Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de
Computação Gráfica

Ano Letivo de 2021/2022

Relatório Fase 1
Primitivas Gráficas

Grupo

Pedro Araújo	a90614
Pedro Fernandes	a84313
João Cardoso	a94595
Rita Gomes	a87960

13 de Março de 2022

Índice

1 Introdução **1**

2 Formalização do problema **2**
2.1 Plano 2
2.2 Caixa 3
2.3 Esfera 4
2.4 Cone 5

3 Desenho das primitivas **7**
3.1 *Engine* 7
3.2 Generator 8

4 Conclusão **9**

1 Introdução

O presente relatório serve de suporte ao trabalho realizado no âmbito da unidade curricular de Computação Gráfica. Este trabalho está dividido em quatro fases, sendo que neste relatório será analisada toda a formalização e modelação envolvida na realização da primeira fase.

O objetivo final deste projeto é desenvolver um motor 3D baseado num minigráfico de cena e consequentemente fornecer um conjunto de exemplos que demonstram o seu potencial.

Nesta primeira fase, foi proposto aos alunos que procedessem à criação de duas aplicações: uma para gerar ficheiros com a informação resultante da elaboração de vértices de acordo com os argumentos dados (generator) e outra que irá permitir a visulização dos modelos a partir dos ficheiros gerados (*engine*). O gerador de vértices de primitivas deverá ser capaz de gerar as seguintes primitivas, de acordo com os argumentos fornecidos:

1. **Plano:** um quadrado no plano XZ, centrado na origem, subdividido nas direções X e Z.
2. **Caixa:** requer dimensões X, Y e Z e o número de divisões por aresta.
3. **Esfera:** necessita de raio, slices e stacks.
4. **Cone:** precisa do raio da base, altura, slices e stacks.

Para além disso, a geração de vértices deverá também, ter como destino, um ficheiro .3d, de forma a facilitar a interpretação do mesmo pelo motor gráfico. O motor gráfico é responsável por receber ficheiros do tipo .xml, capazes de referenciar modelos em formato .3d, e renderizar os mesmo fazendo recurso ao OpenGL. Por último, é solicitado aos alunos a definição de estruturas de dados com intuito de armazenar as informações relativas aos modelos, em memória.

2 Formalização do problema

2.1 Plano

Fazemos uso de 3 parâmetros na criação do plano: comprimento do lado do quadrado, número de divisões ao longo de cada eixo e o respetivo ficheiro .3d (plane.3d).

Para que o quadrado se mantenha centrado na origem e no plano XZ, a coordenada y dos vértices será 0 e a dimensão das restantes irá do ponto *(comprimento/ -2)* até ao ponto (comprimento/2).

Para desenhar um quadrado são necessários quatro pontos, a partir desses são desenhados dois triângulos, de forma a formarem um quadrado, para N divisões são necessários NxN quadrados. Para o desenvolvimento dos quadrados foi utilizada a técnica matricial, desenvolvendo as colunas de cada linha, partindo do referencial (comprimento/-2) e avançando (comprimento/divisões).

O resultado obtido pode ser observado na figura a seguir.

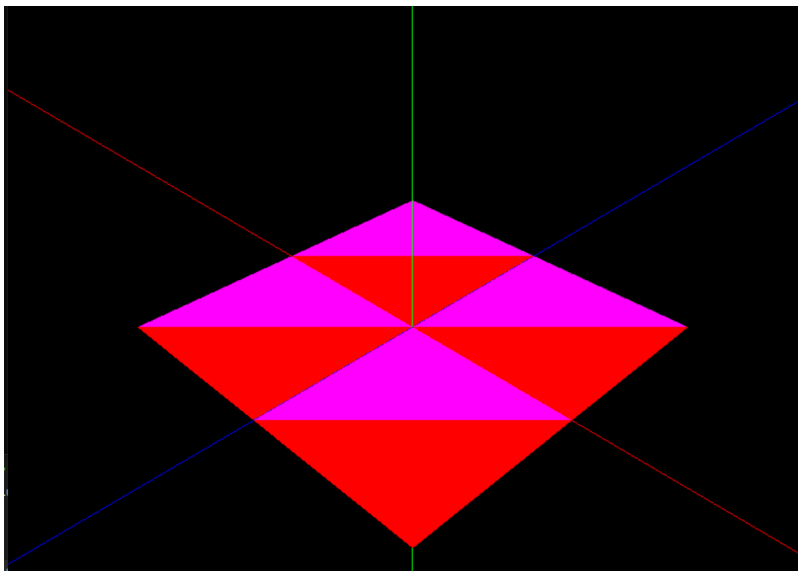

Figura 2.1: Plano XZ centrado, na origem

2.2 Caixa

No processo de criação da caixa são necessários 3 parâmetros: o primeiro corresponde à medida de cada dimensão, o seguinte representa o número de divisões de cada face e o último diz respeito ao nome do ficheiro .3d (box.3d).

Idealizamos a caixa de forma a estar centralizada na origem em cada dimensão e, para isso, dividimos o parâmetro do comprimento de cada dimensão por -2, para assim construir cada face: metade numa parte negativa e a outra na parte positiva.

Para a criação de cada face foi utilizado parte do raciocínio do desenvolvimento do Plano, no que toca a resolução das divisões, utilizando técnicas matriciais.

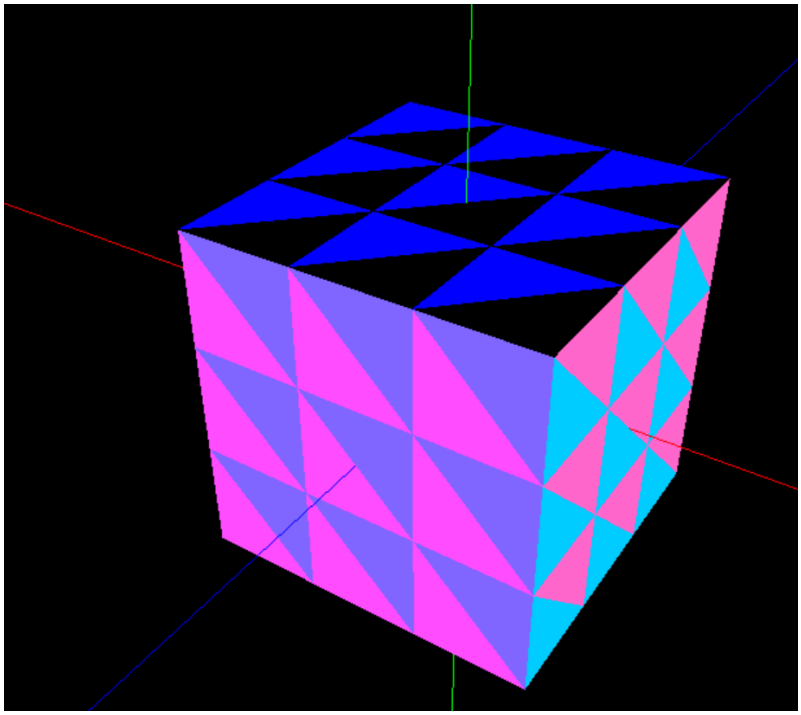

Figura 2.2: Caixa com dimensão 1x1x1, centrada na origem e com 3 divisões.

2.3 Esfera

A esfera, devido a sua natureza geométrica, revela-se mais complexa que as figuras prévias.

Em primeiro lugar, temos que considerar que serão recebidos 3 parâmetros: **número de slices**, **número de stacks** e o **raio** que se pretende que esta primitiva tenha. Estes dados serão ponto de partida para se poder formar a figura, utiliziando triângulos com unidade para construção da mesma, tal como foi feito até agora. É importante realçar a importância dos valores destes argumentos, visto que para que haja coerência, os valores passados também deverão ter algum fundamento.

Deste modo, é essencial que haja um valor relativamente alto de slices e stacks. A lógica por trás disto depende de um fator crucial para o desenvolvimento desta esfera: são precisos 4 pontos inciais, de modo a poder derivar os restantes pontos que irão compor esta primitiva.Esta derivação sucede devido à necessidade de obter coordenadas esféricas, em contraste com o que se fez para outras primitivas geométricas.

Para se obter estas coordenadas foram utilizadas as seguintes fórmulas na função auxiliar *polar2Cart*. Note que o φ da imagem corresponde ao ângulo que com a soma do β, no nosso código, faz um ângulo reto, e o θ coincide com o α:

$$x = r \cdot sin(\theta) \cdot cos(\varphi)$$
$$y = r \cdot sin(\varphi)$$
$$z = r \cdot cos(\theta) \cdot cos(\varphi)$$

$$\text{Onde , } \varphi = [0, \pi] \text{ e } \theta = [0, 2\pi[$$

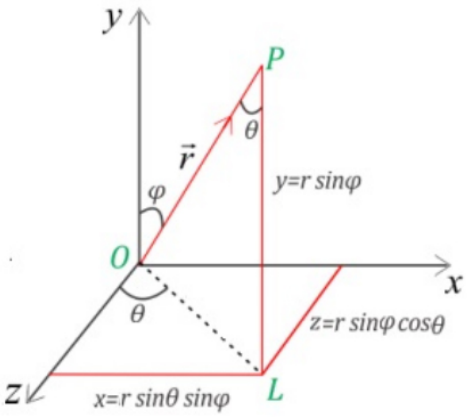

Figura 2.3: Fórmulas utilizadas

A partir deste ponto, tornou-se relativamente fácil a construção da esfera. Cada ponto desloca-se relativamente ao ponto inicial (digamos Ponto X), através da variação de θ e de φ com o respetivo deslocamento, calculado através dos limites superiores, dividindo pela stacks ou slices.

O processo é efetuado com dois ciclos encadenados, onde o ciclo exterior itera sobre as slices e o interior sobre stacks.

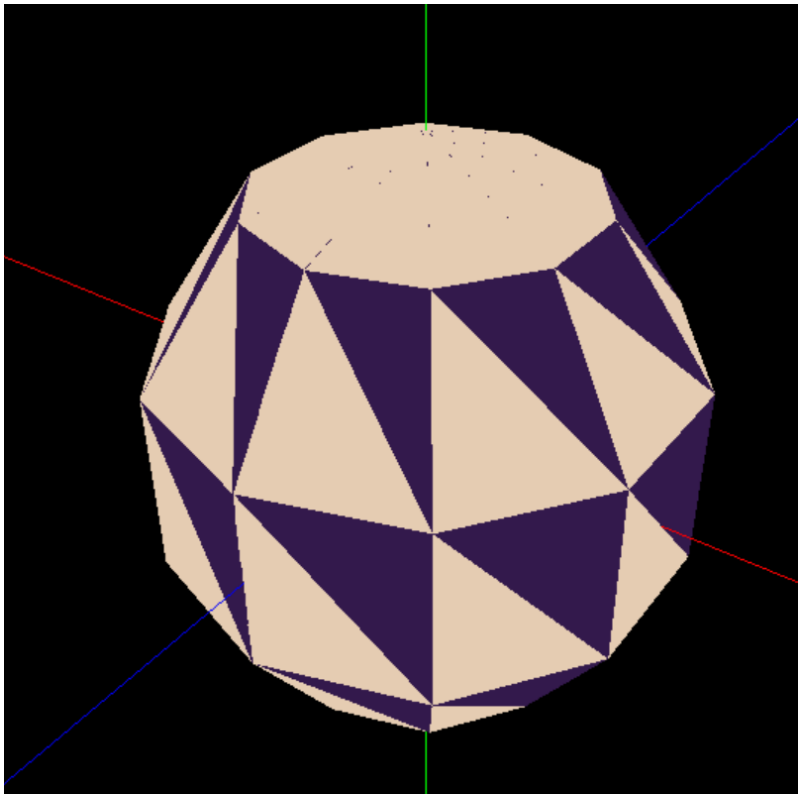

Figura 2.4: Esfera com raio 1, 10 slices e 5 stacks.

2.4 Cone

Para a criação de um cone foram utilizados 4 parâmetros: o raio da base, a altura do cone, o número de slices e o número de stacks que este irá conter.

Para a base do cone temos apenas de calcular as coordenadas em X e Y dos pontos visto que Y é igual para todos e não varia. A base será composta por vários triângulos unidos ao centro e determinamos o ângulo alpha através da divisão de 2*PI pelo número de slices e multiplicando esse valor pelo número da slice que estamos a construir. Somando esse alpha com o tamanho de cada slice chegamos ao próximo ponto do círculo para desenhar novamente o triângulo.
Para a superfície lateral é preciso fazer cada camada horizontal de cada vez e definir a altura inferior e superior da camada em questão. O raio do círculo superior calcula-se através de:
((Stacks totais – nrº próxima Stack) / Stacks totais) * raio do cone
E o raio do círculo inferior através da seguinte fórmula:
((Stacks totais – nrº da Stack atual) / Stacks totais) * raio do cone
Dentro de cada stack vão se calculando as interseções com as slices que originam quatro pontos, dois na stack i e por isso contido no círculo e outros dois na stack i+1 contido no círculo superior. A altura de cada stack calcula-se através da divisão da altura do cone sobre o número de stacks pretendido.

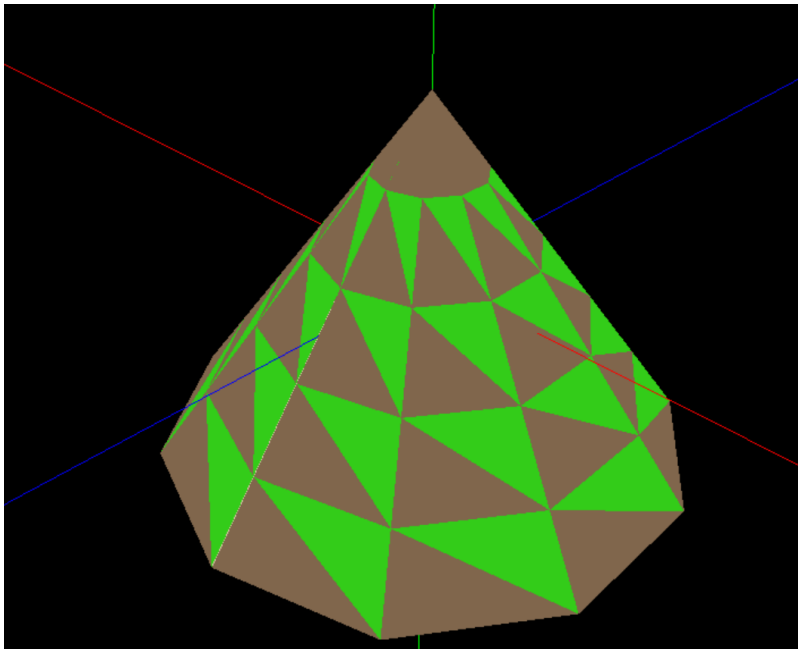

Figura 2.5: Cone com raio 1, 10 slices e 5 stacks.

3 Desenho das primitivas

3.1 *Engine*

A aplicação *engine* irá ler e processar um ficheiro **XML** de configuração que contém o nome dos ficheiros .3d que deverão ser também lidos consequentemente pela aplicação.

O *engine* tem como principal objetivo a leitura do ficheiro xml que contém informações relevantes acerca dos aspectos da câmera e dos modelos das figuras, e a partir dessas informações fazer a renderização dos mesmos.

Para o armazenamento e utilização dos dados do ficheiro 3d, foi criada a estrutura de dados "Vertex", que serve para armazenar os vértices de uma determinada figura geométrica. Para o armazenamento e utilização dos dados do ficheiro xml, foi criada uma estrutura de dados "Camera"que guarda as posições da câmara. E para a transformação de coordenadas polares para cartesianas foi utilizado como auxilio a struct "camPos".

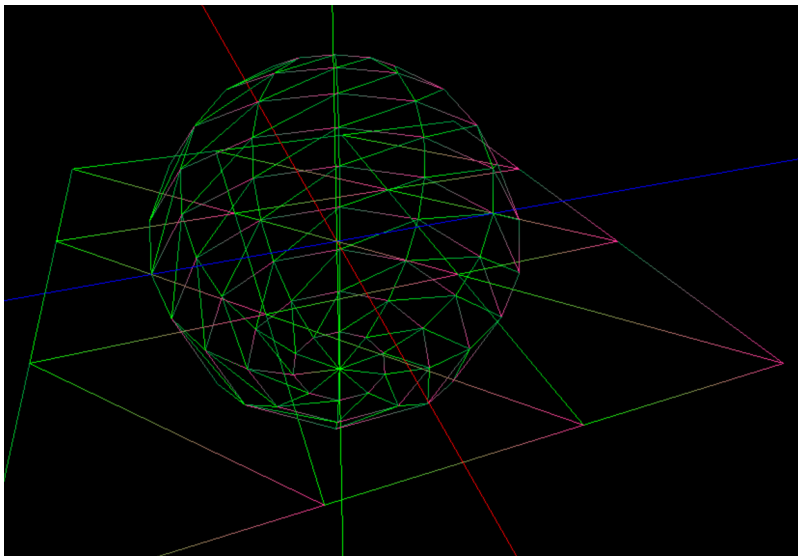

Figura 3.1: Figura renderizada do ficheiro xml teste 5

3.2 Generator

O generator possui a importância de gerar os ficheiros de extensão .3d que irá conter os pontos/vértices constituintes das diferentes primitivas gráficas e conforme os parâmetros escolhidos.

Depois de compilado e gerado o executável, para utilizar esta aplicação ao introduzir o comando "generator sphere 1 10 5 sphere.3d", por exemplo, o generator cria um ficheiro "sphere.3d"que contém as coordenadas de todos os pontos utilizados para contruir graficamente a figura geométrica pedida.

A compilação e execução de cada programa assume que será feito através de uma pasta "build"dentro da pasta que contém o código fonte dos mesmos.

Para efeitos de teste para saber se os pontos que geramos estavam corretos, utilizamos o conhecimento obtido nas aulas para renderizar cada uma das figuras.

4 Conclusão

Terminada a primeira de quatro fases deste projeto, o grupo reconheceu que foi possível consolidar vários conceitos abordados nas aulas de Computação Gráfica. Para além disso, a utilização de novas ferramentas de programação tais como OpenGl, XML e a linguagem de programação C++ fez com que o grupo pudesse abrangir os seus conhecimentos no que toca ao desenvolvimento de software.

A maior dificuldade sentida foi compreender o parsing dos ficheiros ".xml".

Um dos aspetos positivos que o grupo reconheceu foi o facto de não utilizarmos estruturas de dados demasiado complexas. No desenvolvimento do nosso trabalho apenas foi necessário criar arrays para armazenar os vértices de uma determinada primitiva em memória, as posições da câmara e ainda para escrever nos ficheiros ".3d". Para além de ser uma estrutura de simples compreensão, é suficientemente eficiente para aquele que é o objetivo final.

Consideramos que desenvolvemos um bom trabalho, apesar de uma das funcionalidades exigidas nesta fase não estar funcional (engine). No entanto é de referir que a abordagem utilizada poderá vir a sofrer alterações mediante os requisitos das fases seguintes e iremos certamente ter o engine funcional.