3.º Trabalho Prático

Métodos de Programação I
LESI/LMCC
Universidade do Minho

Ano Lectivo de 2001/2002

1 Introdução

Um *fractal* é uma estrutura matemática que se repete infinitamente, e a que corresponde uma representação visual "infinitamente detalhada". Uma imagem possível é a seguinte: se usarmos dispositivos de ampliação cada vez mais potentes (lupas, microscópios), conseguiremos sempre ver cada vez mais detalhes na figura, que pareciam não existir quando a observávamos com o dispositivo anterior. Os fractais têm aplicações importantes que não serão aqui exploradas, nomeadamente na simulação de fenómenos naturais (por exemplo climatéricos).

Neste trabalho estudar-se-á a geração de fractais por hilomorfismos de determinados tipos indutivos.

2 A Biblioteca Gráfica `SOEGraphics`

Para o desenho dos fractais sugere-se a utilização da biblioteca `SOEGraphics` [1][1]. Para isso basta fazer

```
import SOEGraphics
```

O desenho de objectos gráficos não é mais do que um caso especial de processamento de *input / output* monádico. Apresentam-se aqui algumas funções básicas suficientes para o presente trabalho.

Antes de mais, vejamos como criar uma janela gráfica utilizando a seguinte função:

```
type Title = String
type Size = (Int,Int)
openWindow :: Title -> Size -> IO Window
```

O seguinte programa abre uma janela com nome "Sierpinski", de dimensão 300×300; escreve nela uma mensagem; espera que uma tecla seja premida; finalmente fecha a janela.

```
main =
   runGraphics (
   do w <- openWindow "Sierpinski" (300,300)
      drawInWindow w (text (100,200) "Aqui está uma janela")
      k <- getKey w
      closeWindow w
   )
```

A função `drawInWindow :: Window -> Graphic -> IO()` desenha um objecto de tipo `Graphic` numa janela. É naturalmente possível desenhar outro tipo de objectos além de texto; o seguinte será útil para o desenho dos fractais:

[1] distribuída como parte da *Hugs Graphics Library* em `http://haskell.org/graphics/`

```
type Point = (Int,Int)
polygon :: [Point] -> Graphic
```

Objectos gerados por esta função correspondem a polígonos descritos por uma lista de pontos.

Para colorir os fractais sugere-se a utilização da seguinte função:

```
withColor :: Color -> Graphic -> Graphic
```

Como exemplo de utilização, apresenta-se uma função que desenha numa janela um triângulo rectângulo isósceles de cor azul, descrito pelas coordenadas de um dos seus vértices e pelo comprimento dos seus catetos:

```
fillTri :: Window -> Int -> Int -> Int -> IO()
fillTri w x y size = drawInWindow w (withColor Blue
                       (polygon [(x,y),(x+size,y),(x,y-size),(x,y)]))
```

Uma nota final: o ponto $(0, 0)$ numa janela gráfica corresponde ao canto *superior esquerdo* da mesma.

3 O Triângulo de Sierpinski

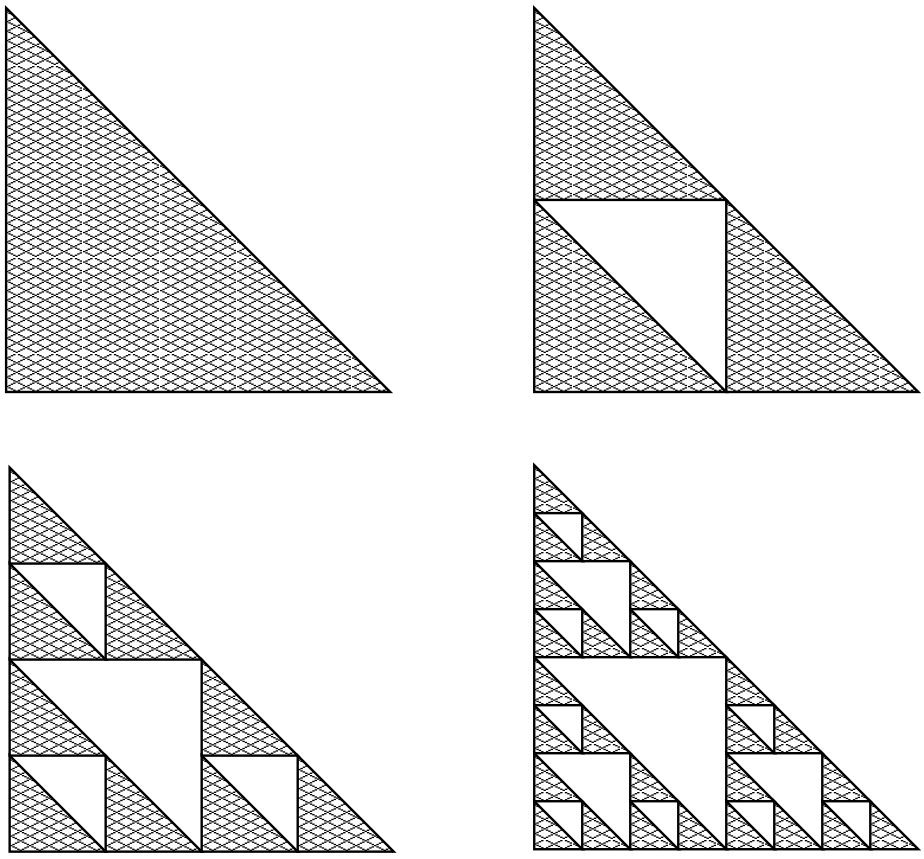

Figure 1: Construção de um triângulo de Sierpinski

3.1 Introdução

O triângulo de Sierpinski é uma figura com carácter fractal que se obtém da seguinte forma: considere-se um triângulo rectângulo e isósceles A cujos catetos têm comprimento s. A estrutura fractal é criada desenhando-se três triângulos no interior de A, todos eles rectângulos e isósceles e com catetos de comprimento $s/2$. Este passo é depois repetido para cada um dos triângulos desenhados, e assim sucessivamente. Os três primeiros passos são apresentados na Fig. 1.

Note-se que um triângulo de Sierpinski é gerado repetindo-se infinitamente o processo acima descrito; no entanto para efeitos de visualização num monitor (de resolução forçosamente finita) será necessário escolher uma representação adequada do triângulo, parando o processo recursivo a um determinado nível. A figura a desenhar é constituída por um conjunto finito de triângulos todos da mesma dimensão (por exemplo, no quarto triângulo da figura, desenhar-se-iam 27 triângulos). Cada triângulo é geometricamente descrito pelas coordenadas do seu vértice inferior esquerdo e o comprimento dos seus catetos:

```
type Side = Int
type Tri = (Point, Side)
```

3.2 Criação de um Triângulo de Sierpinski por um Hilomorfismo

A estrutura recursiva de (uma representação finita de) um triângulo de Sierpinski é captada por uma árvore ternária (cada nó é um triângulo com os respectivos três sub-triângulos). Nas folhas dessa árvore encontram-se os triângulos mais pequenos (todos da mesma dimensão), que deverão ser desenhados. Apenas estes conterão informação de carácter geométrico, tendo os nós da árvore um papel exclusivamente estrutural.

```
data TLTree = Leaf Point Side
            | Node TLTree TLTree TLTree
```

A informação geométrica guardada em cada folha consiste nas coordenadas do vértice inferior esquerdo e no lado dos catetos do respectivo triângulo.

A função `sierpinski :: Point -> Side -> Int -> [Tri]` recebe as coordenadas do vértice inferior esquerdo e a dimensão do triângulo exterior, bem como o número de níveis pretendido (critério de paragem do processo de construção do fractal), e constrói uma lista de triângulos a desenhar. Esta função é um hilomorfismo do tipo `TLTree`, composição das seguintes duas funções:

```
sierpinski :: Point -> Side -> Int -> [Tri]
sierpinski p s = apresentaSierp . (geraSierp p s)

geraSierp :: Point -> Side -> Int -> TLTree
geraSierp (x,y) s 0 = Leaf (x,y) s
geraSierp (x,y) s n = Node (geraSierp (x,y) (s `div` 2) (n-1))
                           (geraSierp (x+s`div`2,y) (s `div` 2) (n-1))
                           (geraSierp (x,y-s`div`2) (s `div` 2) (n-1))

apresentaSierp :: TLTree -> [(Point, Side)]
apresentaSierp (Leaf (x,y) s) = [((x,y),s)]
apresentaSierp (Node a b c) =
             (apresentaSierp a)++(apresentaSierp b)++(apresentaSierp c)
```

4 Trabalho a Desenvolver

4.1 Sobre o Triângulo de Sierpinski

1. Desenvolva a biblioteca "pointfree" `TLTree.hs` de forma análoga a outras bibliotecas que conhece (*eg.* `BTree.hs`, *etc.*).

2. Exprima as funções `geraSierp` e `apresentaSierp` como anamorfismo e catamorfismo, respectivamente, do tipo `TLTree`.

3. Recorrendo às funções relevantes da biblioteca gráfica, escreva uma função

   ```
   desenha :: [Tri] -> IO()
   ```

   que desenha uma lista de triângulos.

4. Teste o seu programa:

   ```
   desenha (sierpinski (50,300) 256 5)
   ```

4.2 Sobre o fractal Floco de Neve

O fractal floco de neve constrói-se a partir de um triângulo equilátero de lado s da seguinte forma: em cada lado do triângulo, e centrado em relação a ele, desenha-se um triângulo (também ele equilátero) de lado $s/3$, como ilustrado na segunda imagem da Fig. 2. Na figura obtida (uma "estrela de David") podemos identificar 6 triângulos de lado $s/3$, e a cada um deles aplicar-se-á de novo o passo anterior, e assim sucessivamente (mas note-se que alguns dos triângulos desenhados depois do primeiro sobrepõem-se a triângulos já previamente desenhados pelo que podem ser omitidos).

1. Defina um tipo de dados T que permita que a construção deste fractal seja efectuada por um hilomorfismo de T. Defina o hilomorfismo e teste-o recorrendo às funções da biblioteca gráfica.

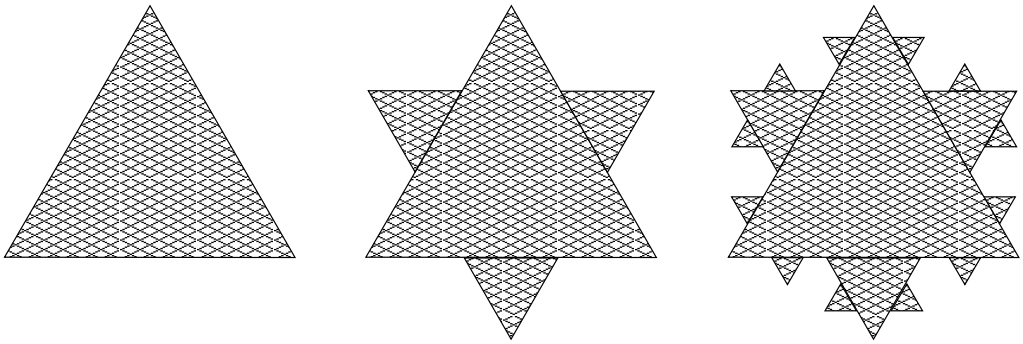

Figure 2: Construção de um floco de neve

5 Valorização

1. Modifique a sua solução para o fractal Floco de Neve por forma a que triângulos de tamanhos diferentes sejam desenhados a cores diferentes.

2. Adapte a sua resposta à alínea anterior por forma a desenhar apenas a fronteira do fractal.

3. Investigue outras formas fractais — dispõe de uma boa tutorial em

   `http://uranus.ee.auth.gr/ ngramm/fractals/tutorial3.html`)

   — e implemente-as tal como nos exercícios anteriores. Escolha exemplos relativamente simples.

References

[1] P. Hudak. The haskell school of expression: Learning functional programming through multimedia, 2000.