Fractais 3D

Diogo Fernandes 89221

Resumo – O texto que se segue pretende apresentar a aplicação desenvolvida para a visualização de fractais em 3D.

I. INTRODUÇÃO

A aplicação Fractais 3D foi desenvolvida para a unidade curricular Computação Visual do Mestrado Integrado em Engenharia de Computadores e Telemática. A aplicação foi desenvolvida em WebGL e o seu objetivo é permitir a visualização de alguns fractais simples 3D com o uso de uma câmera livre em primeira pessoa.

A aplicação permite renderizar os seguintes fractais: “Sierpinski pyramid”, “Koch Snowflake”, “Menger Sponge” e “Jerusalem Cube”.

Na natureza pode-se observar certos objetos com repetições de padrões e simetria, como é o caso de um floco de neve. Os fractais é uma forma de se poder simular tais objetos.

Os fractais têm, geralmente, um padrão ‘self-similar’, isto é, são objetos exatamente similares a uma parte mais pequena deles próprios [1]. Cada parte do fractal sofre o mesmo tipo de padrão sucessivamente, conforme o número de iterações.

Neste artigo vai-se descrever como os fractais acima referidos foram implementados.

O objetivo deste trabalho é adquirir e pôr em prática conhecimentos de como criar e manipular fractais em 3D com iluminação em real time e aplicar esses conhecimentos numa aplicação que pode ser executada numa página web.

II. IMPLEMENTAÇÃO

Nesta secção serão apresentados todos os fractais implementados. Durante toda a implementação foi usada os recursos disponíveis no site da unidade curricular dedicados às aulas práticas.

No desenvolvimentos de alguns algoritmos recursivos para a geração dos fractais teve-se como base alguns exemplos já existentes.

1.Sierpinski Pyramid

Este fractal é baseado na subdivisão recursiva feita por tetraedros, como demonstra a Figura 1, que contém o estado original(tetraedro) e o resultado após 1 e 2 iterações.

Figure 1: Sierpinski Pyramid (0,1 e 2 iterações)

Para obter as próximas iterações é calculado o ponto intermédio de cada par de vertices do tetraedro inicial e com esses pontos formam-se 4 novos tetraedros.

2-Koch Snowflake

Com a mesma estrutura inicial (tetraedro) do exemplo anterior, é possível criar um outro tipo de fractal. Consiste em acrescentar um novo tetraedro mais pequeno, perpendicularmente a cada uma das faces como mostra a Figura2.

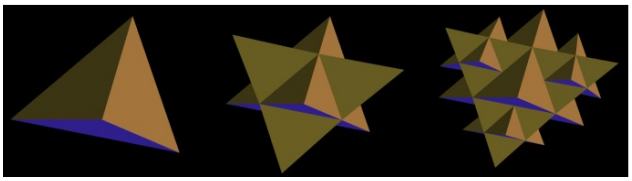

Figure 2: Koch Snowflake(0,1 e 2 iterações)

Para formar este fractal para cada iteração é necessário para cada face calcular o ponto médio de cada par de vertices e o ponto central. Para obter o ponto “do topo” calcula-se o vetor normal da face e as coordenadas da extremidade do vetor são as coordenadas do ponto.

3- Menger Sponge

Para compreender como a Menger Sponge é implementada, é necessário perceber como o tapete de Sierpinski[2](Figura 3) é construído. Começando com um quadrado, divide-se esse quadrado em 9 quadrados iguais e remove-se o quadrado central. Depois aplica-se a mesma operação para cada quadrado novo.

Figure 3: Tapete de Sierpinski

A Menger sponge tem o mesmo conceito só que em 3D. Começando com um cubo, este é dividido em 27 cubos e depois são removidos os cubos do centro de cada face, e do seu interior.

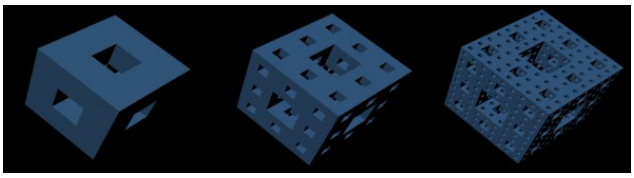

Figure 4: Menger Sponge(0,1 e 2 iterações)

4-Jerusalém Cube

O Jerusalém Cube é um fractal bastante semelhante ao Menger Sponge. A sua principal diferença está no modo de como é feito a subdivisão dos cubos. No fractal Menger Sponge todos os cubos têm a mesma dimensão. Neste fractal existe um cubo mais pequeno no meio das extremidades. Deste modo é obtida uma cruz em cada face, que não é preenchida.

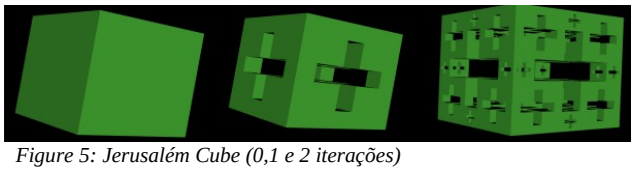

Figure 5: Jerusalém Cube (0,1 e 2 iterações)

Começando com um cubo, subdividimos esse mesmo cubo em 8 cubos, em que 4 deles têm tamanho inferior, ficando estes cubos de tamanho inferior no meio de cada extremidade da face.

III.CÂMERA

Para ajudar a visualização dos fractais gerados foi implementada uma câmera livre em primeira pessoa. Ela permite fazer todo o tipo de movimentos incluindo “olhar” para cima/baixo/lados e andar na direção para o qual se está olhar. Isto permite ao utilizador entrar pelos fractais visualizando assim o seu interior.

Também foi implementada uma forma de fazer o fractal rodar com o uso do rato.

IV. ILLUMINAÇÃO

A aplicação permite escolher um conjunto de cores e modos para a illuminação. No modo ***static*** a fonte de luz fica no mesmo ponto mesmo com o utilizador movimentando-se pelo espaço. No modo ***lantern*** a fonte de luz são os “olhos da câmera” .

Estes modos só se aplicam quando a cor da luz selecionada é vermelha, verde ou azul. Quando a cor da luz selecionada é custom1,2 ou 3 existe mais do que uma fonte de luz no qual podem ou não ter rotação automática, logo , o modo(static ou lantern) deixa de ter efeito.

V. MANDELBULB

O objétivo inicial para o projeto seria a renderização de um mandelbulb[4].

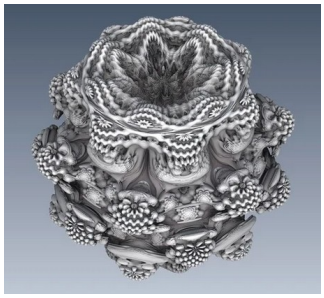

Figure 6: Mandelbulb

Foi feita uma investigação intensiva sobre este fractal e a matemática que o permite gerar, mas não existe muita informação disponível e o que foi encontrado era pouco clara.

$$\mathbf{v}^n := r^n \langle \sin(n\theta)\cos(n\phi), \sin(n\theta)\sin(n\phi), \cos(n\theta) \rangle,$$

where

$$r = \sqrt{x^2 + y^2 + z^2},$$

$$\phi = \arctan\frac{y}{x} = \arg(x + yi),$$

$$\theta = \arctan\frac{\sqrt{x^2 + y^2}}{z} = \arccos\frac{z}{r}.$$

Figure 7: White and Nylander's formula

Conseguiu-se desenvolver uma função que gera os pontos para o mandebulb mas não foi possível avançar apartir deste ponto. Pelo que se percebeu, para a iluminação seria necessário usar ray marching[3], algo que estava além do meu conhecimento atualmente.
Depois de muitas horas perdidas decidi foca-me em outras partes do projeto, como o desenvolvimento da câmera livre.

VI. Conclusão

Para todos os fractais aplicados, usou-se uma solução recursiva, que era o que se esperava devido à natureza 'self-similar' dos fractais.
Conseguiu-se recriar alguns fractais simples e ao fazê-lo aprofundar e treinar conhecimentos adquiridos na unidade curricular de Computação Visual, como a criação e manipulação de figuras 3D em runtime com iluminação, implementação de uma camera livre e criar uma interface capaz de interagir com os modelos.

References

[1] Wikipédia, "Self-similarity", https://en.wikipedia.org/wiki/Self-similarity

[2] Wikipédia, "Tapete de Sierpinski", https://pt.wikipedia.org/wiki/Tapete_de_Sierpinski

[3] Syntopia, "Ray Marching", http://blog.hvidtfeldts.net/index.php/2011/06/distance-estimated-3d-fractals-part-i/

[4] Wikipédia, "Mandelbulb", https://en.wikipedia.org/wiki/Mandelbulb