



**Universidade do Minho**

**Transformações Geométricas**  
**Unidade Curricular de Computação Gráfica**  
Licenciatura em Ciências da Computação  
Universidade do Minho

Manuel Marques  
(A85328)

Pedro Oliveira  
(A86328)

João Rodrigues  
(A84505)

Eduardo Pereira  
(A84098)

10 de Março de 2020

# Índice

<b>1</b>	<b>Contextualização</b>	<b>2</b>
1.1	Enunciado . . . . .	2
1.2	Resumo . . . . .	3
<b>2</b>	<b>Apresentação das soluções</b>	<b>4</b>
2.1	Estruturas de dados . . . . .	4
2.1.1	Vector dos modelos . . . . .	5
2.1.2	Vector das Transformações Geométricas . . . . .	5
2.1.3	Vector Sequência . . . . .	5
2.2	Sistema Solar Estático . . . . .	7
<b>3</b>	<b>Conclusão</b>	<b>8</b>

# Índices de Figuras

2.1	<i>.xml</i> de demonstração das estruturas . . . . .	4
2.2	Resultado gráfico de acordo com o <i>xml</i> . . . . .	6
2.3	<i>xml</i> que caracteriza o sistema solar estático . . . . .	7
2.4	Representação gráfica de acordo com o <i>xml</i> do sistema solar . . . . .	7

# Capítulo 1

## Contextualização

### 1.1 Enunciado

#### Phase 2 – Geometric Transforms

This phase is about creating hierarchical scenes using geometric transforms. A scene is defined as a tree where each node contains a set of geometric transforms (translate, rotate and scale) and optionally a set of models. Each node can also have children nodes.

Example of a configuration XML file with a single group:

```
<scene>
  <group>
    <translate X=5 Y=0 Z=2 />
    <rotate angle=45 axisX=0 axisY=1 axisZ=0 />
    <models>
      <model file="sphere.3d" />
    </models>
  </group>
</scene>
```

Example of a group with a child group:

```
<scene>
  <group>
    <translate X=1 />
    <models>
      <model file="sphere.3d" />
    </models>
    <group>
      <translate Y=1 />
      <models>
        <model file="cone.3d" />
      </models>
    </group>
  </group>
</scene>
```

In the second example the child group will inherit the geometric transforms from the parent.

Geometric transformations can only exist inside a group, and are applied to all models and subgroups.

Note: the order of the geometric transforms is relevant.

## 1.2 Resumo

Na abordagem a este problema, decidimos definir três estruturas da classe *vector*, identificados por *modelos*, *geo\_tr* e *sequencia*. O vector *modelos* guarda a informação dos pontos de cada vértice a desenhar.

*geo\_tr* contém a informação das transformações geométricas depois de lidas no ficheiro xml e *sequencia* guarda a sequência de instruções para desenho da cena descrita no ficheiro xml.

## Capítulo 2

# Apresentação das soluções

### 2.1 Estruturas de dados

Para efeito de apresentação das estruturas de dados utilizadas para guardar a informação necessária para o desenho da cena, criou-se o seguinte ficheiro *xml*. De seguida apresentamos o conteúdo dos *vectors* de acordo com o conteúdo do ficheiro.

```
<scene>
  <group>
    <translate X=-2 Y=0 Z=0/>
    <model file="sun.3d" />
    <group>
      <translate X=2.17 Y=0 Z=0/>
      <model file="mercury.3d" />
    </group>
    <group>
      <translate X=4.6 Y=0 Z=0/>
      <model file="venus.3d" />
    </group>
  </group>
</scene>
```

Figura 2.1: *.xml* de demonstração das estruturas

### 2.1.1 Vector dos modelos

Nesta estrutura guardamos os pontos de acordo com o modelo que foi previamente gerado pelo *Generator*.

Por uma questão de simplificação, na figura seguinte, apresentamos apenas os nomes dos modelos a desenhar.

Para o *xml* de exemplo, o resultado do vector é :

sun.3d	mercury.3d	venus.3d
--------	------------	----------

### 2.1.2 Vector das Transformações Geométricas

Na estrutura referida, guardamos as transformações geométricas detalhadas no ficheiro *xml*. Por definição nossa, atribuímos um inteiro para cada transformação:

0 - *Translate*

1 - *Rotate*

2 - *Scale*

Para este exemplo em concreto , o vector é apenas constituído por 0, pois temos apenas translações mas, a situação é análoga para as restantes transformações.

Para o *xml* de exemplo, o resultado do vector é :

0	0	0
---	---	---

### 2.1.3 Vector Sequência

Nesta estrutura guardamos a sequência de desenho. Para isso atribuímos um valor inteiro a cada função:

0 - *glPushMatrix*

1 - *glPopMatrix*

2 - *writeGeoTransf*

2 - *writeModelo3D*

Para o *xml* de exemplo, o resultado do vector é :

0	2	3	0	2	3	1	0	2	3	1	1
---	---	---	---	---	---	---	---	---	---	---	---

O resultado gráfico da leitura deste exemplo *xml* é o seguinte:

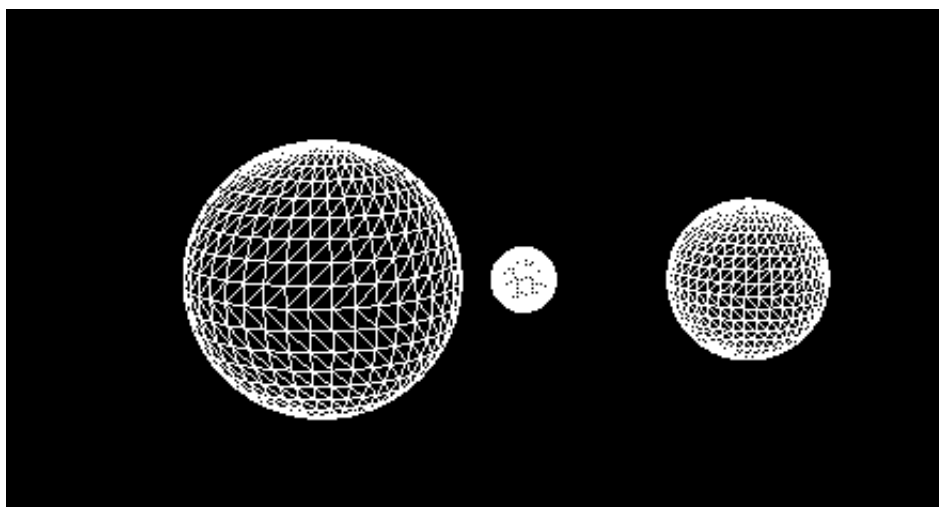


Figura 2.2: Resultado gráfico de acordo com o *xml*

## 2.2 Sistema Solar Estático

```
<scene>
  <group>
    <translate X=-8 Y=0 Z=0/>
    <model file="sun.3d" />
    <group>
      <translate X=2.17 Y=0 Z=0/>
      <model file="mercury.3d" />
    </group>
    <group>
      <translate X=4.6 Y=0 Z=0/>
      <model file="venus.3d" />
    </group>
    <group>
      <translate X=7.05 Y=0 Z=0/>
      <model file="earth.3d" />
      <group>
        <translate X=1 Y=0 Z=-1/>
        <model file="moon.3d" />
      </group>
    </group>
    <group>
      <translate X=9.29 Y=0 Z=0/>
      <model file="mars.3d" />
    </group>
    <group>
      <translate X=11.79 Y=0 Z=0/>
      <model file="jupyter.3d" />
    </group>
    <group>
      <translate X=14.21 Y=0 Z=0/>
      <model file="saturn.3d" />
    </group>
    <group>
      <translate X=16.39 Y=0 Z=0/>
      <model file="uranus.3d" />
    </group>
    <group>
      <translate X=18.56 Y=0 Z=0/>
      <model file="neptune.3d" />
    </group>
  </group>
</scene>
```

Figura 2.3: *xml* que caracteriza o sistema solar estático

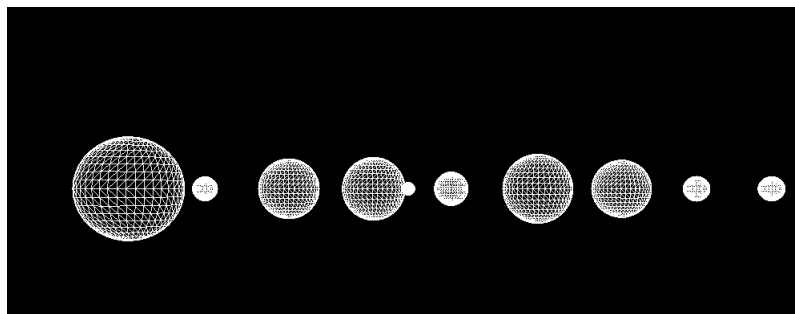


Figura 2.4: Representação gráfica de acordo com o *xml* do sistema solar



## Capítulo 3

# Conclusão

Damos