

TP Récursif - Fractales

1 Trace (suivi) d'exécution

On appelle *trace* de l'exécution d'une fonction l'affichage des valeurs des paramètres reçus ainsi que les résultats renvoyés. En CAML la commande de "traçage" d'exécution est la directive¹ `#trace`.

Un exemple vaut mieux qu'un long discours :

```
# let f x = x + 1 ;;
  val f : int -> int = <fun>
# #trace f ;;          (* oui avec le # ! *)
  f is now traced.
```

CAML nous indique que la fonction `f` est maintenant *tracée*.

```
# f 5 ;;
  f <-- 5          (* indique que le passage de l'argument 5 à f *)
  f --> 6          (* indique que la fonction renvoie la valeur 6 *)
  - : int = 6      (* Le résultat habituel ! *)
```

Le mode trace est utile pour suivre le déroulement des calculs, pour comprendre ce qui se passe réellement. A noter la directive `#untrace` qui permet de sortir du mode.

Exercice 1.1 (Fonctions à plusieurs paramètres)

écrire deux fonctions qui calculent la moyenne de deux entiers :

1. La fonction reçoit deux paramètres distincts.
2. La fonction reçoit ses arguments sous la forme d'un couple (un seul paramètre).

Tracer l'exécution de ces deux versions.

Exercice 1.2 (Fonctions récursives)

Attention, pour les fonctions ci-dessous, ne pas prendre de grandes valeurs pour les applications. Les fonctions tracées doivent être récursives (pas de fonctions chapeaux!) et prendre tous leurs arguments sous la forme d'un seul paramètre (n-uplet).

1. écrire et tracer les fonctions `fibonacci` et `ackermann`
2. écrire et tracer les deux versions de la fonction "puissance"

1. Les directives permettent de contrôler le comportement du "top-level", charger des fichiers, ou, comme ici, de tracer l'exécution. Pour ne pas être confondues avec des valeurs, toutes les directives commencent par `'#'`.

2 Fractales

Principes

Les fractales que nous allons voir sont définies par la limite d'un procédé récursif de fabrication : un segment est remplacé par un motif composé lui-même de plusieurs segments de tailles inférieures, ensuite chacun de ces segments est à nouveau remplacé par ce motif et ainsi de suite. Chaque itération supplémentaire correspond à une nouvelle courbe, de rang supérieur. La fractale correspond à l'ensemble des points contenus dans l'intersection de ces courbes.

A fractal is a mathematical set that typically displays self-similar patterns, which means it is "the same from near as from far". Fractals may be exactly the same at every scale, or, [...], they may be nearly the same at different scales. The concept of fractal extends beyond trivial self-similarity and includes the idea of a detailed pattern repeating itself.

wikipedia²

Bibliothèque graphique

Vous aurez besoin d'utiliser les fonctions de la bibliothèque graphique³.

Tout d'abord, il faut charger le module et ouvrir la fenêtre de sortie (à ne faire qu'une seule fois) :

```
#load "graphics.cma" ;;      (* Chargement de la bibliothèque *)
open Graphics ;;             (* Ouverture du module *)
open_graph "";               (* Ouvre la fenêtre de sortie *)
```

Quelques fonctions utiles :

Pour effacer la fenêtre de sortie :

```
val clear_graph : unit -> unit
```

Positionner le point courant en (x, y) :

```
val moveto : x:int -> y:int -> unit
```

Translater le point courant selon le vecteur (dx, dy) :

```
val rmoveto : dx:int -> dy:int -> unit
```

Tracer une ligne du point courant au point (x, y) :

```
val lineto : x:int -> y:int -> unit
```

A partir du point courant, tracer le vecteur (dx, dy) :

```
val rlineto : dx:int -> dy:int -> unit
```

2. <http://en.wikipedia.org/wiki/Fractal>

3. <http://caml.inria.fr/pub/docs/manual-ocaml/libref/Graphics.html>

Exercice 2.1 (Un exemple à tester)

```

let draw x y l h =
  moveto x (y + h/2);
  rlineto (-l/2) (-h);
  rlineto (l) (2*h/3);
  rlineto (-l) 0;
  rlineto (l) (-2*h/3);
  rlineto (-l/2) (h);;

draw 50 50 100 100;;

```

Exercice 2.2 (Montagne)

On désire générer aléatoirement une "montagne" selon le principe suivant :

étape 1 On trace un segment entre 2 points quelconques.

étape 2 On calcule une nouvelle hauteur pour le milieu du segment (de manière aléatoire⁴).

étape n On applique le même processus sur chacun des nouveaux segments de droite de l'étape précédente.

Méthode :

La "courbe" d'ordre n entre les points (x, y) et (z, t) est :

$n = 0$ le segment $[(x, y), (z, t)]$

$n \neq 0$ la courbe d'ordre $n - 1$ entre (x, y) et (m, h)

suivie de la courbe d'ordre $n - 1$ entre (m, h) et (z, t) ,

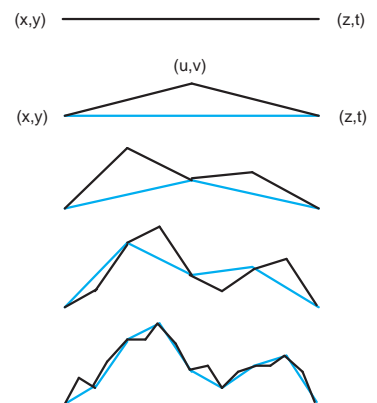
où m est le "milieu" de x et z et h une hauteur calculée aléatoirement.

Conseil : calculer la nouvelle hauteur en fonction des 2 points et éventuellement de n . On peut, par exemple, diminuer la différence de hauteur au fur et à mesure que les points se rapprochent...

Exemples (avec `int(e)` qui donne un entier aléatoire entre 0 et e) :

$$h = (y + t)/2 + \text{int}(10 * n)$$

$$h = (y + t)/2 + \text{int}(\text{abs}(z - x)/5 + 20)$$



4. Vous pouvez utiliser les fonctions du "pseudo" générateur de nombres aléatoires : le module **Random** dont vous trouverez une description dans le manuel CAML . N'oubliez pas d'initialiser le moteur !

Exercice 2.3 (Dragon)

Écrire une fonction qui trace la courbe définie par :

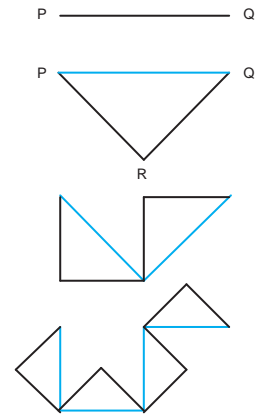
- La courbe d'ordre 1 est un vecteur entre 2 points quelconques P et Q.
- La courbe d'ordre n est la courbe d'ordre $n - 1$ entre P et R suivie de la même courbe d'ordre $n - 1$ entre R et Q (à l'envers), où PRQ est le triangle isocèle rectangle en R, et R est à droite du vecteur PQ.


Un peu d'aide :

Si P et Q sont les points de coordonnées (x,y) et (z,t) , les coordonnées (u,v) de R sont :

$$u = (x + z)/2 + (t - y)/2$$

$$v = (y + t)/2 - (z - x)/2$$




Note : Cette courbe est fondée sur un motif particulièrement simple : 

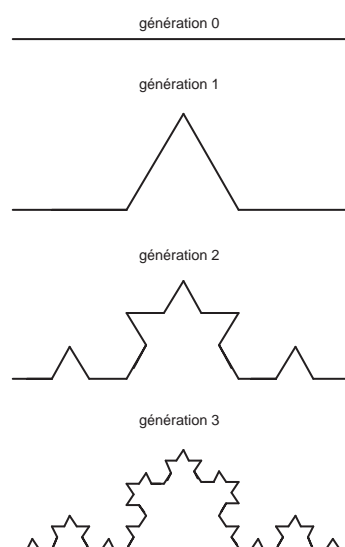
Exercice 2.4 (Flocon de von Koch (bonus))

Le flocon de von Koch est défini par :

- Le flocon d'ordre 0 est un triangle équilatéral.
- Le flocon d'ordre 1 est ce même triangle dont les cotés sont découpés en trois et sur chacun desquels s'appuient un autre triangle équilatéral au milieu.
- Le flocon d'ordre $n + 1$ consiste à prendre un flocon d'ordre n en appliquant la même opération sur chacun de ses cotés.

En terme de motifs, le flocon peut s'expliquer ainsi : il est constitué de trois courbes de von Koch placées sur les côtés d'un triangle équilatéral.

La courbe de von Koch est une fractale dont le motif de rang 1 est le suivant :  Un segment de longueur d sera donc remplacé par 4 segments de longueur $d/3$, l'angle des segments obliques est 60° .

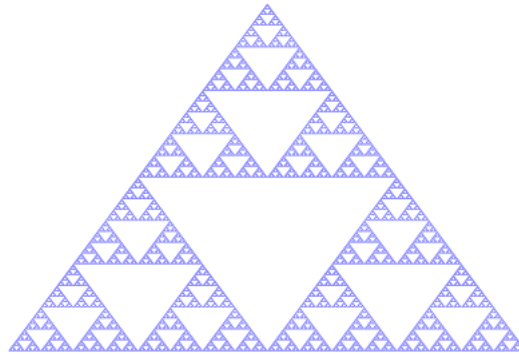


Écrire tout d'abord une fonction récursive qui trace la courbe de von Koch à partir de deux entiers n et d , n le rang de la courbe, et d la longueur du segment de départ. Puis, écrire une fonction qui trace un flocon complet au rang n donné.

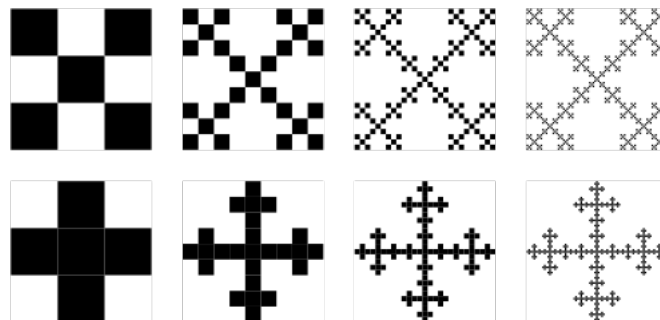
Exercice 2.5 (Triangle de Sierpinski)

Le triangle de Sierpinski est défini par :

- le triangle d'ordre 0 est un triangle équilatéral.
- le triangle d'ordre $n + 1$ est l'homotétie de rapport 0.5 du triangle d'ordre n , dupliqué trois fois et positionnés de sorte que ceux-ci se touchent deux à deux par un sommet.

**Exercice 2.6 (Vicsek)**

La fractale de Vicsek est connue également sous le nom de fractale box, elle résulte d'une construction similaire à celle de Sierpinski. Elle est définie de la façon suivante : la box de départ est une boîte décomposée en 9 sous-carreaux (3×3) dont seulement 5 d'entre eux sont conservés (soit en croix, soit en étoile). On réitère l'opération récursivement sur les carreaux restant.



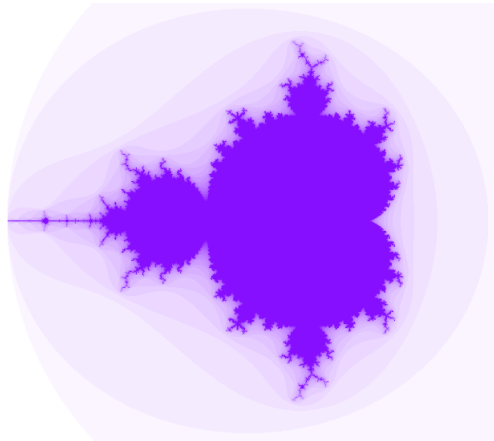
Exercice 2.7 (Ensemble de Mandelbrot (bonus))

L'ensemble de Mandelbrot⁵ est une fractale définie comme l'ensemble des points c du plan complexe pour lesquels la suite définie par récurrence par :

$$\begin{cases} z_0 &= 0 \\ z_{n+1} &= z_n^2 + c \end{cases}$$

Si nous reformulons cela sans utiliser les nombres complexes, en remplaçant z_n par le couple de réels (x_n, y_n) ⁶ et c par le couple (a, b) , alors nous obtenons :

$$\begin{cases} x_0 &= y_0 = 0 \\ x_{n+1} &= x_n^2 - y_n^2 + a \\ y_{n+1} &= 2x_n y_n + b \end{cases}$$



Pour dessiner cette fractale, nous utiliserons les commandes :

- `rgb : int -> int -> int -> Graphics.color = <fun>` construit une couleur format RGB
- `set_color: Graphics.color -> unit = <fun>` change la couleur courante.
- `plot : int -> int -> unit = <fun>` affiche un point aux coordonnées passées.

Exercice 2.8 (Ensemble de Julia (bonus))

L'ensemble de Julia⁷ est une "généralisation" de l'ensemble de Mandelbrot. En réalité, l'ensemble de Mandelbrot est, d'une certaine façon, un ensemble d'indices particuliers pour les ensembles de Julia.

Pour une valeur donnée de c , l'ensemble de Julia correspondant est la frontière de l'ensemble des valeurs initiales z_0 pour lesquelles la suite est bornée (l'ensemble de ces valeurs étant lui désigné comme l'ensemble de Julia rempli).

Valeurs de c intéressantes à tester :

$$c = \begin{cases} \begin{matrix} \mathbf{a} & \mathbf{b} \\ 0.3 & 0.5 \\ 0.285 & 0.01 \\ -1.417022285618 & 0.0099534 \\ -0.038088 & 0.9754633 \\ 0.285 & 0.013 \end{matrix} \end{cases}$$

5. http://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot

6. la condition devenant que ni x_n , ni y_n ne tendent vers l'infini (en valeur absolue).

7. http://fr.wikipedia.org/wiki/Ensemble_de_Julia