Computação Gráfica
Fase 2

Universidade do Minho
Escola de Engenharia

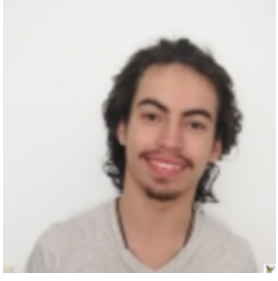
Trabalho realizado por:

Afonso Laureano Barros Amorim
Gonçalo Martins dos Santos
João Carlos Fernandes Novais
Telmo José Pereira Maciel

Grupo 26

A97569
Afonso Amorim

A95354
Gonçalo Santos



A96626
João Novais

A96569
Telmo Maciel

Índice

1 Introdução

Na segunda fase do trabalho tínhamos como objetivo melhorar o que foi feito na fase anterior tornando possível usando transformações geométricas como: translações, rotações e escalamentos.

2 Alterações na arquitetura do projeto

Nesta parte do relatório, vamos apresentar o que foi alterado na arquitetura do projeto para satisfazer o que era pedido no enunciado.

2.1 Generator

O generator sofreu uma alteração pequena, a inserção de uma nova primitiva chamada **Washer** (vamos abordar esta nova primitiva mais detalhadamente na próxima secção).

2.2 Engine

No engine, mudamos a maneira como apresentamos os modelos no ecrã, agora desenhamos e aplicamos transformações aos modelos de maneira hierárquica.

Para além disto, adicionamos a funcionalidade que permite ao utilizador selecionar um elemento no ecrã , obtendo uma resposta do engine que diz qual foi o nome do modelo selecionado.

2.3 Parser

No parser, adicionamos dois parâmetros:

- **Groups**: que serve para guardar os grupos lidos do ficheiro XML;
- **AllModels**: que serve para guardar todos os modelos carregados de ficheiros XML.

Para além disto, ainda adicionamos a possibilidade de dar um nome a um modelo e adicionamos a transformação color para mudar a cor de um elemento.

2.4 Shape.cpp

Neste ficheiro, adicionamos os atributos name e file que servem para guardar o nome e ficheiro onde está guardado o modelo.

2.5 Group.cpp

Nesta fase criamos a classe Group, usadada para guadar a informação de um elemento group do ficheiro XML, que permite gerar a hierarquia também proveniente do ficheiro XML.

Nesta classe, temos um vector de Transformation (transformations), vector de Shape (models) e vector de Group (groups).

- A variável transformations representa as transformações a aplicar;
- A variável models representa os modelos a desenhar;
- A variável groups representa os subgrupos.

2.6 Transformation.cpp

Nesta fase também criamos a classe Transformation que é usada para representar uma transformação, que pode ser uma translation, rotate, scale e color.

3 Primitivas adicionadas

Como já foi referido adicionamos a primitiva *Washer* (em português Arruela).

A maneira de a constuir é muito semelhante à de um cilindro, uma vez que esta é apenas um cilindro "oco" no centro. Ela necessita de 4 argumentos para ser construída: o raio interior, raio exterior, altura e número de fatias.

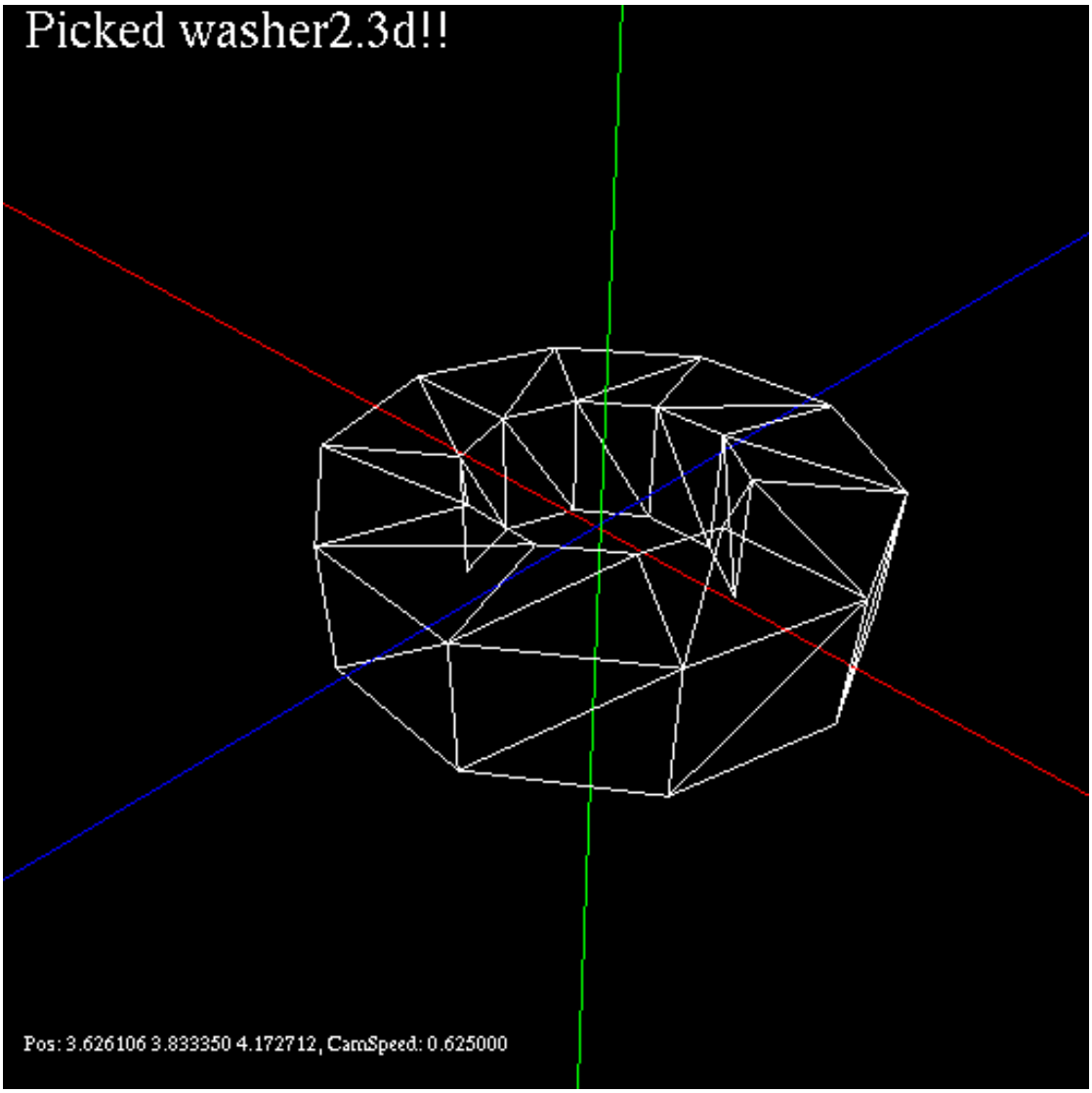


Figure 1: Exemplo de uma Arruela

4 Estrutura e parsing do ficheiro XML

4.1 Estrutura do ficheiro

O ficheiro XML tem um apenas um elemento **world** que é o elemento mais acima na hierarquia. Este elemento pode ter vários elementos group. O elemento group pode ser composto por:

- Elemento translate;
- Elemento rotate;
- Elemento scale;
- Elemento color;
- Elemento models;

4.1.1 Elemento Translate

Este elemento tem 3 atributos: x, y e z. Este elemento representa a translação dos modelos em relação ao eixo representado pelo x, y e z.

4.1.2 Elemento Rotate

Este elemento tem 4 atributos: angle, x, y e z. Este elemento representa uma rotação em angle graus em relação ao eixo formado pelos atributos x, y e z.

4.1.3 Elemento Scale

Este elemento tem 3 atributos: x, y e z. Este elemento representa a escala dos modelos em relação ao eixo especificado pelos atributos.

4.1.4 Elemento Color

Este elemento tem 3 atributos: r, g e b. Este elemento tem por base o formato RGB no qual o atributo r representa a intensidade da cor vermelha, o g reprensenta da cor verde e o b o da cor azul.

O nosso grupo achou por bem adicionar este elemento para poder ser mais fácil diferenciar os diferentes planetas do sistema solar no cenário final.

4.1.5 Elemento Models

Este elemento abriga vários elementos model. Cada um destes elementos representa um modelo que deve ser carregado para este grupo.

O elemento model apenas tem o atributo file que representa o nome do ficheiro que tem os pontos a serem carregados.

Para além de tudo isto, um elemento group pode ter outro elemento **group aninhado**.

Todos os elementos são opcionais.

4.2 Parsing do ficheiro

Para o parsing nesta fase apenas deixamos mais aprimorado o parsing que tínhamos já feito na última fase, tendo apenas adicionado o suporte para múltiplos grupos e para transformações.

A maneira como o nosso grupo implementou o parsing do ficheiro XML levanta algumas questões às quais temos que ter em atenção, essas questões são :

- Só lemos o primeiro elemento transform do ficheiro XML, ou seja, se tiver mais que 1 todos os outros são ignorados;
- Só lemos o primeiro elemento models do ficheiro, ignorando todos os outros.

5 Construção do cenário

5.1 Generator

Para executar a demonstração do Sistema Solar é necessário serem geradas 3 figuras:

1. sphere.3d, usado para representar os planetas. Deverá ser do tipo esfera e com raio 1

2. washer.3d, usado para representar as órbitas dos planetas. Deverá ser do tipo washer e com raio interior 1, raio exterior 1.001 e altura 0.0005 .

3. saturnRing.3d, usado para representar o anel de Saturno. Deverá ser do tipo washer e ter raio interior de 1, raio exterior de 1.2 e altura de 0.0005.

5.2 Engine

Para executar e visualizar o modelo na engine gráfica, apenas temos que passar como argumento para a mesma o nome do ficheiro é "sim.xml", sendo apresentado no terminal as instruções para a utilização da câmara em si.

5.3 Escala Utilizada

De forma a termos um modelo semi-realista mas ainda possível de computar e visualizar decidimos usar as seguintes escalas:

1. Para a escala dos planetas, decidimos usar como referência o tamanho de Mercúrio, ou seja, todos os planetas têm o seu tamanho em função do de Mercúrio, à exceção do Sol, para o qual usamos o tamanho de 60 ao invés de 277, de forma a tornar mais visível o resto dos elementos do sistema solar.

2. Para a distância dos planetas decidimos usar também como referência o tamanho de Mercúrio, mas ainda dividindo por 100, de forma a garantir que nenhum planeta está demasiado longe.

5.4 Sistema Solar

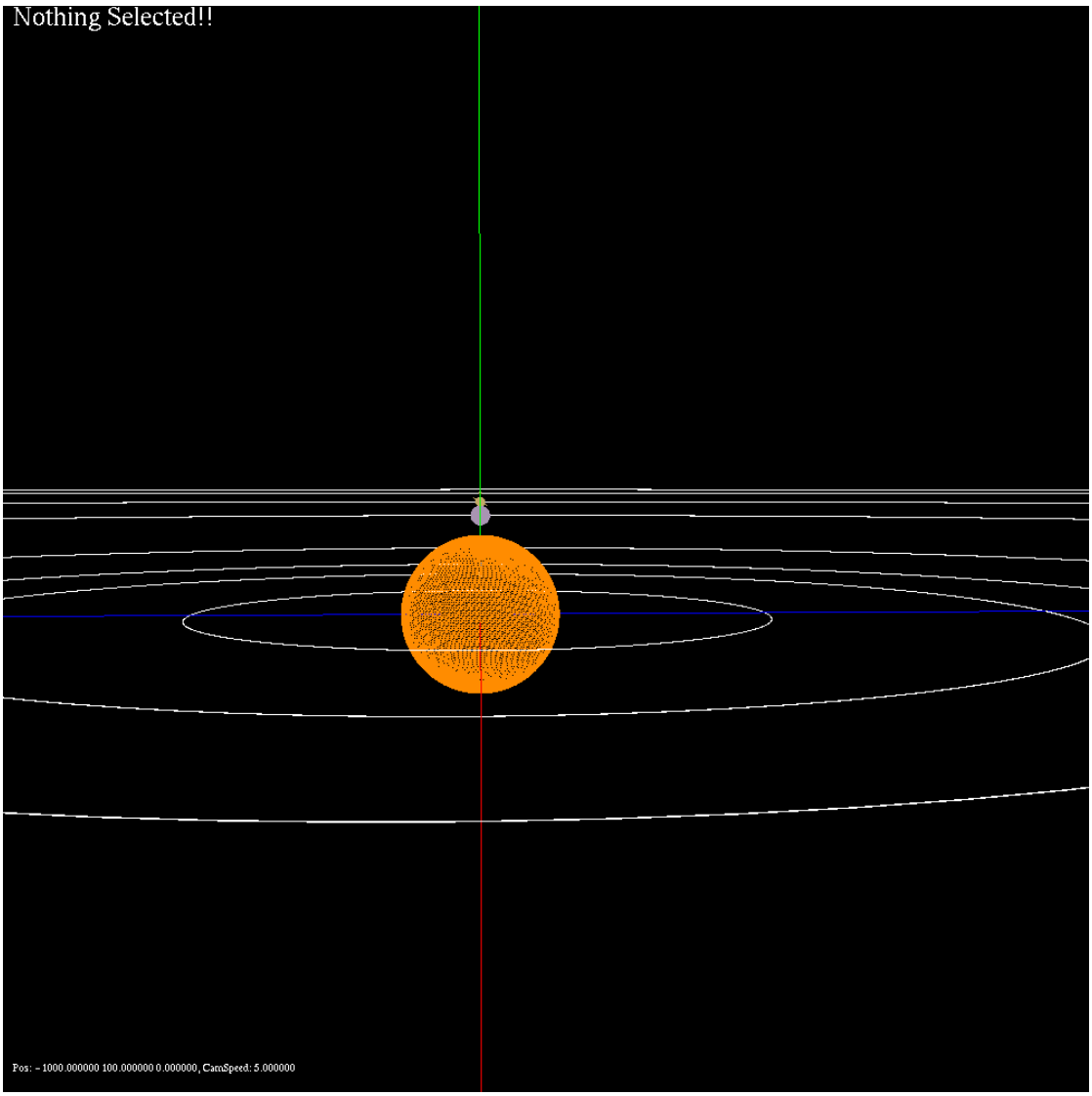


Figure 2: Sistema Solar numa vista Baixa

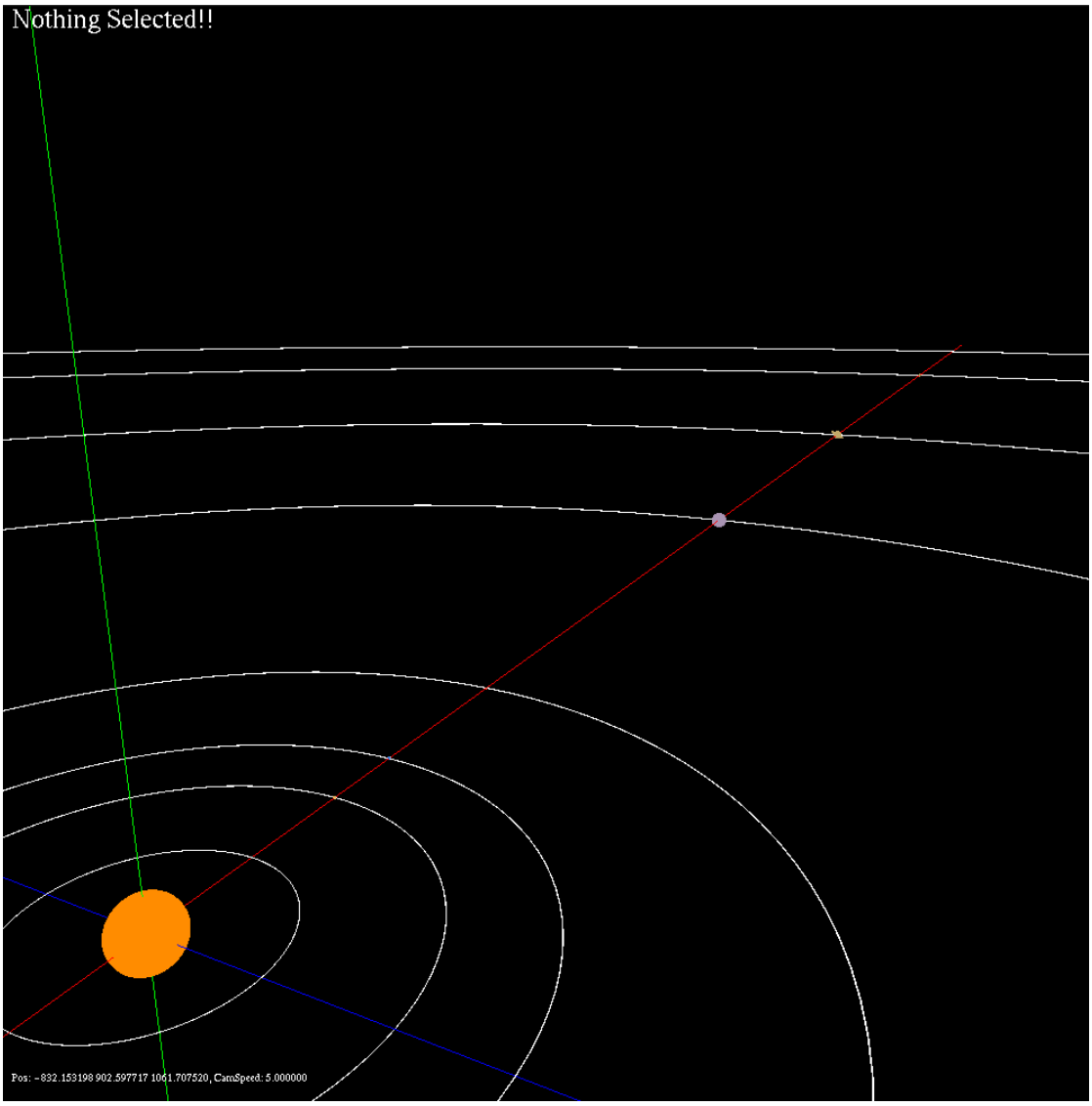


Figure 3: Sistema Solar numa vista Superior Perto do Sol

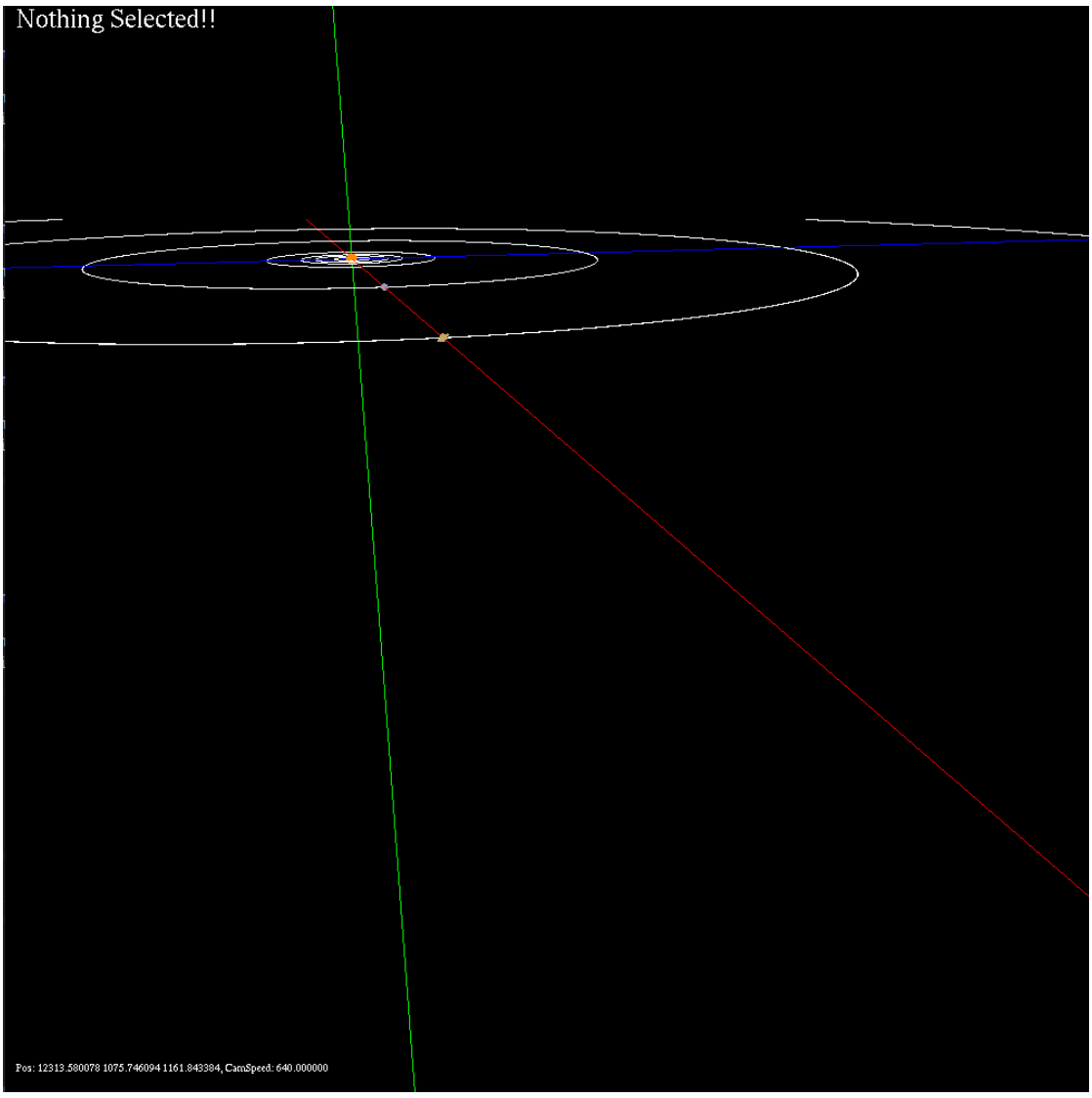


Figure 4: Sistema Solar numa vista Superior Perto de Urano

6 Conclusão

Com a conclusão da fase 2 do trabalho, tivemos a oportunidade de implementar transformações geométricas que testaram os nossos conhecimentos na cadeira e em trigonometria em geral.

Numa perspetiva mais crítica, reparamos que estamos a ter poucos FPS (*Frames per Second*), e a câmara tem alguns problemas ao iniciar, não ficando sempre no sítio suposto. Essas são as coisas que queremos melhorar principalmente para a próxima fase do trabalho.

Esperamos que, com a conclusão da fase 2, tenhamos desenvolvido uma base sólida para a implementação das próximas funcionalidades das próximas fases do trabalho.