



Universidade do Minho

Computação Gráfica

Mestrado Integrado em Engenharia Informática
3ºano-2ºsemestre

José Pinto A81317 Luís Correia A81141
Pedro Barbosa A82068

Trabalho Prático - Parte 1

Resumo

Foi-nos proposto na UC de Computação Gráfica o desenvolvimento de um mini mecanismo 3D baseado num cenário gráfico utilizando as ferramentas utilizadas na cadeira, C++ e OpenGL.

Este relatório é referente à primeira fase onde nos foi proposto a criação de algumas primitivas gráficas.

Março 2019

Conteúdo

1	Introdução	1
2	Arquitetura do código	1
2.1	Generator	1
2.2	Engine	1
3	Primitivas geométricas	2
3.1	Plane	2
3.2	Box	3
3.3	Sphere	4
3.4	Cone	6
4	Modelos	8
5	Exemplos primitivas gráficas	8

1 Introdução

Nesta primeira fase do trabalho prático foram criadas duas aplicações, o *Generator* e o *Engine*.

O *Generator* consegue gerar os modelos para o plano, a caixa, a esfera e o cone. Estes modelos, podem depois ser incorporados em ficheiros de configuração, em XML, que podem ser lidos pelo *Engine*. Este, a partir dessas configurações, desenha os modelos pretendidos.

2 Arquitetura do código

Como foi mencionado anteriormente foram criadas duas aplicações, o *Generator* e o *Engine*.

2.1 Generator

O generator gera os ficheiros onde se encontram os vértices de determinada primitiva geométrica. Este recebe como parâmetros o nome da primitiva, os parâmetros desta e um ficheiro onde vai escrever os vértices.

O generator possui um manual para auxiliar na invocação da aplicação. Para aceder ao manual, basta executar `./generator -h`. Com isto, são apresentadas as primitivas geométricas possíveis, bem como os parâmetros de cada uma. Na figura 1 é possível observar o menu do generator.

```
#####  
#  
# Como invocar:  
#   ./generator {COMANDO} {FICHEIRO}  
#  
# Comandos:  
#   -Plano [Tamanho]  
#   -Caixa [Tamanho X] [Tamanho Y] [Tamanho Z] [DIVISÕES]  
#   -Esfera [RAIO] [SLICES] [STACKS]  
#   -Cone [RAIO] [ALTURA] [SLICES] [STACKS]  
#  
#####
```

Figura 1: Menu do generator.

2.2 Engine

O engine está encarregue de receber um ficheiro de configuração XML, e desenhar os modelos nela presentes. Nesta fase, o ficheiro XML apenas contém os modelos (.3d) previamente gerados, sendo que no futuro irá suportar configurações adicionais (como transformações geométricas). O engine lê e desenha os vértices de cada ficheiro modelo.

O engine possui um manual que pode ser invocado da forma `./engine -h`. Uma vez executado este comando, é apresentado como a aplicação pode ser chamada, e os movimentos que é possível executar com a câmara.

Na figura 2 é possível verificar os controles que podem ser utilizados para facilitar a visualização dos modelos. De notar que é possível mover a câmara verticalmente e horizontalmente, bem como aproximar/afastar a câmara do ponto onde está a olhar. Também é possível modificar a forma como se visualiza a figura. É possível observá-la por pontos, linhas ou totalmente preenchida.

```
#####
# Como invocar: ./engine <XML file>
#
#
# Movimentos:
#   w: Move a câmara para cima
#   s: Move a câmara para baixo
#   a: Move a câmara para a esquerda
#   d: Move a câmara para a direita
#   KEY_UP: Reduz o raio da câmara
#   KEY_DOWN: Aumenta o raio da câmara
#   1: GL_POINT
#   2: GL_LINE
#   3: GL_FILL
#
#####
```

Figura 2: Menu do engine.

3 Primitivas geométricas

3.1 Plane

O plano é composto por dois triângulos. O plano está contido no plano XZ e é centrado na origem.

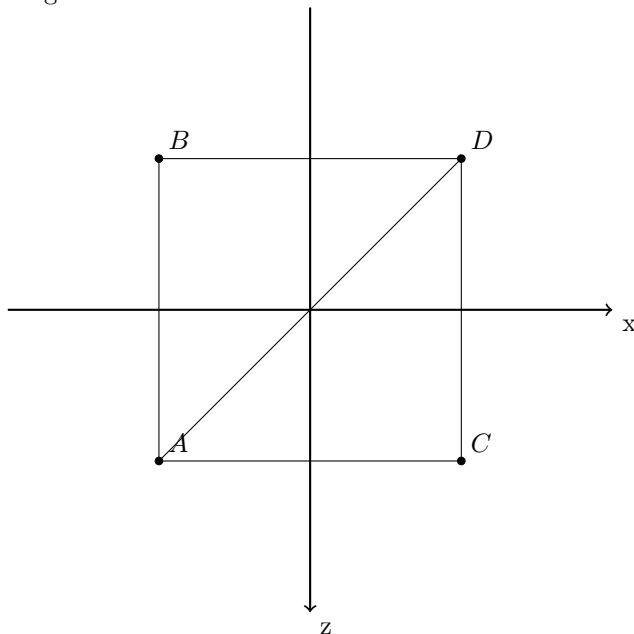


Figura 3: Ilustração do Plano

Na figura acima é possível verificar como foi construído o plano. Dado o lado do plano, este é dividido por 2, obtendo assim as coordenadas dos pontos A, B, C, D.

Dado o lado, o ponto A tem coordenadas $(-lado/2, lado/2)$, o ponto B com coordenadas $(-lado/2, -lado/2)$, o ponto C tem coordenadas $(lado/2, lado/2)$ e o ponto D tem coordenadas $(lado/2, -lado/2)$.

3.2 Box

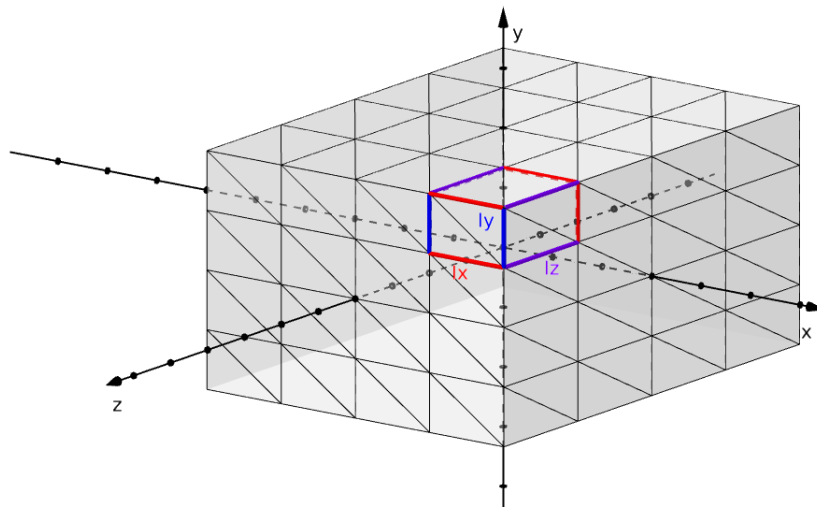


Figura 4: Esquema de uma caixa

Para o desenho da caixa são dadas as dimensões no eixo dos X, dos Y e dos Z, sendo opcionalmente possível indicar também o número de divisões.

Para o desenho da caixa, centra-se a caixa na origem.

Caso não tenha sido indicado o número de divisões, são desenhados dois triângulos em cada um dos lados.

Se tiver sido indicado que a caixa tem n divisões então cada face terá $2 * n^2$ triângulos com os lados diminuídos por um factor de n .

3.3 Sphere

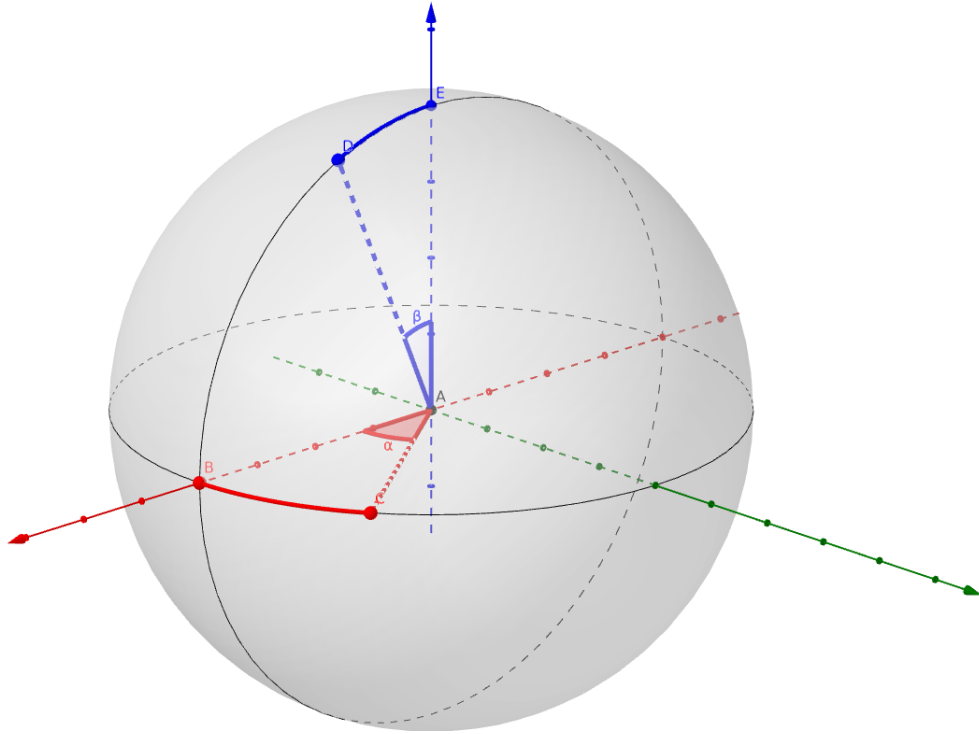


Figura 5: Esquema de uma esfera

$$\alpha = \frac{2\pi}{stacks}$$

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

A esfera é gerada uma camada horizontal de cada vez, a começar pelo topo. No início de cada camada são utilizadas as fórmulas das coordenadas esféricas para calcular a posição de quatro pontos na superfície da esfera de forma a desenhar dois triângulos. Em cada iteração basta rodar mais α graus em relação ao eixo y para calcular as coordenadas de dois novos pontos (não é necessário recalcular as coordenadas y pois estas são constantes na mesma camada horizontal). Estes dois novos pontos mais dois calculados na iteração anterior permitem gerar mais dois triângulos. Estes passos são iterados até ocorrer uma rotação completa em torno do eixo y.

Para gerar os vértices das restantes camadas basta rodar β graus para baixo em relação ao eixo x e repetir o processo.

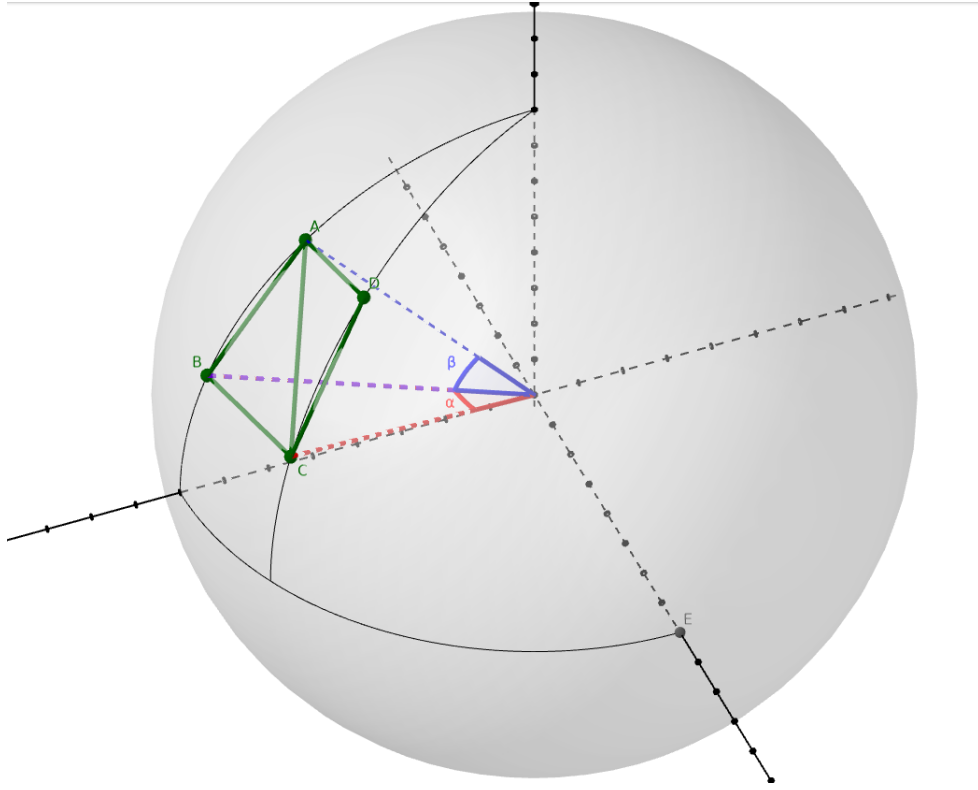


Figura 6: Exemplo de 2 triângulos na superfície da esfera

$$Ax = radius * \cos(\frac{\pi}{2} - \beta * j) * \sin(\alpha * i)$$

$$Ay = radius * \sin(\frac{\pi}{2} - \beta * j)$$

$$Az = radius * \cos(\frac{\pi}{2} - \beta * j) * \cos(\alpha * i)$$

$$Bx = radius * \cos(\frac{\pi}{2} - \beta * (j + 1)) * \sin(\alpha * i)$$

$$By = radius * \sin(\frac{\pi}{2} - \beta * (j + 1))$$

$$Bz = radius * \cos(\frac{\pi}{2} - \beta * (j + 1)) * \cos(\alpha * i)$$

$$Cx = radius * \cos(\frac{\pi}{2} - \beta * (j + 1)) * \sin(\alpha * (i + 1))$$

$$Cy = radius * \sin(\frac{\pi}{2} - \beta * (j + 1))$$

$$Cz = radius * \cos(\frac{\pi}{2} - \beta * (j + 1)) * \cos(\alpha * (i + 1))$$

$$Dx = radius * \cos(\frac{\pi}{2} - \beta * j) * \sin(\alpha * (i + 1))$$

$$Dy = radius * \sin(\frac{\pi}{2} - \beta * j)$$

$$Dz = radius * \cos(\frac{\pi}{2} - \beta * j) * \cos(\alpha * (i + 1))$$

$$0 \leq i < slices$$

$$0 \leq j < stacks$$

3.4 Cone

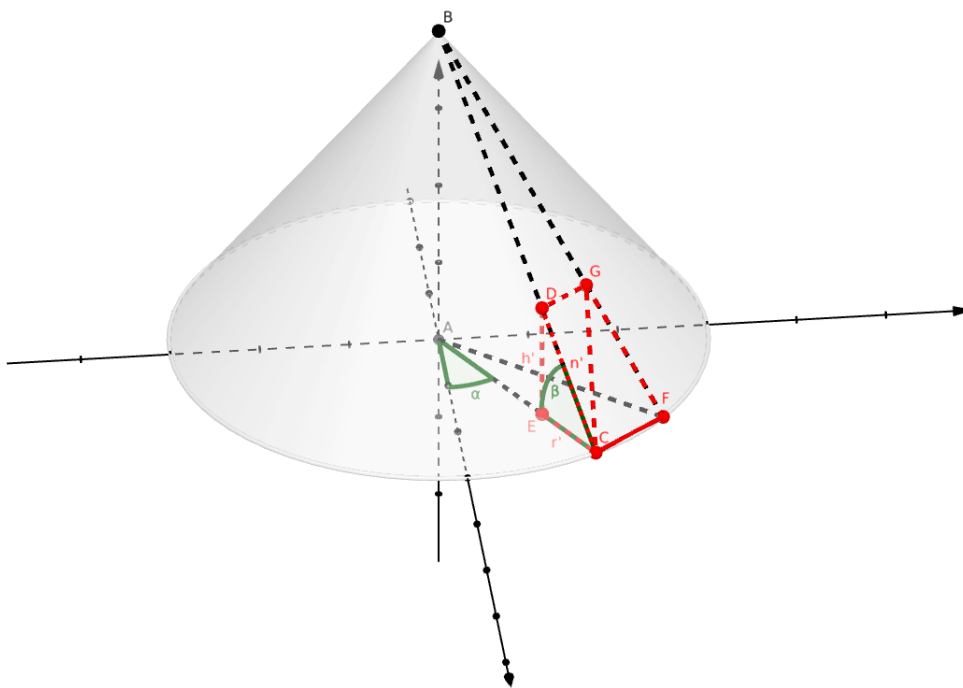


Figura 7: Esquema de um cone

$$\alpha = \frac{2\pi}{stacks}$$

$$\beta = \arctan\left(\frac{height}{radius}\right)$$

$$n = coneSlant = \overline{BC} = \sqrt{height^2 + radius^2}$$

$$n' = segment = \frac{n}{stacks}$$

$$r' = qtdSubRadius = \cos \beta * n'$$

$$h' = qtdAddY = \sin \beta * n'$$

A construção do cone está dividido em duas fases. Na primeira fase é feita a base do cone, na segunda fazemos a superfície lateral. Para construir a superfície lateral, optamos por desenhar por *stacks*, ou seja, desenhemos a primeira *stack* em torno do cone todo e só depois avançamos para a *stack* seguinte. Cada *stack* é composta por (*slices**2) triângulos. Os vértices (Pontos C,D,F,G) utilizados para a construção de cada triângulo são calculados da seguinte maneira:

A variável *yy* é inicializada a 0;
A variável *newRadius* é igual a (*radius-r'*)
A variável *newY* é igual a (*yy+h'*)

$$Cx = radius * \sin(\alpha * i);$$

$$Cy = yy;$$

$$Cz = radius * \cos(\alpha * i);$$

$$Dx = (newRadius) * \sin(\alpha * i);$$

$$Dy = newY;$$

$$Dz = (newRadius) * \cos(\alpha * i);$$

$$Fx = radius * \sin(\alpha * (i + 1));$$

$$Fy = yy;$$

$$Fz = radius * \cos(\alpha * (i + 1));$$

$$Gx = (newRadius) * \sin(\alpha * (i + 1));$$

$$Gy = newY;$$

$$Gz = (newRadius) * \cos(\alpha * (i + 1));$$

$$0 \leq i < slices$$

Num primeiro ciclo fazemos aumentar o valor da variável 'i' de maneira a preencher a primeira *stack*.

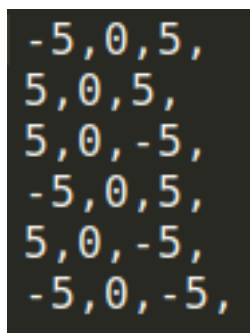
Após terminar o ciclo, o valor das variáveis yy e $radius$ vão ser atualizadas passando a ter o valor de $(yy+h')$ e $(radius-r')$ respetivamente.

Quando passamos para a seguinte *stack*, o valor de $newRadius$ é recalculado, passando a ser $(radius-r')$ (note-se que o $radius$ foi previamente alterado pelo que o novo valor é na realidade $(newRadius-r')$). O mesmo acontece com o $newY$ que passa a ser $yy+h'$ (sendo que o yy também foi previamente alterado).

4 Modelos

Uma vez invocada a aplicação do generator, os vértices são guardados num ficheiro modelo em formato .3d.

Estes ficheiros contém em cada linha as coordenadas dos vértices que compõem a figura geométrica. Na figura 8 temos um exemplo de como as coordenadas são armazenadas no ficheiro modelo. Estas são separadas por vírgulas, facilitando a leitura destas aquando do processamento do ficheiro modelo.



```
-5,0,5,  
5,0,5,  
5,0,-5,  
-5,0,5,  
5,0,-5,  
-5,0,-5,
```

Figura 8: Exemplo de ficheiro modelo contendo os vértices.

5 Exemplos primitivas gráficas

De seguida apresentam-se exemplos das primitivas geográficas realizadas.

Na figura 9 temos um plano com lado 4.

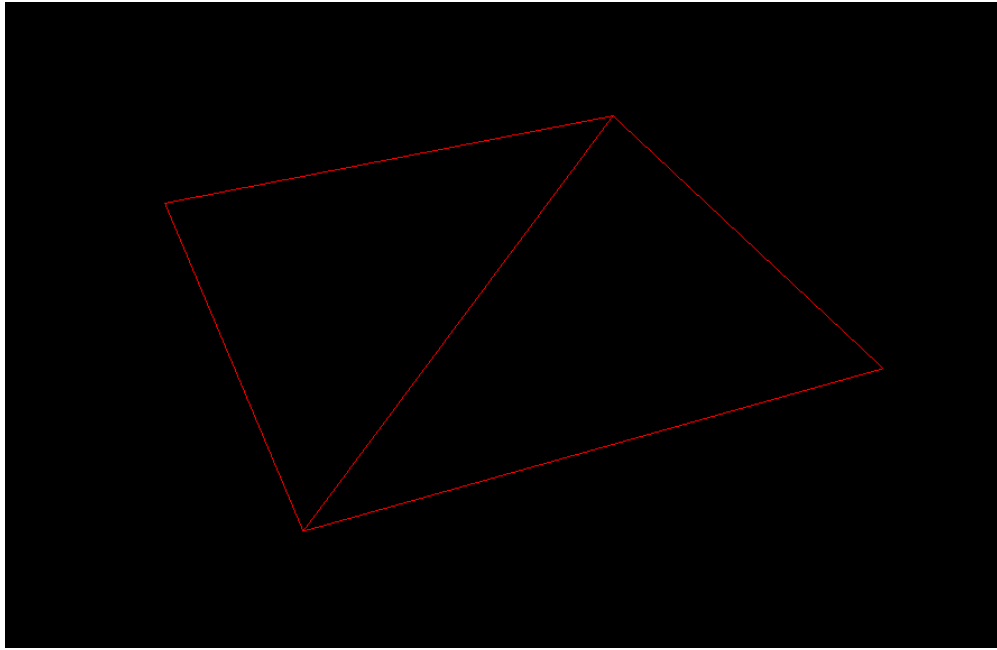


Figura 9: Exemplo de um plano.

Na figura 10 é possível visualizar uma caixa com dimensão igual a 5 no eixo dos X, dos Y e dos Z e com número de divisões igual a 20.

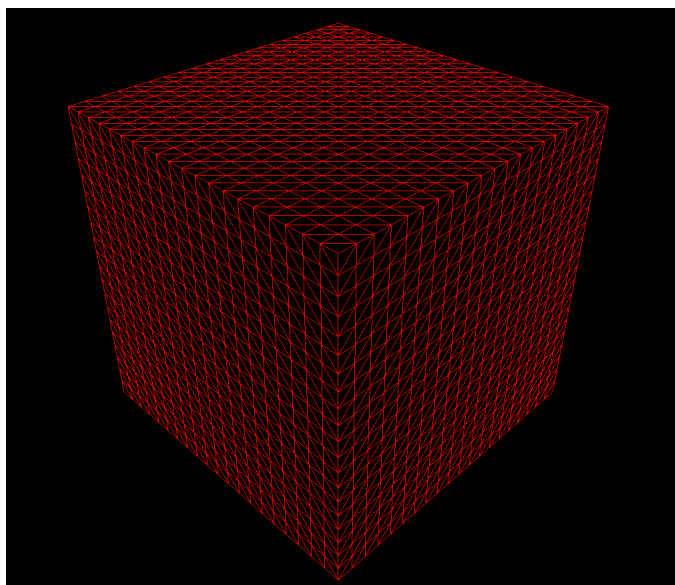


Figura 10: Exemplo de uma caixa.

Tem-se na figura 11 um exemplo de uma esfera com raio 5, com número de *slices* igual a 100 e com 100 *stacks*.

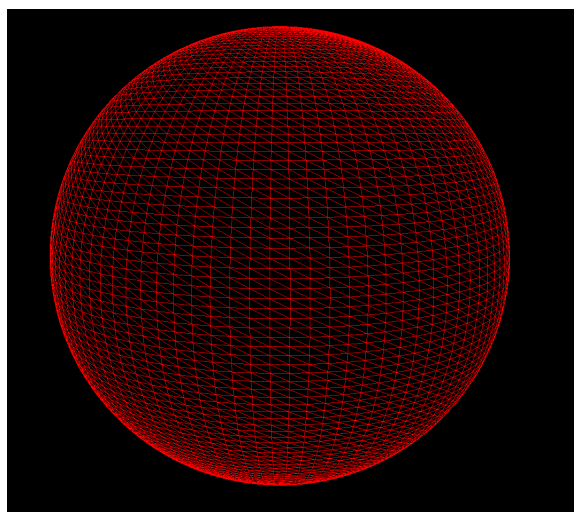


Figura 11: Exemplo de uma esfera.

Na figura 12 pode-se visualizar um exemplo da primitiva cone com raio 5, altura 10, número de *stacks* igual a 20 e número de *slices* igual a 15.

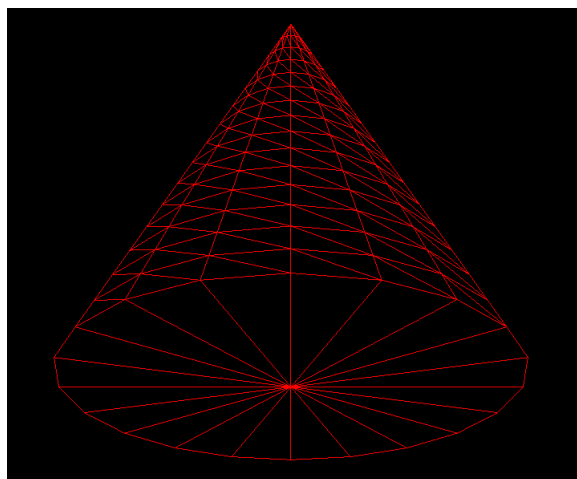


Figura 12: Exemplo de um cone.