

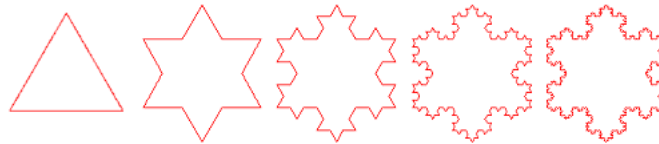
TP1 : Dessiner des Fractales

Nous vous proposons dans ce premier TP d'écrire un programme pour dessiner des images *fractales*.

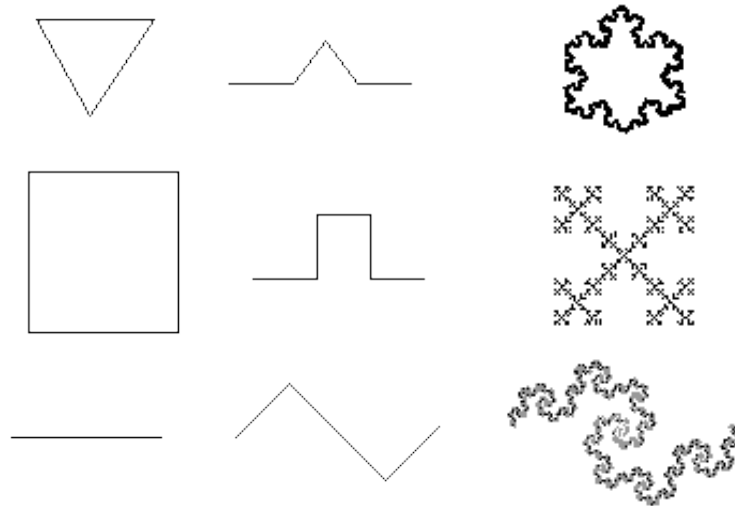
1 Qu'est-ce qu'une image fractale ?

Les images fractales que nous considérons dans ce TP sont obtenues par un processus récursif qui consiste à appliquer une transformation à chaque côté d'une figure géométrique *simple* (comme un segment, un triangle ou un carré) afin d'obtenir une nouvelle figure géométrique *plus complexe*, puis à recommencer ce processus récursivement sur chaque côté de cette nouvelle figure.

Par exemple, la célèbre fractale du “flocon de neige de von Koch” s'obtient en appliquant à chaque côté d'un triangle équilatéral la transformation qui consiste à remplacer le 1/3 central de chaque côté par 2 segments ayant la même longueur que celle prélevée. Au bout de quelques itérations, le résultat ressemble à un flocon de neige, comme le montre la figure suivante :



D'autres images peuvent être obtenues en changeant la figure géométrique de départ ainsi que le motif appliqué pour les transformations. Par exemple, la figure suivante montre trois exemples d'images fractales : la première colonne est la figure géométrique de départ, la seconde est le motif appliqué et enfin la troisième est l'image obtenue après un certain nombre d'itération.



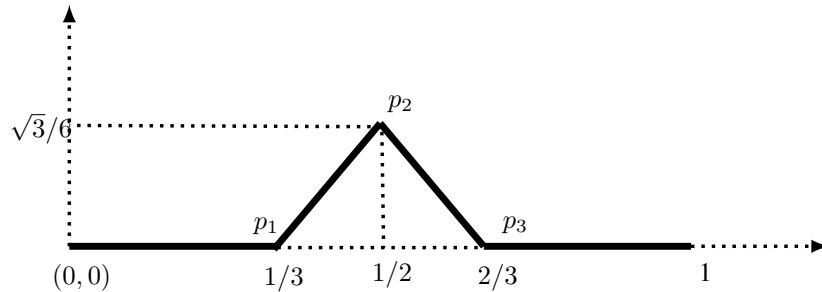
2 Un peu de mathématiques

Avant de pouvoir dessiner ces images fractales, il faut tout d'abord trouver une représentation mathématique des motifs puis déterminer les calculs à effectuer pour appliquer les transformations.

2.1 Représentation des motifs

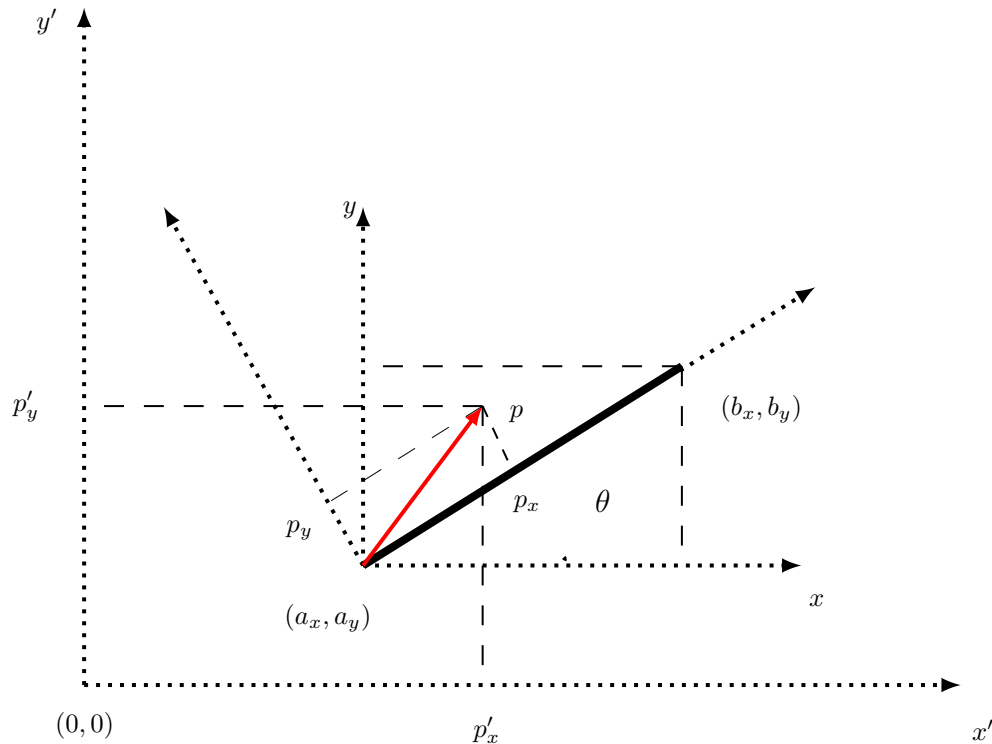
Un motif sera décrit par les coordonnées des segments le constituant, en prenant comme conventions que le point de départ du motif a pour coordonnées $(0, 0)$ et le point d'arrivée $(1, 0)$. Les points de départ et d'arrivée ayant toujours les mêmes coordonnées, seuls les points intermédiaires seront donc nécessaires.

Par exemple, le motif du flocon de neige de *von Koch* sera représenté par trois points p_1 , p_2 et p_3 de coordonnées respectives $(1/3, 0)$, $(1/2, \sqrt{3}/6)$ et $(2/3, 0)$ comme le montre la figure suivante :



2.2 Rotation vectorielle

La difficulté algorithmique de ce TP est celle qui consiste à calculer la position d'un point p du motif pour un segment quelconque $[a; b]$. Soient (a_x, a_y) et (b_x, b_y) les coordonnées de départ et d'arrivée de ce segment. Comme indiqué dans la figure suivante, le problème consiste à trouver les coordonnées (p'_x, p'_y) du point p dans le repère $x'y'$, connaissant ses coordonnées (p_x, p_y) dans le repère xy .



Pour cela, il faut tout d'abord ramener les coordonnées de p dans le repère xy , puis effectuer une rotation vectorielle suivant l'angle θ formé par le segment $[a; b]$ avec l'axe xy . Les équations pour effectuer ces transformations sont résumées ci-dessous.

On commence par calculer la longueur d du segment $[a; b]$, puis on calcule le cosinus et le sinus de θ .

$$\begin{aligned}d_x &= b_x - a_x \\d_y &= b_y - a_y \\d &= \sqrt{d_x^2 + d_y^2} \\ \cos \theta &= d_x/d \\ \sin \theta &= d_y/d\end{aligned}$$

Enfin, les coordonnées (p_x, p_y) sont données par les deux équations suivantes :

$$\begin{aligned}p'_x &= a_x + d \times (p_x \cdot \cos \theta - p_y \cdot \sin \theta) \\p'_y &= a_y + d \times (p_x \cdot \sin \theta + p_y \cdot \cos \theta)\end{aligned}$$

Exercice 1 En utilisant les équations définies dans ci-dessus, écrire une fonction transformation de type

```
(float * float) -> (float * float) -> (float * float) -> (float * float)
```

telle que transformation (ax, ay) (bx, by) (px, py) retourne les coordonnées du point (px', py').

3 Bibliothèque graphique

Pour dessiner les images fractales, nous allons utiliser la bibliothèque graphique Graphics prédéfinie dans OBJECTIVE CAML. Pour utiliser cette bibliothèque dans l'interpréteur, vous devez taper la commande suivante :

```
ocaml graphics.cma
```

puis taper tout de suite la commande open Graphics. Pour compiler votre programme, vous devez exécuter le compilateur avec la commande suivante :

```
ocamlc -o fractales graphics.cma monprogramme.ml
```

Pour utiliser cette bibliothèque graphique, vous devez tout d'abord ouvrir une fenêtre graphique à l'aide de la fonction open_graph : string -> unit. Par exemple, l'appel open_graph " 300x100" ouvre une fenêtre graphique de 300 pixels de large et 100 pixels de haut. Le pixel en (0, 0) étant situé en bas à gauche.

Pour tracer des lignes, on utilisera les fonctions moveto : int -> int -> unit et lineto : int -> int -> unit qui permettent respectivement de positionner le point courant et de tracer une ligne du point courant vers le point passé en argument, ce point devenant ensuite le point courant.

Exercice 2 En utilisant les fonctions précédentes, écrire une fonction tracer_ligne : (float * float) -> (float * float) -> unit telle que tracer_ligne (ax, ay) (bx, by) trace le segment entre les points (ax, ay) et (bx, by).

4 Dessiner les fractales

Il est temps maintenant de dessiner des images fractales.

4.1 Flocon de neige

Exercice 3 En utilisant les fonctions définies dans les sections précédentes, écrire une fonction

```
motif_flocon : (float * float) -> (float * float) -> unit
```

pour dessiner le motif présenté section 2.1 à partir d'un segment de coordonnées (ax, ay) et (bx, by) .

Exercice 4 Afin de généraliser la fonction `motif_flocon` pour afficher un segment d'un flocon de neige, écrire la fonction

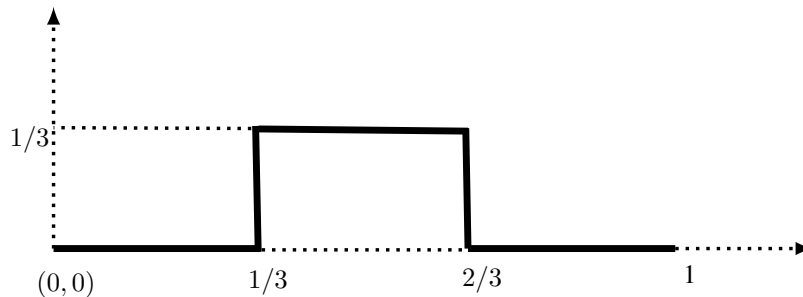
```
flocon : int -> (float * float) -> (float * float) -> unit
```

telle que `flocon n (ax, ay) (bx, by)` applique n fois récursivement le motif du flocon de neige de *von Koch* à partir d'un segment de coordonnées (ax, ay) et (bx, by) .

Exercice 5 Afin de dessiner complètement le flocon de neige, écrire une expression Caml qui appelle la fonction `flocon` pour chaque segment du triangle de départ.

4.2 Une autre fractale

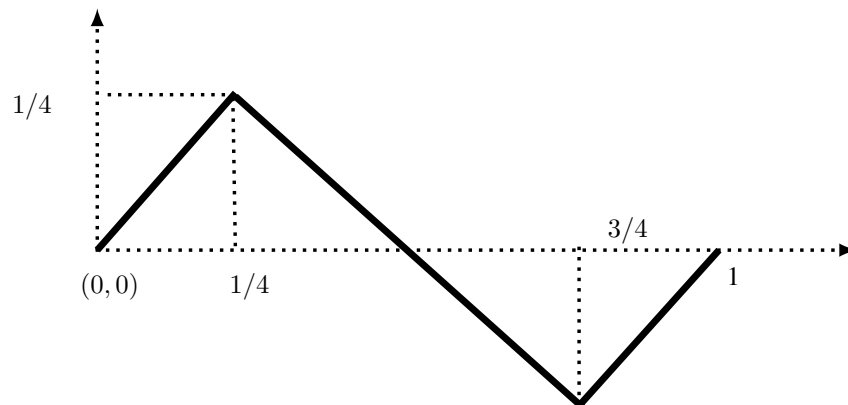
En utilisant les coordonnées du motif ci-dessous :



écrire une fonction `carre : int -> (float * float) -> (float * float) -> unit` pour dessiner une image fractale à partir d'un carré.

4.3 Une dernière fractale

En utilisant le motif ci-dessous :



écrire une fonction `julia : int -> (float * float) -> (float * float) -> unit` pour dessiner une image fractale à partir d'un segment.