Universidade do Minho

Mestrado Integrado em Engenharia Informática

Computação Gráfica

3ª Fase

Grupo 17

Daniela Fernandes	José Gomes	Ricardo Costa	Tiago Rodrigues
A73768	A82418	A85851	A84276

2 de maio de 2021

Conteúdo

1 Introdução

O trabalho prático da unidade curricular de Computação Gráfica tem como base a utilização do OpenGl, recorrendo à biblioteca GLUT, para a construção de modelos 3D.

A produção dos modelos envolve as diferentes temáticas abordadas nas aulas, como as transformações geométricas, curvas, superfícies cúbicas, iluminação e texturas. Com a realização deste projeto, dividido em quatro fases, tem como objetivo consolidar todos estes tópicos.

Este relatório diz respeito à realização da terceira fase do projeto mencionado, que tem como assunto principal as curvas, as superfícies cúbicas e VBOs.

2 Descrição do problema

Para esta fase o objetivo é o desenvolvimento de novas funcionalidades aplicadas ao **Gerador** e ao **Motor** que desenvolvemos nas fases anteriores, utilizando *Bazier patches* e curvas de *Catmull-Rom*, respetivamente.

Os requisitos para esta fase são:

- **Gerador:** os parâmetros que recebe são o tipo da primitiva gráfica, parâmetros relativos ao modelo e nome do ficheiro onde vão ser guardados os vértices.
  - bezier file_bezier_patch tessellation file.3d : vai calcular os pontos necessários para desenhar a primitiva de acordo com o ficheiro que contém o *bezier patch* e guarda-os num ficheiro de acordo com as fórmulas de *Bezier*.
- **Motor:** vai ler o ficheiro *XML* que contém todos os modelos das primitivas, vai desenhar os mesmo com os **VBOs**, realizar as animações com rotações e translações e desenhar curvas de **Catmull-Rom**.
- **Modelo do Sistema Solar:** é a representação dinâmica do sistema solar.
  - como primitivas para representar o sol, planetas, satélites e cometa, utiliza a esfera, o anel e o teapot
  - para as trajetórias dos planetas em volta do sol, dos satélites em volta do planeta e a trajetória do cometa utilizamos as curvas de *Catmull-Rom*
  - para as rotações do sol, dos planetas e dos satélites sobre si mesmo, utilizamos o tempo.

3 Resolução do Problema

Para a resolução desta fase vamos alterar o **gerador** e o **motor** que criamos para as fases anteriores.

3.1 Gerador

O Gerador nesta fase irá gerar pontos para um novo modelo, o *teapot*, através de pontos de controlo presentes num ficheiro. Este modelo denomina-se de *Bezier Patch*. O ficheiro com os pontos de controlo e outras informações relativamente à sua distribuição e número total de pontos é fornecida pela equipa docente.

3.1.1 *Bazier*

Além do ficheiro onde se estão presentes os pontos de controlo já mencionados, é também necessário fornecer um nível de tecelagem e um ficheiro para onde irão ser escritos os pontos gerados para depois ser passado ao Motor, à semelhança das outras primitivas.

Começamos por ler o ficheiro com os pontos de controlo fornecido e a guardar as informações lá presentes em estruturas de dados (no nosso caso usamos *vector*). Nesta leitura do ficheiro identificamos então quantos *patches* tem o ficheiro (um *patch* é um conjunto de 16 pontos de controlo neste caso) e guardamos os índices de cada ponto de controlo num vetor de vetores. Cada vetor nesta estrutura corresponde a um *patch*. Seguidamente, é identificado o número total de pontos de controlo e guardamos estas coordenadas de cada ponto numa outra estrutura (um simples *vector*).

Depois de executada a leitura do ficheiro, o próximo passo é percorrer cada um dos *patches* no nosso vetor de vetores e vamos guardar em 3 matrizes 4 por 4 diferentes (para as coordenadas X, Y e Z) os pontos que estão guardados no vetor nos índices do *patch*. As matrizes são 4 por 4 porque cada *patch* tem 16 pontos de controlo como foi referido. Isto é realizado da seguinte forma:

```
indice = patches[i][k + (4 * j)];

x[j][k] = points[indice * 3];
y[j][k] = points[(indice * 3) + 1];
z[j][k] = points[(indice * 3) + 2];
```

Dado que as coordenadas são armazenadas de 3 em 3, é necessário para cada ponto multiplicar o valor do índice presente no vetor dos pontos por 3 para a coordenada X. Para as outras duas é somar este valor a 1 e a 2, para as coordenadas X e Y, respetivamente.

Podemos agora finalmente realizar as contas necessárias para podermos obter os pontos para desenhar a primitiva. Para isto iremos utilizar os seguintes dados:

- $steps = 1/tecelagem$
- $u\_step = u * steps, u = 1..tecelagem$
- $v\_step = v * steps, v = 1..tecelagem$
- $M = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
- $M^T = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
- $value\_x = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} * M * \begin{pmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{pmatrix} * M^T * \begin{pmatrix} v^3 \\ v^2 \\ v \\ 1 \end{pmatrix}$
- $value\_y = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} * M * \begin{pmatrix} y_{00} & y_{01} & y_{02} & y_{03} \\ y_{10} & y_{11} & y_{12} & y_{13} \\ y_{20} & y_{21} & y_{22} & y_{23} \\ y_{30} & y_{31} & y_{32} & y_{33} \end{pmatrix} * M^T * \begin{pmatrix} v^3 \\ v^2 \\ v \\ 1 \end{pmatrix}$
- $value\_z = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} * M * \begin{pmatrix} z_{00} & z_{01} & z_{02} & z_{03} \\ z_{10} & z_{11} & z_{12} & z_{13} \\ z_{20} & z_{21} & z_{22} & z_{23} \\ z_{30} & z_{31} & z_{32} & z_{33} \end{pmatrix} * M^T * \begin{pmatrix} v^3 \\ v^2 \\ v \\ 1 \end{pmatrix}$

Tendo estas fórmulas, iremos percorrer os valores de u e v de 1 até ao valor da tecelagem e para cada iteração iremos calcular 4 pontos que são os pontos que vão ser escritos para ficheiro. Estes 4 pontos correspondem a 2 triângulos que formam um quadrado. Os valores de *u_step* e *v_step* são os valores que são utilizados juntamente com os valores das matrizes 4 por 4 para cada coordenada (X, Y e Z) para calcular as coordenadas de cada um dos pontos do *teapot*. Não esquecer que a ordem pela qual os pontos são escritos para ficheiro importa e por isso usamos a regra da mão direita para saber por qual ordem escrever.

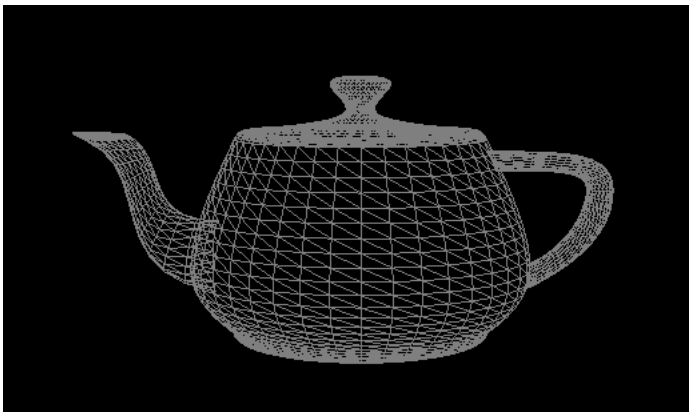

Figura 1: *Teapot* construído através dos *Bezier Patches* com um nível de tecelagem 10

3.2 Motor

Nesta fase foi dividido o Motor em duas partes diferentes:

- **catmullRom.cpp :** contém as funções relativas às curvas de *CatmullRom* para ser possível desenhar as trajetórias dos planetas e fazê-los movimentar nessa trajetória.
- **engine.cpp :** contém tudo o que estava na fase anterior, desde a leitura do ficheiro de configuração, leitura dos ficheiros ".3d" e o desenho das primitivas agora com *VBOs*. Além disso, contém também as funções que permitem realizar as animações através das funções presentes no *catmullRom.cpp*.

3.3 Leitura das novas transformações

Dado que nesta fase existem novas transformações (translação e rotação com animação), foram construídas duas novas funções para ler estas transformações que têm um tempo associado, **load_Translate_Animation** e **load_Rotate_Animation**. Estas funções apenas são executadas se o *translate* e o *rotate* no ficheiro de configuração tiverem associado um tempo, caso contrário, é feita a leitura do modo como era feito na fase anterior para a rotação e translação estática. Não esquecer que não pode haver mais que uma translação ou rotação associada a um grupo, logo ou existe a translação/rotação estática ou estas novas animadas.

3.4 Alterações à estrutura de dados

Para poder realizar a passagem do desenho de modo imediato para o desenho com VBOs foi necessário realizar algumas alterações às estruturas onde os dados do ficheiro de configuração e ficheiros ".3d" são armazenados. Assim, na estrutura *Group* da fase anterior foi

alterado o vetor de *Points* para um vetor de *floats* e foi adicionado um *buffer* e uma outra variável que representa o tamanho relativas ao VBO.

```
struct Group{
    Geometric_Transf transform;
    vector<string> models;
    vector<Group> subgroups;
    vector<float> vertexB;
    GLuint coords[1];
    GLuint num_vertices;
};
```

Também foi necessário realizar alterações à estrutura *Geometric_Transf* para poder lidar com as novas transformações relativas à animação. Para isso foram adicionadas novas variáveis para armazenar os dados relativos a estas transformações.

```
struct Geometric_Transf{
....

/* Translate with Animation */
    float ta_time;
    vector<Point> ta_points;
    float ta_Y[3];
    int ta_turns;

    /* Rotate with Animation */
    float ra_time;
    float ra_x, ra_y, ra_z;
};
```

3.5 VBOs

Para poder implementar o desenho das primitivas através de VBOs foram realizadas alterações na fase de carregar os pontos dos modelos presentes nos ficheiros ".3d" para o respetivo grupo. Assim, são carregadas todas as coordenadas para um vetor de *floats* e é atualizado o contador de vértices. Depois são apenas realizadas as seguintes instruções:

```
glGenBuffers(1, g.coords);
glBindBuffer(GL_ARRAY_BUFFER, g.coords[0]);
glBufferData(GL_ARRAY_BUFFER, g.vertexB.size() * sizeof(float), g.vertexB.data(),
↪  GL_STATIC_DRAW);
```

Depois na fase de desenho das primitivas são adicionadas as seguintes instruções, ao invés de percorrer o vetor onde antigamente estavam guardados os vértices:

```
 glBindBuffer(GL_ARRAY_BUFFER, g.coords[0]);
glVertexPointer(3, GL_FLOAT, 0, 0);
glDrawArrays(GL_TRIANGLES, 0, g.num_vertices);
```

Do modo como realizamos este processo, cada grupo possui um VBO associado para todos os modelos que pertencerem a esse grupo.

3.6 Curvas de *Catmull-Rom*

3.6.1 Posição do objeto

Para criar as trajetórias dos planetas com as curvas de *Catmull-Rom* foi seguida a estratégia do guião 8 e foram utilizados os seguintes dados:

- $T = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix}$
- $M = \begin{pmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$
- $P = \begin{pmatrix} P_0x & P_1x & P_2x & P_3x \\ P_0y & P_1y & P_2y & P_3y \\ P_0z & P_1z & P_2z & P_3z \end{pmatrix}$

A matriz P corresponde a 4 pontos que pertencem à curva que são fornecidos.
Tendo estes dados, podemos aplicar as seguintes fórmulas para obter a posição do objeto:

$$A = P * M$$
$$P(t) = A * T$$

3.6.2 Matriz de rotação do objeto

É necessária calcular a derivada da posição do objeto para podermos obter a matriz de rotação que alinha o objeto com a curva. Para isto, são utilizadas as seguintes fórmulas (A é calculado através da fórmula anterior):

$$T' = \begin{pmatrix} 3t^2 & 2t & 1 & 0 \end{pmatrix}$$
$$P'(t) = A * T'$$

Antes de obter a matriz de rotação é antes necessário calcular o seguinte (para estes considerou-se $Y_0 = (0, 1, 0)$):

$$X = ||P'(t)||$$

$$Z = \frac{X * Y_{i-1}}{||X * Y_{i-1}||}$$

$$Y_i = \frac{Z * X}{||Z * X||}$$

Tendo estes valores podemos agora recorrer à função *buildRotMatrix* (também existente no guião 8) e depois a **glMultMatrixf** para multiplicar a matriz de rotação obtida pela matriz atual.

3.6.3 Cálculo do *t global*

A função **getGlobalCatmullRomPoint** necessita do *t global*. Este valor é calculado através dos seguintes parâmetros:

- *elapsedT* : tempo que passou desde que o programa foi inicializado.
- *actualT* : tempo que passou desde que a volta de um planeta foi inicializada.
- *ta_turns* : número de voltas realizada pelo planeta desde que o programa foi inicializado
- *ta_time* : intervalo de tempo presente no ficheiro de configuração que o planeta deve demorar a completar uma volta.

Com estes parâmetros os valores do *actualT* e *t global* são calculados da seguinte forma:

$$actualT = elapsedT - ta\_time * ta\_turns$$

$$tglobal = \frac{actualT}{ta\_time}$$

O valor do *elapsedT* é obtido através de *glutGet(GLUT_ELAPSED_TIME) / 1000.0*.

3.6.4 Desenho das linhas da trajetória

As linhas das trajetórias dos planetas e do cometa são desenhadas na função **renderCatmullRomCurve** que recebe um conjunto de pontos pertencentes à curva (que são fornecidos no ficheiro de configuração) que depois são passados a outra função, **getGlobalCatmullRomPoint** que os usa para calcular os diferentes pontos pertencentes à curva (no nosso caso 1000 pontos). Estes pontos todos são conectados através do *GL_LINE_LOOP* da seguinte forma:

```
glBegin(GL_LINE_LOOP);
    for (int i = 0; i < 1000; i++) {
        getGlobalCatmullRomPoint( (float) (i/1000.0), points, pos, deriv);
        glVertex3f(pos[0], pos[1], pos[2]);
    }
glEnd();
```

3.6.5 Rotações dos planetas

As rotações permanentes dos planetas foram realizadas da seguinte forma:

```
float ang = 360 / (g.transform.ra_time * 1000);

float elapsedT = glutGet(GLUT_ELAPSED_TIME);

glRotatef(elapsedT * ang, g.transform.ra_x, g.transform.ra_y, g.transform.ra_z);
```

O tempo de uma rotação completa é também fornecido no ficheiro de configuração (*g.transform.ra_time*). O ângulo de rotação é então calculado dividindo os graus de uma volta completa (360^{o}) pelo tempo em milissegundos que um planeta é suposto dar uma volta completa e multiplicando este valor pelo tempo que já passou desde o início do programa. Assim, temos o planeta a rodar permanentemente.

4 Alterações no Modelo do Sistema Solar

Nesta fase, além das transformações com animação já faladas ao longo deste relatório, foi também adicionado um cometa ao modelo do Sistema Solar, através da primitiva *teapot* que é gerada através do *Bezier Patches*. Este cometa irá possuir uma trajetória também definida com as curvas de *Catmull-Rom* tal como as trajetórias dos planetas.

Tal como foi descrito nas fases anteriores recorreu-se às translações, rotações e escalas para representar o Sistema Solar, mas foram feitas alterações significativas às proporções reais do sistema para ser possível melhor visualizar o modelo.

Não esquecer que, tal como na fase anterior, as modelos de um sub-grupo herdam as transformações que forem aplicadas ao grupo em que estão inseridos, sendo necessário ter isso em conta ao realizar transformações nesses ub-grupos. Isto significa que para um satélite de um planeta temos de dividir os valores das rotações, translações e escalas pelo valor da escala do planeta.

Para as trajetórias dos planetas definidas através das curvas de *Catmull-Rom* calculámos 8 pontos para ser inseridos no ficheiro de configuração com base na distância dos planetas ao

Sol ou dos satélites relativamente ao seu respetivo planeta. Estes 8 pontos foram calculados da seguinte forma, em que r é o raio de uma circunferência e a distância de um planeta ao centro do Sol, por exemplo:

- **Ponto 1:** $x = r$ e $z = 0$
- **Ponto 2:** $x = \sqrt{\frac{r^2}{2}}$ e $z = \sqrt{\frac{r^2}{2}}$
- **Ponto 3:** $x = 0$ e $z = -r$
- **Ponto 4:** $x = -\sqrt{\frac{r^2}{2}}$ e $z = -\sqrt{\frac{r^2}{2}}$
- **Ponto 5:** $x = -r$ e $z = 0$
- **Ponto 6:** $x = -\sqrt{\frac{r^2}{2}}$ e $z = \sqrt{\frac{r^2}{2}}$
- **Ponto 7:** $x = 0$ e $z = r$
- **Ponto 8:** $x = \sqrt{\frac{r^2}{2}}$ e $z = \sqrt{\frac{r^2}{2}}$

É através destes pontos que depois são definidos todos os outros pontos pertencentes à curva de *Catmull-Rom* como já vimos.

Para chegar aos valores do tempo (em segundos) a ser usados nas translações e rotações, foram consultados os tempos que os planetas demoram a dar uma volta completa ao Sol e adaptados os tempos uns aos outros para ser possível observar movimento, já que alguns planetas demoram muito tempo. Relativamente aos tempos de rotação dos planetas e satélites e também do Sol, foi também consultada informação relativamente ao tempo real e adaptada ao nosso sistema (sempre de modo a que fosse possível perceber-se do movimento dos planetas com tempo de rotação mais lento). Também foi tido em consideração os planetas e satélites que rodam no sentido dos ponteiros do relógio (Vénus, Urano e Tritão) contrariamente aos outros e foi colocado o seu tempo como negativo para rodarem no sentido contrário.

Por fim, relativamente ao cometa foi definida uma trajetória com 4 pontos de modo a aproximar-se das trajetórias reais dos cometas.

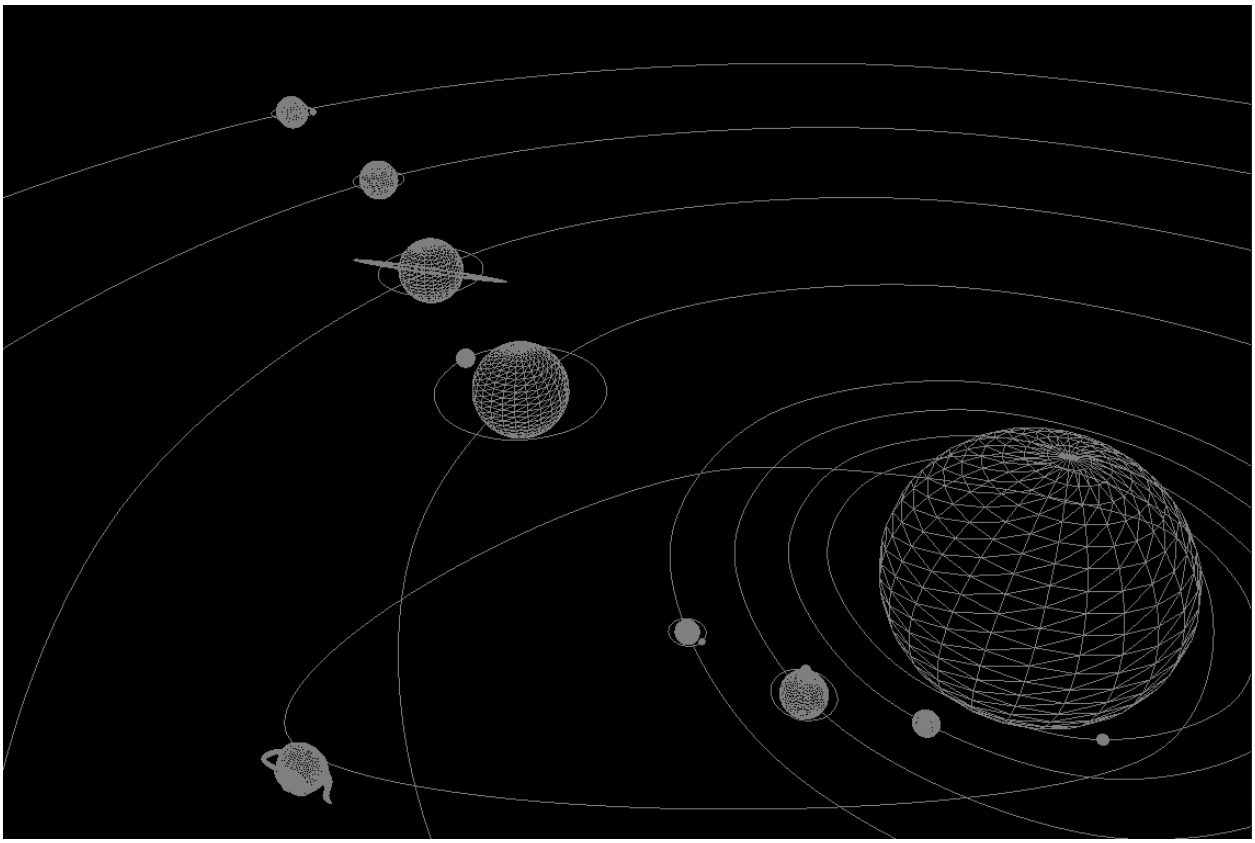

Figura 2: Representação do Sistema Solar dinâmico

5 Conclusão

Achamos que esta fase foi, tanto em termos de requisitos oficiais como em objetivos pessoais, um sucesso.

Esta fase foi importante para a consolidação de temas abordados nas aulas práticas e teóricas da Unidade Curricular. Aprofundamos, em especial, o nosso conhecimento sobre as curvas de *Catmull-Rom*, *Bazier Patches*, *VBOs* e Tecelagem e continuamos também a aprender mais sobre conceitos já aplicados nas fases anteriores como transformações e *XML*.