Universidade do Minho

Unidade Curricular de Computação Gráfica

Ano Letivo 2024/2025

Engenharia Informática

Fase 1 – 'Primitivas Gráficas'

Grupo 4:

Filipa Oliveira da Silva (A104167)

Maria Cleto Rocha (A104441)

Mário Rafael Figueiredo da Silva (A104182)

1. Introdução

Nesta primeira fase do projeto, o objetivo é implementar uma aplicação capaz de gerar e visualizar primitivas gráficas em 3D. O desenvolvimento é dividido em duas aplicações complementares:

- **Generator:** aplicação de linha de comando responsável por gerar arquivos que contêm a informação dos vértices dos modelos. Esta ferramenta recebe, via argumentos, o tipo de primitiva (plane, box, sphere ou cone) e os parâmetros geométricos necessários (por exemplo, dimensões, subdivisões, slices, stacks etc.), e armazena os vértices num arquivo com extensão ".3d". Esses arquivos poderão, posteriormente, ser utilizados para compor cenas no engine.
- **Engine:** aplicação gráfica que, a partir de um arquivo de configuração em XML, lê os parâmetros da câmara, configura a janela e carrega os modelos gerados pelo Generator. Nesta fase, o engine utiliza a API OpenGL com o suporte do FreeGLUT para criar uma janela e desenhar as primitivas (utilizando o modo de desenho imediato). O XML, estruturado conforme um modelo previamente definido, permite definir os parâmetros de visualização e os modelos a serem carregados, garantindo que os dados sejam lidos apenas uma vez no início da execução.

Esta abordagem modular permite uma separação clara entre a geração dos dados (Generator) e a visualização dos mesmos (Engine), facilitando a evolução do projeto nas fases seguintes, onde serão incorporadas transformações geométricas hierárquicas, animações e, posteriormente, técnicas de renderização mais avançadas, como o uso de VBOs e texturização.

2. Arquitetura do Sistema

A arquitetura da fase 1 foi concebida de forma a garantir a independência entre a geração dos modelos e a sua renderização, promovendo a modularidade e a escalabilidade. A seguir, descreve-se a estrutura e os principais componentes:

2.1 Estrutura de Pastas e Componentes

- **Generator:**
  - **CMakeLists.txt:** Arquivo de configuração para o CMake, responsável por definir a compilação dos dois executáveis (Generator e Engine), gerenciar as dependências (OpenGL, FreeGLUT, tinyXML2) e configurar as propriedades do build.
  - **generator.cpp:** Código fonte da aplicação Generator. Responsável por:
    - Interpretar os argumentos passados via linha de comando.
    - Gerar as primitivas gráficas (plane, box, sphere, cone) com base nos parâmetros fornecidos.
    - Calcular e armazenar os vértices num arquivo de saída.
  - **engine.cpp:** Código fonte da aplicação Engine. Responsável por:
    - Ler um arquivo de configuração XML que define os parâmetros de exibição (dimensões da janela, configurações da câmera) e a lista de modelos a carregar.
    - Configurar a cena e inicializar a renderização utilizando OpenGL e FreeGLUT.
    - Carregar os arquivos de modelo gerados pelo Generator e desenhá-los na tela.
  - **Bibliotecas:** A biblioteca tinyXML2 foi utilizada para a leitura e interpretação do ficheiro de configuração em XML, permitindo a extração dos parâmetros necessários para a renderização dos modelos.

2.2 Fluxo de Execução

1. **Gerador de Modelos (Generator):**
    - Ao ser invocado com parâmetros específicos, a aplicação processa a entrada (por exemplo, generator sphere 1 10 10 sphere.3d) e gera um arquivo ".3d" contendo os vértices do modelo.
    - Cada primitiva é gerada com os vértices calculados a partir de fórmulas geométricas (por exemplo, subdivisão de um plano ou geração dos pontos de uma esfera), de modo a centralizar o modelo na origem.
2. **Engine:**
    - No início, a aplicação lê um arquivo XML de configuração (por exemplo, config.xml), que define:
        - **Janela:** Largura e altura.
        - **Câmara:** Posição, ponto de vista (lookAt), vetor up e parâmetros de projeção.
        - **Modelos:** Lista de arquivos ".3d" a serem carregados.
    - Uma vez configurada a cena, o engine inicia o OpenGL (ativando o depth test, face culling etc.) e entra no loop principal de renderização.
    - Durante a renderização, o engine desenha os modelos utilizando o modo de desenho imediato (glBegin(GL_TRIANGLES) ... glEnd()), exibindo as primitivas conforme os dados gerados pelo Generator.

2.3 Tecnologias e Ferramentas Utilizadas

- **C++:** Linguagem de programação utilizada para desenvolver ambas as aplicações.
- **OpenGL e FreeGLUT:** API de renderização e biblioteca de gerenciamento de janelas e eventos.
- **tinyXML2:** Biblioteca utilizada para a leitura e interpretação do arquivo de configuração XML.
- **CMake:** Ferramenta de geração de arquivos de build, que permite compilar o projeto de forma multiplataforma e gerenciar as dependências via vcpkg.

2.4 Vantagens da Arquitetura Adotada

- **Modularidade:** A separação entre Generator e Engine facilita o desenvolvimento e a manutenção, permitindo alterações ou extensões em uma das partes sem afetar a outra.

- **Escalabilidade:** A estrutura modular possibilita a incorporação de novas funcionalidades nas fases subsequentes (como transformações hierárquicas, animações, VBOs e texturização) sem reestruturar a base do sistema.
- **Reusabilidade:** Os arquivos de modelo gerados podem ser reutilizados em diferentes cenas, e a leitura do XML é realizada de forma única, garantindo que os dados sejam carregados eficientemente.

3. Descrição das Imagens e do Gráfico

O conjunto de imagens apresentadas de seguida demonstram diferentes representações gráficas em 3D de primitivas geométricas geradas e renderizadas pelo sistema implementado. As imagens incluem modelos como esferas, cones, cubos e planos, bem como um gráfico de visualização que mostra a câmara e os eixos do sistema de coordenadas.

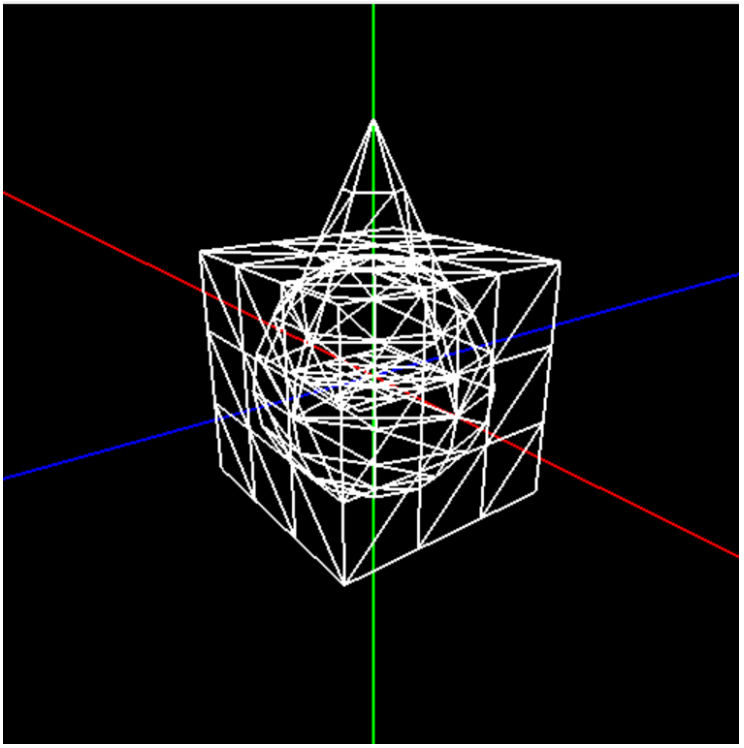

Figura 1 - Quatro primitivas gráficas sobrepostas no eixo

A *Figura 1* representa as 4 figuras geométricas distintas: um cone, uma esfera, um cubo e um plano, todas sobrepostas no espaço, centralizadas e com subdivisões triangulares.

O cone tem sua base no plano XZ e a ponta no eixo Y positivo, enquanto a esfera está posicionada no centro dos eixos. O cubo é subdividido em faces triangulares, e o plano é exibido no eixo XZ. As linhas de contorno de cada figura destacam as suas estruturas geométricas, criando assim uma composição que evidencia a interação entre as primitivas no espaço 3D.

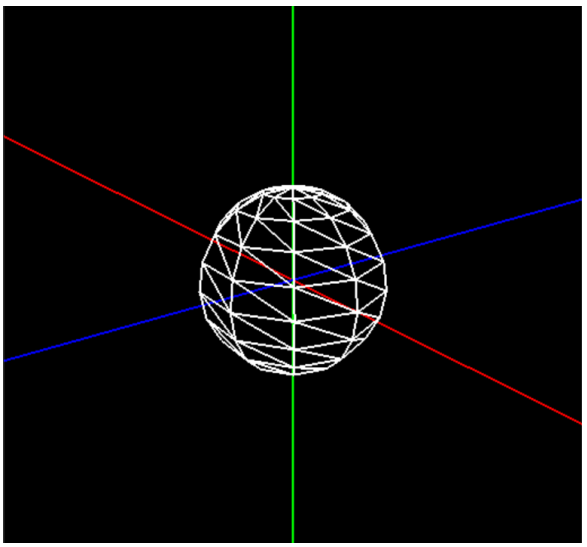

Figura 2 - Primitiva gráfica da esfera no eixo

A *Figura 2* apresenta a esfera gerada numa superfície composta por subdivisões triangulares, com o modelo sendo exibido numa posição que permite visualizar a sua estrutura geométrica em 3 dimensões.

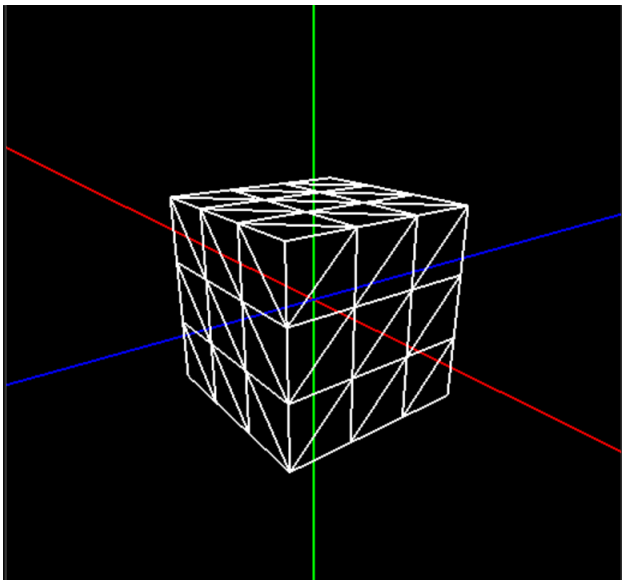

Figura 3 - Primitiva gráfica do cubo no eixo

A *Figura 3* exibe um cubo subdividido em diversas faces triangulares. O modelo está posicionado numa orientação que destaca essas mesmas subdivisões ao longo dos eixos X, Y e Z.

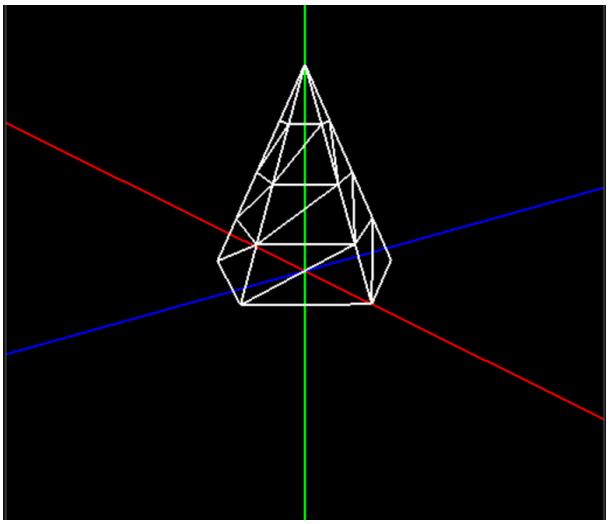

Figura 4 - Primitiva gráfica do cone no eixo

A *Figura 4* é a representação de um cone em 3 dimensões, com uma base na direção XZ e subdivisões visíveis ao longo da sua estrutura, onde as faces triangulares do mesmo conectam a sua base ao vértice superior.

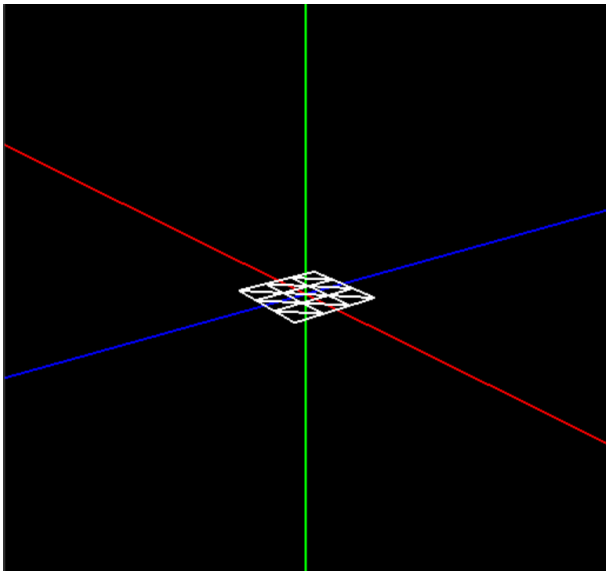

Figura 5 - Primitiva gráfica do plano no eixo

A *Figura 5* mostra um plano subdividido em várias partes, com as divisões claramente visíveis. O plano está orientado no espaço de forma a destacar as suas subdivisões em ambas as direções (X e Z).

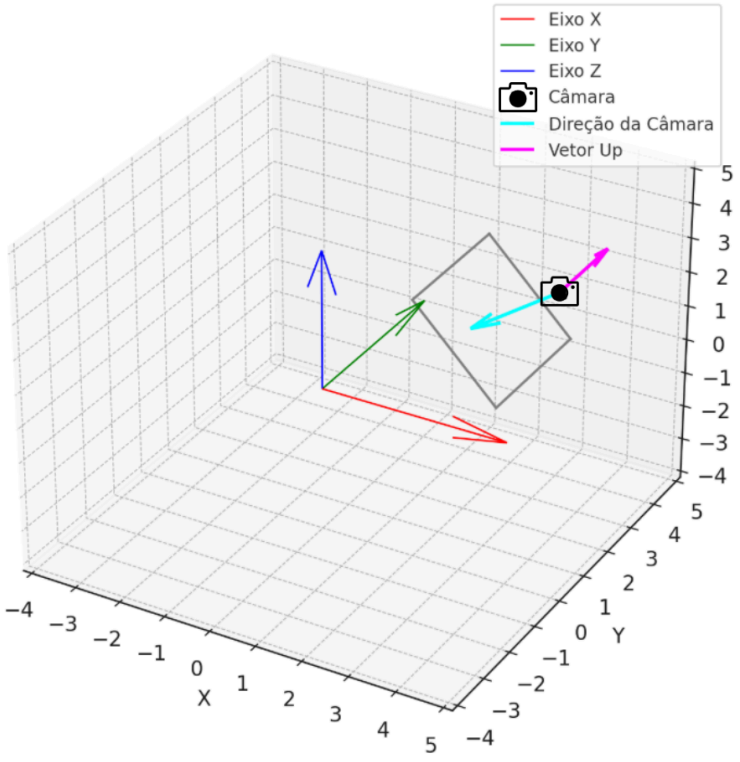

Figura 6 - Representação da visão da câmara em 3 dimensões

Na *Figura 6*, pode-se observar uma representação do gráfico da câmara nos eixos no sistema 3D. A visualização mostra os eixos X, Y e Z com as suas respectivas direções (vermelho, verde e azul). A câmara e o vetor "Up" também são visíveis, destacando a posição da câmara e a direção da visualização da mesma.

4. Conclusão e Trabalhos Futuros

Nesta primeira fase do projeto, foi possível desenvolver uma aplicação capaz de gerar e visualizar primitivas gráficas em 3D, utilizando o OpenGL e o FreeGLUT. A arquitetura modular, com a separação entre o *Generator* e o *Engine*, proporcionou flexibilidade e escalabilidade, permitindo que diferentes tipos de primitivas fossem geradas e visualizadas com facilidade.

Os resultados mostraram que o sistema é capaz de lidar com modelos geométricos simples como cubos, cones, esferas e planos, exibindo-os de maneira eficiente na interface gráfica. A visualização da imagem em 3D, juntamente com a configuração da câmara, permitiu uma análise clara dos modelos gerados.

Para o trabalho futuro, é esperado que o sistema seja expandido para incluir transformações geométricas hierárquicas, animações e suporte para técnicas de renderização mais avançadas, como o uso de VBO's e texturização. Os próximos passos também incluirão a implementação de novas funcionalidades, como o carregamento dinâmico de modelos e a utilização de curvas para movimentação e animação de objetos.

Concluímos e acreditamos então que o sistema esteja de facto pronto para ser integrado com outras fases do projeto, como a manipulação de superfícies cúbicas e a aplicação de curvas Bezier, além de integrar novas funcionalidades como a texturização e a iluminação no motor de renderização.