# Computação Gráfica - 1.ª Fase

Dinis Lourenço Gomes
a87993@alunos.uminho.pt

Rui Miguel Gonçalves Dias
a89480@alunos.uminho.pt

Universidade do Minho - Departamento de Informática

## 1 Introduction

Nesta primeira fase o objetivo do trabalho prático foi desenvolver duas aplicações: uma para gerar ficheiros com a informação dos modelos a partir de inputs e outra para ler ficheiros de configuração, em XML, e dar display aos modelos.

Uma vez que o generator e o engine são duas aplicações distintas optou-se por colocá-los em pastas diferentes.

### 1.1 Generator

A aplicação generator terá de ser capaz de processar inputs do utilizador tais como o tipo de modelo, os parâmetros necessários para a criação do modelo e o nome do ficheiro destino. Os seguintes modelos têm que ser apresentados:

- Plane (um quadrado no plano XZ, centrado na origem, composto por 2 triângulos)
- Box (dimensões X, Y e Z, e opcionalmente o numero de divisões por edge)
- Sphere (raio da esfera, slices e stacks)
- Cone (raio da base, altura, slices e stacks)

### 1.2 Engine

A aplicação engine recebe um ficheiro de configuração, em XML. Nesta fase, este ficheiro apenas apresenta o nome dos ficheiros que contêm os modelos.

O engine é apenas um worker que faz a leitura dos ficheiros *.3d* e desenha os modelos a partir das coordenadas geradas pelo generator.

# 2 Implementation

Os tópicos seguintes vão descrever o raciocínio utilizado durante cada secção do trabalho.

## 2.1 Generator

O generator recebe vários argumentos e executa diferentes funções dependendo do que quer desenhar.

O primeiro argumento é responsavel por decidir qual dos modelos e que o generator vai calcular as coordenadas para e o último será responsável pelo nome do ficheiro.

### 2.1.1 Plane

O plano (aka Plane), ao ser uma figura 2d, optou-se por representá-lo com dois quadrados um para uma perspectiva vista de cima e outro para uma vista de baixo, como conseguimos ver nas imagens seguintes:

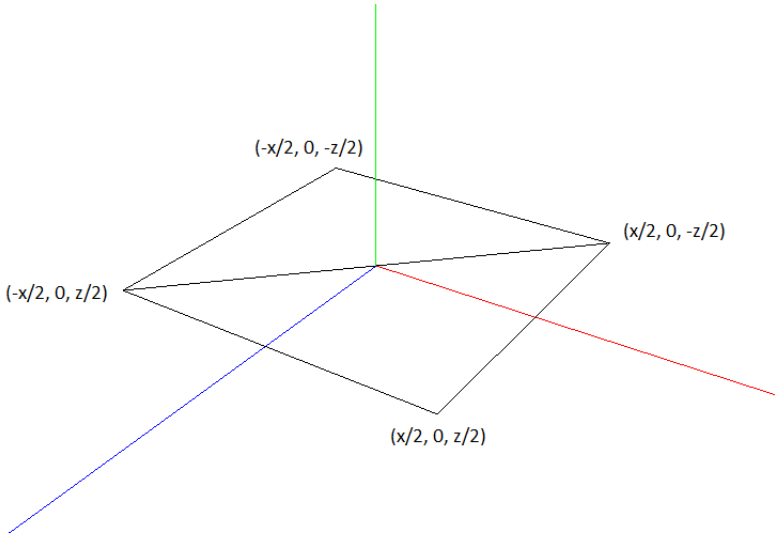



Figure 1: Plano XZ visto de cima

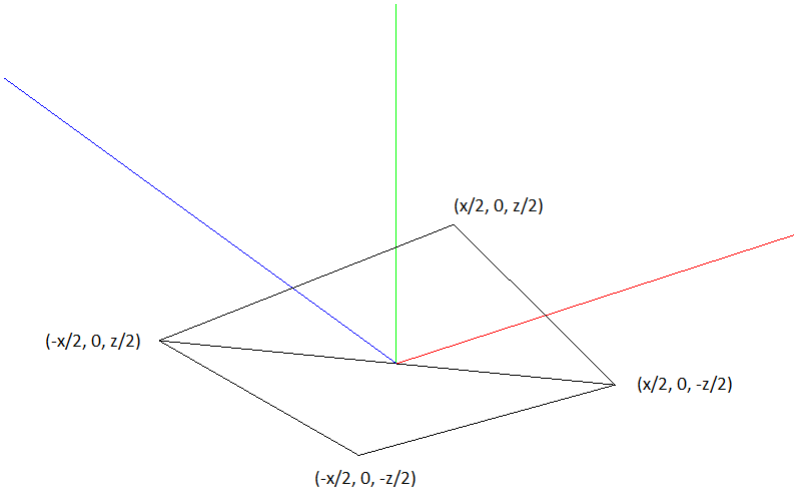



Figure 2: Plano XZ visto de baixo

### 2.1.2 Box

A Box é representada por 6 faces cujas coordenadas são:

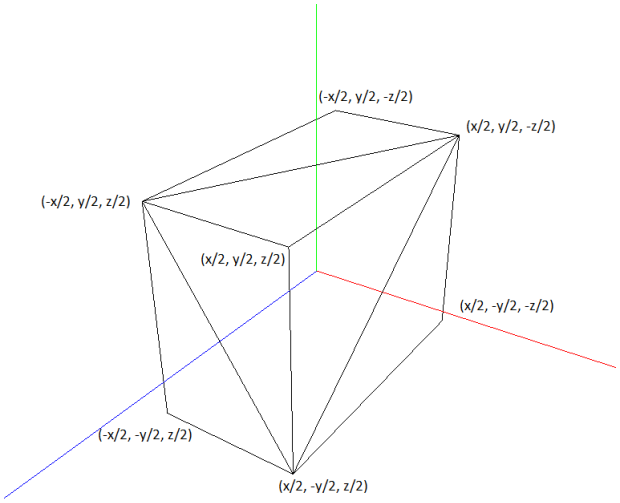
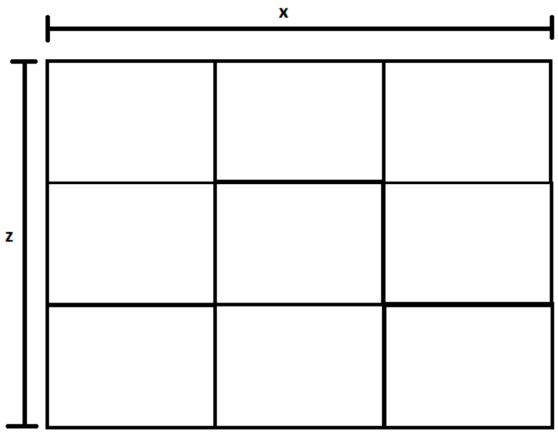



Figure 3: Plano XZ

Cada face pode ser dividida em partes iguais onde as coordenadas das faces horizontais são obtidas da maneira descrita abaixo. Para as restantes faces aplicam-se as mesmas formúlas de forma análoga.




$$x^* = -\frac{x}{2} + \frac{x}{divisionCount} divisionStep$$

$$y^* = y^*$$

$$z^* = -\frac{z}{2} + \frac{z}{divisionCount} divisionStep$$

Figure 4: Divisões de cada face

### 2.1.3 Sphere

A Sphere é dividida em stacks e slices(aka sectors). A representação começa pela parte inferior(com menor valor de y) e é feita ao longo das stacks. Os pontos obtêm-se da seguinte forma:

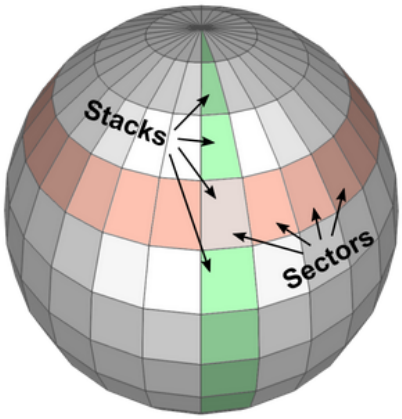



Figure 5: Sectores e stacks de uma esfera

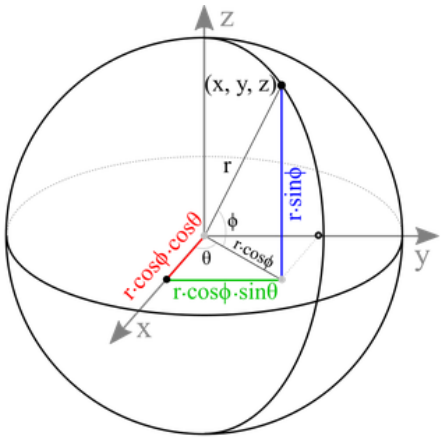



Figure 6: Ponto da esfera usando um sector e angulos de stack

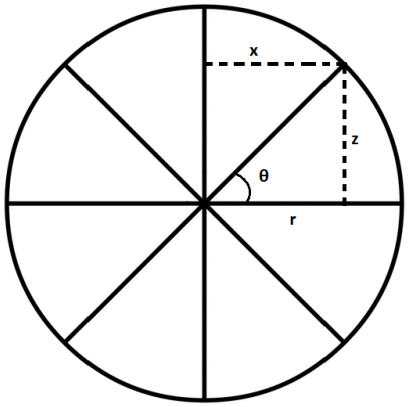
$$\theta = 2\pi \frac{sectorStep}{sectorCount}$$

$$\phi = -\frac{\pi}{2} + \pi \frac{stackStep}{stackCount}$$

$$\begin{aligned} x &= (r\cos\phi)\sin\theta \\ y &= r\sin\phi \\ z &= (r\cos\phi)\cos\theta \end{aligned}$$

### 2.1.4 Cone

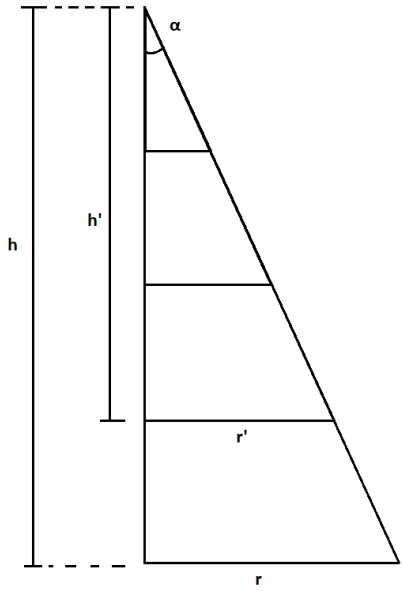
O Cone, tal como a Sphere, é dividido em stacks e slices. Os pontos da base e do topo de uma stack, excepto na tip do Cone onde o topo da stack é apenas um ponto, são calculados da seguinte fórmula:




$$\begin{aligned} \theta &= 2\pi \ / \ sectorCount \\ x^* &= r * \cos(\theta * sectorStep) \\ z^* &= r * \sin(\theta * sectorStep) \end{aligned}$$

Figure 7: Base do cone

O raio das bases criadas pela divisão em stacks é calculado por um razão trignométrica entre o ângulo e a nova altura, como mostrado na seguinte figura:




$$\begin{aligned} r &= \frac{h}{\tan \propto} \\ r' &= \frac{h'}{\tan \propto} \\ h^* &= \frac{h}{stackStep} * stackCount \end{aligned}$$

Figure 8: Vista lateral do cone

Juntando os passos anteriormente descritos obtêm-se que as coordenadas são calculadas pela formúla a seguir descrita:

$$(x,\ y,\ z) = \left( \frac{h^*}{\tan \propto} \cos(\theta * sectorStep),\ h^*, \frac{h^*}{\tan \propto} \sin(\theta * sectorStep) \right) \quad (1)$$

## 2.2 Engine

### 2.2.1 Câmera

A câmera move-se ao longo da superfície de uma esfera de raio variável, focada no centro da mesma. As coordenadas poderão então ser obtidas através dos ângulos, que podem ser definidos, utilizando as equações presentes na fig.6.

### 2.2.2 Estruturas Utilizadas

#### Model

Composto pelo nome do ficheiro e as coordenadas dos pontos necessários para a construção deste modelo.

Estas coordenadas encontram-se guardadas como um vetor de tuplos como se pode observar no seguinte código:

```
class Model{
    public:
        string filename;
        vector< tuple<float, float, float> > coordenadas;
    Model(string f, vector< tuple<float, float, float> > coord){
        filename = f;
          coordenadas = coord;
    }
};

```

### 2.2.3 Parsing do XML

O parsing dos ficheiros de XML optou-se por utilizar a biblioteca *tinyxml2*. É então guardado temporariamente o nome do file e depois associado ao Model.

### 2.2.4 Parsing dos ficheiros .3d

A leitura dos ficheiros foi utilizada a funcionalidade de streams presente no C++, pelo que a informação e transcrita para um vetor de tuplos(x,y,z) e posteriormente associado ao Model.

### 2.2.5 Desenhar os modelos

Os Models são guardados como variáveis globais, ou seja, o processo de os desenhar consiste em iterar sobre o vetor e passar à função drawFigure as coordenadas de cada Model.