Um sistema de cache preditivo para o processamento em tempo-real de grandes volumes de dados gráficos

Sergio Pinheiro
Luiz Velho
Waldemar Celes

IMPA-Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina, 110, Rio de Janeiro,Brasil

Departamento de Informática, PUC-RIO
Rua Marquês de São Vicente, 225, Rio de Janeiro, Brasil

Abstract. Para visualizar objetos gráficos em tempo-real é necessário superar com dois tipos de problema: tempo limitado de para efetuar o processo de rendering e espaço limitado dos dispositivos de alta velocidade (memórias RAM e vídeo). O primeiro problema foi resolvido a partir da representação em multi-resolução dos dados gráficos. Para resolver o segundo problema foi utilizado um sistema de gerenciamento de memória preditivo baseado no modelo de memória virtual. Este trabalho propõe uma arquitetura que permite que qualquer tipo de dispositivo de armazenamento seja inserido. O funcionamento do sistema consiste em reservar um espaço de memória em cada dispositivo e gerenciar esse espaço de forma otimizada. Será mostrado um algoritmo de predição adaptativo específico para o problema de visualização. O sistema de gerenciamento de memória preditivo foi testado em aplicações de visualização em tempo-real de imagens de satélite e panoramas virtuais.

1 Introdução

Os dados bidimensionais são muito utilizados em várias aplicações de computação gráfica. Em geral, essas aplicações são desenvolvidas para permitir a visualização desses dados em tempo real, como por exemplo, visualização de panoramas, imagens e terrenos.

As aplicações de visualização em tempo-real devem manter uma taxa mínima e constante de quadros por segundo. Essa taxa deve ser alta o sufiente para que o usuário não perceba nenhuma latência. A dificuldade de se realizar esta tarefa está intimamente relacionada com a quantidade de dados que se deseja visualizar. Isto ocorre basicamente devido a dois fatores:

- No processo de visualização é necessário que os dados sejam carregados em memórias de alta velocidade que muitas vezes possuem um pequeno poder armazenamento.
- Para visualizar um determinado objeto é necessário realizar certos tipos de processamento, como cálculos geométricos e de iluminação. Esses cálculos são, em geral, bastante complexos.

Para resolver o primeiro caso, é necessário utilizar técnicas de gerenciamento de memória. Esses técnicas exploram informações com relação as características dos objetos gráficos e da aplicação para determinar quais os dados devem permancer na memória e quais devem ser eliminados. Além disso, com base nessas informações é possível carregar antecipadamente os dados que serão necessários num futuro próximo. A operação que determina o que deve ser carregado antecipadmanete é chamada de *predição*.

O segundo problema mencionado acima, onde é necessário reduzir a quantidade de processamento, pode ser resolvido através de técnicas de representação dos dados do objeto. Uma das técnicas mais utilizadas é técnica de representação em multi-resolução.

O objetivo deste trabalho é desenvolver um sistema de gerenciamento de memória baseado no modelo de memória virtual para visualizar em tempo-real uma grande quantidade de dados gráficos 2D. Esses dados são representados em uma estrutura de multi-resolução. A operação de predição é baseada no fato de que os dados visualizados possuem coerência espacial e temporal. Além disso, o sistema também leva em consideração que os dados gráficos são representados em multi-resolução e que em cada nível de resolução os dados serão subdivididos de forma regular. Essa subdivisão dos dados é que permitirá o processo de carregamento, onde cada subconjunto será carregado quando for necessário.

2 Trabalhos Anteriores

Hoje já existem placas gráficas oferecidas no mercado que possuem vários recursos de processamento gráfico. No entanto, a capacidade de armazenamento não evoluiu nesta mesma proporção. Assim, os trabalhos passaram a dar mais atenção ao problema de gerenciamento de memória.

A solução que é amplamente utilizada é baseada num recurso oferecido pelos sistemas operacionais chamado de *arquivos mapeados em memória* ("Memory-mapped files"). O sistema de arquivo mapeado em memória permite que uma região do espaço de endereço virtual seja mapeada em um arquivo de dados. A vantagem de utilizar esse recurso é a facilidade de implementação. No entanto, existem várias desvantagens em utilizar essa técnica como não ter controle sobre as operações de carregamento e restrições em relação ao particionamento dos dados e dispositivos que podem ser suportados. Os trabalhos [8, 9, 7] utilizam esse recurso para realizar operações de transferência de dados geométricos e o trabalho [3] utilizou para as operações de gerenciamento de memória para textura.

Com o objetivo de superar as restrições impostas pelos sistemas operacionais, surgiram trabalhos que desenvolveram sistemas de gerenciamento de memória com seu próprio sistema de paginação. Neste tipo de padadígma surgiram os trabalhos de Parajola [10], Falby [4] e Machado [2] que apresentaram sistemas de gerenciamento de memória adaptados para a visualização de terreno. O trabalho de Tanner [1] apresentou um sistema de gerenciamento de memória paginado que permitia a visualização de grandes texturas representadas com a técnica de mipmap [13]. Devido a isto, Tanner chamou a sua técnica de *clipmap*. O trabalho de Funkhouse [5] apresenta um sistema de gerenciamento de memória segmentado para a visualização ambientes virtuais. Esse sistema é adequado porque cada objeto do ambiente virtual necessita de um quantidade diferente de memória para armazenar suas informações geométricas.

As técnicas propostas nos trabalhos citados acima possuem duas desvantagens. A primeira é que as melhores soluções para o problema de carregamento preditivo são propostas para aplicações bastante específicas. Outra é em relação a arquitetura dos sistemas de gerenciamento que possuem uma estrutura rígida que restringe a quantidade e os tipos de dispositivos que podem ser manipulados.

Este trabalho propõe uma nova arquitetura para o sistema de gerenciamento de memória e uma nova técnica de predição. A arquitura proposta neste trabalho possui as seguintes características:

- A arquitetura do sistem é baseada no modelo de memória virtual que permite ao sistema gerenciar diversos tipos de unidades de armazenamento, como memória de vídeo, RAM, Disco, disco remoto e etc.
- A interface e as operações de transferência de dados entre os estágios não estão relacionadas com nenhum tipo específico de dado. Essa parte do sistema é completamente independente da aplicação ou do dado gráfico. Os sistemas de paginação podem ser implementados com considerando o grau de dependência com a aplicação e com o dado gráfico (Figura 1).
- A arquitetura do sistema permite que um número ilimitado de estágios sejam inseridos no sistema. Os dispositivos são organizados seqüencialmente em estágios.

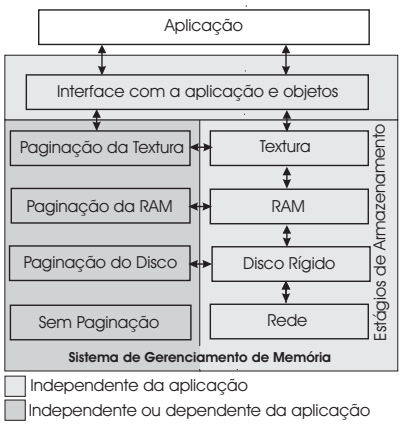

Figura 1: A arquitetura geral do sistema de gerenciamento de memória.

Em relação à técnica de predição pode-se destacar as seguintes características:

- O cálculo dos parâmetros futuros da câmera são realizados a partir de equações de segunda ordem, ou seja, são estimadas a velocidade e a aceleração dos parâmetros. Além disso, o algoritmo de carregamento explora as informações sobre a taxa de transferência para determinar o nível de resolução que deve ser carregado.
- As operações de cancelamento são introduzidas no sistema para minimizar a perda de desempenho provocada pelos erros do sistema de predição.
- A criação de um conjunto de regras que asseguram a consistência entre as operações de liberação e carregamento de dados.

3 Multi-Resolução

Um objeto gráfico [6], do espaço euclidiano $\mathbb{R}^n$, por um suporte geométrico $\mathbf{U} \subset \mathbb{R}^n$ e uma função $f : \mathbf{U} \rightarrow \mathbb{R}^p$. O símbolo utilizado para denotar um objeto gráfico é $\mathcal{O}(\mathbf{U}, f)$. Um objeto gráfico 2D tem um suporte geométrico descrito por uma função g cujo domínio é um subconjunto do $\mathbb{R}^2$.

No processo de representação o objeto gráfico é decomposto em partições uniformes e em seguida é representado em vários níveis de resolução. Para tornar a operação de representação mais intuitiva, objeto gráfico $\mathcal{O}$ será denotado por $\mathcal{O}(\mathbf{S}, z, \mathcal{I})$, onde $\mathbf{S}$ é uma superfície e a função z descreve um mapeamento da imagem $\mathcal{I}(\mathbf{U}, f)$ sobre a superfície $\mathbf{S}$.

Na representação por decomposição os suportes geométricos $\mathbf{S}$ e $\mathbf{U}$ são particionados, tal que cada elemento $S_{i,j}$ e $U_{i,j}$ seja retangular e de mesmo tamanho. Cada partição do objeto gráfico $\mathcal{O}$ é

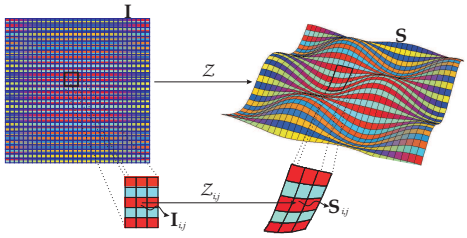

Figura 2: Decomposição retangular uniforme de um objeto gráfico.

definida por $\mathcal{T}(S_{i,j}, z_{i,j}, \mathcal{I}_{i,j})$, onde $z_{i,j}$ é uma função que mapeia os atributos da imagem $\mathcal{I}_{i,j}(U_{i,j}, f_{i,j})$ na superfície $S_{i,j}$. Essa partição $\mathcal{T}$ é chamada de *ladrilho* ("Tile"), como mostra a Figura 2.

Para facilitar a construção dessa estrutura será assumido que as dimensões da matriz de ladrilhos é $2^n \times 2^m$. Na estrutura em multi-resolução cada ladrilho pode ser unicamente identificado de acordo com a sua posição na matriz e o nível de resolução onde a matriz se encontra, assim cada ladrilho será identificado como $T_{i,j,k}$, onde (i, j) representa a posição na matriz e k o nível de resolução (Figura 3).

Para diminuir o espaço ocupado pela estrutura em multiresolução, durante o processo de construção apenas os ladrilhos filhos que acrescentarem informação visual em relação a textura do seu ladrilho pai é que possuirão texturas de maior resolução associadas a eles, caso contrário, suas texturas são eliminadas e a textura do ladrilho pai é associada de forma adequada aos ladrilhos filhos. Este método de representação é chamado de representação em multi-resolução adaptativa (Figura 4).

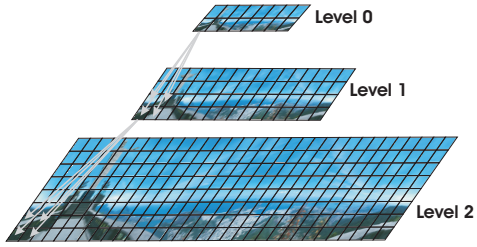

Figura 3: Estrutura em multi-resolução completa.

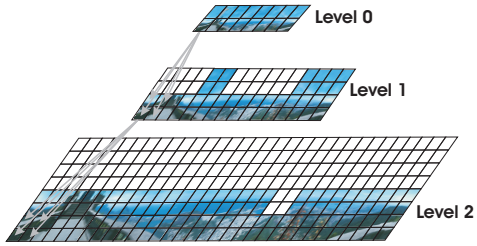

Figura 4: Estrutura em multi-resolução adaptativa.

Na fase de implementação, cada ladrilho é descrito por uma estrutura que contém quatro campos (poteiros para geometria, textura, ladrilho pai e ladrilhos filhos). O primeiro nível de resolução dá origem a uma floresta de quad-trees. Note que agora existem duas formas de acessar um determinado ladrilho na estrutura. A primeira é pelo índice (i, j, k) e a outra é pela quad-tree (veja a Figura 5).

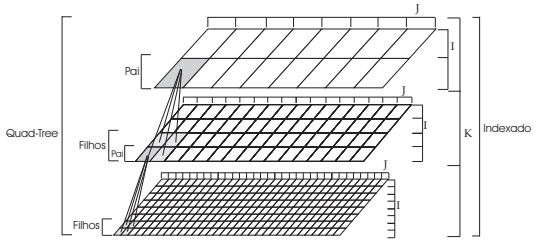

Figura 5: Os ladrilhos podem ser acessados de forma indexada ou a partir da quad-tree.

4 Gerenciamento de Memória

O sistema de gerenciamento de memória para o processo visualização de objetos gráficos foi inspirado no modelo de memória virtual[12][11]. No entanto, é necessário estender esse modelo para permitir que o sistema seja flexível e escalável.

A extensão foi realizada a partir da generalização o conceito de memória primária e secundária. A memória primária é todo tipo de dispositivo de armazenamento que tem taxa de transferência mais rápida e menor capacidade de armazenamento do que o dispositivo de armazenamento com o qual que está trocando informações diretamente. O dispositivo mais lento e com maior capacidade de armazenamento é considerado como a memória secundária, a Figura 6 mostra a relação entre os dispositivos de armazenamento.

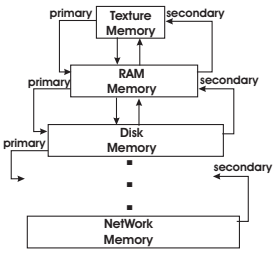

Figura 6: Sequência dos estágios de armazenamento.

Os principais módulos do sistema são os seguintes:

- **Estágios de Armazenamento**: Representa os dispositivos de armazenamento. Esse módulo especifica uma interface comum para as operações de escrita, leitura e alocação de dados nos dispositivos. Essa interface permite que qualquer tipo de dispositivo de armazenamento seja inserido no sistema.
  - **Unidade de Transferência de Dados**: É responsável pela transferência de dados entre os estágios de armazenamento. Esse módulo define a forma como os dados são transferidos. Os dados podem ser transferidos de forma assíncrona ou síncrona.
- **Sistema de Paginação**: Compreende as operações de carregamento, liberação e tratamento de ausência de páginas. Cada estágio de armazenamento possui seu próprio módulo de paginação.
- **Interface**: Trata da comunicação entre a aplicação e o sistema de gerenciamento de memória. A interface permite realizar três operações: operação de sincronismo, operação de acesso aos dados e operações de troca de informações.

Na operação de sincronismo a aplicação informa ao sistema quanto tempo deve levar a operação de predição. As operações de acesso permitem a aplicação acessar os dados gráficos. A operação de troca de informação é utilizada para a aplicação fornecer dados que são utilizados na fase de predição (Figuura 7.

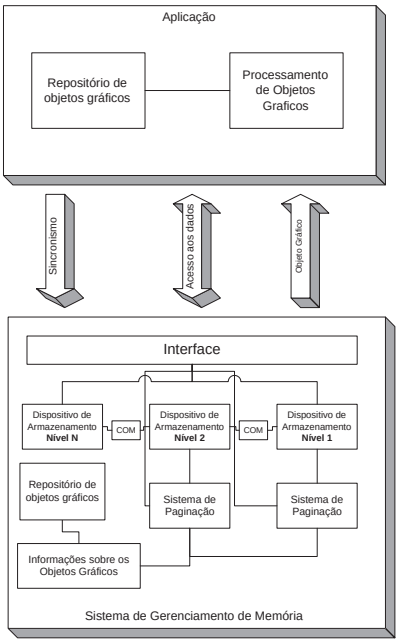

Figura 7: Arquitetura do sistema de gerenciamento de memória.

Os estágios de armazenamento possuem uma tabela de páginas. Essa tabela é utilizada para armazenar os mapeamentos entre endereços lógicos e os endereços físicos e o estado desses mapeamentos. O último estágio de armazenamento não possui tabela de páginas porque é capaz de armazenar todos os dados gráficos (Figura 8). Observe que só é necessário ter

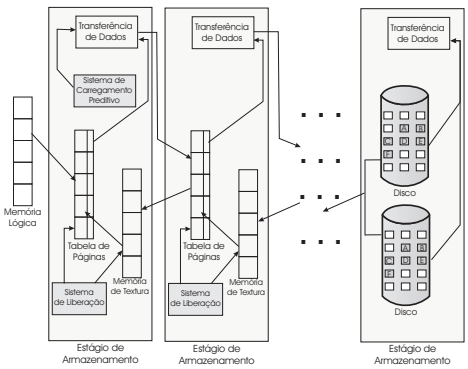

Figura 8: Vizualização do sistema de gerenciamento montado com os estágios.

um sistema de carregamento de página. Esse sistema está ligado ao estágio mais alto de armazenamento de dados. Quando o sistema de predição pede para carregar uma página de dados, esse pedido é passado para os diversos estágios de armazenamento, em forma de cascata, até que a página requerida seja encontrada. Uma vez que a página tenha sido encontra em um determinado estágio, esta é carregada, também em forma de cascata, em todos os estágios acima deste, até estar acessível no estágio desejado. Já o sistema de liberação de página é específico para cada nível de armazenamento. Isto é necessário porque a compexidade do algoritmo de liberação de páginas pode variar dependendo da capacidade de armazenamento de cada estágio.

O sistema divide as regiões de armazenamento em blocos de mesmo tamanho. O tamanho dos blocos de memória são determinados de acordo com o tamanho dos dados armazenados pelos ladrilhos. Uma página de dados é composta por um número finito de blocos. Por questões de otimização, cada página é dividida em quatro blocos. Assim, ladrilhos filhos são armazenados na mesma página. Cada ladrilho $\mathcal{T}(S_{i,j}, z_{i,j}, \mathcal{I}_{i,j})$ é recebe um endereço lógico (Figura 9).

Objeto	Página	Posição
8bits	22bits	2bits

Figura 9: Dois bits identificar o ladrilho filho, vinte e dois bits para identificar a página e oito bits para identificar o objeto gráfico 2D.

O espaço de endereçamento lógico é formado pelos endereços virtuais associados aos ladrilho. Se um ladrilho não tiver textura, este ladrilho receberá o endereço virtual associado a um ladrilho ancestral que tenha textura. A aplicação acessa os dados de cada ladrilho a partir desse sistema de endereçamento. A Figura 10 mostra como é formado o espaço de endereçamento lógico de um objeto gráfico.

5 Sistema de Predição

No processo de visualização o que determina o que deve ser mostrado na cena é a câmera virtual. Desta forma, as páginas são acessadas de acordo com as alterações dos valores dos parâmetros que definem a câmera virtual. Detectando como esses parâmetros estão variando sob o tempo, é possível prever o que será visualizado no futuro próximo.

A operação de predição é realizada em três passos. O primeiro consiste em determinar a região do objeto gráfico que possui a maior probabilidade

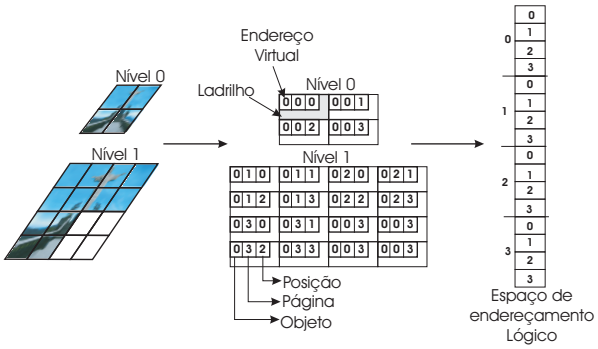

Figura 10: Construção do espaço de endereçamento lógico de um objeto gráfico 2D.

de ser acessada. Essa região é chamada de *área de carregamento*. O segundo é determinar o melhor nível de resolução que deve ser carregado. O terceiro consiste em determinar quais os pedidos de carregamento que devem ser cancelados. A região onde os pedidos são cancelados é chamada de *área de descarregamento*.

A área de carregamento é determinada pela união das janelas de visão calculadas nos quadros futuros (Figura 11). As janelas de visão são estimadas a partir dos cálculos de velocidade e aceleração dos parâmetros da câmera virtual. Supondo que o quadro atual seja i, o sistema de predição calcula os parâmetros da câmera para os quadros futuros $[i+1, i+2, \ldots i+n]$, onde n é o número de quadros que devem ser previstos. Em seguida, cada parâmetro futuro é passado para o objeto gráfico que retorna as coordenadas da janela de visão em relação ao seu sistema de coordenadas.

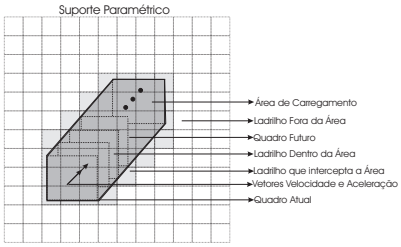

Figura 11: Os ladrilhos que estão dentro ou que interceptam a área de carregamento possuem uma probabilidade maior de serem requeridos pela aplicação de visualização e devem ser carregados.

O nível de resolução dentro da área de carregamento é escolhido considerando critérios da taxa de transmissão e de distância. O critério da taxa de transmissão é baseado no número de

ladrilhos por quadro que podem ser carregados para o estágio mais alto de armazenamento. Observe a Figura 12 e suponha que a taxa de transmissão seja de 25 *ladrilhos/quadro*, então $R_0 = \frac{4}{25} = 0.16$, $R_1 = \frac{11}{25} = 0.44$, $R_2 = \frac{27}{25} = 1.08$ e $R_3 = \frac{83}{25} = 3.32$. Assim, o nível escolhido por esse critério é o nível 2.

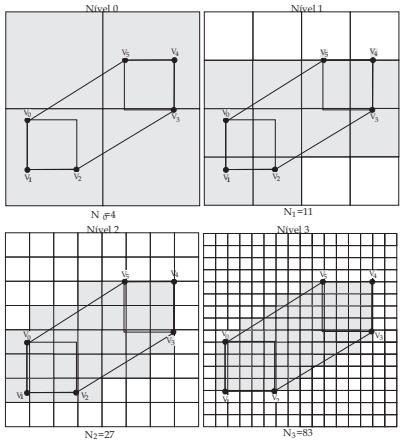

Figura 12: Os ladrilhos do nível 2 são escolhidos para carregamento.

A área de descarregamento é região determinada pela diferença entre área de carregamento calculada no quadro anterior e a área calculada no quadro atual (Figura 13. Os pedidos de carregamento dos ladrilhos que estão dentro dessa área são cancelados.

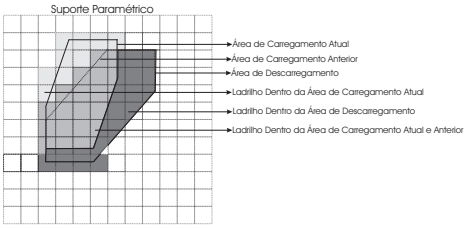

Figura 13: Os pedidos de carregamento dos endereços virtuais dos ladrilhos que estão dentro da área de descarregamento são cancelados.

5.1 Política de Carregamento

A política para o carregamento de páginas foi implementada com base nas seguintes regras:

*R*1**.** Para um objeto gráfico ser visualizado é necessário que as páginas das texturas do nível de mais baixa resolução sejam carregadas. Assim, é assumido que o dispositivo de armazenamento deve ser capaz de armazenar pelo menos o nível mais baixo de resolução da textura. A Figura 14 ilustra esta regra.

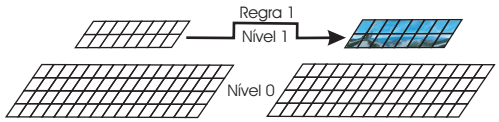

Figura 14: Ilustração da Regra $C1$.

*R*2**.** Se a textura de um ladrilho é carregada, as texturas dos seus irmãos também são. Esta regra é naturalmente atingida visto que uma página armazena as texturas dos ladrilhos que são filhos do mesmo ladrilho pai (Figura 15).

Figura 15: Ilustração da Regra $C2$.

*R*3**.** Se a textura de um ladrilho que está no nível k for carregada, então as texturas dos ancestrais deste ladrilho também deverão ser carregadas (Figura 16).

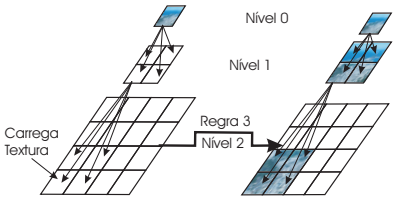

Figura 16: Ilustração da Regra $C3$.

Dado que o sistema de visualização seja configurado para desenhar q quadros por segundos e que o tempo gasto para se desenhar um quadro seja T_q segundos, onde $T_q < \frac{1}{q}$. Então o sistema de gerenciamento de memória tem o tempo de $T_c = \frac{1}{q} - T_q$ para realizar as tarefas de previsão, carregamento e liberação de páginas.

5.2 Política de liberação

O sistema de liberação de páginas decide retirar uma página de memória considerando as informações de

estado atual dos parâmetros da câmera, de velocidade e de aceleração desses parâmetros. A prioridade de uma página é determinada a partir da combinação do critério de ditância e nível de resolução (Figura 17). A página de contiver os ladrilhos do nível de maior resolução e mais distantes da janela de visão seram retiradas do estágio.

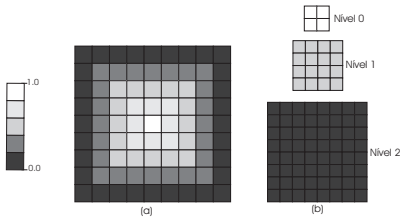

Figura 17: (a) Classficação dos ladrilhos baseado na critério de distância; (b) Classificação dos ladrilhos baseado no critério de profundidade.

O processo de liberação de páginas é executado obedecendo as seguintes regras:

*L*1. As páginas que contém as texturas do nível mais baixo de resolução não podem ser liberadas. Esta regra evita inconsistência durante a operação de tratamento de ausência de páginas.

*L*2. A textura de um ladrilho pai só poder ser liberada depois das texturas dos seus quatro filhos (Figura 18). Esta regra também evita inconsistência durante a operação de tratamento de ausência de páginas.

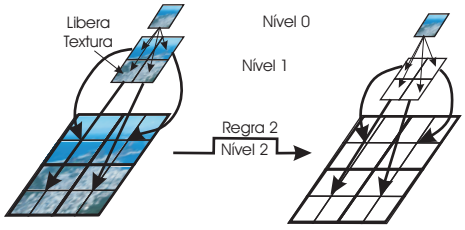

Figura 18: Ilustração da Regra *L*2.

*L*3. As páginas que contém as texturas que fazem parte do quadro atual não podem ser liberadas.

5.3 Tratamento de ausência de páginas

Apesar do sistema de carregamento de páginas se esforçar para carregar as páginas antes delas serem acessadas, podem ocorrer casos onde nem todas as páginas acessadas estejam carregadas a tempo.

O sistema de gerencimento não pode interromper o processo de visualização para carregar uma página ausente. Assim, este problema é resolvido explorando a estrutura de multi-resolução do objeto gráfico.

Quando um acesso a uma página ausente ocorre, o sistema varre a árvore de multi-resolução a procura de um ancestral desse ladrilho que está com sua textura carregada. A devido as regras $R1$ e $L1$ esse ancestral sempre será encontrado. Em seguida, a função de mapeamento de textura do ladrilho desejado é modificada para mapear apenas a região da textura do ladrilho ancestral representa a sua textura (Figura 19).

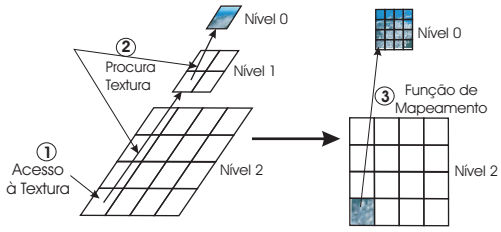

Figura 19: A região da textura ladrilho do ancestral mapeada na geometria do ladrilho desejado.

6 Aplicações

O sistema de gerenciamento de memória foi utilizado em aplicações de visualização de panoramas virtuais e de imagens de satélite (Figura 20). A janela em baixo de cada aplicação mostra o mapa de memória dos estágios de armazenamento gerenciados pelo sistema durante a execução do programa. Neste caso os dados estão sendo acessados localmente.

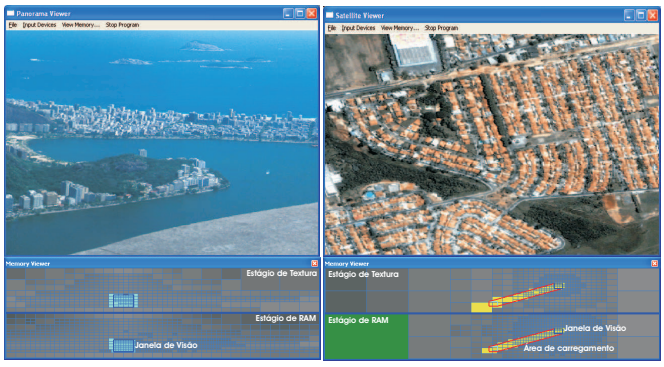

Figura 20: As aplicações de visualização em tempo-real de imagens de satélite e panoramas virtuais.

7 Conclusões

As principais contribuições deste trabalho foram:

- A criação e o desenvolvmento de um sistema de gerenciamento de memória baseado numa arquitetura flexível, escalável e independente da aplicação. A flexibilidade permite ao sistema incorporar vários estágios de armazenamento. Os estágios foram projetados para suportar vários dispositivos físicos de mesma natureza. Essa propriedade permite que o sistema seja distribuído.
- O estudo do problema de carregamento antecipado de dados (predição). Este trabalho propôs uma nova técnica de predição adaptativa para aplicações de visualização em tempo-real. A tarefa de carregamento é realizada levando em consideração o comportamento dos acessos da aplicação e a velocidade com que os dados podem ser carregados para o dispositivo desejado.
- A crição de um conjunto de regras que tornam as operações de carregamento e liberação de dados sempre consistentes.

Os experimentos com o sistema de predição mostraram que o sistema conseguiu se adaptar as variações de configuração de hardware e de espaço de armazenamento nos dispositivos. O sistema obteve mais de 95% de média de acerto.

Os cálculos de estimativa da área de carregamento mostraram que são sensíveis à situações de descontinuidade de movimento. De fato, esses cálculos são feitos com base nas equações do movimento. A descontinuidade de movimento provoca instabilidade nos cálculos da aceleração dos parâmetros da câmera.

Esse problema só ocorreu casos simulados e não chegou a comprometer a robustez do sistema de predição. No entanto, isto mostra que a técnica proposta neste trabalho possui algumas desvantagens.

A arquitetura hierárquica de armazenamento mostrou flexibilidade e provou que pode ser utilizada em aplicações diferentes. O sistema foi utilizado tanto na aplicação cliente como na aplicação de servidor. A aplicação de servidor foi capaz de lidar com as aplicações de visualização de imagens de satélite e de panoramas virtuais. Isto mostra que o sistema é pode ser independente do tipo da aplicação.

Referências

[1] T. Chistopher, M. Christopher, and J. Michael. The clipmap: A virtual mipmap. *SIGGRAF*, pages 151–158, July 1998.

[2] Enylton Machado Coelho. Visualização interativa de terrenos em um arquitetura cliente-servidor. Master's thesis, PUC-Rio, 1999.

[3] Jürgen Döllner, Konstantin Bauman, and Klaus Hinrighs. Texturing techniques for terrain visualization. *IEEE Visualization 2000*, pages 227–234, october 2000.

[4] John S. Falby, Michael J. Zyda, David R. Pratt, and Randy L. Mackey. NPSNET: Hierarchical data structures for real-time three-dimensional visual simulation. *Computers and Graphics*, 17(1):65–69, – 1993.

[5] Thomas A. Funkhouser, Carlo H. Sequin, and Seth J. Teller. Management of large amounts of data in interactive building walkthroughs. *Computer Graphics 1992 Symposium on Interactive 3D Graphics*, 25(2):11–20, 1992.

[6] Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1998.

[7] P. Lindstrom and V. Pascucci. Terrain simplification: A general framework for view-dependent out-of-core visualization. *IEEE Transaction on Visualization and Computer Graphics*, pages 239–254, May 2002.

[8] Peter Lindstrom. Out-of-core simplification of large polygonal models. *Proceeedings of ACM SIGGRAPH*, pages 259–262, July 2000.

[9] Peter Lindstrom and Valerio Pascucci. Visualization of large terrains made easy. *Proceeedings of IEEE Visualization*, pages 363–370, October 2001.

[10] Renato B. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In David Ebert, Hans Hagen, and Holly Rushmeie, editors, *IEEE Visualization '98*, pages 19–26, 1998.

[11] A. Silberschatz and P. Galvin. *Sistemas Operacionais*. Prentice Hall, 2000.

[12] Andrew Tanenbaum. *Operating Systems: Design and Implementation*. Prentice-Hall, 1987.

[13] L. Williams. Pyramidal parametrics. *SIGGRAPH*, 17:1–11, July 1983.