4
Geração Eficiente da Textura

Este capítulo aborda conceitos referentes à simplificação dos dados geográficos a serem renderizados para diminuir o tempo gasto para a geração da textura.

4.1
Simplificação dos Dados Geográficos

Os dados geográficos tratados no presente trabalho são as linhas e polígonos. Para diminuir o tempo de renderização da textura com dados geográficos é preciso que seja possível representar de maneira fiel tais geometrias com um número reduzido de vértices. Para isso foi utilizado o algoritmo de Douglas-Peucker descrito na Seção 2.4 para a simplificação das polilinhas.

Apesar do algoritmo de Douglas-Peucker originalmente ser aplicado apenas a geometria de linhas, é possível uma pequena adaptação para o caso de polígonos. Um polígono pode ser facilmente fragmentado em duas linhas, essas linhas são simplificadas, e depois o polígono é remontado, sendo uma simplificação do polígono original. A heurística utilizada para fragmentar o polígono é calcular os pontos extremos ao longo do eixo mais longo do polígono. A Figura 4.1 mostra o passo a passo da solução.

4.2
Hierarquia de Simplificação

A organização dos dados a serem simplificados é importante para a escalabilidade e a possibilidade de adaptar a qualidade de acordo com a posição da navegação e a distância ao dado visualizado. O objetivo principal é conseguir um processo paralelo à multi-resolução do terreno que garanta a eficiência do desenho das primitivas geográficas de forma independente, mas que ocorra de maneira similar e que possa ser utilizada em conjunto com qualquer algoritmo para a simplificação do terreno.

Para a hierarquia, foi implementado uma variação da *kd-tree* (Ooi87) para indexar geometrias próximas em blocos que tenham aproximadamente o mesmo número de vértices. A estrutura consiste em uma árvore binária

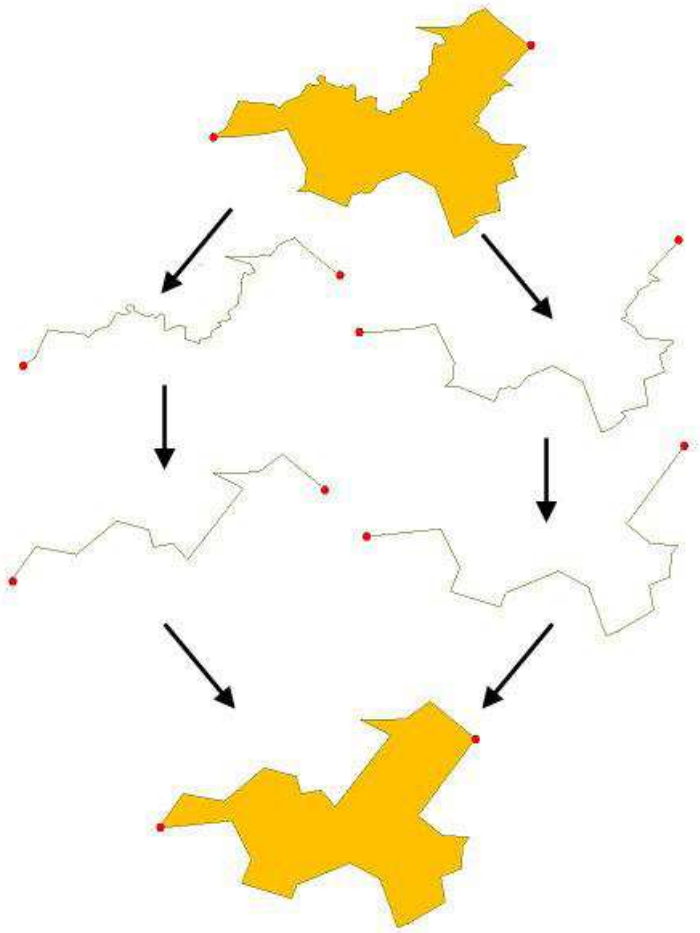

Figura 4.1: Simplificação de polígonos utilizando o algoritmo de Douglas-Peucker

balanceada completa onde cada nó representa uma região do mundo. Para preencher a estrutura precisamos calcular quais são as áreas que representam os blocos e quais objetos devem pertencer a cada nó.

Deve-se montar uma hierarquia para cada tema. Dados um conjunto de informações sobre as geometrias pertencentes ao tema a ser indexado, como a caixa envolvente e o número de vértices, mais um valor n que é o número aproximado de vértices que estarão presentes nos blocos, o processo de criação da estrutura é descrito a seguir. O valor de n é baseado no número de vértices que a API de *arrays* do OpenGL é capaz de manipular eficientemente por vez, otimizando o desenho de cada bloco.

A partir do valor de n, e o número total de vértices nos objetos a serem indexados, estima-se o número necessário de níveis da árvore para que todas as geometrias originais estejam indexadas nas folhas. Partindo da caixa envolvente do conjunto de todas as geometrias, executa-se o seguinte algoritmo:

Encontra-se uma reta que divida a caixa envolvente de modo que o número de vértices presentes em cada uma das subdivisões seja aproximadamente o mesmo. A direção dessa reta é ao longo do maior eixo da caixa envolvente atual. A posição da reta é decidida de forma heurística, é uma média ponderada considerando o número de coordenadas de cada geometria e a posição de sua caixa envolvente. Sendo v_i o número de vértices do i-ésimo objeto, x_i o menor x da caixa do i-ésimo objeto e N_o o número total de objetos a serem inseridos na estrutura, a fórmula para o cálculo da posição x de corte é dada pela equação: (para cortes verticais, o cálculo é análogo)

$$y = \frac{\sum_{i=1}^{N_o} v_i y_i}{\sum_{i=1}^{N_o} v_i}$$

A idéia básica é que os objetos com mais vértices tendem a direcionar a reta de corte para próximo do seu x mínimo, obtendo uma reta que é uma média da distribuição dos vértices dos objetos indexados. O processo de subdivisão é chamado recursivamente para as duas caixas geradas pela divisão da reta escolhida, até que se preencha todos os níveis da árvore.

Partindo do princípio intuitivo que quanto maior a região a ser visualizada maior a quantidade de geometrias a serem processadas e, consequentemente, menor o detalhe necessário para uma visualização satisfatória, calculamos valores para os erros dos nós da estrutura. Os nós folhas das árvores representam as geometrias originais, sem nenhuma simplificação. À medida que se sobe na árvore, mais rudimentar são as linhas que o nó armazena, pois os nós representam uma área maior do mundo. O nó raiz contém a caixa que envolve todas as geometrias, contendo assim o formato mais simples das geometrias da hierarquia.

Os valores passados como tolerância para o algoritmo de Douglas-Peucker (Dou73) para que as geometrias possam ser simplificadas são calculadas para cada nó da árvore. O valor da tolerância para um nó é o valor de um pixel na coordenada de mundo, considerando uma projeção ortográfica ajustada à caixa envolvente do nó, para uma dada janela de visualização com tamanho definido. Assim o nó tem o seu valor associado e as geometrias relacionadas a ele são geradas utilizando o algoritmo de Douglas-Peucker usando esse número como parâmetro.

4.3 Qualidade e Coerência da Simplificação

Como descrito anteriormente na Seção 4.1, usamos uma solução simples para a simplificação dos polígonos utilizando o algoritmo de Douglas-Peucker,

originalmente aplicado às linhas. Na solução apresentada, é facil perceber que a escolha dos vértices para a fragmentação do polígono influencia diretamente no polígono simplificado resultante, ou em outras palavras, cada par de vértices escolhidos para a fragmentação gera um polígono simplificado diferente.

Essa propriedade prejudica potencialmente um caso muito comum de dados geográficos de polígonos, onde os polígonos são adjacentes uns aos outros e contém vários vértices coincidentes, como ocorre em dados que representam fronteiras políticas de municípios, estados ou países, por exemplo. Assim, como a simplificação não garante nenhuma propriedade dos vértices presentes no resultado, tais polígonos teriam suas fronteiras descaracterizadas, podendo ocorrer linhas não coincidentes entre vizinhos. Isso gera obviamente um efeito visual desagradável, além de uma incoerência geográfica conceitual. A Figura 4.2 mostra o problema. As áreas em vermelho são as incoêrencias, sendo sobreposições ou "vazios".

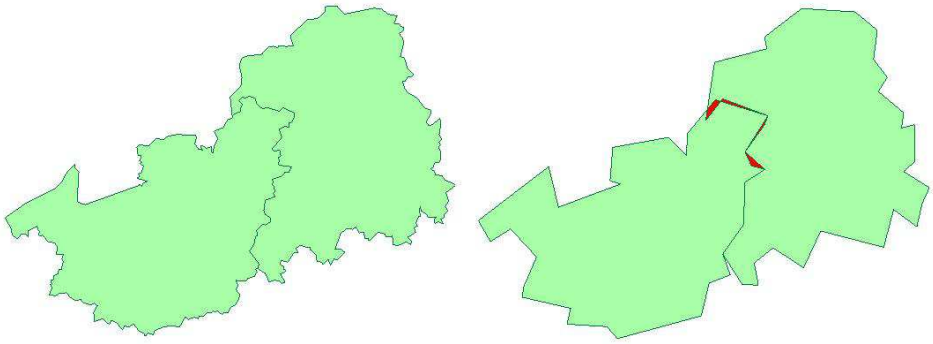

Figura 4.2: Incoerência topológica devido à simplificação

Para manter a coerência nesses casos, seria necessário um tipo especial de fragmentação. Tal processo seria feito de forma que os vértices comuns entre polígonos adjacentes fizesse parte de uma linha a ser simplificada, e os vértices disjuntos desses polígonos formariam outros fragmentos. Após ter esses fragmentos simplificados, os polígonos poderiam ser remontados com coerência entre suas fronteiras. O resultado esperado é exemplificado pela Figura 4.3.

Descobrir os pontos de fragmentação é um processo complexo, que pode ser obtido por adaptações de algoritmos geométricos de custo computacional considerável, como sobreposição de polígonos ou cálculo de interseções de um conjunto de segmentos. Mesmo após a descoberta dos pontos, ainda é necessário organizar os fragmentos após a simplificação para que eles possam ser reunidos e formar os polígonos simplificados finais. Esse tratamento exigiria um pré-processamento muito custoso para o tamanho dos dados que o presente trabalho deseja manipular.

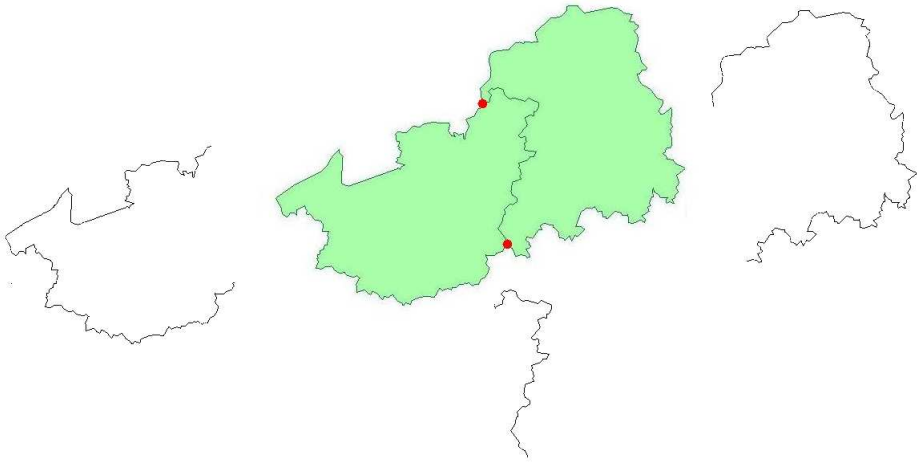

Figura 4.3: Fragmentação que garante a coerência entre fronteiras

Embora a coerência não tenha sido mantida por motivos de eficiência, a decisão do nível de detalhe dos dados e a tolerância torna imperceptível a ocorrência do fenômeno. Por isso a escolha do bloco da hierarquia deve respeitar uma tolerância fornecida pelo usuário, quanto maior a tolerância, maior a percepção do efeito negativo. Caso a tolerância para o erro seja pequena, o efeito praticamente não é percebido durante a navegação.

Um outro caso que a simplificação pode contribuir para a melhoria da qualidade geral da visualização é quando ocorre uma densidade grande de objetos geográficos em uma pequena área da janela de visualização, tanto na visão ortográfica tradicional de SIG quanto na visão tridimensional. Muitos objetos em uma pequena porção da área visível torna indistinguível os objetos individualmente, se transformando em informação visual irrelevante. Como consequência natural do processo de simplificação, são eliminados objetos geográficos cujo tamanho não seja significativo na tolerância definida no bloco reduzindo a saturação de objetos.

4.4 Escolha do Nível de Detalhe dos Dados

A idéia para a simplificação é utilizar maior nível de detalhe das geometrias nas frações da textura que se encontram mais próximas ao observador ou tenha uma importância maior na imagem final por ocupar uma porcentagem considerável da tela. O processo de decisão para qual bloco deve ser visualizado tenta aproximar o erro utilizado para o cálculo da simplificação do bloco para um valor definido pelo usuário, que representa o número de pixels aceitável de diferença da geometria original para a simplificada. O valor padrão que vamos utilizar para o trabalho é de um pixel, pois, como discutido na explicação do algoritmo de Douglas-Peucker, a área em coordenadas de mundo correspon-

dente a um pixel na tela é suficiente para uma boa redução de vértices, além de garantir a qualidade e evitar o efeito da incoerência discutido na seção anterior.

A decisão de qual bloco da hierarquia dos dados vetoriais deve ser enviado para o desenho é adaptada da solução mostrada na dissertação de Magalhães (Mag05), originada do trabalho de Lindstrom (Lin02). A técnica original projeta o erro no espaço do objeto presente no bloco de terreno na tela a partir de um ponto específico, obtendo o erro do citado bloco em pixels no estado atual da visualização. Assim, dado um valor τ fornecido pelo usuário que representa a tolerância do erro em pixels, são decididos quais blocos na hierarquia serão mostrados cujos erros projetados não ultrapassem esse valor.

A diferença básica entre o cálculo do erro projetado é que as caixas envolventes dos blocos dos terrenos são caixas tridimensionais, enquanto nos blocos de dados vetoriais são apenas quadriláteros, pois os dados originalmente bidimensionais são projetados em um plano. O erro projetado é calculado da seguinte forma para a hierarquia de simplificação dos dados geográficos: Dado um bloco de dados geográficos com erro ϵ e uma distância d entre o observador e o ponto mais próximo no quadrilátero envolvente do bloco, o erro projetado ρ é dado por:

$$\rho = \lambda \frac{\epsilon}{d}$$

onde $\lambda = \frac{w}{2 \tan \gamma/2}$, w é a altura da janela de visualização em pixels e γ é o ângulo de abertura da câmera.

É importante salientar que este cálculo é feito no espaço da câmera antes de se aplicar a transformação sobre a textura. A transformação deforma também a noção de distância e o tamanho dos blocos, e a quantidade de detalhes deve ser definida de acordo com a necessidade do observador, independente da transformação aplicada.

A escolha dos blocos é feita como um percorrimento em profundidade na hierarquia de simplificação. Partindo da raiz, que contém as geometrias mais simplificadas, ou seja, com o maior erro no espaço do objeto, a seguinte decisão é feita para cada nó da árvore. Se o valor ρ do bloco for menor que o valor τ fornecido pelo usuário, este bloco é enviado para desenho. Caso contrário, a decisão é chamada recursivamente para os dois filhos do nó, que possui erros menores que o nó atual por definição. Como nos nós folha o erro associado é zero, já que as geometrias não são simplificados nestes blocos, caso a busca chegue em um nó folha este é automaticamente escolhido. Um detalhe importante para a eficiência da hierarquia é que apenas blocos visíveis no estado atual da câmera são considerados na escolha, funcionando como uma técnica similar ao *frustum culling* (Ass00, Cla76). O pseudo-código a seguir

detalha o funcionamento do algoritmo.

Algoritmo 1 Escolha dos blocos na hierarquia de simplificação

```
function EscolheBlocos(x)
if Nó x é visível then
  if Nó x é folha then
    Nó x é escolhido
  else
    if (Erro ρ em x) ≤ τ then
      Nó x é escolhido
    else
      EscolheBlocos(Filho Esquerdo de x)
      EscolheBlocos(Filho Direito de x)
    end if
  end if
end if
```