Computação Gráfica
(3º ano de LCC)
Trabalho Prático - Parte 1
Relatório de Desenvolvimento

Francisco Oliveira (A82066)
Luís Costa (A70070)
Maria Luísa Silva (A82124)
Tierri Monteiro (A76359)

6 de Março de 2020

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

No âmbito da Unidade Curricular de Computação Gráfica, foi-nos proposto a realização deste trabalho prático. Este trabalho foi dividido em 4 fases, sendo esta a primeira. Por sua vez, tem como objetivo, a criação de algumas primitivas gráficas.

1.2 Resumo

Nesta primeira fase foi necessário a criação de um gerador, o qual é responsável por gerar a informação dos modelos(plano,caixa,esfera e cone), guardando os vértices. Além disso, também foi necessário a criação de um *Engine* (por nós assim designado), responsável pela leitura da configuração de um ficheiro XML e exibir os respetivos modelos.

Capítulo 2

Arquitetura do Código

2.1 Primitivas

2.1.1 Plane

A primitiva *plano* ("Plane") é um quadrado no plano XZ, que está centrado na origem, e tem apenas 1 parâmetro: **size** (tamanho), que indica o comprimento dos lados do quadrado (em **float**).

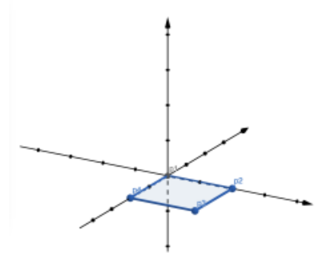

Figura 2.1: Plano

Com isto, podemos ver que o plano tem 4 vértices: *p1 = (0,0,0), p2 = (size,0,0), p3 = (size,0,size) e p4 = (0,0,size).*

Como queremos que o plano se encontre centrado na origem, temos de alterar as coordenadas dos pontos de forma a que isto seja possıvel. Para isto, basta descobrir a metade do size dos lados do quadrado, e fazer uma espécie de "translação"deste valor tanto no eixo do X como no do Z. Seja s' = size / 2, Podemos ver que o plano se encontra centrado, sendo os vértices: *p1 = (-s',0,-s'), p2 = (s',0,-s'), p3 = (s',0,s') e p4 = (-s',0,s').*

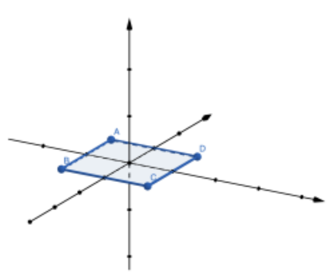

Figura 2.2: Plano Centrado

2.1.2 Box

Uma caixa, tal como um plano, pode ser interpretada como um conjunto de triângulos. Para esta ser gerada, precisamos de uma largura **'X'**, uma altura **'Y'** e um comprimento **'Z'**, e opcionalmente o número de divisões de cada face (caso não seja passado como parâmetro, tem valor *default* 1).

Cada face da caixa irá conter quadriláteros igual ao número de *divisões*∗*divisões*.

Esses quadriláteros vão ter as seguintes dimensões:

- lx = x/div
- ly=y/div
- lz=y/div

Para depois definir um triângulo em cada uma das faces do quadrilátero utiliza-se uma sequência de vértices cujos pontos são calculados desta forma :

- Ponto A : (lx∗i-x/2, ly∗t-y/2 , z/2)
- Ponto B : (lx∗ix/2+lx, ly ∗ ty/2 , z/2)
- Ponto C : (lx∗ix/2,ly∗ty/2+ly, z/2)

Tendo em conta que *i* e *t* vão estar entre 0 e o numero de divisões.

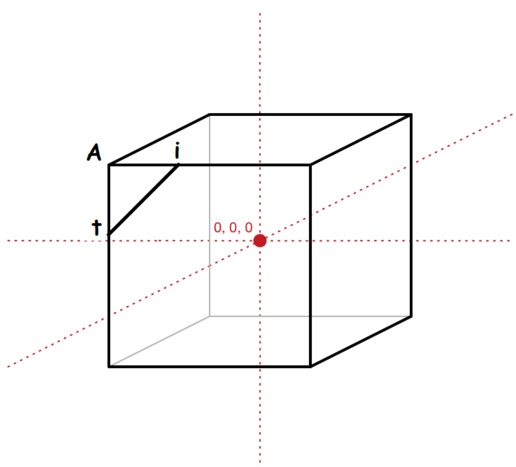

Figura 2.3: Caixa

2.1.3 Cone

A primitiva *cone* ("Cone") está centrada na origem, e tem 4 parâmetros: **radius**, **height**, **slices** e **stacks**.

Sejam **sl** o número de slices e **st** o número de stacks. A base do cone é um círculo com raio = **radius**. Como estamos a trabalhar com triângulos, este circulo é obtido construindo **sl** triângulos apontados para o centro. É então necessário descobrirmos um ângulo α tal que cada triângulo tem este formato, Este ângulo é calculado (em radianos) pela seguinte fórmula:

$$\alpha = 2\pi \ / \ slices$$

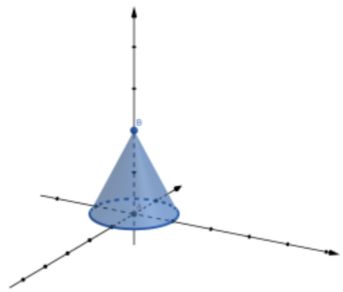

Figura 2.4: Cone

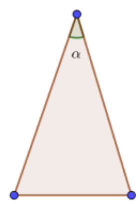

Figura 2.5: Triângulo

Tendo o raio e o α, já temos tudo o que precisamos para construir a base. A face lateral é construıda stack a stack, onde por cada stack construímos **sl** trapézios. Cada stack tem uma altura h, que é calculada pela equação:

$$h = height/\ stacks$$

Cada ponto do trapézio é descoberto utilizando a seguinte fórmula:

$$P = (r' * sin\ \alpha',\ h',\ r' * cos\ \alpha')$$

onde r' é a distância do ponto ao eixo do Y, h' é a distância do ponto ao plano **XZ** e α é o ângulo do ponto em relação ao eixo do **Z**. Se tivermos o trapézio seguinte,

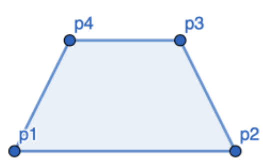

Figura 2.6: Trapézio

e seja $p1 = (r' * sin\ \alpha',\ h',\ r' * cos\ \alpha')$, então conseguimos encontrar os restantes pontos a partir deste:

$$p2 = (r' * sin(\alpha' + \alpha),\ h',\ r' * cos(\alpha' + \alpha))$$
$$p3 = (r' * sin(\alpha' + \alpha),\ h' + h,\ r' * cos(\alpha' + \alpha))$$
$$p4 = (r' * sin\ \alpha',\ h' + h,\ r * cos\ \alpha')$$

Sabendo descobrir as coordenadas de cada ponto, conseguimos construir os trapézios, ou mais especifica- mente, os triângulos que constroem os trapézios, e tendo isto, conseguimos fazer o cone.

2.1.4 Sphere

A primitiva *esfera* (”Sphere”) está centrada na origem, e tem 3 parâmetros: **radius**, **slices** e **stacks**. Começamos por construir os 2 polos da esfera. Estas 2 stacks são feitas separadamente porque , ao Figura 2.7: Esfera contrário das restantes, são formadas por triângulos e não por trapézios. Nestes triângulos, o vértice da ”ponta”situa-se no eixo do Y.

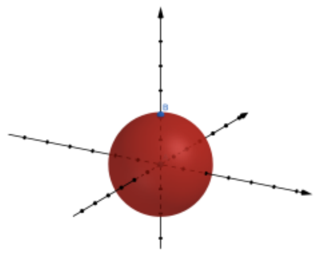

Figura 2.7: Esfera

Seja t o triângulo da figura anterior, os seus vértices são:

1. **Pólo Norte**

   $p1 = (radius * cos(\beta + \beta') * sin(\alpha), radius * sin(\beta + \beta'), radius * cos(\beta + \beta') * cos(\alpha))$

   $p2 = (0, -radius, 0)$

   $p3 = (radius * cos(\beta + \beta') * sin(\alpha + \alpha'), radius * sin(\beta + \beta'), radius * cos(\beta + \beta') * cos(\alpha + \alpha'))$

2. **Pólo Sul**

   $p1 = (radius * cos(\beta) * sin(\alpha), radius * sin(\beta), radius * cos(\beta) * cos(\alpha))$

   $p2 = (0, radius, 0)$

   $p3 = (radius * cos(\beta) * sin(\alpha + \alpha'), radius * sin(\beta), radius * cos(\beta) * cos(\alpha + \alpha'))$

No que toca à face lateral, a estratégia é semelhante à do cone. Construmos stack a stack, e por cada stack construmos trapézios de quantidade igual ao número de slices. Os pontos seguem o mesmo formato do trapézio do cone, mas agora temos de ter em atenção o ângulo β . Vejamos novamente o trapézio da figura 11.

Se $p1 = (r * cos(\beta) * sin(\alpha), r * sin(\beta'), r * cos(\beta') * cos(\alpha))$, então:

$p2 = (r * cos(\beta') * sin(\alpha' + \alpha), r * sin(\beta'), r * cos(\beta') * cos(\alpha' + \alpha))$

$p3 = (r * cos(\beta' + \beta) * sin(\alpha' + \alpha), r * sin(\beta' + \beta), r * cos(\beta') * cos(\alpha' + \alpha))$

$p4 = (r * cos(\beta' + \beta) * sin(\alpha'), r * sin(\beta' + \beta), r * cos(\beta') * cos(\alpha'))$

onde α' e β' são os ângulos dos pontos em relação ao referencial e r o raio.

Capítulo 3

Aplicações

3.1 Generator

O gerador é um programa que recebe como *input* o nome da primitiva e os argumentos necessários para gerar os seus pontos. Todas as primitivas têm como unidade construção triângulos.Para assegurar que o número de argumentos na criação de cada figura está correto, cria-mos uma função que imprime no ecrã uma mensagem de erro caso algo esteja incorreto.

3.2 Engine

Respeitando o enunciado, é necessária a criação de uma aplicação que seja capaz de interpretar ficheiros de configuração **XML** com o correspondente formato.

O **Engine**, assim denominado por nós, é responsável pela leitura desse tal ficheiro em **XML**, que opera lendo um nome de ficheiro passado como primeiro argumento é interpretado conforme o pedido.

Após esta fase inicial de interpretação e armazenamento, o **Engine**, com recurso às bibliotecas **GLUT** e **OpenGL**, inicializa uma cena 3D e renderiza os vértices, respeitando sempre a obrigação de desenhar apenas triângulos, ou seja, desenhando 3 vértice de cada vez com a função do **OpenGL** *glVertex3f*.

3.3 Exemplos

Executando o gerador (*”Generator”*) com as diferentes primitivas obtemos os diversos ficheiros (”plane.3d”,”box.3d”,etc) necessários para que o motor (*”Engine”*), efectuando a leitura do documento *”scene.xml”*, desenhe os modelos pretendidos.

```xml
<scene>
<model file= "../../generator/build/plane.3d"/>
</scene>
```

Figura 3.1: scene.xml (plano)

```
./Generator plane 2 plane.3d
```

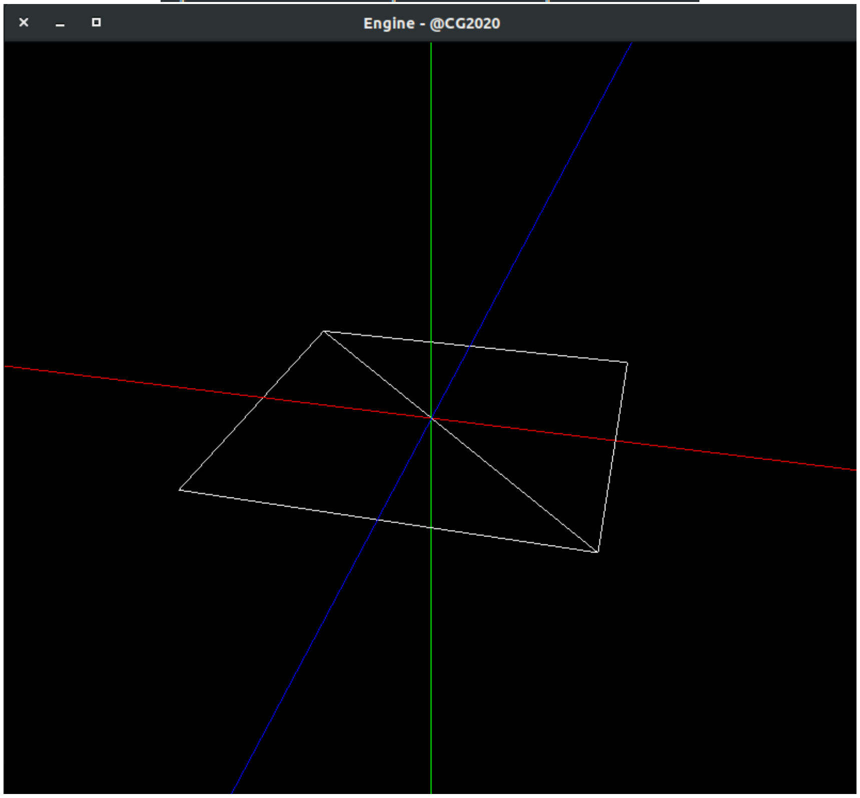

Figura 3.2: Plano

```
./Generator box 1.5 1 2 box.3d
```

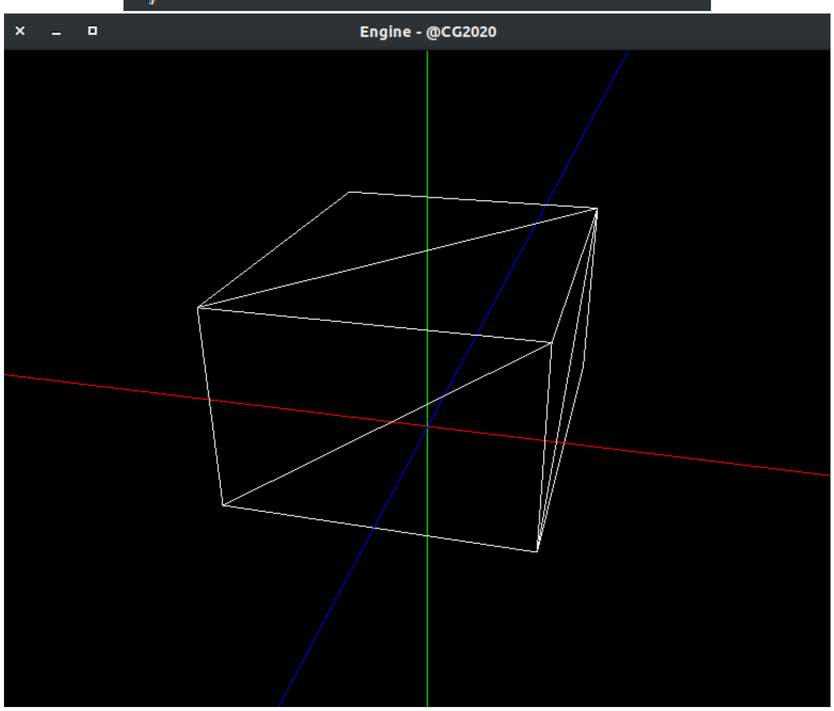

Figura 3.3: Caixa

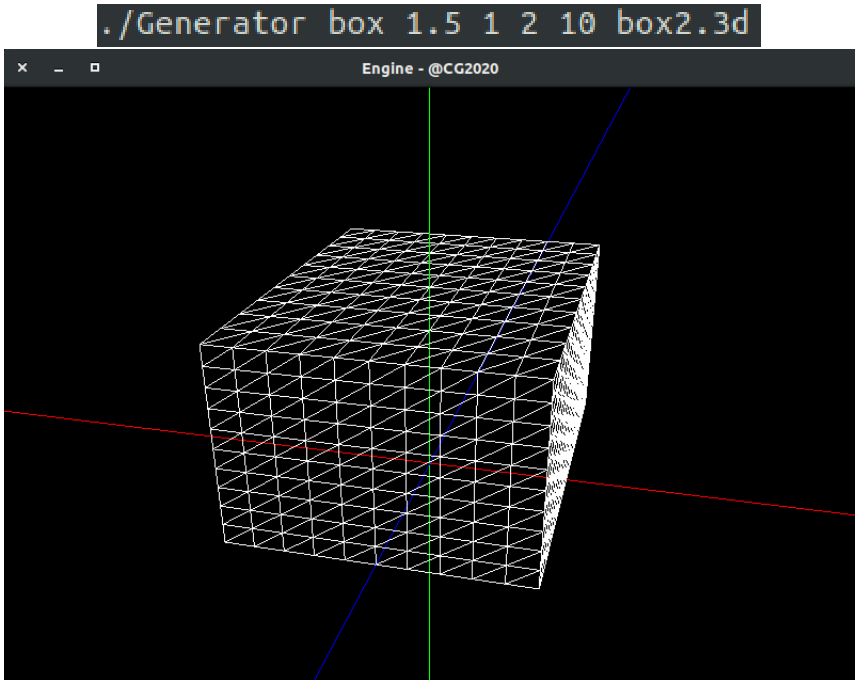

Figura 3.4: Caixa (com divisões)

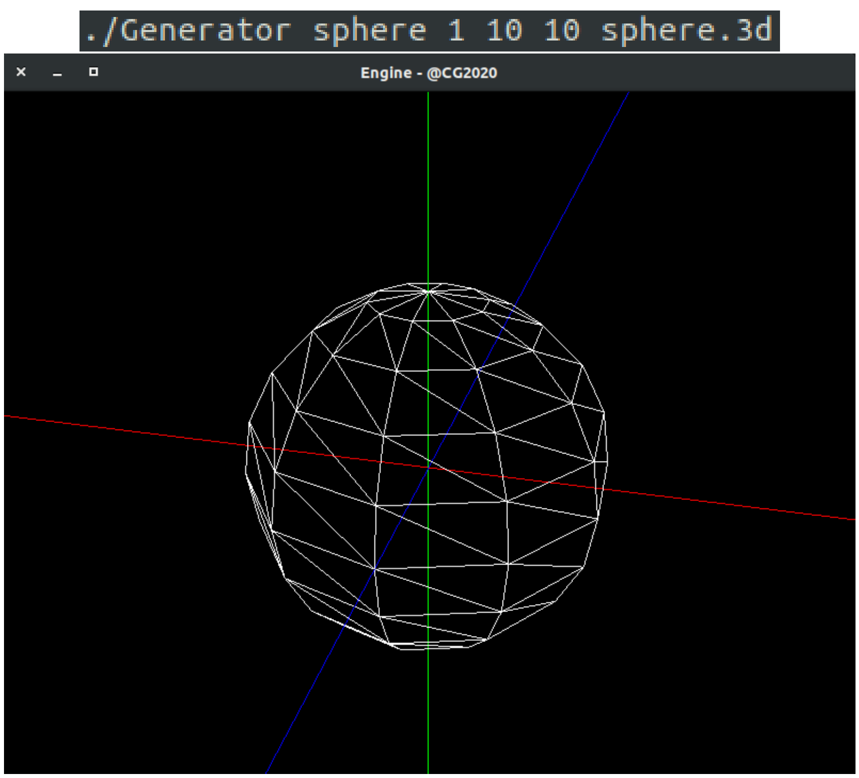

Figura 3.5: Esfera

```
./Generator cone 1 1.5 10 10 cone.3d
```

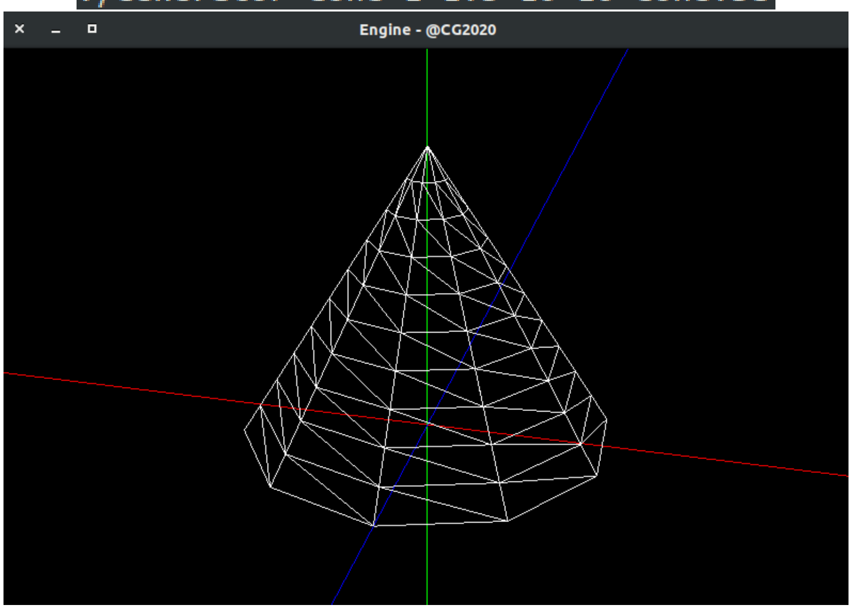

Figura 3.6: Cone

Capítulo 4

Conclusão

Esta primeira fase do trabalho foi bastante importante para nós, porque nos permitiu consolidar os conhecimentos abordados nesta Unidade Curricular.

Para além disto, ensinou-nos a usar ferramentas para modelação 3D e em simultâneo aprender **C++**.

Por último, fazemos um balanço positivo desta primeira fase, e esperemos que com esta parte já feita continuemos com motivação para as próximas partes do trabalho.