Investigação de ferramentas computacionais para construção de uma arquitetura baseada em *Cluster* de PC para aplicação em interface avançada na área de distribuição de energia elétrica

Antônio Valério Netto[1], Gabriel de Queiroz Rezende[1,2]

[1] Cientistas Associados Ltda. – Divisão de Sistemas Interativos
Rua Alfredo Lopes, 1717 – CEP: 13560-460, São Carlos, SP

[2] Escola de Engenharia de São Carlos – Universidade de São Paulo (EESC/USP)
Av. Trabalhador São-carlense, 400 – CEP 13560-970, São Carlos, SP.

{antonio.valerio, gabriel.rezende}@cientistasassociados.com.br

Resumo: Nesse artigo são apresentados os resultados de um estudo relacionado a uma solução de processamento computacional para o projeto intitulado "Sistema computacional para redução de perdas em redes de distribuição de energia com interface em realidade virtual". Um das maiores barreiras encontradas atualmente na utilização de ambientes interativos é a complexidade dos mesmos, fazendo com que requeiram uma grande capacidade de processamento, necessitando assim o uso de supercomputadores de custo elevado, o que muitas vezes inviabiliza diversos projetos. Aliado a esse problema, a limitação física para a alocação de diversos dispositivos em uma única estação, tem sido outro fator que tem dificultado o desenvolvimento de tais aplicações. Foram pesquisadas e avaliadas as funcionalidades de diversas ferramentas computacionais, como o *NetJuggler* e *Cluster Juggler*, que entre outras características, são responsáveis por distribuir o processamento computacional entre vários computadores com o objetivo de aumentar o poder de processamento.
Palavras-chaves: Sistemas Interativos, Realidade Virtual, *Clusters* para PC, Interface Homem-Computador.

Investigation of computational tools for construction of an architecture based on PC Cluster for application in advanced interface in the electric power distribution area

Abstract: In this article, the results of a case study of processing computational solution for the project entitled System computational for losses reduction in energy distribution nets with virtual reality interface are presented. Nowadays, one of the largest barriers found in the use of interactive systems is their complexity, requesting a great processing capacity and needing the use of high cost supercomputers. These factors a lot of times make unfeasible several projects. Ally to that problem, the physical limitation for the allocation of several devices in a single station, it has been other factor that has made it difficult the development of such applications. The funcionalities of several computational tools were researched and appraised, like NetJuggler and Cluster Juggler. Among their characteristics, someone are responsible for distributing the computational processing among several computers with the objective of increasing the processing power.
Keywords: Interactive Systems, Virtual Reality, Clusters for PC, Human-Computer Interface.

(Received April 03, 2005 / Accepted June 15, 2005)

1. INTRODUÇÃO

Este trabalho se insere no contexto da investigação de ferramentas computacionais para serem aplicadas em sistemas com interfaces avançadas baseadas em ambientes virtuais, com o objetivo de garantir um alto nível de imersão do usuário dentro desse ambiente por meio de uma relação homem-máquina eficiente. Dentro do processamento gráfico envolvido nos ambientes virtuais, a renderização é tipicamente a tarefa que exige maiores recursos computacionais, sendo responsável na maioria dos casos, pelo gargalo do sistema [1].

Basicamente, a renderização consiste em calcular o efeito de cada primitiva em cada *pixel*. Conforme as características de modelagem e visualização, as

primitivas podem estar posicionadas em qualquer região dentro ou fora do campo de visão do usuário. Dessa forma, quanto mais complexo for o mundo virtual, mais difícil será atingir uma taxa de atualização adequada. Até recentemente, apenas as plataformas computacionais dedicadas como os da *Silicon Graphics* eram capazes de atingir um desempenho satisfatório para esse tipo de aplicação. As principais desvantagens desses sistemas são o alto custo de aquisição de *hardware* e a dependência tecnológica. Uma alternativa para sistemas com grande demanda de processamento gráfico são os *clusters* de PCs. Basicamente, um *cluster* é um sistema distribuído que consiste de uma rede de PCs interconectados que trabalham em conjunto para uma ou várias tarefas [2].

Durante a última década, vários pesquisadores [3] [4] [8] [13] vêm desenvolvendo algoritmos e técnicas focalizadas na renderização paralela, viabilizando cada vez mais aplicações interativas devido ao aumento de capacidade de processamento dos *clusters*. Entre as várias vantagens citadas por alguns autores [3] [4], destacam-se: o baixo custo, o poder de processamento e escalabilidade. Um dos objetivos do projeto em questão é permitir a interação em tempo real com o modelo 3D da cidade de São Carlos (SP). Dada a complexidade desta aplicação, esse ambiente deve ser capaz de atingir uma taxa de atualização adequada e, para tanto, optou-se pela alternativa de *clusters* de PCs.

O projeto principal [5] está relacionado a um sistema computacional para redução de perdas em redes de energia elétrica. Ele utiliza basicamente duas tecnologias: os Algoritmos Evolutivos (AE) [15] e a Realidade Virtual (RV). Os AE com Representação por Cadeias de Grafo (RCG) são utilizados para determinação do circuito com perdas mínimas. Enquanto que a RV é usada para o desenvolvimento de um ambiente interativo e imersivo. O sistema possui quatro módulos principais: o reconfigurador (AE com RCG), a interface interativa, o codificador, e os dados da rede de distribuição conforme mostra a Figura 1.

Os AEs englobam vários tipos de métodos de aprendizado de máquina (algoritmos genéticos, estratégias evolutivas, programação genética) [15] inspirados em mecanismos biológicos como a seleção natural e a evolução. Estes métodos são freqüentemente usados em problemas de otimização e inteligência computacional [15]. A Representação por Cadeias de Grafos (RCG) é uma estrutura de dados utilizada para representar grafos no formato de floresta. Essa representação é utilizada pelo AE do sistema para minimizar as perdas na distribuição de energia.

Durante o desenvolvimento do projeto, também foram identificadas várias pendências a respeito do *hardware* e *software* necessários para atingir um bom desempenho da aplicação. Inicialmente, foram pesquisados diversos tipos de dispositivos de *hardware* para utilização com o ambiente virtual desenvolvido, com a finalidade de transmitir ao usuário uma maior sensação de imersão.

Dessa forma, foram estudados dispositivos como *DataGloves* (modelos *P5 DataGlove*, *5DT DataGlove*, *CyberGlove* e *PinchGlove*), *Trackers* (modelos *Polhemus* - *Fastrak* e *Ascension* - Flock of Birds), e S*hutterglasses* (*CrystalEyes e Stereo Eyes*).

Após a realização de um estudo avançado sobre esses dispositivos, procurou-se optar por modelos que mais se adequassem e suprissem as necessidades contempladas no projeto. Sendo assim, levando-se em consideração diversos aspectos, inclusive financeiros, os modelos escolhidos foram: a Luva 5DT DataGlove 5-W Right-handed, o Tracker da Acension (Flock of Birds) e o shutterglasses Crystal Eyes 3.

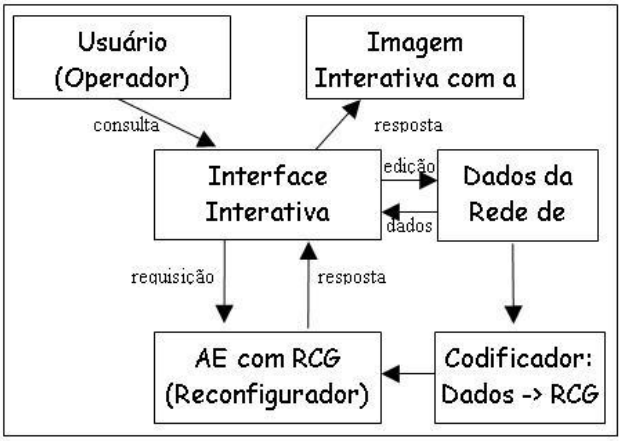

Figura 1: Visão geral dos módulos do sistema computacional para redução de perdas.

A tarefa principal dentro dessa atividade foi o estudo das principais referências a respeito de como inserir esses dispositivos, no ambiente virtual e suas funcionalidades dentro do mesmo, sendo coletadas informações que fornecessem uma metodologia para utilizá-los juntamente com as ferramentas VRJuggler (VRJ) [9] e o *Open Scene Graph* (OSG) [10]. Uma das dificuldades enfrentadas durante essa tarefa foi à falta de uma boa documentação, sendo que a maioria das informações adquirida durante esse estudo foi advinda

de fóruns de discussão de usuários do VRJ. A conclusão parcial foi que, a princípio, não existe uma solução de *software* de fácil implementação, já que um dos problemas cruciais é a inexistência de *drivers* de comunicação eficientes entre os dispositivos e as ferramentas.

Na Seção 2 são apresentadas as abordagens básicas para a renderização paralela. Na Seção 3 são apresentadas as ferramentas computacionais utilizadas neste trabalho. Na Seção 4 são introduzidos os dois modos de *cluster*ização pesquisados, o *Cluster* Juggler e o NetJuggler. Na seção 5 são descritas as principais dificuldades encontradas no decorrer do trabalho de levantamento técnico. A Seção 6 contém as considerações finais sobre esta etapa do trabalho.

2. RENDERIZAÇÃO PARALELA

A renderização de imagens complexas é uma tarefa que exige alto processamento computacional, sendo necessária uma alta taxa de quadros por segundo (*frame rate*) [1] [6]. Aplicações de visualização científica e Realidade Virtual, demandam centenas de MFLOPS (milhões de operações de ponto flutuante por segundo) e uma largura de banda alta, ultrapassando as capacidades computacionais dos PCs comuns. Por essa razão, a paralelização é uma tarefa crucial para a viabilização de sistemas gráficos de alto desempenho. Para realizá-la, é aplicado um *pipeline* de renderização. Esse *pipeline* consiste de duas tarefas principais: processamento geométrico (transformação, recorte ou *clipping*, iluminação, etc) e rasterização (conversão de varredura ou *scan-conversion*, sombreado, e determinação de visibilidade).

Uma das estratégias típicas na paralelização por Processamento Geométrico, é a alocação de um subconjunto de primitivas da cena para cada um dos processadores. No caso da rasterização, cada processador é responsável por uma parte dos cálculos sobre os *pixels*. Conforme Sutherland e colaboradores [7], o problema de renderização pode ser enfocado como um problema de ordenamento de primitivas na tela. No caso da renderização paralela, é esse ordenamento que faz a redistribuição dos dados entre os diferentes processadores, ou seja, o posicionamento do ordenamento determina a estrutura do sistema de renderização paralela resultante. Molnar [6] e colaboradores dividem as abordagens sobre renderização paralela em três grupos. Essa divisão está baseada no momento onde o ordenamento é realizado dentro no *pipeline* de renderização para a determinação da superfície visível. O ordenamento pode ser realizado no processamento geométrico (*sort-first*), entre o processamento geométrico e a rasterização (*sort-middle*) ou durante a rasterização (*sort-last*) [8].

No *sort-first*, os algoritmos distribuem as primitivas durante o processamento geométrico, dessa forma os diferentes processadores realizam o resto dos cálculos de renderização. Isto implica em dividir a tela 2D em regiões descontínuas e alocá-las em diferentes processadores. Na abordagem *sort-middle* as primitivas são distribuídas na metade do *pipeline* de renderização (entre o processamento geométrico e a rasterização), Sendo assim, elas já se encontram pré-processadas e transformadas em coordenadas de tela prontas para a rasterização. Dado que na maioria de sistemas dedicados, o processo geométrico e a rasterização são realizadas em diferentes processadores, essa é a forma natural de separar o *pipeline*. Por fim, na abordagem *sort-last*, os algoritmos distribuem os *pixels* durante a rasterização, sendo assim, um subconjunto de primitivas é alocada em cada processador, que irá calcular os *pixels* do seu subconjunto sem importar a localização dos mesmos. Vale ressaltar que os processadores de renderização transmitem seus *pixels* para os processadores de composição de imagem (compositores) por meio da rede. Esses compositores é que serão encarregados de formar a imagem final a partir dos dados recebidos dos renderizadores.

3. BIBLIOTECAS PARA DESENVOLVIMENTO DE AMBIENTES VIRTUAIS

Para o desenvolvimento do sistema, foram utilizados, principalmente, duas bibliotecas de código aberto, o VRJuggler e o *Open Scene Graph*, descritos sucintamente abaixo:

VRJuggler (VRJ): projeto de pesquisa do Centro de Aplicações para Realidade Virtual da *Iowa State University* [9]. Trata-se de uma biblioteca na linguagem C++ que auxilia no desenvolvimento de aplicações em ambientes imersivos e interativos, propiciando o desenvolvimento e a sua execução, independentes da arquitetura de *hardware* e do sistema operacional utilizados. Dessa forma, o VRJ provê uma plataforma virtual (PVJ) disponibilizando um ambiente operacional unificado, no qual os programadores podem escrever e testar aplicações, usando recursos disponíveis que garantam a portabilidade de seus trabalhos. O sistema básico da PVJ é composto: por uma aplicação objeto, um gerenciador de desenho e o *kernel*. A interface entre

a aplicação objeto e o VRJ consiste de um *kernel* que proporciona a abstração de *hardware* para a plataforma virtual e o gerenciador de desenho proporciona a abstração para as APIs gráficas. O *kernel* é responsável pelo controle de todos os componentes do sistema VRJ e sua interface propicia uma API para o *hardware* do ambiente. O *kernel* não depende de detalhes específicos de APIs gráficas, ele obtém toda informação necessária do gerenciador de desenho, que consiste de um gerenciador externo do *kernel* do VRJ. Vale ressaltar que as aplicações utilizam o gerenciador de desenho para acessar qualquer detalhe específico e necessário de uma API. O conceito de uma plataforma virtual facilita e simplifica os esforços do desenvolvimento de aplicações em sistemas de RV, propiciando um ambiente de trabalho unificado que suporta o desenvolvimento e execução dessas aplicações. Uma plataforma virtual garante a longevidade das aplicações, permitindo que seus programadores acompanhem os avanços tecnológicos sem a necessidade de se investir a cada nova tecnologia desenvolvida, tempo e recursos modificando aplicações para suportá-la.

***Open Scene Graph* (OSG):** biblioteca gráfica de alto nível para o desenvolvimento de aplicações gráficas de alto desempenho como simuladores de vôo, jogos, RV e visualização científica. O OSG [10] é uma biblioteca multi-plataforma desenvolvida em C++ que permite aos programadores, representar objetos em uma cena usando estruturas de dados de árvore, chamadas de *scene graph*. Esta estrutura é usada para representar o mundo virtual. O nó raiz abrange o mundo virtual completo. Este mundo é quebrado em uma hierarquia de nós que podem representar grupos, propriedades de posicionamento, animação e definições de relações entre objetos. As folhas dessa árvore contêm os próprios objetos, sua geometria e propriedades materiais. Entre as principais vantagens do OSG destacam-se a portabilidade, a produtividade, a escalabilidade e o desempenho. Esta ferramenta é desenvolvida dentro da filosofia do *software* livre, e possui muitas versões, cada uma adaptada a novas necessidades e extensões para aplicações gráficas cada vez mais completas e complexas. O OSG não dispõe de uma documentação da linguagem de descrição de cenas que sirva de referência para o programador, mas pode-se obter informação de outras ferramentas similares que possuem a mesma filosofia.

4. ESTUDOS DE PARALELIZAÇÃO DO AMBIENTE INTERATIVO

Considerando que o sistema está planejado para ser executado em um *cluster* de PC e dada a complexidade da aplicação, foram avaliadas diversas alternativas. Na literatura foram encontradas duas ferramentas que permitem trabalhar em *cluster* de PC usando VRJ: o *NetJuggler* (NJ) [11] e o *ClusterJuggler* (CJ) [12]. Ambas as ferramentas são extensões da biblioteca VRJ, e permitem distribuir os diferentes componentes da Plataforma Virtual, como dispositivos *DataGloves*, *ShutterGlasses, Trackers* e *Displays* de visualização, entre as várias máquinas que compõem o *cluster*.

O NetJuggler usa um protocolo desenvolvido para aplicações computacionais de alto desempenho em comunicação distribuída, o MPI [14]. Uma solução imediata para programadores com experiência nesse protocolo seria a utilização do NJ, já que essa ferramenta utiliza a mesma programação paralela que o MPI. Porém, nosso estudo sobre essa ferramenta foi limitado por duas razões principais: a mesma não apresenta compatibilidade com a versão 2.0 do VRJuggler, utilizada no sistema em desenvolvimento e as diversas vantagens oferecidas pelo Cluster Juggler, mencionadas a seguir.

Uma das características mais atraentes do CJ, é que o mesmo permite por meio de configurações de *software*, a adaptação de diferentes tipos de dispositivos de *hardware* para sistemas de RV. Além disso, o CJ provê portabilidade e escalonamento para as aplicações, ocultando os detalhes de implementação do *cluster*. Um componente muito importante do *Cluster Juggler* é o gerenciador de dispositivos de entrada remota *(*RIM - *Remote Input Manager*). O RIM é formado por um conjunto de *plugins* que possibilitam realizar tarefas como a de uma máquina do *cluster* atuar como um servidor de dispositivos de *hardware*, permitindo que aplicações remotas requisitem informações de um dispositivo local, ou mesmo que todos os nós do sistema possuam a mesma informação de entrada de um dispositivo, como se este estivesse fisicamente ligado cada nó. As funções mais importantes do RIM são:

- ✓ **Transparência da localização do dispositivo:** evita que o programador de uma determinada aplicação se preocupe com a localização dos dispositivos, já que uma vez que o *cluster* esteja devidamente configurado, o programador e a aplicação podem atuar como se cada dispositivo de entrada estivesse conectado a todos os nós do sistema;

✓ ***Cluster*** **heterogêneo:** essa função possibilita a remoção de uma típica barreira existente na construção de um *cluster:* a necessidade da utilização de computadores com configurações idênticas. Assim, com a utilização dessa ferramenta se torna dispensável que todos os nós do *cluster* possuam as mesmas configurações de *hardware*. Outra vantagem da utilização desse *plugin* é a de que ele possibilita ainda, a estruturação de um *cluster* com múltiplas plataformas (Linux, Windows, etc.), contanto que elas estejam utilizando o VRJ;

✓ **Suporte para *drivers* de dispositivos em diferentes plataformas:** como o VRJ suporta vários sistemas operacionais diferentes, a existência de *drivers* adequados em cada plataforma, para um ou outro dispositivo de RV, já não é mais um problema. Um *cluster* devidamente configurado e integrado por diferentes sistemas operacionais, um nó pode sediar um dispositivo que possua apenas suporte no *Windows*, e compartilhar esse dispositivo no *cluster*, para ser acessado por um outro nó que esteja configurado com o SO Linux, mesmo que o dispositivo não possua um *driver* adequado para utilização nesse sistema; e

✓ **Distribuição dos dispositivos de entrada:** possibilita que os dispositivos de *hardware* utilizados, sejam distribuídos pelos nós do sistema, evitando assim que um determinado nó seja sobrecarregado com muitos dispositivos, transpondo inclusive a própria limitação física para ligações de dispositivos em um único microcomputador.

Durante a fase de estudo do CJ foram testadas configurações que distribuem dispositivos de entrada (como *mouse*) e a utilização de múltiplos *displays*. Por exemplo, em um dos testes efetuados, utilizou-se um *cluster* de quatro máquinas. Um nó controlava o dispositivo de entrada, outros dois eram responsáveis pelo *display* do mundo virtual, sendo que cada nó era dedicado a ângulos de visão diferentes, e o último nó, ficou encarregado da sincronização do sistema. Essa é uma configuração típica de sistemas imersivos do tipo caverna digital [9].

Porém, o sistema final do projeto em questão possuirá, a princípio, apenas um *display* de saída. Uma das limitações da versão atual de CJ, é que ele somente está preparado para suportar múltiplos *displays.* Para utilizar as ferramentas do CJ, é necessária a existência de uma cópia do mundo virtual inteiro em cada máquina de visualização.

5. DIFICULDADES ENCONTRADAS

Inicialmente, não se tinham as informações sobre os algoritmos existentes para a renderização paralela. Para superar esse problema, foi realizada uma revisão bibliográfica das suas principais abordagens. O estudo da aplicação CJ permitiu a equipe montar e testar um pequeno *cluster* de quatro nós. Esse *cluster* foi capaz de mostrar o ambiente virtual em diferentes pontos de vista, de forma que cada nó ficou responsável pela exibição de uma câmara distinta. Além disso, um nó do *cluster* ficou responsável pelo dispositivo de entrada que, no caso, foi o mouse. Entre as várias dificuldades técnicas superadas para o montagem e teste do *cluster*, pode-se citar a correta configuração da rede em Linux.

Para o correto desempenho dos equipamentos utilizados, foi necessário fazer ajustes nos dispositivos de *hardware* como a placa gráfica e controladora de discos RAID. A placa gráfica de alto desempenho WildCat4 7110, necessita de um *driver* especial que pode ser obtido na *homepage* da 3DLabs (www.3dlab.com). Porém, a instalação e configuração desses *drivers* não representam tarefas triviais no Linux.

A placa controladora de discos em RAID Sil3112 apresenta incompatibilidades com o *driver* padrão instalado pelo RedHat 9.0. Foi necessário obter o *driver* do fabricante para resolver problemas de desempenho dos discos.

Com relação à biblioteca VRJ, a mesma é de fácil instalação. É necessário obter da Internet o arquivo no formato *tar.gz*, e descompactá-lo. A fim de assegurar o correto funcionamento dessa ferramenta, e possibilitar a utilização *VRJconfig*, uma interface gráfica do VRJ é capaz de estruturar e alterar arquivos de configuração. Por outro lado, a biblioteca OSG depende de outras bibliotecas para o seu correto funcionamento, são elas: o *OpenThreads* e o *Producer*. Na *homepage* da OSG existem vários arquivos para serem copiados localmente, embora seja conveniente copiar apenas o arquivo que já contenha as três bibliotecas juntas. Basta então compilar todas as bibliotecas e instalá-las usando o comando *make*. Finalmente, para o correto desenvolvimento da aplicação, foi necessário configurar as variáveis de ambiente utilizadas pelas bibliotecas no arquivo */etc/profile* para não se ter problemas no momento de compilar e "linkar" as aplicações.

Durante as pesquisas realizadas sobre os dispositivos, uma das principais dificuldades foi a de se achar documentações adequadas que comprovassem a existência de *drivers* para Linux, e se esses funcionariam adequadamente. Para a escolha entre os

dispositivos de *hardware* apresentados na introdução deste trabalho, eram necessárias ainda informações mais detalhadas sobre as capacidades técnicas de cada um, para se ter certeza de que o escolhido satisfaria todas as necessidades contempladas no projeto.

6. CONSIDERAÇÕES FINAIS

No decorrer do trabalho foram estudadas diferentes técnicas de renderização paralela. Esta área de pesquisa, inicialmente, foi destinada aos sistemas dedicados ou computadores de memória compartilhada. Posteriormente, devido ao incremento do poder de processamento de microcomputadores de escritório, os pesquisadores abordaram o problema de renderização paralela em *cluster* de PCs. A principal vantagem do *cluster* de PC, é que estes permitem o desenvolvimento de sistemas interativos de baixo custo comparados aos sistemas dedicados. O levantamento bibliográfico permitiu adquirir o conhecimento necessário para implementação dessas técnicas em um *cluster* de PC.

É importante salientar que o êxito do projeto principal depende fortemente do desempenho do sistema implementado. Durante o desenvolvimento das atividades da Fase I do projeto (6 meses) patrocinado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), foram estudadas as ferramentas VRJ e OSG que permitem interagir com o mundo virtual. Ambas as bibliotecas, possuem uma ampla gama de classes e funções, para a configuração e gerenciamento de aplicações em RV.

Uma das atividades realizadas compreendeu o teste e configuração do CJ para um *cluster* de quatro PC, onde se observou a grande utilidade desta ferramenta para aplicações de tipo caverna digital. Porém foi constatada também a limitação dessa ferramenta ao utilizá-la com múltiplos *displays*, deixando para os programadores do sistema o trabalho de aplicar as técnicas para renderização paralela usando apenas um único *display*. Além disso, durante a primeira fase foram estudados os dispositivos que serão necessários para atingir a interatividade e imersão no ambiente virtual.

Dessa forma, está sendo proposta em uma segunda etapa, o estudo de maneiras adequadas para integrar algoritmos entre as bibliotecas de desenvolvimento CJ e OSG, a realização de testes de desempenho do sistema construído, verificando se a taxa de quadros por segundo será adequada para ambientes iterativos, e a perfeita integração dos dispositivos e a aplicação.

AGRADECIMENTOS

Os autores agradecem a Professora Dra. Maria Estela V. de Paiva da Escola de Engenharia de São Carlos da Universidade de São Paulo (EESC/USP), pelo auxílio com informações na área de Computação paralela. Queremos também, agradecer o suporte financeiro da FAPESP (processos: 02/07862-3 e 03/10954-0).

REFERÊNCIAS BIBLIOGRÁFICAS

[1] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A sorting classification of parallel rendering. *IEEE Comput. Graph. Appl.*, 14(4):23–32, 1994.

[2] N. Alves. *Serviços de pertinência para clusters de alta disponibilidade*. Master's thesis, IME-USP, 2004.

[3] R. Samanta, T. Funkhouser, and K. Li. *Parallel rendering with k-way replication*, Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics, p. 75–84, IEEE Press, 2001.

[4] W. T. Corrêa, J. T. Klosowski, and C. T. Silva. *Out-of-core sort-first parallel rendering for cluster-based tiled displays*, p. 89–96, Eurographics Association, 2002.

[5] Cientistas Associados. Projeto EnergiaRV. http://www.cientistasassociados.com.br/projetos/projects.htm [Acessado em 21/02/2005].

[6] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A sorting classification of parallel rendering, *IEEE Comput. Graph. Appl.*, 14(4): 23–32, 1994.

[7] E. E. Sutherland, R. F. Sproull, and R. A. Schumacker. *A characterization of tem hidden-surface algorithms*, ACM Comput. Surv., 6(1):1–55, 1974.

[8] S. Molnar, J. Eyles, and J. Poulton. *Pixel ow: high-speed rendering using image composition*, Proceedings of the 19th annual conference on Computer graphics and interactive techniques, p.231–240. ACM Press, 1992.

[9] VRJuggler. *Open Source Virtual Reality Tools*. http://www.vrjuggler.org/ [10/02/2005].

[10] *Introduction to the OpenSceneGraph*, http://openscenegraph.sourceforge.net/documentation/OpenSceneGraph/doc/introduction.html [Acessado em 22/02/2005].

[11] J. Allard, V. Gouranton, L. Lecointre, and E. Melin. *Net Juggler Guide*, Laboratoire d'Informatique Fondamentale d'Orléans - Université d'Orléans, 45067 Orleans Ce-dex2, France, Setembro 2002. http://netjuggler.sourceforge.net/Download/NetJuggler-HTML/index.html [Acessado em 22/02/2005].

[12] E. Olson. *Cluster juggler - pc cluster virtual reality*, Master's thesis, Iowa State University, 2002.

[13] T. W. Crockett and T. Orloff. *A mimd rendering algorithm for distributed memory architectures*, Proceedings of the 1993 symposium on Parallel rendering, p. 35–42. ACM Press, 1993.

[14] PACHECO, P. Parallel Programming with MPI. San Francisco: Morgan Kaufman,1997. 418 p.

[15] Delbem, A. C. B., "Restabelecimento de energia em sistemas de distribuição por algoritmos evolucionários associado a cadeias de grafo", Tese de Doutorado EESC-USP, 2002.