Codes CAML de Von Koch et de Hanoï

1 Commentaires

Ici nous ne mettons que les codes CAML ainsi que leurs typages. Pour les commentaires, le lecteur est invité à se reporter au compte-rendu. Bien entenu, les fonctions non modifiées d'une version à l'autre (pour Von Koch) ne sont pas réécrites.

2 Von Koch

2.1 Première version

```
open Graphics ;;
open_graph "" ;;

let avance distance angle =
  let (x_courant_int, y_courant_int) = current_point () in
  let x_courant_float = float_of_int x_courant_int
  and y_courant_float = float_of_int y_courant_int in
  let x_cible_float = x_courant_float +. distance *. (cos angle)
  and y_cible_float = y_courant_float +. distance *. (sin angle) in
  let x_cible_int = int_of_float x_cible_float
  and y_cible_int = int_of_float y_cible_float in
  lineto x_cible_int y_cible_int ;;

let pi = 3.1415926535 ;;
let pi_sur_trois = pi /. 3. ;;

let rec motif generation distance angle =
  match generation with
    0 -> avance distance angle
  | n ->
      let nouvelle_distance = distance /. 3. in
      motif (n-1) nouvelle_distance angle ;
      motif (n-1) nouvelle_distance (angle +. pi_sur_trois) ;
      motif (n-1) nouvelle_distance (angle -. pi_sur_trois) ;
      motif (n-1) nouvelle_distance angle ;;
```

2.2 Version 2

```
let arrondi x_float =
  let x_int = int_of_float x_float in
  let différence_float = x_float -. float_of_int x_int in
  if différence_float > 0.5 then
    if différence_float < 0. then
      x_int - 1
    else
      x_int + 1
  else
    x_int ;;

let avance distance angle =
  let (x_courant_int,y_courant_int) = current_point() in
  let x_courant_float = float_of_int x_courant_int
  and y_courant_float = float_of_int y_courant_int in
  let x_cible_float = x_courant_float +. distance *. (cos angle)
  and y_cible_float = y_courant_float +. distance *. (sin angle) in
  let x_cible_int = arrondi x_cible_float
  and y_cible_int = arrondi y_cible_float in
  lineto x_cible_int y_cible_int ;;
```

2.3 Troisième version

```
type point_float =
    {mutable x : float ;
     mutable y : float } ;;

let point = {x = 0.0 ; y = 0.0} ;;

let point_courant_float () = (point.x, point.y) ;;

let moveto_float x_float y_float =
  moveto (int_of_float x_float) (int_of_float y_float) ;
  point.x <- x_float ;
  point.y <- y_float;;

let lineto_float x_float y_float =
  lineto (int_of_float x_float) (int_of_float y_float) ;
  point.x <- x_float ;
  point.y <- y_float ;;

let avance distance angle =
  let (x_courant_float, y_courant_float) = point_courant_float () in
  let x_cible_float = x_courant_float +. distance *. (cos angle)
  and y_cible_float = y_courant_float +. distance *. (sin angle) in
  lineto_float x_cible_float y_cible_float ;;

let flocon n distance =
  clear_graph() ;
  moveto_float 10. (distance /. 3.) ;
  motif n distance pi_sur_trois ;
  motif n distance (-. pi_sur_trois) ;
  motif n distance (-. pi) ;;
```

3 Hanoï

3.1 Version élémentaire

```
let mouvement origine destination cpt =
  print_string("mouvement " ^origine^ " ---> " ^destination);
  print_newline();
  cpt := !cpt + 1;;

let hanoi nb_disques=
  let compt=ref 0 in
  let rec hanoi_aux n origine intermediaire destination = match n with
  | 1 ->
      begin
        mouvement origine destination compt;
      end
  | _ ->
      begin
        hanoi_aux (n-1) origine destination intermediaire;
        mouvement origine destination compt;
        hanoi_aux (n-1) intermediaire origine destination;
      end;
  in hanoi_aux nb_disques "A" "B" "C"; !compt;;
```

3.2 Version graphique

3.2.1 Hanoï,version graphique : le code

Les typages sont ici précisés.

```
open Graphics;;
open_graph "";;

let dessine_piquet i nb_disques=
  set_color black;
  moveto (-180+200*i) 100;
  lineto (-20+200*i) 100;
  moveto (200*i-100) 100;
  lineto (200*i-100) (110+30*nb_disques);
  moveto (200*i-100) 85;
  set_text_size 12;
  draw_string ([|"A";"B";"C"|].(i-1));;
(* val dessine_piquet : int -> int -> unit = <fun> *)

let arc_en_ciel h n=
  [|black;red;green;blue;yellow;cyan;magenta|].(h mod 7);;
(*val arc_en_ciel : int -> 'a -> Graphics.color = <fun>*)

let dessine_rondelle tour hauteur largeur n=
  let px=(-100)+200*tour and py=110+30*(hauteur-1) and dx=(largeur*80)/n in
  set_color (arc_en_ciel largeur n);
  fill_rect (px-dx) py (2*dx) 20;;
(*val dessine_rondelle : int -> int -> int -> int -> unit = <fun>*)

let dessine_etat etat_piquets n=
  clear_graph ();
  set_color black;
  moveto 300 30;
  set_text_size 100;
  draw_string ("Hanoï "^(string_of_int n));
  for i=1 to 3 do dessine_piquet i n done;
  for i=1 to 3 do
    for k=1 to etat_piquets.(i-1).(0) do
      dessine_rondelle i k etat_piquets.(i-1).(k) n
    done
  done;;
(*val dessine_etat : int array array -> int -> unit = <fun>*)
```

```
let mouvement origine destination etat_piquets nb_disques=
  let t1=etat_piquets.(origine-1).(0) and t2=etat_piquets.(destination-1).(0) in
  etat_piquets.(destination-1).(t2+1)<-etat_piquets.(origine-1).(t1);
  etat_piquets.(origine-1).(0)<-t1-1;
  etat_piquets.(destination-1).(0)<-t2+1;
  dessine_etat etat_piquets nb_disques;
  for i=1 to 3000000/nb_disques do () done;;
(* val mouvement : int -> int -> int array array -> int -> unit = <fun> *)

let init_etats nb_disques=
  let etats=Array.make_matrix 3 (nb_disques+1) 0 in
  etats.(0).(0)<- nb_disques;
  for i=1 to nb_disques do etats.(0).(i)<-(nb_disques+1-i) done;
  dessine_etat etats nb_disques;
  for i=1 to 900000 do () done;
  etats;;
(*val init_etats : int -> int array array = <fun>*)

let hanoi nb_disques=
  let etat_piquets=init_etats nb_disques in
  let rec hanoi_aux n origine intermediaire destination =
    match n with
    | 1 -> mouvement origine destination etat_piquets nb_disques
    | _ ->
        begin
        hanoi_aux (n-1) origine destination intermediaire; (* dest est vide *)
        mouvement origine destination etat_piquets nb_disques;
        hanoi_aux (n-1) intermediaire origine destination;
        end;
  in hanoi_aux nb_disques 1 2 3;;
(*val hanoi : int -> unit = <fun>*)

close_graph();;
```