



UNIVERSIDADE DO MINHO
Departamento de Informática

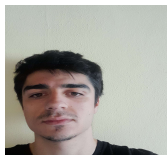
COMPUTAÇÃO GRÁFICA

Phase 2 - Geometric Transforms

Grupo 37

Feito por:

Dinis Gonçalves Estrada (A97503)
Emanuel Lopes Monteiro da Silva (A95114)



A97393



A95114

April 29, 2023
Ano Letivo 2022/23

Conteúdo

1	Introdução	2
2	Objetivos	2
3	Arquitetura atualizada	3
4	World atualizado	3
4.1	Tranformations	3
4.2	Groups	3
5	Generator	4
5.1	Comandos	4
5.2	Cálculo dos Pontos	4
5.2.1	Torus	4
6	Engine - Atualização	6
6.1	Estrutura de Dados	6
6.2	Leitura e “Parse” do ficheiro XML	6
7	Sistema Solar	7
7.1	Modelos utilizados	7
7.2	Criação do ficheiro XML	7
7.2.1	Sol	8
7.2.2	Mercúrio, Vénus, Terra e Marte	8
7.2.3	Júpiter, Saturno, Úrano e Neptuno	9
7.3	Output	10
8	Conclusão	11

1 Introdução

Nesta segunda fase do projeto da unidade curricular de Computação Gráfica foi necessário fazer alterações ao trabalho elaborado na primeira fase. Estas alterações consistem na introdução das transformações geométricas como a translação, a escala e a rotação aos objetos criados.

Na realização desta etapa foi necessário alterar o parser dos ficheiros XML, uma vez que estes sofreram alterações. Ou seja, uma vez que estes ficheiros possuem uma estrutura hierárquica em árvore, cada nodo possui o nome de um ficheiro 3d onde estão os vértices dos objetos e as transformações que serão aplicadas a estes.

Para que isto fosse possível o "engine" sofreu alterações assim como o "generator" com uma nova figura geométrica, o Torus.

2 Objetivos

Na parte do "generator" foi necessário adicionar uma nova figura, o Torus, para ser utilizado mais tarde na criação do Sistema Solar.

Relativamente ao "engine" efetuaram-se alterações no parser de leitura dos ficheiros XML para que fosse possível ler também as transformações a aplicar à figura e por isso foram criadas novas estruturas de dados para guardar essa informação.

Com as modificações todas feitas tivemos como último objetivo criar um modelo de um sistema solar a partir de um ficheiro XML.

3 Arquitetura atualizada

De um modo geral, a arquitetura do programa não foi alterada, com a exceção do package World em que tivemos de alterar as estruturas de dados.

Uma das modificações então feitas foi aprimorar a estrutura de dados devido à necessidade de armazenar os novos elementos da leitura do ficheiro XML.

4 World atualizado

4.1 Transformations

Adicionamos o namespace transformations que inclui uma classe transform constituída por uma variável float relativa ao angle (que apenas é usado para definir ângulos no caso de rotações) e um array de floats fixo de dimensão 3 (correspondente às coordenadas x,y,z) e uma variável transtype, que corresponde a uma classe enum também definida no mesmo namespace. Este transtype não é nada mais do que apenas um identificador do tipo de transformação, podendo então tratar-se de uma rotação, translação, escala ou mudança de cor.

A acompanhar esta classe transform estão definidas todas as funções get e set necessárias para aceder a qualquer uma das suas variáveis e para definir variáveis transform com valores específicos, mantendo o encapsulamento.

4.2 Groups

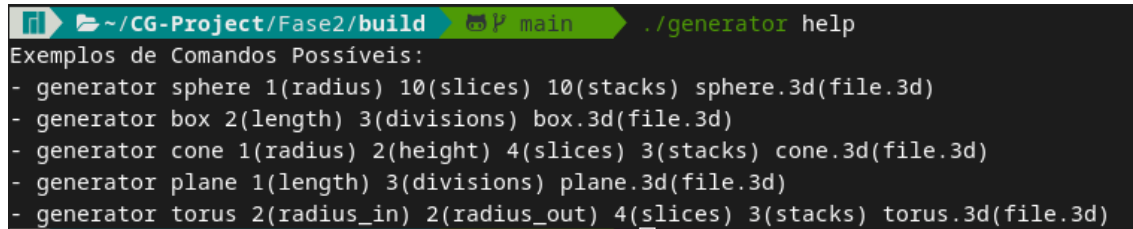
Tal como o namespace transformations adicionamos o a classe Group que é composta por vetores de transforms, figures e groups. Esta classe será a principal estrutura de dados responsável por armazenar a informação obtida a partir de ficheiros XML, sendo que cada grupo poderá conter não só transformações específicas mas também grupos "filhos" aos quais estas também terão de ser aplicadas.

A acompanhar esta classe Group estão definidas todas as funções get e set necessárias para aceder a qualquer um dos vetores e funções responsáveis por adicionar novas transformações, figuras, ou grupos aos mesmos.

5 Generator

5.1 Comandos

Para que possa ser mais fácil o uso desta aplicação existe um comando help que disponibiliza os possíveis comandos.



```
~/CG-Project/Fase2/build main ./generator help
Exemplos de Comandos Possíveis:
- generator sphere 1(radius) 10(slices) 10(stacks) sphere.3d(file.3d)
- generator box 2(length) 3(divisions) box.3d(file.3d)
- generator cone 1(radius) 2(height) 4(slices) 3(stacks) cone.3d(file.3d)
- generator plane 1(length) 3(divisions) plane.3d(file.3d)
- generator torus 2(radius_in) 2(radius_out) 4(slices) 3(stacks) torus.3d(file.3d)
```

Figure 1: Menu Help Atualizado

5.2 Cálculo dos Pontos

Na parte do Generator podemos encontrar um ficheiro .cpp que terá todas as funções relacionadas com o cálculo dos pontos dos triângulos de forma representar as figuras que o utilizador pretender. Estes pontos são inicialmente guardados numa struct denominada por figure que mais tarde será interpretada e guardaremos os seus pontos num ficheiro .3d para que o engine o possa ler.

5.2.1 Torus

A função presente no generator que gera os pontos dos triângulos para representar um torus denomina-se por calcTorus e recebe o raio da circunferência interior, o raio da circunferência entre a circunferência exterior e a interior, o número de stacks e o números de slices.

Para desenhar esta figura recorreremos à sua parametrização, sendo que o r é o raio interno e R é o raio externo:

$$\begin{aligned}x &= (R + r \cos v) \cos u \\y &= (R + r \cos v) \sin u \\z &= r \sin v \\u, v &\in [0, 2\pi]\end{aligned}$$

Figure 2: Parametrização do Torus

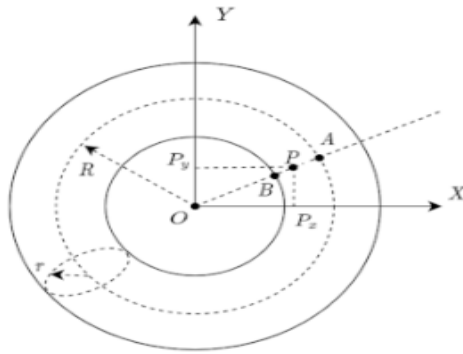


Figure 3: Formato do Torus

Após o cálculo de cada ponto dos triângulos, a interpretação destes pelo engine corresponde aos exemplo seguinte:

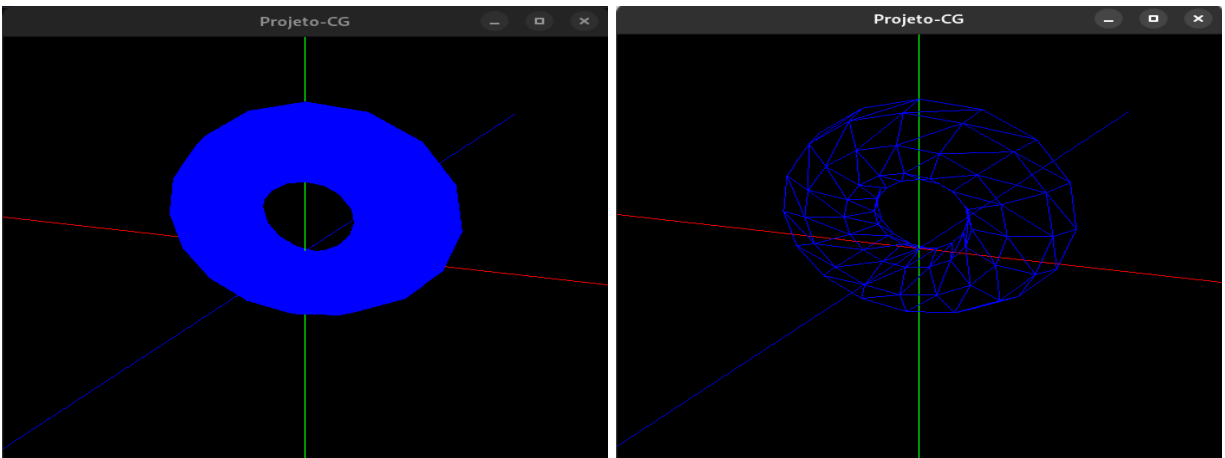


Figure 4: Torus com 2 radius_in, 1 radius_out, 16 slices e 8 stacks

6 Engine - Atualização

Nesta fase do projeto, foi necessário atualizar a aplicação “Engine” de forma a conseguir ler o novo formato de ficheiro XML proposto no enunciado. Além disso, foi fundamental também alterar a estrutura de dados utilizada para armazenar os conteúdos do ficheiro lido.

6.1 Estrutura de Dados

Para armazenar os dados obtidos através do ficheiro XML dado pelo utilizador, nesta segunda fase, usamos a classe criada Group no ficheiro world.hpp. Esta classe é usada como estrutura de dados, e é constituída por 3 vetores, um vetor dedicado às figuras obtidas do ficheiro, outro às transformações e finalmente um último que contém os group filhos presentes nesse mesmo group.

```
class Group {
private:
    vector<figure> models;
    vector<transformations::Transform> transforms;
    vector<Group> child_nodes;
public:
    Group() {};;
    Group(vector<figure> figs, vector<transformations::Transform> transforms, vector<Group> child_nodes);
    vector<figure> getModels();
    vector<transformations::Transform> getTrans();
    vector<Group> getChilds();
    void addGroup(Group g);
    void addTransform(transformations::Transform);
    void addFigure(utis::figure);
};
```

Figure 5: Class Group

Foi essencial criar uma classe recursiva devido ao formato do ficheiro XML, uma vez que, um elemento group pode ter subelementos group como filhos.

6.2 Leitura e “Parse” do ficheiro XML

Para ler o ficheiro e fazer “parse” do mesmo, foi necessário modificar a função “readXMLfile” feita na 1ª fase deste trabalho. Dividimos a função em três: a função “readXMLfile” que será responsável por ler os elementos “window” e “camera” e por inicializar a class World que irá armazenar todos os dados contidos no ficheiro XML; a função “readXMLmodels” que será responsável por ler os models, ficheiros .3d; e a função “readXMLgroup” que é responsável por dividir e identificar os vários elementos presentes num “group” do ficheiro XML. Esta função recebe um apontador para um elemento “group” do ficheiro e a estrutura de dados group onde irá armazenar os diversos dados lidos.

Primeiramente, irá criar um ciclo, de forma a percorrer cada elemento filho do apontador passado no input. Dentro do ciclo, é analisado o value de cada um deles, que poderá corresponder a 3 tipos: “transform”, “models” e “group”. Quando o value é do tipo “transform” teremos que analisar os filhos deste mesmo elemento, que podem ser de um dos seguintes tipos: “translate”, “rotate”, “scale” ou “color”, e nesse caso iremos analisar os atributos do elemento para fazermos a inicialização, com os respetivos valores, da classe Transform e em seguida adicioná-la à estrutura group que estamos a usar. Caso o valor do elemento em questão seja “models”, teremos de analisar os filhos desse mesmo elemento, de forma a obter os modelos, que contém a referência para o ficheiro .3d que deve ser lido. De seguida, teremos de chamar a função “readXMLmodels”, de forma a obter a Figure equivalente ao ficheiro, e por fim, adicioná-la à estrutura group que estamos a usar. Quando o elemento filho de um group for do próprio tipo Group, iremos inicializar uma nova classe group, com o nome de “grupo”. Depois, iremos chamar a própria função “readXMLgroup”, passando como input o apontador para o elemento filho onde se encontra, e o “grupo”, voltando assim, a fazer todo este processo sucessivamente até chegar ao final do ficheiro.

7 Sistema Solar

Nesta segunda fase do projeto foi pedido como demo um modelo estático do sistema solar, incluindo o sol, os planetas e respectivas luas, em hierarquia com objetivo principal de demonstrar as novas funcionalidades, para tal, foi necessário elaborar um ficheiro XML, contendo todas as transformações e ficheiros .3d necessários. O respetivo ficheiro XML encontra-se em "tests/tests_files_phase_2/solar_system.xml".

Não foram utilizadas as dimensões reais pois os tamanhos dos astros iriam diferir muito, pelo que, por exemplo, as luas seriam quase impercetíveis.

As cores e dimensões dos astros do sistema solar para o desenho do sistema solar tiveram como base a representação do sistema solar da figura 6. Para além do Sol, dos planetas e da Lua foram incluídas apenas 4 luas de Júpiter e Saturno, respetivamente.

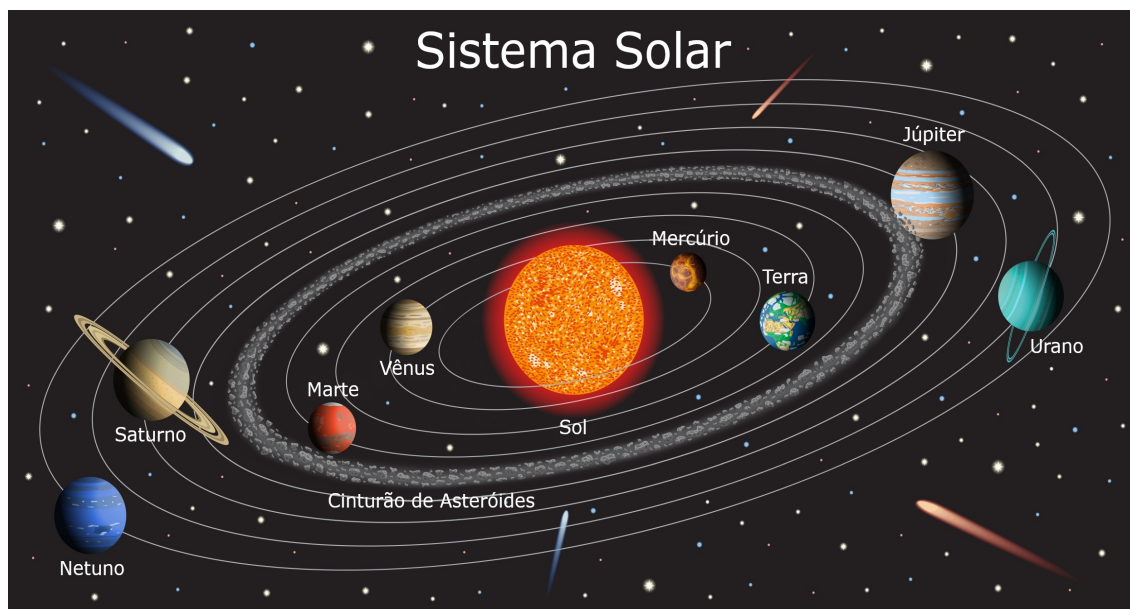


Figure 6: Sistema Solar

7.1 Modelos utilizados

Com recurso ao gerador implementado na fase anterior, foram criados três ficheiros .3d para servirem de modelos no ficheiro XML. Em primeiro lugar foi gerado o ficheiro sphere.3d (esfera de raio 2, com 20 stacks e 20 slices) que será responsável por desenhar o sol e todos os planetas e respectivas luas. Para os anéis de saturno foi gerado o ficheiro torus.3d (toro de raio interior 0.5, raio exterior 3, com 2 stacks e 30 slices).

7.2 Criação do ficheiro XML

O ficheiro XML final contém vários grupos principais que referenciam um dos seguintes desenhos: o sol, um planeta (que poderá incluir subgrupos que referenciam as suas luas ou anéis).

De acordo com o output desejado foram escritos todos os conteúdos seguindo a ordem a seguir referida.

7.2.1 Sol

Desenho do sol no centro do referencial (sem translações):

```
<group>
  <transform>
    <color R="1" G="0.6" B="0"/>
  </transform>
  <models>
    <model file="sphere.3d"/>
  </models>
</group>
```

Figure 7: Sol

7.2.2 Mercúrio, Vénus, Terra e Marte

Desenho dos planetas telúricos num ponto qualquer da sua orbita (rotação em torno do eixo y) com intervalo de 0.5 unidades no eixo do x entre si:

```
<group>
  <transform>
    <rotate angle="-30" x="0" y="1" z="0"/>
    <color R="0.65" G="0.3" B="0.15"/>
    <translate x="3" y="0" z="0"/>
    <scale x="0.05" y="0.05" z="0.05"/>
  </transform>
  <models>
    <model file="sphere.3d"/>
  </models>
</group>
<group>
  <transform>
    <rotate angle="12" x="0" y="1" z="0"/>
    <color R="0.7" G="0.5" B="0.3"/>
    <translate x="3.5" y="0" z="0"/>
    <scale x="0.06" y="0.06" z="0.06"/>
  </transform>
  <models>
    <model file="sphere.3d"/>
  </models>
</group>
```

```
<group>
  <transform>
    <rotate angle="234" x="0" y="1" z="0"/>
    <color R="0" G="0" B="0.8"/>
    <translate x="4" y="0" z="0"/>
    <scale x="0.08" y="0.08" z="0.08"/>
  </transform>
  <models>
    <model file="sphere.3d"/>
  </models>
  <group>
    <transform>
      <translate x="0" y="0" z="3"/>
      <color R="0.9" G="0.9" B="0.9"/>
      <scale x="0.25" y="0.25" z="0.25"/>
    </transform>
    <models>
      <model file="sphere.3d"/>
    </models>
  </group>
</group>
<group>
  <transform>
    <rotate angle="332" x="0" y="1" z="0"/>
    <color R="0.85" G="0.2" B="0.2"/>
    <translate x="4.5" y="0" z="0"/>
    <scale x="0.06" y="0.06" z="0.06"/>
  </transform>
  <models>
    <model file="sphere.3d"/>
  </models>
</group>
```

Figure 8: Mercúrio e Vénus, Terra, Lua e Marte, respetivamente

7.2.3 Júpiter, Saturno, Úrano e Neptuno

Desenho dos planetas gasosos num ponto qualquer da sua orbita (rotação em torno do eixo y) com intervalo de 1.5 unidades no eixo do x entre si:

```
<group>
  <transform>
    <rotate angle="275" x="0" y="1" z="0"/>
    <color R="0.7" G="0.5" B="0.3"/>
    <translate x="8" y="0" z="0"/>
    <scale x="0.3" y="0.3" z="0.3"/>
  </transform>
  <models>
    <model file="sphere.3d"/>
  </models>
</group>
```

Figure 9: Júpiter

<pre><group> <transform> <color R="0.95" G="0.5" B="0.3"/> <rotate angle="145" x="0" y="1" z="0"/> <translate x="9.5" y="0" z="0"/> <scale x="0.22" y="0.22" z="0.22"/> </transform> <models> <model file="sphere.3d"/> </models> <group> <transform> <color R="0.8" G="0.5" B="0.3"/> <rotate angle="70" x="1" y="0" z="0"/> </transform> <models> <model file="torus.3d"/> </models> </group> </group></pre>	<pre><group> <transform> <color R="0" G="1" B="1"/> <rotate angle="90" x="0" y="1" z="0"/> <translate x="11" y="0" z="0"/> <scale x="0.23" y="0.23" z="0.23"/> </transform> <models> <model file="sphere.3d"/> </models> </group> <group> <transform> <color R="0" G="0.666" B="1"/> <translate x="12.5" y="0" z="0"/> <scale x="0.17" y="0.17" z="0.17"/> </transform> <models> <model file="sphere.3d"/> </models> </group></pre>
--	---

Figure 10: Saturno e Anel, Úrano e Neptuno, respetivamente

7.3 Output

Após a leitura do ficheiro XML final pode-se observar o seguinte output:

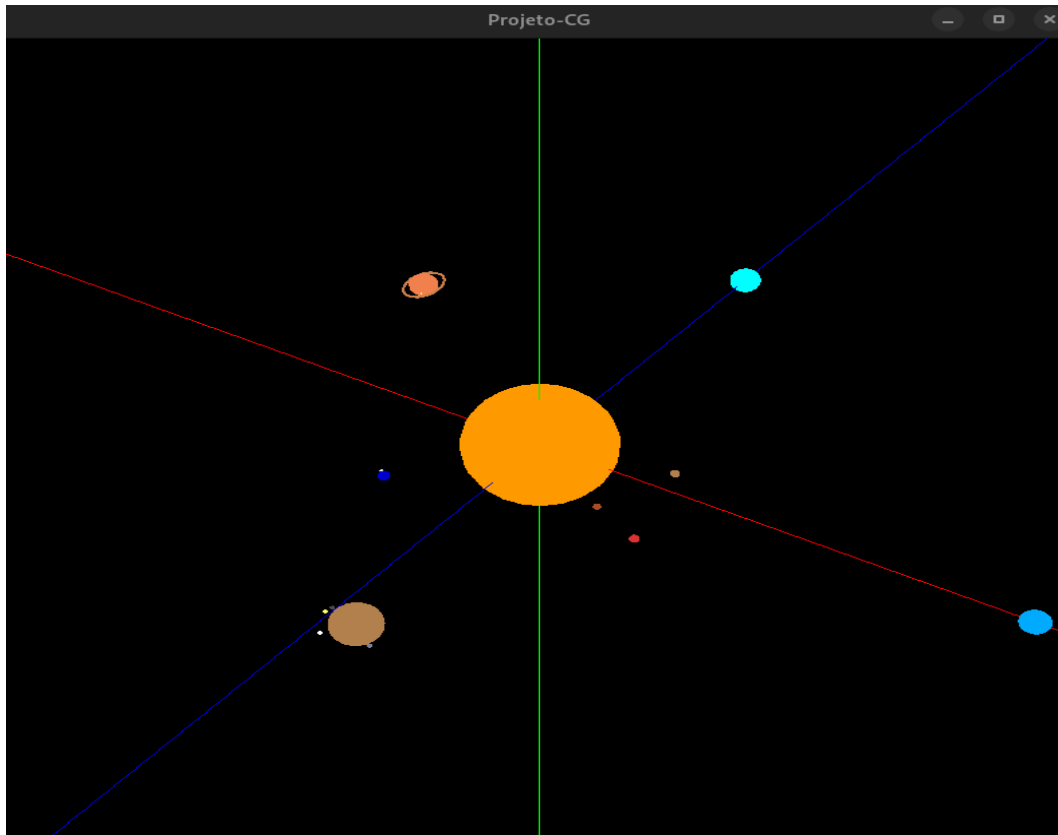


Figure 11: Sistema Solar

8 Conclusão

Nesta segunda fase, foi possível, uma vez mais, aplicar e consolidar vários conceitos abordados nas aulas teóricas e práticas, bem como aprofundar o nosso conhecimento em OpenGL.

Com isto, o grupo considera que realizou com sucesso esta etapa uma vez que tudo está a funcionar e que o modelo do sistema solar também correspondeu ao esperado.