Projeto Final
IA725 - Computação Gráfica I

106304 - Yu Tzu Wu

1o semestre 2016

1 Introdução

Este trabalho teve como objetivo montar uma cena foto realística em OpenGL. Procurou-se desenvolver a cena por meio de formas geométricas simples, reproduzindo alguns objetos do mundo real de simplicidade geométrica. Para conferir maior realismo à cena, focou-se no efeito da iluminação e na adição de texturas e de rugosidade aos objetos criados.

O trabalho foi desenvolvido no sistema operacional Windows 8.1 com o software Visual Studio Express 2013 for Desktop na linguagem C# e com o uso do Tao Framework.

2 Objetivos

1. Criar formas geométricas
   (a) Circunferência
   (b) Cilindro com raio variável
   (c) Cubo
   (d) Disco de raio variável
   (e) Esfera
   (f) Plano
   (g) Retângulo
   (h) Tetraedro
2. Criar objetos
   (a) Bolo
   (b) Chão
   (c) Copo
   (d) Folha
   (e) Janela
   (f) Livro
   (g) Mesa
   (h) Parede
   (i) Porta
   (j) Prato
3. Aplicar texturas sobre os objetos
4. Aplicar o modelo de iluminação Blinn-Phong
5. Aplicar os tipos de tonalização

(a) Flat (b) Gouraud (c) Phong

6. Aplicar a textura *bump mapping* sobre o objeto
7. Controle por esfera virtual
8. Função *auto rotate* da cena em torno do eixo Y global
9. Projeção perspectiva com 1, 2 e 3 pontos de fuga
10. Projeção perspectiva com vista frontal, de cima e de lado
11. Opção de mostrar ou não o sistema de referência do objeto

Os objetos foram criados explorando as formas geométricas. Por exemplo, os livros, a porta e as janelas são construídos como retângulos; o bolo e o prato, como cilindros; o chão e a parede como planos; e o copo, unindo os vértices de algumas circunferências. Neste projeto, os vértices dos objetos foram calculados e ordenados manualmente para formar as superfícies dos objetos.

3 Desenvolvimento

3.1 Objetos

Os objetos que constituem a cena são definidos pela classe Objeto. Essa classe contém como atributos as informações necessárias para renderizar o objeto em OpenGL e nos métodos os construtores dos objetos, as funções relacionadas ao OpenGL e demais funções comuns a todos os objetos.

Devido à grande quantidade de elementos na cena, optou-se por trabalhar com classes, pois isso facilita bastante na hora de alterar os atributos de cada objeto e de desenhá-los em tela.

3.2 Propriedade dos materiais

A classe Material define as propriedades k_a, k_d, k_s e *shininess* para o cálculo do efeito de iluminação. Antes de desenhar cada objeto em tela, as propriedades do material são enviadas para o shader.

As constantes k_a, k_d e k_s são definidas como vetores 3x1, cada elemento define o comportamento do material para os canais r, g e b. Quando o comportamento é o mesmo para os três canais, é possível passar somente valores escalares como parâmetros do construtor e este automaticamente gerará a forma vetorial das constantes k_a, k_d e k_s.

3.3 Iluminação

Na parte da iluminação, aplicou-se uma fonte de luz branca, localizada na posição (0, 10, 15), com o observador na posição (1, 1, 1), ambas as posições definidas no sistema de coordenadas global da cena.

Na cena, existe a possibilidade de escolher o tipo de tonalização sobre os objetos. O controle do tipo de tonalização a ser aplicada na cena é realizado por meio de duas variáveis booleanas: *flatShading* e *phongShading*. O primeiro define o tipo de normal de

cada vértice a ser enviado ao shader e o segundo determina se o modelo de iluminação será computado no vertex shader ou no fragment shader. O valor de ambas as variáveis é modificado automaticamente por meio das teclas 5 (Flat shading), 6 (Gouraud shading) e 7 (Phong shading).

A normal e a tangente de cada vértice foram calculadas matematicamente para as formas cubo, tetraedro, retângulo, plano e cilindro. Para a esfera, a normal foi aproximada para o caso de uma esfera ideal, ou seja, na direção do vetor raio para cada vértice. A tangente da esfera não foi calculada, assim como a normal e a tangente do copo. Um tratamento foi feito para não aplicar texturas sobre a esfera e o copo, pois para isso seriam necessárias as informações da normal e da tangente de cada vértice.

3.4 Textura

Neste projeto, foram utilizados dois tipos de texturas: um para enriquecer a aparência dos objetos e outro de *bump mapping* para conferir um aspecto de rugosidade aos objetos. As texturas se encontram na pasta "textures"do projeto e em geral foram baixados no Google Images, sendo alguns montados ou gerados manualmente. A seguir, são listadas todas as texturas utilizadas com a respectiva fonte.

bookCover0.jpg Link, acesso: junho 2016.

bookCover1.jpg Montagem das imagens Link Capa e Link Orelha, acesso: junho 2016.

bookCover2.jpg Montagem das imagens Link Capa (acesso: junho 2016) com a orelha da imagem bookCover0.jpg.

cake.jpg Recorte de Link, acesso: junho 2016.

door.jpg Link, acesso: junho 2016.

paper.jpg Link, acesso: junho 2016.

paperRecycled.jpg Link, acesso: junho 2016.

window.jpg Link, acesso: junho 2016.

wood2.jpg Link, acesso: junho 2016.

wood-floor.jpg Link, acesso: junho 2016.

bump.jpg Essa textura foi gerada no MATLAB.

bumpLeather.jpg Link, acesso: junho 2016.

bumpPaper.jpg Link, acesso: junho 2016.

bumpWall.jpg Link, acesso: junho 2016.

bumpWood.jpg Link, acesso: junho 2016.

bumpWood2.jpg Link, acesso: junho 2016.

3.5 Rotação

O controle de rotação automática em torno do eixo Y é feito pela tecla "w". Quanto ativado, o ângulo de rotação é definido por um *stopwatch*. A rotação também pode ser realizada por meio da implementação do controle por esfera virtual em torno de qualquer eixo.

3.6 Perspectiva

Na cena, não foi implementada a projeção ortográfica pelo fato de que o olho humano possui uma projeção perspectiva. Assim, buscou-se aumentar a impressão de realidade da cena implementando apenas a projeção perspectiva. As diferentes vistas foram obtidas alterando a posição da câmera, que atuaria como o observador que está olhando para a tela. No entanto, isso não afeta no valor da posição do observador utilizado para computar o efeito de iluminação. A seleção da vista é feita através do menu de alternativas (botão direito do mouse).

3.7 Sistema de coordenadas

Cada objeto possui o seu próprio sistema de coordenadas local criado no momento da construção do objeto. Além disso, existe a opção de gerar um sistema de coordenadas global por meio de um dos construtores da classe Objeto. Esses sistemas não são desenhados em tela por default, mas existe a opção de visualizá-los por meio do botão 2 do teclado.

Referências

[1] J. F. Hughes, A. van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner, and K. Akeley. *Computer Graphics: Principles and Practice*. Addison-Wesley, third edition, 2013.

[2] D. Shreiner, G. Sellers, J. Kessenich, and B. Licea-Kane. *OpenGL Programming Guide : The Official Guide to Learning OpenGL, Version 4.3*. Addison-Wesley, eighth edition, 2013.