Computação Gráfica
Universidade do Minho

Licenciatura em Ciências da Computação

Fase 1 - Relatório de Desenvolvimento

Hugo Costa (a96059)

Gonçalo Macedo (a87946)

Sara Fontes (a92999)

March 10, 2023

Contents

1 Introdução **2**
1.1 Enunciado 2

2 Apresentação das Soluções **3**
2.1 Gerador 3
2.1.1 Plano 3
2.1.2 Box 4
2.1.3 Esfera 5
2.1.4 Cone 7
2.2 Engine 8

3 Conclusão **9**

Chapter 1

Introdução

1.1 Enunciado

Esta fase requer duas aplicações: uma para gerar arquivos com as informações dos modelos (nesta fase apenas gera os vértices para o modelo) e o próprio motor que irá ler um arquivo de configuração, escrito em XML, e exibe os modelos.

Gerador

Tem como objetivo gerar ficheiros com os pontos no espaço necessários para desenhar primitivas gráficas.

- Plano: é um quadrado no plano XZ, centrado na origem, subdividido nos eixos X e Z;
- Box: requer dimensões e o número de divisões por arestas, centrado na origem;
- Esfera: requer raio, altura, slices e stacks;
- Cone: requer o raio da circunferência que serve de base, altura, slices e stacks e a base deve estar no plano XZ.

Engine

Tem como funcionalidade interpretar um ficheiro XML com a configuração da câmera e os ficheiros com os pontos previamente gerados, e desenhar as primitivas.

Chapter 2

Apresentação das Soluções

2.1 Gerador

Com o auxílio de um conjunto de parâmetros, o gerador e capaz de produzir primitivas gráficas. Para tal, regista as coordenadas dos pontos que compõem a figura num arquivo com extensão ".3d", que pode ser posteriormente carregado e utilizado pelo motor gráfico para exibir a primitiva em questão.

2.1.1 Plano

Para criar um plano com o OpenGL criamos, primeiramente, a função 'createPlane' que cria um ficheiro com extensão ".3d" de coordenadas de vértices de um plano. Esta função recebe 3 parâmetros:

- 'length' - cumprimento do plano;
- 'divisions' - número de divisões do lado do plano;
- 'filename' - ficheiro onde se pretende escrever as coordenadas.

Este ficheiro é criado, mais em baixo, usando a função 'fopen' que abre o ficheiro para escrita.

A função começa por definir as variáveis a serem usadas na criação das coordenadas. Em seguida, após a criação do ficheiro, ocorre a iteração das divisões do plano, calculando as coordenadas x e z de cada vértice.

Para cada divisão do plano são escritos dois triângulos que compõem a superfície do plano. Por fim, o ficheiro é fechado e, se ocorrer um erro ao cria-lo, a função escreve uma mensagem de erro no ecrã.

O ficheiro .3d que será criado com as coordenadas dos vértices do plano será posteriormente executado e, usando o OpenGL, obtemos o seguinte resultado:

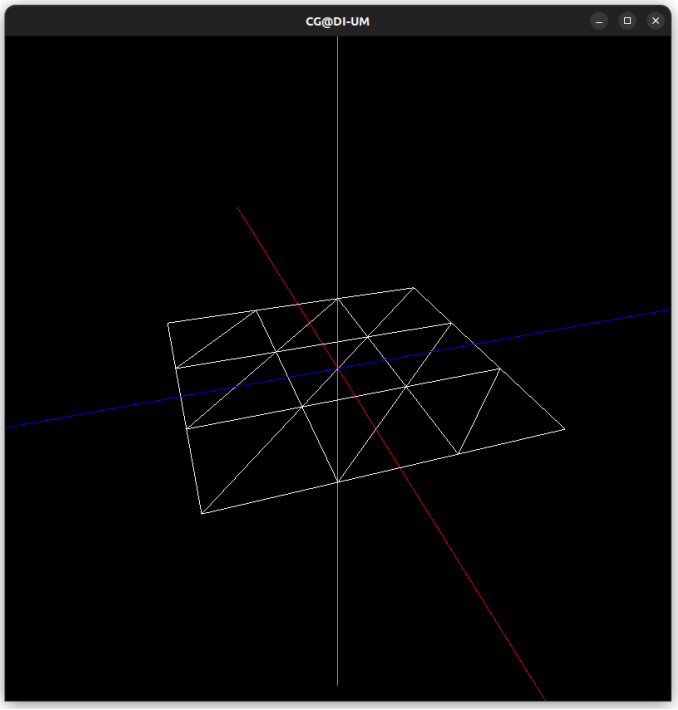

Figura 1: Plano gerado apartir do test_1_6.xml

2.1.2 Box

Para criar uma box com o OpenGL criamos, primeiramente, a função ‘createBox‘ que cria um ficheiro com extensão ”.3d” de coordenadas de vértices de uma caixa. A função ‘createBox‘ recebe 3 parâmetros:

- ’length’ - comprimento da caixa;
- ’divisions’ - número de divisões da caixa;
- ’filename’ - nome do ficheiro onde as coordenadas serão salvas.

A função começa por definir uma variável unidade que representa a largura de cada segmento da caixa, dividindo o comprimento pela quantidade de divisões. Em seguida, abre o ficheiro e se ele for aberto com sucesso, itera sobre cada segmento da caixa e escreve as coordenadas dos vértices correspondentes no ficheiro.

A caixa e construída a partir de 6 faces. Para cada face, a função itera sobre as divisões ao longo de cada eixo da caixa e calcula as coordenadas dos vértices usando as variáveis x1, x2, y1, y2, z1, z2. As coordenadas dos vértices são escritas no ficheiro, pela ordem x, y, z, separadas por espaços, e um caracter é adicionado ao final de cada linha para separar as coordenadas dos diferentes vértices.

O ficheiro .3d que ser a criado com as coordenadas dos vértices da caixa será posteriormente executado e, usando o OpenGL, obtemos o seguinte resultado:

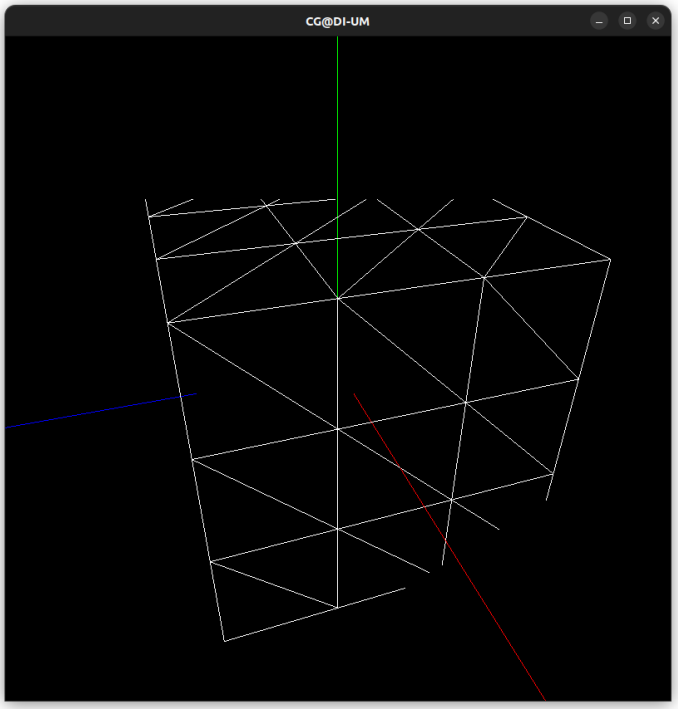

Figura 2: Caixa gerada apartir do test_1_4.xml

2.1.3 Esfera

Para criar uma esfera com o OpenGL criamos, primeiramente, a função ‘createSphere‘ que cria um ficheiro com extensão ".3d" de coordenadas de vértices de uma esfera. A função ‘createShpere‘ recebe 4 parâmetros:

- 'radius' - raio da esfera;
- 'slices' - número de fatias da esfera;
- 'stacks' - n umero de pilhas que compõem a esfera;
- 'filename' - nome do ficheiro onde as coordenadas serão salvas.

A função começa por definir algumas váriveis float que são usadas nos cálculos de coordenadas da esfera. Em seguida, abre o ficheiro. Se o ficheiro foi aberto com sucesso, a função usa dois ciclos "for" aninhados para iterar sobre as fatias e pilhas da esfera, calculando as coordenadas dos pontos que formam a superfície da esfera e escrevendo-as no ficheiro aberto.

Para cada fatia e pilha, a função calcula as coordenadas x, y e z de quatro pontos (x1, x2, x3, x4, y1, y2, z1, z2, z3, z4) usando as fórmulas trigonométricas da esfera. Em seguida, a função escreve as coordenadas dos quatro pontos num formato legível em ficheiros, desde que a pilha atual não esteja na borda inferior (j != 0) ou na borda superior (j != stacks - 1) da esfera.

Finalmente, a função fecha o ficheiro. Se o ficheiro não puder ser aberto, a função emite uma mensagem de erro.

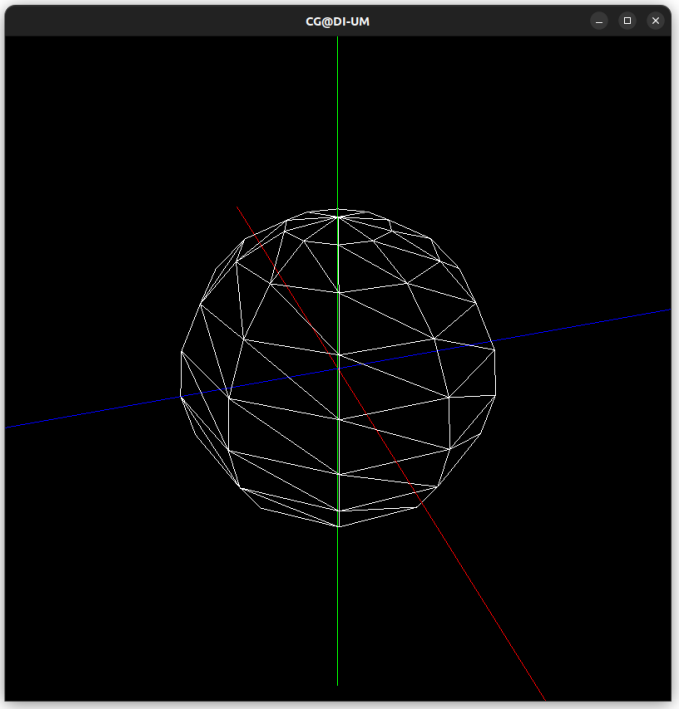

Figura 3: Esfera gerada apartir do test_1_3.xml

O ficheiro .3d que será criado com as coordenadas dos vértices da esfera será posteriormente executado e, usando o OpenGL, obtemos o resultado que podemos verificar na Figura 3.

2.1.4 Cone

Para criar um cone com o OpenGL criamos a função ‘createCone‘ que cria um ficheiro com extensão ”.3d” de coordenadas de vértices de um cone. A função ‘createCone‘ recebe 5 parâmetros:

- ’radius’ - raio da base do cone;
- ’height’ - altura do cone;
- ’slices’ - número de fatias horizontais que dividem o cone;
- ’stacks’ - número de fatias verticais que dividem o cone;
- ’filename’ - nome do ficheiro onde as coordenadas serão salvas.

A primeira parte do código cria a base do cone, dividindo-o em fatias horizontais. Para cada fatia, a função calcula as coordenadas dos três vértices do triângulo que a compõe. Os vértices são calculados usando trigonometria para determinar as coordenadas x, y e z de cada ponto.

A segunda parte do código cria as fatias verticais do cone. Para cada fatia, a função calcula as coordenadas de quatro vértices, que formam dois triângulos que cobrem a superfície do cone nessa fatia. As coordenadas são calculadas usando trigonometria, assim como na primeira parte do código, mas tendo em consideração a altura da fatia em questão.

As coordenadas de cada triângulo são escritas no ficheiro de saída. No final da função, o ficheiro é fechado. Se ocorrer algum erro ao abrir o ficheiro de saída, a função exibe uma mensagem de erro.

O ficheiro .3d que será criado com as coordenadas dos vértices do cone será posteriormente executado e, usando o OpenGL, obtemos o seguinte resultado:

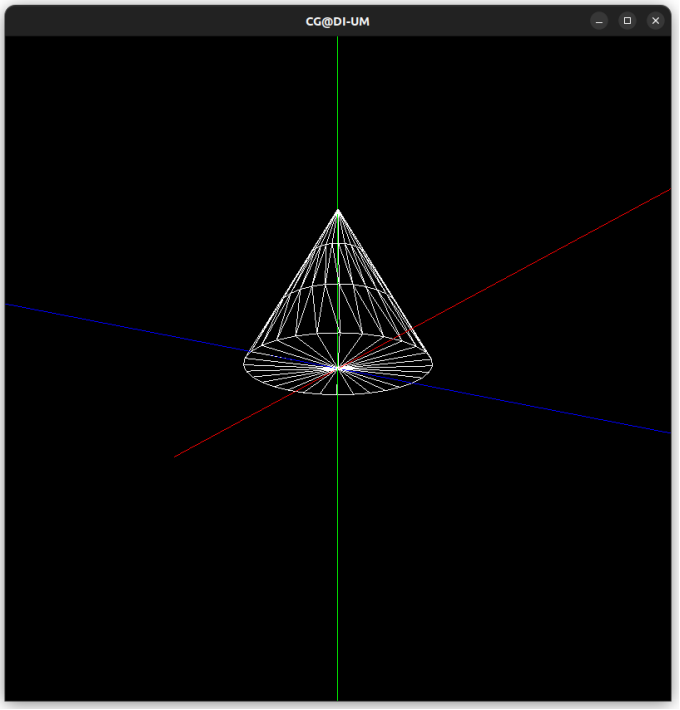

Figura 4: Cone gerado apartir do test_1_1.xml

2.2 Engine

O engine interpreta um ficheiro XML e com a informação nele contido irá gerar uma cena. Para que o engine tivesse esta funcionalidade, tomou-se vantagem do parser tinyXML. Após ser executado o parser extrai a seguinte informação do ficheiro:

- Câmara
  - Posição da câmara
  - Ponto para onde a câmara está a olhar (lookAt)
  - Inclinação da câmara;
- Modelos
  - Nome dos ficheiros .3d a carregar

O engine tem implementado uma câmara que permite rodar a imagem em torno da cena através de teclas que possibilitam aproximar e afastar a câmara da origem do referêncial.

Chapter 3

Conclusão

Numa fase inicial da realização desta primeira fase do trabalho surgiram alguns desafios tais como a compreensão de como funcionam dos ângulos de figuras curvas, como o cone e a esfera. No entanto, consideramos que foi ultrapassada facilmente.

Para além disso, vimos também alguma dificuldade em compreender o ficheiro tinyXML e como implementá-lo.

Finalmente, apesar das dificuldades, os objetivos foram atingidos, pelo que consideramos que obtivemos um trabalho bom.