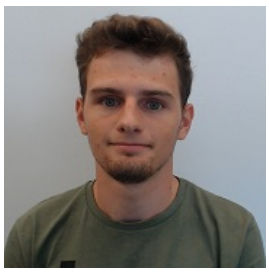
Universidade do Minho
Licenciatura em Engenharia Informática

Computação Gráfica

Trabalho Prático - Fase 2

Grupo 33


Bohdan Malanka
a93300


Diogo Rebelo
a93278


Henrique Alvelos
a93316


Lídia Sousa
a93205

4 de abril de 2022

Conteúdo

Lista de Figuras

1 Introdução

O presente documento é alusivo à **segunda** fase do projeto prático desenvolvido com recurso à linguagem de programação C++, no âmbito da Unidade Curricular de Computação Gráfica que integra a Licenciatura em Engenharia Informática da Universidade do Minho. Este projeto encontra-se dividido em quatro fases de trabalho, cada uma com uma data de entrega específica. Esta divisão em fases, pretende fomentar uma simplificação e organização do trabalho, contribuindo para a sua melhor compreensão.

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, que surgem a seguir:

- Compreender a utilização do *OpenGL*, recorrendo à biblioteca GLUT, para a construção de modelos 3D;
- Aprofundar temas alusivos à produção destes modelos 3D, nomeadamente, em relação a transformações geométricas, curvas, superfícies, iluminação, texturas e modo de construção geométrico básico;
- Relacionar todo o conceito de construir modelos 3D com o auxílio da criação de ficheiros que guardam informação relevante nesse âmbito;
- Relacionar aspetos mais teóricos com a sua aplicação a nível mais prático.

Naturalmente, é indispensável que o conjunto de objetivos supracitados seja concretizado com sucesso e, para isso, o formato do relatório está organizado de acordo com uma descrição do problema inicial, seguindo-se o conjunto de aspetos relevantes em relação ao **Gerador** e chegando aos aspetos primordiais sobre o **Motor**. O grupo decidiu incluir também uma secção direccionada para a descrição das funcionalidades adicionais.

2 Descrição do Problema

Nesta segunda fase do projeto o objetivo é desenvolver novas funcionalidades no **Motor** para interpretar e processar grupos de transformações geométricas, tais como translações, escalas e rotações especificadas no ficheiro XML. Além disso, também é necessário criar um ficheiro desta categoria de forma a desenhar um modelo estático do sistema solar.

3 Estrutura do projeto

Tal como na primeira fase a estrutura do projeto é mantida, isto é, as aplicações são divididas por diretorias de *Generator*, *Engine* e *Models*.

De um modo geral, a aplicação é construída pelas seguintes componentes:

- **Gerador:** contém o conjunto de funções e estruturas responsáveis por gerar o ficheiro com o conjunto de pontos de cada modelo, com a extensão `.3d`;
- **Motor:** contém o conjunto de funções e estruturas responsáveis por ler o ficheiro de configuração XML e representar graficamente cada modelo;
- **Models:** diretoria onde os ficheiros .3d e os ficheiros XML ficam guardados.

4 Gerador

Quanto ao Gerador, nesta fase não houve necessidade de se fazerem algumas alterações pelo que esta aplicação se encontra igual à primeira fase.

5 Motor

Como houve uma alteração dos ficheiros XML dos que eram tratados na primeira fase, isto é, agora há mais informação(transformações geométricas) a ser armazenada houve a necessidade de alterar a nossa estrutura de dados para tal.

A estrutura de dados que armazena os dados relativos à **câmara** é a seguinte:

```
struct camera {
    float position[3] = {0,0,0}
    float lookAt[3] = {0,0,0}
    float up[3] = {0,0,0}
    float projection[3] = {0,0,0}
}
```

A estrutura que armazena uma **transformação geométrica** consiste em:

```
struct Transform {
    string name;
    float x;
    float y;
    float z;
    float angle;
}
```

Por outro lado, para armazenar os **dados de um determinado grupo** recorremos à seguinte estrutura:

```
struct Group{
    vector<Transform> transformation; //geometric transforms
    vector<Point> points;             //vertices of the models
    vector<Group> subGroups;          //subgroups of the group

}
```

Quanto à **estrutura de dados principal**:

```
struct World{
    camera *cam = new camera;
    vector<Group> groups;
};
```

Nesse sentido, a estrutura principal (*World*) armazena a estrutura de dados da câmara e um vector de estruturas de dados que armazena um grupo. Já um grupo possui um vector de transformações aplicados a figuras desse grupo, um vector de pontos carregados no momento do parse do ficheiro XML e um vector de subgrupos.

5.1 Leitura do ficheiro XML

Para a leitura e *parse* deste tipo de ficheiros continuamos a usar a biblioteca *tinyxml2*. Porém, nesta fase já é necessário interpretar as transformações geométricas de um grupo e dos seus subgrupos, pelo que o grupo decidiu alterar a função *parseInput* e dividi-la em duas em que a leitura dos dados da câmara é feita pela função *parsecamera* e posteriormente a função *parseGroup* que interpreta os grupos e subgrupos.

Além disso foi alterada a função *readFile* pelo que já não retorna um booleano mas um vector dos respetivos pontos lidos e que é chamada a medida que se efetua o *parse* do ficheiro XML, ou seja, já não guardamos os nomes dos ficheiros .3d.

5.2 Desenho das primitivas

Para desenhar as primitivas com a nova estrutura de dados foi criada a função *drawGroups*.

Esta função itera por cada grupo presente na estrutura principal *world.groups* e para cada um faz *push* da matriz das transformações através da função do *glut glPushMatrix*. Posteriormente percorremos o *array* das transformações e executamos por ordem que aparece no ficheiro, atraves das funções *glTranslatef*, *glScalef* e *glRotatef*. Depois desenhamos a respetiva figura deste grupo com os pontos armazenados com ajuda da função *drawPrimitives*. Antes de fazer *pop* voltamos a chamar recursivamente a função (*drawGroups*) para executar as transformações dos subgrupos e só depois disso é que fazemos *glPopMatrix*. Deste modo, os subgrupos herdam as transformações do grupo pai.

5.3 Extras

Quanto aos extras, nesta fase apenas implementamos a contagem dos FPS's(*Frame per Second*) desenvolvido na aula prática 4. Desta forma podemos comparar o desempenho com um número maior de vértices por figura e posteriormente a implementação dos VBO's.

5.4 Modelo do sistema solar

Antes de desenvolver o modelo do sistema solar é preciso ter em consideração as seguintes propriedades:

- o *translate* de um grupo é somado ao do subgrupo;
- o valor do*scale* de um grupo é multiplicado ao valor do subgrupo;
- o valor do *rotate* do grupo é somado ao do subgrupo;

Para compor o sistema solar num ficheiro .xml, decidimos, primeiramente, por saber as distâncias dos planetas em relação ao Sol e da Lua em relação ao planeta Terra e o diâmetro de cada elemento do modelo. Percebemos que, para desenhar, seria bastante difícil usar apenas uma escala para distâncias e diâmetros. Para resolver este problema, decidimos que, para as distâncias a escala seria 1/20000 e para os diâmetros seria 1/20000000. Contudo, ao colocar estes valores, verificamos que os planetas se sobrepunham, isto porque a distância estava mal calculada: teria de ser calculada pela distância entre os centros do planeta.

Quanto às inclinações dos planetas, começamos por definir os valores de acordo com a seguinte tabela:

Planeta	Inclinação do eixo de rotação (º)
Mercúrio	7
Vénus	177 *
Terra	23.5
Marte	25
Júpiter	3
Saturno	27
Urano	98 *
Neptuno	30
	* Rotação retrógrada

Assim, aplicamos um *rotate* sobre o eixo dos zz com o valor do ângulo de cada inclinação. Como este ângulo é positivo para o lado esquerdo, é necessário aplicar o valor negativo ao ângulo para a inclinação se verificar para o lado direito.

6 Testes

Para mostrar em funcionamento o trabalho exposto ao longo deste relatório realizamos os seguintes testes:

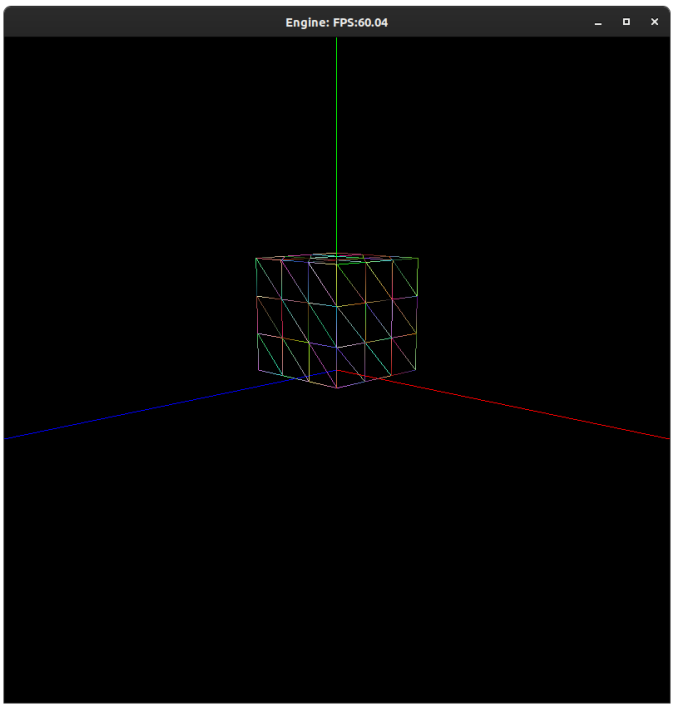


Figura 1: Ficheiro test_2_1.xml

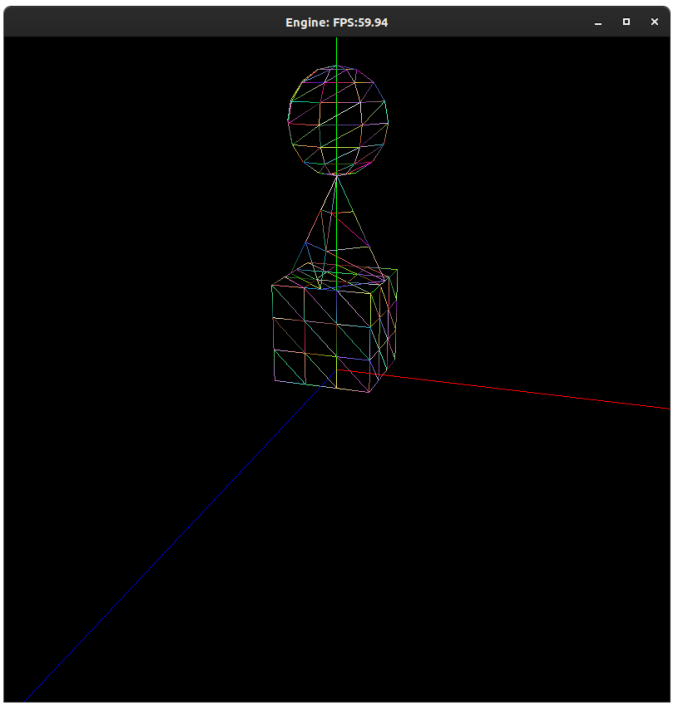


Figura 2: Ficheiro test_2_2.xml

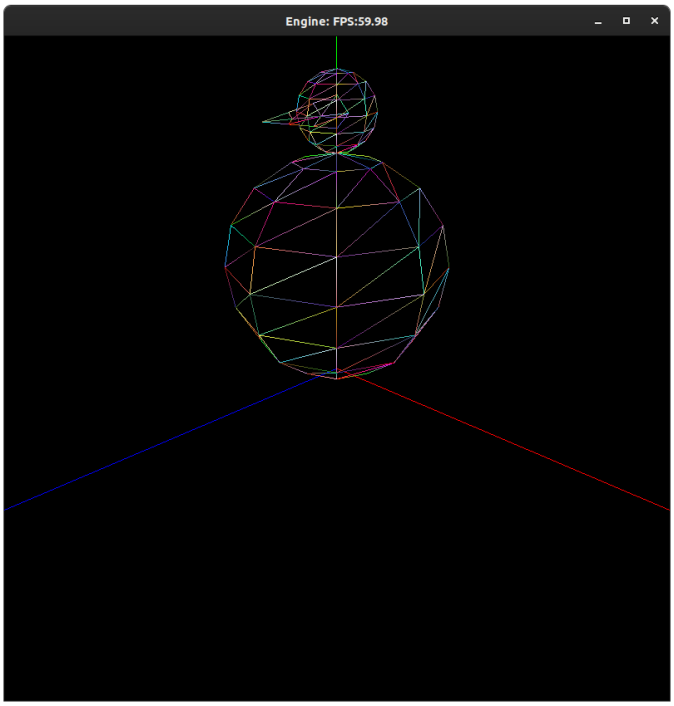


Figura 3: Ficheiro test_2_3.xml

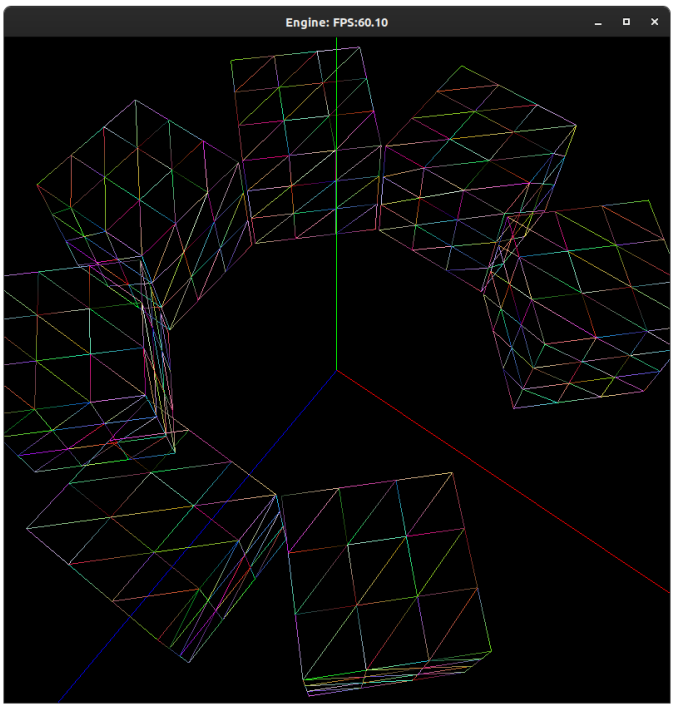


Figura 4: Ficheiro test_2_4.xml

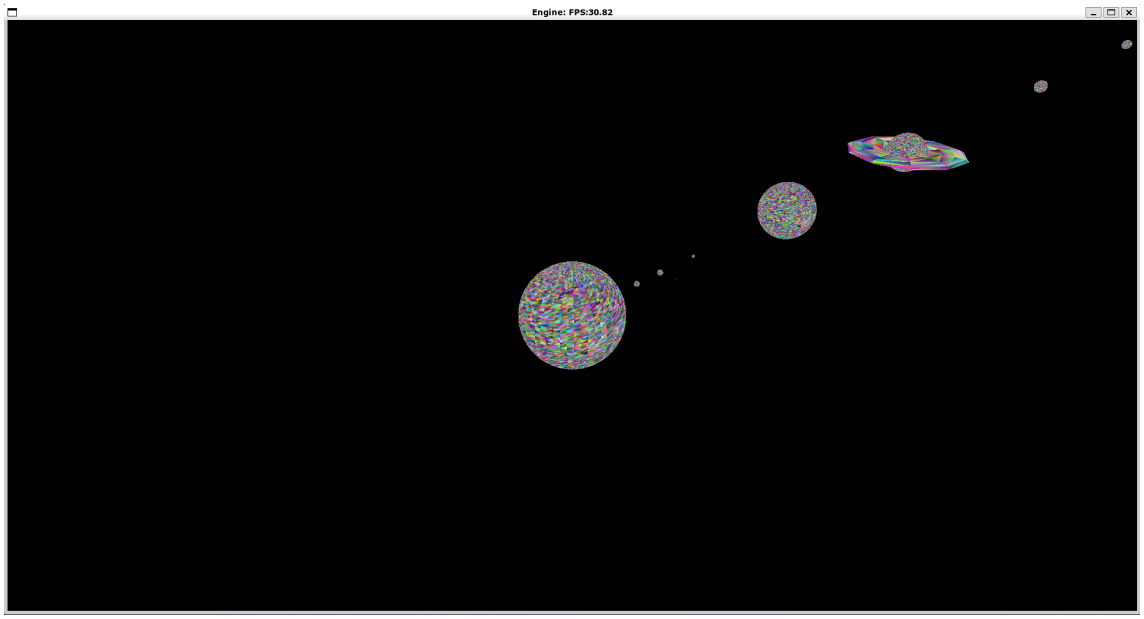


Figura 5: Ficheiro solarSystem.xml

7 Conclusão

Da realização desta segunda fase, o grupo considera que a mesma foi bem conseguida, já que se realizaram todas as funcionalidades requisitadas pelo próprio enunciado.

No espetro positivo, consideramos conveniente destacar o correto funcionamento do nosso programa. Além disso, as estruturas implementadas estão em concordância com a estrutura do XML permitindo uma visualização mais clara daquilo que é armazenado.

Por outro lado, também existiram algumas dificuldades, tais como a familiarização com as funções do *tinyXML2* e a implementação de funções recursivas. Além disso, na representação do modelo do sistema solar o principal problema foi determinar uma escala o mais próxima da realidade e com isso descobrir as translações, escalas e rotações dos planetas e seus satélites. Apesar disso, as dificuldades foram ultrapassadas.

Para concluir, consideramos que houve um balanço positivo do trabalho realizado dado que as dificuldades sentidas foram superadas a ser cumpridos todos os requisitos.