PROJET DE THÉORIE DES LANGAGES
STRUCTURES COMBINATOIRES ET FRACTALES

ANNEXE 1
PREMIER TEMPS : RECONNAISSANCE DES FRACTALES DE TYPE L-SYSTEM

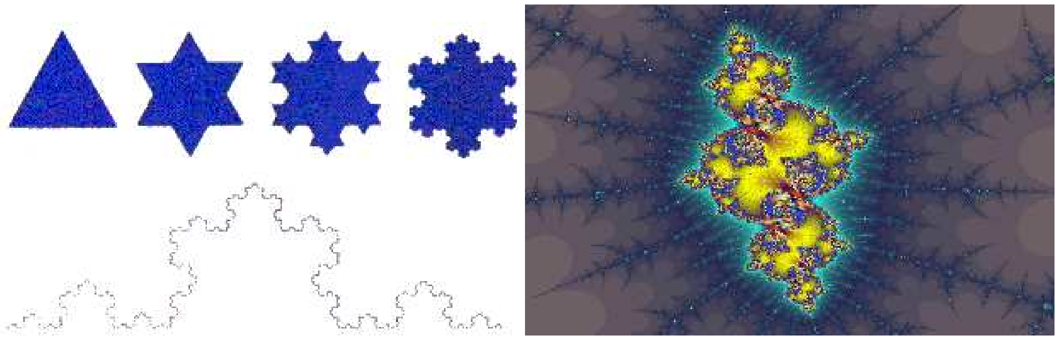
1. Introduction

Avant de dire que $4 + 2 * (1 + 2)^2$ vaut 22, il faut avoir *lu* l'expression en question, non seulement lu comme le ferait un enfant de 7 ans, mais bien plus que cela : il faut avoir *découpé* cette expression pour en isoler les éléments significatifs et, alors, reconstruire l'expression en calculant la valeur au fur et à mesure.

En clair, avant de pouvoir *évaluer*, il faut faire une *analyse lexicale* et *analyse syntaxique*. Il en est de même pour le projet, avant de pouvoir dessiner une fractale (donnée sous la forme d'une certain jeu de règles de réécriture), il va falloir apprendre à lire ce jeu de règles, c'est-à-dire le décomposer et voir son architecture.

2. Fractales et Fractint

Il y a deux grandes classes de fractales : les fractales "géométriques" (flocon de neige, feuilles de chou fleur, formes d'arbres...) et les fractales "analytiques" (comme le "bonhomme en pain d'épices" de Mandelbrot ou encore l'ensemble de Julia...). Les fractales de type analytique nécessitent la manipulation de suite de nombres complexes, comme ce type n'est pas implémanté en Pascal et que pour dessiner une telle fractale, il faut manipuler en permanence des nombres complexes de façon efficace, nous laissons de côté cette classe de fractales pour l'instant (les élèves les plus avancés pourront toujours traiter cette classe par la suite s'ils le désirent). En revanche, les fractales de type géométrique (introduites par Lindenmayer, d'où leur nom de L-systems) sont décrites par une certaine "grammaire", ce qui nous fait d'ores et déjà présentir l'utilité du couple Lex/Yacc.



Fractale de type géométrique et fractale de type analytique

Nous allons dans ce projet employer une façon de décrire les L-systems qui est compatible avec le plus grand logiciel dédié aux fractales : Fractint. Ce logiciel est une interface qui permet de dessiner des fractales, il a par ailleurs une grande quantité de formules préenregistrées et il est possible de créer ses propres formules et de voir à quel dessin elles vont aboutir. La dernière version du logiciel (la version 20.0) se trouve sous "voisinage réseau, Bdc, Logiciels, Fractint" ; comme c'est un *freeware* (= un logiciel libre), vous avez le droit de le recopier et de l'installer chez vous.

QUESTION 0. Allez dans le sous-répertoire de Fractint, puis sur la ligne de commande de DOS, tapez **`fractint`**. Choisissez alors le mode d'affichage 640.480 256 couleurs. Puis appuyez sur "t", choisissez "lsystems" puis Koch1. Validez par entrée. Vous voyez alors apparaître le flocon de neige ou "courbe de Von Koch". Faites varier le nombre d'itérations (0, puis 1, puis 2, puis 3, puis 4...) pour comprendre comment la construction se fait. Recommencez la même opération avec d'autres fractales dans **`lsystem`**.

3. L-systems

Lorsque l'on observe certains arbres, il n'est pas difficile d'imaginer qu'une branche est relativement semblable à l'arbre entier. Les L-systems utilisent cette idée. Dans le logiciel `Fractint`, un L-system est une description, sur plusieurs lignes, donnant : le nom choisi pour la fractale, un certain angle α, un axiome, la liste des règles de réécriture. Voici un exemple (il y en a d'autres à la fin de cette annexe) :

```
Koch1 {            ; après le point-virgule, ce sont des commentaires
      Angle 6       ; l'angle est fixé à 2π/6
      Axiom F- -F- -F    ; ici on déclare qu'on part de la figure F- -F- -F
      F=F+F- -F+F    ; puis à chaque itération, on fait la substitution ci-contre
   }
```

Chaque ligne de description peut éventuellement être suivie d'un point-virgule, indiquant le début d'un commentaire. Dans les membres droits des règles de réécriture, chaque non-terminal peut être précédé de m signes $+$, ce qui signifie que l'on doit tourner d'un angle de $m\alpha$ dans le sens contraire des aiguilles d'une montre. De même, si un terminal est précédé de m signes $-$, cela signifie que l'on doit tourner d'un angle de $m\alpha$ dans le sens des aiguilles d'une montre. Il n'y a pas de distinction majuscule /minuscule, les espaces et les indentations (= tabulations, touche “Tab”) multiples sont ignorées. Plusieurs instrutions peuvent rassemblées en une seule ligne si elles sont séparées par une virgule.

Voici un autre L-system : `Weed {Angle 50, Axiom +++++++++++++X,`
`X=f[@.5+++++++++X]-f[@.4- - - - - - - - - - - -!X]@.6X}`

Les symboles “+”, “-”, “f”, “@”, “[”, “]”, et “!” sont des symboles réservés. Le symbole “X” n'a pas de sens graphique, c'est juste un élément qui se réécrit selon la ligne “`X=`”. Le symbole “+” signifie “tourne à gauche de $7.2 = 360/\alpha$ degrés” (α est la valeur donnée à `Angle`). Le symbole “-” signifie “tourne à droite de 7.2 degrés”. La ligne avec “`Angle 50`” indique quelle est la valeur de référence des rotations, ici 360/50 degrés. Le symbole “f” signifie “trace une ligne dans la direction actuelle”. Les symboles “[” et “]” vont de pairs. “[” signifie “sauvegarder les données actuelles (position, orientation, longueur de ligne, etc.)” “]” signifie “réactiver les valeurs sauvegardées”. Le symbole “@” signifie “multiplie la longueur actuelle du segment par tel taux”. Par exemple “@.5” signifie que cette longueur est multiplié par $0,5$ Le symbole “!” signifie “inverser la signification du + et du -”.

4. La grammaire

QUESTION 1. En fonction des indications ci-dessus et des exemples (ci-après), proposez une grammaire $G_{L-System}$ (celle que vous utiliserez pour le couple Lex/Yacc) qui reconnaisse les L-systems. Vous devez aboutir (au nom près) à quelques choses du genre

$$\text{Lsystem} \to \text{Name } \{ \text{ Declarations } \}$$
$$\text{Declarations} \to \text{Declaration } \mathbf{,} \text{ Declarations} \mid \text{Declaration retourchariot Declarations} \mid \text{Declaration}$$
$$\text{Declaration} \to \text{DecAngle} \mid \text{DecAxiom} \mid \text{DecRegle}$$
$$\text{DecAngle} \to \text{Angle Entier}$$
$$\text{DecAxiom} \to \text{Axiom Expr}$$
$$\text{DecRegle} \to \text{Lettre} = \text{MembreDroit}$$
$$\text{MembreDroit} \to (\text{Expression, Environnement [MembreDroit] })^*$$
$$\text{Expression} \to \text{Environnement Lettre}$$
$$\text{Environnement} \to (\text{Rotation , Longueur , Inverse})^*$$
$$\text{Rotation} \to \epsilon \mid -^+ \mid +^+$$
$$\text{Longueur} \to \epsilon \mid \text{@ Decimal}$$

(Faites attention à ne pas confondre ci-dessus le vocabulaire terminal de $G_{L-System}$ et différents caractères utilisés par Lex-Yacc.) Sous Lex, vous utiliserez entre autres les lexèmes (tokens) suivants : Name, Entier, Decimal, Lettre, Inverse. Vous devez donc avoir des déclarations du type :
Lettre = [a-z,A-Z], Name = Lettre $[a-z, A-Z, 0-9]^*$, Entier = $[1-9][0-9]^*$,
Decimal = Entier.$[0-9]^*$, Inverse = [!], ...

Quand vous pensez avoir trouvé la bonne grammaire, montrez-là à votre enseignant.

QUESTION 2. Implémentez votre grammaire en Lex-Yacc. Vous pouvez tester votre programme sur le fichier **fractint.l**, qui contient nombre de L-systems. Vérifiez que votre programme ne déclare une “syntax error” uniquement pour des formules qui ne correpondent pas à des L-systems comme définis ci-dessus. Quand vous avez fini, montrez votre programme à votre enseignant.

Vous êtes alors prêts à passer au deuxième temps : le dessin de la fractale que vous venez de reconnaître.

5. Exemples

Koch1 { ; Adrian Mariano
; from The Fractal Geometry of Nature by Mandelbrot
Angle 6
Axiom F- -F- -F
F=F+F- -F+F
}

Koch2 { Angle 12 ; blablabla ignoré @;#&?!
Axiom F- - -F- - -F- - -F
F=-F+++F- - -F+
}

Koch3 { Angle 4
Axiom F-F-F-F
F=F-F+F+FF-F-F+F
}

Koch6 { axiom f+f+f+f
f=f-ff+ff+f+f-f-ff+f+f-f-ff-ff+f
angle 4
}

Dragon { Angle 8
Axiom FX
F=
y=+FX- -FY+
x=-FX++FY-
}

Peano1 { Angle 4
Axiom F-F-F-F
F=F-F+F+F+F-F-F-F+F
}

Cesaro { Angle 34
Axiom FX
F=
X=- - - -F!X!++++++++F!X!- - - -
}

FlowSnake { angle=6
axiom FL
L=FL-FR- -FR+FL++FLFL+FR-,
R=+FL-FRFR- -FR-FL++FL+FR,
F=
}

CantorDust { Angle 6
Axiom F
F=FGF
G=GGG
}

Plant09 { axiom y
x=X[-FFF][+FFF]FX
y=YFX[+Y][-Y]
angle 14
}

Bush { Angle 16
Axiom ++++F
F=FF-[-F+F+F]+[+F-F-F]
}

DragonCurve { Angle 4
Axiom X
X=X-YF-
Y=+FX+Y
}

KochCurve { Angle 6
Axiom F
F=F+F- -F+F
}

Man { ; From Ivan Daunis daunis@teleline.es
; looks like man with an odd number of iterations
Angle 8
Axiom F++F++F++F
F=-F-FF+++F+FF-F
}

Lace { ; From Ivan Daunis daunis@teleline.es
Angle 8
Axiom F++F++F++F
F=F+++F—F+F—F++F–F++F
}