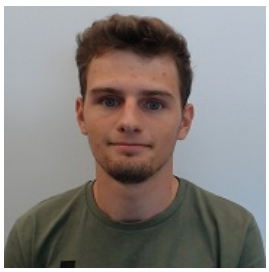
Universidade do Minho
Licenciatura em Engenharia Informática

Computação Gráfica

Trabalho Prático - Fase 4

Grupo 33


Bohdan Malanka
a93300


Diogo Rebelo
a93278


Henrique Alvelos
a93316


Lídia Sousa
a93205

27 de abril de 2024

Conteúdo

Lista de Figuras

1 Introdução

O presente documento é alusivo à **quarta** 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**.

2 Descrição do Problema

Nesta fase o **Generator** deve gerar coordenadas de textura e normais para cada vértice. Quanto ao **Engine**, as funcionalidades de iluminação e texturização devem ser implementadas, assim como ler e aplicar as normais e as coordenadas de textura dos modelos criados pelo **Generator**.

Já o ficheiro XML deve permitir a definição da cor difusa, especular, emissiva e ambiente, bem como brilho. Além disso, deve permitir definir as fontes de luz.

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

Para aplicar as novas funcionalidades, foi necessário alterar o ficheiro *output*. Assim, cada linha desse ficheiro possui três pontos, seguidos de três pontos para a normal e, por fim, dois valores para as coordenadas da textura. Segue-se uma explicação para cada objeto:

Figura 1: Estrutura do ficheiro 3d

4.1 Plano

4.1.1 Normais

Como o plano está virado para cima, bastou verificar que a normal de cada ponto é **(0, 1, 0)**.

4.1.2 Textura

Para a textura, em vez de cobrir o plano todo com uma única textura, a imagem vai-se repetindo ao longo da grelha do plano.

4.2 Cone

4.2.1 Normais

Podemos dividir as normais em dois segmentos: a base do cone e a superfície lateral. Quanto à base, todos os pontos têm a normal como **(0, -1, 0)**.

4.3 Caixa

4.3.1 Normais

Sendo que a caixa é formada por 6 "planos", serão o número de vetores das normais será 6 (um para cada face). Além disso, duas faces são perpendiculares com o plano xy, outras duas são com o plano yz e as restantes são com o plano xz. A única diferença de cada "par"de faces perpendicular com qualquer plano é que as faces têm direções opostas. Assim, as normais serão **(0, 0, 1), (0, 0, -1)** para as faces de frente e de trás, **(0, 1, 0) para as faces de cima e de baixo, (0, -1, 0)** e **(1, 0, 0), (-1, 0, 0)** para as faces laterais.

5 Engine

De modo a suportar novas extensões, desta vez sobre definições da cor difusa, especular, emissiva e ambiente, brilho e fontes de luz, foi necessário alterar a leitura do ficheiro XML. A função *parseInput*, que trata da análise deste documento e do povoamento das estruturas de dados, possui, agora, capacidade de capturar grupos do ficheiro .XML relacionados com as definições anteriormente citadas, guardando na alterada estrutura **World**. Agora, esta *struct* possui um vetor de **Light**.

```
struct Light{
    string type;
    Point pos;
    Point dir;
    float cutoff;
};
```

Figura 2: Estrutura *Light*

Além disso, a struct **Model** foi alterada para processar as cores e texturas. Quanto às cores, foi criada uma estrutura **Color** e para as texturas apenas um inteiro. Este inteiro é um ID que foi criado quando a textura estava a ser carregada.

```
struct Color{
    float diffuse[4] = {0.8f, 0.8f, 0.8f, 1.0f};
    float ambient[4] = {0.2f, 0.2f, 0.2f, 1.0f};
    float specular[4] = {0,0,0, 1.0f};
    float emissive[4] = {0,0,0, 1.0f};
    float shininess = -1;
};
```

Figura 3: Estrutura *Color*

```
struct Model{
    string file3D;
    int textureID;
    int vertexCount;
    int indexBegin;
    //int indexEnd;
    Color color;
};
```

Figura 4: Estrutura *Model*

De modo a suportar as definições de luzes, depois de ler o ficheiro XML, ativa-se as opções de luz. Além disso, na função de renderização, a seguir à criação da câmara, "liga-se"cada luz, isto é, para cada elemento da lista de **Light**, dependendo do tipo, define-se os parâmetros de fonte de luz.

5.1 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;

As diferenças, em relação à terceira fase, não são muitas, visto que só se adicionaram campos relacionados com luz e textura. Quanto a estas últimas, retiramos do Planet Texture Map Collection.

6 Testes

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

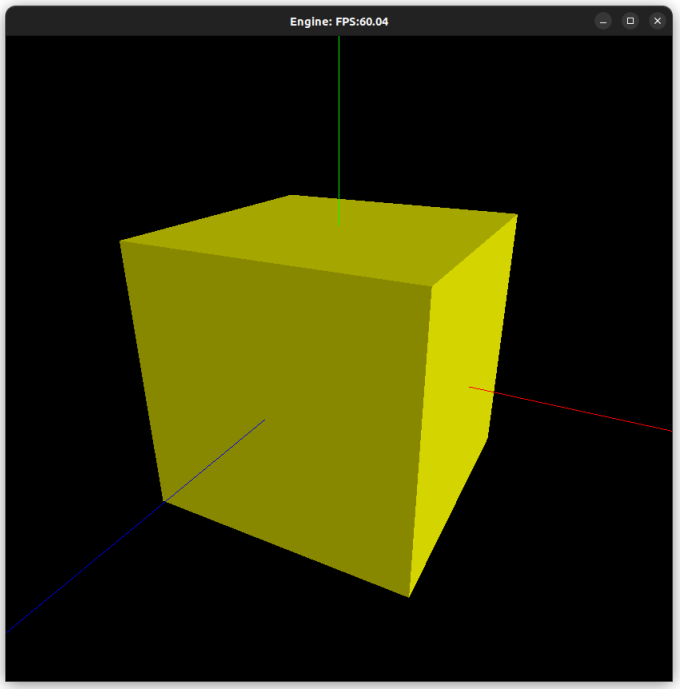


Figura 5: Ficheiro test_4_1.xml

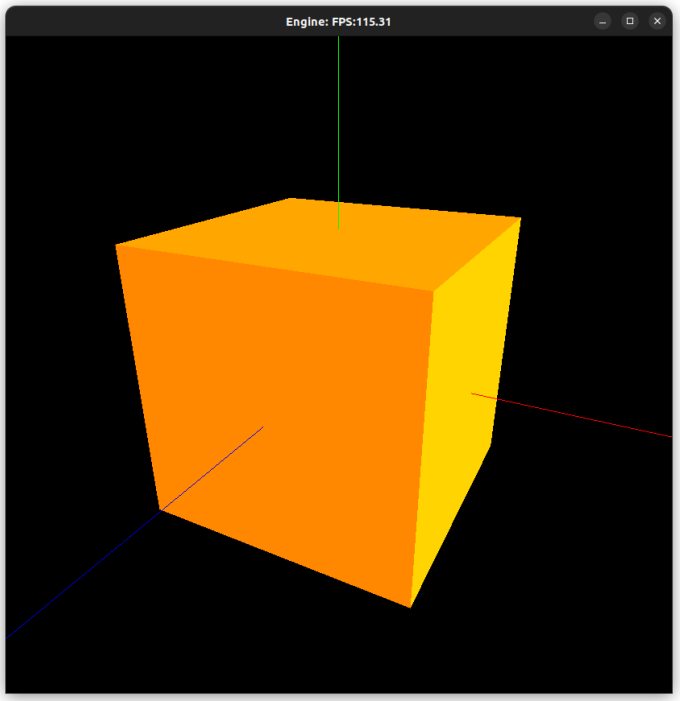


Figura 6: Ficheiro test_4_2.xml

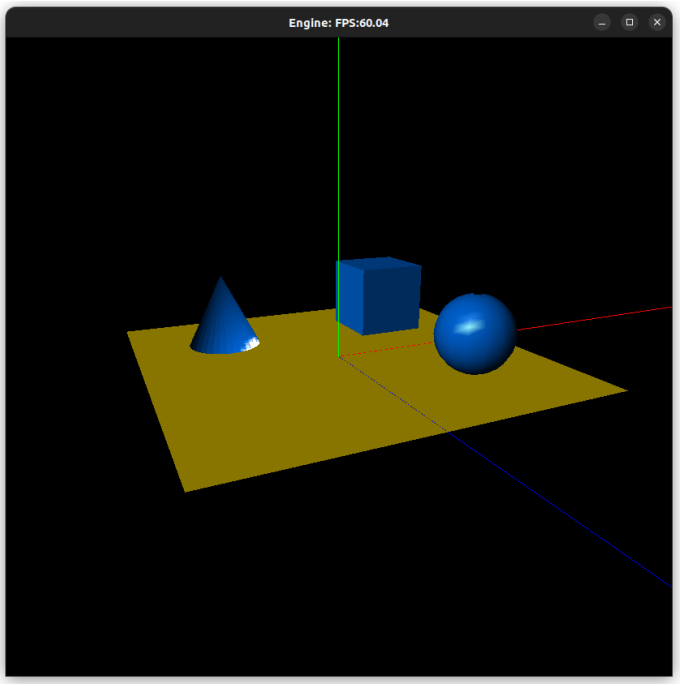


Figura 7: Ficheiro test_4_3.xml

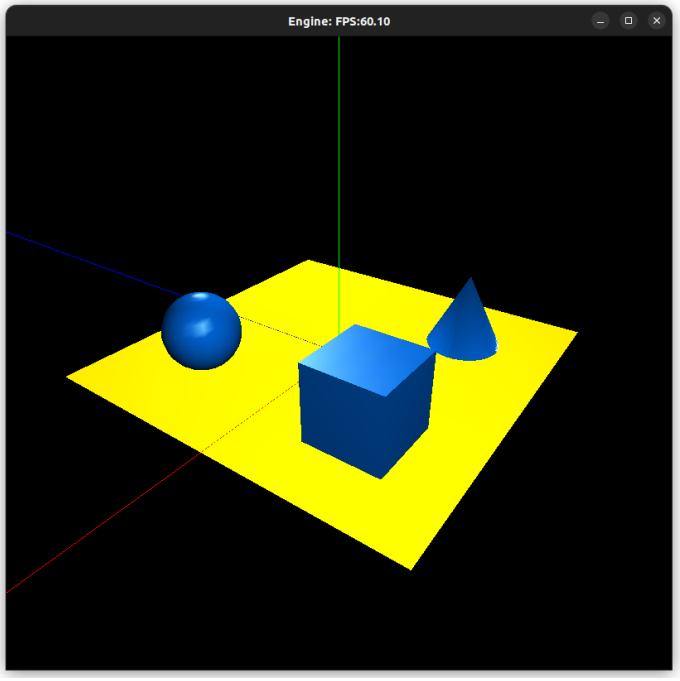


Figura 8: Ficheiro test_4_4.xml

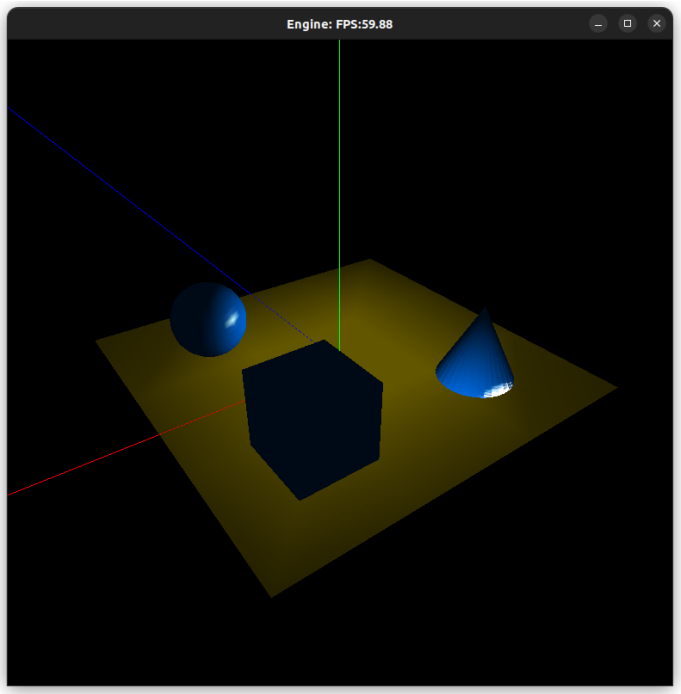


Figura 9: Ficheiro test_4_5.xml

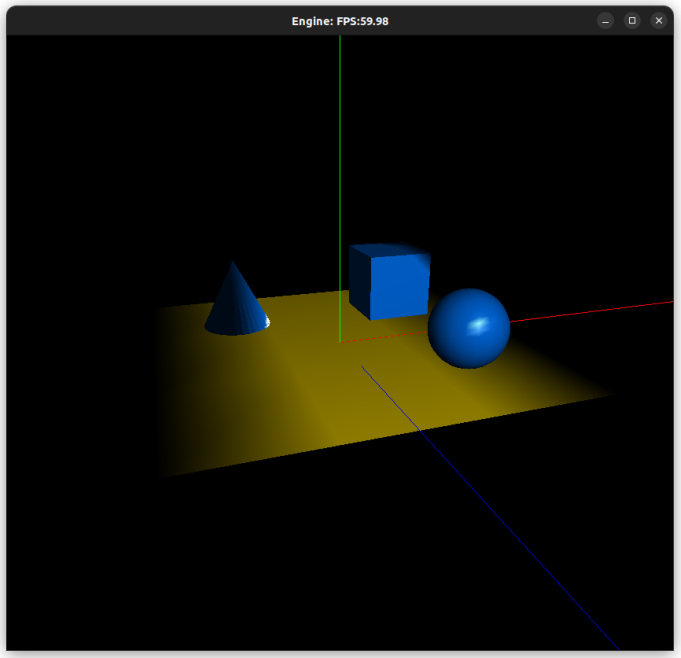


Figura 10: Ficheiro test_4_6.xml

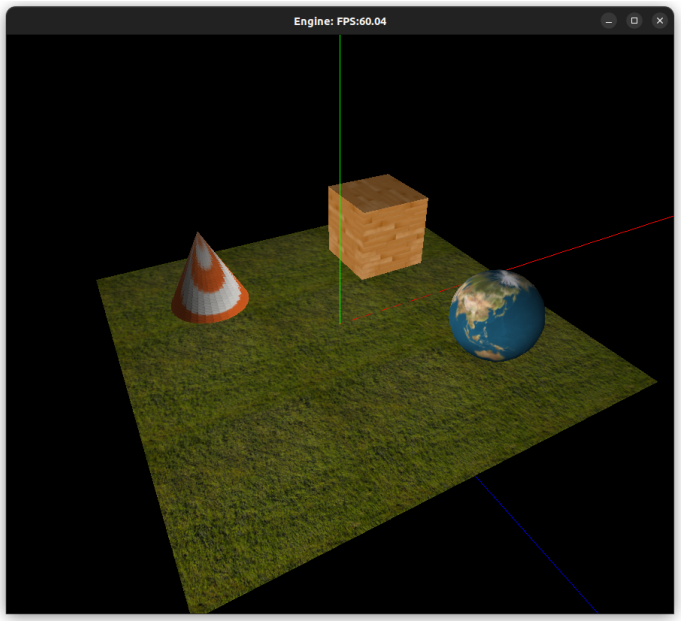


Figura 11: Ficheiro test_4_7.xml

7 Conclusão

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

No espetro positivo, consideramos conveniente destacar o correto funcionamento do nosso programa em termos de cores, luzes e texturas. 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, não conseguimos compilar o resultado do Sistema Solar com as texturas e o resultado não é de todo favorável visto que demorou imenso e os FPS são inferiores a 1.

Para concluir, consideramos que houve um balanço positivo do trabalho realizado dado que, apesar das dificuldades, conseguimos fazer grande parte das tarefas, de todas as fases.