4
Decomposição e Multi-Resolução

Para processar os objetos gráficos a partir de sistemas computacionais é necessário representá-lo a partir de um número finito de parâmetros e de variáveis. A operação que transfere um objeto gráfico do universo matemático para o universo de representação é chamada de *discretização*. Intuitivamente, a operação de discretização é uma ponte que liga o universo matemático ao universo de representação. Essa operação associa ao objeto gráfico uma descrição finita de sua geometria, topologia e de seus atributos. O processo de discretização é feito em dois passos: representação do suporte geométrico e representação da função de atributos.

Existem várias técnicas de representação de objetos gráficos. Elas são descritas em [21]. Neste trabalho os objetos gráficos 2D são representados a partir da combinação das técnicas de decomposição e multi-resolução. A combinação das técnicas de decomposição e multi-resolução é essencial para resolver os problemas de gerenciamento de memória e visualização em tempo-real, respectivamente.

Por comodidade, as operações de discretização serão discutidas considerando que o objeto gráfico é descrito por uma única parte $\mathcal{O}(U, f) = \mathcal{O}(G, f)$. A extensão do processo de discretização para objetos gráficos descrito por múltiplas partes é feita de forma imediata, basta considerar que cada parte do objeto gráfico $\mathcal{O}(G_i, f_i)$ é discretizada independentemente.

No processo de discretização o objeto gráfico é decomposto em partições uniformes e em seguida é representado em vários níveis de resolução. As operações de discretização são realizadas em objeto gráficos $\mathcal{O}(\mathbf{S}, z)$, onde o conjunto $\mathbf{S}$ é descrito através de funções paramétricas e a função z descreve um mapeamento da imagem $\mathcal{I}(\mathbf{U}, f)$ sobre a superfície $\mathbf{S}$. A imagem $\mathcal{I}$ é chamada de textura e a função z é chamada de mapeamento de textura. Para tornar a operação de representação mais intuitiva, objeto gráfico $\mathcal{O}$ será denotado por $\mathcal{O}(\mathbf{S}, z, \mathcal{I})$.

Após a descrição de uma técnica de representação, será mostrado o pseudo-código que representa a sua implementação. Os nomes dos

algoritmos e das estruturas de dados estão em inglês devido ao estilo de programação adotado.

4.1 Decomposição do Objeto Gráfico

Dado um objeto gráfico $\mathcal{O}(\mathbf{S}, z, \mathcal{I})$, a representação por decomposição consiste em particionar os suportes geométricos $S = \bigcup_\lambda S_\lambda$ e $U = \bigcup_\lambda U_\lambda$, tal que os conjuntos S_i e U_i sejam disjuntos entre si, isto é, $S_{\lambda 1} \cap S_{\lambda 2} = \emptyset$ e $U_{\lambda 1} \cap U_{\lambda 2} = \emptyset$. Existem várias técnicas de decomposição do suporte geométrico[21]. Este trabalho assume que o suporte geométrico do objeto gráfico é contínuo e retangular. Desta forma, a técnica decomposição retangular uniforme é a forma mais natural de discretizar o suporte geométrico. A Figura 4.1 mostra o resultado do processo de discretização. Devido as características do objeto gráfico, esta decomposição pode ser armazenada a partir de matrizes. Assim, os elementos S_λ e U_λ podem ser unicamente identificados pela sua posição (i, j) na matriz. Logo podemos denominar cada S_λ e U_λ como $S_{i,j}$ e $U_{i,j}$, respectivamente.

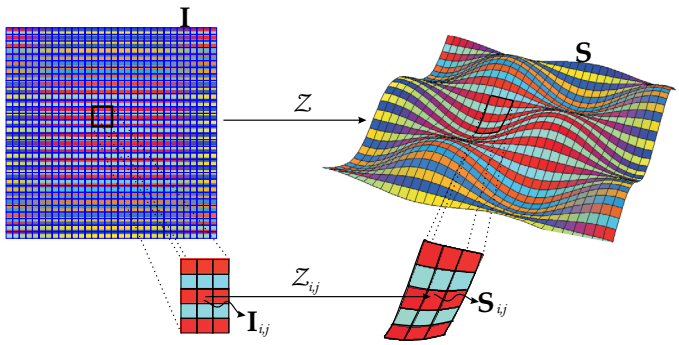

Figura 4.1: Decomposição retangular uniforme de um objeto gráfico.

A decomposição do suporte geométrico da imagem define de forma natural um particionamento da função f em várias funções de atributos $f_{i,j}$, onde cada função $f_{i,j}$ é associada a um elemento $U_{i,j}$. Como a função f cobre o suporte geométrico, pode-se definir a função $f_{i,j}$ como $f_{i,j} = f \mid U_{i,j}$. Definindo-se $\mathcal{I}_{i,j}(U_{i,j}, f_{i,j})$, tem-se que cada partição do objeto gráfico $\mathcal{O}$ é 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 $\mathcal{I}_{i,j}$ na superfície $S_{i,j}$. Essa partição $\mathcal{T}$ é chamada de *ladrilho*(“Tile”), como mostra a Figura 4.1. O pseudo-código 1 mostra a implementação da função de decomposição.

Algoritmo 1 TileMatrix Decomposition (**TileMatrix** $\mathcal{MAT}$, $nTilesW$, $nTilesH$, $\mathcal{O}(\mathbf{S}, z, \mathcal{I})$)

$\mathcal{MAT}$=CreateTileMatrix($nTilesW$,$nTilesH$) {Cria a matrix de ladrilhos}
for ($i = 0$; $i < nTilesH$; $i = i + 1$) **do**
  for ($j = 0$; $j < nTilesW$; $j = j + 1$) **do**
    $\mathcal{MAT}[i, j].S$ =CreateTileGeometry($i, j, \mathbf{S}$){Função que cria a forma $S_{i,j}$}
    $\mathcal{MAT}[i, j].I$ =CreateTileTexture($i, j, \mathcal{I}$){Função que cria a imagem $I_{i,j}$}
    $\mathcal{MAT}[i, j].z$ =CalculateTextureMap($i, j, \mathbf{S}, \mathcal{I}$){Função que define o mapeamento $z_{i,j}$}
  end for
end for
end function

A fase de decomposição do objeto gráfico está intimamente relacionada com desenvolvimento do sistema de gerenciamento de memória. No sistema de gerenciamento de memória cada ladrilho é tratado como a menor unidade de dados que pode ser transferida de um nível de armazenamento para outro. Devido a decomposição ser uniforme, o espaço de memória de cada nível de armazenamento é divido em blocos de mesmo tamanho. Cada bloco de memória é capaz de armazenar todos os dados referentes a um determinado ladrilho. E finalmente, como cada ladrilho pode ser identificado unicamente, é possível criar um sistema de endereçamento. Baseado nesse sistema pode-se criar interfaces para operações de acesso e carregamento de ladrilhos. Assim, as aplicações podem utilizar essa interface para acessar ou carregar um determinado ladrilho.

Note que o conceito de ladrilho é independente de um tipo específico de objeto gráfico. Conseqüentemente, o sistema de gerenciamento de memória também é independente, pois as suas operações e a sua forma de armazenamento são baseadas em cima deste conceito.

4.2 Multi-Resolução Adaptativa

Após o processo de decomposição, o objeto gráfico é representado por uma matriz de ladrilhos. A partir dessa matriz são criadas outras matrizes de menor resolução. Assim, obtém-se uma representação em multi-resolução do objeto gráfico. Essa estrutura em multi-resolução é construída segundo uma escala diádica, isto é, em potência de 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 $\mathcal{T}_{i,j,k}$, onde (i, j) representa a posição na matriz e k o nível de resolução.

Para construir uma matriz de ladrilhos que representa o nível k de resolução, utiliza-se a matriz de ladrilhos que está no nível $k + 1$. Cada ladrilho $\mathcal{T}_{i,j,k}$ é obtido a partir de operações de união e reamostragem dos ladrilhos $\mathcal{T}_{2i,2j,k+1}$, $\mathcal{T}_{2i,2j+1,k+1}$, $\mathcal{T}_{2i+1,2j,k+1}$, $\mathcal{T}_{2i+1,2j+1,k+1}$. O ladrilho $\mathcal{T}_{i,j,k}$ é chamado de *ladrilho pai* e os quatros ladrilhos que formam o ladrilho $\mathcal{T}_{i,j,k}$ são chamados de *ladrilhos filhos*. O suporte geométrico do ladrilho $S_{i,j,k}$ é obtido a partir da união dos suportes geométricos dos ladrilhos filhos, isto é, $S'_{i,j,k} = S_{2i,2j,k+1} \cup S_{2i,2j+1,k+1} \cup S_{2i+1,2j,k+1} \cup S_{2i+1,2j+1,k+1}$, como mostra a Figura 4.2. Em seguida, é feita a operação de reamostragem em $S'_{i,j,k}$ para se obter $S_{i,j,k}$, veja a Figura 4.2.

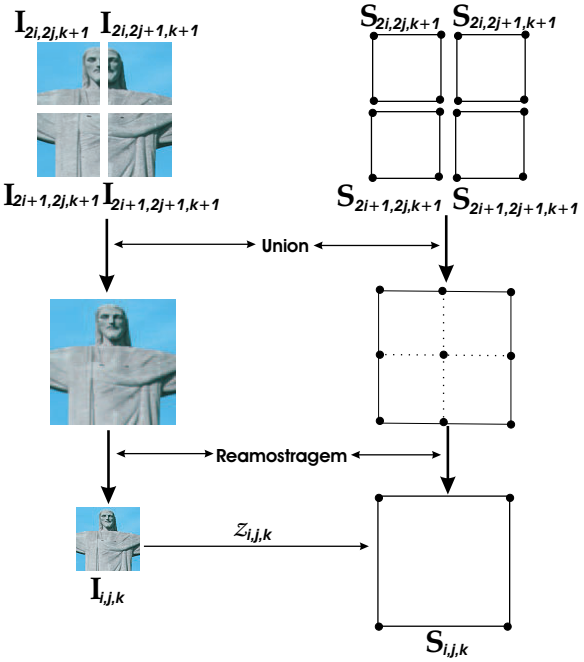

Figura 4.2: Construção da estrutura em multi-resolução de um objeto gráfico. A função de atributo representa um mapeamento de textura.

Para obter a textura $\mathcal{I}_{i,k,j}$ do ladrilho $\mathcal{T}_{i,j,k}$, primeiro realiza-se uma operação de união das texturas dos ladrilhos filhos obtendo $\mathcal{I}_{i,j,k} = \mathcal{I}_{2i,2j,k+1} \cup \mathcal{I}_{2i,2j+1,k+1} \cup \mathcal{I}_{2i+1,2j,k+1} \cup \mathcal{I}_{2i+1,2j+1,k+1}$. Em seguida, realiza-se uma operação de reamostragem para que a textura $\mathcal{I}_{i,j,k}$ fique com mesmo

número de amostras que as texturas dos ladrilhos filhos. A Figura 4.2 mostra como é feita a operação de se criar a textura de um o ladrilho de menor resolução. A Figura 4.3 mostra a representação em multi-resolução de uma panorama virtual. Supondo que o objeto foi representado com k níveis de resolução, então cada nível $k-t$, onde $0 < t \leq k$, é representado por uma matriz de ladrilhos de dimensão $2^{n-t} \times 2^{m-t}$.

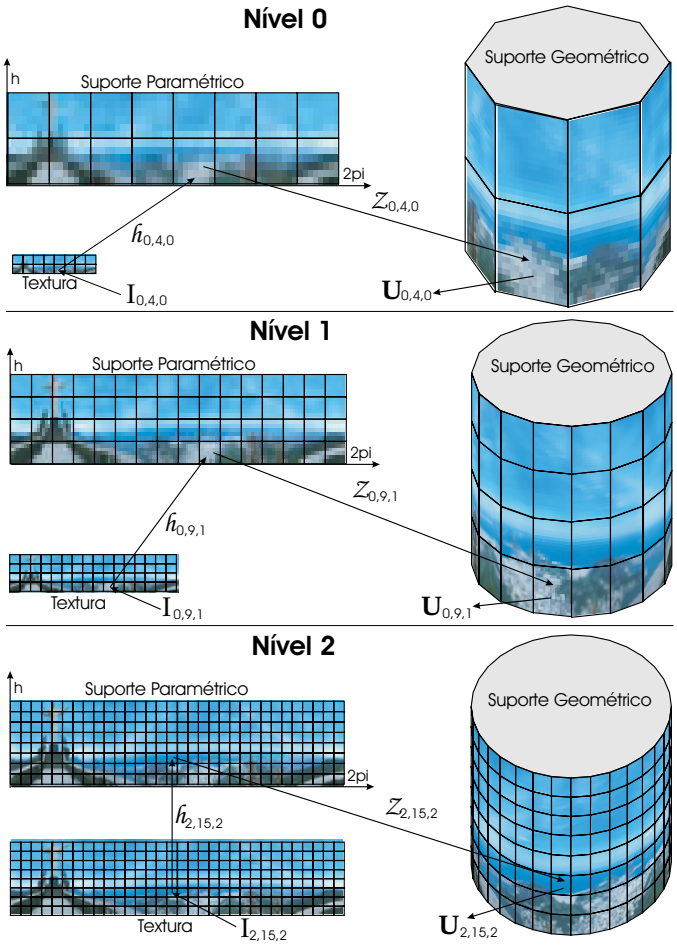

Figura 4.3: Representação em multi-resolução de uma panorama virtual do Rio de Janeiro.

Intuitivamente, a operação de união tem o objetivo de reconstruir uma região do objeto gráfico do nível mais alto de resolução. Enquanto que a operação de reamostragem é utilizada para se obter um nível de menor resolução dessa região. O pseudo-código 2 mostra a implementação da construção da estrutura em multi-resolução. Terminado o processo de construção, obtém-se duas estruturas em multi-resolução. Uma estrutura representa o suporte geométrico em diferentes níveis de detalhes. E a outra representa a textura em diferentes níveis de detalhes.

Algoritmo 2 NormalMutiResolution (**Matrix** $\mathcal{MAT}$, **MultiRes** $\mathcal{MR}_{tex}$, **MultiRes** $\mathcal{MR}_{geo}$, $nLevels$)

$\mathcal{MR}_{tex}$=CreateTexLevels($nLevels$)
$\mathcal{MR}_{geo}$=CreateGeoLevels($nLevels$)
$\mathcal{MR}_{tex}.Level[nLevels-1]$ =CreateTextureMatrix($\mathcal{T}.nTilesW$,$\mathcal{T}.nTilesH$)
$\mathcal{MR}_{geo}.Level[nLevels-1]$ =CreateGeometricMatrix($\mathcal{T}.nTilesW$,$\mathcal{T}.nTilesH$)
for ($i = 0$; $i < \mathcal{T}.nTilesH$; $i = i + 1$) **do**
  for ($j = 0$; $j < \mathcal{T}.nTilesW$; $j = j + 1$) **do**
    $\mathcal{MR}_{tex}.Level[nLevels-1, i, j].\mathcal{T}.I$=$\mathcal{MAT}[i, j].I$
    $\mathcal{MR}_{geo}.Level[nLevels-1, i, j].\mathcal{T}.S$=$\mathcal{MAT}[i.j].S$
  end for
end for
ntw=$\mathcal{MAT}.nTilesW$
nth=$\mathcal{MAT}.nTilesH$
for ($l = (nLevels - 2)$; $l \geq 0$; $l = l - 1$) **do**
  $ntw = ntw/2$
  $nth = nth/2$
  $\mathcal{MR}_{tex}.Level[l]$ =CreateTexMatrix(ntw,nth)
  $\mathcal{MR}_{geo}.Level[l]$ =CreateGeoMatrix(ntw,nth)
  for ($i = 0$; $i < nth$; $i = i + 1$) **do**
    for ($j = 0$; $j < ntw$; $j = j + 1$) **do**
      Texture Tile Processing
      I_{union}=TexUnion($\mathcal{MR}_{tex}.Level[l + 1, 2i, 2j].\mathcal{T}.I$, $\mathcal{MR}_{tex}.Level[l + 1, 2i, 2j + 1].\mathcal{T}.I$, $\mathcal{MR}_{tex}.Level[l + 1, 2i + 1, 2j].\mathcal{T}.I$, $\mathcal{MR}_{tex}.Level[l + 1, 2i + 1, 2j + 1].\mathcal{T}.I$)
      $\mathcal{MR}_{tex}.Level[l, i, j].\mathcal{T}.I$=TexResample($I_{union}$)
      LinkTexTile($\mathcal{MR}_{tex}.Level[l, i, j].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l + 1, 2i, 2j].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l + 1, 2i, 2j + 1].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l + 1, 2i + 1, 2j].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l + 1, 2i + 1, 2j + 1].\mathcal{T}$)
      Geometric Tile Processing
      S_{union}=GeoUnion($\mathcal{MR}_{geo}.Level[l + 1, 2i, 2j].\mathcal{T}.S$, $\mathcal{MR}_{geo}.Level[l + 1, 2i, 2j + 1].\mathcal{T}.S$, $\mathcal{MR}_{geo}.Level[l + 1, 2i + 1, 2j].\mathcal{T}.S$, $\mathcal{MR}_{geo}.Level[l + 1, 2i + 1, 2j + 1].\mathcal{T}.S$)
      $\mathcal{MR}_{geo}.Level[l, i, j].\mathcal{T}.S$=GeoResample($S_{union}$)
      LinkGeoTile($\mathcal{MR}_{geo}.Level[l, i, j].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l + 1, 2i, 2j].\mathcal{T}$, $\mathcal{MR}_{geo}.Level[l + 1, 2i, 2j + 1].\mathcal{T}$, $\mathcal{MR}_{geo}.Level[l + 1, 2i + 1, 2j].\mathcal{T}$, $\mathcal{MR}_{geo}.Level[l + 1, 2i + 1, 2j + 1].\mathcal{T}$)
    end for
  end for
end for
end function

Observe que existem as funções `LinkGeoTile` e `LinkTexTile` que ligam o ladrilho pai nos ladrilhos filhos. Essas ligações formam uma quadtree cuja raiz é o ladrilho que está no menor nível de resolução. Assim, 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 4.4).

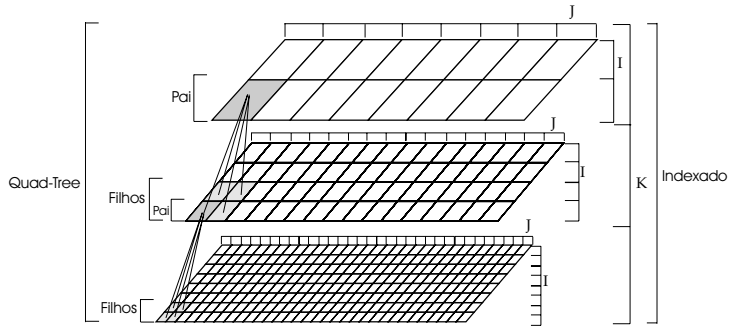

Figura 4.4: A estrutura em multi-resolução permite acessar os ladrilhos de forma indexada ou a partir do percorrimento na quad-tree.

Um dos problemas da representação em multi-resolução é que essa estrutura necessita de um grande espaço de armazenamento. Dado que a amostragem original ocupe M Bytes para ser amazenada. Para armazená-la em vários níveis de resolução são necessários $M+M\sum_{i=1}^{k}\frac{1}{4^i} \approx M+\frac{M}{3}$ Bytes, onde k é o número de níveis de resolução. Ou seja, a estrutura em multi-resolução ocupará um espaço $\frac{1}{3}$ maior do que o ocupado pela amostragem original. Para grandes valores de M, essa a quantidade a mais passa a ser relevante.

Um outro problema bastante relevante é com relação ao desempenho do sistema de gerenciamento de memória. Existem regiões na textura que possuem poucas informações de detalhes, ou seja, regiões de baixas freqüências. Essas regiões poderiam ser representadas apenas em níveis de baixa resolução sem alterar a qualidade visual da textura. O benefício de carregar e armazenar essas regiões nos níveis mais altos de armazenamento é mínimo. Isso prejudica o desempenho do sistema de gerenciamento, pois o tempo que poderia ser gasto carregando regiões que acrescentariam mais informações visuais é gasto em carregar essas regiões que não acrescentam nenhuma informação a mais que sua representação em menor resolução.

Para minimizar esses dois problemas, durante o processo de construção da estrutura, apenas os ladrilhos filhos que acrescentarem um certo nível de detalhe em relação a textura associada ao ladrilho pai é que possuirão texturas de maior resolução. Caso contrário, as texturas dos ladrilhos filhos são eliminadas e a textura do ladrilho pai é associada de forma adequada aos mesmos. Esse processo de eliminação é realizado em dois passos. Primeiro,

após a construção da textura do ladrilho pai, a sua textura é super-amostrada para ficar com o mesmo número de amostras que a textura obtida a partir da operação de união dos ladrilhos filhos, como mostra a Figura 4.5.

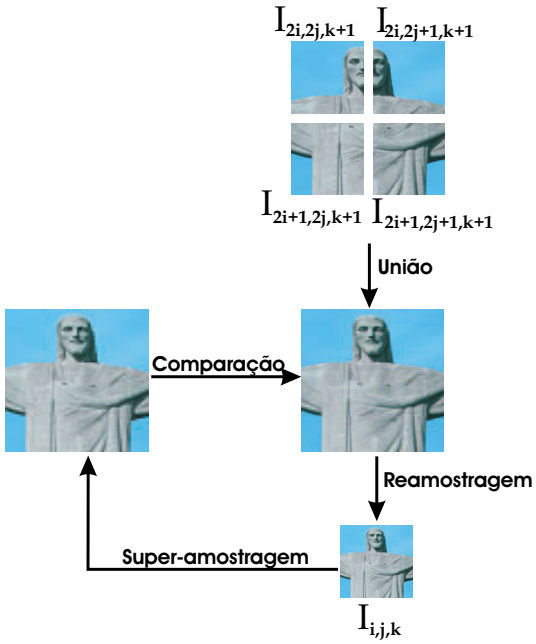

Figura 4.5: Método de construção da estrutura em multi-resolução adaptativa para objeto gráficos cuja função de atributo é um mapeamento de textura.

Em seguida, é feita uma comparação para ver se existe uma diferença significativa entre as duas texturas (Figura 4.5). Caso não haja diferença significativa entre as duas texturas, as amostras associadas aos ladrilhos filhos são eliminadas. Este processo de comparação pode ser realizado a partir da determinação do critério de erro e da especificação de um valor de tolerância com relação ao erro calculado. O critério de erro pode ser calculado utilizando-se várias técnicas de processamento de sinais[19]. Uma técnica bastante utilizada é calcular o erro a partir de transformadas. Essa técnica permite analisar as perdas de altas freqüências que ocorrem no processo de construção da textura do ladrilho pai.

No trabalho [25] a decisão de eliminar a textura de um ladrilho é baseada apenas no cálculo do erro entre a textura de uma nível k (ladrilho pai) e a textura obtida da união das quatro textura do um nível $k+1$ (ladrilhos filhos). Ou seja, são ignorados os erros que foram cometidos na construção dos ladrilhos do nível $k+1$ em relação aos ladrilhos do nível $k+2$, veja a Figura 4.6(a). Assim, algumas regiões da textura representada

em multi-resolução terão erros acumulados maiores do que a tolerância especificada. A desvantagem desse método de decisão é que não proporciona uma forma precisa de controlar o erro na fase de representação da textura. Para aplicações que lidam com imagens de satélite isso é inaceitável.

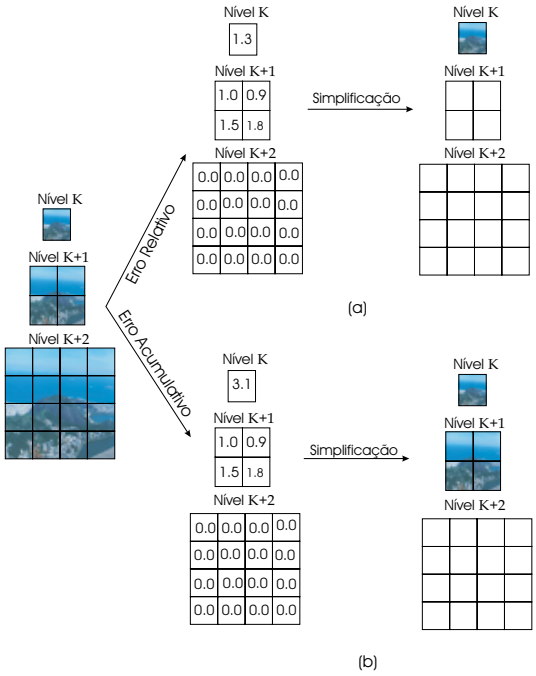

Figura 4.6: Especificando uma tolerância igual a dois; (a)Na simplificação com erro relativo as texturas do nível $k+2$ e $K=+1$ são eliminadas. A reconstrução da textura utilizando apenas a textura do nível k resultará num erro maior que o especificado; (b)Na simplificação com erro acumulativo apenas as texturas do nível $K+2$ são eliminadas. Assim, a textura será reconstruída utilizando o nível $k+1$, onde o erro médio é menor que a tolerância especificada.

Neste trabalho o método de decisão para a eliminação de texturas é baseado na técnica do erro cumulativo. Essa técnica consiste em armazenar em cada ladrilho o erro relacionado com a sua textura. Assim, o erro de um ladrilho pai de um nível k é obtido somando-se o erro cometido na construção de sua textura mais o erro acumulado nos ladrilhos filhos do nível $k+1$, como mostra a Figura 4.6(b). Esse erro acumulado pode ser definido como a média dos erros dos ladrilhos filhos ou como o erro máximo entre os ladrilhos filhos. Assim, sejam Err_{f1}^{k+1}, Err_{f2}^{k+1}, Err_{f3}^{k+1} e Err_{f4}^{k+1} os erros dos ladrilhos filhos do nivel $k+1$ e Err_{pf} o erro do ladrilho pai em relação aos seus filhos. Então o erro associado ao ladirlho pai poder ser

definido como

$$Err_p^k = Err_{pf} + \frac{Err_{f1}^{k+1} + Err_{f2}^{k+1} + Err_{f3}^{k+1} + Err_{f4}^{k+1}}{4}$$

ou como

$$Err_p^k = Err_{pf} + Max(Err_{f1}^{k+1}, Err_{f2}^{k+1}, Err_{f3}^{k+1}, Err_{f4}^{k+1}).$$

Se o Err_p^k for menor que a tolerância estipulada então as texturas dos ladrilhos filhos são eliminadas.

O problema de utilizar a média dos erros dos ladrilhos filhos é que pode acontecer de um ladrilho ter um erro muito alto e os outros três terem erros muito baixos. Neste caso, a média irá suavizar o erro acumulado nos ladrilhos filhos. Desta forma, existe a possibilidade das texturas serem eliminadas mesmo que o erro em um dos ladrilhos seja maior do que a tolerância desejada. Devido a esse problema, neste trabalho foi utilizada a medida de erro máximo. Note que com o cálculo de erro máximo qualquer ladrilho terá um erro associado sempre menor do que a tolerância desejada.

Após o processo de simplificação, utilizando a técnica de erro acumulativo, a estrutura em muli-resolução fica como mostra a Figura 4.7. Esse método de representação é chamado de *representação em multi-resolução adaptativa*. O pseudo-código 3 mostra a implementação dessa técnica de representação.

Observe que a multi-resolução adaptativa é feita apenas na textura do objeto gráfico. O processo para se representar a geometria em multi-resolução adaptativa consiste nas mesmas operações realizadas na textura. No entanto, no caso da geometria essas operações são mais complexas, pois envolve problemas topológicos que, em geral, são difíceis de se resolver. Como já foi mencionado, o objetivo deste trabalho não é lidar com problemas topológicos que sejam complexos. Além disso, existem várias técnicas de visualização em tempo-real que visualizam de forma adaptativa a geometria do objeto gráfico (veja [14], [33], [40] e [23]).

4.3 Função de Mapeamento de Textura

A função de mapeamento de textura é quem faz a ligação entre o suporte geométrico do objeto gráfico com os atributos de cor de uma imagem. Nas seções anteriores, foi mostrado como decompor e representar

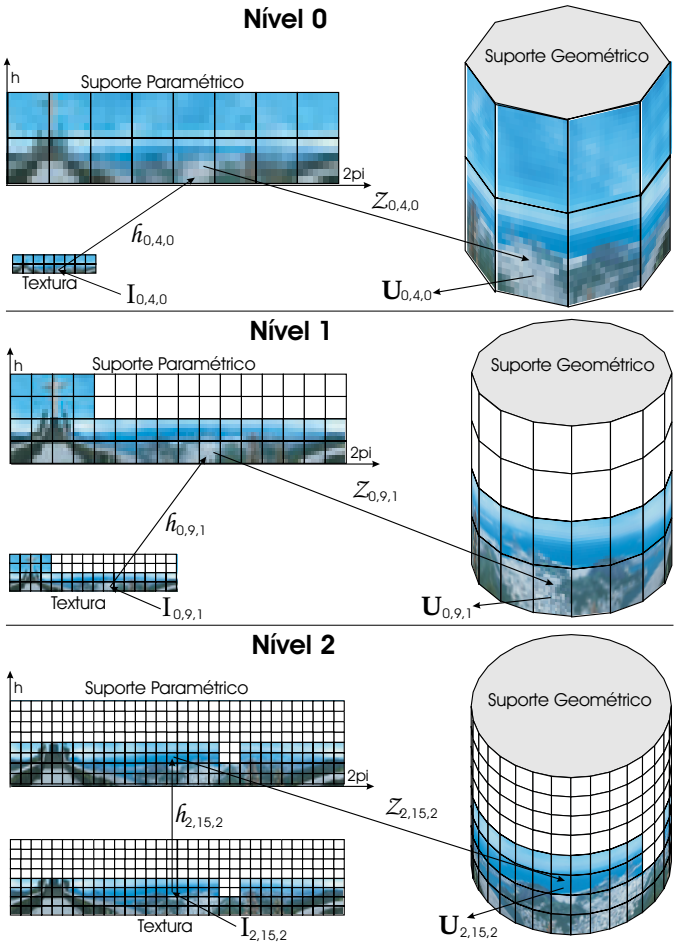

Figura 4.7: Representção em multi-resolução adaptativa obtida a partir do método de simplicação para textura.

em multi-resolução o suporte geométrico e a textura de um objeto gráfico 2D. Após a operação de representação, o objeto gráfico é formado por um conjunto de ladrilhos $\mathcal{T}_{i,j,k}(S_{i,j,k}, z_{i,j,k}, \mathcal{I}_{i,j,k})$. Agora, é necessário calcular a função $z_{i,j,k}$ que irá mapear a textura $\mathcal{I}_{i,j,k}$ na superfície $S_{i,j,k}$.

Foi mostrado no capítulo anterior, que se um objeto gráfico $\mathcal{O}(\mathbf{S}, z)$, onde S é uma superfície paramétrica descrita por uma função $j : \mathbf{V} \subset \mathbb{R}^2 \rightarrow \mathbf{S}$ e z é um mapeamento de textura, então a função z é definida por $z(p) = f(h^{-1}[j^{-1}(p)])$, $p \in S$, h é uma mudança de sistema de coordenadas do suporte da imagem para o suporte paramétrico de $\mathbf{S}$ e f é a função de atributos de cor da textura. Como as funções f e j são dadas e fixas para um determinado tipo de textura e superfície, então o que define a forma como a textura será mapeada na superfície é a transformação h. Desta forma, a função de mapeamento de textura z é caracterizada pela função h.

Neste trabalho, os suportes paramétricos do objeto gráfico e da textura são retangulares, logo a função h é descrita por uma transformação afim.

Consire uma textura $\mathcal{I}(\mathbf{U}, f)$, com $\mathbf{U} = [a, b] \times [c, d]$ e $\mathbf{V} = [m, n] \times [g, h]$. Assim, tem-se que a função de mudança de coordenadas de $\mathbf{U}$ para $\mathbf{V}$ é definida por

$$h(u_x, u_y) = \left(m + \frac{u_x - a}{c - a} \cdot (g - m), n + \frac{u_y - b}{d - b} \cdot (h - n) \right),$$

onde $(u_x, u_y) \in \mathbf{U}$. Logo, h^{-1} é dado por

$$h^{-1}(v_x, v_y) = \left(a + \frac{v_x - m}{g - m} \cdot (c - a), b + \frac{v_y - n}{h - n} \cdot (d - b) \right),$$

$(v_x, v_y) \in \mathbf{V}$. Dado que $v_x = f_x^{-1}(p)$ e $v_y = f_y^{-1}(p)$, tem-se que

$$h^{-1}(p) = \left(a + \frac{f_x^{-1}(p) - m}{g - m} \cdot (c - a), b + \frac{f_y^{-1}(p) - n}{h - n} \cdot (d - b) \right),$$

é a função que mapeia cada ponto $p \in \mathbf{S}$ da superfície paramétrica no suporte da textura. Com isso, o mapeamento de textura $z(p) = f(h^{-1}(p))$ associa os atributos de cor da textura em cada ponto da superfície.

Observe que a transformação h é completamente descrita pelas coordenadas dos dois vértices que definem os suportes $\mathbf{U}$ e $\mathbf{V}$. A especificação desses vértices define como os atributos de cor do conjunto $\mathbf{U}$ serão mapeados no conjunto $\mathbf{V}$. A figura 4.8(a) ilustra o mapeamento representado pela transformação h. Os pontos $u \in \mathbf{U}$ são chamados de coordenadas de textura. Um recurso que é muito utilizado é mapear um subconjunto retangular $\mathbf{U}' \subset \mathbf{U}$ do suporte da textura na superfície. Com efeito, dado $\mathbf{U}' = [s, t] \times [k, l]$, onde $s > a$, $t > b$, $k < c$ e $l < d$, basta substituir os vértices que definem $\mathbf{U}'$ na função h para obter o mapeamento desejado. A Figura 4.8(b) mostra o efeito desse mapeamento.

Para obter a função $z_{i,j,k}$ é necessário calcular a transformação $h^{-1}_{i,j,k}$ para cada ladrilho $\mathcal{T}_{i,j,k}$, onde a função $h^{-1}_{i,j,k}$ representa o mapeamento de cada ponto da superfície $\mathbf{S}_{i,j,k}$ no suporte $\mathbf{U}_{i,j,k}$ da textura $\mathcal{I}_{i,j,k}$. Neste caso, a transformação h^{-1} é definida de acordo com a região retangular da textura $\mathcal{I}_{i,j,k}$ que deve ser mapeada na superfície $\mathbf{S}_{i,j,k}$. Isso é feito a partir da especificação das coordenadas de textura (s, k) e (t, l) que definem uma região retangular $\mathbf{U}'_{i,j,k} = [s, t] \times [k, l] \subset \mathbf{U}_{i,j,k}$. O cálculo das coordenadas de textura (s, k) e (t, l) será realizado considerando que o suporte $\mathbf{U}_{i,j,k}$ é um quadrado unitário, ou seja, $\mathbf{U}_{i,j,k} = [0, 1] \times [0, 1]$.

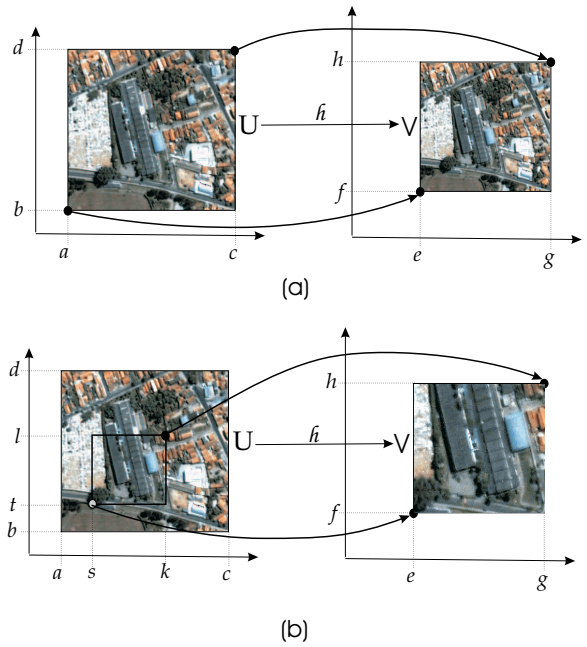

Figura 4.8: Tipos de mapeamento de textura: (a)Mapeamento de todo o conjunto **U** no conjunto **V**; (b)Mapeamento de uma pequena região retangular de **U** em **V**.

O procedimento para o cálculo das coordenadas consiste em varrer paralelamente as estruturas em multi-resolução da textura e da geometria. Esse percorrimento é feito a partir do menor nível de resolução. Os ladrilhos são acessados através da estrutura quad-tree formada pelas ligações entre os ladrilhos pais e seus quatro filhos. Durante cada passo no percorrimento das estruturas podem ocorrer duas situações. A primeira, ocorre quando os ladrilhos das duas quad-trees estão no mesmo nível de resolução. Ou seja, dados uma superfície $\mathbf{S}_{k_1}$ e uma textura $\mathcal{I}_{k_2}$, tem-se que $k_1 = k_2$. Neste caso, toda a textura $\mathcal{I}_{k_2}$ deve ser mapeada na superfície $\mathcal{I}_{k_1}$. Desta forma, as coordenadas de textura são definidas por $(s, k) = (0, 0)$ e $(t, l) = (1, 1)$ (veja a Figura 4.9(a)).

Na segunda situação, os ladrilhos das duas quad-trees não estão no mesmo nível de resolução. Dados uma superfície $\mathbf{S}_{k_1}$ e uma $\mathcal{I}_{k_2}$, tem-se que $k_1 > k_2$. Neste caso, calcula-se as coordenadas de textura (s, k) e (t, l) de tal forma que apenas uma parte da região da textura $\mathcal{I}_{k_2}$ seja mapeada na superfície $\mathbf{S}_{k_1}$. Essa região é a representação em baixa resolução da textura $\mathcal{I}_{k_1}$ que foi eliminada no processo de simplificação. As dimensões dessa região

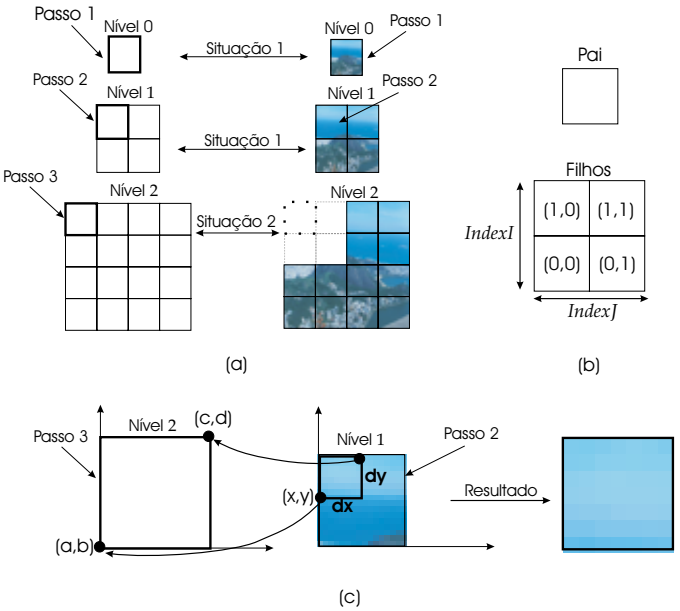

Figura 4.9: Mapeamento de texture em representação em muli-resolução; (a)Mostra as duas situações que podem ocorrer durante a fase de mapeamento; (b)Identificação dos ladrillhos filhos em relação ao seu pai;(c)Mapeamento de textura para a situação onde o nível de resolução do ladrilho geométrico difere do nível de resolução do ladrilho de textura.

são definidas por $dx = dy = \frac{1}{2^{k_1-k_2}}$. E as coordenadas iniciais são

$$x = \sum_{r=1}^{k_1-k_2} IndexJ(\mathcal{T}_r) \cdot \frac{1}{2^r} \; e \; y = \sum_{r=1}^{k_1-k_2} IndexI(\mathcal{T}_r) \cdot \frac{1}{2^r}.$$

Suponha que os quadro ladrilhos estejam organizados numa matriz 2×2 e indexados como $[0,1] \times [0,1]$. Os operadores $IndexI$ e $IndexJ$ retornam os índices (i, j) de um ladrilho filho com relação ao seu pai, como mostra a Figura 4.9(b). Assim, as coordenadas de texturas são definidas por $s = x$, $k = y$, $t = x + dx$ e $l = y + dy$. A Figura 4.9(c) mostra como é feito desse mapeamento.

O pseudo-código 4 mostra a implementação do mapeamento de textura responsável pela ligação entre a superfície e textura de um objeto gráfico. A utilização de um sistema unitário para especificar as coordenadas de textura não foi por acaso. A biblioteca de visualização OpenGL utiliza o mesmo sistema de coordenadas para especificar mapeamentos de textura (veja [20]). Assim, as coordenadas de texturas resultantes dos cálculos realizados logo acima são passadas diretamente para a biblioteca durante o processo de visualização.

4.4 Estrutura de Dados

Após a fase de representação, passa-se para a fase de implementação. Nessa fase, a representação do objeto gráfico é implementada em uma estrutura de dados. O planejamento dessa estrutura de dados leva em consideração as operações que são realizadas sobre os dados contidos nos objetos. Neste trabalho, a estrutura de dados foi planejada para possibilitar que os dados sejam acessados de forma eficiente durante o processo de visualização.

O objeto gráfico é implementado como uma estrutura que possui um ponteiro para uma estrutura em multi-resolução que armazena as informações geométricas e uma lista de ponteiros para estruturas em multi-resolução contendo informações de textura, veja a Figura 4.10.

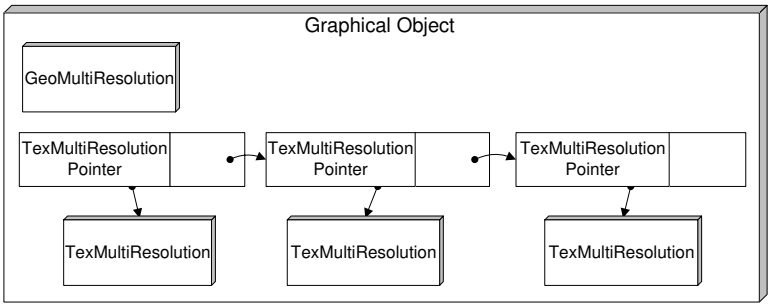

Figura 4.10: Estrutura de dados que representa um objeto gráfico no nível de implementação.

Nas duas estruturas em multi-resolução de cada nível de resolução é descrito a partir de um vetor unidimensional. Cada campo desse vetor possui um ponteiro para um ladrilho. Esses vetores são armazenados em uma lista duplamente encadeada, como mostra a Figura 4.11. A estrutura em multi-resolução geométrica é chamada de `GeoMultiResolution` e a multi-resolução de textura é chamada de `TexMultiResolution`.

Dentro de `GeoMultiResolution` existe uma estrutura de dados chamada `GeoMultiResInfo` que possui seis campos:

```
struct GeoMultiResInfo
{
  nTilesW : Número de colunas da matriz de ladrilhos do primeiro nível.
  nTilesH : Número de linhas da matriz de ladrilhos do primeiro nível.
  MinBox  : Vetor bidimensional que armazena
            a coordenada mínima do suporta paramétrico.
```

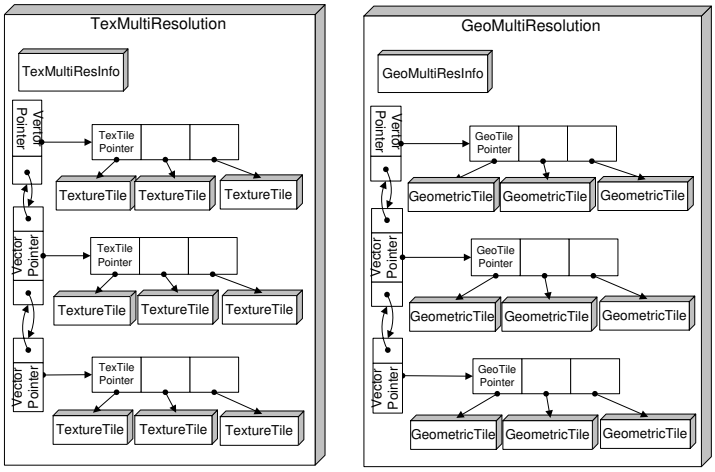

Figura 4.11: Estrutura de dados que implementa a representação em multi-resolução da geometria e da textura de um objeto gráfico.

```
  MaxBox  : Vetor bidimensional que armazena
            a coordenada mínima do suporta paramétrico.
  GeoTileW: Largura do ladrilho dentro no primeiro
            nível de resolução. GeoTileW=(MaxBox.x-MinBox.x)/nTilesW
  GeoTileH: Altura do ladrilho no primeiro
            de nível de resolução. GeoTileH=(MaxBox.y-MinBox.y)/nTilesH
}
```

A Figura 4.12 ilustra a estrutura `GeoMultiResInfo`. Os dois primeiros campos permitem que os níveis de resolução sejam acessados como se fossem matrizes. O índice de um ladrilho (i, j, k) é convertido para o índice $v = j + i \cdot (nTilesW \cdot 2^k)$ do vetor. No entanto, essa conversão só funciona se nenhum dos ladrilhos do nível acessado foi eliminado durante o processo de construção. No caso da estrutura em multi-resolução geométrica essa a operação de conversão é válida, pois nenhum ladrilho é eliminado.

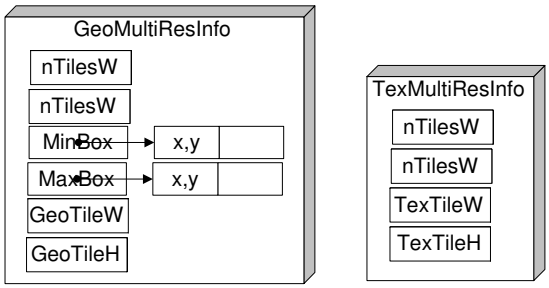

Figura 4.12: Estrutura de dados que contém informação gerais sobre a representaçõa em multi-resolução.

Os ladrilho geométricos são descrito por uma estrutura que contém os seguintes campos:

```
struct GeometricTile
{
  GeoPointer   : Ponteiro para informações geométricas.
  MapTexture   : É uma lista de mapeamentos de textura.
                 Cada elemento desse vetor possui três campos:
                 struct Mapping
                 {
                   TexPos        : Vetor bidimensional que contém
                                   a posição inicial da região da textura
                                   que deve ser mapeada.
                   TexSize       : Dimensão da região da textura que
                                   deve ser mapeada.
                   TexTilePointer: Ponteiro para um ladrilho de textura.
                 }
  ChildPointer : Vetor com quatro ponteiros para os ladrilhos filhos.
  FatherPointer: Pointeiro para o ladrilho pai.
}
```

A estrutura de dados que descreve os ladrilhos geométricos é chamada de `GeometricTile`. A Figura 4.13 ilustra essa estrutura.

A estrutura `TexMultiResolution` também possui uma estrutura de dados que armazena informações gerais sobre a multi-resolução. Essa estrutura é chamada de `TexMultiResInfo` e possui quatro campos (veja a Figura 4.12):

```
struct TexMultiResInfo
{
  nTilesW : Número de colunas da matriz de ladrilhos do primeiro nível.
  nTilesH : Número de linhas da matriz de ladrilhos do primeiro nível.
  TexTileW: Largura em pixels da textura armazenada nos ladrilhos.
  TexTileH: Altura em pixels da textura nos ladrilhos.
}
```

Os dois primeiros permitem que os vetores sejam acessados como se fossem matrizes. No entanto, na multi-resolução de textura esse tipo acesso não pode ser utilizado porque os níveis de resolução, em geral, não possuem todos os ladrilhos. Mas, observe que os ladrilhos geométricos possuem ponteiros para as texturas. Assim, uma maneira alternativa de se acessar

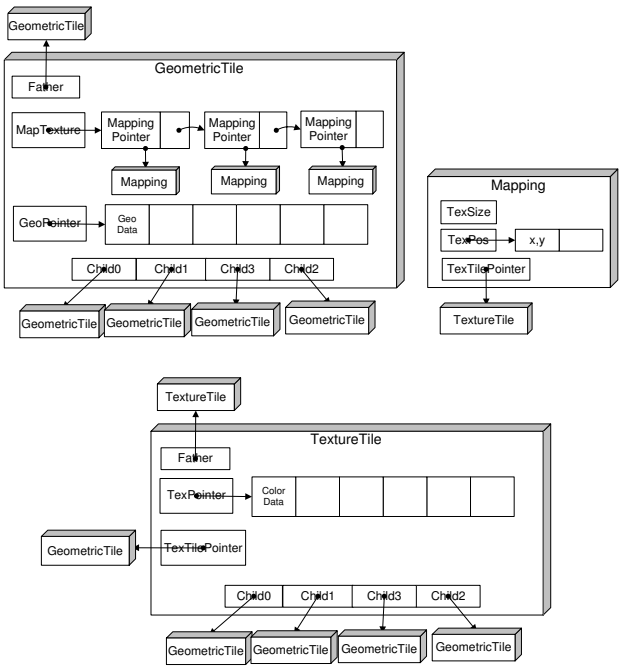

Figura 4.13: Estrutura de dados para representar os ladrilhos de textura e os ladrilhos geométricos.

um ladrilho de textura a partir dos índices (i, j, k) é acessando o ladrilho geométrico que está na posição (i, j, k) da estrutura em multi-resolução geométrica. Como já foi discutido, todos os níveis da multi-resolução geométrica estão sempre completos. Esse "truque" será utilizado mais adiante no processo de visualização.

Os ladrilhos de textura são formados por quadro campos, como mostra a Figura 4.13. Esses campos são os seguintes:

```
struct TextureTile
{
  TexPointer    : Ponteiro para uma textura.
  GeoTilePointer: Ponteiro para um ladrilho geométrico.
  ChildPointer  : Vetor com quatro ponteiros para os ladilhos filhos.
  FatherPointer : Pointeiro para o ladrilho pai.
}
```

Observe que existe uma ligação bi-direcional entre as multi-resoluções geométrica e as de texturas. Esta ligação é representada pelos ponteiros `TexTilePointer` e `GeoTilePointer`. Assim, é possível passar de uma estrutura para a outra a qualquer momento através dessa ligação.

Algoritmo 3 AdptativeMutiResolution (**Matrix** $\mathcal{MAT}$, **MultiRes** $\mathcal{MR}_{tex}$, **MultiRes** $\mathcal{MR}_{geo}$, $nLevels$,Tol)

$\mathcal{MR}_{tex}$=CreateTexLevels($nLevels$); $\mathcal{MR}_{geo}$=CreateGeoLevels($nLevels$)
$\mathcal{MR}_{tex}.Level[nLevels-1]$ =CreateTexMatrix($\mathcal{T}.nTilesW$,$\mathcal{T}.nTilesH$)
$\mathcal{MR}_{geo}.Level[nLevels-1]$ =CreateGeoMatrix($\mathcal{T}.nTilesW$,$\mathcal{T}.nTilesH$)
for ($i=0$; $i<\mathcal{T}.nTilesH$; $i=i+1$) **do**
  for ($j=0$; $j<\mathcal{T}.nTilesW$; $j=j+1$) **do**
    $\mathcal{MR}_{tex}.Level[nLevels-1,i,j].\mathcal{T}.I$=$\mathcal{MAT}[i,j].I$
    $\mathcal{MR}_{geo}.Level[nLevels-1,i,j].\mathcal{T}.S$=$\mathcal{MAT}[i,j].S$
    $\mathcal{MR}_{geo}.Level[nLevels-1,i,j].\mathcal{T}.Err$=0
  end for
end for
ntw=$\mathcal{MAT}.nTilesW$
nth=$\mathcal{MAT}.nTilesH$
for ($l=(nLevels-2)$; $l\geq 0$; $l=l-1$) **do**
  $ntw=ntw/2$
  $nth=nth/2$
  $\mathcal{MR}_{tex}.Level[l]$ =CreateTexMatrix(ntw,nth)
  $\mathcal{MR}_{geo}.Level[l]$ =CreateGeoMatrix(ntw,nth)
  for ($i=0$; $i<nth$; $i=i+1$) **do**
    for ($j=0$; $j<ntw$; $j=j+1$) **do**
      ——————**Texture Tile Processing**——————
      I_{union}=TexUnion($\mathcal{MR}_{tex}.Level[l+1,2i,2j].\mathcal{T}.I$,$\mathcal{MR}_{tex}.Level[l+1,2i,2j+1].\mathcal{T}.I$,$\mathcal{MR}_{tex}.Level[l+1,2i+1,2j].\mathcal{T}.I$,$\mathcal{MR}_{tex}.Level[l+1,2i+1,2j+1].\mathcal{T}.I$)
      I_{rsamp}=TexResample(I_{union})
      I_{ssamp}=TexSupersample(I_rsamp)
      Err_{pf}=CalculeTexError(I_ssamp,I_{union})
      $\mathcal{MR}_{tex}.Level[l,i,j].\mathcal{T}.Err$=$Err_{pf}$+TexErrorAcum($\mathcal{MR}_{tex}.Level[l+1,2i,2j].\mathcal{T}.Err$, $\mathcal{MR}_{tex}.Level[l+1,2i,2j+1].\mathcal{T}.Err$, $\mathcal{MR}_{tex}.Level[l+1,2i+1,2j].\mathcal{T}.Err$, $\mathcal{MR}_{tex}.Level[l+1,2i+1,2j+1].\mathcal{T}.Err$)
      $\mathcal{MR}_{tex}.Level[l,i,j].\mathcal{T}.I$=$I_{rsamp}$
      if $\mathcal{MR}_{tex}.Level[l,i,j].\mathcal{T}.Err\leq Tol$ **then**
        DeleteTexTile($\mathcal{MR}_{tex}.Level[l+1,2i,2j].\mathcal{T}.I$, $\mathcal{MR}_{tex}.Level[l+1,2i,2j+1].\mathcal{T}.I$, $\mathcal{MR}_{tex}.Level[l+1,2i+1,2j].\mathcal{T}.I$, $\mathcal{MR}_{tex}.Level[l+1,2i+1,2j+1].\mathcal{T}.I$)
      else
        LinkTexTile($\mathcal{MR}_{tex}.Level[l,i,j].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l+1,2i,2j].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l+1,2i,2j+1].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l+1,2i+1,2j].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l+1,2i+1,2j+1].\mathcal{T}$)
      end if
      ——————**Geometric Tile Processing**——————
      S_{union}=GeoUnion($\mathcal{MR}_{tex}.Level[l+1,2i,2j].\mathcal{T}.S$, $\mathcal{MR}_{geo}.Level[l+1,2i,2j+1].\mathcal{T}.S$, $\mathcal{MR}_{geo}.Level[l+1,2i+1,2j].\mathcal{T}.S$, $\mathcal{MR}_{geo}.Level[l+1,2i+1,2j+1].\mathcal{T}.S$)
      $\mathcal{MR}_{geo}.Level[l,i,j].\mathcal{T}.S$=GeoResample($S_{union}$)
      LinkGeoTile($\mathcal{MR}_{geo}.Level[l,i,j].\mathcal{T}$, $\mathcal{MR}_{tex}.Level[l+1,2i,2j].\mathcal{T}$, $\mathcal{MR}_{geo}.Level[l+1,2i,2j+1].\mathcal{T}$, $\mathcal{MR}_{geo}.Level[l+1,2i+1,2j].\mathcal{T}$, $\mathcal{MR}_{geo}.Level[l+1,2i+1,2j+1].\mathcal{T}$)
    end for
  end for
end for
end function

Algoritmo 4 TextureMapping (**MultiRes** $\mathcal{MR}_{tex}$, **MultiRes** $\mathcal{MR}_{geo}$)

for ($i = 0$; $i < \mathcal{MR}_{geo}.Level[0].nTilesH$; $i = i + 1$) **do**
  for ($j = 0$; $j < \mathcal{MR}_{geo}.Level[0].nTilesW$; $j = i + 1$) **do**
    CalculteTexMapping ($\mathcal{MR}_{geo}.Level[0].\mathcal{T}$,$\mathcal{MR}_{tex}.Level[0].\mathcal{T}$, 0.0, 0.0, 1.0, 1.0)
  end for
end for
end function

———————CALCULATETEXMAPPING IMPLEMENTATION———————

CalculteTexMapping (**Tile** $\mathcal{T}_{geo}$, **Tile** $\mathcal{T}_{tex}$, $TexLevel$, x, y, dx, dy)
if HaveChildren($\mathcal{T}_{geo}$) **then**
  for ($i = 0$; $i < 2$; $i = i + 1$) **do**
    for ($j = 0$; $j < 2$; $j = j + 1$) **do**
      if HaveChildren($\mathcal{T}_{tex}$) **then**
        CalculateTexMapping(GotoChild($\mathcal{T}_{geo}$, i, j), GotoChild($\mathcal{T}_{tex}$, i, j), 0.0, 0.0, 1.0, 1.0)
      else
        $Newdx$=$dx/2$;
        $Newdy = dy/2$
        $Newx$=$x + jdx$;
        $Newy$=$y + idy$
        CalculateTexMapping(GotoChild($\mathcal{T}_{geo}$, i, j),$\mathcal{T}_{tex}$, $Newx$, $Newy$, $Newdx$, $Newdy$)
      end if
    end for
  end for
else
  $\mathcal{T}_{geo}.Z.s$=x;
  $\mathcal{T}_{geo}.Z.k$=y
  $\mathcal{T}_{geo}.Z.t$=$x + dx$;
  $\mathcal{T}_{geo}.Z.l$=$y + dy$
  $\mathcal{T}_{geo}.Z.\mathcal{T}_{tex} = \mathcal{T}_{tex}$
end if
end function