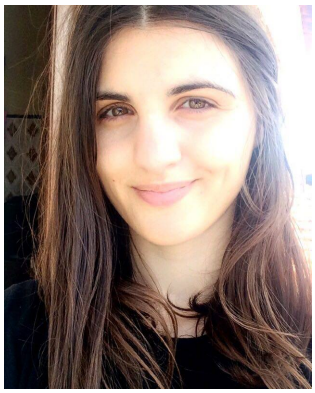
Escola de Engenharia
Universidade do Minho

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
Mestrado Integrado em Engenharia Informática
Computação Gráfica

Sistema Solar

Fase 3
Curves, Cubic Surfaces and VBOs

Grupo 18



Célia Figueiredo a67637

Luís Pedro Fonseca a60993

Nelson Carvalho Vieira a54764

Braga, 19 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 vamos precisar de 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.

2.1 Luzes

De forma a manter a simplicidade do código, criou-se uma nova classe Light que detém a posiçãoo da mesma. Esta define valores genéricos para as componentes difusa e ambiente, apesar de ser relativamente fácil obter estes a partir do xml tal como será abordado nos materiais pois essa funcionalidade foi utilizada lá. A classe Group detém portanto um conjunto de luzes. que são desenhadas antes dos objetos do mesmo grupo. Utilizou-se só 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. Para tal criou-se uma nova classe Material para albergar todas as informações necessárias que são as cores RGB para as componentes ambiente, difusa, especular e emissiva. O XMLParser proporciona portanto novas funções para extrair cada uma das componentes do .xml, colocando como placeholders 0.2 para cada cor ambiente, 0.8 para difusa, e 0 para as restantes. Para o sol utilizou-se as propriedades difusas a 0 e as emissivas a 1 para poder ser visível apesar da luz estar no seu interior e portanto não o afetar.

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.

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 posiçõa da luz.

2.3.6 Superficies de Bezier

O vector da normal em qualquer ponto da Superfície é definido pelo produto cruzado dos vetores das tangentes normalizadas.", ou seja, começa-se por calcular as tangentes em u e em v normalizando-se em seguida os vetores, por fim, faz-se o produto cruzado entre os dois vetores e obtem-se uma normal.

Para calcular as tangentes usa-se uma função que recebe como argumentos o valor de u e v a matriz m os pontos da patch de bezier e um inteiro de serve para indicar se estamos a calcular em u ou em v, esta funçao devolve-nos um ponto, referente a x, y ou z que é passado na matrix p. Cross e normalize, assim como o nome indica são as funções do cálculo do produto cruzado e da normaliza¸c~ao. De real¸car que para a normalização, verifica-se se o divisor é diferente de 0, pois se for 0 será devolvido o erro de not a number, pois é impossivel dividir por zero. A funçãoo bezierTangent calcula os diversos pontos em u e em v. De realçar que os pontos são guardados numa queue de nome derivada para serem posteriormente escritos no ficheiro .3d.

2.4 Texturas

As texturas são só obtidas quando no xml se pode encontrar um atributo texture num file e para as representar deveria-se ter utilizado uma classe Texture para não sobrecarregar o Group.

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 sucesso. Apesar de as texturas terem sido aplicadas com sucesso relativamente às esferas, visto que era o necessário para a correta representaçãoo do sistema solar, o mesmo não se verifica relativamente ao resto das figuras pois, por exemplo, o cone e o cilindro requerem texturas mais complexas devido à sua forma geométrica e, não eram pontos cruciais para terminar com sucesso o projeto. Para o movimeto do cometa foi utilizada com sucesso uma curva de catmull-rom com vários pontos de controlo, contudo, por falta de tempo, o cometa não muda a sua orientação ao longo da curva. Apesar destes pequenos pontos em falta real¸cam-se alguns pontos, como por exemplo, o facto de ser possível utilizar uma imagem como background da cena, imagem essa que é aplicada a uma esfera que engloba a câmara e se move com a mesma.

Concluindo, tentamos sempre manter um código relativamente simples e fácil de expandir por forma a acrescentar novas funcionalidades. Para além disso a performance do engine é boa, já que o número de fps ronda o limite de 60 quando está correr o sistema solar.