Corrigé feuille d'exercices n°10 de l'option d'informatique.

Du bon usage du graphisme en Caml (bis)

Exercices n°1

Voici les principales fonctions sur les nombres complexes :

```
1   type complexe = {re : float;im : float};;
2
3   let zero = {re = 0.; im = 0.};;   (* Définition de quelques constantes *)
4   let un = {re = 1.; im = 0.};;
5   let Pi = 4. *. atan 1.;;
6
7   let somme (x : complexe) (y : complexe) = (* somme de deux complexes *)
8       {re = x.re  +. y.re;
9       im = x.im +. y.im};;
10
11  let difference (x : complexe) (y : complexe) = (* différence de deux complexes *)
12      {re = x.re  -. y.re;
13      im = x.im -. y.im};;
14
15  let minus_compl (z : complexe) =  (* opposée d'un complexe *)
16       {re = -. z.re ; im = -. z.im};;
17
18  let produit (x : complexe) (y : complexe) = (* produit de deux complexes *)
19      {re = (x.re  *. y.re) -. (x.im *. y.im);
20      im = (x.re *. y.im) +. (x.im *. y.re)};;
21
22  let rec puissance z n = (* complexe élevé à la puisance n *)
23      match n with
24        0 -> un
25      | n -> produit z (puissance z (n-1)) ;;
26
27  let module (x : complexe) =  (* module d'un complexe *)
28    sqrt (x.re *. x.re +. x.im *. x.im);;
29
30  let module1 (x : complexe) = (* autre fonction "module" plus précise *)
31     if abs_float(x.re) >. abs_float(x.im)
32         then
33              let r = x.im /. x.re in
34              abs_float(x.re) *. sqrt( 1. +. r *. r)
35        else
36              if x.im =. 0.0
37                   then 0.0
38                   else
39                         let r = x.re /. x.im in
40                         abs_float(x.im) *. sqrt( 1. +. r *. r);;
41
42  let Re (x : complexe) = x.re;; (* partie réelle d'un complexe *)
```

```
43  let Im (x : complexe) = x.im;; (* partie imaginaire d'un complexe *)
44
45  let print_compl z =  (* affichage d'un complexe *)
46        print_float (Re z); print_string "+I*"; print_float (Im z);
47        print_newline ();;
48
49  let racine (z:complexe) = (* racine carrée d'un complexe *)
50     if Im(z) >. 0. then
51           {re = sqrt((Re(z) +. module(z)) /. 2.) ; im = sqrt((module(z) -. Re(z)) /. 2.)}
52       else
53          {re = sqrt((Re(z) +. module(z)) /. 2.) ; im = -. sqrt((module(z) -. Re(z)) /. 2.)}

54  #open "graphics";;
55  open_graph "";;
56
57  let WinXmin = ref 0.;; let WinXmax = ref 0.;;
58  let WinYmin = ref 0.;; let WinYmax = ref 0.;;
59  let WinXcor = ref 0.;; let WinYcor = ref 0.;;
60
61  (* Pour définir la fenêtre du plan complexe correspondant à la fenêtre graphique *)
62  let window x1 y1 x2 y2 =
63        WinXmin := x1;  WinXmax := x2;
64        WinYmin := y1;  WinYmax := y2;
65        WinXcor := float_of_int (size_x ()) /. (x2 -. x1);
66        WinYcor := float_of_int (size_y ()) /. (y2 -. y1);;
67
68  let couleurs = [|black;red;blue;yellow;cyan;green;magenta|];;
69  let nbcouleurs = vect_length couleurs;;
70
71  (* pointe le point de l'écran correspondant au complexe z avec la couleur n *)
72  let affiche z n =
73         set_color couleurs.(n mod nbcouleurs);
74         plot  (int_of_float(( Re(z) -. !WinXmin) *. !WinXcor))
75                     (int_of_float(( Im(z) -. !WinYmin) *. !WinYcor));;
76
77  (* Déssine l'ensemble de Julia ayant le paramètre complexe c avec k itérations *)
78  let Julia c k =
79       let rec suivant z n =
80             affiche z n;
81             if n < k then
82                 let z1 = racine(difference z c) in
83                      let z2 = minus_compl z1 in
84                            suivant z1 (n+1); suivant z2 (n+1)
85       in
86            suivant zero 0;;
87
88  (* définition de la fenêtre de visualisation *)
89  window (-. 12. /. 7.) (-. 1.) (12. /. 7.) 1.;;
90  Julia (minus_compl un) 10;; (* tracé de l'ensemble de Julia pour c=-1 *)
```

Exercices n°2

Tracé de l'ensemble des complexes de la forme $a_0 + a_1 * c + ... + a_n * c^n$ où $(a_0, a_1, ..., a_n)$ appartiennent à $\{0, 1\}$.

```
#let Ensemble c k =
   let rec suivant z n =
   if n<k then
       let z2 = somme z (puissance c n) in
             affiche z2 n;
             suivant z (n+1); suivant z2 (n+1)
       in
           affiche zero 0;
           suivant zero 0;
           suivant un 0;;

Ensemble : complexe -> int -> unit = <fun>
```