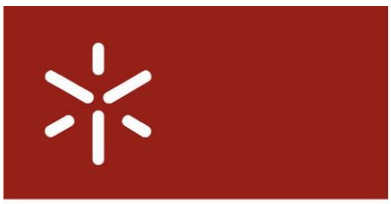


Universidade do Minho

Computação Gráfrica

MIEI - 3º ano - 2º semestre
Universidade do Minho

Trabalho Prático

João Leal
A75569

Rui Vieira
A74658

Carlos Faria
A67638

Ricardo Leal
A75411

29 de Abril de 2018

Conteúdo

1. *Introdução*

1.1 Contextualização

No âmbito da Unidade Curricular de Computação Gráfica, foi nos posposto a realização de um mecanismo de 3D onde foram sendo aplicados múltiplas ferramentas usadas ao longo do semestre.

Este projeto foi seccionado, em 4 partes, sendo esta a terceira fase que tem como objetivo a inclusão de curvas de superfícies cúbicas ao trabalho previamente realizado, com a finalidade de criar um modelo dinâmico do Sistema Solar.

1.2 Resumo

Tratando-se da terceira parte de um projeto prático, é natural que mantenham funcionalidades desenvolvidas na primeira e segunda fase, e que outras sejam alteradas de forma a cumprir os requisitos necessários.

Esta fase traz consigo várias novidades que irão levar a várias alterações, tanto ao nível do engine como do generator.

Começando pelo generator, nesta fase, este passará a conseguir ser capaz de criar um novo tipo de modelo baseado em Bezier patches. Este passará a poder receber como parâmetros o nome de um ficheiro, no qual se encontram definidos os pontos de controlo dos vários patches, assim como, um nível de tecelagem e, a partir destes, retornará um ficheiro contendo uma lista de triângulos que definem essa mesma superfície.

Passando para o engine, este sofrerá várias modificações e, ao mesmo tempo, receberá também novas funcionalidades. Os elementos translate e rotate, presentes nos ficheiros XML, sofrer˜ao algumas modificações. O elemento translation será agora acompanhado por um conjunto de pontos, que no seu todo, definirão uma catmull-rom curve, assim como o número de segundos para percorrer toda essa curva. Isto surge com o objetivo de passar a ser possível criar animações baseadas nessas mesmas curvas. Passando para o elemento rotation, o ângulo anteriormente definido poderá agora ser substituído pelo tempo, correspondente este ao número de segundos que o objeto demora para completar uma rotação de 360 graus em torno do eixo definido. Deste modo, ter´a que ser alterado não só o parser responsável por ler esses mesmos ficheiros, assim como, o modo como é processada toda a informação recebida com o intuito de gerar todo o cenário pretendido. A última modificação, e não menos importante, que o engine sofrerá, está relacionada com os modelos gráficos, pois, estes agora passarão a ser desenhados com o auxílio de VBOs, ao contrário da fase anterior, no qual estes eram desenhados de forma imediata.

Todas as alterações tem como finalidade gerar de forma eficaz um modelo do Sistema Solar ainda mais realista, passando de um modelo estático a um dinâmico.

2. *Arquitetura de Código*

Tratando-se de uma continuação do trabalho desenvolvido nas fases anteriores, mantendo inalterado grande parte do código, contudo de forma a cumprir os requisitos necessários propostos para esta fase parte do código foi alterado e criado outro.

2.1 Aplicações

Nesta secção são apresentadas as aplicações fundamentais que permitem gerar e exibir os diferentes cenários pretendidos. Na realização desta fase houve alterações significativas em ambas as aplicações do projeto, de modo a que os requisitos propostos pela mesma fossem devidamente cumpridos.

2.1.1 Generator

generator.cpp -Tal como explicado em fases anteriores, trata-se da aplicação onde estão definidas as estruturações das diferentes primitivas geométricas a desenvolver de forma a gerar os respetivos vértices. A aplicação cria uma pasta "files"e guarda os ficheiros gerados dentro da mesma. Para alem das primitivas gráficas previamente desenvolvidas, na realização desta fase foi introduzido um novo método de construção de modelos com base em curvas de Bézier, implicando novas funcionalidades ao gerador.

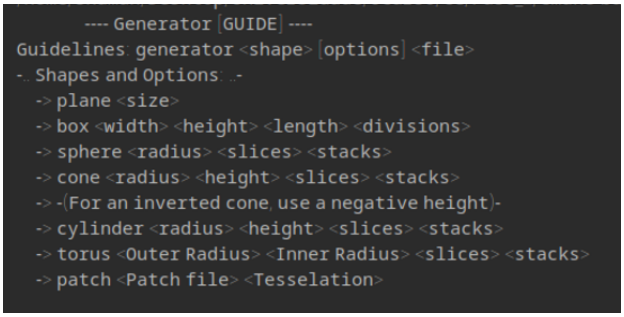


Figura 2.1: Apresentação do ficheiro generator.h

2.1.2 Motor

engine.cpp - Aplicação que permite a apresentação de uma janela exibindo os modelos pretendidos, lidos a partir de um ficheiro XML (criado pelo utilizador), e ainda a interação com estes mesmos modelos, através de comandos. Durante a realização desta fase do projeto inevitavelmente surgiram alterações relativamente à fase anterior.

Devido a mudanças relativas à estruturação do ficheiro de configuração obrigaram a pequenas mudanças ao nível do parser e as consequentes estruturas de dados. A implementação de Catmull-Rom Cubic Curves e de Vertex Buffer Objects, por sua vez impulsionou também a mudanças nesta aplicação.

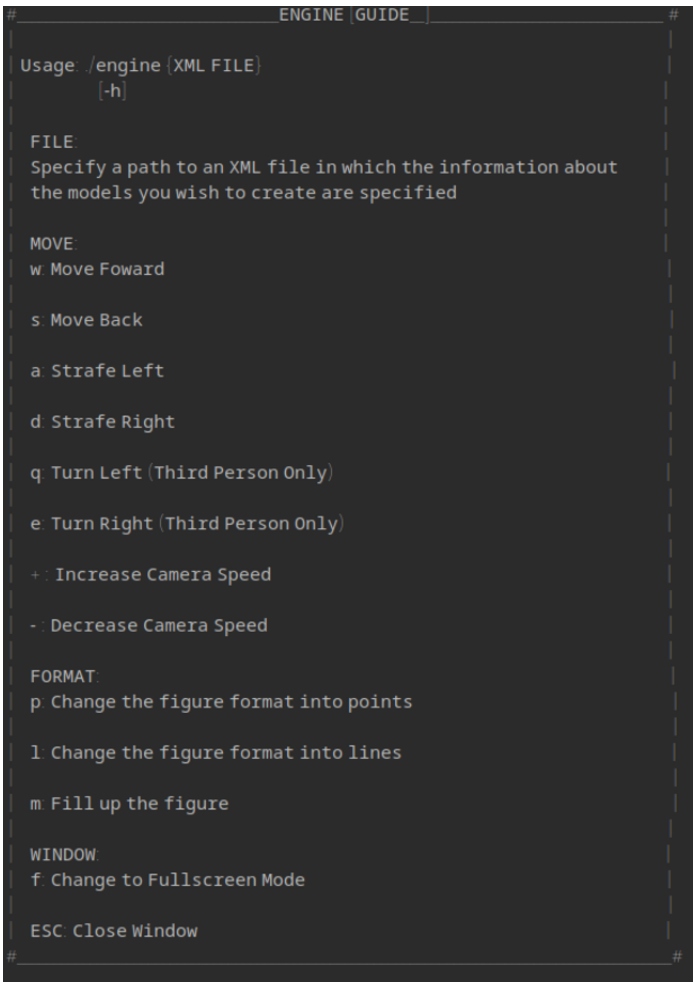


Figura 2.2: Apresentação do ficheiro engine.h

2.2 Classes

Ao contrário da fase anterior foi necessária onde foi necessária a criação de varias classes de maneira a facilitar o processamento e organização de informação, nesta fase houve apenas a criação de uma única classe e alteração das já existentes.

2.2.1 Vertex

vertex.cpp - Define a estrutura de um vértice, necessário para o desenho de um triângulo. Assim sendo, estão aqui definidas 3 coordenadas: X, Y e Z.

```
#ifndef _VERTEX_H_
#define _VERTEX_H_

#include <vector>
#include <string>

class Vertex {

    private:
        float x;
        float y;
        float z;

    public:
        Vertex();
        Vertex(float xx, float yy, float zz);
        Vertex(const Vertex &p);
        Vertex(std::string str);
        float getX();
        float getY();
        float getZ();
        virtual ~Vertex(void);
};

#endif
```

Figura 2.3: Apresentação do ficheiro vertex.h

2.2.2 Transform

Transforme.h - Classe responsável pelo armazenamento da informação necessária à execução de uma translação, obrigando à existência de variáveis de instância x, y e z, representando o vetor aplicado na translação. A implementação das curvas *Catmull-Rom* implicou ainda à presença de um conjunto de pontos que a definam com correspondente tempo em segundos que demorará a percorrer a curva. O conteúdo de cada um destes conjuntos de pontos será explicado detalhadamente mais adiante. Esta classe armazena ainda de toda informação pertinentes à execução de uma rotação, necessitando a existência de variáveis de um plano cartesiano (x, y e z), representando o eixo de aplicação e ainda uma variável correspondendo ao ângulo de rotação do vetor. Já a implementação de movimento sobre os modelo exigiu uma variável correspondente ao número de segundos a percorrer uma rotação de 360ª sobre o eixo especificado.

```
#ifndef _TRANSFORMS_H
#define _TRANSFORMS_H

#include<vector>
#include <cmath>
#include "../src/vertex.h"

class Transform{

    float x_var;
    float y_var;
    float z_var;

    public:
        Transform();
        Transform(float x, float y, float z);
        float getX();
        float getY();
        float getZ();
        virtual void perform() { };
};

class Translation : public Transform{
    float time;
    std::vector<Vertex*> curve_points;
    std::vector<Vertex*> catmull_points;

public:
    Translation();
    Translation(float x, float y, float z,float t);
    void addPoint(Vertex* v);
    void generateCurve();
    void drawCurve();
    int getCurveSize();
    void perform();
    virtual ~Translation(void);
};

class Rotation : public Transform{
    float angle;
    float time;

public:
    Rotation(float t, float a, float x, float y, float z);
    void perform();
    virtual ~Rotation(void);
};
```

```
class Translation : public Transform{
    float time;
    std::vector<Vertex*> curve_points;
    std::vector<Vertex*> catmull_points;

public:
    Translation();
    Translation(float x, float y, float z,float t);
    void addPoint(Vertex* v);
    void generateCurve();
    void drawCurve();
    int getCurveSize();
    void perform();
    virtual ~Translation(void);
};

class Rotation : public Transform{
    float angle;
    float time;

public:
    Rotation(float t, float a, float x, float y, float z);
    void perform();
    virtual ~Rotation(void);
};

class Scale : public Transform{

public:
    Scale(float x, float y, float z);
    void perform();
    virtual ~Scale(void);
};

class Colour : public Transform{

public:
    Colour(float r, float g, float b);
    void perform();
    virtual  ~Colour(void);
};

#endif
```

Figura 2.4: Apresentação do ficheiro Transform.h

2.2.3 Group

group.cpp-Armazena toda a informação que diz respeito ao grupo hierárquico atual. Acrescentamos uma classe *Model*, responsável pela inicialização/geração de VBO's e pelo desenho dos mesmos.

```
#ifndef _GROUP_H_
#define _GROUP_H_

#include <vector>
#include "transforms.h"
#include "../src/shape.h"

class Group {

        std::vector<Transform*> transforms;
        std::vector<Shape*> models;
        std::vector<Group*> children;

    public:
        Group();
        Group(std::vector<Transform*> trfms, std::vector<Shape*> m, std::vector<Group*> c);
        std::vector<Transform*> getTransforms();
        std::vector<Shape*> getModels();
        std::vector<Group*> getChilds();
        void pushTransform(Transform* t);
        void setModels(std::vector<Shape*> m_list);
        void pushChild(Group* c);
        ~Group(void);
};

#endif
```

Figura 2.5: Header da classe group

2.2.4 Shape

shape.cpp - Classe responsável pelo armazenamento de todo o conjunto de pontos necessários à representação de um determinado modelo, contendo um cojunto de vértices e o seu nome. Necessitando um buffer para implementação de VBOs.

```
#ifndef _SHAPE_H_
#define _SHAPE_H_

#include <vector>
#include "vertex.h"

class Shape {

private:
    std::vector<Vertex*> vertexes;

public:
    void pushVertex(Vertex* v);
    void pushShape(Shape* s);
    void getVertexAt(int i, Vertex** v);
    void reverse();
    int getSize();
    virtual ~Shape(void);
};

#endif
```

Figura 2.6: Apresentação do ficheiro Shape.h

2.2.5 Bezier

bezier.cpp- Classe indispensável ao processamento das curvas Bezier responsável ao armazenamento do conjuntos de pontos de controlo.

```
#ifndef _BEZIER_H_
#define _BEZIER_H_

#include "../src/shape.h"
#include <string>
#include <fstream>
#include <iostream>

Shape* bezier_Parse(std::string patch_path, int tesselation);

#endif
```

Figura 2.7: Apresentação do ficheiro Bezier.h

2.2.6 Models

models.cpp -Classe responsável pelo armazenamento de todos os algoritmos necessários à criação de cada modelo a que queremos gerar, com as suas respetivas caraterísticas.

```
#ifndef _MODELS_H
#define _MODELS_H

#include <vector>
#include <math.h>
#include "../src/shape.h"

Shape* createPlane(float size);
Shape* createBox(float width, float height, float length, int div);
Shape* createCone(float radius, float height, int slices, int stacks);
Shape* createCylinder(float radius, float height, int slices, int stacks);
Shape* createSphere(float radius, int slices, int stacks);

#endif
```

Figura 2.8: Apresentação do ficheiro Models.h

3. *Generator*

3.1 Descrição

O Gerador, tal como na fase anterior, é responsável por gerar ficheiros que contêm o conjunto de vértices das primitivas gráficas que se pretende gerar, conforme os parâmetros escolhidos. A única mudança que ocorreu nesta transição de fases foi a inclusão de um processo de modo a poder gerar modelos baseados em *Bezier Patches*.

3.2 Bezier Patches

3.2.1 Leitura de Ficheiro

O ficheiro que contém as informações para gerar os vértices do modelo Bezier em questão, está dividido em 4 partes fundamentais:

Número de Patches (n_patches): Apresentado na primeira linha do ficheiro;

Patches: Apresentadas nas *n_patches* linhas seguintes, cada uma contendo 16 pontos de controlo, constituintes dessa mesma patch;

Número de Pontos de Controlo (n_control): Apresentado na linha seguinte às *Patches*;

Pontos de Controlo: Apresentados nas *n_control* linhas seguintes;

Tendo, então, que a informação é representada desta forma, começamos primeiro por retirar a informação relativa ao número de patches existente *(n_patches)*. De seguida, para cada uma delas, guardamos os indíces dos pontos de controlo de cada uma, num array. Por fim, guardamos as coordenadas dos pontos de controlo num *vector<Vertex*> controls*, para posteriormente serem acedidos (através dos indices guardados anteriormente) para calcular os vértices do modelo a representar.

3.2.2 Curvas de Bezier

Para definir uma curva de Bezier, são precisos 4 pontos. Tais pontos, são denominados de *Pontos de Controlo*, e estão representados em XYZ.

Como se trata de uma curva, é de esperar que esta possa ser definida por uma equação, e como de uma equação se trata, é normal que esta contenha uma variável. Tal varíavel é chamada de *tesselagem* e está contida num intervalo [0,1]. Assim, para cada valor de *tesselagem* (t), dentro desse intervalo, obtemos uma posição em XYZ desta curva, portanto, para podermos ser capazes de a representar, apenas necessitamos de calcular o resultado da equação da curva para estes valores de t, que será incrementado em intervalos fixos. Desta forma, conseguimos obter os vários pontos da curva.

Segue-se em seguida a equação mencionada acima:

P(t) = P1*bz1 + P2*bz2 + P3*bz3 + P4*bz4

P(i)$_{i>0\&i<5}$) , representam os Pontos de Controlo da curva e **bz(i)**$_{i>0\&i<5}$) os Coeficientes que auxiliam no cálculo dos Pontos da Curva. Estes coeficientes podem ser calculados da seguinte forma:

bz1 = (1-t) * (1-t) * (1-t);

bz2 = 3 * t * (1-t) * (1-t);

bz3 = 3 * t * t * (1-t);

bz4 = t * t * t;

Podemos assim concluir que, para calcular a posição da curva em determinado valor de t, estes valores bz(i) terão de ser novamente calculados e multiplicados pelos Pontos de Controlo.

Sabendo agora definir curvas de Bezier, estamos aptos para passar para as Patches de Bezier, que segue um princípio bastante parecido com o das Curvas, só que neste caso, em vez de 4 Pontos de Controlo, temos 16 que podem ser representados numa grelha de 4x4. Desta forma, podemos considerar cada linha da grelha, como uma curva bezier, percorrendo horizontalmente a linha de modo a calcular os pontos correspondentes às curvas, posteriormente passando para a linha seguinte.

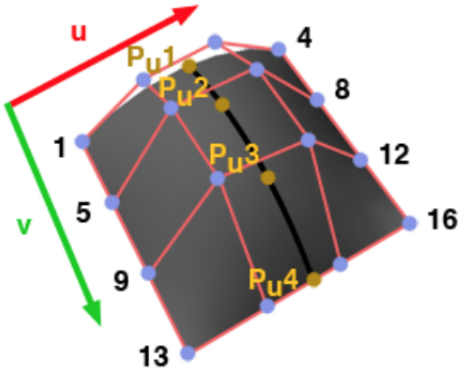


Figura 3.1

Após os 4 pontos serem calculados, resta calcular o Ponto final desta curva que corresponde a um par (u,v) e à posição da superfície de Bezier para um desses pares.

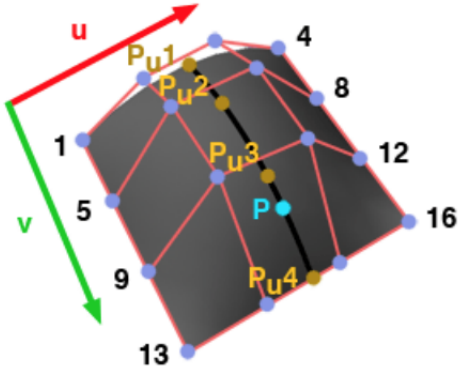


Figura 3.2

Dependendo do nível de tesselagem dado, serão calculados os vários pares (u,v) que posteriormente serão escritos para um ficheiro , de modo a gerar triângulos que representem corretamente o modelo a desenhar. Assim, facilmente concluímos que quanto maior a tesselagem, mais uniforme e "bonito"será o modelo gerado.

4. *Engine*

4.1 Descrição

O *engine* é responsável pelo armazenamento da informação dos modelos a representar, assim como as respetivas transformações geométricas de cada um desses modelos, que no seu conjunto, irão formar o sistema solar que queremos representar. Findando o processo de leitura do ficheiro, passamos para a fase seguinte, a qual é responsável pela renderização da informação, previamente obtida.

4.2 VBOs

Esta nova fase do projeto conta com a implementação de VBOs (Vertex Buffer Object) que substitui a forma mais primitiva de desenhar os modelos, renderização imediata. Esta funcionalidade do OpenGL oferece métodos capazes de inserir a informação sobre os vértices diretamente na placa de vídeo. A performance obtido pelos VBOs é feito á custa de vertex buffers, arrays contendo todos os vértices do modelo a desenhar que são passados á placa de vídeo, suprimindo a necessidade de os armazenar na memória do sistema, sendo estes diretamente renderizados pela placa o que diminui a sobrecarga do sistema e permite obter ganhos muito significados que se traduzem no aumento dos FPS (frames per second). No nosso projecto, estes são gerados na função *Model::setUp()* e desenhados na função *Model::renderModel()* que utiliza a função *glDrawArrays* para esse fim.

4.3 Catmull-Rom curve

4.3.1 Rotate

Surge uma alteração à transformação geométrica *Rotate*, que traduz-se na adição de uma variável *time*, que indica o tempo em segundos, necessário para uma rotação de 360º.

Para traduzir esse movimento recorremos aos seguintes cálculos:

r = *(TEMPO DECORRIDO DESDE A EXECUÇÃO) % (time % 1000)* rot = *(r*360)/(time*1000)*

Sendo *rot* o ângulo de rotação para usar na função *glRotatef*, responsável pela aplicação desta transformação.

4.3.2 Translate

Assim como a *Rotate*, esta transformação geométrica, *Translate* também sofre alterações com a adição da variável *time*, que desta vez corresponde ao tempo necessário ao objeto para completar uma volta, na sua trajetória.

Para traduzir esta trajetória, recorremos ao cálculo de curvas Catmull-Rom, que resultará num conjunto de pontos que posteriormente serão percorridos e usados na aplicação da função *glTranslatef*, responsável pela representação da movimentação do objeto ao longo da curva Catmull-Rom.

5. *Analise de Resultados*

O resultado final correspondeu ao esperado pelo grupo, todos os planetas foram representados à escala de modo a criar uma perceção semelhante à realidade.

5.1 Visualização

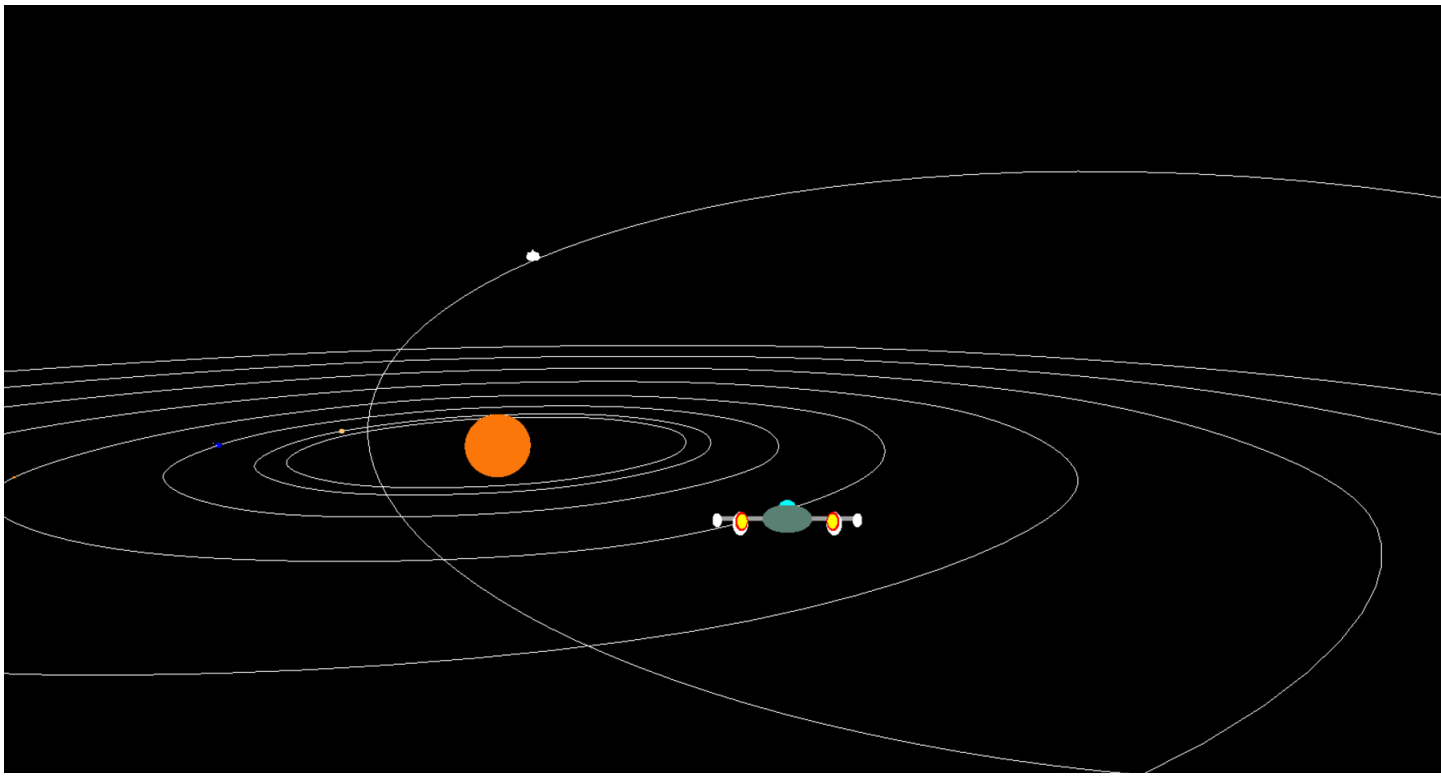


Figura 5.1: Representação do Sistema solar

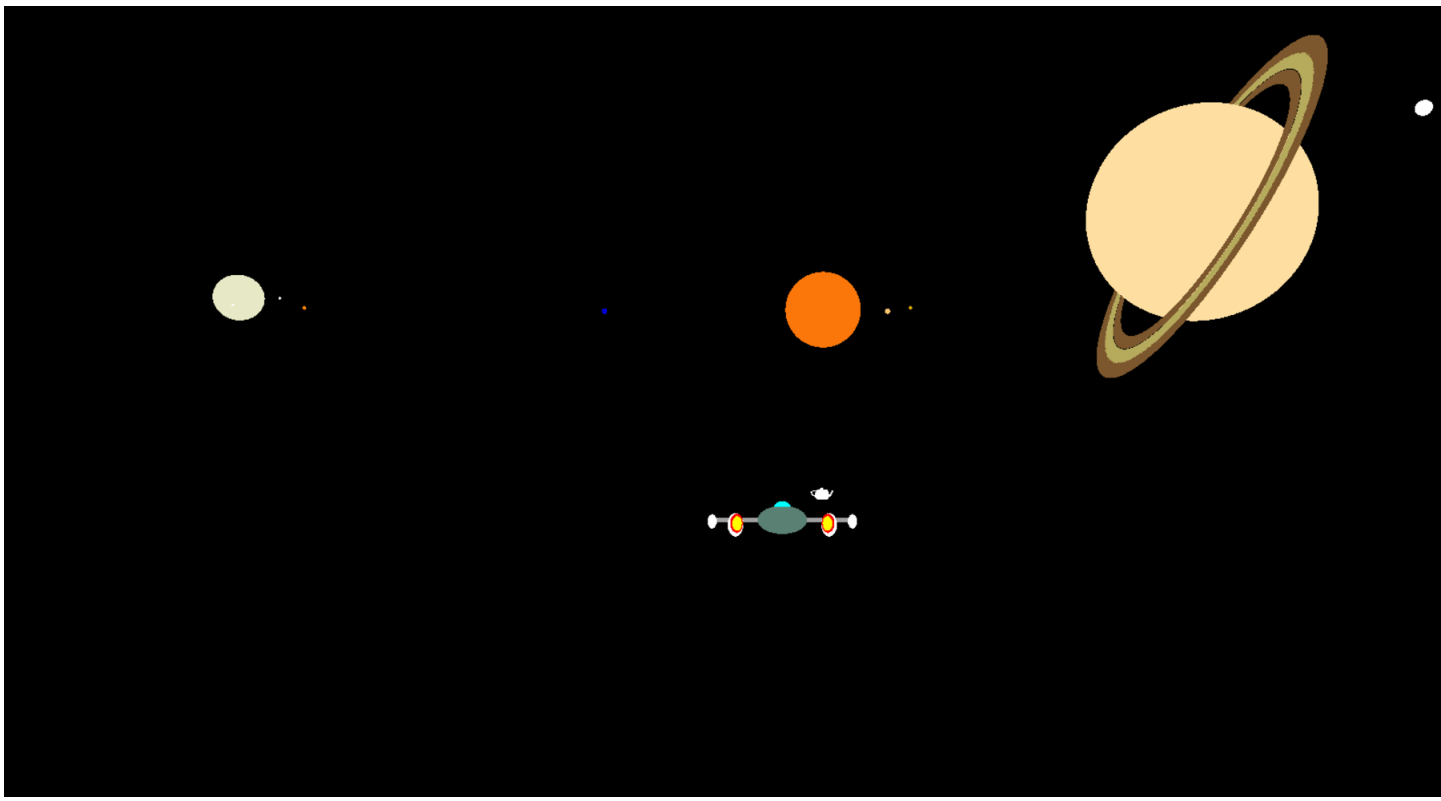
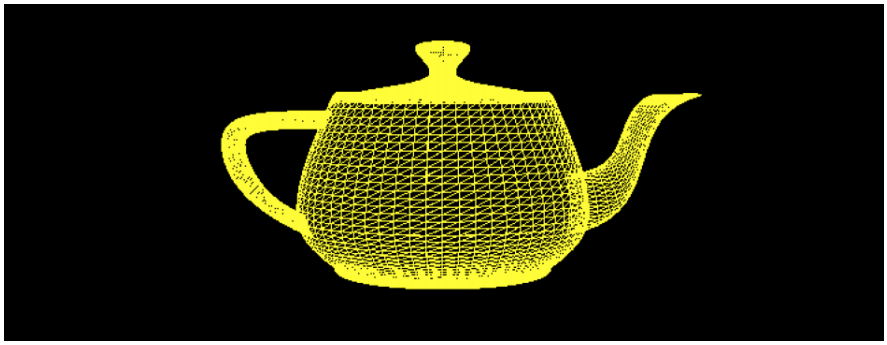


Figura 5.2: Representação do Sistema solar



6. *Conclusão*

Nesta fase, ao contrário da anterior foi bastante demorada e desafiante, muito pelo maior que requisitos da mesma em relação à anterior.

Ao longo da elaboração desta fase, o maior desafio foi encontrar a melhor solução possível para a implementação dos requisitos obrigando a bastante pesquisa.

Em suma sentimos bastante orgulho no trabalho realizado, sentindo que todas as horas de trabalho no projeto foram bem gastas, o que é dizer muito num semestre tão exigente em volume de trabalhos a realizar.

7. *Anexos*

7.1 Ficheiro XML - Sistema Solar

```
<scene>
    <group>
        <!-- SUN -->
        <group>
            <scale X='5' Y='5' Z='5' />
            <colour R='251' G='119' B='9' />
            <rotate time='50' X='0' Y='1' Z='0"'/>
            <models>
                <model file='../files3d/sphere.3d'/>
            </models>
        </group>

        <!-- PLANETS AND SATELITES -->

        <!-- MERCURY -->
        <group>
            <translate time="60">
                <point X="29.8775" Y="0" Z="0"/>
                <point X="21.1266" Y="0" Z="21.1266"/>
                <point X="0" Y="0" Z="29.8775"/>
                <point X="-21.1266" Y="0" Z="21.1266"/>
                <point X="-29.8775" Y="0" Z="0"/>
                <point X="-21.1266" Y="0" Z="-21.1266"/>
                <point X="0" Y="0" Z="-29.8775"/>
                <point X="21.1266" Y="0" Z="-21.1266"/>
            </translate>
            <scale X='0.17625' Y='0.17625' Z='0.17625'/>
            <colour R='219' G='170' B='0' />
            <models>
                <model file='../files3d/sphere.3d' />
            </models>
        </group>
        <!-- VENUS -->
        <group>
            <translate time="90">
                <point X="33.9233" Y="0" Z="0"/>
                <point X="23.98740" Y="0" Z="23.98740"/>
                <point X="0" Y="0" Z="33.9233"/>
```

```
        <point X="-23.98740" Y="0" Z="23.98740"/>
        <point X="-33.9233" Y="0" Z="0"/>
        <point X="-23.98740" Y="0" Z="-23.98740"/>
        <point X="0" Y="0" Z="-33.9233"/>
        <point X="23.98740" Y="0" Z="-23.98740"/>
    </translate>
    <scale X='0.4395' Y='0.4395' Z='0.4395'/>
    <colour R='254' G='198' B='115' />
    <models>
        <model file='../files3d/sphere.3d' />
    </models>
</group>
<!-- EARTH -->
<group>
    <translate time="120">
        <point X="44.8061" Y="0" Z="0"/>
        <point X="31.68270" Y="0" Z="31.68270"/>
        <point X="0" Y="0" Z="44.8061"/>
        <point X="-31.68270" Y="0" Z="31.68270"/>
        <point X="-44.8061" Y="0" Z="0"/>
        <point X="-31.68270" Y="0" Z="-31.68270"/>
        <point X="0" Y="0" Z="-44.8061"/>
        <point X="31.68270" Y="0" Z="-31.68270"/>
    </translate>
    <rotate angle="23" X="0" Y="1" Z="1"/>
    <colour R='0' G='0' B='220' />
    <scale X='0.4425' Y='0.4425' Z='0.4425'/>
    <group>
        <rotate time="22" X="0" Y="1" Z="0"/>
        <models>
            <model file='../files3d/sphere.3d' />
        </models>
    <!-- MOON -->
    <group>
           <rotate angle="-23" X="0" Y="1" Z="1"/>
        <translate Y='0.75' Z='1.75' />
        <colour R='255' G='255' B='255' />
        <scale X='0.15' Y='0.15' Z='0.15'/>
        <group>
            <rotate time="616" X="0" Y="1" Z="0"/>
             <models>
                <model file='../files3d/sphere.3d' />
             </models>
        </group>
    </group>
    </group>
</group>

<!-- MARS -->
<group>
```

```
    <translate time="170">
        <point X="60.9523" Y="0" Z="0"/>
        <point X="44.51400" Y="0" Z="44.51400"/>
        <point X="0" Y="0" Z="60.9523"/>
        <point X="-44.51400" Y="0" Z="44.51400"/>
        <point X="-60.9523" Y="0" Z="0"/>
        <point X="-44.51400" Y="0" Z="-44.51400"/>
        <point X="0" Y="0" Z="-60.9523"/>
        <point X="44.51400" Y="0" Z="-44.51400"/>
    </translate>
    <scale X='0.1860' Y='0.1860' Z='0.1860'/>
    <colour R='254' G='135' B='1' />
        <rotate time="28" X="0" Y="1" Z="0"/>
        <models>
        <model file='../files3d/sphere.3d' />
    </models>
</group>
<!-- JUPITER -->
<group>
    <translate time="240">
        <point X="88.12987" Y="0" Z="0"/>
        <point X="62.31723" Y="0" Z="62.31723"/>
        <point X="0" Y="0" Z="88.12987"/>
        <point X="-62.31723" Y="0" Z="62.31723"/>
        <point X="-88.12987" Y="0" Z="0"/>
        <point X="-62.31723" Y="0" Z="-62.31723"/>
        <point X="0" Y="0" Z="-88.12987"/>
        <point X="62.31723" Y="0" Z="-62.31723"/>
    </translate>
    <rotate time="33" X="0" Y="1" Z="0"/>
    <scale X='1.125' Y='1.125' Z='1.125'/>
    <colour R='231' G='232' B='197' />
    <models>
        <model file='../files3d/sphere.3d' />
    </models>

    <!--SATELITE-->
    <group>
        <translate X='0' Y='0' Z='1.875' />
        <rotate time="33" X="0" Y="1" Z="0"/>
        <colour R='255' G='255' B='255' />
        <scale X='0.025' Y='0.025' Z='0.025' />
        <models>
            <model file='../files3d/sphere.3d' />
        </models>
    </group>
    <!--SATELITE-->
    <group>
        <translate X='1.425' Y='0.25' Z='-1.8074' />
        <rotate time="34" X="0" Y="1" Z="0"/>
```

```
        <colour R='255' G='255' B='255' />
        <scale X='0.0175' Y='0.0175' Z='0.0175' />
        <models>
            <model file='../files3d/sphere.3d' />
        </models>
    </group>
    <!--SATELITE-->
    <group>
        <translate X='-1.00875' Y='-0.35' Z='1.3666' />
        <rotate time="35" X="0" Y="1" Z="0"/>
        <colour R='255' G='255' B='255' />
        <scale X='0.05' Y='0.05' Z='0.05' />
        <models>
            <model file='../files3d/sphere.3d' />
        </models>
    </group>
    <!--SATELITE-->
    <group>
        <translate X='1.85175' Y='0' Z='0' />
        <rotate time="36" X="0" Y="1" Z="0"/>
        <colour R='255' G='255' B='255' />
        <scale X='0.055' Y='0.055' Z='0.055' />
        <models>
            <model file='../files3d/sphere.3d' />
        </models>
    </group>

</group>
<!-- SATURN-->
<group>
    <translate time="300">
        <point X="120.5005" Y="0" Z="0"/>
        <point X="85.20672" Y="0" Z="85.20672"/>
        <point X="0" Y="0" Z="120.5005"/>
        <point X="-85.20672" Y="0" Z="85.20672"/>
        <point X="-120.5005" Y="0" Z="0"/>
        <point X="-85.20672" Y="0" Z="-85.20672"/>
        <point X="0" Y="0" Z="-120.5005"/>
        <point X="85.20672" Y="0" Z="-85.20672"/>
    </translate>
    <rotate angle='90' Y='1' />
    <scale X='0.9975' Y='0.9975' Z='0.9975'/>
    <colour R='255' G='223' B='161' />
    <group>
        <rotate time="39" X="0" Y="1" Z="0"/>
        <models>
            <model file='../files3d/sphere.3d' />
        </models>
        <!--SATELITE-->
        <group>
```

```
            <translate X="0" Y="0.6" Z="2..275" />
            <rotate time="39" X="0" Y="1" Z="0"/>
            <colour R="255" G="255" B="255" />
            <scale X="0.0575" Y="0.0575" Z="0.0575" />
            <models>
                <model file="../files3d/sphere.3d" />
            </models>
        </group>
    </group>
    <!-- SATURN'S RINGS -->
    <group>
        <!-- SATURN'S RING 1 -->
        <group>
            <rotate angle="27" X="1"/>
            <scale X='1.35' Y='1.35' />
            <colour R='124' G='87' B='45' />
            <group>
                <rotate time="25" Z="1"/>
            <models>
                <model file='../files3d/ring2.3d' />
            </models>
            </group>
        </group>
        <!-- SATURN'S RING 2 -->
        <group>
            <rotate angle="27" X="1"/>
            <scale X='1.5' Y='1.5' />
            <colour R='182' G='170' B='93' />
            <group>
                <rotate time ="-20" Z="1"/>
            <models>
                <model file='../files3d/ring2.3d' />
            </models>
            </group>
        </group>
        <!-- SATURN'S RING 3-->
        <group>
            <rotate angle="27" X="1"/>
            <scale X='1.65' Y='1.65' />
            <colour R='124' G='87' B='45' />
            <group>
                <rotate time=" 30" Z="1" />
                <models>
                <model file='../files3d/ring2.3d' />
            </models>
            </group>
        </group>
    </group>

</group>
```

```
<!-- URANUS -->
<group>
    <translate time="375">
        <point X="158.0747" Y="0" Z="0"/>
        <point X="111.77569" Y="0" Z="111.77569"/>
        <point X="0" Y="0" Z="158.0747"/>
        <point X="-111.77569" Y="0" Z="111.77569"/>
        <point X="-158.0747" Y="0" Z="0"/>
        <point X="-111.77569" Y="0" Z="-111.77569"/>
        <point X="0" Y="0" Z="-158.0747"/>
        <point X="111.77569" Y="0" Z="-111.77569"/>
    </translate>
    <rotate angle="98" X="1" Y="0" Z="0"/>
    <scale X='0.645' Y='0.645' Z='0.645'/>
    <colour R='148' G='193' B='255' />
    <group>
        <rotate time="48" X="0" Y="-1" Z="0"/>
        <models>
            <model file='../files3d/sphere.3d' />
        </models>
    </group>
</group>
<!-- NEPTUN -->
<group>
    <translate time="450">
        <point X="201.35529" Y="0" Z="0"/>
        <point X="142.37969" Y="0" Z="142.37969"/>
        <point X="0" Y="0" Z="201.35529"/>
        <point X="-142.37969" Y="0" Z="142.37969"/>
        <point X="-201.35529" Y="0" Z="0"/>
        <point X="-142.37969" Y="0" Z="-142.37969"/>
        <point X="0" Y="0" Z="-201.35529"/>
        <point X="142.37969" Y="0" Z="-142.37969"/>
    </translate>
    <rotate angle="30" X="0" Y="1" Z="1"/>
    <scale X='0.6375' Y='0.6375' Z='0.6375'/>
    <colour R='0' G='83' B='255' />
    <group>
        <rotate time="60" X="0" Y="1" Z="0"/>
        <models>
            <model file='../files3d/sphere.3d' />
        </models>
    <!--SATELITE-->
    <group>
        <translate X="0" Y="0.75" Z="1.85" />
        <rotate time="60" X="0" Y="1" Z="0"/>
        <colour R="255" G="255" B="255" />
        <scale X="0.0575" Y="0.0575" Z="0.0575" />
        <models>
            <model file="../files3d/sphere.3d" />
```

```
            </models>
        </group>
        </group>
    </group>

    <!--COMETA -->
    <group>
        <translate time="200">
            <point Y="40" Z="-80" />
            <point Y="36.9552" Z="-39.8182" />
            <point Y="28.2843" Z="-5.75379" />
            <point Y="15.3073" Z="17.0074" />
            <point Y="2.44929e-15" Z="25" />
            <point Y="-15.3073" Z="17.0074" />
            <point Y="-28.2843" Z="-5.75379" />
            <point Y="-36.9552" Z="-39.8182" />
            <point Y="-40" Z="-80" />
            <point Y="-36.9552" Z="-120.182" />
            <point Y="-28.2843" Z="-154.246" />
            <point Y="-15.3073" Z="-177.007" />
            <point Y="-7.34788e-15" Z="-185" />
            <point Y="15.3073" Z="-177.007" />
            <point Y="28.2843" Z="-154.246" />
            <point Y="36.9552" Z="-120.182" />
        </translate>
        <rotate angle="270" X="1" Y="0" Z="0"/>
        <scale X="0.5" Y="0.5" Z="0.5" />
        <models>
            <model file="../files3d/teapot.3d"/>
        </models>
    </group>
    </group>
</scene>
```