Universidade do Minho
Escola de Engenharia

Computação Gráfica

Relatório da Fase 1 – Primitivas Gráficas

Grupo: 32

14 de Março de 2022

LEI – 3º Ano – 2º Semestre

Carlos Filipe Almeida Dias - A93185

José Pedro Martins Magalhães - A93273

Francisco Reis Izquierdo - A93241

Duarte Augusto Rodrigues Lucas - A89526

Conteúdo

Introdução 4

Capítulo 1 5

1. Generator 5

1.1 Plano 6

1.2 Cubo 7

1.3 Cone 7

1.3.1 Base do Cone 7

1.3.2 Superfície lateral do Cone 8

1.4 Esfera 10

Capítulo 2 12

2. Engine 12

Capítulo 3 14

3. Conclusão 14

Lista de Figuras

Figura 1: plane 1 3 6
Figura 2: box 1 3 7
Figura 3: Ângulo das Coordenadas Polares 8
Figura 4: Stacks de um Triângulo 9
Figura 5: cone 1 2 8 5 10
Figura 6: sphere 1 12 10 12

Lista de Tabelas

Tabela 1: Argumentos do Generator 5

Introdução

O presente relatório foi elaborado no âmbito da Unidade Curricular de Computação Gráfica, onde nos foi proposto nesta primeira fase a criação de primitivas gráficas. A implementação das primitivas gráficas irá permitir gerar o ponto de partida no âmbito do resultado final proposto pela equipa docente e que irá ser desenvolvido nas adjacentes fases. No entanto, o foco deste relatório visa a detalhar e a explicar o desenvolvimento do trabalho ao longo desta fase, no qual o mesmo se dividiu em dois componentes chave, o *generator* e o *engine*.

Estes dois componentes de requisito obrigatório foram desenvolvidos com o intuito de trabalharem paralelamente e cuja implementação irá ser detalhada ao longo do presente relatório. O *generator* tem como funcionalidade a criação dos ficheiros relativos aos modelos pretendidos, nomeadamente esfera, caixa, plano e cone, no qual para cada modelo é guardado as informações providenciais para a projeção dos mesmos, isto é, são guardados os vértices.

Posteriormente, a informação presente em cada ficheiro é passada ao outro componente mencionado, o *engine* sendo este o responsável pela modelação e projeção dos modelos, através de um ficheiro em formato XML que define a configuração da projeção mencionada.

Além disso, ao longo desta primeira fase existem outros componentes que convém ressalvar tal como a câmara e o seu posicionamento, teclas de ação condição e também os níveis de zoom, sendo estes componentes importantes na medida em que permitem uma melhor avaliação dos resultados, mas acima de tudo uma boa implementação desta fase.

Capítulo 1

1. Generator

Modo de Uso:

cd generator

mkdir build

cd *build*

cmake ...

sudo make generator

O *Generator* deverá ser capaz de gerar as primitivas, de acordo com os seguintes argumentos fornecidos:

Primitivas	Argumentos
Plano	Comprimento e divisões ao longo de cada eixo
Cubo	Comprimento e divisões ao longo de cada eixo
Cone	Raio da base, Altura, *slices e stacks*
Esfera	Raio, *slices e stacks*

Tabela 1: Argumentos do Generator

Exemplos:

```
C:\>generator sphere 1 10 10 sphere.3d
```

```
C:\>generator box 2 3 box.3d
```

```
C:\>generator cone 1 2 4 3 cone.3d
```

```
C:\>generator plane 1 3 plane.3d
```

O *generator* é aplicação responsável por criar ficheiros 3d com todos os vértices associados para a criação de um modelo. Isto é, após o reconhecimento

do modelo pedido, o programa segue uma série de passos, específicos para cada modelo, gerando todos os vértices necessários para a construção do pedido, sendo guardados numa estrutura de dados.

No fim da configuração de todos os vértices, estes são escritos para o ficheiro de destino, sendo a sua escrita bastante importante devido à disposição de cada vértice no mesmo. A ordem dos vértices no ficheiro revela-se de extrema importância devido à configuração final, isto é, a troca de ordem de vértices compromete toda a configuração desejada. Deste modo, a primeira linha do ficheiro irá enunciar o número de vértices existentes, e as restantes linhas irá conter as coordenadas dos vértices por uma ordem específica. Estas especificações permitem que uma posterior leitura deste ficheiro a configuração do modelo seja a correta.

1.1 Plano

O Plano é constituído por subretângulos dependendo do número de divisões "*div*" ao longo de cada eixo, isto é, é constituído por *div*div* subretângulos.

Para construirmos cada subretângulo é necessária a construção de dois triângulos, sendo que cada um é constituído por três vértices organizados de maneira contrária ao ponteiro do relógio.

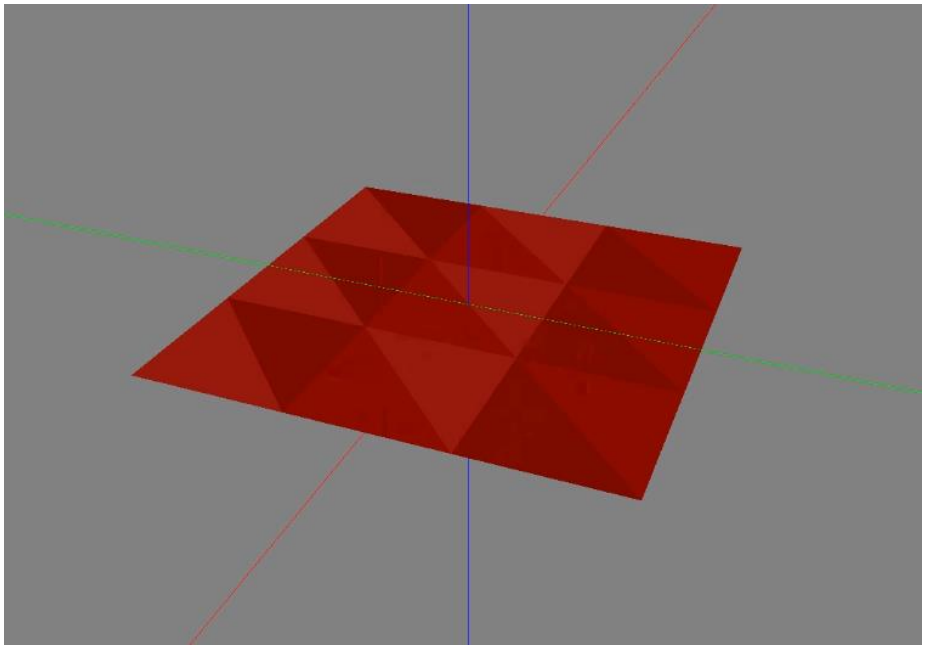

Figura 1: plane 1 3

1.2 Cubo

O Cubo nada mais é do que a junção de seis planos, dois em cada eixo.

Na construção de cada plano é necessário ter em atenção a sua orientação bastando para isso trocar os eixos que são iterados na sua construção e a ordem dos vértices de cada triângulo para que as faces estejam sempre direcionadas para o exterior do cubo.

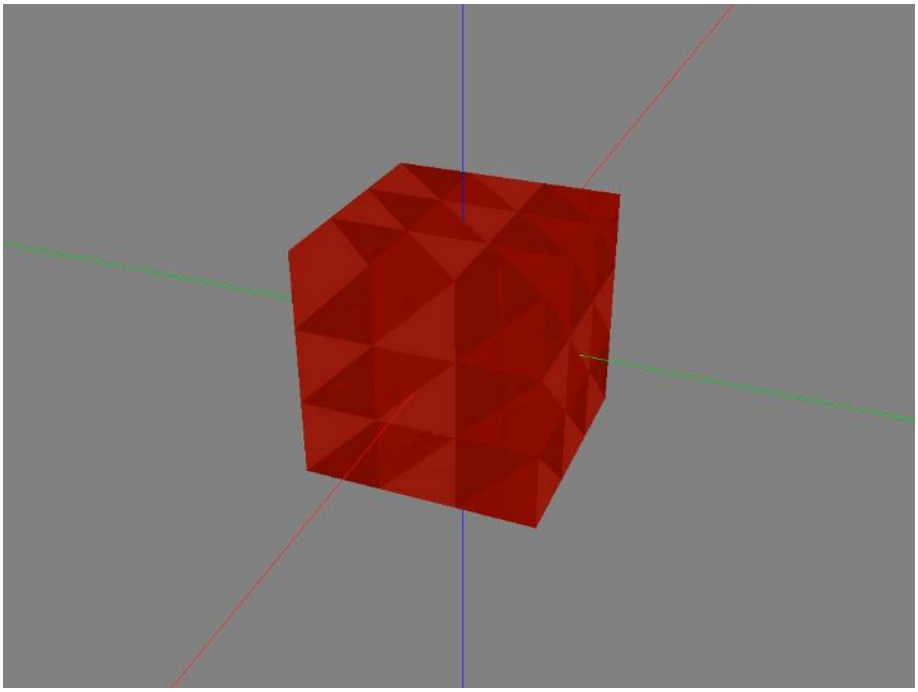

Figura 2: box 1 3

1.3 Cone

Para criarmos um cone, foram utilizados 4 parâmetros: o raio da base, uma altura do cone, o número de *slices* e o número de *stacks* que irá conter. Desta forma, foi necessário dividir a construção do cone em duas partes, que se detalham da seguinte forma.

1.3.1 Base do Cone

Para calcular o número de vértices da base (exceto a origem), é necessário fornecer o número de *slices.* O ângulo entre os 2 vértices consecutivos, relativamente ao eixo do XX, é calculado pela expressão seguinte:

$$\hat{a}ngulo = \frac{2\pi}{slices}$$

Desta forma, com o ângulo entre os vértices e o raio da circunferência, é possível definir as coordenadas de cada vértice, recorrendo as coordenadas polares.

$$x = raio \times \cos(\hat{a}ngulo\ inicial)$$
$$y = raio \times \sin(\hat{a}ngulo\ inicial)$$

Para definir um triângulo da base, utilizamos os seguintes vértices, como é representado na figura 3:

$\hat{A}ngulo\ Inicial = 0$

$Ponto\ O\ (0,0,0)$

$Ponto\ A\ (raio \times \cos(\hat{a}ngulo\ Inicial), 0\ , raio \times \sin(\hat{a}ngulo\ Inicial))$

$Ponto\ B\ (raio \times \cos(\hat{a}ngulo\ Inicial + \hat{a}ngulo), 0\ , raio \times \sin(\hat{a}ngulo\ Inicial + \hat{a}ngulo))$

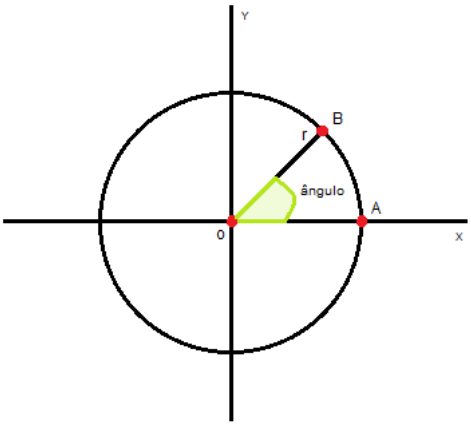

Figura 3: Ângulo das Coordenadas Polares

1.3.2 Superfície lateral do Cone

A abordagem utilizada para a criação da superfície do cone foi gerar uma sequência de triângulos (igual ao número de *slices*). Para cada triângulo, cada coordenada dos vértices no plano XOY são os mesmos que foram definidos para

a base, referido anteriormente. Cada triângulo é limitado pelo número de *stacks*, sendo isto um “corte” horizontal nos triângulos, passando a ser formado por quadriláteros, como na figura 4.

Para a altura do cone, usou-se o centro da figura como o ponto 0. Assim sendo, a altura da figura vai de ***(-h/2)*** até ***(h/2).***

Para calcular o tamanho de cada *stack* (divisões), é utlizado a altura do cone dividindo pelo número de *stacks*, como é dado pela expressão seguintes:

$$Divisão = \frac{altura}{stacks}$$

Ao longo de cada *stack*, o ângulo irá diminui, assim foi necessário calcular os raios seguintes, como também a altura seguinte:

$$rSeguinte = rAtual - \frac{radius}{stacks}$$

$$hSeguintes = hAtual + \frac{height}{stacks}$$

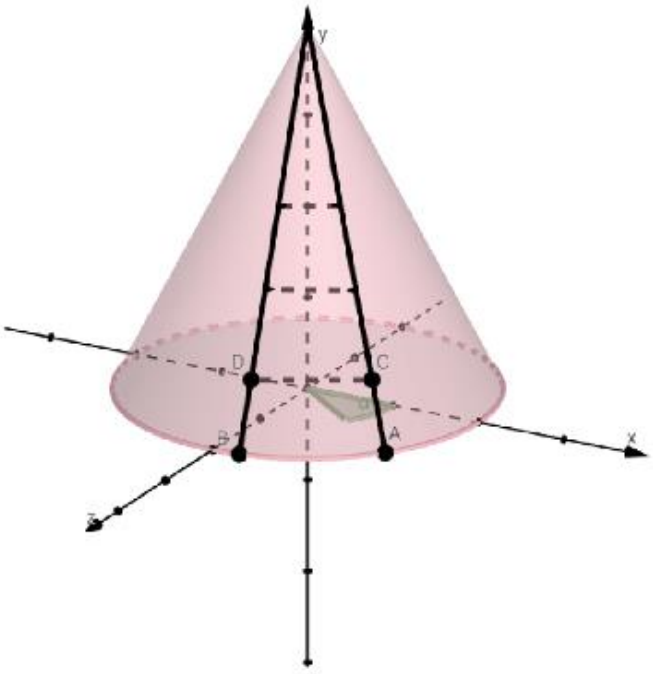

Figura 4: *Stacks* de um Triângulo

Posto isto, foi dividido cada quadrilátero em dois triângulos, sendo representado pelas seguintes funções:

$$Insere\_ponto(fich, rAtual \times \cos(anguloInicial), hAtual, rAtual \times \sin(anguloInicial))$$

$$Insere\_ponto(fich, rSeguinte \times \cos(anguloInicial), hSeguinte, rSeguinte \times \sin(anguloInicial))$$

$$Insere\_ponto(fich, rSeguinte \times \cos(anguloInicial + angulo), hSeguinte, rSeguinte \times \sin(anguloInicial + angulo))$$

$$Insere\_ponto(fich, rAtual \times \cos(anguloInicial), hAtual, rAtual \times \sin(anguloInicia))$$

$$Insere\_ponto(fich, rSeguinte \times \cos(anguloInicial + angulo), hSeguinte, rSeguinte \times \sin(anguloInicial + angulo))$$

$$Insere\_ponto(fich, rAtual \times \cos(anguloInicial + angulo), hAtual, rAtual \times \sin(anguloInicial + angulo))$$

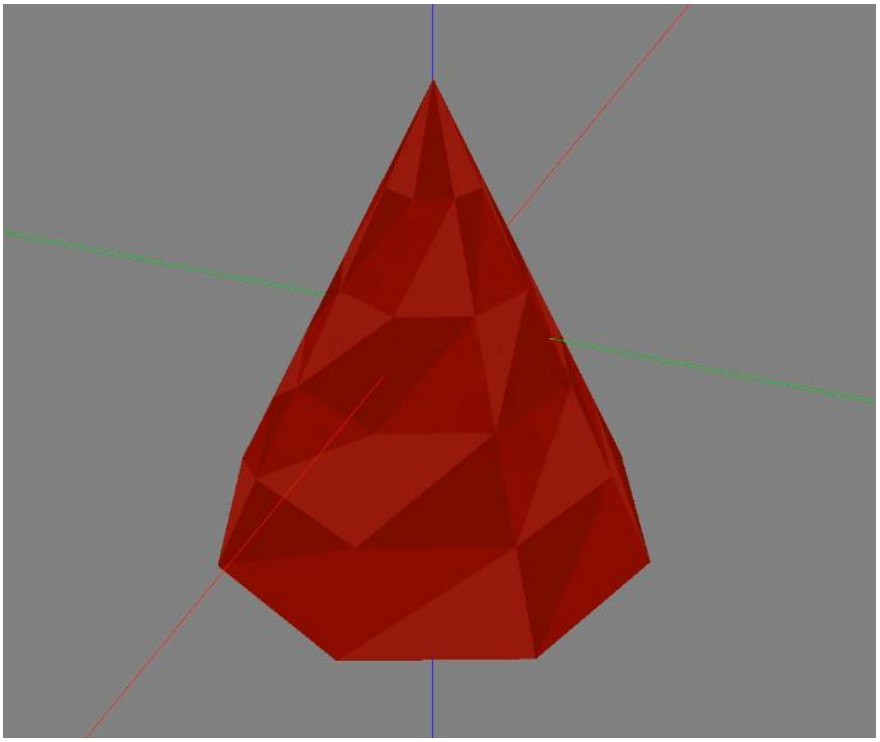

Figura 5: cone 1 2 8 5

1.4 Esfera

A Esfera é uma figura dividida verticalmente em *stacks*, horizontalmente em *slices* e constituída por retângulos. De modo a desenharmos todos os retângulos decidimos fazer dois ciclos: um interior que corre os graus de 0 a 360 (longitude)

que avança num intervalo de (***2 * π / Slices***) graus; e um ciclo exterior que corre os graus de 0 a 180 (latitude) que avança num intervalo de (***π / Stacks***) graus.

Por cada ciclo são construídos os dois triângulos que formam um retângulo. Cada um desses triângulos é construído através das latitudes e longitudes atuais assim como as do próximo ciclo através das seguintes formulas:

Triangulo 1:

Vértice 1:

$$x = raio \times \cos(longitude\_atual) \times \sin(latitude\_atual)$$

$$y = raio \times \sin(longitude\_atual) \times \sin(latitude\_atual)$$

$$z = raio \times \cos(latitude\_atual)$$

Vértice 2:

$$x = raio \times \cos(longitude\_atual) \times sin(latitude\_seguinte)$$

$$y = raio \times \sin(longitude\_atual) \times \sin(latitude\_seguinte)$$

$$z = raio \times \cos(latitude\_seguinte)$$

Vértice 3:

$$x = raio \times \cos(longitude\_seguinte) \times \sin(latitude\_atual)$$

$$y = raio \times \sin(longitude\_seguinte) \times \sin(latitude\_atual)$$

$$z = raio \times \cos(latitude\_atual)$$

Triangulo 2:

Vértice 1:

$$x = raio \times \cos(longitude\_seguinte) \times \sin(latitude\_atual)$$

$$y = raio \times \sin(longitude\_seguinte) \times \sin(latitude\_atual)$$

$$z = raio \times \cos(latitude\_atual)$$

Vértice 2:

$$x = raio \times \cos(longitude\_atual) \times \sin(latitude\_seguinte)$$

$$y = raio \times \sin(longitude\_atual) \times \sin(latitude\_seguinte)$$

$$z = raio \times \cos(latitude\_seguinte)$$

Vértice 3:

$$x = raio \times \cos(longitude\_seguinte) \times \sin(latitude\_seguinte)$$

$$y = raio \times \sin(longitude\_seguinte) \times \sin(latitude\_seguinte)$$

$$z = raio \times \cos(latitude\_seguinte)$$

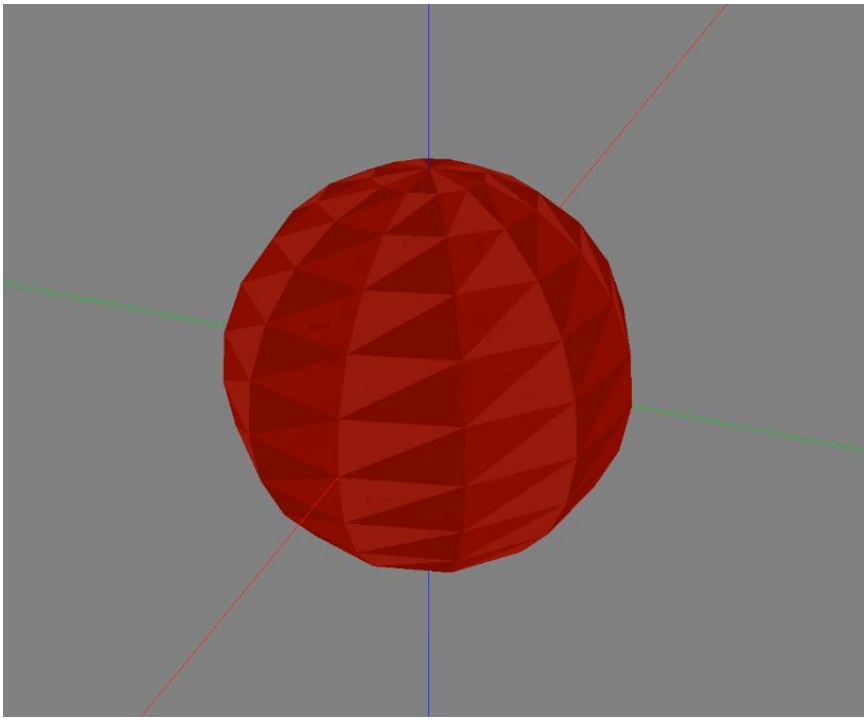

Figura 6: sphere 1 12 10

Capítulo 2

2. Engine

Modo de Uso:

cd engine

mkdir build

cd build

cmake ...

sudo make engine

O *Engine* tem como parâmetro um ficheiro XML que contem uma secção com a posição da câmara, o ponto para onde olhar, o vetor "***up***" e as definições de projeção, e outra secção com os caminhos para os modelos que se pretende importar.

Correr o *engine:*

./engine configure.xml

Desenhar as figuras:

O *Engine* é a aplicação responsável por desenhar os modelos 3d dos caminhos especificados no ficheiro xml. Cada modelo 3d é um ficheiro de texto que contém todos os vértices da figura agrupados três a três (em triângulos).

Quando um ficheiro .3d é lido, os vértices nele contidos são armazenados numa estrutura de dados chamada "*object*" que também contém a posição do objeto assim como a sua cor. Estas estruturas "*object*" são depois agrupadas numa lista global que contém todos os objetos em cena.

Este modo de armazenar os modelos ajuda a simplificar o processo de desenho, uma vez que apenas temos de nos preocupar em desenhar um triângulo de cada vez independentemente da figura a que este pertence. Como nesta primeira fase ainda não nos foi pedido que implementássemos luzes, decidimos colorir cada triângulo com um pequeno desvio da cor original de modo a conseguirmos visualizar cada um deles.

Manipulação da câmara:

De modo a podermos facilmente manipular a câmara com o rato assim como fazer zoom com a *scrollwheel* decidimos guardar as coordenadas da câmara não como (x, y, z) (coordenadas cartesianas) mas como (raio, latitude, longitude), isto permite-nos facilmente orbitar à volta do modelo 3d e aproximarmo-nos.

A função "*gluLookAt*" do *glut* necessita de coordenadas cartesianas para posicionar a câmara, assim utilizámos as seguintes formulas para traduzir entre coordenadas:

x = raio * cos(longitude) * sin(latitude)

y = raio * sin(longitude) * sin(latitude)

z = raio * cos(latitude)

Como o ficheiro XML contém um campo para a posição da câmara, optámos por bloquear a opção de movimento por *default* mas esta pode ser desbloqueada clicando na tecla "f", pode-se também clicar na tecla "r" para voltar à posição especificada no ficheiro XML.

O *Engine* desenha também no canto superior esquerdo um menu de ajuda com as funcionalidades do programa e as teclas que podem ser primidas.

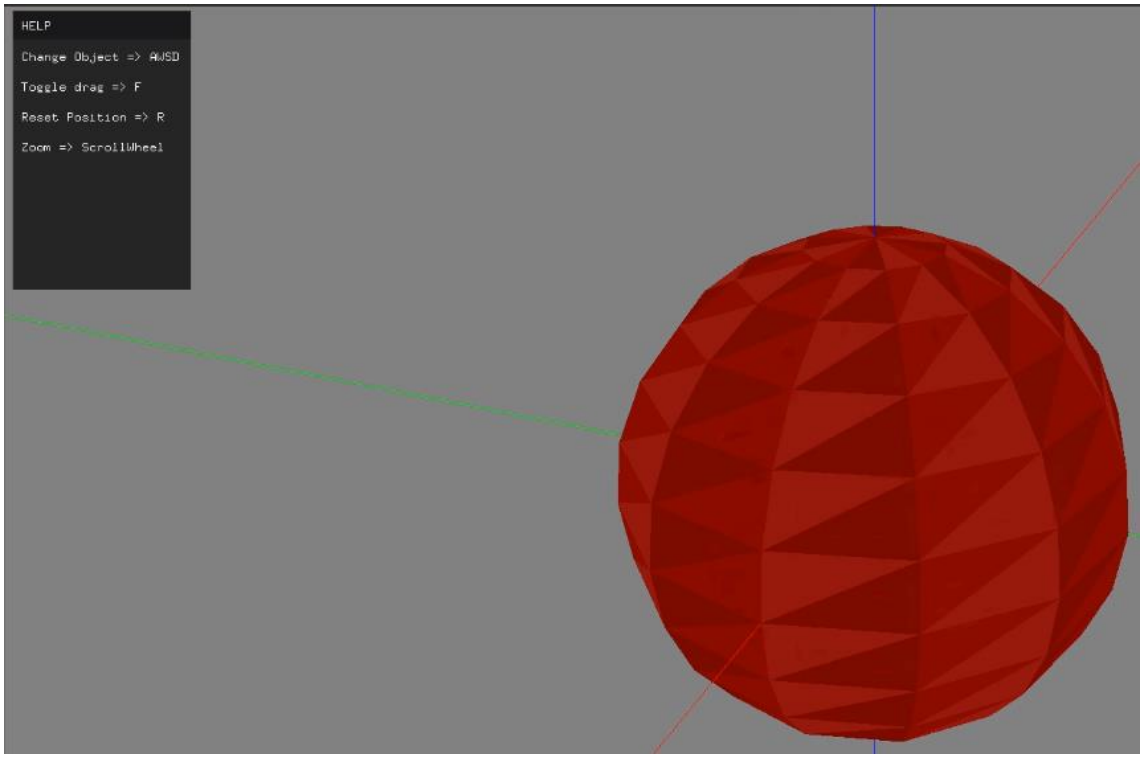

Tabela 2: Menu(Help)

Capítulo 3

3. Conclusão

Em suma esta primeira fase mostrou ser bastante desafiadora na medida em que tivemos de correlacionar os tópicos abordados nas aulas práticas da disciplina e explorar os mesmos de forma a atingir o objetivo proposto. É importante mencionar alguns aspetos relativos que foram abordados e desenvolvidos nesta fase, tal como o generator que se mostrou ser uma peça fulcral no que concerne à geração de toda a informação providencial para a projeção dos modelos propostos. Além deste componente e talvez o mais desafiante de implementar, foi o engine uma vez que tivemos de ter em conta diversos aspetos ao longo da sua implementação, tal como a câmara, teclas ação condição e zoom, uma vez que estes aspetos são relevantes para as futuras fases, uma vez que uma boa prática na implementação das mesmas, consegue facilitar o trabalho do grupo. Além disso, nesta fase procurámos averiguar quais as medidas que teríamos de seguir com o intuito de atingir os objetivos proposto comparando os resultados com os exemplos prestados pela equipa docente.