Universidade do Minho
Escola de Engenharia

Licenciatura em Engenharia Informática 2021/2022

Computação Gráfica

Trabalho prático - Fase 3

- Curves, Cubic Surfaces and VBOs -

1 de maio

Benjamim Miranda Costa (A87985)
Maria Sofia Rocha Gomes (A93314)
Marisa Ferreira Soares (A92926)
Miguel Rodrigues Santa Cruz (A93194)

Índice

1. Introdução

No presente relatório será explicado a estratégia da conceção desta terceira fase, as principais alterações em relação às fases anteriores, uma vez que foi necessário adicionar novas funcionalidades.

O principal objetivo desta terceira fase do trabalho é permitir o desenho de curvas de Catmull-Rom definindo os diversos pontos de controlo, a capacidade de desenhar as cenas 3D utilizando *VBOs* em vez do modo imediato e ainda permitir gerar superfícies de Bézier utilizando para isso o generator desenvolvido nas fases anteriores com esta nova funcionalidade.

2. Mudanças relativamente às fases anteriores

2.1. Engine

Relativamente ao engine, foi adicionada a capacidade de visualizar os frames por segundo, sendo esta informação mostrada no título da janela atual.

Foi também implementada a capacidade de desenhar com VBOs os modelos carregados a partir do ficheiro XML. Através da tecla F5 é possível alternar entre o desenho no modo imediato e o desenho com VBOs.

Figura 1. Título da janela mostrando opções ativas no momento

As teclas definidas para alterar cada uma das definições são as seguintes:

- ➔ Com a tecla *F1* é possível alternar entre visualizar os eixos XYZ.
- ➔ Com a tecla *F2* é possível alternar o *Face Culling* (*Front Face Culling* ou *Back Face Culling*).
- ➔ Com a tecla F3 é possível alternar entre a vista de modo cheio ou wireframe.
- ➔ Com a tecla *F4* podemos alternar entre visualizar a cor do modelo contida no ficheiro XML (quando não estiver definida é utilizado branco por padrão) ou visualizar alternadamente cada face com uma cor diferente, como ilustrado no relatório da Fase II.
- ➔ Com a tecla *F5* podemos alternar entre desenho com VBOs ou modo imediato
- ➔ Com as teclas F6 e F7 é possível aumentar/diminuir o nível de tesselação das curvas de Catmull-Rom desenhadas.

Foi implementada a capacidade de desenhar curvas de Catmull-Rom utilizando para isso pontos de controlo definidos no ficheiro XML. Utilizando estes pontos é construída uma lista que será posteriormente passada à função `renderCatmullRomCurve` que irá efetuar os cálculos

necessários para o seu desenho. Foi utilizada a função `glutGet(GLUT_ELAPSED_TIME)` para obter o tempo decorrido desde a execução do engine.

2.2. Generator

No generator foi adicionado a capacidade de gerar um ficheiro ".3d" tendo por base um ficheiro que contém patches de Bézier com a extensão ".patch".
Para a utilização desta funcionalidade utiliza-se o comando `./generator bezier ".patch file" [tessellation level] ".3d output file"`.
Foi definida a função `createBezierPatch` que recebe como argumentos o ficheiro ".patch", o nível de tesselação e ainda o nome do ficheiro de output.
Esta função é responsável por efetuar o parse do conteúdo do ficheiro ".patch" e calcula as curvas de Bézier resultantes.

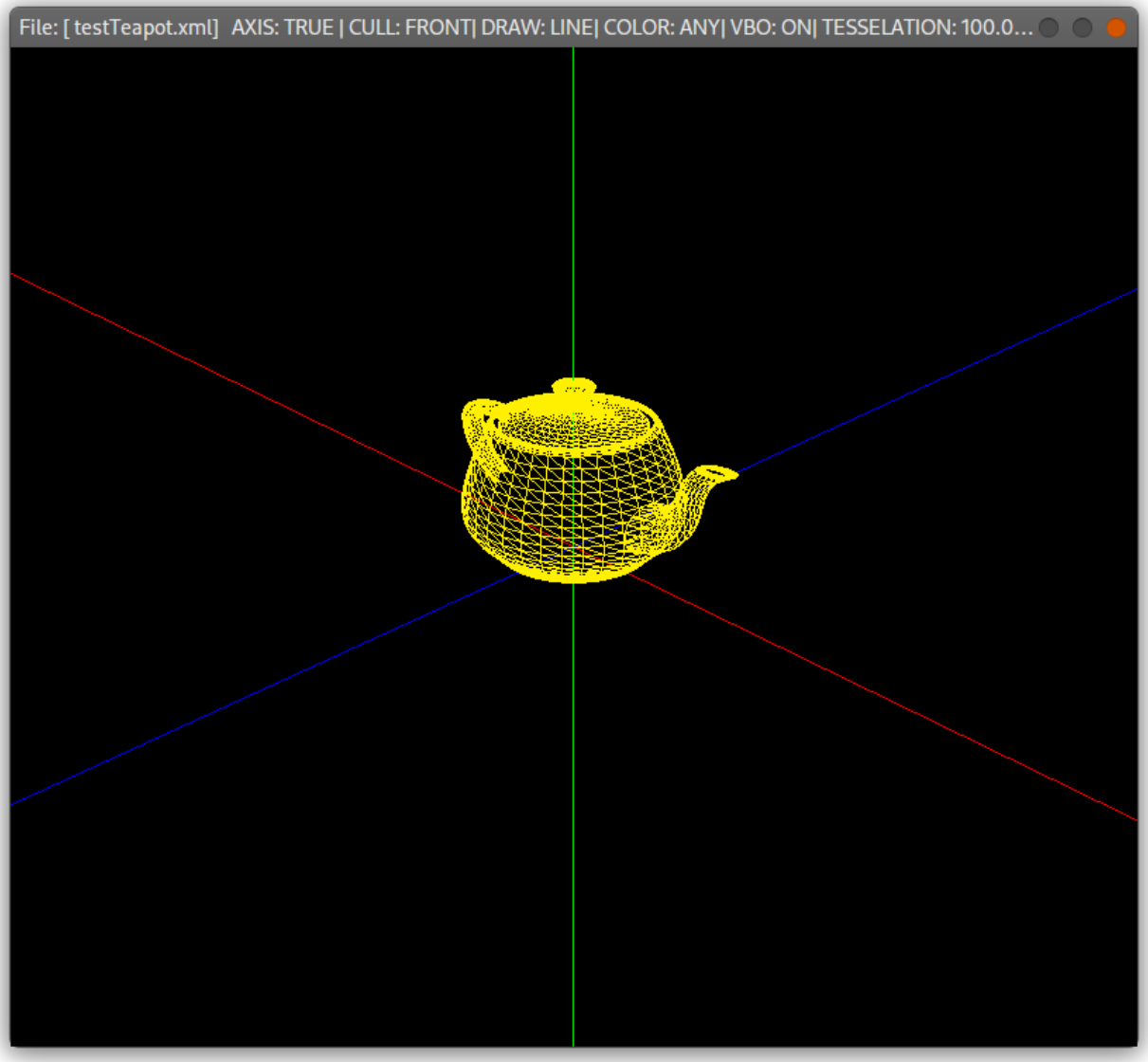

Figura 2. Teapot gerado com o ficheiro teapot.patch

3. Classes Atualizadas

3.1. Classe *Transformation*

Para permitir animar os modelos, tanto nas translações como nas rotações, foi adicionado à classe Transformation o campo *time,* uma lista de pontos de controlo para o desenho das curvas de Catmull-Rom e ainda um boolean correspondente ao atributo *align* que as transformações nesta fase podem possuir.

```
class Transformation{
public:
    /* 0 - Transform
     * 1 - Rotate
     * 2 - Scale
     * 3 - Color
     * */
    int type;
    float color_r = 1;
    float color_g = 1;
    float color_b = 1;
    float x;
    float y;
    float z;
    float angle;
    /* Catmull-Rom curves */
    float time = 0; // Time to move in the curve
    std::list<Point> catmullRomPoints; // List of points for catmull-rom curve
    bool align = true; // Align with the curve
    float prev_y[3] = { [0]: 0, [1]: 1, [2]: 0};
public:
    void applyTransformation();
    std::string toString();
};
```

Figura 3. Classe *Transformation*

3.2. Classe Model

A classe que é responsável por guardar em memória os modelos lidos do ficheiro *XML* é a classe Model. Esta classe contém o nome do ficheiro "*.3d*" correspondente ao modelo atual, uma lista de objetos do tipo *Point* (contém as coordenadas *xyz* de um vértice) e uma lista de objetos da classe *transformation* (descrita acima) que foi adicionada nesta segunda fase.

```
class Model {
public:
    std::string filename = "NaN";
    std::list<Point> points;
    std::list<Transformation> transformations{};
    void drawModel() const;
    void printOut();
};
```

Figura 4. Classe *Model*

Nesta classe foi atualizado o método *drawModel()* para permitir que o desenho do modelo se passe a efetuar usando VBOs.

4. Demo Sistema Solar

Como ficheiro de teste desta terceira fase foi elaborado um ficheiro XML denominado "*SolarSystemPhase3.xml*" que contém a informação para o desenho dos planetas do sistema solar, bem como as transformações a aplicar a cada planeta e a sua respectiva cor.

Foram definidas as órbitas dos planetas e das respectivas luas com curvas de Catmull-Rom e foi utilizado o elemento XML `<rotate time='t' x='0' y='1' z='0'/>` para descrever o movimento de rotação dos planetas.

A título de exemplo apresenta-se de seguida o conteúdo do ficheiro *XML* responsável por desenhar o planeta Terra e a Lua.

```
<!-- Earth -->
<group>
    <transform>
        <color r="9" g="116" b="255" />
        <translate time="10" align="true">
            <point x="100" y="0" z="0" />
            <point x="70.7" y="0" z="70.7" />
            <point x="0" y="0" z="100"/>
            <point x="-70.7" y="0" z="70.7" />
            <point x="-100" y="0" z="0" />
            <point x="-70.7" y="0" z="-70.7" />
            <point x="0" y="0" z="-100" />
            <point x="70.7" y="0" z="-70.7" />
        </translate>
        <scale x="2" y="2" z="2" />
    </transform>
    <models>
        <model file="sphere.3d" />
    </models>
```

Figura 5. XML Terra

```
<!-- Earth's Moon -->
<group>
    <transform>
        <color r="164" g="153" b="124" />
        <translate time="2" align="true">
            <point x="10" y="0" z="0" />
            <point x="7" y="0" z="7" />
            <point x="0" y="0" z="10"/>
            <point x="-7" y="0" z="7" />
            <point x="-10" y="0" z="0" />
            <point x="-7" y="0" z="-7" />
            <point x="0" y="0" z="-10" />
            <point x="7" y="0" z="-7" />
        </translate>
        <scale x="0.5" y="0.5" z="0.5" />
    </transform>
    <models>
        <model file="sphere.3d" />
    </models>
</group>
```

Figura 6. XML Lua

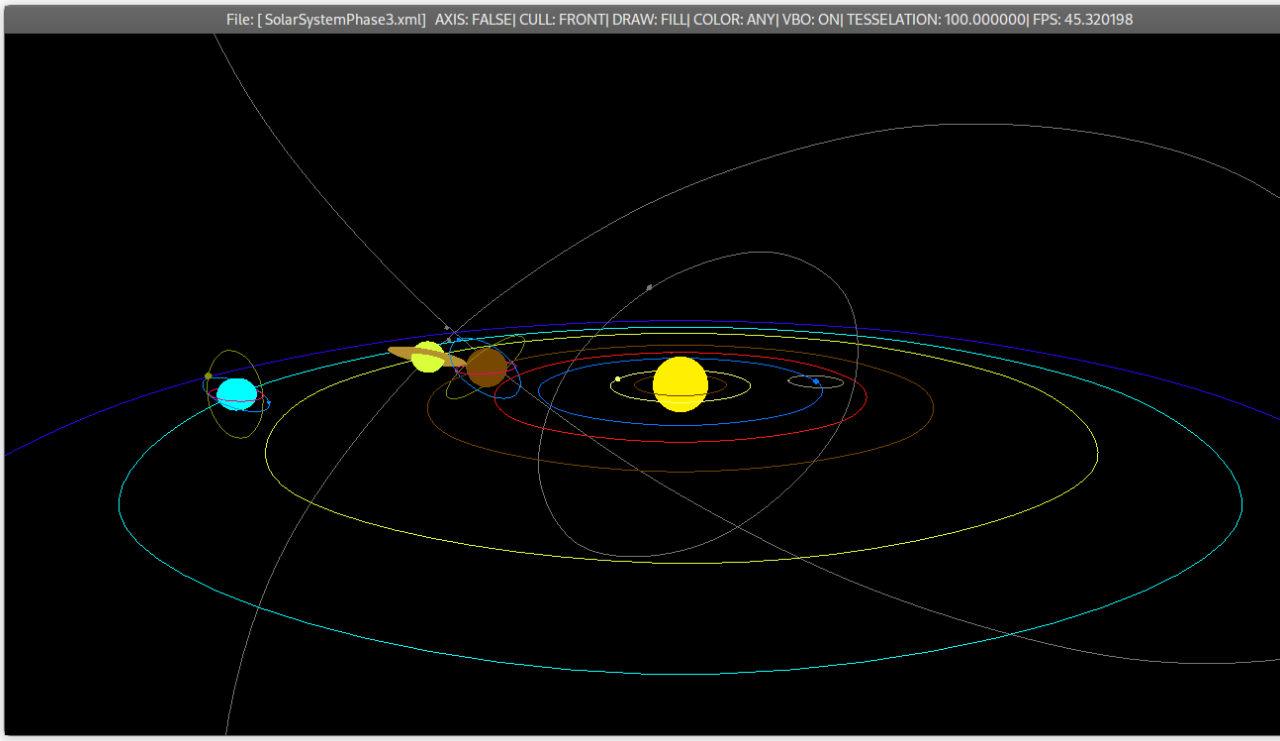

Figura 7. Sistema Solar Fase 3

No *stdout* foi atualizada a informação que é possível visualizar (resultado de invocar os métodos *printOut* das classes *Model* e *CameraConfig*). A figura abaixo representa parte do *output* obtido ao correr o engine com o ficheiro "*SolarSystemPhase3.xml*" como argumento.

```
miguel@fedora build $ ./engine SolarSystemPhase3.xml
XML File: SolarSystemPhase3.xml
==================================================
Camera Configurations:
Position x: 300 y: 100 z :300
LookAt x: 0 y: 0 z :0
Up x: 0 y: 1 z :0
FOV: 60
Near: 1
Far: 1000
==================================================
Model Filename: teapot.3d
Transformations List:
Color x: 120.000000 y: 120.000000 z: 120.000000
Rotate angle: 45.000000 x: 1.000000 y: 0.000000 z: 0.000000
Translate time: 14.000000 align: 1
List of points -------
Point x: 130.000000 y: 0.000000z: 0.000000
Point x: 91.000000 y: 0.000000z: 91.000000
Point x: 0.000000 y: 0.000000z: 130.000000
Point x: -91.000000 y: 0.000000z: 91.000000
Point x: -130.000000 y: 0.000000z: 0.000000
Point x: -91.000000 y: 0.000000z: -91.000000
Point x: 0.000000 y: 0.000000z: -130.000000
Point x: 91.000000 y: 0.000000z: -91.000000
----------------------
Scale x: 1.200000 y: 1.200000 z: 1.200000
Rotate time: 1.000000 x: 0.000000 y: 1.000000 z: 0.000000
Rotate angle: -90.000000 x: 1.000000 y: 0.000000 z: 0.000000
==================================================
Model Filename: box.3d
Transformations List:
Color x: 120.000000 y: 120.000000 z: 120.000000
Rotate angle: -145.000000 x: 1.000000 y: 0.000000 z: 0.000000
Translate time: 17.000000 align: 1
List of points -------
Point x: 360.000000 y: 0.000000z: 0.000000
```

Figura 8. *Output* de `./engine SolarSystemPhase3.xml`

5. Conclusão

Depois de realizada esta terceira fase do trabalho prático, podemos afirmar que concluímos todos os objetivos que foram propostos e ainda implementamos alguns dos objetivos extra, como a capacidade de visualizar os *frames por segundo,* a capacidade de alterar o nível de tesselação para o desenho das curvas de Catmull-Rom e a capacidade de poder alternar entre o modo imediato e o desenho com VBOs*,* para além das funcionalidades extra que tinham sido implementadas nas fases anteriores (como informação sobre a cor dos objetos e a capacidade de ver as definições que estão ativas no título da janela).

Foi possível observar o ganho óbvio de performance com a utilização de VBOs, particularmente quando aumentou a complexidade do ficheiro de *demo* do sistema solar.

Concluímos que o nosso trabalho representa satisfatoriamente o resultado das aprendizagens que obtivemos nesta unidade curricular até ao presente momento.