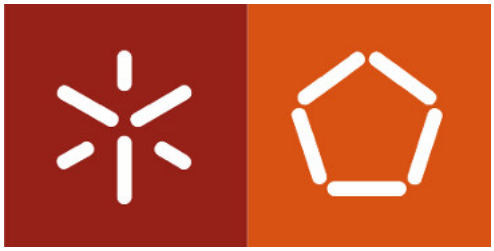


Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Computação Gráfica

Ano Letivo de 2022/2023

Fase 1 – Primitivas Gráficas

Grupo 6

A96434 Francisca Quintas Monteiro de Barros
A97588 Joana Isabel Freitas Pereira
A94870 Rafael Picão Ferreira Correia

10 de março de 2023

Índice

Lista de Figuras

1 Introdução

O presente relatório é referente ao trabalho prático desenvolvido pelo grupo 06 no âmbito da Unidade Curricular de Computação Gráfica, lecionada no curso de Licenciatura em Engenharia Informática no 2º Semestre do ano letivo 2022/2023.

Este projeto tem como objetivo o desenvolvimento de um mecanisno 3D baseado num cenário gráfico, através de ferramentas como o OpenGL e o C++, também usadas nas aulas práticas.

O projeto está dividido em 4 fases e para cada fase, serão fornecidos arquivos de configuração XML para testes e avaliação.

Esta primeira fase foca na representação de algumas primitivas gráficas e para a sua realização é necessário o desenvolvimento de duas aplicações:

- **Generator:** Destina-se a gerar os vértices consoante as primitivas gráficas recebidas.
- **Engine:** Destina-se a ler um ficheiro de configuração, em XML, de modo a posteriormente desenhar os vértices das primitivas gráficas anteriormente geradas. Além disso, o ficheiro de configuração contém informação sobre as configurações da câmera e sobre a janela.

2 Generator

Esta aplicação destina-se a gerar os vários vértices que constituem as diferentes primitivas gráficas que nos foram requeridas nesta fase:

- **Plano:** um quadrado no plano XZ, centrado na origem, subdividido em ambas as direções X e Z
- **Caixa:** requer dimensão e o número de divisões por aresta, centrada na origem
- **Esfera:** requer raio, fatias e camadas, centrada na origem
- **Cone:** requer raio inferior, altura, fatias e camadas, a base do cone deve estar no plano XZ)

Graficamente, o conjunto de 3 destes vértices correspondem a triângulos, estes são a unidade de constituição de todas as primitivas.

2.1 Modo de execução

Para a execução da aplicação *Generator* é necessário indicar, primeiramente o tipo de figura a gerar, de seguida os seus parâmetros (que serão aprofundados na secção de cada figura) e finalmente o local onde o ficheiro deve ser guardado.

De modo a facilitar a sincronização das duas aplicações, é aconselhável que o ficheiro gerado pela primeira aplicação (*Generator*), seja guardado na pasta *models*. Deste modo, no *input*, o nome do ficheiro deve vir precedido de *"../../engine/models/"*.

```
code> generator plane 2 3 ../../engine/models/plane_2_3.3d
code>
```

Figura 2.1: Comando

2.2 Plano

A criação de um plano requer a definição de dois parâmetros importantes: o número de divisões e o comprimento dos lados.

O número de divisões irá influenciar o tamanho do plano. Considerando n o número de divisões, o plano irá ter n x n quadrados. Cada quadrado pode ainda ser visto como a conjunção de dois triângulos.

O desenho dos pontos que definem os triângulos segue um padrão específico, começando da esquerda para a direita e de frente para trás. A primeira aresta a ser construída é a mais à esquerda (com menor valor de x), e essa aresta é construída de frente para trás (com maior valor de z para menor valor de z).

É importante notar ainda que o plano será desenhado no plano XZ, mantando a coordenada de Y com o valor 0 e centrado na origem pelo que as coordenadas de cada ponto correspondem a metade do valor de X e a metade do valor de Z.

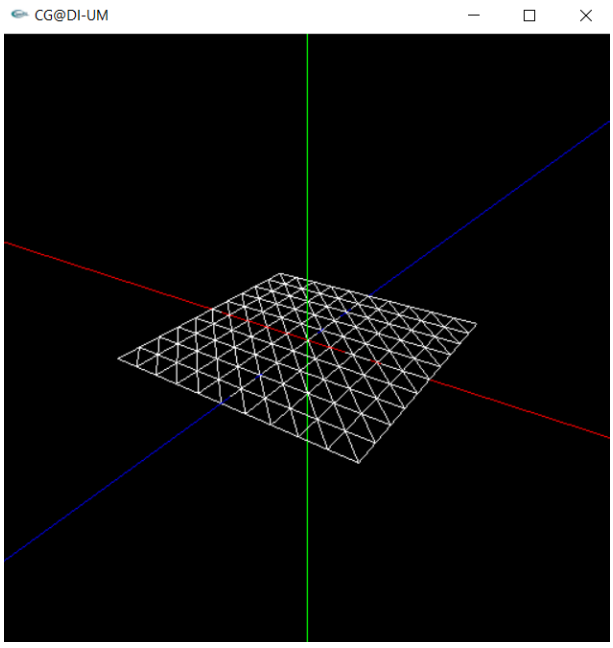


Figura 2.2: Exemplo de plano com comprimento 2 e 10 divisões

2.3 Caixa

Para a criação da caixa, são pedidos dois parâmetros, tal como no plano: o comprimento dos lados e o número de divisões.

Como tal, podemos ver a caixa como a união de 6 planos concorrentes, que representam as 6 faces da caixa.

Logo para cada plano iremos ter a mesma abordagem explicada anteriormente em relação à criação dos pontos.

A caixa vai também estar centrada na origem e as suas faces vão ter uma estrutura semelhante ao plano.

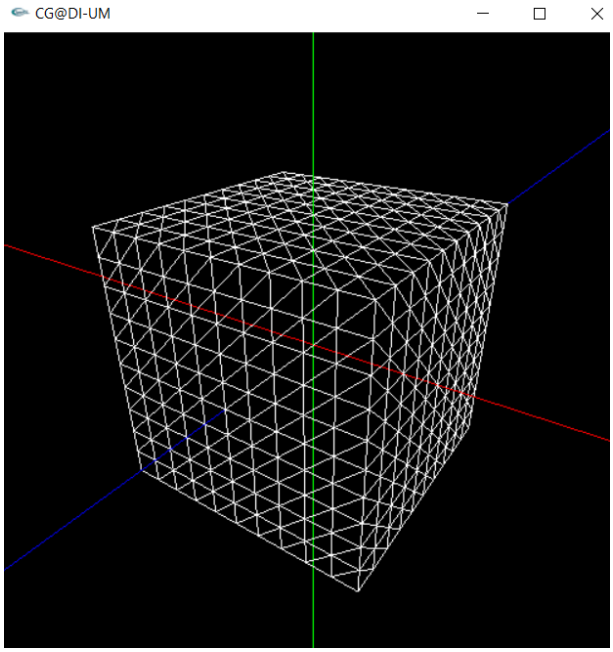


Figura 2.3: Exemplo de caixa com comprimento 2 e 10 divisões

2.4 Esfera

A criação da esfera já é mais complexo uma vez que já são pedidos três parâmetros: o raio, o número de camadas e o número de fatias.

Ao criar a esfera, ela deve estar centrada na origem. Além disso, é importante lembrar que a esfera é dividida em n x m retângulos, cada um formado por 2 triângulos, em que n é o número de camadas e o m o número de fatias.

A esfera é construída com base em coordenadas esféricas, ou seja, é constituída por um raio e dois ângulos, alfa e beta. O processo de criação começa do centro para as pontas, com os ângulos iniciais definidos em 0, como de pode ver na figura seguinte:

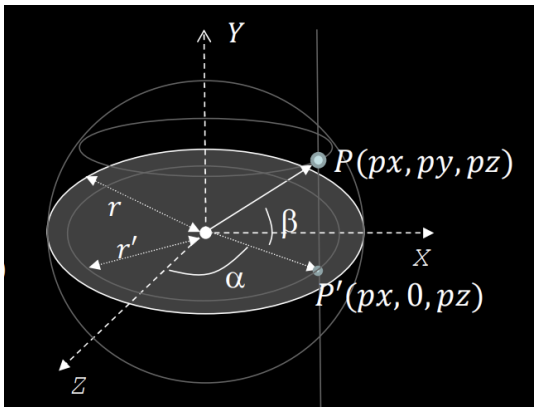


Figura 2.4: Coordenadas esféricas

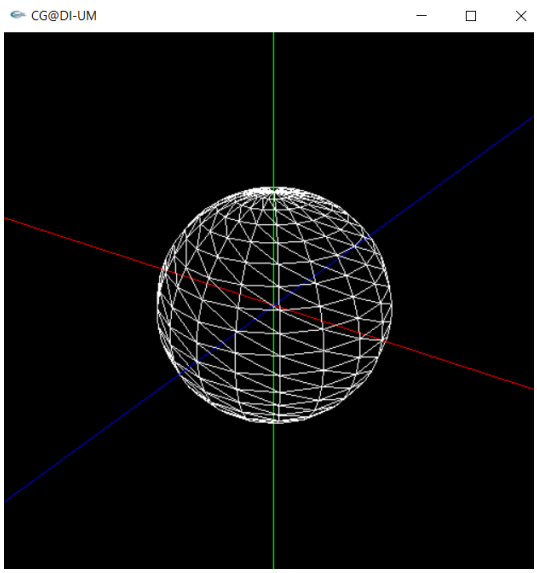


Figura 2.5: Exemplo de esfera com raio 1, 20 camadas e 20 fatias

2.5 Cone

Para a criação do cone, são pedidos quatro parâmetros: o raio, a altura, o número de camadas e o número de fatias.

O cone vai estar centrada na origem e a base no plano XZ. Este vai estar dividida em n x m retângulos, cada um constituído por 2 triângulos, em que n é o número de camadas e m o número de fatias.

A sua realização foi dividida em duas partes: a base (circunferência) e a superfície lateral.

Primeiro são gerados os pontos da base. A base é feita com triângulos no plano XZ direcionados para o centro da circunferência, ou sejam todos os triângulos partilham o ponto (0,0,0). O numero de triângulos depende do número de fatias verticais, tendo em conta que quanto maior o seu número, mais "redonda"ficará a circunferência.

De seguida, é preciso construir os triângulos que formam a superfície lateral do cone. Estes triângulos são construídos por camadas, de baixo para cima. Para tal, de camada para camada, o raio diminui e as coordenadas do y aumentam proporcionalmente.

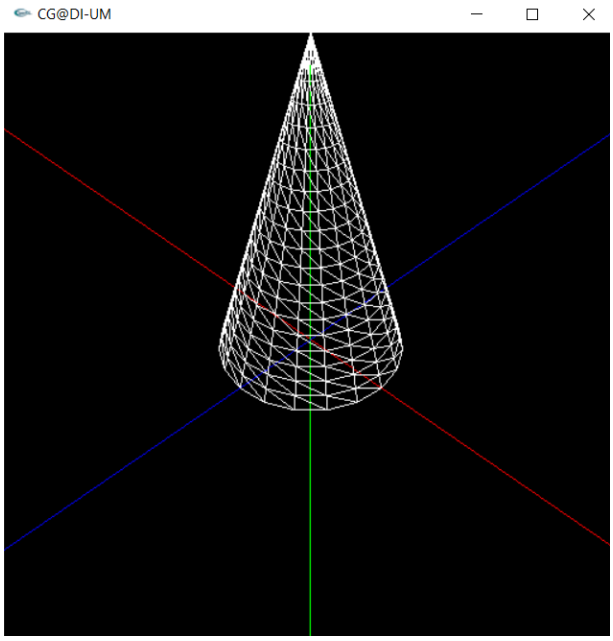


Figura 2.6: Exemplo de cone com raio 1, altura 3, 20 camadas e 20 fatias

3 Engine

A aplicação da *Engine* que deverá ser capaz de receber um ficheiro de configuração em XML e gerar os modelos 3D dinâmicos criados através do *Generator*.

Nesta primeira fase, os ficheiros contêm apenas informações básicas, como as definições da câmara, o tamanho da janela e informações de quais dos modelos carregar.

Para o parse desses ficheiros XML, foi utilizada a biblioteca *RapidXML*, que é capaz de guardar todas as informações em uma árvore para facilitar o acesso posterior.

Após a realização do parse, para cada ficheiro, é necessário contar o número total de pontos em cada um, para posteriormente alocar espaço em um buffer e armazenar esses pontos. Essa informação é armazenada em uma variável global e, no momento de renderização, esses pontos são desenhados na tela através da função *renderScene*.

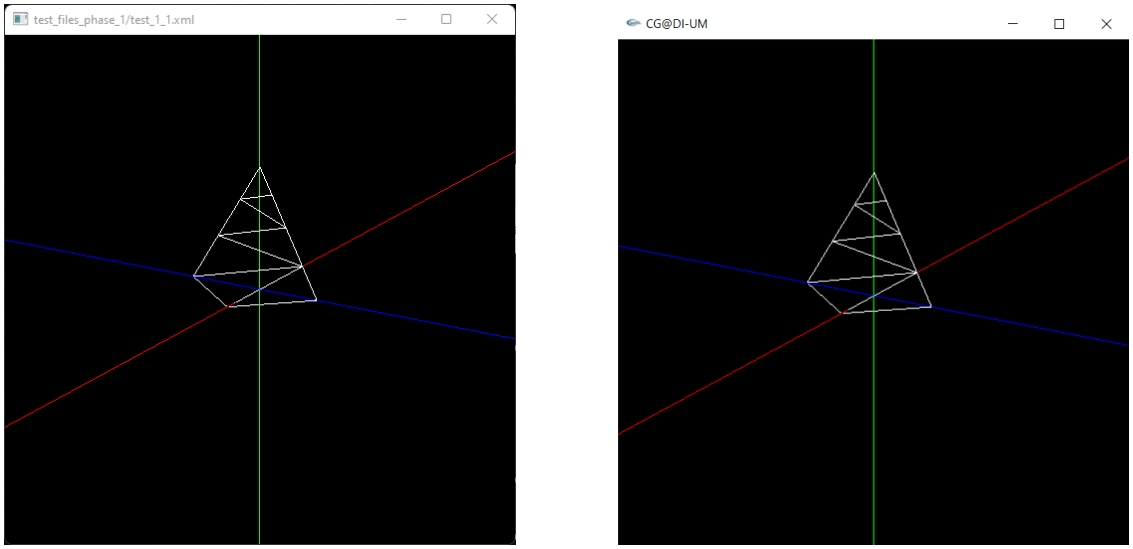
Para utilizar a *Engine*, basta modificar o valor do ficheiro de configuração no código e executar os comandos do *Cmake*. Depois, é só executar o programa no *Visual Studio* para visualizar os modelos 3D dinâmicos gerados a partir do *Generator*.

3.1 Testes

Para garantir a qualidade e eficácia do nosso programa, o grupo decidiu testá-lo com os ficheiros de teste disponibilizados na plataforma *Blackboard*.

Os resultados obtidos foram extremamente satisfatórios uma vez que os resultados foram idênticos.

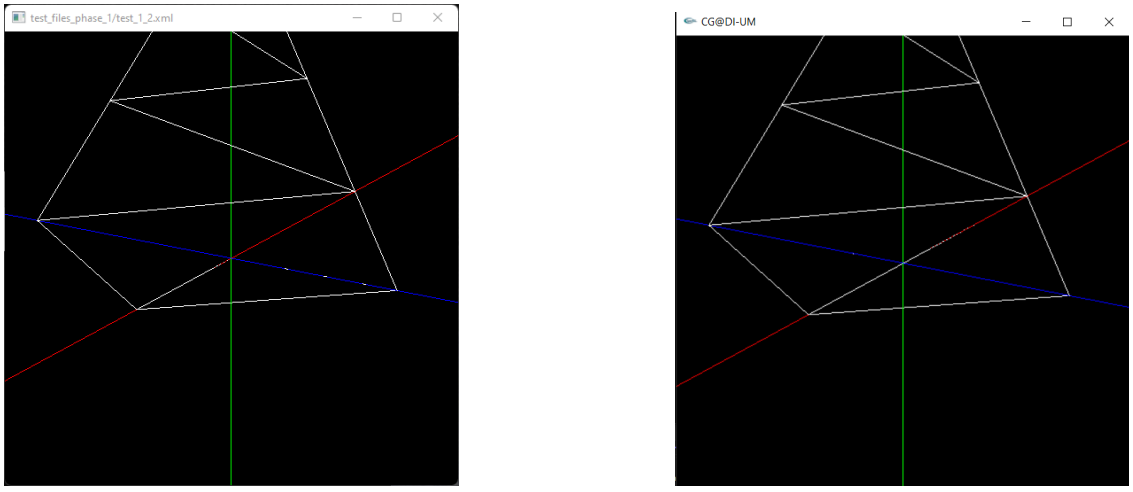
3.1.1 Teste Cone 1



(a) Resultado esperado (b) Nosso resultado

Figura 3.1: Teste de resultados - Cone 1

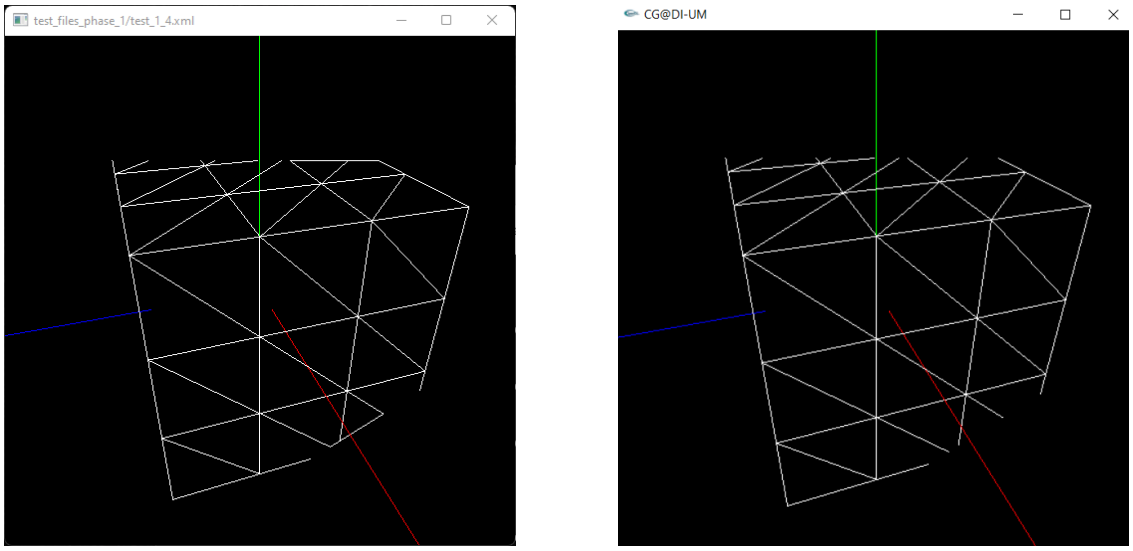
3.1.2 Teste Cone



(a) Resultado esperado (b) Nosso resultado

Figura 3.2: Teste de resultados - Cone 2

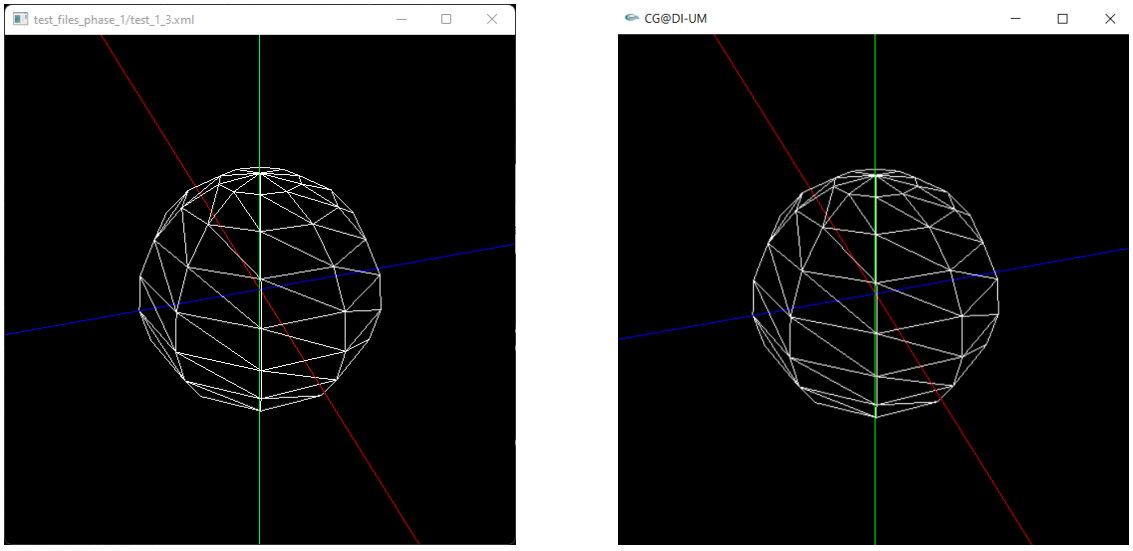
3.1.3 Teste Caixa



(a) Resultado esperado (b) Nosso resultado

Figura 3.3: Teste de resultados - Caixa

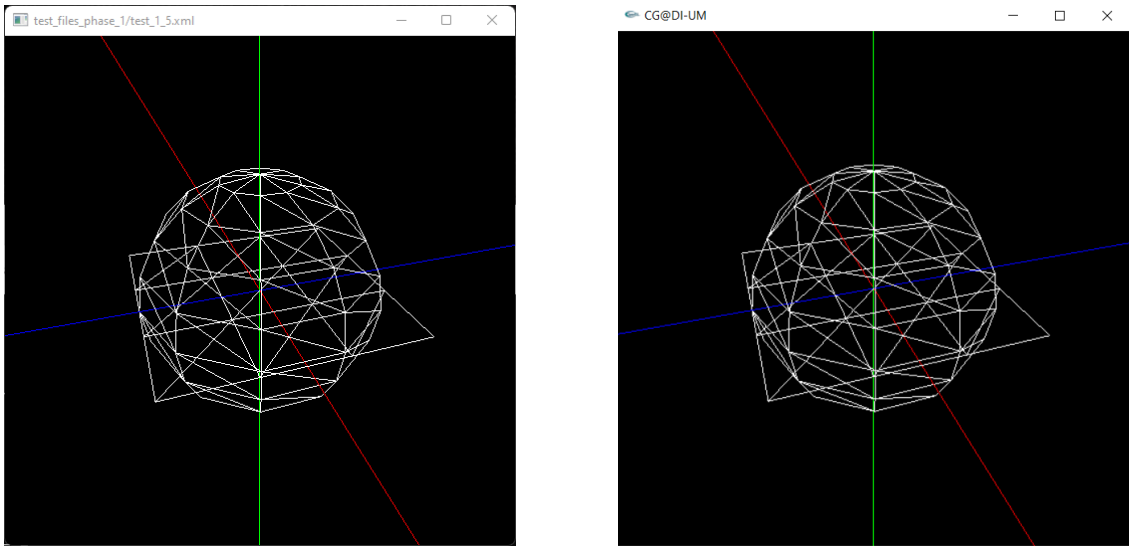
3.1.4 Teste Esfera



(a) Resultado esperado (b) Nosso resultado

Figura 3.4: Teste de resultados - Esfera

3.1.5 Teste Esfera e Plano



(a) Resultado esperado (b) Nosso resultado

Figura 3.5: Teste de resultados - Esfera e Plano

4 Conclusão

Concluímos que a primeira fase deste projeto foi um sucesso, uma vez que fomos capazes de cumprir todos os requisitos propostos no enunciado, desenvolvendo com sucesso dois programas essenciais: *Generator* e *Engine*. Esses programas foram criados de forma extremamente satisfatória, demonstrando a nossa capacidade de aplicar os conhecimentos lecionados até o momento tanto nas aulas teóricas como nas práticas.

Estamos também satisfeitos com os conhecimentos adquiridos em ferramentas associadas à Computação Gráfica, como o OpenGL GLUT e em C++, conhecimentos que nos serão muito úteis no futuro.

Estamos confiantes que o sucesso desta primeira fase, nos ajudará na conclusão das próximas fases e num bom projeto final.