Universidade do Minho

Mestrado Integrado de Engenharia Informática

Primitivas Gráficas

Trabalho Prático
Graphical Primitives

Autores:

Sara Pereira	A73700
Filipe Fortunato	A75008
Frederico Pinto	A73639

4 de Abril de 2018

Conteúdo

1 Introdução

1.1 Contextualização

No âmbito da UC Computação Gráfica, foi-nos proposto o desenvolvimento de um mecanismo 3D baseado num cenário gráfico com o auxílio de duas ferramentas utilizadas, também, nas aulas práticas, sendo elas OpenGL e C++. Este projeto consistem em 4 fases distintas, sendo esta a primeira que consiste na representação de algumas primitivas gráficas.

2 Arquitetura do código

Para esta fase do projeto, foi decidido que se ia dividir o mecanismo em duas aplicações principais: o gerador e o motor. Ao mesmo tempo, foram criados ficheiros com as estruturações dos diferentes modelos geométricos, sendo os requisitados o plano, a box, o cone e esfera.

2.1 Aplicações

Nesta secção vão ser retratadas as aplicações supracitadas, assim como os respetivos ficheiros gerados.

2.1.1 Gerador

Como o nome indica, esta aplicação destina-se a gerar os vários pontos constituintes das diferentes primitivas gráficas requisitadas- plano, caixa, o cone e a esfera-, conforme os parâmetros escolhidos (dimensões, fatias e camadas). Graficamente, estes vértices correspondem a triângulos, pois estes são a unidade de constituição de todas as primitivas. O código desta aplicação encontra-se no ficheiro gerador.cpp, sendo esta uma complementar ao motor.

2.1.2 Utilização

Após a utilização do comando *g++ gerador.cpp -o gen* dá-se a criação do executável. Com a invocação do comando *./gen plane 2 5 plano.3d*, por exemplo, dá-se a criação de um ficheiro plano.3d que contém no seu interior as coordenadas x,y,z de todos os pontos que constituem a figura geométrica pedida. Em seguida é apresentado o menu de ajuda da parte do gerador.

Figura 1: Menu ajuda da parte do gerador

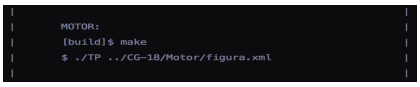
Figura 2: Menu ajuda da parte dos modelos geométricos

3 Motor

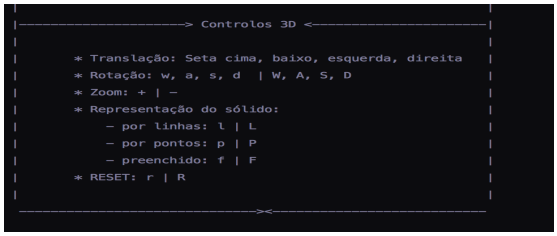
Esta aplicação destina-se a receber os ficheiros escritos em XML, que indicam apenas a localização do ficheiro criado pelo gerador, fazendo o parser desses ficheiros gerados interpretando e representando os modelos no seu conteúdo. Para a implementação deste motor3D, partimos do esqueleto que é utilizado nas aulas práticas contendo diretrizes obrigatórias do Glut e fomos acrescentando funções e variáveis que fazem sentido para o problema em causa. Para lidar com os pontos foi criada uma estrutura Point que armazena cada uma das coordenadas x,y,z que identificam um ponto, sendo que implementamos um vetor de Points para armazenar os diversos pontos que vão sendo lidos do ficheiro XML.

3.1 Utilização e Demonstração

De seguida é apresentado o manual de ajuda para a utilização do motor, este pode ser visualizado através da invocação do comando *./motor help*. São, também, referidos os diferentes inputs válidos para a interação com os modelos.



(a) Menu ajuda da parte do motor



(b) Menu ajuda com comandos para interação com os modelos

Antes de demonstrar o funcionamento desta aplicação é importante voltar a referir a criação de um ficheiro xml, que vai ser lido pelo motor. No entanto, esta criação é feita manualmente pelo utilizador e os ficheiros presentes no mesmo criados previamente pelo gerador. Após a criação dos modelos pretendidos, é possível a interação com os mesmos com a utilizaçao dos comandos supracitados. Segue um exemplo do funcionamento do motor, dado, por exemplo, o seguinte ficheiro xml como input:

```
<scene>
    <model file='cone.3d'/>
</scene>
```

Figura 4: Exemplo de um ficheiro em XML

O output deverá ser uma representação *GLUT* de um cone

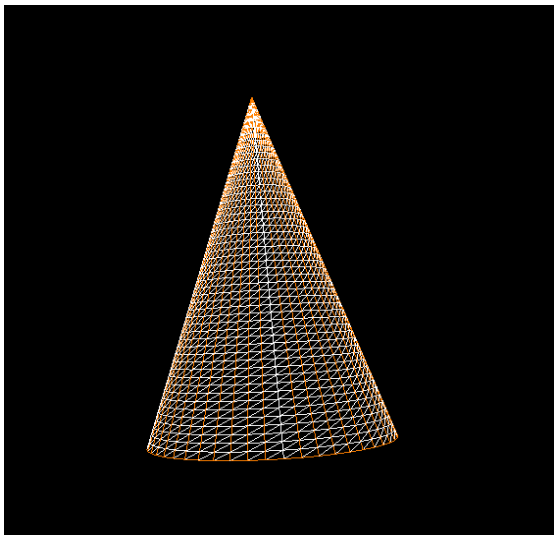


Figura 5: Exemplo de um output do Motor

Após a visualização do modelo requisitado, é possível interagir com os mesmos através do teclado. As teclas A,W,S,D permitem efetuar rotações no modelo apresentado, enquanto as setas permitem efetuar translações nos mesmos. Para além disso, é possível representar os modelos de diferentes maneiras:

- Comando L

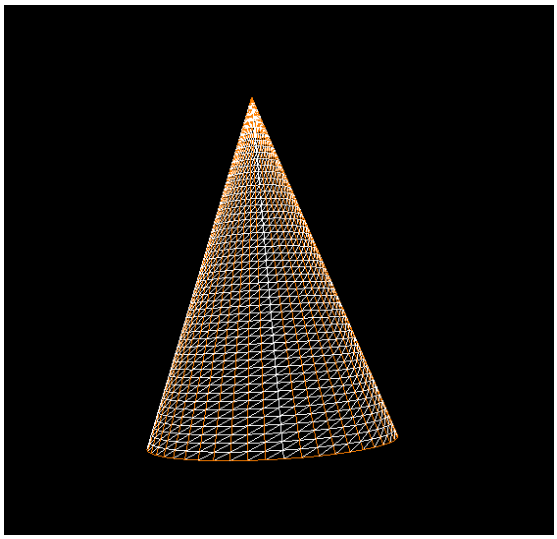


Figura 6: Modelos representados por linhas

-Comando P

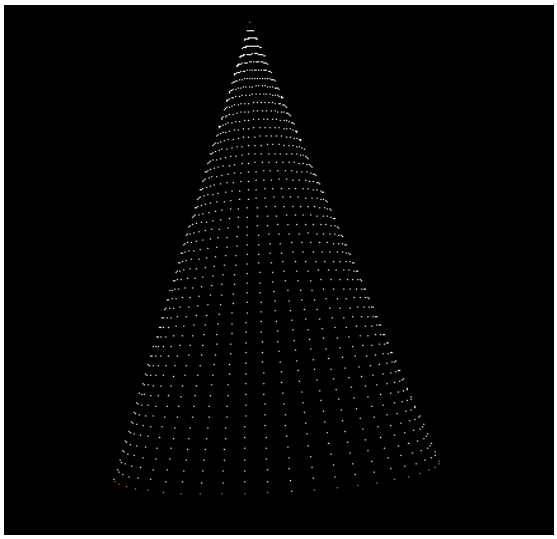


Figura 7: Modelos representados por Pontos

-Comando F

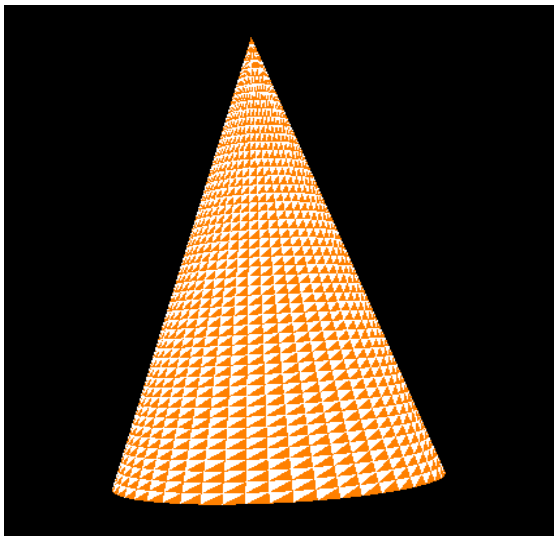


Figura 8: Modelos representados em cores

4 Primitivas Gráficas

4.1 Plano

Um plano é constituido por dois triângulos que partilham, também, dois pontos entre si. Todos os planos gerados estarão contidos no plano **XZ** e centrados na origem, sendo que as suas dimensões serão dadas pelas variáveis X e Z.

4.1.1 Algoritmo

Os triângulos que compõem o plano, para poderem ser observados, têm de ser desenhados por uma certa ordem. Neste caso, como queremos observar a face voltada para cima, o desenho dos pontos que constituem os triângulos deve-se dar no sentido oposto ao dos ponteiros do relógio. Ao mesmo tempo, é necessário ter em atenção que foi requisitado que o plano se encontrasse centrado na origem. Desta maneira, as coordenadas de cada ponto correspondem a metade do valor de **x** e metade do valor de **z**. Por exemplo, se quisermos gerar um plano de dimensões 2, os pontos gerados devem ser *(1,0,-1), (-1,0,-1), (-1,0,1)* para o triângulo da esquerda e *(1,0,1), (1,0,-1), (-1,0,1)* para o triângulo da direita.

4.1.2 Modelo 3D

Figura 9: Plano de dimensão 4

4.2 Caixa

Uma caixa considera-se um prisma de seis faces, onde temos que ter em consideração a existência de mais uma dimensão, sendo as três que a constituem o comprimento (X), a largura (Z) e a altura (Y). No entanto, é necessário considerar, também o número de divisões (camadas).

4.2.1 Algoritmo

O desenvolvimento das faces da caixa segue a mesma teoria do plano, no entanto, como são implementadas camadas, precisamos de um algoritmo mais complexo. Assim, desta vez, para a geração dos pontos dos triângulos, temos que considerar o número de camadas, então o espaçamento entre dois pontos pertencentes ao eixo X é dado pela *divisão do comprimento pelo número de camadas*. Para o espaçamento entre dois pontos do eixo do Z e Y substitiu-se o comprimento pela largura e altura, respetivamente. Considerando a face voltada para a frente, onde o valor de z se mantém sempre constante em todos os pontos, podemos obter a mesma face que, anteriormente, era composta por dois triângulos, agora é composta por várias divisões. Assim, o desenho dos triângulos faz-se da esquerda para a direita, sendo as coordenadas dos vértices calculadas pelo espaçamento e, chegando ao fim da linha incrementa-se a altura. Assim, foi desenvolvida uma fórmula para a aplicação do algoritmo, sendo i (variável que permite a incrementação da altura) menor que j (variável para percorrer eixo do x) e ambos menor que o número de camadas:

1. (xx + (espC * j), yy + (espA * i), z)
2. (xx + (espC * j) + espC, yy + (espA * i), z)
3. (xx + (espC * j) + espC, yy + (espA * i) + espA, z)
4. (xx + (espC * j) + espC, yy + (espA * i) + espA, z)
5. (xx + (espC * j), yy + (espA * i) + espA, z)
6. (xx + (espC * j), yy + (espA * i), z)

Os pontos 1,2,3 formam um triângulo e 4,5,6 formam outro, ambos da face da frente, sendo espC o espaçamento entre os pontos pertencentes ao eixo X, espA o espaçamento entre os pontos do eixo Y.

4.2.2 Modelo 3D

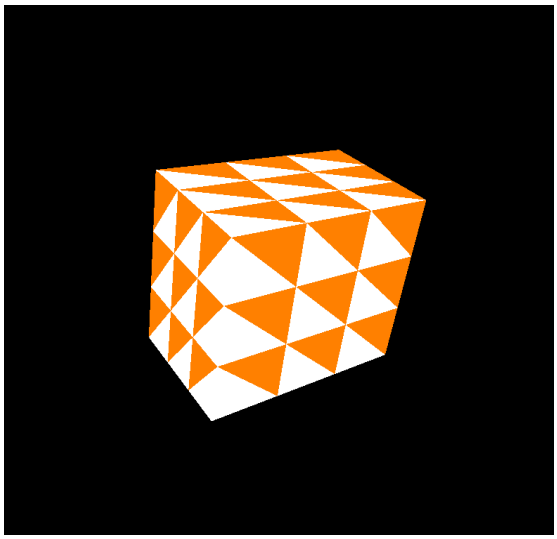


Figura 10: Caixa de tamanho (x,y,z) (3,4,5) e com 3 Camadas

4.3 Esfera

Uma esfera é um sólido geométrico que é formado por uma superficie curva em que os pontos são equidistantes do centro. Para se construir uma esfera os seguintes parâmetros têm que ser inseridos, raio (**r**), número de fatias (**cv**) e número de camadas (**ch**). Quando maior forem os valores de fatias e número de camadas, melhor será a curvatura resultante.

4.3.1 Algoritmo

Para fazer uma esfera é importante estabelecer algumas variáveis importantes .

- Espaçamento entre fatias (**espV**) = **2*π / cv**.
- Espaçamento entre camadas (**espH**) = π **/ ch**.

Após o cálculo destas variáveis podemos começar a desenhar a figura. Para isso utilizamos fórmulas para descobrir o valor das variáveis X,Y e Z nos pontos pertencentes à superficie da esfera. Essas fórmulas utilizam os valores dos ângulos α e β que são respectivamente, o ângulo no plano horizontal XZ e o ângulo no plano vertical YZ.

Aplicando as fórmulas para descobrir quais os quatro pontos pertencentes a um trapézio criado na superficie da esfera conseguimos desenhar os dois triângulos que fazem esse trapézio.

- Fórmula para calcular valor de X: **r * sin(α) * sin(β)**;
- Fórmula para calcular valor de Y: **r * cos(β)**;
- Fórmula para calcular valor de Z : **r * sin(β) * cos(α)**;
- Fórmula para calcular valor de X: **r * sin(α+espV) * sin(β+espH)**;
- Fórmula para calcular valor de Y: **r * cos(β)+espH**;
- Fórmula para calcular valor de Z : **r * sin(β+espH) * cos(α+espV)**;

4.3.2 Modelo 3D

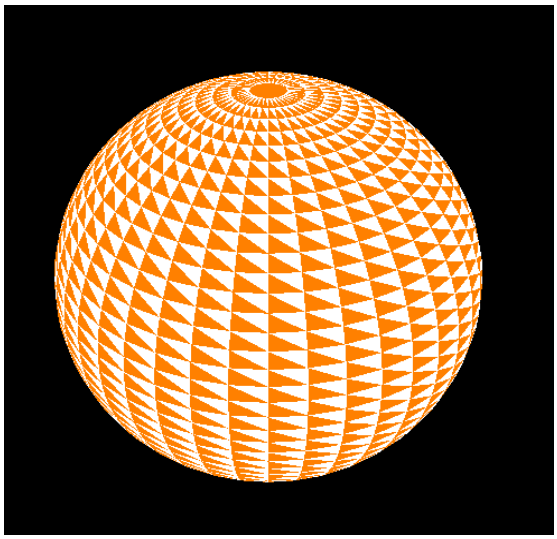


Figura 11: esfera com raio 3, com 40 Fatias e 40 Camadas

4.4 Cone

O cone é um sólido geométrico que é obtido através de uma pirâmide com uma circunferência como base, logo, quanto maior for o número de fatias melhor será a curvatura do cone. É preciso inserir os seguintes parâmetros quando se quer fazer um cone, sendo eles, um raio (**r**), uma altura (**a**), um número de fatias (**slices**) e um número de camadas horizontais (**cH**).

4.4.1 Algoritmo

Tal como na esfera, no cone também é preciso definir algumas variáveis importantes primeiro.

- Espaçamento entre fatias (**espS**) : **2*π / slices**;
- Espaçamento entre camadas (**espH**) : **a/cH**;
- Altura da base do cone (**alt**) : **-(a/2)**;

A construção do cone tem duas fases. Numa primeira fase é feita a circunferência, ou seja, a base do triângulo. Para tal utilizamos as seguintes fórmulas que utilizam um ângulo (α) no plano horizontal XZ.

- Fórmula para calcular valor de X: **r * sin(**α**)**;
- Fórmula para calcular valor de X: **r * sin(**α**+espS)**;
- Fórmula para calcular valor de Y: **alt**;
- Fórmula para calcular valor de Z : **r * cos(**α**)**;
- Fórmula para calcular valor de Z : **r * cos(**α**+espS)**;

Utilizando as fórmulas acima, conseguimos descobrir os pontos para construir os triângulos correspondentes à base. Um dos pontos em todos os triângulos da circunferência é o ponto (0,-(a/2),0) que corresponde ao centro da mesma.

Após isso, chega a fase de construção da parte de cima do cone em cH camadas horizontais. Para tal é feito um ciclo que calcula algumas variáveis em cada iteração que permitem obter informações sobre a camada em que se está a trabalhar. Esse ciclo é inicializado com um inteiro i que começa em zero e em cada iteração soma um.

- Camada de baixo (**camadaB**) : **alt + (i*espH)**;
- Camada de acima (**camadaA**) : **alt + ((i+1) * espH)**;
- Raio da camada de baixo (**raioB**) : **((r/cH)*i)**;
- Raio da camada de cima (**raioA**) : **r - ((r/cH) * (i+1))**;

Depois de saber este valores conseguimos aplicar um ciclo dentro do ciclo em que estamos, que utiliza as fórmulas que são utilizadas na fase de criação da base para descobrir os pontos do trapézio na superficie do cone. Após isso é possível criar os dois triângulos que compõe o trapézio.

4.4.2 Modelo 3D

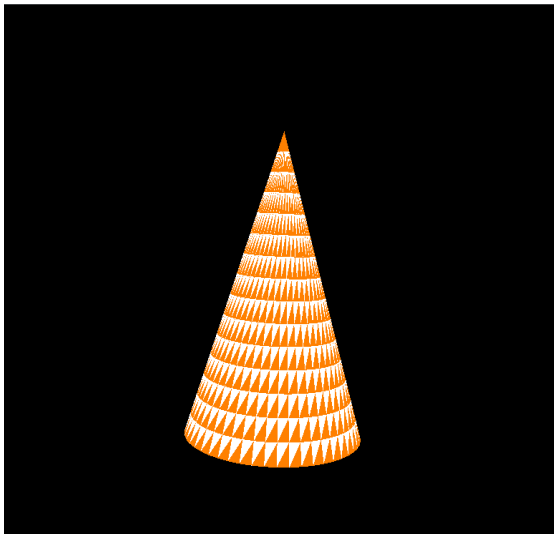


Figura 12: Cone com raio 5, altura 15, 50 fatias e com 15 camadas

4.5 Cilindro

O cilindro é um sólido geométrico que é constituido por duas bases, sendo elas circunferências. Para construir um cilindro é necessário inserir como parâmetro um (**raio**), uma (**altura**), o número de camadas verticais (**slices**) e o número de camadas horizontais (**slicesH**). É de salientar, que quanto maior for o valor de slices melhor será a curvatura do cilindro.

4.5.1 Algoritmo

Tal como no cone, é necessário calcular algumas variáveis importantes.

- Espaçamento entre fatias (**espS**) : **2*π / slices**;
- Espaçamento entre camadas (**espH**) : **altura / slicesH**;
- Altura da base de baixo do cilindro (**alt**) : **-(altura/2)**;

Comparando com a construção do cone, também o cilindro tem duas fases de construção. Na primeira fase são feitas ambas faces do cilindro. São utilizadas as fórmulas que se utilizam também na construção da base do cone, incluindo um ângulo α.

A segunda fase da construção do cilindro diz respeito à construção da superficie lateral. Para isso, é utilizado um ciclo que permite a construção da superficie entre camadas horizontais. Os pontos que fazem essa superficie são descobertos noutro ciclo que utiliza as seguintes fórmulas.

- Fórmula para calcular valor de X: **r * sin**(α);
- Fórmula para calcular próximo valor de X: **r * sin**(α+**espS**);
- Fórmula para calcular valor de Y: **alt**;
- Fórmula para calcular valor de Y acima: **alt+espH**;
- Fórmula para calcular valor de Z : **r * cos**(α);
- Fórmula para calcular próximp valor de Z : **r * cos**(α+**espS**);

Utilizando as fórmulas conseguimos descobrir os pontos que formam um quadrado na superficie lateral. Posteriormente, construindo dois triângulos conseguimos representar esse quadrado em cada iteração do ciclo.

4.5.2 Modelo 3D

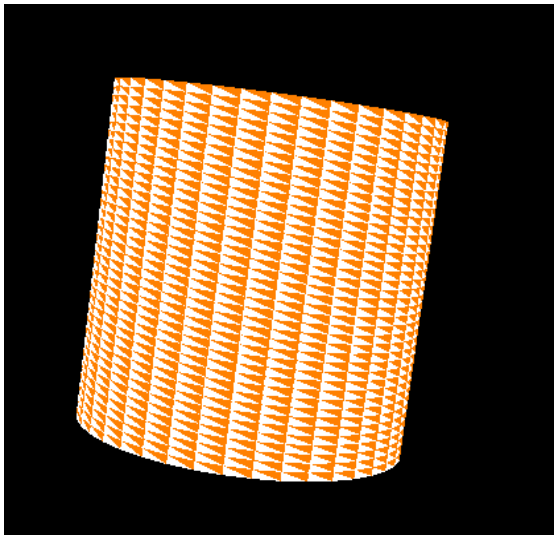


Figura 13: Cilindro com raio 1, altura 2, 40 fatias e com 40 camadas

5 Conteudo Adicional

5.1 TinyXML2

Para fazermos o *parsing* dos ficheiros XML usamos o ***TinyXML2*** de maneira a explorar o conteúdo dos mesmos.

6 Conclusão/Trabalho Futuro

A realização desta primeira fase do projeto é bastante importante devido ao facto de que nos permitiu consolidar o conhecimento em ferramentas de associadas à computação gráfica como o *OpenGL* e o *GLUT*, permitiu-nos adquirir também conhecimentos da linguagem *C++*, esta que é essencial para o desenvolvimento deste trabalho. Para além disso conseguimos ter uma noção dos algoritmos que estão associados à criação de primitivas gráficas. Deste modo esperamos que o trabalho realizado na primeira fase do trabalho, de algum modo, nos ajude na a realização das consequentes fases do projeto.