TÉCNICAS DE PROCESSAMENTO PARALELO PARA SISTEMAS GRÁFICOS DE ALTO DESEMPENHO

Manuel Lois Anido

Núcleo de Computacão Eletrônica da UFRJ
Cx. Postal 2324 - CEP 20.001 - Rio de Janeiro - Brasil
e-mail: ncd10121@ufrj.bitnet

Resumo

Este artigo descreve as principais técnicas de processamento paralelo empregadas na geração de imagens por computador e apresenta as arquiteturas de alguns sistemas de visualização de alto desempenho. Inicialmente este artigo analisa os problemas envolvidos na geração de imagens, de forma qualitativa e quantitativa. Em seguida é apresentada uma taxonomia de sistemas de geração de imagens, discutindo as principais vantagens e restrições de cada um.

Palavras chave: Computação Gráfica, Processamento Paralelo, Geração de Imagens, Arquiteturas Gráficas.

1. Introdução

Tem-se conseguido grande progresso no campo de geração de imagens por computador nas duas últimas décadas. A maioria dos avanços tem sido nos campos de síntese e de geração de imagens. O desenvolvimento destes campos tem tido lugar em pelo menos duas principais frentes. Uma direção de pesquisa tem sido melhorar o realismo das cenas mostradas, independente do tempo necessário para criar a imagem. A outra principal direção tem sido construir sistemas altamente interativos, que devem produzir imagens sucessivas dentro de um período fixo de tempo (usualmente 1/30 do segundo, ou menos), ou seja, que devem operar em tempo real.

Sistemas de visualização tridimensional (3D) tem sido utilizados em sua grande maioria em aplicações de simuladores de vôo. Entretanto, como resultado dos avanços da tecnologia VLSI (Very Large Scale Integration), outros campos de aplicação estão sendo desenvolvidos. Estes campos incluem: visualização científica interativa, películas de cinema, animação por computador, análise de movimento, simulação de tráfego, videogames, etc. Estas novas aplicações tem aumentado a demanda por sistemas capazes de produzir imagens de alta qualidade e resposta rápida aos comandos do usuário.

2. Conceituação do Problema

2.1. Descrição Formal dos Objetos

Imagens são produzidas por um conjunto de transformações aplicadas a uma representação 3D de modelos de objetos armazenada em disco. O processo de geração de imagens consiste na criação de uma imagem 2D, definida pela cor e luminancia dos pixels no espaço da imagem formada pelo tubo de raios catódicos. Modelos para GIC (Geração de Imagens por Computador) concentram-se na estrutura das cenas, na sua forma, e na relação espacial. Diferentes formas de representação podem ser usadas para o modelo tridimensional.

Um modelo na forma geométrica computacional é representado por um conjunto de objetos sólidos definidos por expressões matemáticas, que incluem superfícies bicúbicas paramétricas, B-splines e curvas de Bezier [1]. Entretanto, a descrição de superfícies em forma matemática conduz a uma enorme carga computacional. De forma a facililtar a construção de um modelo complexo, pode ser usado um esquema de Geometria de Construção de Sólidos (CSG), o qual explora a noção de *adicionar* e *subtrair* blocos sólidos simples. Esta representação tem muitas vantagens, particularmente durante a fase de construção, mas é rígida com relação às transformações geométricas necessárias para criar vistas do modelo, dentro das limitações de tempo real.

A técnica consagrada de representação do modelo, para atender às restrições das transformações em tempo real, consiste em definir o modelo indiretamente, através de conjuntos de faces que são representadas por conjuntos de arestas. Estas faces podem ter a forma de polígonos, usualmente triangulos ou quadriláteros. Esta representação é usualmente conhecida como B-rep [1].

Fractais são uma outra forma de modelamento de objetos e tem recebido muita atenção da comunidade científica nos últimos anos [2]. As imagens resultantes desta técnica são espetaculares, e muitos métodos de gerar fractais tem sido desenvolvidos. O termo *fractal* tem sido generalizado pela comunidade científica para incluir objetos fora da definição original de

Mandelbrot. Esta forma de modelamento também leva a uma elevada carga computacional, podendo requerer horas de processamento para gerar uma imagem.

2.2 Técnicas de Visualização

A. Método Projetivo com Pipeline Gráfica

A técnica de visualização mais comum é o *método projetivo*, no qual os componentes da cena são submetidos a transformações geométricas (para transladá-los ou mudar o ponto de vista) e em seguida são projetados na tela. Este é o método utilizado em sistemas gráficos que operam em tempo real, onde um certo grau de realismo é sacrificado em prol de interatividade.

Uma organização em pipeline é uma estrutura natural para geração de imagens porque cada estágio geralmente só depende dos resultados do estágio anterior. Além disso tal organização diminui a alta demanda de processamento, pois diversas tarefas podem ser executadas simultaneamente, embora não necessáriamente na mesma imagem. A figura 1 ilustra um pipeline típico para geração de imagens que emprega primitivas (linhas e polígonos) e técnicas de sombreamento convencionais. Esta subdivisão do pipeline é lógica e não física, pois alguns estágios do pipeline podem ser implementados tanto em software como em hardware.

B. Ray-Casting e Ray-Tracing

Ray-Casting e Ray-Tracing [3] são técnicas que consistem em determinar a visibilidade de superfícies, através da emissão de raios imaginários de luz, partindo do olho do observador até os objetos da cena. A interseção dos raios com a tela especifica os pontos de formação da imagem. O método Ray-Tracing é usualmente uma extensão do algoritmo de ray-casting, para incluir reflexões, translucencia, sombras e refrações e apresenta bons resultados para luz especular. A cor de cada pixel é determinada pela contribuiçao de todos os raios.

Estes métodos propicíam imagens altamente realísticas, porém o número de operações envolvidas é extremamente elevado, fazendo com que as imagens levem horas para serem geradas nos computadores atuais. Uma tela de 1024x1024 pixels requer mais de um milhão de operações de cálculo de interseções raio-objetos, que por si só são operações sofisticadas. A carga de processamento cresce quadráticamente com o número de pixels em uma das laterais do CRT ($O(N^2)$). Por outro lado, o método é favorável à utilização de processamento altamente paralelo, permitindo diversos níveis de paralelismo.

C. Radiosidade

A técnica de radiosidade surgiu a partir dos modelos de transferência de calor e, ao contrário de ray-tracing, apresenta bons resultados para luz difusa, embora demande elevada capacidade de processamento. Diferentemente de Ray-Tracing, a carga computacional cresce quadráticamente ($O(N^2)$) com o número de objetos, ao invés do número de pixels por objeto. As superfícies podem ser refletoras ou emissoras de energia. O objetivo do método é calcular o balanço de energia do sistema.

2.3 A Imagem e a Memória de Vídeo (Frame Buffer)

Uma imagem pode ser imaginada como uma matriz tridimensional, onde linhas e colunas correspondem aos eixos x e y da tela e a terceira dimensão (z) corresponde às características de cada pixel, ou seja, R,G,B, profundidade, intensidade, etc. Os eixos x e y são normalmente mapeados numa memória de vídeo, enquanto o eixo z é normalmente mapeado em diversos planos de memória.

Nos sistemas que utilizam uma memória para armazenar a imagem mostrada ("frame buffer"), são normalmente empregadas memórias de vídeo ("video RAMs" - VRAM), projetadas especificamente para permitir que a saída de vídeo seja independente de outras operações no frame buffer. Um chip VRAM é similar a um chip DRAM convencional, mas contém um registro de deslocamento com entrada paralela e saída serial, conectado a uma segunda porta de dados, por onde saem os bits em série que irão endereçar a tabela de cores.

2.4 Principais Barreiras ao Desempenho

Sistemas gráficos podem ser construídos com os mais diversos desempenhos. No entanto, para projetar arquiteturas gráficas de alto desempenho, exisem três barreiras importantes que devem ser ultrapassadas, a saber:

a. *Processamento Geométrico:* Acelerar transformações, recorte e perspectiva, além dos limites possíveis com um só processador de ponto flutuante (ou ponto fixo).

b. *Processamento Inteiro de Pixels*: Acelerar o preenchimento de polígonos e o processamento de pixels, além da taxa obtenível com um simples processador gráfico.

c. *Banda Passande do Frame-Buffer*: Prover leituras e escritas no frame-buffer mais rápidas

do que as suportadas por um sistema de memória convencional.

2.5 Demanda Computacional

Consideremos uma cena sofisticada composta de 10.000 polígonos e uma taxa de atualização de 20 cenas por segundo (a nave espacial ENTERPRISE do filme Guerra nas Estrelas foi formada com 450.000 polígonos !!).

O número total de instruções de ponto flutuante necessárias para realizar as operações geométricas é de aproximadamente 40 milhões [1]. Nas operações de rastreio e acesso ao "frame buffer", é necessário realizar 65 milhões de somas e 102 milhões de acessos ao "frame-buffer", por segundo [1].

Da estimativa de carga computacional acima é possível concluir que é mandatório o uso de processamento paralelo e que é necessária uma organização adequada do frame-buffer para atender tal formidável demanda computacional.

3. Arquiteturas Para Computação Gráfica Usando Processamento Paralelo

3.1. Introdução

Sistemas gráficos de alto desempenho geralmente possuem arquiteturas com múltiplos processadores, grupados em três grandes blocos funcionais, que são: sistema de acesso ao banco de dados e de percurso da lista de polígonos, sistema de operações geométricas, e sistema de rasterização. É usual dizer que o sistema de operações geométricas constitui o 'front-end' do sistema gráfico e que o sistema de rasterização constitui o 'back-end'. Em sistemas que realizam a simulação do comportamento de aparelhos complexos, como aviões (simuladores de vôo) ou automóveis, normalmente é utilizado um processador adicional dedicado a tal tarefa, o qual transmite informações ao resto do sistema gráfico.

3.2. Estratégias Típicas de Paralelização em Computação Gráfica

As abordagens típicas para paralelizar são:

- *Paralelização no Espaço dos Objetos:*
  - Objetos são processados em paralelo;
  - Pixels sao gerados em sequência;
  - O aumento do número de objetos implica no aumento do número de processadore
  - O aumento do número de pixels implica em processadores mais rápidos;
  - No caso extremo tem-se 1 processador por objeto;
- *Paralelização no espaço da Imagem:*
  - Objetos são processados em sequência;
  - Pixels são processados em paralelo;
  - O aumento do número de objetos implica em processadores mais rápidos
  - O aumento do número de pixels implica no aumento do número de processadores;
  - No caso extremo tem-se 1 processador por pixel;
- *Paralelização Funcional*
  - Objetos são processados em paralel ou série;
  - Processadores alocados por tarefa;
  - Pipeline gráfica;

4. Arquiteturas para Operações Geométricas com Múltiplos Processadores

Num sistema gráfico que transforma objetos do espaço tridimensional para o espaço da tela, uma possível sequência de operações para alcançar as transformações necessárias é:

- Remoção de faces de costas para o observador;
- Translação, Rotação, e Escalamento;
- Recorte;
- Projeção Perspectiva;
- Iluminação dos Polígonos;

4.1. Organização em Pipeline (ou Paralelismo Horizontal)

Um dos mais conhecidos pipelines gráficos para operações geométricas foi introduzido por Clark [4], denominado *geometry engine*, o qual é ilustrado na figura 2. O *geometry engine* original consiste de um pipeline de 12 circuitos integrados microprogramados identicos que podem realizar funções distintas, de acordo com sua posição no pipeline. Uma versão mais recente deste pipeline (com 5 circuitos integrados) é utilizada na última linha de sistemas gráficos da Silicon Graphics.

Este tipo de solução com circuitos integrados especializados microprogramados é eficiente, mas não apresenta flexibilidade suficiente para permitir a inclusão de funções adicionais. Um outro cuidado adicional que deve ser tomado neste tipo de organização é fazer com que haja um balanceamento de carga entre os diversos estágios do pipeline.

Um outro exemplo empregando pipeline é descrito por Atkin e Ghee [5]. Neste sistema, é empregado um pipeline de transputers T800 para realizar as operações geométricas, conforme ilustrado na figura 6. O desempenho mencionado no artigo é baixo (quando

comparado a sistemas comerciais) em virtude da utilização dos canais seriais para passar informação de um estágio do pipeline para outro e também em função de ser um sistema simples de teste. Entretanto, o artigo serve para mostrar que modificações radicais devem ser feitas na arquitetura de pipeline de transputers para conseguir alto desempenho em operações gráficas. Por outro lado, empregando uma metodologia Ray-Tracing, ao invés de malhas de polígonos, é possível conseguir alto desempenho (quando comparado cou outras arquiteturas) empregando uma organização adequada de transputers. Isto se deve ao fácil mapeamento do problema numa rede de transputers, onde o tempo de processamento é muito maior que o tempo de comunicação.

Vale lembrar que uma solução que empregue simplesmente pipeline no 'front-end' tem limitações para obter elevado desempenho, pois o processador alocado para percorrer a lista de polígonos ('display list') provavelmente não conseguirá suprir o resto do pipeline. Portanto uma solução paralela poderá ser necessária.

4.2. Organização em Paralelo (ou Paralelismo Vertical)

A utilização de imagens descritas por malhas de polígonos permite repartir a carga de trabalho entre diversos processadores operando concorrentemente, pois em geral não há interdependencia nas operações geométricas entre as diversas primitivas gráficas. Tendo por base este fato, diversas arquiteturas paralelas tem sido propostas e utilizadas comercialmente.

O sistema visual IMAGE IV do simulador de vôo da Singer Link-Miles utiliza diversos processadores do tipo DSP (Digital Signal Processor) operando em paralelo, organizados conforme ilustra a figura 6. Os canais DMA controlam a transferencia dos dados de entrada e de saída dos DSPs. O uso deste tipo de processadores nesta aplicação deve-se ao fato de os mesmos possuírem operações aritméticas rápidas, que é um requisito fundamental para acelerar as operações geométricas.

Foi desenvolvido, na Universidade de Southampton, (UK) um sistema visual de baixo custo que utiliza até 30 microprocessadores MC68000-12MHz operando em paralelo, com uma organização semelhante à do sistema IMAGE IV [6]. As principais diferenças entre este sistema e o sistema IMAGE IV estão no fato de o sistema de Southampton também realizar o preenchimento de polígonos, além transmitir os dados de saída numa forma codificada e fácil de ser interpretada pelo controlador da memória de video para preenchimento da memória.

Com o intuito de aumentar o desempenho do sistema gráfico de Southampton em operações geométricas e também reduzir o número de processadores necessários, foi desenvolvida uma nova arquitetura utilizando processadores RISC especialmente projetados para esta aplicação [7,8]. Diversos processadores RISC são organizados em paralelo (em grupos de 4), conforme ilustrado na figura 7. Cada processador RIG é capaz de operar a 30 MIPS (pico), possuindo buffers de entrada e saída embutidos na área de registros gerais do sistema, para acelerar a transferência de dados de E/S. Canais DMA de entrada e saída coordenam a transferência de dados do sistema. A transferência de dados é feita por acesso direto à memória pois o tempo de processamento é da mesma ordem de grandeza do tempo de transferência. O sistema é organizado de tal forma que o balanceamento de carga pode ser fàcilmente distribuído. A ordem dos pacotes de dados de entrada é monitorada e passada ao sistema de rasterização de polígonos que irá recompo-la, caso necessário.

Torborg descreve uma arquitetura para operações geométricas utilizando processadores de 32 bits microprogramados operando em paralelo [9]. Cada processador opera a 12.5 MIPs e 25 MFLOPs. Esta arquitetura foi especialmente projetada para executar o conjunto de comandos do standard para gráficos 3D - PHIGS+. A organização geral desta arquitetura assemelha-se às outras arquiteturas paralelas descritas nesta seção, com um barramento de entrada de dados 3D e comandos e um barramento de saída para dados 2D e comandos para o controlador da memória de tela.

4.3. Organização com Pipelines em Paralelo (Paralelismo H & V)

Grimdale e seu grupo [10] apresentaram uma proposta de arquitetura que consiste de circuitos integrados funcionalmente semelhantes ao *geometry engine*, organizados em diversos pipelines em paralelo, visando obter maior desempenho nas operações geométricas. Um pipeline completo é composto de 16 circuitos integrados, e o objetivo do grupo de Sussex (UK) é conseguir um desempenho superior a 10.000 polígonos em tempo real.

5. Arquiteturas Para Geração da Imagem com Múltiplos Processadores

O subsistema de rasterização ('back-end') cria a imagem final através da conversão do conjunto de primitivas passadas pelo subsistema de operações geométricas. As principais operações necessárias neste estágio são:

- Determinar a visibilidade de cada pixel de uma primitiva (polígono ou objeto);
- Determinar o sombreamento de cada pixel;
- Acrescentar funções especiais como por exemplo texturas;

O artigo de Gharachorloo, Gupta, Sproull e Sutherland [11] apresenta uma excelente caracterização das principais técnicas de rasterização.

5.1. Arquiteturas com Pipeline para Geração de Imagens

5.1.1. Pipeline por Ordem de Objetos

Esta forma de organização é um método direto de prover concorrencia aos cálculos de rasterização, distribuindo os vários passos de um algoritmo de software, num pipeline de hardware. Esta técnica pode ser usada com os dois principais métodos de rasterização: ordem de objetos (algoritmos z-buffer, depth-sort, e BSP-tree) e ordem de imagem (algoritmos de linhas de vídeo - 'scan-line'). Vale notar que na técnica por ordem de objeto, um determinado objeto é todo rasterizado antes de passar para o próximo objeto, mesmo que o próximo objeto seja interceptado por linhas de vídeo comuns a ambos.

Os principais blocos deste pipeline, que geralmente tem um processador dedicado, executam:

1. Processamento de Polígono - Determinar a linha de vídeo inicial interceptada pelo polígono e calcular a inclinação dos lados (ou calcular os valores *Delta* para x,z,R,G,B para cada lado).

2. Processamento de Lados - As linhas de vídeo dentro de cada primitiva são processadas uma por uma. Os valores *Delta* calculados no estágio anterior são usados para calcular x,z,R,G,B, na interseção dos lados com a linha de vídeo. Os valores *Delta* para incrementar z,R,G, e B de pixel para pixel dentro de um span (sequência contígua de pixels) são então calculados.

3. Processamento do Span - Para cada span, usando o *Delta* calculado anteriormente, calcular os valores x,z,R,G,B para os pixels dentro do .span. O sombreamento Gourad [12] também pode ser feito neste bloco do pipeline. O valor da profundidade z (que indica se o pixel será ocultado) é comparado com o z anterior armazenado naquela posição e caso seja menor (novo mais próximo do observador), o novo pixel substitui o velho.

Este algoritmo só é valido para polígonos convexos. O tempo de processamento do método por ordem de objeto tende a crescer com o número de objetos na cena, sejam eles visíveis ou não.

5.1.2. Pipeline por Ordem de Imagem

Neste tipo de método, a imagem é calculada linha de vídeo por linha de vídeo (ou pixel por pixel), ao invés de primitiva por primitiva. Para evitar primitivas que não foram transformadas para o espaço da tela, primitivas são ordenadas em caixas, de acordo com a primeira linha de vídeo em que aparecem. Os principais blocos do pipeline por ordem de imagem (que geralmente possuem um processador dedicado) são:

a. Ordenador em Y - Coloca cada lado do polígono na caixa correspondente à linha de vídeo em que ele aparece primeiro.

b. Ordenador de segmentos ativos - Le lados das caixas mantendo uma tabela de lados ativos para a linha de vídeo corrente. Desta tabela ele constrói uma lista de segmentos ativos (span dentro de um polígono), o qualé ordenado pelo valor *x* do extremo esquerdo de cada segmento.

c. Gerador de Spans Visíveis - Percorre a lista de segmentos ativos, comparando os valores *z* onde necessário, e dá saída da sequência de spans visíveis na linha de vídeo corrente.

d. Sombreador - Realiza o sombreamento Gourad ou Phong, calculando a intensidade nos pixels do span.

Deve-se notar que não é necessária memória de vídeo neste tipo de sistema, desde que o sistema gráfico consiga gerar pixels à taxa do vídeo. Isto é muito difícil atualmente devido ao grande número de objetos e ao número de pixels dos sistemas atuais, que demandam uma enorme banda passante do vídeo. No entanto, este método foi a base de muitos sistemas gráficos dos simuladores de vôo de Evans & Sutherland nos anos 70.

Este tipo de método é limitado pela linha de vídeo mais ocupada (com maior quantidade de spans), o que limita a complexidade da cena. O tempo de processamento tende a crescer com a complexidade das partes visíveis da imagem, ao invés do número de objetos.

5.2. Arquiteturas Paralelas para Geração de Imagens

Paralelismo de imagem é atrativo porque os pixels da tela podem ser gerados em paralelo de diversas formas. As formas mais comuns de subdividir a tela são: em filas, em colunas, e entrelaçada. Um sistema que gera a imagem em paralelo, rasteriza em ordem de objetos e portanto uma memória de vídeo (frame buffer) é necessária para armazenar os resultados intermediários.

5.2.1. Arquiteturas com Memória Particionada

A idéia básica desta forma de ataque do problema é associar um processador a cada partição da memória de vídeo. Esta organização permite computar regiões em paralelo, aumentando o desempenho do sistema gráfico. A banda passante da memória de vídeo pode ser aumentada, provendo um canal de acesso separado à partição para cada processador. As duas formas mais comuns de subdividir a memória de vídeo são: partição

contígua e partição entrelaçada, as quais são ilustradas na figura 3.

A principal vantagem da partição contígua é que as primitivas só tem que ser processadas nas regiões em que podem ser visíveis, e estas regiões podem ser determinadas rapidamente. Como principal desvantagem está o fato de ser mais susceptível a gargalos, os quais podem ocorrer caso todas as primitivas caiam na região de um processador.

A partição entrelaçada foi proposta inicialmente por Fuchs [13] e permite atingir uma melhor divisão da carga de trabalho, pois quase todos os polígonos (exceto os minúsculos) caem em todas as partições da memória de vídeo. Cada processador lida com todas as primitivas, embora só uma fração dos pixels. O mais importante é que o pior caso desta organização depende do numero total de primitivas e não da região mais densa. Esta é a técnica mais utilizada atualmente.

Uma melhoria da partição entrelaçada foi proposta por Clark e Hannah [4], para tomar vantagem de cálculos comuns a múltiplos processadores. Este método é utilizado nos sistemas mais recentes da Silicon Graphics, cuja arquitetura é ilustrada na figura 8.

5.2.2. Memória de Vídeo com Lógica

Neste tipo de método, processamento é incorporado a partições da memória de vídeo, fornecendo novos modos de endereçamento dos pixels, bem como a habilidade de endereçar um retangulo inteiro de pixels da memória de vídeo, em paralelo. Um exemplo desta organização é o *8x8 Display* desenvolvido por Gupta, Sproul e Sutherland.

Henry Fuchs propos e desenvolveu uma arquitetura denominada *Pixel Planes 4* que explora o paralelismo de imagem ao extremo [13]. Nesta arquitetura, um processador muito simples (1 bit) e uma pequena memória (72 bits) são incorporados a cada pixel, formando uma memória de vídeo inteligente. O processador executa a equação básica Ax+By+C, que é a equação de uma reta e reflete os lados dos polígonos. Cada expressão é avaliada em paralelo em todos os pixels da tela, para saber se cada pixel está fora ou dentro do polígono. A figura 9 ilustra a organização básica da arquitetura *Pixel Planes 4*.

5.3. Rasterização por Objeto em Paralelo

Numa arquitetura de rasterização por objeto em paralelo, múltiplas primitivas são processadas em paralelo, de forma que os pixels podem ser gerados mais rápido. O método usual é associar primitivas a um certo número de processadores (idealmente um processador por objeto), cada qual sendo capaz de gerar a imagem completa contendo suas primitivas. Durante a rasterização cada processador de objeto enumera os pixels da tela em ordem de linha de vídeo (em geral), gerando z, R,G e B. As cadeias de pixels de cada processador de objeto tem que ser combinadas para produzir uma única cadeia de pixels para o estágio final. Um dos primeiros sistemas a utilizar este método foi o simulador de vôo General Electric NASA II.

5.3.1. Pipeline de Processador por Primitiva

Um modo simples de combinar as multiplas cadeias de cores de pixels e de valores z, produzidas por múltiplos processadores de objetos é organizar os processadores em pipeline, de forma que uma cadeia de valores de cor e de profundidade z passe através deles (ver figura 4). Cada processador gera valores de cor de de z para sua primitiva no pixel corrente, e compara seu valor z com o valor z do processador anterior no pipeline. Se o valor z de entrada é menor (polígono mais próximo), os valores de cor e z são passados adiante. Se z for maior que o z de entrada, os valores da primitiva são enviados para o estágio seguinte e os valores do estágio anterior são inibidos.

Uma das primeiras arquiteturas projetadas utilizando este princípio foi apresentada por Cohen e Demetrescu. As principais desvantagens deste tipo de arquitetura são o elevado número de processadores necessários num sistema com muitos polígonos e o fato de não utilizar memória de vídeo, a qual é essencial em certas aplicações, como por exemplo a adição de texturas baseadas em mapas. Uma outra desvantagem é o tempo de latencia do pipeline para gerar a imagem. A figura 11 ilustra este tipo de arquitetura.

5.3.2. Árvore de Processador por Primitiva

Este tipo de organização é uma alternativa à organização em pipeline e consiste em usar uma árvore binária com blocos de composição para misturar as cadeias de cor e de coordenadas z. Este método permite que a rasterização nos processadores de objetos seja verdadeiramente síncrona (não há atraso para computar o mesmo pixel), e reduz a latencia ocasionada por um pipeline longo.
A desvantagem é que é necessária lógica especial para realizar a composição.

A primeira arquitetura em árvore com um processador por primitiva foi apresentada por Fussel. Kedem e Ellis [14] desenvolveram a arquitetura denominada Ray Casting Machine, que rasteriza diretamente imagens a partir de uma representação em árvore de Geometria de Construção de Sólidos (CSG).

5.4. Rasterização Híbrida Paralela

Quando as filosofias de paralelismo de objeto e paralelismo de imagem são levadas ao extremo, resultam sistemas pouco eficientes. Isto pode ser

melhorado usando uma combinação das duas técnicas e produzindo sistemas híbridos paralelos, que embora sejam mais complexos que os outros, são mais eficazes.

5.4.1. Buffers Virtuais e Processadores Virtuais

A principal desvantagem das arquiteturas altamente paralelas por objeto e por imagem está na baixa utilização dos elementos processadores (PEs). Uma maneira de aumentar a utilização é construir sòmente uma fração do hardware, mas alocar estes recursos dinamicamente na tela, conforme necessário. Existem duas variantes desta técnica: *buffers virtuais* (para sistemas paralelos por imagem) e *processadores virtuais* (para sistemas paralelos por objetos). De forma similar à memória virtual, ambos os métodos tentam aumentar os recursos físicos aparentes de um sistema, através do remanejamento dinamico de recursos. Um exemplo desta arquitetura é o sistema SAGE (Sistolic Array Graphics Engine).

5.4.2. Arquitetura Paralela com Buffer Virtual

Existe um outro nível de paralelismo em sistemas com Buffer Virtual que usam áreas retangulares e ordenação em caixas completa. Um exemplo deste tipo de arquitetura é o sistema Pixel Planes 5. Similarmente ao sistema Pixel Planes 4, este sistema também possui uma memória de vídeo inteligente, com um processador que permite calcular expressões quadráticas do tipo $Ax+By+Cx+Dx^2+Exy+Fy^2$, que são úteis para desenhar superfífies curvas e o modelo de iluminação de radiosidade esférica.

5.5. Arquiteturas para Ray-Tracing

Conforme já descrito, ray-tracing é um método de visualização poderoso que pode gerar figuras altamente realistas. Infelizmente acarreta uma carga computacional muito grande fazendo com que imagens levem minutos ou horas para serem geradas. Felizmente, algoritmos de ray-tracing podem ser paralelizados de diversos modos e em diversos níveis.

A maioria das arquiteturas para ray-tracing utilizam um grande número de elementos de processamento, numa organização MIMD. Como muitas arquiteturas para ray-tracing assemelham-se a computadores paralelos comerciais, os sistemas mais empregados em geração de imagens ray-tracing são sistemas comerciais paralelos. Entre esses sistemas podemos destacar redes de Transputers (IBM VICTOR, PARSYS SuperNode, MEIKO Computing Surface) , Connection Machine [DELA88], AT&T Pixel Machine [POTM89], iPSC [Intel], e outros mais.

6. Conclusões

Este artigo tentou resumir as principais técnicas e arquiteturas empregadas em sistemas gráficos de alto desempenho. Do exposto ficou patente que a demanda computacional é tão elevada que o uso de pipelining e/ou processamento paralelo são mandatórios para alcançar alto desempenho. Foi também mostrado que é necessária uma elevada banda passante entre a memória de vídeo e a tela, principalemte em monitores de alta resolução.

O aumento do desempenho traduz-se para o usuário em imagens mais realistas (como alto nível de detalhes) e alta interatividade, ou seja, o usuário interage com o sistema gráfico e recebe respostas aos seus comandos em tempo real, tal qual a ação de dirigir um automóvel.

Foi mostrado que ainda há muitas barreiras a serem ultrapassadas, para gerar imagens altamente realistas de forma economica, em tempo real. No entanto, os resultados obtidos nos últimos 20 anos tem sido espetaculares.

7. Referencias

[1] Foley, J.D., Van Dam, A., Feiner, S. and Hughes, J., *Computer Graphics - Principles and Practice*, Addison Wesley, Reading, Mass., 1990.

[2] Mandelbrot, B.B., *The Fractal Geometry of Nature*, Freeman, San Francisco, 1982.

[3] Roth, S. D., *Ray Casting for Modelling Solids*, Computer Graphics and Image Processing, Vol. 18, No. 2, pp. 109-144, February 1982.

[4] Clark, J.H., *Structuring a VLSI System Architecture*, Lambda, Vol.1 No.2, pp.25-30, Second Quarter, 1980.

[5] Atkin, P. and Ghee, S., *A Transputer Based Multi-User Flight Simulator*, Technical note 36, Inmos Ltd., Bristol, UK, 1988.

[6] Allerton, D.J and Zaluska E. J, *A Multi-processor Approach to Image Generation*, IEE Int. Conf. on Simulators, Warwick, England, pp. 226-231, 1986.

[7] Anido, M.L. and Allerton, D.J., *RISC Design for Computer Image Generation*, Microprocessors and Microsystems Journal, Butterworth-Heinemann Publ., Vol. 14, No.6. pp. 341-350, August 1990.

[8] Anido, M.L., *Design and Implementation of a RISC Processor for Computer Image Generation Geometric Computations*, Ph.D. Thesis, Dept. of Electronics & CS, University of Southampton, UK, 1991.

[9] Torborg, J.G., *A Parallel Processor Architecture for Graphics Arithmetic Operations*, SIGGRAPH, vol. 21,

No.4, pp. 197-204, July 1987.

[10] Grimsdale, R.L., *Techniques for Real-Time Image Generation*, Proc. Int. Conf. on Parallel Processing for Computer Vision and Display, Leeds, UK, January, 1988.

[11] Gharachorloo, N, Gupta, S., Sproull R.F. and Sutherland, I.E., *A Characterization of Ten Rasterization Techniques*, SIGGRAPH 89, Boston, August, pp.355-367, 1989.

[12] Gourad, H., *Continuous Shading of Curved Surfaces*,
IEEE Transactions on Computers, Vol.20/6, pp.623-628, June, 1971.

[13] Fuchs, H. and Poulton, J., *Pixel-Planes: a VLSI-Oriented Design for a Raster Graphics Engine*, VLSI Design, Third Quarter, pp. 20-28, 1981.

[14] Kedem, G. and Ellis, J.L., *The Raycasting Machine*, in Proceedings of the 1984 Int. Conf. on Computer Design, pp. 533-538, October 1984.

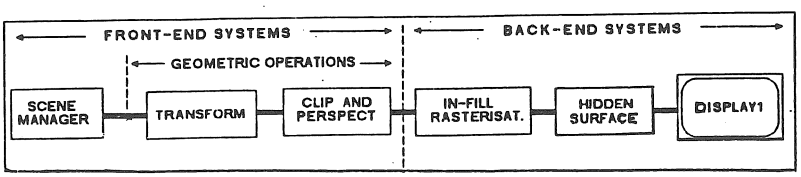

Figura 1 - Pipeline tipica para geração de imagens

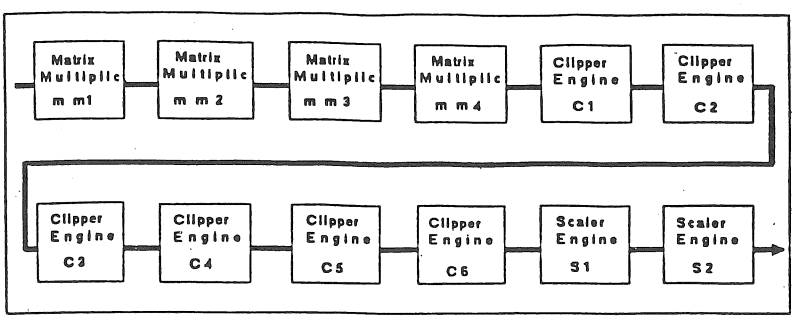

Figura 2 - Elementos do Pipeline do Geometry Engine

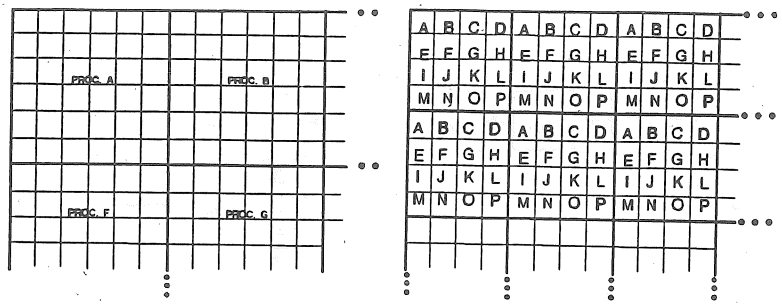

Figura 3 - Partição Contígua e partição entrelaçada da Memória de vídeo

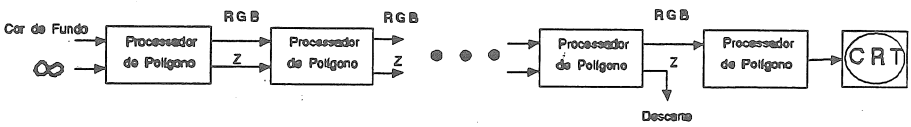

Figura 4 - Arquitetura de Cohen e Demetrescu - pipeline de processador por primitiva

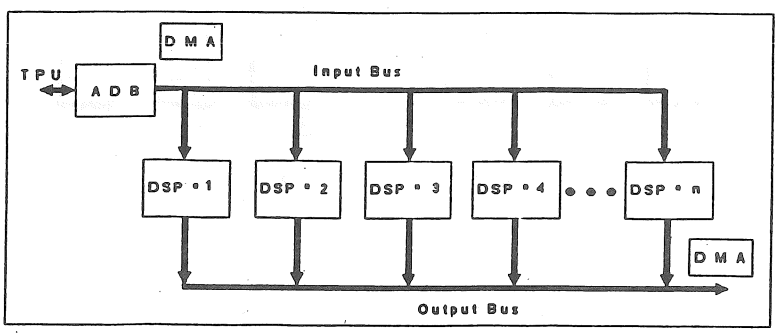

Figura 5 - Paralelismo de DSPs no simulador de Vôo da Singer-Link Miles.

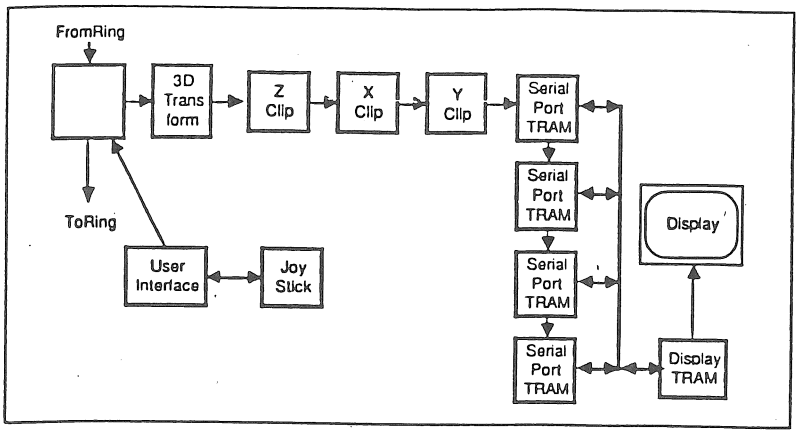

Figura 6 - Sistema Gráfico com Pipeline de Transputers

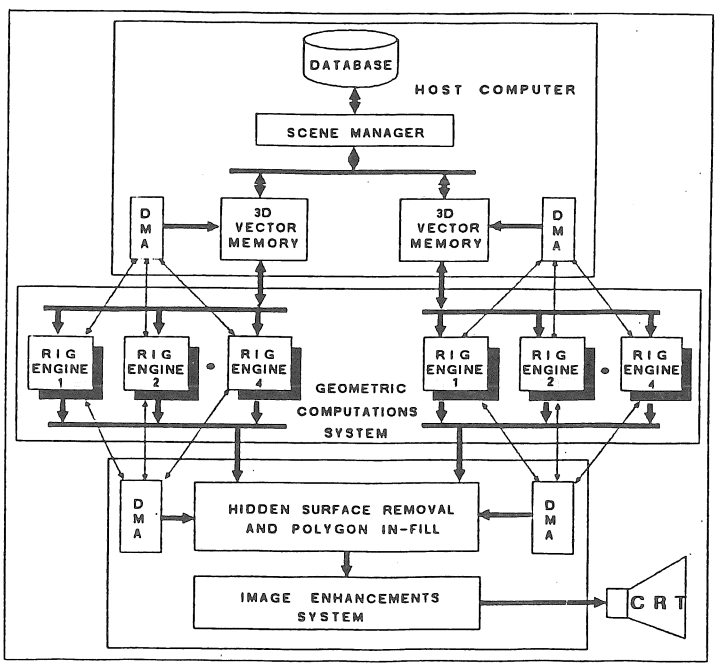

Figura 7 - Sistema MIGS com Processadores RISC Especializados para Operações Geométricas

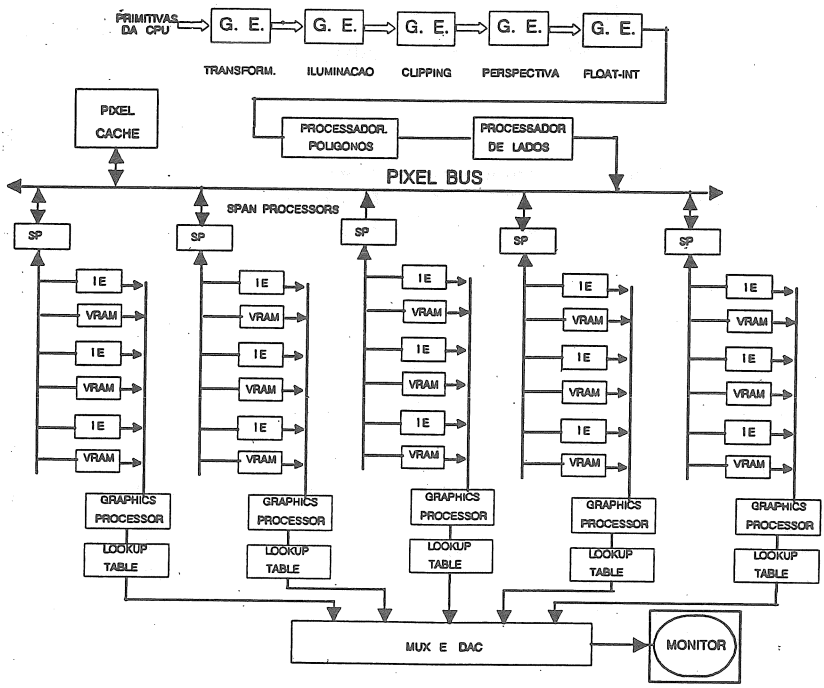

Figura 8 - Arquitetura do Sistema Silicon Graphics 4D-240/GTX

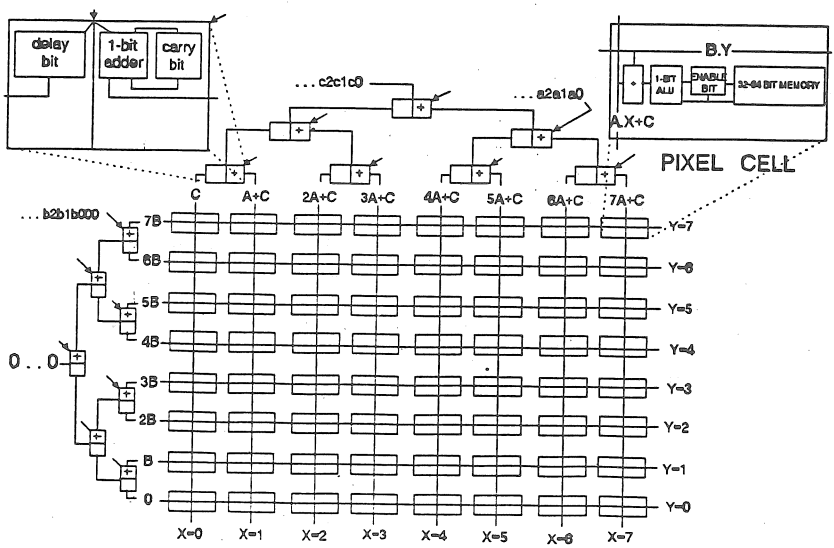

Figura 9 - Arquitetura do Sistema Pixel-Planes 4