Comparação entre Sistemas para Renderização em Tempo Real WireGL e OpenRT

Fábio P. Basso[1], Daniel Welfer[1], Diego Kreutz[1], Raquel M. Pillat[2], *
[1]UFSM - Universidade Federal de Santa Maria
[2]UNIJUÍ- Universidade Regional do Noroeste do Estado do Rio Grande do Sul
{fbasso, kreutz, welfer}@inf.ufsm.br, pillati@detec.unijui.tche.br

Introdução

O processo de renderização cria uma imagem calculando os fatores ópticos que exercem influência na visualização de objetos. Para que uma imagem seja formada por este processo é necessário que se defina uma cena com objetos, tridimensionais ou não, com suas propriedades, e os parâmetros de um visualizador, uma câmara, por exemplo. A imagem final é o resultado de todos os fatores ópticos calculados em cada ponto da cena que está definida nos parâmetros do visualizador.

A renderização pode ser feita de duas formas, ou por traçado de raios, ou por rasterização, sendo que ambas necessitam de alto desempenho computacional para efetuá-la em tempo real. Para efetuá-la de forma econômica, como alternativa ao uso de máquinas multiprocessadas, foram desenvolvidos dois sistemas, WireGL e OpenRT, que fazem isso em aglomerados de computadores. O primeiro realiza a renderização distribuída utilizando o processo de rasterização enquanto que o segundo utiliza o traçado de raios *(Ray Tracing)*. O objetivo deste testo é comparar estes dois sistemas e apresentar uma análise teórica inicial. Esta poderá servir de base para a escolha entre a utilização do WireGL ou do OpenRT na implementação de uma aplicação de animação gráfica 3D, que necessita efetuar a renderização de cenas tridimensionais em tempo real.

Complexidade Geométrica, Realismo e Interatividade X Custo Computacional

Complexidade geométrica e custo computacional possuem uma relação intrínseca, sendo que o custo computacional aumenta conforme o faz a complexidade geométrica. Sistemas gráficos de tempo real que trabalham com dados geométricos complexos geralmente utilizam NURBS (*free-form geometry*) e usam um processo de tecelagem para convertê-los em triângulos [BEN02]. Freqüentemente, como solução para sistemas de tempo real, o que se tem feito para reduzir a complexidade geométrica é utilizar técnicas de redução de malha que reduzem o número de triângulos dos objetos. Porém, em sistemas que necessitem de objetos reais, é necessário que se mantenha a complexidade geométrica, sendo necessário um poder de processamento elevado.

Alguns efeitos ópticos, na maioria das vezes, não podem ser percebidos, a não ser que os objetos estejam em movimento. A interatividade provê a visualização destes efeitos. O processo de rasterização utilizado por OpenGL é computacionalmente

* Trabalho desenvolvido na disciplina de Programação Paralela do curso de Engenharia da Produção da UFSM, ministrada pelo professor Benhur O. Stein.

econômico, mas deixa de avaliar estes efeitos na renderização de objetos [BEN02], não podendo calcular a refração e reflexão da luz em superfícies curvas, por exemplo. O processo de rasterização apresenta boa qualidade nas imagens renderizadas, mas, não é capaz de representar efeitos reais. Já o processo de traçado de raios pode simular tais efeitos ópticos e, portanto, criar objetos com aparência real, porém necessita de mais tempo de processamento do que a rasterização.

Em se tratando de realismo de movimento, é necessário que a cena seja renderizada o mais rápido possível. Sistemas interativos necessitam que as imagens processadas sejam mostradas numa certa freqüência, cerca de 30 quadros por segundo, para representar um movimento real, sendo necessário que se processem, no sistema, 30 cenas por segundo. Neste caso, a rasterização é mais indicada pois é um processo mais rápido que o traçado de raios.

Arquiteturas Gráficas Paralelas

Arquiteturas gráficas paralelas podem geralmente ser classificadas de acordo como o ponto no *pipeline* gráfico em que os dados serão redistribuídos [HUM01]. Sistemas gráficos desenvolvidos para aglomerados de computadores tem, na grande maioria, o mesmo modo de funcionamento. Existem as máquinas servidoras de processamento, que efetuarão cálculos de renderização, e as máquinas clientes, que fazem requisições de processamento às máquinas servidoras.

A distribuição da cena entre as máquinas servidoras de processamento pode ser feita de várias formas. A forma mais utilizada por traçado de raios é, se possível, manter um mesmo objeto sempre numa mesma máquina servidora, por motivos de desempenho, dividindo a cena em partes heterogêneas [WAL03]. O processo de rasterização, geralmente, divide a cena em partes homogêneas, definidas como ladrilhos. Define-se, para uso geral, as divisões da cena como "segmentos" para ambos os casos, traçado de raios e rasterização.

Cada máquina servidora é responsável em calcular e retornar para a máquina cliente a cor dos pixels do segmento recebido. Estes segmentos são armazenados em um ou mais *framebuffers* nas máquinas clientes que posteriormente, após a conclusão do processo de renderização, farão a composição destes segmentos numa imagem final. No princípio, as máquinas clientes que efetuam requisições de processamento, escolhem as máquinas servidoras que serão responsáveis pelo processo de renderização. O programa gráfico deverá conter as requisições que serão paralelisáveis, sendo encargo das máquinas clientes distribuí-las entre as máquinas servidoras. Escolhidas as máquinas e estabelecidas as conexões, as clientes delegam a cada servidor um segmento da cena e o efeito a ser calculado. Então, aguardam até que todos os segmentos estejam armazenados nos *framebuffers* para ordená-los e reconstruir a imagem.

WireGL

WireGL provê uma API com sintaxe igual a de OpenGL[**] devido a essa biblioteca ser amplamente utilizada no desenvolvimento de aplicações gráficas. Assim

[**] http://www.opengl.org

como OpenGL, WireGl utiliza, no processo de *pipeline*, a rasterização, poupando recursos computacionais na redução dos dados a serem processados [HUM99]. WireGL foi desenvolvida na universidade de Stanford, no laboratório de Computação Gráfica, e tem sido utilizada por muitas ferramentas de produção 3D.

O sistema de renderização WireGl consiste de um ou mais clientes submetendo comandos OpenGL simultaneamente para um ou mais servidores gráficos, chamados *pipeservers* [HUM01]. Os servidores estão conectados por uma rede de alta velocidade e todos possuem um acelerador gráfico, podendo administrar, cada um, muitos segmentos da cena. Os clientes visualizam um simples *pipeline* gráfico conceitual, pois WireGL virtualiza esta arquitetura que pode ser vista na Figura 1.

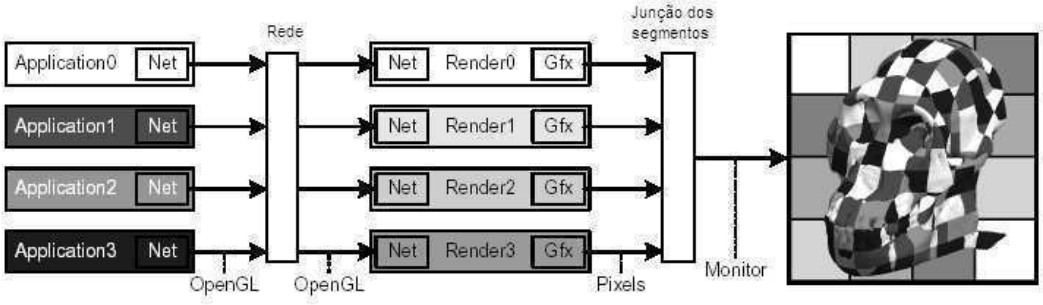

Figura 1: WireGL é composto de nós de aplicação, nós de renderização e display [HUM01].

OpenRT

A API OpenRT é muito semelhante à OpenGL, utilizando funções de nomes similares, trocando apenas o gl por rt. Até mesmo a semântica das funções é a mesma da OpenGL [WAL02]. OpenRT utiliza no *pipeline* de produção o traçado de raios, efetuando operações computacionais a mais que as efetuadas por WireGL. Sombreamento programável, semântica de quadros e retenção de objetos são fatores que são agregados em OpenRT [WAL03]. Portanto, a qualidade e realidade das imagens renderizadas são superiores às feitas por um algoritmo de rasterização.

A arquitetura utilizada por OpenRT é a mesma de WireGL, cliente-servidor, no qual a cena é dividida em partes homogêneas, diferente de outros sistemas de traçado de raios, obtendo uma escalabilidade melhor. Enquanto os servidores estão processando um quadro N, os clientes já enviam a requisição para processamento do quadro N+1, reduzindo a latência da rede que interliga o aglomerado de computadores. O sistema OpenRT é desenvolvido pelo grupo de Computação Gráfica da Universidade de Saarland, Alemanha.

Comparação

Wald [WAL03] utilizou OpenRT em um aglomerado de computadores para renderizar uma cena complexa com 16 PCs AthlonMP 1800, com dois processadores cada, totalizando 32 UCP. Os servidores de processamento estavam interconectados por uma rede Fast Ethernet. Pode-se obter uma taxa de quadros acima de 10 quadros por segundo com uma resolução de vídeo de 640x480, num total de 800.000 triângulos por segundo. Humpheys [HUM01] utilizou WireGL para renderizar *Nurbs* em um

aglomerado de computadores composto de 16 Compaq SP750 para processamento, cada uma contendo dois processadores Pentium III 800 com uma placa aceleradora gráfica NVIDIA Quadro2 e interconexão de rede Myricom com largura de banda de 101MB/seg, atingiu-se, para um display de 1024x768, 467.000 triângulos/seg, sem utilizar redução de malha.

Levando em conta que as máquinas utilizadas para testar o sistema OpenRT eram superiores às utilizadas para testar WireGL, é possível se chegar a conclusão que este último tem um desempenho, na proporção de processamento por triângulos calculados em um segundo, melhor que o sistema OpenRT.

Conclusões e Trabalhos Futuros

Este documento apresentou as principais características do WireGL e do OpenRT. Além disso, foram abordados alguns aspectos relativos as vantagens e desvantagens da utilização do traçado de raios ou rasterização no processo de renderização de imagens.

Apesar das publicações de Humpheys e Benthin, que relatam os aspectos dos dois sistemas, WireGL e OpenRT respectivamente, informarem a velocidade alcançada no processo de renderização, não é possível de ser feita uma análise de desempenho correta, devido à utilização de duas estruturas computacionais diferentes, dificultando a escolha de um dos sistemas. Portanto, é necessário a realização de testes comparativos práticos e reais em uma mesma estrutura computacional. E, a partir dos resultados obtidos, permitir a escolha do sistema de renderização em tempo real que melhor se enquadre para o desenvolvimento de uma aplicação de animação de personagens compostos de articulações em tempo real.

Bibliografia

[BEN02] BENTHIN, C.; DAHMEN, T.; WALD, I.; SLUSALLCK, P. **Interactive Headlight Simulation** – A Case Study of Interactive Distributed Ray Tracing. In: Proceedings of EUROGRAPHICS Workshop on Parallel Graphics and Visualization (PGV), 2002.

[HUM00] HUMPHEYS, G; ELDRIDGE M.; BUCK I; STOLL G.; HANRAHAN P. **Distributed Rendering for Scalable Displays**. In: Proceedings of SC2000. Disponível por www em http://graphics.stanford.edu/software/wiregl/

[HUM01] HUMPHEYS, G; ELDRIDGE M.; BUCK I; STOLL G.; EVERETT, M.; HANRAHAN P. **WireGL:** A Scalable Graphics System for Clusters. In: Proceedings of SIGGRAPH 2001. Disponível por www em http://graphics.stanford.edu/software/wiregl/

[HUM99] HUMPHEYS, G.; HANRAHAN, P. **A distributed Graphics System for Large Tiled Displays**. In: Proceedings of IEE Visualization 1999. Disponível por www em http://graphics.stanford.edu/software/wiregl/

[WAL02] WALD, I.; BENTHIN C.; DIETRICH A.; SLUSALLEK P. **A Scalable and Flexible Rendering for Interactive 3D Graphics**. Technical Report 2002-01. Disponível por www em http://www.openrt.de/Publications/

[WAL03] WALD, I; BENTHIN C.; DIETRICH A.; SLUSALLEK P. **Interactive Ray Tracing on Commodity PC clusters** – State of the Art and Pratical Applications. In: Proceedings of EuroPar 2003.