

Computação Gráfica (3º ano de Ciências da Computação)

**Trabalho Prático - *Fase 1* – Primitivas Gráficas**

Grupo 16

**Feito por:**

* André Filipe Barbosa de Sousa (A87999)
* João Nuno Rodrigues Fernandes (A87971)
* José Carlos Batista Loureiro (A96467)
* Oliver Favero Teixeira (A102506)

**Em:**

* 2 de março de 2025

[1. Introdução 3](#_Toc191805696)

[2. Generator 3](#_Toc191805697)

[2.1. Definição 3](#_Toc191805698)

[2.2. Funcionalidades 3](#_Toc191805699)

[2.2.1. Plano 4](#_Toc191805700)

[2.2.2. Box 4](#_Toc191805701)

[2.2.3. Cone 5](#_Toc191805702)

[2.2.4. Esfera 6](#_Toc191805703)

[3. Engine 7](#_Toc191805704)

[3.1. Leitura do ficheiro de configuração 7](#_Toc191805705)

[3.2. Desenho dos modelos 7](#_Toc191805706)

[3.3. Movimento das Figuras 8](#_Toc191805707)

[4. Resultados 9](#_Toc191805708)

[5. Conclusão 15](#_Toc191805709)

# 1. Introdução

O objetivo proposto nesta primeira fase do trabalho prático é o desenvolvimento de um programa para a criação e desenho de algumas primitivas gráficas, como por exemplo, um plano, uma box, uma esfera e um cone.

Para isso, foram necessárias criar, com base o enunciado proposto pelo docente, duas aplicações o Generator e o Engine.

# 2. Generator

## 2.1. Definição

O Generator através de um conjunto de parâmetros efetua os cálculos necessários para produzir a forma geométricas desejada e assim gerar um ficheiro .3d para, posteriormente, ser utilizado no Engine.

## 2.2. Funcionalidades

O generator tem a capacidade de gerar as formas geométricas seguintes:

• Plano:

Cria um plano paralelo ao plano xOz, centrado na origem do referencial, recebendo como parâmetros o comprimento das arestas, o número de divises e o nome do ficheiro 3d resultante.

• Box:

Cria uma caixa, centrada na origem do referencial, recebendo como parametros o comprimento das arestas, o número de divisões em cada face e o nome do ficheiro 3d resultante.

• Esfera:

Cria uma esfera, centrada na origem do referencial, recebendo como parâmetros o raio, as “slices” (divisões na vertical), as “stacks” (divisões na horizontal) e o nome do ficheiro 3d resultante.

• Cone:

Cria um cone com a base paralela ao plano xOz, centrado na origem do referencial, recebendo como parâmetros o raio da base, a altura, as “slices” (divisões na vertical), as “stacks” (divisões na horizontal) e o nome do ficheiro 3d resultante.

### 2.2.1. Plano

Para a construção de um plano, precisamos da definição de dois parâmetros: length – que, logicamente, define o tamanho do lado, e division, que define a quantidade de divisões.

A função **plane\_generator** começa por definir o ponto inicial (, ). A construção do plano consiste em progredir no eixo x até que uma linha esteja completa, progredindo em passo . Depois que a linha estiver completa, a função move-se ao longo do eixo z, que vai aumentar uma unidade para estabelecer a nova linha, onde o valor de y é sempre 0.

Com isto, os pontos gerados são organizados em triângulos, cada divisão do plano é composta por dois triângulos. Os vértices desses triângulos são armazenados num ficheiro, assim a função termina, criando um ficheiro .3d que contém as coordenadas dos pontos que foram gerados.

### 2.2.2. Box

A função box\_generator, como o próprio nome indica, gera a forma geometrica que representa uma caixa, onde que é exportada para um ficheiro .3d.

Primeiro, calculamos o tamanho de cada divisão (eachSquareLength) tendo em conta o comprimento e o número de divisões. O número total de pontos a serem criados é calculado segundo o número de faces, de divisões por face e em triângulos por cada divisão. Sendo assim, a fórmula passa a ser .

A caixa é gerada a partir do quadrante negativo dos eixos X e Z de modo que fique centrada na origem. Para cada lado da caixa, neste caso: inferior, superior, direita, esquerda, parte frontal e parte traseira, aproveitando a logica de construção de um plano, mas repetindo seis vezes, uma por cada face da caixa os vértices ficam armazenados no ficheiro e quando tudo é completado há uma notificação para confirmar que o ficheiro foi gerado.

### 2.2.3. Cone

A construção de um cone é baseada na transformação de coordenadas polares para cartesianas em três dimensões. Usamos funções trigonométricas mais apropriadas para posicionar os pontos em ângulos corretos. Desta forma calculamos os pontos em três dimensões e é armazenado as suas respetivas extensões num ficheiro .3d.

O cone é dividido em fatias verticais e camadas horizontais. O ângulo da base é dado por angleBase=*.* Desta forma definimos a rotação que deve acontecer para cada fatia ao redor do eixo Y, a altura calculada por respetivada camada é determinada por angleSides=, ou seja, define quanto cada camada se desloca verticalmente.

Para a montagem de um ponto da base, faz-se a definição em x com a projeção em coordenadas polares:

onde r é o raio do cone na camada correspondente, e o angulo θ (theta) é o ângulo da fatia calculado como .

A base do cone é um círculo dividido em vários triângulos, onde cada triângulo é representado por dois pontos na circunferência e pelo centro (0,0,0) da camada. As faces laterais do cone, são compostas por triângulos entre os quatro pontos de duas camadas consecutivas, conforme vamos construindo os triangulos e “subindo” nas “stacks” do cone o raio diminui de acordo com a suguinte proporção:

Nota: “I" representa a stack atual.

Cada ponto das faces laterais é posicionado usando as coordenadas polares, com altura , finalmente, dois triângulos são criados entre cada slice das camadas consecutivas para formar a respetiva superfície lateral.

### 2.2.4. Esfera

A função que gera a esfera é composta por coordenadas esféricas para criar um modelo de esfera em 3D com pontos calculados na sua superfície, assim a esfera é cortada ao longo do eixo Y em camadas (neste caso em altura) e fatias (anéis).

Os ângulos necessários para a criação da esfera são:

angleCirc = – isto define a rotação ao redor do eixo Y.

angleSides = – este determina a mudança na latitude da esfera.

Deste modo, os pontos da esfera são gerados em coordenadas esféricas e depois transformados para coordenadas cartesianas, através da seguinte formula:

onde: r é o raio da esfera, φ (phi) é o ângulo de latitude. θ (theta) é o ângulo de rotação em relação ao eixo Y.

A superfície da esfera é construída a partir de quadriláteros divididos por dois triângulos, os triângulos dividem cada quadrilátero em duas partes, cada quadrilátero é definido por quatro pontos em duas camadas consecutivas que triangulam a esfera por toda a sua superfície

# 

# 3. Engine

A engine é responsável por ler os ficheiros XML. Esta lê os ficheiros”.3d” gerados pelo “generator” e gera as figuras pretendidas.

## 3.1. Leitura do ficheiro de configuração

Como sugerido, utilizou-se o tinyxml2 no auxílio à leitura do XML proveniente do ficheiro de configuração. O código do “engine” possui uma função responsavel pela mesma,

**int readXML(char\* filename)**

Onde lê um ficheiro XML que contém configurações da cena, como dimensões da janela, posição da câmara, alvo da visualização, vetor "up", parâmetros de projeção e referências para os modelos 3D a serem carregados.

## 3.2. Desenho dos modelos

No que toca ao desenho dos modelos, desenvolveu-se as seguintes funções:

* **void Axis()**

Que desenha os eixos (x,y,z) em cores diferentes (vermelho, verde e azul, respetivamente)

* **changeSize(int w, int h)**:

Ajusta a perspetiva da janela quando o tamanho desta é alterado, reconfigurando a matriz de projeção e o viewport.

* **void renderScene(void)**

Função principal de renderização que limpa os buffers, configura a câmara, aplica transformações geométricas e desenha os primitivos 3D carregados dos ficheiros.

## 3.3. Movimento das Figuras

Relativamente ao movimentar a nossa figura na cena criada, foram criadas as seguintes funções:

* **void keysFunc() e void specialKeysFunc()**

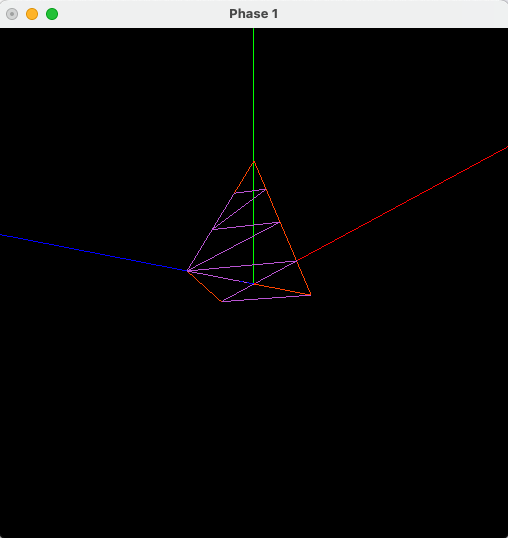
Que nos permite movimentar as nossas figuras pelo eixo do X, Y e Z, e fazer a sua rotação em torno do eixo do Y

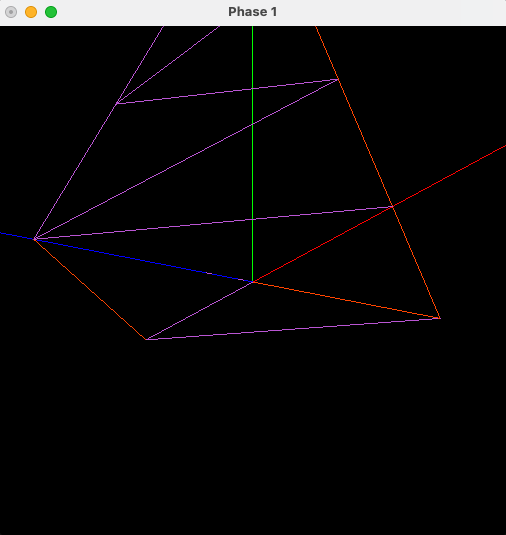
* **void mouseFunc() e mouseMotionFunc(int x, int y):**

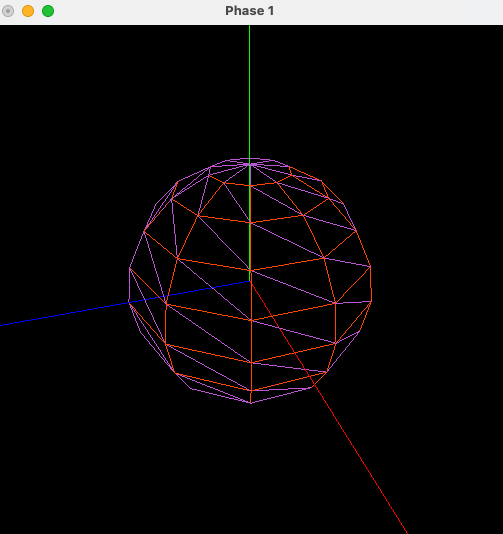
Onde ele permite rodar a figura criada pelo eixo do Y ao clicar no botão esquerdo do rato, ao clicar no botão direito fazemos “zoom in” e “zoom out” e por último, ao clicar no botão do meio do mouse conseguimos segurar a nossa figura livremente pelo plano

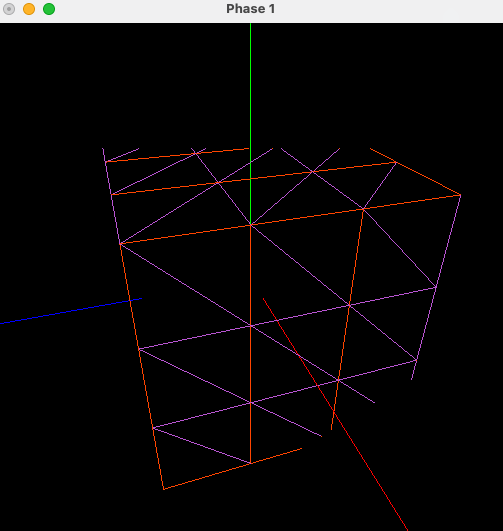
# 4. Resultados

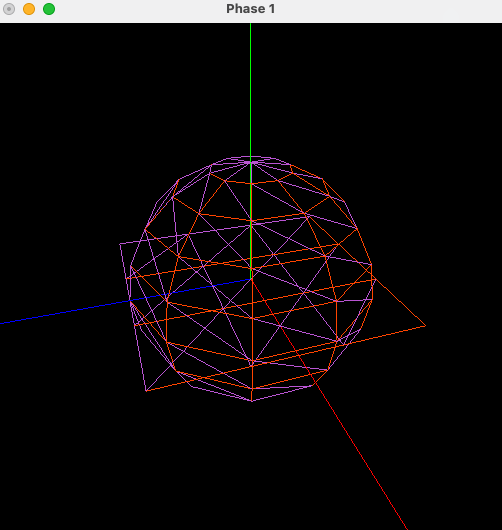
Nesta secção demonstramos os exemplos fornecidos pelos professores e o seu respetivo resultado:

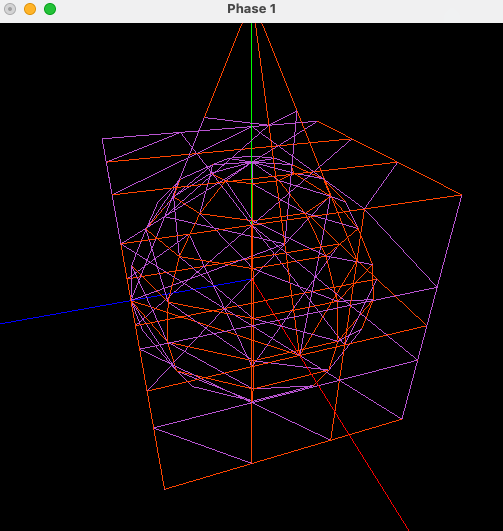
 Figura 1: test\_1\_1.xml

Figura 1: test\_1\_2.xml

Figura 1: test\_1\_3.xml

Figura 1: test\_1\_4.xml

Figura 1: test\_1\_5.xml

Figura 1: test.xml

# 5. Conclusão

Na primeira fase do projeto, conseguimos alcançar os objetivos estabelecidos, apesar de algumas dificuldades na implementação das fórmulas necessárias para a construção das figuras geométricas. Desenvolvemos com sucesso a aplicação geradora, que permite a criação dos modelos tridimensionais básicos (plano, caixa, esfera e cone), bem como o motor gráfico, capaz de interpretar os ficheiros gerados e exibi-los corretamente através da leitura do XML de configuração.

As dificuldades enfrentadas centraram-se principalmente na definição e implementação das equações matemáticas para gerar os vértices das figuras com precisão. No entanto, com uma abordagem metódica e validação dos resultados através de testes, conseguimos superar estes desafios e produzir um sistema funcional e coerente com os requisitos da fase.

A conclusão desta fase proporciona uma base sólida para as fases seguintes, permitindo-nos avançar para a aplicação de transformações geométricas e a construção de cenas hierárquicas de forma estruturada e eficiente.