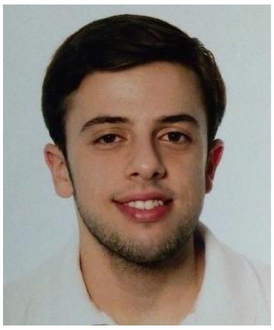
Universidade do Minho

Departamento de Informática

Graphical primitives

Mestrado Integrado em Engenharia Informática

Computação gráfica



Jaime Leite (A80757)

Bruno Veloso (A78352)

Índice

Introdução………3

Fase 1…….4

Primitivas gráficas ……………………………………………………………………………………………4

1. Plano……………………………………………………………………………………………………4

2. Caixa……………………………………………………………………………………………………5

3. Esfera…………………………………………………………………………………………………… 7

4. Cone……………………………………………………………………………………………………9

Generator ……………………………………………………………………………………………………11

Engine……12

Conclusão……………………………………………………………………………………………………..13

Índice de figuras

Figura 1 – desenho de um plano…………………………………………………………………………… 4

Figura 2 – desenho de uma caixa…………………………………………………………………………6

Figura 3 - demonstração dos pontos da esfera…………………………………………………… 7

Figura 4 – desenho de uma esfera………………………………………………………………………… 8

Figura 5 – desenho de uma slice para a base do cone…………………………………………… 9

Figura 6 – desenho de um lado do cone……………………………………………………………10

Figura 7 – cone visto da parte de baixo……………………………………………………………10

Figura 8 – cone visto da parte de cima……………………………………………………………10

Figura 9 - desenho do plano, caixa, cone e esfera……………………………………………12

Introdução

Partindo do enunciado que nos foi proposto, apresentaremos neste relatório o trabalho desenvolvido na primeira fase do projeto relativo à unidade curricular de *Computação Gráfica*.

Apresentaremos, primeiramente, os vários passos para a construção de figuras 3D, nomeadamente o plano, caixa, esfera e cone, tendo por base a figura geométrica *triângulo*.

Na segunda parte do relatório, iremos abordar a elaboração de um generator e de um engine. O primeiro irá escrever em vários ficheiros todos os pontos relativos às figuras 3D referidas anteriormente e o segundo é responsável por ler um ficheiro XML (que contem informação das figuras a serem desenhadas) e a partir do mesmo desenhar os modelos indicados no enunciado do projeto, partindo dos pontos guardados pelo generator.

Fase 1

Primitivas gráficas

Apresentaremos de seguida os passos para a construção das figuras propostas no enunciado.

1. Plano

No desenho de um plano, são usados seis pontos, derivados de dois triângulos. A função responsável por tal desenho é designada por “drawPlane” e recebe como parâmetros o tamanho do lado do plano e o ficheiro onde irá armazenar os seus pontos.

No corpo da função, os pontos são chamados pela seguinte ordem: A,B,C,D, respeitando a regra da mão direita.

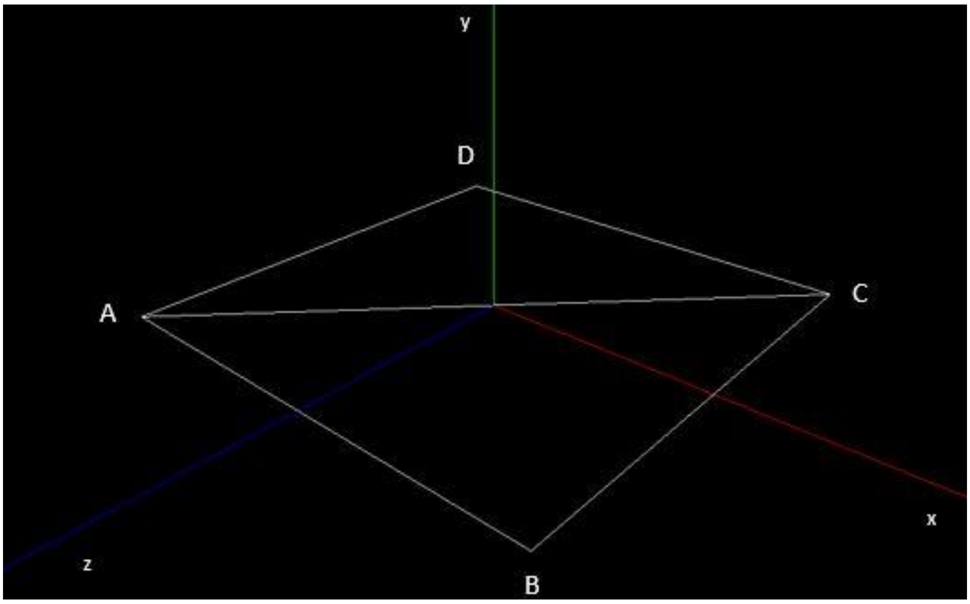


Figura 1: desenho de um plano

Através da figura, verifica-se que (0,0,0) é o ponto central do plano. Todos os pontos do plano têm ordenada 0, e as suas abcissas e cotas têm como valor escalar metade do comprimento passado na função “drawPlane”, permitindo assim que os mesmos fiquem uniformemente distribuídos relativamente à origem do referencial.

2. Caixa

A função responsável por desenhar uma caixa é a “drawBox”. Recebe como parâmetros as medidas necessárias para a sua construção (como sendo a largura, altura, comprimento e número de divisões) e o nome de um ficheiro (onde irão ficar armazenados os pontos da figura).

Sabendo que a caixa pode ter várias divisões(quadrados), cada uma das faces da mesma tem uma função auxiliar associada, que representa o desenho de uma das suas divisões. Estas funções auxiliares, invocadas várias vezes dentro de ciclos, criam as diferentes faces contidas nos diferentes planos do referencial.

O processo de construção de uma divisão é o seguinte: partindo de dois pontos, que vão variando em função dos parâmetros inicialmente passados à função e que representam a diagonal de um quadrado, obtêm-se os restantes pontos, sendo assim possível formar uma divisória.

Para as divisórias relativas às faces contidas no eixo dos XX: a função que desenha uma divisão vai ser invocada para desenhar tanto na parte positiva como na parte negativa do eixo. Para a parte positiva do eixo irão ser passados os pontos ((largura / 2),-(altura / 2),-(comprimento / 2)) e ((largura / 2),-(altura / 2) + deltaY1,-(comprimento / 2) + deltaZ1). Já para a parte negativa do eixo irão ser referenciados os pontos (-(largura / 2),-(altura / 2),-(comprimento / 2)) e (-(largura / 2),-(altura / 2) + deltaY1,-(comprimento / 2) + deltaZ1).

Por cada iteração dos ciclos, estes valores variam da seguinte forma:

z = -comprimento / 2 + j * deltaZ1;

y = (-altura / 2) + (i * deltaY1),

em que i e j estão limitados ao número de divisões;

deltaY1 = altura / divisões;

deltaZ1 = comprimento / divisões;

Para as faces contidas no eixo dos YY: a função que desenha uma divisão vai ser invocada para desenhar tanto na parte positiva como na parte negativa do eixo. Para a parte positiva do eixo irão ser passados os pontos ((largura / 2), (altura / 2),(comprimento / 2)) e ((largura / 2) - deltaX, (altura / 2),(comprimento / 2) - deltaZ). Já para a parte negativa do eixo irão ser referenciados os pontos ((largura / 2), -(altura / 2),(comprimento / 2)) e ((largura / 2) - deltaX, -(altura / 2),(comprimento / 2) - deltaZ).

Em cada iteração dos ciclos, estes valores variam da seguinte forma:

x = (largura / 2) - j * deltaX;

z = (comprimento / 2) - (i * deltaZ),

em que i e j estão limitados ao número de divisões;

deltaX = largura / divisões;
deltaZ = comprimento / divisiões;

Para as faces contidas no eixo dos ZZ: a função que desenha uma divisão vai ser invocada para desenhar tanto na parte positiva como na parte negativa do eixo. Para a parte positiva do eixo irão ser passados os pontos ((largura / 2), -(altura / 2),(comprimento / 2)) e ((largura / 2) - deltaX, -(altura / 2) + deltaY,(comprimento / 2)). Já para a parte negativa do eixo irão ser referenciados os pontos ((largura / 2), -(altura / 2),-(comprimento / 2)) e ((largura / 2) - deltaX, -(altura / 2) + deltaY,-(comprimento / 2)).

Em cada iteração dos ciclos, estes valores variam da seguinte forma:
x = (largura / 2) - j * deltaX;
y = (-altura / 2) + i * deltaY;
em que i e j estão limitados ao número de divisões;

deltaX = largura / divisiões;
deltaY = altura / divisiões;

No final, será possível ver cada superfície do sólido (centrado no ponto (0,0,0)) com (2*divisões) quadrados.

Assim sendo, a implementação de uma caixa assenta no desenho recursivo das repartições relativas às seis faces da mesma, sendo os seus tamanhos dependentes dos argumentos introduzidos na função “drawBox”.

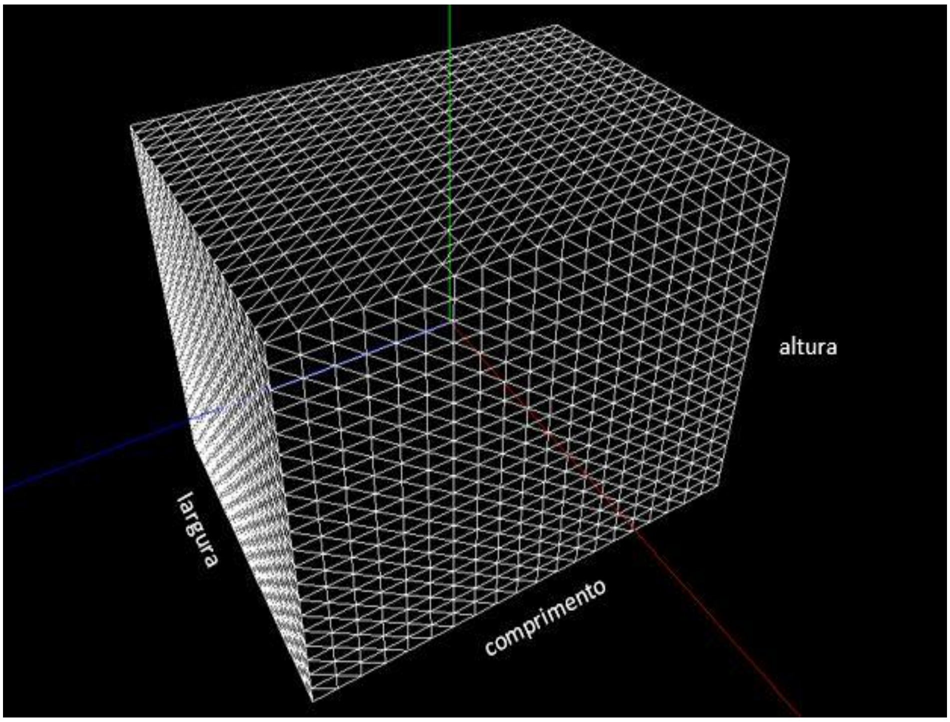


Figura 2: desenho de uma caixa

3. Esfera

Para ser possível construir a esfera era necessário saber o raio da esfera, bem como o número de slices e stacks desta. A metodologia utilizada passou por usar dois ângulos, um θ (beta) que vai desde 0 a 2*pi (0-360 graus) e um φ(fi) que que vai desde 0 a pi (0-180 graus) que representará só a superfície esférica superior. A ideia passa por termos um valor de ângulo “ang1” e um “ang2”, em que “ang1” representa o valor de (2*pi) / slices (o nosso beta), e “ang2” representa pi/stacks (nosso fi). Para desenhar basta ter 2 ciclos, um ciclo que vai desde 0 até stacks/2 (desenhar parte superior toda) em que tem um ciclo interior que vai desde 0 até slices para desenhar os triângulos todos à volta.

Por cada uma desta (stack/2)*slices iterações são feitos 4 triângulos, 2 para a parte superior da esfera e 2 para a parte inferior. O primeiro triângulo é formado pelos pontos P1,P2 e P3. P1 tem de coordenadas: x=radius*cos(nível)*sin(angulo), y = radius*sin(nível) e z=radius*cos(nível)*cos(angulo); P2 tem de coordenadas: x=radius*cos(nivel)*sin(angulo2);y=radius*sin(nivel);z=radius*cos(nivel)*cos(angulo2); P3 que tem de coordenadas: x=radius*cos(nivel2)*sin(angulo2);y=radius*sin(nivel2), z=radius* cos(nivel2) *cos(angulo2), em que radius é o raio da esfera, “nivel” é o valor i* (pi/stacks) em que i é a stack onde estamos (vai desde 0 a stacks/2) representa o φ, “nível2” é a camada superior ou seja (i+1)*(pi/stacks), o nosso “angulo” representa a slice onde estamos vezes o (2*pi)/slices ou seja o beta, e por fim “angulo2” representa o “angulo” + (2*pi)/slices ou seja o. O x,y, e z são calculados através das fórmulas que se podem ver na figura abaixo.

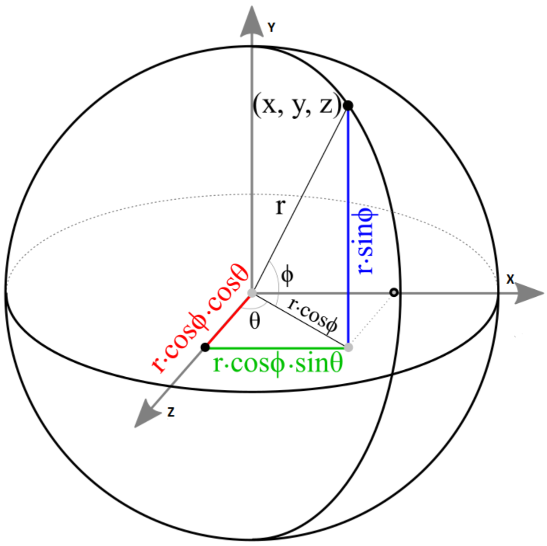


Figura 3: demonstração dos pontos da esfera

O triângulo complementar ao primeiro desenhado para a face superior é representado pelos pontos P4,P5 e P6. P4 tem de coordenadas x=radius*cos(nivel2)*sin(angulo2),y=radius*sin(nivel2),z=radius*cos(nivel2)*cos(angulo2), P5 tem x=radius*cos(nivel2)*sin(angulo), y=radius*sin(nivel2) e z=radius* cos(nivel2)*cos(angulo), e P6 tem de coordenadas x=radius*cos(nivel)*sin(angulo), y=radius*sin(nivel) e z=radius * cos(nivel)*cos(angulo), assim sendo com este triângulo formado e executando as (stacks/2) * slices iterações temos a parte superior da esfera desenhada.

Quando à parte inferior, bastou mudar os ângulos para negativos, ou seja, em cada iteração do ciclo bastou desenhar 2 triângulos em que “angulo” passa a “-angulo”, “angulo2” passa a “-angulo2”, “nivel” a “-nivel” e “nivel2” a “-nivel2”. Assim sendo ficou construída na sua totalidade a esfera.

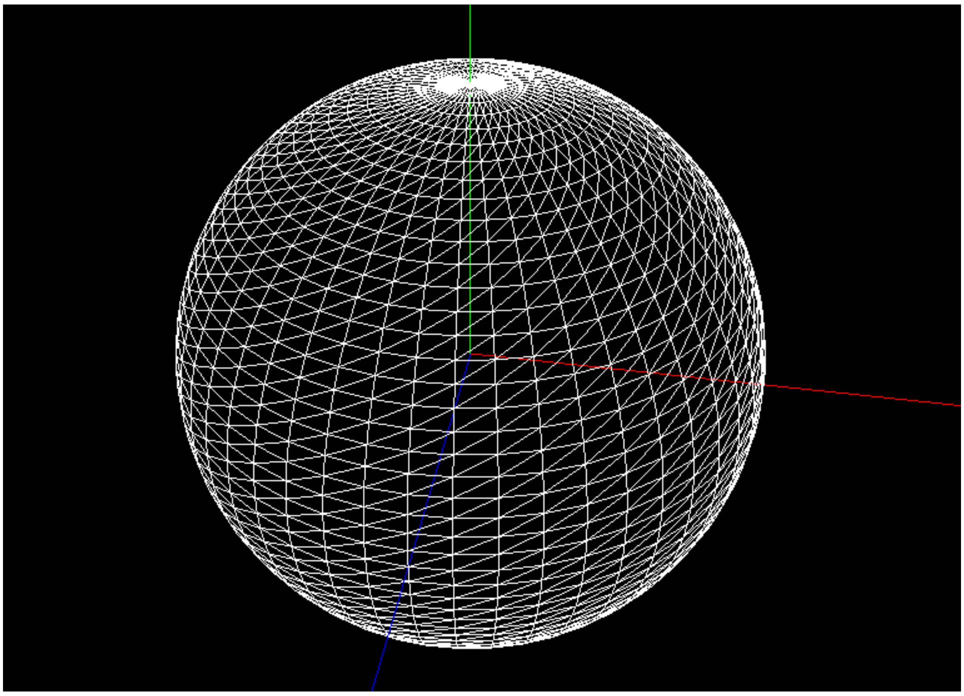


Figura 4: desenho de uma esfera

4. Cone

Para conseguirmos desenhar um cone necessitamos de ter um raio, uma altura, o número de slices e o número de stacks que este deve ter, de forma a obter com sucesso a sua criação.

Para desenhar a sua base precisamos do seu raio e o número de fatias(slices). Uma vez que a base tem como ângulo total 360 graus (2*pi radianos), sabemos que cada slice vai ter de ângulo ang= (2*pi) / slices. Depois desta análise para desenhar, basta um ciclo que cria uma slice de cada vez. As coordenadas da slice serão x= radius*sin(z * ang),y= 0 e z=radius * cos(z * ang) para o primeiro ponto, em que z corresponde à fatia a desenhar começando pela slice 0. O segundo ponto será x=0, y=0 e z=0, uma vez que o cone está centrado no referencial inicial. Já o terceiro ponto será x = radius*sin(ang * (z + 1)), y= 0 e z= radius*cos(ang * (z + 1)), que corresponde ao primeiro ponto da próxima slice. Tendo isto estabelecido, este triângulo é desenhado no sentido dos ponteiros do relógio para poder ser visto por baixo.

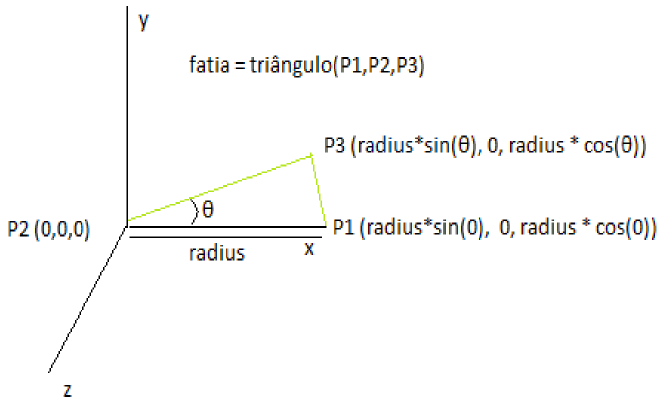


Figura 5: desenho de uma slice para a base do cone

Quanto aos lados do cone, o número de triângulos que serão precisos corresponde a slices* stacks* 2, logo para cada camada formam-se as slices todas e só depois se passa para a próxima stack, em que de cada vez se desenham 2 triângulos. Cada camada superior tem menor raio que a inferior usando a fórmula novoRaio = radius - i * (radius / stacks) em que 'i' corresponde à stack em que nós estamos a desenhar sendo que a última stack terá novoRaio = 0.

O primeiro triângulo será formado por, ponto P1 que será x = raio2*sin(ang* z), y = (i + 1) * (height / stacks) e z = raio2*cos(ang * z), 'z' é a slice que estamos, raio2 é o raio da camada superior e 'i' é a stack. Ponto P2 será x=raio*sin(z*ang), y=i * (height / stacks) e z=raio*cos(z*ang). E o ponto P3 x=raio*sin(ang * (z + 1)), y=i * (height / stacks) e z=raio*cos(ang*(z + 1)), com estes 3 pontos desenhamos 1 triângulo (P1,P2,P3).

O segundo triângulo será formado por, ponto P4 será x=raio2*sin(z*ang), y=(i + 1) * (height / stacks) e z=raio2*cos(z * ang) e este ponto corresponde ao ponto P1. P5 será x= raio*sin(ang * (z + 1)), y= i * (height / stacks) e z=raio*cos(ang * (z + 1)) e o P6 x= raio2*sin(ang * (z + 1)),y=(i + 1) * (height / stacks) e z= raio2*cos(ang * (z + 1)), triângulo final fica (P4,P5,P6). Ambos os triângulos são desenhados no sentido contrário ao ponteiro dos relógios.

Efetuando isto para as stacks * slices iterações formamos o cone inteiro.

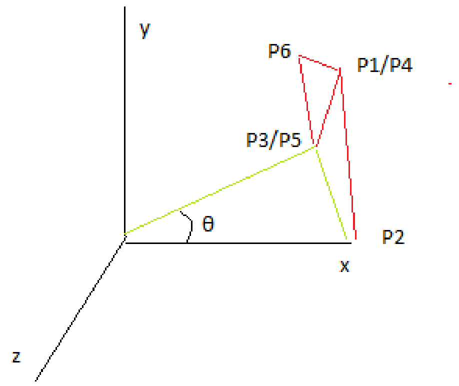


Figura 6: desenho de um lado do cone

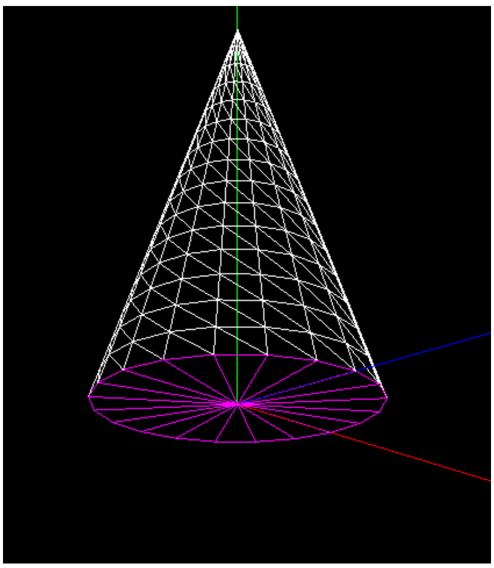


Figura 7: cone visto da parte de baixo

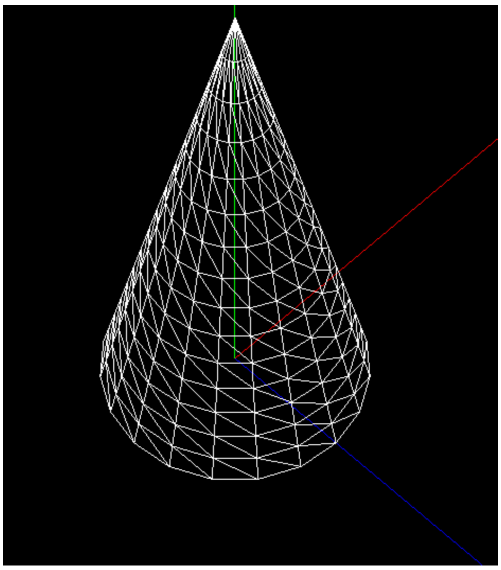


Figura 8: cone visto da parte de cima

Generator

O Generator é composto por todas as funções que desenham o plano, a caixa, a esfera e o cone. Essas funções são designadas por “drawPlane”, “drawBox”, “drawSphere” e “drawCone”, respetivamente, e, conforme o algoritmo desenvolvido, é guardada em ficheiros a informação relativa aos triângulos de cada figura geométrica. A linha inicial desses ficheiros tem um número representativo da quantidade de triângulos necessários para se formar a figura e as restantes linhas contêm nove valores, ou seja, as coordenadas de cada ponto que está contido num triângulo.

Na main do programa, o segundo argumento indicado na linha de comandos é analisado e, de seguida, é invocada a função respetiva a determinada figura. Na linha de comandos é também obrigatória a indicação do ficheiro para onde serão escritos os pontos da figura, bem como todos os parâmetros necessários para a sua construção.

Engine

O objetivo do “engine” passa por ler um ficheiro xml, no qual este terá indicação de ficheiros em que cada um terá uma figura geométrica na forma de pontos e o número de triângulos na primeira linha de cada.

Para ser possível, usámos o “tinyxml2” para dar parse da informação do xml. Guardamos esta informação do xml numa “struct Document” e fizemos um parse que através de um ficheiro, de uma figura geométrica (array de “Triangulo” em que “Triangulo” é uma “struct” por nós definida que contém 9 valores que correspondem a todos os valores para formar um triângulo), e também do tamanho da figura geométrica (inicialmente a 0). No final deste segundo parse, temos armazenada toda a informação de cada ficheiro que estava no xml e com o uso de uma outra função conseguimos desenhar cada figura geométrica tendo em conta o seu tamanho.

No fim serão desenhadas todas as figuras, umas em cima de outras.

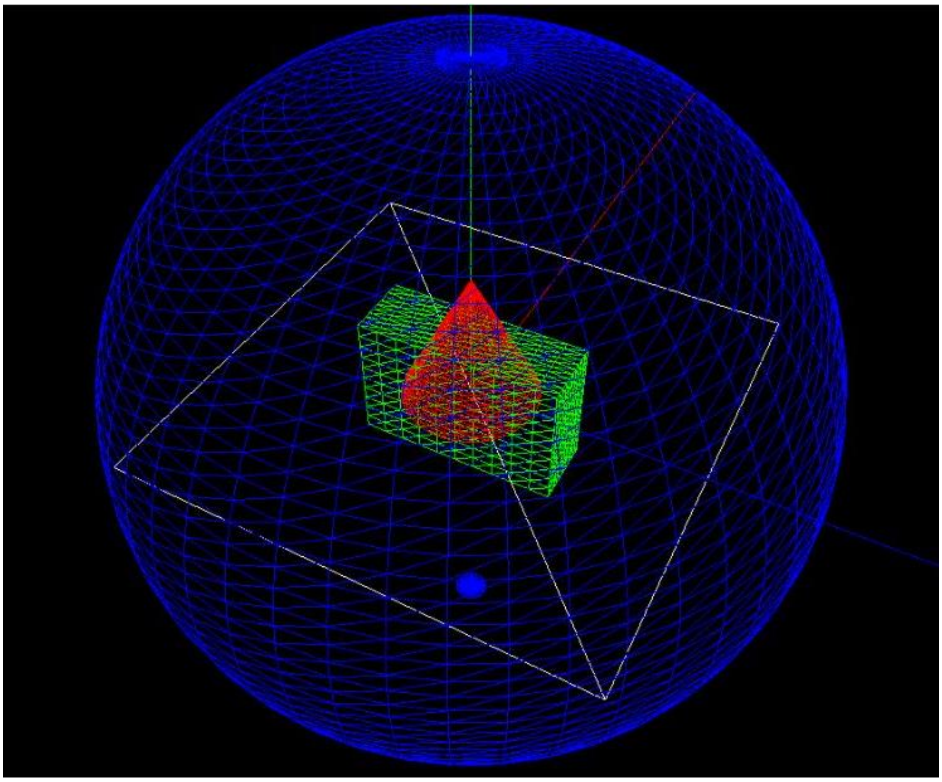


Figura 9: desenho do plano, caixa, cone e esfera

Conclusão

O grupo considera que alcançou os objetivos propostos para a primeira fase do projeto. Desenhamos, na totalidade, as figuras que eram referidas no enunciado, bem como desenvolvemos o generator e o engine, sendo que o primeiro nos permite guardar as informações relativas às várias figuras e o segundo constrói as figuras a apresenta o seu aspeto para o utilizador.

Consideramos benéfica esta primeira etapa, pois apesar de estas figuras já estarem implementadas no *OpenGl*, compreendemos melhor como este processa a sua construção.

Concluindo, o processo de aprendizagem nesta primeira etapa será importante para as fases seguintes do projeto.