Universidade do Minho

Departamento de Informática

Trabalho Prático

Ciências da Computação - LCC

Computação Gráfica

Grupo 25

Gonçalo Rodrigues A91641

Hugo Sousa A91654

Março de 2023

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

O presente relatório provém da primeira fase do projeto proposto no âmbito da Unidade Curricular de Computação Gráfica, com o objetivo de implementar um *generator* capaz de criar ficheiros com informações de figuras primitivas e uma *engine* 3D capaz de desenhar as mesmas.

1.1.1 Estrutura do Relatório

Neste relatório será descrita a interpretação efetuada do enunciado do trabalho prático e a abordagem usada para resolver os problemas propostos.
Começamos por descrever o *generator* e a *engine*, os requisitos para o funcionamento dos mesmos e de seguida será descrita a abordagem usada para as várias figuras. Serão ainda descritas as ferramentas usadas na execução deste trabalho e os aspetos do *generator* e da *engine*. Por fim, são apresentados diversos exemplos do funcionamento dos mesmos.

Capítulo 2

Análise e Especificação

2.1 Descrição e Enunciado

2.1.1 Generator

O *generator* é responsável por criar os ficheiros que serão usados pela *engine*. Este recebe como parâmetros a figura a ser criada, os parâmetros para a mesma e o ficheiro de destino onde esta será guardada. Nesta fase o *generator* é capaz de criar as primitivas para planos, caixas, esferas e cones, guardando os vértices resultantes num ficheiro com a extensão .3d.

2.1.2 Engine

A *engine* recebe a sua configuração através de um ficheiro XML, que contém o tamanho da janela, as definições da câmara e ainda os ficheiros que serão posteriormente carregados. É usada a ferramenta *tinyXML2* para facilitar a leitura do ficheiro com a configuração da *engine*.

Para além de ler uma configuração e desenhar as figuras pretendidas, a *engine* também permite movimentar a câmara de modo a facilitar a visualização das mesmas.

2.2 Requesitos

2.2.1 Generator

O *generator* têm a capacidade de gerar as seguintes primitivas:
Plano : Recebe a dimensão da aresta e o número de divisões e desenha um quadrado no plano **XZ**, centrado na origem e subdividido nas direções **X** e **Z**
Caixa : Recebe a dimensão da aresta e o número de divisões por aresta e desenha uma caixa centrada na origem.
Esfera : Recebe o raio, o número de número de slices e stacks e desenha uma esfera centrada na origem.
Cone : Recebe o raio da base, a altura, o número de slices e stacks e desenha um cone pousado no plano **XZ**.

2.2.2 Engine

A *engine* lê a configuração de um ficheiro XML (config.xml), altera os valores da janela e da câmara consoante os parâmetros no ficheiro, identifica os ficheiros que serão lidos e por fim desenha as figuras no ecrã.

Capítulo 3

Concepção e Desenho da Resolução

3.1 Plano

Para construir o plano utilizamos 2 triângulos para cada divisão (cada divisão vai ser um quadrado de *unit* por *unit*, sendo esta recebida por argumento), assim o resultado final do plano será um quadrado de *unit*divisions* por *unit*divisions* (sendo *divisions* recebido por argumento). Como o plano é construído no plano **XZ**, fixamos o valor do eixo **Y** a **0**. Seguidamente construímos os dois triângulos no segundo quadrante do plano **XZ** e aplicamos transformações a cada ponto de modo a construir os outros triângulos, sendo que para cada incremento do ponto do eixo **Z** incrementa *divisions* vezes o ponto do eixo **X**.

3.2 Caixa

A construção da caixa segue a mesma filosofia da construção do plano, visto que é construída com a "junção"de seis planos de comprimento *unit*divisions* (passados por argumento), mas com a condicionante de que não podemos fixar o valor do eixo de **Y** a **0**, assim o valor de **Y** será *unit*divisions/2* para que a caixa fique com centro no ponto *(0,0,0)*. Assim sendo, apenas precisamos de construir seis planos, dois em cada eixo, um no lado positivo e outro no lado negativo do eixo em questão.

3.3 Cone

Para a construção do cone começamos por construir a base, que está no eixo **XZ** e por esse motivo fixamos **Y** a **0**. Para construir construir os triângulos da base utilizamos as fórmulas ***sin(α)*radius*** e ***cos(α)*radius***. Assim para construir um dos triângulo usamos três vértices que são do tipo ***(sin(α)*radius, 0, cos(α)*radius)***, ***(0,0,0)***, ***(sin(β)*radius, 0, cos(β)*radius)***, onde α é o ângulo que é inicializado a **0** e β é o ângulo após ser aplicado o *step* que é traduzido pela fórmula *2*π/slices* (sendo *slices* e *radius* passado por argumento). Na iteração seguinte α passa a ser β e este é incrementado com o *step*, e assim sucessivamente. A base estará completamente construída quando o ângulo β for igual a π, ou seja, quando tiver feito tantos triângulos quando o número de *slices+1*.

As *stacks* são construídas ao mesmo tempo que a *slice* a que pertencem. Cada *stack* tem dois triângulos exceto a última que tem apenas uma, por este motivo num primeiro momento construímos as *stacks* que tem dois triângulos e no fim construímos a que tem apenas um. O raciocínio para construir as *stacks* é semelhante ao da base, mas neste caso tendo em consideração a altura, ou seja o **Y** de cada vértice, sendo este dado pelas fórmulas *(j)*heigth/stacks* ou *(j+1)*heigth/stacks* dependendo do vértice em questão (onde *height* é passado como argumento e *j* é a variável que é incrementada no ciclo), o **X** e **Z** de cada vértices seguem as fórmulas dadas na base sendo incrementados com *stacks-2* ou *stacks-1* dependendo mais uma vez do vértice em questão. Resta então construir o triângulo da última *stack* que segue a mesma lógica dos restantes mas apenas constrói um dos triângulos.

3.4 Esfera

Para a construção da esfera, temos dois ângulos (*a* e *b*) que determinam respetivamente as *slices* e as *stacks*. Fazendo *((2*π)/slices)* e *(π/stacks)* obtemos a variação dos ângulos *a* e *b* e, a cada iteração dos ciclos que percorrem todos os valores destes ângulos, obtemos quatro pontos (extremos do sector limitado por duas *stacks* e duas *slices* consecutivas) *p1*,*p2*,*p3*,*p4* (as coordenadas destes pontos são obtidas através das coordenadas esféricas dependentes de *a* e *b*), que usamos para desenhar dois triangulos, desenhando então o sector. Para os sectores entre as duas primeiras e as duas últimas *stacks*, apenas necessitamos de 3 pontos, pois a primeira e a ultima *stack* são formadas por um só ponto. Estas excepções são desenhadas respetivamente antes e depois dos ciclos que desenham a restante esfera.

Capítulo 4

Codificação

4.1 Código e Ferramentas Utilizadas

Para a execução deste trabalho foi utilizada a linguagem C++ tanto na construção do *generator* como da *engine*. Para permitir a leitura da configuração do motor foi usada a biblioteca *tinyXML2*.

4.2 Generator

A função principal do *generator* é escrever para um dado ficheiro os vértices que serão posteriormente representados numa determinada cena. Para este efeito foram usadas as funcionalidades da linguagem C++. A abertura e/ou criação deste ficheiro é feita através da função ofstream e a escrita de cada vértice é realizada usando os operadores de inserção "<<".

4.3 Engine

4.3.1 Leitura do Ficheiro XML

Foi utilizado o *tinyXML2* para permitir a leitura da configuração, guardada em XML, da *engine*. Nesta configuração está presente tanto a definição da janela como a da câmara, assim como os ficheiros com as informações dos vértices que têm de ser lidos.
Começamos por procurar a raiz do ficheiro, e de seguida vamos percorrendo os elementos com o objetivo de retirar as informações tanto da janela como da câmara. Para este efeito é procurado o elemento "window" e "camera" respetivamente. Por fim usamos um ciclo para procurar todos elementos "model" para que se possa guardar o nome dos ficheiros a ser lidos num vetor de strings.

4.3.2 Leitura dos Ficheiros .3D

Para a leitura dos ficheiros 3D, anteriormente guardados num vetor de strings, é percorrido cada ficheiro linha a linha para que as coordenadas dos vértices possam ser guardadas noutro vetor. Este processo permitirá que futuramente todos os vértices sejam desenhados sem ter de se estar a aceder ao ficheiro novamente.

4.3.3 Câmara

Foram utilizadas funções do Glut para processar *inputs* do utilizador de modo a permitir uma melhor visualização dos objetos no espaço tridimensional. Para este efeito serão usadas as *arrow keys* para movimentar a câmara no plano, e o *"w"* e *"s"* para aproximar e afastar a câmara da figura.

Capítulo 5

Testes

5.1 Plano

```
$ ./generator plane 5 3 ../../3d/plano.3d
```

Figura 5.1: Comando para criar o ficheiro .3D com os vértices do plano.

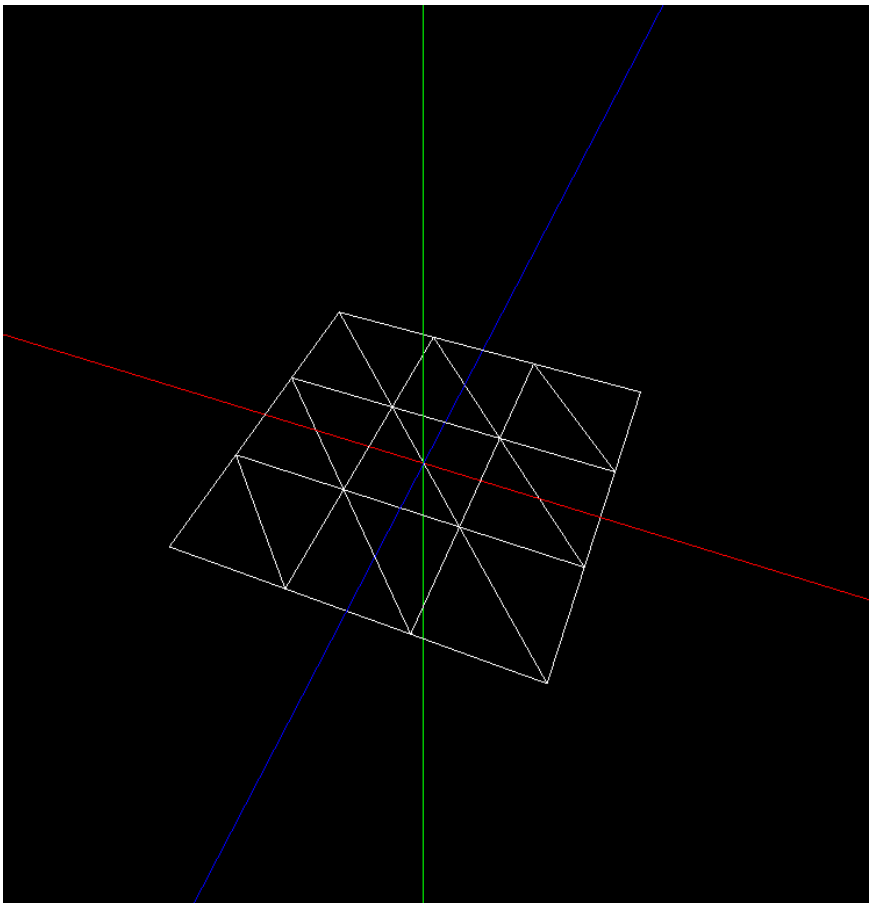

Figura 5.2: Representação do plano.

5.2 Caixa

```
$ ./generator box 5 3 ../../3d/caixa.3d
```

Figura 5.3: Comando para criar o ficheiro .3D com os vértices da caixa.

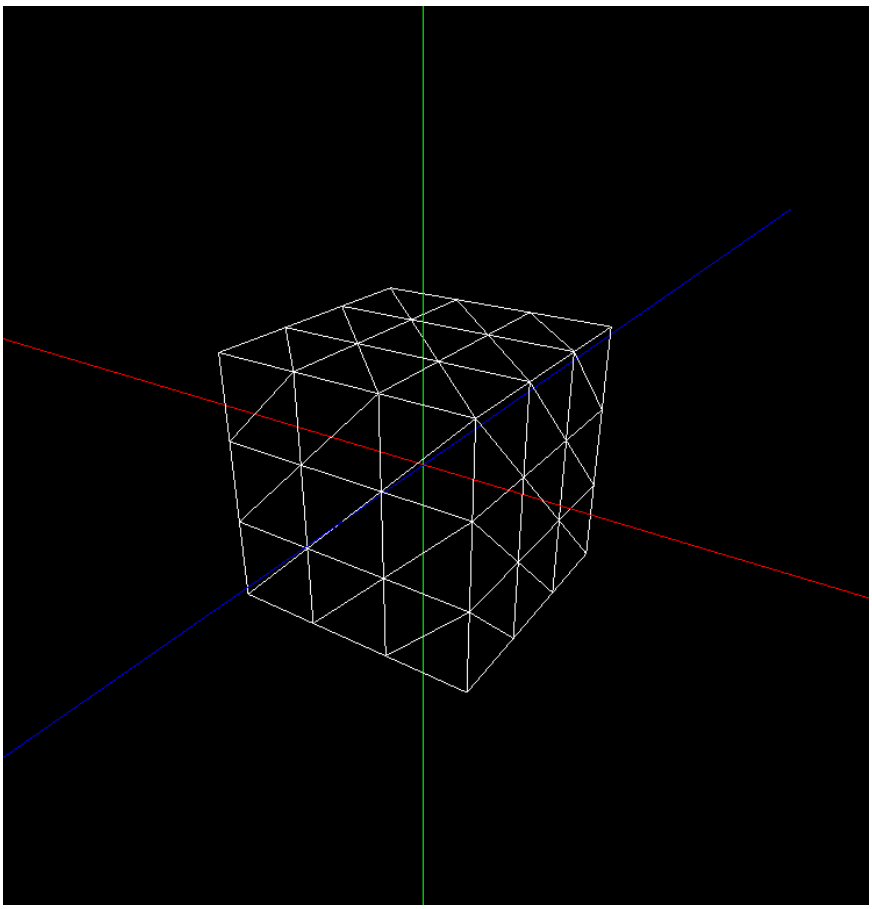

Figura 5.4: Representação da caixa.

5.3 Cone

```
$ ./generator cone 3 7 10 10 ../../3d/cone.3d
```

Figura 5.5: Comando para criar o ficheiro .3D com os vértices do cone.

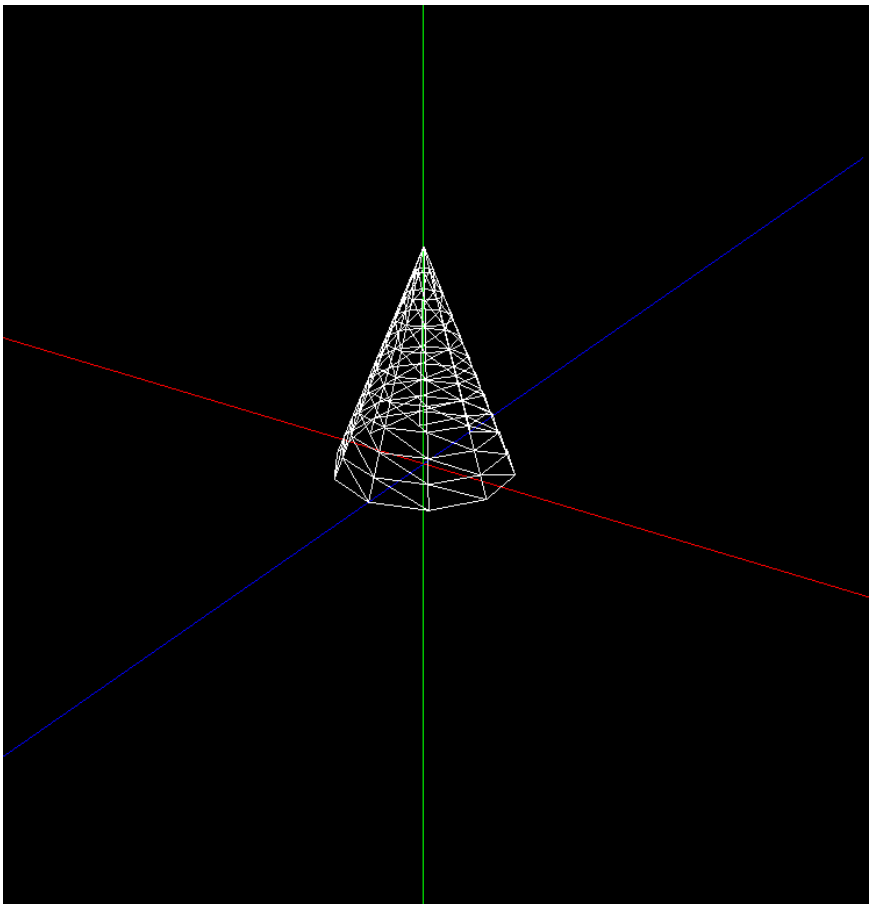

Figura 5.6: Representação do cone.

5.4 Esfera

```
$ ./generator box 5 3 ../../3d/caixa.3d
```

Figura 5.7: Comando para criar o ficheiro .3D com os vértices da esfera.

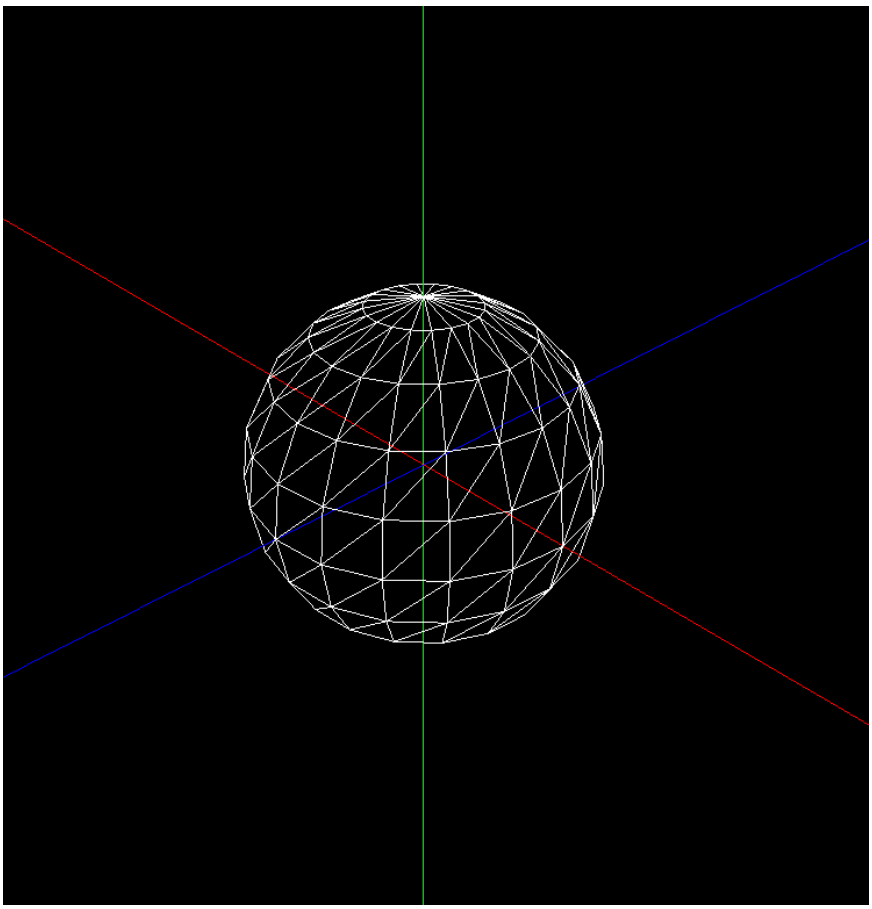

Figura 5.8: Representação da esfera.

Capítulo 6

Conclusão

Concluindo o relatório desta primeira fase do nosso trabalho prático, após esta demonstração percebemos que podíamos ter feitos alguns tópicos de maneira diferente. Íamos alterando o projeto de forma a otimizá-lo consoante os erros que nos iam aparecendo e perante as vantagens que verificávamos com essas alterações.

Um desses tópicos foi como iríamos centrar o plano na origem do plano cartesiano. No começo pensamos que seria necessário ter uma condição para os casos da dimensão ser par ou ímpar pois o centro varia, ou seja, no caso da dimensão ser par o centro ficaria num vértice do triângulo e caso fosse ímpar o centro ficaria no ponto médio da hipotenusa de um triângulo, mas com o decorrer do trabalho percebemos que é possível fazer translações que nos permitem colocar os pontos no sitio correto sem precisar de condições.

Outro contratempo, e este não conseguimos resolver, é o facto de que quando queremos movimentar a câmara pela primeira vez a posição em que a câmara se encontra é alterada drasticamente e assim notasse um pequeno movimento anormal. Pensamos que a causa deste problema é o cálculo dos ângulos dos vetores dos eixos com o vetor diretor da câmara ao centro na sua posição inicial não ser o correto, e isso afeta o primeiro movimento que fazemos. Depois desse movimento a câmara desloca-se normalmente.

Apesar destes contratempos, este projeto permitiu-nos rever todo o que nos foi ensinado durante as aulas da unidade curricular de Computação Gráfica e alguns conceitos sobre geometria e trignometria, principalmente na parte de desenhar as figuras.