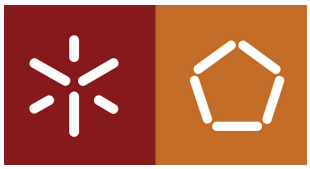
Universidade do Minho

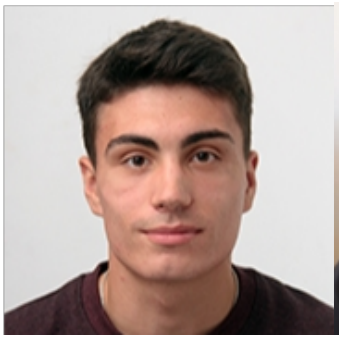
Departamento de Informática



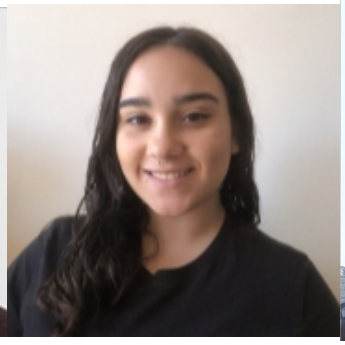
licenciatura em engenharia informática

# Computação gráfica - 2ª fase

Grupo 38



Henrique Pereira



Mariana Morais

Ana João

Simão

Henrique Morais Pereira A100831
Mariana Filipa Morais Gonçalves A100662
Ana João A95128
Simão A100597

Março de 2024

# Conteúdo

# 1 Introdução

A unidade curricular de Computação Gráfica propõe, recorrendo à biblioteca GLUT, a utilização do OpenGL para a construção de modelos 3D com recurso à linguagem de programação C++. Pretende-se, assim, que o relatório sirva de suporte ao trabalho realizado para esta fase, mais propriamente, dando uma explicação e elucidando o conjunto de decisões tomadas ao longo da construção de todo o código fonte e descrevendo a estratégia utilizada para a concretização dos principais objetivos propostos.

Com a criação dos modelos propostos no enunciado pretendemos consolidar os conceitos abordados nas aulas, mais propriamente transformações geométricas.

Posto isto, temos de forma muito breve o assunto do presente relatório que se refere à fase 2 de um projeto dividido em 4 fases, tal como indicado pela equipa docente.

# 2 Descrição do Problema

## 2.1 Fase 2

A segunda fase deste projeto tem como objetivo o desenvolvimento de novas funcionalidades aplicadas ao Motor desenvolvido anteriormente. Este deve agora processar transformações geométricas, tais como translações, escalas e rotações, descritas num ficheiro XML de configuração. Este ficheiro corresponde à representação estática de um modelo do sistema solar.

### 2.1.1 Engine

O *Engine* tem como requisito os seguintes pontos:

- Ler o ficheiro XML onde estão os modelos das primitivas.
- Armazenar as transformações e os modelos de cada grupo em memória.
- Desenhar, utilizando triângulos, as primitivas relativas aos modelos desejados em cada grupo, com as transformações associadas.

### 2.1.2 Modelo do Sistema Solar

O *Modelo do Sistema Solar* é um modo estático que representa o sistema solar que servirá como base para o resultado final do projeto. Para tal serão utilizadas primitivas como a esfera e o torus para a estruturação de planetas e satélites. Os planetas, o sol e a lua serão desenvolvidos com esferas de diferentes dimensões, sendo os anéis de Saturno representados por dois torus de dimensões diferentes. Para tal servimo-nos de uma escala relativa, priorizando a legibilidade e a Ênfase em certos elementos.

# 3 Resolução do Problema

Visto que nesta fase os ficheiros XML possuem novas propriedades comparativamente à fase anterior, foi necessário não só reestruturar o nosso parser, mas também as estruturas que armazenam informações relevantes para o desencadeamento da criação dos novos modelos.

A estrutura Transformation é responsável por representar uma transformação geométrica. Esta é composta por 4 variáveis para armazenar os valores necessários para realizar qualquer tipo de transformação.

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

Figura 1: Struct Transformation

A estrutura Transformations é responsável por representar um conjunto de transformações aplicadas a um grupo (só pode existir uma transformação de cada tipo).

```
struct Transformations {
    Transformation scale;
    Transformation translate;
    Transformation rotate;
};
```

Figura 2: Struct Transformations

A estrutura Group é responsável por representar um grupo existente no ficheiro XML. Este é composto por um conjunto de transformações, um vetor para armazenar os modelos, outro para armazenar os pontos lidos dos modelos e mais um para guardar os subgrupos presentes. Desta forma, as transformações aplicadas ao grupo principal serão aplicadas aos seus subgrupos.

```
struct Group {
    Transformations transformations;
    vector<string> models;
    vector<Point> points;
    vector<Group> subGroups;
};
```

Figura 3: Struct Group

### 3.0.1 Leitura do ficheiro XML

Para efetuar o parsing continuamos a utilizar o (*TinyXML-2*). No entanto, para esta fase, é importante perceber que o foco principal não é decifrar as primitivas a desenhar, mas sim entender a constituição de cada grupo existente no ficheiro XML, passado como argumento ao Engine.

- **initTransformations**:

  É responsável por inicializar uma estrutura Transformations com todos os valores a 0 exceto os da escala que têm de ser inicializados a 1, uma vez que da maneira como o projeto está estruturado iríamos aplicar à primitiva uma escala 0.

- **parseGroups**:

  É a função crucial desta fase do projeto. Cria uma nova estrutura Group e preenche os respetivos campos, retornando-a no final. A nossa abordagem para tratar dos subgrupos utilizou um ciclo while de maneira a iterar sobre os mesmos recursivamente através da chamada da própria função sobre cada subgrupo e adicionando o resultado ao vetor.

- **loadModels**:

  Percorre os modelos de um grupo e armazena o conjunto de pontos lidos num vector<Point>points e devolve-o.

#### 3.0.2 Desenho das primitivas

Surgindo a necessidade de desenhar as primitivas, cujos vértices foram carregados para memória, criou-se a função drawGroups.

A função drawGroups percorre o vetor mainGroups e, para cada grupo encontrado, faz push das transformações geométricas com o auxílio das funções glPushMatrix, glTranslatef, glScalef e glRotatef. Para além de realizar as transformações necessárias, desenha os triângulos relativos aos pontos presentes no vetor points desse grupo, recorrendo à função drawPoints. Antes de fazer glPopMatrix sâo também desenhadas as primitivas dos seus subgrupos, invocando novamente drawGroups recursivamente. Desta forma, os subgrupos herdam as transformações do grupo no qual estão contidos.

#### 3.0.3 Modelo do Sistema solar

No modelo do sistema solar, descrito no ficheiro XML, serão representados o Sol, os 9 planetas(Mercúrio, Vénus, Terra, Marte, Júpiter, Saturno, Urano, Neptuno e Plutão), a lua e os anéis de Saturno. Para esta tarefa, as primitivas utilizadas serão a esfera e o torus.

Visto que estas primitivas possuem uma dimensão e uma posição fixas, é necessário recorrer a transformações para alterar a orientação, dimensão e localização destas relativamente a um referencial. Assim, é necessário recorrer a escalas, rotações e translações de modo a desenhar estas figuras o mais próximo da realidade possível. No entanto, devido a existir imensa discrepância entre as dimensões reais dos diferentes elementos do sistema, não foi possível utilizar uma escala completamente de acordo com a real, tendo sido necessário realizar pequenas aproximações de modo a permitir visualizar decentemente todas as figuras.

Relativamente aos anéis de Saturno, embora estes sejam um extenso sistema de anéis, serão representados apenas por 2 torus uma vez que a nosso ver é a forma que mais se assemelha ao visivelmente desejado.

É necessário, ao desenvolver o modelo, ter em atenção o facto de que as primitivas de um subgrupo herdam as transformações do grupo em que estão inseridas. Isto faz com que:

- O translate de um grupo seja somado ao translate do subgrupo;
- O scale de um grupo multiplique o valor do scale do subgrupo;
- O valor do scale do grupo multiplique o valor do translate no subgrupo;
- O valor do rotate do grupo seja somado ao rotate do subgrupo.

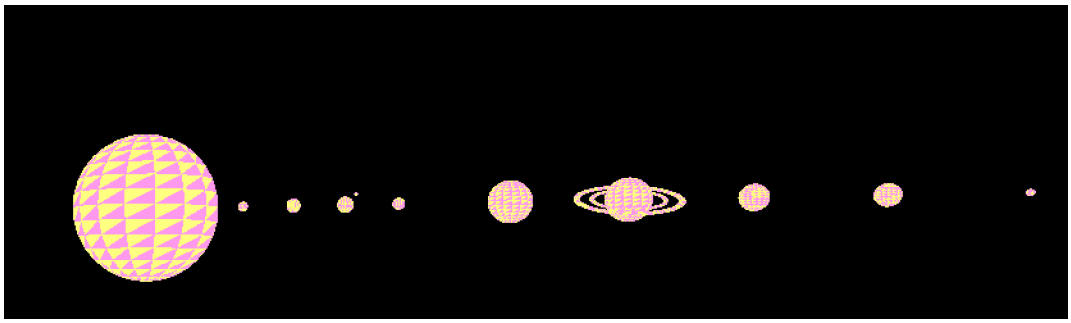


Figura 4: Modelo do Sistema Solar

### 3.0.4 Modelo do Sistema solar com escalas

Após criar um sistema solar com as componentes que desejávamos, de modo a pormenorizar o nosso sistema solar, testamos escalas e parâmetros que seriam interessantes para a UC. Começamos por investigar a distância dos planetas ao sol. Inicialmente quisemos aplicar a escala de 1 unidade a representar 57,9 milhoes de quilómetros que é a escala usada até Marte. Após este planeta, a nosso ver, seria interessante mudar a escala visto que há uma grande diferença de valores que iriam fazer com que o nosso sistema solar não fosse agradável de explorar. De seguida, pelos mesmos motivos, é aplicada outra escala, 1 unidade para 250 milhões de quilómetros. Após isto, temos a coordenada x da posição dos nossos planetas conforme a escala usada.

| | Distância ao sol (Km) | Escala 1:57,9 milhões | Escala 1:155 milhões | Escala 1:250 milhões |
|---|---|---|---|---|
| Sol | - | | | |
| Mercúrio | 57,9 milhões | X=1 | - | - |
| Vénus | 108,2 milhões | X=1,869 | - | - |
| Terra | 149,6 milhões | X=2,584 | - | - |
| Marte | 227,9 milhões | X=3,936 | - | - |
| Júpiter | 778,9 milhões | - | X=5,025 | - |
| Saturno | 1429 milhões | - | X=9,219 | - |
| Urano | 2871 milhões | - | - | X= 11,484 |
| Neptuno | 4495 milhões | - | - | X=17,980 |
| Plutão | 5906 milhões | - | - | X=23,624 |

Uma vez que não quisemos deixar o nosso sistema por aqui, exploramos também a relação de tamanho entre eles, ou seja, o raio. A escala que a nosso ver era pertinente para o sol, seria de 0,8. Após fixar este valor, vimos a relação entre o raio do sol e dos planetas e aplicamos à escala desejada. Ou seja, O sol tem raio de 696340 km e Mercúrio tem 2440 km de raio, ou seja $\frac{RaioSol}{RaioMercúrio} = 285,39$ logo Mercúrio é 285,39 vezes menor que o Sol. Dito isto, se a escala do Sol é 0,8 mercúrio teria de ser $\frac{0,8}{285,39} = 0,0028$. Isto para se cumprir efetivamente a proporção dos planetas. Contudo, isto acaba por fazer com que o nosso sistema seja de difícil compreensão e desagradável de manipular, pois falamos de algo grandioso reduzido a muito pouco. Então, após aplicar uma escala totalmente proporcional, de modo a melhorar o aspeto/desempenho, testamos algumas manipulações de escalas que seriam úteis e credíveis. Deste estudo temos o seguinte aproveitamento:

| | Raio(Km) | Escala | Escala*10 | Escala*10*1.5 |
|---|---|---|---|---|
| Sol | 696340 | 0,8 | - | - |
| Mercúrio | 2440 | 0,0028 | 0,028 | - |
| Vénus | 6052 | 0,0070 | 0,070 | - |
| Terra | 6371 | 0,0073 | 0,073 | - |
| Marte | 3389 | 0,0039 | 0,039 | - |
| Júpiter | 69911 | 0,0803 | 0,803 | 0,535 |
| Saturno | 58232 | 0,0669 | 0,669 | 0,446 |
| Urano | 25362 | 0,0291 | 0,291 | 0,194 |
| Neptuno | 24622 | 0,0283 | 0,283 | 0,189 |
| Plutão | 2377 | 0,0027 | 0,027 | - |

Desta tabela, são usados os últimos valores calculados para cada componente.

Após isto e alguma análise do sistema solar e do que gostaríamos de representar decidimos implementar as rotações dos planetas. Para isto aplicaremos um rotate sobre o eixo Z com o valor do ângulo de cada inclinação. Como este ângulo é positivo para o lado esquerdo, iremos aplicá-lo com valor negativo para se verificar a rotação para o lado direito. Isto para os planetas com uma rotação normal, para os que apresentam rotação retrógrada, será aplicada a rotação positiva.

| | Inclinação | Rotate |
|---|---|---|
| Mercúrio | 7 | -7 |
| Vénus | 177 | 177 |
| Terra | 23,5 | -23,5 |
| Marte | 25 | -25 |
| Júpiter | 3 | -3 |
| Saturno | 27 | -27 |
| Urano | 98 | 98 |
| Neptuno | 30 | -30 |
| Plutão | 120 | -120 |

Dada a tabela acima, podemos concluir que Vénus e Urano apresentam uma rotação retrógrada tal como prevista pela explicação acima.

Os anéis de Saturno acabaram por não ser escalonados uma vez que não encontramos informação para tal, a lua embora seja também representada não está conforme os padrões reais do sistema solar.

## 3.1 Testes

De forma a testar todo o trabalho elaborado, utilizamos os ficheiros disponibilizados na pasta de testes relativamente à fase 1 do projeto. Como é possível observar, todas as cenas estão idênticas ao esperado, o que consideramos que tenha sido um sucesso.

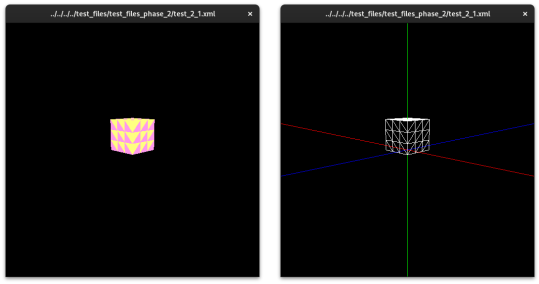


Figura 5: Ficheiro test_2_2.xml

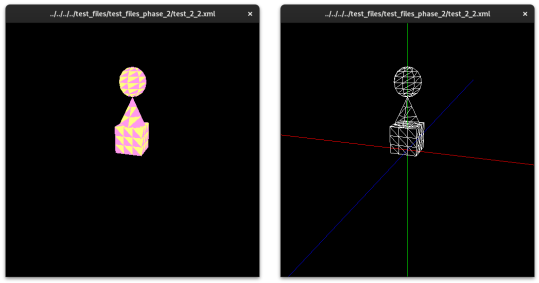


Figura 6: Ficheiro test_2_2.xml

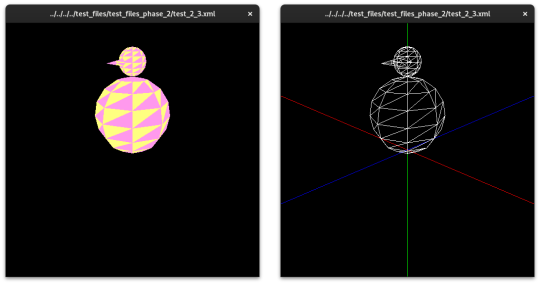


Figura 7: Ficheiro test_2_3.xml

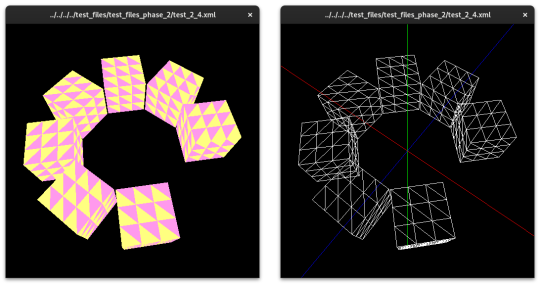


Figura 8: Ficheiro test_2_4.xml

# 4 Conclusão

Fazendo uma retrospetiva da fase consideramos que foi um sucesso uma vez que cumprimos todos os requisitos estabelecidos.

A realização desta segunda fase foi crucial para estabelecer uma base sólida para o restante desenvolvimento do projeto. Agora, com uma compreensão mais profunda das bibliotecas e ferramentas utilizadas, acreditamos que estamos numa posição mais vantajosa para avançar para a próxima fase.

A familiaridade adquirida com as ferramentas, uma melhor compressão de funções e a linguagem de programação durante a fase dois simplificará a execução da próxima tarefa. Isso porque consideramos que estamos mais confortáveis com os procedimentos e métodos necessários para manipular geometrias e implementar as transformações desejadas.