Simulação de Iluminação Volumétrica

Saulo Andrade Pessoa
Universidade Federal de Pernambuco
Centro de Informática
Recife, Pernambuco, Brasil
sap@cin.ufpe.br

Resumo

Em computação gráfica, inúmeros métodos vêm sendo criados para aumentar o realismo das imagens sintetizadas. Tais métodos são inspirados principalmente em fenômenos físicos. Um fenômeno bastante estudado é a interação ocorrida entre a luz e os objetos, porém, tal interação é geralmente simplificada devido às limitações computacionais. Uma simplificação comumente realizada é a de desconsiderar a interação ocorrida entre a luz e a atmosfera. Tal simplificação pode ser entendida como se a luz estivesse viajando no vácuo, fazendo com que as imagens careçam de efeitos atmosféricos. Um efeito comumente observado é quando gelo seco é espalhado sobre o palco e holofotes são direcionados sobre as partículas suspensas, gerando um volume de luz cônico. O foco deste trabalho será neste último efeito, que é chamado de iluminação volumétrica.

1. Introdução

Determinados fenômenos naturais só podem ser reproduzidos em computação gráfica caso o meio seja participativo. Com um meio participativo, a luz interage com as partículas suspensas no ar à medida que se propaga. Como o planeta Terra é naturalmente envolvido por uma camada gasosa (atmosfera), a interação entre luz e meio se torna inevitável. Além da atmosfera, a luz também pode interagir com qualquer outro tipo de partícula que esteja suspensa no ar, como, por exemplo, fumaça, gelo seco, poeira, etc.

2. Contexto

Atualmente, inúmeros pesquisadores já realizaram estudos e desenvolveram métodos para simulação de iluminação volumétrica. Basicamente, tais métodos resolvem o problema integrando-se numericamente a luz que é espalhada ao longo do raio de visão. Os métodos propostos diferem basicamente em como a integral é calculada, o que leva à classificação dos métodos em dois tipos: aplicáveis a GPUs e os não aplicáveis a GPUs.

Dentre os não aplicáveis a GPUs, [1] propôs um método baseado no traçado de raios para encontrar os pontos de interseção entre os volumes de luz e o raio de visão. [2] fez uso de mapas de fótons para obter tal efeito. Por serem custosos, estes métodos não são indicados para aplicações interativas.

Os métodos aplicáveis a GPUs só começaram a surgir com o início da popularização das placas gráficas no final da década passada. [3] propôs um método aplicável a GPUs que faz uso de técnicas de renderização volumétrica para integrar a luz espalhada. [4] propôs otimizações para o método anterior. [5] e [6] propuseram métodos que integram a luz espalhada utilizando *shaders* programáveis e que fazem uso de geometria para representar os volumes de luz. [7] propôs uma abordagem para tratar partículas com densidade não uniforme.

3. Espalhamento de Luz

Efeitos atmosféricos ocorrem devido ao espalhamento e absorção da luz ao se chocar com partículas suspensas. O Espalhamento de Mie explica a formação dos volumes de luz. Esse tipo de espalhamento ocorre em atmosferas pesadas, como água ou fumaça [5], isto é, quando o tamanho das partículas é grande em relação ao comprimento de onda da luz. Existem quatro tipos de interação que a luz pode sofrer: absorção, emissão, espalhamento e ajuntamento [8]. A Figura 1 mostra esses tipos de interação.

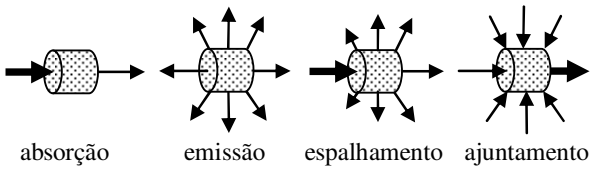


Figura 1. Interações que a luz pode sofrer

4. Modelo de *Shading*

O método [3] foi tomado como referência devido a sua adequação a GPUs. Por uma questão de simplicidade, será assumido o caso em que existe apenas uma fonte de luz. O método pode ser facilmente estendido para suportar múltiplas fontes de luz apenas calculando a contribuição de cada uma.

Basicamente, o objetivo final é obter a intensidade de luz que alcança a câmera que é expressa como

$$I_c = I_o \beta(T) + \int_0^T F(\alpha) H(r) I_p(r) \beta(t) \sigma dt , \qquad (1)$$

onde I_c é a intensidade de luz que chega a câmera, I_o é a intensidade radiada pelo objeto, T é a distância entre o objeto e a câmera, $\beta(T)$ é a taxa de absorção, α é o ângulo de fase, $F(\alpha)$ é a função de fase, r é a distância entre o ponto P e a fonte de luz, $H(r)$ é a função de visibilidade, $I_p(r)$ é a intensidade de luz que chega ao ponto P, t é a distância entre o ponto P e a câmera, σ é o coeficiente de extinção que é proporcional à densidade das partículas (ver **Figura 2**).

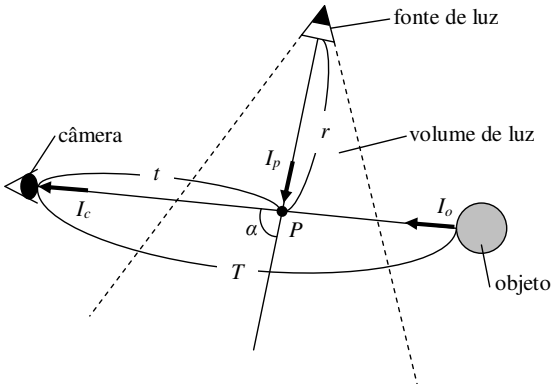


Figura 2. Modelo de shading

Para uma densidade uniforme de partículas, a taxa de absorção $\beta(t)$ é expressa como

$$\beta(t) = e^{-\sigma t} . \qquad (2)$$

A função de fase $F(\alpha)$ diz respeito à intensidade de luz que é espalhada na direção da câmera quando esta faz um ângulo α com a direção do raio de luz. Devido à complexidade da teoria de Mie, a função de fase utilizada é uma aproximação experimental proposta por [9]. $F(\alpha)$ é expressa como

$$F(\alpha) = K(1 + 9\cos^{16}(\frac{\alpha}{2})) , \qquad (3)$$

onde K é uma constante. Por fim, a intensidade de luz que alcança o ponto P é dada por

$$I_p(r) = \frac{I(\theta,\Phi)}{r^2} B(r) , \qquad (4)$$

onde $I(\theta,\Phi)$ é a intensidade emitida pela fonte de luz na direção de P e (θ,Φ) indica essa direção.

5. Implementando na GPU

Em (1), o primeiro termo representa a luz que está sendo irradiada pelo objeto atenuada pela absorção das partículas suspensas, enquanto o segundo diz respeito ao espalhamento atmosférico. É fácil perceber que (1) sem o segundo termo nada mais é que o modelo de *fog* implementado nas placas gráficas [10]. Como o primeiro termo pode ser computado na placa gráfica, é nela que ele será avaliado, restando apenas o segundo termo. Por não existir uma solução analítica para a integral de (1), ela será avaliada numericamente.

A integral é avaliada fazendo uso de planos de amostragem. Cada plano sempre estará localizado em uma posição fixa relativa à câmera. Os planos são dispostos de forma que fiquem paralelos ao plano de projeção da câmera, e a distância entre eles é uniforme. Os planos de amostragem são formados a partir de várias geometrias do tipo GL_QUAD. Esta divisão é necessária pois adiante cada vértice de cada pedaço deverá ser tratado independentemente. A

Figura **3** mostra como os planos de amostragem devem ser dispostos e subdivididos. A idéia é utilizar cada plano para amostrar a luz que é espalhada na direção da câmera e, ao final, acumular a contribuição de cada um no *color buffer*.

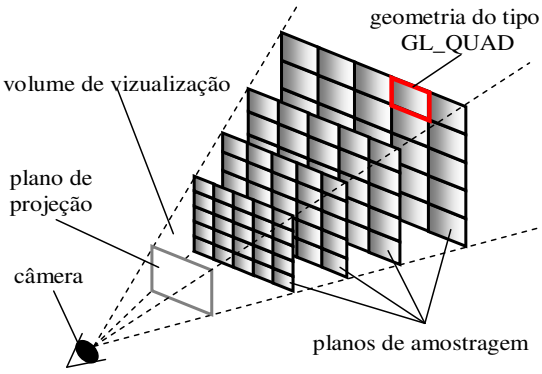


Figura 3. Planos de amostragem

Considerando-se I_s o termo relativo ao espalhamento atmosférico na equação (1), obtém-se a seguinte equação discretizando a integral:

$$I_s = \sum_{k=1}^{n} F(\alpha_k) H(r_k) I_p(r_k) B(t_k) \Delta t \quad , \qquad (5)$$

onde n é a quantidade de planos de amostragem e k é o k-ésimo plano. A equação (5) pode ser reescrita como

$$I_s = \sum_{k=1}^{n} I(\theta_k, \Phi_k) H(r_k) \xi(t_k) \, , \qquad (6)$$

onde

$$\xi(t_k) = \frac{F(\alpha_k) e^{-\rho(r_k + t_k)}}{r_k^{\,2}} \Delta t \quad . \qquad (7)$$

(6) possui três termos independentes que serão avaliados separadamente. Os dois primeiros, que representam respectivamente a função de iluminação e a função de visibilidade, são avaliados através de texturas. Já o último termo, representado por (7), será avaliado em cada vértice dos planos de amostragem.

A equação (8) é avaliada na CPU para cada vértice de cada plano de amostragem e, ao final, o resultado é passado na forma de uma cor através da função glColor. A vantagem de se utilizar este meio é que os *pixels* entre os vértices serão avaliados pela GPU através de uma interpolação. Isto é possível habilitando-se o modelo de *shading* de Gouraud.

A função de iluminação, $I(\theta_k,\Phi_k)$, será representada por uma textura, isto é, um mapa de iluminação. Ao utilizar um mapa de iluminação, a especificação da função de iluminação torna-se bastante flexível, pois apenas alterando-se uma imagem pode-se obter inúmeras e diferentes funções. Mesmo com tanta flexibilidade, é bom tomar alguns cuidados na escolha do mapa de iluminação, pois podem surgir problemas de *aliasing* em certas circunstâncias. Os mapas que possuem altas freqüências são os candidatos fortes a gerarem *aliasing*. Por exemplo, na Figura 4 o mapa (a) é muito bom; o (b) é razoável; e o mapa (c) é ruim. O mapa de iluminação é aplicado utilizando-se técnicas de mapeamento de texturas projetivas. [11] explica em detalhes como se deve proceder para criar texturas projetivas em OpenGL.

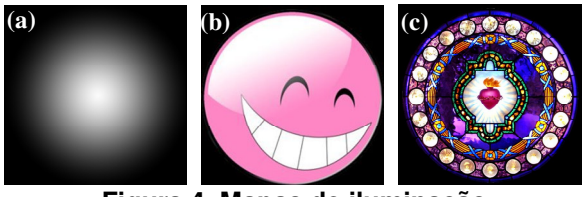
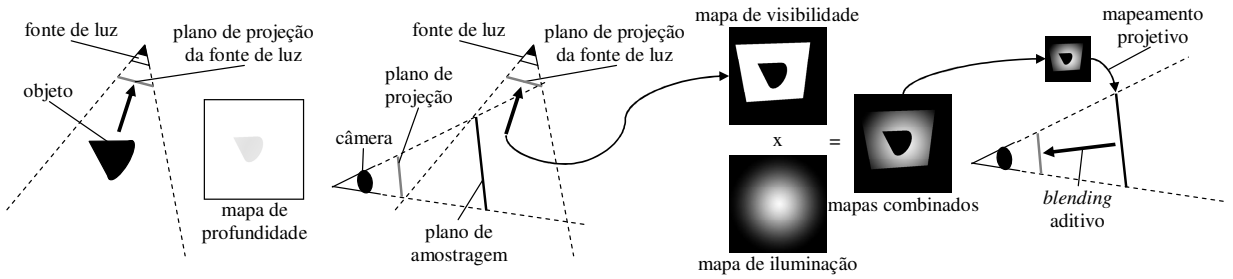


Figura 4. Mapas de iluminação

Os volumes de sombra que surgem quando existe algum objeto dentro do volume de luz são imprescindíveis para o realismo da simulação. A função de visibilidade é responsável pelo surgimento dos volumes de sombra. Esta função basicamente retorna zero caso um ponto não seja visível do ponto de luz, ou um caso seja visível. Da mesma forma que a função de iluminação, esta função será implementada utilizando-se texturas projetivas, sendo que para a função de visibilidade a textura será uma imagem em preto e branco. As partes em preto representam os pontos que não são visíveis, enquanto as em branco representam os pontos visíveis. Diferentemente da função de iluminação, a função de visibilidade necessita que seja criado um mapa de visibilidade (textura) diferente para cada plano de amostragem. A Figura 5 ilustra os passos necessários para a geração e aplicação destes mapas. Inicialmente em (a), um mapa de profundidade é gerado para os objetos da cena considerando-se a fonte de luz como ponto de vista. Este mapa é gerado renderizando-se apenas os objetos que devem gerar sombra e em seguida copiando-se os valores do *buffer* de profundidade para uma textura. Este mapa é único para todos os planos de amostragem e só necessita ser gerado uma vez no início de cada quadro. Em (b), o mapa de profundidade é aplicado sobre o plano de amostragem e a renderização desta vez é feita da câmera. A função de textura habilitada para este mapa é similar a utilizada na técnica de geração de sombras *shadow mapping*, onde os valores armazenados no mapa são testados com as profundidades de cada fragmento gerado da geometria (plano de amostragem), caso o teste seja satisfeito a cor branca é escrita, caso contrário a cor preta é escrita. O resultado dessa renderização já é propriamente o mapa de visibilidade que deve ser copiado para uma textura para ser utilizado posteriormente. Em (c), é ilustrado como o mapa de visibilidade é combinado com o mapa de iluminação, entretanto, o que de fato é feito é ligar cada textura a uma unidade de textura de OpenGL e escolher a função de textura GL_MODULATE para ambas. Por fim, em (d), o resultado da combinação dos mapas é aplicado sobra cada um dos planos de amostragem fazendo uso de técnicas de texturas projetivas.

6. Resultados

Os resultados (ver Figura 6) foram obtidos em um AMD Athlon 64 3200+ com 1024MB RAM e uma placa gráfica NVidia GeForce 6600 GT de 128MB.



(a) Criando mapa de profundidade **(b) Criando mapa de visibilidade** **(c) Combinando mapas** **(d) Mapeando mapa combinado**

Figura 5. Mapas de visibilidade

	n	QUAD	σ	*K*	Δt	*fps*	mapa
(a)	40	30x30	0.5	30	0.07	37	
(b)	55	10x10	0.2	25	0.05	28	
(c)	40	20x20	0.1	20	0.1	41	

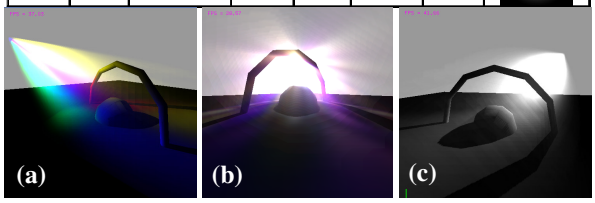


Figura 6. Resultados obtidos

7. Conclusões e Trabalhos Futuros

Este trabalho apresentou um modelo para simulação de iluminação volumétrica que faz uso da GPU para acelerar a simulação. Este modelo mostrou-se bastante adequado para ser utilizado em aplicações interativas, como os jogos eletrônicos.

Alguns tópicos não foram abordados neste trabalho, ficando como trabalhos futuros: tratar o caso em que a densidade das partículas é variável; utilizar *shaders* programáveis para aumentar a performance; tratar o caso em que a fonte de luz é onidirecional; combinar o efeito de iluminação volumétrica com o efeito de *bloom*. Utilizar técnicas de HDRR (*High Dynamic Range Rendering*).

Referências

[1] T. Nishita, Y. Miyawaki, e E. Nakamae, "A shading model for atmospheric scattering considering luminous intensity distribution of light sources", *In SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 1987, pp. 303–310.

[2] H. Jensen, e P. Christensen, "Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps", *In Proceedings of SIGGRAPH'98*, Orlando, July 1998, pp 311-320.

[3] Y. Dobashi, T. Yamamoto, e T. Nishita, "Interactive rendering method for displaying shafts of light", *In PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, 2000, pp. 31.

[4]. Y. Dobashi, T. Yamamoto, e T. Nishita, "Interactive Rendering of Atmospheric Scattering Effects Using Graphics Hardware," *Graphics Hardware*, 2002.

[5]. R. James, *Graphics programming methods*, Charles River Media, Rockland, MA, USA. 2003.

[6]. Y. Zhu, G. Owen, F. Liu, e A. Aquilio, "Gpu-based volumetric lighting simulation", *In CGIM '04: Proceedings of The 7th IASTED International Conference on COMPUTER GRAPHICS AND IMAGING*, IASTED/ACTA Press, Kauai, Hawaii, USA, 2004, pp. 479.

[7]. J. Mitchell, *ShaderX3: Advanced Rendering with DirectX and OpenGL*, Charles River Media, 2004.

[8]. V. Biri, S. Michelin, e D. Arques, "Real-Time Animation of Realistic Fog", *Thirteenth Eurographics Workshop on Rendering*, 2002.

[9] M. Gibbons, "Radiation Received by Uncollimated Receiver from a 4-PI Source", *Journal of the Optical Society of America*, vol. 48, issue 8, 1958, pp. 550-555.

[10]. M. Woo, J. Neider, T. Davis e OpenGL ARB, *Opengl Programming Guide: The Official Guide to Learning Opengl, Version 1.1*, Addison-Wesley Pub, 1997.

[11]. C. Everitt. "Projective Texture Mapping", 2001. Disponível em: http://developer.nvidia.com/object/Projective_Texture_Mapping.html. Último acesso: 16/08/2007.