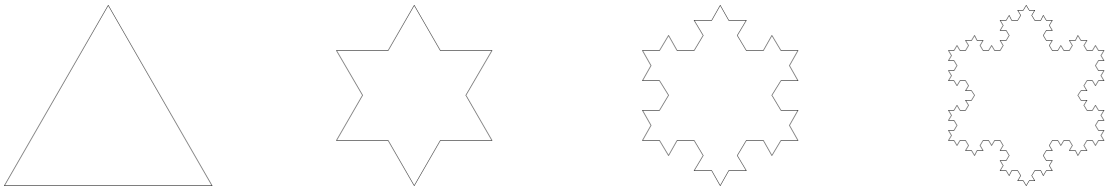
TP 8 – Introduction aux fractales

Une fractale est un objet géométrique autosimilaire, c'est à dire que l'on peut retrouver la fractale dans son intégralité en zoomant sur une de ses parties. C'est naturellement un objet limite, tous les détails d'une fractale ne peuvent pas être représentés sur une figure. Dans cette séance, nous nous intéresserons uniquement aux fractales définies par application successives d'une **transformation**. D'autres fractales existent mais leur compréhension est un peu plus compliquée. Nous les aborderons peut-être en fin de semestre si nous en avons le temps.

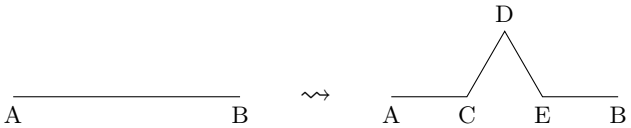
1 Flocon de von Koch

Voici les différentes étapes de la construction du flocon de von Koch par application successive de la même transformation.



Attention. Les figures précédentes ont été crées avec `Maxima` et une mise à l'échelle a été effectuée pour celle de gauche. Normalement, le triangle de cette figure de gauche peut être placé dans celle de droite et ceci en ne la touchant que sur les bords du triangle.

Le triangle correspond au flocon de von Koch avec un degré de détail valant 1 et la dernière est le flocon de Koch de degré 4. On note F_d le flocon de degré d. Pour passer du flocon F_d au flocon F_{d+1} on applique la transformation suivante à tous les segments de F_d.



Le but de cette partie est de faire dessiner à `Maxima` le flocon de von Koch F_d pour un degré de précision d quelconque et raisonnable. Pour mener à bien ce projet, nous avons besoin de quelques outils.

1.1 Fonction procédurale

Imaginons que l'on veuille créer une fonction nommée `foo` dépendant d'un paramètre `n` et qui retourne `n`/2 si `n` est pair et 3`n` + 1 si `n` est impair. Vous savez déjà comment faire appel à la commande `if` ou créer des fonctions mathématiques simples mais vous ne savez pas créer des fonctions combinant les deux. Pour cela nous avons besoin de la commande `block` d'usage très simple. Voici par exemple comment créer notre fonction `foo` :

```
(%i1) foo(n):=block(
         [resultat],
         if mod(n,2)=0 then(
            resultat:n/2
         )
         else(
            resultat:3*n+1
         )
       )$
(%i2) foo(3);foo(4);
(%o2) 10
(%o3) 2
```

La syntaxe de la fonction `block` est très simple : `block([v1,...,vn],exp1,...,expm)` exécute successivement les expressions `exp1`,...`expm`. Le premier argument de `block` signifie que les variables `v1`,...`vn` sont locales à la fonction, c'est-à-dire qu'après l'appel de la fonction et si elles ont déjà été utilisées, elles auront les mêmes valeurs qu'avant l'appel de la fonction. La valeur retournée par la fonction est celle de la dernière expression évaluée.

Exercice 1.

1. Tester l'exemple précédent et constater que la variable `resultat` est bien locale à la fonction `foo`.
2. Ecrire une fonction `affixe` qui étant donné un point sous la forme `[x,y]`, retourne le complexe `x` $+ i$ `y`. Par exemple, `affixe([1,2])` retournera `1+2%i`.
3. Ecrire une fonction `point` qui étant donné un complexe `a` $+ i$`b`, retourne le point `[a,b]`. Par exemple, `point(4%i+3)` retournera `[4,3]`. `[realpart,imagepart]`

1.2 Transformation élémentaire

L'utilisation de nombres complexes va nous faciliter la tâche. Retrouvons au préalable quelques liens entre géométrie plane et nombres complexes.

Exercice 2. Soient trois points A, B et C d'affixes respectifs z_A, z_B et z_C.

1. Quelle est l'affixe du milieu du segment $[AB]$?
2. Quelle est l'affixe du translaté de A par le vecteur $\overrightarrow{BC}$?
3. Quelle est l'affixe de l'image de A par la rotation de centre O et d'angle θ ?
4. Quelle est l'affixe de l'image de B par la rotation de centre A et d'angle θ ?

Nous avons maintenant les outils mathématiques pour créer une fonction effectuant la transformation élémentaire.

Exercice 3. On reprend la figure représentant la transformation élémentaire.

1. Trouver comment obtenir les points C, D et E à partir des deux points A, B et des opérations géométriques introduites à l'exercice précédent.
2. Ecrire une fonction `elemKoch` qui étant donnée une liste de deux points A et B retourne la liste des cinq points A, C, D, E, B correspondant à la transforamation élémentaire. On utilisera naturellement les résultats de l'exercice précédent et les fonctions `affixe` et `point`.

1.3 Mise en place

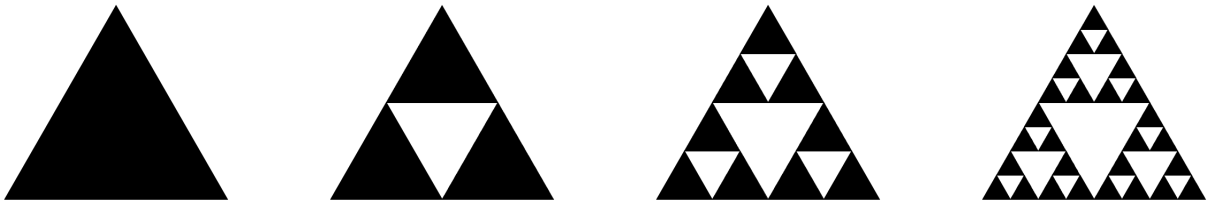
Nous sommes maintenant prêts à dessiner le flocon de von Koch avec un degré de précision donné.

Exercice 4.

1. Déterminer les coordonnées des sommets d'un triangle équilatéral de côté 1.
2. Ecrire une fonction `flocon` qui étant donné un entier `d`, retourne la liste des points du flocon de von Koch F_d de degré `d`.
3. Afficher les flocons F_1, F_2, F_3, F_4 et F_5. `[point_size]`.

2 Triangle de Sierpiński

Voici les fractales de Sierpiński S_d avec un degré de détail d valant respectivement $1, 2, 3$ et 4.



Contrairement au flocon de Koch, la transformation élémentaire ne porte sur les segments mais directement sur les triangles : un triangle (dessiné en noir) donne lieu à trois triangles plus petits. La transformation élémentaire correspond exactement au passage de la fractale S_1 à la fractale S_2.
Pour cette partie nous allons représenter les triangles comme une liste de trois points (eux-mêmes représentés comme des listes de deux nombres). Une figure de Sierpiński sera donc naturellement représentée par une liste de triangles. Par exemple, la figure S_1 est donnée par :

`[[[0,0],[0.5,sqrt(3)/2],[1,0]]].`

Il y a bien trois niveaux de crochets ci-dessus.

Exercice 5.

1. Ecrire une fonction `milieu` qui étant donnés deux points `A` et `B`, retourne le milieu du segment `[A,B]`.
2. Ecrire une fonction `elemSierpinsky` qui étant donné un triangle `T`, retourne une liste de quatre triangles correspondant à la transformation élémentaire de Sierpiński.

Si `T` est un triangle, la commande `wxdraw2d(polygon(T))` permet d'afficher `T`. Les options `color` et `fill_color` permettent alors de contrôler la couleur du bord et de remplissage. Mieux si `T1,T2,T3` sont des triangles, alors

```
(%i8) L:[polygon(T1),polygon(T2),polgon(T3)]$
(%i9) wxdraw2d(L)
```

affiche les trois triangles `T1`,`T2` et `T3`.

Exercice 6.

1. Construisez la liste `L:[1,2,5,2]` et utiliser les commandes `factorial` et `map` pour créer la liste `[1!,2!,5!,2!]`.
2. Ecrire une fonction `Sierpinsky` qui étant donné un paramètre entier `d`, retourne une liste de triangles correspondant à la fractale de Sierpińsky S_d.
3. Dessiner les fractales de Sierpińsky S_1, S_2, S_3, S_4 et S_5.

3 Tapis de Sierpińsky

Voici le tapis de Sierpińsky T_d avec un degré de détail d valant $1, 2, 3$ et 4.

Il peut être utile de représenter le carré de coté `c` ayant `A=(x,y)` comme sommet inférieur gauche, par le triplet `[x,y,c]`.

Exercice 7. En vous aidant de ce qui a été fait pour le triangle de Sierpińsky, tracer le tapis de Sierpińsky pour différents niveaux de détails.