```
open Types
let foi = float_of_int
let iof = int_of_float

let fst (a,_,_) = a
let snd (_,b,_) = b
let trd (_,_,c) = c

let r3_to_vec3 (x,y,z) = Raylib.Vector3.create x z y
let (+$) (a,b,c) (x,y,z) = (a+.x, b+.y, c+.z)
let (-$) (a,b,c) (x,y,z) = (a-.x, b-.y, c-.z)
let ( *$) (a,b,c) q = (a*.q, b*.q, c*.q)
let (/$) (a,b,c) q = (a/.q, b/.q, c/.q)

let zero = 0.0,0.0,0.0
let zero_r = Raylib.Vector3.create 0.0 0.0 0.0

(*dot product, fin le produit scalaire quoi*)
let ps (a,b,c) (x,y,z) = a*.x +. b*.y +. c*.z

(*cross product, produit vectoriel*)
let pv (a,b,c) (x,y,z) = (b*.z -. c*.y, c*.x -. a*.z, a*.y -. b*.x)

let abs_f x = if x < 0.0 then -.x else x
let sign_f x = if x < 0.0 then -1.0 else 1.0

let dist_square (xa,ya,za) (xb,yb,zb) = (xa-.xb)**2.0 +. (ya-.yb)**2.0 +. (za-.zb)**
2.0
let dist a b = sqrt (dist_square a b)
let norme = dist zero

let vect_elem a b = (b -$ a) /$ dist a b
let shmidtz = vect_elem zero

let det3 (a,b,c) (d,e,f) (g,h,i) =
  a *. (e*.i -. h*.f) -. d*. (b*.i -. h*.c) +. g*. (b*.f -. e*.c)

let order a b c center =
   (*on veut que a b c soit dans le sens horraire, vus depuis le centre center*)
   let det = det3 (a-$center) (b-$center) (c-$center) in
   if det > 0.0 then a,b,c
   else a,c,b

(* vecteur normal à la surface a b c vu depuis le centre center*)
let normal a b c center =
   let a,b,c = order a b c center in
   pv (b-$a) (c-$a)

(* fonctions modifiables pour plus de détail*)
let collision (_,_,z) =
   z < 0.01

let fix_collision (x,y,_) =
   (x,y,0.01)

let somme_forces l p =
   let s = List.fold_left (fun x (f,_) -> x +$ f ) zero l in
   if collision p.pos then zero  else s

(*l'aire d'un triangle situé sur les sommets a b c*)
let area (a,b,c) = norme (pv (b-$a) (c-$a)) /. 2.0

(*On approxime le volume de l par le volume d'une ellipsoide*)
let volume l =
   let l = Array.map (fun x-> x.pos) l in
   let minx,miny,minz = Array.fold_left
      (fun (a,b,c) (x,y,z) -> (min a x, min b y, min c z)) (max_float,max_float,max_fl
oat) l in
   let maxx,maxy,maxz = Array.fold_left
      (fun (a,b,c) (x,y,z) -> (max a x, max b y, max c z)) (min_float,min_float,min_fl
oat) l in
   4.0/.3.0 *. 3.14 *. (maxx-.minx)*.(maxy-.miny)*.(maxz-.minz)
```