Universidade do Minho

3º ano do Mestrado Integrado em Engenharia Informática

Computação Gráfica

Ano lectivo 2016/2017

Fase 1 - Primitivas Gráficas

Autores:
Diana Oliveira (a67652)
Manuel Moreno (a67713)
Xavier Francisco (a67725)

Braga, 6 de Março de 2017

Conteúdo

Resumo

1 Introducao

No âmbito para a Unidade Curricular de Computação Gráfica foi-nos proposto o desenvolvimento de um sistema de renderização 3D. Este sistema corresponde a duas componentes principais diferentes: o gerador, um programa que cria modelos das figuras geométricas implementadas e o motor de renderização que renderiza um cenário dado pelo utilizador.

Neste documento estão contidos todos os passos relevantes e decisões tomadas para o desenvolvimento deste projeto.

1.1 Estrutura do relatório

O Relatório está dividido em 4 capítulos:

No primeiro capítulo, é descrito mais detalhadamente o problema em mão e a especificação da nossa solução, tendo em conta os vários aspetos importantes para a realização do trabalho.

No segundo capítulo teremos a análise do gerador.

No terceiro capítulo teremos a análise do engine (Motor 3D).

E por último, no quarto capítulo, em que é feita uma conclusão, relativa ao estado actual do trabalho bem como o futuro trabalho que será desenvolvido.

2 Introdução ao problema

Começamos por analisar o enunciado e podemos separar em dois problemas diferentes. O primeiro consiste na criação de um programa, denominado por gerador, que ao receber os parâmetros necessários, gera todos os vértices dos triângulos necessários à construção das figuras pretendidas. Enquanto que os pontos vão sendo obtidos, estes vão sendo impressos num ficheiro. A segunda parte será o motor de renderização. Este vai receber um cenário descrito em XML, que para esta primeira fase, só irá referenciar os modelos a serem desenhados. Os triângulos contidos nos modelos referenciados no cenário serão então desenhados pelo motor. Este motor foi criado usando uma *toolkit* do OpenGL, conhecida por GLUT.

3 Gerador

A primeira etapa de desenvolvimento do projeto foi o gerador. Este gerador recebe o nome da figura geométrica, os argumentos que o caracterizam (dependendo da figura) e o nome do ficheiro de output. Depois de validar os argumentos passados, a função apropriada (definida em `create_models.c` é usada para criar um ficheiro da figura pedida. O ficheiro criado contém os pontos necessários (agrupados por triângulos), para a sua posterior renderização.

3.1 Formato do ficheiro de modelo

No decorrer do nosso projecto usaremos a extensão .3d para os ficheiros criados pelo nosso gerador. O formato do ficheiro criado segue a seguinte gramática:

```
modelo <- triângulo*
triângulo <- ponto ponto ponto
ponto  <- float float float newline
```

Este formato foi escolhido principalmente pela sua simplicidade, de forma a usar as funções standards de C e tornar o `parsing` deste ficheiro um *não-problema*. É de notar que apesar da sua simplicidade, o formato preenche bem os requisitos necessários, visto não haver mais informações que se queira passar ao motor nesta fase.

3.2 Plano

Para a implementação do plano, o gerador cria dois triângulos de iguais dimensões, coplanares e adjacentes pela hipotenusa. É de notar que este plano é de tamanho finito de lado igual ao cateto dos triângulos.

3.2.1 Resultados

A seguir apresenta-se o plano gerado, com o seguinte comando:

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

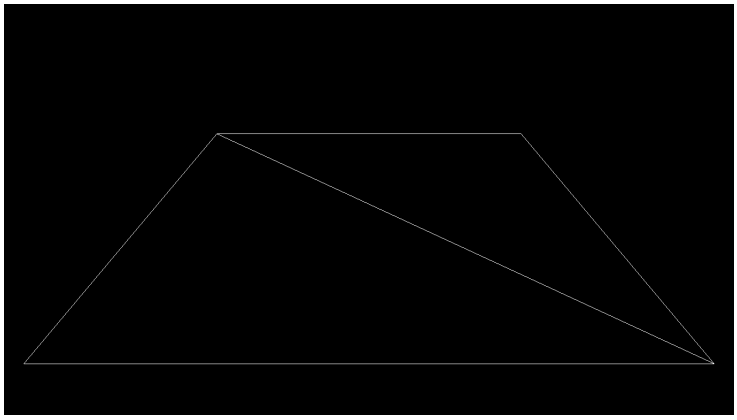

Figura 1: Representação do plano gerado

3.3 Caixa

As caixas tem como argumento, o seu comprimento, largura, altura e o número de divisões `n`. São criadas centradas na origem.

Na criação das caixas são desenhados n^2 retângulos (da mesma maneira que o plano) por cada face, formando uma matriz quadrada. Cada retângulo tem a mesma forma que a face que a preenche (por forma, entende-se a razão entre o comprimento e a largura).

Este processo pode ser melhor percebido, tendo em conta que uma das componentes das coordenadas dos pontos de uma face, será constante. Assim podemos reduzir o problema a desenhar um quadrado, como o que vemos no seguinte exemplo:

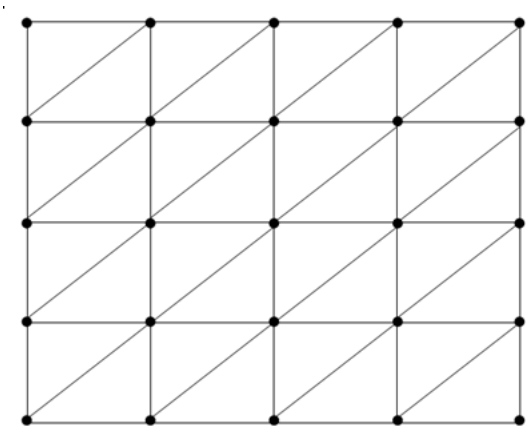

Figura 2: Desenho de uma face

De seguida é apresentado dois exemplos de paralelepípedo gerados:
O primeiro gerado com o seguinte comando:

```
\$ ./generator box 1 1 1 1 simple_box.3d
```

Este modelo com uma só divisão permite-nos ter uma visão simples do cubo criado e confirma a nossa afirmação. Cada face tem `1` retângulo e todas as faces tem igual tamanho à face que a preenche (trivialmente, têm também a mesma forma).

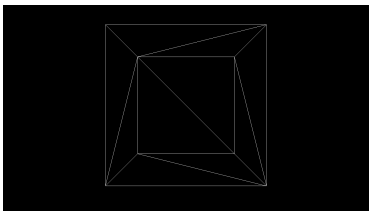

Figura 3: Visão de cima de um cubo 1x1x1

O segundo é gerado com o seguinte comando:

```
\$ ./generator box 1 3 3 5 complex_box.3d
```

Esta figura é vista de dentro, para podermos melhor ver os diferentes elementos. Podemos logo ver a semelhança entre o quadrado de cima e esta imagem. Também podemos ver que os

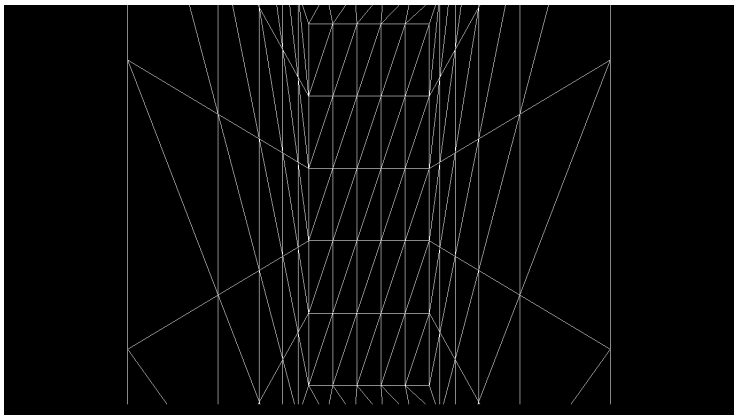

Figura 4: Visão de dentro de um paralelepípedo 3x3x5

rectângulos têm diferentes tamanhos dependendo da face (os de frente w:3, h:5 e os de lado w:5, h:5).

3.4 Esfera

Para criar uma esfera é nos passado três argumentos: o seu raio, o número de stacks e o número de slices. O número de stacks vão ser o número de divisões horizontais da esfera, e o número das slices as divisões verticais.

A seguinte imagem ajuda-nos a esclarecer o resultado ao que queremos chegar:

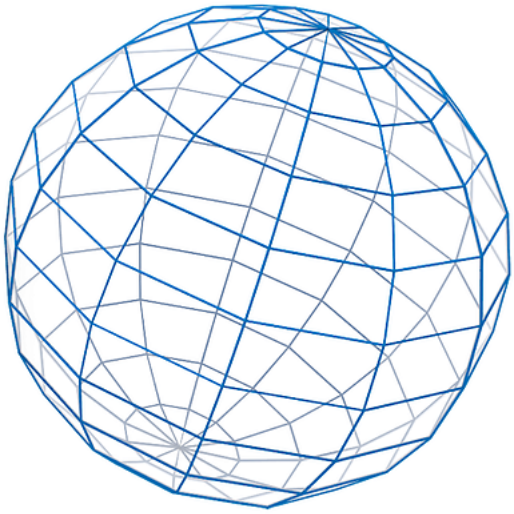

Figura 5: Esfera dividida por slices e stacks

Para criar esta esfera, definimos duas constantes: sliceAngle e stackAngle, respectivamente o ângulo de cada slice e de cada stack e duas variáveis: alpha e beta, respectivamente a longitude e a latitude da esfera. A esfera todo então é descrita, por esta figura para todos os alphas e betas:

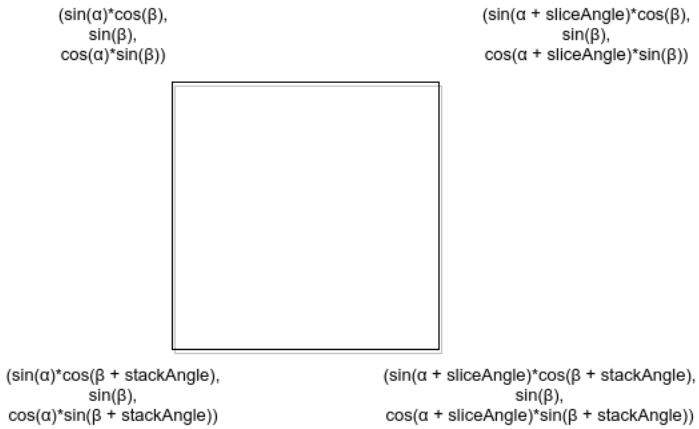

Figura 6: Como são descobertos as coordenadas da esfera.

Este quadrado representa os quadrados que podemos ver no fig5.
Por fim, acabamos com visualizações da esfera criada.

Figura 7: Uma slice e uma stack respetivamente, exibidas singularmente.

3.5 Cone

O cone receberá como argumentos, o raio da base, a sua altura, as slices e as stacks. As slices e stacks funcionam de forma análoga à esfera.

A criação do cone começa pelos triângulos da base, formando um polígno regular com o mesmo número de lados que o número de slides dado como argumento. Na criação do cone, como na esfera, temos uma constante em que o ângulo é divido pelo número de slices(sliceAngle), no entanto, é também preciso dividir a altura pretendida pelo número de stacks(stackHeight).

A seguir apresenta-se dois exemplo de cones. O segundo com 50 stacks e 50 slices:

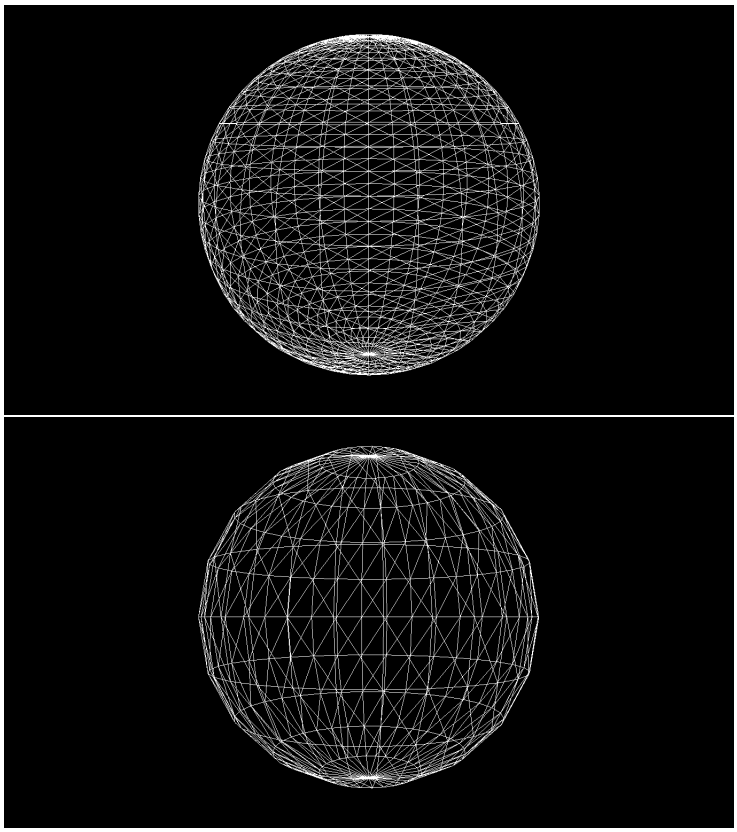

Figura 8: Diversas esferas

Figura 9: Diagrama do cone que queremos

4 Motor-3D

O "engine" desempenha um papel muito importante no projeto. O objetivo deste motor é ler um ficheiro XML com a informação dos modelos a desenhar, e assim desenhar o sólido correspondente. Para desenvolver o motor baseamos nos no esqueleto que foi utilizado nas aulas práticas.

Adicionamos rotações e translações da câmara através das teclas normais (opwsad) para uma melhor visualização dos sólidos construidos.

Os triângulos são desenhados através do **GL_TRIANGLES** e do **glVertex3f** cujos pontos estão guardados no ficheiro lido, na ordem correta para a obtenção de figuras corretas.

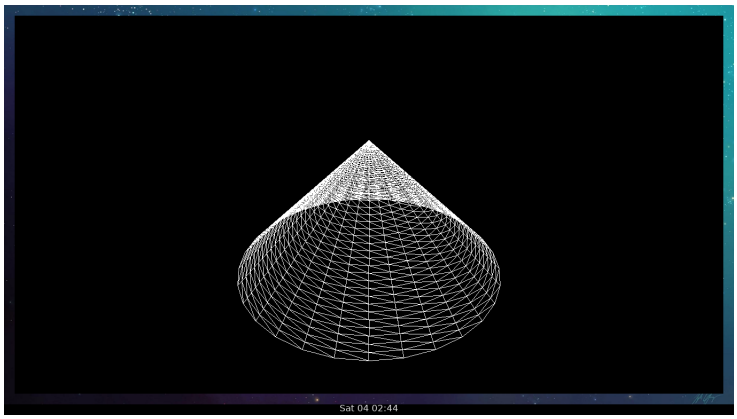

Figura 10: Cone

5 Conclusão e Trabalho Futuro

Concluida esta primeira fase do projeto de Computação Gráfica, podemos verificar que conseguimos consolidar os conhecimentos adquiridos nas aulas tanto teóricas como práticas. No entanto, tivemos algumas dificuldades nas coordenadas polares e esféricas relacionadas com a visualização dos pontos no espaço, contudo conseguimos formar todas as figuras geométricas sugeridas. Conseguimos também criar as duas aplicações pedidas, o gerador e o motor3D. Como trabalho futuro, adicionaremos transformações geométicas: translação, rotação e escala.