

TP Récursif Dessins

1 Patterns

Rappel

Deux méthodes pour passer à la ligne :

- Utiliser la fonction `print_newline : unit -> unit`.
- Afficher le caractère `'\n'` (peut être inséré dans une chaîne de caractères).

1.1 Must-do

1.1.1 Build me a line

Écrire la fonction `build_line n str` qui construit une chaîne de n fois la chaîne `str`.

```
# build_line ;;
- : int -> string -> string = <fun>

# build_line 5 "*" ;;
- : string = "*****"
# build_line 4 ".*" ;;
- : string = ".*.*.*.*"
```

1.1.2 Draw me a square

Écrire la fonction `square` qui affiche le motif suivant :

```
# square 5 "*" ;;
*****
*****
*****
*****
*****
- : unit = ()
```

```
# square 6 "@" ;;
@@@@@@
@@@@@@
@@@@@@
@@@@@@
@@@@@@
@@@@@@
- : unit = ()
```

```
# square ;;
- : int -> string -> unit = <fun>
```

1.1.3 Draw me a square - bis

Écrire la fonction `square2` qui affiche le motif suivant :

```
# square2 5 ("*", ".") ;;
*.*.*.*.*
.*.*.*.*.*
*.*.*.*.*
.*.*.*.*.*
*.*.*.*.*
- : unit = ()
```

```
# square2 6 ("&", " ") ;;
& & & & & &
  & & & & &
& & & & & &
  & & & & &
& & & & & &
  & & & & &
- : unit = ()
```

```
# square2 ;;
- : int -> string * string -> unit = <fun>
```

1.1.4 Draw me a triangle

Écrire la fonction `triangle` qui affiche le motif suivant :

```
# triangle 5 "*" ;;
*
**
***
****
*****
- : unit = ()
```

```
# triangle 6 "+" ;;
+
++
+++
++++
+++++
++++++
- : unit = ()
```

```
# triangle ;;
- : int -> string -> unit = <fun>
```

1.2 Bonus

1.2.1 Draw me a pyramid

Écrire la fonction `pyramid` qui affiche le motif suivant :

```
# pyramid 5 (".", "&") ;;
...**...
...***...
..****...
.*****.
.*****.
*****
- : unit = ()
```

```
# pyramid 6 ("-", "|") ;;
-----||-----
----|||-----
---|||||---
--|||||||---
-|||||||---
-|||||||---
|||||||
- : unit = ()
```

```
# pyramid ;;
- : int -> string * string -> unit = <fun>
```

1.2.2 Draw me a cross

Écrire la fonction `cross` qui affiche le motif suivant :

```
# cross 5 (".", "&") ;;
&.....&
.&.....&
..&...&..
...&.&...
....&....
...&.&...
..&...&..
.&.....&
&.....&
- : unit = ()
```

```
# cross 6 (" ", "o") ;;
o      o
o      o
o      o
o      o
o      o
o      o
o      o
o      o
o      o
o      o
o      o
- : unit = ()
```

```
# cross ;;
- : int -> string * string -> unit = <fun>
```

2 Fractales

Principe

*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."

Les fractales que nous allons voir sont définies par la limite d'un procédé récursif de fabrication : un segment (dans la première partie) 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. Les exercices de la deuxième partie sont basés sur le même principe, mais appliqué à des surfaces.

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" ;;      (* Load the library *)
open Graphics ;;             (* Open the module *)
open_graph "";;              (* Open the window *)
open_graph " 300x100";;      (* 300w × 100h window *)
```

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.
```

(draw a line from point (x, y) to point (z, t) *)*

```
let draw_line (x, y) (z, t) =
  moveto x y ;
  lineto z t ;;
```

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

2. <https://caml.inria.fr/pub/docs/manual-ocaml-4.07/libref/Graphics.html>

2.1 Les courbes

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

2.1.1 Mountain

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 (aléatoire³).

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 paramètres l'ordre n et les deux points (x, y) et (z, t) .

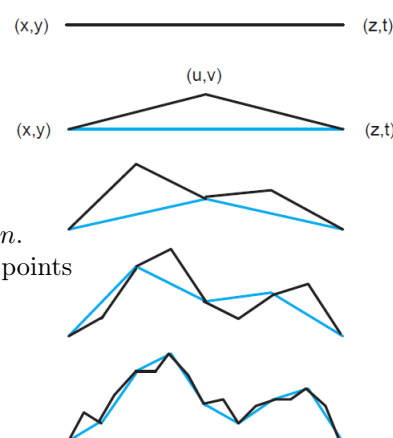
Conseil : calculer la nouvelle hauteur en fonction des 2 points et éventuellement de n .

On peut, par exemple, diminuer la différence de hauteur au fur et à mesure que les points se rapprochent...

Quelques exemples (`int(e)` donne un entier aléatoire entre 0 et $e - 1$) :

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

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



2.1.2 Dragon

Écrire une fonction qui trace la courbe définie par :

- La courbe d'ordre 0 est un segment 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 segment PQ .

Ce qui se traduit simplement comme suit : partir d'un segment de base ; puis en suivant la courbe, remplacer chaque segment par deux segments à angle droit en effectuant une rotation de 45° alternativement à droite puis à gauche.

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 :

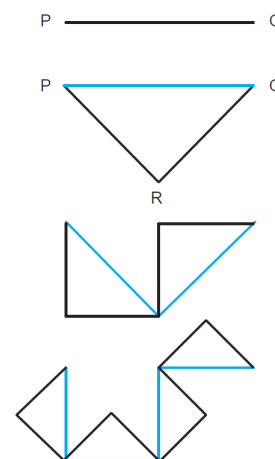
$$u = (x + z)/2 + (t - y)/2$$

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



Exemple d'application pour obtenir un joli dragon :

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



Des "bonus du dragon" (et d'autres bonus) 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 !

2.2 Les surfaces

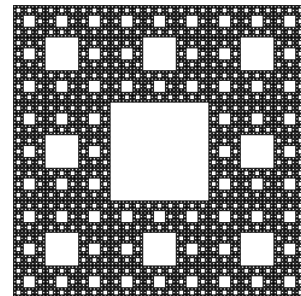
2.2.1 Sierpinski carpet (Éponge de Sierpinski)

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 obtenue par :

```
let sponge (x,y) n =
  clear_graph () ;
  moveto x y ;
  draw_sponge n ;;

sponge (10,10) 243 ;;
```



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

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

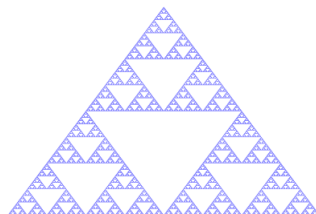
`fill_rect x y w h` fills the rectangle with lower left corner at (x,y) , width w and height h , with the current color. Raise `Invalid_argument` if w or h is negative.

Le tapis de Sierpinski se fabrique en découpant un carré départ en neuf sous carrés égaux (une grille de trois par trois) et la suppression du carré central. On réappliquera récursivement cette opération sur les huit carrés restants, et ainsi de suite (n fois).

2.2.2 Sierpinski triangle

Le triangle de Sierpinski est défini par :

- Le triangle d'ordre 0 est un triangle équilatéral.
- Le triangle d'ordre $n + 1$ est l'homothétie 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.



2.3 Bonuses...

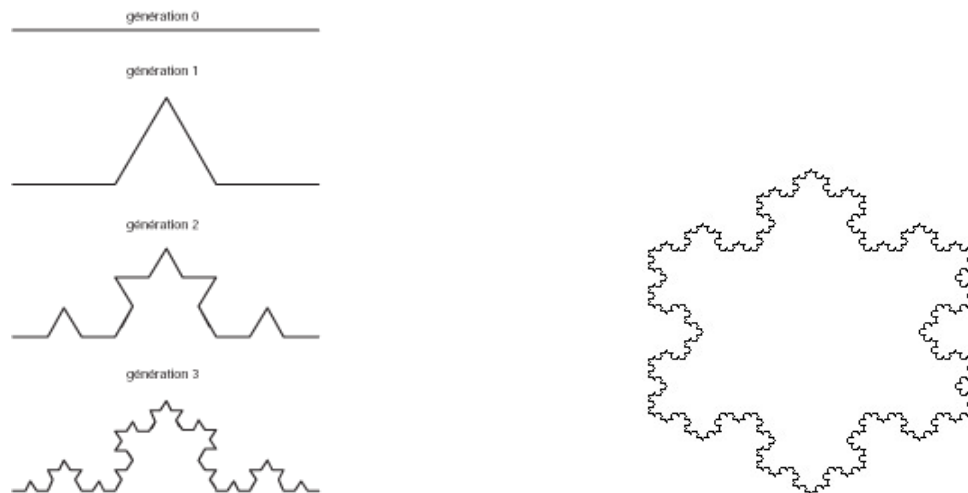
2.3.1 Koch snowflake

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'appuie 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.

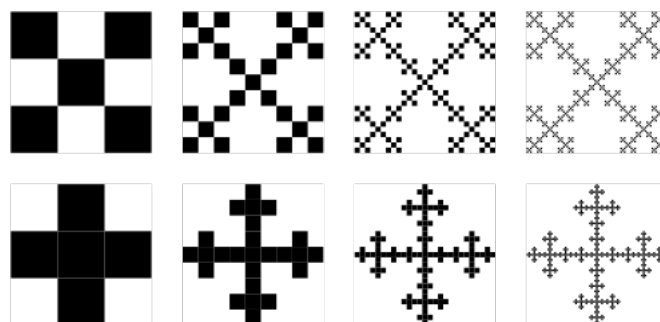
La courbe de von Koch : un segment de longueur d sera remplacé par 4 segments de longueur $d/3$, l'angle des segments obliques est 60° (voir figure ci-dessous).



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.

2.3.2 Vicsek

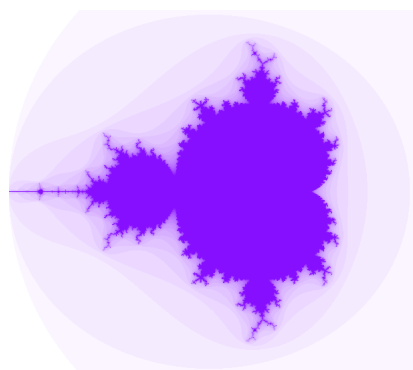
La *fractale de Vicsek* est connue également sous le nom de *fractale box*. Elle résulte d'une construction similaire à celle de Sierpinski. Elle est définie de la façon suivante : la box de départ est une boîte décomposée en 9 sous-carreaux (3 x 3) dont seulement 5 d'entre eux sont conservés (soit en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



2.3.3 Mandelbrot

Pour dessiner cette fractale de Mandelbrot ⁴, 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.
```



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