Sintetizador Paralelo de Imagens de Alta Qualidade para ANIMAKER

Dinamérico Schwingel
Rodrigo de Losina Silva
Dante Augusto Couto Barone

Universidade Federal do Rio Grande do Sul - UFRGS
Instituto de Informática - II
Curso de Pós-Graduação em Ciência da Computação - CPGCC
Caixa Postal 15064 - CEP 91591-970
Porto Alegre, RS, Brasil
dino, rodrigo, barone@inf.ufrgs.br

Abstract. This paper describes a parallel high-quality render for the ANIMAKER system. The render is based on a ray-tracing algorithm and it was built in a distributed network of workstations. The integration with ANIMAKER and the parallel model of programming are described, then results and further directions are presented.

1 Introdução

Visualização de animações computadorizadas em tempo real, ou próximo deste, com alta qualidade é uma tarefa ainda realizada por poucas máquinas que, em sua maioria, possuem *hardware* especial para capacitá-las a realizar essa tarefa.

Este trabalho discute uma alternativa a esse enfoque, dado o surgimento de ilhas com alto poder computacional formadas por redes locais de estações de trabalho que, muitas vezes, estão com seu desempenho potencial subutilizado. Dessa forma pode-se obter uma máquina paralela/distribuída a um custo muito baixo e sem ter que atrelar o *software* a um determinado *hardware*.

Um sintetizador de imagens de alta qualidade baseado na técnica de *ray-tracing* foi implementado em tal ambiente para funcionar em conjunto com o sistema de animação modelada ANIMAKER [SIL92].

O texto deste trabalho está estruturado de forma que será, em primeiro lugar, introduzida a técnica de *ray-tracing* seguida de uma explanação sobre síntese paralela de imagens. Após serão descritos a integração e o sintetizador e, ao final, serão apresentados os resultados obtidos com o sistema.

2 Ray-Tracing

A técnica de *ray-tracing* foi introduzida por Whitted [WHI80] baseado em trabalhos anteriores que usavam técnicas conhecidas como *ray-casting*.

O algoritmo consiste, basicamente, em perseguir a trajetória de raios de luz em um espaço tridimensional a medida que vão se refletindo e refratando nos objetos posicionados nesse espaço. É possível obter imagens com elevado grau de realismo fotográfico, principalmente através da incorporação de modelos mais aperfeiçoados como o de [COO84] que permite a geração de imagens com profundidade de campo.

Devido a alta qualidade das imagens geradas e à facilidade de simular fenômemos óticos como reflexão e refração, muitos esforços tem sido feitos no sentido de diminuir o tempo de geração das imagens, o principal ponto fraco deste algoritmo.

Diversas técnicas foram criadas para acelerar o processamento do algoritmo seqüencial, entre elas estão: subdivisão espacial adaptativa [GLA84], subdivisão regular com propagação dos raios em aritmética inteira [FUJ86], eliminação da árvore de recursão e criação de hierarquias com eficientes cálculos de intersecção com envelopes [KAY86].

3 Síntese paralela de imagens

Além da aceleração do algoritmo seqüencial, trabalhos foram feitos com o objetivo de propor arquiteturas paralelas para obter melhor desempenho. Entre esses trabalhos estão uma proposta para uma arquitetura paralela com subdivisão adaptativa [DIP84], um sistema *pipeline* triprocessado modular [GAU88], uma máquina composta por uma rede tridimensional de processadores *Transputer* [PIT90] e uma implementação sobre uma máquina hipercúbica [BAD90].

Como outras técnicas de síntese de imagens, *ray-tracing* trabalha a partir do espaço imagem, de onde são projetados os raios para dentro da cena, fazendo um caminho inverso ao que ocorre fisicamente em uma fotografia, por exemplo. A cada *pixel* da imagem está associado um raio que pode ser calculado de forma independente, *i.e.*, sem depender de dados

que só estarão disponíveis após o processamento de algum outro raio.

Essa concorrência lógica configura um modelo de paralelismo de dados e a exploração de tal paradigma de programação em ambiente distribuído é uma forma eficiente de paralelizar/distribuir aplicações [SCH92b], notadamente uma aplicação com o algoritmo em questão.

Para se atingir boa eficiência no projeto de sistemas paralelos em ambiente distribuído deve-se ter uma relação de compromisso com distribuição de carga e eficiência de comunicação. Tal relação advém de que uma melhor distribuição de carga implica a utilização de um número maior de mensagens do que para, por exemplo, fazer uma distribuição estática que não exige troca de informações, entretanto pode aumentar o tempo total de execução.

4 Integração com ANIMAKER

ANIMAKER foi construído para funcionar em conjunto com sistemas de modelagem de objetos e de síntese de imagens, definindo animações através da técnica de quadros-chaves, através de um roteiro que contém instruções para a geração da animação.

A comunicação com os sistemas de síntese é feita através de arquivos gerados pelo ANIMAKER com a descrição de cenas de uma determinada animação. Estes arquivos descrevem todos os parâmetros de cada entidade presente na cena (câmera, atores, etc.), e referenciam arquivos contendo a forma dos objetos.

Os arquivos com a forma dos objetos eram definidos, inicialmente, no padrão OBT, um formato interno do CPGCC/UFRGS, que permitia a compatibilidade do sistema ANIMAKER com alguns modeladores de objetos.

Na definição do sistema, optou-se por esta forma de comunicação porque o tempo para a síntese de uma imagem de alta qualidade torna o custo de leitura de um arquivo de cena desprezível e, limitando a comunicação apenas a arquivos de descrição de cena há liberdade para utilizar qualquer sistema de síntese de imagens em conjunto com o ANIMAKER.

Para integrar os dois sistemas bastaria que o sintetizador aceitasse como entrada os arquivos de cena e de objetos, sob o controle do ANIMAKER. Optou-se, no entanto, por fazer modificações mais profundas em ambos os sistemas, visando uma integração mais eficiente.

Os sistemas de síntese de imagens baseados na técnica *ray-tracing* são mais eficientes para a geração de objetos definidos analiticamente do que para objetos definidos por malhas de polígonos. Porém, a exibição através de aramado (*wire-frame*) ou por técnicas como *z-buffer* exige, normalmente, a representação dos objetos na forma poligonal.

O formato então utilizado pelo ANIMAKER oferecia apenas esta última alternativa de representação. Portanto, para permitir uma comunicação eficiente entre o sistema de síntese e o sistema de animação, tornou-se necessário definir um padrão para a representação dos objetos, compatível com ambos os sistemas. Este padrão teria que ser capaz de representar os objetos tanto como malha de polígonos como na forma de primitivas modeladas analiticamente para aproveitar as características especializadas do sintetizador.

Considerando-se isso, optou-se por definir uma expansão do padrão OBT, denominada OBX (OBt eXtendido), que possui uma sintaxe mais aperfeiçoada.

Das informações contidas no formato OBX, aquelas associadas a faces e a vértices foram aproveitadas do formato OBT e definem uma malha de polígonos. Esta malha é utilizada pelo mecanismo de visualização por aramado do ANIMAKER e por sistemas de síntese de imagens baseados em malhas de polígonos.

A informação mais importante, acrescentada no formato OBX, é a da entidade ou entidades que descrevem o objeto analiticamente.

5 Aplicação Desenvolvida

A aplicação foi desenvolvida com base no compromisso entre uma boa distribuição de carga e a minimização do número de mensagens, conforme a seção 3. Considerando esse compromisso, foi utilizada uma modelagem do tipo mestre/escravos, na qual um processo distribui dinamicamente trabalho para um conjunto de outros que executam o algoritmo intensivo em computação e retornam seus resultados ao processo distribuidor.

Pitot et al. [PIT90] dividem os algoritmos paralelos de *ray-tracing* em dois tipos: aqueles que copiam toda a estrutura de dados para todos os processadores e os que copiam apenas um determinado volume para cada processador. Este algoritmo encontra-se na primeira classificação e isso foi projetado assim para que não houvesse um grande fluxo de mensagens que tenderia a saturar o meio físico compartilhado de comunicação, uma vez que *ray-tracing* possui um algoritmo global de iluminação.

Nesse ponto seguiu-se um enfoque diferente do adotado em [BAD90] e em [SCH92a], notadamente devido às características da interconexão entre os processadores que, naquele trabalho, possui canais dedicados, enquanto que neste é utilizado o meio físico comum de uma rede local.

A distribuição de trabalho consistiu da divisão do espaço imagem em subconjuntos disjuntos, deno-

minados pacotes de trabalho, a serem processados nos processos escravos. Cada pacote apresenta uma quantidade variável de carga, dependente do número de objetos atingidos pelos raios daquele pacote, e isso diminui a probabilidade da ocorrência de colisões na rede, quando do envio dos resultados ao mestre.

O tamanho dos pacotes de trabalho, medido em coordenadas do espaço imagem, é um parâmetro que está a disposição do usuário. Essa característica é usada para que seja feito um controle fino da distribuição, de acordo com a complexidade da cena. Baseado em testes realizados, foi estabelecido o valor padrão de uma linha para cada processador, pois assim obteve-se a melhor eficiência para cenas com número de objetos em torno de 100. Outra estratégia, esta visando otimizar o uso da rede de interconexão, pode ser usada através da manutenção de pacotes com tamanho múltiplo daquele usado pela rede. Cabe salientar que o tamanho físico dos pacotes de trabalho é sempre o mesmo, mudando apenas o tamanho dos pacotes com a imagem gerada, com diferentes parametrizações.

A figura 1 apresenta a estrutura de comunicação entre o ANIMAKER e o sintetizador de imagens, considerando o uso de n processadores para os processos escravo.

A comunicação entre os processos — locais e remotos — foi implementada através de primitivas disponíveis no sistema operacional SunOS 4.1,

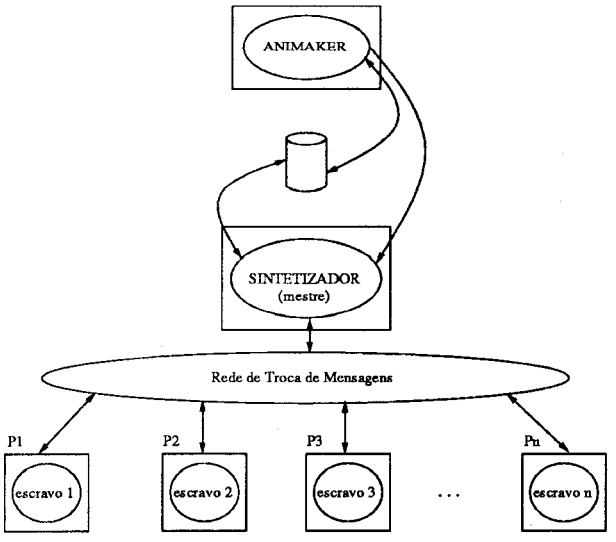

Figura 1: Estrutura de Comunicação do Sistema

5.1 Descrição dos Algoritmos

Os algoritmo implementado aqui foi, inicialmente, desenvolvidos para uma máquina paralela fortemente acoplada [SCH92a], posteriormente sendo adaptado para execução em ambiente distribuído. Trata-se de um sistema de *ray-tracing* básico que não incorpora técnicas especiais de aceleração como as citadas na seção 2. Contudo, uma vez que o paralelismo é explorado ao nível de processo, pode-se adicionar qualquer uma das técnicas sem modificação na estrutura de paralelização.

O paradigma de programação utilizado divide a implementação em dois programas: mestre e escravo.

O programa mestre é responsável pela leitura da descrição da cena, execução dos escravos, controle da comunicação, detecção de falhas e envio de trabalho, concorrentemente com o recebimento dos resultados.

O programa escravo implementa o algoritmo completo de síntese de imagens e executa-o para as coordenadas especificadas no pacote de trabalho até que receba sinalização para encerrar sua execução.

Observa-se que os escravos executam um algoritmo estritamente seqüencial que pode ser desenvolvido e testado dessa forma até que seja realizada sua implementação paralela.

5.2 Avaliação do Sistema

O sintetizador se encontra em fase final de integração com o sistema ANIMAKER e, ainda em sua versão seqüencial, os dois foram utilizados na realização de uma vinheta para a Secretaria de Ciência e Tecnologia do Rio Grande do Sul pelo CPGCC.

A estratégia de paralelização mostrou-se bastante eficaz e na figura 2 pode-se ver o ganho obtido pela execução paralela, em relação ao mesmo algoritmo seqüencial, do sintetizador na criação de uma cena composta por 97 objetos. Este é um exemplo escolhido aleatoriamente dentre os testes realizados com mais de 40 objetos que apresentaram comportamento bastante similar. Os tempos de execução foram tomados em horários de ociosidade da rede (tipicamente à noite) para que processos de outros usuários não interferissem nas medidas. O ganho apresentado foi obtido em estações de trabalho Sparc 1+ interconectadas por rede *Ethernet*.

6 Conclusões

Foi apresentado um sistema paralelo de síntese de imagens de alta qualidade baseado na técnica de *ray-tracing* e sua integração com um sistema de animação.

A integração com o sistema ANIMAKER deve prosseguir e evoluir para, talvez, um esquema de comunicação mais estreito, sendo para isso necessário definir um protocolo de comunicação daquele sistema para com sistemas de síntese.

O uso do formato OBX oferece uma alternativa

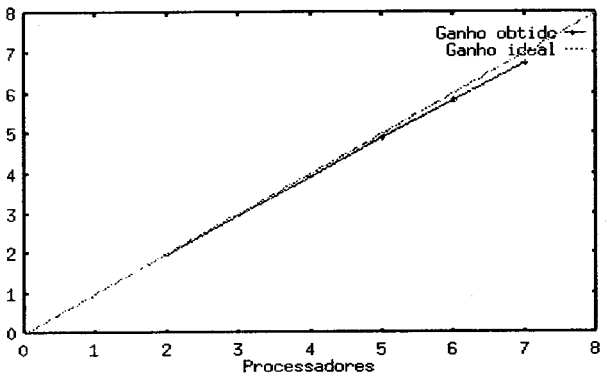

Figura 2: Ganho de Desempenho Obtido

eficiente para a criação de animações que explorem a representação analítica utilizada pelo sintetizador.

O sistema de síntese mostrou uma boa eficiência com relação ao seqüencial, aproximando-se muito do ganho ideal nos testes realizados com até sete máquinas.

Com o aumento da velocidade do processo de síntese pode-se utilizar de imagens com qualidade final, ainda na criação de animações, com mais freqüência, possibilitando, dessa forma, maior sinergia no processo de criação.

Referências

[BAD90] BADOUEL, D.; PRIOL, T. VM_pRAY: An Efficient Ray Tracing Algorithm on a Distributed Memory Parallel Computer. IRISA. Publication interne 506. Jan.1990. 52p.

[COO84] COOK, R. L.; PORTER, T.; CARPENTER, L. Distributed Ray-Tracing. **ACM Computer Graphics**, New York, v.18, n.3, p.137-145, Jul.1984.

[DIP84] DIPPÉ, M; SWENSEN, J. An Adaptive Subdivision Algorithm and Parallel Architecture for Realistic Image Synthesis. **ACM Computer Graphics**, New York, v.18, n.3, p.149-158, Jul.1984.

[DUN90] DUNCAN, R. Survey of Parallel Computer Architectures. **IEEE Computer**, New York, v.23, n.2, p.5-16, Feb.1990.

[FUJ86] FUJIMOTO, A.; TANAKA T.; IWATA K. ARTS: Accelerated Ray-Tracing System, **IEEE Computer Graphics and Applications**, New York, v.6, n.4, p.16-26, Apr.1986.

[GAU88] GAUDET, S. *et al.* Multiprocessor Experiments for High-Speed *Ray-Tracing*. **ACM Transactions on Graphics**, New York, v.7, n.3, Jul.1988, p.151-179.

[GLA84] GLASSNER, A. S. Space Subdivision for Fast Ray Tracing. **IEEE Computer Graphics and Applications**, New York, v.4, n.10, p.15-22, Oct.1984.

[GUS88] GUSTAFSON, J. L. Reevaluating Amdahl's Law. **Communications of the ACM**. New York, v.31, n.5, p.532-533, May 1988.

[KAY86] KAY, T. L.; KAJIYA J. T. Ray Tracing Complex Scenes. **ACM Computer Graphics**. v.20, n.4, p.269-278, 1986.

[PIT90] PITOT, P.; MOISAN B.; DUTHEN Y.; CAUBET R. A Transputer Based Implementation of the VOXAR Project. **Microprocessing and Microprogramming**, Amsterdam, v.30, n.1-5, p.347-354, Aug.1990.

[QUI87] QUINN, M. J. **Designing Efficient Algorithms for Parallel Computers**. New York:McGraw-Hill. 1987. 288p.

[SCH92a] SCHWINGEL, D. **APART^2 — Uma Arquitetura Paralela Assíncrona para** *Ray-tracing* **em** *Transputers*. Porto Alegre:II-UFRGS. 1992. (Projeto de Diplomação). 68p.

[SCH92b] SCHWINGEL, D.; BARONE, D. A. C. Exploração de Paralelismo de Dados em Ambiente Distribuído. **IV Simpósio Brasileiro de Arquitetura de Computadores – Processamento de Alto Desempenho**. Anais. São Paulo:LSI-USP. p.459-470. 1992.

[SEI88] SEITZ, C.; MAYR E.; DALLY B.; JOHNSSON, L.; ABU-MOSTAFA, Y.; LYON, D.; ACKLAND, B. **VLSI and Parallel Computation**. Morgan Kaufmann. 1988. 470p.

[SIL92] SILVA, R. L.; OLABARRIAGA, S. D. ANIMAKER - Sistema de Animação Modelada. **V SIBGRAPI**. Anais. p. 259-267. 1992. 1990. 353p.

[WHI80] WHITTED, T. An Improved Illumination Model for Shaded Display. **Communications of the ACM**, New York, v.23, n.6, p.343-349, Jun.1980.