Universidade do Minho - Escola de Engenharia
Licenciatura em Engenharia Informática - 3º ano

Computação Gráfica

Fase 1

Grupo 49

Ano Letivo 2022/2023

José Miguel Carvalho Neiva - a92945
Marco António Dias Martins - a90072
Miguel Ângelo Mesquita Rego - a94017
Paula Frederika Janovská Marques - a90088

Índice

Introdução

O presente relatório serve como suporte à Fase 1 do projeto da unidade curricular Computação Gráfica (CG) do ano letivo 2022/2023, que visa alcançar os seguintes objetivos:

- Criar uma aplicação que gere ficheiros com a informação dos modelos (apenas a informação dos vértices).
- Criar o programa que lê o ficheiro de configuração, escrito em XML, e mostra os modelos.

Para atingir estes objetivos compaginamos reuniões e divisões de trabalho convenientes para que todos pudéssemos retirar o máximo deste projeto.

Procedimento

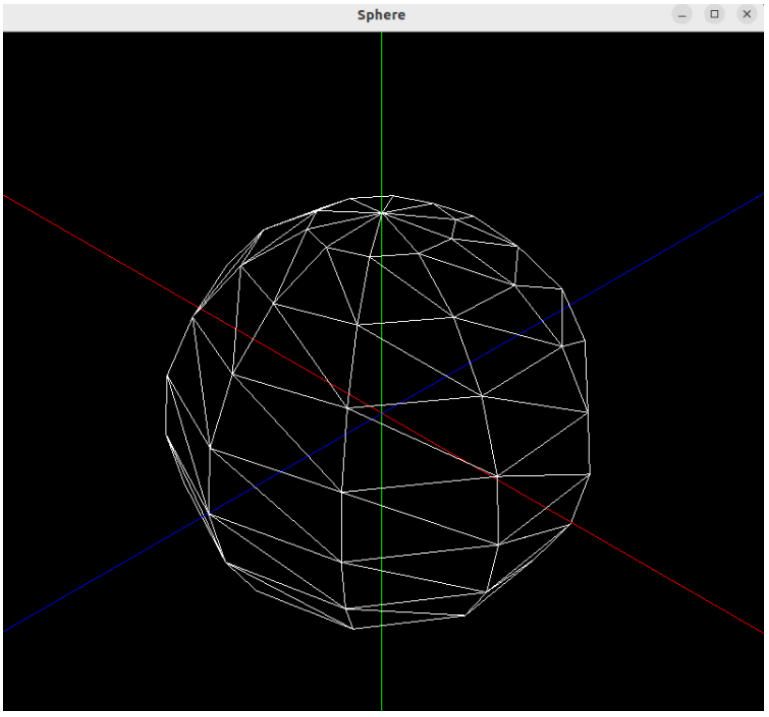
Generator:

Temos 3 tipos de coordenadas: coordenadas cartesianas, coordenadas esféricas e coordenadas cilíndricas. Nas coordenadas cartesianas temos a classe "Cordinates3D" onde vai receber o X, Y e o Z. Nas coordenadas esféricas temos a classe "Spherical" que vai receber o ângulo alfa, beta e radius(raio), onde vai transformar as coordenadas esféricas em coordenadas cartesianas. Nas coordenadas cilíndricas temos a classe " "

A função "**void DrawAxis()**" – desenha os eixos XYZ

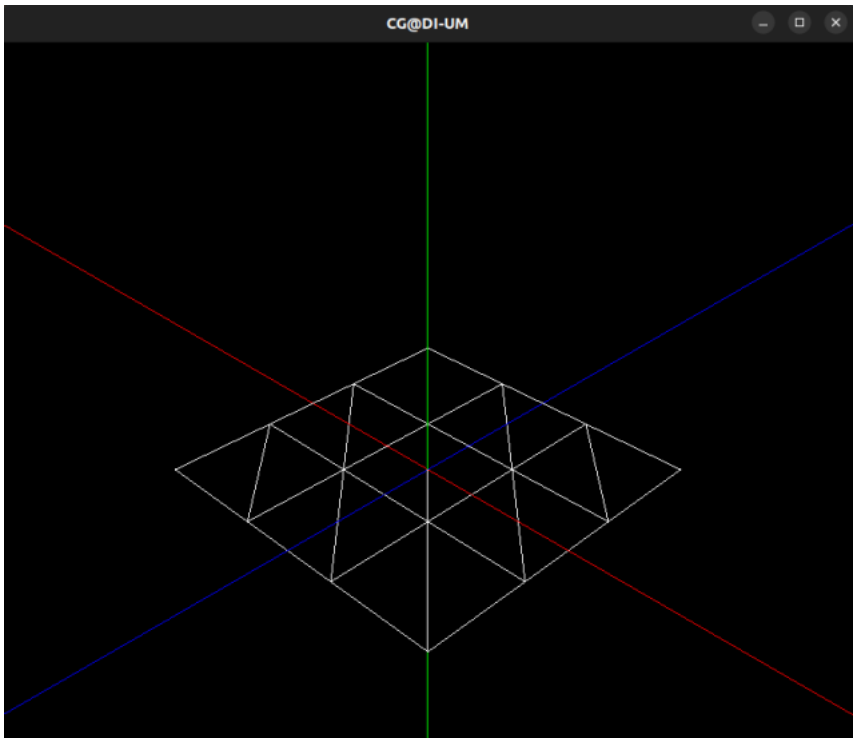
- **ESFERA**

  Para criar a esfera usamos a função "**std::list<Coordinates3D> drawSphere(float radius, int slices, int stacks)** " recebendo como parâmetros o seu raio (radius), o número de subdivisões verticais da esfera (slices), o número de subdivisões horizontais da esfera (stacks)



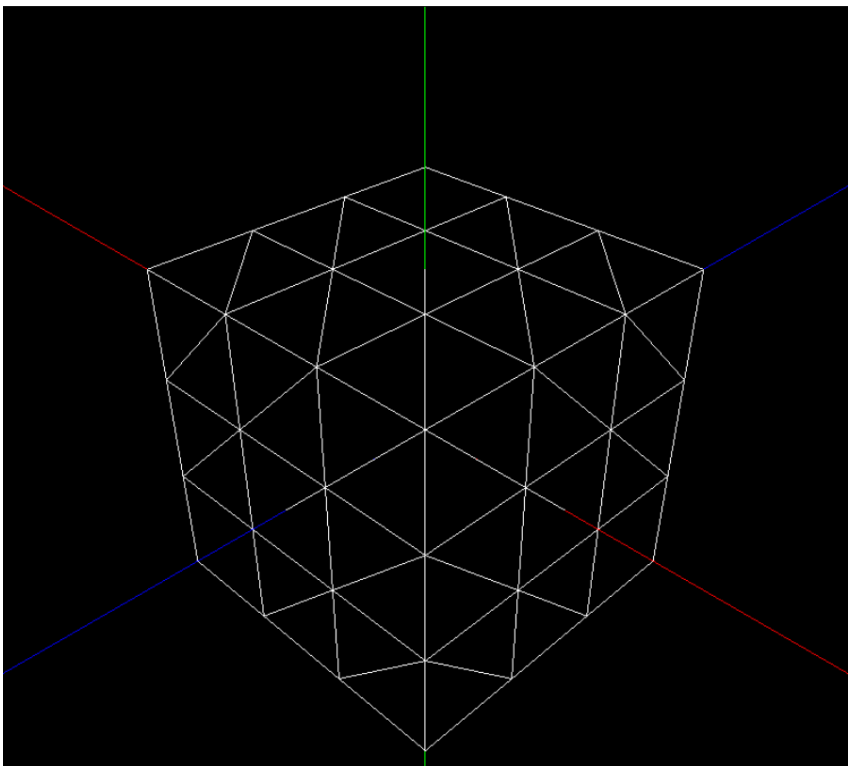
- # PLANO

Para criar o plano usamos a função "**std::list<Coordinates3D> drawPlane(float length, float divisions)**" onde este vai receber os parâmetros do tamanho e das divisões que vão ser efetuadas.



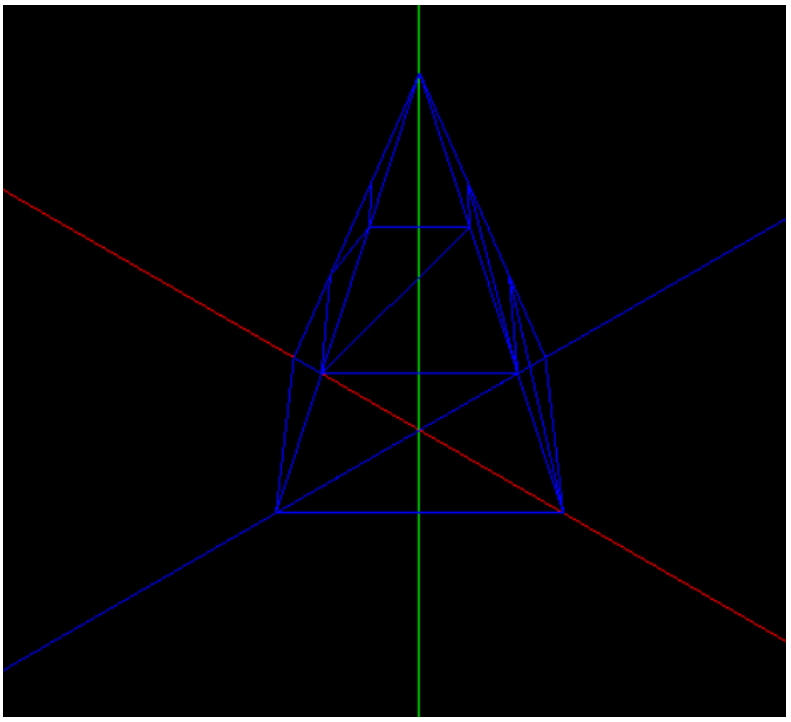
- # CUBO

Para construirmos o cubo usamos a função "**std::list<Coordinates3D> drawCube(float length, float division)"** onde este recebe **"std::list<Coordinates3D> result, temp;"** em que o "result" são os varios planos "x0z", "z0x", "y0x", "x0y", "z0y" e "y0z", e o "temp" é usado para colocar os pontos de cada plano no result.



- **CONE**

Para construirmos o cone usamos a função **"void drawCone(float radius, float height, int slices, int stacks)"** e que recebe como parâmetros o radius(raio da base), height(a altura), o número de subdivisões verticais da esfera (slices) e número de subdivisões horizontais da esfera (stacks)



ENGINE:

Nesta fase inicialmente necessitamos de ler um ficheiro XML que contém: a configuração da câmara e o nome dos ficheiros .3d. Procede depois com a leitura destes ficheiros .3d, guardando o seu conteúdo em memória. Finalmente, desenha os vértices lidos no ecrã, tendo em conta a configuração da câmara.

Decidimos usar o pugiXML em alternativa ao tinyXML pois achamos que era mais fácil de entender e mais acessível tendo em conta esta primeira fase.

Conclusão

Apesar das dificuldades, consideramos que esta fase foi bem conseguida. Após a triagem dos nossos resultados pelos testes fornecidos verificamos que passavam nos mesmos.

As dificuldades que encontramos centraram-se principalmente no parser onde nos sentimos um pouco perdidos mas rapidamente conseguimos entender o que era proposto e chegar a um resultado favorável.

Conseguimos desafiar os nossos conhecimentos já existentes e por sua vez compreender melhor os novos conceitos que exploramos.