Computação Gráfica

1ª Fase – Primitivas Gráficas Simples

André Geraldes

Patrícia Barros 67665

Sandra Ferreira 67709

Conteúdo

[Introdução 2](#_Toc414522243)

[Desenvolvimento 4](#_Toc414522244)

[Gerador 4](#_Toc414522245)

[Implementação do Plano 4](#_Toc414522246)

[Resultado 4](#_Toc414522247)

[Implementação do Paralelepípedo 4](#_Toc414522248)

[Resultado 5](#_Toc414522249)

[Implementação do Cone 5](#_Toc414522250)

[Implementação da Esfera 5](#_Toc414522251)

[Resultado 5](#_Toc414522252)

[Implementação do Cilindro 5](#_Toc414522253)

[Resultado 6](#_Toc414522254)

[Motor 3D 6](#_Toc414522255)

[Conclusão 7](#_Toc414522256)

Introdução

Este trabalho foi desenvolvido no âmbito da Unidade Curricular de Computação Gráfica, pertencente ao plano de estudos do 3º ano da licenciatura em Engenharia Informática.

Este projeto será constituído por 4 fases distintas com o objetivo final de criar um *motor 3D*. Nesta primeira fase foi-nos proposto o desenvolvimento de uma aplicação que crie um ficheiro onde irão estar armazenados os triângulos necessários ao desenho de primitivas de sólidos geométricos e também o desenvolvimento de um pequeno motor que lê de um ficheiro XML os triângulos previamente gerados e constrói a respetiva primitiva.

Desenvolvimento

Gerador

A primeira etapa do desenvolvimento do nosso projeto foi a aplicação “gerador” que recebendo o nome do sólido a desenhar e alguns argumentos que o caracterizam (variáveis consoante o sólido) gera os pontos necessários para o seu desenho através de triângulos. Esses pontos são guardados num ficheiro ordenados pela ordem correta para os triângulos serem desenhados como pretendido.

# Implementação do Plano

Para a implementação do plano o gerador recebe 4 argumentos: o comprimento, a largura, o número de camadas horizontais e o número de camadas verticais.

O plano é desenhado a partir do seu canto superior esquerdo, cujas coordenadas são facilmente determinadas através do seu comprimento e largura. A partir desse ponto é possível descobrir todos os outros da seguinte forma: o ponto seguinte terá um deslocamento vertical em relação ao anterior, sendo esse deslocamento dado pela largura dividida pelo número de camadas horizontais. O ponto seguinte terá um deslocamento horizontal em relação ao inicial. Da mesma forma esse deslocamento é dado pelo comprimento dividido pelo número de camadas verticais. Assim temos já o primeiro triângulo formado. O segundo terá estes últimos dois pontos e ainda um outro que é dado pelo ponto inicial com um deslocamento tanto vertical como horizontal calculado da mesma forma que os anteriores. Temos assim dois triângulos formados. A partir daí descobrem-se todos os outros iterando este algoritmo por toda a largura e comprimento do plano.

## Resultado

# Implementação do Paralelepípedo

Para a implementação do paralelepípedo o gerador recebe 6 argumentos: comprimento, largura, altura, número de camadas verticais, número de camadas horizontais e número de camadas longitudinais.

As bases do paralelepípedo são geradas da mesma forma que o plano, apenas com uma translação no eixo do y de tantas unidades quanta a altura do sólido a dividir por dois (para ficar centrado), uma translação no sentido negativo e outra no sentido positivo.

48 completa aqui que eu não sei.

## Resultado

# Implementação do Cone

Para o desenho do cone são utilizados 4 argumentos: o raio, a altura, as camadas verticais e as camadas horizontais.

Para desenhar a base iteramos sobre o número de camadas verticais, utilizando a razão entre e o número de camadas verticais para incrementar um ângulo , desde que este toma o valor de 0 até completar o círculo. A base foi desenhada com um valor de y correspondente a para que o cone ficasse centrado em relação à origem.

O esquema seguinte é ilustrativo da técnica usada para encontrar os pontos relativos à base: (vou fazer um esqueminha à mão e digitalizar e ponho aqui sexta)

Resultado

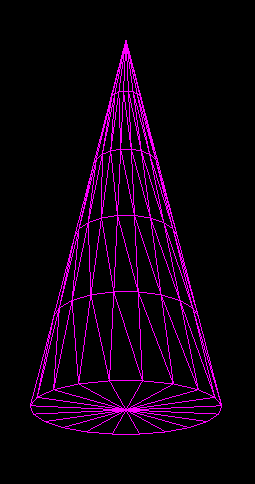


Figura 1 - Implementação do cone

# Implementação da Esfera

## Resultado

# Implementação do Cilindro

A implementação do cilindro foi feita de forma semelhante ao cone, com a exceção de que o raio não sofre nenhuma diminuição ao longo das iterações e em vez de uma só base tem duas bases simétricas em relação ao eixo do y.

## Resultado

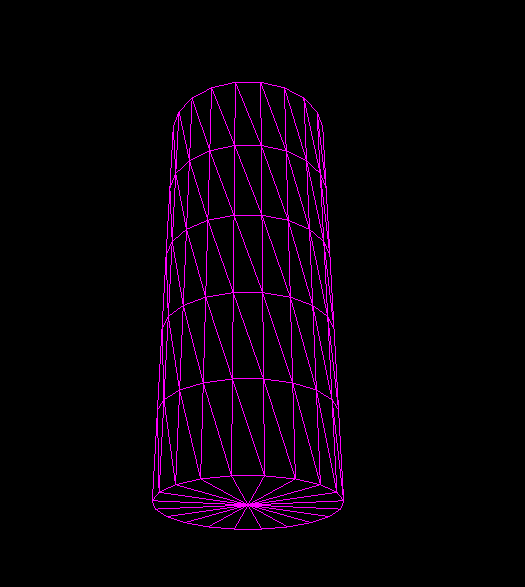


Figura 2 - Visualização do cilindro

Motor 3D

O motor 3D foi criado para ler os pontos gerados e guardados previamente pelo gerador num ficheiro e através deles desenhar os sólidos correspondentes.

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

Acrescentamos rotações da câmara através das teclas especiais e do rato e também translações do sólido através das teclas normais para melhor visualização do trabalho realizado. Temos ainda disponível um menu quando é efetuado um clique no botão direito do rato que permite mudar o modo de apresentação dos sólidos (linhas, sólido ou pontos).

Para guardar os pontos lidos foi criada uma estrutura **Ponto** constituída por três *doubles* para guardar cada valor das coordenadas e um vetor **Pontos** para guardar vários elementos da estrutura anterior. Foi feita uma função **readFile** para fazer a leitura e *parsing* do ficheiro e guardar os pontos lidos nas estruturas mencionadas anteriormente.

O desenho das primitivas é feito iterando sobre o vetor **Pontos** e através do **GL\_TRIANGLES** e do **glVertex3f** são desenhados todos os triângulos cujos pontos estavam guardados no ficheiro lido. Os pontos gravados no ficheiro já se encontram devidamente ordenados de forma a desenhar o sólido pretendido logo isso não é algo com que tivéssemos que nos preocupar nesta fase da construção do motor.

Conclusão