```
open Maths
open Raylib.Color
open Constantes
open Types

(*force elastique avec les voisins *)
let ressort src dst l0 k_ressort = vect_elem dst.pos src.pos *$
  (-.k_ressort*.(dist src.pos dst.pos -.l0))

(*force de repultion avec les voisins*)
let repultion src dst = vect_elem dst.pos src.pos *$
  (k_repultion/.(dist src.pos dst.pos)**4.0)

(*force d'amortissment avec les voisins *)
let amortisseur src dst =
  let er = vect_elem src.pos dst.pos in
  er *$ (k_damping *. ps (src.vit -$ dst.vit) er)

(*force de gonflement du gaz*)
let gaz id vol blob center=
  List.fold_left (+$) zero
    (List.map (fun (surface,norm) -> shmidtz norm *$ (nRT *. surface /. vol))
    (Graph.triangles_with id blob center))

(*force induite par le champs gravitationnel*)
let poids penche src =
  (gravity +$ if penche then (5.0,0.0,0.0) else zero) *$ (1.0*.src.mass)

(*renvoie une liste de couples (force, couleur)*)
let bilan_des_forces src id volume blob {penche;k_ressort;center;_} =
   [
    poids penche src, yellow;
    gaz id volume blob center, green
   ]
   @ List.concat
    (
      List.map
      (fun (dst,l0) ->
        [
          ressort src dst l0 k_ressort, Raylib.fade blue 0.4;
          amortisseur src dst, raywhite;
          repultion src dst, red;
        ]
      )
      (Graph.linked_to blob id)
    )
```