Universite Paris Diderot
Licence

Programmation Fonctionnelle
Annee 2010-2011

TP n°7

Graphismes

Dans ce TP, on introduit le module `Graphics` de la librairie de OCaml. Pour plus de details, voir `http://caml.inria.fr/pub/docs/manual-ocaml/libref/Graphics.html`.

Voici des extraits de la documentation du module `Graphics` qui illustrent les primitives utilisees dans la solution.

{ `open_graph : string -> unit`
Show the graphics window or switch the screen to graphic mode. The graphics window is cleared and the current point is set to (0, 0). The string argument is used to pass optional information on the desired graphics mode, the graphics window size, and so on. Its interpretation is implementation-dependent. If the empty string is given, a sensible default is selected.

{ `moveto : int -> int -> unit`
Position the current point.

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

Pour pouvoir utiliser le mode interactif, il est necessaire de charger le module graphics.cma dans le toplevel OCaml. Il est possible de le charger de deux manieres di erentes :

{ Lancer le toplevel OCaml avec comme argument graphics.cma

{ Charger le module Graphics dans une session toplevel avec la directive toplevel suivante : `#load "graphics.cma";;`.

Pour pouvoir tester vos programmes, vous pourrez utiliser la fonction `Graphics.read_key` qui attend que l'utilisateur appuie sur une touche.

Exercice 1 [Fonctions trigonometriques] A cher dans une fenêtre graphique le graphe de la fonction sinus entre 0 et 2π en rouge, et aussi du cosinus en bleu.

Exercice 2 [La courbe du dragon]

{ La courbe du dragon d'ordre 1 entre les points A et B est le segment $[AB]$;

{ Soit C le point formant un triangle isocele rectangle en C avec A et B, a droite de $\overrightarrow{AB}$. Si A et B sont de coordonnees respectivement (x, y) et (z, t), alors C est de coordonnees (u, v) avec :
$u = (x + z)/2 + (t - y)/2$
$v = (y + t)/2 - (z - x)/2$
La courbe du dragon d'ordre n est :
{ La courbe du dragon d'ordre $n - 1$ entre A et C
{ La courbe du dragon d'ordre $n - 1$ entre B et C

Ecrire un programme qui dessine la courbe du dragon.

1 Ensemble de Mandelbrot

Exercice 3 Pour cette exercice, nous utiliserons le module et `Complex` de la bibliotheque standard OCaml. On rappelle juste qu'a chaque point du plan de coordonnees (x, y), on peut associer un nombre complexe note $x + iy$. Pour representer les complexes, la bibliotheque `Complex` de nit le type :

```
type t = {
    re : float;
    im : float;
   }
```

ainsi que toutes les operations de base (addition, multiplication...).

1. Commencez par ouvrir une fenêtre graphique de 500 par 500 et ecrivez une fonction `remplit : unit -> unit` qui va remplir point par point cette fenêtre de rouge. On utilisera uniquement des fonctions recursives (pas de boucles). Le but etant de parcourir tout les points, il est interdit d'uliser les fonctions de remplissages du module `Graphics`.
2. Pour chaque point du plan d'a xe c, on de nit la suite de Mandelbrot par :

$$\begin{array}{rcl} c_0 & = & c \\ c_{n+1} & = & c_n^2 + c \end{array}$$

   L'ensemble de Mandelbrot est l'ensemble des points pour lesquels la norme des elements de cette suite ne tend pas vers l'in ni.
   On peut montrer que si la norme depasse 2, alors la suite tend vers l'in ni.
   On fera l'approximation suivante : si au bout de 50 iterations la norme est toujours inferieure a 2, alors le point est dans l'ensemble.
   Ecrire une fonction
   `diverge : int -> Complex.t -> bool`
   telle que `diverge nb_iter c` renvoie `true` lorsqu'en moins de `nb_iter` iterations, la norme de `c` est plus grande que 2, et `false` sinon.
   Veri er que la suite diverge en $(1, 1)$ mais pas en $(0, 0)$.
3. Soit `c : Complex.t` le point en bas a gauche du carre que nous voulons a cher et `largeur :` oat sa largeur. La fonction suivante renvoie l'a xe correspondant au point de coordonnees (x, y) sur la fenêtre graphique 500x500 :

```
let calcule_affixe c largeur x y =
  Complex.add c
      {re= (float_of_int x) *. largeur /. 500. ;
       im= (float_of_int y) *. largeur /. 500. }
```

   transformer la fonction `remplit` en une fonction
   `mandelbrot : int -> Complex.t -> float -> unit`
   qui prend en parametres le nombre maximum d'iterations pour chaque point, `c` et `largeur`, et qui dessine l'ensemble de Mandelbrot.
   Calculer `mandelbrot 50 re= -1.5; im= -1. 2.`
4. Modi er le programme pour que la couleur soit choisie en fonction de la vitesse de divergence de la suite (nombre d'iterations pour depasser 2).
5. L'ensemble de Julia pour une valeur complexe z se calcule comme l'ensemble de Mandelbrot en remplacant la suite de Mandelbrot par :

$$\begin{array}{rcl} c_0 & = & c \\ c_{n+1} & = & c_n^2 + z \end{array}$$

   Dessiner l'ensemble de Julia, par exemple pour $z = 0,75i$