宣言型プログラム論 ミニプロジェクト 2

200911434 青木大祐

2012 年 11 月 22 日

1　座標 (x,y) に，指定した色で半径 r の円に内接する正 n 角形を描画する関数

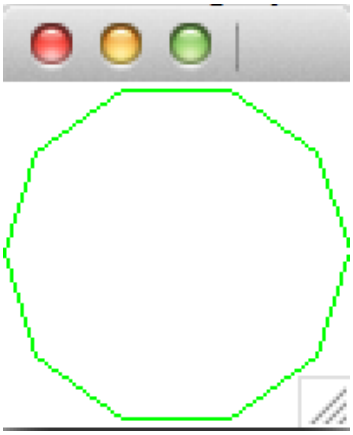
1.1　ソースコード

```
let rec range n m =
  if m > n then (range n (pred m)) @ [m]
  else [n]
;;

let pi = 3.141592;;

let draw_regular_ngon x y r c n =
    List.fold_left (fun acc i ->
        let v_x = r *. (cos (2. *. pi *. (float acc) /. (float n))) in
        let v_y = r *. (sin (2. *. pi *. (float acc) /. (float n))) in
        let start_x = x +. v_x in
        let start_y = y +. v_y in

        let v_x = r *. (cos (2. *. pi *. (float i) /. (float n))) in
        let v_y = r *. (sin (2. *. pi *. (float i) /. (float n))) in
        let dest_x = x +. v_x in
        let dest_y = y +. v_y in
        Graphics.set_color c;
        Graphics.moveto (int_of_float start_x) (int_of_float start_y);
        Graphics.lineto (int_of_float dest_x) (int_of_float dest_y);
        i
    ) 0 (range 1 n)
;;

let main x y r c n =
  let rec loop () =
    let status = Graphics.wait_next_event
      [Graphics.Key_pressed] in
    if status.Graphics.keypressed then ()
    else loop () in
  Graphics.open_graph " 100x100";
  draw_regular_ngon x y r c n;
  loop ();
  Graphics.close_graph ()
;;
```

range 関数は n から m まで 1 つずつ増加する数のリストを生成する関数である。これを用いてそれぞれの頂点を *fold_left* で処理している。

1.2　実行例

```
# #use "1.ml";;
val range : int -> int -> int list = <fun>
val pi : float = 3.141592
val draw_regular_ngon :
  float -> float -> float -> Graphics.color -> int -> int = <fun>
File "1.ml", line 33, characters 2-29:
Warning 10: this expression should have type unit.
val main : float -> float -> float -> Graphics.color -> int -> unit = <fun>

# main 50.0 50.0 50.0 Graphics.green 10;;
```



2　座標 (x,y) に，指定された色で，与えられた図形を描画する

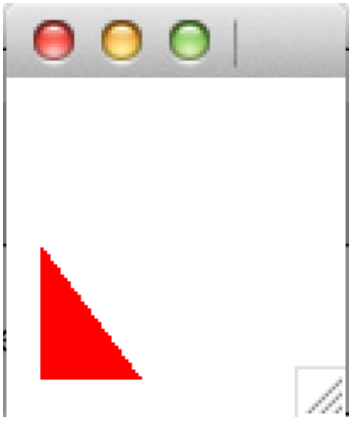
2.1　ソースコード

```
open Graphics;;

type figure =
Rectangle of float * float (* 幅 * 高さ *)
  | Circle of float (* 半径 *)
  | Triangle of float * float (* 幅 * 高さ *)

let draw_figure x y fig =
  match fig with
    | Rectangle(n1, n2) -> fill_rect (int_of_float x) (int_of_float y) (int_of_float n1) (int_of_float n2)
    | Circle r -> fill_circle (int_of_float x) (int_of_float y) (int_of_float r)
    | Triangle(width, height) -> fill_poly [| (int_of_float x, int_of_float y); (int_of_float (x +. width),int_of_float y); (
        int_of_float x, int_of_float (y +. height) )|]
;;



let main x y c figure =
  let rec loop () =
    let status = Graphics.wait_next_event
      [Graphics.Key_pressed] in
    if status.Graphics.keypressed then ()
    else loop () in
  Graphics.open_graph " 100x100";
  set_color c;
  draw_figure x y figure;
  loop ();
  Graphics.close_graph ();;
```

2.2　実行例

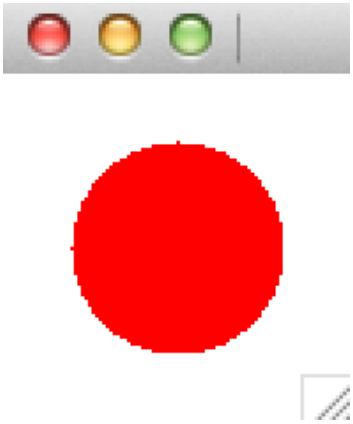
2.2.1　三角形

```
# #use "2.ml";;
type figure =
    Rectangle of float * float
  | Circle of float
  | Triangle of float * float
val draw_figure : float -> float -> figure -> unit = <fun>
val main : float -> float -> Graphics.color -> figure -> unit = <fun>

# main 10.0 10.0 red (Triangle(30.0,40.0));;
```



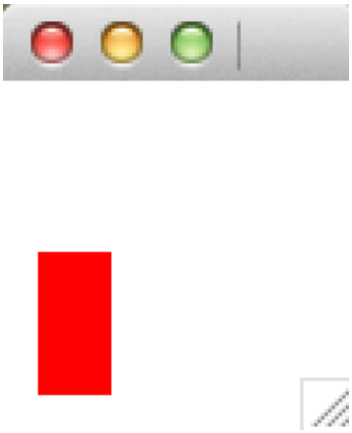
2.2.2 円

```
# main 50.0 50.0 red (Circle(30.0));;
```



2.2.3 四角形

```
# main 10.0 10.0 red (Rectangle(20.0, 40.0));;
```



3 図形を組み合わせた図形

3.1 ソースコード

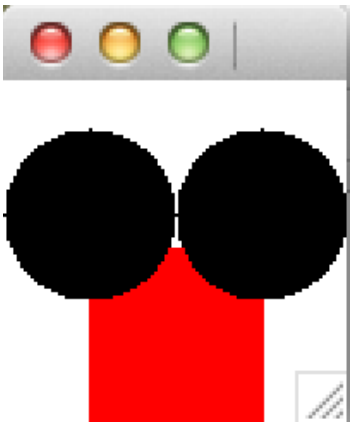
```
open Graphics;;

type figure =
Rectangle of float * float (* 幅 * 高さ *)
  | Circle of float (* 半径 *)
  | Triangle of float * float (* 幅 * 高さ *)
;;

let draw_figure x y scale fig =
  match fig with
    | Rectangle(n1, n2) -> fill_rect (int_of_float x) (int_of_float y) (int_of_float (scale *. n1)) (int_of_float (scale *. n2))
    | Circle r -> fill_circle (int_of_float x) (int_of_float y) (int_of_float (scale *. r))
    | Triangle(width, height) -> fill_poly [| (int_of_float x, int_of_float y); (int_of_float (x +. (scale *. width)),
        int_of_float y); (int_of_float x, int_of_float (y +. (scale *. height)))|]
;;

type region =
Figure of figure
  | Translate of float * float * region (* 図形を x 座標, y 座標で指定される位置に移動 *)
  | Scale of float * region (* float で指定される倍率で, 図形を縮小拡大 *)
  | Union of region * region (* 二つの図形が合わさった図形 *)
;;

type picture = (Graphics.color * region) list
;;

let rec draw_region x y scale region =
  match region with
    | Figure(fig) -> draw_figure x y scale fig
    | Translate(x, y, reg) -> draw_region x y scale reg
    | Scale(scale, reg) -> draw_region x y scale reg
    | Union(reg1, reg2) -> draw_region x y scale reg1; draw_region x y scale reg2
;;

let draw_picture pic =
  List.fold_right (
    fun next prev ->
      match next with
        | (c, reg) -> set_color c; draw_region 0.0 0.0 1.0 reg
  ) pic ();;

let ex =
  let circle = Figure (Circle 25.) in
  let r1 = Union (Translate (25.,60., circle),
                  Translate (75.,60., circle)) in
  let rectangle = Figure (Rectangle (50., 50.)) in
  let r2 = Translate (25., 0., rectangle) in
  [(Graphics.black, r1); (Graphics.red, r2)]
;;



let main p =
  let rec loop () =
    let status = Graphics.wait_next_event
      [Graphics.Key_pressed] in
    if status.Graphics.keypressed then ()
    else loop () in
  Graphics.open_graph " 100x100";
  draw_picture p;
  loop ();
  Graphics.close_graph ()
;;


let mypict =
  let circle = Figure (Circle 25.) in
  let r1 = Union (Translate (25.,60. ,Scale(1.25, circle)),
            Translate (75.,60. ,Scale(0.75, circle))) in
  let rectangle = Figure (Rectangle (50., 50.)) in
  let triangle = Figure (Triangle(35., 35.)) in
  let r2 = Union(Translate( 25., 0., rectangle),
                 Translate( 75., 75., Scale(0.25, triangle))) in
  [(Graphics.green, r1); (Graphics.blue, r2)]
;;
```

3.2 実行例

3.2.1 例 1

```
 # #use "3.ml";;
type figure =
```

```
    Rectangle of float * float
  | Circle of float
  | Triangle of float * float
val draw_figure : float -> float -> float -> figure -> unit = <fun>
type region =
    Figure of figure
  | Translate of float * float * region
  | Scale of float * region
  | Union of region * region
type picture = (Graphics.color * region) list
val draw_region : float -> float -> float -> region -> unit = <fun>
val draw_picture : (Graphics.color * region) list -> unit = <fun>
val ex : (Graphics.color * region) list =
  [(0,
    Union (Translate (25., 60., Figure (Circle 25.)),
     Translate (75., 60., Figure (Circle 25.))));
   (16711680, Translate (25., 0., Figure (Rectangle (50., 50.))))]
val main : (Graphics.color * region) list -> unit = <fun>
# main ex;;
```



3.2.2 例 2

```
let mypict =
  let circle = Figure (Circle 25.) in
  let r1 = Union (Translate (25.,60. ,Scale(1.25, circle)),
            Translate (75.,60. ,Scale(0.75, circle))) in
  let rectangle = Figure (Rectangle (50., 50.)) in
  let triangle = Figure (Triangle(35., 35.)) in
  let r2 = Union(Translate( 25., 0., rectangle),
                 Translate( 75., 75., Scale(0.25, triangle))) in
  [(Graphics.green, r1); (Graphics.blue, r2)]
;;
```