Universidade do Minho
Escola de Engenharia

Computação Gráfica
Fase 1-Grupo 11
2022/2023

Afonso Xavier Cardoso Marques a94940
Ana Filipa da Cunha Rebelo a90234
Tomás Cardoso Francisco a93193
Simão Paulo da Gama Castel-Branco e Brito a89482

Março 2023




Índice

List of Figures

1 Introdução

O projeto proposto na Unidade Curricular de Computação Gráfica consiste em desenvolver um motor gráfico 3D e fornecer exemplos práticos que demonstrem a eficácia na sua utilização, utilizando a API OpenGL e recorrendo à biblioteca GLUT. O projeto está dividido em quatro fases, tendo cada uma delas os seus próprios objetivos e desafios e a obrigatoriedade de implementar certas funcionalidades.

O presente documento visa apresentar e descrever a primeira fase de desenvolvimento do projeto, cujo objetivo foi o de criar um gerador de vértices de quatro primitivas gráficas diferentes: plano(plane), caixa(box), esfera(sphere) e cone(cone), tendo em consideração diferentes parâmetros tais como a largura, altura, profundidade e número de divisões(slices). A sua representação gráfica assenta no desenho de vários triângulos, sendo que para tal é necessário determinar os vários vértices que constituem esses mesmos triângulos.

Esta fase requer duas aplicações - um gerador e um motor - e por isso foi dividida em duas partes, cada uma correspondente a cada uma das aplicações:

1. GERADOR: Cria ficheiros ".3d" com a informação dos vértices do modelo, recebendo como parâmetros o tipo de primitiva gráfica, outros parâmetros requeridos para a criação do modelo e o ficheiro destino onde os vértices vão ser guardados.
2. MOTOR: Recebe e lê o ficheiro de configuração XML, que inclui as definições da câmera e a indicação de quais os ficheiros pré-gerados que devem ser carregados.

2 Estrutura do projeto

Para uma melhor organização do projeto decidimos estruturá-lo recorrendo a diretorias separadas tais como engine,generator,includes e rapidxml-1.13. Apresentamos em seguida,um diagrama de packages para uma melhor representação da estrutura do projeto.

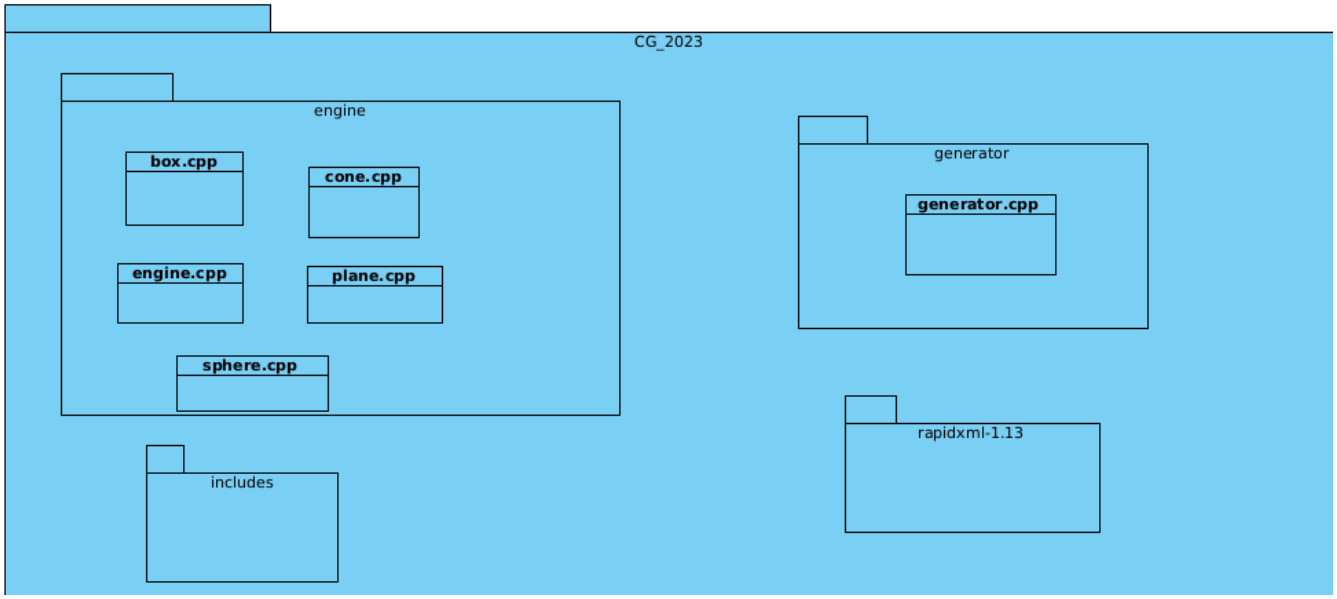


Figure 1: Diagrama de Packages

O package Generator contém o código referente à geração de todos os vértices necessários para a representação das figuras implementadas. No package Engine encontra-se o código referente à leitura dos ficheiros .xml e .3d.

Encontra-se também nesta diretoria o desenho de todas as figuras geradas e todos os comandos disponibilizados durante a visualização das primitivas. De modo a facilitar a leitura dos ficheiros .xml este package utiliza o package rapidxml-1.13.

3 Generator

A primeira aplicação desenvolvida tem como principio básico e fundamental a geração de ficheiros com a extensão "3d" que contêm, linha por linha, os vértices necessários para a criação de uma figura a três dimensões. Quando invocados pelo Engine, é feito um parse a cada linha do ficheiro, criando no processo um vértice que vai ser desenhado.

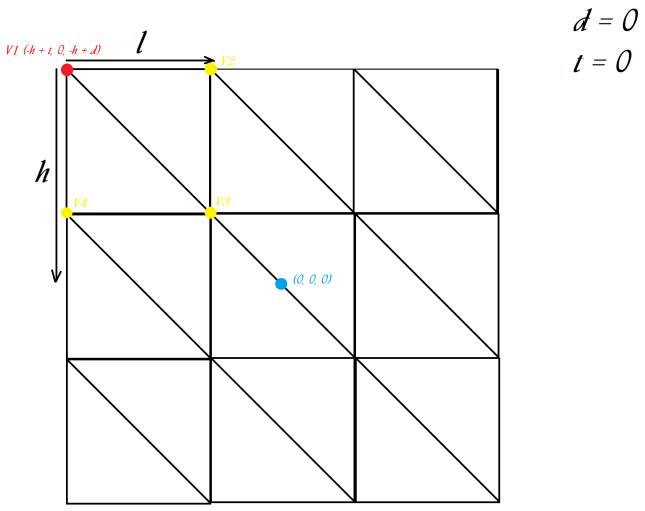
A lógica por trás do gerador consiste em aplicar algoritmos que calculam, com base nos parâmetros passados à aplicação, as coordenadas dos vértices que compõem os triângulos que vão construir a figura pretendida. O processo foi feito de forma a que cada vértice seja gerado segundo a ordem que é mais adequada à figura, fazendo com que seja mais acessivel a geração de triângulos através do Glut.

De seguida encontram-se explicados os algoritmos aplicados para a geração de vértices de cada figura:

3.1 Plano

A geração do plano tem como parâmetros um número de virgula flutuante correspondente ao comprimento do lado, um inteiro que representa as divisões que o lado irá ter e o nome do ficheiro que vai ser gerado (geralmente é "plane.3d").

A seguinte imagem permite visualizar o estado inicial do algortimo:



Podemos ver que V1 é o primeiro vértice a ser criado. Para cada quadrado (dois triângulos juntos) a ordem de geração é a seguinte:

- V1-V2-V3

- V1-V4-V3

Esta lógica é aplicada a cada um dos quadrados pequenos que compõem o plano. O avanço é feito da esquerda para a direita. Por se tratar de um algoritmo de poucas linhas, decidimos expô-lo a seguir em pseudo-código.

```
(...)

    float h = length / 2;
    float l = length / divisions;
    float d = 0;
    float t = 0;

    for(d = 0; d < divisions * l; d += l) {

        for(t = 0; t < divisions * l; t += l) {

            gerar_v1(-h + t, 0.0f, -h + d);
            gerar_v2(-h + l + t, 0.0f, -h + d);
            gerar_v3(-h + l + t, 0.0f, -h + l + d);

            gerar_v1(-h + t, 0.0f, -h + d);
            gerar_v4(-h + t, 0.0f, -h + l + d);
            gerar_v3(-h + l + t, 0.0f, -h + l + d);

        }
    }

(...)
```

Tal como dito anteriormente, os vértices já estão a ser gerados numa ordem que permite uma melhor transição para o Glut.

3.2 Caixa

A geração da caixa segue uma lógica semelhante à do plano onde, no fundo, estamos a gerar seis planos, cada um correspondente a cada face da caixa. Em cada face foi apenas necessário trocar os valores das coordenadas de sitio.

Tem os mesmos parâmetros que o plano, isto é, um número de virgula flutuante correspondente ao comprimento do lado, um inteiro que representa as divisões que o lado irá ter e o nome do ficheiro que vai ser gerado (por convenção será "box.3d").

3.3 Cone

O algortimo usado para a geração dos vértices do cone consiste em gerar dois triângulos para cada *slice*. A função generateCone recebe como parâmetros o raio da base (*radius*), a altura do cone (*heigth*), o número de *slices*, o número de *stacks* e o nome do ficheiro onde vão ser guardadas as coordenadas dos pontos.

Pensamos na base da figura como uma piza dividida em fatias, cada uma de ângulo igual. Para obter esse valor de ângulo dividimos 360^{o} pelo número de *slices* passado à função. De seguida o algoritmo entra num loop que permite criar dois triângulos, um "deitado" e outro "de pé", ou seja um na base e outro na superficie lateral do cone. Todos os triângulos da base têm o ponto (0, 0, 0) em comum e os da superficie lateral tem em comum o ponto (0, *heigth*, 0).

Em cada iteração do loop, multiplicamos o valor do ângulo pelo raio e pelo número da *slice* atual, permitindo assim que os triângulos formem superfícies contínuas.

O grupo está consciente que a geração do cone não está a ser feita dentro dos padrões que são pedidos. Referimos-nos ao facto do parâmetro *stacks* não ser usado para a criação dos vértices. Assim, o grupo compromete-se a fazer a devida correção ao longo do desenvolvimento da segunda fase.

4 Engine

O motor é responsável por, após a criação dos modelos, ler um ficheiro XML - passado como argumento - que contém informações sobre o ambiente em que os modelos 3D serão renderizados, bem como os nomes dos ficheiros dos modelos a serem desenhados. Para isto, recorremos à biblioteca **rapidxml.hpp**.

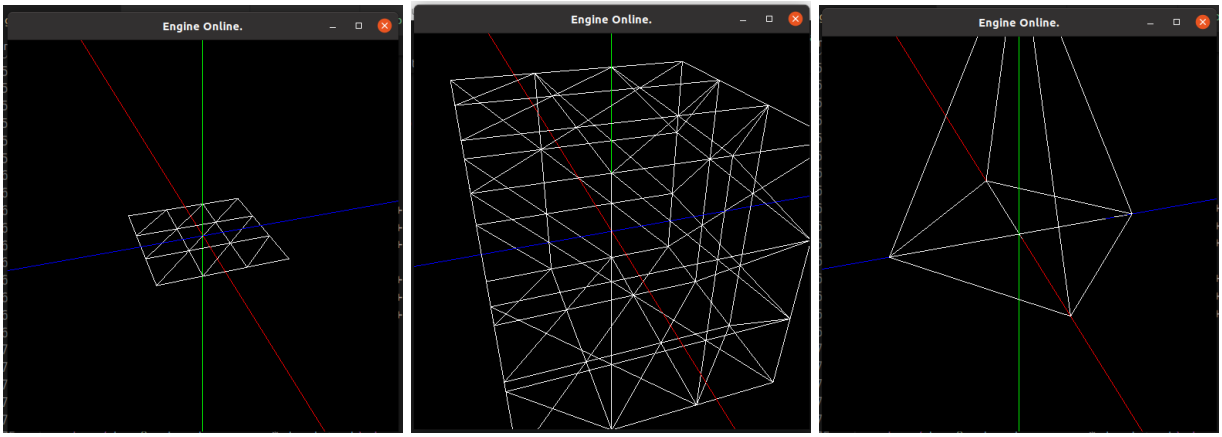
4.1 Leitura do ficheiro XML

A função **set_up_from_files** é responsável por ler as informações do ficheiro XML e retornar um vetor com os nomes dos ficheiros dos modelos. São extraídas - e armazenadas em variáveis globais - as informações necessárias relativas à janela, à câmara e aos nomes dos ficheiros dos modelos, as quais irão ser utilizadas na inicialização da janela e na renderização. No final, o vetor files3D é preenchido com os nomes dos ficheiros dos modelos a serem desenhados.

4.2 Leitura dos modelos e desenho das primitivas

A função **fromFileToShape** é responsável por ler cada ficheiro .3d especificado no vetor files3D e desenhar as primitivas necessárias (triângulos) utilizando a função glBegin(GL_TRIANGLES). Nesta função, o vector que contém todos os nomes dos ficheiros é iterado sobre, cada um dos nomes sendo utilizado para abrir o ficheiro com o nome correspondente, onde depois são extraídas coordenadas por linhas do ficheiro, de forma a obter novos pontos que serão depois usados na função glBegin de forma a desenhar as figuras.

Após compilação e execução do engine conseguimos gerar as seguintes figuras:



(a) Plane (b) Box (c) Cone

Figure 2: Desenhos das primitivas

5 Conclusão

No decorrer desta fase foi possível aplicar conhecimentos adquiridos nas aulas da Unidade Curricular de Computação Gráfica, com a utilização do OpenGL e Glut, trabalhando com a linguagem de programação C++ e explorando novos domínios como a leitura de ficheiros XML.

Dada por concluída a primeira fase do projeto, consideramos importante realizar uma análise crítica, e ainda, realçar os pontos positivos e negativos. Durante o desenvolvimento do projeto surgiram algumas dificuldades, o que tornou impossível para o grupo, de modo a cumprir os prazos, implementar a totalidade das funcionalidades ficando a geração da esfera por fazer. Além disso, a geração do cone não está a ser feita dentro dos padrões pedidos tal como referimos anteriormente. Assim, na próxima fase começaremos por concluir a totalidade das funcionalidades e melhorar a geração do cone.

Por outro lado, apesar das dificuldades encontradas o grupo conseguiu implementar a maioria das funcionalidades propostas o que consideramos ser um ponto positivo do trabalho.