Técnicas Simples para Visualização Foto-realista de Terrenos em Tempo Real Usando Placas Gráficas Programáveis

Ricardo Gomes Leal Costa
Waldemar Celes (orientador)
Tecgraf - Departamento de Informática, PUC-Rio
{rcosta,celes}@inf.puc-rio.br

Resumo

Este trabalho consiste em explorar técnicas simples de programação em placas gráficas para gerar imagens de terrenos virtuais com boa qualidade. Implementamos vários efeitos de iluminação que, somados ao mapeamento procedural de camadas de textura em função da altitude, resultam numa aparência mais natural. Empregamos o algoritmo Variance Shadow Map para geração de sombras suaves em terrenos. Apresentamos ainda uma solução simples e eficiente para simulação de neblina e a variação da cor do céu em relação à hora do dia. Os resultados obtidos foram positivos, alcançando uma aparência realista com ótimo desempenho.

1. Introdução

A visualização de terrenos é utilizada em diversas áreas da computação como jogos, simuladores de vôo, mapeamento de planetas e simulações científicas. Em muitos casos, deseja-se obter uma aparência foto-realista dada pelas características geométricas (relevo e rugosidade da superfície), materiais (cor e reflexão luminosa de cada tipo de solo) e ambientais (efeitos climáticos e atmosféricos). A simulação destes efeitos geralmente requer algoritmos computacionalmente caros que nem sempre são adequados para visualização em tempo real.

O objetivo deste trabalho é apresentar uma solução simples para esses problemas através de programação em placa gráfica. Todos os cálculos de geometria e iluminação são feitos na unidade de processamento gráfico (GPU) a partir de parâmetros definidos na unidade de processamento central (CPU). Com isso, espera-se obter um bom desempenho mesmo com a grande quantidade de cálculos necessária.

Os vértices do terreno, que formam uma malha inicialmente plana, são deslocados por um mapa de elevação. Em seguida, o programa de fragmentos mapeia a textura correspondente à altitude daquele ponto e calcula a iluminação difusa e especular, aplicando também mapas de rugosidade e de brilho. As sombras são geradas pelo algoritmo *Variance Shadow Map* [3], que apresenta diversas vantagens quando usado em terrenos. Finalmente, o terreno recebe um efeito de névoa seca. O Sol e a cor do céu também são desenhados proceduralmente pelo programa de fragmentos com base na hora do dia.

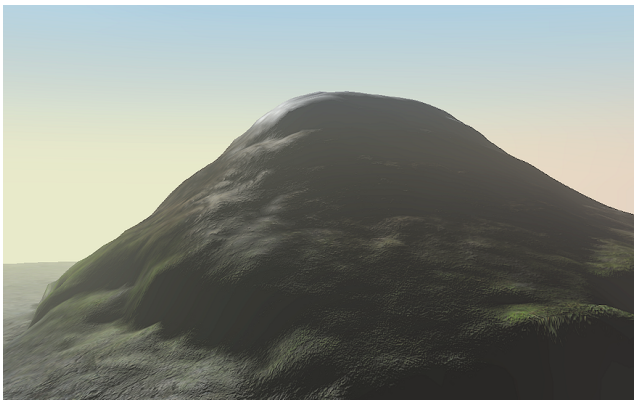

Figura 1. Terreno ao amanhecer.

2. Implementação das técnicas

2.1. Geometria

O relevo do terreno é gerado a partir de uma malha de triângulos plana com $y = 0$ para todos os vértices. A quantidade de vértices não varia ao longo do tempo, o que torna vantajoso seu armazenamento em memória de vídeo por meio de um *vertex buffer object*. O relevo do terreno é descrito por uma textura chamada *mapa de elevação*. Esta textura é acessada pelo programa de vértices e funciona como um *mapa de deslocamento* para a superfície do terreno, ou seja, a nova coordenada y dos vértices é proporcional ao va-

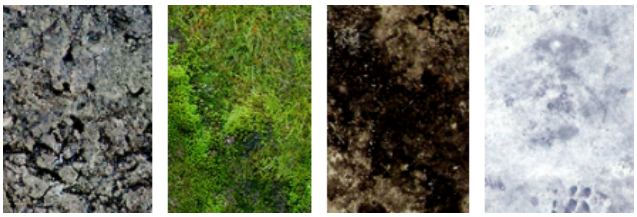

Figura 2. Texturas com os tipos de solo.

lor dos texels correspondentes.

Em seguida, é necessário calcular as normais dos vértices. Para isso, calculamos a elevação dos vértices vizinhos, acessando seus respectivos texels no mapa de elevação. São necessários apenas dois vértices vizinhos para formarmos uma base de três vértices, portanto um acesso é feito na coordenada de textura $(s + d, t)$ e outro em $(s, t + d)$, onde d é a distância entre dois texels em coordenadas de textura. Em posse da posição dos vértices vizinhos, o produto vetorial dos vetores formados por esses pontos resulta na normal daquele vértice, que é passada para o programa de fragmentos como coordenada de textura para cálculo da iluminação.

2.2. Aparência

Em geral, um terreno possui camadas de diferentes materiais que variam com a altitude. Neste trabalho, consideramos que o terreno apresenta quatro tipos de solo de acordo com sua elevação: arenoso, gramado, rochoso e com neve (Fig. 2). Estas quatro texturas são passadas em conjunto ao programa de fragmentos através de um *array de texturas* [5]. A vantagem desta técnica em relação ao uso de quatro texturas separadas é que apenas uma unidade de processamento de texturas da GPU é utilizada. Também é mais conveniente que uma textura volumétrica, pois os filtros de redução e magnificação são aplicados apenas nas coordenadas s e t, dando liberdade para fazermos a interpolação entre camadas da maneira desejada, proceduralmente.

No programa de fragmentos, cada fragmento recebe a altitude dos vértices interpolada linearmente pelo rasterizador. Com base nesta altitude, uma das quatro texturas é selecionada e seu texel contribui para a cor difusa do fragmento. Nas interseções entre camadas, as duas texturas são interpoladas. A Tabela 1 mostra um exemplo de como cada textura pode ser escolhida em função da altitude do fragmento normalizada para o intervalo [0, 1].

Técnicas procedurais são utilizadas para tornar as transições de camadas menos artificiais. Primeiro, uma textura volumétrica contendo *Perlin Noise* [6] é acessada com as mesmas coordenadas de textura utilizadas para o mapeamento dos tipos de solo, e o valor resultante é somado aos

Altitude	Textura
0.00 - 0.25	Areia
0.20 - 0.45	Grama
0.40 - 0.85	Rocha
0.80 - 1.00	Neve

Tabela 1. Camadas de textura.

limites de altitude entre camadas da Tabela 1, deixando as transições menos lineares e homogêneas. Inicialmente, tentamos somar a perturbação diretamente à altitude do ponto, mas isto resultou numa aparência granulosa para superfícies planas situadas nas transições, o que não parecia muito natural. Em seguida, a função *smoothstep* calcula a interpolação entre as camadas. Ela cria uma transição não linear entre dois valores, o que também deixa as transições de camadas menos perceptíveis.

2.3. Iluminação

A iluminação difusa, especular e ambiente é calculada por fragmento pelo modelo de reflexão de *Phong* [7]. É possível definir coeficientes distintos de reflexão difusa, especular, ambiente, emissiva e fator de brilho para cada camada do terreno. Assim, a camada de gelo pode ter reflexão especular mais forte que a camada de areia, por exemplo. Nas áreas de transição de camadas, estes coeficientes são interpolados do mesmo modo que as texturas.

2.3.1. Mapeamento de rugosidade Para dar aparência rugosa ao terreno, não podemos simplesmente alterar a geometria, pois a quantidade de triângulos necessária para representar essas pequenas deformações seria inviável do ponto de vista do desempenho. Para contornar este problema, a técnica de *mapeamento de rugosidade* é utilizada. Ela cria perturbações nas normais da superfície para que a iluminação dê à superfície uma aparência rugosa. Para isso, um *mapa de normais* contém em seus canais (R, G, B) as coordenadas (x, y, z) das normais perturbadas de uma superfície plana. Assim como as texturas do terreno, o mapa de normais também é passado ao programa de fragmentos por um array de texturas, já que cada camada pode apresentar rugosidade diferente. Porém, como a superfície do terreno é irregular, não podemos aplicar diretamente essas normais. Em vez disso, os parâmetros para o cálculo da iluminação são transformados para o *espaço tangente*. Ele é obtido através de uma base ortonormal contendo os vetores normal, binormal e tangente de cada ponto, previamente calculados no programa de vértices. Dessa forma, as perturbações das normais tornam-se relativas às normais originais do terreno.

Para que a iluminação com mapeamento de rugosidade pareça correta, é necessário considerar não apenas a nor-

mal perturbada como também a original no cálculo de iluminação difusa e especular. Existem casos em que a normal original está voltada contra a luz, mas a perturbada não, e vice-versa. Quando isso acontece, surge rugosidade em áreas não iluminadas, ou áreas iluminadas ficam sem rugosidade. A solução mais simples para este problema é iluminar apenas quando ambas as normais não estão contra a luz. Porém, esta definição implica numa transição abrupta entre regiões iluminadas e regiões em sombra (Fig. 3a). Por conta disso, optamos por interpolar o fator de auto-sombreamento de ambas as normais suavemente com uso da função *smoothstep*, resultando numa aparência mais natural (Fig. 3b).

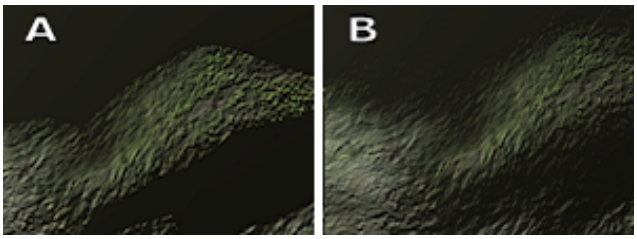

Figura 3. Correção da iluminação com mapeamento de rugosidade.

2.3.2. Mapeamento de brilho Além da rugosidade, algumas regiões da superfície do terreno podem apresentar mais ou menos reflexão especular. Por exemplo, uma rachadura na rocha não apresentaria nenhuma reflexão especular, enquanto que pequenos cristais de silicato teriam uma reflexão especular forte. Para simular esse efeito, uma textura especial em tons de cinza mapeia as diferentes intensidades de brilho especular, funcionando como um coeficiente para o valor especular resultante daquele fragmento.

2.3.3. Geração de sombras Para que as montanhas do terreno gerem sombras ao serem iluminadas pelo sol, precisamos implementar uma técnica de sombreamento. O *Variance Shadow Map* [3] gera sombras suaves a partir do mapa de profundidade da cena em relação à fonte de luz, sem exigir tanto processamento como algoritmos baseados em *Percentage-Closer Filtering*. Sua desvantagem é a presença de "vazamentos" de luz em áreas onde existem sombras de mais de um objeto no mesmo local, porém isto não é um problema para terrenos, já que as sombras do relevo não costumam sobrepor-se. Outra vantagem de sua aplicação em terrenos é que o Sol pode ser considerado uma fonte de luz direcional. Assim, não é necessário calcular a distância euclidiana de cada fragmento até a fonte de luz para obtenção do mapa de profundidade, já que este pode ser obtido diretamente da coordenada z dos fragmentos vistos a partir da posição da luz, que por sua vez é fornecida pelo programa de vértices e interpolada linearmente por fragmento.

2.4. Céu e atmosfera

Para que a cena fique mais realista, precisamos considerar os efeitos causados pela presença da atmosfera. O céu é representado geometricamente por um cubo simples, envolvendo todo o terreno. A aparência do céu é gerada de forma procedural pelo programa de fragmentos aplicado ao cubo. A cor calculada para cada ponto é dada pela posição do Sol e pelo espalhamento de luz na atmosfera.

O desenho do Sol também é gerado proceduralmente, pela Eq. 1, onde $\vec{L}$ é a direção da fonte de luz (Sol) e $\vec{V}$ é a direção da câmera até o fragmento, normalizados, com um certo fator de potência k_i. O produto é feito diversas vezes com diferentes fatores de potência e acumulado na cor final do pixel, para que ocorra uma variação gradual de iluminação do centro do Sol até sua vizinhança. O resultado pode ser visto na Fig. 4a.

$$Sol_{rgb} = \sum_{i=0}^{n} (\vec{L} \cdot \vec{V})^{k_i} \quad (1)$$

A cor do céu é obtida a partir de três texturas que representam os diversos tons de acordo com a hora do dia, noite e pôr-do-sol (Fig. 4b), simulando o espalhamento de luz na atmosfera. A coordenada de textura s, calculada pela Eq. 2, indica a posição longitudinal do Sol dado o ângulo plano entre $\vec{L}$ e $\vec{V}$, enquanto que a coordenada t, descrita pela Eq. 3, onde h é um coeficiente de altitude dado pela posição da câmera, indica sua latitude. Em seguida, dependendo da coordenada y da direção do Sol, é feita uma interpolação entre as texturas "dia" e "pôr-do-sol" ou "pôr-do-sol" e "noite".

Figura 4. (a) Representação procedural do Sol. (b) Texturas para dia, pôr-do-sol e noite.

Para tornar o fenômeno de espalhamento da luz mais plausível, o efeito de névoa seca (*haze effect*) é simulado no programa de fragmentos. Uma camada de neblina próxima à superfície contribui para a cor final do fragmento com a mesma cor que seria aplicada ao céu naquele ponto.

Sua intensidade depende da altitude e da distância da câmera até o ponto, calculada pelo programa de vértices e interpolada na rasterização, para melhor desempenho.

$$s = \frac{(\vec{L}_{xy} \cdot \vec{V}_{xy})}{2} + 0.5 \quad (2)$$

$$t = max(0, V_y)^h \quad (3)$$

2.5. Superfície molhada

Um efeito interessante é o de superfície molhada. Ele pode ser usado quando quisermos simular chuva ou áreas banhadas pelo oceano. Sua implementação é simples: as áreas molhadas recebem um aumento no fator de reflexão especular e uma redução no fator de iluminação difusa [4]. A intensidade do efeito pode ser regulada de acordo com o tipo de solo, então as áreas rochosas são as que geralmente sofrem maior variação quando estão molhadas (Fig. 5).

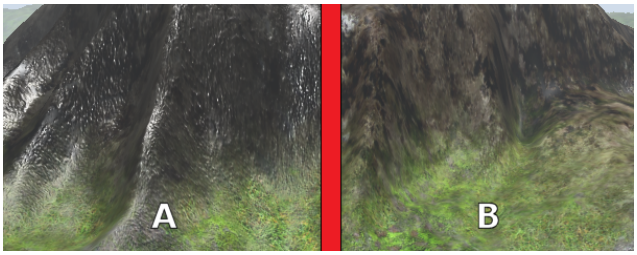

Figura 5. Superfície (a) molhada, (b) seca.

3. Resultados e conclusão

O uso de programação em placas gráficas possibilitou a implementação de técnicas simples para obtenção de uma aparência mais natural e realista. O mapeamento procedural de camadas de textura permite a visualização de terrenos com infinitas variações de relevo. A técnica *Variance Shadow Map* mostrou-se muito adequada para geração de sombras em terrenos (Fig. 6). A nossa solução para simulação atmosférica, com a cor do céu variando em função da hora do dia e com efeito de névoa seca sobre o terreno, é de fácil implementação e melhorou o realismo da cena (Fig. 7).

O programa obteve bom desempenho, com uma média de 270 quadros por segundo[1] para resolução de 1280x1024.

Em trabalhos futuros, a adição de nuvens e efeitos climáticos poderia tornar a cena ainda mais natural [1]. O uso de iluminação *High Dynamic Range* (HDR) [2] permitiria efeitos de iluminação mais realistas. Finalmente, para maior desempenho em terrenos muito grandes, técnicas de multirresolução de geometria poderiam ser implementadas.

1 Testado num Intel Core 2 Duo E6600 com NVIDIA GeForce 8800 GTS 320MB.

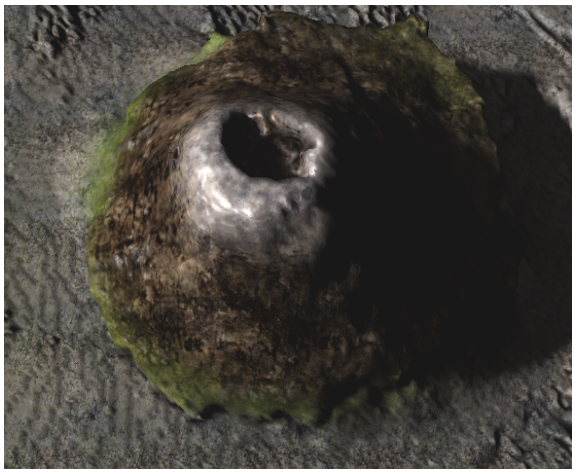

Figura 6. Terreno com sombras suaves.

Figura 7. Dia, pôr-do-sol e noite.

Referências

[1] Crytek. "Real-Time Atmospheric Effects in Games, Revisted". Disponível em `http://developer.nvidia.com/`.

[2] P. E. Debevec. "Rendering with Natural Light". *SIGGRAPH*, 1998.

[3] W. Donnelly and A. Lauritzen. "Variance Shadow Maps". Technical report, Computer Graphics Lab, School of Computer Science, University of Waterloo, 2006.

[4] NVIDIA Corporation. "Team Secrets: Cascades". Disponível em `http://developer.nvidia.com/`.

[5] NVIDIA Corporation. "Texture Array Terrain". Disponível em `http://developer.nvidia.com/`.

[6] K. Perlin. "Improving Noise". Technical report, Media Research Laboratory, Dept. of Computer Science, New York University, 2002.

[7] B.-T. Phong. "Illumination for computer generated images". *ACM Computing Surveys*, 18(6), June 1975.