

Universidade do Minho

Departamento de Informática

**Computação Gráfica**

Grupo nº 5

*Mafalda Costa (a83919)*



Carlos Preto (a89587)



Pedro Veloso (a89557)



*Simão Monteiro (a85489)*



***Índice***

[Introdução 3](#_Toc68079902)

[Leitura dos ficheiros 4](#_Toc68079903)

[Registo em memória 4](#_Toc68079904)

[Desenho dos triângulos 5](#_Toc68079905)

[Interação com o utilizador 6](#_Toc68079906)

[Sistema Solar 8](#_Toc68079907)

[Mercúrio 9](#_Toc68079908)

[Vênus 9](#_Toc68079909)

[Terra e Lua 10](#_Toc68079910)

[Marte 10](#_Toc68079911)

[Júpiter 10](#_Toc68079912)

[Saturno 11](#_Toc68079913)

[Úrano 11](#_Toc68079914)

[Neptuno 11](#_Toc68079915)

[Conclusão 12](#_Toc68079916)

[Bibliografia 13](#_Toc68079917)

# Introdução

No âmbito da UC de Computação Gráfica da Universidade do Minho foi proposto o desenvolvimento de uma simulação de um Sistema Solar.

Nesta segunda fase este Sistema começa a ganhar forma através de algumas transformações geométricas e tendo em conta um modelo hierárquico. Este modelo hierárquico ganha a forma de uma árvore, da qual cada nodo contém uma série de transformações geométricas modelos e nodos filhos dependentes das transformações dos nodos pais.

O Sistema Solar estático desenvolvido nesta fase permite a visualização do Sol, dos planetas, das suas órbitas e respetivas luas.

# Leitura dos ficheiros

Para a realização desta fase do trabalho, continua a ser necessário realizar a leitura do ficheiro XML e dos ficheiros 3D. Contudo, ao contrário da fase 1, é possível realizar translações, rotações e escalas sobre os vários objetos 3D, que constam no ficheiro. Para a realização destas operações é indispensável que os objetos estejam dentro de um grupo, sobre o qual serão aplicadas.

Considerando, a estrutura do ficheiro XML, dividiu-se a sua leitura em 2 fases: fase de leitura do ficheiro 3D e a fase de leitura dos dados do grupo.

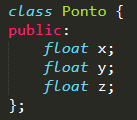
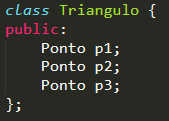
Na fase de leitura do ficheiro 3D, não se realizaram alterações, sendo que a leitura é constituída por determinar primeiramente o número de pontos no ficheiro, seguida da leitura dos pontos.

Na fase de leitura de um grupo, verifica-se a existência de translações, rotações e escalas, que em caso de existência devem ser lidas e registadas em memória. Também é registado em memória, nesta fase, a ordem em que as operações aparecem. Posteriormente, é analisada a presença de modelos no grupo, que uma vez mais devem ser lidos e associados ao grupo. Por fim, uma vez que um grupo pode conter subgrupos é realizada a função de forma recursiva, enquanto existirem subgrupos.

# Registo em memória

Após a leitura dos pontos é necessário registá-los em memória, possibilitando o desenho dos triângulos, de uma forma mais simples e rápida.

Como, nesta segunda fase, um modelo 3D continua a ser representado por triângulos, tendo associado a cada triângulo uma cor, as classes ponto, triangulo e modelo não sofreram alterações, sendo representadas da seguinte forma:

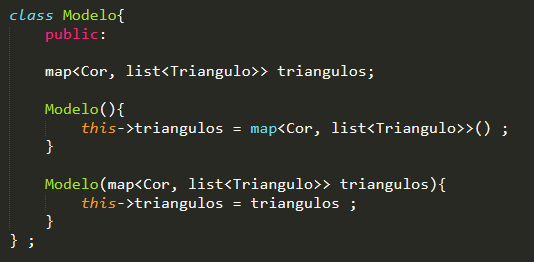


Figura 1: Classe Ponto/Triângulo/Modelo

Após o registo de um modelo, é necessário ler os dados referentes às suas operações na cena – translação, rotação e escala. Assim criou-se a classe “Group”, onde são registados os dados associados ao grupo. A classe, contém, como se observa na figura 2, uma lista com translações, rotações e escalas, para além dos seus modelos e subgrupos.

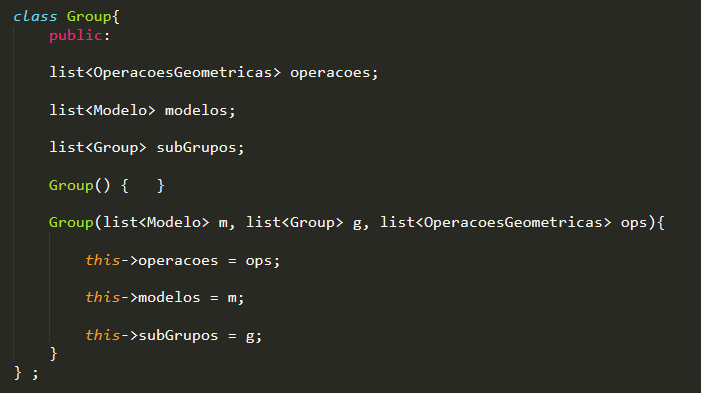


Figura 2: Classe Group

Por fim, uma vez que uma cena contém um grupo principal responsável por conter todos os modelos, regista-se esse grupo na classe “*Scene*”. É na classe “*Scene”* onde se encontra a função *load* responsável por ler um ficheiro em formato XML e povoar todas as estruturas de dados acima referidas.

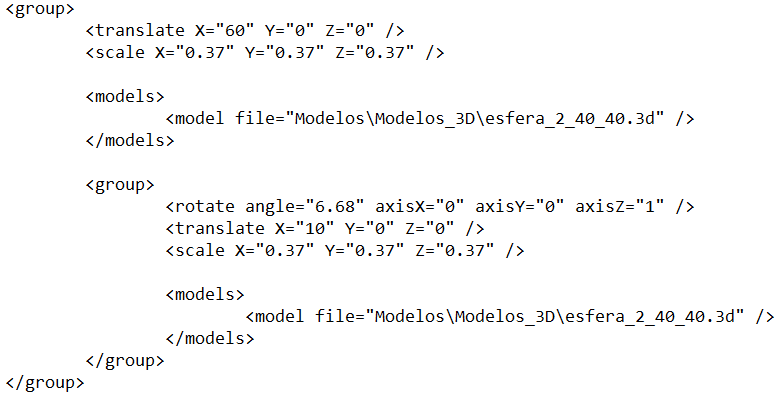
# Desenho dos triângulos

Para desenhar os triângulos no ecrã é necessário percorrer a estrutura de dados e recorrendo à função “glVertex3f” desenhar os triângulos.

Uma vez que, as operações – translação, rotação e escalas – apenas se aplicam a elementos dentro do grupo, devendo ser realizadas quando entramos no grupo e desfeitas quando saímos. Para desfazer as operações, em vez de calcular o inverso das operações, opta-se por retirar de uma *stack* a matriz existente antes da entrada no grupo, recorrendo à função “glPopMatrix()”. Assim é também é necessário colocar a matriz atual, na *stack*, quando é realizada uma entrada num grupo, recorrendo à função “glPushMatrix()”.

Assim para o exemplo apresentado na figura 3, é realizado uma amostra do que acontece no algoritmo desenvolvido.

Primeiramente é realizado um *push* da matriz atual, para a *stack* que passa a conter a matriz 1. Posteriormente são realizadas todas as operações, alterando a matriz, e desenha-se o modelo considerando essas operações. Como existem subgrupos deve-se voltar a chamar a função de uma forma recursiva, voltando a realizar a operação de *push*, o que leva a *stack* a ter registadas as matrizes 1 e 2. Após realizar as operações do subgrupo e desenhar o modelo, deve-se sair da função recursiva voltando à função que a chamou (função pai). Contudo, é necessário registar um *pop* da matriz, de modo a obter-se a matriz existente antes da entrada na função recursiva (matriz 2). Saindo da função pai, deve voltar a realizar-se, uma vez mais, um *pop*, obtendo como matriz a matriz 1 da *stack*.



1111

2111

2

1

Figura 3: Exemplo de pushMatrix/popMatrix num XML

Com a utilização deste algoritmo, é possível iterar sobre grupos realizando as suas operações e à saída voltar a ter como matriz de transformações uma matriz igual à de entrada.

# Interação com o utilizador

Durante a realização desta fase foram implementados comandos que permitem interagir com as renderizações obtidas. É possível realizar translações, rotações e escalas no modelo completo, bem como modificar a forma de visualização dos modelos.

Assim a tabela seguinte representa as funcionalidades, de interação com o modelo, associadas a cada tecla.

Tabela 1: Associação de funcionalidade-tecla

|  |  |
| --- | --- |
| **Tecla** | **Funcionalidade** |
| X | Define que a translação, rotação e escala será realizada considerando o eixo do X; |
| Y | Define que a translação, rotação e escala será realizada considerando o eixo do Y; |
| Z | Define que a translação, rotação e escala será realizada considerando o eixo do Z; |
| A | Diminui a coordenada selecionada. Como *default* utiliza a coordenada X; |
| D | Aumenta a coordenada selecionada. Como *default* utiliza a coordenada X; |
| I | Define a visualização como linhas; |
| O | Define a visualização como pontos; |
| P | Define a visualização como sólido; |
| Q | Aumenta a escala em função do eixo selecionado. Como *default* utiliza o eixo do X; |
| E | Diminui a escala em função do eixo selecionado. Como *default* utiliza o eixo do X; |
| W | Aumenta o ângulo de rotação em torno do eixo selecionado. Como *default* utiliza o eixo do X; |
| S | Diminui o ângulo de rotação em torno do eixo selecionado. Como *default* utiliza o eixo do X; |
| T | Desenha/Apaga as orbitas dos planetas. |
| R | Desenha/Apaga as orbitas os eixos do referencial. |

Além da iteração direta com o modelo renderizado, através das várias operações, também é possível interagir com a câmara.

A interação com a câmara pode ser realizada através do modo Explorador ou do modo FPS. Para realizar a troca de modos é utilizada a tecla 0.

No modo de exploração utilizando o rato é, então possível interagir com a câmara. Assim temos:

Tabela 2: Modo Explorador

|  |  |
| --- | --- |
| **Comando** | **Funcionalidade** |
| Botão esquerdo do rato + movimento | Movimenta a câmara, mantendo o foco no ponto (0,0,0). |
| Botão direito do rato + movimento positivo no eixo do Y (cima) | Diminui a distância da câmara ao ponto (0,0,0). |
| Botão direito do rato + movimento negativo no eixo do Y (baixo) | Aumenta a distância da câmara ao ponto (0,0,0). |
| 1 | Diminui a distância ao centro da origem para 15. |

No modo FPS utiliza-se o rato, para orientar a vista, e o teclado, para mudar a posição da câmara. Assim temos:

Tabela 3: Modo FPS

|  |  |
| --- | --- |
| **Comando** | **Funcionalidade** |
| Botão esquerdo do rato + movimento | Movimenta a direção que a câmara aponta. |
| KEY RIGHT | Avança para a direita. |
| KEY LEFT | Avança para a esquerda. |
| KEY UP | Avança na direção do olhar. |
| KEY DOWN | Recua na direção do olhar. |

# Sistema Solar

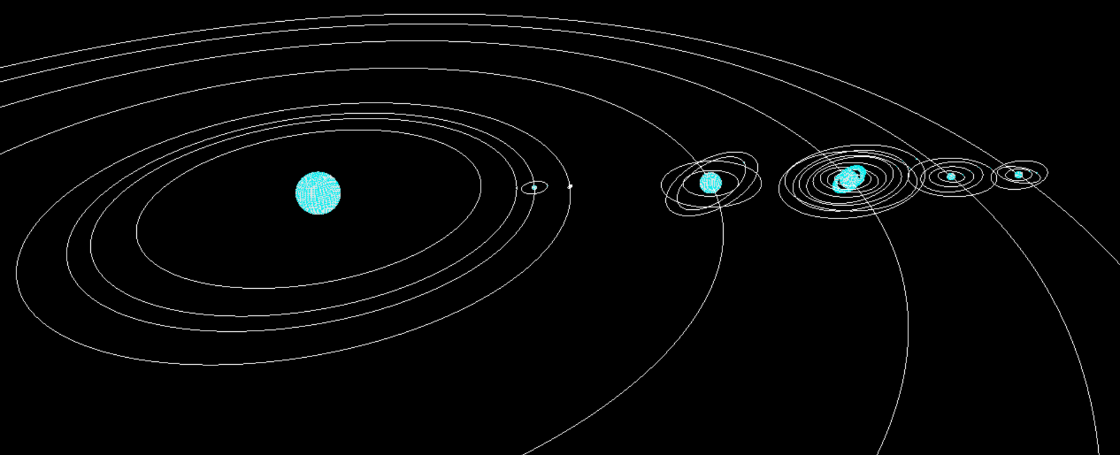
O sistema solar, formado há cerca de 4,66 mil milhões de anos atrás é constituído por 8 planetas – Mercúrio, Vênus, Terra, Marte, Júpiter, Saturno, Urano e Neptuno.

Para a realização deste trabalho desenvolveu-se um ficheiro XML, que lido pela *engine* permite criar uma representação abstrata, em 3D, do sistema solar.

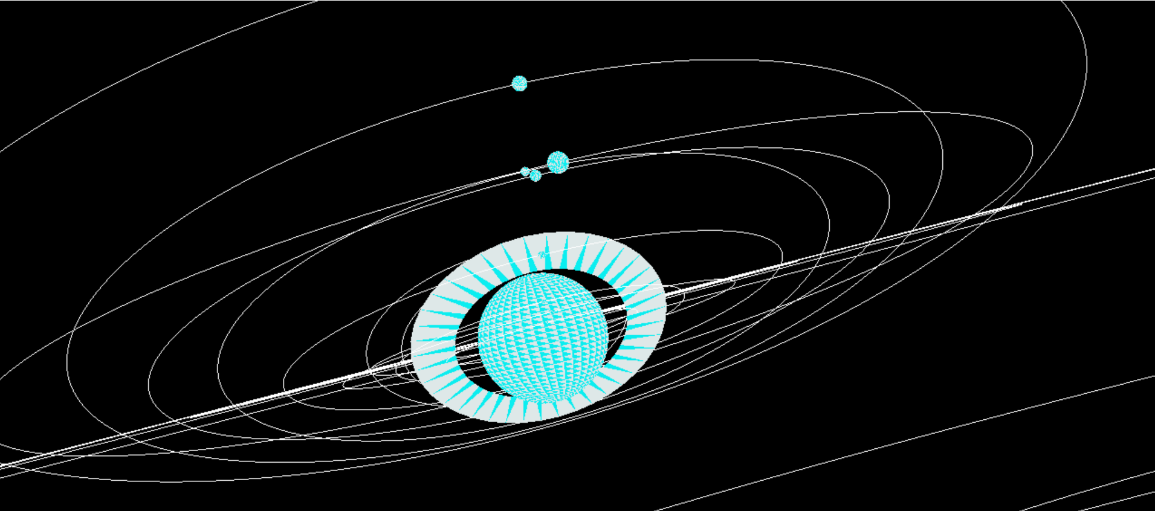
Para a construção do ficheiro XML optou-se por construir o sol e os planetas à custa do ficheiro “esfera\_2\_40\_40.3d”, isto é, todas as escalas que são feitas para a construção de cada planeta são calculadas em relação à esfera que este ficheiro produz. As translações são calculadas a partir da origem do eixo xyz. Pelo mesmo raciocínio construíram-se as luas de cada planeta, sendo que neste caso os cálculos dimensionais são efetuados a partir do planeta que estas orbitam, pois, as luas são subgrupos do grupo que constrói cada planeta. Para todos os planetas e luas desenharam-se também as suas órbitas. Para esta construção teve-se em conta que nem todos os planetas realizam a mesma órbita e por isso fazem-se algumas rotações no ficheiro XML para ajustar o ângulo da órbita em relação ao planeta. No caso de saturno, foi necessário construir o seu famoso anel que se pode encontrar no ficheiro “anelsaturno.3d”. Dado o elevado número de luas de alguns planetas e visto que o processo de construção das luas é semelhante para todas elas, para que o ficheiro XML não ficasse demasiado extenso, decidiu-se apenas construir aquelas que se consideraram mais relevantes, como forma de exemplo. Tentou-se que a escala fosse o mais realistas possível, no entanto, as proporções reais não permitiriam a visualização de alguns planetas. Desta forma, surgiu a necessidade de alterar a escala para aquela que é uma proximidade muito imperfeita da realidade, mas percetível a olho nu. Para a elaboração desta escala não houve nenhum tipo de precisão, olhando para algumas imagens que foram encontradas do Sistema Solar na Internet, foi-se testando e ajustando os valores *translate* do XML.



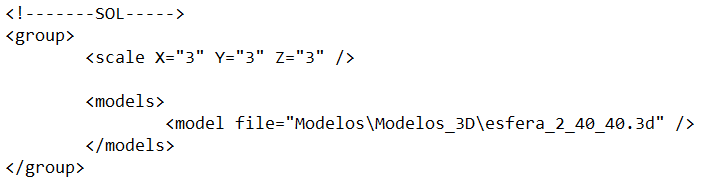
*Figura 4: Exemplo de uma imagem do Sistema Solar completo*



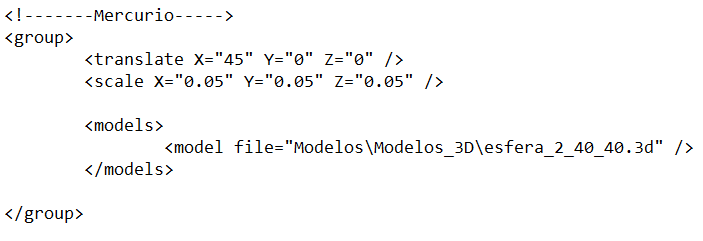
*Figura 5: Sistema Solar desenvolvido*

*Figura 6: Saturno e as respetivas luas*

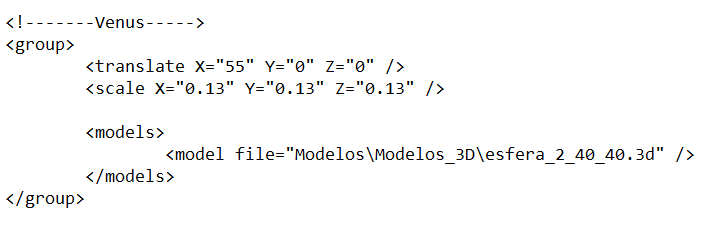
**Sol**



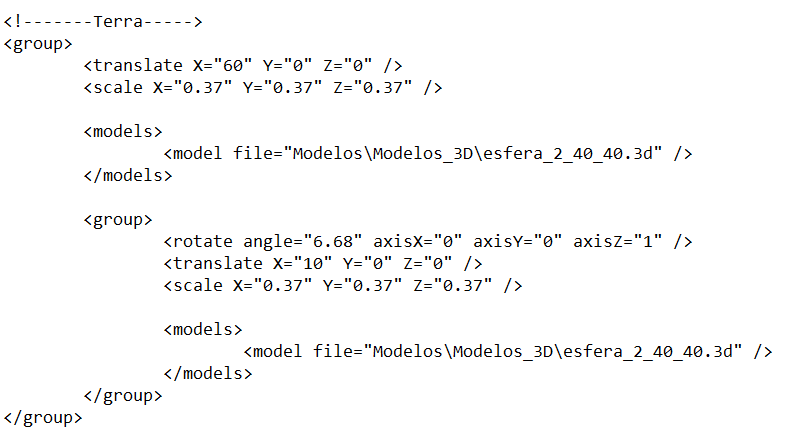
## Mercúrio



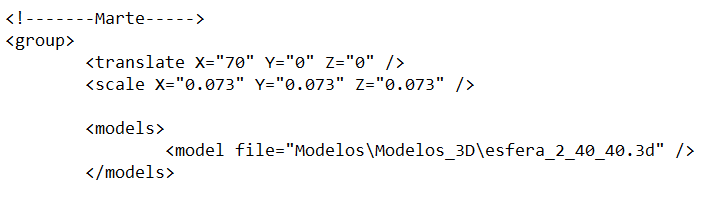
## Vênus



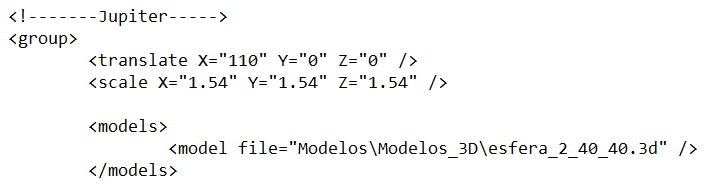
## Terra e Lua



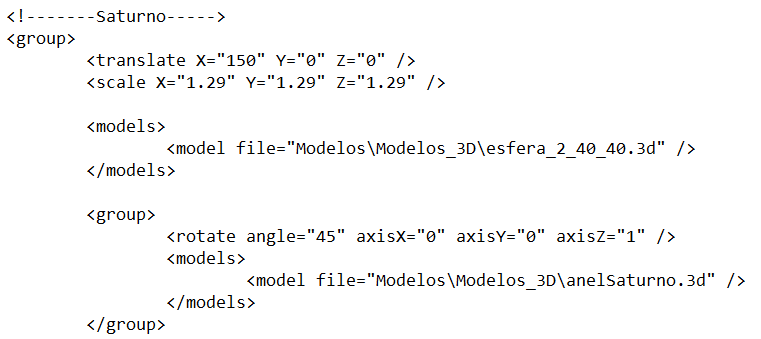
## Marte



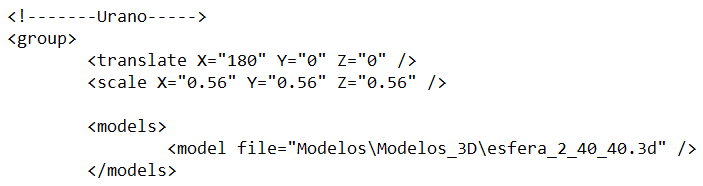
## Júpiter



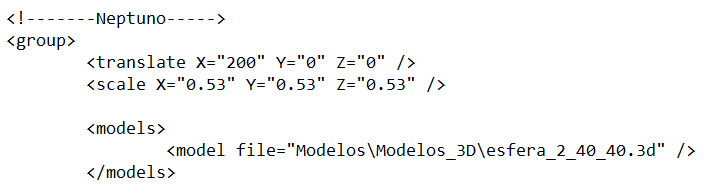
## Saturno



## Úrano



## Neptuno



# Conclusão

Em forma de conclusão, o grupo sente que conseguiu atingir os objetivos propostos com sucesso e sem consideráveis dificuldades.

Foi criada uma representação do Sistema Solar semelhante, a olho nu, com a realidade e ainda, adicionado o modo FPS à aplicação, que vem facilitar a visualização do sistema desenvolvido.

Para além disso, esta fase permitiu uma melhor aprendizagem através da consolidação de alguns conceitos abordados nas aulas de Computação Gráfica. Uma grande motivação que sentimos, para trabalho futuro, é o facto de o Sistema Solar começar a ganhar forma.

# Bibliografia

* [TinyXML-2](http://www.grinninglizard.com/tinyxml2/index.html): [*www.github.com/leethomason/tinyxml2/releases/tag/8.0.0*](http://www.github.com/leethomason/tinyxml2/releases/tag/8.0.0)
* http://www1.ci.uc.pt/iguc/