UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

NICOLAS VON DOLINGER MOREIRA ROCHA

DOCUMENTAÇÃO

BELO HORIZONTE

2023

Introdução

Este documento descreve a implementação de três diferentes fractais, seguindo as especificações fornecidas. Os fractais selecionados serão determinados com base na soma dos dígitos do número de matrícula do aluno, seguindo as regras indicadas.

Os três fractais serão implementados usando estratégias iterativas, e serão fornecidas equações de recorrência para calcular a quantidade de segmentos e símbolos em cada estágio. Além disso, será discutida a complexidade dos algoritmos utilizados e serão apresentadas opções de software para desenhar os fractais gerados.

1. Fractais a serem Implementados

(i) Fractal da Ilha de Koch: Este fractal será implementado quando a soma dos dígitos do número de matrícula, dividida por 4, resultar em 3.

(ii) Fractal de Preenchimento de Espaço de Peano: Este fractal será implementado quando o número de matrícula for ímpar.

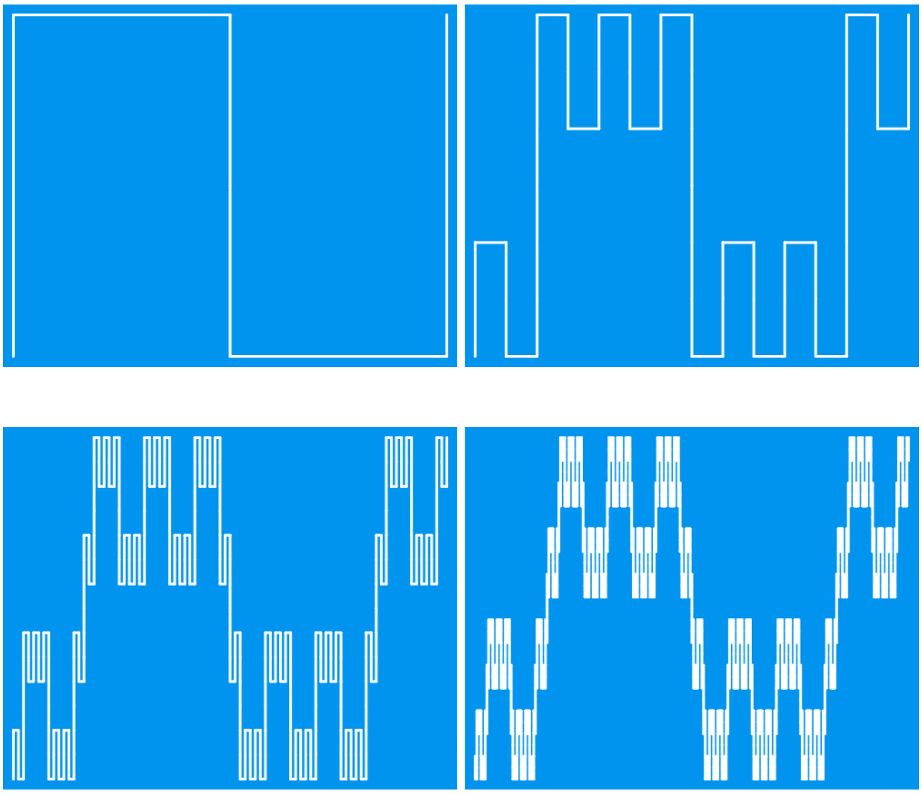
(iii) Fractal com Regra Própria: Será implementado um fractal personalizado, definido pelo próprio aluno, que gere uma cadeia de polígonos com pelo menos duas regras semelhantes aos fractais de Preenchimento de Espaço de Peano e Ilha de Koch. Abaixo segue a descrição da regra escolhida:

Axioma: X

$\theta = \pi/2$

Regras: X → XFFX+F+YFFY-F-XFFX
Y → YFFY-F-XFFX+F+YFFY

As imagens abaixo representam, respectivamente, da esquerda para a direita, os estágios 1, 2, 3 e 4:



Para a implementação dos fractais selecionados, foi escolhida a abordagem iterativa mencionada anteriormente, também é possível adotar a implementação recursiva para gerar os fractais selecionados. A implementação iterativa envolve a geração dos caracteres de cada estágio intermediário, gravando-os em um arquivo ou em uma estrutura de dados, e processando esses caracteres para gerar o próximo estágio. Esse processo é repetido até que o estágio desejado seja alcançado.

A implementação iterativa pode ser descrita da seguinte maneira:

- Definição das regras de construção: São estabelecidas as regras que determinam como o fractal é construído. Essas regras podem envolver transformações geométricas, substituições de símbolos ou outras operações específicas para cada fractal.
- Inicialização: É iniciado um estágio inicial, geralmente contendo apenas um ou alguns caracteres iniciais.
- Loop iterativo: Um loop iterativo é usado para percorrer cada estágio intermediário até o estágio desejado ser alcançado. Dentro do loop, os

caracteres do estágio atual são processados de acordo com as regras de construção.

- Geração do próximo estágio: Com base nos caracteres do estágio atual, são gerados os caracteres do próximo estágio. Isso pode envolver a substituição de certos caracteres por sequências de caracteres mais complexas ou a aplicação de transformações geométricas para adicionar detalhes ao fractal.
- Atualização do estágio atual: O estágio atual é atualizado com os caracteres gerados no passo anterior.
- Verificação da condição de parada: É verificado se o estágio atual atingiu o estágio desejado. Se sim, o processo é encerrado. Caso contrário, o loop iterativo continua executando.

A implementação iterativa é útil para fractais que podem ser construídos seguindo uma sequência clara de transformações ou regras de construção. Ela permite o controle preciso dos estágios intermediários e a visualização passo a passo da construção do fractal.

Ao utilizar a implementação iterativa, é importante garantir que as regras de construção estejam corretamente definidas e que o processo iterativo não resulte em loops infinitos. Também é necessário considerar o armazenamento e a manipulação dos caracteres de cada estágio intermediário, garantindo que os recursos do sistema sejam utilizados de forma eficiente.

A implementação recursiva de fractais possui uma série de pontos positivos e negativos a serem considerados. Vamos discorrer sobre esses aspectos abaixo:

Pontos Positivos:

- Simplicidade conceitual: A implementação recursiva reflete diretamente a natureza recursiva dos fractais, o que torna a compreensão do algoritmo de construção mais intuitiva. A estrutura recursiva permite que os fractais sejam construídos a partir de versões menores de si mesmos, resultando em uma lógica mais clara e concisa.

- Estrutura modular: A implementação recursiva permite que o código seja dividido em funções recursivas que resolvem partes específicas do problema. Isso facilita a compreensão e manutenção do código, uma vez que cada função tem um propósito bem definido.
- Flexibilidade e generalização: A implementação recursiva é altamente flexível e pode ser facilmente adaptada para diferentes tipos de fractais. Ao ajustar as regras de construção e as condições de parada, é possível explorar uma ampla variedade de fractais com base na mesma estrutura recursiva.
- Controle da complexidade: Através da recursão, é possível controlar a complexidade do algoritmo e definir um limite para o número de iterações ou detalhes do fractal. Isso permite a geração de fractais com diferentes níveis de detalhe, dependendo dos requisitos do projeto ou da capacidade de processamento disponível.

Pontos Negativos:

- Consumo de memória: A implementação recursiva pode consumir uma quantidade significativa de memória, especialmente para fractais que possuem muitos estágios ou uma recursão profunda. Cada chamada recursiva adiciona uma nova camada de ativação à pilha de chamadas, o que pode levar a um consumo excessivo de recursos em casos extremos.
- Desempenho: Em alguns casos, a implementação recursiva pode ser menos eficiente em termos de desempenho em comparação com outras abordagens, como a implementação iterativa. Isso ocorre devido ao custo adicional de chamadas de função e à necessidade de repetir cálculos para estágios intermediários.
- Estouro de pilha: Se a recursão não for adequadamente controlada e limitada, pode ocorrer um estouro de pilha, o que resulta em um erro e na interrupção da execução do programa. É importante definir uma

condição de parada correta e garantir que a recursão não se torne excessivamente profunda para evitar esse problema.

- Dificuldade na depuração: A recursão pode tornar a depuração do código mais desafiadora. O rastreamento do fluxo de execução pode se tornar complicado, especialmente em fractais com múltiplas chamadas recursivas e interações complexas entre elas.

Portanto, embora a implementação recursiva possua pontos positivos, como simplicidade conceitual e modularidade, é necessário considerar os pontos negativos, como o consumo de memória e o desempenho potencialmente inferior.

Neste projeto, a estratégia de implementação iterativa pode ser adotada para os fractais selecionados, permitindo a geração gradual dos estágios intermediários e facilitando a visualização da construção do fractal.

2. Equações de Recorrência

Para cada fractal implementado, serão apresentadas as equações de recorrência para calcular a quantidade de segmentos (F) gerados e a quantidade de símbolos existentes em cada estágio. Essas equações fornecerão uma representação matemática das sequências geradas pelos fractais e permitirão prever a quantidade de elementos em qualquer estágio.

(i) Fractal da Ilha de Koch

Iterações	F	Símbolos
0	4	7
1	36	63
2	324	567
3	2916	5103
4	26244	45927

Dessa forma, analisando as informações disponíveis, é possível perceber que a quantidade total de símbolos gerados em cada estágio é 9 vezes o número de caracteres do estágio anterior.

E(0) = 7

E(1) = E(0) * 9

E(2) = E(1) * 9

E(3) = E(2) * 9

E(4) = E(3) * 9

Portanto, tem-se a seguinte equação de recorrência:

F(0) = 7

F(n) = F(n-1) * 9

(ii) Fractal de Preenchimento de Espaço de Peano

Iterações	F	Símbolos
0	0	1
1	8	21
2	80	201
3	728	1821
4	6560	16401

Dessa forma, analisando as informações disponíveis, é possível perceber que a quantidade total de símbolos gerados em cada estágio é 9 vezes o número de caracteres do estágio anterior mais 12.

E(0) = 1

E(1) = 9 * E(0) + 12

E(2) = 9 * E(1) + 12

E(3) = 9 * E(2) + 12

E(4) = 9 * E(3) + 12

Portanto, tem-se a seguinte equação de recorrência:

F(0) = 1

F(n) = 9 * F(n-1) + 12

(iii) Fractal com Regra Própria

Iterações	F	Símbolos
0	0	1
1	8	18
2	56	120
3	344	732
4	2072	4404

Dessa forma, analisando as informações disponíveis, é possível perceber que a quantidade total de símbolos gerados em cada estágio é igual à anterior mais 2 vezes 6.

E(0) = 1

E(1) = (E(0) + 2) * 6

E(2) = (E(1) + 2) * 6

E(3) = (E(2) + 2) * 6

E(4) = (E(3) + 2) * 6

Equação de recorrência:

F(0) = 1

F(n) = (F(n-1) + 2) * 6

3. Complexidade dos Algoritmos

Será realizada uma análise da complexidade dos algoritmos utilizados para implementar os fractais, considerando a notação assintótica mais precisa possível. Serão discutidos o tempo de execução e o uso de recursos, como memória e processamento, para avaliar a eficiência dos algoritmos em diferentes cenários.

Para determinar a complexidade assintótica dessa recorrência, vamos analisar o crescimento da função F(n) à medida que n aumenta. A complexidade assintótica é uma medida do crescimento da função quando n se aproxima do infinito.

(i) Fractal da Ilha de Koch

$\Theta(9^n)$ - "Teta de 9 elevado a n"

(ii) Fractal de Preenchimento de Espaço de Peano

$\Theta(9^n)$ - "Teta de 9 elevado a n"

(iii) Fractal com Regra Própria

$\Theta(n)$ - "Teta de n"

Foi utilizada a notação Θ porque tanto o limite superior quanto o limite inferior da sequência são exponenciais em relação a n. A constante x multiplicada por 9^n é uma função assintótica tanto inferior quanto superior para a sequência F(n). Portanto, a notação Θ é apropriada para descrever o custo assintótico dessas sequências.

4. Opções de Software para Desenho dos Fractais

O site Online Math Tools é uma ferramenta online que oferece uma variedade de recursos matemáticos, incluindo a capacidade de gerar fractais por meio de sistemas L-System. Essa abordagem permite criar e explorar fractais de maneira interativa e visual, sem a necessidade de implementar a renderização gráfica manualmente em C.

Ao acessar o site, você encontrará uma interface intuitiva que permite definir as regras do sistema L-System. Essas regras especificam como os símbolos devem ser substituídos e transformados em sequências de caracteres. Com a definição adequada das regras, é possível gerar padrões complexos que se repetem e formam os fractais desejados.

Além das regras, você também pode configurar outros parâmetros importantes, como o ângulo de rotação, o número de iterações e o tamanho inicial. Esses parâmetros influenciam diretamente na aparência e detalhamento dos fractais gerados.

Após configurar as regras e os parâmetros, o site exibirá o fractal resultante na tela, permitindo que você visualize e explore o padrão gerado. É possível interagir com o fractal, aplicar zoom, ajustar a escala e mover a visualização para explorar diferentes partes do fractal.

Essa abordagem oferece a vantagem de simplicidade e conveniência, pois você pode gerar e visualizar fractais diretamente no navegador, sem a necessidade de implementar a lógica de renderização em C. Isso pode economizar tempo e esforço, especialmente se o objetivo principal do trabalho for a exploração dos fractais gerados.

Em resumo, o uso do site Online Math Tools permite gerar e visualizar fractais de maneira interativa e conveniente, sem a necessidade de implementação manual da renderização gráfica em C. É uma alternativa prática para explorar fractais, proporcionando uma experiência visual imersiva.
Para acessar o site apresentado e aplicar a regra de qualquer fractal, clique nesse link: https://onlinemathtools.com/l-system-generator . As instruções para uso estão no próprio site.

5. Conclusão

Este documento apresentou a implementação de três fractais com base em especificações fornecidas, discutindo diferentes estratégias de implementação, equações de recorrência, complexidade dos algoritmos e opções de software para desenhar os fractais gerados. Espera-se que essa documentação forneça uma visão geral abrangente do projeto e auxilie na compreensão e implementação dos fractais propostos.