Relatório de Computação Gráfica

Fase 1- Primitivas Gráficas

Uma imagem com objeto

Descrição gerada automaticamente

2019/2020

Carlos Afonso A82529, Gonçalo Nogueira A86617, Luís Fernandes A76712, Pedro Fernandes A84313

Índice

Introdução……………………………………………………………3

Estruturação do projeto……………………………………….4

CMakeLists…………………………………………………………..4

Gerador………………………………………………………………..5

Motor……………………………………………………………………6

Plano………………………………………………………………….10

Caixa…………..……………………………………………………..11

Esfera………….……….…………………………………………….12

Cone……………………………………………………………………14

TinyXML………………………………………………………………16

Conclusão……………………………………………………………17

Introdução

No âmbito da cadeira de Computação Gráfica foi-nos proposto, na primeira fase, da conceção e desenvolvimento de um gerador de vértices de algumas primitivas gráficas, tais como um plano, caixa, esfera e cone.

Para além do gerador foi também pedida a implementação de um motor que através da leitura de um ficheiro XML consiga desenhar os vértices das primitivas gráficas gerados anteriormente pelo utilizador.

Neste relatório faremos uma descrição detalhada das etapas do trabalho, explicando as decisões que efetuamos bem como os algoritmos para a criação dos vértices de cada primitiva gráfica.

Estruturação do projeto

Para este projeto foi decidido a divisão do trabalho em duas aplicações principais, o motor e o gerador e o seu código encontra-se em /src.

Para além disso os ficheiros XML estão em /scenes e os ficheiros .3d com as coordenadas dos pontos das primitivas gráficas estão em /models. Todos estes ficheiros encontram-se já previamente definidos contudo, ficheiros .3d podem ser gerados desde que o nome seja o que está configurado para ser lido no motor no ficheiro XML.

CMakeLists

O cmakelists cria a makefile para criar a aplicação gerador com o /src/generator.c++ e a makefile para criar motor com /src/engine.c++ e para usar as bibliotecas que estão em /include

Gerador

Esta aplicação tem como objetivo a criação de um ficheiro com o nome fornecido pelo utilizador onde constam os vários pontos constituintes das diferentes primitivas gráficas e conforme os parâmetros escolhidos.

Depois de compilado e gerado o executável, para utilizar esta aplicação ao introduzir o comando “./generator box 3 3 4 box.3d”, por exemplo, o gerador cria um ficheiro ‘box.3d’ que contém as coordenadas de todos os pontos utilizados para construir graficamente a figura geométrica pedida.

Motor

Esta aplicação tem como objetivo receber e ler um ficheiro XML, que indicam a localização do ficheiro criado pelo gerador anteriormente, fazendo parser desses ficheiros gerados e construindo os modelos que esses ficheiros representam.

Além disso, foi utilizada a biblioteca Tinyxml para ser feito o parser do ficheiro XML. De salientar que a criação deste ficheiro é realizada pelo utilizador e os ficheiros presentes neste foram previamente gerados.

Nesta aplicação foram também criados vários comandos para a interação do utilizador com os modelos representados, tais como a forma do objeto ser representado, rotações e translações.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 1- Ficheiro XML

O output do motor ao ler o ficheiro XML será a representação GLUT de uma caixa.

Uma imagem com objeto

Descrição gerada automaticamente

Figura 2 - Output do Motor

Após a apresentação do output é possível também a interação com o objeto através de vários comandos criados e explicados abaixo.

-Comando F

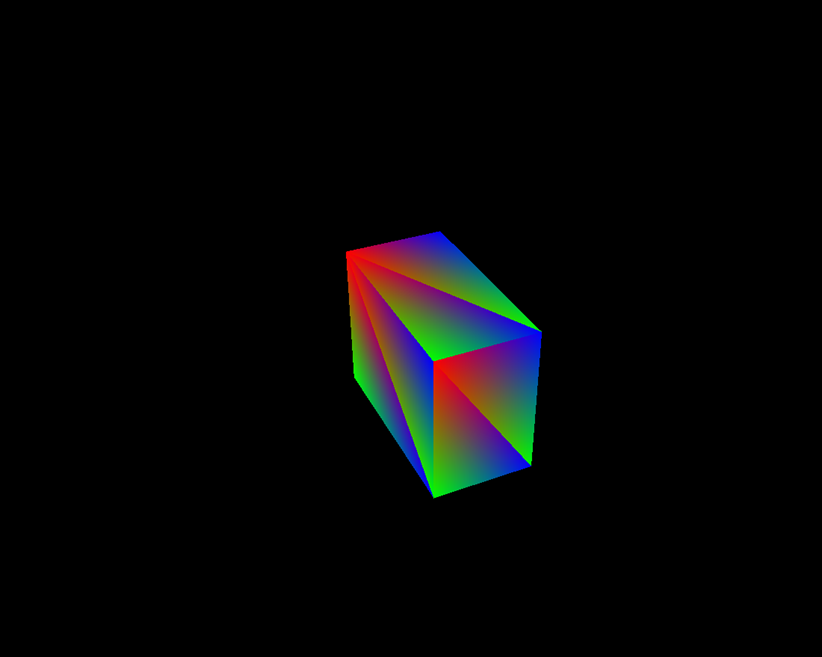


Figura 3 - Modelo representado a cores

- Comando L

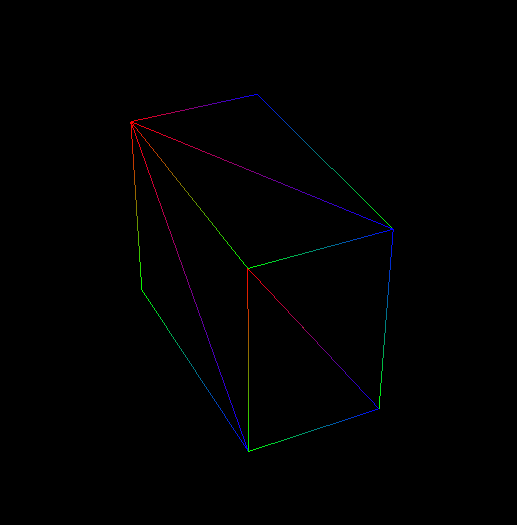


Figura 4 - Modelo representado por linhas

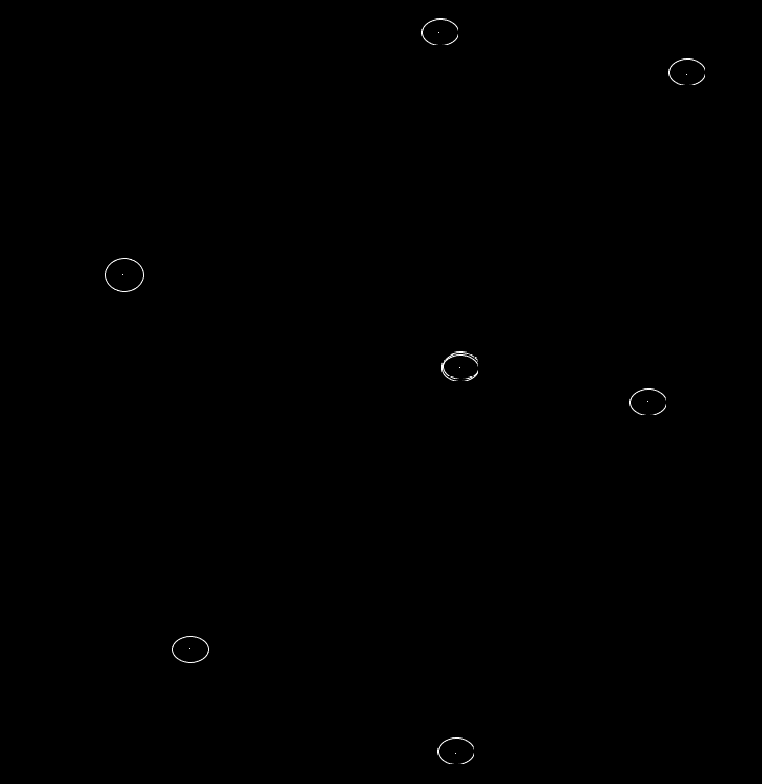
- Comando P

Figura 5- Modelo representado por pontos

Os três comandos mencionados acima são relativos ás diferentes maneiras de representar os modelos, no entanto, podemos também interagir com os mesmos através das teclas A e D que permitem rodar o modelo, as setas permitem efetuar translações em X e em Z e as teclas W e S em Y e as teclas ‘–‘ e ‘+’ permitem fazer scale .

Primitivas gráficas

**PLANO:**

Um plano é constituído por dois triângulos que contêm dois pontos em comum. Todos os planos gerados estarão no plano XZ e centrados na origem e as suas dimensões serão dadas pelos argumentos dados á aplicação gerador.

Para observarmos a face voltada para cima, o desenho sequencial dos pontos que constituem o triângulo deve seguir a regra da mão direita e por isso devem ser desenhados no sentido contrário dos ponteiros do relógio.

De salientar também que como é pedido que o plano seja centrado na origem os valores das coordenadas de cada ponto serão metade do valor pretendido das dimensões X e Z.

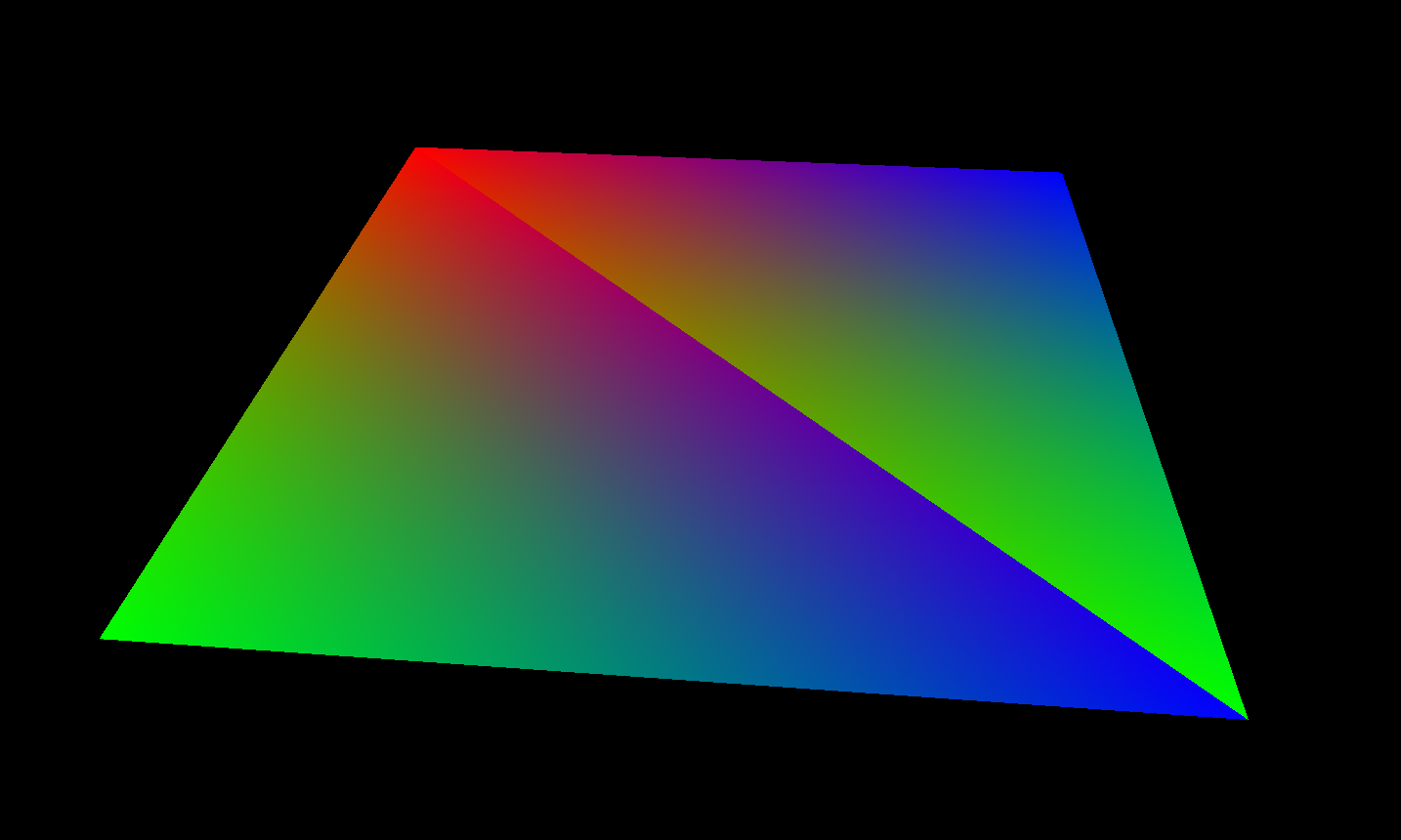
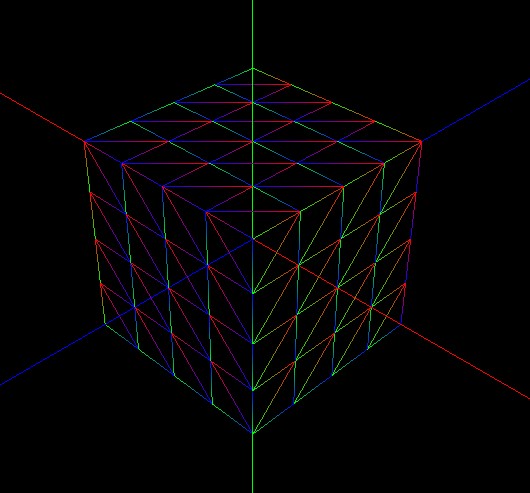


Figura 6 - Modelo do plano

**CAIXA:**

A caixa será feita por dois triângulos por face se só forem passados os parâmetros (x,y,z), mas se for passado outro parâmetro (div) serão gerados 2 triângulos por cada divisão da face. Neste último caso cada face terá então div\*div\*2 triângulos. Para cada face, os triângulos começam a ser feitos nos menores valores dos eixos que não são fixos e para cada divisão são feitos os 2 triângulos com os pontos por ordem contrária aos ponteiros do relógio.



**ESFERA:**

Para o cálculo da são necessários três parâmetros sendo eles o raio, as slices (camadas na vertical) e stacks (camadas na horizontal).

Para fazer uma esfera é importante determinar algumas variáveis como a sliceSize calculada através de 2\*PI/numero de slice e a stackSize obtida através de PI/numero de stacks.

Depois destas variáveis temos de calcular as coordenadas dos quatro pontos originados pela interseção das stacks com as slices através de fórmulas que usam o angulo alpha e beta que são respetivamente o ângulo no plano horizontal XY e o ângulo no plano horizontal XZ.

Sendo alpha=0.0f e beta=PI/-2 na figura abaixo temos a ordem da leitura de coordenadas de forma a seguir a regra da mão direita.

Uma imagem com captura de ecrã

Descrição gerada automaticamenteUma imagem com captura de ecrã

Descrição gerada automaticamente

Uma imagem com transporte, aeronaves, balão

Descrição gerada automaticamente

Figura 7- Modelo da esfera

**CONE:**

Para o cálculo do cone são necessários quatros parâmetros sendo eles o raio, a altura, as slices (camadas na vertical) e stacks (camadas na horizontal).

O cálculo dos pontos consiste em duas partes, a primeira sendo o cálculo dos pontos da circunferência base divida nas slices pretendidas e a segunda os pontos da parte lateral do cone.

Para a base do cone temos apenas de calcular as coordenadas em X e Y dos pontos visto que Y é igual para todos e não varia. A base será composta por vários triângulos unidos ao centro e determinamos o ângulo alpha através da divisão de 2\*PI pelo número de slices e multiplicando esse valor pelo número da slice que estamos a construir. Somando esse alpha com o tamanho de cada slice chegamos ao próximo ponto do círculo para desenhar novamente o triângulo.

Para a superfície lateral é preciso fazer cada camada horizontal de cada vez e definir a altura inferior e superior da camada em questão. O raio do círculo superior calcula-se através de:

***((Stacks totais – nrº próxima Stack) / Stacks totais) \* raio do cone***

E o raio do círculo inferior através da seguinte fórmula:

***((Stacks totais – nrº da Stack atual) / Stacks totais) \* raio do cone***

Dentro de cada stack vão se calculando as interseções com as slices que originam quatro pontos, dois na stack i e por isso contido no círculo e outros dois na stack i+1 contido no círculo superior. A altura de cada stack calcula-se através da divisão da altura do cone sobre o número de stacks pretendido.

Depois de determinadas as coordenadas dos pontos é só garantir que a ordem está correta para o OpenGL desenhar corretamente a superfície lateral.

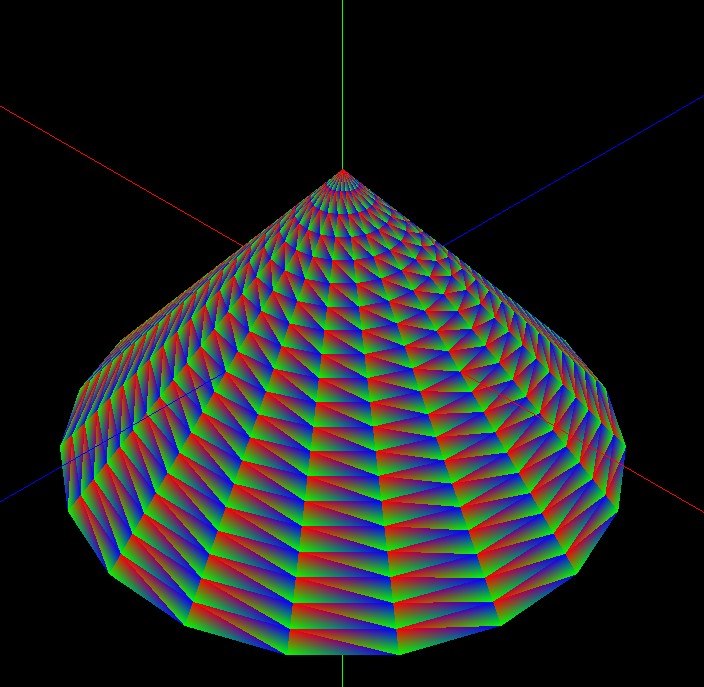


Figura 8 -Modelo do cone

TinyXML

Para realizarmos parsing do ficheiro XML utilizamos esta biblioteca tirando partido das ferramentas disponibilizadas pela própria biblioteca para obter a localização do ficheiro 3d gerada previamente.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 9 - Função utilizada com TinyXML para o parsing do ficheiro XML

Conclusão

Com a realização desta primeira fase do trabalho pratico foi permitido a todo o grupo a consolidação de conhecimentos acerca das ferramentas associadas á computação gráfica como o OpenGL e a biblioteca do GLUT, permitiu-nos também um maior contacto e aprofundamento do conhecimento com a linguagem c++ utilizada nesta disciplina e que é essencial para a mesma.

Para além disso conseguimos também adquirir a noção dos algoritmos que estão relacionados com a criação de primitivas gráficas. Desta forma, esperamos que todos estes conhecimentos adquiridos e reforçados sejam uteis para as próximas fases do projeto.