



Universidade do Minho
Escola de Engenharia

Computação Gráfica

Fase 1- Primitivas gráficas
Grupo 7

Braga, Março de 2023

Bernardo Amado Pereira da Costa, A95052
Eduardo Miguel Pacheco Silva, A95345
José Carlos Gonçalves Braz, A96168

Índice

| | |
|---|----------|
| 1. Introdução | 3 |
| 2. Modelos tridimensionais..... | 4 |
| 2.1. Plano..... | 4 |
| 2.2. Caixa..... | 5 |
| 2.3. Esfera..... | 5 |
| 2.4. Cone..... | 6 |
| 3. Generator | 7 |
| 3.1. Estrutura de ficheiro..... | 7 |
| 4. Engine..... | 7 |
| 5. Gerar modelos e demais comandos..... | 8 |
| 6. Conclusão e análise de resultados | 9 |

1. Introdução

Nesta primeira fase foi-nos pedido para criar primitivas gráficas. Assim sendo, e já com os conhecimentos básicos de programação no âmbito da modelação que temos aprendido nas aulas, pudemos começar a planear e executar o início deste projeto. No que concerne ao planeamento e organização, nesta fase inicial, entre os demais ficheiros programados, temos três essenciais: *geometricShapes*, *generator* e *engine*. O primeiro tem como objetivo definir as classes das primitivas e a própria estrutura de pontos. O *generator* cria objetos geométricos tridimensionais, como plano, caixa, cone e esfera, e grava as suas informações num arquivo 3D. O *engine* é importantíssimo, sendo responsável pela leitura de ficheiros de configuração e renderização das formas geométricas.

2. Modelos tridimensionais

2.1. Plano

Para a realização do plano e das demais figuras trabalhamos à base de triângulos como visto nas aulas. Primeiramente, definimos os construtores de classe para o mesmo e em seguida criamos vetores de vértices através da função *push_back* que nos permitem o desenho desta forma geométrica.

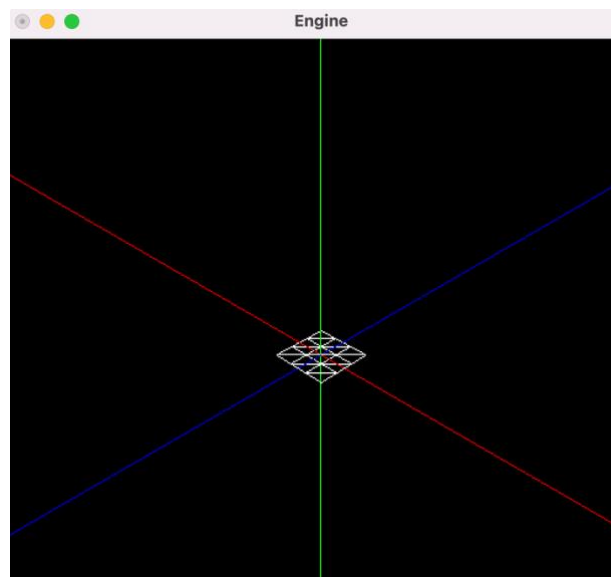


Figura 1-Exemplo de Plano "plane.3d"

2.2. Caixa

Após a criação de construtores para o que é a nossa *box* trabalhamos com as vistas de baixo e de cima, frontal e traseira e direita e esquerda, fazendo 6 planos e trabalhando com os 3 diferentes eixos.

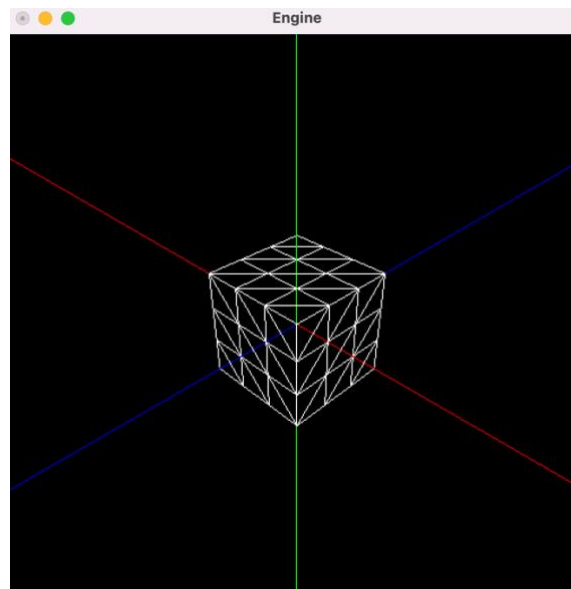


Figura 2- Exemplo da caixa "box.3d"

2.3. Esfera

Para a construção da esfera decidimos utilizar coordenadas esféricas devido á facilidade de encontrar um vértice através dos ângulos alfa e beta (e das operações trigonométricas associadas aos mesmos) e dum raio. Sabemos que considerando como ponto de referência o centro da esfera, o raio será constante e os ângulos alfa e beta para o ponto seguinte serão fáceis de encontrar (contando com a colaboração de stacks e slices).

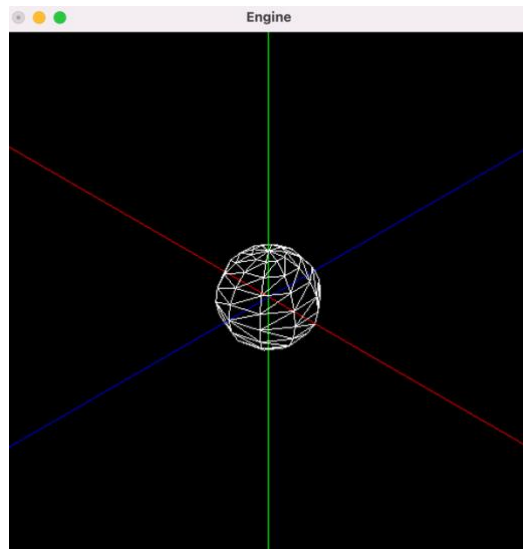


Figura 3- Exemplo de esfera "sphere.3d"

2.4. Cone

A classe possui um método que calcula os pontos que formam o cone. Esse método utiliza as fórmulas matemáticas (onde vr e vh são respectivamente a variação do raio e da altura) para calcular os pontos de cada slice e stack do cone e armazena os pontos num vetor de objetos.

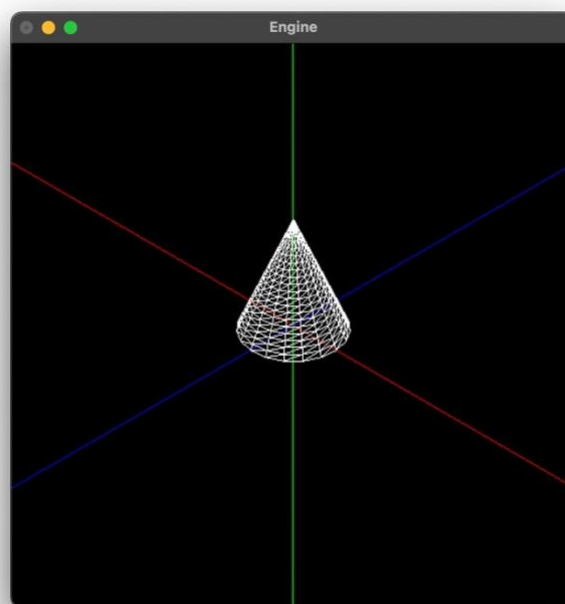


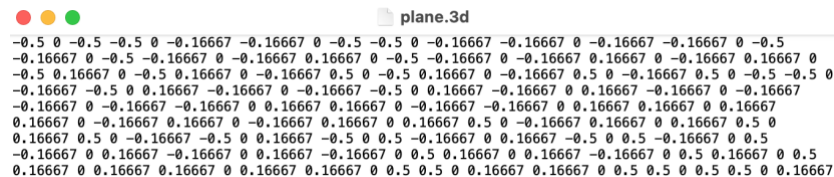
Figura 4- Exemplo de cone "cone.3d"

3. Generator

Na função *main*, *GeometricShape* é inicializado com base nos argumentos de linha de comando. Dependendo da forma geométrica especificada, uma nova instância da classe correspondente é criada usando os argumentos fornecidos.

3.1. Estrutura de ficheiro

Não existe propriamente separação entre pontos que constituem um vértice, sendo que por exemplo na linha de comandos aparecerão num esquema de linha-a-linha. A identificação do que é um ponto é bastante fácil. Abaixo anexamos um exemplo do ficheiro *plane.3d*. Este contém os pontos pertencentes ao plano.



```
-0.5 0 -0.5 -0.5 0 -0.16667 -0.16667 0 -0.5 -0.5 0 -0.16667 -0.16667 0 -0.16667 -0.16667 0 -0.5
-0.16667 0 -0.5 -0.16667 0 -0.16667 0.16667 0 -0.5 -0.16667 0 -0.16667 0.16667 0 -0.16667 0.16667 0
-0.5 0.16667 0 -0.5 0.16667 0 -0.16667 0.5 0 -0.5 0.16667 0 -0.16667 0.5 0 -0.16667 0.5 0 -0.5 -0.5 0
-0.16667 -0.5 0 0.16667 -0.16667 0 -0.16667 -0.5 0 0.16667 -0.16667 0 0.16667 -0.16667 0 0.16667 -0.16667 0
-0.16667 0 -0.16667 -0.16667 0 0.16667 0.16667 0 -0.16667 -0.16667 0 0.16667 0.16667 0 0.16667 0.16667
0.16667 0 -0.16667 0.16667 0 -0.16667 0.16667 0 0.16667 0.5 0 -0.16667 0.16667 0 0.16667 0.5 0
0.16667 0.5 0 -0.16667 -0.5 0 0.16667 -0.5 0 0.5 -0.16667 0 0.16667 -0.5 0 0.5 -0.16667 0 0.5
-0.16667 0 0.16667 -0.16667 0 0.16667 -0.16667 0 0.5 0.16667 0 0.16667 -0.16667 0 0.5 0.16667 0 0.5
0.16667 0 0.16667 0.16667 0 0.16667 0.16667 0 0.5 0.5 0 0.16667 0.16667 0 0.5 0.5 0 0.5 0.5 0 0.16667
```

Figura 5- Ficheiro *plane.3d*

4. Engine

Neste ficheiro criámos um “World” a partir do arquivo .XML de configuração que contém informações sobre a cena a ser renderizada. Em seguida o programa obtém as informações da câmara, projeção e arquivos de objetos a serem renderizados. Quando o utilizador pressiona a tecla “X”, existe alteração entre o modo de preenchimento e o contorno das figuras.

```
<group>
  <models>
    <model file="cone.3d" />
    <model file="sphere.3d" />
    <model file="box.3d" />
    <model file="plane.3d" />
  </models>
</group>
```

Figura 6- Exemplo do *config.xml*

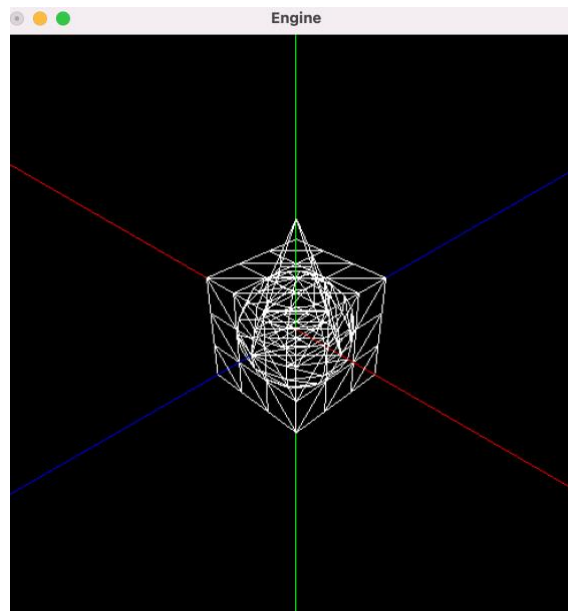


Figura 7- Exemplo de todas as figuras pretendidas nesta fase

5. Gerar modelos e demais comandos

Para a geração do ficheiro corremos os seguintes comandos:

```
$ cmake ..
```

```
$/group_project <nome_da_figura> <argumentos>
```

O parâmetro <nome_da_figura> pode ser "plane", "box", "cone" ou "sphere", dependendo do tipo de figura que se deseja gerar. Os argumentos necessários para cada figura são os seguintes:

- Para "plane": <dimensão> <divisão_edge>
- Para "box": <dimensão> <divisão_edge>
- Para "cone": <raio> <altura> <fatias> <camadas> <nome_do_arquivo>
- Para "sphere": <raio> <fatias> <camadas>

Para visualização das figuras alteramos o ficheiro que é referente à execução do programa, alterando a *source* de generator.cpp para engine.cpp. Com isto, conseguimos, através do seguinte comando: `$/group_project` ter a visualização completa e requerida de qualquer primitiva.

6. Conclusão e análise de resultados

Após a conclusão desta primeira fase do trabalho podemos afirmar que obtivemos os conhecimentos e destreza necessários para as próximas etapas. Conseguimos, com sucesso, gerar as primitivas requeridas (plano, caixa, esfera e cone) através de figuras mais simples como triângulos.