Universidade do Minho
Escola de Ciências

Computação Gráfica

Ciências da Computação
Fase 3

Bruno Fernandes (A95972) | Tiago Silva (A97450) | Tomás Pereira (A97402)

5 de maio de 2023

Índice

Capítulo 1

Introdução

Este relatório foi elaborado no âmbito da terceira fase do trabalho prático da unidade curricular de Computação Gráfica, no qual foi proposto implementar curvas de Bezier no *generator*, através de um ficheiro com os pontos de controlo, e atualizar a *engine* de forma a implementar translações e rotações com base em curvas de Catmull-Rom.

Capítulo 2

Generator

Ao *generator* foi acrescentada a função *drawbezier* que desenha curvas de Bezier com base nos pontos de controlo especificados num ficheiro *.patch*.

2.1 Primitivas Gráficas

2.1.1 Curvas de Bezier

Do ficheiro *.patch* temos a informação do número de *patches*, cada um com 16 índices dos pontos de controlo. Armazenamos os índices numa estrutura do tipo vector <vector <int>> e os pontos numa estrutura vector <Coordenada>, onde "Coordenada" é uma *struct* com um vector <float> c, servindo os índices 0, 1 e 2 para os valores de x, y e z, respetivamente.

De seguida, são calculados os pontos, segundo o algoritmo de Bezier. Durante a execução, a informação é guardada numa *string* e, no fim, copiada para um ficheiro *.3d*.

```
1   1.4 0 2.4
2   1.3778 -0.254497 2.4
3   1.38491 0 2.43889
4   1.3778 -0.254497 2.4
5   1.36295 -0.251754 2.43889
6   1.38491 0 2.43889
7   1.38491 0 2.43889
8   1.36295 -0.251754 2.43889
9   1.38021 0 2.46806
10  1.36295 -0.251754 2.43889
11  1.35833 -0.2509 2.46806
12  1.38021 0 2.46806
13  1.38021 0 2.46806
14  1.35833 -0.2509 2.46806
15  1.38426 0 2.4875
16  1.35833 -0.2509 2.46806
17  1.36231 -0.251635 2.4875
18  1.38426 0 2.4875
19  1.38426 0 2.4875
20  1.36231 -0.251635 2.4875
```

Figura 2.1: `Excerto do resultado da execução da função`

Capítulo 3

Engine

3.1 Arquitetura do Código

Nesta fase foi nos exigido que os modelos fossem desenhados a partir de *VBOs*, que nos permitem inserir os vértices diretamente na placa gráfica, o que melhora o desempenho na renderização de gráficos. Para isso, foi necessário gerar os *buffers* com as coordenadas dos vértices dos modelos. São gerados tantos *buffers* quanto o número de objetos que são desenhados. Em seguida, é carregada a informação de cada primitiva no respetivo *buffer*.

```
glGenBuffers(objetos.size(), buffers);

for (int i = 0; i < objetos.size(); i++) {
    vertexCount[i] = objetos[i].model.size() / 3;
    glBindBuffer(GL_ARRAY_BUFFER, buffers[i]);
    glBufferData(GL_ARRAY_BUFFER, objetos[i].model.size() * sizeof(float), objetos[i].model.data(), GL_STATIC_DRAW);
}
```

Figura 3.1: `Criação dos` *`VBOs`*

Tendo a informação sido colocada em cada *buffer*, percorremo-los e desenhamos os modelos.

```
glBindBuffer(GL_ARRAY_BUFFER, buffers[i]);
glVertexPointer(3, GL_FLOAT, 0, 0);
glDrawArrays(GL_TRIANGLES, 0, vertexCount[i]);
```

Figura 3.2: `Desenho dos modelos a partir dos` *`VBOs`*

Para além disso, acrescentamos a possibilidade de fazer translações, com base em curvas de Catmull-Rom, e rotações animadas. Para tal, guardamos a informação necessária, proveniente do *XML*, para cada objeto, na seguinte *struct*:

```
struct transformacao {
    string nome;
    float x;
    float y;
    float z;
    float angle = 0;
    float time = 0;
    string align;
    vector <Coordenada> coord;
    float current_time = 0;
    float t = 0;
};
```

Figura 3.3: *Struct* `transformacao`

Visto que o novo tipo de translação necessita de, pelo menos, quatro pontos, criamos a estrutura "Coordenada" para armazenar cada ponto. No vetor "coord" estão várias coordenadas. As variáveis "time" e "align" servem para guardar a informação dos atributos "time" e "align" presentes nas transformações do *XML*. As variáveis "current_time" e "t" surgem pela necessidade de controlar o tempo que as primitivas demoram a completar uma volta.

Com a informação armazenada, resta efetuar os cálculos seguindo o método de Catmull-Rom.

3.2 Resultados Gerados

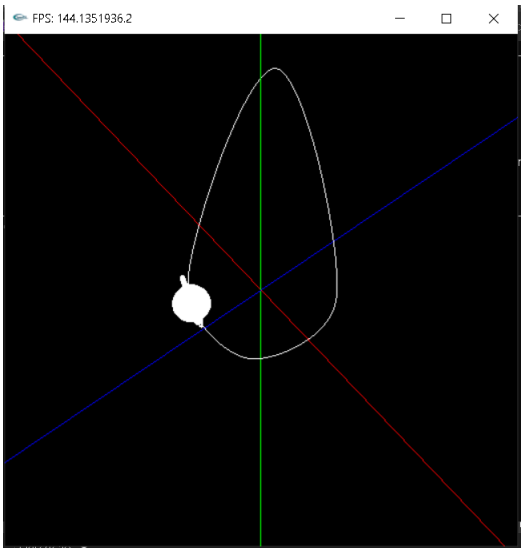

Figura 3.4: `Resultado do` *test_3_1.xml*

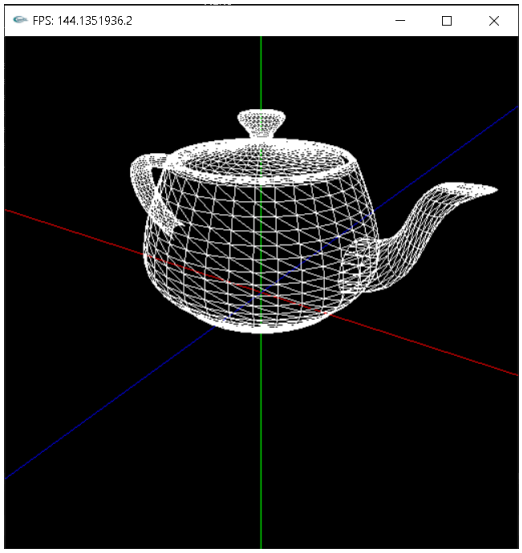

Figura 3.5: `Resultado do` *test_3_2.xml*

3.3 Modelo do Sistema Solar

Ao modelo da fase anterior, foram acrescentadas as órbitas (com animação) dos planetas e luas, através das curvas de Catmull-Rom, e rotação dos planetas em torno de si próprios. Foi também acrescentado um cometa a partir de um *patch* de Bezier, utilizando os pontos de controlo do *teapot*.

```xml
<group>                                                    <!-- TERRA -->
    <transform>
        <translate time = "10" align="true"> <!-- O campo align diz se o objecto deve ser orientado na curva -->
            <point x = "55" y = "0" z = "0" />
            <point x = "38.89" y = "0" z = "-38.89" />
            <point x = "0" y = "0" z = "-55" />
            <point x = "-38.89" y = "0" z = "-38.89" />
            <point x = "-55" y = "0" z = "0" />
            <point x = "-38.89" y = "0" z = "38.89" />
            <point x = "0" y = "0" z = "55" />
            <point x = "38.89" y = "0" z = "38.89" />
        </translate>
        <scale x="2" y="2" z="2" />
    </transform>
    <group>
        <transform>
            <rotate time="1" x="0" y="1" z="0" />
        </transform>
        <models>
            <model file="sphere_1_8_8.3d" />
        </models>
    </group>
    <group>                                                    <!-- TERRA - LUA -->
        <transform>
            <scale x="0.5" y="0.5" z="0.5" />
            <translate time = "10" align="true"> <!-- O campo align diz se o objecto deve ser orientado na curva -->
            <point x = "3.5" y = "0" z = "0" />
            <point x = "2.475" y = "0" z = "-2.475" />
            <point x = "0" y = "0" z = "-3.5" />
            <point x = "-2.475" y = "0" z = "-2.475" />
            <point x = "-3.5" y = "0" z = "0" />
            <point x = "-2.475" y = "0" z = "2.475" />
            <point x = "0" y = "0" z = "3.5" />
            <point x = "2.475" y = "0" z = "2.475" />
        </translate>
            <scale x="0.4" y="0.4" z="0.4" />
        </transform>
        <models>
            <model file="sphere_1_8_8.3d" /> <!-- generator sphere 1 8 8 sphere_1_8_8.3d -->
        </models>
    </group>
</group>
```

Figura 3.6: Excerto do documento *XML* para a geração do sistema solar

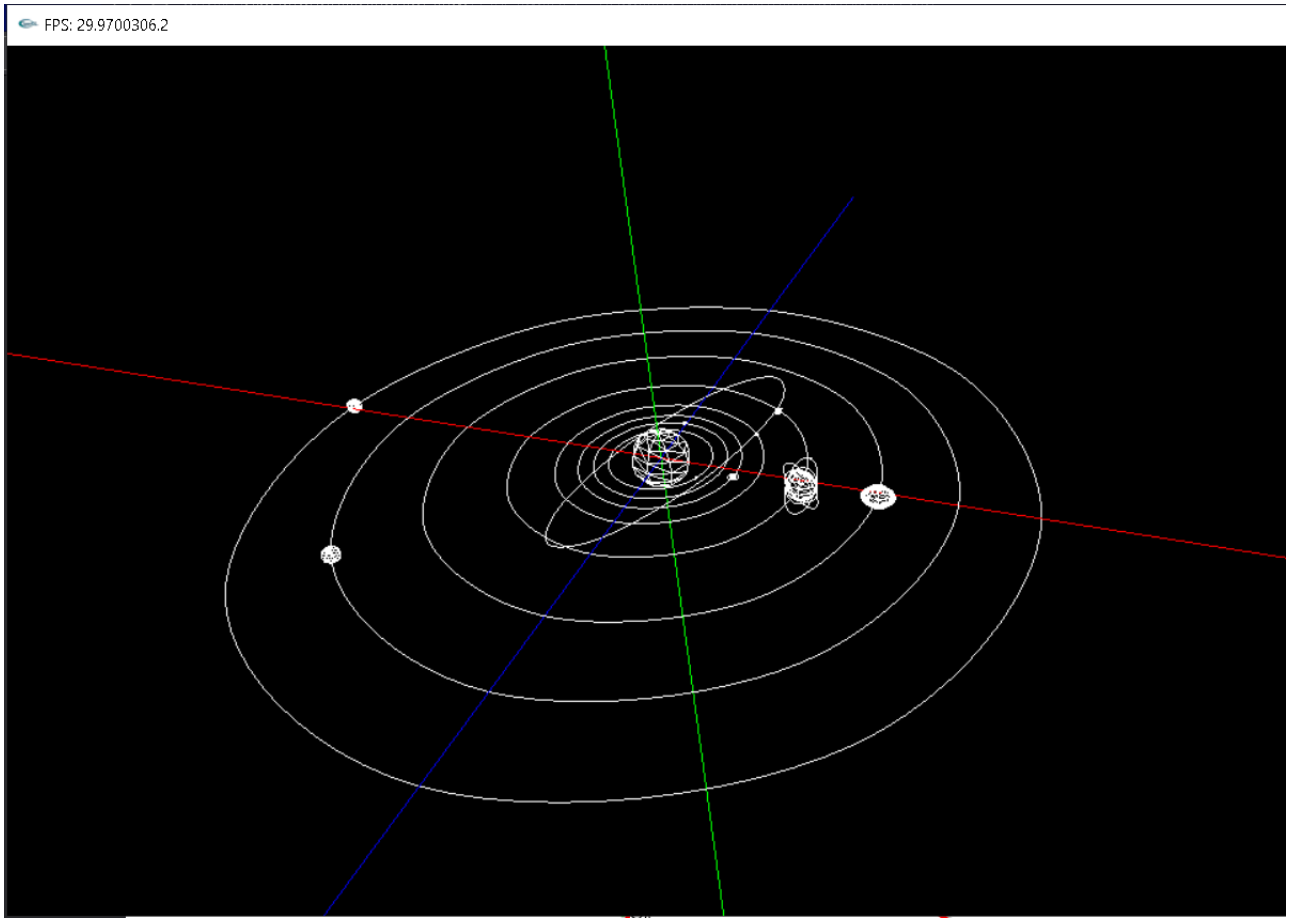

Figura 3.7: `Modelo do sistema solar dinâmico`

Adicionamos, também, a hipótese de visualizar o sistema solar com, ou sem, as curvas das órbitas, pressionando a tecla "O", e de esconder os eixos com a tecla "E".

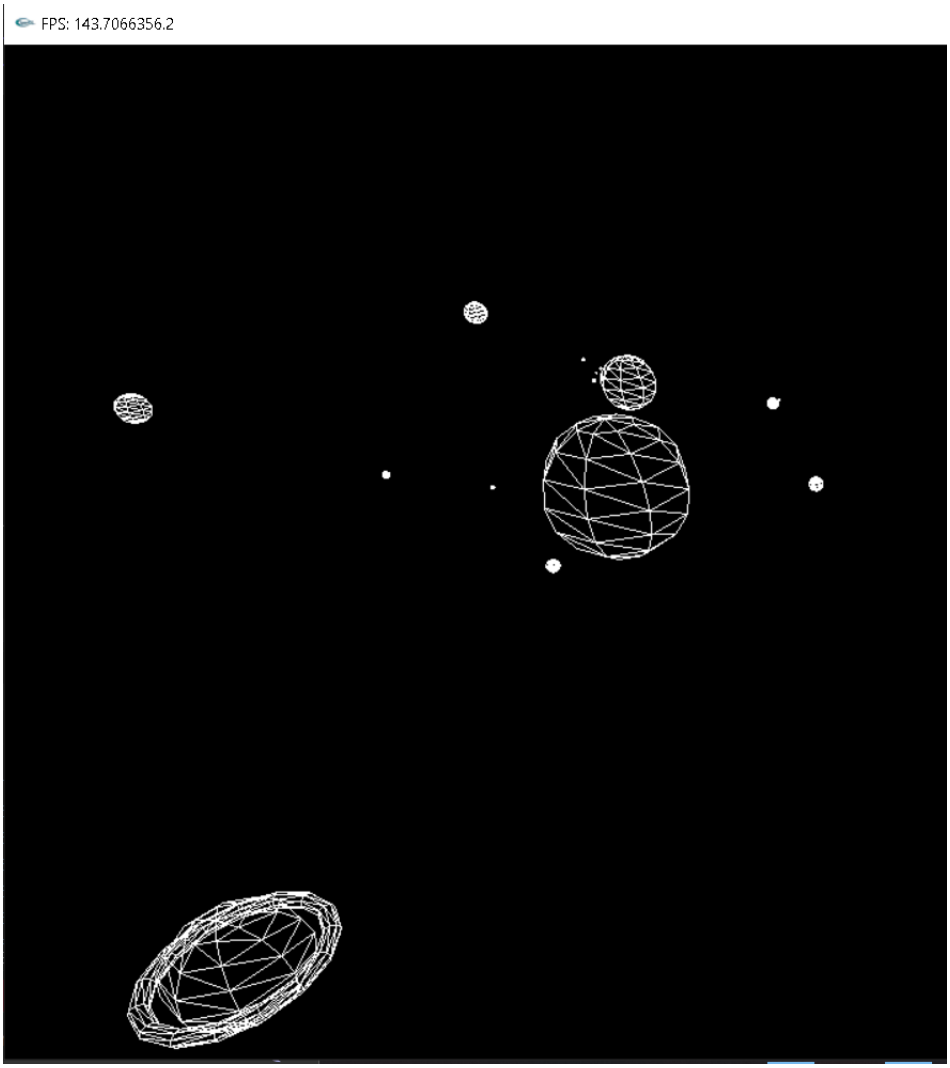

Figura 3.8: `Modelo sem eixos e curvas`

Capítulo 4

Conclusão

Nesta fase, o maior desafio foi a função que gera as curvas de Bezier pois foi necessário armazenar toda a informação do ficheiro *.patch* e manipulá-la de forma a calcular os pontos que constituem cada curva. Além disso, consideramos ter feito uma boa implementação das curvas de Catmull-Rom e dos *VBOs*.

Esta fase foi importante para aprofundar-mos o nosso conhecimento sobre estes dois tipos de curvas, visto que nos permitiu desenvolver algoritmos para as implementar e manipular. Deste modo, achamos que cumprimos os requisitos exigidos nesta fase, visto que, conseguimos criar um modelo dinâmico do sistema solar, como era pretendido.