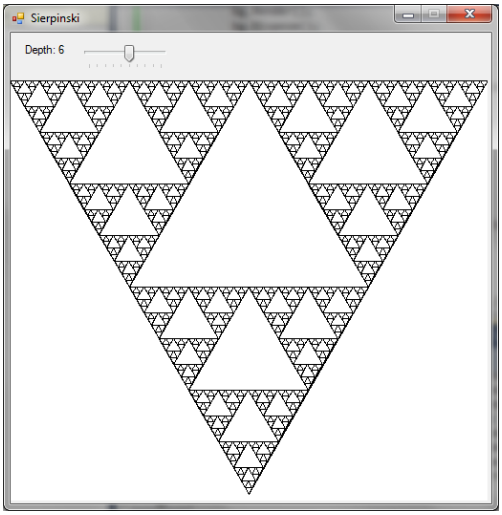
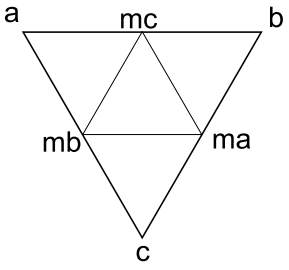
TP2

Exercice 1) Triangle de Sierpinski

Le triangle de Sierpinski est un objet fractal, c'est-à-dire que ce dernier est similaire quelque soit l'échelle à laquelle on l'observe (son aspect ne change pas si l'on zoom à l'intérieur).



Sa construction est simple : chaque triangle de Sierpinski de sommets (a,b,c), comportent trois sous triangles de Sierpinski de sommets (a,mc,mb), (mc,b,ma) et (mb,ma,c) avec ma, mb et mc les milieux des segments [bc], [ac] et [ab] :



Rappel : le milieu du segment [ab] où les coordonnées de a sont (ax,ay) et celles de b sont (bx,by) est le point de coordonnées ((ax+bx)/2,(ay+by)/2).

Télécharger le squelette d'application (http://www.esiee.fr/~perretb/I3FM/AlgoProg/Sierpinski.zip) et compléter les fonctions `drawTriangle` et `drawSierpinski` pour dessiner un triangle de Sierpinski.

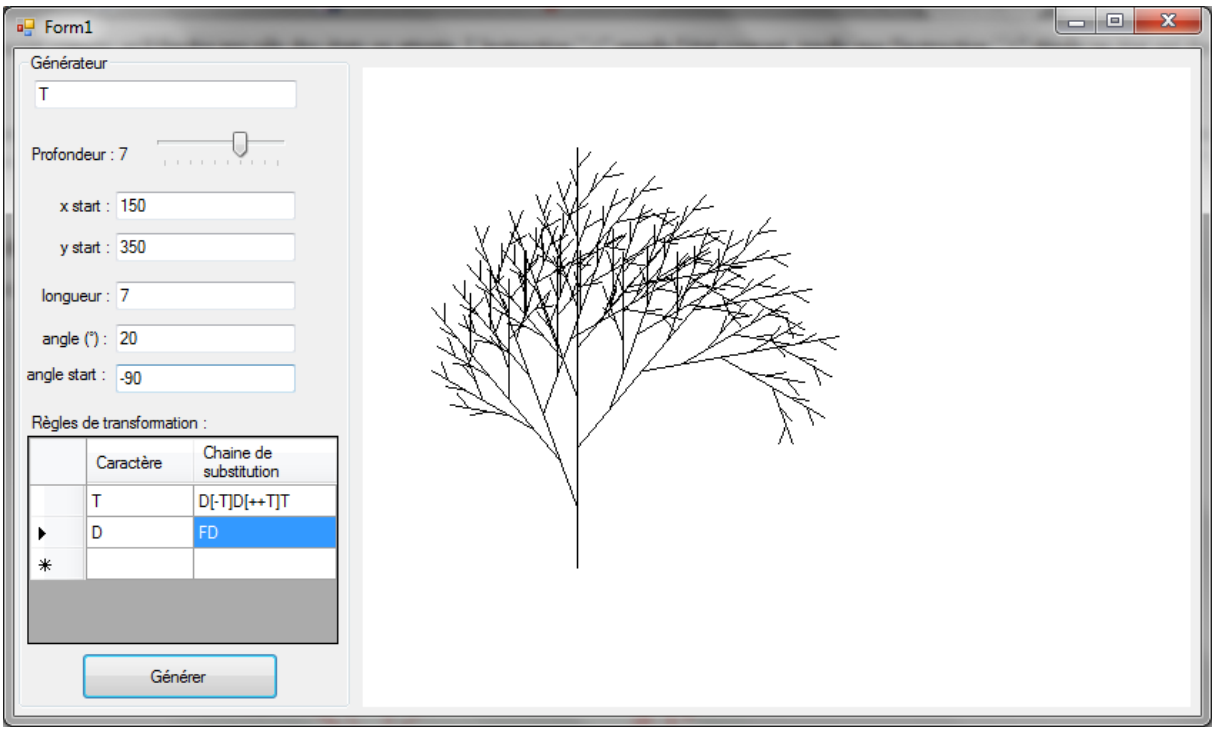
- La fonction `drawTriangle (Point a, Point b, Point c, Graphics g)` doit dessiner le triangle de sommets `(a,b,c)` dans le graphique `g.`
- La fonction `drawSierpinski(Point a, Point b, Point c, int depth, Graphics g)` doit :
  - si `depth` est égal à zéro, dessiner le triangle de sommets (a,b,c) en utilisant la fonction `drawTriangle`
  - si `depth` est supérieur à zéro il faut dessiner les 3 triangles de Sierpinski) de sommets (a,mc,mb), (mc,b,ma) et (mb,ma,c) et de profondeur `depth-1` (appel récursif)

```
1
```

Exercice 2) Système de Lindermayer

Les systèmes de Lindenmayer (ou L-System) ont été inventés pour étudier le processus de croissance des organismes vivants (algues, plantes, bactéries, ...) et sont maintenant utilisés pour simuler des objets naturels dans les applications multimédias.

Le projet à l'adresse http://www.esiee.fr/~perretb/I3FM/AlgoProg/Lindenmayer.zip contient un squelette pour l'implémentation d'un L-System que vous complèterez.



Langage de la tortue : Une tortue graphique est un petit algorithme de dessin qui repose sur un langage de programmation extrêmement simple. A chaque instant, la tortue se trouve à une position donnée et regarde dans une direction donnée. La tortue obéit à 4 instructions :

- "F" : Avancer d'un pas dans la direction courante en laissant une trace.
- "f" : Avancer d'un pas dans la direction courante en sautant.
- "-" Tourner à gauche d'un angle θ.
- "+" Tourner à droite du même angle θ.

La longueur d'un pas et la valeur de θ sont des paramètres fixes de la tortue. Si le programme contient d'autres caractères que les 4 instructions ils sont simplement ignorés par la tortue.

Par exemple avec θ=90°, et un pas de 100, le programme "F+F+F+F" permet de tracer un carré avec des côtés de longueur 100.

Pour vous aidez, la fonction suivante est donnée: `PointF Avance(PointF start, float angle, float length)` elle calcule la position de la tortue après un déplacement partant du point start selon la direction angle et sur une distance length.

Travail : codez la fonction qui implémente l'algorithme de la tortue et testez son bon fonctionnement.

```
private void trace(Graphics g, String sequence, PointF start, float curAngle, float
length)
```

Cette fonction doit par parcourir la chaine de caractère sequence et appliquer les règles de dessin de la tortue: si le caractère est

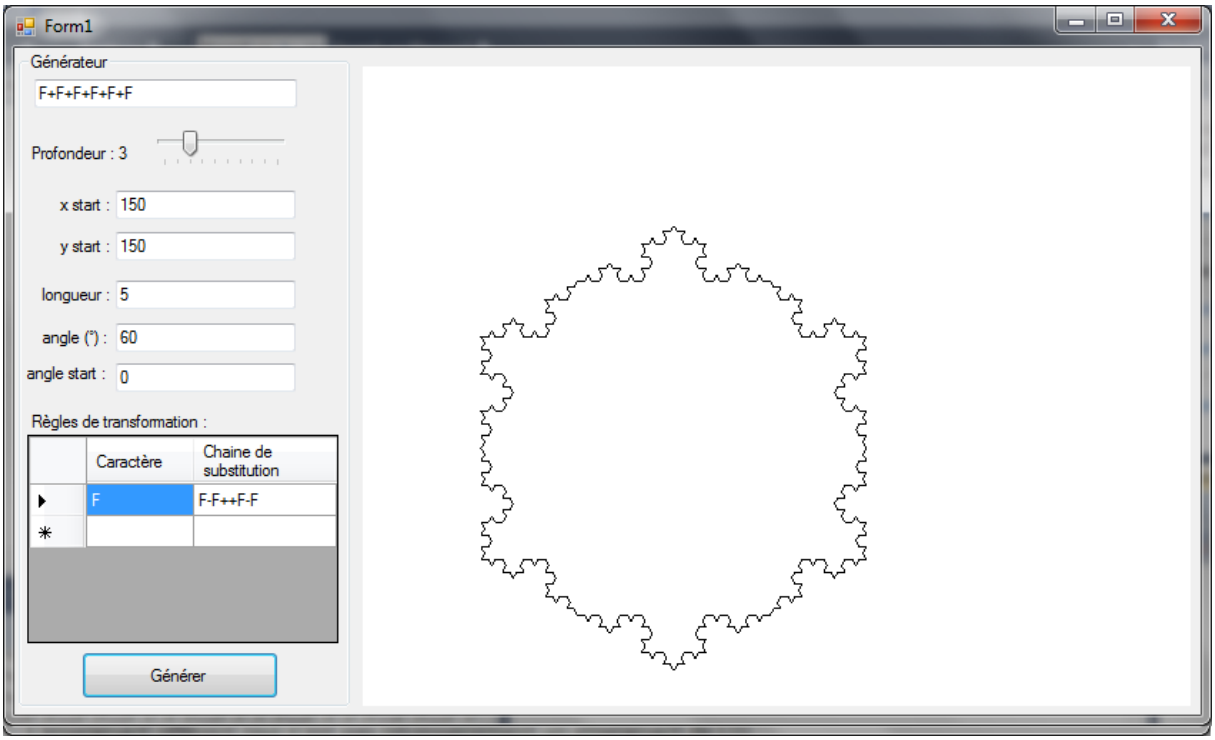
- "F" : déplacer la tortue sur la distance length en partant de la position courante suivant l'angle courant tout dessinant un trait.
- "f" : déplacer la tortue sur la distance length en partant de la position courante suivant l'angle courant sans dessiner de trait.
- "-" augmenter l'angle courant de la valeur angle (champs de la classe).
- "+" diminuer l'angle courant de la valeur angle (champs de la classe).
- tout autre caractère : ne rien faire.

L-System : Un L-System est basé sur un ensemble de règles de réécriture qui vont permettre de transformer un programme pour notre tortue en un nouveau programme. Une règle de réécriture associe un caractère à une chaine de caractères. Par exemple, la règle " T -> +FT-F" dit que le caractère "T" doit être remplacé par la chaine "+FT-F". L'application récursive de règles de transformation permet de simuler la croissance du système.

Exemple : Soit la règle : F->F-F++F-F, on souhaite appliquer 3 fois cette règle (on parle de profondeur 3) à la séquence initiale F, on obtient:

F puis F-F++F-F puis F-F++F-F- F-F++F-F++ F-F++F-F- F-F++F-F

Cet exemple correspond à un côté du flocon de Koch :



Travail :

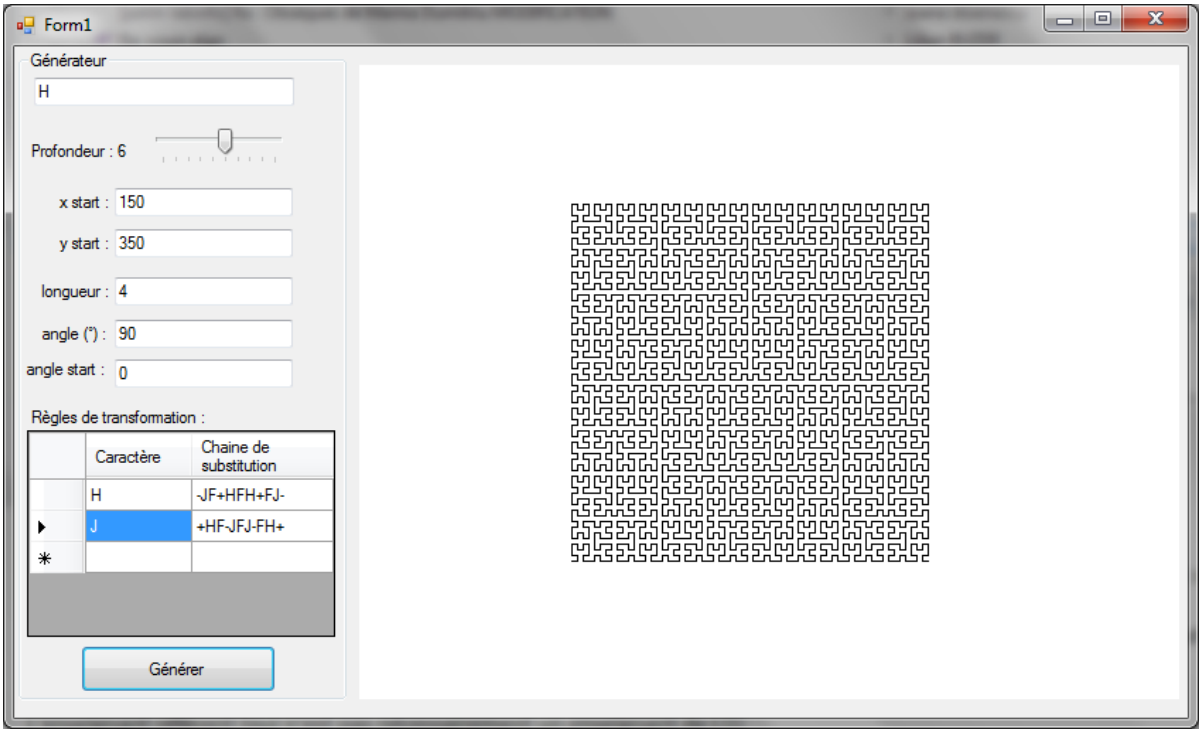
- Commencez par compléter la fonction `private Regle TrouverRegle(List<Regle> regles, char c)` qui cherche la règle de substitution d'un caractère parmi un ensemble de règles.
  - Règle est une classe qui contient un champ de type caractère (appelé Caractere) et un champ de type String (voir la définition dans le code source)
  - La fonction TrouverRegle reçoit en paramètre une liste de règles ainsi qu'un caractère et doit trouver, dans la liste, une règle dont le champ Caractere est égal au paramètre.
  - Si on trouve une telle règle, la fonction la renvoie, sinon elle renvoie null.
- Ecrivez ensuite la fonction `private StringBuilder AppliquerReglesDeSubstitution(StringBuilder sequence, int profondeur)` qui applique récursivement les règles de substitution à la séquence donnée autant de fois que demandée.
  - si profondeur est égale à zéro : la fonction renvoie simplement la séquence reçue en paramètre
  - si profondeur est supérieure à zéro :
    - la fonction parcourt tous les caractères de la séquence. Pour chaque caractère, elle regarde si une règle de substitution existe (grâce à la méthode TrouverRegle). Si une règle de substitution existe, elle remplace le caractère courant par la chaine de caractère contenu dans la règle trouvée (champs Substitution).
    - Elle s'appelle récursivement en diminuant la profondeur de 1.

Cette fonction utilise la classe StringBuilder pour travailler sur la séquence de manière efficace. Contrairement à la classe string, la classe StringBuilder permet de modifier la chaine qu'elle représente, elle permet notamment de supprimer un caractère à une position donnée (Remove) et d'insérer une chaine de caractères à une position donnée (Append).

Autres exemples de L-System:

La courbe de Hilbert-Peano : c'est un parcours fractal d'un espace carré discret à 2 dimensions avec une courbe qui maximise les relations de voisinages. Elle s'obtient à partir des règles :

- "H -> -JF+HFH+FJ-"
- "J -> +HF-JFJ-FH+"



Ile de Van Koch : Elle s'obtient avec la règle "F -> F+F-F-FF+F+F-F F+F+F+F"