Fractais 3D

Álvaro Martins 72447 & André Rodrigues 73152

***Resumo* - O presente artigo apresenta fractais em 3D, tal como a sua análise e implementação.**

Usando uma abordagem que se pretende pedagógica, o artigo começa por estabelecer o problema Fractais 3D de um ponto de vista geral e teórico. De seguida, a apresentação das várias técnicas é abordada detalhadamente relativa a cada tipo de fractal abordado. De seguida é abordada a arquitetura do projeto relativa a interface e Webgl e a iluminação utilizada no projeto.

I. Introdução

No âmbito da Unidade Curricular Computação Visual do 4º ano do curso Mestrado Integrado em Engenharia de Computadores e Telemática escolhemos a realização de um projecto em WebGL no qual desenvolvemos uma aplicação de manipulação de Fractais em 3D. Optamos por este tema pois o conceito de fractais e a sua implementação é de interesse para ambos os elementos do grupo.

WebGL é uma API de javaScript para renderização de gráficos 3D e 2D no browser, permite a utilização da GPU para acelerar a renderização. Para expressar os shaders WebGL é utilizada a linguagem GLSL[6] [7] .

Fractais[1] são figuras geométricas que exprimem um padrão repetido representado em várias escalas, também conhecidos por 'simetria expandida'. Quando a simetria é exacta a cada escala (nível de recursividade), podemos classificar a replica como 'self-similar'. Alguns exemplos conhecidos desse tipo de simetria é o Menger Sponge(Fig.1) e a Sierpinski triangle(Fig.2).

O artigo está dividido em apresentação do fractal, a sua visualização em 3D, a sua origem em 2D, a sua implementação, arquitectura do sistema e a iluminação do fractal, terminando com uma breve conclusão.

Com a execução deste trabalho pretendemos, essencialmente, alargar os nossos conhecimentos no que respeita à criação e manipulação de fractais em 3D, bem como a sua renderização e iluminação.

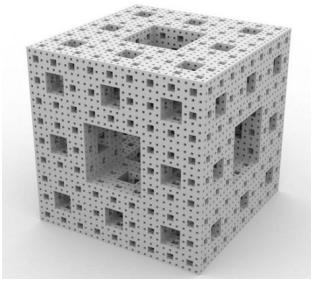

Fig. 1 - Menger Sponge, self-similar 3D Fractal

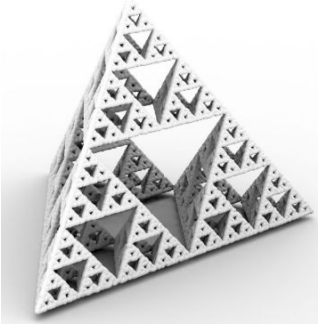

Fig. 2 - Sierpinski Pyramid, self-similar 3D Fractal

II. Menger Sponge

Numa abordagem relativa ao Menger Sponge[2], não podemos esquecer a sua origem, a Sierpinski carpet.

A Sierpinski carpet é um fractal plano descrita por Waclaw Sierpinski onde foi utilizada a técnica de subdivisão em cópias do mesmo objecto em pequenas dimensões recursivamente, esta mesma técnica é aplicada no Sierpinski triangle.

Processo

No caso da Sierpinski carpet, na fase inicial, um quadrado é dividido em 9 quadrados, retirando o quadrado central, na próxima iteração, em cada quadrado, é aplicado o mesmo processo de divisão (Fig3.).

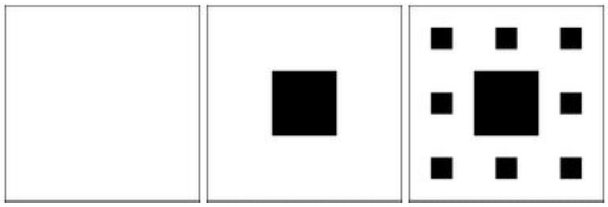

Fig. 3 - Sierpinski carpet, processo de divisão (2 iterações)

Em 3D

O Menger Sponge resume-se a passar o Sierpinski carpet para 3D, neste caso a subdivisão é feita em cubos (3D) enquanto que posteriormente era feita em quadrados (2D)(Fig4.).

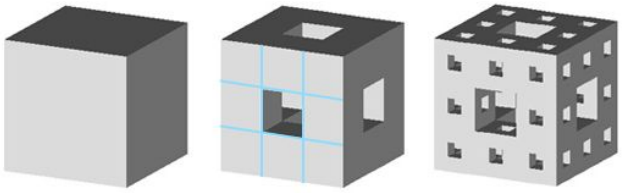

Fig. 4 - Menger Sponge, processo de divisão (2 iterações)

Implementação

Na implementação do algoritmo de divisão começamos por identificar cada vértice do cubo de modo a obter um modelo ordenado dos vértices, de seguida são efectuadas 20 subdivisões respectivas ao novo modelo (3x3x3) excluindo os 7 cubos centrais, 6 equivalente a cada face e o seu 'nucleo'.

O objectivo do algoritmo é transformar um Cubo1x1x1, num Cubo3x3x3 em que são removidos os cubos centrais de cada face e o cubo do centro do cubo original originando um efeito de "túnel" entre todas as faces paralelas.

São necessárias no total 20 divisões na transformação de um Cubo1x1x1 num Cubo3x3x3 com as remoções incluídas (Fig.5).

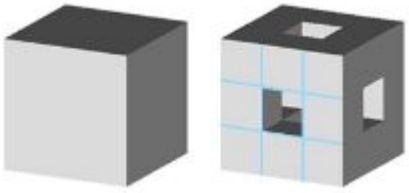

Fig. 5 - Cubo1x1x1 e Cubo3x3x3 com as remoções incluídas.

III. Sierpinski Pyramid

Tal como o Menger Sponge, o Sierpinski Pyramide é baseado numa subdivisão recursiva 2D, o Sierpinski triangle.

Esta subdivisão recursiva tem como fim a subdivisão de um triângulo em 4 triângulos equiláteros formando um padrão repetitivo.

Processo

No caso do Sierpinski Triangle[3], na fase inicial, um triângulo equilátero é dividido em 4 triângulos, retirando o triângulo central.

Na próxima iteração, em cada triângulo restante, é aplicado o mesmo processo de divisão (Fig6.).

Fig. 6 - Sierpinski triangle, processo de divisão (2 iterações)

Em 3D

Tal como o Menger Sponge, o Sierpinski Pyramid resume-se a passar o Sierpinski triangle para 3D, neste caso a subdivisão é feita em tetraedros (3D) enquanto que posteriormente era feita em triângulos (2D)(Fig7.).

Fig. 7 - Sierpinski Pyramid, processo de divisão (2 iterações)

Implementação

Na implementação do algoritmo de divisão começamos por identificar cada vértice do tetraedro de modo a obter um modelo ordenado dos vértices, de seguida efetuamos 4 divisões por face, restando 1 cubo 'vazio' por face.

O objectivo do algoritmo é transformar um Tetraedro, num Tetraedro composto por 4 Tetraedros, a Sierpinski pyramid.

São necessárias no total 4 divisões na transformação de um Tetraedro numa Sierpinski pyramid, com as remoções incluídas (Fig.8).

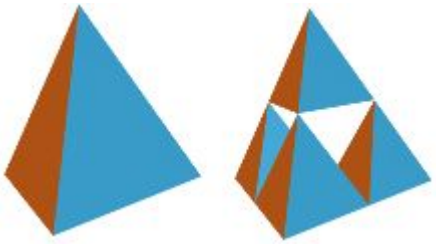

Fig. 8 -Tetraedro e Sierpinski pyramid, com as remoções incluídas.

IV. Koch SnowFlake 3D

O Koch SnowFlake[4] baseia-se num modelo 2D designado por 'Curva de Koch'[5], mais conhecido como 'Floco de neve de Koch'. Este modelo(curva geométrica) é considerado um dos primeiros fractais a ser descrito.

Processo

Na construção do 'Floco de neve de Koch', divide-se as arestas em 3 segmentos de igual comprimento, de seguida desenha-se um triângulo equilátero (60 graus) em que a base do novo triângulo será o segmento central calculado (⅓ a ⅔ da aresta), no final é removido este segmento base após a inserção do triângulo(Fig.9).

Fig. 9 - Floco de neve de Koch, 2 iterações.

Em 3D

Para obter este efeito em 3D começou se com um tetraedro e para cada face do tetraedro gera-se recursivamente um tetraedro novo com tamanho inferior(Fig.9).

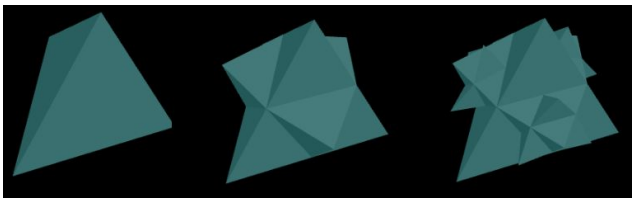

Fig. 9 - Floco de neve de Koch 3D, 2 iterações.

Implementação

Para implementar este fractal reutilizou-se a ideia da fractal Sierpinski Gasket.

Na implementação do algoritmo de divisão começamos por identificar cada vértice do tetraedro de modo a obter um modelo ordenado dos vértices, de seguida efetuamos 4 divisões por face e acabamos por inserir o tetraedro original(ao contrario do Sierpinski gasket).

O objectivo do algoritmo é transformar um Tetraedro, numa figura composta por 5 Tetraedros, Koch SnowFlake 3D.

São necessárias no total 4 divisões na transformação de um Tetraedro num Koch snowflake 3D.

V. 'Xadrez 3D'

Uma ideia que surgiu na realização deste projecto foi o efeito 'Xadrez 3D' e achamos interessante aplicar este efeito de uma forma recursiva.

Processo

Este efeito aplica sob um quadrado preto um novo xadrez, isto é, são gerados 2 novos quadrados com cor e 2 transparentes(Fig.10).

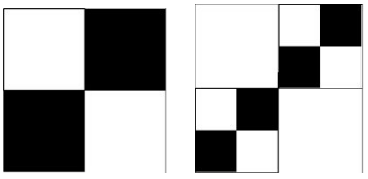

Fig. 10 - Efeito Xadrez

Em 3D

Em 3D são feitas 4 divisões no cubo, em primeiro lugar o no cubo canto inferior direito frente, cubo canto superior direito back, cubo canto inferior esquerdo back e finalmente cubo canto superior esquerdo frente(Fig.11).

Fig.11 - Efeito Xadrez

Implementação

Na implementação do algoritmo de divisão começamos por identificar cada vértice do tetraedro de modo a obter um modelo ordenado dos vértices, de seguida efetuamos 4 divisões resultantes nos 4 cubos que formam o 'Efeito Xadrez'.

Similaridades

Se continuaremos a iterar com o algoritmo recursivo 'Efeito Xadrez', o modelo do cubo irá tender para uma Sierpinski Pyramid(Fig.12).

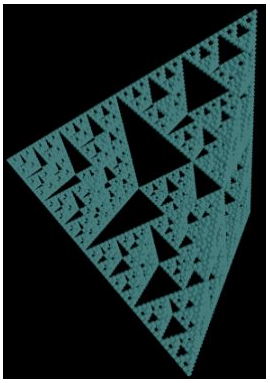

Fig. 12 - Efeito Xadrez 3D

VI. Arquitetura

O sistema é composto por vários módulos auxiliares e o programa principal(Fig.13).

É composto por um módulo que contém estruturas de dados e algumas operações matemáticas básicas(math.js), outro módulo que contém operações matemáticas mais complexas(models.js), um módulo exterior que contém funções auxiliares externas(webgl-utils.js by google), a pagina web em si onde todo o layout está exposto e os shaders definidos(index.html) e finalmente o programa que contém as funcionalidades necessárias para a criação de fractais 3D(proj.js).

Para o design da interface principal o grupo decidiu fazer algo simples e funcional, onde é apresentado ao utilizador um conjunto de botões do lado esquerdo e o canvas que contém os objectos do lado direito juntamente com um dropdown list que contém os fractais suportados.

O utilizador ao selecionar um fractal diferente na lista ou a aumentar o nível de recursividade irá fazer com que o sistema reinicie os buffers. As únicas formas de reiniciar os buffers são as descritas em cima pois o grupo concluiu que reiniciar os buffers era muito pesado para o sistema e várias vezes desnecessário. Também para reduzir a carga do sistema o grupo optou por alterar a iluminação, que irá ser descrito no próximo capítulo.

Ao premir algum botão irá apenas modificar valores de variáveis que irão ser utilizadas pelo sistema na fase de drawScene onde são aplicadas transformações à matriz modelview e o sistema irá apenas redesenhar o objecto.

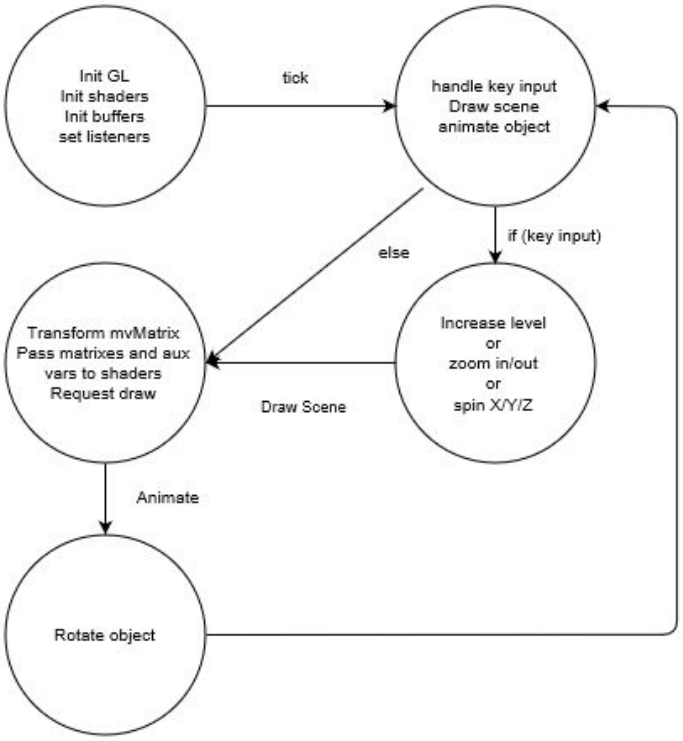

Fig. 13 - Maquina de estados do sistema

VI. Iluminação

Como o projeto utiliza extensivamente recursividade e é pesado para o CPU do computador o grupo necessitou de explorar a utilização da GPU para alguma da carga. Para o cálculo da iluminação do objeto o grupo decidiu utilizar os shaders, nomeadamente o fragment shader, e para tal necessitou de criar uma matriz nova, que contém a transposição da matriz inversa da modelview matrix.

O grupo decidiu apenas utilizar luz ambiente e difusa pois utilizar especular adicionaria mais carga ao sistema e ambiente e difusa são suficientes para o projecto em causa.

Foi escolhido o fragment shader pois calcula a iluminação para cada pixel em vez de ser por vértices.

VII. Conclusão

Neste trabalho foi abordado o tema Fractais 3D, referimos vários tipos de Fractais 3D como, Sierpinski Pyramid e Menger Sponge, foi feita a devida descrição de cada fractal desde a sua origem, a sua versão 2D, a sua forma em 3D, a implementação do algoritmo recursivo de divisão e concluímos que os fractais são objetos complexos e difíceis de manipular, requerem um exercício mental elevado relativo à sua geometria no espaço e na sua descrição, isto é, definição dos vértices e faces pela ordem correta.

Cumprimos todos os objetivos que nos foram propostos neste projeto em que o principal era a projeção de um Fractal 3D.

Para um aumento da performance do sistema o grupo optou pela utilização da GPU para alguns cálculos(iluminação).

Este trabalho foi muito importante para o nosso conhecimento, o aprofundamento deste tema fez-nos aprender e compreender o método utilizado na manipulação de objetos 3D utilizando algoritmos recursivos.

VIII. Referências

[1] “Fractal”. Em “Wikipedia”: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Fractal> Acesso em Novembro 2016

[2] “Menger Sponge”. Em “Wikipedia”: a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Menger_sponge> Acesso em Novembro 2016

[3] “Sierpinski Triangle”. Em “Wikipedia”: a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Sierpinski_triangle> Acesso em Novembro 2016

[4] “Koch Snowflake”. Em “Wikipedia”: a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Koch_snowflake> Acesso em Novembro 2016

[5] “Curva de Koch”. Em “Wikipedia”: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Curva_de_Koch> Acesso em Novembro 2016

[6] “GLSL Programming”. Em “Wikibooks”: Open books for an open world. Disponível em: <https://en.wikibooks.org/wiki/GLSL_Programming/Applying_Matrix_Transformations> Acesso em Novembro 2016

[7] “WebGL Shaders and GLSL”. Em “WebGL Fundamentals”: WebGL from the ground up. No magic.. Disponível em: <http://webglfundamentals.org/webgl/lessons/webgl-shaders-and-glsl.html>
Acesso em Novembro 2016

[8] “Computação Visual, WebGL”. Em “Sweet.ua.pt/jmadeira”. Disponível em: <http://sweet.ua.pt/jmadeira/WebGL>
Acesso em Novembro 2016

[9] “The Lessons”. Em “Learning WebGL”. Disponível em: <http://learningwebgl.com/blog/?page_id=1217>
Acesso em Novembro 2016