Computação Gráfica

Fase 1/4

"Primitivas Gráficas"

Hugo Sousa, A76257 Matias Capitão, A82726

06 de Março 2020

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Esta primeira fase do trabalho prático vai incidir na criação de dois programas, nomeadamente, um gerador que irá produzir um ficheiro do tipo ".3d"que contém os vértices das figuras que foram pedidas como argumento e um motor que irá ler o ficheiro referido anteriormente e desenhar as figuras, depois de verificar a sua presença num ficheiro XML.

Capítulo 2

Análise e Especificação

2.1 Gerador

A função main do gerador vê qual é o modelo que se pretende gerar (dependendo do input do utilizador), chama a função geradora da forma pretendida (tendo em conta o número de argumentos fornecidos) e escreve o resultado num ficheiro com o nome fornecido como primeiro argumento, todas as figuras são escritas como conjuntos de triângulos.
No caso do input ser errado, mensagens de erro, com as instruções para desenhar cada modelo, serão lançadas para o stdout. Os ficheiros criados são guardados na pasta "assets"para serem posteriormente lidos pelo engine.

2.1.1 Plano

Sintaxe do input:

`gen <outfile> plane <sizewidth> ou opcionalmente gen <outfile> plane <xsize> <zsize>`

(na primeira opção assume-se que os dois lados têm o mesmo tamanho então será criado um quadrado de lado <sizewidth>na segunda opção o comprimento e largura do modelo poderão ser diferentes.)

Depois de lidos os dados do input, estes são processados e guardados numa estrutura "struct Plane"que define um rectangulo, de seguida os pontos vão ser escritos como um conjunto de dois triangulos atendendo à ordem de escrita no sentido "counter clock-wise",

2.1.2 Box

Sintaxe do input:

`gen <outfile> box <xsize> <ysize> <zsize> <divisions>`

ou, opcionalmente,

`gen <outfile> box <xsize> <ysize> <zsize>`

(na primeira opção assume-se que o cubo não tem divisioes)

1. Posicionamento inicial na coordenada (-x/2, -y/2, -z/2);

2. Vamos criar os pontos de cada face:
    - O conjuntos de pontos dos triângulos da face pertencente ao plano X=-x/2, por incremento dos saltos no eixo dos ZZ e YY;
    - O conjunto dos pontos dos triângulos das faces pertencentes aos planos perpendiculares ao eixo dos XX, por incremento dos saltos no eixo dos ZZ, YY e XX;
    - O conjunto dos pontos dos triângulos da face pertencente ao plano X=x/2, por incremento dos saltos no eixo dos ZZ e YY;

2.1.3 Esfera

Sintaxe do input:

```
gen <outfile> sphere <radius> <slices> <stacks>
```

1. Posicionamento inicial na coordenada (0, 0, 0);
2. Determinamos os pontos na esfera;
3. Guardamos esses pontos num vetor;
4. Vamos buscar 4 ponts de cada vez para fazer um plano que vai ser escrito como dois triangulos.

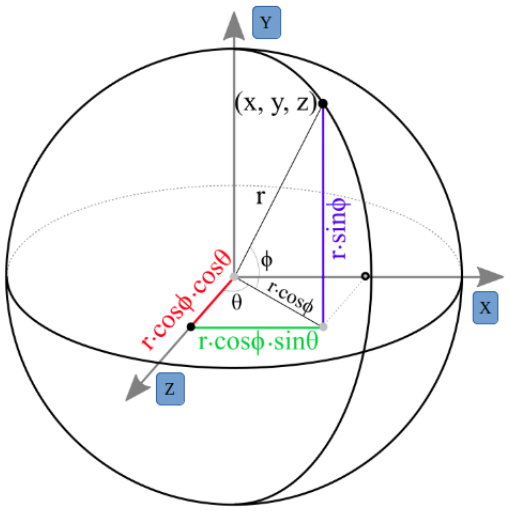

Figura 2.1: Coordenadas dos pontos na matriz

2.1.4 Cone

Sintaxe do input:

```
gen <outfile> cone <radius> <height> <slices> <stacks>
```

1. centro da base é o ponto B_origin(0,0,0) e o topo da base é o ponto T_origin(0,height,0);

2. O cone é contruido slice a slice começando com o mini cone no topo depois a base e de seguida cada lado(aqui entram as stacks e vamos construir todos os pontos das stacks como um plano de 4 pontos (isto é dois triangulos), excluindo os ultimos pontos que são os do mini cone no topo que já foram construidos).

2.2 Motor

A segunda parte do trabalho consiste na criação de um programa que, a partir da leitura de um ficheiro XML gera essas mesmas figuras. Nesta fase o ficheiro XML apenas contém o nome do ficheiro criado com o generator onde por sua vez estarão contido os dados sobre as figuras a desenhar. O ficheiro XML lido pelo drawer tem, nesta etapa, este formato:

```
<scene>
        <model file="sphere.3d" />
</scene>
```

Figura 2.2: XML

A aplicação apenas desenha a(s) figura(s) a partir dos pontos dos modelos contidos no ficheiro xml. Estes pontos são lidos para memória, fazendo recurso à biblioteca "tinyxml"e a uma estrutura de dados global models que guarda a informação dos modelos, e só depois é feito o rendering dos triângulos gerados.
Este programa permite ainda:

- Zoom in e Zoom out usando as teclas 'w' e 's' respectivamente;
- A rotação da camera em torno do objeto usando as setas do teclado;
- Mudar o modo de desenho com a tecla esquerda do rato.

Capítulo 3

Resultados

Depois de correrrmos o nosso gerador o resultado será algo do género:

```
0x0p+0 0x1p+1 0x0p+0
0x0p+0 0x1p+1 0x0p+0
0x1.4060b8p-2 0x1.f9b24ap+0 0x0p+0
0x1.4060b8p-2 0x1.f9b24ap+0 0x0p+0
0x0p+0 0x1p+1 0x0p+0
0x0p+0 0x1p+1 0x0p+0
0x1.4060b8p-2 0x1.f9b24ap+0 0x0p+0
0x0p+0 0x1p+1 0x0p+0
0x1.30b28ap-2 0x1.f9b24ap+0 0x1.8c023ep-4
0x1.30b28ap-2 0x1.f9b24ap+0 0x1.8c023ep-4
0x0p+0 0x1p+1 0x0p+0
0x0p+0 0x1p+1 0x0p+0
```

Figura 3.1: Ficheiro do tipo .3d para uma esfera

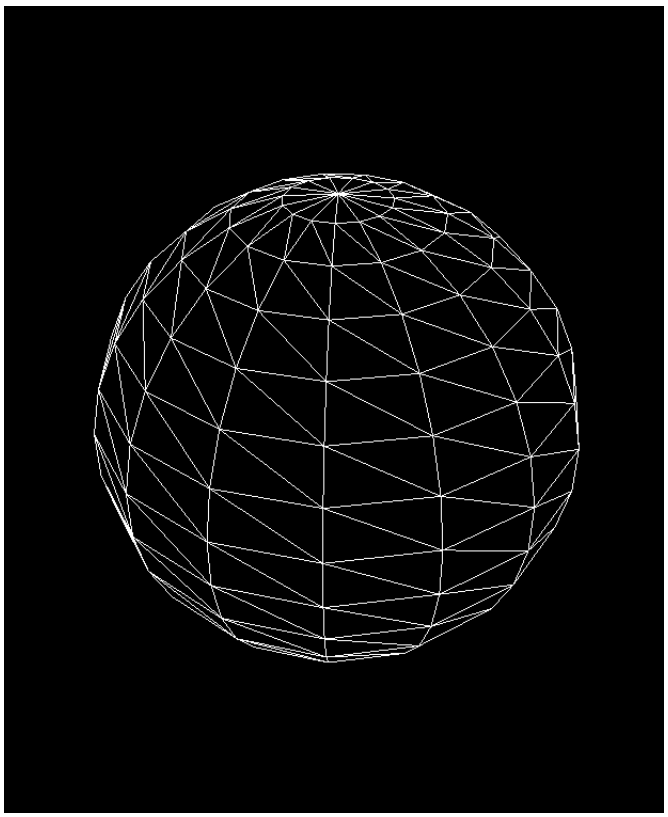

Figura 3.2: Esfera com raio 2, 15 stacks e 15 slices

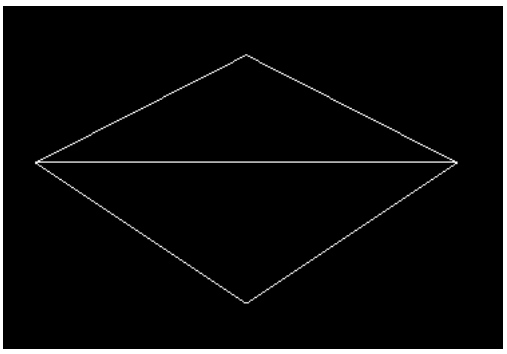

Figura 3.3: Plano com largura 2 e comprimento 2

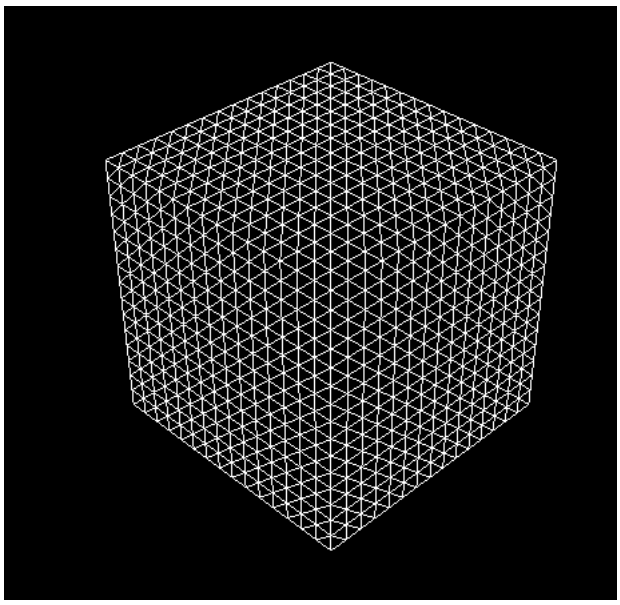

Figura 3.4: Box com largura, altura e comprimento 2 e 15 divisões

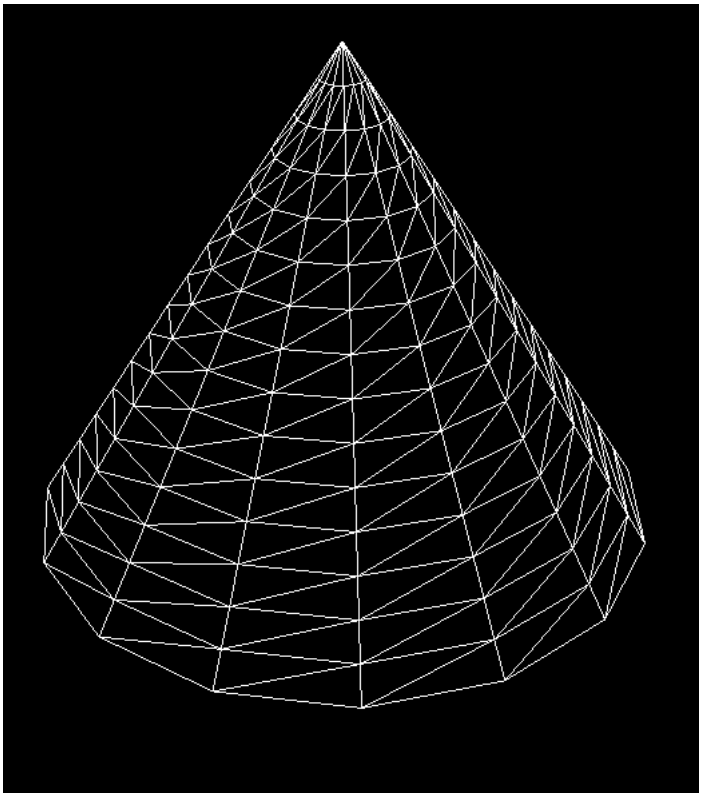

Figura 3.5: Cone com raio 2, altura 3, 15 stacks e 15 slices

Capítulo 4

Bibliografia

http://www.songho.ca/opengl/gl_sphere.html

Capítulo 5

Conclusão

Esta primeira fase do trabalho permitiu-nos compreender melhor como trabalhar com o OpenGL. Houveram algumas situações que criaram certas dificuladades mas acreditamos que conseguimos supera-las e concluir com um trabalho que no global nos parace bem conseguido. De notar que, a ferramenta TinyXML se mostrou extremamente útil aquando da leitura do ficheiro XML.