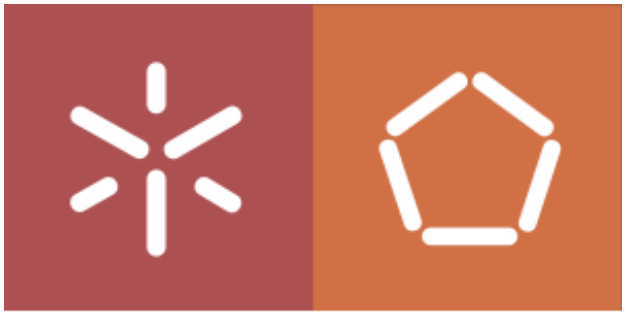


Universidade do Minho
Escola de Engenharia

# Computação Gráfica

Departamento de Informática

## Phase 1 – Graphical Primitives

Grupo 36

A93247 - André Lucas Silva Verdelho
A93184 - Diogo Camacho Barbosa
A93188 - Nelson Miranda Ribeiro

# Índice

# Introdução

Para a realização da primeira fase do trabalho, foi apresentado como objetivo o desenvolvimento de um pequeno cenário 3D através da utilização das seguintes primitivas gráficas requeridas: um plano, uma caixa, uma esfera e um cone.

Para esta fase foi necessário implementar duas aplicações: o *Generator*, que gera ficheiros 3D com a informação relativa dos modelos, contendo todos os vértices constituintes de cada primitiva; e o *Engine* – que lê os ficheiros de configuração escritos em XML, guarda os dados em estruturas e apresenta os modelos em 3D.

Nos seguintes capítulos deste relatório será apresentado como ficou estruturado esta fase do projeto e explicações mais detalhadas que sustentam a criação das duas aplicações, *Generator* e *Engine*.

# Estrutura do Projeto

De forma a obter uma melhor organização para o nosso projeto, decidimos estruturá-lo em diferentes diretorias: uma contém a informação relativa ao *Generator*, outra com a informação relativa ao *Engine*, uma diretoria *Models* que armazena os ficheiros .3d e por último, uma diretoria que contém os ficheiros XML, para posteriormente serem usados pelo Engine.

Dentro da diretoria *Generator* encontra-se o código relativo à geração de coordenadas para a representação dos diferentes modelos (plano, caixa, esfera e cone).

No *Engine* encontra-se todo o código relativo à implementação em 3D dos vários modelos. Durante a leitura dos ficheiros XML e consequente leitura dos ficheiros .3d por parte do *Engine*, os diversos pontos são armazenados em estruturas próprias, *Point* e *Triangle*.

Em ambas as aplicações, *Generator* e *Engine*, foi utilizada a biblioteca tinyxml2 para a configuração dos ficheiros XML.

## Ficheiro 3D

A estrutura dos ficheiros .3d é análoga para as diferentes primitivas. Na primeira linha do ficheiro encontra-se o número de vértices da primitiva em questão e nas restantes linhas estão presentes os vértices, em que as suas coordenadas (x,y,z) estão separadas por vírgulas.

Na seguinte figura encontra-se um exemplo de um ficheiro .3d:

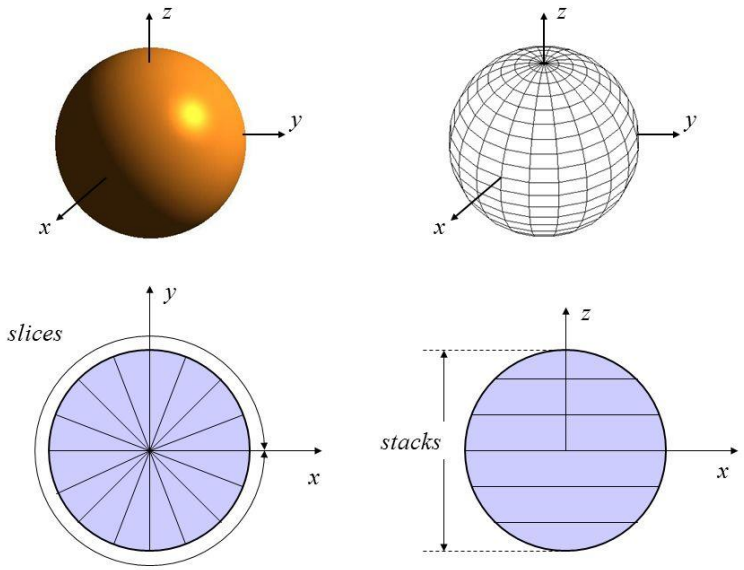
```
1   540
2   0.000000,1.000000,-0.000000
3   0.600000,0.800000,-0.000000
4   0.485410,0.800000,-0.352671
5   0.600000,-0.800000,-0.000000
```

# Generator

Esta aplicação é responsável por gerar os vértices que irão dar origem aos triângulos das diferentes primitivas, através de algoritmos específicos.

## Esfera

Para a construção da esfera foram necessários os vários inputs: o raio, os *stacks* e as *slices*. As *stacks* correspondem ao número de cortes horizontais utilizados para calcular as coordenadas dos pontos para gerar os triângulos, enquanto que os *slices* se referem aos cortes verticais, centrados no eixo Y. Para realizar a divisão destas *stacks* calculamos a altura para cada uma das mesmas. Por outro lado, para obter as coordenadas dos *slices*, recorremos a um ângulo entre as diferentes fatias.

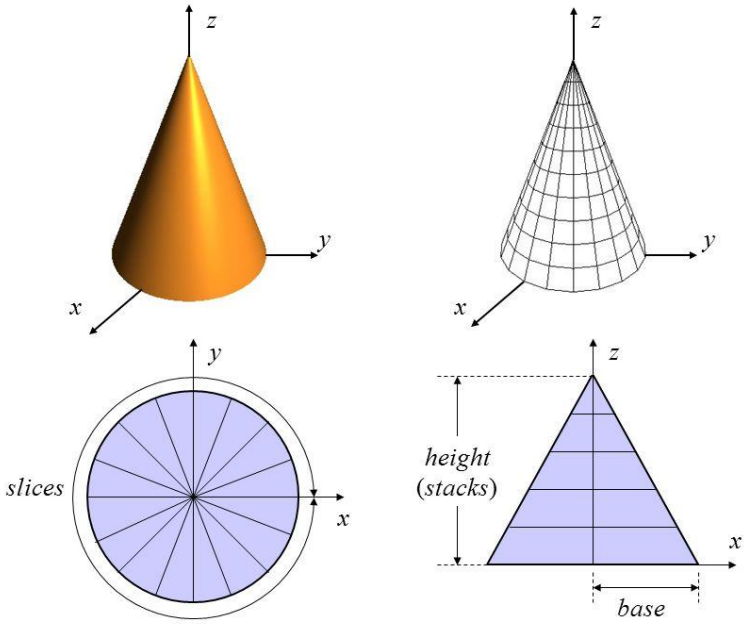



Para podermos calcular as coordenadas de forma correta tivemos de ter em consideração 3 parâmetros. Primeiro calculamos a altura da *stack* a partir de uma simples fórmula, tendo em conta o raio da esfera principal e o número de *stacks*. De seguida, obtivemos o ângulo entre cada *slice*, apenas sendo necessário o número de fatias a considerar. Por fim, era necessário saber o raio de cada circunferência resultante do corte horizontal (da *stack*).

Para calcular cada par de triângulos que formulam uma das faces, tivemos de obter 4 pontos da esfera, sendo que 2 deles estariam numa stack superior e os outros 2 na *stack* inferior, sendo que 2 pares de pontos - um da *stack* superior outro da inferior - pertencem à mesma *slice* respetivamente.

## Cone

A construção do cone demonstrou ser de semelhante raciocínio à construção de uma esfera, contudo, provida de algumas diferenças notáveis. Para além do raio, *stacks* e *slices*, era também fornecido uma altura do cone. Tendo esta altura do cone, foi facilitado o cálculo da altura de cada *stack*, sendo que a altura se obtém dividindo a altura do cone pelo número de *stacks*. Foi também necessário calcular o ângulo entre *slices* e o raio de cada *stack*.




De igual modo ao cálculo das coordenadas dos pontos de uma esfera, recorremos à estratégia de obter 4 pontos capazes de formar 2 triângulos na mesma face.

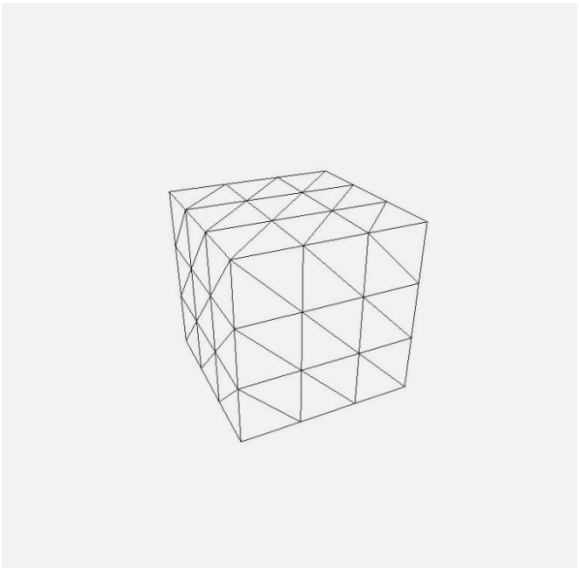
## Cubo

Para calcular os pontos de um cubo, optámos por tirar partido da liberdade de posição dos pontos, tendo estes sido escolhidos de modo a formarem planos perpendiculares aos diferentes eixos.

As faces paralelas ao plano ZY terão um x constante, sendo este obtido pela metade do comprimento do lado do cubo. Como existem duas faces, o lado oposto ao primeiro com abcissa (x) positivo terá a abcissa simétrica do mesmo. Repetiu-se este processo para os restantes planos paralelos ao plano XY e também ao plano ZX.

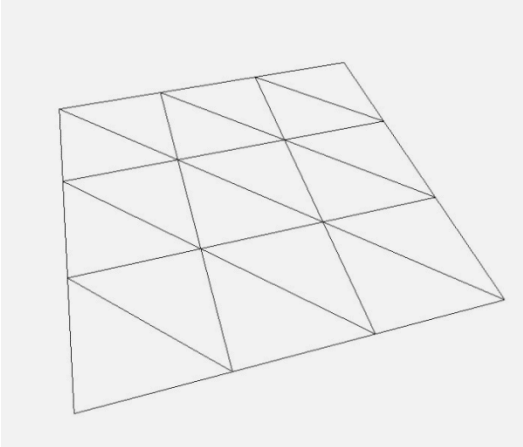
De seguida, começamos por calcular os pontos de cada 2 planos paralelos entre si, armazenando as suas coordenadas numa matriz que irá seguir um padrão, guardando em posições pré-determinadas cada ponto.

Para calcular os triângulos, tirando proveito do facto de termos padronizado o armazenamento dos pontos, conseguimos calcular cada trio capaz de formar um triângulo seguindo uma fórmula geral.



## Plano

Para obter as coordenadas de um plano singular, reutilizamos o algoritmo da construção das faces do cubo, nomeadamente, do plano paralelo ao XZ.



# Engine

Nesta parte do Projeto é realizado um parse dos documentos XML já fornecidos, podendo assim produzir os modelos neles representados.

Criamos classes Ponto e Triangulo para uma melhor organização e manuseamento de código. Depois da leitura do XML, obtemos as indicações para criar a cena e quais as figuras a serem desenhadas, incluindo as suas coordenadas, onde serão guardadas numa lista global.

Na fase de desenho, a lista que contém os diversos triângulos é iterada e desenhada pela ordem que os triângulos foram introduzidos.

O grupo implementou uma funcionalidade de rotação da câmara, semelhante à abordada nas aulas práticas. A câmara é desloca-se numa superfície esférica imaginária apontada sempre ao centro do referencial, onde se encontra a figura desenhada.

As teclas 'a' e 'd' são usadas para rodar em torno do eixo y e para deslocar a câmara verticalmente são usadas as teclas 'w' e 's'. Também foram implementadas as funcionalidades de aproximação e afastamento das figuras controlado pelas teclas '+' e '-', respetivamente.

No entanto movimentação da câmara implementada segue uma fórmula fixa de cálculo da posição, ou seja, a mesma desloca-se a uma dada posição para poder ser manipulada e não se começa a deslocar a partir da posição inicial. Sendo assim foi implementada uma funcionalidade de "reset", em que a visão da câmara volta à posição inicial.

# Conclusão

A realização deste trabalho forneceu-nos uma consolidação dos conhecimentos lecionados nas aulas teóricas e práticas, assim como nos permitiu ter uma experiência mais próxima com as ferramentas e com a nova programação em C++.

Como grupo acreditamos que poderíamos ter feito, dada a hipótese, um trabalho muito superior em qualidade, sendo que não conseguimos implementar na totalidade a construção do cubo. Temos várias teorias que potencialmente conseguem explicar o porquê de os triângulos não estarem a ser corretamente desenhados, contudo, dada a uma má organização do tempo disponível para realização do trabalho, e por uma oscilação nas prioridades, não tivemos a oportunidade de implementar a sua construção a 100%.

Por outro lado, consideramos que foi um sucesso, na medida em que conseguimos entender os erros e falhas do nosso projeto, e caso realizássemos novamente o trabalho com os conhecimentos adquiridos durante esta primeira fase, pensamos que o resultado final seria muito mais satisfatório.

Concluindo, consideramos que foi um trabalho que estava bem definido, os objetivos eram claros, e a aprendizagem resultante foi valiosa.