Trabalho Parcial Sobre o Estudo Desenvolvido

Aluna: Giordanna De Gregoriis
Orientadora: Cristina Lúcia Dias Vaz
Co-Orientador: Dionne Cavalcante Monteiro

Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Faculdade de Computação
Bacharelado em Ciência da Computação

28 de setembro de 2016

Introdução

- Este seminário tem como objetivo mostrar os resultados do que já foi obtido até o momento;

Introdução

• Este seminário tem como objetivo mostrar os resultados do que já foi obtido até o momento;

• Na primeira seção é mostrado os resultados do estudo feito em torno do artigo *"The Art of Random Fractals"*, escrito por Douglas Dunham e John Shier, onde é explicado o funcionamento e a matemática do algoritmo de demonstração, com as devidas considerações finais e o resumo dos resultados obtidos;

Introdução

• Este seminário tem como objetivo mostrar os resultados do que já foi obtido até o momento;

• Na primeira seção é mostrado os resultados do estudo feito em torno do artigo *"The Art of Random Fractals"*, escrito por Douglas Dunham e John Shier, onde é explicado o funcionamento e a matemática do algoritmo de demonstração, com as devidas considerações finais e o resumo dos resultados obtidos;

• Na segunda seção tem-se o início de um estudo a respeito do fractal Apolloniano, onde para melhor compreensão foi feita a sua implementação.

Estudo do Artigo "The Art Of Random Fractals"

• O algoritmo criado por John Shier possui uma base matemática que permite preencher uma região espacial com uma sequência finita de formas (ou "estampas", como ele descreve) colocadas aleatoriamente e cada vez menores;

Estudo do Artigo "The Art Of Random Fractals"

- O algoritmo criado por John Shier possui uma base matemática que permite preencher uma região espacial com uma sequência finita de formas (ou "estampas", como ele descreve) colocadas aleatoriamente e cada vez menores;
- Este algoritmo pode ser usado para produzir uma variedade de padrões que são esteticamente agradáveis, que pela sua aparente autossimilaridade são chamados de padrões fractais.

Início

Inicialmente desejava-se preencher área A com estampas em posições aleatórias, progressivamente menores, sem que estas tenham intersecção entre si. Foi descoberto através de testes que isso pode ser alcançado se for obedecido uma regra de potência inversa;

Início

Inicialmente desejava-se preencher área A com estampas em posições aleatórias, progressivamente menores, sem que estas tenham intersecção entre si. Foi descoberto através de testes que isso pode ser alcançado se for obedecido uma regra de potência inversa;

Para $i = 0, 1, 2, ...$ a área da estampa de ordem i, A_i pode ser definido por:

$$A_i = \frac{A}{\zeta(c, N)(N + i)^c} \tag{1}$$

Onde $c > 1$, $c_{max} \approx 1.48$ e $N > 1$ são parâmetros, e $\zeta(c, N)$ é a função zeta de Hurwitz:

$$\zeta(c, N) = \sum_{k=0}^{\infty} \frac{1}{(N + k)^c}$$

Função Zeta de Hurtwitz

A função zeta de Hurwitz implementada no algoritmo original é feita apenas no domínio dos números reais, feita com somatórios aproximados. Esta função aproximada é dada por:

$$\zeta(c, N) = \left(\sum_{i=N}^{100000} i^{-c} \right) + \left(\frac{1}{c-1} \times N^{1-c} \right) \tag{2}$$

Função Zeta de Hurtwitz

No algoritmo de demonstração o c é um valor encontrado aleatoriamente entre 1 e 1.48 e $N = 2$.

O número retornado por esta função será utilizado para determinar a porcentagem da área A que será preenchida por A_0.

Função Zeta de Hurtwitz

No algoritmo de demonstração o c é um valor encontrado aleatoriamente entre 1 e 1.48 e $N = 2$.

O número retornado por esta função será utilizado para determinar a porcentagem da área A que será preenchida por A_0. **Exemplo:**

$$\zeta(1.263, 2) = \left(\sum_{i=2}^{100000} i^{-1.263}\right) + \left(\frac{1}{1.263 - 1} \times 2^{1-1.263}\right) \tag{3}$$
$$\approx 3.23 + 0.19 \approx 3.42$$

Início

Com $c = 1.263$ e $N = 2$, o valor obtido de $\zeta(1.263, 2)$ foi aproximadamente 3.42. O algoritmo então procede calculando a razão através de:

$$Razão = \frac{1}{\zeta(1.263, 2)} = \frac{1}{3.42} \approx 0.29 \tag{4}$$

Então a área A_0 da primeira forma a ser posicionada no plano deverá preencher aproximadamente 29% da área original.

Início

Na demonstração original deseja-se preencher o plano com círculos. Para isso utiliza-se a razão para descobrir qual será o raio do primeiro círculo que terá área A_0, dada por:

$$Raio_{A_0} = (\sqrt{A_{total}} \times \sqrt{\frac{Razão}{\pi}}) \times N^{-\frac{c}{2}} \quad (5)$$

Onde $N^{-\frac{c}{2}}$ serve para reduzir o valor obtido de acordo com a iteração N, assim servindo para a parte iterativa do algoritmo também.

Início

Substituindo a fórmula anterior pelos valores do exemplo dado alguns slides atrás e considerando que a nossa área é uma tela de 600×600 pixels, temos:

$$Raio_{A_0} = (\sqrt{360000} \times \sqrt{\frac{0.29}{\pi}}) \times 2^{-\frac{1.263}{2}} \tag{6}$$

$$\approx (600 \times 0.3) \times 2^{-0.6315} \approx 180 \times 0.64 \approx 115.2$$

Iteração

• Quando $i = 0$, a primeira forma com área de A_0 é colocada aleatóriamente no interior da região R tal que não se sobreponha a fronteira de R. Isso normalmente requer várias tentativas em posições aleatórias antes da obtenção de um posicionamento bem sucedido em que a forma é completamente dentro de R;

Iteração

• Quando $i = 0$, a primeira forma com área de A_0 é colocada aleatóriamente no interior da região R tal que não se sobreponha a fronteira de R. Isso normalmente requer várias tentativas em posições aleatórias antes da obtenção de um posicionamento bem sucedido em que a forma é completamente dentro de R;

• Quando $i > 0$, para cada $i = 1, 2, ..., n$ coloca-se de forma iterativa aleatoriamente uma cópia da estampa com área A_i dentro de R e de modo que não intersecte com qualquer cópia anteriormente colocada. Em seguida, passamos a colocar a próxima estampa, com área de A_{i+1}, ou parar o algoritmo se foi colocado a n-ésima forma ou atingiu uma outra condição de parada.

Iteração

O raio da forma A_i i para $i > 0$ é determinado por:

$$Raio_{A_i} = Raio_{A_0} \times i^{-\frac{c}{2}} \tag{7}$$

Onde $Raio_{A_0}$ é o raio da primeira forma inserida no plano.

Iteração

• Após determinar uma posição aleatória no plano, verifica-se caso a forma A_i não intersecta com nenhuma outra já posicionada. Este teste depende de cada forma, devendo ser implementado de acordo com suas características. Caso seja desejável inserir formas diferentes no mesmo plano, o teste deve ser abstraído para aceitar qualquer entrada e produzir uma resposta correta.

Iteração

• Após determinar uma posição aleatória no plano, verifica-se caso a forma A_i não intersecta com nenhuma outra já posicionada. Este teste depende de cada forma, devendo ser implementado de acordo com suas características. Caso seja desejável inserir formas diferentes no mesmo plano, o teste deve ser abstraído para aceitar qualquer entrada e produzir uma resposta correta.

• Neste algoritmo demonstratívo, o teste é feito apenas para círculos. Além disso ele verifica se a distância entre os círculos é maior que a soma de seus raios, evitando assim que os círculos sejam posicionados muito próximo entre si e tornando a imagem mais esteticamente agradável.

Iteração

Algoritmo 1: Teste de intersecção dos círculos

Entrada: $x_1, y_1, raio_1, x_2, y_2, raio_2$

Saída: *true* se não tem intersecção e *false* caso contrário

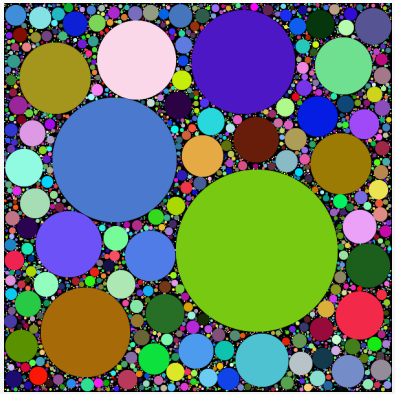
```
1  início
2      se |x1 − x2| < (raio1 + raio2) então
3          se |y1 − y2| < (raio1 + raio2) então
4              se √((x1 − x2)² + (y1 − y2)²) >= (raio1 + raio2) então
5                  retorna true
6              fim
7          fim
8      fim
9  fim
10 retorna false
```

Finalização

Terminado a iteração temos uma figura que se assemelha a um fractal geométrico aleatório onde nenhuma das estampas se tocam, e a área não preenchida ou carpete é um conjunto conectado contínuo (isso se forem estampas sem oco).

Finalização

Figura: Exemplo de uma execução, com $c = 1,3$



Testes e Resultados

Vários testes foram realizados para entender melhor a relação entre os diversos parâmetros presentes no algoritmo.

Foi desenvolvido um aplicativo em Java onde se insere um valor c, sendo seus limites $1,01 \leqslant c \leqslant 1,48$. Depois de inserido o valor, inicia o processo de geração da imagem em uma tela de 600×600 pixels e $N = 2$.

Círculo

Dados

$c = 1,48$

$\zeta(c, N) = 1,694351369474375$

Razão $= 0,5901963536112477$

Raio $= 155,70852052297627$

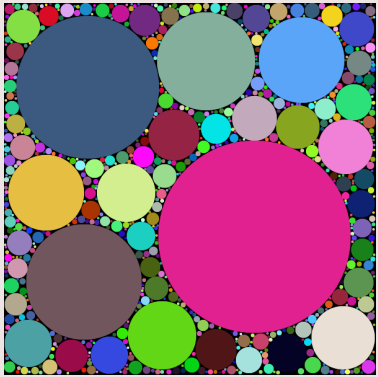
$A_{preenchida} \approx 93\%$

N° de iterações $= 400000$

N° de formas $= 2434$

tempo de execução $= 0,274$ segundos.

Resultado



Círculo

Dados

$c = 1,24$

$\zeta(c, N) = 3,7610743206468737$

Razão $= 0,26588147820168795$

Raio $= 113,57474202778332$

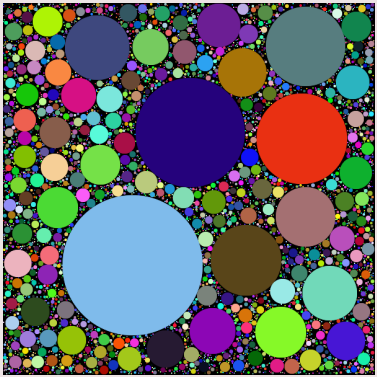
$A_{preenchida} \approx 78\%$

N° de iterações $= 374108$

N° de formas $= 90001$

tempo de execução $= 47,798$ segundos.

Resultado



Círculo

Dados

$c = 1,01$ $\zeta(c, N) = 99,5779388822347$

Razão $= 0,010042385002391388$

Raio $= 23,90422124959249$

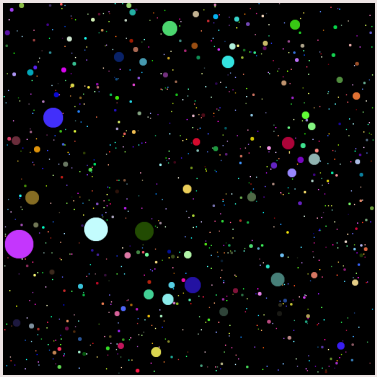
$A_{preenchida} \approx 5\%$

Nº de iterações $= 94017$

Nº de formas $= 90001$

tempo de execução $= 61,888$ segundos.

Resultado



Quadrado

Dados

$c = 1,48$

$\zeta(c, N) = 1,694351369474375$

Razão $= 0,5901963536112477$

Raio $= 137,99308340987494$

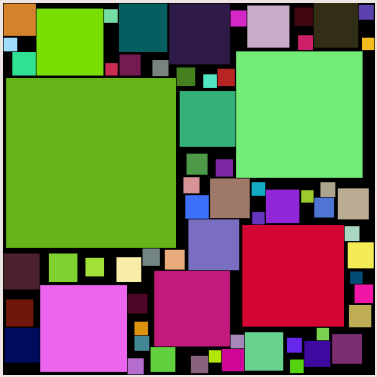
$A_{preenchida} \approx 86\%$

N° de iterações $= 406508$

N° de formas $= 120$

tempo de execução $= 0,271$ segundos.

Resultado



Quadrado

Dados

$c = 1,24$

$\zeta(c, N) = 3,7610743206468737$

Razão $= 0,26588147820168795$

Raio $= 100,65299443637255$

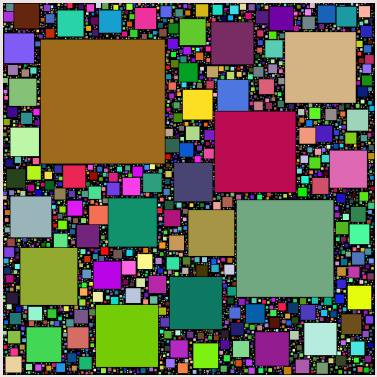
$A_{preenchida} \approx 77\%$

N° de iterações $= 400038$

N° de formas $= 33432$

tempo de execução $= 12,492$ segundos.

Resultado



Quadrado

Dados

$c = 1,01$ $\zeta(c, N) = 99,5779388822347$

Razão $= 0,010042385002391388$

Raio $= 21,184564503368836$

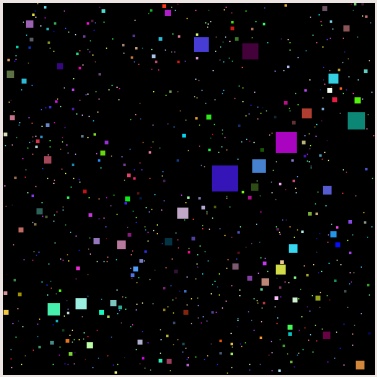
$A_{preenchida} \approx 4\%$

Nº de iterações $= 111583$

Nº de formas $= 90001$

tempo de execução $= 256,638$ segundos.

Resultado



Considerações Finais desta Seção

• Os métodos apresentados criam imagens artísticas, que são uma mistura de geometria e aleatoriedade;

• Quanto maior o fator c, maior o tamanho do raio da primeira forma a ser colocada no quadro, maior a porcentagem do preenchimento da área, menor será a quantidade de formas, menor o tempo de execução e mais rapidamente as formas vão diminuindo de tamanho;

• Pode-se observar que a colocação de novas formas geralmente é interrompida quando o raio da próxima forma é inferior a $3,5\%$ do raio da maior forma (a primeira).

Utilização e Implementação do Fractal Apolloniano (em andamento)

• O fractal criado pelo matemático grego Apollonius of Perga baseia-se em gerar "trincas" de círculos, onde cada um é tangente aos outros dois;

Utilização e Implementação do Fractal Apolloniano (em andamento)

- O fractal criado pelo matemático grego Apollonius of Perga baseia-se em gerar "trincas" de círculos, onde cada um é tangente aos outros dois;
- Pode ser usado para preencher com mais eficácia uma determinada área, além de produzir resultados esteticamente agradáveis.

Algoritmo

Para programar este fractal foi utilizado o algoritmo de Soddy, no qual inicia-se com 3 círculos de curvaturas $k(k_{ia}, k_{ib}, k_{ic})$.

O algoritmo utiliza os círculos definidos pelos centros c_n e curvaturas k_n, e também o teorema dos círculos de Descartes para encontrar a curvatura do próximo círculo k_4.

$$k_4 = k_1 + k_2 + k_3 \pm 2\sqrt{k_1k_2 + k_2k_3 + k_3k_1} \tag{8}$$

Algoritmo

Utiliza-se do teorema de círculos complexos de Descartes para encontrar o centro do círculo c_4.

$$c_4 = \frac{c_1k_1 + c_2k_2 + c_3k_3 \pm 2\sqrt{k_1k_2c_1c_2 + k_2k_3c_2c_3 + k_3k_1c_3c_1}}{k_4} \tag{9}$$

Algoritmo

Encontra-se um raio do primeiro círculo a ser colocado. Isto pode ser aleatório ou pré-definido. Neste programa estes valores são computados por porcentagens, ou seja, valores de $0,5 < raio_0 < 1$. Assim o primeiro círculo deve ser no mínimo um pouco maior que a metade do lado do quadrado do nosso quadro.

Acha-se o valor que será usado para encontrar as posições X e Y do centro do primeiro círculo.

$$valor = \frac{\pi}{2} - arcseno(aleatorio \times (1 - raio_0)/raio_0)$$

$$X_{primeiro} = raio_0 \times cos(valor)$$

$$Y_{primeiro} = raio_0 \times sen(valor)$$

Algoritmo

Número de círculos novos no estágio $n = 2 \times 3^n$

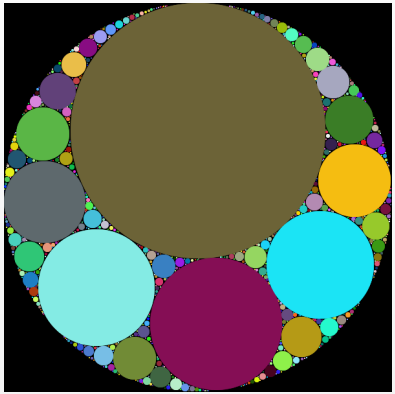
Número total de círculos depois de n estágios $= 2 \times 3^{n+1}$

Algoritmo

Terminado a iteração temos um fractal Apolloniano composto por círculos que se tangenciam levemente.

Algoritmo

Figura: Exemplo de uma execução, com $c = 1,3$



Testes e Resultados

Alguns testes foram realizados para verificar se era possível reaproveitar a razão do raio gerada a partir da função Zeta de Hurtwitz na geração do primeiro círculo do fractal.

Foi desenvolvido um aplicativo em Java onde se insere um valor c, sendo seus limites $1,01 \leqslant c \leqslant 1,48$. Depois de inserido o valor, inicia o processo de geração da imagem em uma tela de 600×600 pixels.

Testes e Resultados

Dados

$c = 1,48$

$\zeta(c, N) = 1,694351369474375$

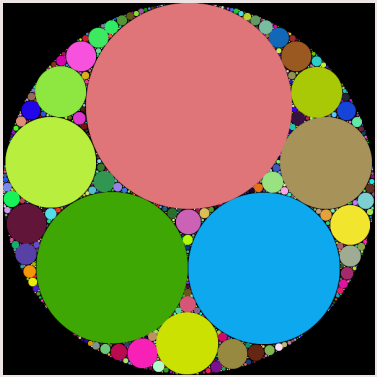
Razão $= 0,5901963536112477$

$A_{preenchida} \approx 96\%$

Nº de formas $= 3325$

tempo de execução $= 0,187$ segundos.

Resultado



Testes e Resultados

Dados

$c = 1,24$

$\zeta(c, N) = 3,7610743206468737$

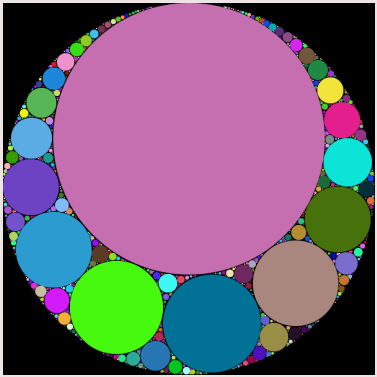
Razão $= 0,26588147820168795$

$A_{preenchida} \approx 95\%$

N° de formas $= 3140$

tempo de execução $= 0,19$ segundos.

Resultado



Testes e Resultados

Dados

$c = 1,01$

$\zeta(c, N) = 99,5779388822347$

Razão $= 0,010042385002391388$

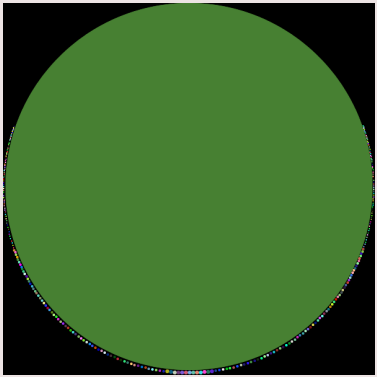
Neste caso foi utilizado:
$1-$Razão$= 0,989957614997608612$

$A_{preenchida} \approx 98\%$

Nº de formas $= 1353$

tempo de execução $= 0,74$ segundos.

Resultado



Considerações Finais desta Seção

• O método apresentado cria um fractal que preenche a área desejada com uma boa porcentagem;

• O fator c neste caso serviu para obtermos a razão da área, que foi justamente o valor utilizado para definir o tamanho do primeiro círculo da série. Como ele não permite que o primeiro círculo tenha uma razão menor que $0,5$ realiza-se uma "inversão" do valor realizando a operação $\text{Razão}_{nova} = 1 - \text{Razão}_{antiga}$, assim quanto menor a razão antes de $0,5$, maior será o raio do primeiro círculo;

• A sua formação é diferente dos métodos apresentados ateriormente pois neste caso há intersecção entre as formas, porém para resolver esteticamente este problema basta decrementar $0,5$ do raio antes de desenhar as formas no programa.

Referências

Dunham, Douglas, and John Shier. "The Art of Random Fractals."Bridges Seoul,(eds. Gary Greenfield, George Hart, and Reza Sarhangi), Seoul, Korea (2014): 79-86.

Wikibooks, The Free Textbook Project (2013) "Fractals/Apollonian fractals.", <https://en.wikibooks.org/wiki/Fractals/Apollonian_fractals>. Setembro.