

Universidade do Minho
Escola de Ciências

Computação Gráfica
Fase I
Relatório de Desenvolvimento

André Oliveira Barbosa
A91684

Francisco António Borges Paulino
A91666

12 de março de 2023

Resumo

Este relatório foi elaborado no âmbito da primeira fase do trabalho prático da disciplina de Computação Gráfica, com a criação de primitivas gráficas.

Conteúdo

1	Introdução	2
1.1	Enquadramento e contexto	2
2	Análise e Especificação	3
2.1	Descrição do Problema	3
3	Concepção da Resolução	4
3.1	Gerador	4
3.1.1	Plane	4
3.1.2	Box	4
3.1.3	Sphere	4
3.1.4	Cone	5
3.1.5	Exemplo de execução	5
3.2	Engine	5
3.2.1	Plane and Sphere	6
3.2.2	Plane	6
3.2.3	Sphere	6
3.2.4	Box	7
3.2.5	Cone	7
4	Conclusão	10

Capítulo 1

Introdução

1.1 Enquadramento e contexto

Na primeira fase do trabalho prático da Unidade Curricular de Computação Gráfica foi-nos pedido gerar o ponto de partida do projeto, de forma a servir como base para as próximas fases.

Sendo assim, o objetivo deste relatório será detalhar o desenvolvimento desta fase, que se subdividiu na construção de um gerador e de um engine.

Capítulo 2

Análise e Especificação

2.1 Descrição do Problema

Tal como referido anteriormente, esta fase está essencialmente dividida entre a construção do gerador e do engine. Assim, deverá ser criado um gerador de modelos e um engine que possa, por sua vez, ler e exibí-los. A aplicação geradora deve ser capaz de criar quatro tipos de primitivas gráficas: plano, caixa, esfera e cone. O engine deve ser capaz de ler um arquivo de configuração em formato XML, que contém as configurações da janela e da câmera, e os arquivos gerados pela aplicação geradora.

Capítulo 3

Concepção da Resolução

3.1 Gerador

Primeiramente, construímos um gerador com o objetivo de gerar as primitivas de acordo com os seguintes parâmetros:

- Plano: Comprimento e divisões ao longo de cada eixo ;
- Cubo: Comprimento e divisões ao longo de cada eixo;
- Cone: Raio da base, altura, slices e stacks ;
- Esfera: Raio, slices, stacks ;

O gerador é responsável por criar ficheiros.3d com todos os vértices associados para a criação de um modelo, e com referência ao número total de vértices na primeira linha.

3.1.1 Plane

Para gerar o plano, foi utilizado um processo iterativo em que se percorreu os parâmetros do plano através de dois ciclos "for". Iniciou-se a iteração no local onde as coordenadas x e z são negativas, e percorreu-se cada linha do plano, desenhando simultaneamente os dois triângulos que compõem cada secção do plano.

3.1.2 Box

A box é formada por um conjunto de seis planos, dois em cada eixo. Ao construir cada plano, é fundamental considerar a sua orientação. Garantimos que as faces estejam sempre voltadas para o exterior do cubo simplesmente alternando os eixos utilizados na sua construção e a ordem dos vértices em cada triângulo.

3.1.3 Sphere

A esfera foi construída começando pelo polo superior e iterando através de cada slice da mesma. Para garantir uma descrição precisa das coordenadas de cada ponto da esfera, utilizou-se coordenadas polares, resultando numa representação realista da mesma.

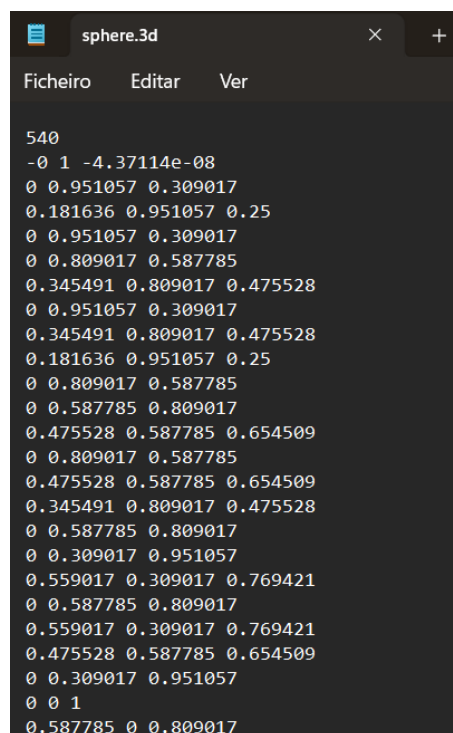
3.1.4 Cone

Para a construção da base do cone, e partindo da origem do referencial, foi criada uma base sobre o plano xz. A partir disso, a face do cone foi desenhada, utilizando novamente coordenadas polares e o teorema de Tales para determinar o valor do raio em cada "stack".

3.1.5 Exemplo de execução

```
C:\Users\infin\Desktop\Gerador_fase1\x64\Debug>Gerador_fase1.exe sphere 1 10 10 sphere.3d
Sucesso
```

Figura 3.1: Exemplo de criação de uma sphere com raio 1, 10 slices e 10 stacks



```
540
-0 1 -4.37114e-08
0 0.951057 0.309017
0.181636 0.951057 0.25
0 0.951057 0.309017
0 0.809017 0.587785
0.345491 0.809017 0.475528
0 0.951057 0.309017
0.345491 0.809017 0.475528
0.181636 0.951057 0.25
0 0.809017 0.587785
0 0.587785 0.809017
0.475528 0.587785 0.654509
0 0.809017 0.587785
0.475528 0.587785 0.654509
0.345491 0.809017 0.475528
0 0.587785 0.809017
0 0.309017 0.951057
0.559017 0.309017 0.769421
0 0.587785 0.809017
0.559017 0.309017 0.769421
0.475528 0.587785 0.654509
0 0.309017 0.951057
0 0 1
0.587785 0 0.809017
```

Figura 3.2: ficheiro sphere.3d com 540 vértices, criado após a execução do comando acima

3.2 Engine

O Engine tem como parâmetro um ficheiro XML que contém uma secção com os parâmetros da janela, posição da câmara, o ponto para onde olhar, o vetor "up" e as definições de projeção. Para além disso, existe outra secção com os caminhos para os modelos. A partir deste ficheiro irá ser gerado um modelo. Para que o motor tivesse esta funcionalidade, utilizamos o parser tinyXML2.

Depois de carregar os ficheiros .3d, os pontos neles contidos são armazenados num vetor de vetores de Ponto, uma estrutura de dados previamente definida que representa um ponto no espaço. Cada vetor armazena os pontos constituintes de uma primitiva gráfica, permitindo assim carregar várias primitivas. Adicionalmente,

o motor gráfico possui uma câmara que permite rodar em torno do modelo utilizando as setas do teclado, e as teclas F1 e F2 permitem afastar ou aproximar a câmara da origem do referencial, respectivamente. De seguida apresentamos exemplos de modelos gerados pelo engine a partir dos ficheiros .3d, após ler os ficheiros XML:

3.2.1 Plane and Sphere

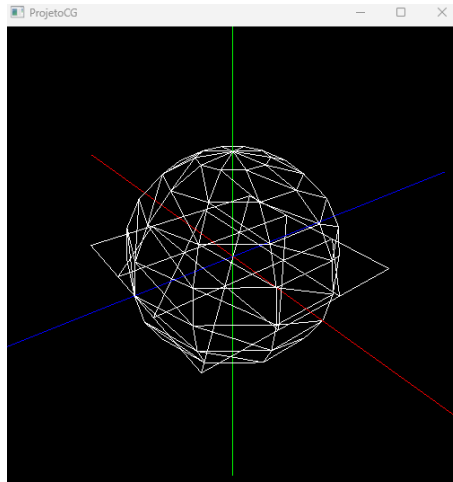


Figura 3.3: Plano e esfera gerados pelo engine

3.2.2 Plane

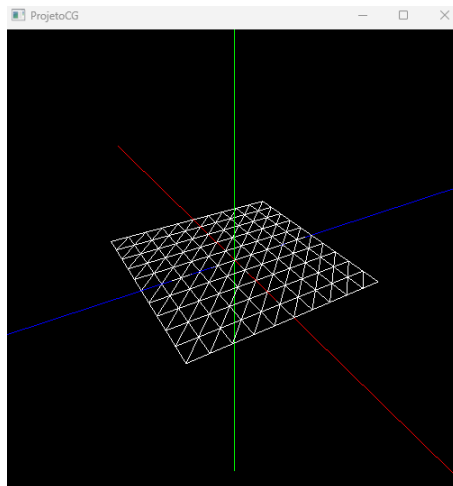


Figura 3.4: Plane gerado pelo engine

3.2.3 Sphere

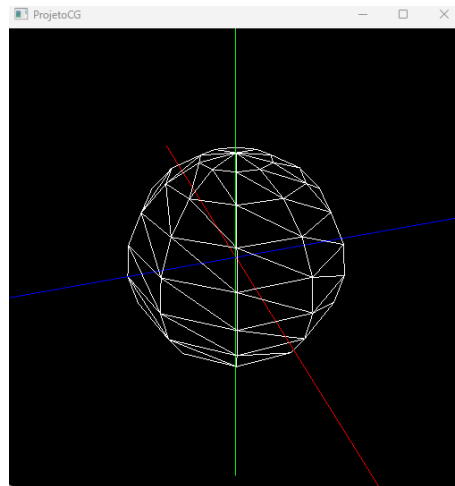


Figura 3.5: Sphere gerado pelo engine

3.2.4 Box

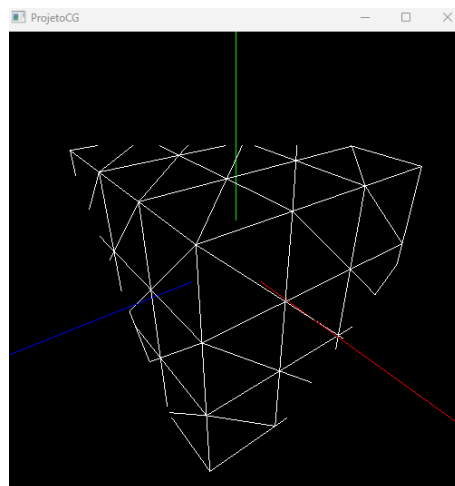


Figura 3.6: Box gerada pelo engine

3.2.5 Cone

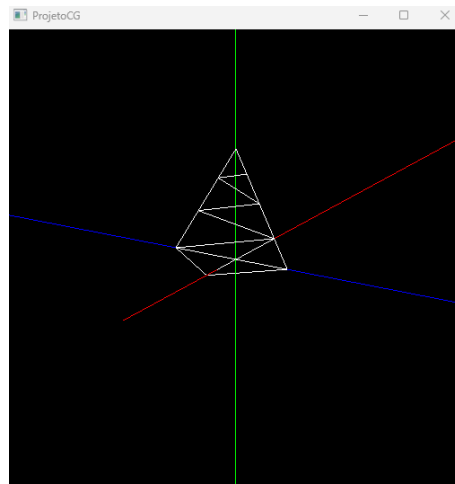


Figura 3.7: Cone gerado pelo engine

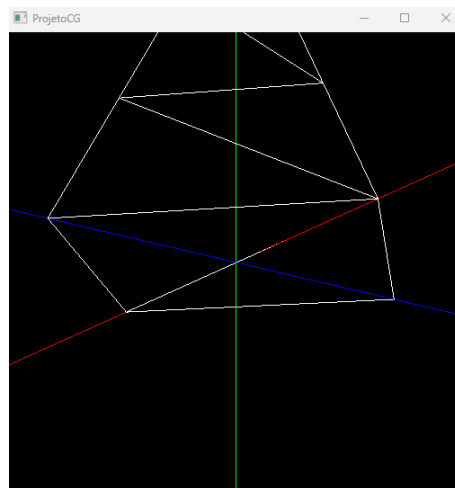


Figura 3.8: Cone gerado pelo engine

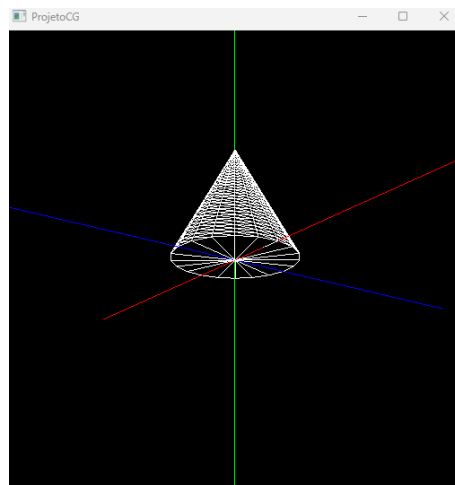


Figura 3.9: Cone gerado pelo engine

Capítulo 4

Conclusão

Apesar do projeto não ter sido concluído na data estipulada, e apesar das inúmeras dificuldades que enfrentamos, consideramos que esta fase do projeto foi bastante importante para colocar em prática a matéria aprendida nas aulas, bem como para nos atualizarmos na linguagem de c++, linguagem essa com a qual nunca tínhamos tido contacto. Consideramos também que os vários erros com que nos deparamos do visual studio, do CMake e do glut foram um entrave muito relevante para a não conclusão do projeto na data estipulada dado que não encontrávamos resolução para alguns deles. A leitura do ficheiro XML foi igualmente desafiante. Por fim, pensamos que construímos um gerador e um engine robustos e que respondem aos requisitos e objetivos propostos pela equipa docente.