```
open Maths
open Constantes

(*renvoie la liste contenant les mêmes élèments mais de façon unique*)
let unique l =
  let h = Hashtbl.create 10 in
  List.iter (fun x -> Hashtbl.replace h x ()) l;
  Hashtbl.to_seq_keys h |> List.of_seq

let subdivide positions triangles =
  (* rajoute du détail à une sphère de rayon 1 *)
  let q = Array.length positions in (* nombre de points actuels *)
  let edges = (* propriété 1 : les arrêtes a --- b sont telles que a < b *)
    List.map (fun (i,j,k) -> [(i,j);(i,k);(j,k)]) triangles |> List.concat |> unique
  in
  (* tous les points entre les sommets i et j*)
  let middle = Hashtbl.create 100 in
  let newverticieslist = List.mapi
    (fun i (a,b) ->
      Hashtbl.replace middle (a,b) (q+i);
      shmidtz (positions.(a) +$ positions.(b))
    ) edges in
  let positions' = Array.concat [positions;Array.of_list newverticieslist] in
  let triangles' =
    List.map
      (fun (a,b,c) ->
        let d = Hashtbl.find middle (a,b) in
        let e = Hashtbl.find middle (a,c) in
        let f = Hashtbl.find middle (b,c) in
        (* on renvoie les 4 triangles formés par le triangle abc,
         tout en conservant la propriété 2 grâce à la propriété 1*)
        [(b,d,f);(a,d,e);(c,e,f);(d,e,f)]
      )
    triangles |> List.concat
  in positions', triangles'

(*icosahèdre brut*)
let icosahedron =
  let x = 0.525731112119133606 in
  let z = 0.850650808352039932 in
  [|
    (-.x, 0.0, z);(x, 0.0, z);(-.x, 0.0, -.z);
    (x, 0.0, -.z);(0.0, z, x);(0.0, z, -.x);
    (0.0, -.z, x);(0.0, -.z, -.x);(z, x, 0.0);
    (-.z, x, 0.0);(z, -.x, 0.0);(-.z, -.x, 0.0)
  |]

(*Propriété 2 : un triangle est une liste de 3 indices de sommets, dans l'ordre croi
ssant*)
let indices_triangles = [
  (0, 1, 4); (0, 4, 9); (4, 5, 9);
  (4, 5, 8); (1, 4, 8); (1, 8, 10);
  (3, 8, 10); (3, 5, 8); (2, 3, 5);
  (2, 3, 7); (3, 7, 10); (6, 7, 10);
  (6, 7, 11); (0, 6, 11); (0, 1, 6);
  (1, 6, 10); (0, 9, 11); (2, 9, 11);
  (2, 5, 9); (2, 7, 11)
]

let icosphere, indices_triangles =
  let rec aux n (positions,triangles) =
    if n = 0 then positions,triangles
    else aux (n-1) (subdivide positions triangles)
  in aux deep (icosahedron,indices_triangles)

let n = Array.length icosphere
```