Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Computação Gráfica

Ano Letivo de 2022/2023

Grupo 24

André Castro Alves (a95033) Cláudio Alexandre Freitas Bessa (a97063)
José Fernando Monteiro Martins (a97903)

4 de junho de 2023

Indice

1 Introdução

O documento apresentado visa apresentar as metodologias praticadas e processos utilizados durante a quarta e útlima fase do projeto que tem vindo a ser realizado no âmbito da UC de Computação Gráfica.

1.1 Generator

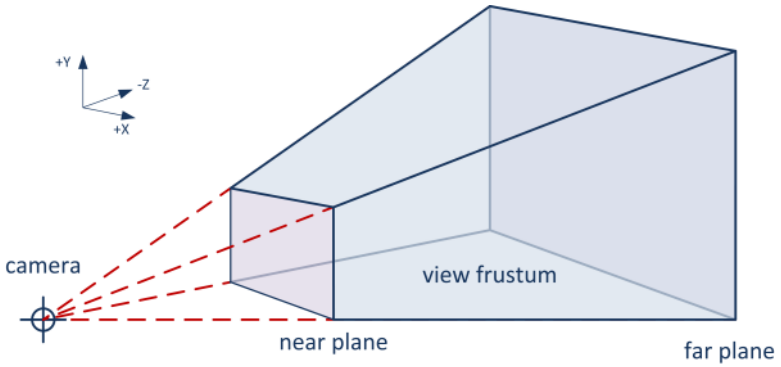
De forma a implementar as luzes e as texturas é necessario efetuar a criação de normais e pontos para textura às nossas primitavas.

1.2 Engine

Por sua vez, o *engine* sofre alterações, acionando as texturas. Foram adicionadas as texturas de maneira a tornar o projeto *visual appealing* e mais realista, de forma a ir de encontro as necessidades do mesmo. Foi da mesma forma iniciado a implementação do *frustum culling* de modo a proporcionar uma melhor visibilidade para o utilizador.

2 Frustum

A noção de *frustum* é, em linguagem corrente, o que nós conseguimos ver. Mais especificamente o que a nossa câmara vai conseguir captar, sendo que esta tem um limite maximo de visão, conhecido como *far plane*, tendo também um limite mínimo, conhecido como *near plane*.



Tudo que se vai encontrar dentro desta pirâmide vai ser desenhado.
Como se pode ver no exemplo abaixo, a visão do snowman fica deturpada pela proximidade que este tem, ou seja já sai da piramide de visão em que se encontrava e passa para antes do near plane, afetando assim a visualização do mesmo.

3 Primitivas

As texturas e as luzes eram um dos requsitos a implementar neste fase. Para este ser cumprido teve de se alterar o *generator* de forma a aceitar toda a informação necessaria para a geração das figuras. Os métodos de aplicação a cada uma das figuras difere, caso este seja um plano, cubo, teapot, etc.

3.1 Plano

Para gerar as normais apenas é necessário gerar 4 uma vez que um plano é composto por 2 triângulos e portanto 4 vértices. Estando o plano virado para cima os seus valores tomam propriedades como (0,1,0).

3.2 Cubo

No caso do cubo o processo a implementação das normais resultou em alguns problemas iniciais uma vez que existem 3 normais possíveis para cada vértice, um por cada face. Cada ponto mapeia 3 normais com direções perpendiculares a cada um dos 3 planos sendo os seus valores, (1,0,0) , (0,1,0) e (0,0,1) não necessariamente por esta ordem. Havendo no total 24 pontos (8 x 3).

3.3 Esfera

Umva vez que a esfera possui uma infinidade de normais, a forma mais simples de os calcular é simplesmente efetuar uma normalização do vértice da superfície e as coordenadas do ponto central.

3.4 Cone

Para calcular as normais da base do cone simplemente tomou-se uma postura igual à dos planos visto que todos os vértices possuem normais iguais.

Já para descobrir as normais relativas aos restantes vértices da primitiva é necessário o vértice da *stack* superior e os dois vértices, um da *slice* seguinte e outro da *slice anterior*. Isto para quê? Isto para conseguir pegar em dois dos vetores, visto que existem três, e efetuar o produto externo entre os mesmos dois a dois de forma a calcular a normal do plano definido. Conseguindo obter a normal após normalização do vetor resultante de descobrir a equação do plano tangente.

3.5 Torus

No desenvolvimento das normais da *torus* é necessário ter cuidado ao caso especial de apenas possuír dois "planos", i.e. 2 lados. Devido a isso é tratado da mesma forma do plano, caso contrário resultaria em problemas de iluminação.

No caso geral, as normais da *torus* são calculadas da mesma forma que na esfera, sendo a distância nromalizada entre o vértice e o ponto central.

3.6 Patch

Por últimos, nos *patches* de *bezier* foi onde se desenvolveu uma componente mais matemática. Para calcular as normais são calculados os vértices e as derivadas parciais sobre as variáveis u e v, que originam vetores em 2 dimensões, horizontal e vertical. Depois calculamos o produto externo destes dois vetores e seguidamente é efetuado uma normalização. Isto claramente para cada vértice do *patch*.

4 Dificuldades

As nossas maiores dificuldades residiram nesta última fase. Sendo que tivemos de começar de novo mais de 3 vezes para tentar que esta parte ficasse feita, mas os nossos problemas foram mudando de parte para parte, mas nunca desapareceram por completo. Ficamos com a ideia final que o problema deveria residir numa função que tinhamos implementado, a *addFile*, sendo que se o problema fosse realmente nesta, tanto as texturas como as luzes iriam funcionar. Em suma, nesta última fase ficou por fazer as texturas, cores e luzes na parte do engine. Sendo que as normais foram implementadas, mas não testadas, sendo que não o conseguiamos fazer. Quanto à parte 3, apesar de termos as curvas de *Bezier* implementadas, mas *catmull-Rom* também não funciona, mais uma vez não conseguimos identificar o erro, mas as curvas não são geradas, e os nossos planetas não giram em torno de nenhum eixo, sendo por isso estaticos.

5 Conclusão

Sendo esta a última fase de entrega do trabalho prático, nas fases anteriores conseguimos concluir quase todos os requisitos, sendo que nesta não conseguimos concluir todos os que eram expectáveis. Durante a realização interceptamos nos com múltiplos erros, não conseguindo ter uma perceção total do seu motivo não nos permitindo resolver os mesmos a tempo. Outras coisas, e.g. como as luzes, que no fim da nossa implementação obtiamos erros sem qualquer conotação por parte do interpretador. Outros fatores consideramos que estejam também implementados, na totalidade ou perto da mesma mas não conseguimos testar 100%. Quanto ao *frustum culling* começamos a implementação, mas não a conseguimos terminar.