TP Récursif
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.

Extrait Wikipedia (en) [1] :

> "A fractal is a mathematical set that has a fractal dimension that usually exceeds its topological dimension and may fall between the integers. Fractals are typically self-similar patterns, where self-similar means they are "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 definition of fractal goes beyond self-similarity per se to exclude trivial self-similarity and include the idea of a detailed pattern repeating itself."

Bibliothèque graphique

Vous aurez besoin d'utiliser les fonctions de la bibliothèque graphique. [2]

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

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

Quelques fonctions utiles (extraits du manuel) :

`val clear_graph : unit -> unit`
Erase the graphics window.

`val moveto : x:int -> y:int -> unit`
Position the current point.

`val rmoveto : dx:int -> dy:int -> unit`
`rmoveto dx dy` translates the current point by the given vector.

`val lineto : x:int -> y:int -> unit`
Draw a line with endpoints the current point and the given point, and move the current point to the given point.

`val rlineto : dx:int -> dy:int -> unit`
Draw a line with endpoints the current point and the current point translated of the given vector, and move the current point to this point.

Exemple à tester :

```
let test (x,y) (z,t) =
  clear_graph ();
  set_color red;
  moveto x y ;
  lineto z t ;
  set_color blue ;
  rmoveto x y ;                      test (50,50) (50,150) ;;
  rlineto z t ;;                     test (100,100) (200,100) ;;
```

1. `http://en.wikipedia.org/wiki/Fractal`
2. `http://caml.inria.fr/pub/docs/manual-ocaml/libref/Graphics.html`

Les courbes

Conseil pour tous les exercices qui suivent : commencer les tests avec des petites valeurs pour l'ordre de la courbe àă tracer...

Exercice 1 (Montagne)

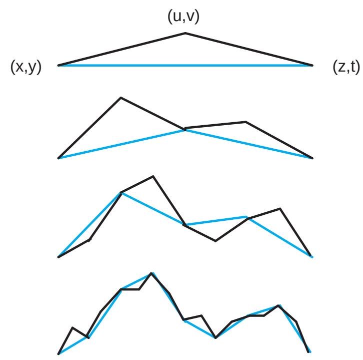
On désire générer aléatoirement une "montagne" selon le principe suivant :

étape 0 On trace un segment entre 2 points quelconques.

étape 1 On calcule une nouvelle hauteur pour le milieu du segment (aléatoire [3]).

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.

La fonction prend donc en paramètre l'odre n et les deux points (x, y) et (z, t).

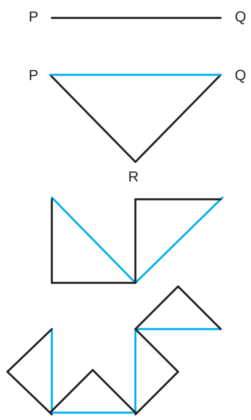
Conseil : calculer la nouvelle hauteur en fonction des 2 points et éventuellement de n. On peut, par exemple, diminuer la différence de hauteur au fur et à mesure que les points se rapprochent...

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

$h = (y + t)/2 + int(10 * n)$
$h = (y + t)/2 + int(abs(z - x)/5 + 20)$

Exercice 2 (Dragon)

Écrire une fonction qui trace la courbe définie par :



- La courbe d'ordre 0 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 :

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

Exemple d'application pour obtenir un joli dragon :

```
clear_graph ();
dragon 19 (150,150) (350,350) ;;
```

Des "bonus du dragon" en ligne (`http://algo-td.infoprepa.epita.fr/`) !

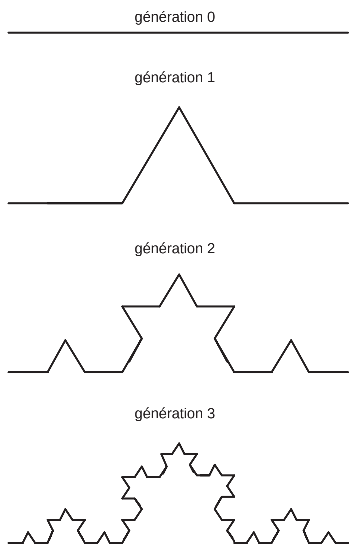
3. Vous pouvez utiliser les fonctions du "pseudo" générateur de nombres aléatoires : le module `Random` dont vous trouverez une description dans le manuel CAML . N'oubliez pas d'initialiser le moteur !

Exercice 3 (Bonus : Flocon de von Koch)

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 : un segment de longueur d sera remplacé par 4 segments de longueur d/3, l'angle des segments obliques est 60°(voir figure ci-dessous).



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

Les surfaces

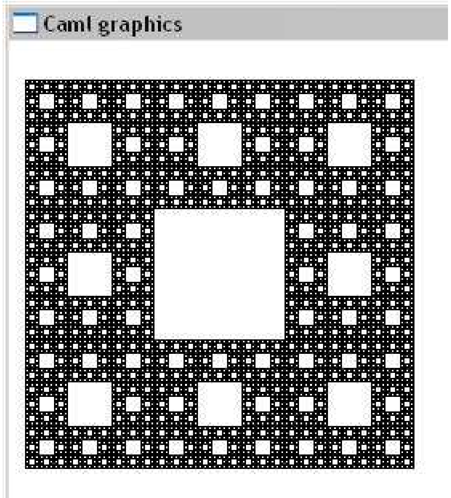
Exercice 4 (Éponge de Sierpinski)

Une fractale très simple (lorsqu'on connaît le principe !) : écrire une fonction qui génère cette "éponge" à partir d'un point de départ (coordonnées (x, y)) et de la taille du carré.

Par exemple, l'éponge ci-contre a été obtenu par :

```
let eponge (x,y) n =
  clear_graph () ;
  moveto x y ;
  dessine_carre n ;;

eponge (10,10) 243 ;;
```



La fonction suivante vous sera utile (extrait du manuel) :

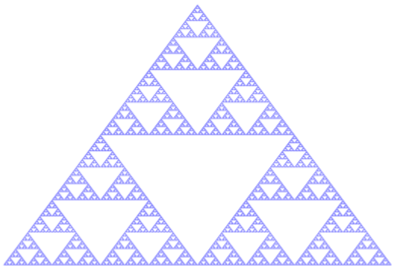
`val fill_rect : int -> int -> int -> int -> unit`

`fill_rect` x y w h fills the rectangle with lower left corner at (x, y), width w and height h, with the current color. Raise `Invalid_argument` if w or h is negative.

Exercice 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 positionés de sorte que ceux-ci se touchent deux à deux par un sommet.



Exercice 6 (Bonus : 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 definie de la façon suivante : la box de depart est une boite décomposée en 9 sous-carreaux (3 x 3) dont seulement 5 d'entre eux sont conversés (soit en croix, soit en étoile). On réitère l'opération récursivement sur les carreaux restant.