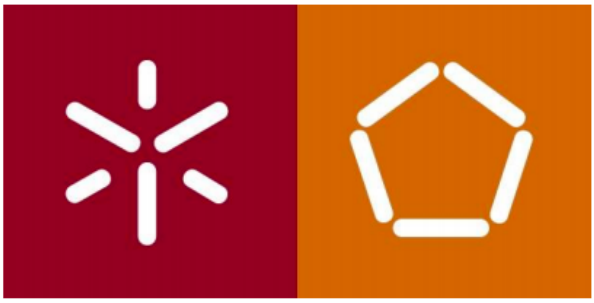

Universidade do Minho

Mestrado Integrado em Engenharia Informática

29/03/2020

Computação Gráfica
Fase 2

António Gonçalves (A85516)
João Fernandes (A84034)
Eduardo Conceição (A83870)
Rita Rosendo (A84475)

Contents

1 Introdução

Este relatório irá descrever a segunda fase do projeto da cadeira de Computação Gráfica. Nesta fase do trabalho foi-nos proposto que, a partir da primeira fase, construíssemos um programa que conseguisse ler e efetuar transformações geométricas, desenhando a partir destas um modelo do Sistema Solar, representando os vários planetas que o constituem, com as devidas dimensões e distâncias entre eles.

2 Análise do Problema

Nesta parte do projeto iremos utilizar a fase anterior como base, alterando alguns pormenores que são necessários para que a presentação do sistema solar seja realizada.

2.1 *Generator*

Uma vez que nesta fase só iremos necessitar de desenhar esferas, não será preciso alterar o *Generator* criado para a fase anterior.

2.2 *Engine*

Será necessário modificar o *engine*. Este terá que distinguir os vários grupos e subgrupos do XML para poder realizar a leitura adequada das dimensões e transformações necessárias para representar o modelo do sistema solar.

Nesta parte do trabalho, o ficheiro que contem a informação do modelo será bastante diferente. Em vez de receber qual a figura a desenhar e as suas dimensões este irá conter a informação sobre todos os planetas, agrupando os satélites de cada planeta em subconjuntos.

3 Resolução

3.1 Engine

Para guardar as diversas informações necessárias criamos vários vetores para representar todos os dados que estarão no ficheiro XML:

```
struct Group{
    GeometricTransf transforms;
    vector<string> models;
    vector<Point> points;
    vector<Group> childGroups;
};
```

Figure 1: Estrutura principal do projeto.

Nesta estrutura iremos guardar a informação de cada parte do ficheiro. De seguida iremos explicar a utilidade de cada um destes vetores e da estrutura *GeometricTransf* criada por nós.

3.1.1 *GeometricTransf*

```
struct GeometricTransf{
    GeoTPoint translate;
    GeoTPoint rotate;
    GeoTPoint scale;
};
```

Figure 2: Representação das transformações.

O objetivo desta estrutura é guardar e separar as diferentes transformações. Para isto temos uma estutura, a *GeoTPoint* genérica com os dados dos valores de x, y e z, bem como o valor de um ângulo.

```
struct GeoTPoint{
    float angle;
    float x;
    float y;
    float z;
};
```

Figure 3: Estrutura *GeoTPoint*.

Nem todas as transformações necessitam de toda estas informações, mas decidimos criar uma estrutura que pudesse ser utilizada por todas.

3.1.2 Vetores

Os restantes vetores são utilizados para guardar a informação dos vertices e dos modelos na memória, sendo esse uma das maiores diferenças em relação à fase anterior, onde estes eram guardados num ficheiro.

Por fim, iremos guardar as informações referentes aos *childGroups* noutro grupo dentro do pai, assim, todas as transformações referentes ao grupo do pai serão também efetuadas ao grupo filho, que neste caso serão os planetas (grupo pai) e os satélites (grupo filho). Assim, ao realizar o *translate* para desenhar o planeta na devida posição, o satelite só terá de realizar o *translate* a partir do novo referencial do grupo, que é o centro do planeta em questão.

3.1.3 Grupo Principal

Cada um destes grupos irá representar o planeta e os seu satélites (caso os tenha). Assim, criámos ainda outro vetor, *PrincipalGroups*, para agrupar todos os Grupos criados anteriormente de forma a puder ler estes sequencialmente para a representação do modelo.

3.2 Leitura

A leitura do ficheiero XML é feita utilizando várias funções. Começamos por fazer *parsing* dos grupos e sub-grupos, inicializando a *GeometricTransf* de cada um. No *parsing* de cada grupo terão de ser lidas as transformações e os modelos para que a informação esteja organizada devidamente. Por fim esta será guardada no vetor *PrincipalGroups*.

3.2.1 Desenho dos modelos

Após ler toda a informação será necessário desenhar os vários modelos. Para isto usamos duas funções: a *drawPs* e a *drawGs*.

A *drawPs* irá receber um vetor de pontos, sendo estes oriundos do ficheiro criado pelo *Generator*, com as coordenadas necessárias para desenhar o modelo. Este vetor será percorrido enquanto que os seus pontos são desenhados.

A *drawGs* recebe o conjunto de todos os grupos (*PrincipalGroup*), aplicando a cada um a *drawPs* e aos seus *childGroups*.

3.3 Câmara

O controlo da câmara é feito a partir dos inputs do teclado e do rato, que serão explicados em mais detalhe nas sub-secções seguintes.

3.3.1 Rato

Utilizando a função *processMouseButtons* e a função *processMouseMotion* podemos alterar a direção a partir do qual a câmara observa o modelo e aproximar ou afastar a ponto de vista da mesma. Assim, ao clicar no botão direito do rato alteramos a variável *tracking* na primeira função, sendo esta recebida na segunda, permitindo alterar o raio da câmara. O mesmo acontece com o botão esquerdo do rato, mas este faz com que seja possível alterar a posição da câmara.

3.3.2 Teclado

Podemos utilizar as setas do teclado para alterar o centro do referencial da câmara. Para isto usamos a função *specialKeys* que altera as variaveis camX, camZ, centerX e centerZ, sendo estas utilizadas na função gluLookAt. Para além disso a tecla 'r' servirá para dar *reset* às variáveis referentes à câmara, voltando esta ao seu estado inicial.

3.4 Escala e ficheiro XML

Tal como o enunciado indica, no ficheiro XML, encontra-se o modelo do sistema solar onde estão representados o Sol, os 8 planetas (Mercúrio, Vénus, Terra, Marte, Júpiter, Saturno, Urano e Neptuno) e alguns satélites dos planetas(Lua, Fobos, Europa, Titã, Umbriel e Tritão).

Inicialmente, definimos as escalas para desenhar os planetas e satélites conforme escalas reais, no entanto, optamos por definir escalas que nos permitissem representar o modelo no ecrã uma vez que as escalas iniciais faziam com que os desenhos ficassem demasiado grandes em relação a outros que ficavam quase invisíveis. Apesar disto, tentamos representar a realidade e ficamos com um desenho em que a esfera correspondente ao Sol é a maior, seguida da esfera do planeta Júpiter, etc.

Para representar a inclinação de cada planeta fizemos uma aproximação aos valores reais para realizar os devidos *rotates*. As diferentes distâncias entre os diversos planetas foram, mais uma vez, calculadas de forma a que o modelo final tivesse um tamanho aceitável, estando assim um pouco afastadas de uma representação real.

Tivemos uma dificuldade em conseguir que o *engine* conseguisse encontrar o ficheiro *sphere.3d* criado pelo *Generator*, por isso decidimos indicar-lhe o caminho direto para este conseguir aceder-lhe.

```
<scene>
 <!--Sol-->
 <group>
 <translate X="-40" />
 <scale X="9.5" Y="9.5" Z="9.5"/>
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
 </group>
 <!--Mercúrio-->
 <group>
 <translate X="-25" />
 <rotate angle="-0.1" axisX="0" axisY="0" axisZ="1" />
 <scale X="0.4878" Y="0.4878" Z="0.4878"/>
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
</group>
 <!--Venus-->
 <group>
 <translate X="-18" />
 <rotate angle="177.3" axisX="0" axisY="0" axisZ="1" />
 <scale X="1.2104" Y="1.2104" Z="1.2104"/>
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
 </group>
 <!--Terra-->
 <group>
 <translate X="-12.5" />
 <rotate angle="-23.26" axisX="0" axisY="0" axisZ="1" />
 <scale X="1.2742" Y="1.2742" Z="1.2742"/>
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
 <!--Lua-->
 <group>
  <translate X="-0.5" Y="1.8" />
  <rotate angle="-2" axisX="0" axisY="0" axisZ="1" />
  <scale X="0.28" Y="0.28" Z="0.28" />
  <models>
   <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
  </models>
 </group>
 </group>
```

```
<!--Marte-->
<group>
<translate X="-3.5" />
<rotate angle="-25.19" axisX="0" axisY="0" axisZ="1" />
<scale X="0.678" Y="0.678" Z="0.678"/>
<models>
 <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
</models>
<!--Fobos-->
<group>
 <translate X="-0.9" Y="1.3" />
 <rotate angle="-2" axisX="0" axisY="0" axisZ="1" />
 <scale X="0.21" Y="0.21" Z="0.21" />
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
</group>
</group>
<!--Jupiter-->
<group>
<translate X="19" />
<rotate angle="-3.13" axisX="0" axisY="0" axisZ="1" />
<scale X="3" Y="3" Z="3"/>
<models>
 <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
</models>
<!--Europa-->
<group>
 <translate X="-0.9" Y="1.8" />
 <rotate angle="-0.5" axisX="0" axisY="0" axisZ="1" />
 <scale X="0.11" Y="0.11" Z="0.11" />
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
</group>
</group>
<!--Saturno-->
<group>
<translate X="30" />
<rotate angle="-26.73" axisX="0" axisY="0" axisZ="1" />
<scale X="2.4" Y="2.4" Z="2.4"/>
<models>
 <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
</models>
<!--Titã-->
<group>
 <translate X="-0.9" Y="1.2" />
 <rotate angle="0.4" axisX="0" axisY="0" axisZ="1" />
 <scale X="0.15" Y="0.15" Z="0.15" />
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
</group>
</group>
```

```
<!--Urano-->
<group>
<translate X="45" />
<rotate angle="-97.77" axisX="0" axisY="0" axisZ="1" />
<scale X="1.9" Y="1.9" Z="1.9"/>
<models>
 <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
</models>
<!--Umbriel-->
<group>
 <translate X="-1.2" Y="1.2" />
 <rotate angle="0.2" axisX="0" axisY="0" axisZ="1" />
 <scale X="0.18" Y="0.18" Z="0.18" />
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
</group>
</group>
<!--Neptuno-->
<group>
<translate X="56" />
<rotate angle="-28.32" axisX="0" axisY="0" axisZ="1" />
<scale X="1.5" Y="1.5" Z="1.5"/>
<models>
 <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
</models>
<!--Tritão-->
<group>
 <translate Y="1.62" Z="-1.23" />
 <rotate angle="157" axisX="0" axisY="0" axisZ="1" />
 <scale X="0.14" Y="0.14" Z="0.14" />
 <models>
  <model file="/home/antonio/Desktop/TrabM2/Generator/cmake-build-debug/sphere.3d" />
 </models>
</group>
</group>
</scene>
```

Figure 4: Ficheiro XML

4 Prints do modelo

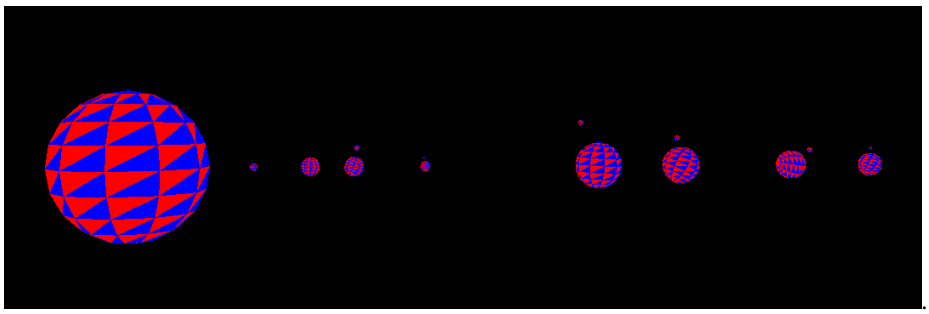

Figure 5: Modelo do Sistema Solar

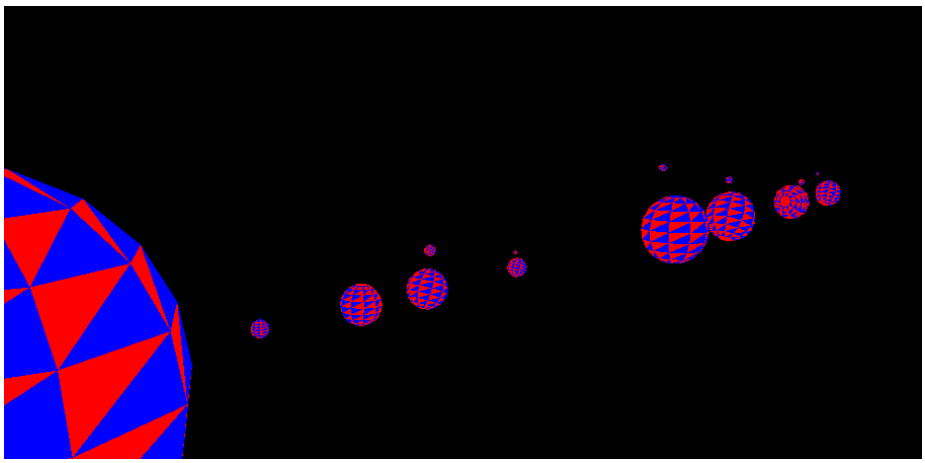

Figure 6: Modelo visto em perspetiva

5 Conclusão

Conseguimos ultrapassar alguns dos nossos problemas que foram aparecendo ao longo do trabalho, como a falha ao efetuar o load do ficheiro sphere.3d, que tem de ser feito com o path direto para o ficheiro, ou os problemas que tínhamos inicialmente com os movimentos da câmara. No entanto, pensamos que ainda existem alguns detalhes que poderão ser alterados para as futuras entregas de forma a tornar o modelo mais realista e mais eficaz.

Esta fase do projeto ajudou-nos a aprofundar ainda mais nosso conhecimento de C++, para além de nos obrigar a utilizar outras funcionalidades do GLUT que não tinham sido abordadas no trabalho anterior, como o movimento da câmara com o rato, algo que já tínhamos tentado realizar na fase anterior mas que não conseguimos completar.

Concluindo, acreditamos ter atingido o objetivo desta fase, que seria conseguir recriar um modelo do Sistema Solar aproximado ao real a partir da informação contida no ficheiro XML.