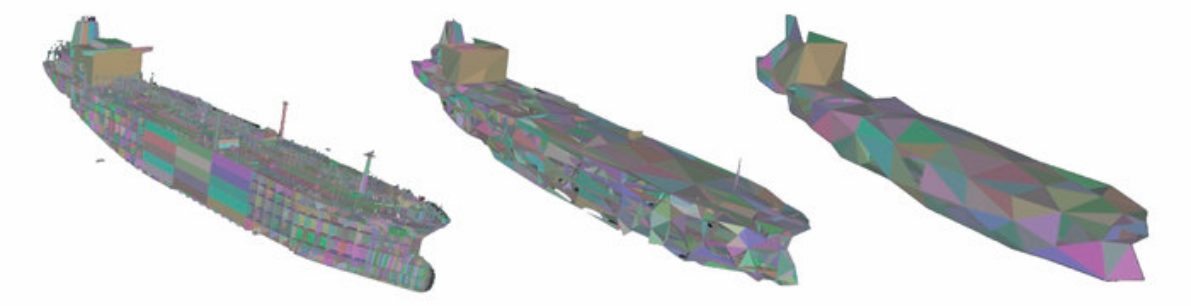

Figura 2.3 - O mesmo modelo simplificado usando HLODs. Figura extraída da publicação de Erikson et al. (2001)

Maciel & Shirley (1995) introduziu o conceito de impostores para substituir objetos ou clusters de objetos. Os impostores são imagens geradas para substituir parte da geometria do modelo. As imagens são geradas para ângulos pré-determinados de cada objeto ou cluster de objetos. Uma vantagem é que, como os impostores são gerados usando o próprio renderizador, superfícies ocultas são escondidas automaticamente, de forma que não afetam a aparência final da representação simplificada.

Rusinkiewicz & Levoy (2000) introduz um algoritmo para visualizar grandes modelos escaneados usando uma hierarquia de nuvens de pontos. Esses modelos são formados por 100 milhões a 1 bilhão de pontos amostrados.

Técnicas tradicionais de visualizar esse tipo de modelo envolvem criar uma triangulação em cima desses pontos amostrados e simplificá-la usando uma técnica de *view-dependend LOD*. Esse novo algoritmo de hierarquias de nuvens de pontos propõe que os pontos amostrados sejam renderizados usando pequenos pontos de pouco mais de um pixel, renderizandos usando uma primitiva de ponto do OpenGL. Quando visualizados a partir de uma certa distância, esses pontos começam a ser agrupados em um único ponto um pouco maior que represente as suas características combinadas. Esses pontos gerados por agrupamento são novamente agrupados para formar pontos maiores, que represente aquela parte do modelo quando for visualizado de longe (Figura 2.4).

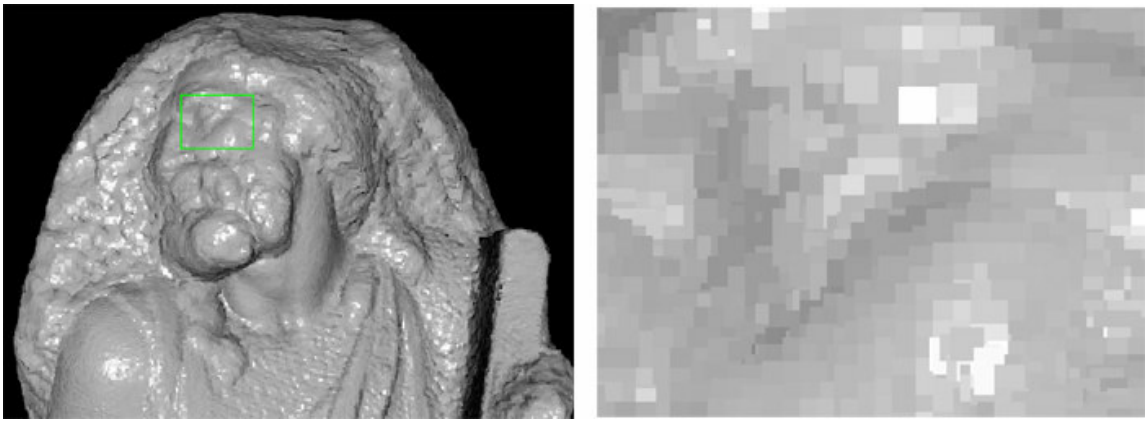

Figura 2.4 - Rusinkiewicz & Levoy (2000) renderizando um modelo escaneado usando uma hierarquia de pontos (esquerda). Quando vistos de perto (direita), podemos ver que os pontos são na verdade primitivas de pontos do OpenGL, que são desenhados como pequenos quadrados. Figura extraída da publicação de Rusinkiewicz & Levoy (2000).

Gobbetti & Marton (2005) apresentam o algoritmo de Voxels Distantes (*far voxels*) que também usa hierarquias de nuvens de pontos, mas voltado para a visualização de modelos CAD, conseguindo um desempenho interativo para modelos com 350 milhões de triângulos (Figura 2.5). Nesse trabalho as nuvens de pontos são chamadas de voxels, e são usadas para representar as versões simplificadas dos objetos da cena numa estrutura de HLODs. Os níveis mais detalhados, correspondentes às folhas da árvore de HLODs, ainda são representados da forma tradicional, usando a geometria original do modelo.

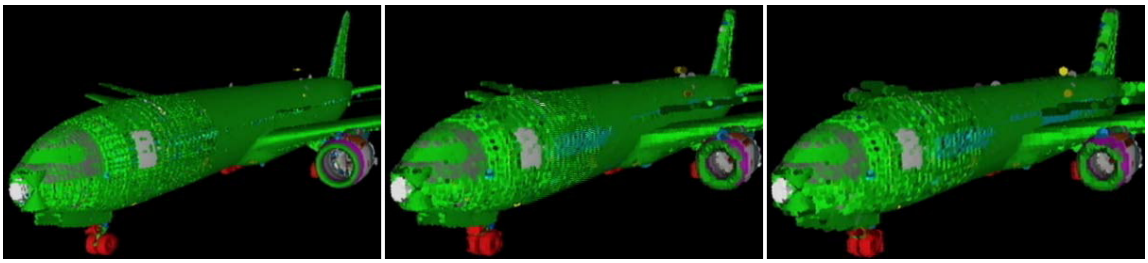

Figura 2.5 – Diferentes níveis de resolução usados para representar um modelo CAD de um Boeing 777 (350 milhões de triângulos) usando a técnica de Voxels Distantes. Figura extraída da publicação de Gobbetti & Marton (2005)

Simplificações baseadas em polígonos permitem um maior desempenho, ao custo de se sacrificar a aparência exata do modelo e qualidade final de imagem. Simplificações baseadas em pontos, por outro lado, permitem um controle maior da aparência de cada região do modelo simplificado, o que melhora a qualidade da imagem ao custo de exigir mais primitivas na representação.

Uma alternativa que recentemente começou a ganhar força para resolver o problema de visualização de modelos massivos é o uso de um algoritmo inteiramente baseado em traçado de raios, sem nenhuma aceleração em placa gráfica. Dietrich et al. (2003) propõe o OpenRT, uma API padronizada para aplicações de traçado de raios interativas. Essa API é orientada para ser simples, fácil de usar e difundida como o OpenGL, mas é estruturada de forma diferente. (Wald et al., 2005) propõe um visualizador que usa essa API para renderizar modelos massivos usando paginação, estratégias de carregamento sob demanda e um esquema de aproximações hierárquicas para representar geometrias que ainda não tenha sido carregadas.

Segundo Yoon et al. (2006), a velocidade atual dos processadores está avançando a ponto de conseguir renderizar modelos a taxas interativas usando traçado de raios e o atual problema a ser resolvido para permitir que essa técnica seja usada para modelos massivos é reduzir o tamanho do *working set* necessário. O *working set* é o conjunto de dados que precisam estar alocados na memória principal para permitir a renderização de um quadro. Nessa publicação, é proposto o uso de R-LODs para substituir grupos de triângulos inteiros nos nós internos da KD-Tree usada. Segundo a publicação, R-LODs são planos que contêm atributos de material, como as cores da geometria substituída. O ganho

de performance obtido, quando comparado a um sistema de traçado de raios sem R-LODs, varia de 2 a 20 vezes, com pouca perda de qualidade visual (Figura 2.6).

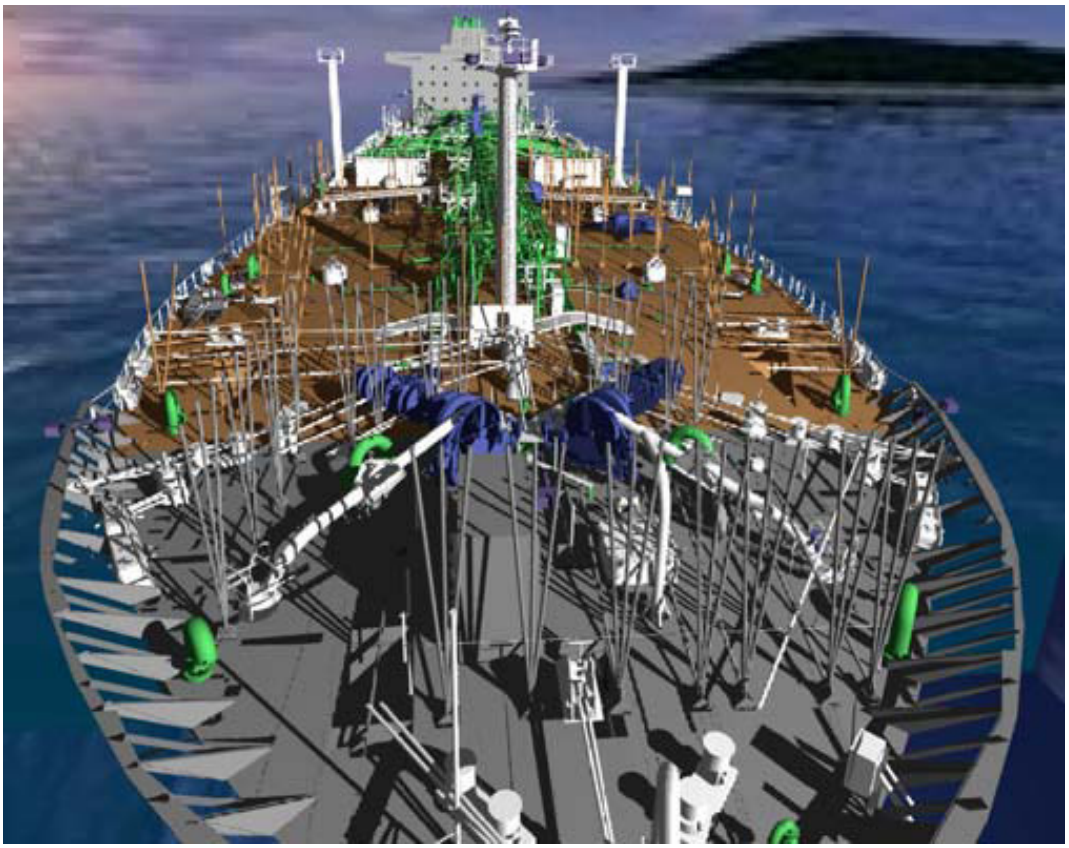

Figura 2.6 – Modelo de um navio com 77 milhões de triângulos renderizado com a técnica de R-LOD apresentada por Yoon et al. (2006)

Soluções baseadas em traçado de raios têm três grandes vantagens sobre outros métodos de renderização. A primeira vantagem é que a visibilidade de superfícies pode ser determinada em tempo logarítmico sobre o tamanho da cena, desde que o modelo seja estruturado de forma adequada. Outra vantagem é que o traçado de raios é altamente paralelizável, bastando que cada pixel a ser renderizado seja atribuído a um processador diferente. E por último, o cálculo de sombras e reflexões pode ser feitos com um custo não muito maior que o da própria renderização usando traçado de raios.

Frisken et al. (2000) sugere a adoção de *Adaptively Sampled Distance Fields*, ou ADF, como uma estrutura de dados fundamental para a Computação Gráfica. Segundo a publicação, sua estrutura é simples e direta e é extremamente efetiva na reconstrução de formas complexas com boa qualidade.

ADF representam superfícies ou volumes através de um campo de valores de distância amostrados adaptativamente em uma estrutura espacial (no caso, uma *octree*). Nesse campo, cada valor representa a distância do ponto em questão até o ponto mais próximo na superfície ou volume sendo representado na ADF. De posse desse campo, é possível usar um algoritmo de traçado de raios para gerar uma representação para a superfície ou volume em questão de forma eficiente.

Até o momento, existem poucas publicações que abordem o uso de ADF para modelos de engenharia. Duguet et al. (2006) apresenta um relatório de pesquisa não publicado, onde são propostas técnicas para se criar uma estrutura de ADF em *octree* para modelos massivos, técnicas de aceleração para a geração da ADF e formas automáticas de preencher buracos existentes nos modelos massivos usados.

Esta dissertação apresenta uma implementação da técnica de Voxels Distantes introduzida por Gobbetti & Marton (2005). Juntamente com o traçado de raios usando impostores apresentada por Yoon et al. (2006) e ADF, a técnica de Voxels Distantes representa uma das idéias mais atuais para tratar a renderização de modelos complexos com taxas interativas.

A escolha por essa técnica foi tomada porque a técnica de traçado de raios não consegue tirar vantagem das placas 3D existentes atualmente e ainda depende do uso de máquinas com diversos processadores para conseguir uma performance interativa. A técnica de ADF, por outro lado, é voltada para modelos com formas orgânicas, tendo dificuldades para lidar com objetos com arestas, que constituem a maioria dos objetos existentes em modelos de engenharia. Além disso, a técnica de ADF ainda não foi aplicada com eficiência em modelos massivos. Assim, acreditamos que a técnica de Voxels Distantes é atualmente a mais apropriada para a visualização dos modelos massivos de engenharia.