XX Encontro de Iniciação à Pesquisa
Universidade de Fortaleza
20 à 24 de Outubro de 2014

Construção de fractais com a linguagem de programação Logo

Bruno Macabeus Mendes de Aquino[1*] (IC), Ricardo Bezerra De Menezes Guedes[2] (PQ), Francisco José Alves de Aquino[3] (PQ)

1. Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Curso de Engenharia da Computação
2. Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Professor Departamento de Telemática
3. Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Professor Departamento de Telemática
bruno.macabeus@gmail.com

Palavras-chave: Fractal. Logo. Linguagem de programação.

Resumo

Neste artigo são apresentados alguns conceitos básicos sobre fractais, a sua importância, algumas aplicações e a construção de alguns tipos usando a linguagem Logo e a IDE (*ambiente de desenvolvimento integrado*) xLogo. Os fractais podem ser usados para modelar a natureza, objetos geométricos que não podem ser modelados facilmente pela geometria euclidiana, pois podem apresentar padrões recursivos detalhados.

Introdução

A linguagem Logo foi criada em 1967 com o objetivo didático, a fim de facilitar o ensino de lógica de programação para pessoas que não tinham o domínio da matemática e principalmente para crianças [1]. O Logo é voltado principalmente para a área gráfica, facilitando a geração de desenhos. Originalmente, usava-se um robô, chamado de *turtle*, como saída gráfica. Essa origem influenciou muito a forma que se gera os gráficos. Como veremos adiante, com códigos simples e curtos é possível gerar interessantes fractais.

A geometria fractal é fascinante, não apenas pelas belas imagens resultante de pequenas sequências, mas também pela aplicação em diversos ramos, tais como: telecomunicações (construção de antenas mais eficientes [2]), e computação (compreensão de imagens [3]). Entretanto, essa é uma área do conhecimento ainda nova e que necessita de mais estudos e definições. Construir fractais com a linguagem de programação Logo é útil por facilitar análises de forma rápida e simples.

A IDE xLogo é *open source* e escrita em Java. Nela há uma tradução dos procedimentos para a língua portuguesa. É importante lembrar que na linguagem Logo não há uma rigorosa especificação, assim cada interpretador tem diversas variações. Possivelmente, um código escrito nesse artigo não será executável corretamente em outro ambiente. A escolha desse ambiente de desenvolvimento deu-se por executar nos sistemas operacionais mais usados e interpretar códigos em português.

Metodologia

Fractal

Recentemente adotado como novo ramo da ciência, ainda há necessidade de uma definição formalmente aceita [4]. K. J. Falconer definiu algumas regras que caracterizam se um objeto é ou não um fractal, como a necessidade de possuir auto-similaridade e ser gerado através de um procedimento recursivo ou iterativo [5].

A auto-similaridade descreve um objeto que apresenta um padrão de repetição independente da escala da qual é apresentado e uma parte que representa o todo [6]. Essa é uma característica marcante dos fractais e muito presente na natureza, seja na formação geográfica ou nas estruturas de seres vivos.

Uma outra definição de fractal, dita por Guzmán, são objetos matemáticos gerados por meio da iteração infinita [7]. Chamamos de “geração” quantas vezes o processo de construção foi repetido.
O matemático polonês Helge Von Koch criou uma das primeiras imagens de um fractal, a Curva de Koch, também conhecido como Koch Snowflake, em 1904, servindo de marco para o estudo [5]. A construção consiste em:

1. considere uma linha reta;
2. divida-a em três segmentos de mesmo tamanho;
3. substitua o segundo segmento por um triângulo equilátero, com a base igual ao tamanho da reta substituída;
4. apague a base do triângulo;
5. repita o segundo, terceiro e quarto passo com todos os segmentos.

Basta escolher uma seção e comparar com a geração inicial para confirmar a auto-similaridade.

O matemático polonês Waclaw Sierpinski criou os fractais Triângulo de Sierpinski e Tapete de Sierpinski usando um método diferente: o da eliminação [4].
A construção do Tapete de Sierpinski consiste em:

1. considere um quadrado;
2. divida-o em 9 quadrados de mesmo tamanho;
3. elimine o quadrado central;
4. repita o segundo e terceiro passo em cada quadrado restante.

Linguagem Logo

A linguagem Logo oferece muitos recursos para geração de gráficos. O “cursor gráfico” do ambiente é chamado de *tartaruga*, e muitas vezes abreviado de *tat*. Tal cursor não é apenas um ponto, pois além de ter coordenadas, têm direção e cor própria.
Ao mover a tartaruga, usando comandos com `parafrente` ou `paratrás` ela deixa um rastro contínuo. Por padrão, a tartaruga começa com a direção de 0°; para alterar usa-se os comandos `paradireita` e `paraesquerda`, segundo do número que corresponderá ao movimento. A cor do rastro pode ser alterada usando o comando `mudecl` seguido pelo número da cor. Um procedimento começa com `aprenda` seguido do nome dele, e termina com o comando `fim`.
Na geometria euclidiana podemos desenhar um triângulo com retas que passam em três pontos distintos. Com Logo, desenhamos um triângulo com os seguintes comandos:

```
aprenda triângulo
  paraesquerda 30
  parafrente 100
  paraesquerda 120
  parafrente 100
  paraesquerda 120
  parafrente 100
fim
```

Logo é uma linguagem inspirada em Lisp e, como herança, possui dois tipos de variáveis: palavra e lista. Variáveis começam com dois pontos e devem ser declaradas com o comando `atribua` no caso de variáveis globais e `atrlocal` para variáveis locais. Para definir que tal sentença é uma palavra, deve começar com aspa dupla. No caso de variáveis como parâmetro de uma função, devem ser definidas logo após o nome dela.

Há diversos comandos de repetição, sendo os mais usados o `repita`, `para` e `enquanto`. O escopo usado para delimitar rotinas são os colchetes.

Exemplo de um código para criar um triângulo equilátero com o tamanho fornecido:

```
aprenda triângulo :tamanho
  paraesquerda 30
  parafrente :tamanho
  paraesquerda 120
  parafrente :tamanho
  paraesquerda 120
  parafrente :tamanho
fim
```

Exemplo de um código para desenhar um quadrado:

```
aprenda quadrado
 repita 4 [
  parafrente 100
  paradireita 90
 ]
fim
```

Exemplo de um código para exibir os 10 primeiros números divisíveis por três:

```
aprenda múltiplos
 atrlocal "valoresdivisíveis []
 atrlocal "valoratual 1
 enquanto [(conte :valoresdivisíveis) < 10] [
  se (resto :valoratual 3) = 0 [
   atrlocal "valoresdivisíveis juntenofim :valoratual :valoresdivisíveis
  ]
  atrlocal "valoratual :valoratual + 1
 ]

 mostre :valoresdivisíveis
fim
```

Uma função é recursiva quando ela chama a si mesma. A linguagem Logo suporta esse tipo de chamada. Ela é muito útil para a geração de desenhos, principalmente fractais, pois permite a chamada da própria função alterando apenas os argumentos que recebe. Para evitar chamadas infinitas, sempre usa-se um critério para encerra-se.

Exemplo de função recursiva para gerar um polígono:

```
aprenda polígono :lados :geração
  se :geração = 0 [ pare ]
  parafrente 100
  paradireita 360 / :lados
  polígono :lados :geração-1
fim

aprenda desenharpolígono :lados
  polígono :lados :lados
fim
```

Use, por exemplo, `desenharpolígono 7`, para verificar o resultado. A função `desenharpolígono` chamará `polígono` passando como parâmetro a quantidade desejada de lados e a geração – quantas vezes a função será executada – que, nesse caso, é a mesma quantidade de lados da imagem.

No exemplo, a função `polígono` será executada 7 vezes. A cada vez que é chamada verifica se a geração é igual a zero, para, caso seja, encerrar a função imediatamente. Caso não seja, move a tartaruga para frente e a vira e, por fim, chama novamente a própria função passando como argumento a mesma quantidade de lados e a geração atual menos um.

É importante a próxima chamada da função receber o valor da geração menos um e verificar se é igual a zero, para não entrar em um *loop* infinito.

Ambiente de desenvolvimento xLogo

O xLogo pode ser baixo através do site oficial: http://xlogo.tuxfamily.org/

Na primeira vez que for executá-lo exibirá um painel perguntando em qual idioma você vai usar. Escolha português para ficar conforme esse artigo.

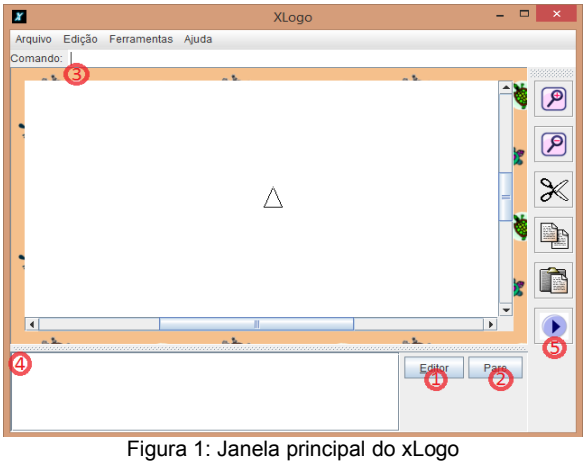

Figura 1: Janela principal do xLogo

1 – Abrir o editor de código. Para que os comandos sejam salvo e lido, clique no primeiro botão da janela.

2 – Se, durante a execução de algum código, desejar parar a execução, clique nesse botão. Isso irá apenas parar a execução do código, não apagará procedimentos e variáveis globais.

3 – Digitar o código aqui e tecle “enter” para executar algum comando.

4 – Registro dos comandos enviados e mensagens.

5 – No painel de edição de código, no canto inferior, há um campo de texto. Os comandos inseridos nele serão executados ao clicar nesse botão.

Resultados e Discussão

Curva de Koch

Com Logo é possível criar de forma simples a Curva de Koch, com um código recursivo:

```
aprenda koch :tamanho :geração
  se :geração = 0 [parafrente :tamanho pare]
  koch :tamanho/3 :geração-1
  paraesquerda 60
  koch :tamanho/3 :geração-1
  paradireita 120
  koch :tamanho/3 :geração-1
  paraesquerda 60
  koch :tamanho/3 :geração-1
fim
```

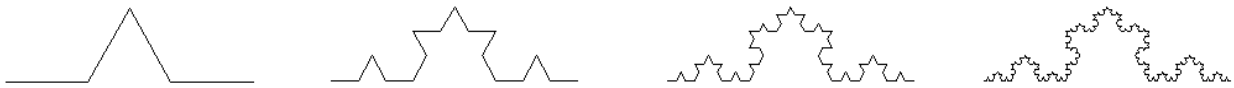

Figura 2: Resultado da função "koch" com as gerações 1, 2, 3 e 4

Exemplo de comando para executar esse código: `paradireita 90 koch 250 3`

A cada chamada da função koch enviará apenas um terço do tamanho fornecido, de forma recursiva, até a geração zero, quando, enfim, desenhará a reta.

Alterando o triângulo equilátero por outra forma geométrica, como o da imagem abaixo, é possível analisar imagens bem mais complexa e interessante.

```
aprenda koch2 :tam :ger
  se :ger = 0 [ parafrente :tam pare]
  koch2 :tam/3 :ger-1
  paraesquerda 60
  koch2 :tam/3 :ger-1
  paraesquerda 60
```

```
  koch2 :tam/3 :ger-1
  paradireita 60
  koch2 :tam/3 :ger-1
  paradireita 60
  koch2 :tam/3 :ger-1
  paradireita 60
  koch2 :tam/3 :ger-1
  paradireita 60
  koch2 :tam/3 :ger-1
  paraesquerda 60
  koch2 :tam/3 :ger-1
  paraesquerda 60
  koch2 :tam/3 :ger-1
fim
```

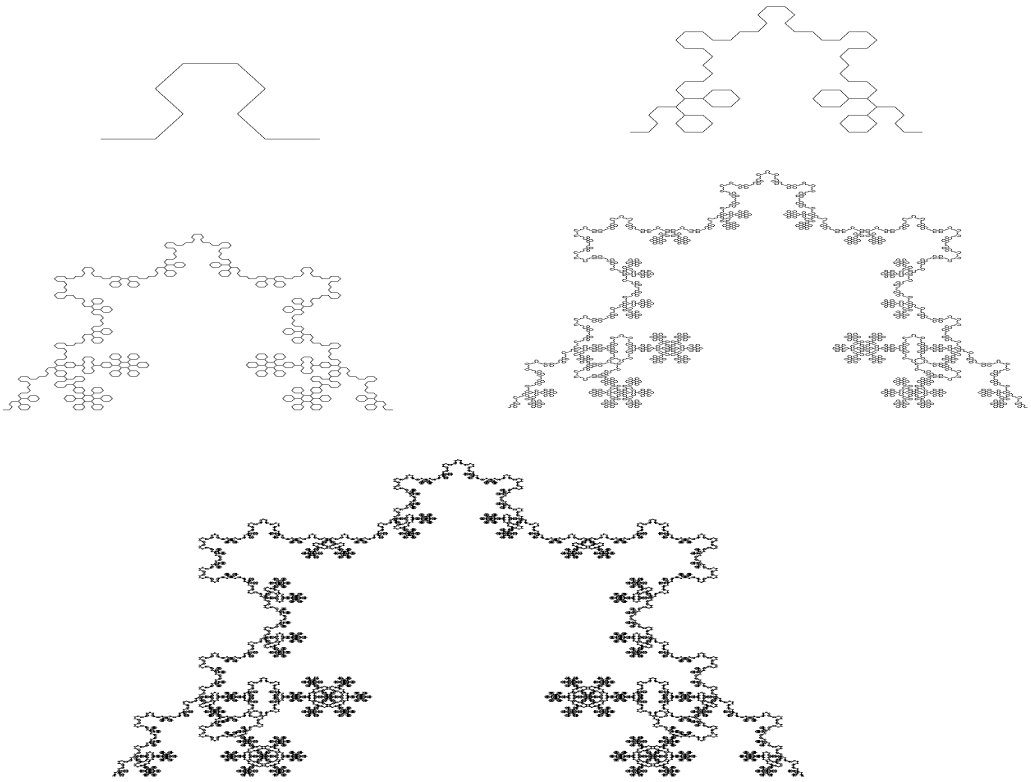

Figura 3: Resultado da função "koch2" com as gerações 1, 2, 3, 4 e 5

Note que apenas foi alterado o triângulo para outra forma geométrica. Em ambos os casos, a tartaruga começa e termina na mesma altura e não houve variação no comprimento das retas.

Tapete de Sierpinski

Em Logo, fica mais fácil usar um método diferente para gerar esse fractal. No lugar de desenhar um quadrado e dividi-lo em 9 pequenos quadrados, é melhor desenhar um quadrado para o fundo, chamar a função do tapete para desenhar os demais quadrados pequenos, ir para frente e repetir essa ação, girar a tartaruga, repetir esse processo 4 vezes, como mostrado no Quadro 1.

Quadro 1. Código para Tapete de Sierpinski

```
aprenda quadrado :tam
  repita 4 [
    parafrente :tam
    paradireita 90
  ]
fim

aprenda quadradocolorido :tam :cor
  quadrado :tam

  paradireita 45
  usenada
  parafrente 3
```

```
aprenda tapete :tam :geração
  se :geração = 0 [ pare ]

  repita 4 [
    quadradocolorido :tam :geração-1
    tapete :tam/3 :geração-1
    parafrente :tam

    quadradocolorido :tam :geração-1
    tapete :tam/3 :geração-1
    parafrente :tam

    parafrente :tam
```

```
        mudecl :cor
        pinte
        mudecl preto
        paratrás 3
        paraesquerda 45
        uselápis
      fim
```

```
          paradireita 90
        ]
      fim
```

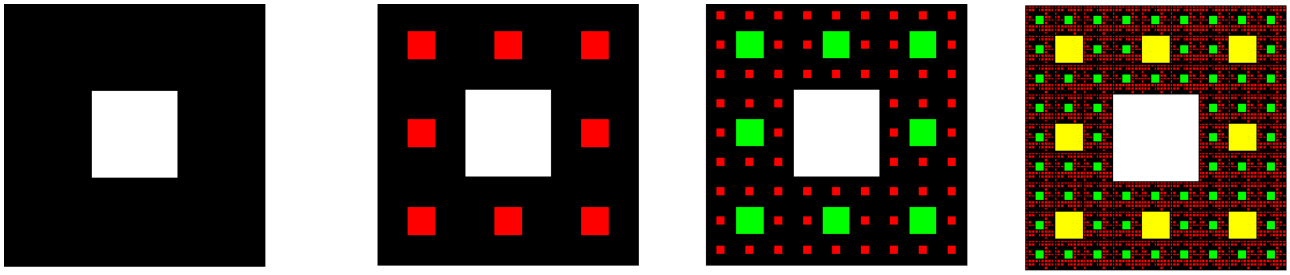

Figura 4: Resultado da função "tapete" com as gerações 1, 2, 3 e 4

Para facilitar a leitura, foi modulado o código: uma função para desenhar um quadrado, uma função para desenhar e pintar um quadrado e uma função para desenhar o tapete.

Conclusão

Neste artigo foi apresentado a construção de alguns tipos de fractais usando a linguagem Logo no ambiente de desenvolvimento xLogo. A geometria fractal consegue modelar melhor o mundo real que a geometria euclidiana. Os fractais podem ser facilmente construídos usando-se código recursivo e uma linguagem similar ao Logo, pois esta é direcionada para a criação de gráficos.

Referências

[1] GUEDES, R. M B. INTELIGÊNCIA COMPUTACIONAL: Métodos Procedimentais para Pensar, Aprender e Resolver Problemas. Dissertação (Mestrado). Universidade Federal de Pernambuco, Centro de Ciências Exatas e da Natureza, Departamento de Informática. 1998.

[2] COHEN, N. Fractal antennas and fractal resonators. Patente US 6452553 B1. Disponível no endereço https://www.google.com/patents/US6452553. Acesso 24 de agosto de 2014.

[3] HAFNER, U. FIASCO ([F]ractal [I]mage [A]nd [S]equence [CO]dec). Disponível no endereço: https://github.com/l-tamas/Fiasco. Acesso: 24 de agosto de 2014.

[4] BARBOSA, R. M. Livro Descobrindo a Geometria Fractal – para sala de aula. Ed. Autêntica, 2002.

[5] PICKOVER, C. A. The Math Book: From Pythagoras to the 57th Dimension, 250 Milestones in the History of Mathematics. Disponível em: http://books.google.com.br/books?id=JrslMKTgSZwC&pg=PA310, acesso: 24 de agosto de 2014.

[6] MANDELBROT, B. How long is the coast of britain? Statistical self-similarity and fractional dimension Science 156:636–638. 1967.

[7] CARDOSO, A.; SOUZA Jr., J. C.; OLIVEIRA, M. J. R. Fractais e Progressões Geométricas. Anais do CNMAC (2010), v.3. Disponível em: http://www.sbmac.org.br/eventos/cnmac/xxxiii_cnmac/pdf/667.pdf. Acesso: 24 de agosto de 2014.