Universidade do Minho

Mestrado Integrado em Engenharia Informática

Computação Gráfica

Primitivas Gráficas

Grupo:
Etienne Costa A76089
Joana Cruz A76270
Rafael Alves A72629
Maurício Salgado A71407

Docente:
António Ramires

9 de Março de 2019

Conteúdo

1 Introdução

O relátorio apresentado diz respeito à primeira fase do trabalho proposto no âmbito da unidade curricular de Computação Gráfica. O trabalho consiste no desenvolvimento de um gerador de vértices de algumas primitivas gráficas sendo estas um plano, uma caixa, uma esfera e um cone). Para além disto, também foi desenvolvida uma aplicação de leitura de ficheiros de configuração em XML(Engine) que servirá para desenhar os vértices anteriormente gerados. Para o desenvolvimento deste projeto foi necessário utilizar certos recursos como a linguagem C++ e o OpenGL.

2 Gerador

Esta aplicação tem como objetivo gerar todos os vértices necessários para a elaboração dos triângulos que constituem a figura pretendida e guardar num ficheiro. A função main do gerador vê qual a primitiva gráfica que se pretende gerar, como já referimos, pode ser um plano, uma caixa, uma esfera ou um cone. De seguida chama a função geradora correspondente, tendo em conta o número de argumentos pedidos, e escreve como resultado os vértices gerados.

```
Generator.exe <modelo> [parâmetros] <outputFile>
```

- **Plano** - apenas recebe um argumento, por ser pedido um plano quadrado e corresponde ao lado do plano.
- **Caixa** - recebe como argumentos o comprimento, a altura, a largura e o número de divisões. Caso não se pretenda dividir este último argumento tomará o valor 1.
- **Esfera** - recebe como argumentos o raio, o número de slices(divisões verticais) e o número de stacks(divisões horizontais).
- **Cone** - recebe os mesmos argumentos da esfera, incluindo a altura do cone.

3 Plano

É pretendido um plano XZ quadrado, centrado na origem e obtido com 2 triângulos. Para calcular os pontos que constituem o plano precisamos do tamanho que nos dará informação sobre a dimensão do plano no eixo dos xx, e dimensão do plano no eixo dos zz. O plano contém 4 pontos e sendo centrado na origem temos que efetuar os seguintes cálculos:

$$
\begin{gathered}
x = tamanho/2 \\
y = 0 \\
z = tamanho/2
\end{gathered}
$$

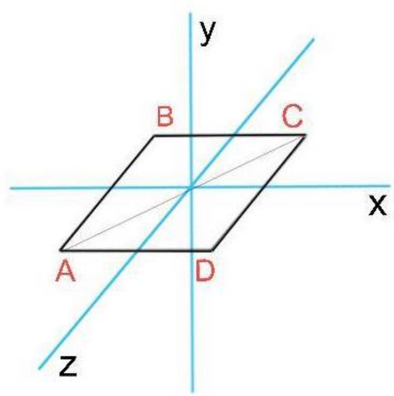

Figura 1: Figura ilustrativa de um plano XZ

Efetuando os cálculos, obtemos os seguintes pontos:

$$
\begin{gathered}
A = (-x, y, z) \\
B = (-x, y, -z) \\
C = (x, y, -z) \\
D = (x, y, z)
\end{gathered}
$$

Gerando o plano a partir do ponto A, e segundo o openGL, para a superfície do plano ficar visível utilizamos a regra da mão direita no sentido inverso aos ponteiros do relógio, obtemos que os vértices dos triângulos ACB e ADC, terão como coordenadas:

$$ACB \rightarrow (-x, y, z)(x, y, -z)(-x, y, -z)$$
$$ADC \rightarrow (-x, y, z)(x, y, z)(x, y, -z)$$

Exemplo: *plane 5*

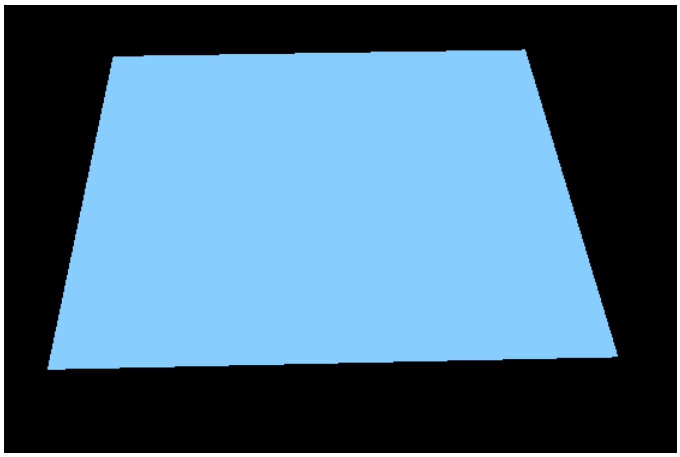

Figura 2: Exemplo de um plano

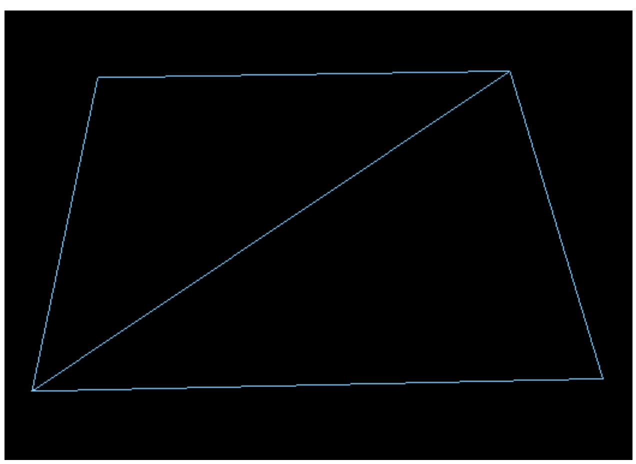

Figura 3: Exemplo de um plano com a representação dos triângulos

Caso fosse pedido qualquer plano XZ teríamos que receber dois parâmetros: as dimensões do plano no eixo do xx e no eixo zz.

4 Caixa

O cálculo dos pontos de uma caixa precisa dos seguintes parâmetros: *comprimento*(dimensão no eixo dos xx), *altura*(dimensão no eixo dos yy), *largura*(dimensão no eixo dos zz) e o número de *divisões*. Uma caixa pode ou não conter divisões pelo que precisamos de guardar informação sobre o número de divisões, sendo estas calculadas pelas seguintes equações:

$$divX = \frac{dimX}{div}$$

$$divY = \frac{dimY}{div}$$

$$divZ = \frac{dimZ}{div}$$

E de modo a que a caixa fique centrada na origem precisamos das coordenadas x, y, e z do seu centro:

$$dimX = \frac{comprimento}{2}$$

$$dimY = \frac{altura}{2}$$

$$dimZ = \frac{largura}{2}$$

Necessitamos de calcular todos os vértices que constituem a caixa, que é constítuida por 6 faces. Primeiramente, calculamos as faces XZ, de seguida as faces XY e por último as faces YZ. Todas estas fases seguem o mesmo processo: calcula-se os seis vértices pertencentes aos dois triângulos de uma divisão, usamos a regra da mão direita no sentido contrário aos ponteiros do relógio nas faces visíveis para nós, enquanto as faces opostas são calculadas no sentido dos ponteiros do relógio. O posicionamento inicial é sempre a origem e este processo é repetido quantas vezes o número de divisões.

4.1 Cálculo das faces XZ

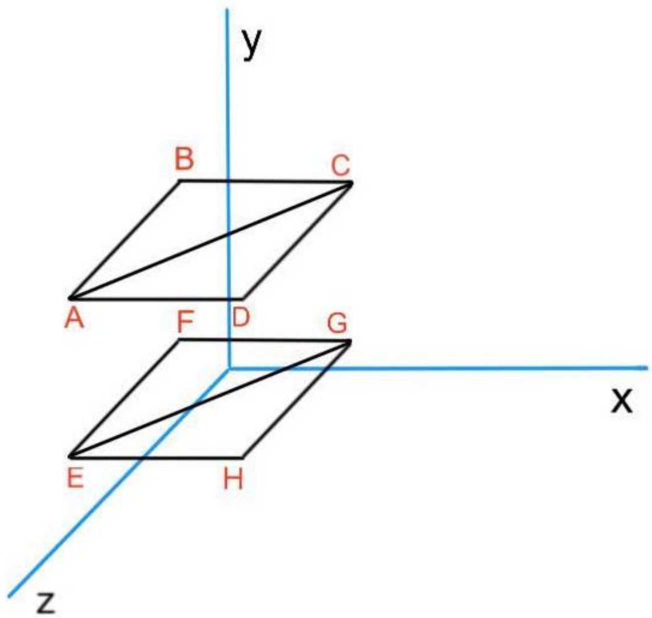

Figura 4: Faces XZ de uma caixa sem divisões

Para $i = 0$ **até div fazer** $i++\{$

$x = i \times divX$

Para $j = 0$ **até div fazer** $j++\{$

$z = j \times divZ$

Cálculo da face de cima - Triângulos ACB e ADC

$xa = x - dimX$

$ya = dimY$

$za = z - dimZ + divZ$

$xb = x - dimX$

$yb = dimY$

$zb = z - dimZ$

$xc = x - dimX + divX$

$yc = dimY$

$zc = z - dimZ$

$xd = x - dimX + divX$

$yd = dimY$

$zd = z - dimZ + divZ$

Cálculo da face de baixo - Triângulos EFG e EGH

$xe = x - dimX$

$ye = -dimY$

$ze = z - dimZ + divZ$

$xf = x - dimX$

$yf = -dimY$

$zf = z - dimZ$

$xf = x - dimX + divX$

$yg = -dimY$

$zg = z - dimZ$

$xh = x - dimX + divX$

$yh = -dimY$

$zh = z - dimZ + divZ$

}
}

4.2 Cálculo das faces XY

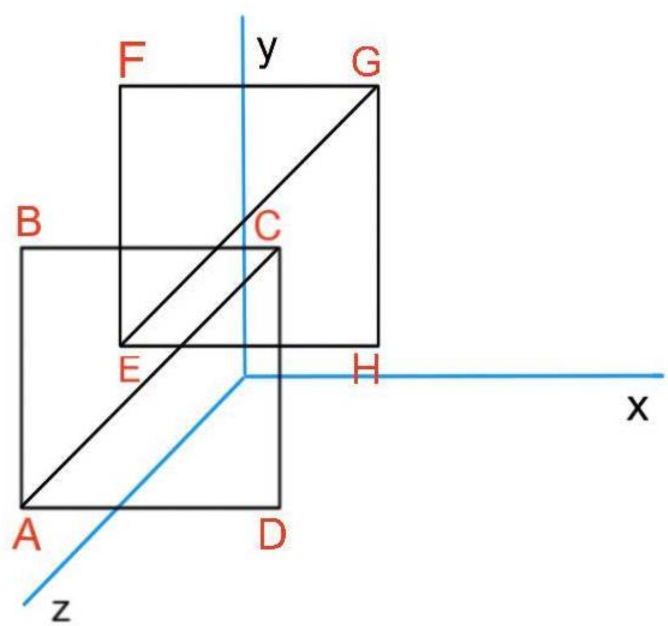

Figura 5: Faces XY de uma caixa sem divisões

Para $i = 0$ **até div fazer** $i + +\{$

$x = i \times divX$

Para $j = 0$ **até div fazer** $j + +\{$

$y = j \times divY$

Cálculo da face da frente - Triângulos ACB e ADC

$xa = x - dimX$

$ya = y - dimY$

$za = dimZ$

$xb = x - dimX$

$yb = y - dimY + divY$

$zb = dimZ$

$xc = x - dimX + divX$

$yc = y - dimY + divY$

$zc = dimZ$

$xd = x - dimX + divX$

$yd = y - dimY$

$zd = dimZ$

Cálculo da face de trás - Triângulos EFG e EGH

$xe = x - dimX$

$ye = y - dimY$

$ze = -dimZ$

$xf = x - dimX$

$yf = y - dimY + divY$

$zf = -dimZ$

$xg = x - dimX + divX$

$yg = y - dimY + divY$

$zg = -dimZ$

$xh = x - dimX + divX$

$yh = y - dimY$

$zh = -dimZ$

}
}

4.3 Cálculo das faces YZ

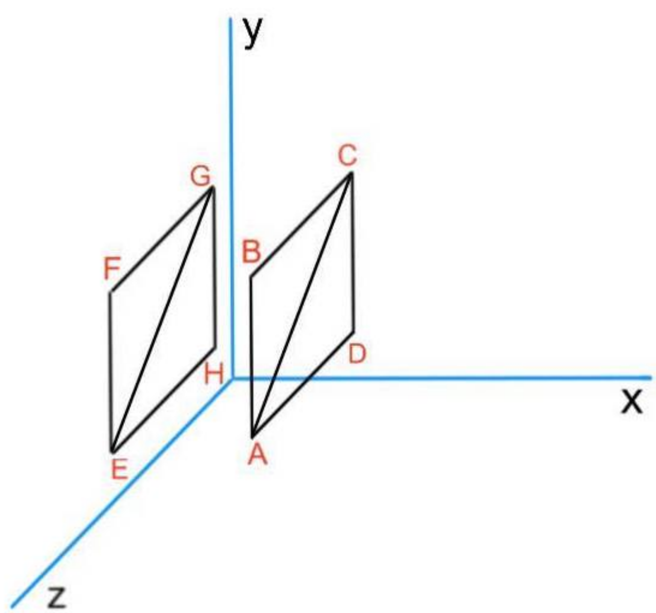

Figura 6: Faces YZ de uma caixa sem divisões

Para $i = 0$ **até div fazer** $i++\{$

$y = i \times divY$

Para $j = 0$ **até div fazer** $j++\{$

$z = j \times divZ$

Cálculo da face lateral direita - Triângulos ACB e ADC

$xa = dimX$

$ya = y - dimY$

$za = z - dimZ + divZ$

$xb = dimX$

$yb = y - dimY + divY$

$zb = z - dimZ + divZ$

$xc = dimX$

$yc = y - dimY + divY$

$zc = z - dimZ$

$xd = dimX$

$yd = y - dimY$

$zd = z - dimZ$

Cálculo da face lateral esquerda - Triângulos EFG e EGH

$xe = -dimX$

$ye = y - dimY$

$ze = z - dimZ + divZ$

$xf = -dimX$

$yf = y - dimY + divY$

$zf = z - dimZ + divZ$

$xg = -dimX$

$yg = y - dimY + divY$

$zg = z - dimZ$

$xh = -dimX$

$yh = -dimY$

$zh = z - dimZ$

}
}

Exemplo: *box 4 4 4 2*

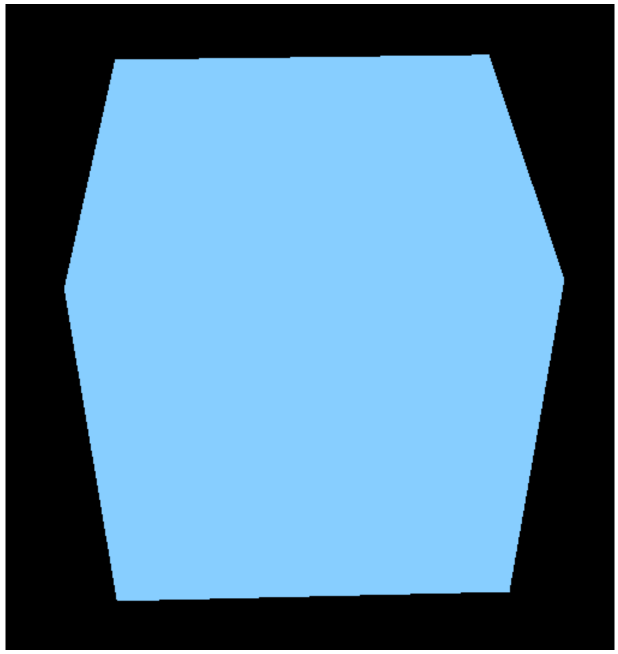

Figura 7: Exemplo de uma caixa com divisões

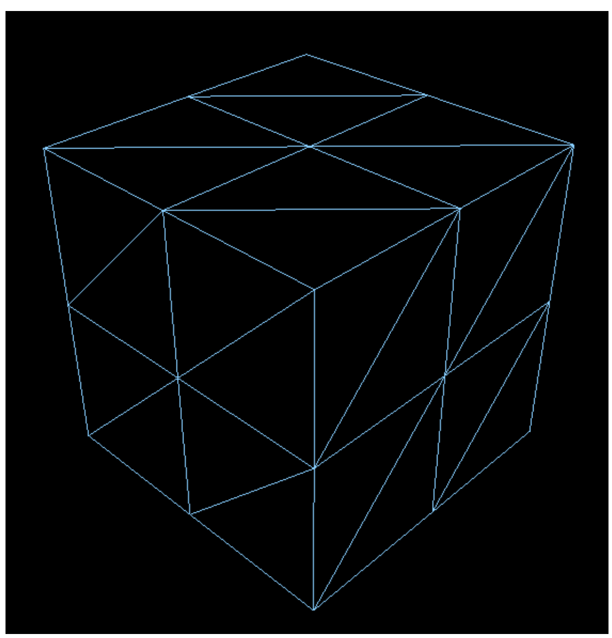

Figura 8: Exemplo de uma caixa com divisões com a representação dos triângulos

5 Esfera

O cálculo dos pontos de uma esfera necessita de 3 parâmetros: raio, slices que correspondem às divisões na vertical ao longo da esfera e stacks que correspondem às divisões na horizontal ao longo da esfera Quanto maior o número de slices e stacks, maior será o número de pontos a determinar, ou seja, melhor será a precisão da esfera. Sabemos que:

- A intersecção entre uma slice e uma stack origina 4 pontos
- A distância entre cada um destes pontos ao centro é o raio
- Podemos ter um vetor para cada ponto, e esse vetor tem dois ângulos, um relativo ao eixo dos yy(β) e outro relativo ao eixo dos zz(α). Em vez de termos relativo ao eixo dos zz, poderíamos ter relativo ao eixo dos xx
- O ângulo α$\in$ [0; 2π] e depende do número de slices
- O ângulo β$\in$ [0; π], e depende do número de stacks
- Temos um Δα que será calculado através $\frac{2\times\pi}{slices}$
- Temos um Δβ que será calculado através $\frac{\pi}{stacks}$

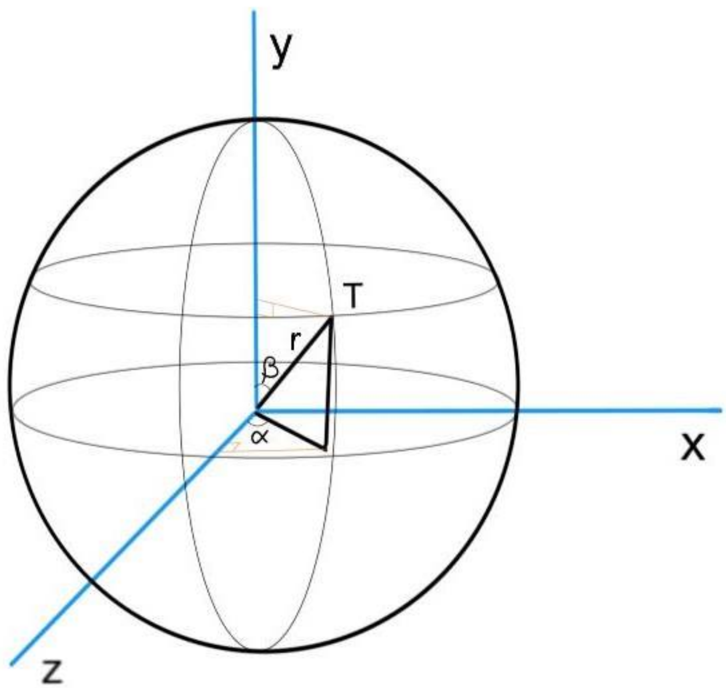

Figura 9: Representação de um ponto T na superfície de uma esfera e os respetivos ângulos

Após a nossa representação da esfera, facilmente conseguimos retirar as equações para transformar as coordenadas polares em cartesianas do ponto T:

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

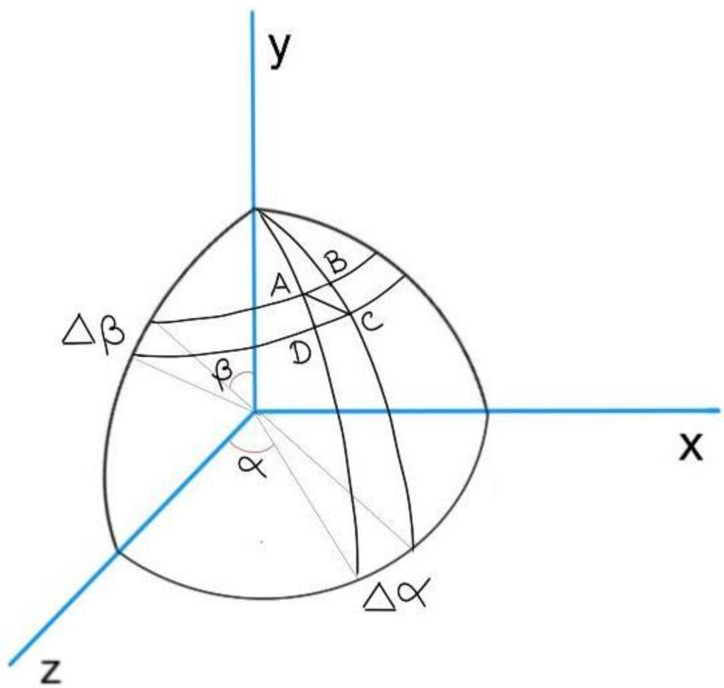

Figura 10: Representação dos 4 pontos de uma interseção

Na figura acima apresentada demonstramos um exemplo de uma interseção entre slices e stacks em que geramos os pontos A, B, C e D. Estes pontos servirão para calcular os vértices correspondentes aos triângulos ABC e ACD. Por último falta-nos determinar, os ângulos correspondentes aos pontos B, C e D.

Ponto	α	β
B	α+ Δα	β
C	α+ Δα	β+ Δβ
D	α	β+ Δβ

Para cada stack i {

$\beta = i \times \Delta\beta$

Para cada slice j {

$\alpha = j \times \Delta\alpha$

Ponto A

$x = r \times \sin(\beta) \times \sin(\alpha)$

$y = r \times \cos(\beta)$

$z = r \times \sin(\beta) \times \cos(\alpha)$

Ponto B

$x = r \times \sin(\beta) \times \sin(\alpha + \Delta\alpha)$

$y = r \times \cos(\beta)$

$z = r \times \sin(\beta) \times \cos(\alpha + \Delta\alpha)$

Ponto C

$x = r \times \sin(\beta + \Delta\beta) \times \sin(\alpha + \Delta\alpha)$

$y = r \times \cos(\beta + \Delta\beta)$

$z = r \times \sin(\beta + \Delta\beta) \times \cos(\alpha + \Delta\alpha)$

Ponto D

$x = r \times \sin(\beta + \Delta\beta) \times \sin(\alpha)$

$y = r \times \cos(\beta + \Delta\beta)$

$z = r \times \sin(\beta + \Delta\beta) \times \cos(\alpha)$

}

}

Exemplo: *sphere 4 100 100*

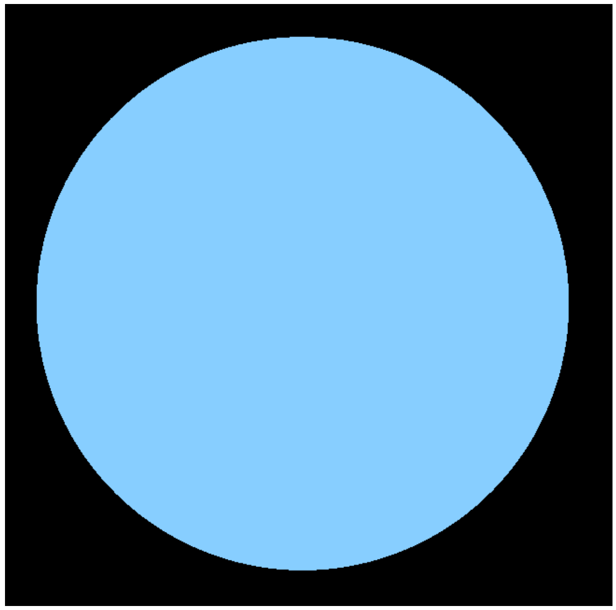

Figura 11: Exemplo de uma esfera

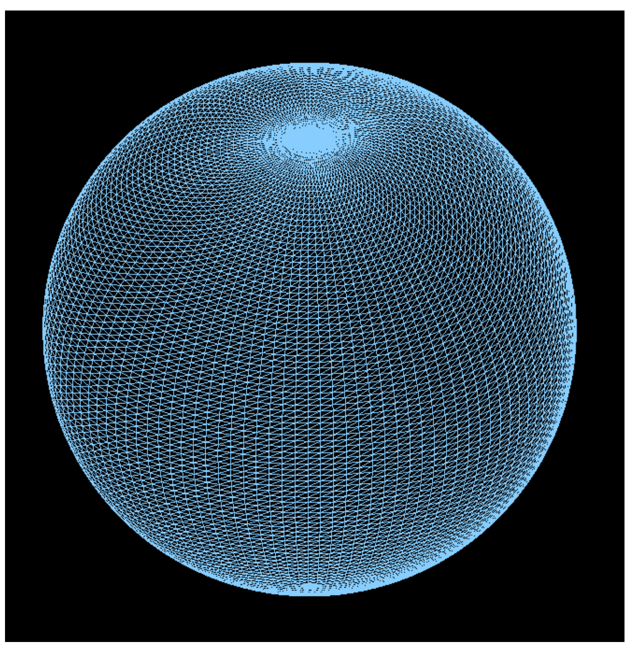

Figura 12: Exemplo de uma esfera com a representação dos triângulos

6 Cone

Para calcular os pontos que constituem um cone é necessário os seguintes parâmetros: raio, altura, slices e stacks. Dado não se tratar de uma primitiva regular, foi preciso optar por uma prática recorrente Divide and Conquer, resolvendo inicialmente a base do nosso cone, e só após isso a superfície lateral. Sabemos que a base vai ser constituída por vários triângulos unidos ao centro, cada um com o seu ângulo α respetivo. Para a geração dos pontos que irão representar a base do cone, foi necessário recorrer ao sistema de coordenadas polares, para transformar em coordenadas cartesianas. Apresentamos então as fórmulas para o cálculo da base:

$x = r \times \sin(\alpha)$

$z = r \times \cos(\alpha)$

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

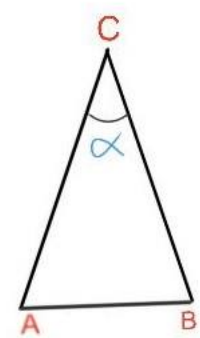

Figura 13: Imagem ilustrativa de um triângulo da base do cone

Para cada slice i $\{$

$\alpha = i \times \Delta\alpha$

Ponto A

$x = r \times \sin(\alpha)$

$y = 0$

$z = r \times \cos(\alpha)$

Ponto B

$x = r \times \sin(\alpha + \Delta\alpha)$

$y = 0$

$z = r \times \cos(\alpha + \Delta\alpha)$

Ponto C

$x = 0$

$y = 0$

$z = 0$

}

Os ciclos que foram implementados posteriormente para a geração dos vértices da superfície lateral, seguem a mesma lógica da esfera, a intersecção entre uma slice e uma stack resulta em quatro pontos. Porém existe a necessidade de definir um novo raio e aumentar a respectiva altura do modelo a cada camada ao longo da iteração. Além das fórmulas já declaradas acima que também iremos precisar para a superfície, temos mais esta equação:

$\Delta y = \frac{altura}{stacks}$

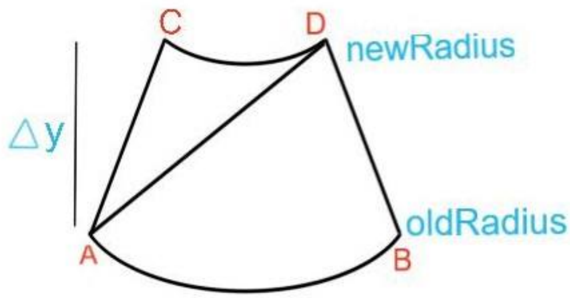

Figura 14: Imagem ilustrativa de uma intersecção na superfície lateral do cone

Para cada stack i {

$y = i \times \Delta y$

$newR = \frac{r}{altura} \times (altura - ((i+1) \times \Delta y))$

Para cada slice j {

$\alpha = j \times \Delta\alpha$

Ponto A

$x = oldR \times \sin(\alpha)$

$y = y$

$z = oldR \times \cos(\alpha)$

Ponto B

$x = oldR \times \sin(\alpha + \Delta\alpha)$

$y = y$

$z = oldR \times \cos(\alpha + \Delta\alpha)$

Ponto C

$x = newR \times \sin(\alpha)$

$y = y + \Delta y$

$z = newR \times \cos(\alpha)$

Ponto D

$x = newR \times \sin(\alpha + \Delta\alpha)$

$y = y + \Delta y$

$z = newR \times \cos(\alpha + \Delta\alpha)$

}

$oldR = newR;$

}

Exemplo: *cone 1 2 50 50*

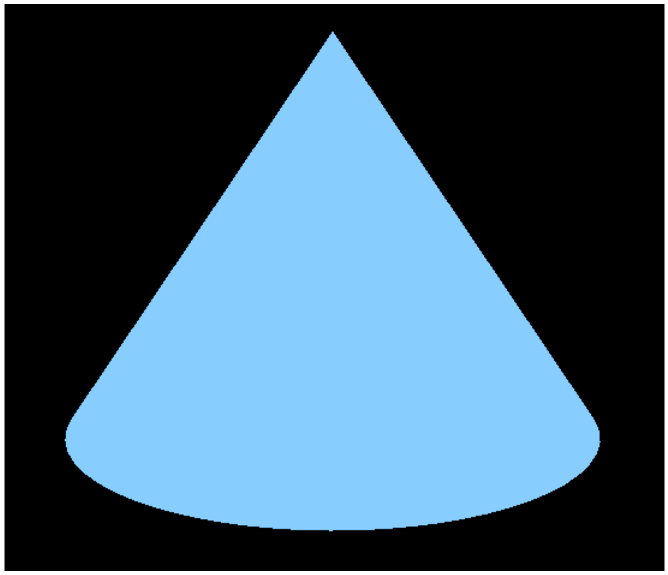

Figura 15: Exemplo de um cone

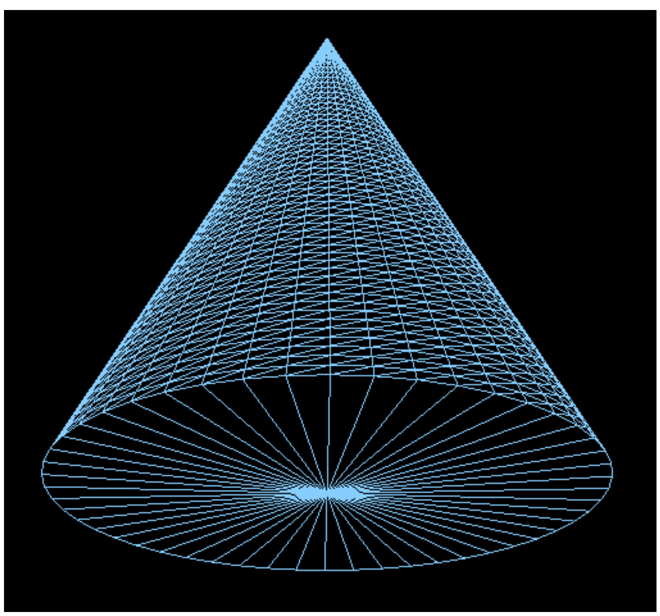

Figura 16: Exemplo de um cone com a representação dos triângulos

7 Motor

A aplicação Engine tem como objetivo ler e interpretar um ficheiro de configuração XML, com recurso à biblioteca tinyXML2, que contém uma lista com os ficheiros anteriormente criados pelo Gerador, e desenhar em OpenGL as figuras contidas nesse ficheiro.

Engine.exe <xmlFile>

8 Extras

Durante a realização do trabalho decidimos implementar algumas funcionalidades que por sua vez facilitaram a verificação dos resultados obtidos e permitiram uma melhor interação com os modelos produzidos pelo generator.

- PolygonMode : Polígonos podem ser desenhados de forma preenchida,apenas com as linhas e contorno ou somente os vértices. Um polígono tem dois lados (front e back), que podem ser renderizados diferentemente dependendo do lado que observador estiver a observar. Para tal utilizamos os seguintes comandos: *P, L e F.*

- Zoom: Optou-se por implementar o zoom de modo a ser possível fazer aproximações da câmera sobre os diferentes modelos , para tal efeito utilizamos as seguintes teclas:
  Zoom in : X
  Zoom out : Z

- Rotações e Translações : Durante a resolução dos problemas propostos, sentimos a necessidade de interagir com os modelos através de rotações e translações de forma a podermos concluir com toda a certeza que todas as faces/triângulos do modelo eram desenhadas. Para tal efeito utilizamos as seguintes teclas:
  Rotações : Q, E e as setas
  Translações: W, S, A e D

- ChangeModel: Para uma troca entre os diferentes modelos optou-se por realizar a função changeModel,para efectuar as mudanças utilizamos as seguintes teclas: *N e M.*

9 Conclusão

Com a elaboração desta fase do projeto aprofundamos os nossos conhecimentos em relação à ferramenta usada OpenGL, assim como exploramos a linguagem C++. Todos os requisitos propostos nesta fase foram cumpridos, estando todos funcionais, e ainda foram implementados algumas funcionalidades extra no motor para verificar a correta construção das primitivas.