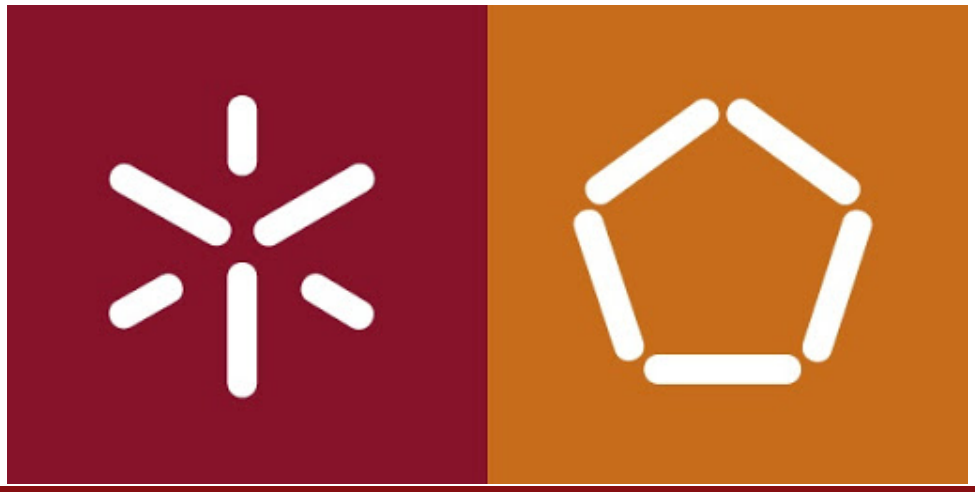

Computação Grafica
Universidade do Minho

Henrique Paz (a84372), João Queiros (a82422), José Santos (a84288), Pedro Gomes (a84220)

7 de Março de 2020

Conteúdo

1 Introdução

A primeira fase deste trabalho pratico têm como objetivo a criação de duas aplicações: uma aplicação para gerar os ficheiros com a informação dos modelos, ou seja os vertices de cada modelo, nomeadamente de um plano, uma caixa que poderá ser um cubo ou um paralelepípedo retangular, um cinlindro e uma esfera cada uma com parametros diferentes, e outra para criar uma *engine* que lê as configurações num documento xml para depois criar os modelos referidos.

2 Motor

O motor é o programa utilizado para desenhar os modelos contidos no ficheiro .xml que lhe é passado como parâmetro de entrada.

Para isso utilizamos duas estruturas auxiliares, uma para guardar os vértices, e contém 3 floats correspondendo ás 3 coordenadas, e utilizamos ainda outra estrutura, o modelo, para agregar os vértices num vetor para depois os desenhar.

Cada linha do ficheiro .3d corresponde ás 3 coordenadas necessárias para formar um vértice.

3 Gerador

O gerador, como o nome indica, gera os ficheiros .3d que posteriormente são referenciados num ficheiro .xml para serem desenhados pelo engine. Para tal, este recebe a informação acerca da figura a gerar, assim como os vários parâmetros necessários para gerar cada uma das figuras.

3.1 Plano

Para a criação do plano, é utilizado apenas um parâmetro: o comprimento do lado (*size*), sendo que o plano se trata de um quadrado centrado na origem formado por dois triângulos.

3.1.1 Equações

As equações das coordenadas dos vértices do plano tomam as seguintes formas (sendo s = size/2):

```
A = (s,0,s)
B = (s,0,-s)
C = (-s,0,-s)
D = (-s,0,s)
```

3.1.2 Algoritmos

Para gerar o plano através do size, este é dividido por 2, de modo a que as coordenadas x e z fiquem centradas na origem, e a partir daí são gerados os 6 vértices necessários para os 2 triângulos que formam o plano

```
V1 = A;
V2 = B;
V3 = C;
V4 = C;
V5 = D;
V6 = A;
```

3.1.3 Diagramas

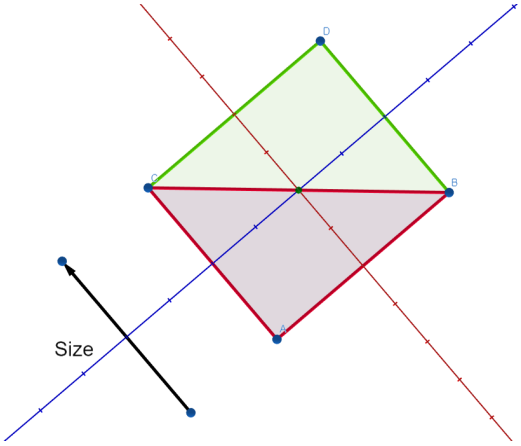

Figura 1: Triangulos do plano

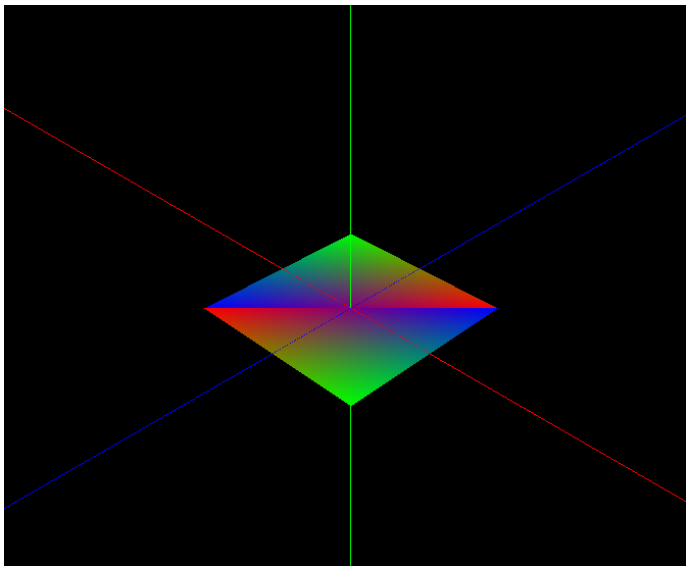

Figura 2: Plano centrado na origem de tamanho 4

3.2 Caixa

Para a criação da caixa, foram utilizados quatro parâmetros: o tamanho da caixa em cada eixo *x,y,z* e o numero de divisões por cada face *slices*, sendo que trata-se de paralelepípedo retangular (pode ou não ser um cubo), centrado na origem, formado por um conjunto de planos em cada face da caixa.

3.2.1 Equações

Cada face é composta por quadriláteros de número igual a:

```
divisions^2
```

Sendo que as dimensões de cada quadrilátero são dadas por:

```
sX = X / divisions
sY = Y / divisions
sZ = Z / divisions
```

3.2.2 Algoritmos

Para gerar a caixa, optamos por gerar cada uma das faces independentemente uma da outra. Assim sendo, para formar uma face da caixa, os vértices de cada quadrilátero que a compõe são definidos da seguinte forma: (usando como exemplo a face frontal da caixa)

```
for(int i = 0; i < divisions; i++){
    for(int j = 0; j < divisions; j++){
        A = (X/2, i*sY - Y/2, Z/2 - sZ*j);
        B = (X/2, i*sY - Y/2, Z/2 - sZ*(j+1));
        C = (X/2, (i+1)*sY - Y/2, Z/2 - sZ*(j+1));
        D = (X/2, (i+1)*sY - Y/2, Z/2 - sZ*j);

        V1 = A;
        V2 = B;
        V3 = C;
        V4 = C;
        V5 = D;
        V6 = A;
    }
}
```

3.2.3 Diagramas

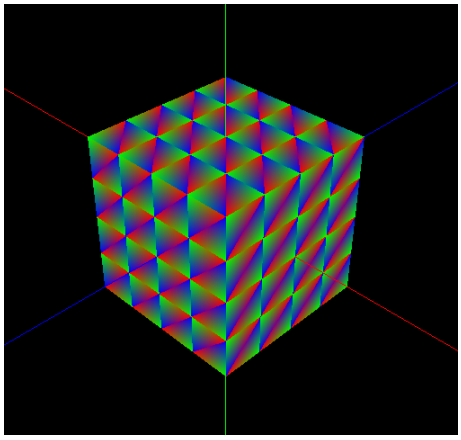

Figura 3: Caixa em forma de cubo com tamanho 3 nos 3 eixos e com 4 divisões

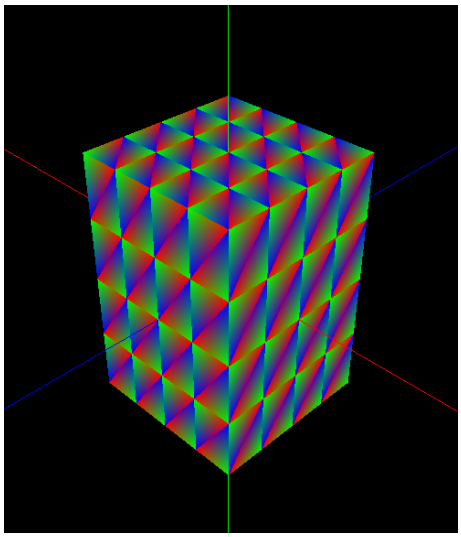

Figura 4: Caixa em forma de paralelepípedo retangular

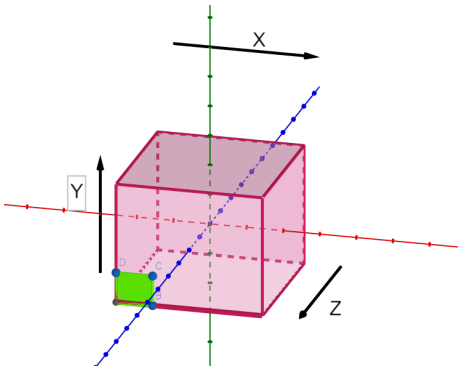

Figura 5: Diagrama da caixa

3.3 Cone

Para a criação do cone são necessários 4 parâmetros: O raio da base(*raio*), a altura do cone(*altura*), o número de slices (*slices*) e o número de stacks(*stacks*).

3.3.1 Equações

Para gerar o cone, começamos por gerar o círculo correspondente á base do cone, sendo que este está divido em triângulos de número igual ao número de slices, sendo que o ângulo formado pela origem e pelos outros dois vértices de cada triângulo é dado por:

```
anguloH = 2 * pi / raio
```

Dado este ângulo, é possível assim calcular as coordenadas x e z para qualquer vértice que não se encontre na origem: (i corresponde a uma variavél cujos valores vão de 0 ao número de slices)

```
x1 = raio * cos(anguloH * i)
z1 = raio * sin(anguloH * i)
x2 = raio * cos(anguloH * (i+1))
z2 = raio * sin(anguloH * (i+1))
```

, onde x1,z1 correspondem ás coordenadas de um dos vértices e x2,z2 ás coordenadas do segundo, sendo que o terceiro vértice possui como coordenadas a origem.

Para gerar o resto do cone, é necessário ainda uma componente que diminua o valor das coordenadas x e z e á medida que se gera os vértices das stacks superiores. (j corresponde a uma variável cujos valores vão de 0 ao número de stacks)

```
altura1 = (1-(j/stacks))
altura2 = (1-((j+1)/stacks))
```

, onde a altura de cada stack é dada por:

```
alturaS = altura / stacks
```

3.3.2 Algoritmos

Para gerar a base do cone, iteramos a variável i pelo número de slices, gerando deste modo todos os seus triângulos constituintes:

```
for(int i = 0; i < slices; i++){
    V1 = (0,0,0);
    V2 = (x1,0,z1);
    V3 = (x2,0,z2);
}
```

Para gerar o resto do cone, é necessário ter um ciclo para iterar pelas stacks, assim como um segundo ciclo dentro do primeiro para iterar pelas slices: (x1,x2,z1 e z2 são calculados segundo a mesma equação que é utilizada para a base do cone)

```
for(int j = 0; j < stacks; j++){
    for(int i = 0; i < slices; i++){
        A = (x1*altura1,alturaS*j,z1*altura1);
        B = (x1*altura2,alturaS*(j+1),z1*altura2);
        C = (x2*altura1,alturaS*j,z2*altura1);
        D = (x2*altura2,alturaS*(j+1),x2*altura2);

        V1= A;
        V2= B
        V3= C;
        V4= C;
        V5= D;
        V6= A;

    }
}
```

3.3.3 Diagramas

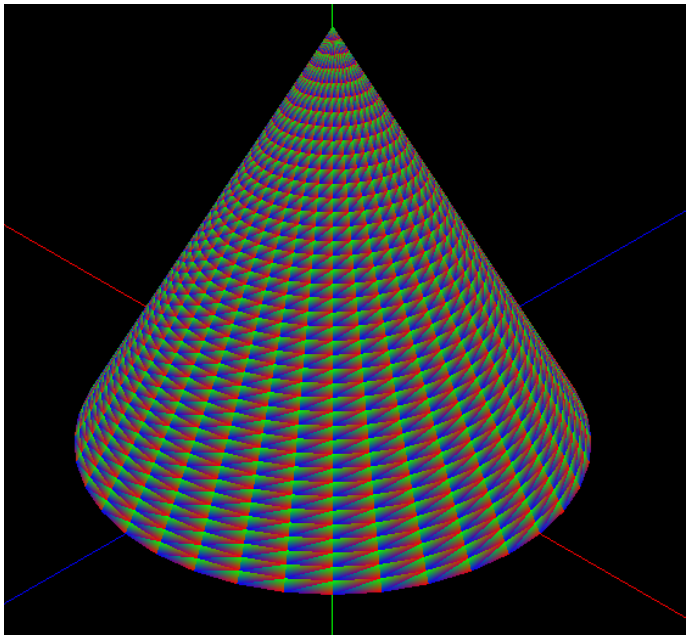

Figura 6: Cone com raio 2, altura 3

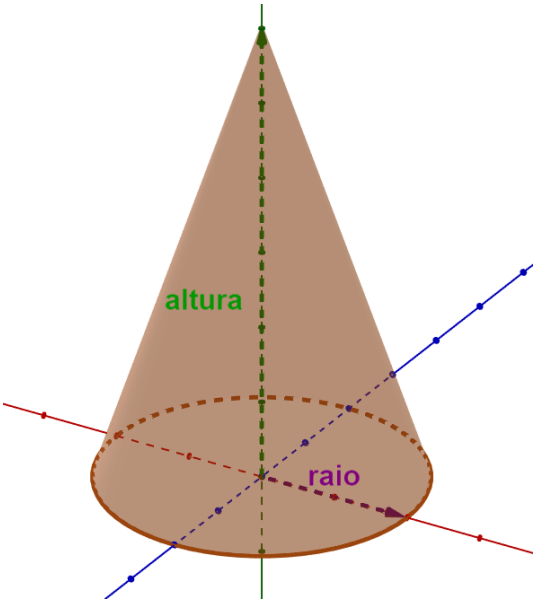

Figura 7: Diagrama do cone

3.4 Esfera

Para a criação da esfera são necessários 3 parâmetros: o seu raio(*raio*), o número de slices(*slices*) e o número de stacks(*stacks*).

3.4.1 Equações

Tal como no cone, é necessário definir o ângulo que faz variar as componentes de x e z dos vértices dos triângulos, e tal como no cone este ângulo é dado por:

```
anguloH = 2 * pi / raio
```

No entanto, no caso da esfera, é ainda necessário um outro ângulo, que corresponde á variação vertical das coordenadas x e z, e é dado por:

```
anguloV = pi / stacks
```

Como estamos a gerar a esfera a partir da sua base, e a desenhar uma stack de cada vez, este ângulo varia apenas entre e 0 e pi, pois apenas varia de 0º a 180º graus, enquanto que o primeiro varia de 0º a 360º. Deste modo podemos definir a componente que altera os valores

3.4.2 Algoritmos

Para gerar a esfera, iteramos uma variável j pelo número de stacks, e i pelo número de slices

```
    for(int j = 0; j <stacks; j++){
        for(int i = 0; i < slices; i++){
            x1 = raio * cos(anguloH * i);
z1 = raio * sin(anguloH * i);
x2 = raio * cos(anguloH * (i + 1));
z2 = raio * sin(anguloH * (i + 1));

y1 = -raio + anguloV * j);
y2 = -raio + anguloV *(j + 1));


            A= (x1 *altura1, y1, z1*altura1);
            B= (x1*altura2, y2, z1*altura2);
            C= (x2*altura1,y1,z2*altura1);
            D= (x2*altura2,y1,z2*altura1);


        }
    }
```

3.4.3 Diagramas

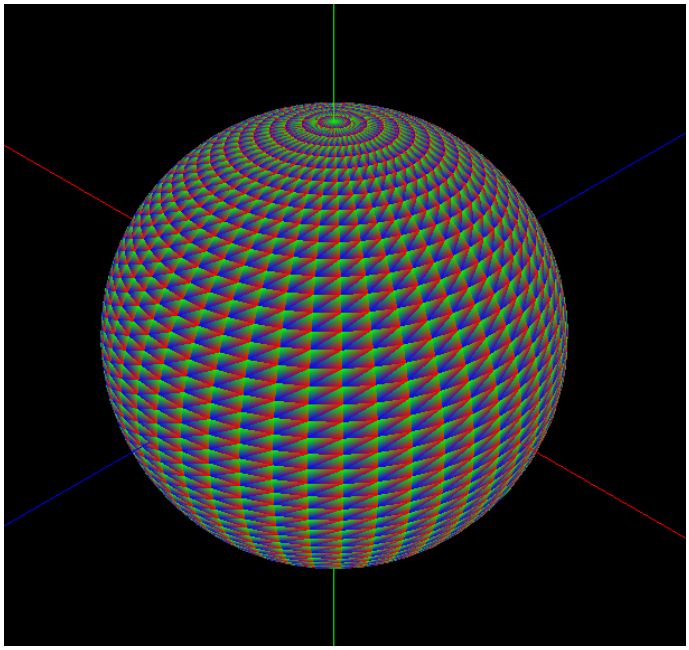

Figura 8: Esfera com raio 2

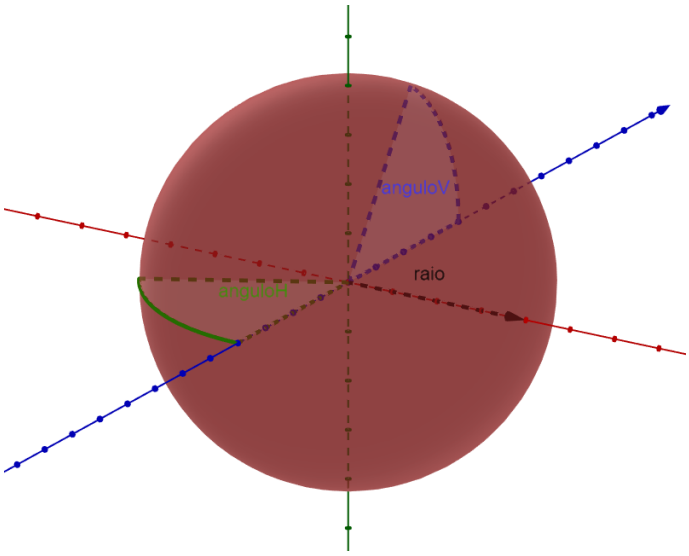

Figura 9: Diagrama de Esfera