



**Universidade do Minho**

Escola de Engenharia

Licenciatura em Engenharia Informática

Licenciatura em Engenharia Informática

## **Unidade Curricular - Computação Gráfica**

Ano Letivo de 2024/2025

### **Fase 2 - Transformações Geométricas Grupo 36**

**Tomás Henrique Alves Melo**

a104529

**Carlos Eduardo Martins de Sá Fernandes**

a100890

**Hugo Rafael Lima Pereira**

a93752

**Alexandre Marques Miranda**

a104445



Março, 2025

## Resumo

Este relatório tem como objetivo mostrar detalhadamente as alterações feitas nesta segunda fase do projeto, focando principalmente na aplicação Engine e nas novas estruturas de dados criadas no ficheiro XMLDataFormat que garantiram que a segunda fase fosse cumprida com sucesso com base nos requisitos especificados no enunciado do trabalho prático. O relatório cobre ainda a forma como o ficheiro XML que representa o sistema solar foi construído e a nova primitiva que foi necessária de desenhar para que a construção do sistema solar fosse a mais real e fidedigna possível com base no conhecimento geral que temos sobre o sol, planetas (e as suas luas).

**Área de Aplicação:** Construção de cenas hierárquicas 3D com transformações geométricas e visualização em tempo real com recurso à linguagem de programação C++ e OpenGL.

**Palavras-Chave:** Computação Gráfica, Primitivas Geométricas, Transformações Geométricas, Hierarquia de Modelos, Modelação 3D, Motor gráfico 3D, OpenGL

# Índice

<b>1. Introdução .....</b>	<b>1</b>
<b>2. Objetivos Impostos .....</b>	<b>2</b>
<b>3. Arquitetura desenvolvida .....</b>	<b>3</b>
<b>4. Generator .....</b>	<b>4</b>
4.1. buildPrimitives .....	4
4.1.1. buildSaturnRing .....	4
<b>5. Engine .....</b>	<b>6</b>
5.1. Ficheiros XML .....	6
5.2. Nova funcionalidade .....	6
5.3. Desenho das primitivas .....	6
5.3.1. Caixa com translação de 1 no eixo Y .....	7
5.3.2. Torre (caixa, cone e esfera) .....	8
5.3.3. Boneco de Neve .....	9
5.3.4. Caixas - Rotações e Translações .....	10
5.3.5. SaturnRing (com rotação aplicada no eixo do X) .....	11
5.3.6. Esfera juntamente com o anel (simulação do planeta Saturno) .....	12
5.4. Sistema Solar .....	13
5.4.1. solar_system_final.xml .....	13
5.4.2. solar_system_real_scale.xml .....	14
<b>6. Utils .....</b>	<b>15</b>
6.1. XMLDataFormat .....	15
6.1.1. Novas estruturas de dados .....	15
6.1.1.1. Transform .....	15
6.1.1.2. Group .....	15
6.1.2. buildGroup & buildTransform .....	16
6.2. Ficheiro XML para gerar Saturno recorrendo à tag 'group' .....	16
<b>7. Conclusão .....</b>	<b>18</b>
<b>Referências - APA .....</b>	<b>19</b>

## Lista de Figuras

Figura 1	Arquitetura para a Fase 2 .....	3
Figura 2	Cálculo dos pontos das slices do Ring .....	5
Figura 3	Visualizar coordenadas da câmera pelo terminal .....	6
Figura 4	test_2_1.xml .....	7
Figura 5	test_2_2.xml .....	8
Figura 6	test_2_3.xml .....	9
Figura 7	test_2_4.xml .....	10
Figura 8	SaturnRing: inner radius: 3, outer radius: 5, height: 0.1, slices: 30, stacks:30 .....	11
Figura 9	Planeta Saturno .....	12
Figura 10	Sistema solar (estético) - Respeito pelas distâncias e tamanhos .....	13
Figura 11	Zoom-in para visualização da lua da Terra e luas de Marte .....	14
Figura 12	Sistema solar (escala real dos objetos) - Escala Real .....	14

# 1. Introdução

No contexto da unidade curricular de Computação Gráfica, foi desenvolvida a segunda etapa do trabalho prático proposto. Esta fase consiste em criar cenas hierárquicas recorrendo a transformações geométricas. Para tal, os ficheiros XML ganham um novo formato e é necessário adaptar a aplicação Engine de modo a atender aos novos requisitos impostos nesta segunda fase do trabalho prático, para além do desenvolvimento de uma nova primitiva e criação de novas estruturas de dados.

## **2. Objetivos Impostos**

Para esta fase do trabalho prático, é proposto criar cenas hierárquicas com recurso a transformações geométricas. Desta forma, uma cena será definida como uma árvore em que cada nó contém um conjunto de transformações geométricas, quer sejam estas rotações, translações ou escalas e, opcionalmente, um conjunto de modelos. Para além disso cada nó pode ter vários nós-filho, tal como especificado no enunciado do trabalho prático.

### 3. Arquitetura desenvolvida

A arquitetura do foi mantida. As alterações apenas precisaram de ser feitas nos programas já elaborados na primeira fase, adicionando a estes as novas funcionalidades exigidas na segunda fase.



Figura 1: Arquitetura para a Fase 2

## 4. Generator

Esta aplicação tem como função principal criar os vértices e índices que servirão de base para a construção dos triângulos das diversas primitivas geométricas, com recurso a algoritmos específicos para cada tipo de primitiva, tal como descrito no relatório referente à fase 1 do trabalho prático. Nesta fase, surgiu a necessidade de desenvolver uma nova primitiva que representa o anel do planeta Saturno.

### 4.1. buildPrimitives

No programa responsável pela construção das funções que representam os cálculos necessários para o desenho correto das primitivas, adicionámos uma nova primitiva representante do anel de Saturno.

Esta primitiva teve a necessidade de ser criada no momento em que estávamos a desenvolver o ficheiro que representa o sistema solar. De modo a representarmos graficamente da forma mais fidedigna possível, com base nos conhecimentos adquiridos até ao momento, achamos necessário a construção da primitiva do anel, dado que ajudaria na representação correta do sistema solar, não sendo algo complexo de implementar com base nos conhecimentos que já obtivemos até ao momento.

#### 4.1.1. buildSaturnRing

A função `buildSaturnRing` é responsável por criar um anel tridimensional (representando o anel de Saturno) utilizando coordenadas polares, slices e stacks. O anel gerado corresponde a uma superfície formada por duas faces (superior e inferior) e faces laterais que conectam as camadas de pontos.

O anel é composto por camadas horizontais (stacks) em torno de um círculo. Cada camada é composta por pontos tanto na face superior como na face inferior, sendo gerado um ponto para cada slice de cada camada. A função `addUniquePoint` é usada para garantir que pontos duplicados não sejam adicionados, otimizando assim o uso de memória.

Os pontos são calculados para as faces superior e inferior do anel, considerando a altura do anel e distribuindo os pontos ao longo de um círculo. Para cada fatia, são calculadas as coordenadas  $x$  e  $z$  de cada ponto usando a fórmula polar ( $x = r * \cos(\theta)$  e  $z = r * \sin(\theta)$ ), enquanto a coordenada  $y$  é ajustada para a posição superior ou inferior.



$$\Delta\theta = 360^\circ / \text{slices}$$

$$\theta_i = i \times \Delta\theta$$

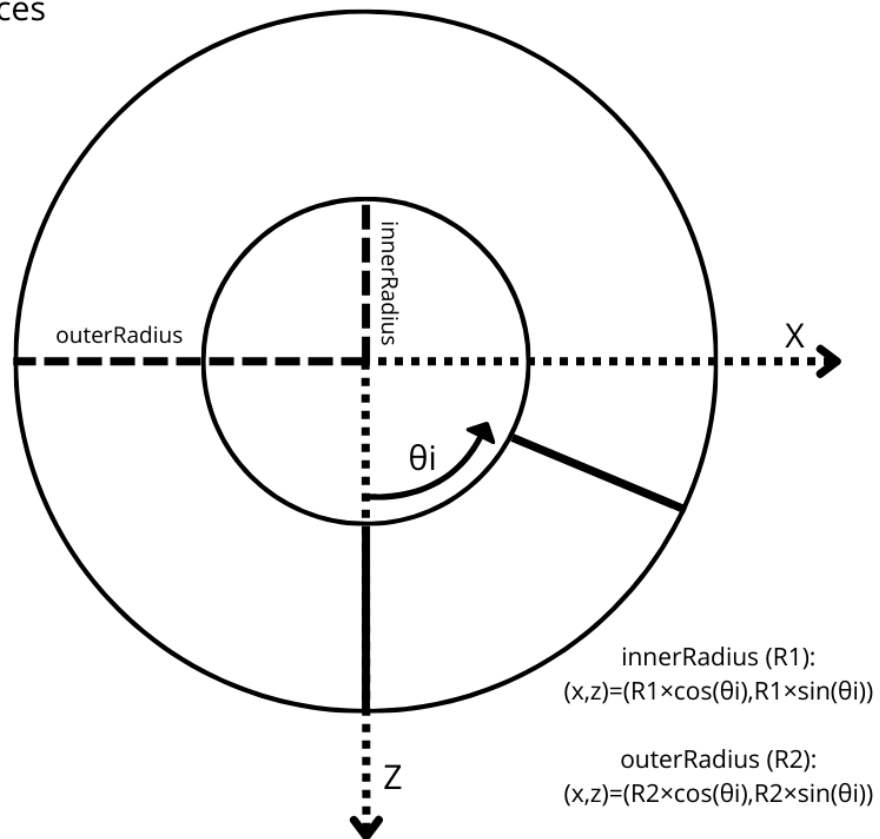


Figura 2: Cálculo dos pontos das slices do Ring

A função de seguida gera os índices que formam os triângulos que representam a superfície do anel. Os triângulos são formados tanto para as faces superior e inferior (top-facing e bottom-facing) quanto para as faces laterais do anel, ligando as camadas superiores e inferiores.

Após gerar todos os pontos e índices, a função adiciona os pontos ao objeto ring e define os índices para a renderização.

Por fim, a função retorna o objeto ring, que contém todos os dados necessários para renderizar o anel de Saturno, incluindo os pontos e a ordem correta dos índices para formar os triângulos.

## 5. Engine

A maior parte das modificações solicitadas na fase 2 serão implementadas no Engine, para que este consiga criar cenas que utilizem transformações geométricas dispostas de forma hierárquica.

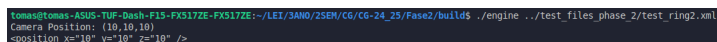
### 5.1. Ficheiros XML

Na segunda fase, o formato do ficheiro XML foi alterado, expandido-se, de modo a incluir as transformações geométricas aplicadas aos grupos e à hierarquia dos modelos. Em vez de apenas especificar os modelos a serem carregados, como na primeira fase, agora o ficheiro XML também inclui informações sobre transformações como translação, rotação e escala, além de suportar a estrutura hierárquica dos objetos. Este tema é aprofundado na Secção 6 deste relatório.

### 5.2. Nova funcionalidade

Ao construir o ficheiro XML que representa o sistema solar, surgiu a necessidade de implementarmos uma nova funcionalidade que fosse capaz de indicar as coordenadas exatas de onde a câmara está posicionada. Para tal, adicionamos juntamente às funcionalidades de zoom-out, omissão de eixos, entre outras, a funcionalidade de visualizar as coordenadas da posição da câmara quando o utilizador assim o desejar, ao premir a tecla 'C' do teclado.

C - Mostrar as coordenadas da posição da câmara no terminal

A terminal window with a dark background. The prompt is 'tomas@tomas-ASUS-TUF-Dash-F15-FX517ZE:~/LEI/3ANO/2SEM/CG/CG-24\_25/Fase2/build\$'. The command executed is './engine ../test\_files/phase\_2/test\_ring2.xml'. The output is 'Camera Position: (10,10,10)' followed by an XML snippet: '<position x="10" y="10" z="10" />'.

```
tomas@tomas-ASUS-TUF-Dash-F15-FX517ZE:~/LEI/3ANO/2SEM/CG/CG-24_25/Fase2/build$ ./engine ../test_files/phase_2/test_ring2.xml
Camera Position: (10,10,10)
<position x="10" y="10" z="10" />
```

Figura 3: Visualizar coordenadas da câmara pelo terminal

### 5.3. Desenho das primitivas

Após concluído o parse do ficheiro XML, as primitivas estão organizadas em groups. A função `renderGroup` percorre a estrutura de grupos e aplica as transformações definidas, através da `applyTransform` antes de chamar a `drawPrimitives` para desenhar os modelos associados. Se um grupo possuir subgrupos, a função `renderGroup` processa-os recursivamente. As transformações aplicadas na `applyTransform` incluem rotação, translação e escala, modificando a matriz de transformação do OpenGL. A hierarquia permite que objetos dependam de outros, por exemplo, um carro e as respetivas rodas, onde o carro é o grupo principal e cada roda é um subgrupo dentro dele.

### 5.3.1. Caixa com translação de 1 no eixo Y

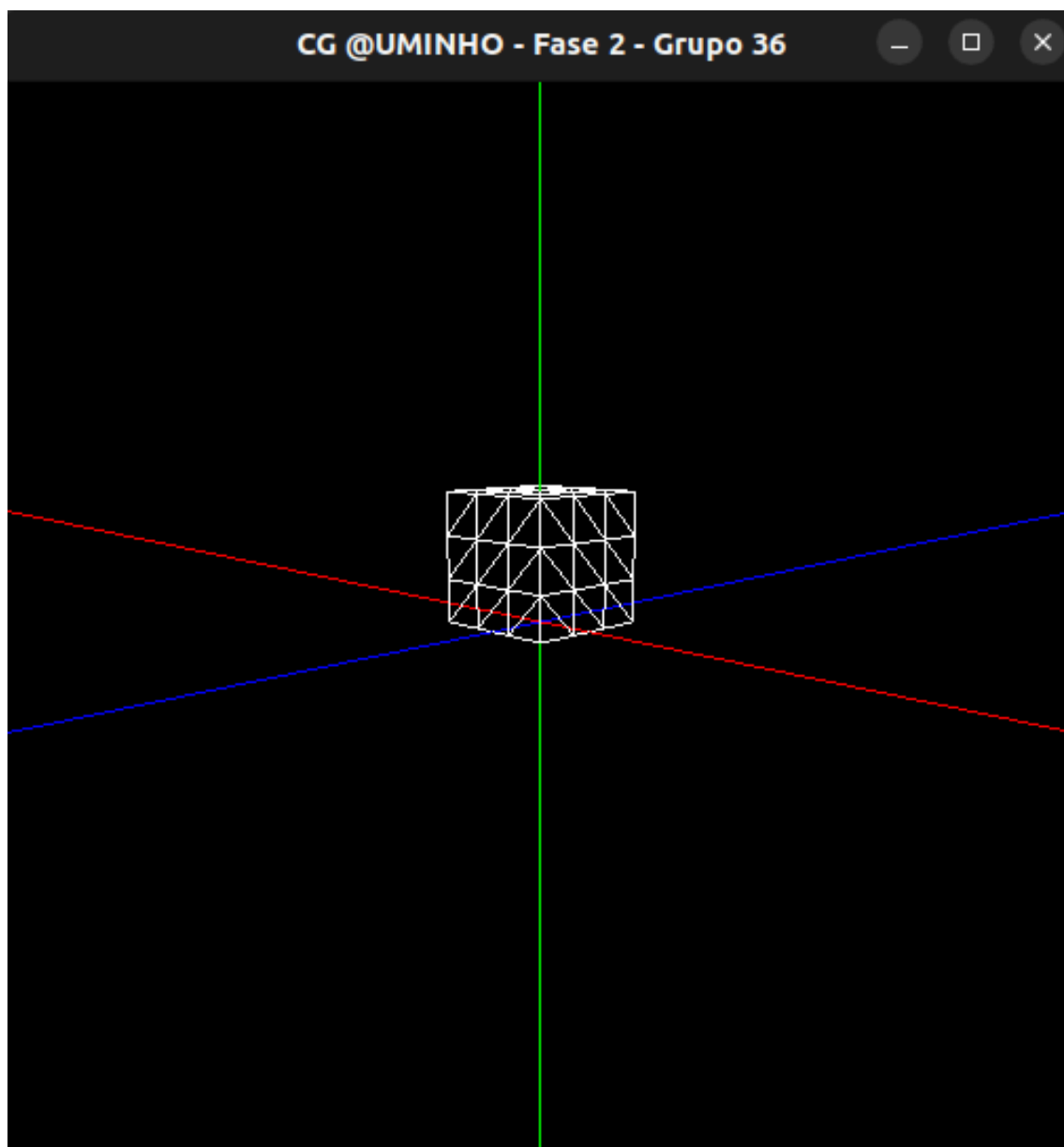


Figura 4: test\_2\_1.xml

### 5.3.2. Torre (caixa, cone e esfera)

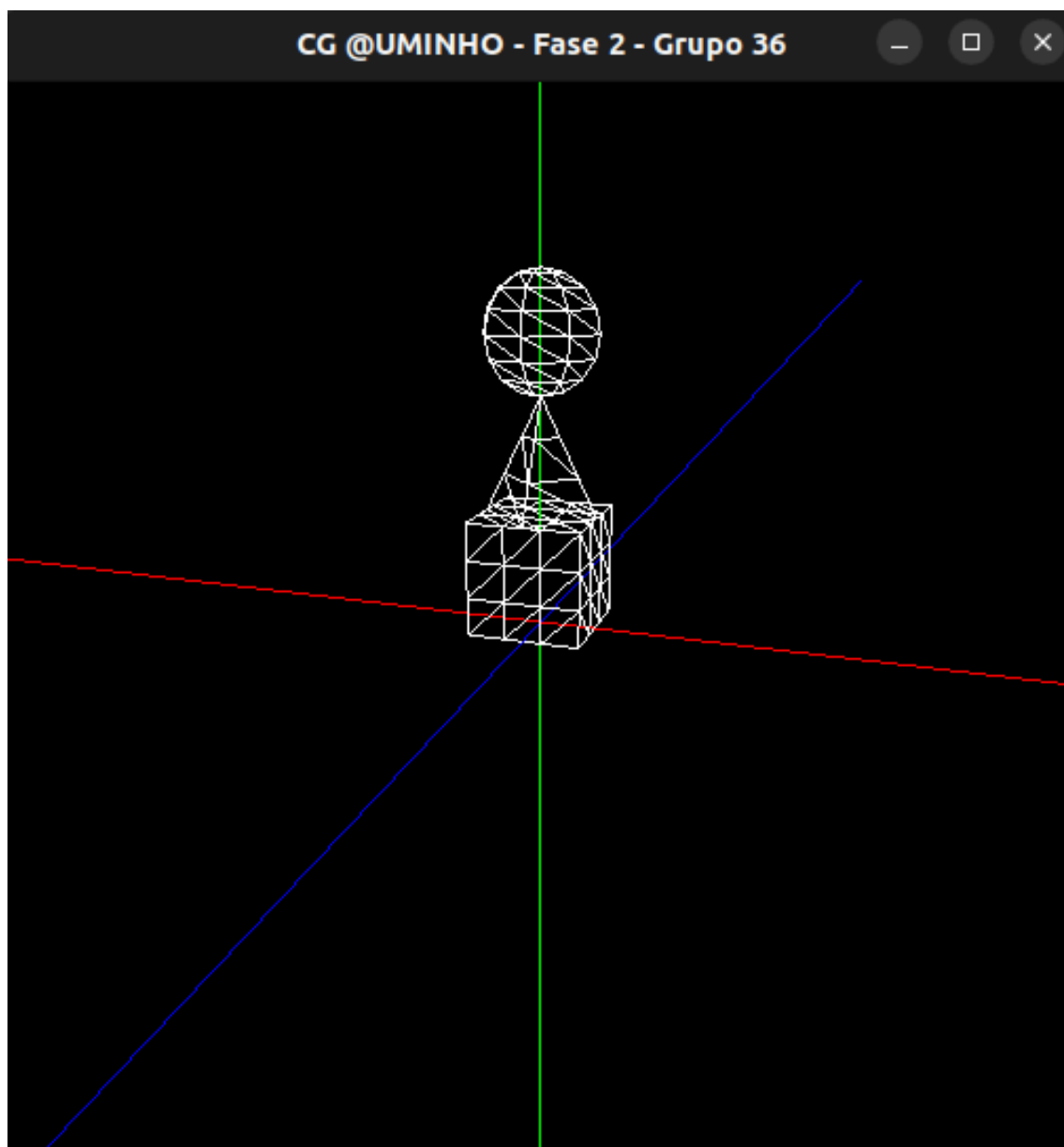


Figura 5: test\_2\_2.xml

### 5.3.3. Boneco de Neve

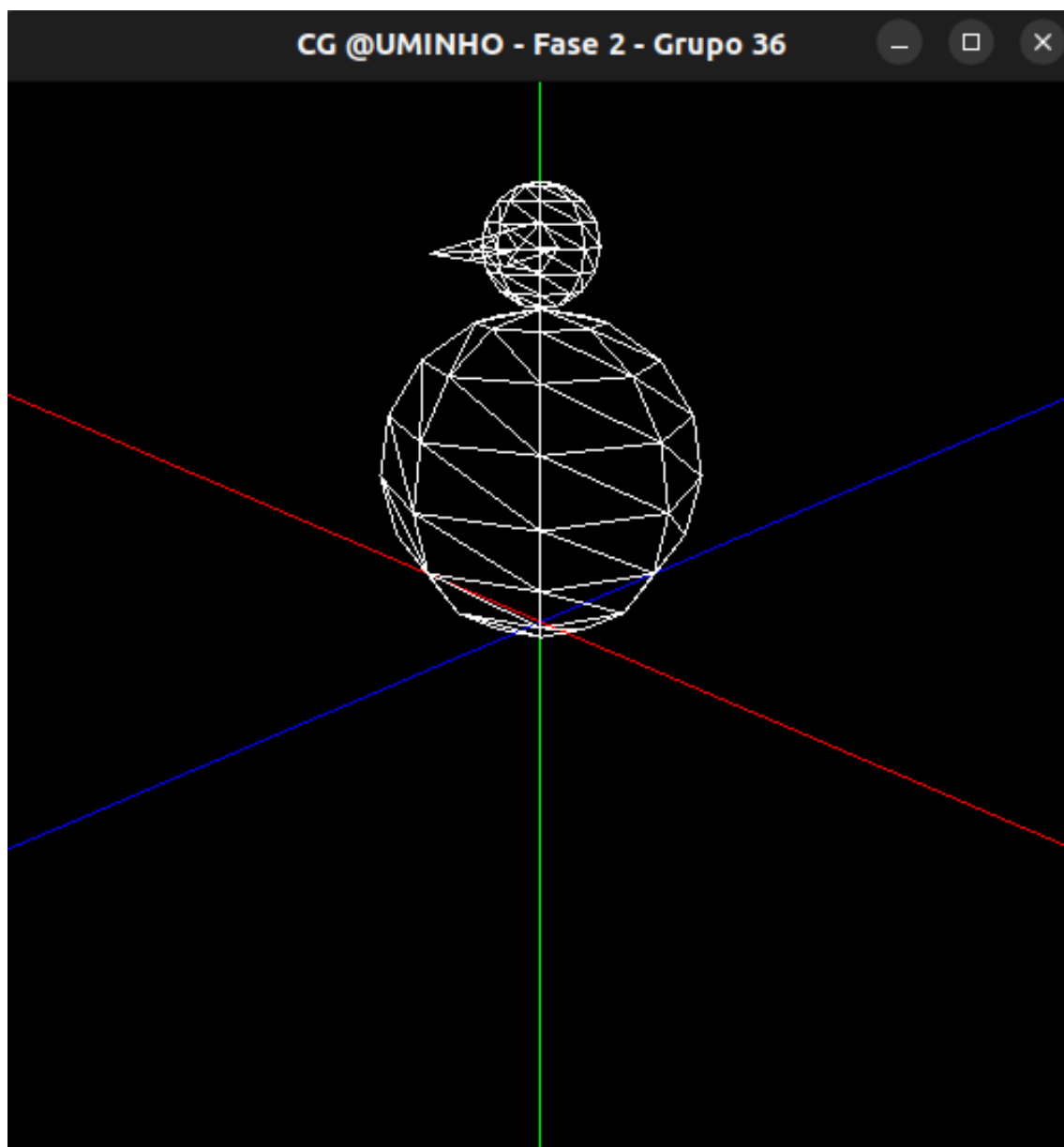


Figura 6: test\_2\_3.xml

#### 5.3.4. Caixas - Rotações e Translações

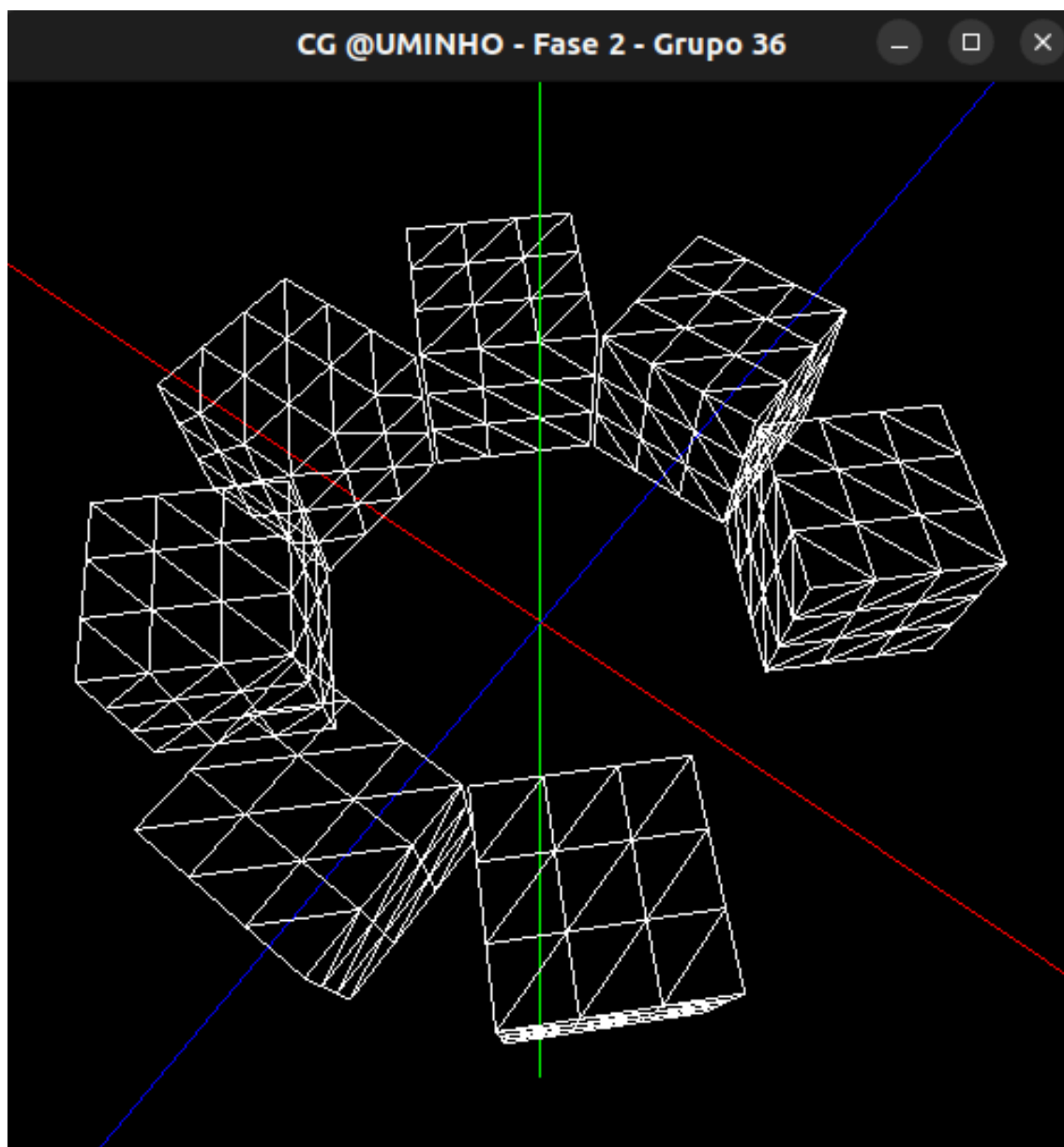


Figura 7: test\_2\_4.xml

Representação do anel e de seguida a junção da esfera e anel, de modo a simular o planeta Saturno.

### 5.3.5. SaturnRing (com rotação aplicada no eixo do X)

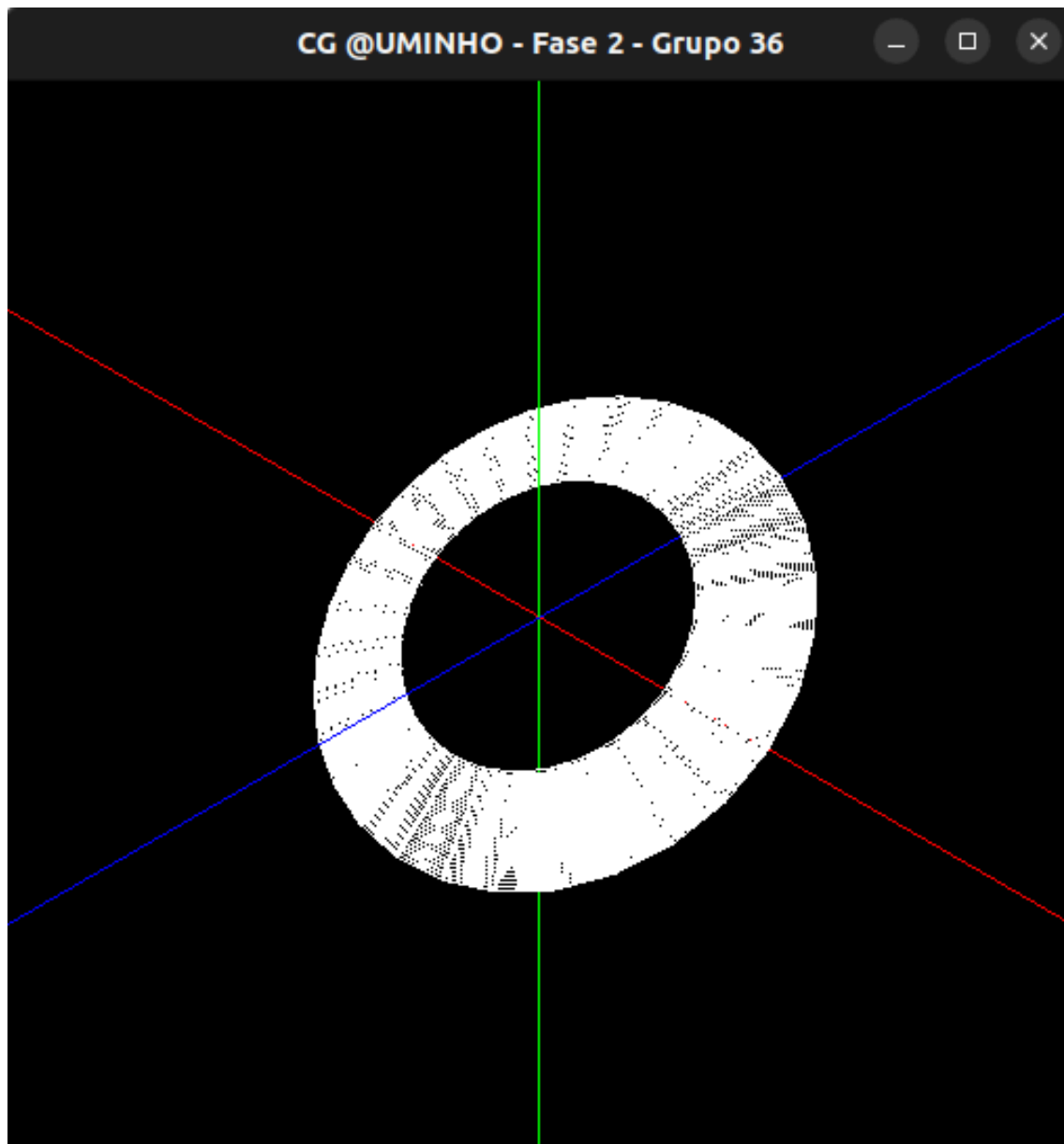


Figura 8: SaturnRing: inner radius: 3, outer radius: 5, height: 0.1, slices: 30, stacks:30

5.3.6. Esfera juntamente com o anel (simulação do planeta Saturno)

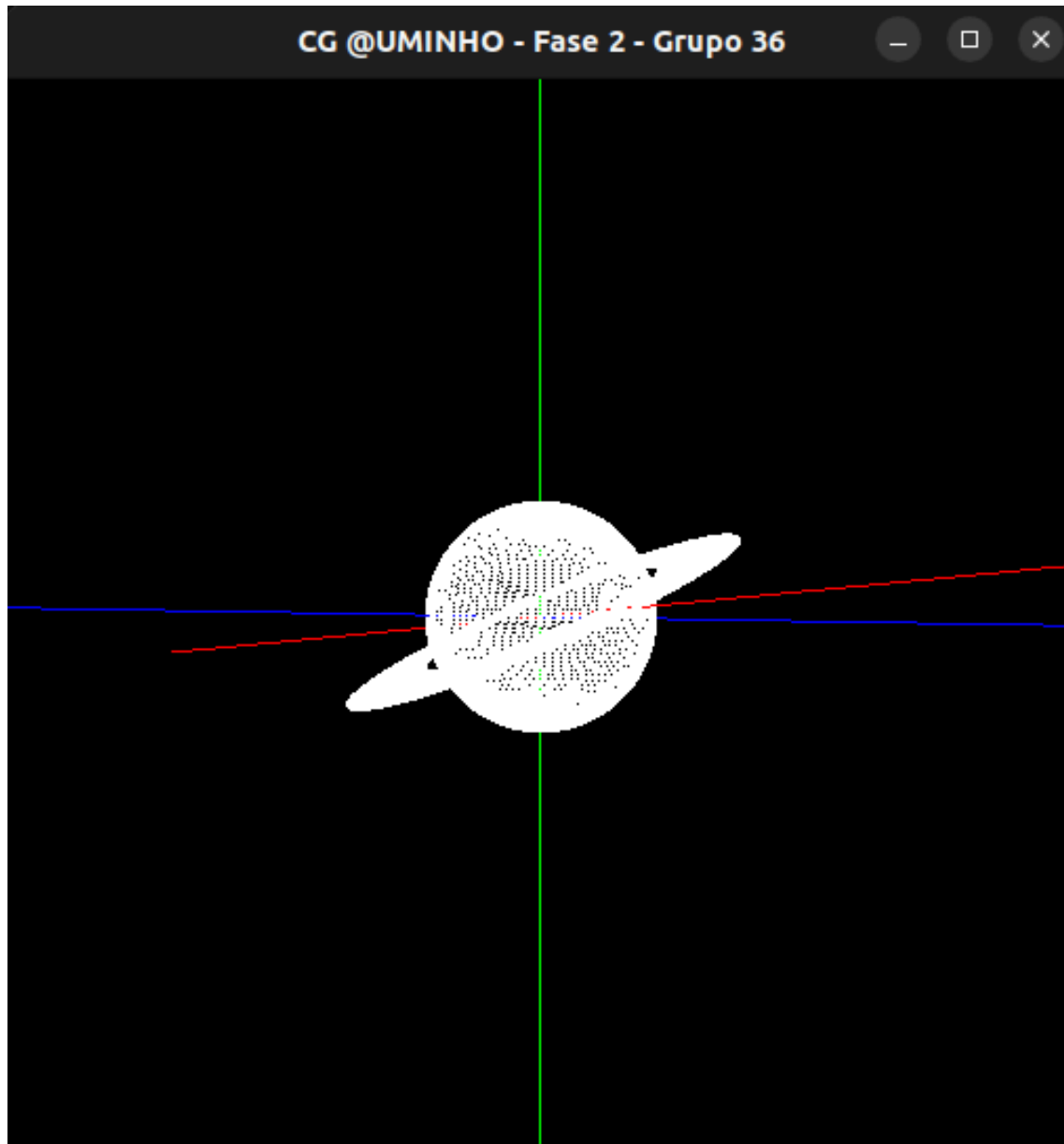


Figura 9: Planeta Saturno



## 5.4. Sistema Solar

O processo para o desenvolvimento do sistema solar foi trabalhoso, porém monótono, sendo a sua construção difícil numa fase inicial, porém rapidamente o seu desenvolvimento tornou-se num processo bastante intuitivo.

Para a elaboração deste ficheiro, foi construído um grupo para cada planeta e, no caso de o planeta possuir luas, foram criados nós-filho de modo a que estas pudessem adotar as características do grupo pai, adotando assim a rotação correspondente, sendo apenas preciso ajustar a translação e escala.

Para não adotarmos o formato linear do sistema solar e para trazermos mais realismo ao desenho, recorremos a rotações para todos os planetas, rodando assim o eixo de coordenadas para a posição desenhada em cada planeta, bastando apenas após isso gerir a distância do objeto ao sol e regular a sua escala.

Para o desenvolvimento do sistema solar, foram criadas várias versões (diferentes tentativas), chegando como resultado final ao ficheiro `solar_system_final.xml`. De seguida, ajustamos a partir deste ficheiro as distâncias entre objetos e tamanho dos objetos para trazer mais realismo ao sistema solar, desenvolvendo o `solar_system_real_scale.xml`. Apenas a dimensão real do sol não foi tida em conta para o desenvolvimento deste último ficheiro por razões estéticas, uma vez que este possui um tamanho absurdamente maior quando comparado com planetas grandes como Júpiter (sendo 10x maior do que este), porém ainda assim continua a ser o maior objeto do sistema solar para manter a proporção de tamanho, mantendo um pouco a realidade.

### 5.4.1. `solar_system_final.xml`

Representação do sistema solar.

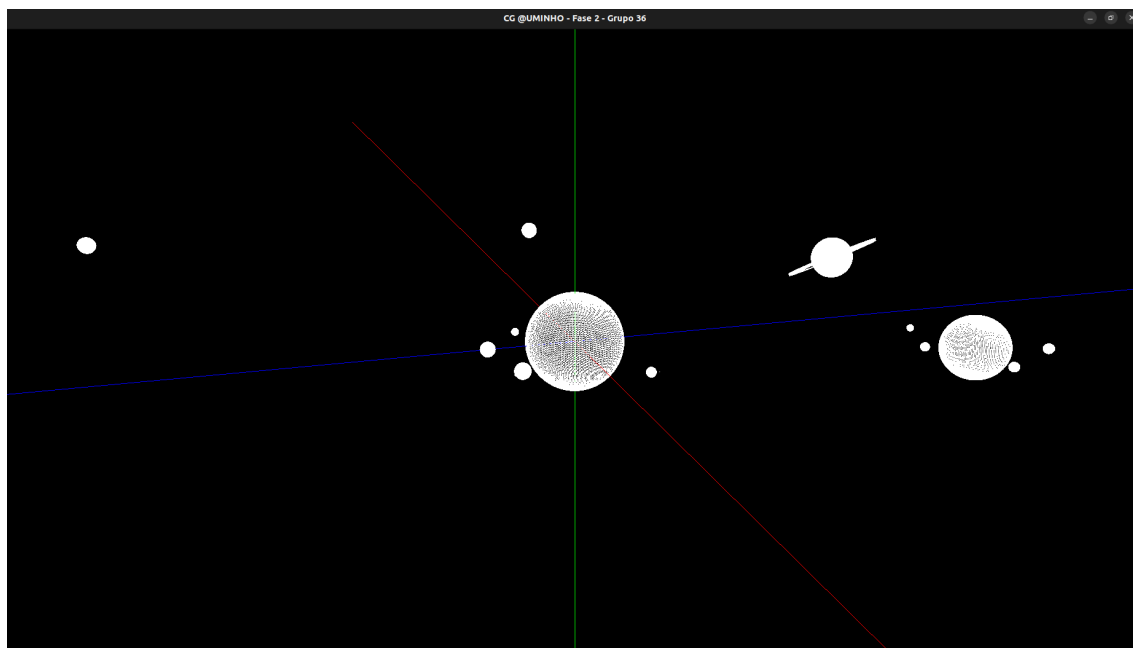


Figura 10: Sistema solar (estético) - Respeito pelas distâncias e tamanhos

Na imagem acima é representado o Sol, no centro, de seguida Mercúrio, Vénus, Terra (juntamente com a lua da Terra) e Marte (juntamente com as luas Deimos e Phobos), muito próximos do Sol. Um pouco mais distante, encontra-se Júpiter rodeado pelas 4 principais luas (Io, Europa, Ganimedes e Calisto), Saturno com o grandioso anel um pouco mais distante, e por fim, Urano e Neptuno.

Abaixo encontra-se uma representação do sistema solar com maior aproximação para visualização da lua da Terra e as luas de Marte que também foram implementadas.

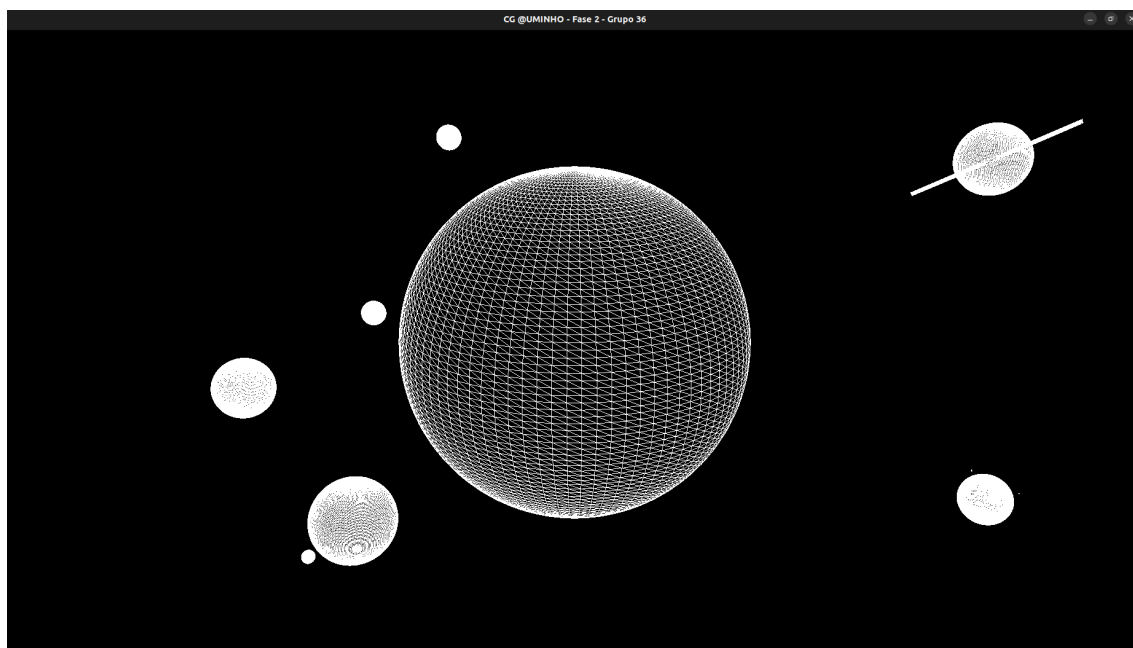


Figura 11: Zoom-in para visualização da lua da Terra e luas de Marte

#### 5.4.2. solar\_system\_real\_scale.xml

Representação do sistema solar à escala real.

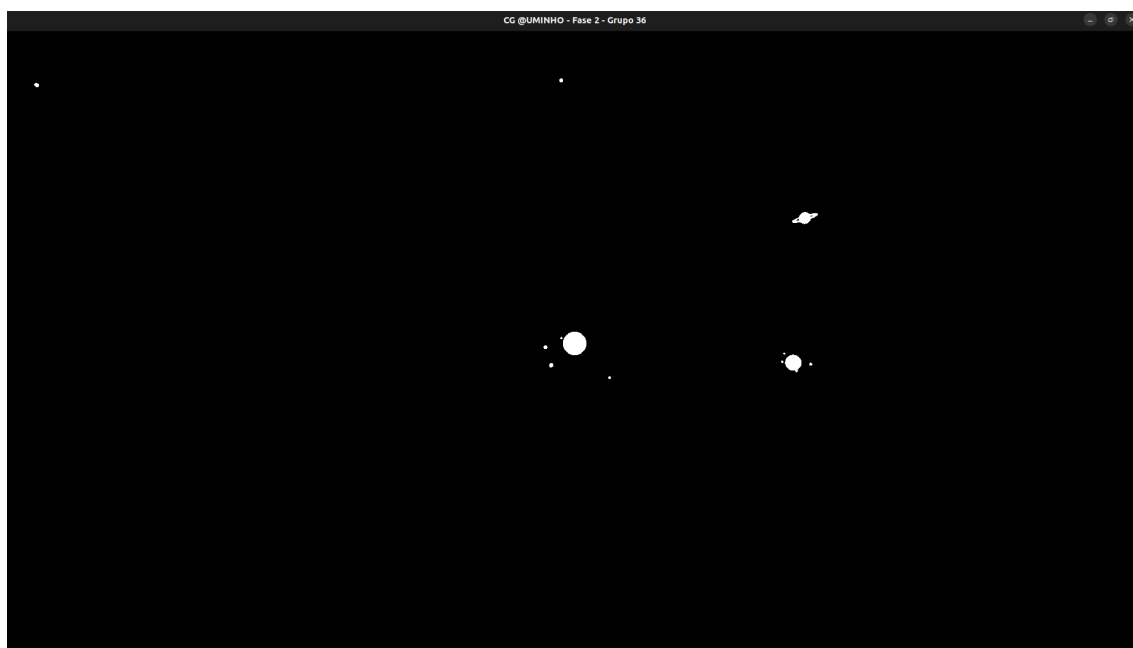


Figura 12: Sistema solar (escala real dos objetos) - Escala Real

Ainda foi desenvolvida uma versão do sistema solar adaptada aos tamanhos e distâncias reais dos objetos, tendo esta implementação como objetivo a representação real e fiável do sistema solar, tendo sido desenvolvida com especial atenção, como referido, à distância entre os objetos e distância entre os objetos, quer seja com as luas, planetas ou Sol. Tal como no ficheiro desenvolvido acima, não foi tido em conta o real tamanho do sol por motivos estéticos e para mais fácil visualização com dimensões razoáveis de todos os outros objetos numa mesma imagem.

## 6. Utils

Para alcançar o objetivo de representar cenas que utilizam transformações geométricas dispostas hierarquicamente, foi preciso desenvolver novas estruturas de dados que armazenem toda a informação requerida pelo ficheiro XML, possibilitando assim a geração das cenas conforme especificado. Para tal, adicionámos novas estruturas de dados no XMLDataFormat, tais como a struct Transform que armazena os tipos de transformações geométricas (rotação, translação e escala) e ainda uma struct Group que representa um nó na hierarquia da cena, contendo uma transformação geométrica, uma lista de modelos associados ao grupo e uma lista de subgrupos (children).

### 6.1. XMLDataFormat

Este ficheiro contém funções de leitura para interpretar o XML e preencher a estrutura XMLDataFormat, com a ajuda de funções auxiliares, como a buildTransform, para processar os diferentes elementos do XML, garantindo que as informações necessárias são extraídas corretamente. A cena é estruturada como uma árvore, onde cada grupo pode herdar transformações geométricas do grupo pai. Isto permite representar hierarquias como um sistema solar, onde os planetas estão posicionados ao redor do Sol e as luas em torno dos planetas. As transformações são aplicadas sequencialmente, respeitando a ordem especificada no XML.

#### 6.1.1. Novas estruturas de dados

Criámos duas structs para esta fase, sendo estas a Transform e a Group.

##### 6.1.1.1. Transform

```
struct Transform {  
    float translate[3];  
    float rotate[4];  
    float scale[3];  
};
```

Contém um vetor associado à translação, outro à rotação e por fim, outro à escala, foi criada para facilitar as manipulações para translações, rotações e escalas que surgiram como novas funcionalidades nesta fase.

##### 6.1.1.2. Group

```
struct Group {  
    Transform transform;  
    std::list<std::string> models;  
    std::list<Group*> children;  
};
```

Dado que o parsing dos ficheiros XML é feito com base na tag 'group', a struct Group armazena os valores de transformação na struct Transform e os modelos na lista de modelos e, ao aplicar esta mesma lógica recursivamente para os grupos filho, garantimos que toda a exploração dos grupos filho é feita até ao final respeitando a hierarquia de prioridades entre grupos, fazendo com que os grupos filho adotem as transformações dos grupos pai.

### 6.1.2. buildGroup & buildTransform

A função `buildTransform` é responsável por extrair e armazenar as transformações aplicadas a um grupo a partir do ficheiro XML. Recebe como parâmetros um apontador para o elemento `transform` e um objeto do tipo `Transform`, onde os valores extraídos são armazenados. A função verifica a existência de três tipos de transformações: translação, rotação e escala. Para cada uma delas, os valores dos atributos `x`, `y` e `z` são lidos e guardados na struct correspondente à transformação. No caso da rotação, além das coordenadas do eixo de rotação, também é lido o ângulo da transformação tal como descrito no XML. Estes dados são posteriormente utilizados para aplicar as transformações na fase de renderização.

Já a função `buildGroup` é responsável por processar os grupos do ficheiro XML e construir a hierarquia da cena. Recebe um apontador para um elemento `group` e um objeto do tipo `Group`, onde toda a informação extraída do XML será armazenada. O primeiro passo da função é verificar se o grupo possui um elemento `transform`. Caso exista, a `buildTransform` é chamada para preencher o objeto `Transform` do grupo com as devidas transformações. Em seguida, a função analisa o conteúdo do grupo para verificar a existência de modelos 3D, armazenados dentro do elemento `models`. Se forem encontrados elementos `model`, os seus atributos `file` são lidos e guardados na lista `models` do grupo, garantindo que cada grupo saiba quais modelos precisa carregar.

Por fim, a `buildGroup` processa também os subgrupos (grupos filho). Cada `group` pode conter outros `group` dentro de si, formando uma estrutura de árvore onde cada nó pode ter múltiplos filhos. Para cada subgrupo encontrado, a função adota uma chamada recursiva, garantindo que toda a estrutura hierárquica seja corretamente construída. Cada subgrupo criado é armazenado na lista `children` do grupo pai, criando um sistema de dependência entre os diferentes grupos da cena.

A implementação desta hierarquia permite que as transformações sejam corretamente aplicadas aos diferentes níveis da cena.

Assim, quando um grupo é desenhado, a sua matriz de transformação é aplicada primeiro, e as transformações dos seus filhos são multiplicadas por essa matriz antes de serem desenhadas, garantindo assim que os objetos sejam corretamente posicionados e escalados, respeitando a organização definida no ficheiro XML, garantindo a hierarquia especificada no enunciado do trabalho prático.

## 6.2. Ficheiro XML para gerar Saturno recorrendo à tag 'group'

```
<world>
  <window width="512" height="512" />
  <camera>
    <position x="10" y="10" z="20" />
    <lookAt x="0" y="0" z="0" />
    <up x="0" y="1" z="0" />
    <projection fov="60" near="1" far="1000" />
  </camera>

  <group>
    <!-- Saturno -->
    <models>
      <model file="sphere.3d" /> <!-- generator sphere 3 60 60 sphere.3d -->
    </models>

    <!-- Anel -->
    <group>
      <transform>
        <scale x="1.1" y="0" z="1.1" />
        <rotate angle="25" x="1" y="0" z="0" />
      </transform>

      <models>
        <model file="ring.3d" /> <!-- generator ring 3.2 5.1 0.1 40 40 ring.3d -->
      </models>
    </group>
  </group>
</world>
```

```
-->
    </models>

    </group>
</group>

</world>
```

## 7. Conclusão

Durante o desenvolvimento desta segunda fase do projeto, conseguimos consolidar os conhecimentos sobre transformações geométricas e a manipulação de matrizes de transformação.

Estamos satisfeitos com o trabalho realizado, pois conseguimos implementar todas as funcionalidades exigidas para esta etapa, além de algumas adicionais, como a criação de um anel (Ring) — uma primitiva que adicionámos para representar Saturno de forma mais realista no modelo do Sistema Solar. Acrescentámos também a funcionalidade de visualizar as coordenadas da câmara, motivada pela necessidade de posicionar a câmara a gosto no momento de escrita da posição da câmara no ficheiro XML.

Em suma, acreditamos que esta fase nos permitiu reunir os conhecimentos e bases necessárias para avançarmos com confiança para a terceira fase do trabalho prático.

## Referências - APA

Solar System Sizes - NASA Science. (s.d.). NASA Science. <https://science.nasa.gov/resource/solar-system-sizes/>

Planet Sizes and Locations in Our Solar System - NASA Science. (s.d.). NASA Science. <https://science.nasa.gov/solar-system/planets/planet-sizes-and-locations-in-our-solar-system/>

The Scale of the Solar System. (s.d.). Las Cumbres Observatory. <https://lco.global/spacebook/solar-system/scale-solar-system/>