Computação Gráfica

Relatório

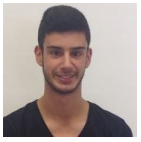
1º Fase - Primitivas gráficas

Ana Paula Carvalho
a61855

Bruno Manuel Arieira
a70565

João Miguel Palmeira
a73864

Pedro Manuel Almeida
a74301



12 de Março de 2018

Índice

List of Figures

1 Introdução

Para esta primeira fase do trabalho prático, foi-nos proposto o desenvolvimento de um gerador de pontos de algumas primitivas gráficas – um plano, uma caixa, uma esfera e um cone – dada a sua designação. Para tal primeiro é necessário a criação de um motor capaz de desenhar modelos 3D armazenados em ficheiros individuais (ficheiros .3d) que foram gerados pela aplicação geradora (gerador).

Ao longo deste relatório incluímos descrições detalhadas de todas as etapas relativas a presente fase, documentando e explicando o trabalho desenvolvido.

2 Desenvolvimento

2.1 Gerador

Recebe como argumentos o nome da primitiva desejada, os parâmetros necessários para a sua construção e o nome ficheiro .3d que irá gerar.

Dependendo do primeiro argumento passado – Plano, Caixa, Esfera, ou Cone – é feita a verificação se os restantes argumentos estão corretos. É gerado o ficheiro XML e, seguidamente, são invocadas as funções draw correspondentes à primitiva pretendida, criado o ficheiro.3d e calculadas as coordenadas dos pontos, sendo estas depois escritas para o ficheiro.

Este é o funcionamento genérico do gerador. Para cada uma das primitivas gráficas, ilustraremos ainda o funcionamento específico.

2.1.1 Plano

```
void drawPlano(float length, float width, char* filename){
```

Figure 1: Função que desenha o plano

Calcula os pontos necessários para criar um plano, centrado na origem, com dimensões de length x width passadas como argumentos. Para gerar esta primitiva, são necessários seis pontos pertencentes a dois triângulos que, concatenados, irão formar o plano almejado.

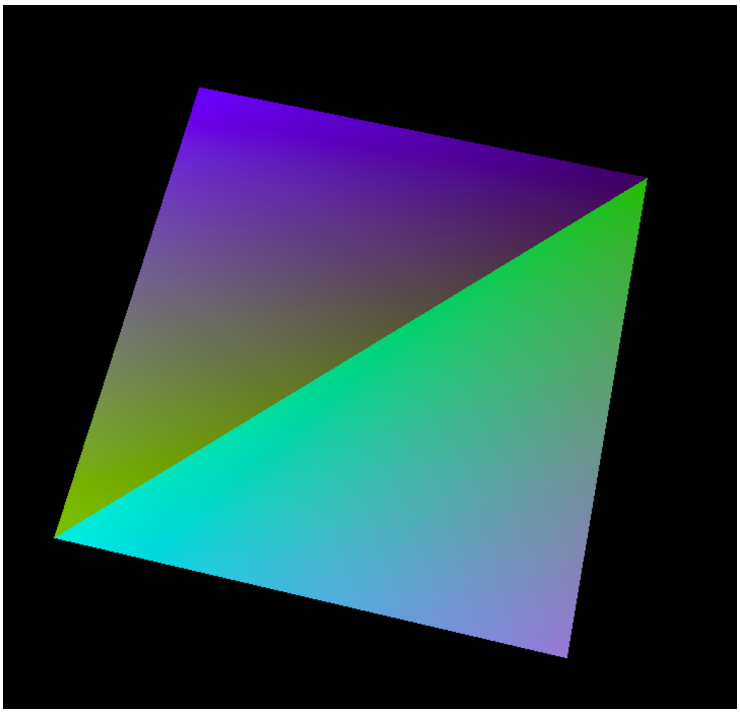


Figure 2: Exemplo de um plano gerado

2.1.2 Caixa

```
void drawCaixa(float length, float height, float width, char* filename){
```

Figure 3: Função que desenha uma caixa

Para esta primitiva gráfica são necessários três parâmetros para calcular pontos: length, height e width. Para se construir uma caixa são necessários seis planos que, como já referido no ponto acima, requerem seis pontos cada. Portanto, para a caixa precisa-se de calcular trinta e seis pontos.

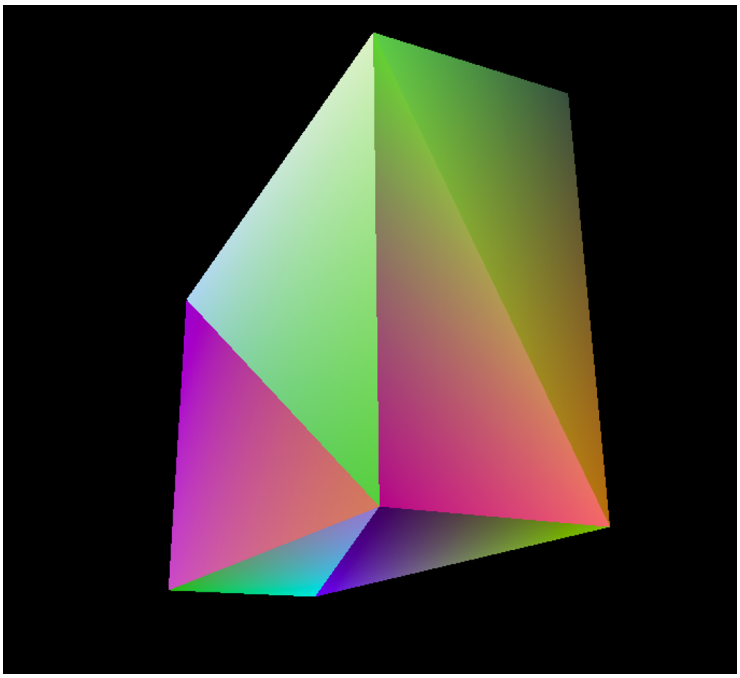


Figure 4: Exemplo de uma caixa gerada

2.1.3 Esfera

```
void drawEsfera(float radius, int slices, int stacks, char* filename){
```

Figure 5: Função que desenha uma esfera

De modo a obter a criação de uma nova esfera são necessários passar como parâmetros o raio desejado, em quantas "fatias" queremos dividir dado como slices e o numero de camadas desejadas passadas no parâmetro stacks. Com estes 3 parâmetros procede-se ao calculo dos pontos necessários para acriação da esfera. Esses pontos são calculados através de 2 ciclos onde para cada camada, são calculados 6 pontos por cada slice entre 0 e 2pi. Isto começa a criação da esfera na parte inferior e irá sendo desenvolvida camada a camada até ao topo.

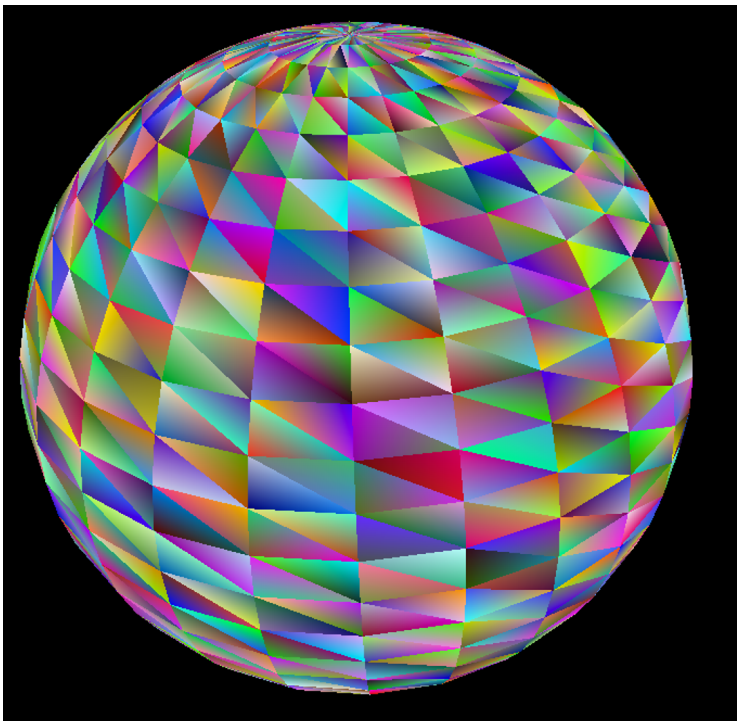


Figure 6: Exemplo de uma esfera gerada

2.1.4 Cone

```
void drawCone(float radius, float height,  int slices, int stacks, char* filename){
```

Figure 7: Função que desenha um cone

Para a criação da primitiva gráfica correspondente ao cone serão necessários 4 argumentos. O primeiro irá indicar qual o raio da base circular do cone, o raio da base. De seguida, temos o parâmetro correspondente á altura total do mesmo. Por ultimo, tal como se encontra na esfera, damos as slices e stacks pretendidas para a construção do cone. Através do raio e slices, obteremos os pontos necessários para o desenho da base, sendo os restantes dos pontos obtidos através dos restantes parâmetros através de ciclos que terminam no topo do cone.

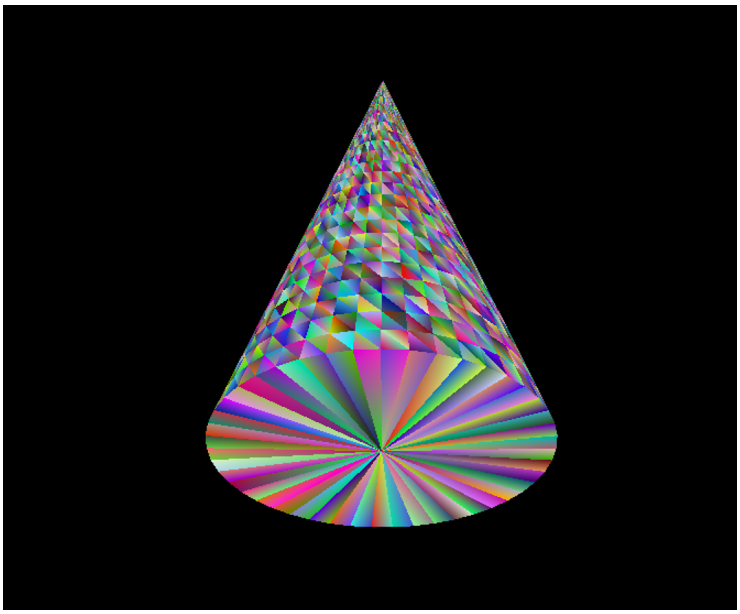


Figure 8: Exemplo de um cone gerado

2.2 Motor

O motor tem como principais funções executar parsing de ficheiros XML e executar um rendering da informação recolhida no parsing, isto é, o Motor 3D lê o ficheiro XML com a informação dos modelos dos objetos a desenhar e através dessa informação desenha o sólido correspondente.

Para o desenvolvimento do motor baseamo-nos no esqueleto que nos foi fornecido para utilização nas aulas práticas.

Elaboramos ainda um ficheiro, estruturas.h, com as diferentes estruturas que utilizamos e a função que carrega e faz o parsing do ficheiro XML, função load. Para realizar o parsing foi também necessário a criação de algumas classes, tais como: Ponto, Modelo e Figura.

```
class Ponto {
public:
    float coordx;
    float coordy;
    float coordz;

};

class Modelo{
    public:
    string ficheiro;
    list<Ponto> pontos ;
    Modelo(){ this->ficheiro = "untitled" ; this->pontos = list<Ponto>() ; }
    Modelo( const string& ficheiro, list<Ponto> pontos){
        this->ficheiro = ficheiro ;
        this->pontos = pontos ;
    }
} ;

class Figura {
public:
    list<Modelo> modelos;
    Figura() { this->modelos = list<Modelo>(); }
    Figura(list<Modelo> modelos) { this->modelos = modelos; }
    void load(const char* pFilename) ;
};
```

Figure 9: Código das Estruturas criadas

O Ponto apenas é composto pelas coordenadas de um ponto, por isso, esta classe é apenas composta por três floats que representam os eixos x, y e z. Já o Modelo é uma estrutura composta unicamente por uma lista de pontos. Das três referidas, o Figura é a estrutura mais abrangente uma vez que engloba uma lista de modelos.

A função Main estará exposta no ficheiro .cpp que realizará o load da Figura e através do parse do ficheiro lido e será desenhado todos os triangulos obtidos a partir dos pontos guardados no mesmo. Este processo é realizado com sucesso através do GL_TRIANGLES e do glVertex3f.

Foram também implementados comandos de modo a navegar sobre os desenhos obtidos como, por exemplo, a tecla do lado esquerdo do rato permitir o reposicionamento da camara ou através das setas presentes no teclado.

Translações serão obtidas através das teclas J, K, L e I, bem como o zoom que se pode fazer no objeto através das teclas S (menos zoom) e W (mais zoom).

3 Conclusão

A primeira parte do trabalho serviu para nos introduzir aos modelos 3D que foram lecionados até ao momento nesta Unidade Curricular e desenvolvê-los numa área mais prática. Para tal, foi proposto a criação de um motor 3D que recebe um ficheiro de extensão .3d, criado num gerador com o nome do sólido e os respetivos dados.

Dos modelos 3D gerados, aquele que suscitou mais dificuldades foi o cone, dado que contêm uma complexidade acrescida em relação aos outros modelos. No entanto, após algumas pesquisas e com os conhecimentos adquiridos na aula, achamos que o resultado foi satisfatório.

Por outro lado, o facto de ser necessário o uso e leitura de ficheiros XML resultou também na necessidade de alguma pesquisa para ultrapassar esta adversidade, o que nos levou a criação das estruturas que mencionamos anteriormente.

Por fim, este trabalho ajudou-nos a consolidar os conhecimentos lecionados nas aulas práticas, o que nos será muito útil no futuro.