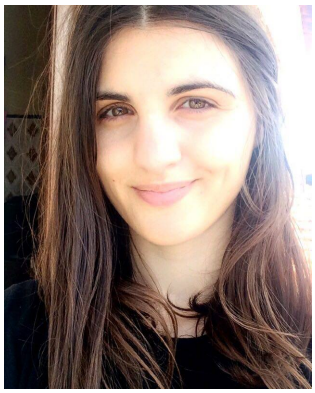
Escola de Engenharia
Universidade do Minho

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, 1 de Maio de 2017

Conteúdo

Resumo

O presente relatório descreve o trabalho efetuado para a realização da terceira fase do projeto, onde foi pedido a implementação e representação de um Sistema Solar com o uso da ferramenta *OpenGL*. Nesta fase serão utilizadas VBOs para permitir uma melhor perfomance da aplicação. Neste relatório não só é abordada a forma como foram adicionados os VBOs, mas também é abordada a forma como se implementou a rotação dos planetas à volta do Sol, e a própria rotação dos planetas, usando para tal curvas de Catmull-Rom com vários pontos de controlo. Por fim é falado sobre a criação de uma superficie de Bezier.

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 segunda fase tem-se agora de introduzir VBOS, este método consegue aumentar consideravelmente a performance de uma aplicação pois reduz o número de acessos à memória do computador, visto que todos os pontos são carregados para um buffer no GPU, sendo isto dos pontos mais importantes desta terceira fase do projeto.

Por forma a tornar o sistema solar mais próximo do real foram implementadas funcionalidades no *engine* que permitem definir pontos de controlo de uma curva de *Catmull-Rom* para um dado grupo, sendo assim possível criar curvas que os planetas devem percorrer de forma a realizar a translação à volta do sol.

Além dos pontos de controlo da curva que devem seguir também é possivel definir quanto tempo a translação deverá demorar. O último ponto desenvolvido é a utilização de superfícies de *Bezier*.

2. VBOs

Uma das melhorias desta fase em relação à primeira foi a introdução de VBOs no projeto. Com isto a criação das variadas figuras geométricas torna-se mais eficiente.

As definições que foram apresentadas anteriormente para as primitivas são do tipo imediato isto é para cada ponto é usada a função do *OpenGL glVertex3f()*. Contudo o *OpenGL* permite a utilização de *buffers* de arrays. Nestes podemos organizar os atributos dos vértices (coordenadas, coordenadas de texturas e normais). O procedimento a seguir com VBOs é o seguinte:

1. Ativar buffers (glEnableClientState(GL VERTEX ARRAY);
2. Alocar e preencher os arrays;
3. Gerar VBOs;
4. Preparar desenho dos VBOs;
5. Desenhar com VBOs.

A estrutura de dados da implementação em VBO's é semelhante à implementada nos modelos, mas existe uma pequena alteração, pois quando estamos nos modelos temos os pontos que vamos usar numa estrutura definida por nós (Ponto3D), em VBO's temos de seguir o que é pré-definido, sendo que os pontos são passados em arrays de valores.

De seguida pode ser visualizada a estrutura de armazenamento das figuras em VBO's

```
struct sVBO {
char name[256];
GLuint *buffers;
unsigned short indices;
int n_ind;
}
```

3. Curvas de Catmull-Rom

A matriz M utilizada para o cálculo da suavidade da curva é a seguinte:

$$\text{float m[4][4]} = \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1.0 & -2.5 & 2 & -0.5 \\ -0.5 & 0.0 & 0.5 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix}$$

3.1 Translações

As translações à volta de um ponto, que são usadas para os planetas à volta do sol, e os satélites à volta dos planetas têm por base curvas de *Catmull-Rom*. Para tal, são definidos pelo menos quatro pontos de controlo no *xml* e, a cada instante, o objeto é deslocado para a próxima posição nessa curva.

```
currentTime = sum(glutGet(GLUT_ELAPSED_TIME))
t=(currentTime%period)/period
```

Ou seja, calculando a divisão entre o resto do tempo atual com a duração do periodo e a duração do período obté-se um valor entre 0 e 1 que permite calcular ciclicamente qual o próximo ponto onde, dentro da curva, o objeto se deve encontrar.

Para além do tempo, só os pontos de controlo, que permitem gerar a curva, é que são precisos guardar na estrutura de dados.

4. Rotações

As rotações à volta de um eixo, são usadas para a rotação dos planetas em torno de si mesmos. Para tal, a cada iteração é necessário ajustar o ângulo da rotação.

```
currentTime = sum(glutGet(GLUT_ELAPSED_TIME))
angle=((currentTime%period)/period) *360
```

Ou seja, como anteriormente, calculando a divisão entre o resto do tempo atual com a duração do periodo e a duração do periodo obtém-se um valor entre 0 e 1 que é de seguida multiplicado por 360 para obter o ângulo da rotação.

5. Superfícies de Bezier

Nesta terceira fase foi proposta a criação de Superfícies de *Bezier*, através de um ficheiro passado como argumento, com as *patches* e os *control points* necessários para a criação da superfície, com o qual o *generator* deve criar a lista de vértices necessários para a criação da superfície. Para ser possível criar as superfícies teve de se fazer algumas alterações no *generator*, assim como novas funções que permitem gerar os vértices pretendidos.

5.1 Ficheiro

O ficheiro, passado como argumento no *generator*, contém diversas linhas com significados diferentes. A primeira linha contem o número, n, de *patches* necessárias para a criação das superficies, as n linhas seguintes são as linhas das *patches*, cada uma destas aponta para 16 *control points* cada. A seguir aparece uma linha com o número, m, de *control points* presentes no ficheiro, porém, as últimas m linhas são as linhas dos *control points*, cada uma destas linhas contém três valores diferentes. Se aparecer um ficheiro da seguinte forma:

```
2
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
3, 16, 17, 18, 7, 19, 20, 21, 11, 22, 23, 24, 15, 25, 26, 27
16
1.4, 0, 2.4
1.4, -0.784, 2.4
0.784, -1.4, 2.4
0, -1.4, 2.4
1.3375, 0, 2.53125
1.3375, -0.749, 2.53125
0.749, -1.3375, 2.53125
0, -1.3375, 2.53125
1.4375, 0, 2.53125
1.4375, -0.805, 2.53125
0.805, -1.4375, 2.53125
0, -1.4375, 2.53125
1.5, 0, 2.4
1.5, -0.84, 2.4
0.84, -1.5, 2.4
0, -1.5, 2.4
```

Sabemos que há duas patches:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
```

e

```
3, 16, 17, 18, 7, 19, 20, 21, 11, 22, 23, 24, 15, 25, 26, 27
```

Haverá também dezasseis *control points* onde 1.4, 0, 2.4 corresponde ao *control point* de índice 0 e 1.4, -0.784, 2.4 *control point* de índice 1 e assim sucessivamente.

5.2 Criação da Superfície

Para o cálculo dos vértices é usada a seguinte fórmula:

$$B(u,v) = [u^3 \quad u^2 \quad u \quad 1]M\begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

Figura 5.1: Fórmula da superfície de Bezier

Portanto utilizou-se a seguinte matriz M:

float m[4][4] = $\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

Para cada vértice precisamos de um valor em x, outro em y e por fim em z. Para tal ser possivel, vamos ter que calcular B(u,v) três vezes, uma com a matriz de todos os x de uma dada *patch*, outra com a matriz de pontos de todos os y e a última para a matriz de pontos de todos os z.

6. Demo

Neste fase atualizou-se o demo para, ao invés de uma estrutura estática, apresentar movimento. A principal alteração foi animar os objetos. Para tal, introduziu-se extensões aos elementos *translate* e *rotate*, um atributo tempo descreve a duraçãoo para cada rotação ou translação em 360^o.

Assim um planeta pode estar afeto por uma rotação (inclinação), uma velocidade de rotação (rotação em torno de um eixo próprio), uma translação (em redor a um ponto) e uma deslocação (translação). Isto com o minímo de dois grupos, visto que cada grupo só pode conter uma translação e uma rotação.

7. Conclusão

Concluídos três quartos do projecto, estamos agora numa fase em que já temos VBO's, capacidade de criação de superfícies de Bezier rotações e translaçõesdos planetas e das luas. Foram realizadas mais melhorias a nível da câmara, para que a visualização do demo fosse mais fluída tendo em conta as distâncias. Com a implementação das extensões à rotação e translação, permitiu que o modelo fosse mais próximo da realidade, faltando só iluminação e texturas para ter o mínimo necessário à realização do projeto. Ao longo da execução do projeto foram encontrados algumas difuculdades a nível de implementação, maioritariamente devido a impossibildade de comparência a determinadas aulas teórico-práticas. Em suma, esta fase, exigiu melhorias e adições às funcionalidades já implementadas,