Computação Gráfica

Primeira Fase: Primitivas Gráficas

Ciências da Computação

Bruna Alvarães	Igor Braga	Guilherme Santos	Ricardo Teixeira
(A84729)	(A85965)	(A85115)	(A85688)

14 de março de 2021

Resumo

No seguimento das aulas deste semestre, o seguinte trabalho elabora um motor 3D e fornece exemplos de figuras geométricas em 3D para mostrar o potencial deste motor.

Conteúdo

Capítulo 1

Introdução

Nesta primeira fase do trabalho foi criadas duas aplicações, um gerador para gerar ficheiros com a informação dos vértices de primitivas gráficas e um motor, para ler o ficheiro com esta informação e desenhar as primitivas no ecrã.

1.1 Estrutura do Relatório

Neste relatório começamos por analisar o enunciado da primeira fase, descrevemos tanto o gerador como o motor e quais são os seus requisitos. De seguida falamos da nossa abordagem às várias figuras. No próximo capítulo discutimos os problemas de implementação, as ferramentas utilizadas e os aspectos do gerador e do motor. Por fim apresentamos os resultados obtidos e apresentamos as nossas conclusões.

Capítulo 2

Análise e Especificação

2.1 Descrição e Enunciado

2.1.1 Gerador

O gerador cria os ficheiros para serem usados pelo motor. Recebe os parâmetros que são necessários para a criação do modelo e o ficheiro destino onde os vértices serão guardados. Nesta fase, o gerador é capaz de criar as primitivas para planos, caixas, esferas e cones e guardar os vértices resultantes num ficheiro dado com a extensão .3d . Neste momento os ficheiros contêm apenas os vértices das figuras.

2.1.2 Motor

O motor recebe um ficheiro escrito em XML, que contem o nome dos ficheiro que queremos carregar, e usando a ferramenta *tinyXML2* para perceber quais os ficheiros que queremos, carrega esses ficheiros para mostra-los no ecrã.

2.2 Especificação do Requisitos

2.2.1 Gerador

O gerador consegue gerar as seguintes primitivas:
Plano (Que desenha um quadrado centrado na origem)
Caixa (Que recebe as dimensões X,Y e Z e o número de divisões)
Esfera (Que recebe o raio, as slices e as stacks)
Cone (Que recebe o raio da base, a altura, as slices e as stacks)

2.2.2 Motor

O motor lê um ficheiro XML que serve de configuração , identifica os ficheiros a abrir e desenha a figura no ecrã.

Capítulo 3

Concepção/desenho da Resolução

3.1 Abordagem às figuras 3D

3.1.1 Plano

Uma vez que o centro do plano se situa na origem, para dados valores de X e Z, utilizaremos a metade destes valores para os vértices de cada triângulo Para a construção do plano utilizámos 2 triângulos. Fixamos o valor do Y e alternamos, de positivo para negativo e vice-versa, os valores de X e Z. Esta figura podia também ser construída com slices, aumentado assim o número de triângulos mas mantendo a mesma estratégia de construção.

3.1.2 Caixa

Semelhante à construção do plano, mas desta vez com 6 figuras e com divisões nas arestas que nós tratamos como slices ,em que a origem encontra-se no centro da caixa. Para a face de cima (respetivamente face de baixo), fixamos o Y positivo (respetivamente Y negativo) e alternamos, de positivo para negativo e vice-versa, o valor de Z e começando do valor negativo de X e somando um certo valor, valor este que é o resultado de (valor de X)/slices, até termos todos os triângulos pretendidos. Para as restantes faces, seguimos a mesma estratégia. Uma coordenada está fixa, uma alterna e a outra começa no negativo e vamos somando um certo valor ((valor inteiro)/slices), até termos todos os triângulos pretendidos.

Todas as faces construídas têm a face de fora pintada.

3.1.3 Esfera

Para a construção da esfera, temos dois ângulos (alpha e beta) que determinam respetivamente a longitude (slices) e a latitude (stacks). Fazendo ((2 * pi)/slices) e (pi/stacks) obtemos a variação dos ângulos alpha e beta e, a cada iteração dos ciclos que percorrem todos os valores destes ângulos, obtemos quatro pontos (extremos do sector limitado por duas stacks e duas slices consecutivas) p1,p2,p3,p4 (as coordenadas destes pontos são obtidas através das coordenadas esféricas dependentes de alpha e beta), que usamos para desenhar dois triangulos (como indicado no código), desenhando então o sector.

Para os sectores entre as duas primeiras e as duas últimas stacks, apenas necessitamos de 3 pontos, pois a primeira e a ultima stack são formadas por um só ponto. Estas excepções são desenhadas respetivamente antes e depois dos ciclos que desenham a restante esfera.

3.1.4 Cone

Para representar o cone começamos por escrever os vértices que pertencem ao círculo da base (tantos quanto o número de *slices*). Depois para representar a superfície cónica usamos triângulos consecutivos de maneira a completar as *stacks*, primeiro com a base em baixo e depois com a base em cima. Por fim para completar a ultima *stack*, usamos triângulos com base em baixo que convergem no vértice do cone

Capítulo 4

Codificação e Testes

4.1 Decisões e Problemas de Implementação

Numa primeira fase tivemos alguma dificuldade a entender o funcionamento da ferramenta *tinyXML2* para o motor mas após várias experiências, conseguimos perceber o seu funcionamento. Em termos da criação das figuras e da escrita dos seus vértices para os ficheiros, não sentimos grandes dificuldades pois achámos que era algo muito similar àquilo que foi apresentado nas aulas em termos da criação das primitivas.

4.2 Código e Ferramentas Utilizadas

Recorrendo à linguagem C++, desenvolvemos um motor que usa *tinyXML2* e reconhece os ficheiros com a extensão .3D . Estes foram criados pelo gerador que também foi desenvolvido usando C++.

4.3 Gerador

O gerador tem como função primária a escrita para um dado ficheiro dos vértices que irão ser representados mais tarde numa dada cena. Assim, a geração destes vértices é feita da mesma maneira que seria feita se estes estivessem para ser desenhados directamente no ecrã, e em vez de serem utilizados pelas funções do OpenGL, são escritos para ficheiro.

4.3.1 Escrita para o Ficheiro

A escrita para um ficheiro é bastante simples devido a simplicidade de abertura de ficheiros na linguagem C++ , utilizando a função ofstream. Após a abertura dos ficheiros, são utilizados os operadores de inserção '¡¡' para a escrita das coordenadas de cada vértice no ficheiro. Após a escrita de todos os vértices, a stream do ficheiro é fechada para que os dados em cache sejam escritos.

4.4 Motor

O motor tem como função primária a representação gráfica dos vértices criados pelo gerador. Para isto, o motor utiliza a ferramenta TinyXML2 para a leitura de um ficheiro de configuração do tipo XML que contém o nome dos ficheiros que irão ser lidos. Os vértices dos ficheiros são guardados num vector todos

de uma só vez para que a leitura do ficheiro não seja repetida todas as frames e não tenha impacto na performance do motor.

4.4.1 Leitura do Ficheiro XML

Utilizando o TinyXML2 para a leitura de ficheiros em formato XML, o reconhecimento de quais primitivas guardadas em ficheiro devem ser mostradas em ecrã tornou-se bastante simples. Iremos primeiramente buscar a tag que serve de raíz do ficheiro, que no nosso caso será "model", e com a indicação desta raíz iremos buscar o primeiro elemento "path"para que se possa entrar num loop que permita a leitura de todos os elementos "path"guardados em "model", guardando o nome dos ficheiros num vector de strings.

4.4.2 Leitura dos Ficheiros .3D

Com o vector que contém o nome dos ficheiros a serem lidos preenchido, podemos passar a abertura dos ficheiros .3D . Percorremos os ficheiros linha a linha, e guardamos num vector as coordenadas de cada vértice. Isto garante que a leitura dos pontos é feita da memória e não de um ficheiro, que aumenta significativamente a nossa performance.

4.4.3 Câmara

Foi decidido que se deveria utilizar uma câmara em primeira pessoa pois permite uma melhor visualização dos objectos no espaço tridimensional. Utilizando funções do Glut para processar input do utilizador, a nossa câmera utiliza o rato para alterar a direção em que a câmera aponta e as setas para alterar a posição da câmera no espaço.

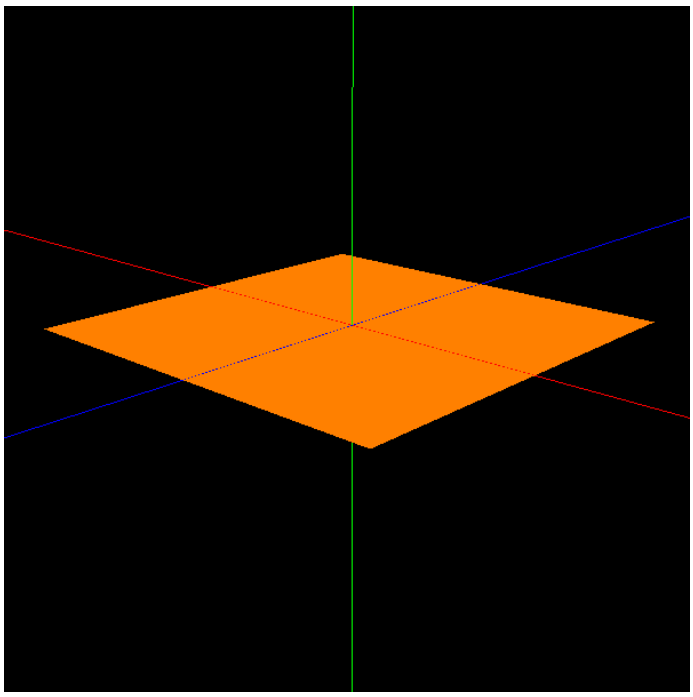
4.4.4 Contador de FPS

Para medir a performance do motor foi utilizado o código das aulas para um contador de FPS.

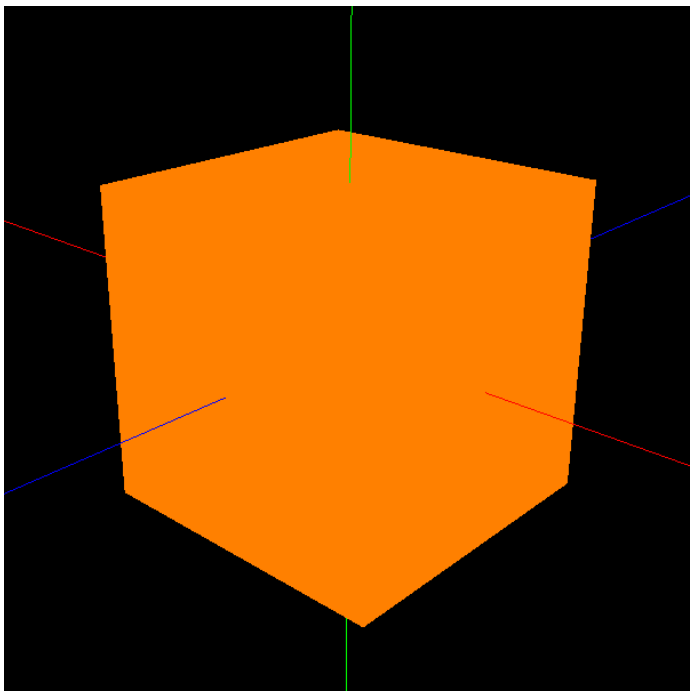
4.5 Testes realizados e Resultados

Estes foram os resultados obtidos após correr no motor os ficheiros criados pelo gerador:

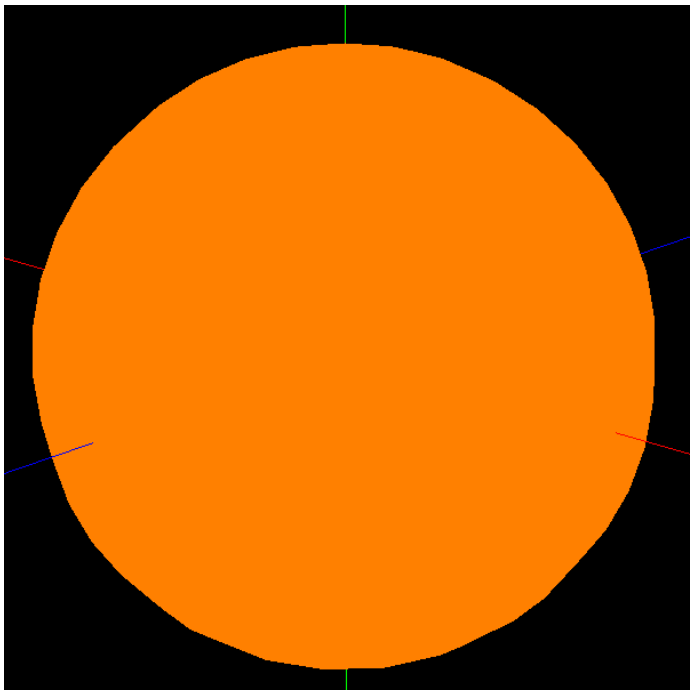
4.5.1 Plano



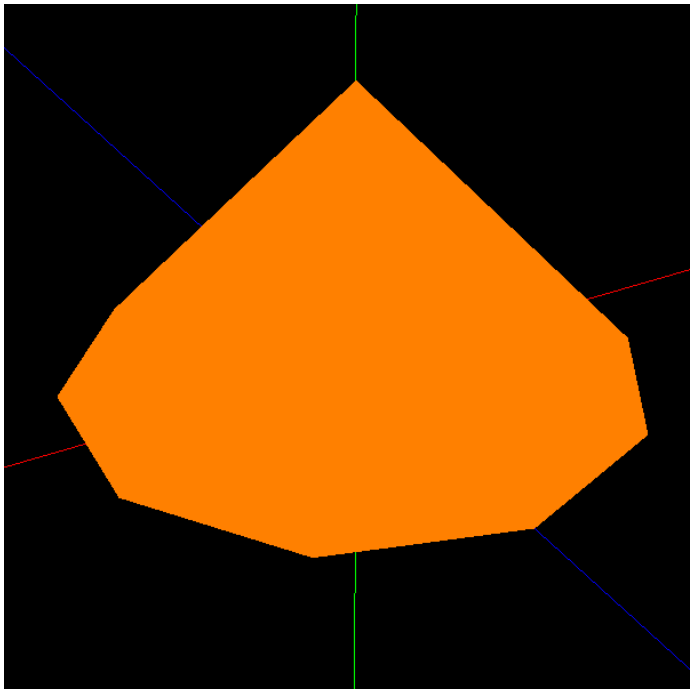
4.5.2 Caixa



4.5.3 Esfera



4.5.4 Cone



Capítulo 5

Conclusão

A realização desta primeira fase ajudou-nos a por em prática os conhecimentos que aprendemos nas aulas, e também ajudou-nos a aprofundar o nosso conhecimento na linguagem C++.