Primeiro Exercício Programa

Prof. Alair — MAC122 — versão preliminar

22 de setembro de 2004

Um compilador de fractais

Resumo: Neste exercício programa, cada aluno deverá escrever um compilador suficientemente geral de fractais. A definição de um fractal, como veremos a seguir, naturalmente leva ao uso de recursão e o exercício-programa *deve* fazer uso de recursividade para o desenho dos fractais. Para a visualização de um fractal será fornecida uma ferramenta que cuidará de toda a parte gráfica. As linguagens de programação aceitas são java, C, python. Pode-se entregar em pascal desde que seja compilado com compilador a ser combiando com a monitora.

Antes de dar mais detalhes e descrever os fractais, faremos uma breve descrição de como poder codificar o desenho de curvas no plano.

Descrição de curvas

Daremos a seguir uma descrição bastante simples e muito freqüentemente utilizada para descrever uma seqüência de comandos capazes de desenhar uma seqüências contígua de segmentos de retas. Por simplicidade, uma tal seqüência contígua é chamada de *curva*. Se o ponto final da curva é o mesmo que o inicial, a curva é dita *fechada*, e é também chamada de *polígono*.

Para tanto, definimos um *estado* como sendo uma tripla[1] formada por três componentes[2]: as coordenadas x e y de um ponto no plano e; um ângulo α que designa a direção por onde será desenhado um segmento de reta que parte do ponto (x, y).

[1] Uma classe.

[2] Atributos da classe.

Alguns comandos serão dados de forma a se alterar os estados, e, eventualmente, o ponto do plano associado a um estado. Um estado pode ser alterado de forma limitada mas geral o suficente para nossos propósitos. Estas alterações são regidas por alguns parâmetros. Os parâmetros globais[3] que regem estas alterações são definidos para todos os estados: o comprimento u de um tal segmento; o incremento/decremento β que será aplicado às direções dos estados.

Assim, dado um estado, podemos alterá-lo através três comandos (métodos) possíveis:

1. um que, altera as coordenadas do ponto do plano associado ao estado. A partir do ponto (x, y), move-se na direção α por uma distância u, de forma que o novo ponto seja $(x + u\cos\alpha, y + u\sin\alpha))$. Ligam-se com um segmento de reta de comprimento u os dois pontos dos do plano, o antigo e o novo. Este comando é normalmalmente chamado de *forward* na linguagem de programação logo e por isto é simbolizado pela letra `F`. Neste enunciado, chamaremos de comando `F`.

2. um que vira a direção α para a direita (subtrai β de α). Este comando produz uma rotação positiva no sentido horário e por isto é simbolizado pela letra `+`. Neste enunciado, chamaremos de comando `+`.

3. um que vira a direção α para a esquerda (soma β a α). Este comando produz uma rotação no sentido anti-horário e por isto é simbolizado pela letra `-`. Neste enunciado, chamaremos de comando `-`.

Observe que o segundo e o terceiro método podem ser fundidos num único método se o ângulo a ser acrescido a α for passado como parâmetro. Num dos casos se passa β como parâmetro e no outro $-\beta$.

Assim, uma seqüência de comandos que permita desenhar uma curva, pode ser representada por uma cadeia de caracteres (string) no alfabeto $\{\texttt{F}, \texttt{+}, \texttt{-}\}$. Em nossos exemplos, suporemos um estado inicial no ponto $(0, 0)$ e direção inicial 0. Suporemos ainda os parâmetros globais[4] $u = 1$ e $\beta = \pi/3$. A string `F` corresponde ao desenho do segmento de reta visto na Figura 2 enquanto que a string `-F++F++F` corresponde a uma seqüência de comandos que desenha o triângulo equilátero visto na Figura 1

[3]Recomenda-se que estes parâmetros sejam fornecidos como parâmetros ao métodos que deles venham a depender, mas também poderiam ser atributos estáticos da classe, ou variáveis globais.

[4]Os ângulos são dados sempre em radianos.

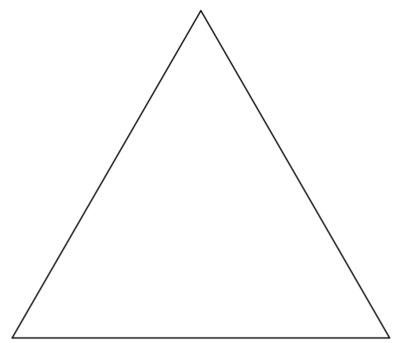


Figura 1: Curva correspondente ao string `-F++F++F`

Um fractal

Um fractal é uma estrutura geométrica onde se pode identificar numa fração de um pequeno detalhe uma reprodução bastante selhante ao todo. Costumeiramente, um fractal é definido através de recorrências. Vamos considerar como exemplo uma curva de Koch, aqui representada por K_n, onde n é o nível da recorrência da curva de Koch. Uma curva de Koch de nível 0, K_0, corresponde ao string `F`, como é visto na Figura 2, enquanto[5] que K_1 corres-

Figura 2: Curva K_0, correspondente ao string `F`

ponde ao string `F-F++F-F`, como é visto na Figura 3. De forma geral, bem

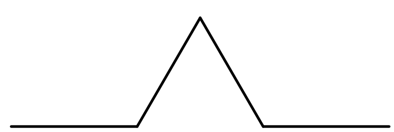


Figura 3: Curva K_1, corresponde ao string `F-F++F-F`

semelhante à curva K_1, uma curva K_n corresponde à seguinte seqüência de comandos:

1. desenha-se uma curva K_{n-1};
2. vira-se à esquerda de $\pi/3$;

[5] Para manter a escala, estamos adotando $u = 3^{-n}$ em cada curva K_n.

3. desenha-se uma curva K_{n-1};
4. vira-se à direita de $2\pi/3$;
5. desenha-se uma curva K_{n-1};
6. vira-se à esquerda de $\pi/3$;
7. desenha-se uma curva K_{n-1};

Assim, temos que a string associada à curva K_2 é[6]
`F-F++F-F - F-F++F-F ++ F-F++F-F - F-F++F-F`.

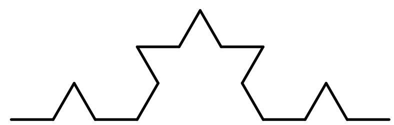


Figura 4: Curva K_2, (`F-F++F-F - F-F++F-F ++ F-F++F-F - F-F++F-F`)

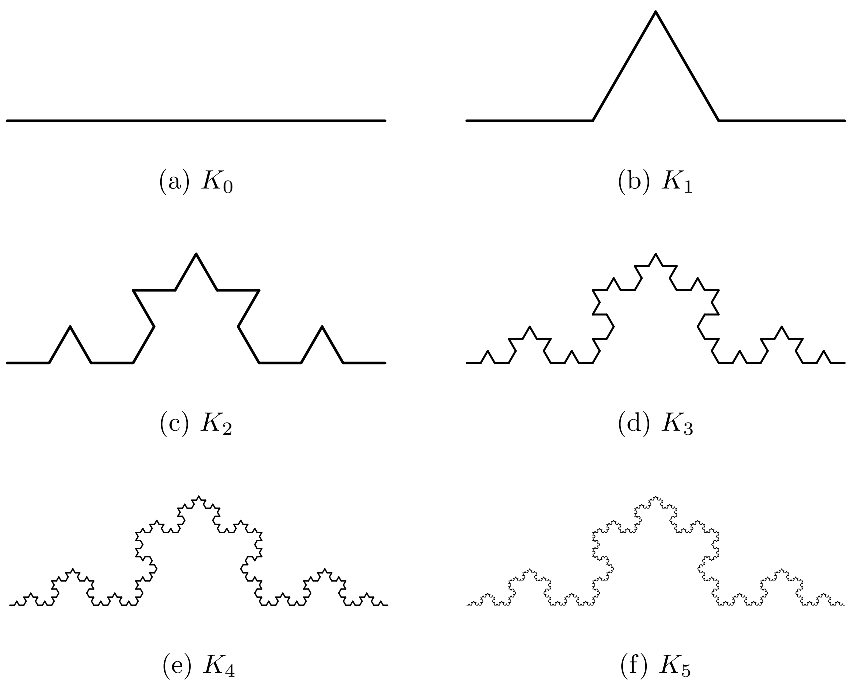
De forma geral, representando a curva K_{n-1} pela string `K`, teremos a a string `K-K++K-K`. Normalmente, na hora de descrever uma curva fractal, usa-se a mesma letra `F` ao invés de `K`. Assim sendo, a *recorrência* de uma curva de Koch é descrita pela string:

`F-F++F-F`.

Repetimos que neste caso, a letra `F` não designa um segmento de reta mas uma curva K_{n-1}, que por sinal é um segmento de reta quando $n = 1$. Na Figura 5, vemos as curvas de Koch K_n, para $n = 0, 1, \ldots, 5$.

Nos casos da Figura 5, exibimos as curvas K_n. Isto equivale a pensar que queremos partir de uma string inicial F, também chamada de *axioma* e substituir ocorrências de F por K_n. Uma aplicação bastante comum das curvas de Koch é a de desenhar figuras mais complexas a partir de axiomas diferentes e possivelmente não relacionados à estrutura da recorrência da curva fractal em questão. Assim, poderíamos partir de um axioma diferente como aquele do triângulo equilátero representado na Figura 1. Neste caso, o axioma é `-F++F++F`, e cada ocorrência de F é substituida pelo desenho de uma curva de Koch K_n, descrita pela recorrência `F-F++F-F`. Figuras como estas são chamadas de ilhas de Koch. Para o caso $n = 5$, temos então uma figura como a Figura 6.

[6] Os espaços em branco foram colocados somente para tornar mais evidente a estrutura fractal.



(a) K_0 (b) K_1

(c) K_2 (d) K_3

(e) K_4 (f) K_5

Figura 5: Primeiras curvas de Koch

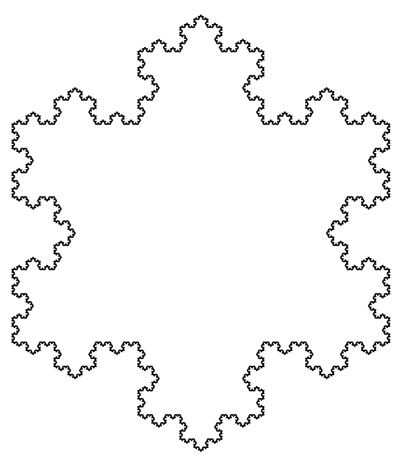


Figura 6: Ilha de Koch formada por triângulo equilátero de lado K_5

Outros fractais

Vários outros fractais podem ser gerados através de parâmetros apropriados. Basta para isto escolher o axioma, a recorrência e o ângulo β de acordo. A menos de um fator de escala, o parâmetro u não importa tanto. Também importa pouco o ponto do estado inicial. A direção do estado inicial determina a orientação final da figura.

Por exemplo, dado o axioma `F--F--F--F--F--`, a recorrência `F--F++++F--F` e o ângulo $\beta = \pi/5$ e o nível de recursão 5, obtemos o fractal da Figura 7.

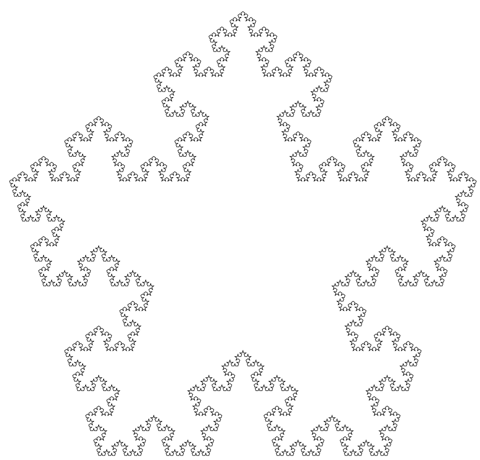


Figura 7: Fractal associado à recorrência `F--F++++F--F` com $\beta = \pi/5$

O que deve ser feito no EP

O exercício-programa a ser entregue é individual e deve ser feito usando-se as linguagens de programação: java, python, ou C. Pode-se entregar em pascal desde que seja compilado com compilador a ser combinado com a monitora.

Ele deve receber alguns parâmetros da linha de comando (será dada uma receita de como isto será feito) e imprimir os vértices que definem uma seqüência contígua de segmentos de reta (curva) que desenham o fractal relativo aos parâmetros fornecidos.

O programa deve receber os seguintes parâmetros ordenados na linha de comando:

1. o nível n de recursão da curva fractal;

Nome	β	Axioma	Recorrência
tri	$\pi/3$	`-F++F++F+++`	`F-F++F-F`
triI	$\pi/3$	`F--F--F--`	`F-F++F-F`
tetra	$\pi/4$	`F++F++F++F++`	`F-F++F-F`
tetra	$\pi/4$	`+F--F--F--F--`	`F-F++F-`
tetra	$\pi/2$	`F+F+F+F+`	`F-F+F+FF-F-F+`
tetra	$\pi/2$	`F+F+F+F+`	`F-FF+FF+F+F-F-FF+F+F-F-FF-FF+`
quad	$\pi/2$	`F+F+F+F+`	`F-F++F-F`
quadI	$\pi/2$	`F-F-F-F-`	`F-F++F-F`
quad16	$\pi/16$	`F++++++++F++++++++F++++++++F`	`F-------F++++++++++++++F-------F`
quad16I	$\pi/16$	`F--------F--------F--------F`	`F-------F++++++++++++++F-------F`
quad16EI	$\pi/16$	`F++++++++F++++++++F++++++++F++++++++` `++++++++F--------F--------F--------F`	`F-------F++++++++++++++F-------F`
koch4	$\pi/2$	`F+F+F+F`	`F-F+F+F-F`
penta	$\pi/5$	`---F++F++F++F++F+++++`	`F--F++++F--F`
penta2	$\pi/5$	`-FF--FF--FF--FF--FF-`	`F--F++++F--F`
pentaI	$\pi/5$	`F--F--F--F--F--`	`F--F++++F--F`
pentaI3	$\pi/5$	`+F++F----F++F----F++F----F++F----F++F-`	`F--F++++F--F`
hexa	$\pi/6$	`F++F++F++F++F++F++`	`F--F++++F--F`
hexaI	$\pi/6$	`-F--F--F--F--F--F-`	`F--F++++F--F`
six	$\pi/6$	`F++F++F++F++F++F++`	`F--F--F++++F++++F--F--F`
sixI	$\pi/6$	`-F--F--F--F--F--F-`	`F--F--F++++F++++F--F--F`
hepta	$\pi/7$	`+F++F++F++F++F++F++F+`	`F---F++++++F---F`
heptaI	$\pi/7$	`-F--F--F--F--F--F--F-`	`F---F++++++F---F`
32seg	$\pi/2$	`F+F+F+F+`	`-F+F-F-F+F+FF-F+F+FF+F-F-FF+F` `F-FF+F+F-FF-F-F+FF-F-F+F+F-F+`

Tabela 1: Diversos outras curvas fractais

2. o quociente π/β;
3. o string R correspondente à recorrência;
4. o string A correspondente ao axioma;
5. o comprimento u;
6. a coordenada x do estado inicial;
7. a coordenada y do estado inicial;
8. o quociente π/α, onde α é a direção do estado inicial.

Com os parâmetros 2 e 8 o que se deseja, de fato, é poder saber quais são os parâmetros β e α.

O exercício-programa *deve* fazer uso de recursividade para o desenho dos fractais quando for usar a recorrência R.

É importante que todos façam este este exercício-programa, já que o EP2 será feito a partir de modificações deste EP1.

Visualização do fractal

Como vimos, o exercício-programa deve imprimir as coordenadas cartesianas dos pontos associados aos sucessivos estados percorridos. Esta saída é extremamente simples, mas não permite uma visualização adequada do fractal gerado. Para tanto deve-se usar o programa python *dots2svg* que lê os pontos impressos e gera um arquivo gráfico em formato SVG, que é um formato gráfico vetorial e apresenta várias vantagens sobre formatos do tipo bitmap.

Uma vez gerado o arquivo SVG, pode-se visualizar o mesmo através de um plugin da Adobe a ser instalado no navegador internet, ou através de vários programas que vizualizam arquivos SVG como o inkscape, sodipodi, ou ksvg no linux.

Informações adicionais

Informações adicionais sobre o EP1 serão encontradas na URL: `http://www.ime.usp.br/~alair/mac122-04/ep1/`, inclusive alguns exemplos de execução de um protótipo.