Universidade do Minho – 2016/2017

Relatório de Computação Gráfica

Fase 1 – Primitivas Gráficas

Grupo 10:
Carlos Pereira, a61887
João Barreira, a73831
Rafael Braga, a61799

Índice

Introdução

No âmbito da Unidade Curricular de Computação Gráfica, foi-nos proposto, nesta primeira fase, o desenvolvimento de um gerador de vértices de algumas primitivas gráficas (plano, caixa, esfera e cone). Para além disto, também foi desenvolvido um mecanismo de leitura de ficheiros de configuração em XML que servirão para desenhar os vértices das primitivas gráficas anteriormente gerados, a partir de ficheiros escolhidos pelo utilizador.

Neste relatório incluímos uma descrição pormenorizada de todas as etapas do trabalho, dando uma maior ênfase à descrição técnica do mesmo, com o recurso a equações, diagramas, figuras e algoritmos que ajudam a documentar e a perceber melhor as escolhas por detrás da elaboração deste projeto.

Estruturação do projeto

De forma a melhorar a organização do nosso projeto, e tirando proveito da programação orientada aos objetos do *C++*, resolvemos criar uma hierarquia de classes.

Em primeiro lugar, temos a classe *Primitive* que corresponde à classe base do projeto e que é uma classe abstrata que apenas representa um conjunto de vértices (definidos pela classe auxiliar *Vertex*).

Herdando as características da classe *Primitive*, temos as classes referentes a cada uma das primitivas gráficas: *Plane*, *Box*, *Sphere* e *Cone*. Cada uma desta classes possui um método que se encarrega de calcular as coordenadas de todos os seus vértices.

Plane

- Equações

O cálculo dos pontos que constituem um plano apenas necessita de dois parâmetros: dimensão do plano no eixo dos xx (*dimX*) e dimensão do plano no eixo dos zz (*dimZ*).

Um plano possui quatro pontos distintos. Para se determinar as coordenadas x, y e z de cada um desses pontos e, para que o plano fique centrado na origem, é apenas necessário realizar os seguintes cálculos:

$$x = \frac{dimX}{2}$$
$$y = 0$$
$$z = \frac{dimZ}{2}$$

- Algoritmo

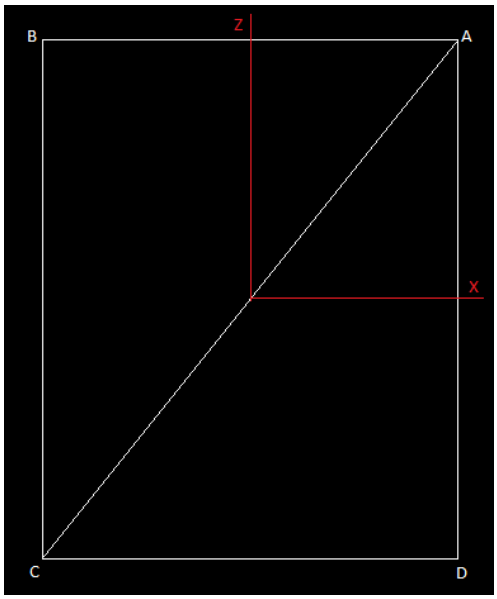

Figura 1 – Esquematização de um plano

A **Figura 1** exemplifica o modelo de um plano XZ. O plano possui quatro pontos: A, B, C e D. Como se pode verificar o plano contém apenas dois triângulos: o superior (ABC) e o inferior (ACD).

Os pontos destes triângulos são determinados por esta ordem para que, segundo o *OpenGL*, a superfície do plano fique do lado de fora pela regra da mão direita. O seguinte algoritmo traduz o conjunto de passos necessários para o cálculo desses triângulos.

$x_A = x$

$y_A = y$

$z_A = -z$

$x_B = -x$

$y_B = y$

$z_B = -z$

$x_C = -x$

$y_C = y$

$z_C = z$

$x_D = x$

$y_D = y$

$z_D = z$

Box

- Equações

O cálculo dos pontos que constituem uma caixa necessita de quatro parâmetros: largura (*dimX*), altura (*dimY*), comprimento (*dimZ*) e número de divisões (*numDiv*). A **Figura 2** e a **Figura 3** exemplificam, respetivamente, uma caixa sem divisões e uma caixa com divisões.

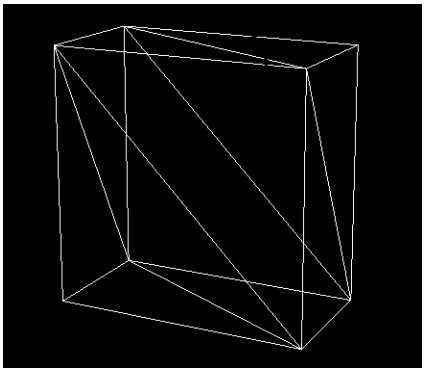

Figura 2 – Caixa sem divisões

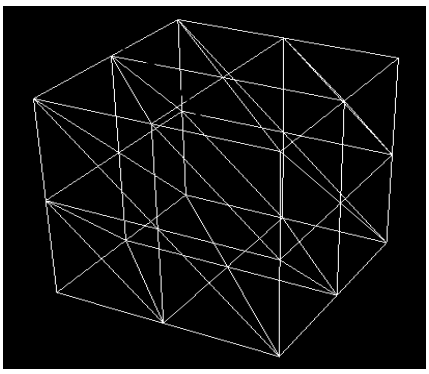

Figura 3 – Caixa com duas divisões

Como foi dito, uma caixa pode conter uma ou mais divisões. Torna-se então necessário guardar informação acerca das dimensões de cada divisão. As dimensões de uma divisão são calculadas da seguinte forma:

$$divX = \frac{dimX}{numDiv}$$
$$divY = \frac{dimY}{numDiv}$$
$$divZ = \frac{dimZ}{numDiv}$$

De modo a que a caixa fique centrada na origem, as coordenadas x, y e z do seu centro são calculadas usando as seguintes expressões:

$$centerX = \frac{dimX}{2}$$
$$centerY = \frac{dimY}{2}$$
$$centerZ = \frac{dimZ}{2}$$

- Algoritmo

O algoritmo para o cálculo de todos os vértices que constituem uma caixa é dividido em três fases: cálculo das faces XY, cálculo das faces XZ e, finalmente, cálculo das faces YZ da caixa. Todas estas três fases seguem o mesmo conjunto de passos. Inicialmente calculam-se os vértices de uma divisão (6 vértices pertencentes a 2 triângulos). Este processo é repetido *numDiv* vezes. Os vértices da face oposta são calculados somando os valores das dimensões da caixa e usando a regra da mão direita no sentido dos ponteiros do relógio. É apresentado, de seguida, o calculo detalhado dos vértices que constituem as duas faces XY de uma caixa.

- Faces XY

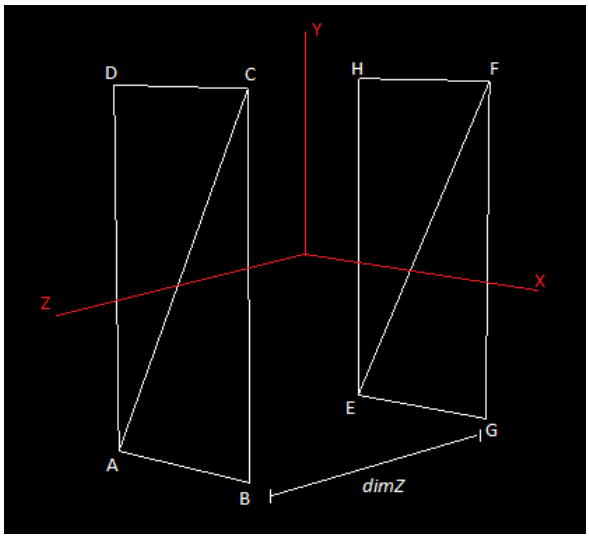

Figura 4 - Esquematização das duas faces XY de uma caixa (sem divisões)

O ponto de partida para o cálculo das faces XY corresponde às seguintes coordenadas x, y e z: (0, 0, *dimZ*).

Para *i* = x **até** $dimX$ **fazer *i* = *i* +** $divX$ **{**

Para *j* = y **até** $dimY$ **fazer *j* = *j* +** $divY$ **{**

Cálculo da face da frente

$$x_A = i - centerX$$
$$y_A = j - centerY$$
$$z_A = z - centerZ$$

$$x_B = i + divX - centerX$$
$$y_B = j - centerY$$
$$z_B = z - centerZ$$

$$x_C = i + divX - centerX$$
$$y_C = j + divY - centerY$$
$$z_C = z - centerZ$$

$$x_D = i + divX - centerX$$
$$y_D = j + divY - centerY$$
$$z_D = z - centerZ$$

Cálculo da face de trás

$$x_E = i - centerX$$
$$y_E = j - centerY$$
$$z_E = z - centerZ - dimZ$$

$$x_F = i + divX - centerX$$
$$y_F = j + divY - centerY$$
$$z_F = z - centerZ - dimZ$$

$$x_G = i + divX - centerX$$
$$y_G = j - centerY$$
$$z_G = z - centerZ - dimZ$$

$$x_H = i - centerX$$
$$y_H = j + divY - centerY$$
$$z_H = z - centerZ - dimZ$$

}

}

- Faces XZ

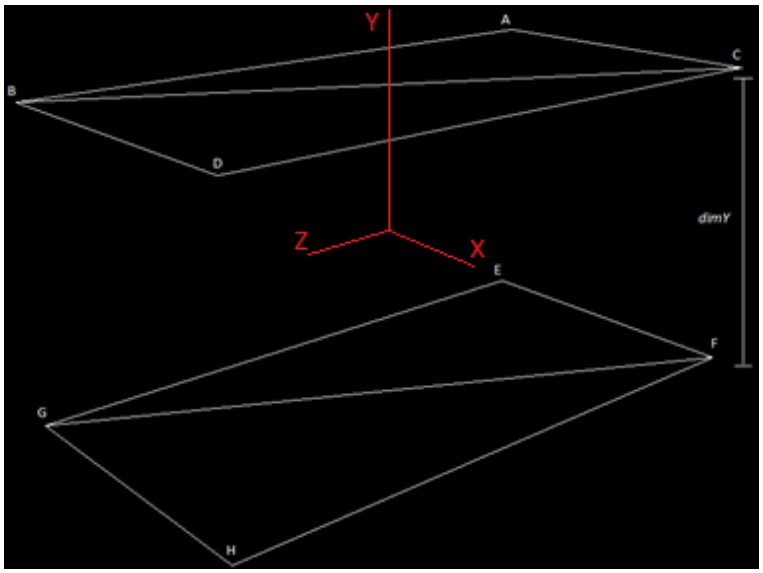

Figura 5 **-** Esquematização das duas faces XZ de uma caixa (sem divisões)

O ponto de partida para o cálculo das faces XY corresponde às seguintes coordenadas x, y e z: (0, *dimY*, 0).

Para *i* = x **até** $dimX$ **fazer *i = i +*** $divX$ ***{***

Para *j* = z **até** $dimZ$ **fazer *j = j +*** $divZ$ ***{***

Cálculo da face de cima

$$x_A = i - centerX$$
$$y_A = y - centerY$$
$$z_A = j - centerZ$$

$$x_B = i - centerX$$
$$y_B = y - centerY$$
$$z_B = j + divZ - centerZ$$

$$x_C = i + divX - centerX$$
$$y_C = y - centerY$$
$$z_C = j - centerZ$$

$$x_D = i + divX - centerX$$
$$y_D = y - centerY$$
$$z_D = j + divZ - centerZ$$

Cálculo da face de baixo

$$x_E = i - centerX$$
$$y_E = y - centerY - dimY$$
$$z_E = j - centerZ$$

$$x_F = i + divX - centerX$$
$$y_F = y - centerY - dimY$$
$$z_F = j - centerZ$$

$$x_G = i - centerX$$
$$y_G = y - centerY - dimY$$
$$z_G = j + divZ - centerZ$$

$$x_H = i + divX - centerX$$
$$y_H = y - centerY - dimY$$
$$z_H = j + divX - centerZ$$

}

}

- Faces YZ

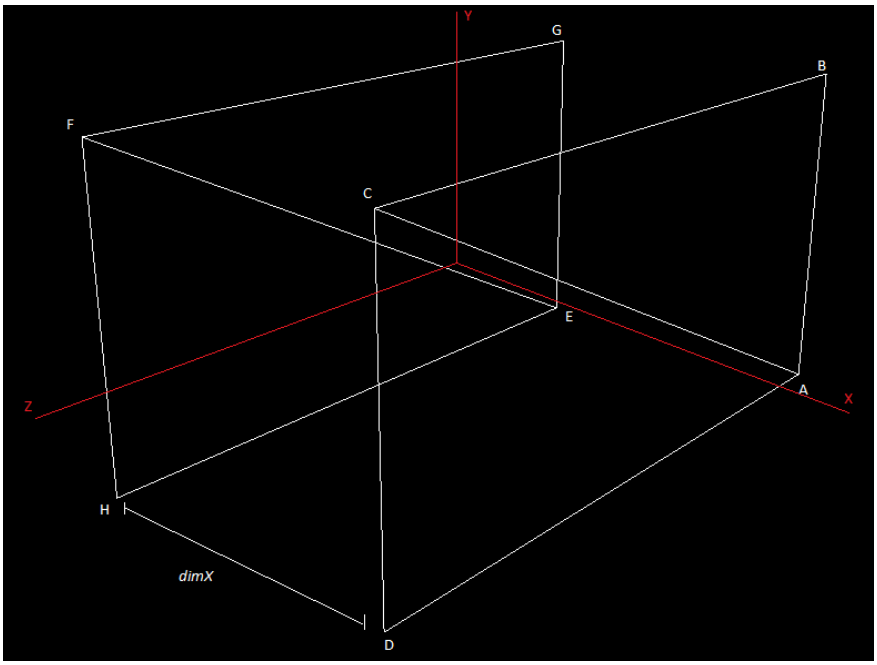

Figura 6 - Esquematização das duas faces YZ de uma caixa (sem divisões)

O ponto de partida para o cálculo das faces YZ corresponde às seguintes coordenadas x, y e z: (*dimX*, 0, 0).

Para *i =* y **até** $dimY$ **fazer *i = i +*** $divY$ ***{***

Para *j =* z **até** $dimZ$ **fazer *j = j +*** $divZ$ ***{***

Cálculo da face da frente

$$x_A = x - centerX$$
$$y_A = i - centerY$$
$$z_A = j - centerZ$$

$$x_B = x - centerX$$
$$y_B = i + divY - centerY$$
$$z_B = j - centerZ$$

$$x_C = x - centerX$$
$$y_C = i + divY - centerY$$
$$z_C = j + divZ - centerZ$$

$$x_D = x - centerX$$
$$y_D = i - centerY$$
$$z_D = j + divZ - centerZ$$

Cálculo da face de trás

$$x_E = x - centerX - dimX$$
$$y_E = i - centerY$$
$$z_E = j - centerZ$$

$$x_F = x - centerX - dimX$$
$$y_F = i + divY - centerY$$
$$z_F = j + divZ - centerZ$$

$$x_G = x - centerX - dimX$$
$$y_G = i + divY - centerY$$
$$z_G = j - centerZ$$

$$x_H = x - centerX - dimX$$
$$y_H = i - centerY$$
$$z_H = j + divZ - centerZ$$

}

}

Sphere

- Equações

O cálculo dos pontos que constituem uma esfera necessita de três parâmetros: raio, *slices* e *stacks*. Estas duas últimas são, respetivamente, camadas na vertical e horizontal ao longo da superfície da esfera. Quanto maior forem estes dois campos maior será o número de pontos a determinar e maior será a precisão no desenho da esfera.

A interseção de uma camada vertical (*slice*) e uma camada horizontal (*stack*) resulta em quatros pontos diferentes. Cada um destes pontos tem a mesma distância ao centro da esfera: raio. Podemos então ter um vetor para cada ponto. Este vetor tem origem no centro e termina no ponto em questão, logo a sua norma será igual ao raio. Tal como está explicado na **Figura 7**, o vetor tem dois ângulos: um relativo ao eixo Z (α) e outro em relação ao eixo Y (β).

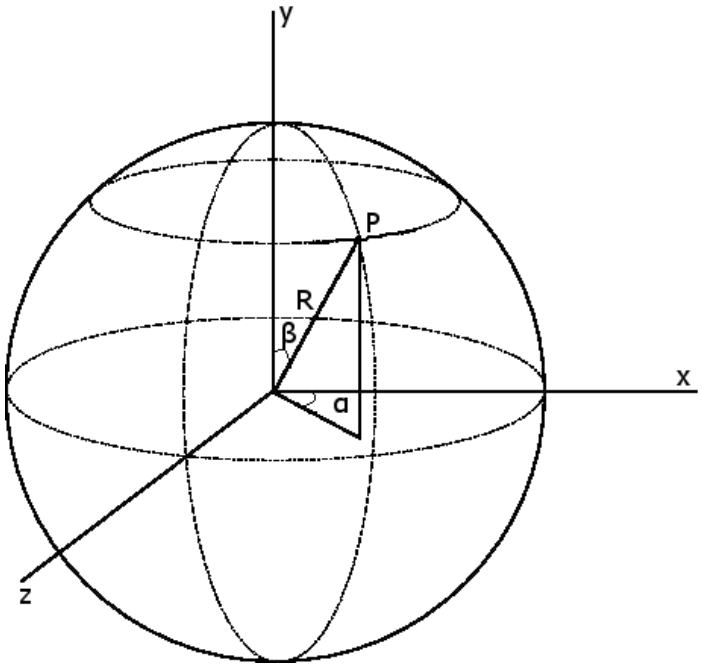

Figura 7 – Ângulos de um ponto na superfície de uma esfera

Sabendo isto, é possível determinar as equações para calcular as coordenadas x, y e z do ponto P a partir dos ângulos:

$$x = raio \times sin(\beta) \times \cos(\alpha)$$
$$y = raio \times cos(\beta)$$
$$z = raio \times sin(\beta) \times \sin(\alpha)$$

Para se determinar as coordenadas em x e em z, primeiro é necessário projetar o vetor que contém o ponto P para o plano XZ. Só depois é que se projeta o ponto para os eixos X e Z. Daí fazer-se a multiplicação por $sin(\beta)$ no cálculo destas coordenadas. Os ângulos α e β dependem do número de *slices* e *stacks*, visto serem os ângulos em relação aos eixos X e Y, respetivamente:

$$\alpha = \frac{2 \times \pi}{slices}$$

$$\beta = \frac{\pi}{stacks}$$

O nosso programa, para determinar os pontos duma esfera, calcula as coordenadas dos quatros pontos resultantes da interseção de *slices* com *stacks* – **Figura 8**. A estes pontos chamamos A, B, C e D para facilitar a ordem da escrita do código. O ponto A é a nossa referência, a partir donde calculamos os pontos B, C e D. Para cada interseção temos quatro pontos, mas estamos a utilizar triângulos, sendo necessário calcular as coordenadas necessárias para dois triângulos: o inferior (ACD) e o superior (ABC).

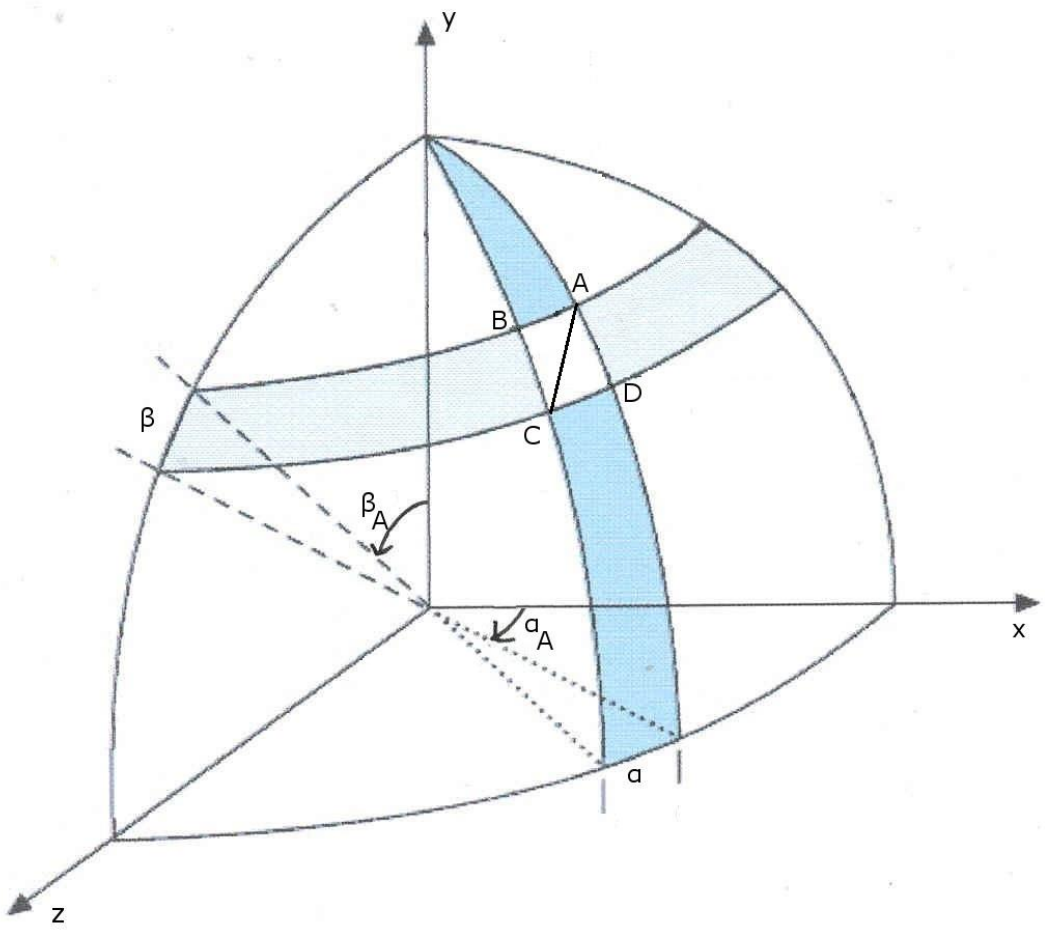

Figura 8 – Esquematização de uma interseção de *slices* e *stack*

Os pontos destes triângulos são determinados por esta ordem para que, segundo o *OpenGL*, a superfície da esfera fique do lado de fora pela regra da mão direita.

Sabendo tudo o que já foi dito, é possível determinar os ângulos α e β dos pontos B, C e D a partir dos de A (**Tabela 1**):

$\alpha_B = \alpha_A + \alpha$	$\beta_B = \beta_A$
$\alpha_C = \alpha_A + \alpha$	$\beta_C = \beta_A + \beta$
$\alpha_D = \alpha_A$	$\beta_D = \beta_A + \beta$

Tabela 1 – Equações para calcular os ângulos α_P *e* β_P de um ponto P a partir dos ângulos de A

α_P e β_P são os ângulos aos eixos X e Y do vetor que contém o ponto P.

- Algoritmo

Sabendo as equações a aplicar para determinar as coordenadas x, y e z, como também a dependência entre os pontos de uma interseção de *slice* e *stack* a partir de um ponto, utilizámos o seguinte algoritmo para calcular as coordenadas:

Para cada *stack i* {

$\beta_A = \beta * i$

Para cada *slice j* {

$\alpha_A = \alpha * j$

$$x_A = raio \times sin(\beta_A) \times \cos(\alpha_A)$$
$$y_A = raio \times cos(\beta_A)$$
$$z_A = raio \times sin(\beta_A) \times \sin(\alpha_A)$$

$$x_B = raio \times sin(\beta_A) \times \cos(\alpha_A + \alpha)$$
$$y_B = raio \times cos(\beta_A)$$
$$z_B = raio \times sin(\beta_A) \times \sin(\alpha_A + \alpha)$$

$$x_C = raio \times sin(\beta_A + \beta) \times \cos(\alpha_A + \alpha)$$
$$y_C = raio \times cos(\beta_A + \beta)$$
$$z_C = raio \times sin(\beta_A + \beta) \times \sin(\alpha_A + \alpha)$$

$$x_D = raio \times sin(\beta_A + \beta) \times \cos(\alpha_A)$$
$$y_D = raio \times cos(\beta_A + \beta)$$
$$z_D = raio \times sin(\beta_A + \beta) \times \sin(\alpha_A)$$

}

}

Exemplo de uma esfera calculada pelo algoritmo anterior:

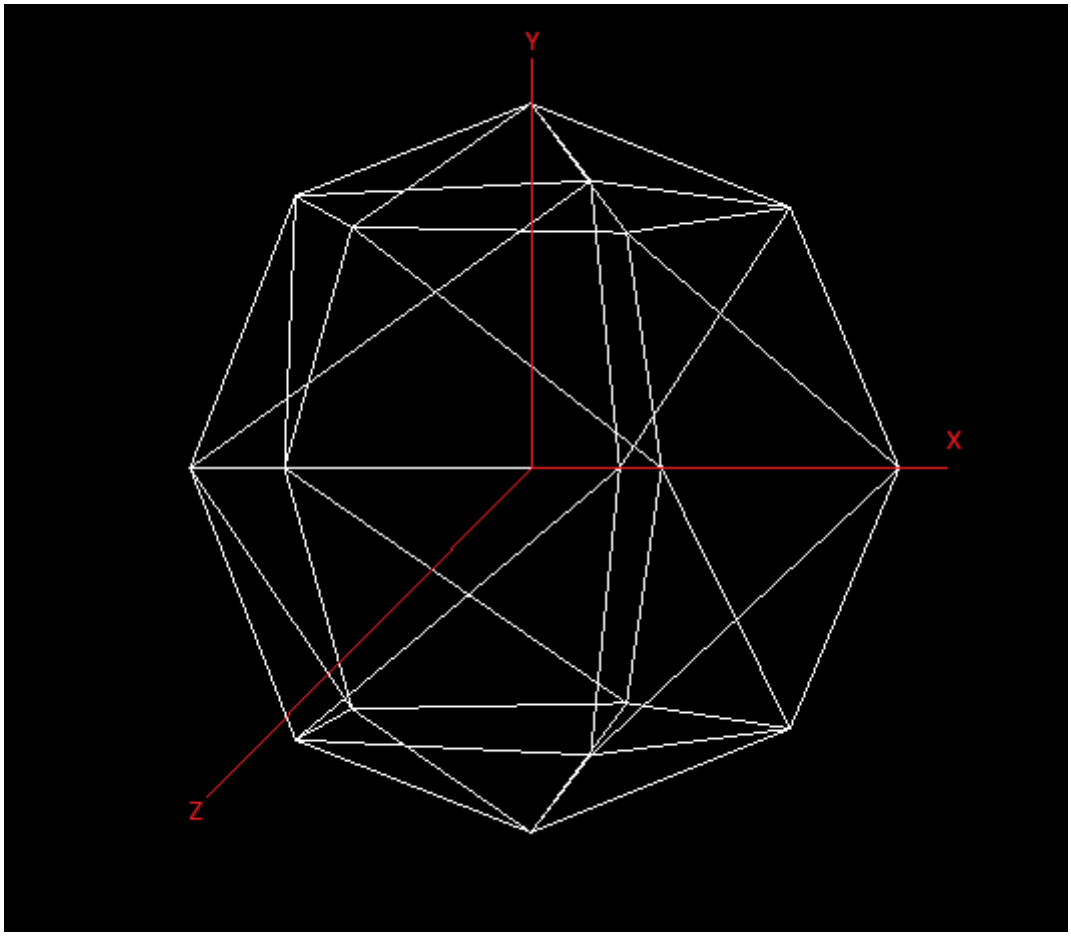

Figura 9 – Esfera com 4 stacks e 5 slices

Cone

- Equações

Para o cálculo do cone são necessários quatro parâmetros: raio, altura, *slices* e s*tacks*. Tal como no caso da esfera, as *slices* e as *stacks* são, respetivamente, camadas na vertical e na horizontal ao longo da superfície do cone. Quanto maior forem em número, maior será o número de pontos e maior será a precisão no desenho da primitiva.

Ao contrário da esfera, não existe uma equação para calcular os pontos do cone em cada eixo. Isto porque não se trata de uma primitiva regular. Assim, o cálculo dos pontos dividido em duas partes: pontos que constituem a base e pontos que constituem a superfície lateral.

A base de um cone é um círculo que será paralelo ao plano XZ. Ou seja, apenas temos que calcular as coordenas em X e em Z dos pontos, visto que a coordenada Y é igual para todos e não varia. Como só estamos a desenhar com triângulos, a base será constituída por vários triângulos unidos ao centro (**Figura 10**), onde C é o centro. Tal como no caso da esfera, temos um ângulo α entre o vetor que vai da origem a um ponto P da base e o eixo Y. Este ângulo é igual a:

$$\alpha = \frac{2 \times \pi}{slices}$$

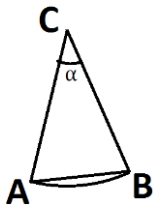

Figura 10 *– Esquema de um triângulo pertencente à base do cone*

Pela **Figura 10**, verifica-se que as coordenadas X e Z de um ponto P, em função de α, são:

$$x = raio \times \sin(\alpha)$$
$$z = raio \times \cos(\alpha)$$

Devido à semântica do *OpenGL*, a declaração das coordenadas da base segue a regra da mão direita, de forma que a base fique com a superfície virada para fora. Ou seja, para o triângulo da **Figura 10**, as coordenadas seriam declaradas pela ordem: A, C e B.

O cálculo das coordenadas dos pontos da superfície lateral do cone segue a lógica da esfera: a interseção entre uma *slice* e uma *stack* resulta em quatro pontos. Ou seja, temos o triângulo ABC e ACD. Os pontos são definidos por esta ordem para que, segunda a regra da mão direita, a superfície fique para o lado de fora (**Figura 11**):

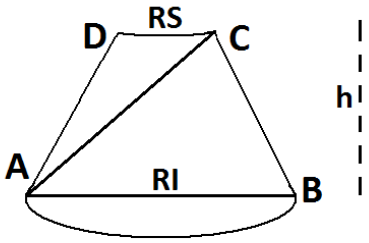

Figura 11 – Interseção de uma *slice* e uma *stack* na superfície lateral do cone

Os pontos são calculados, de *stack* em *stack*, da base até ao vértice do cone. Para uma *stack* i, D e C coincidem com os pontos A e B da *stack i* + 1. Pode-se afirmar que uma *stack* é delimitada por dois círculos horizontais de raios diferentes, em que:

- A e B pertencem ao círculo inferior de raio *RI*

- D e C pertencem ao círculo superior de raio *RS*

A altura dos pontos que pertencem ao círculo superior é igual à dos pontos pertencentes ao círculo inferior mais a altura *h* de uma *stack*:

$$h = \frac{altura\ do\ cone}{stacks}$$

$$h_D = h_C = h_A + h = h_B + h$$

Sabendo a altura dos pontos em relação à base, calculam-se os raios *RI* e *RS* através da semelhança de triângulos:

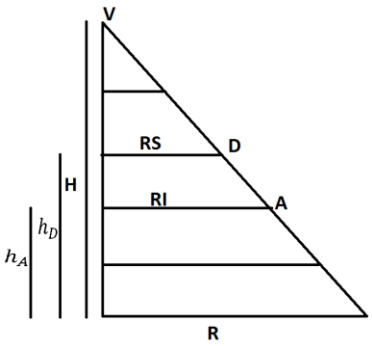

Figura 12 – Vista lateral do cone, em que H é a altura, R o raio e V o vértice no topo

$$RI = R - \frac{R \times h_A}{H}$$

$$RS = R - \frac{R \times h_D}{H}$$

Posto isto, consegue-se facilmente calcular as coordenadas dos pontos A, B, C e D. Utilizamos, novamente, o ponto A como referência, determinando-se as restantes coordenadas a partir dele:

$$\alpha_A = \alpha \times número\ da\ stack$$

$x_A = RI \times \sin(\alpha)$	$y_A = h \times número\ da\ stack$	$z_A = RI \times \cos(\alpha)$
$x_B = RI \times \sin(\alpha_A + \alpha)$	$y_B = y_A$	$z_B = RI \times \cos(\alpha_A + \alpha)$
$x_C = RS \times \sin(\alpha_A + \alpha)$	$y_C = y_A + h$	$z_C = RS \times \cos(\alpha_A + \alpha)$
$x_D = RS \times \sin(\alpha_A)$	$y_D = y_A + h$	$z_D = RS \times \cos(\alpha_A)$

Tabela 2 – Equações para determinar as coordenadas dos pontos na superfície lateral do cone

- Algoritmos

Sabendo as equações a aplicar para determinar as coordenadas x, y e z, como também a dependência entre os pontos de uma interseção de *slice* e *stack* a partir de um ponto, utilizámos os seguintes algoritmos para calcular as coordenadas da base e da superfície lateral, respetivamente:

Base do cone

$x_C = 0$

$z_C = 0$

Para cada *slice i* {

$x_A = raio \times \sin(\alpha_A)$

$z_A = raio \times \cos(\alpha_A)$

$x_B = raio \times \sin(\alpha_A + \alpha)$

$z_B = raio \times \cos(\alpha_A + \alpha)$

}

Superfície lateral

Para cada *stack i* {

$altura\ inferior\ (HI) = i \times altura\ de\ uma\ stack\ (h)$

$altura\ superior\ (HS) = (i + 1) \times h$

$raio\ inferior\ (RI) = raio\ (R) - \frac{R \times HI}{altura\ do\ cone\ (H)}$

$raio\ superior\ (RS) = R - \frac{R \times HS}{H}$

Para cada *slice j* {

$\alpha_A = \alpha \times j$

$x_A = RI \times \sin(\alpha_A)$

$$y_A = HI$$
$$z_A = RI \times \cos(\alpha_A)$$

$$x_B = RI \times \sin(\alpha_A + \alpha)$$
$$y_B = HI$$
$$z_B = RI \times \cos(\alpha_A + \alpha)$$

$$x_C = RS \times \sin(\alpha_A + \alpha)$$
$$y_C = HS$$
$$z_C = RS \times \cos(\alpha_A + \alpha)$$

$$x_D = RS \times \sin(\alpha_A)$$
$$y_D = HS$$
$$z_D = RS \times \cos(\alpha_A)$$

}

}

Exemplo de um cone calculado pelo algoritmo anterior:

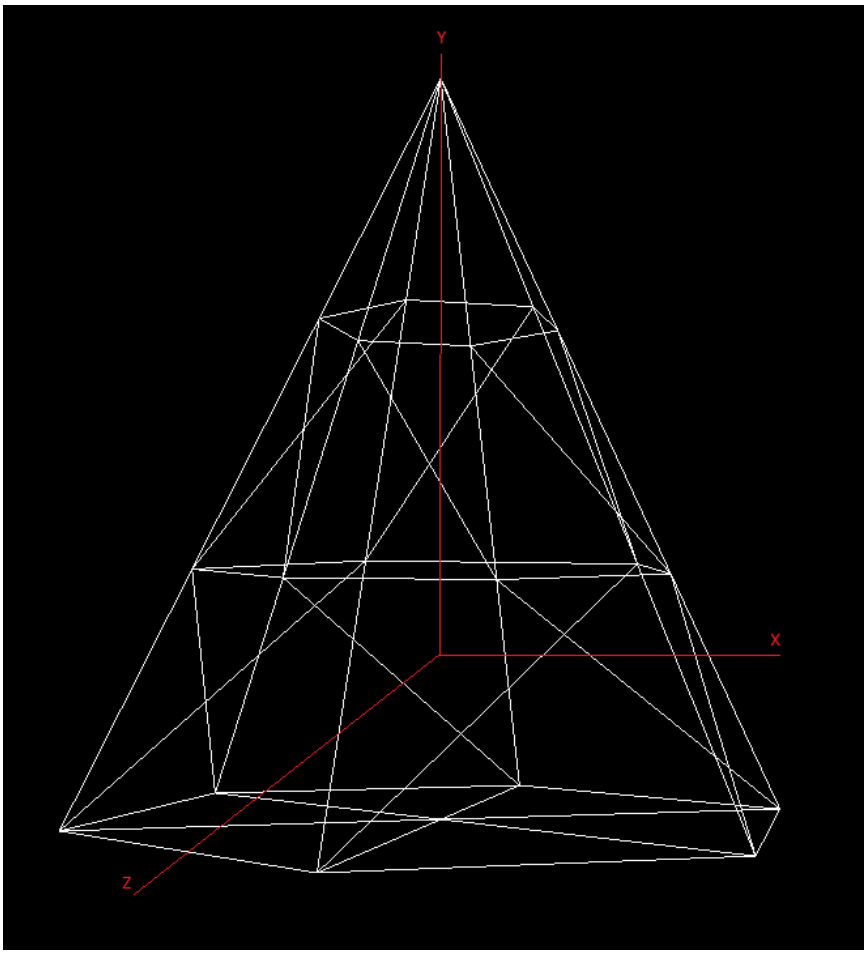

Figura 2 – Cone com 3 *stacks* e 6 *slices*