```
open Raylib
open Maths
open Graph
open Force
open Types
open Constantes
open Icosphere

(*affiche les points ou les forces*)
let draw st center =
  let draw_tri a b c =
    let a,b,c = order a b c center in
    draw_triangle_3d (r3_to_vec3 a) (r3_to_vec3 b) (r3_to_vec3 c)
  in
  let draw_vec src f col =
    let f = f *$ fac_newt in
    let end_force = src +$ f in
    let mid_force = src +$ (f *$ 0.75) in
    begin
      draw_line_3d (r3_to_vec3 src) (r3_to_vec3 end_force) col;
      draw_cylinder_ex (r3_to_vec3 mid_force) (r3_to_vec3 end_force) (norme f/.50.0)
 0.0 10 col;
    end
  in
  begin_drawing ();
  clear_background Color.black;
  begin_mode_3d st.cam;
  (* repère du 0*)
  draw_cube zero_r 1.0 1.0 1.0 Color.raywhite;
  draw_grid 100 10.0;
  if not (is_key_down Key.Z) then
  (*dessin des triangles*)
  List.iter (fun (a,b,c) -> draw_tri a.pos b.pos c.pos (fade Color.green 0.5)) (surf
aces st.blob);
  (*dessin des arrêtes*)
  Graph.iteri
    (fun i s -> let f = bilan_des_forces s i (volume st.blob) st.blob
      {penche = st.penche;points = st.blob; center = center;k_ressort = st.k_ressort
}
      in
      if is_key_down Key.F then (*juste les forces*)
        List.iter (fun (f,col) -> draw_vec s.pos f col) f
      else if is_key_down Key.G then (*l'accélération*)
        draw_vec s.pos (somme_forces f s) Color.orange;
        begin
          List.iter
            (fun (posb,_) ->
              let a = r3_to_vec3 s.pos in
              let b = r3_to_vec3 posb.pos in
              draw_line_3d a b Color.orange)
           (Graph.linked_to st.blob i);
        end
    )
    st.blob;
  end_mode_3d ();
  draw_text (
"F pour le mode forces
Space pour le \"saut temporel\"
W pour l'affichage de la surface
R pour pencher la surface (" ^ string_of_bool st.penche ^ ")
" ^ string_of_float st.k_ressort ^ "N/m force de ressort elastique, modifiable avec
h/y" )
    10 20 20 Color.raywhite;
  end_drawing ()

(*boucle principale*)
let rec loop st =
  let center = Graph.fold_left (+$) zero (Graph.map (fun x-> x.pos) st.blob) *$ (1.0
/.(foi n)) in
  if Raylib.window_should_close () then Raylib.close_window () else
  (*calcul du prochain état en parallèle du dessin*)
  let integrationDomain = Domain.spawn
    (fun _ -> Integration.integrate
```

```
      {points=st.blob;k_ressort = st.k_ressort;penche = st.penche;center = center})
in
  let time_jumping = is_key_down Key.Space in
  if not (time_jumping && st.t mod 10 <> 0) then draw st center;
  let rmed = r3_to_vec3 center in
  Vector3.set_x (Camera3D.target st.cam) (Vector3.x rmed);
  Vector3.set_y (Camera3D.target st.cam) (Vector3.y rmed);
  Vector3.set_z (Camera3D.target st.cam) (Vector3.z rmed);
  Camera3D.set_projection st.cam CameraProjection.Perspective;
  update_camera (addr st.cam) CameraMode.Third_person;
  let blob' = Domain.join integrationDomain in
  loop {
    t = st.t + 1;
    blob = blob';
    k_ressort = max 0.0 (st.k_ressort +.
      1.0 *.
      if is_key_down Key.H then 1.0
      else if is_key_down Key.Y then -.1.0
      else 0.0);
    penche = if is_key_pressed Key.R then not st.penche else st.penche;
    cam = st.cam
  }

let setup () =
  Raylib.init_window w h "Blob";
  (*Raylib.set_target_fps 60;*)
  if is_window_ready () then
    let camera = Camera3D.create zero_r zero_r (Vector3.create 0. 1. 0.) 45.
      CameraProjection.Perspective
    in let open Camera3D in
    set_position camera (Vector3.create 0. 20. 40.);
    set_target camera (Vector3.create 0. 1. 0.);
    set_up camera (Vector3.create 0. 1. 0.);
    set_fovy camera 120.;
    set_projection camera CameraProjection.Perspective;
    disable_cursor ();
    {
      t = 0;
      blob = Graph.initial ();
      penche = false;
      k_ressort = k_ressort;
      cam = camera;
    }
  else failwith "Problème de fenêtre"

let () =
  setup () |> loop
```