



Universidade do Minho  
Escola de Engenharia  
Licenciatura em Engenharia Informática

## Unidade Curricular de Computação Gráfica

Ano Letivo de 2022/2023

# Phase 2 – Geometric Trans- forms

Grupo 34

João Moreira a93326  
Daniel Faria a93191  
Paulo Filipe Cruz a80949

14 de abril de 2023

# CG

# Índice

<b>1. Introdução</b>	<b>1</b>
<b>2. Generator</b>	<b>2</b>
2.1. Toro . . . . .	2
<b>3. Engine</b>	<b>4</b>
3.1. Leitura do ficheiro XML . . . . .	5
3.2. Desenho das primitivas . . . . .	5
<b>4. Extra</b>	<b>7</b>
4.1. Modelo do Sistema Solar . . . . .	7
<b>5. Testes</b>	<b>9</b>
5.1. Ficheiros <i>test</i> . . . . .	9
5.2. Sistema Solar . . . . .	11
<b>6. Conclusão</b>	<b>13</b>
<b>A. Ficheiro XML - Sistema Solar</b>	<b>14</b>

# Lista de Figuras

2.1. Variáveis representadas graficamente no Toro. . . . .	2
2.2. Toro dividido em <i>slices</i> e <i>stacks</i> . . . . .	3
4.1. Informações sobre as funcionalidades implementadas. . . . .	7
5.1. Ficheiro test_2_1.xml . . . . .	9
5.2. Ficheiro test_2_2.xml . . . . .	10
5.3. Ficheiro test_2_3.xml . . . . .	10
5.4. Ficheiro test_2_4.xml . . . . .	11
5.5. Ficheiro solar_system.xml . . . . .	12

# 1. Introdução

O presente documento é alusivo à segunda fase do projeto prático desenvolvido com recurso à linguagem de programação C++, no âmbito da Unidade Curricular de Computação Gráfica que integra a Licenciatura em Engenharia Informática da Universidade do Minho. Este projeto encontra-se dividido em quatro fases de trabalho, cada uma com uma data de entrega específica. Esta divisão em fases, pretende fomentar uma simplificação e organização do trabalho, contribuindo para a sua melhor compreensão.

Pretende-se, assim, que o relatório sirva de suporte ao trabalho realizado para esta fase, mais propriamente, dando uma explicação e elucidando o conjunto de decisões tomadas ao longo da construção de todo o código fonte e descrevendo a estratégia utilizada para a concretização dos principais objetivos propostos, que surgem a seguir:

- Compreender a utilização do *OpenGL*, recorrendo à biblioteca *GLUT*, para a construção de modelos 3D;
- Aprofundar temas alusivos à produção destes modelos 3D, nomeadamente, em relação a transformações geométricas, curvas, superfícies, iluminação, texturas e modo de construção geométrico básico;
- Relacionar todo o conceito de construir modelos 3D com o auxílio da criação de ficheiros que guardam informação relevante nesse âmbito;
- Relacionar aspetos mais teóricos com a sua aplicação a nível mais prático.

Naturalmente, é indispensável que o conjunto de objetivos supracitados seja concretizado com sucesso e, para isso, o formato do relatório está organizado de acordo com uma descrição do problema inicial, seguindo-se o conjunto de aspetos relevantes em relação ao *Generator* e chegando aos aspetos primordiais sobre o *Engine*. O grupo decidiu incluir também uma secção direccionada para a descrição das funcionalidades adicionais e, por fim, amostra dos testes realizados.

Nesta segunda fase do projeto o objetivo é desenvolver novas funcionalidades no *Engine* para interpretar e processar grupos de transformações geométricas, tais como translações, escalas e rotações especificadas no ficheiro XML. Além disso, também é necessário criar um ficheiro desta categoria de forma a desenhar um modelo estático do sistema solar.

## 2. Generator

Nesta fase, para viabilizar a execução do arquivo de demonstração solicitado, foi necessário adicionar uma nova figura, o toro.

### 2.1. Toro

Para construir o toro, procedemos à parametrização deste, segundo as fórmulas que se seguem e onde constam as variáveis  $R$  e  $r$  onde a primeira é o raio da origem ao interior do toro - “radius”, o segundo é o raio do anel interior do toro - “ring radius”, como se vê na figura seguinte.

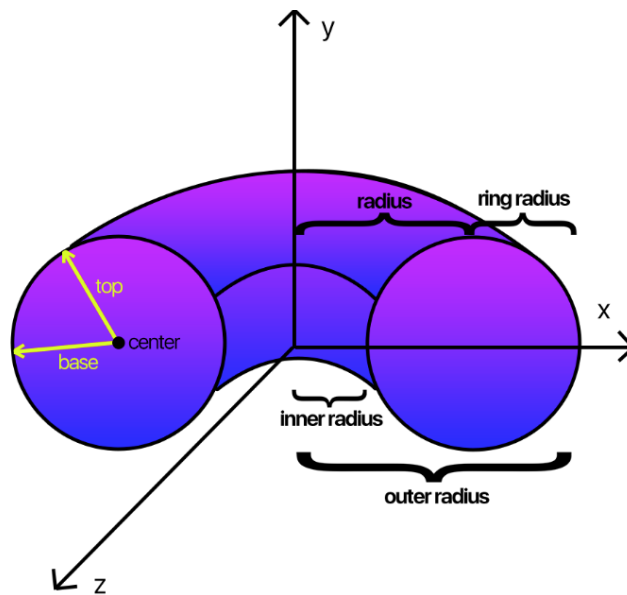


Figura 2.1.: Variáveis representadas graficamente no Toro.

$$x = R * \cos\theta + r * \cos\phi * \cos\theta$$

$$y = R * \sin\theta + r * \cos\phi * \sin\theta$$

$$z = r * \sin\phi$$

$$0 \leq \phi < 2\pi$$

$$0 \leq \theta < 2\pi$$

A função responsável por esta criação designa-se por *generate\_torus* e recebe o conjunto de parâmetros seguintes: *in\_radius*, *out\_radius*, *slices*, *stacks*, *.3d filename*, que a partir dos dois primeiros conseguimos calcular o *R* e o *r*, necessários para gerar os pontos do toro.

Listing 2.1: Parametrização do Toro

```
float R = ((out_radius - in_radius) / 2) + in_radius;
float r = ((out_radius - in_radius) / 2);
float teta = (M_PI * 2) / slices;
float fi = (M_PI * 2) / stacks;
Point p1, p2, p3, p4;
```

Para além disso, vamos dividir as variáveis  $\phi$  e  $\theta$  em *stacks* e *slices*, respetivamente. As variáveis *in\_radius*, *out\_radius* são o raio interior e o raio superior do toro respetivamente.

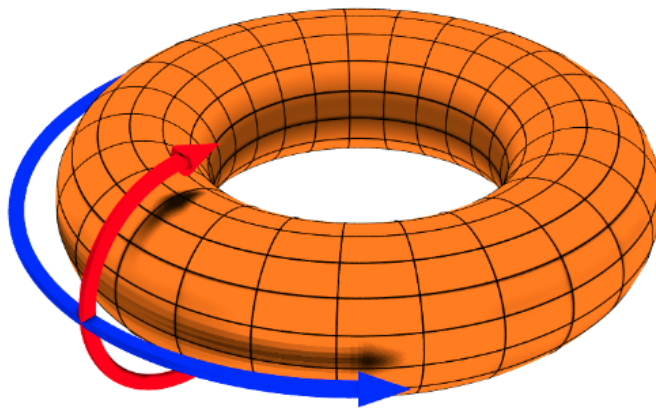


Figura 2.2.: Toro dividido em *slices* e *stacks*.

## 3. Engine

Como houve uma alteração dos ficheiros XML dos que eram tratados na primeira fase, isto é, agora há mais informação (transformações geométricas) a ser armazenada houve a necessidade de alterar a nossa estrutura de dados para tal.

A estrutura que representa e armazena uma **transformação geométrica** consiste em:

---

```
// Enumerator que representa as transformações geométricas das figuras.
enum transformation{
    translate,
    rotate,
    scale
};

// Representação de transformações geométricas com dados relativos para as mesmas.
struct Transform{
    float x;
    float y;
    float z;
    float angle;
    transformation type;
};
```

---

Por outro lado, para armazenar os **dados de um determinado grupo** recorreremos à seguinte estrutura:

---

```
/* Representação de um Group, do ficheiro xml, com as transformações geométricas
que são aplicadas, triângulos necessários para a construção da figura
e subgrupos que possa conter. */
struct Group{
    std :: vector<Transform> transformations;
    std :: vector<Triangle> triangles;
    std :: vector<Group> subGroups;
};
```

---

Quanto à **estrutura de dados principal**:

---

```
/* Estrutura que guarda toda a informação pertinente para a representação do
problema. */
struct WorldInfo {
    int width;
    int height;
    CameraParameters camera;
    std :: vector<Group> groups;
};

// Representação do modelo.
WorldInfo world;
```

---

Nesse sentido, a estrutura principal (*WorldInfo*) armazena a estrutura de dados da câmara, as propriedades da janela a abrir e um vetor de estruturas de dados que armazena um grupo. Já um grupo possui um vetor de transformações aplicados a figuras desse grupo, um vetor de triângulos carregados no momento do *parse* do ficheiro XML e um vetor de subgrupos.

## 3.1. Leitura do ficheiro XML

Para a leitura e *parse* deste tipo de ficheiros continuamos a usar a biblioteca *tinyxml2*. Porém, nesta fase já é necessário interpretar as transformações geométricas de um grupo e dos seus subgrupos, pelo que o grupo decidiu alterar a função *parseGroups* que interpreta todos os dados referentes aos grupos e subgrupos, guardando nas estruturas de dados atualizadas.

Além disso foi alterada a função *readTrianglesFromFile* pelo que agora possui um valor de retorno, nomeadamente, um vetor dos triângulos lidos e que é chamada a medida que se efetua o *parse* do ficheiro XML.

## 3.2. Desenho das primitivas

Para desenhar as primitivas com a nova estrutura de dados foi atualizada a função *drawGroup*.

Esta função itera por cada grupo presente na estrutura principal *world.groups* e para cada um faz *push* da matriz das transformações através da função do *glut glPushMatrix*. Posteriormente percorremos o *array* das transformações e executamos por ordem que aparece no ficheiro,

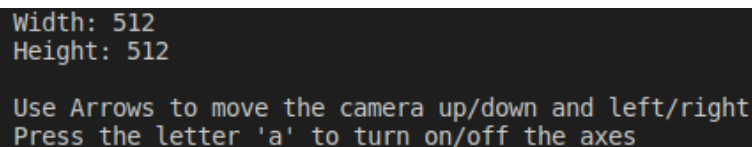
atraves das funções *glTranslatef*, *glScalef* e *glRotatef*. Depois desenhamos a respetiva figura deste grupo com os pontos armazenados com ajuda da função *drawPrimitives*. Antes de fazer *pop* voltamos a chamar recursivamente a função (*drawGroup*) para executar as transformações dos subgrupos e só depois disso é que fazemos *glPopMatrix*. Deste modo, os subgrupos herdam as transformações do grupo pai.



## 4. Extra

Quanto aos extras, nesta fase implementamos uma nova figura geométrica, ou seja, o toro que é necessário para representar a *demo* pedida. Para além disso, também adicionamos a opção de mudar o estado do sistema de eixos, ou seja, desativar ou ativar conforme a necessidade de querer ver os eixos ou não.

Por fim, foi implementado a movimentação da câmara o que permite ver o ficheiro executado de diferentes ângulos, cima para baixo e/ou esquerda para a direita.



```
Width: 512
Height: 512

Use Arrows to move the camera up/down and left/right
Press the letter 'a' to turn on/off the axes
```

Figura 4.1.: Informações sobre as funcionalidades implementadas.

### 4.1. Modelo do Sistema Solar

Antes de desenvolver o modelo do sistema solar é preciso ter em consideração as seguintes propriedades:

- o *translate* de um grupo é somado ao do subgrupo;
- o valor do *scale* de um grupo é multiplicado ao valor do subgrupo;
- o valor do *rotate* do grupo é somado ao do subgrupo;

Para compor o sistema solar num ficheiro .xml, decidimos, primeiramente, por saber as distâncias dos planetas em relação ao Sol e da Lua em relação ao planeta Terra e o diâmetro de cada elemento do modelo. Percebemos que, para desenhar, seria bastante difícil usar apenas uma escala para distâncias e diâmetros. Para resolver este problema, decidimos que, para as distâncias a escala seria  $1/20000$  e para os diâmetros seria  $1/20000000$ . Contudo, ao colocar estes valores, verificamos que os planetas se sobrepunham, isto porque a distância estava mal calculada: teria de ser calculada pela distância entre os centros do planeta.

Quanto às inclinações dos planetas, começamos por definir os valores de acordo com a seguinte tabela:

Planeta	Inclinação do eixo de rotação (°)
Mercúrio	7
Vénus	177
Terra	23.5
Marte	25
Júpiter	3
Saturno	27
Urano	98
Neptuno	30

Assim, aplicamos um *rotate* sobre o eixo dos zz com o valor do ângulo de cada inclinação. Como este ângulo é positivo para o lado esquerdo, é necessário aplicar o valor negativo ao ângulo para a inclinação se verificar para o lado direito.

## 5. Testes

Para mostrar em funcionamento o trabalho exposto ao longo deste relatório realizamos os seguintes testes:

### 5.1. Ficheiros *test*

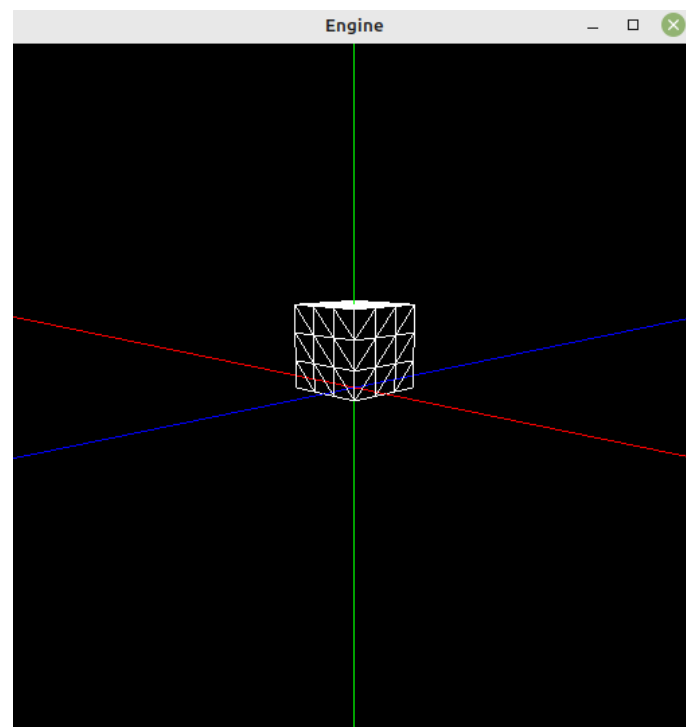


Figura 5.1.: Ficheiro test\_2\_1.xml

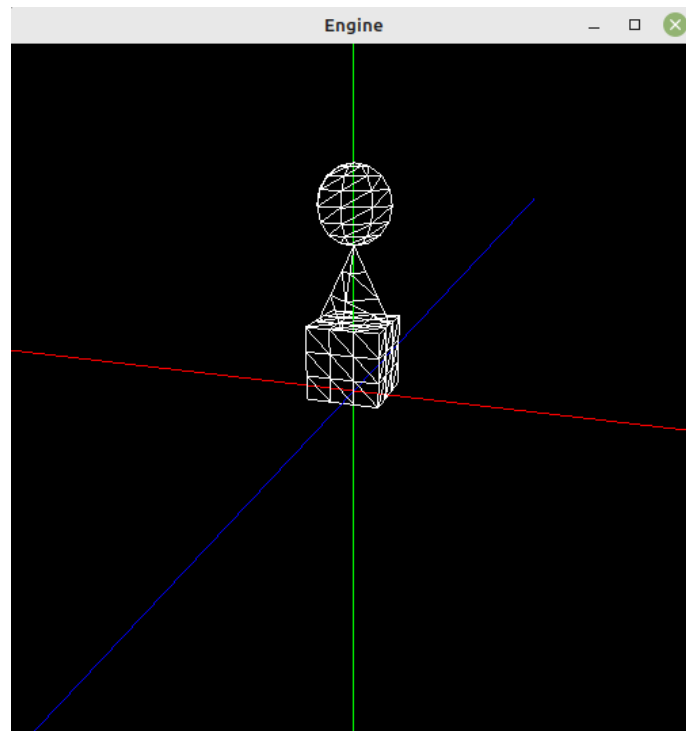


Figura 5.2.: Ficheiro test\_2\_2.xml

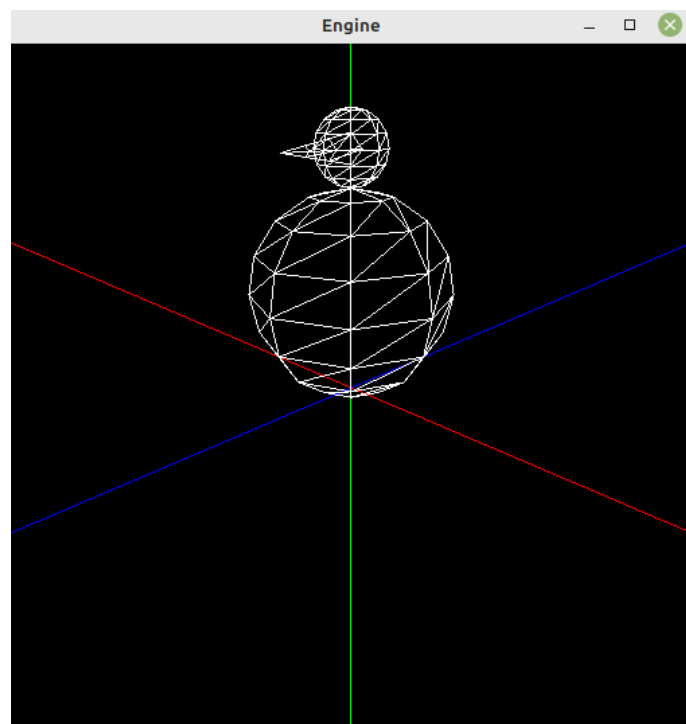


Figura 5.3.: Ficheiro test\_2\_3.xml

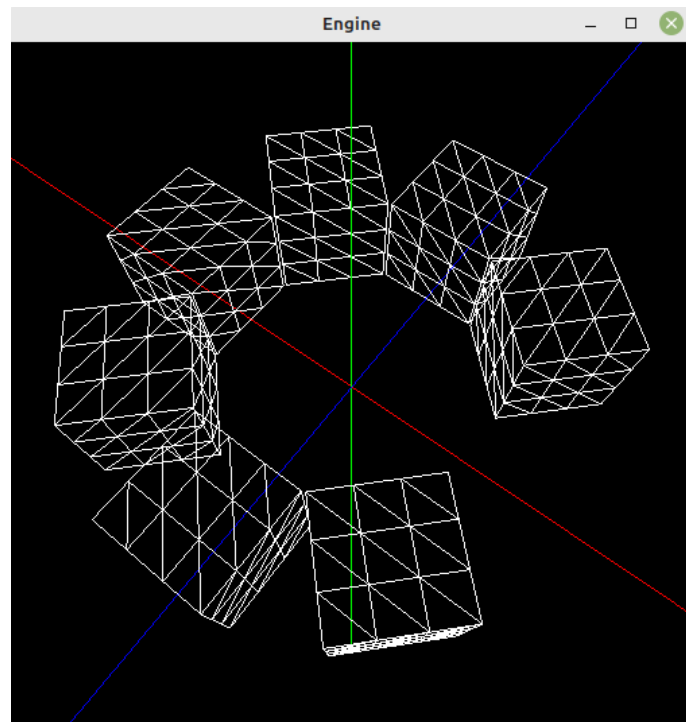


Figura 5.4.: Ficheiro test\_2\_4.xml

## 5.2. Sistema Solar

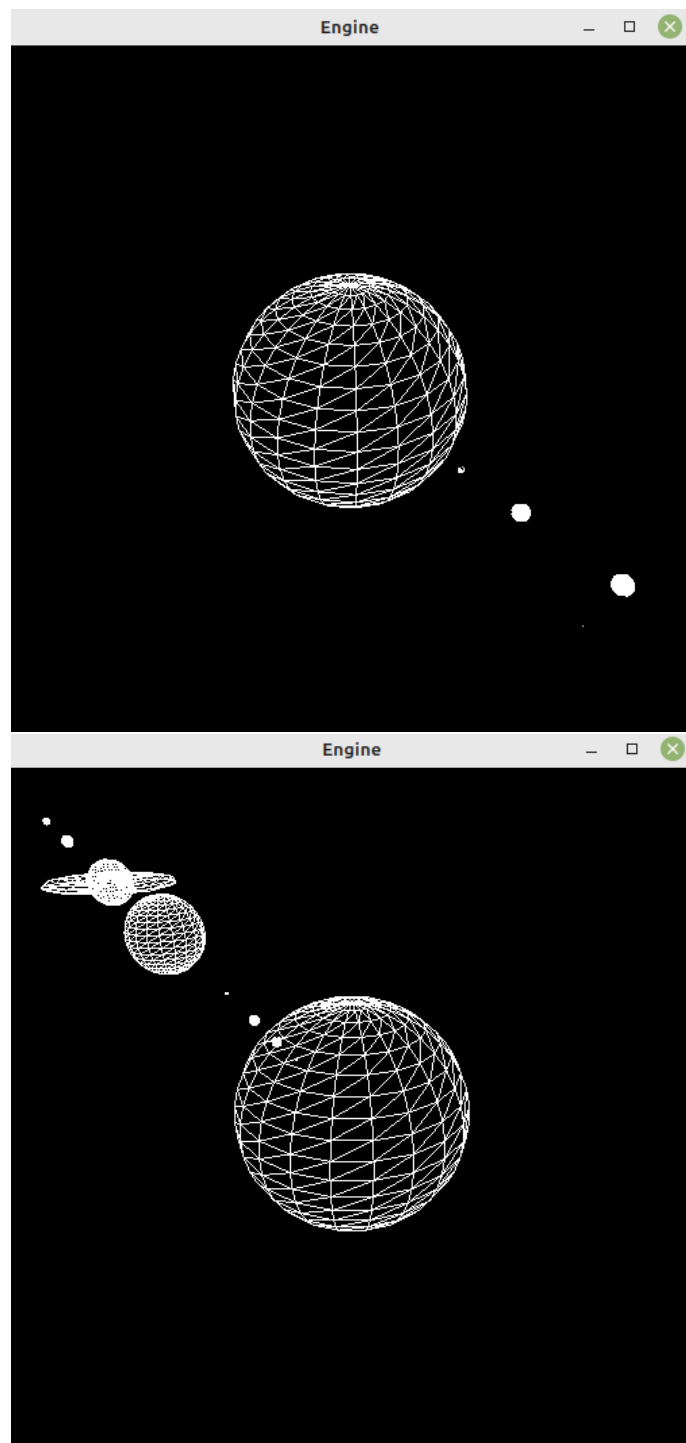


Figura 5.5.: Ficheiro solar\_system.xml

## 6. Conclusão

Da realização desta segunda fase, o grupo considera que a mesma foi bem conseguida, já que se realizaram todas as funcionalidades requisitadas pelo próprio enunciado.

No espectro positivo, consideramos conveniente destacar o correto funcionamento do nosso programa. Além disso, as estruturas implementadas estão em concordância com a estrutura do XML permitindo uma visualização mais clara daquilo que é armazenado.

Por outro lado, também existiram algumas dificuldades, tais como a familiarização com as funções do *tinyXML2* e a implementação de funções recursivas. Além disso, na representação do modelo do sistema solar o principal problema foi determinar uma escala o mais próxima da realidade e com isso descobrir as translações, escalas e rotações dos planetas e seus satélites. Apesar disso, as dificuldades foram ultrapassadas.

Para concluir, consideramos que houve um balanço positivo do trabalho realizado dado que as dificuldades sentidas foram superadas a ser cumpridos todos os requisitos.

## A. Ficheiro XML - Sistema Solar

```
<world>
  <window width="512" height="512" />
  <camera>
    <position x="30" y="30" z="30" />
    <lookAt x="0" y="0" z="0.0" />
    <up x="0" y="1" z="0.0" />
    <projection fov="60" near="1" far="1000" />
  </camera>

  <group>
    <group>
      <!--Sol-->
      <transform>
        <scale x="10" y="10" z="10" />
      </transform>
      <models>
        <model file="sphere.3d" />
      </models>
    </group>

    <group>
      <!--Mercúrio-->
      <transform>
        <translate x="12.8955" y="0.0" z="0.0" />
        <rotate angle="-0.1" x="0" y="0" z="1" />
        <scale x="0.244" y="0.244" z="0.244" />
      </transform>
      <models>
        <model file="sphere.3d" />
      </models>
    </group>

    <group>
      <!--Vénus-->
      <transform>
        <translate x="18.3055" y="0.0" z="0.0" />
        <rotate angle="-177.4" x="0" y="0" z="1" />
        <scale x="0.6052" y="0.6052" z="0.6052" />
      </transform>
```



```

    <models>
        <model file="sphere.3d" />
    </models>
</group>

<group>
    <!--Terra-->
    <transform>
        <translate x="25.7855" y="0.0" z="0.0" />
        <rotate angle="-23.5" x="0" y="0" z="1" />
        <scale x="0.6378" y="0.6378" z="0.6378" />
    </transform>
    <models>
        <model file="sphere.3d" />
    </models>
    <!-- Lua -->
    <group>
        <transform>
            <translate x="0.0" y="0.0" z="6.55174" />
            <rotate angle="18,35" x="0" y="0" z="1" />
            <scale x="0.17374" y="0.17374" z="0.17374" />
        </transform>
        <models>
            <model file="sphere.3d" />
        </models>
    </group>
</group>

<group>
    <!--Marte-->
    <transform>
        <translate x="37.1825" y="0.0" z="0.0" />
        <rotate angle="-25" x="0" y="0" z="1" />
        <scale x="0.3397" y="0.3397" z="0.3397" />
    </transform>
    <models>
        <model file="sphere.3d" />
    </models>
</group>

<group>
    <!--Júpiter-->
    <transform>
        <translate x="76.099" y="0.0" z="0.0" />
        <rotate angle="-3" x="0" y="0" z="1" />
        <scale x="7.1492" y="7.1492" z="7.1492" />
    </transform>

```

```

    <models>
      <model file="sphere.3d" />
    </models>
  </group>

  <group>
    <!--Saturno-->
    <transform>
      <translate x="147.549" y="0.0" z="0.0" />
      <rotate angle="-27" x="0" y="0" z="1" />
      <scale x="6.0268" y="6.0268" z="6.0268"/>
    </transform>
    <models>
      <model file="sphere.3d" />
    </models>
  </group>

  <group>
    <!--Saturno (ring)-->
    <transform>
      <translate x="147.549" y="0.0" z="0.0" />
      <scale x="6.0268" y="1.0268" z="6.0268"/>
      <rotate angle="90" x="1" y="0" z="0"/>
      <rotate angle="-27" x="0" y="0" z="1" />
    </transform>
    <models>
      <model file="torus.3d" />
    </models>
  </group>

  <group>
    <!--Urano-->
    <transform>
      <translate x="287.099" y="0" z="0.0" />
      <rotate angle="-98" x="0" y="0" z="1" />
      <scale x="2.5559" y="2.5559" z="2.5559"/>
    </transform>
    <models>
      <model file="sphere.3d" />
    </models>
  </group>

  <group>
    <!--Neptuno-->
    <transform>
      <translate x="450.4" y="0" z="0.0" />
      <rotate angle="-30" x="0" y="0" z="1" />
      <scale x="2.4766" y="2.4766" z="2.4766"/>

```

```
        </transform>
        <models>
            <model file="sphere.3d" />
        </models>
    </group>
</group>
</world>
```