2
Trabalhos Relacionados

Como já foi mencionado anteriormente, para realizar as operações de visualização e processamento numa grande quantidade de dados gráficos é necessário unir técnicas de multi-resolução com técnicas de gerenciamento de memória.

O problema de gerenciamento de memória para dados gráficos é bastante antigo. Já em 1969, o trabalho desenvolvido por Norman [1] apresentava um sistema de gerenciamento de memória utilizado para realizar processamento de áudio com a transformada rápida de fourier.

As técnicas de representação de dados gráficos em vários níveis de resolução são mais recentes. No processo de visualização essas técnicas são utilizadas para diminuir a quantidade de dados que deve ser processada. Um dos primeiros trabalhos que utilizou a técnica de multi-resolução para visualização foi apresentado por Williams [2]. Esse trabalho apresenta uma técnica para a representação em multi-resolução de texturas. Essa representação é bastante conhecida e é chamada *mipmap*. Apesar de recente, já existe uma vasta literatura sobre multi-resolução. O trabalho de Mallat[22] aborda os problemas de representação em multi-resolução de sinais (imagens) e o trabalho de Garland[29] descreve das principais técnicas de multi-resolução adequadas para a representação topológica e a geométrica.

2.1
Técnicas de Multi-Resolução

O objetivo das técnicas de multi-resolução é reduzir a quantidade de dados que deve ser processada durante a fase de visualização. Essa técnica consiste em representar e armazenar os dados em vários níveis de detalhes, como mostra a Figura 2.1.

Pode-se definir uma família de algoritmos (ou operadores) para processar os dados, em geral malha ou textura, e representá-los a partir de

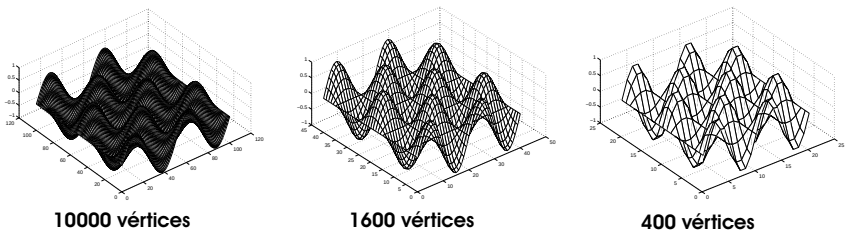

Figura 2.1: Superfície representada em três níveis de resolução

uma estrutura em multi-resolução. Quando esses algoritmos são aplicados nas malhas, eles têm a função de reduzir o número de polígonos a partir de operações de remoção, combinação e reestruturação. Essas operações são feitas de tal forma que visualmente o objeto ainda pareça o mesmo, no entanto, com um número reduzido de polígonos. No caso de processamento de textura, esses algoritmos realizam operações de filtragem e escalamento para se obter texturas de menor resolução.

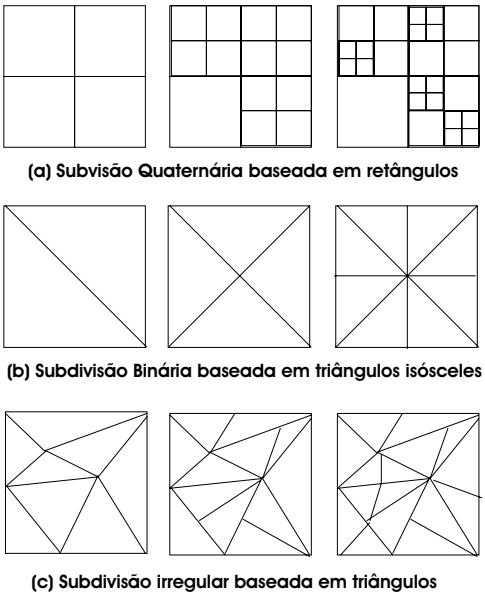

Figura 2.2: Malhas regulares e irregulares

Esses algoritmos podem ser classificados segundo três critérios [39][31]:

– **Tipo da Malha:** Os algoritmos podem ser desenvolvidos para processar grades regulares ou malhas com triangulação irregular(TINs). Os algoritmos baseados em grades regulares geram as malhas através de subdivisão quaternária ou binária (veja a Figura 2.2(a)(b)), enquanto que, os algoritmos baseados em TIN geram malhas com triângulos irregulares, como mostra a figura 2.2(c).

- **Tipo das Operações:** De acordo com esse critério as operações de simplificação ou refinamento da malha podem levar ou não em consideração os parâmetros da câmera. Os algoritmos que levam em consideração os parâmetros da câmera são chamados de *dependentes da câmera* (*"View-Dependent"*) e os que não levam são chamados de *independentes da câmera* (*"View-Independent"*).
- **Tipo de Multi-resolução:** Segundo esse critério, os algoritmos podem ser classificados como *estáticos* ou *dinâmicos*. Os algoritmos estáticos geram um número fixo de níveis de resolução. Os níveis são criados na fase de pré-processamento. Esses algoritmos aplicam operações de simplificação para se obter representações de menor resolução de uma determinada malha. Os algoritmos dinâmicos realizam operações de simplificação e refinamento em tempo-real, obtendo assim vários níveis de resolução de uma determinada malha inicial.

As desvantagens dos algoritmos baseados em grade retangular é que produzem malhas com um número de faces longe do ótimo e necessitam de operações especiais que evitam falhas ("crack") nessas malhas. As principais vantagens são que necessitam de uma estrutura de dados simples (árvore binária ou quaternária), as operações sobre a malha são fáceis de serem implementadas e tornam mais fácil a implementação de um sistema de gerenciamento de memória. Assim, surgiram muitos trabalhos de visualização em tempo-real de objetos gráficos 2D representados por grades regulares, dentre eles podemos citar [39, 6, 23, 10, 40, 14, 30].

A principal vantagem dos algoritmos baseados em TINs é que podem gerar triângulos irregulares, tendo assim, mais flexibilidade na construção das malhas. Essa flexibilidade permite que as malhas construídas possuam um número menor de vértices e que se aproximem mais da forma da malha inicial. As desvantagens desses algoritmos são que necessitam de uma estrutura de dados mais complexa, são mais difíceis de serem implementados e necessitam de uma grande quantidade de espaço para armazenamento. Além disso, torna-se mais complexo elaborar sistemas de gerenciamento de memória para lidar com esse tipo de estrutura. Assim, esses algoritmos foram pouco utilizados em aplicações em tempo-real que lidam com uma grande quantidade de dados. Os principais trabalhos que obtiveram bons resultados utilizando essa técnica de representação foram [8, 15, 11, 16, 24].

Os algoritmos estáticos constroem a estrutura em multi-resolução a partir de operadores *view-independent*. Isto ocorre porque os níveis de detalhes são pré-processados, ou seja, são criados antes da execução da

aplicação responsável pela visualização da malha de polígonos. A vantagem de utilizar os algoritmos estáticos é que proporcionam maior velocidade no processo de visualização. A principal desvantagem é que introduzem o problema chamado de *"popping"*. Esse problema ocorre quando o observador percebe que houve uma mudança entre os níveis de resolução de uma malha. Isso pode ser amenizado utilizando técnicas de morphing durante o processo de transição (veja [30, 25, 6]).

Os algoritmos dinâmicos tiveram um grande impacto nas aplicações em tempo-real. Esses algoritmos podem utilizar operadores *view-dependent* ou *view-independent*. Inicialmente os primeiros trabalhos desenvolvidos [11, 15, 8] utilizavam operadores *view-independent* para construir os níveis de resolução. No entanto, essa abordagem mostrou-se ineficiente nos processos de visualização porque muitas vezes gerava malhas com um número de polígonos maior do que o necessário e não levava em consideração a coerência espacial ou temporal na fase de construção das malhas. Devido a essas desvantagens, surgiram trabalhos que desenvolveram algoritmos dinâmicos que utilizavam operadores *view-dependent*. O objetivo principal era melhorar o desempenho das aplicações de visualização (veja [16, 24, 40, 42, 32, 14, 10]). A vantagem dos algoritmos dinâmicos é que eles evitam redundância de informações geométricas e topológicas (ao contrário dos algoritmos estáticos) e a desvantagem é que a implementação é mais complexa e requerem maior poder de processamento.

Os algoritmos utilizados para representar a geometria dos dados em multi-resolução podem ser ainda classificados de acordo com a forma como eles reduzem os detalhes de uma malha, se preservam a topologia da malha inicial e que tipo de métrica de erro (local ou global) é utilizada para decidir como simplificar a malha ou uma textura original. Para mais detalhes veja [17, 31, 33].

Este trabalho irá utilizar como exemplo principalmente imagens de satélites e panoramas virtuais. Esses objetos possuem uma geometria de baixa complexidade. Assim, optou-se por representar a geometria desses objetos utilizando multi-resolução estática, que pode ser facilmente implementada e permite que o processo de visualização seja realizado mais rápido.

A representação de uma textura em multi-resolução é feita a partir de algoritmos estáticos. Esses algoritmos utilizam operações de simplificação para obter níveis de menor resolução de uma textura inicial. A representação em multi-resolução de um textura consiste de uma seqüência de imagens de ordem descresente de resolução. Cada imagem é criada utilizando um

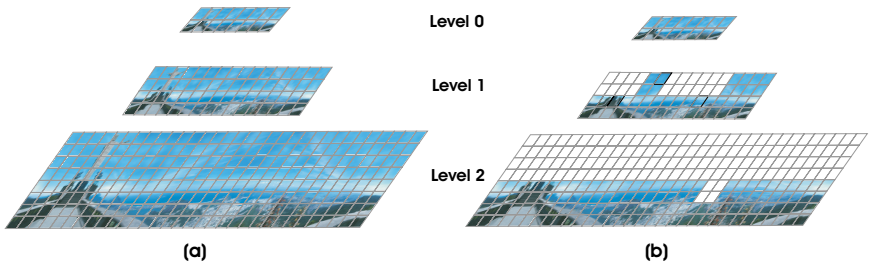

Figura 2.3: Representação em multi-resolução de uma textura; (a)Todos os níveis estão completos; (b)Eliminação das texturas dos ladrilhos que não acrescentam informação em relação ao nível de menor resolução.

fator de escala $\frac{1}{2}$ com relação a imagem predecessora (Figura 2.3(a)). Os dispositivos de armazenamento de alta velocidade (memórias de textura e RAM), em geral, não tem capacidade de armazenar todos níveis de textura. Esse problema é resolvido particionando os níveis de resolução em pequenas regiões. Essas regiões são chamadas de *ladrilhos* ("*Tile*"). Isto permite que apenas uma fração da representação em multi-resolução seja carregada para os dispositivos de alta velocidade.

Nos trabalhos [30, 32, 34] as texturas são representadas pelo método normal de multi-resolução, onde todos os ladrilhos possuem texturas. Ou seja, todos os níveis são completos. As vantagens desse tipo de representação é a simplicidade na estrutura de dados, a facilidade no cálculo do mapeamento de textura e a facilidade de implementação. As desvantagens é que consomem uma grande quantidade de recurso de armazenamento e de transmissão de dados. O trabalho [26] permite que os níveis de resolução possuam ladrilhos sem textura. No entanto, não fornece nenhum método intuitivo ou automático para gerar esses níveis.

O trabalho de Matos [25] baseou-se no fato de que existem regiões na textura que podem ser representadas em baixa resolução sem perdas visuais. Assim, baseado numa métrica de erro foi criado um método automático de eliminação de ladrilhos. Os ladrilhos que não acrescentavam informações visuais relevantes em relação a sua representação em baixa resolução são eliminados (Figura 2.3(b)). Essa métrica de erro é calculada apenas em relação ao nível anterior. Os erros cometidos na criação dos níveis mais baixos são esquecidos. Isso impossibilita um controle preciso do erro que é cometido na representação da textura. Em geral, a textura reconstruída a partir de sua representação em multi-resolução possui um erro maior do que o desejado.

O sistema desenvolvido neste trabalho constroi uma representação de textura em multi-resolução a partir do conceito de erro acumulado. O cálculo deste erro consiste em levar em consideração o erro cometido em todos os ladrilhos dos níveis anteriores que estão contribuindo para a construção do nível atual. Assim, garante-se que cada região da textura será reconstruída com um erro menor ou igual a um determinado valor que foi previamente especificado.

2.2 Técnicas de Gerenciamento de Memória

Inicialmente os sistemas de gerenciamento de memória foram desenvolvidos para que sistemas operacionais pudessem executar várias aplicações ao mesmo tempo sem a necessidade de carregá-las completamente no dispositivo de memória (veja [35][4]). O sistema de gerenciamento de memória mais conhecido e utilizado é o sistema de memória virtual.

O sistema de memória virtual baseia-se no simples conceito de mapear um grande espaço de endereço virtual em um pequeno espaço físico ou real de armazenamento. Geralmente a memória virtual é particionada em pequenos blocos. Esse particionamento da memória virtual permite que as operações de movimentação de dados na memória real sejam realizadas em apenas uma região da mesma. Essas operações de movimentação de dados são chamadas de *carregamento* quando os dados são movidos para a memória real e chamadas *liberação* quando os dados são retirados da memória real.

Os sistemas de memória virtual podem ser classificados dependendo da forma como o espaço de memória virtual é particionado. Assim, os sistemas de memória virtual são classificados em três grandes grupos [35][4]:

- **Sistema de memória paginado**: O espaço de memória virtual é divido em blocos de tamanhos fixos. Os blocos são chamados de *páginas* e as operações de movimentação de dados são chamadas de *paginação*.
- **Sistema de memória segmentado**: O espaço de memória virtual é divido em blocos de memória de tamanho variável. Neste caso, os blocos são chamados de *segmentos*.
- **Sistema de memória segmento-paginado**: O espaço de memória é divido em blocos de memória de tamanho variável e cada bloco é subdividido em partes de tamanho fixo.

Os sistemas de memória virtual realizam as operações de movimentação de dados apenas quando elas são necessárias. Esses sistemas, independentemente do tipo de partição que utilizam, são chamados de *sistemas de paginação sob-demanda*.

A maioria das aplicações que realizam operações de processamento ou visualização de dados gráficos conta com o sistema de memória virtual oferecido pelos sistemas operacionais. Neste caso as aplicações alocam espaço de memória suficiente para armazenar todos os dados. Se esse conjunto de dados é maior do que a memória física, então o sistema de gerenciamento de memória do sistema operacional gerencia a quantidade excedente. O principal problema é que essas aplicações perdem desempenho devido as operações de paginação. Isto ocorre porque se gasta mais tempo realizando operações de movimentação de dados na memória física do que realizando as operações realmente necessárias para se executar uma determinada tarefa de visualização. As principais desvantagens de se utilizar o sistema de memória virtual dos sistemas operacionais são:

- As operações de paginação são lentas. O tamanho dos blocos de memória não é adequado para armazenar os dados das aplicações de visualização.
- Os sistemas operacionais não oferecem o sistema de memória virtual para outros dispositivos de armazenamento além da memória RAM. Isso impossibilita que as aplicações possam utilizar outros dispositivos de armazenamento que estejam instalados localmente ou distribuídos em rede.
- O sistema de gerenciamento de memória não é preditivo.
- A aplicação não tem controle da forma como é particionado o espaço de endereçamento e também não pode controlar o tempo que deve ser gasto com paginação e com processamento.

Os primeiros trabalhos de visualização em tempo-real [10, 14, 11, 16, 24, 8, 15, 7] estavam mais preocupados em apresentar novas técnicas para acelerar o processo de visualização. Essa preocupação era decorrente do pequeno poder de processamento das placas gráficas da época. Assim, as técnicas de visualização visavam obter a melhor malha possível dentro das restrições de tempo exigidas pelas aplicações em tempo-real. Como não utilizam um sistema de gerenciamento de memória apropriado, não se comprometiam em visualizar dados que não pudessem ser armazenados na memória principal. Com o objetivo de amenizar esse problema alguns trabalhos desenvolveram técnicas e estruturas de dados que permitem reduzir o consumo de memória.

Por exemplo, o trabalho [39] utiliza técnicas de compressão de dados em memória.

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. Esse arquivo de dados deve estar armazenado em alguma unidade de disco. Uma vez feito o mapeamento, os dados do arquivo são acessados como se estivessem na memória principal. A vantagem de utilizar esse recurso é a facilidade de implementação. Em geral, só são necessárias três linhas de código: uma para abrir o arquivo, uma para reservar uma região da memória principal e uma última para dizer qual região do arquivo que deve ser mapeada na memória reservada. No entanto, existem várias desvantagens em utilizar essa técnica:

- A aplicação não tem controle sobre a operação de carregamento dos dados para a memória principal. Como é o sistema operacional quem decide isto, não é possível desenvolver algoritmos que antecipam o carregamento de páginas.
- A aplicação não tem controle sobre o particionamento dos dados, ou seja, a granularidade também é imposta pelo sistema operacional. No Windows, por exemplo, a granularidade é de 64Kbytes [18]. Em outras palavras, só permite alocar e acessar regiões do disco que sejam múltiplos dessa granularidade.
- Esse recurso é fornecido apenas para transferências de dados entre memória principal e unidades de discos.

Os trabalhos [36, 40, 42] utilizam esse recurso para realizar operações de transferência de dados geométricos e o trabalho [34] utilizou para as operações de gerenciamento de memória para textura. Esses trabalhos utilizam um sistema de carregamento bastante simples. O carregamento é feito calculando a posição da câmera para o próximo quadro. Em seguida, a posição do arquivo que contém os dados da região que será visualizada no futuro, é mapeada em memória. A operação de liberação é feita com a operação de desmapeamento. Intuitivamente, esse mecanismo de paginação consiste em mapear os dados que estão na direção de visão da

câmera e desmapear os dados que estão na direção oposta, como mostra a Figura 2.4(a). A desvantagem é que em casos onde a câmera muda bruscamente de direção as páginas que acabaram de ser desmapeadas devem ser mapeadas novamente (Figura 2.4(b)). No processo de mapeamento é realizada a operação alocação de memória e no processo de desmapeamento é realizada a operação de liberação. Se as operações de alocação e liberação forem realizadas repetidamente ocorrerá perda de desempenho da aplicação porque são operações computationalmente caras de serem executadas.

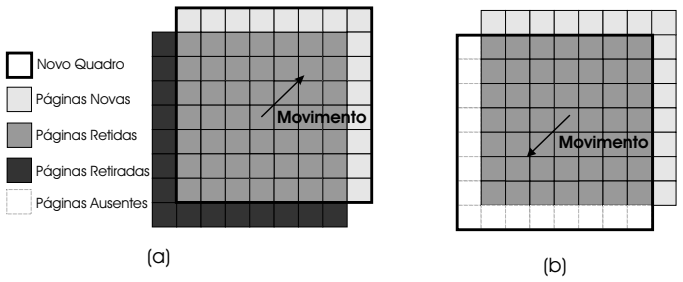

Figura 2.4: Mecanismo de mapeamento de arquivo em memória; (a) As regiões do arquivo que contêm dados vísiveis são mapeadas em memória; (b) O problema de ausência de páginas no caso de mudança de movimento.

Para diminuir o problema de perda de desempenho foram exploradas três estratégias:

1. Mapear grandes regiões de dados para diminuir as operações de alocação e liberação. A desvantagem é que torna necessária a utilização de máquinas com grande quantidade de memória RAM.

2. Realizar as operações de paginação em paralelo com as operações de visualização [34]. As operações de carregamento em paralelo são realizadas a partir de pedidos feitos pela aplicação. Quando o sistema carrega os dados para memória, ele notifica a aplicação que pode acessá-los. Nesse caso, a desvantagem é que haverá uma certa demora (dependendo do tipo de hardware utilizado) para visualizar os dados na resolução correta.

3. Armazenar os dados de forma apropriada para diminuir as operações de busca nos dispositivos mais lentos[36, 40, 42].

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. As operações de movimentação de dados foram implementadas levando em

consideração vários fatores como: estrutura de dados, a forma como os dados estão armazenados e a forma como os dados são acessados. Assim, é possível desenvolver sistemas de paginação mais sofisticados que possam predizer as páginas que devem ser carregadas na memória antes que elas sejam requisitadas pela aplicação. Também é possível determinar as páginas que não serão acessadas num futuro próximo. Essas páginas serão candidatas a serem retiradas da memória. Esses sistemas são chamados de *sistemas de gerenciamento de memória preditivos*. Os sistemas preditivos permitem que aplicações sejam executadas sem grande perda de desempenho devido às seguintes razões:

- Possuem inteiro controle das operações de carregamento e liberação de dados. Isto permite que a aplicação nunca seja interrompida em situações onde o processamento é crítico.
- Permitem que a página de dados seja configurada para obter o melhor desempenho dos dispostivos de armazenamento. Assim, as operações de carregamento são realizadas com maior eficiência.
- Possibilitam o desenvolmento de algoritmos de predição, determinam com maior precisão os dados que devem ser carregados ou retirados dos dispositivos. Desta forma, o tempo para visualizar os dados na resolução correta diminui.

Devido à pequena perda de desempenho, é possível desenvolver aplicações em tempo-real utilizando hardwares que são facilmente encontrados no mercado. Assim, surgiram vários trabalhos de visualização em tempo-real que propuseram sistemas preditivos e algoritmos de predição adequados ao tipo de aplicação de interesse.

Os trabalhos de Parajola [23], Falby [6] e Machado [30] apresentaram sistemas de gerenciamento de memória adaptados para a visualização de terreno. O trabalho de Blow[32] mostrou um sistema para visualização de dinâmica dos fluídos. Os sistemas de gerenciamento de memória são implementados como uma camada acima do sistema operacional com o objetivo de substituir seu sistema de paginação. Por exemplo, no trabalho de Blow[32] foi desenvolvido um mecanismo que permite que mais de 4GBytes de textura sejam acessados utilizando endereços de 32bits. No entando, esses trabalhos utilizam um sistema de predição simples baseado na velocidade dos parâmetros da câmera. Com base nessa informação são derminados dos dados que devem ser carregados para sintetizar o próximo quadro no nível de resolução ótimo. A desvantagem dessa abordagem é que ainda não resolve o problema de perda de resolução nos casos de movimentos rápidos de câmera.

Alem disso, só leva em consideração o fator espacial para determinar o nível de resolução que deve ser carregado. No caso da câmera estar se deslocando rapidamente sobre o objeto, o custo/benefício de carregar os dados dos níveis de alta resolução é mínimo.

O trabalho de Tanner [26] apresentou um sistema de gerenciamento de memória paginado que permitia a visualização de grandes texturas representadas com a técnica de mipmap [2]. Devido a isto, Tanner chamou a sua técnica de *clipmap*. A técnica de clipmap consiste em armazenar parcialmente o mipmap nos estágios mais altos de armazenamento (memórias de textura e RAM). A quantidade de memória utilizada pelo sistema é especificada pela variável *ClipSize*. A variável ClipSize define uma área retangular nos estágios de armazenamento de $ClipSize \times ClipSize$ pixels. Cada nível de resolução do mipmap que for maior que essa área é recortado (Figura 2.5(a)). A área definida pelo *ClipSize* é dividida em duas regiões. A primeira é a *EffectiveSize* que representa a região onde a textura pode ser acessada pela aplicação. A segunda é a *InvalidBorder* que armazena as texturas que são carregadas antecipadamente (Figura 2.5(b)). O endereçamento toroidal [12] é utilizado para acessar os níveis de textura que estão armazenados nos estágios. No endereçamento toroidal as texturas são carregadas na direção oposta ao movimento da câmera e são acessadas de forma circular (Figura 2.5(c)). Isto evita a operação de translação da textura em memória para dar espaço para a nova textura. Este trabalho implementou a técnica clipmap em hardware na placa gráfica. A vantagem é que ganha muita velocidade de processamento. A grande desvantagem é que seu funcionamento fica limitado a um tipo específico de hardware. O sistema de predição é basedo no algoritmo guloso que sempre tenta carregar a área *InvalidBorder* no nível ótimo de resolução. A vantagem de utilizar a área de segurança é que resolve o problema de perda de resolução com os movimentos bruscos de câmera. A desvantagem dessa técnica é que disperdiça banda de transmissão carregando texturas que certamente não serão utilizadas pela aplicação. Esse disperdício é responsável pela grande perda de resolução dos quadros quando a câmera se move com maior velocidade e pela demora do quadro ser sintetizado na resolução correta. Assim, essa técnica só produz bons resultados em dispositivos com altas taxas de transmissão e com as texturas armazenadas localmente.

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

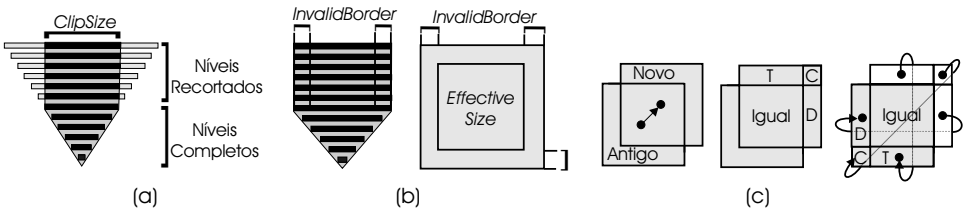

Figura 2.5: (a) O *ClipSize* defini o tamanho região do mipmap que ficará armazenada em memória; (b) O *InvalidBorder* defini a área de memória reservada para armazenar as texturas carrgadas pelo sistema de predição; (c) No endereçamento toroidal a textura do topo é carregada em baixo, a textura da direita é carregada na esquerda, a textura do canto superior é carregada no canto inferior.

geométricas. Esse sistema de gerenciamento foi desenvolvido especificamente para visualização de interiores de casas e edifícios. O ambiente virtual é dividido em células com base nas paredes e portas (Figura 2.6(a)). Cada célula possui um conjunto de objetos. Todo o mecanismo de paginação é construído com base nessa divisão espacial. A operação de carregamento é feita da seguinte forma: Dado que a câmera esteja posicionada em uma determinada célula, estima-se para cada célula do ambiente o tempo que levará para que ela seja visitada pela câmera. Em seguida, reserva-se memória para células visíveis ou que estão estão dentro do volume de visão. Todas as células que estão dentro de um tempo futuro máximo especificado são carregadas. O nível de resolução dos objetos dentro das células é calculado combinando-se o tempo com o fator de visibilidade (Figura 2.6(b)). As células que não são marcadas são retiradas da memória.

Os trabalhos de Hesina [27], Capps [37] e Schmalstieg [13] apresentaram sistemas de gerenciamento preditivos para a visualização de ambientes virtuais distribuídos em tempo-real. Os sistemas desenvolvidos consideram que o ambiente virtual e os objeto contidos nele são representados em multi-resolução. Nos três trabalhos os operadores de movimentação de dados são implementados apenas para lidar com dados geométricos. As operações de paginação são realizadas definindo-se uma região em volta da câmera. Essa região é chamada de *área de interesse* e é representada por uma esfera (Figura 2.7(a)). Todos os objetos do ambiente virtual que estão dentro da área de interesse devem ser transferidos do servidor para o cliente. A área de interesse é dividida em zonas. As zonas definem qual nível de resolução do objeto deve transferido (Figura 2.7(b)). Os objetos que estão fora da área de interesse são retirados da memória. A

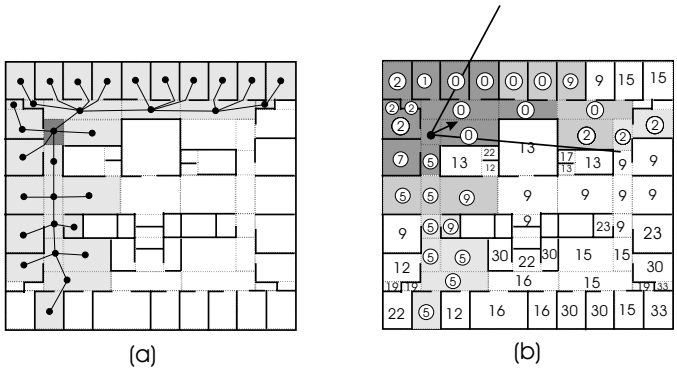

Figura 2.6: Técnica de gerenciamento de memória para interiores de edifícios e casas; (a)Árvove que representa a visibilidade de uma célula em relação as demais células; (b) As células são classificadas de acordo com o tempo que levará para a câmera visitá-las. O sombreamento das células mostra nível de resolução dos objetos que estão na memória (quanto mais escuro maior o nível de resolução).

vantagem desse mecanismo de paginação é que resolve o problema da câmera mudar de direção bruscamente. A desvantagem é que objetos são mantidos em alta resolução(zona3) mesmo sem serem visualizados pela câmera. O trabalho [41] tenta melhorar o sistema de paginação adicionando um sistema de monitoração. Esse sistema verifica quais são os locais mais visitados pelo cliente e dá uma prioridade maior de carregamento para os objetos desses locais.

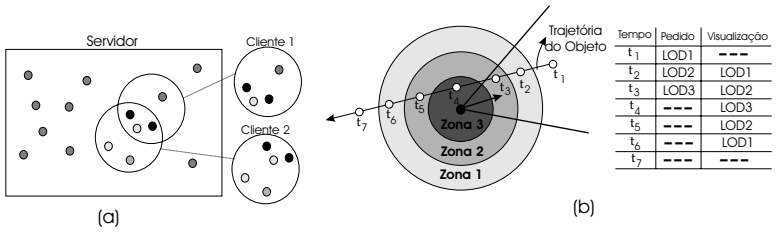

Figura 2.7: Mecanismo de carregamento de dados para ambientes virtuais;(a) Região esférica em torno do observador; (b) A região esférica é dividida em zonas. Essas zonas definem o nível de resolução do objeto que deve ser pedido e carregado na memória.

No trabalho de Matos *et al.* [25, 28] foi proposto um sistema de gerenciamento de memória para a visualização em tempo-real de panoramas virtuais. Esse sistema foi desenvolvido com o objetivo de realizar o gerenciamento de textura. O carregamento das páginas de textura é realizado explorando-se coerência temporal e espacial. No critério temporal calcula-se a velocidade dos parâmetros da câmera. A partir desse cálculo é

possível estimar o que será visualizado em quadros futuros. A prioridade de carregamento de cada página, que é necessária nos quadros futuros, é calculada de acordo com as posições estimadas da janela de visão (Figura 2.8(a)). Apenas esse critério gera resultados ruins no caso dos parâmetros da câmera sofrem variações bruscas. Adicionando-se o critério espacial (veja a Figura 2.8(b)), onde cada página recebe uma prioridade de acordo com a sua proximidade em relação a janela de visualização, resolve-se esse problema. As páginas de texturas que não pertencem aos quadros futuros e estão mais distantes da janela de visualização são candidatas a serem retiradas da memória. O principal problema é que o sistema não suporta texturas representadas em multi-resolução. Isso praticamente impossibilita visualizar grandes texturas.

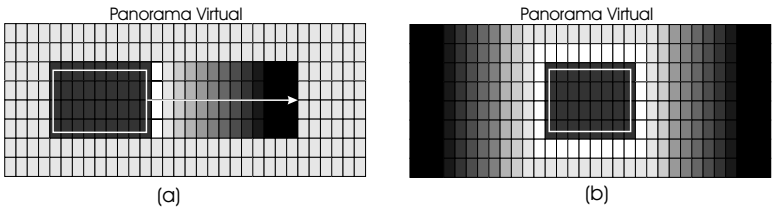

Figura 2.8: Mecanismo de predição para visualização de panoramas virtuais;(a) As páginas de texturas classificadas segundo o critério temporal; (b) As páginas de textura classificadas segundo o critério espacial (quando mais claro maior a prioridade da página).

Não se pode deixar de mencionar que existem sistemas comerciais que oferecem serviços de visualização de dados gráficos (mapas, terrenos, imagens) via internet. Um sistema bastante conhecido no momento é o *EarthViewer* [43]. O EarthViewer é um sistema formado por um servidor de dados de terreno e imagens de satélites de todo o globo terrestre. A consulta a esse banco de dados é feita a partir de aplicações clientes, chamadas de *EarthViewer3D*. O EarthViewer3D permite que os dados vindos do servidor sejam visualizados em tempo-real. Através de testes percebe-se perfeitamente que os dados contidos no servidor são representados em multi-resolução. Presume-se também que o sistema de carregamento de dados existente dentro da aplicação é simples, pois diminuindo-se o espaço em disco, onde ficam os dados baixados da rede, a aplicação demora muito para mostrar os dados na resolução correta. O *TerraServer* [44] é outro sistema que permite a visualização de mapas via internet. Assim como o *EarthViewer*, possui um grande banco de dados com informações geográficas do mundo inteiro. O serviço de consulta é fornecido a partir de uma página

na web. No entanto, a visualização dos dados não é feita em tempo-real e os dados são carregados sob-demanda.

2.3 Discussão Contextual

Fazendo uma análise dos trabalhos que foram apresentados chega-se as seguintes conclusões:

- As melhores soluções para o problema de carregamento preditivo são propostas para aplicações bastante específicas.
- Os sistemas de gerenciamento possuem um número fixo de unidades de armazenamento.
- As equações que estimam os parâmetros futuros da câmera são representadas por equações de primeira ordem. Ou seja, estimam apenas a velocidade dos parâmetros. Essa estimativa só é precisa quando os parâmetros estão variando de forma constante, algo que raramente acontece. Assim, é necessário definir uma região de segurança em torno do ponto central da câmera. Todos os dados dentro dessa região são carregados mesmo que não sejam visíveis. Essa solução para o problema consome mais memória e tempo de transmissão do que realmente é necessário.
- O cálculo que envolve a predição é um processo onde sempre existirá um erro inerente. Assim, apenas com operações de pedidos de carregamento é impossível evitar que dados que não são mais necessários sejam carregados para a memória. Isto aumenta ainda mais o desperdício de recursos de armazenamento e transmissão.

Este trabalho propõe um sistema de gerenciamento de memória baseado no modelo de memória virtual. Esse sistema possui as seguintes características:

- A arquitetura do sistem é baseada no modelo de memória virtual. Esse modelo cria um nível abstrato de acesso à memória, separando completamente o nível de memória lógica do nível de memória física. Esse modelo de abstração permite ao sistema gerenciar diversos tipos de unidades de armazenamento, como memória de vídeo, RAM, Disco, disco remoto e etc.
- A estrutura do sistema é composta por uma interface, estágios de armazenamento e sistemas de paginação. A interface e as operações

de transferência de dados entre os estágios não estão relacionadas com nenhum tipo específico de dado. Desta forma, essa parte do sistema é completamente independente do tipo da aplicação ou do tipo de objeto gráfico. Os sistemas de paginação podem ser implementados com considerando o grau de dependência com a aplicação e com o objeto gráfico (Figura 2.9).

– A arquitetura do sistema permite que qualquer tipo dispositivo de armazenamento seja inserido no sistema. Os dispositivos são organizados seqüencialmente em estágios. O sistema reserva um espaço de memória em cada dispositivo e gerencia esse espaço de forma otimizada.

– 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. A introdução do fator de aceleração torna desnecessária a utilização da área de segurança. 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. Essas operações evitam o carregamento dos dados que foram solicitados pelo sistema de predição.

– A consistência entre as operações de liberação e carregamento de dados é assegurada a partir de um conjunto de regras. Essas regras exploram a representação em multi-resolução para determinar um conjunto de dependências para efetuar o carregamento ou a liberação de uma página de dados.

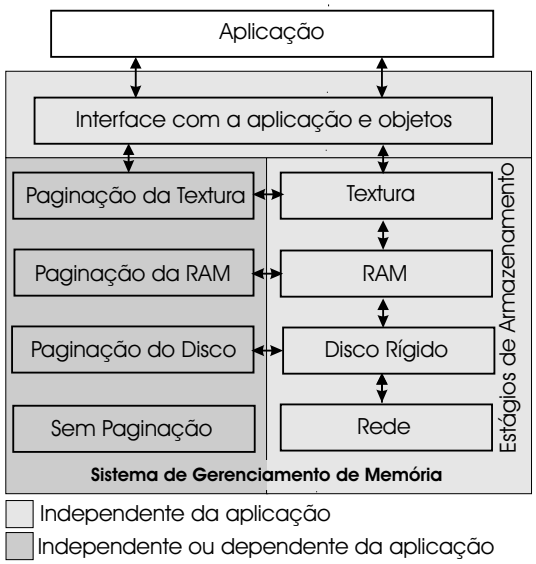

Figura 2.9: A interface, os estágios de armazenamento e as operações de movimentação de dados são independentes da aplicação. Os sistemas de paginação podem ser independentes ou dependentes da aplicação.