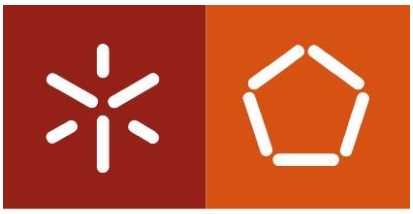

Universidade do Minho

Licenciatura em Engenharia Informática

Computação Gráfica

Fase 2 - Geometric Transformations

Grupo 21

Duarte Parente (A95844) Gonçalo Pereira (A96849)

José Moreira (A95522) Santiago Domingues (A96886)

Ano Letivo 2022/2023

Índice

Capítulo 1

Introdução

O presente relatório visa apresentar a segunda fase do projeto a desenvolver no âmbito da Unidade Curricular de Computação Gráfica. Ao longo do documento irá ser explicada a metodologia adotada para a resolução dos problemas, assim como as diversas opções tomadas ao longo desta segunda fase.

O principal objetivo desta etapa passou por atualizar o `Engine` desenvolvido na fase anterior, de forma a suportar a criação de cenários hierárquicos recorrendo a transformações geométricas. Esta alteração identificou-se particularmente no novo formato do ficheiro de configuração que passaria a permitir a existência de múltiplos grupos aninhados. A estes grupos poderiam estar associados três tipos de transformações geométricas: **Translação**, **Rotação** e **Escala**.

Isto levou a uma reformulação do `parser` quase na sua totalidade, para além das estruturas de dados e classes que também foram repensadas face às complicações levantadas pelas alterações enunciadas.

A `scene` necessária para realizar com sucesso esta segunda fase passava pela criação de um ficheiro de configuração que representasse graficamente um modelo estático do **Sistema Solar**. Para isso foi necessário recorrer às primitivas gráficas definidas assim como tirar proveito das transformações hierárquicas definidas para esta etapa.

Além dos requisitos propostos no enunciado procedemos à adição de duas novas primitivas gráficas: o **Cilindro** e o **Torus**. Esta última revelou-se necessária para a constituição do Sistema Solar através da representação do anel de um planeta.

Capítulo 2

Extras implementados

2.1 Funcionalidades implementadas

Previamente à resolução desta fase procedemos à implementação de algumas funcionalidades através do teclado e rato que serviriam para ajudar no debug do programa e também numa melhor exploração e aproveitamento dos cenários. Em relação ao teclado foram usadas as teclas:

- **1**, **2**, **3** - Mudam o modo de desenho da função *glPolygonMode()* para `GL_FILL`, `GL_LINE` e `GL_POINT`, respetivamente.
- **f** - controla o aparecimento dos eixos cartesianos.

Para além do aproveitamento do teclado foi também usado o rato para promover uma exploração facilmente controlada do cenário produzido. Nomeadamente, usa-se o botão direito para mover a câmara num formato bastante intuitivo, havendo também a possibilidade de controlo de zoom através do botão de *scroll*.

2.2 Novas Primitivas Gráficas

2.2.1 Cilindro

A construção do cilindro faz-se passando como valores o raio das suas bases, a altura do cone, o número de arestas da base (e consequentemente número de faces laterais), as *slices* e o número de níveis ou divisões horizontais (as *stacks*).

A técnica adotada foi calcular a altura que cada *stack* iria ter com base no número de *stacks* que o cilindro contém e a altura do mesmo, e de seguida fazer num ciclo cada *stack*, de baixo para cima. Nesse ciclo é feita também uma verificação para saber se se trata da primeira ou da última iteração e, nesses casos, fazer as respetivas bases (a primeira iteração corresponde à base do cilindro e a última ao topo).

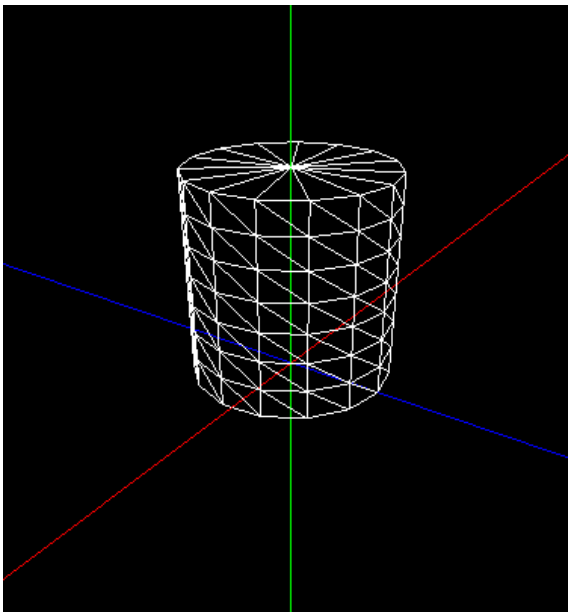

Figura 1: Exemplo de cilindro

2.2.2 Torus

O torus recebe como argumento para a sua construção o raio exterior, o raio interior, o número de fatias (*slices*), e o número de *stacks* (as divisões de cada fatia).

A estratégia adotada teve por base a interpretação de cada porção denominada "slice"que iria constituir os 360^{o} do corpo do "torus". Com isto, era do conhecimento que cada uma das mesmas seria constituída por um número dado de "stacks"e que o processo de divisão nessas mesmas divisões seria igual para todas as fatias, onde, mais uma vez, a totalidade dessas divisões teria que completar os 360^{o} do anel da respetiva porção. Deste modo, foi apenas preciso aplicar cálculos de seno e cosseno, recorrendo às distâncias externa e interna da primitiva ao centro.

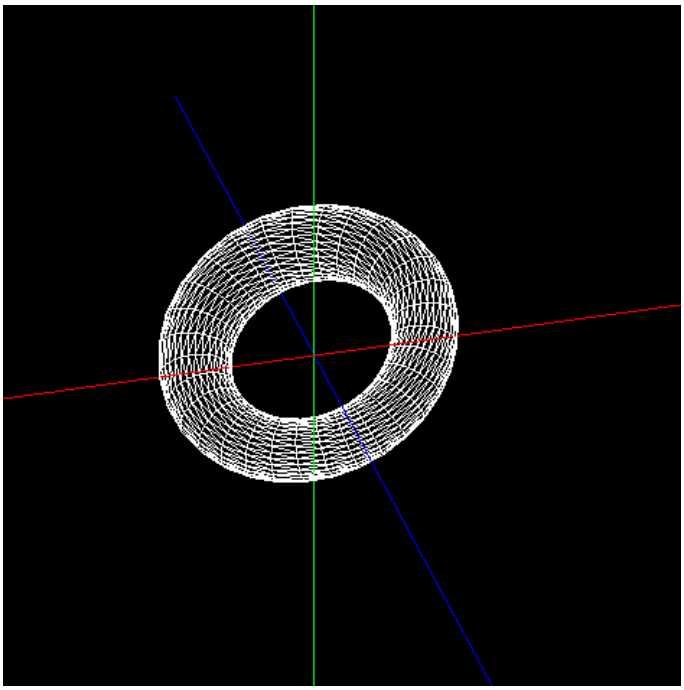

Figura 2: Exemplo de torus

Capítulo 3

Leitura e Armazenamento do Ficheiro de Configurações

3.1 Novo Formato do Ficheiro

Nesta fase do projeto o ficheiro de configurações necessário como argumento do `Engine` sofreu importantes alterações em termos estruturais e de sintaxe. Segue-se um exemplo de um possível ficheiro de configuração por forma a permitir a visualização e assim uma melhor interpretação dos tópicos que serão abordados.

```
<world>
    <window width="512" height="512" />
    <camera>
        <position x="5" y="-2" z="3" />
        <lookAt x="0" y="0" z="0" />
        <up x="0" y="1" z="0" />
        <projection fov="20" near="1" far="1000" />
    </camera>
    <group>
        <transform>
            <translate x="4" y="0" z="0" />
        </transform>
        <models>
            <model file="cone.3d" />
        </models>
        <group>
            <transform>
                <translate x="4" y="0" z="0"/>
            </transform>
            <models>
                <model file="sphere.3d" />
            </models>
        </group>
    </group>
    <group>
```

```
        <transform>
            <translate x="0" y="4" z="0"/>
        </transform>
        <models>
            <model file="cone.3d" />
        </models>
    </group>
</world>
```

A partir deste exemplo consegue-se notar por um lado a ausência de alterações nas informações das definições da câmara e da janela, e por outro a enorme reforma aplicada à definição dos grupos. Estes apresentam-se com uma estrutura em árvore onde cada nodo contém os conjuntos de transformações a aplicar e opcionalmente um conjunto de modelos. Além disso, cada nodo pai pode também apresentar um conjunto de nodos filho que irão herdar as transformações dos seus antecessores.

3.1.1 Nova Extensão XML

Além destas alterações obrigatórias optamos por adicionar uma nova extensão XML, no sentido de enriquecer a qualidade dos modelos gerados, nomeadamente o do Sistema Solar. Esta extensão passou por permitir a especificação da cor com que os modelos de um grupo serão gerados, sendo que o branco é a cor usada caso esta não seja especificada.

Tendo as fases futuras do projeto em consideração acordamos o uso da tag `color`, que ficará contida dentro do domínio do `models`. Para já a color apenas poderá receber uma outra denominada `rgb` que receberá 3 floats como atributos e que serão os argumentos da função *glColor3f()*. Segue-se a representação de um grupo com uma cor especificada:

```
    <group>
        <transform>
            <translate x="0" y="0" z="15.2"/>
        </transform>
        <models>
            <model file="sphere_10_20_20.3d" />
            <color>
                <rgb R="0.67" G="0.67" B="0.67"/>
            </color>
        </models>
    </group>
```

3.2 Armazenamento dos dados

Enquanto que na fase anterior recorríamos a uma estrutura de dados denominada `Settings`, e que guardava apenas a informação das definições da câmara, da janela e conjunto de modelos a representar, foi necessário criar novas classes para armazenar as informações associadas a cada um dos grupos.

```
class XML {
    public:
```

```
        int w_width;            // Largura da Janela
        int w_height;           // Comprimento da Janela
        Settings settings;      // Definições da Câmara
        vector<Group*> groups;  // Conjunto de Grupos
}
```

Em relação às definições de configuração da câmara e da janela não houve qualquer tipo de alteração, não havendo por isso necessidade de alterar a estrutura de dados usada:

```
class Settings{
    public:
        Point position;
        Point lookAt;
        Point up;
        Point projection;
}
```

Já a estrutura associada ao armazenamento de um grupo foi a que envolveu uma maior ponderação, tendo-se optado pela seguinte resolução:

```
class Group{
    public:
        vector<Transformation*> transformations; // Conjunto de Transformações
        vector<string> models;                   // Conjunto de Modelos
        vector<Group*> groupChildren;            // Grupos descendentes
        Color color;                             // Côr do grupo
}
```

De forma a facilitar o armazenamento e forma como são aplicadas as transformações definiu-se uma super classe `Transformation`, onde `Translate`, `Rotate` e `Scale` se apresentam como extensões da mesma.

Capítulo 4

Análise de Resultados

Neste capítulo serão apresentados os resultados do programa quando confrontado com os testes e respetivos ficheiros de configuração disponibilizados pela equipa docente. Foram disponibilizados 4 testes, cada um focando em vários aspetos definidos para esta fase, tanto a nível da aplicação das diferentes transformações como a definição de grupos hierárquicos.

Seguem-se representados os resultados obtidos com o programa desenvolvido (metade esquerda da figura), e respetiva comparação com o resultado esperado (metade direita da figura).

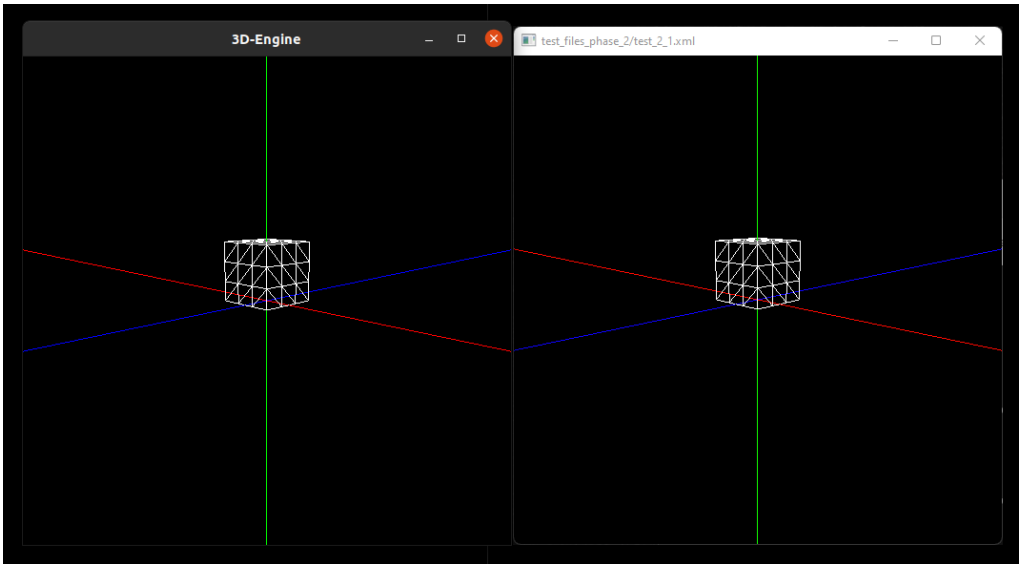

Figura 3: Teste 1

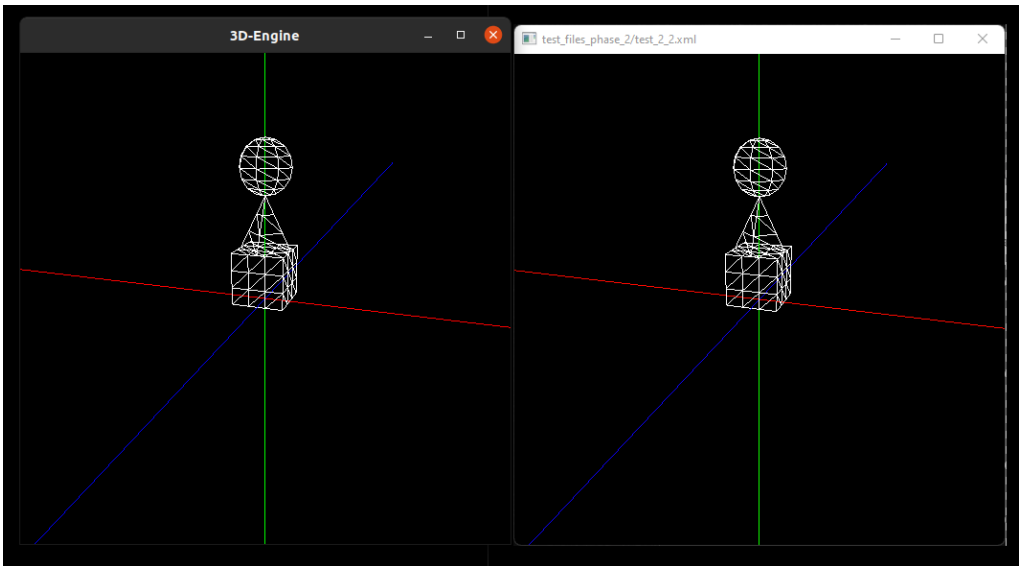

Figura 4: Teste 2

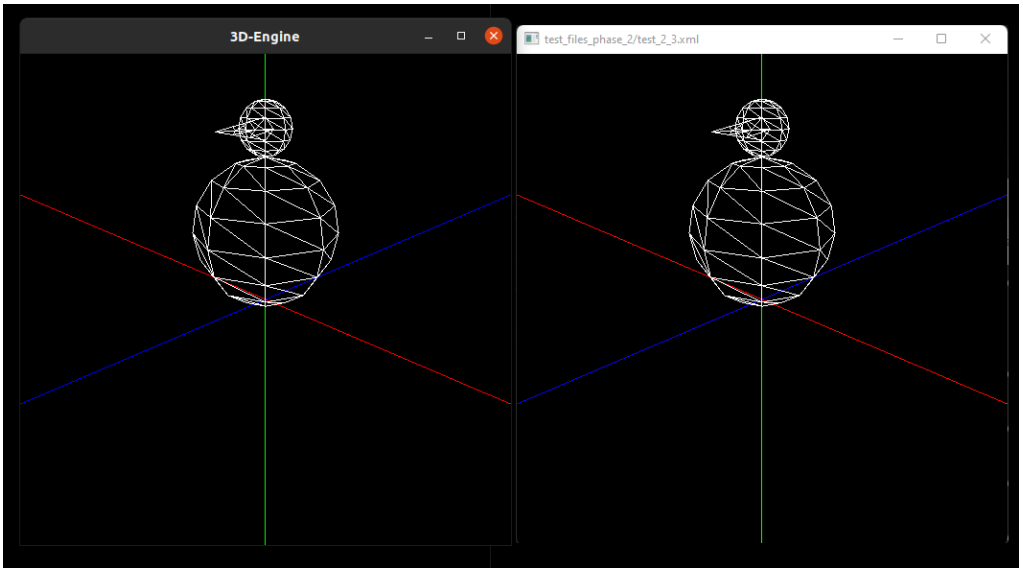

Figura 5: Teste 3

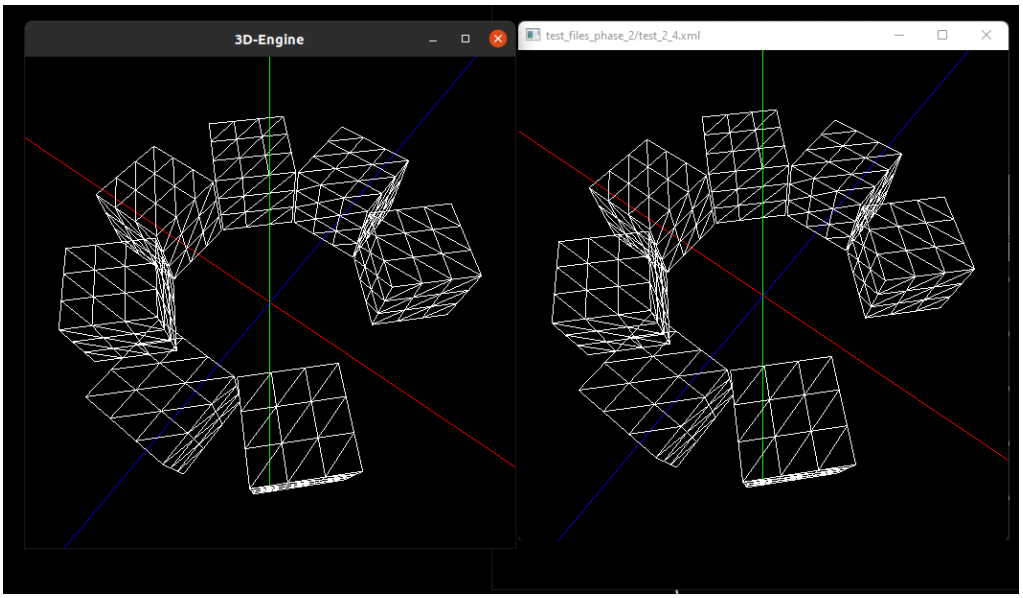

Figura 6: Teste 4

A partir da análise das figura conseguimos constatar que todos os testes passaram com sucesso.

Capítulo 5

Sistema Solar

De forma a cumprir com o requisito principal desta segunda fase do projeto procedemos à criação de um ficheiro de configuração personalizado que representasse graficamente o **Sistema Solar**, sendo nesta fase apenas necessária a sua representação estática.

Para a sua concretização recorremos a algumas das primitvas gráficas definidas na primeira fase, para além do `Torus` implementado já no decorrer desta etapa. Este serviu para representar o anel de Saturno, enquando que a esfera foi a figura escolhida para a representação do sol e dos planetas, assim como os seus satélites naturias.

Seguem-se imagens do desenho 3D produzido pelo `Engine`:

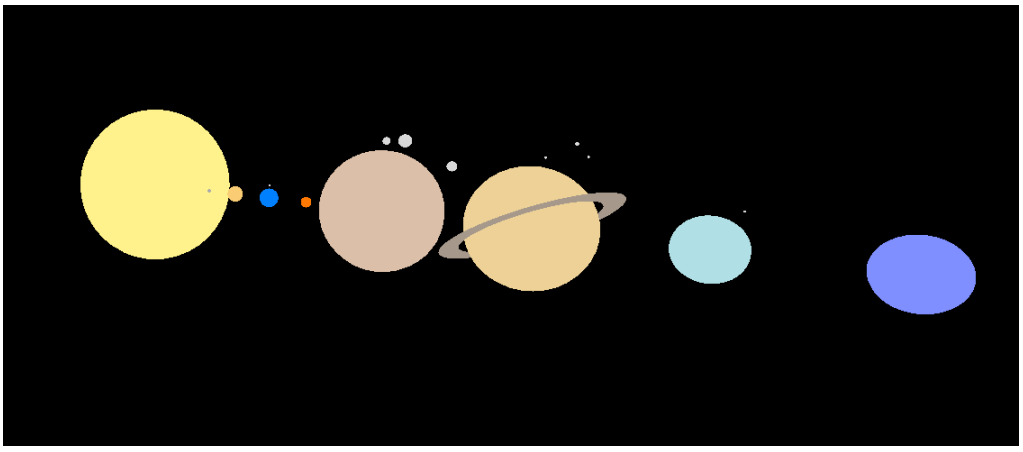

Figura 7: Sistema Solar

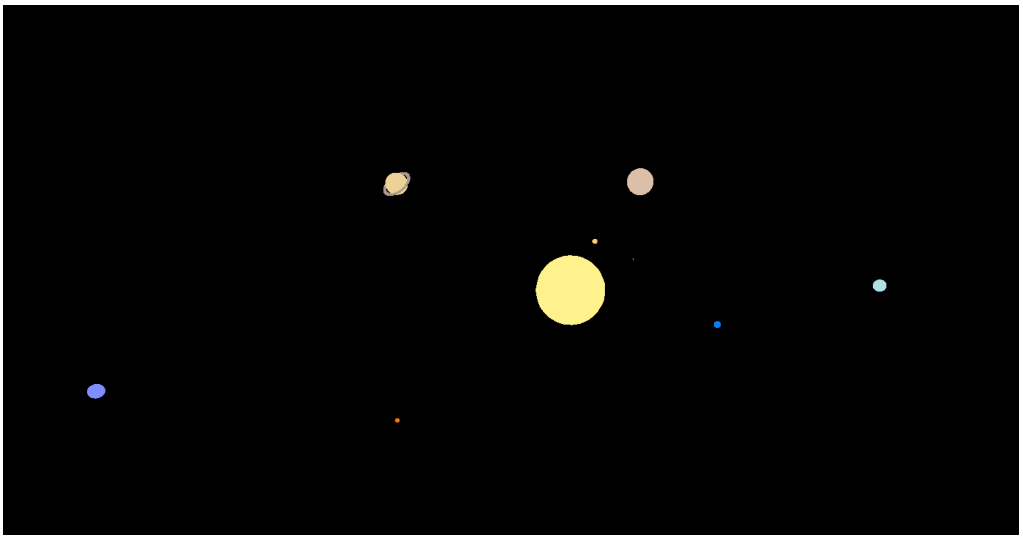

Figura 8: Sistema Solar

De notar que o ficheiro de configuração que traduz a configuração do sistema solar definido através de grupos hierárquicos, encontra-se na pasta `config` da diretoria `Engine` com o nome de *solar_system.xml*.

Capítulo 6

Conclusão

Dando por terminada a realização da segunda fase do projeto estamos em condições de efetuar uma análise crítica ao trabalho realizado.

A dificuldade inicial desta fase prendeu-se na necessidade de reformulação de grande parte do processo de parsing, assim como a sua ligação às estruturas de dados necessárias para o armazenamento do novo formato de definição dos grupos. Ainda assim acreditamos que essas dificuldades foram ultrapassadas e apresentamos uma solução capaz de corresponder aos requisitos propostos e preparada para o que iremos encontrar nas próximas fases.

Salientamos ainda os extras implementados, nomeadamente ao nível do `Generator`, com a adição das duas novas primitivas gráficas, assim como a possibilidade de configurar a côr de um grupo a partir do ficheiro de configuração. Outro extra com um enorme peso na qualidade do trabalho apresentado, ao nível da exploração e aproveitamento dos cenários produzidos pelo `Engine`, passa pela implementação de um modo de exploração da câmara.

No entanto, há ainda aspetos que podem ser melhorados, sendo o mais urgente o uso de VBOs, e que esperamos implementar já na próxima fase.