

UNIVERSIDADE DO MINHO
DEPARTAMENTO DE INFORMÁTICA

Computação Gráfica - Fase 1
Grupo N°2

João Correia (A84414)

Marco Pereira (A89556)

Pedro António (A58062)

Rúben Cerqueira (A89593)

28 de junho de 2021



João



Marco



Pedro



Rúben

Capítulo 1

Introdução

O presente relatório visa apresentar a primeira fase do trabalho proposto no âmbito da Unidade Curricular de Computação Gráfica.

Esta fase é composta pelo desenvolvimento de duas aplicações. A primeira sendo um generator, responsável por gerar as primitivas geométricas abaixo apresentadas, assim como guardar em ficheiros .3d as informações sobre os pontos e/ou triangulos de cada primitiva. Seguidamente, a segunda aplicação, o engine, encarrega-se de interpretar o ficheiro com a ajuda de um ficheiro de configuração em XML, a fim de apresentar ao utilizador a primitiva geométrica desejada.

As primitivas criadas nesta fase, bem como a informação dada para a sua criação, estão a seguir descritas:

- Plane (a square in the xz plane, centered in the origin, made with 2 triangles)
- Box (requires x, y and z dimensions, and optionally the number of divisions)
- Sphere (requires radius, slices and stacks)
- Cone (requires bottom radius, height, slices and stacks)

Capítulo 2

Descrição das Aplicações

2.1 Generator

O generator é responsável por criar cada primitiva geométrica. Para este efeito, criaram-se classes para suportar a definir cada Ponto e Triângulo a serem escritos para o ficheiro de output.

O ficheiro de output é formado por uma linha na qual está a informação de quantos vértices e triângulos irá ler, seguida de linhas de três valores separados por espaços, que representam as coordenadas dos pontos de cada triângulo.

Depois da indicação de todos os Pontos, escrevemos adicionalmente a informação relativa a cada triângulo. Os 3 valores apresentados em cada uma das linhas significam as posições do vetor onde os pontos do triângulo se encontram (e.g.: 0 1 2 é o triângulo formado pelos pontos das linhas 1 2 e 3).

2.2 Engine

O engine está apenas responsável por ler o ficheiro criado pelo generator e por criar o vetor de triângulos a desenhar, apresentando os mesmos no ecrã do utilizador. É ainda possível utilizar os seguintes controlos:

- 'w' e 's' para aproximar/afastar a câmara em relação à origem
- 'a' e 'd' ou utilizar o rato para rodar a primitiva
- apresentar os eixos pressionando *spacebar*
- 'p' para apresentar a primitiva em modo wired

Este programa tem como suporte um ficheiro de configuração XML para que a aplicação saiba quais os ficheiros *.3d* a ler para sua visualização.

A aplicação suporta vários ficheiros XML que contenham uma tag *scene* englobando várias tag *model* aninhadas, correspondentes aos modelos gerados pelo Generator que pretendemos importar.

Existe a possibilidade de ler uma configuração específica correndo a engine, dando como argumento o ficheiro de configuração que se pretende utilizar.

Caso o utilizador não especifique o ficheiro XML a utilizar, será aberto por defeito o ficheiro "config.xml", que contém um plano e uma esfera.

Adicionalmente, estão disponíveis os seguintes ficheiros XML já previamente gerados:

- "sphere.xml": contém uma esfera.
- "box.xml": contém uma caixa.
- "cone.xml": contém um cone.
- "plane.xml": contém um plano.

Capítulo 3

Formulações de cada primitiva gráfica

3.1 Plano

O plano foi a primitiva mais simples de desenhar, sendo apenas constituído por 4 pontos. Estes são calculados através de metade do valor introduzido pelo utilizador, de forma a centrar o plano na origem. Depois, basta apenas criar cada um dos triângulos.

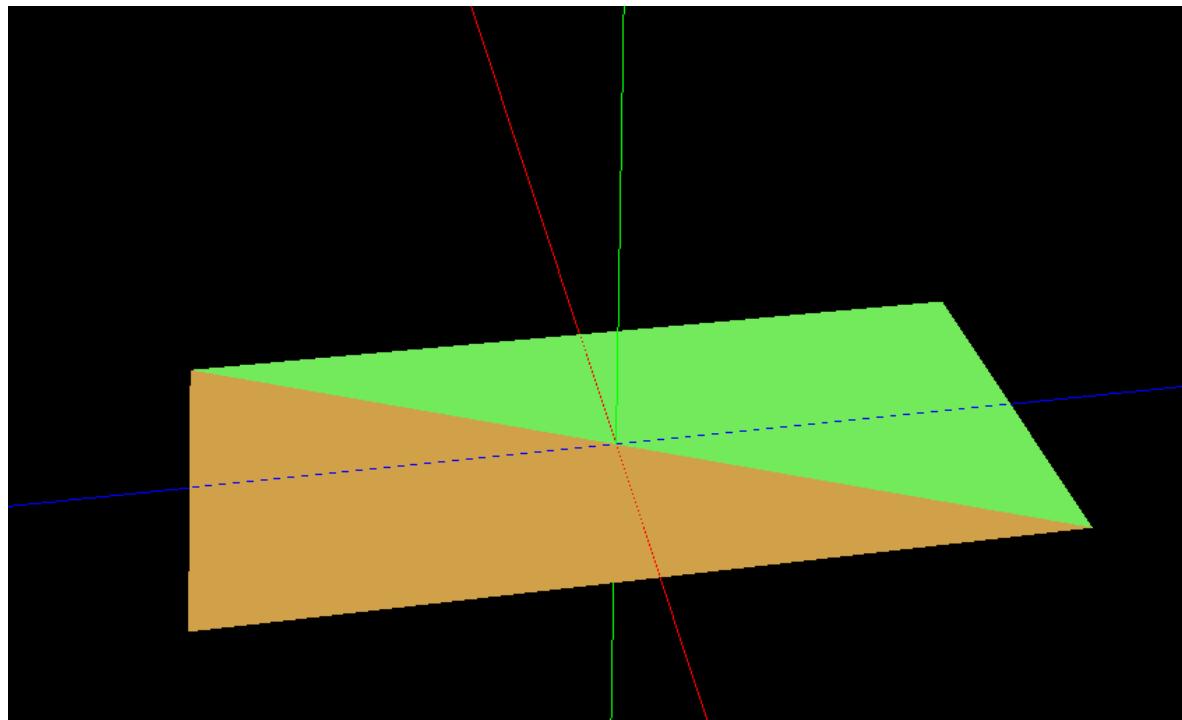


Figura 3.1: Plano

3.2 Box

A caixa recebe como valores a altura, o comprimento, a profundidade e também o número de divisões. A forma como calculamos os pontos na caixa, passa por saber qual será o espaço ocupado por cada divisão, que é calculado dividindo o espaço de uma divisão pelo número de divisões.

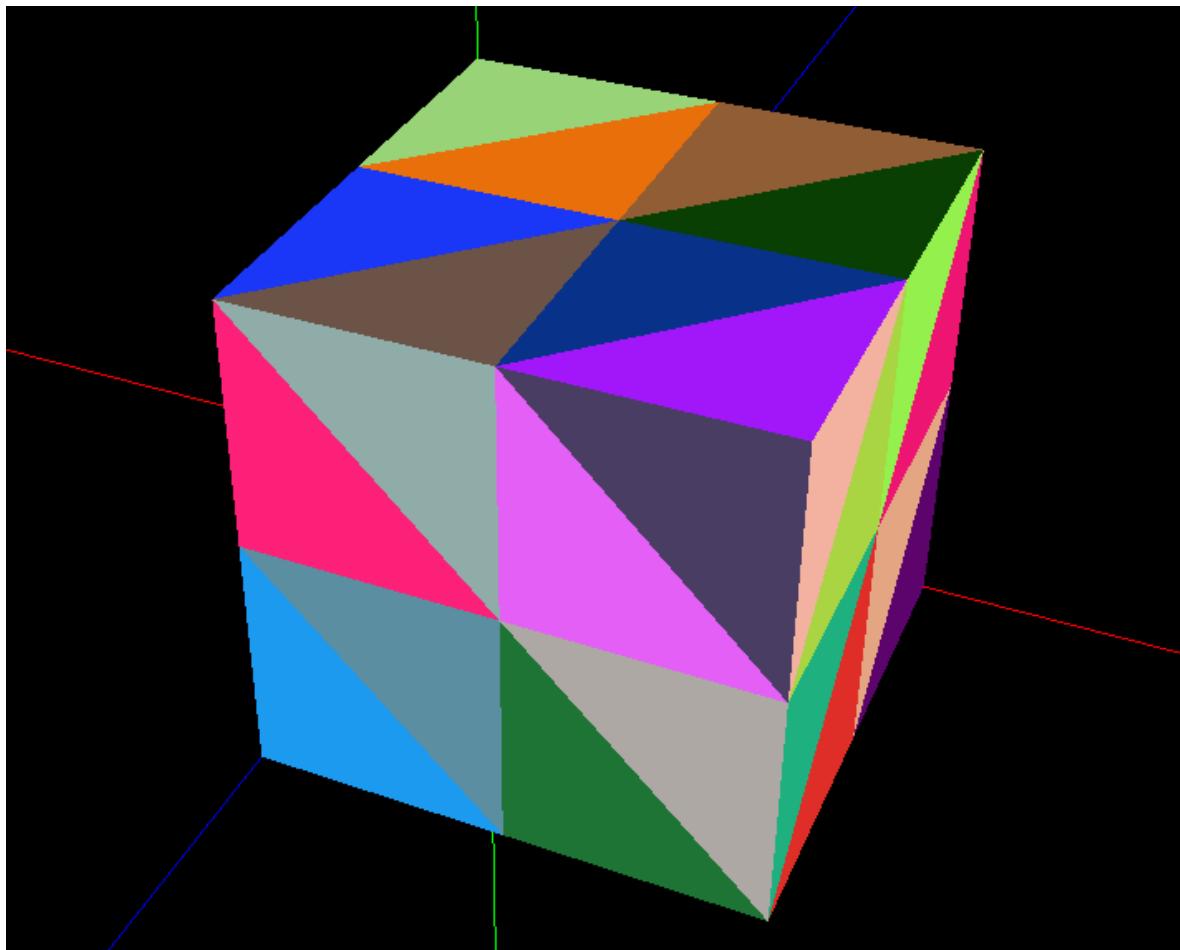


Figura 3.2: Box

3.3 Esfera

Para o cálculo dos pontos pertencentes à esfera recorremos aos ângulos e ao raio da mesma, usando-os como coordenadas polares e obtendo depois coordenadas cartesianas. A seguir, apresentamos as fórmulas de cálculo de cada ponto (Assumindo θ como o ângulo formado na vertical começando a partir da parte positiva do eixo Y, direcionado à parte negativa do eixo Y e α como o ângulo formado na horizontal, que começa na parte positiva do eixo X, em direção contrária aos ponteiros do relógio, para a parte negativa do eixo Z):

$$y = \text{radius} * \cos(\theta)$$
$$x = \cos(\alpha) * \sin(\theta) * \text{radius}$$
$$z = -\sin(\alpha) * \sin(\theta) * \text{radius}$$

Depois do cálculo de cada ponto precisamos apenas de ir formando os triângulos dois a dois a cada iteração, por forma a obter quadrados. Na primeira e última iterações é utilizado apenas um triângulo por iteração de modo a não repetir pontos e triângulos, obtendo-se assim o seguinte sólido:

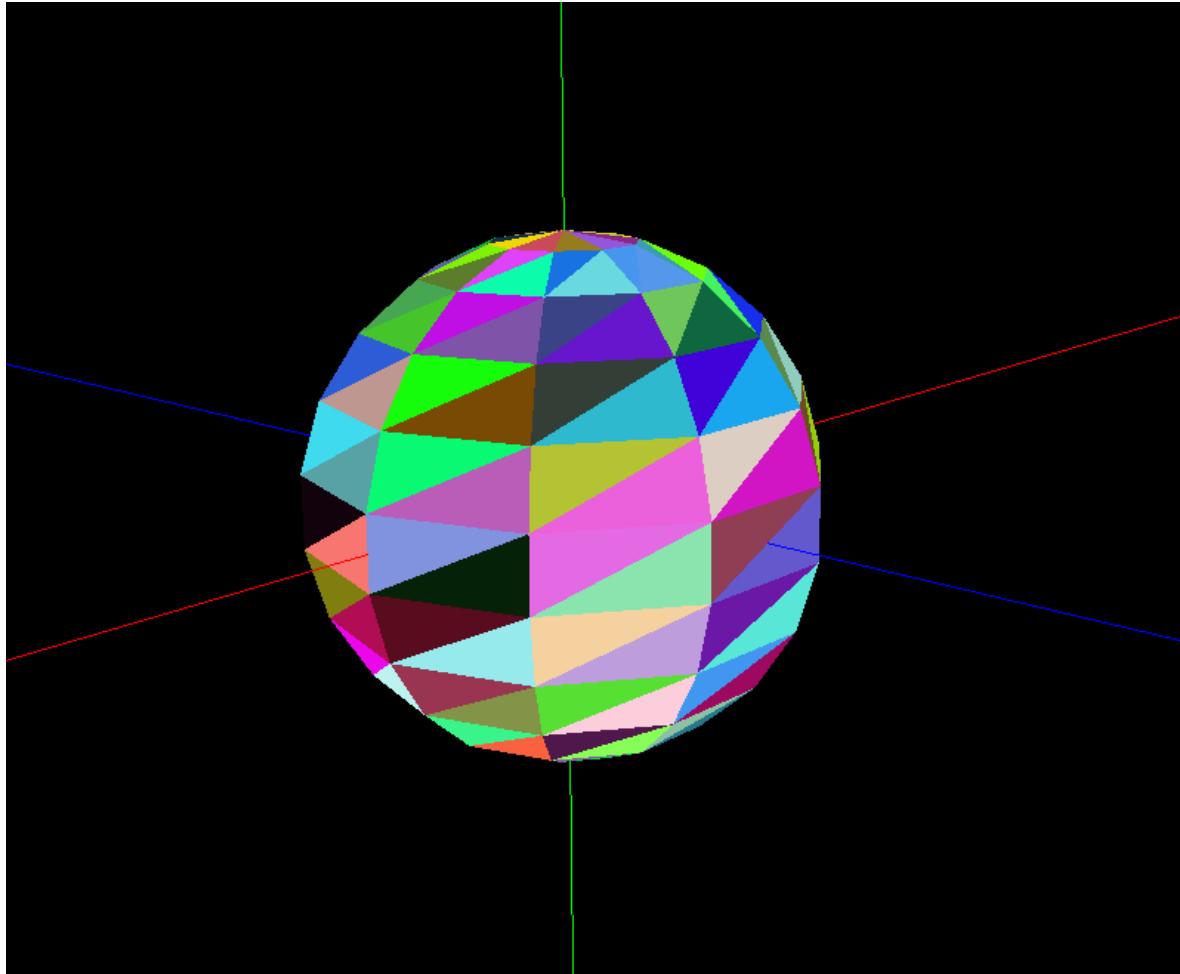


Figura 3.3: Esfera

3.4 Cone

Para o cálculo dos pontos pertencentes ao cone começámos por fazer a divisão da altura pelo número de stacks+1, de modo a obter o incremento de altura por stack. Para o cálculo das componentes x, y, e z foi usado o valor do raio de cada stack, e aplicadas as seguintes fórmulas:

$$height_increment = height/(nStacks + 1)$$

$$y = height_increment * stack$$

$$radiusStack = (height - y) * radius/height$$

$$x = \cos(\alpha) * radiusStack$$

$$z = -\sin(\alpha) * radiusStack$$

Após estes cálculos tivemos apenas que iterar os pontos de modo a criar os devidos triângulos, obtendo o seguinte:

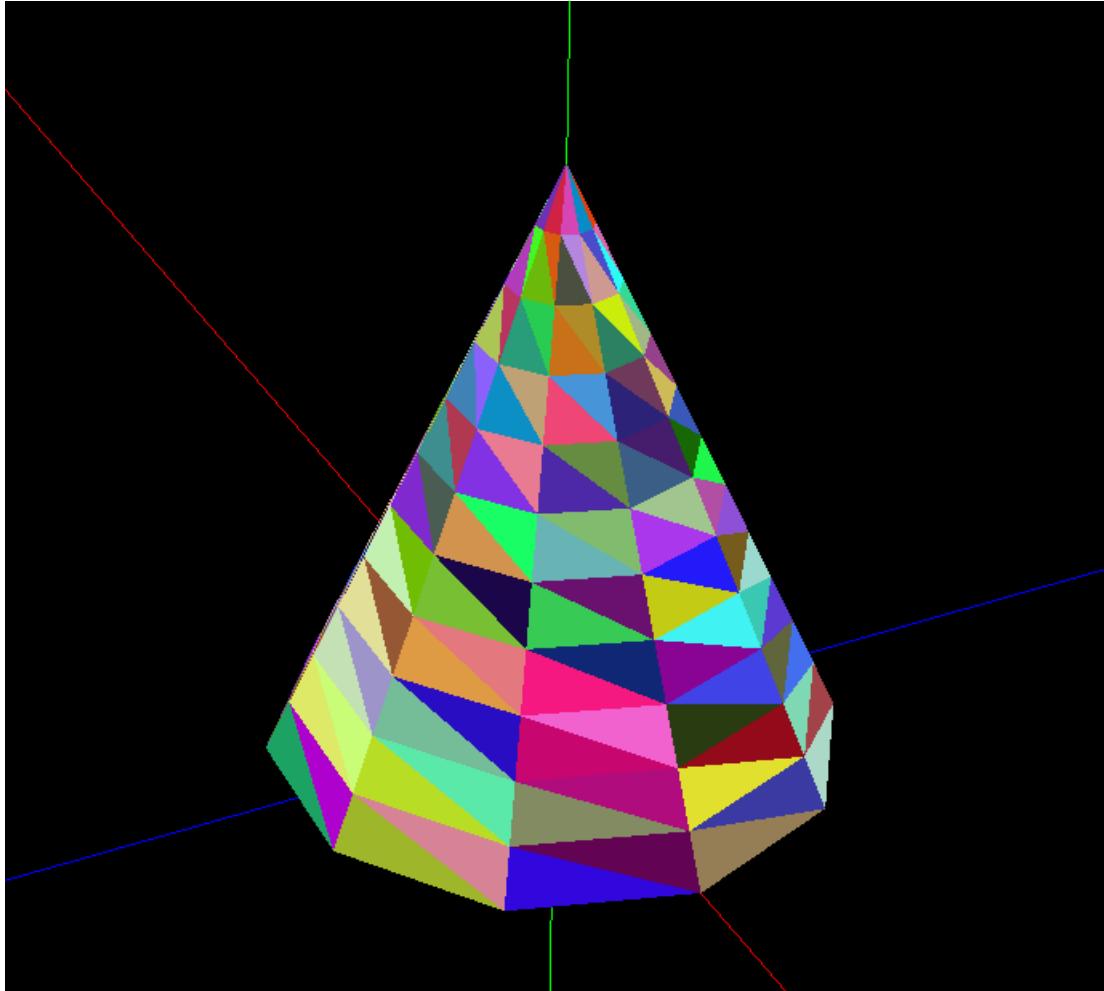


Figura 3.4: Cone

Capítulo 4

Conclusão

A realização da primeira fase do trabalho permitiu a consolidação dos conhecimentos aprendidos nas aulas da UC, bem como o aperfeiçoamento da utilização das ferramentas que o OpenGL facilita, e das particularidades da programação em C++.

Ao longo desta fase, procedeu-se ao desenvolvimento de um simples gerador (*generator*) de ficheiros .3d com as instruções de desenho, e de um interpretador (*engine*) que lê os ficheiros, desenhando conforme os dados obtidos. Através da elaboração dos mesmos pudemos obter uma base sólida para a progressão do trabalho nas seguintes etapas, que se vão centrar nestes dois programas de forma completamente iterativa.