Relatório Segunda Fase Trabalho Prático - Computação Gráfica

Grupo 34 :
António Luís de Macedo Fernandes (a93312)
José Diogo Martins Vieira (a93251)
João Silva Torres (a93231)
Ricardo Lopes Santos Silva (a93195)

13 de Março 2022

Contents

1 Introdução

Esta segunda fase do projeto da cadeira de Computação Gráfica, consistiu em efetuar alterações no trabalho feito na fase 1, de forma a adicionar transformações geométricas, tais como a translação, a escala e a rotação. Para realizar esta etapa, foi necessário considerar que os ficheiros XML possuem uma estrutura hierárquica em árvore, na qual cada nodo possui o ficheiro 3d onde estão os vértices, e transformações que são obrigatoriamente aplicadas aos nodos filhos.

2 Torus

Mudanças feitas no generator. Será usado para o sistema solar, nomeadamente para o anel de Saturno. Para tal, colocamos dois torus com um pequeno espaço entre eles para tornar mais realista a ilustração do anel de Saturno. Passamos então a explicar como desenvolvemos o mesmo.

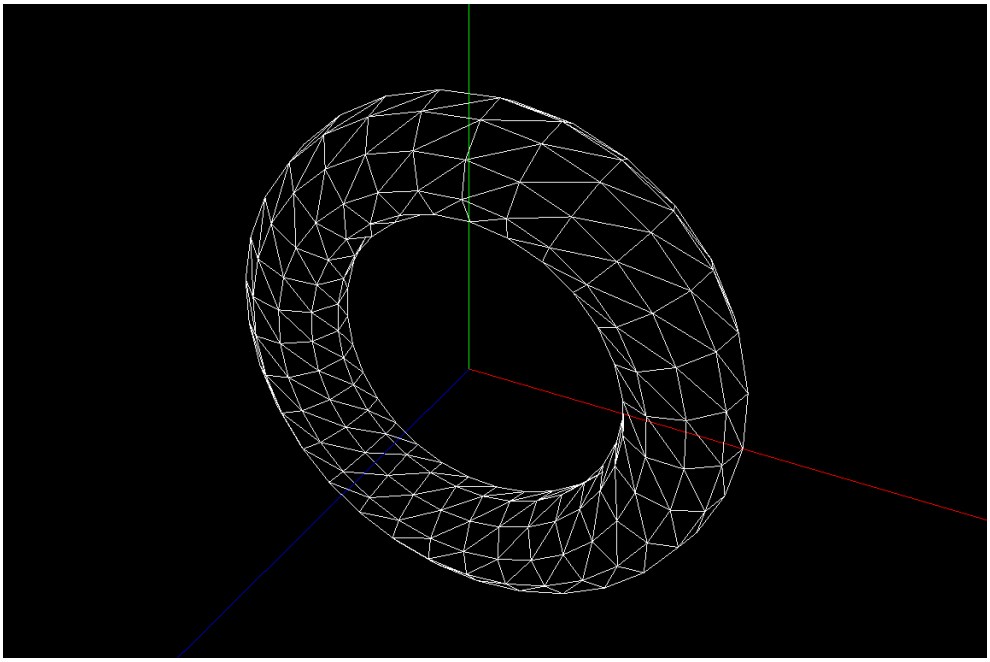

Figure 1: Torus

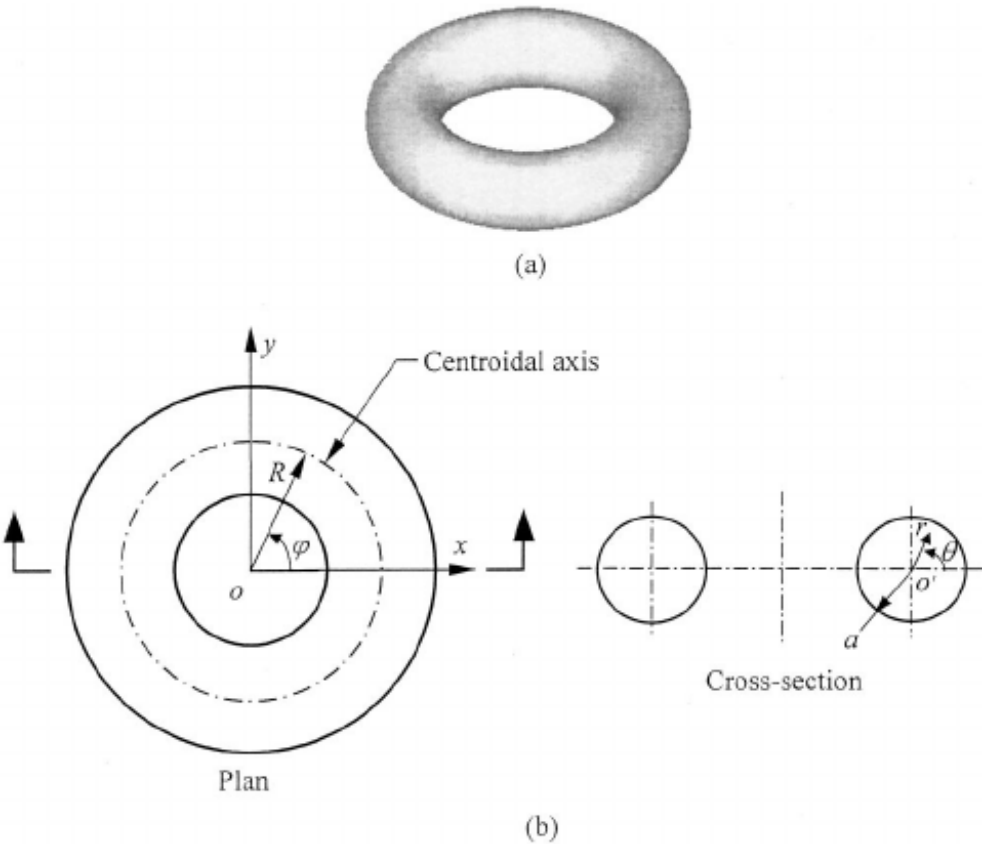

Figure 2: Vista tridimensional do Torus

Para representar o Torus consideramos:

```
x = (r_interno + r_externo * cos(phi)) * cos(theta);
y = (r_interno + r_externo * cos(phi)) * sin(theta);
z = r_externo * sin(phi);
```

2 ciclos: interior percorre slices e o exterior percorre stacks.
phi é alterado consoante as stacks.
theta é alterado consoante as slices.

3 Transformações Geométricas

Nesta fase, tivemos de implementar as transformações geométricas no engine.
Essas transformações são:

- Translate
- Rotate
- Scale

Para isso, tivemos de atualizar o parser do XML. Mas antes disso, fizemos uma reestruturação do projeto para facilitar a introdução destas.

4 Reestruturação projeto

Anteriormente tínhamos uma estrutura muito simples que apenas guardava os pontos de cada modelo. Achamos que é uma estrutura demasiado simples para guardar toda a informação que precisamos.

Para isso, criamos classes para uma mais eficiente e melhor organização.

Explicaremos de seguida as classes criadas.

4.1 Ponto

Esta classe é bastante simples, guarda apenas as coordenadas (x,y,z) de um ponto.

4.2 Model

Esta classe tem como função guardar a informação relativa a um model (ficheiro que contém pontos). Para isso, contém um vector de Ponto, que guarda todos os pontos necessários para de desenhar o modelo. Estes encontram-se pela ordem que devem ser desenhados. Contém também o nome do ficheiro ao qual vai buscar a informação dos pontos.

4.3 Transform

Classe pai das transformações. Contém as três variáveis comuns a todas as transformações(x,y,z). Possui também um método virtual `doAction`, que terá de ser implementado por cada transformação que corresponderá a execução da transformação através do respetivo comando do OpenGL.

As sublasses desta são:

- Translate
- Scale
- Rotate - contém uma variável para guardar o ângulo

4.4 Group

Classe com função de guardar toda a informação relativamente a um grupo.

Assim, vai ter:

- Um vector parar guardar os models.
- Um vector para guardar as transformações.
- Um vector para guardar os grupos que serão subgrupos deste.
- Uma variável da classe Cor.

4.5 Cor

Classe simple que irá guardar os valores (R,G,B).

5 Parsing ficheiros XML

Para o parsing do XML, tivemos de adicionar o parsing do transform. Para tal, vemos os elementos dentro do transform. Como a ordem aqui é importante, percorremos os valores pela ordem que aparecem no ficheiro, e depois verificamos de que tipo de transformação se trata, criamos a classe correspondente e adicionamos ao grupo.

Foi preciso adicionar também a existência de subgrupos. Para isso, no fim de fazer o parsing do grupo principal, vamos verificar se existe subgrupos, se existirem, fazemos de modo recursivo o parsing desse grupo, adicionando ao grupo a que pertence e assim sucessivamente.

Com a implementação das transformações geométricas, falta agora fazer a demo scene que se trata do sistema solar estático.

6 Sistema Solar

Para o sistema solar começamos por fazer o sol, os planetas e as luas, seguidamente a cintura de asteróides, e por fim, as estrelas.

6.1 Sol, Planetas e Luas

Começamos por definir o tamanho para o sol, para efeitos de facilidade de visualização de todos os planetas, decidimos não fazer o sol e as distâncias dos planetas à sua escala verdadeira.

Para a referência das escalas dos tamanhos dos planetas relativamente à terra, usamos o site (https://www.education.com/scie
fair/article/scale-model-planets-solar-system/)

É usado o torus para os anéis de Saturno.

Definimos também as cores para os planetas.

Para as luas, decidimos fazer algumas das mais importantes. Para tal, criamos um subgrupo para o planeta correspondente, fazendo depois as devidas transformações.

6.2 Cintura de Asteróides

Para cintura de asteróides tornava-se impossível adicionar grupos ou sub-grupos manualmente no XML, devido a ser uma quantidade muito elevada.

Para tal criamos um novo atributo para o elemento "group" do XML: "units".

Este atributo vai dizer o número de unidades que vamos desenhar.

Dentro do group teremos um transform que terá algumas alterações para conseguirmos a aleatorieade dos asteróides.

O transform contem os atributos normais (x,y,z), mas pode conter ainda (xR,yR,zR). Se o valor do xR for 10 e do x 20, significa que vai ser gerado um valor entre 0 e 10 que vai ser somado ao 20 para obter o valor da variável x.

Isto também se aplica ao angle do Rotate.

Com esta adição e com a especificação abaixo, conseguimos fazer o cintura de asteróides.

```
<!--Asteroides-->
<group units="1000">
    <color r="100" g="100" b="100"/>
    <models>
        <model file="asteroid.3d"/>
    </models>
    <transform>
        <rotate angle="0" angleR="360" x="0" y="1" z="0"/>
        <translate x="0" xR="7" y="0" yR="3" z="18"/>
    </transform>
</group>
```

Figure 3: Especificação do XML para a cintura de asteróides

6.3 Estrelas

Com uso do atributo "units" explicado anteriormente, fizemos um cubo grande onde o sistema solar se encontra no meio, nas faces do cubo se vão encontrar pequenos planos. Deste modo, dá a perspetiva que há estrelas.

Para isso especificamos as 6 faces dos cubos:

```
<!--Estrelas-->
<group units = "100">
    <models>
        <model file="star.3d"/>
    </models>
    <color r="255" g="255" b="255"/>
        <transform>
            <translate x="-250" y= "-250" yR="500" z="-250" zR="500"/>
            <rotate angle="90" x="0" y="0" z="1"/>
        </transform>
</group>
<group units="100">
    <models>
        <model file="star.3d"/>
    </models>
    <color r="255" g="255" b="255"/>
    <transform>
        <translate x="250"  y="-250" yR="500" z="-250" zR="500"/>
        <rotate angle="90" x="0" y="0" z="1"/>
    </transform>
</group>
<group units="100">
    <models>
        <model file="star.3d"/>
    </models>
    <color r="255" g="255" b="255"/>
    <transform>
        <translate x="-250"  xR="500" y="-250" yR="500" z="250"/>
        <rotate angle="90" x="1" y="0" z="0"/>
    </transform>
</group>
<group units="100">
    <models>
        <model file="star.3d"/>
    </models>
    <color r="255" g="255" b="255"/>
    <transform>
        <translate x="-250" xR="500" y="-250" yR="500" z="-250"/>
        <rotate angle="90" x="1" y="0" z="0"/>
    </transform>
</group>
<group units="100">
    <models>
        <model file="star.3d"/>
    </models>
    <color r="255" g="255" b="255"/>
    <transform>
        <translate x="-250" xR="500" y="-250" z="-250" zR="500"/>
    </transform>
</group>
<group units="100">
    <models>
        <model file="star.3d"/>
    </models>
    <color r="255" g="255" b="255"/>
    <transform>
        <translate x="-250" xR="500" y="250" z="-250" zR="500"/>
    </transform>
</group>
```

Figure 4: Especificação do XML para as estrelas

7 Melhoria na gestão de memória

Para o sistema solar serão necesessários 4 ficheiros:

- sphere.3d (sphere 0.5 30 30)
- torus.3d (torus 3.3 0.3 50 2)
- asteroid.3d(sphere 0.1 5 5)
- star.3d (plane 0.5 1)

Anteriormente seriam guardados os pontos para cada model, isto é desnecessário visto que muitos models tem o mesmo ficheiro associado, por isso, em vez disso, para um uso mais eficiente memória decidimos utilizar um mapa que associasse o nome do ficheiro à lista de pontos, assim em vez de cada model guardar a sua lista de pontos, seria apenas uma referência para o mapa.

Assim na leitura do ficheiro XML, criamos o mapa com os ficheiros que serão necessários ler. Depois disso, quando iríamos fazer a leitura do ficheiro para cada model, desta vez, vemos qual o nome do ficheiro de pontos, caso o ficheiro já tenha sido lido, passa-se a referência do vector do pontos que se encontra no mapa para o vetor de pontos do model, caso ainda não tenha sido lido, faz-se a devida leitura e depois disso, a respetiva associação.

8 Demonstrações

8.1 Ficheiros de Teste

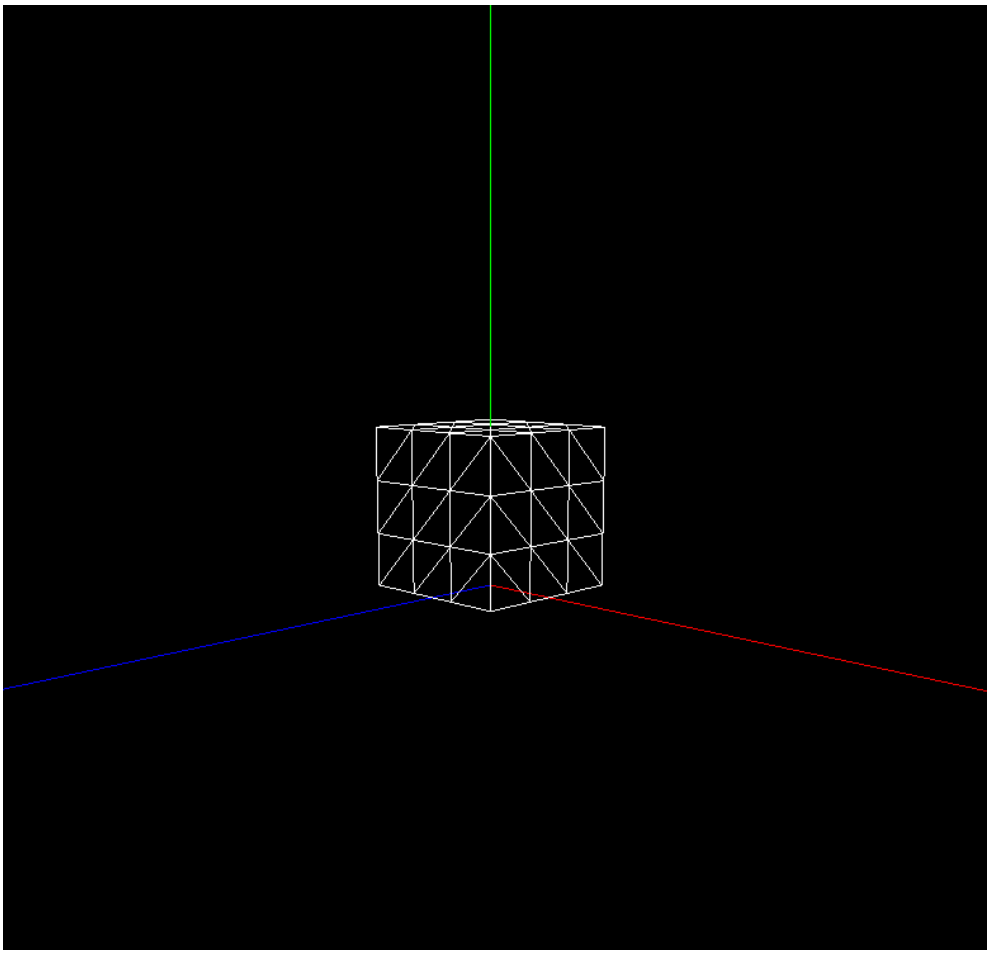

Figure 5: Ficheiro 2.1

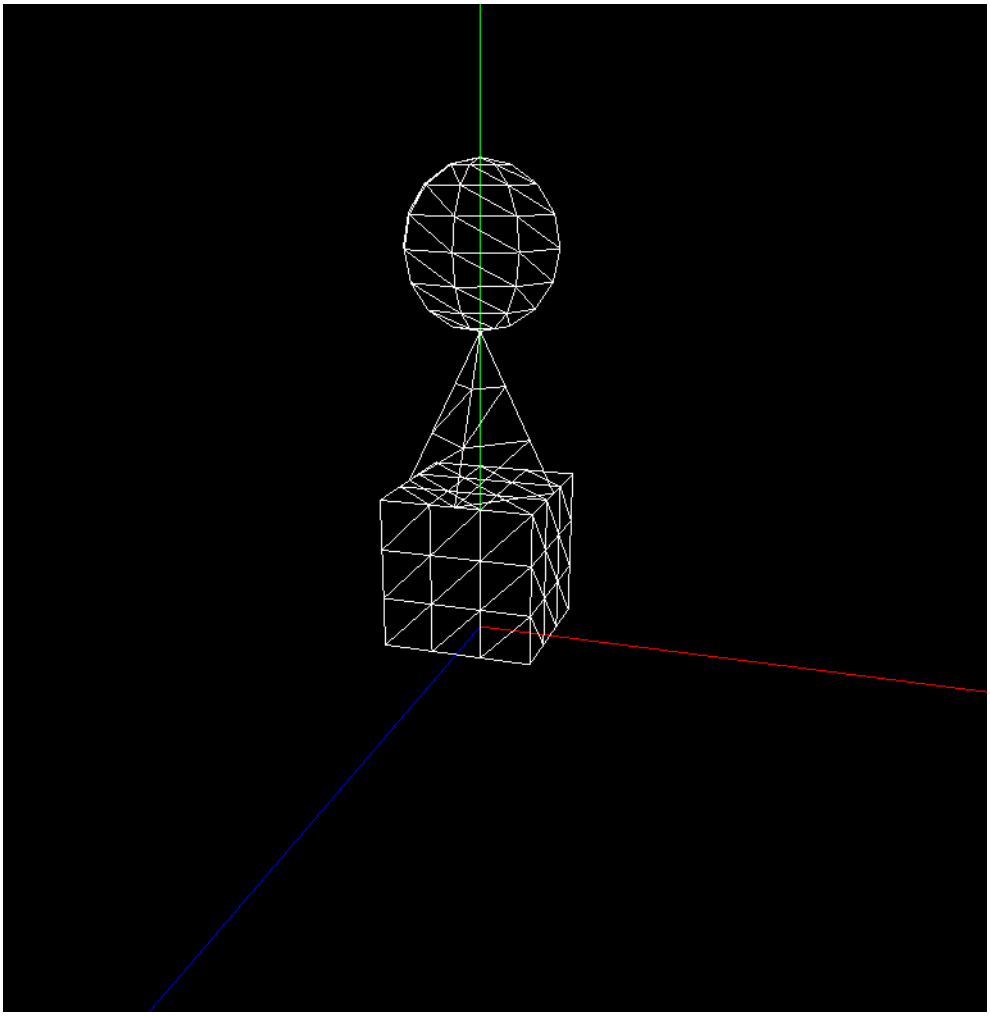

Figure 6: Ficheiro 2.2

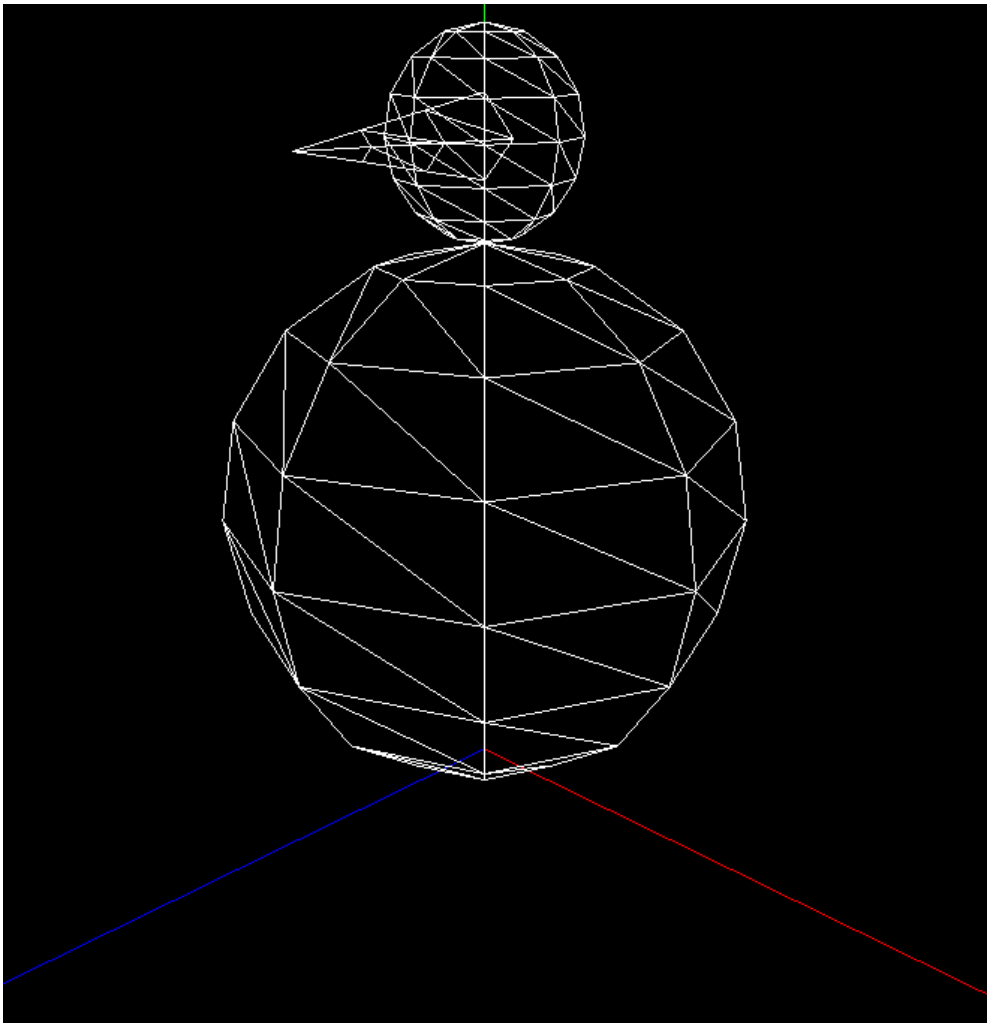

Figure 7: Ficheiro 2.3

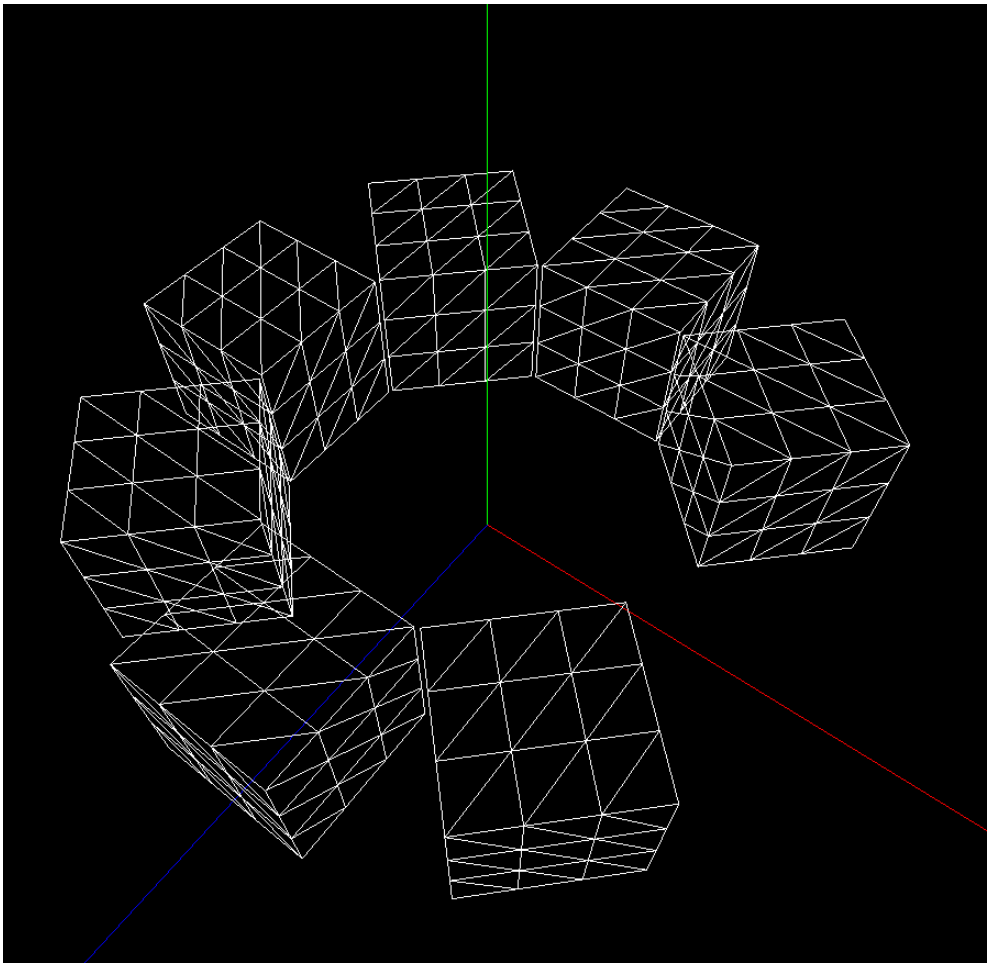

Figure 8: Ficheiro 2.4

8.2 Sistema Solar

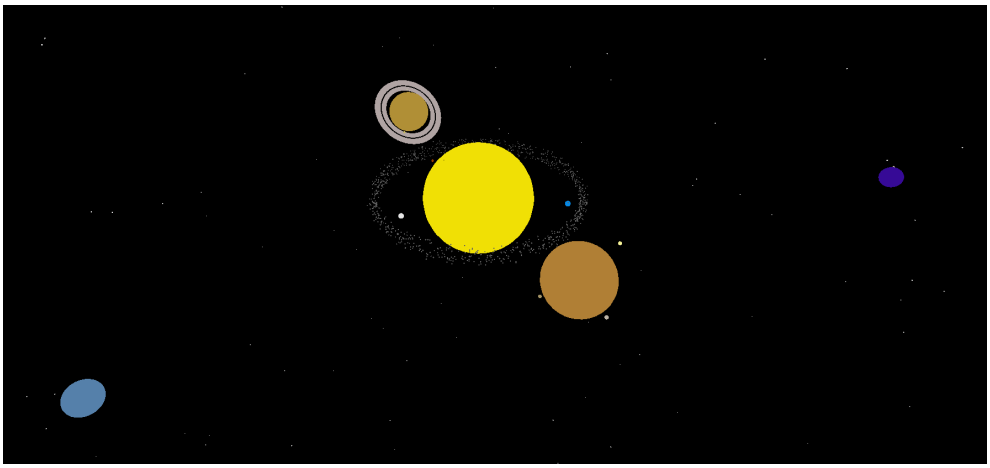

Figure 9: Sistema Solar - vista total

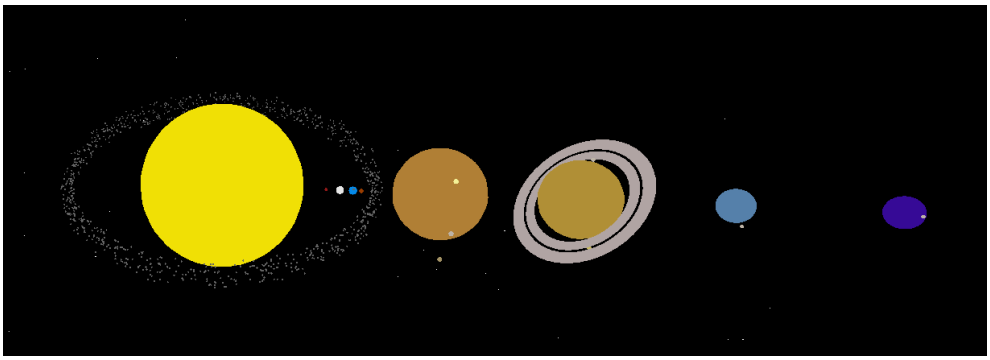

Figure 10: Sistema Solar - alinhados, sem rotações (excepto asteróides e anel de Saturno)

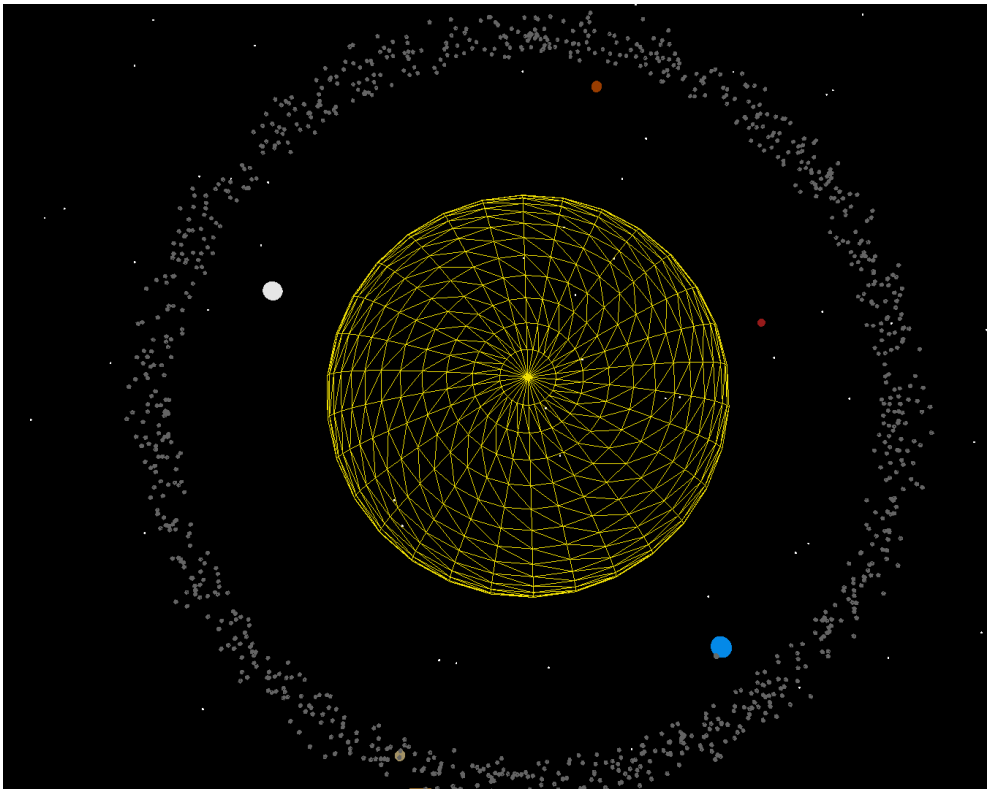

Figure 11: Sistema Solar - primeiros 4 e cintura de asteróides vista de cima (com linhas)

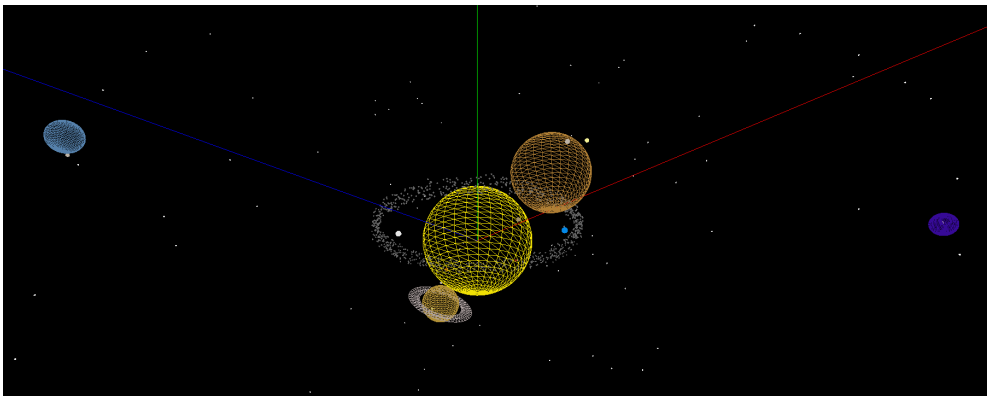

Figure 12: Sistema Solar - vista geral com eixos (com linhas)

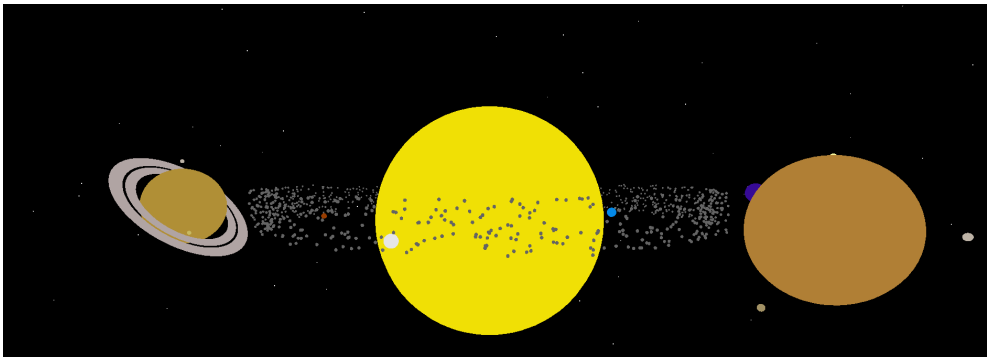

Figure 13: Sistema Solar - Vista com y=0

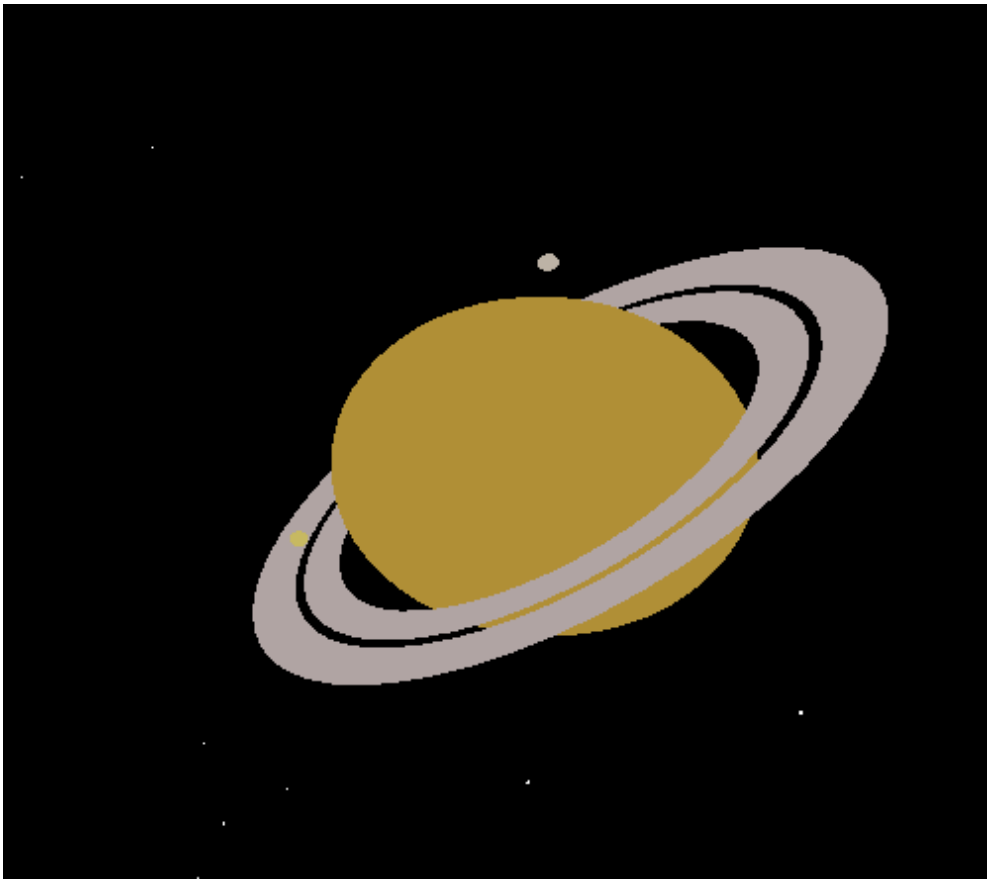

Figure 14: Sistema Solar - Planeta Saturno

9 Conclusão

Nesta segunda fase, consideramos que este projeto foi-nos útil para aprofundar o conhecimento adquirido em aula, nomeadamente na extensão do conhecimento da linguagem de programação C++ e no parsing e criação de ficheiros *XML*.

Por outro lado, a reestruturação do projeto achamos ter sido bem conseguida, pois ajudou-nos a organizar melhor as classes e os conceitos fornecidos. Além disso, estamos contentes com o produto final do nosso sistema solar que, como referido anteriormente, não corresponde à realidade, mas visualmente dá a ideia que pretendemos para o *Sistema Solar*.

Posto isto, esta segunda fase teve um aproveitamento bastante positivo a nível de expansão do conhecimento da *UC* e do cumprimento dos requisitos do enunciado.