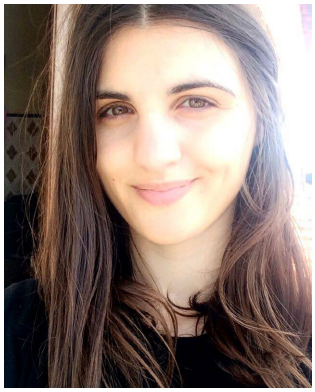
Escola de Engenharia
Universidade do Minho

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
Mestrado Integrado em Engenharia Informática
Computação Gráfica

Sistema Solar

Fase 4
Normals and Texture Coordinates

Grupo 18



Célia Figueiredo a67637

Luís Pedro Fonseca a60993

Nelson Carvalho Vieira a54764

Braga, 22 de Maio de 2017

Conteúdo

Resumo

O presente relatório descreve o trabalho efetuado para a realização da quarta fase do projeto, onde foi pedido a implementação e representação de um Sistema Solar com o uso da ferramenta *OpenGL*. Nesta fase é preciso implementar texturas nas diversas esferas para as conseguirmos destinguir umas das outras. É preciso criar também a iluminaçãoo dos diversos planetas e é para este ponto que é necessário calcular as normais das diversas formas geométricas.

1. Introdução

No âmbito da Unidade Curricular de Computação Gráfica pertencente ao plano de estudos do 3º ano do Mestrado Integrado em Engenharia Informática foi proposto o desenvolvimento de um sistema solar.

Concluída a terceira fase tem-se agora de tratar da implementação da iluminação e texturas. Para implementar a iluminação é necessário calcular as normais no generator e fazer alterações no *engine* de forma a conseguir utilizar as normais para iluminar a cena. Quanto às texturas será necessário calcular os pontos de textura para cada uma das superfícies, na parte do *generator* e, na parte do *engine* é necessário implementar a capacidade de carregar texturas a partir de um ficheiro e aplicar essa mesma textura num objeto, utilizando os pontos gerados pelo *generator*.

2. Iluminação

Com o objetivo de obter mais dinamismo do cenário, implementou-se iluminação. Para tal, e como todas as figuras estão a ser geradas ponto a ponto, sem recurso aos modelos já oferecidos pelo *OpenGL*, foi preciso especificar as normais de cada objeto. Além disso, foi necessário expandir as funcionalidades do *xml* para obter informação relativamente aos materiais dos objetos, ou seja a forma como reagem com a luz, bem como as luzes em si.

É com os vetores das normais que nós definimos como a luz é refletida por determinado objeto, isto é, com a emissão da luz de determinado ponto é necessário saber se quando esta luz atinge o objeto como é que o ilumina. Quando falamos de iluminação o existem vários tipos de iluminação que podemos usar para iluminar um objeto:

- Difusa;
- Ambiente;
- Especular;
- Emissiva;

2.1 Luzes

No nosso caso, fizemos a implementação que permite a um dado objeto ser incidido com um dos tipos de luz, para tal, temos no *xml* a *tag* luz. Na *tag* luzes definimos os tipos de luzes que são aplicadas ao planetas quando este recebem a luz que é emitida pelo sol, pois no sistema solar o sol não é ”iluminado”, mas sim uma fonte de luz. Para definirmos a luz no sol usamos pontos que se encontram ”dentro” da área representada pelo sol no *xml*. portanto um conjunto de luzes. que são desenhadas antes dos objetos do mesmo grupo. Utilizar-se-á uma luz para evitar aumentar a complexididade do sistema, como tal só se faz *enable* a *lightning* e *light0*.

2.2 Materiais

De forma a ajustar como os objetos reagem à luz, foi necessário introduzir materiais.

Os materiais são diferentes, pois nos materiais temos que cada modelo possui o seu próprio material. Para isso definimos uma estrutura de dados que nos permite aramazenar as características que cada modelo tem para o material, isto é, para cada material temos de armazenar o modelo em causa, qual o material associado a esse modelo, qual a cor desse modelo.

2.3 Normais

As normais servirão para se ativar a funcionalidade de iluminação de cada figura geométrica. É graças a estas que é possível determinar o ângulo de incidência da luz e como tal variar a intensidade observada. Foi necessário adicionar o glEnable(GL_NORMALIZE) para recalcular as normais após um *glScale*.

2.3.1 Esfera

Para a Esfera, as normais são os próprios pontos. Como as normais são um vetor não se torna necessário retirar o raio dos pontos. Desta forma, como as coordenadas do ponto são relativas ao seu centro, é exatamente esse vetor que define a sua normal.

```
int drawSphere_VBO(float radius, float centerX, float centerY,
float centerZ, int stacks, int slices, Ponto3D points, int* indexes,
Ponto3D normals){

// calculo do angulo de cada slice e stack, pasado para radianos
divBeta = (360.0f / (float) slices) * M_PI / 180.0f;
divAlpha = M_PI / (float) stacks;  // == (180.0f / (float) stacks) * M_PI

// criação dos arrays com os angulos todos os slices e stacks sendo que o
// os angulos alpha (do Norte ao sul) começam em 90 e vao até -90, i.e., 
alphas[0] = M_PI / 2.0f;
alphas[stacks] = - M_PI / 2.0f;
// os angulos beta (horizontal) começam no 0 até 360, i.e., de 0 a 2PI
betas[0] = 0;
betas[slices] = 0;

for(i = 1;i < slices;i++)
betas[i] = betas[i-1] + divBeta;
for(i = 1;i < stacks;i++)
alphas[i] = alphas[i-1] - divAlpha;

// Triangulos do topo da esfera
stHeightDown = radius * sin(alphas[1]);
stHeightUp = radius * sin(alphas[stacks - 1]);
radDown = radius * cos(alphas[1]);
radUp = radius * cos(alphas[stacks - 1]);
     Calcula o ponto do topo da esfera;
     Calcula a normal nesse ponto;
     Calcula o segundo ponto  esfera;
     Calcula a normal nesse ponto;
     Guarda pontos e normais;
indp += 2;
for(i = 1;i < slices; i++) {
    Calcula o ponto seguinte da stack da esfera;
    Guarda o ponto;
    Calcula a normal para esse ponto;
```

```
     Guarda a normal calculada
      Incrementa o indice;
       Desenha triâgulo;
      Incrementa o indice dos indices;
}
   desenha o último triangulo da stack;
   Incrementa o indice dos indices;

// resto da esfera
 for(i = 2;i < stacks;i++){
     Calcula o ponto seguinte da stack da esfera;
     Guarda o ponto;
 for(j = 1;j < slices;j++) {
      Calcula as normais dos quadrados a desenhar
       desenha quadrado ;
       indi += 6;
  }
     desenha quadrado ;
    indi += 6;
}

// Triangulos da base da esfera
stHeightUp = radius * sin(alphas[stacks - 1]);
radUp = radius * cos(alphas[stacks - 1]);
  Calcula o ponto do topo da esfera;
   Calcula a normal nesse ponto;
indp++;
for(i = 1;i < slices; i++) {
  desenha triangulo;
  indi += 3;
}
  desenha triangulo ;
  indi += 3;
return indp;
}
```

2.3.2 Plano

As normais do plano criado na horizontal, ou seja em que o y não varia, são fáceis de se calcular. As normais dos quatro vértices do apontam para 0 1 0.

2.3.3 Box

Para as normais dos oito vértices da box temos de ter em mente que cada vértice vai ter três normais diferentes, referente às três faces da box que estes fazem parte. Por isso especificou-se as normais por face. Cada face tem portanto as normais à semelhança do plano, com só uma das coordenadas a 1 e as outras a 0.

2.3.4 Cone

É preciso ter-se em atenção os ângulos dos diversos pontos do Cone. Conseguindo-se desenhar o Cone a diferençaa do x e z dos diversos pontos e as suas respectivas normais é que só precisamos dos senos e cosenos, não os multiplicamos pelo raio. Os y de todas as normais calculam-se da mesma maneira: divisão entre a altura do cone e a raiz quadrada entre a soma do quadrado da altura com o quadrado do raio.

2.3.5 Torus

Antes de calcular as normais do tórus, é necessário perceber o que deve ser esperado. Como não se estão a implementar sombras (uma objeto à frente da luz, escurecer o que está a trás de si, um tórus quando horizontal perante a luz, deve ter 2 superficies iluminadas. Ou seja, ao contrário de uma esfera que os seus vetores são relativos ao ponto central, um tórus deve ser analisado como um cilindro em que os vetores são relativos ao eixo à volta. Desta forma, consegue-se obter simultaneamente a parte exterior e interior iluminada de acordo com a posição da luz.

2.4 Texturas

As texturas são armazenadas de modo a que quando o ficheiro é lido saber o tipo de textura que o objeto vai ter.

Nas texturas temos duas vertentes, isto é, temos a aplicação de uma imagem como textura, mas temos também uma vertente, que é a definição do sistema solar que não é definir as texturas como imagens, mas sim ter uma espécie de material, isto é ter o sistema solar em que cada planeta possui um material definido por nós. O carregar de uma textura é feito através da leitura no *xml*de um campo textura que indica o nome da textura a carregar.

3. Conclusão

Dada por terminada esta quarta e última fase do trabalho consideramos que a maior parte dos pontos que eram pedidos foram implementados com alguma dificuldade.

Apesar de sabermos o que seria necessário para a implementação da funcionalidade das texturas, não conseguimos terminar essa tarefa com sucesso.

Concluindo, estas duas últimas fases não foram concluidas com sucesso, devido a problemas de implementação e falta de tempo, no entanto tentaremos que o projeto fique funcional até à altura da apresentação.