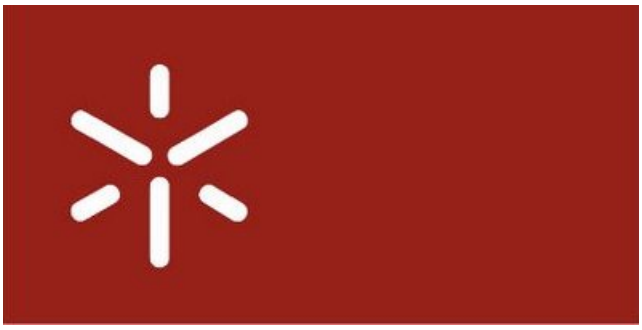

Universidade do Minho

Mestrado Integrado de Engenharia Informática

Computação Gráfica

Fase II: Transformações Geométricas

Adriana Guedes	A74545
José Bastos	A74696
Marco Barbosa	A75278
Ricardo Certo	A75315

Conteúdo

1 Introdução

No seguimento da primeira fase do trabalho, esta fase tem como objetivo efetuar alterações no trabalho já desenvolvido de forma a acrescentar transformações geométricas, tais como, rotações, translações e escalamento. Uma transformação geométrica é, portanto, uma correspondência, um a um, entre pontos de um mesmo plano ou de planos diferentes.

Assim, o principal objetivo deste trabalho é a implementação das transformações geométricas acima referidas, aplicadas a um modelo do Sistema Solar que irá incluir o sol, os planetas e as suas luas, organizado hierarquicamente.

2 Transformações Geométricas

2.1 Rotação

Uma rotação é uma transformação geométrica de um sistema de coordenadas. Tomemos como exemplo a figura que ilustra a rotação em torno do eixo Oz.

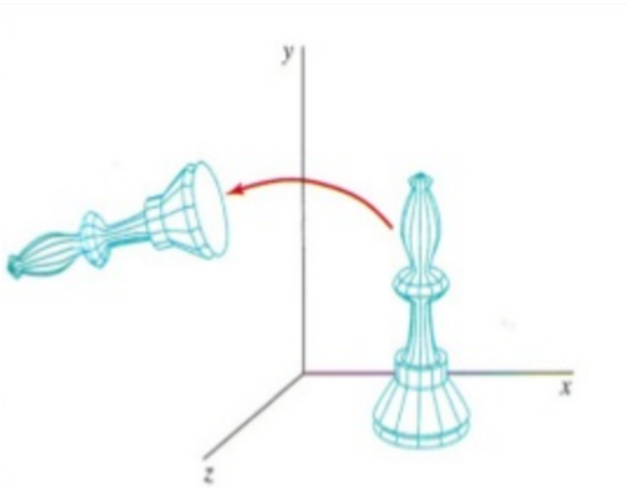

Figura 1: Exemplo de rotação em tordo no eixo Oz

Das transformações gráficas em 2D conseguimos saber que:

```
x' = x * cos(a) { y*sin(a)
y' = x * sin(a) + y*cos(a)
z' = z
```

O parâmetro ***a*** refere-se ao ângulo de rotação escolhido.

Assim, a rotação em torno do eixo Oz pode ser definida de forma matricial da seguinte forma:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figura 2: Matriz relativa à rotação sobre o eixo Oz

2.2 Translação

A translação, quando aplicada a um objeto, reposiciona o mesmo mudando as suas coordenadas (x,y,z) no espaço tridimensional por fatores *Tx*, *Ty* e *Tz*, respetivamente.

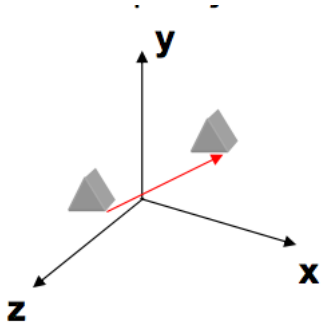

Figura 3: Exemplo da translação de um objeto

Considerando um objeto com coordenadas (x,y,z), a sua nova posição (x',y',z') após uma translação será:

```
x'= x + Tx
y'= y + Ty
z'= z + Tz
```

$$\begin{bmatrix} x_T \\ y_T \\ z_T \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figura 4: Definição em forma matricial da translação

2.3 Escalamento

O escalamento de um objeto altera o tamanho deste, multiplicando as coordenadas (x,y,z) por fatores *sx*, *sy* e *sz*, não nulos.

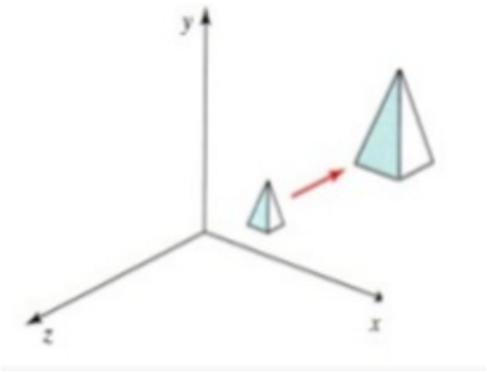

Figura 5: Exemplo do escalamento de um objeto

Pegando num ponto de um objeto com coordenadas (x,y,z), as novas coordenadas (x',y',z') serão:

```
x' = x * sx
y' = y * sy
z' = z * sz
```

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figura 6: Definição em forma matricial do escalamento

3 Estrutura de Dados

3.1 Classes

De modo a simplificar a implementação do sistema solar, construímos diversas classes, classes essas que são: **Ponto, Rotação, Aplicação, Translação, Escala** e **Transformação.**

Todas estas classes tem os seus respetivos construtores e as respetivas funções que permitem alterar ou obter os valores de cada variável (gets e sets).

- **Ponto**

  A classe **Ponto** guarda a informação relativa a um ponto representado no espaço, ou seja, guarda os valores de cada coordenada em forma de *float*.

```
#pragma once

class Ponto{
    float x;
    float y;
    float z;

public:
     Ponto();
     Ponto(float, float, float);
     float getX() { return x; }
     float getY() { return y; }
     float getZ() { return z; }
     void setX( float a) {  x = a; }
     void setY( float a) {  y = a; }
     void setZ( float a) {  z = a; }

};
```

- **Escala**

  A classe Escala guarda a informação correspondente aos valores da escala em relação a cada eixo, em forma de *float*.

```
#pragma once

class Escala{
    float x;
    float y;
    float z;

public:
     Escala();
     Escala(float, float, float);
     float getX() { return x; }
     float getY() { return y; }
     float getZ() { return z; }
     void setX( float a) {  x = a; }
     void setY( float a) {  y = a; }
     void setZ( float a) {  z = a; }
};
```

- **Rotação**

  A classe Rotação armazena a informação relativa a um ângulo e a três variáveis. O ângulo indica-nos como vai rodar a figura, e os outros três parâmetros, se algum deles tiver o valor 0, significa que a rotação não é feita nesse eixo, mas se obtiver o valor 1, significa que a rotação é realizada em torno desse eixo.

```
#pragma once

class Rotacao{
    float angulo;
    float eixoX;
    float eixoY;
    float eixoZ;

public:
     Rotacao();
     Rotacao(float, float, float, float);
     float getAngulo() { return angulo; }
     float geteixoX() { return eixoX; }
     float geteixoY() { return eixoY; }
     float geteixoZ() { return eixoZ; }
     void setAngulo(float a){ angulo = a; }
     void setEixoX(float x){ eixoX = x; }
     void setEixoY(float y){ eixoY = y; }
     void setEixoZ(float z){ eixoZ = z; }
};
```

- **Translação**

  A classe Translação guarda as coordenadas do ponto para qual um objeto deve ser movido.

```
#pragma once
#define _MAIN
#include <vector>

class Translacao{
     float x;
     float y;
     float z;

public:
     Translacao();
     Translacao(float x, float y, float z);
     float getX() { return x; }
```

```
     float getY() { return y; }
     float getZ() { return z; }
     void setX( float a) {  x = a; }
     void setY( float a) {  y = a; }
     void setZ( float a) {  z = a; }
};
```

- **Cor**

  A classe Cor vai ser responsável por atribuir uma cor em formato RGB a um dado planeta.

```
#pragma once

class Cor{
    float r;
    float g;
    float b;

public:
    Cor();
    Cor(float, float, float);
    float getR() { return r; }
    float getG() { return g; }
    float getB() { return b; }
    void setR( float a) {  r = a; }
    void setG( float a) {  g = a; }
    void setB( float a) {  b = a; }
};
```

- **Transformação**

  A classe Transformação vai ser a classe responsável por guardar um objeto do tipo Translação, um do tipo Escala, um do tipo Cor e um do tipo Rotação.

```
#pragma once
#include "translacao.h"
#include "rotacao.h"
#include "escala.h"
#include "cor.h"

class Transformacao{
    Translacao translacao;
    Rotacao rotacao;
    Escala escala;
```

```
        Cor cor;

    public:
        Transformacao();
        Transformacao(Translacao, Rotacao, Escala, Cor);
        Translacao getTranslacao() { return translacao; }
        Rotacao getRotacao() { return rotacao; }
        Escala getEscala() { return escala; }
        Cor getCor() { return cor; }
        void setTranslacao(Translacao t){ translacao = t; }
        void setRotacao(Rotacao r){ rotacao = r; }
        void setEscala(Escala e){ escala = e; }
        void setCor(Cor c){ cor = c; }
    };
```

- **Aplicação**

  A classe Aplicação é responsável por guardar toda a informação de uma dada aplicação, ou seja, o seu respetivo nome, a transformação que irá sofrer, os seus filhos (em que cada um destes também representa uma aplicação), e os pontos necessários para ser possível o desenho da figura.

```
#pragma once
#include <vector>
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
#include "ponto.h"
#include "transformacao.h"
#include <fstream>
#include <iostream>
#include <string>

using namespace std;

class Aplicacao{

    string nome;
    vector<Aplicacao> filhos;
    vector<Ponto> pontos;
    Transformacao transformacao;


public:
```

```
    Aplicacao();
    Aplicacao(string, vector<Aplicacao>, vector<Ponto>, Transformacao);
    vector<Ponto> getPontos() {return pontos;}
    vector<Aplicacao> getFilhos() { return filhos;}
    string getNome() { return nome;}
    Transformacao getTransformacao() { return transformacao;}
    void setNome( string n) { nome=n;}
    void setPontos(vector<Ponto> p) {pontos = p;}
    void setFilhos(vector<Aplicacao> f) {filhos=f;}
    void setTransformacao(Transformacao t) { transformacao = t;}
};
```

4 Leitura do XML

Para esta segunda fase do trabalho tivemos que adaptar o *parsing* do XML que tínhamos na fase anterior de modo a cumprirmos com os novos requisitos desta segunda fase. Para que isso fosse possível tivemos de utilizar os métodos **FirstChildElement()** e **NextSiblingElement()**. Ao usarmos o **FirstChildElement()** vamos obter a primeira tag do nível imediatamente a seguir à anterior, e quando usamos o **NextSiblingElement()** vamos obter a próxima tag do mesmo nível da anterior.

Com o auxílio destes métodos percorremos as todas as transformações, verificando para cada uma quais eram os seus tipos e consoante isso guardar os valores obtidos nas respetivas classes previamente criadas. Uma vez que o filho tem de herdar as transformações do pai, o próximo passo foi fazer com que isso fosse possível. Para a translação e para rotação somamos os novos valores aos antigos, mas para a escala fazemos a multiplicação dos novos valores pelos antigos e para a cor apenas herdamos a cor do pai.

De seguida vamos verificar quais os modelos que sofreram transformações e depois guardar toda essa informação obtida numa classe aplicação de forma que mais adiante essa aplicação possa ser desenhada com as devidas transformações. Por fim, vai verificar se ainda falta realizar o *parse* aos seus irmãos e/ou aos seus filhos, se faltar algum volta aplicar este processo acima descrito novamente.

```
<cena>
    <grupo>
        <escala X="0.1" Y="0.1" Z="0.1"/>
        <modelos>
                <modelo ficheiro = "esfera.3d"/>
        </modelos>
            <grupo>
                <escala X="1.0" Y="1.0" Z="1.0"/>
                <translacao X="10.0" Y="0" Z="0"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>
            </grupo>
    </grupo>
        <translacao X="10.0" Y="20.0" Z="0"/>
        <modelos>
                <modelo ficheiro = "esfera.3d"/>
        </modelos>
    <grupo>
    </grupo>
</cena>
```

1 -> FirstChildElement()

2 -> NextSiblingElement()

Figura 7: Exemplo de um ficheiro XML

Na figura acima podemos observar como as funções **FirstChildElement()** e **NextSiblingElement()** percorrem um ficheiro XML. Ao longo da imple-

mentação do código tivemos que as usar dependendo dos parâmetros do ficheiro que queríamos aceder.

5 Ciclo de Rendering

Para o desenho das aplicações implementamos um ciclo que é capaz de percorrer todas elas. Por cada iteração do ciclo fazemos um **getTransformacao()** para obtermos uma transformação. De seguida, utilizando a transformação para obtermos os dados respetivos às rotações, às translações, às cores e à escala , utilizamos esses dados como parâmetros nas funções do GLUT para o processo de desenho. Para isso, recorremos às funções **glRotate() , glScale() , glTranslated()** e **glColor3f()**. Depois de realizadas todas as transformações vamos proceder ao desenho dos triângulos usando a função **glBegin(GL_TRIANGLES)**.

6 Análise de Resultados

6.1 Ficheiro XML

Como forma de teste ao nosso Motor3D tivemos que criar vários ficheiros XML de teste. A realização destes testes ao nosso Motor3D foi uma parte importante no trabalho desenvolvido, pois foi-nos possível verificar se toda a nossa implementação estava correta.

O nosso ficheiro final possui diferentes hierarquias entre os diferentes grupos de forma a que consigamos desenhar os planetas e os seus respetivos satélites naturais. As esferas foram criadas pelo gerador concebido numa primeira fase do projeto. No ficheiro que se segue pode-mos observar como foi concebido o XML:

```
<cena>

    <grupo>

        <!-- SOL -->
        <grupo>
            <escala X="1" Y="1" Z="1"/>
            <cor R="1" G="1" B="0"/>
            <modelos>
                <modelo ficheiro = "esfera.3d"/>
            </modelos>
        </grupo>

        <!--MERCÚRIO-->
        <grupo>
            <escala X="0.3" Y="0.3" Z="0.3"/>
            <translacao X="8.0" Y="0" Z="0"/>
            <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
            <cor R="9.6" G="0.4" B="0.3"/>
            <modelos>
                <modelo ficheiro = "esfera.3d"/>
            </modelos>
        </grupo>

        <!--VÉNUS-->
        <grupo>
            <escala X="0.32" Y="0.32" Z="0.32"/>
            <translacao X="18.0" Y="0" Z="0"/>
            <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
            <cor R="0.6" G="0.4" B="0.3"/>
```

```
    <modelos>
        <modelo ficheiro = "esfera.3d"/>
    </modelos>
</grupo>

<!--TERRA-->
<grupo>
    <escala X="0.35" Y="0.35" Z="0.35"/>
    <translacao X="25.0" Y="0" Z="0"/>
    <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
    <cor R="0" G="0" B="3.3"/>
    <modelos>
        <modelo ficheiro = "esfera.3d"/>
    </modelos>
        <!--LUA-->
        <grupo>
                <escala X="0.15" Y="0.15" Z="0.15"/>
                <translacao X="0" Y="1.2" Z="0"/>
                <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                <cor R="1" G="2" B="24"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>
        </grupo>
</grupo>

<!--MARTE-->
<grupo>
    <escala X="0.31" Y="0.31" Z="0.31"/>
    <translacao X="45.0" Y="0" Z="0"/>
    <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
    <cor R="1" G="0" B="0.3"/>
    <modelos>
        <modelo ficheiro = "esfera.3d"/>
    </modelos>
</grupo>

<!--JÚPITER-->
<grupo>
    <escala X="2.3" Y="2.3" Z="2.3"/>
    <translacao X="65.0" Y="0" Z="0"/>
    <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
    <cor R="3.6" G="0.4" B="0.0"/>
    <modelos>
        <modelo ficheiro = "esfera.3d"/>
    </modelos>
```

```
        <!--JÚPITER_Ganymede-->
        <grupo>
                <escala X="0.15" Y="0.15" Z="0.15"/>
                <translacao X="4.9" Y="2.5" Z="2.2"/>
                <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>
        </grupo>
        <!--JÚPITER_Callisto-->
        <grupo>
                <escala X="0.15" Y="0.15" Z="0.15"/>
                <translacao X="5.8" Y="5.5" Z="0"/>
                <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>
        </grupo>
        <!--JÚPITER_Io-->
        <grupo>
                <escala X="0.15" Y="0.15" Z="0.15"/>
                <translacao X="-5.8" Y="-4.2" Z="0"/>
                <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>
        </grupo>
        <!--JÚPITER_Europa-->
        <grupo>
                <escala X="0.15" Y="0.15" Z="0.15"/>
                <translacao X="-5.8" Y="5.30" Z="-7.0"/>
                <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>
        </grupo>
</grupo>

<!--SATURNO-->
<grupo>
    <escala X="1.9" Y="1.9" Z="1.9"/>
    <translacao X="95.0" Y="0" Z="0"/>
    <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
    <cor R="1.0" G="0.8" B="0.7"/>
    <modelos>
        <modelo ficheiro = "esfera.3d"/>
```

```
    </modelos>

    <!--SATURNO_Titan-->
     <grupo>
                <escala X="0.16" Y="0.16" Z="0.16"/>
                <translacao X="4.9" Y="3" Z="2.4"/>
                <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>
    </grupo>
    <!--SATURNO_Rhea-->
        <grupo>
            <escala X="0.16" Y="0.16" Z="0.16"/>
            <translacao X="4.8" Y="-4.5" Z="-4"/>
            <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
            <modelos>
                <modelo ficheiro = "esfera.3d"/>
            </modelos>
        </grupo>
    <!--SATURNO_Kapetus-->
        <grupo>
            <escala X="0.16" Y="0.16" Z="0.16"/>
            <translacao X="-3.9" Y="0" Z="-4"/>
            <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
            <modelos>
                <modelo ficheiro = "esfera.3d"/>
            </modelos>
        </grupo>
    <!--SATURNO_Dione-->
        <grupo>
                <escala X="0.16" Y="0.16" Z="0.16"/>
                <translacao X="-5.9" Y="0" Z="5.2"/>
                <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>
        </grupo>
</grupo>


<!--URANO-->
<grupo>
    <escala X="0.9" Y="0.9" Z="0.9"/>
    <translacao X="125.0" Y="0" Z="0"/>
    <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
```

```
            <cor R="0.3" G="2.4" B="3.3"/>
            <modelos>
                <modelo ficheiro = "esfera.3d"/>
            </modelos>
        </grupo>

        <!--NEPTUNO-->
        <grupo>
                <escala X="0.7" Y="0.7" Z="0.7"/>
                <translacao X="135.0" Y="0" Z="0"/>
                <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                <cor R="0.3" G="0.3" B="3.3"/>
                <modelos>
                    <modelo ficheiro = "esfera.3d"/>
                </modelos>

                <!--NEPTUNO_Tritão-->
                <grupo>
                        <escala X="0.1" Y="0.1" Z="0.1"/>
                        <translacao X="0" Y="-2.5" Z="2.2"/>
                        <rotacao angulo="45" eixoX="0" eixoY="0" eixoZ="0"/>
                        <modelos>
                            <modelo ficheiro = "esfera.3d"/>
                        </modelos>
            </grupo>
        </grupo>


    </grupo>

</cena>
```

6.2 Primitivas Extra

De maneira a completar e enaltecer um pouco mais o trabalho realizado, tomámos a iniciativa de "brincar"um pouco com o openGL e conseguimos criar:

- **Nave**

  Decidimos implementar uma pequena "nave", que não é mais do que um conjunto de primitas do *glut* (esferas e um cone) dispostas de uma determinada maneira no espaço. Para conseguirmos tal feito, implementamos uma função *void defineShip()* que contém todas as transformações necessárias, desde translações, mudanças de cor e escala,etc. Para realizarmos tal feito, foi desenhado o seguinte esquema para a nave:

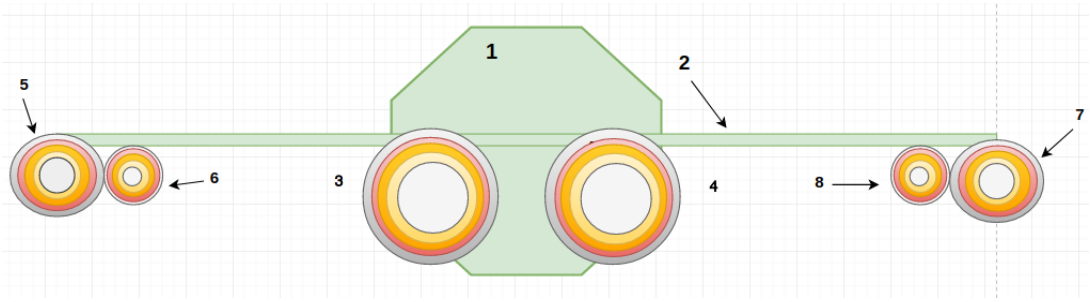

Figura 8: Nave Esquematizada

Note-se que no esquema apresentado, as esferas representadas com os números 3,4,5,6,7 e 8 também contêm outras esferas no seu "interior", apenas não foram numeradas no esquema.

Posto este esquema, considere-se o seguinte excerto de código que permite a sua materialização:

```
 void defineShip(){
    glPushMatrix();
    glScalef(0.2,0.2,0.2);
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    //1
    glColor3f(0.0f,1.0f,0.0f);
    glScalef(0.3, 0.3, 0.3);
    glutSolidCone(1,1,10,10);
    glColor3f(0.0f,1.0f,0.0f);
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    //2
    glPushMatrix();
    glTranslated(0,0,0);
    glScalef(5.0,0.15,0);
    glutSolidCube(2);
    glPopMatrix();
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
  //3
   glColor3f(0.25f,0.25f,0.25f);
   glPushMatrix();
    glScalef(0.35,0.35,0.35);
    glTranslated(-3,-1,4);
    glutWireSphere(2,50,50);
    glPopMatrix();
  //13
```

```
  glColor3f(1.0f,0.0f,0.0f);
  glPushMatrix();
   glScalef(0.55,0.55,0.55);
    glTranslated(-1.79,-0.15,5);
    glutWireSphere(1,50,50);
glPopMatrix();
```

Tendo sido produzido o seguinte resultado final:

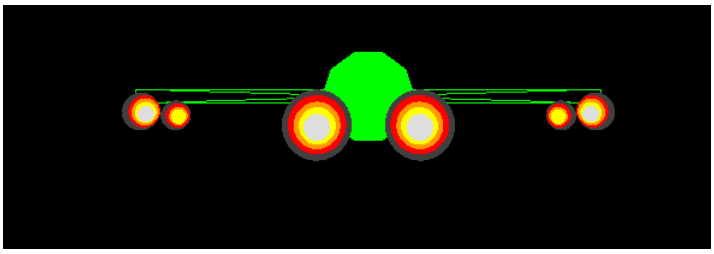

Figura 9: Nave Final

- **Câmara e movimentação em 3rd Person**

  Para a nave criada não ser algo estático e apenas decorativo, decidimos dar-lhe alguma vivacidade e aumentar a interabilidade do utilizador com o programa, através da implementação de uma câmara na terceira pessoa. Esta câmara, permite uma visualização da nave no espaço e, associada à movimentação da nave, permite simular uma viagem pelos diversos planetas, de forma a que possam ser melhor apreciados todos os detalhes implementados, como por exemplo, as luas existentes e o anel de Saturno.

- **Anel de Saturno**

  Para dar uma maior veracidade ao sistema solar desenvolvido, foi criado um anel à volta de Saturno, o qual permite facilmente identificar este planeta. Para tal feito, e como se tratava de algo extra, implementamo-lo diretamente a partir do torus do *Glut*, da seguinte forma:

```
glColor3f(0.4f,0.21f,0.0f);
glPushMatrix();
    glScalef(1.8,0.5,1.8);
    glTranslated(52.7,0,0);
    glRotatef(120.0, 1.0, 0.0, 0.0);
    glutWireTorus(0.25,1.75,28,28);
```

```
glPopMatrix();
```

Como se pode ver, o torus assume uma cor castanha, encontra-se achatado no eixo do y e esticado no eixo do x e do z. De forma a alinha-lo com o planeta Saturno foi necessário coloca-lo na posição correta do espaço e, para uma maior notabilidade, decidimos inclina-lo segundo o eixo do x

- **Cor**

  Como já foi referido, considerámos interessante colorir os planetas, sendo que para isso incluimos uma diretriz *cor* no ficheiro *XML*, que recebe três valores (um para R, outro G e outro para B), permitindo-nos assim especificar a cor que desejámos para cada planeta. Note-se que decidimos deixar todas as luas com cor branca, de forma a não serem confundidas com planetas.

6.3 Resultado Final

O resultado final foi a elaboração de um esboço do Sistema Solar a três dimensões. Não usamos os valores reais para as escalas e nem para as distâncias entre os planetas mas sim valores arbitrários na ordem de grandezas que desejávamos visualizar.

Os resultados que obtivemos podemos verificar nas imagens que se seguem que são o sistema solar visto de várias perspetivas:

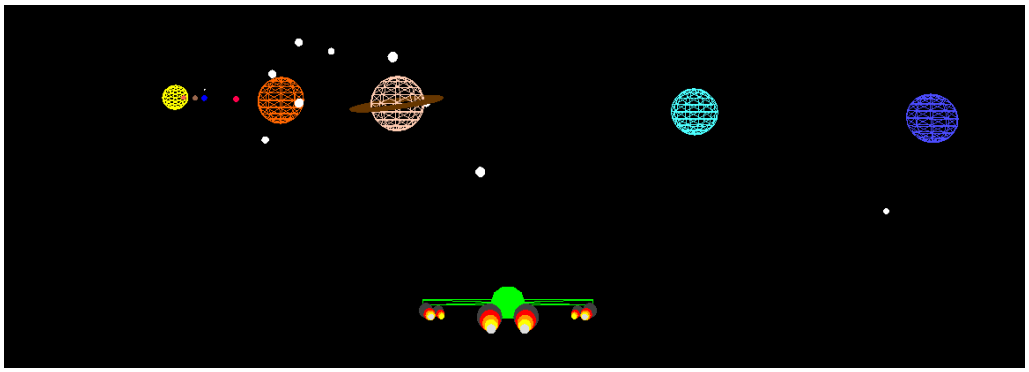

Figura 10: Perspetiva 1 do Sistema Solar

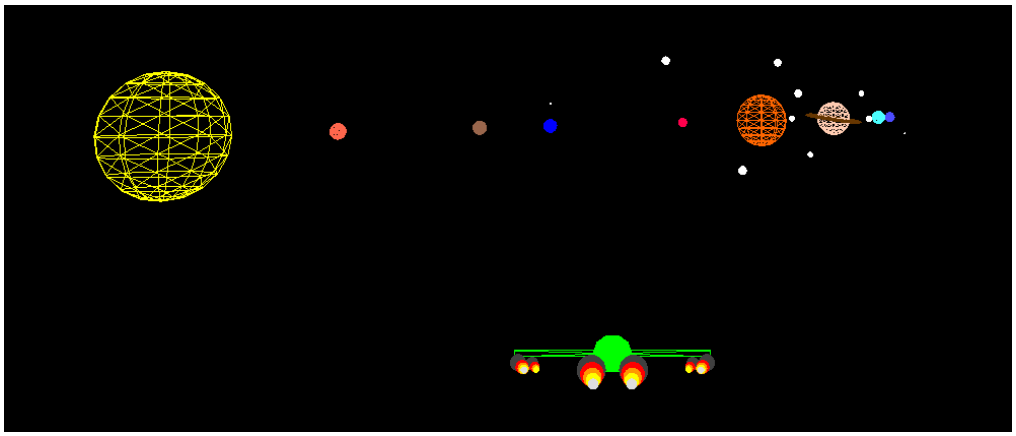

Figura 11: Perspetiva 2 do Sistema Solar

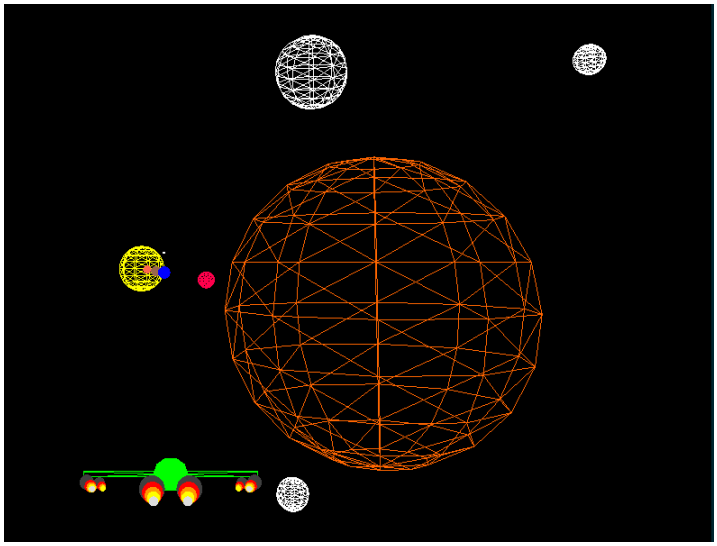

Figura 12: Perspetiva 3 do Sistema Solar

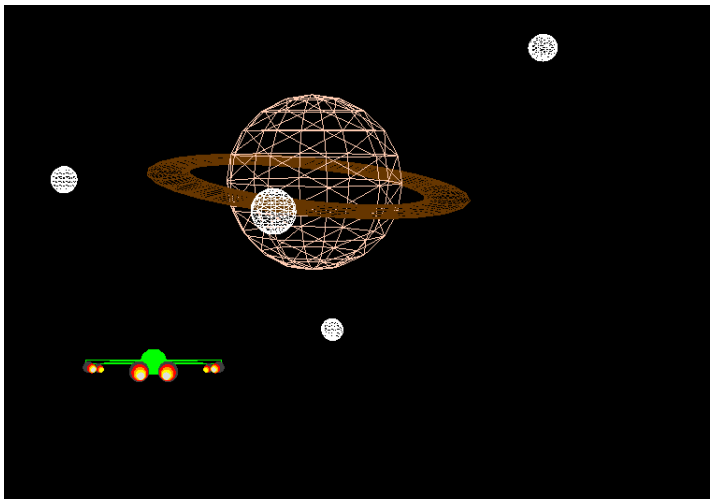

Figura 13: Planeta Saturno, com o seu anel e luas

7 Conclusão e trabalho futuro

Nesta segunda fase do trabalho foi-nos possível aprofundar e por em práticas os conceitos de transformações abordados e trabalhados nas aulas. Apresentamos assim um Sistema Solar, ainda bastante simples mas com o objetivo de nas fases que se seguem melhorar alguns aspetos e ainda implementar outros, como por exemplo, as órbitas onde os planetas se situam, a cintura de asteróides, etc..