Projeto 4: Surface Rendering

Marcos Teixeira

10 de junho de 2018

1 Descrição

Um modelo de iluminação é um conjunto de equações que determinam quantitativamente a cor em um ponto da superfície de um objeto como uma função das propriedades da superfície da superfície e da luz incidente diretamente e indiretamente no ponto. Um modelo de iluminação local é aquele que leva em conta apenas a luz emitida pelas fontes virtuais de luz da cena, ou seja, iluminação direta. Um modelo global de iluminação também leva em conta a iluminação indireta, isto é, proveniente da inter-relação e refração da luz entre os objetos da cena.

A determinação da energia luminosa em torno de um ponto é fisicamente desqualificada pela equação de radiância (difícil de ser resolvida), portanto, em computação simples, adotamos equações de iluminação não físicas mais simples. Neste trabalho pretendemos implementar um modelo simples chamado modelo de iluminação Phong. Aqui, iremos desconsiderar a característica de opacidade dos objetos.

2 Modelo de Iluminação de Phong

Um modelo local considera uma iluminação de um ponto de raios de luz provindos digitais como fontes de luz virtual. Um modelo local desconsidera a iluminação indireta resultante da inter-reflexo difusa e especular e da transmissao difusa e especular que ocorre entre superficies da cena e que afetam a iluminacao de um ponto. O modelo de Phong é o modelo local que considera apenas reflexao difusa e reflexao especular. A luz que chega aos olhos do observador combina a reflexão uniforme da luz ambiente com as reflexões difusa e especular da superfície visível do objeto.

Neste laboratório, iremos implementar o modelo de iluminação de Phong [1] um modelo simplificado no qual o observador e uma única fonte de luz se encontram na mesma posição, longe o suficiente do plano de visualização para obter uma projeção é ortogonal a cena. Como o objetivo é realizar um rendering de superfície, nós devemos interromper o algoritmo DDA quando encontrar um ponto p_i encontra uma superfície opaca de um objeto visível da cena. Um exemplo do processo de iluminação pelo modelo de Phong é apresentado na Figura 1.

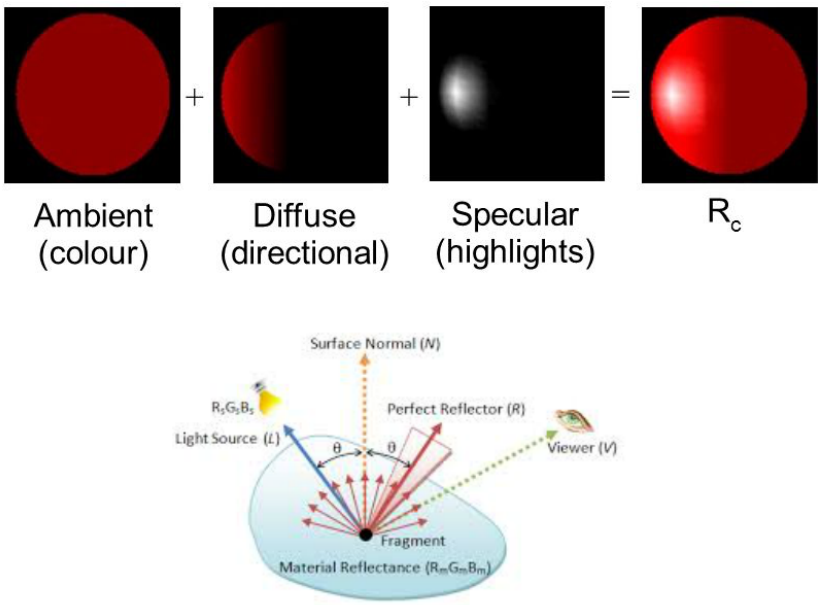

Figura 1: Exemplo do funcionamento do modelo de Phong e o resultado em um objeto 3D.

2.1 Luz ambiente

A iluminação e um ponto P derivada de da luz ambiente é

$$I = k_a L_a, \tag{1}$$

onde k_a, $0 \leq k_a \leq 1$ é o coeficiente de luz ambiente e L_a a cor de luz ambiente.

2.2 Reflexão difusa

Raios de luz que incidem em um ponto P de uma r=superfície refletem, após chocar-se com P, em uma direção arbitrária. Este fenômeno é proveniente da rugosidade do material da superfície, provocando a difusão dos raios de luz em torno do ponto P.

A iluminação de um ponto devido â iluminação difusa é definida pela lei dos cossenos de Lambert. No modelo de Phong, a iluminação da reflexão reflexiva é expressa pela seguinte equação:

$$I = k_d D(p) \cos\theta, \tag{2}$$

$$\cos\theta = (-N\dot{L}_l), \tag{3}$$

onde k_d, $0 \leq k_d \leq 1$ é o coeficiente de reflexão difusa, $D(p)$ é a tonalização baseada em profundidade (depth shading) e θ é o ângulo entre o vetor $\vec{v}$ e o vetor -$\vec{N}(p)$ normal a superfície em p. A tonalização pelo modelo de Phong só é executada para $0 \leq \theta \leq \pi/2$.

2.3 Reflexão especular

A reflexão especular ocorre em superfícies polidas e seu efeito depende da posição do observador. O componente especular só é calculado se $0 \leq \theta \leq \pi/4$. Esta característica é modelada no Phong através da seguinte equação:

$$I = k_s D(p) \cos 2\theta^{ns}, \tag{4}$$

$$\cos\theta = (-N\dot{L}_l), \tag{5}$$

A combinação das Equações 1, 2 e 6, para a iluminação direta de um ponto P, culminam no modelo geral de Phong descrito através de :

$$L(p) = k_a L_a + D(p)(k_d \cos\theta + k_s \cos 2\theta^{ns}), \tag{6}$$

3 Surface Rendering

Um modelo de Surface Rendering é usado para simular os efeitos da luz interagindo em uma superfície. O pipeline de execução do Surface Rendering utilizando o modelo de Phong pode ser expresso da seguinte maneira:

i Criar a imagem de saída R(D, D)

ii Criar a matriz de transformação M_{-1}

iii Definição das faces da cena

iv Criação da tabela de normais

v Criação do vetor de Depth Shading

vi Aplicar M_{-1} na matriz normal

vii Para cada ponto p_i do plano de visualização, faça

- (a) Aplicar M_{-1} em p_i
- (b) Se houver intersecção
  - i. Calcular os λ válidos que dam origem a $p1$ e pn
  - ii. Aplicar o DDA para percorrer os pontos $p_i \in p1, ..., pn$ em direção a cena
  - iii. Ao encontrar um ponto p_i válido da superfície do objeto
    - A. Calcular $I = k_a L_a + D(p)(k_d \cos\theta + k_s \cos 2\theta^{ns})$
- (c) Caso contrário, $R = 0$

viii Retornar R

, onde a matriz M_{-1} = T(N_x/2,N_y/2,N_z/2) * $R_x(theta_x)$ * $R_y(theta_y)$ * T(-D/2,-D/2,-D/2), com $D = \sqrt{nx^2 + ny^2 + nz^2}$. Como são várias estruturas que precisam ser manipuladas durante a execução do Surface Rendering, nós optamos por criar uma estrutura chamada GraphicalContext para facilitar a composição das funções. A estrutura do contexto gráfico é apresentado no pedaço de código 1.

```
typedef struct gc {
  int                numberOfObjects;
  ObjectAttributes  *object;
  PhongModel        *phong;
  iftMatrix         *Tinv;
  iftVector         vDir;
  iftImage          *tde;
  iftImage          *scene;
  iftImage          *label;
  iftVolumeFaces    *faces;
  iftImage          *normal;
} GraphicalContext;
```

Listing 1: Estrutura do Contexto Gráfico

Para cada uma das 6 faces, realizamos os cálculos apresentados na equação abaixo. Para cada um dos lambdas obtidos, nós pegamos o menor e o maior válidos para conseguir obter $p1$ e pn. O valor de menor lambda será atribuído a equação da reta para encontrar $p1$ e o maior lamba para encontrar pn. Note que é necessário garantir que $(\vec{n'} \cdot \vec{n_j})$ não sejam 0 para que os testes das equações acima funcionem.

$$\lambda_j = ((c_j - p_0) \cdot \vec{n_j})/(\vec{n'} \cdot \vec{n_j}),$$
$$p_{'} = p_0 + \lambda\vec{n}$$

Posteriormente, o algoritmo DDA será utilizado para encontrar todos os pontos da linha entre $p1$ e pn. Além de encontrar o primeiro ponto p_i válido de um objeto da cena, calculamos o valor de $L(p)$. Para tal, vamos calcular as informações necessárias para o cálculo do Phong. Primeiramente, calculamos a distancia d de p_i ao observado (ponto P0), dado pela distância euclidiana. Obtemos o valor da normal pré calculada no ponto p_i observado. A distância d transformada em brilho através de uma transformação linear $T(d)$ no qual, pixels mais distantes tem valores baixos e os mais pertos valores mais altos.

Uma vez tendo estas informações, chamamos a função *PhongShading*, que vai calcular o iluminação no ponto p_i encontrado. Calculamos inicialmente o valor o produto interno de V' e N, que resulta em $\cos\theta$. Avaliamos se $\theta \leq pi/2$ para ver se é possível realizar o shading e se $\theta \leq pi/4$ para verificar a necessidade do componente especular. Ao final teremos o valor de $L(p)$, então atribuímos este valor a imagem de saída (multiplicando por 255., pois os resultados do DDA são normalizados em [0,1]).

4 Implementação

Nesta seção são elencados alguns detalhes de implementação do projeto. Primeiramente, as constantes de Phong escolhidas foram as seguintes: $ka = 0.1, kd = 0.7, ks = 0.2ens = 5..$ O vetor de depth shading $T(d)$ foi construído a partir da distância máxima de um ponto P para a origem $p0$. Com isso, valores mais próximos da origem do vetor $T(d)$ tem maiores brilhos do que os mais próximos do fim de vetor. Também calculamos uma tabela de objetos, com atributos como sua cor, visibilidade, e opacidade.

As normais foram pré-calculadas e alocadas em uma tabela, para serem obtidas quando necessário. Uma função foi implementada para mapear o índice de um voxel com a tabela de normais. Experimentos com a estratégia de cálculo on-the-fly das normais também foram conduzidos, mas como demoraram demais (+10 min) para realizar o Rendering, optamos por descartar esta ideia. O cálculo das normais foi realizado de duas maneiras: (i) baseado na cena e (ii) baseado na transformada de distância euclidiana e mapa de rótulos. Em ambos os casos foi considerado um raior $r = 3$ para definir a vizinhança em 3D na etapa de busca de pontos da borda. Na estratégia baseada na cena, um voxel p pertence a borda de um objeto da cena quando:

$$\exists q \in A_1(p), L(q) \neq L(p) e L(p) > 0 \tag{7}$$

Se o contraste entre os objetos (incluindo fundo) for bom, o vetor normal $\vec{N}_p$ pode ser aproximado pelo gradiente G_p. O gradiente em 3D é computado em todas as 6 direções.

$$\vec{G}_p = \sum_{\forall q \in A(p)} [I[q] - I[p]] \tag{8}$$

Nesta estratégia é preciso sempre verificar se a normal $\vec{N}_p$ está apontando para dentro ou para fora o objeto.

Na estimativa baseada na transformada de distância D_E^p, associamos a cada voxel $p \in D_i$ a uma distância ao voxel mais próximo na borda do objeto cujo rótulo é $L(p) \neq 0$.

$$\hat{L} = (D_i, L), D_i \implies Z \tag{9}$$

$$\hat{D}_E = (D_i, D_p) \tag{10}$$

onde $\hat{D}_E$ armazena iterativamente a distancia quadrada para os valores da borda. Após obter a transformada de distâncias, realizamos um cálculo semelhante ao da Equação 8, mas agora com a diferença entre $\hat{D}_E(p)$ e I[p].

5 Resultados

Nesta seção apresentamos os resultados obtidos na imagem de teste **skull.scn** As imagens de saída são da forma (D, D), onde D é igual a $D = \sqrt{nx^2 + ny^2 + nz^2}$, representando a

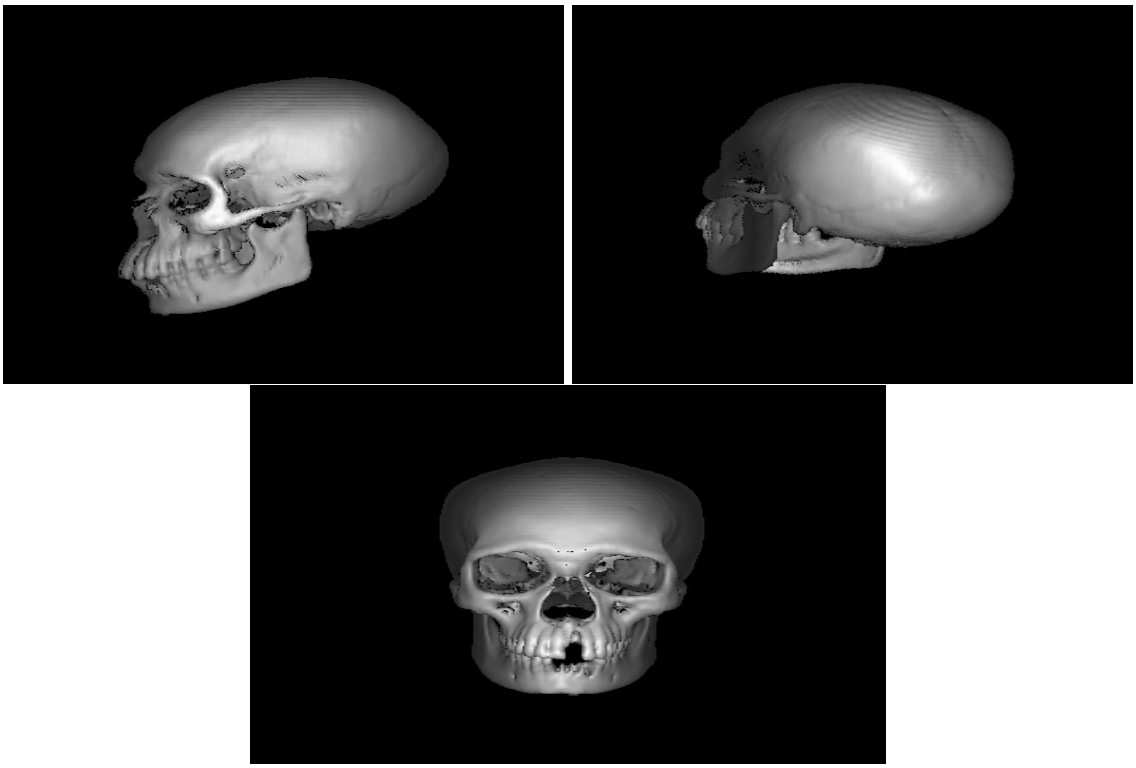

Figura 2: Resultados do Surface Rendering com iluminação de Phong utilizando estimativa da normal pela cena.

diagonal da imagem. Para padronizar as figuras nesse relatório, dado que esse D depende de cada imagem, nós limitamos a altura para 5cm e a largura para 7.5cm. Uma normalização de (0,255) é feita após a finalização do algoritmo de tonalização de Phong, para melhor visualização. Inicialmente, o plano de visualização é reposicionado em $z = D/2$, para afastar um pouco da visão do observador.

Na Figura 5 mostramos os resultados utilizando estimativa das normais pela cena, para rotações de $R_x = 90$, $R_y = 0$ e $R_x = 90$, $R_y = 90$.

Posteriormente, fizemos experimentos com a estimativa baseada na transformada de distâncias e mapa de labels. Os resultados estão apresentados na Figura 5, para rotações de $R_x = 90$, $R_y = 0$ e $R_x = 90$, $R_y = 90$. Para encontrar as bordas com TDE, utilizamos um raio $r = 3.0$ na função *iftSpheric*. Curiosamente, os resultados não ficaram melhores que utilizando a cena para obter as normais, dado que podemos observar alguns "buracos"na imagem. Para tentar solucionar este problema várias alternativas foram testadas, desde modificação dos raios quanto modificações no algoritmo do cálculo da normal. No entanto, não foi possível corrigir este problema.

Esta técnica, por requerer a transformada de distâncias, demora um pouco mais que a por cena. Nos casos onde não ocorrem buracos, a estimativa por transformada de distâncias

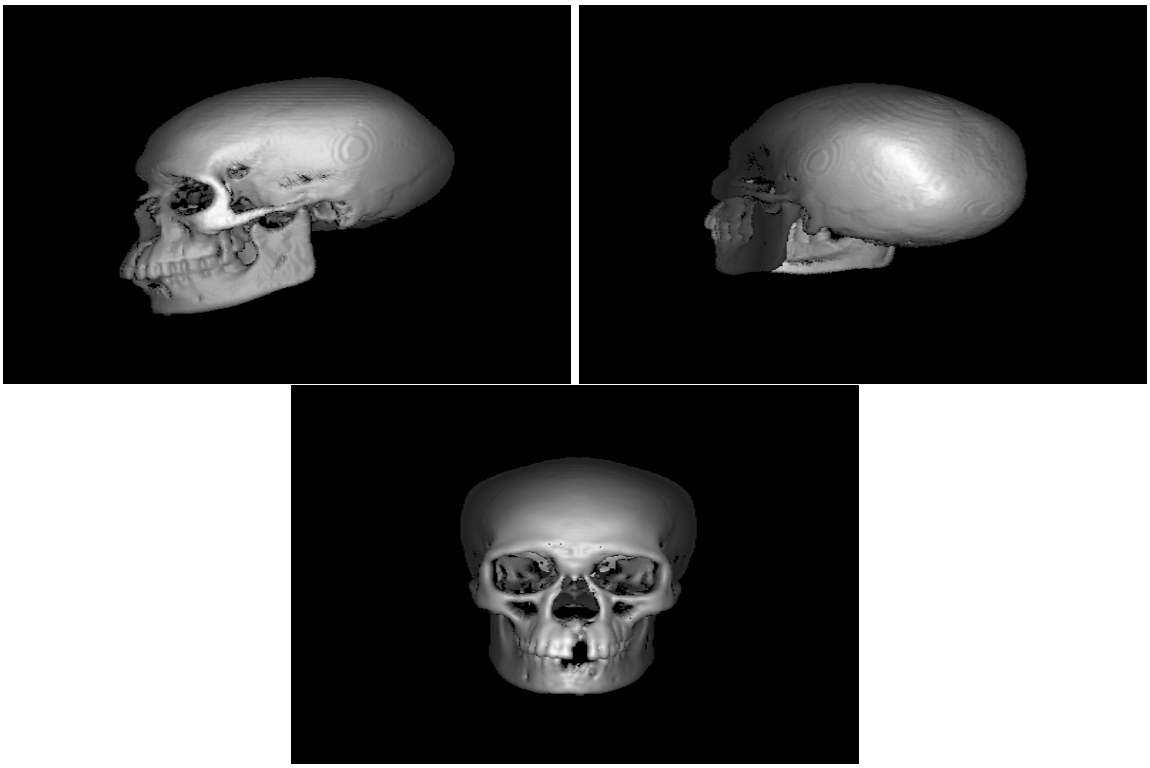

Figura 3: Resultados do Surface Rendering com iluminação de Phong utilizando estimativa da normal pela cena.

funcionou bem.

6 Conclusão

Neste trabalho foi apresentado uma implementação para o algoritmo de surface rendering baseado no modelo de iluminação de Phong. Através da teoria vista em sala de aula, fizemos uma modelagem que levasse em conta todos os passos do algoritmo de forma simples e objetiva. Os experimentos foram conduzidos levando em consideração duas formas de se realizar a estimativa das normais: pela cena e pela transformada de distâncias (mapa de objetos).

Este trabalho apresentou vários desafios ao longo de seu desenvolvimentom, muito mais que o MIP a Transformada de Radon. Muitos dos bugs só foram encontrados após vários dias e conversando com os colegas de disciplina. Alguns bugs ainda não foram solucionados por inteiro, sendo o mais notável os buracos na figura quando utiliza-se a estimativa das normais pela transformada de distância. Do ponto de vista conceitual, esse trabalho utiliza muito do que foi usado para o Projeto 2 (Maximum Intensity Project), o que facilitou um pouco o trabalho. De qualquer forma, foi um aprendizado relevante para concretizar a

teoria vista em sala.

Algumas observações técnicas para a implementação. Foi necessário utilizar a biblioteca compilada para Mac OS disponibilizada pelo monitor para o código. Além disso, mudamos também a versão do gcc no Makefile para o gcc-7, para poder funcionar.

Referências

[1] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, 1975.