2
Trabalhos Relacionados

Neste capítulo é feita uma recapitulação dos trabalhos relacionados à área de visualização de grandes modelos. São apresentados os trabalhos ligados às técnicas convencionais de descarte de primitivas gráficas, que são úteis em todas as situações em que a linha de produção gráfica está saturada. Em seguida são apresentados os trabalhos nas duas áreas que tratam modelos massivos: visualização distribuída e visualização com níveis de detalhe e multiresolução.

2.1
Técnicas de Descarte de Primitivas Gráficas

Esta seção apresenta as técnicas de aceleração que visam descartar primitivas que não contribuem para a geração de uma imagem final.

Uma das técnicas de aceleração mais consolidadas na área de visualização é o descarte por volume de visão (*view frustum culling*) (4). Nessa técnica é calculada uma hierarquia de volumes envolventes para a malha dos objetos do modelo, permitindo o descarte de grupos de objetos fora do campo de visão do observador. Esse descarte é feito com testes simples de interseção entre os volumes e os planos do volume de visão. Assarsson et al. (5) apresentaram diversas otimizações para acelerar essa técnica para hierarquias de caixas alinhadas (*axis-aligned bounding boxes*) e caixas orientadas (*oriented bounding boxes*), tirando proveito da coerência entre quadros consecutivos e de propriedades de volumes de visão simétricos.

O descarte por volume de visão não é capaz de eliminar objetos que estão dentro do volume de visão mas que estão ocultos por outros objetos. Isso pode gerar muito redesenho desnecessário, causando no caso de primitivas dentro do volume de visão um desperdício de poder computacional não só na transformação de vértices como também no processamento de fragmentos (candidatos a *pixels*). A eliminação de objetos ocultos é tratada por algoritmos de descarte por oclusão (*occlusion culling*). Praticamente todas as placas gráficas modernas dão suporte a testes de oclusão, chamados *occlusion queries*, que possibilitam identificar, após o desenho de uma dada geometria, quantos *pixels* passaram no teste de profundidade. Embora o teste em si seja processado rapidamente pela placa gráfica, o seu resultado não fica disponível imediatamente devido à latência entre o pedido de teste e o seu processamento pelo *pipeline* gráfico. Torna-se necessário então tratar o atraso nessa resposta para

evitar travamentos na linha de produção.

Bittner et al. (7) apresentaram um algoritmo intitulado *Coherent Hierarchical Culling* (CHC), que tenta lidar com os problemas supracitados diminuindo o número de testes enviados e utilizando a coerência espacial e temporal de quadros consecutivos para agendar os testes, de forma a manter a linha de produção gráfica sempre em uso. Ivson e Celes (41) analisaram diferentes métodos para a construção da hierarquia de volumes envolventes e o seu impacto no algoritmo CHC. Foi feita também uma análise da qualidade do ajuste dos volumes envolventes experimentados. Mattausch et al. (36) apresentaram melhorias ao algoritmo CHC com o objetivo de reduzir ainda mais o número de testes de oclusão e minimizar o número de mudanças de estado na linha de produção gráfica.

Shirman e Abi-Ezzi (49) apresentaram uma técnica para o descarte rápido de grupos de primitivas com base nas suas orientações. Para isso, primitivas com normais parecidas são agrupadas e um *cone truncado de normais* é criado, contendo todos os vértices e todas as direções de seus vetores normais. O cone é definido por sua direção n, seu ângulo de abertura α, seu vértice a e distâncias ao vértice a ao longo de n que truncam o cone. Shirman e Abi-Ezzi provam que se o observador está dentro do cone chamado cone da frente (*front cone*), então todas as faces no cone estão voltadas para o observador. Similarmente, se o observador se encontra dentro do cone de trás (*back cone*), então todas as faces no cone estão voltadas para o lado oposto do observador (são *back-faces*). Com isso, dado um cone de normais, é possível detectar se ele só possui faces voltadas para trás com um teste simples, possibilitando, em caso de modelos fechados, que todas as faces contidas dentro do cone sejam descartadas de uma vez. De forma análoga, é possível detectar se todas as faces estão voltadas para o observador, sendo assim possível desabilitar o teste automático de faces voltadas para trás no estágio de rasterização do *pipeline* gráfico. Esse trabalho deixa em aberto a forma como se particiona as primitivas do modelo para a construção dos cones de normais.

2.2 Visualização Distribuída

Esta seção fará uma breve recapitulação dos trabalhos relacionados à área de renderização paralela e distribuída, com foco especial em sistemas que particionam o modelo entre os nós do agrupamento de PCs.

Molnar et al. (37) apresentaram uma classificação para estratégias de renderização paralela com base no estágio da linha de produção gráfica em que ocorre a ordenação de visibilidade. Três classes de arquiteturas foram

concebidas com base neste critério: arquiteturas com ordenação no início (*Sort-First*), no meio (*Sort-Middle*) e no fim (*Sort-Last*).

Na estratégia com ordenação no início a tela é dividida em um conjunto de ladrilhos disjuntos, e cada processador é responsável pela renderização (e portanto ordenação) de todas as primitivas que possuem algum vértice dentro da região determinada pelo ladrilho correspondente.

Sistemas com ordenação no meio distribuem primitivas após o estágio da geometria e antes do estágio da rasterização, portanto não se adaptando bem ao *hardware* gráfico moderno.

Já os sistemas com ordenação no fim dividem o modelo entre os nós de renderização, e cada nó executa o *pipeline* gráfico inteiro no seu submodelo correspondente. A ordenação das imagens parciais de cada nó resolve o problema de visibilidade, sendo o último estágio do *pipeline*, comumente executado em um nó mestre.

Os sistemas que utilizam ordenação no início e ordenação no meio devem replicar o modelo inteiro em cada nó de renderização, o que é problemático quando se trata de modelos massivos. Molnar et al. (37) elegeram os sistemas com ordenação no fim como sendo a estratégia com maior escalabilidade em termos de números de primitivas, sendo definitivamente a escolha certa quando se trata da renderização de modelos massivamente complexos.

Wylie et al. (59) construíram um sistema escalável com ordenação no fim utilizando agrupamento de PCs (*PC clusters*) para a renderização de modelos poligonais gigantes. Eles implementaram estratégias simples para a partição inicial dos dados para a renderização paralela, onde a maioria das estratégias teve como base o número de triângulos. Essas estratégias demonstraram um balanceamento de carga excelente em termos de processamento de geometria e um bom balanceamento de carga em termos de rasterização quando o número de triângulos é relativamente grande.

Humphreys et al. (30) apresentaram Chromium, o sucessor do WireGL (29), um sistema para a manipulação de sequências (*streams*) de comandos da API do OpenGL em um agrupamento de PCs. O sistema permite a distribuição de processamento gráfico utilizando ordenação no início ou ordenação no fim de forma completamente transparente, sem a necessidade de mudanças no código fonte da aplicação a ser distribuída. No entanto, acreditamos que otimizações dependentes da aplicação, apesar de menos genéricas e de aumentar o acoplamento da solução, são necessárias no contexto de conjuntos de dados massivamente grandes.

Van der Schaaf et al. (46) compararam o uso de arquiteturas em *modo imediato*, como o WireGL (29) (antecessor do Chromium), com arquiteturas

seguindo o paradigma de *modo retido*. Eles demonstraram que o modo imediato expõe o sistema a problemas sérios de escalabilidade, o que eles tentaram resolver utilizando o modo retido. Dois métodos foram experimentados: replicar os dados nos nós e transmitir comandos gráficos pela rede. O método com replicação de dados foi o melhor em termos de desempenho enquanto o método de transmissão de comandos simplificou a transparência da aplicação e facilitou o tratamento de entradas do usuário e da sincronização de dados.

Abraham et al. (2) apresentaram um sistema distribuído para renderização paralela em agrupamentos de PCs com ordenação no início. O trabalho apresenta um sistema em forma de linha de produção distribuída, que apresenta bom desempenho dado o uso de múltiplas linhas de execução e da paralelização entre operações de transferência de dados e renderização. O trabalho também apresenta um novo algoritmo de balanceamento de carga para sistemas com ordenação no início, utilizando apenas os tempos de renderização dos quadros anteriores para redimensionar os ladrilhos designados aos nós renderizadores.

Cavin et al. (13) implementaram uma linha de produção com ordenação no fim para a renderização de modelos volumétricos e poligonais gigantes utilizando agrupamentos de PCs com componentes de prateleira (*off-the-shelf*). Sua solução demonstra que agrupamentos de baixo custo, que dispensam redes caras e especializadas como Myrinet e Infiniband e *hardwares* de composição de imagem, podem ser competidoras viáveis de soluções mais caras. Em seguida, Cavin et al. (12) apresentaram uma análise teórica e prática do desempenho de implementações em linha de produção com ordenação no fim para ambos tipos de modelos, poligonais e volumétricos.

Marchesin et al. (35) apresentaram um sistema de visualização volumétrica com ordenação no fim que se baseou em um único PC gerenciando múltiplas GPUs. Os resultados demonstram como o sistema proposto compete bem com agrupamentos de PCs de médio porte onde cada nó é equipado com uma única GPU. Isso torna o uso de múltiplas placas gráficas um adicional definitivo para aumentar o poder de processamento de um único PC com baixo custo. Outros trabalhos anteriores em renderização utilizando múltiplas placas gráficas, como o SLI (39) e o Quadro Plex (38) da NVIDIA, utilizam ordenação no início para distribuir a carga entre as GPUs, o que não escala bem com o tamanho do modelo.

2.3
Visualização com Níveis de Detalhe e Multiresolução

Conforme mencionado anteriormente, modelos com geometria massivamente complexa possuem um grande número de primitivas que, quando pro-

jetadas para a tela, ocupam poucos *pixels* ou até menos que um *pixel*. Os trabalhos na área de visualização com níveis de detalhe e multi-resolução exploram esse fato para limitar a quantidade de dados a serem processados a cada quadro, o que é feito pela filtragem seletiva de detalhe geométrico. O cálculo de representações filtradas dos dados e sua extração eficiente em tempo de visualização possui vasta literatura, e numerosos métodos para o cálculo de uma boa aproximação de uma superfície, dada uma métrica de erro de aproximação, tem sido propostos. O cálculo da aproximação ótima é um problema conhecidamente NP-difícil (3), então a maior parte da pesquisa na área tem focado em métodos heurísticos (31).

Esta seção apresenta os trabalhos diretamente relacionados à area de simplificação de malhas de quadriláteros e hexaedros, cujo foco é a geração eficiente desse tipo de malha com um número reduzido de primitivas e com uma boa aproximação da malha original. Também serão apresentados os trabalhos relacionados à área de multi-resolução para renderização em tempo real, cujo objetivo é armazenar e permitir a extração de múltiplos níveis de detalhe de uma forma eficiente, tanto em termos de tempo de processamento quanto em termos da transferência de dados de armazenamento secundário para a memória principal.

2.3.1 Simplificação de Malhas

Os métodos de simplificação de malhas podem ser classificados em métodos com *estratégias globais*, que são aplicadas à malha inteira de uma vez, e com *estratégias locais*, que simplificam interativamente a malha pela repetida aplicação de um ou de um conjunto de operações locais. Estratégiais locais são bem mais comuns que estratégias globais, principalmente pela sua simplicidade, robustez e eficiência. A vasta maioria dos algoritmos simplifica a malha com sequências de contrações de arestas ou vértices. A sequência é decidida na maioria das vezes seguindo uma estratégia gulosa, que mantém uma lista ordenada de operações candidatas, selecionando a operação que causa o mínimo erro a cada passo.

Hoppe (26) introduziu a representação conhecida por *malhas progressivas* (*progressive meshes*) como o primeiro algoritmo dinâmico de simplificação de modelos poligonais. Uma malha progressiva consiste em uma malha base simples, criada a partir de uma sequência de operações de colapso de arestas e uma série de divisões de vértices (*vertex splits*). A divisão de vértices é a operação inversa de um colapso de arestas, substituindo um vértice por dois vértices conectados por uma aresta, criando um vértice e dois triângulos adicionais.

As operações de divisão de vértices correspondem às operações de colapso de arestas utilizadas para criar a malha base. Ao se aplicar todas as operações de divisão de vértices, é possível reconstruir a malha original exatamente. Além da nova representação, o trabalho descreve um algoritmo cuidadoso de simplificação, modelando explicitamente a complexidade da malha e a fidelidade à malha original como uma função de energia a ser minimizada. O algoritmo avalia as arestas a serem colapsadas com base nessa função de energia e as ordena utilizando uma fila de prioridades. A função de energia é então minimizada utilizando uma estratégia gulosa, e o processo termina quando restrições topológicas previnem novas simplificações. Os vértices e arestas remanescentes formam portanto a malha base, e a sequência de colapsos de arestas forma, em ordem reversa, uma hierarquia de divisões de vértices. O trabalho trata de diversos problemas práticos em computação gráfica: transição suave entre níveis de detalhe, transmissão progressiva, compressão de malhas e refinamento seletivo. Além disso, ele introduziu um esquema para lidar com atributos da superfície durante a simplificação, como cores, normais, coordenadas de textura e atributos discretos, como identificadores de material e textura. Posteriormente, Hoppe (27) estendeu o trabalho para permitir a simplificação dependente do observador (*view-dependent simplification*), considerando três critérios de refinamento: posição em relação ao volume de visão, simplificando agressivamente pedaços da representação fora do volume de visão; orientação, simplificando agressivamente pedaços voltados para o lado oposto do observador (*back-facing*) e erro de simplificação projetado, garantindo um erro inferior a um determinado número de *pixels*.

Garland et al. (24) apresentaram um algoritmo de simplificação de superfícies com base na contração iterativa de pares de vértices. O algoritmo mantém uma aproximação do erro cometido em cada operação de contração utilizando matrizes quádricas. Uma matriz quádrica é uma matriz simétrica 4x4 que pode ser usada para calcular, dada uma posição no espaço, a soma dos quadrados das distâncias a um conjunto de planos, nesse caso, os planos das faces da malha em questão. Associa-se então uma matriz quádrica a cada vértice da malha. Matrizes quádricas são aditivas, ou seja, dada uma contração $(v1, v2) \rightarrow \bar{v}$, é possível calcular uma matriz $\bar{Q} = Q_1 + Q_2$, que permite uma boa aproximação na avaliação do erro geométrico cometido em $\bar{v}$. É também simples calcular a posição para o vértice $\bar{v}$ que minimiza o erro cometido, derivável diretamente da matriz $\bar{Q}$. Pelo fato de acumularem a distância a um conjunto possivelmente grande de pontos e por serem matrizes simétricas, sendo armazenáveis em apenas 10 valores de ponto flutuante, essa métrica de erro é compacta, o que é uma propriedade importante quando

se lida com modelos massivos. O algoritmo de simplificação de malhas é um processo iterativo de otimização que se baseia em uma estratégia gulosa, onde contrai-se iterativamente os pares de vértices que causam o menor erro quádrico. Hoppe (28) estendeu essa métrica de erro para lidar com atributos de aparência, como cor e textura. Sua nova métrica se baseia em correspondência geométrica em 3D, permitindo uma interpretação geométrica intuitiva, menos requisitos de armazenamento e avaliação mais rápida quando comparada a outras métricas.

Wu e Kobbelt (58) propuseram um novo *framework* para a decimação de malhas com base na técnica de otimização probabilística de algoritmos de múltipla escolha (*Multiple-Choice algorithms*). A idéia é substituir a fila de prioridades comumente usada por algoritmos iterativos gulosos de simplificação, como por exemplo a utilizada no algoritmo proposto por Garland e Heckbert (24). Ao selecionar o melhor entre um pequeno número de operações locais selecionadas aleatoriamente, muitas operações pesadas são evitadas, como a construção da fila de prioridades na inicialização, o processamento da mudança geométrica de todas as entidades envolvidas em uma operação local e a subsequente atualização de suas posições na fila de prioridades. Essa simples modificação alcançou tempos de simplificação em média 2,5 vezes menores que a versão gulosa do algoritmo, produzindo malhas com praticamente a mesma qualidade.

Daniels et al. (21) propuseram um algoritmo para a simplificação de malhas de quadriláteros que trata o difícil problema de manter a conectividade dos quadriláteros ao longo da simplificação. O método consiste no uso de operações unitárias aplicadas à representação dual da malha, com o firme objetivo de melhorar a estrutura da malha e manter o *genus* topológico. A proposta inclui uma extensão da métrica de erro quádrico para malhas de quadriláteros, priorizando também colapsos que restaurem a valência dos vértices ao valor ideal (quatro vértices adjacentes) e colapsos que criem elementos quadrados.

Tarini et al. (54) apresentaram outro método incremental para a simplificação de malhas de quadriláteros, cuja função objetivo permite a geração progressiva de uma malha com quadriláteros convexos, com ângulos próximos a 90 graus e de lados iguais. O seu conjunto de operações locais é simples porém poderoso, sendo dividido em três conjuntos: operações de simplificação, operações de otimização e operações de limpeza da malha. O conjunto de operações do algoritmo somente utiliza elementos quadrilaterais, sem a necessidade de introduzir elementos triangulares temporários.

2.3.2 Estruturas de Multi-resolução

Pinheiro e Velho (43) propuseram um sistema de gerenciamento preditivo de memória que permite a visualização de objetos gráficos 2D com multi-resolução em tempo real. O sistema teve como base um modelo de memória virtual, onde um componente de carregamento de páginas tenta predizer e carregar dados antes deles serem acessados pela aplicação, além de decidir quais dados devem ser removidos da memória. O mecanismo de memória virtual possui diversos níveis, onde o nível seguinte é sempre mais rápido e possui menor capacidade de armazenamento: rede, disco, memória RAM e memória de vídeo. O trabalho apresenta um conjunto de regras para o carregamento de páginas, forçando que o nível de menor resolução esteja sempre carregado e que a qualidade da visualização aumente crescentemente enquanto as páginas vão sendo carregadas. O sistema é demonstrado com a visualização de panoramas em tempo real.

Erikson et al. (11) utilizaram uma hierarquia de níveis de detalhe para acelerar o desenho de modelos massivos de CAD. Eles representam o modelo geométrico utilizando um grafo de cena e calculam níveis de detalhe em cada nó folha do grafo. Em seguida, eles completam a representação hierárquica gerando o que chamaram de níveis de detalhe hierárquicos (*HLODs*): cada nó interno contém uma versão simplificada da geometria de todos os seus descendentes. O trabalho contempla o agrupamento de objetos distintos e é capaz de lidar com ambientes dinâmicos.

De uma forma geral, esse foi um dentre vários trabalhos que notou que, conforme as placas gráficas foram evoluindo, a razão entre o custo entre decidir se uma primitiva é ou não uma boa aproximação da malha original (26, 27) e o custo de desenhar a primitiva aumentou enormemente (25). Dessa forma, algoritmos tradicionais de visualização de níveis de detalhe dependentes do observador (*view-dependent LOD rendering*), que construíam uma triangulação analisando o erro geométrico no uso de cada triângulo, passaram a ter gargalo na CPU, desperdiçando o poder computacional das GPUs modernas. Essas GPUs possuem mecanismos para armazenar malhas em formatos otimizados em memória de vídeo, eliminando também o possível gargalo na transferência de dados da CPU para a GPU. Assim, a idéia de simplificar grupos maiores de triângulos em conjunto, que podem ser armazenados e renderizados de forma otimizada, e de fazer poucas decisões sobre o erro cometido em cada conjunto de triângulos foi utilizada em diversos trabalhos (42, 17, 18, 19, 47, 34, 10).

Cignoni et al. (18) apresentaram uma técnica chamada *Adaptive Tetra-Puzzles*, que utiliza uma hierarquia regular conforme de tetraedros para partici-

onar um modelo 3D espacialmente. Cada célula tetraédrica possui uma simplificação pré-computada do modelo original, que é armazenada em um formato otimizado. A representação do modelo é construída em uma simplificação da hierarquia de tetraedros de baixo para cima, simplificando em conjunto a malha contida em *diamantes*, que são os conjuntos de tetraedros que compartilham sua maior aresta. Esse algoritmo possui uma característica interessante que é a possibilidade de se simplificar os diamantes de um dado nível da hierarquia em paralelo. Dadas algumas restrições simples para a simplificação dentro de um diamante, todas as possíveis subdivisões seletivas conformes da hierarquia de tetraedros resultam em uma malha com pedaços que casam perfeitamente. Em tempo de visualização, dados os parâmetros da câmera e da tela, a hierarquia é varrida de cima para baixo, carregando e renderizando tetraedros com resolução aceitável, dada uma medição do erro de simplificação projetado na tela.

Gobbetti et al. (25) apresentaram um rico levantamento de estratégias para o tratamento de grandes modelos. Eles discutem formas eficientes de se lidar com dados de tamanha magnitude, incluindo particularmente o descarte por visibilidade, a simplificação de malhas, a organização eficiente dos dados em disco e a compressão de dados.