```
(*************************************************************************)
(*       Rectangle d'aire maximale dans une grille de mots croises       *)
(*************************************************************************)

let N = 20;;
let CN = N;;

let grille = make_vect N [||];;
for i = 0 to N-1 do
  grille.(i) <- make_vect N false
done;;

let raz () =
  for i = 0 to N-1 do
    for j = 0 to N-1 do
      grille.(i).(j) <- false
    done
  done
;;

let init () =
  let rec noircit = function
        0 -> ()
      | n -> let i = random__int N in
             let j = random__int N in
             if grille.(i).(j) = false then begin
                grille.(i).(j) <- true ; noircit (pred n)
             end else
                noircit n
  in noircit CN
;;

#open "graphics";;
open_graph "";;

let w = (size_x () - 60)/N;;
let h = (size_y () - 40)/N;;
let off_x = ((size_x()) - N*w) / 2;;
let off_y = ((size_y()) - N*h) / 2;;

let x_of_i i = off_x + w*i;;
let y_of_j j = off_y + h*j;;

let affiche_case i j b =
  let x = x_of_i i in
  let y = y_of_j j in
  set_color blue ;
  if not b then begin
    moveto (x+w) y ;
    lineto x y ;
    lineto x (y+h)
  end else begin
    fill_rect x y w h
  end
;;

let affiche_grille () =
  for i = 0 to N-1 do
    for j = 0 to N-1 do
       affiche_case i j grille.(i).(j)
    done
  done ;
  moveto (off_x + N*w) off_y ;
  lineto (off_x + N*w) (off_y + N*h) ;
  lineto off_x (off_y + N*h)
;;

type intervale = Interv of int * int;;
type rectangle = Rect   of intervale * intervale;;

let affiche_rectangle (Rect (Interv (i,di), Interv (j,dj) )) =
  set_color red ;
  fill_rect (x_of_i i) (y_of_j j) (w*di) (h*dj)
;;

let aire (Rect (Interv (_,di), Interv (_,dj))) =
  di * dj
;;

let max_rect r1 r2 =
  if (aire r1) > (aire r2) then r1 else r2
;;

let bidon = Rect (Interv (0,0), Interv (0,0));;

let case_noire (Interv (i,di)) (Interv (j,dj)) =
  let rec cherche ci cj =
    if ci = di then
      raise Not_found
    else if cj = dj then
      cherche (succ ci) 0
    else if grille.(i+ci).(j+cj) then
      (ci,cj)
    else
      cherche ci (succ cj)
  in
  cherche 0 0
;;

let trouve () =
  let rec trouve_rec solm ((Interv (i,di)) as ii) ((Interv (j,dj)) as ij) =
    try  let (ci,cj) = case_noire ii ij in
         let sol1 = if cj>0 then
                      trouve_rec solm ii (Interv (j,cj))
                    else solm in
         let sol2 = if cj<dj-1 then
                      trouve_rec sol1 ii (Interv (j+cj+1,dj-cj-1))
                    else sol1 in
         let sol3 = if ci<di-1 then
                      trouve_rec sol2 (Interv (i+ci+1,di-ci-1)) ij
                    else sol2 in
         max_rect sol3 (Rect (Interv (i,ci), ij))
    with Not_found -> max_rect solm (Rect (ii,ij))
  in
  trouve_rec bidon (Interv (0,N)) (Interv (0,N))
;;

let main () =
  raz () ; init () ; clear_graph () ; affiche_grille () ;
  let sol = trouve () in affiche_rectangle sol ;;
```