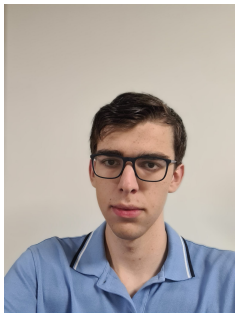
Computação Gráfica

Trabalho Prático

Fase 1 - Primitivas Gráficas

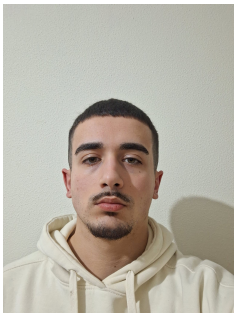
LEI - 2023/2024

Luís Costa
A100749



Rui Pinto
A100649

Ricardo Jesus
A100066



João
A94015

Grupo 36

Universidade do Minho

Índice

1 Introdução

Neste projeto, recebemos a responsabilidade de criar um mecanismo tridimensional baseado em um ambiente gráfico, dividido em quatro fases distintas. Este relatório foca especificamente na primeira fase, na qual fomos encarregados de desenvolver duas aplicações cruciais: um generator e um engine.

O generator é concebido para criar ficheiros que contêm os dados essenciais para o modelo tridimensional, nesta fase, estes ficheiros apenas contém todos os pontos necessários para gerar os pontos através do programa engine. O engine desempenha um papel fundamental ao ler ficheiros XML. Nestes ficheiros XML lidos pelo engine estão contidas informações sobre a camara(posição, projeção, etc) e os modelos 3d a ler, que irão ser utilizados pelo engine. SEle é responsável por interpretar esses arquivos e apresentar os modelos correspondentes para visualização no ambiente gráfico tridimensional. Em suma, o engine serve como a ponte entre as configurações definidas nos ficheiros e a representação visual dos modelos no ambiente tridimensional.

2 Arquitetura

2.1 Estrutura

De forma a facilitar a leitura dos dados por parte do engine, criamos várias estruturas localizadas no ficheiro parser.cpp e as respetivas funções que efetuam o parsing da respetiva informação.

Estas estruturas, como dito em cima, são utilizadas pelo engine para desenhar as figuras pretendidas, tendo a informação de todos os pontos a desenhar, a localização da câmara entre outras informações.

2.1.1 Janela

Esta struct serve apenas para armazenar as dimensões da janela que mostra as figuras desenhadas pelo engine.

2.1.2 Câmara

Para evitar a repetição de vértices, assunto que será tratado mais adiante, esta classe representa um triângulo que inclui as posições no vetor de pontos de cada ponto que o compõe, gerado ao criar a primitiva gráfica.

2.1.3 StringArray

Estrutura que representa um array dinámico de strings, contendo em si um apontador para um array de strings, o tamanho do array(size) e a capacidade(capacity) desse mesmo array.

2.1.4 Modelo

Esta struct armazena dentro de si outras estruturas como as struct CAMARA, JANELA e StringArray com os ficheiros que devem ser utilizados pela engine para desenhar as figuras.

2.2 Ferramentas Utilizadas

2.2.1 TinyXML2

A biblioteca TinyXML2 foi utilizada para efetuar a análise dos ficheiros XML, recolhendo a informação relevante nesta fase e armazenando a mesma nas structs anteriormente mencionadas.

3 Aplicações

3.1 Generator

Como mencionado anteriormente, o gerador deve criar os ficheiros que contêm a informação sobre as primitivas gráficas a desenhar, ou seja, neste caso, o conjunto de vértices que as compõem, de acordo com os parâmetros recebidos.

3.1.1 Parsing dos argumentos

É realizado primeiramente, um parsing dos argumentos passados como input, com o objetivo de verificar se o sólido que se pretende construir é realmente possível.

3.1.2 Geração do modelo

Após a verificação do passo anterior, segue-se a geração do sólido pedido. É realizado o cálculo dos vértices da primitiva gráfica escolhida, fase esta que requer uma especial atenção, para conseguirmos fornecer ao motor um modelo o mais eficiente possível. Para alcançar esse objetivo, exploramos duas abordagens diferentes.

Primeiramente, a aplicação gerava apenas os pontos de todos os triângulos presentes na primitiva a ser construída. O problema desta abordagem foi ser muito pouco eficaz, uma vez que resultava em muitos pontos repetidos devido à presença de um mesmo ponto em vários triângulos. Ora todos estes pontos repetivos acabavam por consumir uma quantidade grande de memória o que não se demonstrava ser o ideal, o que nos levou a pensar noutra abordagem.

Finalmente, optamos por uma abordagem que procurava equilibrar um pouco a abordagem anterior, isto é, permitir a repetição dos vértices mas de uma forma moderada para que ainda assim podessemos garantir a localidade espacial. Assim continuamos a conseguir ter a localidade espacial necessária, e poupamos a memória que com a outra abordagem seria gasta em demasia.

Depois da geração do modelo, com todo o cálculo necessário para os vértices e triângulos, é devolvido o objeto "Modelo".

3.1.3 Estrutura do ficheiro gerado

O generator nesta fase, gera um ficheiro com os pontos necessários para o desenho da figura pretendida. Neste ficheiro, em cada linha, encontra-se representado um ponto, com 3 dígitos que representam as coordenadas de cada ponto nos eixos do x, y e z, respetivamente.

3.2 Engine

O sistema utiliza uma engine que processa um arquivo de configuração em XML (utilizando as funções do parser), usando a biblioteca TinyXML2. Essa engine tem a responsabilidade de ajustar as configurações da câmera, definir a janela de visualização e exibir os resultados gráficos dos modelos 3D gerados pelo "generator".

4 Primitivas Gráficas

Para esta fase, as primitivas geométricas implementadas foram o plano, a caixa, a esfera e o cone.

4.1 Plano

Ao criar um plano, é crucial definir dois parâmetros essenciais: o número de divisões e o comprimento dos lados. O número de divisões (nn) determinará o tamanho total do plano, resultando em n×nn×n quadrados, cada um formado pela junção de dois triângulos.

A disposição dos pontos que delimitam os triângulos segue um padrão específico, começando da esquerda para a direita e de frente para trás. A primeira aresta a ser construída é a mais à esquerda, e essa construção ocorre de trás para frente, ao longo do eixo ZZ. O desenho é gerado a partir de um ponto inicial, com os demais pontos calculados por meio de dois ciclos aninhados que percorrem os eixos XX e ZZ. Cada ciclo possui nn iterações, correspondentes ao número de divisões.

É crucial destacar que o plano é desenhado no espaço XZXZ, mantendo a coordenada YY constante em 0 e centrado na origem. Assim, as coordenadas de cada ponto representam metade dos valores de XX e ZZ. Este processo permite uma representação visual organizada e estruturada do plano no espaço tridimensional.

Assim, depois deste raciocínio, foi-nos possivel encontrar uma fórmula geral para todos os pontos do plano, sendo esta a seguinte:

$$P\left(\text{initialX} - j \times \text{squareLength}, 0, \text{initialZ} - i \times \text{squareLength}\right) \tag{1}$$

4.2 Caixa

Ao criar esta primitiva gráfica, são necessários dois parâmetros essenciais: o comprimento dos lados e o número de divisões, semelhante ao plano previamente mencionado. A representação visual da caixa é concebida como a combinação de seis planos perpendiculares aos eixos x0zx0z, y0zy0z e x0yx0y, representando as seis faces da caixa.

Para cada plano, adotamos a mesma estratégia descrita anteriormente na criação de pontos. Ou seja, cada plano começa com um ponto inicial, e em seguida, dois ciclos aninhados são utilizados para construir os pontos restantes, seguindo o padrão explicado anteriormente. Essa abordagem sistemática é aplicada de maneira consistente para cada um dos seis planos, resultando na formação da caixa tridimensional.

4.3 Esfera

A criação da esfera envolve a consideração de três parâmetros fundamentais: raio, número de camadas e número de fatias. Essa geometria esférica, centrada na origem, é subdividida em n×mn×m retângulos, cada um composto por dois triângulos.

A estratégia para gerar a esfera baseia-se em iterações sobre os parâmetros de camadas e fatias. Cada iteração dos ciclos corresponde a uma camada e uma fatia da esfera, respectivamente. Durante essas iterações, os vértices são calculados e adicionados ao vetor correspondente.

O raio da esfera é crucial para determinar as coordenadas dos pontos na superfície esférica, usando fórmulas trigonométricas baseadas nos ângulos, incrementados de acordo com o número de fatias e camadas. A formação dos triângulos ocorre pela conexão apropriada dos vértices, garantindo uma representação fiel da superfície esférica.

Para otimizar a localização espacial e a eficiência do motor gráfico, durante cada iteração de uma fatia, todos os vértices da fatia são adicionados ao vetor, incluindo os pontos mais alto e mais baixo da esfera. Isso cria uma sobreposição, mas permite uma melhor organização espacial. No cálculo dos triângulos, partimos do triângulo mais baixo, seguido pelos intermédios e, por último, o triângulo mais alto da esfera. Essa abordagem assegura a correta representação visual da esfera tridimensional.

4.4 Cone

A criação do cone envolve quatro parâmetros essenciais: raio, altura, número de camadas e número de fatias. Este cone, centrado na origem, tem sua base no plano XZXZ e é subdividido em n×mn×m retângulos, cada um composto por dois triângulos, onde nn representa as camadas e mm as fatias.

O processo de criação é dividido em duas partes distintas: a base (circunferência) e a superfície lateral. Inicialmente, são gerados os pontos da base, onde triângulos no plano XZXZ convergem para o centro da circunferência (0,0,00,0,0). O número de triângulos depende das fatias verticais, influenciando a forma circular da base.

Em seguida, concentramo-nos na construção dos triângulos que formam a superfície lateral do cone. Esses triângulos são organizados em camadas, de baixo para cima. A medida que avançamos de camada para camada, o raio diminui, e as coordenadas yy aumentam proporcionalmente.

Ao combinar as informações da base e da superfície lateral, obtemos uma representação completa e precisa do cone. Este modelo está pronto para ser incorporado em aplicações gráficas, proporcionando uma visualização tridimensional fiel.

5 Resultados Finais

5.1 Testes

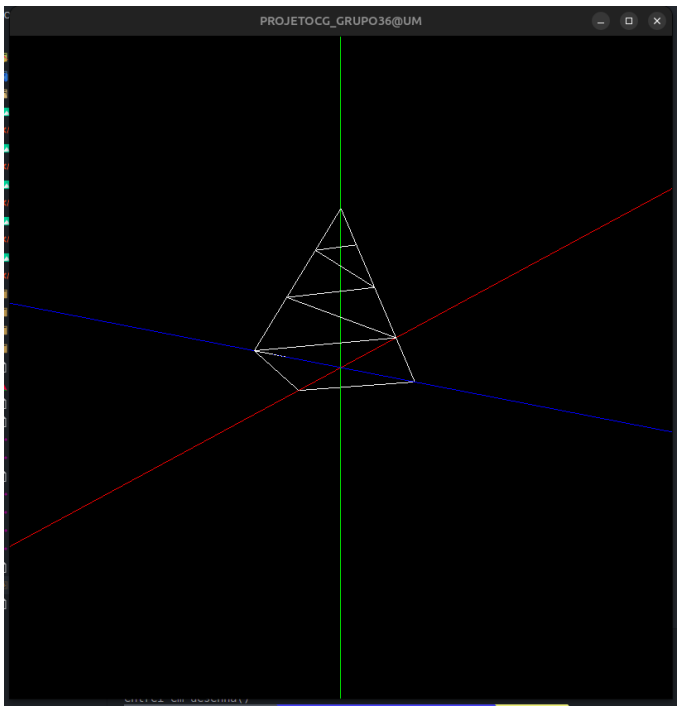


Figura 1: resultado do test_1_1

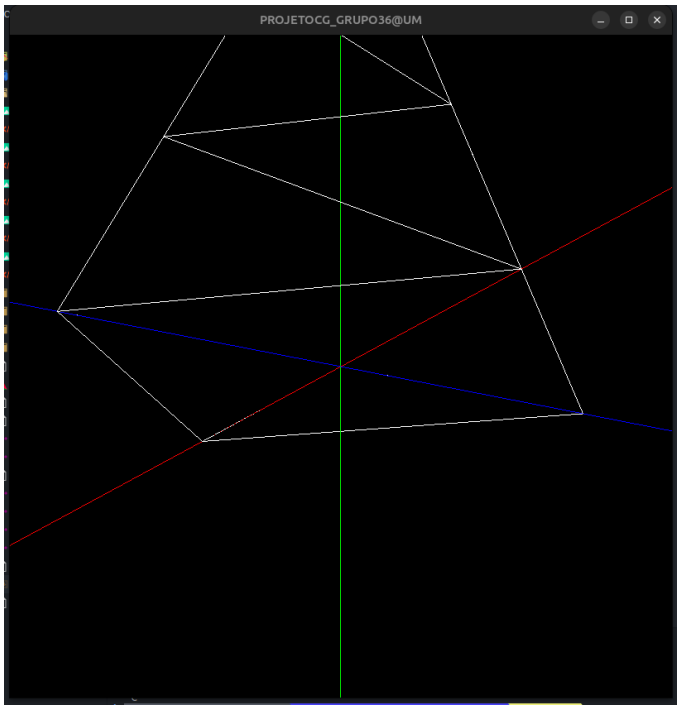


Figura 2: resultado do test_1_2

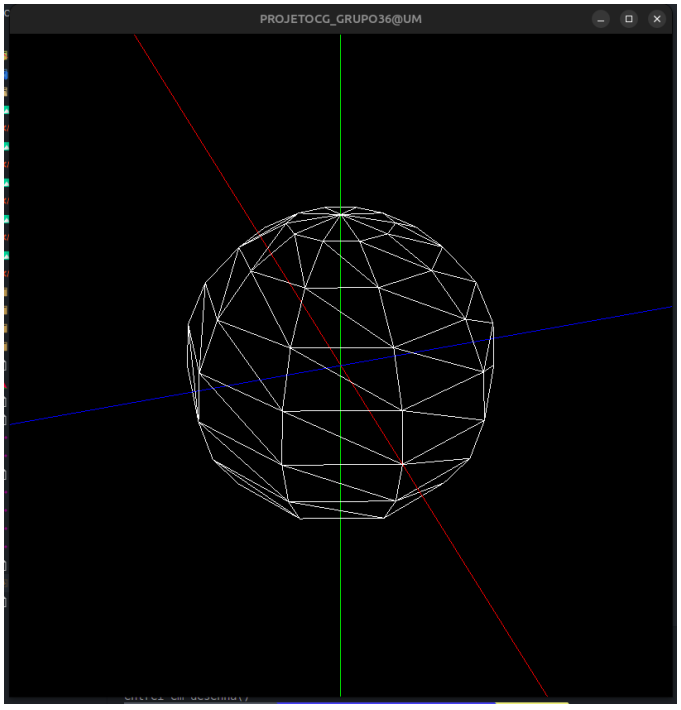


Figura 3: resultado do test_1_3

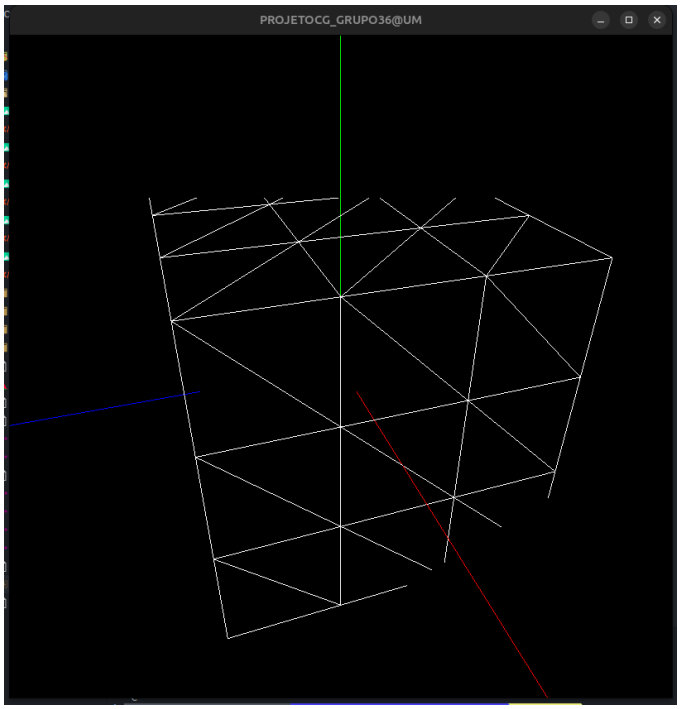


Figura 4: resultado do test_1_4

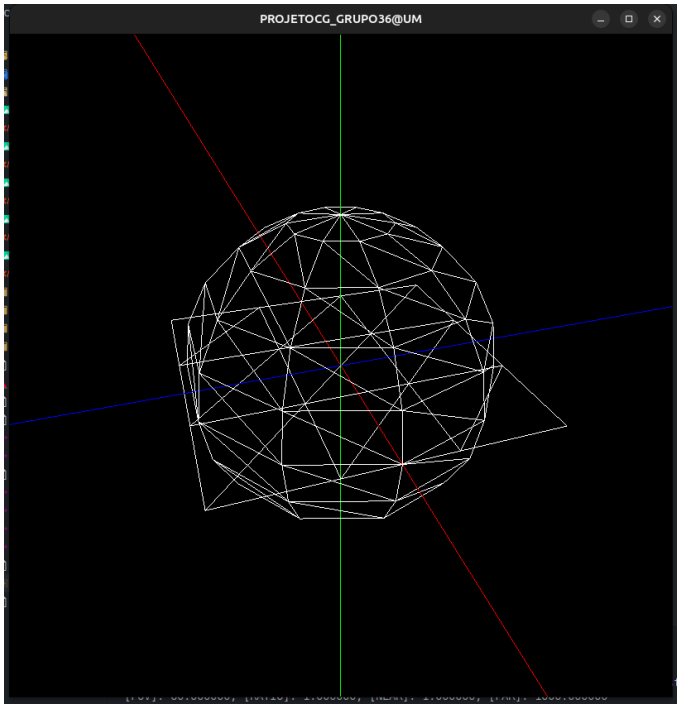


Figura 5: resultado do test_1_5

6 Conclusão

Esta primeira etapa do projeto permitiu consolidar os conhecimentos adquiridos em sala de aula, especialmente no que diz respeito à utilização e domínio de ferramentas de Computação Gráfica, como OpenGL e Glut. Também ganhamos familiaridade com C++, uma nova linguagem para nós, e exploramos conceitos relacionados à leitura de arquivos XML.

No geral, consideramos que atingimos plenamente os objetivos estabelecidos para esta fase.

Na parte do desenho das figuras, realizado na engine, houve uma maior dificuldade em efetuar o desenho de várias figuras de uma vez só, contudo, esta dificuldade foi ultrapassada com sucesso.

Destacamos que, ao longo deste período, mantivemos um foco constante na eficiência e no desempenho das aplicações desenvolvidas. Em resumo, avaliamos o nosso trabalho de forma positiva e estamos bastante satisfeitos com o resultado do mesmo.