LOLscad.scad

```
/*═══════════════════════════════════════════════════════════════════════
═══╕
│                          Liège Openscad Library
│
│            Module de déformation et modification pour OpenSCAD
│
├═══════════════════╤═══════════════════════════════════════════╤═══════
═══┤
│ marc@vanlindt.be │   LGPL 2.1 marc@vanlindt.be 2022   │ v0.98 ----
wip |
├═══════════════════╧═══════════════════════════════════════════╧═══════
═══┘
│           Variables générales             /
└═══════════════════════════════════════════*/

LogoFB= [[4.46567, 4.99666], [3.06433, 4.99666], [3.06433, 0],
[0.987666, 0], [0.987666, 4.99666], [0, 4.99666], [0, 6.76134],
[0.987666, 6.76134], [0.987666, 7.90467], [1.00769, 8.22956], [1.07504,
8.57716], [1.20063, 8.92712], [1.39537, 9.25909], [1.6702, 9.55271],
[2.03602, 9.78764], [2.50376, 9.94352], [3.08433, 10], [4.62167,
9.99402], [4.62167, 8.28068], [3.505, 8.28068], [3.35933, 8.26038],
[3.21662, 8.18648], [3.10811, 8.0397], [3.065, 7.80073], [3.065,
6.76073], [4.64833, 6.76073]];

LetterL=[[0,0],[0,70],[22,73],[19,21],[50,25],[48,-1],[0,0]];
//LetterO=[[0,35],[6.25,61.25],[25,70],[25,70],[43.75,61.25],[50,35],[4
3.75,8.75],[25,0],[6.25,8.75],[0,35]];
LetterO=[[0,0],[0,70],[50,70],[50,0],[0,0]];
blue            = [0,0,1,1];
red             = [1,0,0,1];
green           = [0,1,0,1];
violet      = [0.5,0,0.5,1];
yellow      = [1,1,0,1];
cyan            = [0,1,1,1];
black           = [0,0,0,1];
white           = [1,1,1,1];
oak             = RVB(200,50,90,255);
orange      = [1,0.5,0,1];
olive           = [0.5,0.5,0,1];
sarcelle    = [0,0.5,0.5,1];
marine      = [0,0,0.5,1];
fuschia     = [1,0,1,1];
glass         = [1,0,1,0.2];


bleu            = [0,0,1,1];
rouge           = [1,0,0,1];
vert            = [0,1,0,1];
jaune           = [1,1,0,1];
```

```
noir            = [0,0,0,1];
blanc           = [1,1,1,1];
gris            = [0.5,0.5,0.5,1];
gray            = [0.5,0.5,0.5,1];
pink            = RVB(255,107,219,255);

phi       = 1.61803399;
aphi      = phi-1;
biphi     = phi+1;
angledor  = 360/biphi;
py        = sqrt(0.5);
bipy      = sqrt(2);
pi        = 3.141592654;
tau       = pi*2;

/*═══════════╕
│  Exemples   \
╘═══════════*/

/* MOEBIUS / ELLIPSE
  moebius(n=32,d=40,t=0.5)
  ellipse([5,20],$fn=128);
  moebius(n=32,d=40,t=0.5)
  ellipse([20,5],$fn=128);
*/
/* CHULL

chull(m=true){
  sphere(d=1,$fn=64);
  translate([10,10,00]) sphere(d=1,$fn=16);
  translate([20,0,00]) sphere(d=1,$fn=16);
  translate([30,0,00]) sphere(d=1,$fn=16);
  translate([30,-10,00]) sphere(d=1,$fn=16);
  translate([0,-10,00]) sphere(d=1,$fn=16);
}

*/



 /* PYTHATREE / BONE*/
//pythatree(d="z",h=50,maxit=6,r1=30,r2=30)
//bone(h=50,d1=20,d2=14.14214,c=40,$fn=64);

/*pythatree(d="z",h=10*biphi,maxit=8,a=90,r1=90,r2=90)
hull(){sphere(d=10,$fn=16);
translate([0,0,10*biphi])sphere(d=10*aphi,$fn=16);}
*/



//pythatree(d="y",h=10,sp=5,maxit=8)
//polygon(square([10,10],center=true));
```

```
/*pythatree(d="y",h=50,maxit=9,a=90,s=sqrt(0.5))
hull(){
  circle(d=10,$fn=64);
  translate([0,50])
  circle(d=sqrt(0.5)*10,$fn=64);}
*/


/**/

/* ROTATE2
  rotate2()
  cube(center=true);
  translate([0,2,0])
  cube(center=true);
*/

/* RING
  ring(d=10,n=11){
  cylinder(d=1,h=5);
  translate([0,0,5.6]) scale([1,1,2]) sphere(d=0.6,$fn=64);
  }
*/

/* SKEW
  skew(YX=1)
  cube([2,2,2]);
*/

/* ROUNDSQUARE
  roundsquare(s=[40,20],d=[5,10,5,15],$fn=64,center=true);
*/

/* NGON
  ngon(d=20,fn=3,inside=true);
  translate([0,0,-1])   #cylinder(d=20,$fn=64);

  translate([25,0,0])   ngon(d=20,fn=3,inside=false);
  translate([25,0,-1])  #cylinder(d=20,$fn=64);

  translate([50,0,0])ngon(d=20,f=4,inside=true);
  translate([50,0,-1])#cylinder(d=20,$fn=64);

  translate([75,0,0])ngon(d=20,fn=4,inside=false);
  translate([75,0,-1])#cylinder(d=20,$fn=64);

  translate([100,0,0])ngon(d=20,fn=5,inside=true);
  translate([100,0,-1])#cylinder(d=20,$fn=64);

  translate([125,0,0])ngon(d=20,fn=5,inside=false);
```

```
  translate([125,0,-1])#cylinder(d=20,$fn=64);
*/
/* OUTLINE*/
/*outline(w=1,t="in"){
  ellipse([10,20],fn=64);ellipse([20,10],fn=64);}
translate([0,0,1]){#ellipse([10,20],fn=64);#ellipse([20,10],fn=64);}
*/
/*
outline(w=1,t="out"){
  ellipse([10,20],fn=64);ellipse([20,10],fn=64);}
translate([0,0,1]){#ellipse([10,20],fn=64);#ellipse([20,10],fn=64);}
*/
/*outline(w=1,t="on"){
  ellipse([10,20],fn=64);ellipse([20,10],fn=64);}
translate([0,0,1]){#ellipse([10,20],fn=64);#ellipse([20,10],fn=64);}
  */
/**/
/* RANDOM
  for(i=[1:10]){
    echo(random(10,s=i));
  }
*/
/* FIBONACCI
for(i=[1:15]){
  echo(fibonacci(i));
}
*/

/* TEARDROP / RANDOM
 for(i=[1:500]){
   translate([random(500,i*10),random(500,i*50),random(500,i*60)])
   color([0.6,0.6,0.9,0.5])teardrop(a=30+random(30,i));
 }
*/

/* STAR
  star(d1=10,d2=20,fn=9);
*/

/* TUBE - COUDE
   tube(d1=10,d2=8,h=15,$fn=64);

   translate([0,0,15])
   coude(d1=10,d2=8,a=45,$fn=64);

   translate([0,0,20])
   rotate([0,045,0])
   translate([0,0,5])
   tube(d1=10,d2=8,h=5,$fn=64);
*/
```

```
/* ROUNDCUBE
roundcube(s=[50,100,150],b=[5,15,15,20],t=[25,35,40,5],$fn=64,center=tr
ue);
*/
/* PAIR
for(i=[0:20]){
  echo(str(i,pair(i)==true?" est pair!":" est impair!"));
}
*/
/* GRID - CNC
  grid(s=[100,100],x=5,y=10,w=2)
  {
    cnc(0.5,$fn=32){
      ellipse([2,1],$fn=32);
      ellipse([1,2],$fn=32);
  }}
*/

/* PIEPART
    piepart(d=10,p=20/100);
    rotate([0,0,360*21/100])
    piepart(d=10,p=78/100);

*/

/* PIE
pie(d=10,p=[1,2,1,2,1,2,3,2,1,3,2,1]);
*/
/* SUM
echo(sum([5,10,15,20]));
*/
/* MYANGLE/length*/
/* CORRECT / CHULL
rotate_extrude()
rotate_extrude_correct()
chull(){
  circle(d=3);
  translate([60,0])
  circle(d=3);
  translate([70,120])
  circle(d=3);
}
*/

/* FRACTSHAPE
    fractshape(d=40,fn=4,maxit=3);
    translate([40,0,0]) fractshape(d=40,fn=5,it=3);
    translate([80,0,0]) fractshape(d=40,fn=6,it=3);
```

```
*/
/* clean
test=[[0,0],[0,0],[10,10],[10,10],[20,20],[20,20],[30,30],[30,30],[40,4
0],[40,40],[50,50],[50,50]];
echo(clean(test));
*/

/* Kochflake
    for(i=[0:3]){
    translate([i*10,0,0])kochflake(d=10,maxit=i);}
    */

/* Chaincurve / TRACE

//points=[[0,0],[sin(30)*10,cos(30)*10],[10,0],[0,0]];

points=[[0,0],[0,10],[10,10],[20,0],[30,10],[40,20],[60,-20],[10,-10],[
10,-5],[15,-5],[15,0],[0,0]];

color([0.4,1.0,0.4,1])
linear_extrude(1)
polygon(chaincurve(points,8));

color("red") linear_extrude(3)
trace(points,0.2);

color([0.5,0.5,1,1]) linear_extrude(3)
trace(chaincurve(points,8),0.1);


*/







/*
difference(){
rotate2()             cube([10,10,10],center=true);
translate([0,0,10])  cube([20,20,20],center=true);}
/*
rotate2()             cube([10,10,10],center=true);
*/




/*═══════════════════════════════════════╗
│  Modificateurs / déformations   \
╚═══════════════════════════════════════*/
```

```
module outline                (w,t){
  w=w==undef?1:w;
  t=t==undef?"on":t;
  difference()
  {
    offset(t=="out"?w:t=="in"?0:w/2)
    children();
    offset(t=="out"?0:t=="in"?-w:-w/2)
    children();
  }
}

module pythatree              (a,h,sp,maxit,b,r1,r2,s,d){
  a  = a  == undef ? 45  : a;
  h  = h  == undef ? 1   : h;
  sp = sp == undef ? 0   : sp;
  maxit  = maxit  == undef ? 3   : maxit;
  b  = b  == undef ? 1   : b;
  r1 = r1 == undef ? 0   : r1;
  r2 = r2 == undef ? 0   : r2;
  s  = s  == undef ? py  : s;
  d  = d  == undef ? "y" : d;
  children();
  if(b<=maxit)
  {
    translate([d=="x"?h:d=="y"?sp:-sp, d=="x"?-sp:d=="y"?h:0,
d=="x"?0:d=="y"?0:h])
    rotate([d=="x"?0:d=="y"?r2:0, d=="x"?r2:d=="y"?0:-a, d=="x"?-
a:d=="y"?-a:r2])
    scale([s,s,s])
    pythatree(a=a,h=h,sp=sp,maxit=maxit,b=b+1,r1=r1,r2=r2,s=s,d=d)
    {
      children();
    };
    translate([d=="x"?h:d=="y"?-sp:sp, d=="x"?sp:d=="y"?h:0,
d=="x"?0:d=="y"?0:h])
    rotate([d=="x"?0:d=="y"?r1:0, d=="x"?r1:d=="y"?0:a,
d=="x"?a:d=="y"?a:r1])
    scale([s,s,s])
    pythatree(a=a,h=h,sp=sp,maxit=maxit,b=b+1,r1=r1,r2=r2,s=s,d=d)
    {
      children();
    };
  }
}
module chull                  (m){
  union()
  for(i=[0:$children-2]){
    hull(){
      children(m==true?0:i);
      children(i+1);
```

```
    }
  }
}

module rotate2                (){
  rotate([45,90-atan(sqrt(2)),0])
  children();
}
module ring                   (d,n){
  d=d==undef?10:d;
  n=n==undef?5:n;
  for(i=[0:n-1]){
    rotate([0,0,360/n*i]){
      translate([d/2,0,0])
      children();
    }
  }
}

module fibo                   (s,n,r){
  r=r==undef?true:r;
  s=s==undef?1:s;
  n=n==undef?128:n;


  for(i=[1:n]){
    rotate([0,0,angledor*i])
    translate([s*i,0,0])
    scale(r==true?s+pow(1.003,i):1)
    children();
  }
}

module skew                   (XY,XZ,YX,YZ,ZX,ZY){
  matrice=[
    [1,XY,XZ,0], //[redimX, skewXY, skewXZ,translateX]
    [YX,1,YZ,0], //[SkewYX,RedimY,SkewYZ,translateY]
    [ZX,ZY,1,0] //[SkewZX, SkewZY,redimZ,TranslateZ]
  ];
  multmatrix(matrice){
    children();
  }
}

module cnc(d,show,fn){
  show=show==undef?false:show;
  d = d == undef ? 3:d;
  fn = fn == undef ? 32:fn;
  if(show==false){
    offset(-d/2,$fn=fn)
    offset(d/2,$fn=fn)
```

```
    children();
  }
  else
  {
    color("green")
    linear_extrude(1)
    children();
    color("red")
    linear_extrude(0.5)
    difference()
    {
      offset(-d/2,$fn=fn)
      offset(d/2,$fn=fn)
      children();
      children();
    }
  }
}




module moebius(d,t,fn){ // version 2.0
  fn = fn == undef ? 128  :fn;
  d = d == undef ? 30   :d;
  t = t == undef ? 0.5 :t;
  union(){
   for(j=[0:$children-1])
   {
    for(i=[1:fn]){
      hull(){
        rotate([0,360/fn*i,0])
        translate([d/2,0,0])
        rotate([0,0,i*(360*t)/fn])
        linear_extrude(0.1)
        children(j);

        rotate([0,360/fn*(i+1),0])
        translate([d/2,0,0])
        rotate([0,0,(i+1)*(360*t)/fn])
        linear_extrude(0.1)
        children(j);
      }
    }
  }
 }
}



module grid                    (dim,x,y){
```

```
  dim=dim==undef?[100,100]:dim;
  x=x==undef?10:x;
  y=y==undef?10:y;
  for(i=[1:x-1])
    translate([i*dim[0]/x,0,0])
    rotate([-90,0,0])
    linear_extrude(dim[1])
    children();

  for(i=[1:y-1])
    translate([0,i*dim[1]/y,0])
    rotate([-90,0,-90])
    linear_extrude(dim[0])
    children();
}




module rotate_extrude_correct (){
  difference(){
    children();
    translate([-100000,-50000])
    square([100000,100000]);
  }
}


/*═══════════════════════╗
│  Nouvelles formes   \
╠═══════════════════════╝
╠═══════╗
│  2D   │
╚═══════*/
module teardrop (d,a,fn){
polygon(teardrop(d=d==undef?10:d,a=a==undef?30:a,fn=fn==undef?16:fn));
}


module star         (d1,d2,fn){
polygon(star(d1=d1==undef?10:d1,d2=d2==undef?20:d2,fn=fn==undef?7:fn));
}


module ellipse      (dim,fn){
  polygon(ellipse(dim,fn));
}
module losange      (dim){
  polygon(losange(dim));
}
module roundsquare  (table,d,fn){
  polygon(roundsquare(s=table,d=d,fn=fn));
```

```
}
module kochflake    (d,maxit){
  polygon  (
    koch
    (
      ngon(d=d,fn=3),
      maxit=maxit
    )
  );
}
module ngon         (d,fn,inside){
  polygon(ngon(d,fn,inside));
}



module piepart      (d,a,p){
  polygon(piepart(d=d,a=a,p=p));
}

module triangle     (w,h){
  polygon(triangle(w=w,h=h));
}

module fractshape   (d,fn,inside,maxit){
  polygon(fractshape(fn=fn,d=d,maxit=maxit,inside=inside));
}



module lghs         (){
  for(i=[-1:2:1])
  {
    rotate([0,0,i*45]){
      translate([0,-50,0])
      difference(){
        cnc(10)
        union(){
          hull(){
            circle(d=20);
            translate([0,100,0])
            circle(d=15);
          }
          translate([0,100,0])
          circle(d=50);
        }
        hull(){
          translate([0,150,0])
          rotate([0,0,30])
          circle(d=25,$fn=6);
          translate([0,110,0])
```

```
          rotate([0,0,30])
          circle(d=25,$fn=6);
        }
        circle(d=10);
      }
    }
  }
}




module trace(table,d,fn,dot,dotfn,tr,d2){
  tr=tr==undef?true:tr;
  fn=fn==undef?8:dot==true?4:fn;
  dotfn=dotfn==undef?16:dotfn;
  dot=dot==undef?true:dot;
  d=d==undef?1:d;
    d2=d2==undef?d*2:d2;

  for(i=[0:len(table)-2]){
    if(tr==true)
    {
      hull(){
        translate(table[i])
        circle(d=d,$fn=fn);
        translate(table[i+1])
        circle(d=d,$fn=fn);
      }
    }
    if(dot==true){
    translate(table[i]) circle(d=d2,$fn=dotfn);}
    if(dot==true){
    translate(table[i+1]) circle(d=d2,$fn=dotfn);}
  }
}

module voronoi(dim, w, t, c, n,seed){

  n=n==undef?100:n;
  dim=dim==undef?[1000,500]:dim;
  table=pointgrid([dim[0],dim[1]],n=n,seed=seed);
  c=c==undef?0:c;
  t=table[0][0]+table[0][1];
  w=w==undef?1:w;
  seed=seed==undef?1/fn*n*c/w:seed;

  outline(w=w*2,t="in") square(dim);

  difference(){
    square(dim);
```

```
    cnc(-c/2)
    for (p=table){
      intersection_for(p2=table){
        if (p!=p2){
          translate((p+p2)/2 -normal(p2-p)*w){
            rotate([0,0,-myangle(p,p2)])
            translate([-t,-t])
            square([2*t, t]);
}}}}}}


/*═══════╗
│  2D/3D  │
╘═══════*/
/*═══════╗
│  3D  │
╘═══════*/
module tube       (d1,d2,h,center){
  d1=d1==undef?10:d1;
  d2=d2==undef?8:d2;
  h=h==undef?30:h;
  center=center==undef?false:center;
  translate([0,0,center==true?-h/2:0])
  difference(){
  cylinder(d=d1,h=h);
  translate([0,0,-1])
  cylinder(d=d2,h=h+2);
  }
}

module coude      (d1,d2,a){
  d1=d1==undef?10:d1;
  d2=d2==undef?8:d2;
  a=a==undef?90:a<=-90?-90:a>=90?90:a;
  difference(){
    union(){
      cylinder(d=d1,h=d1/2);
      translate([0,0,d1/2])
      sphere(d=d1);
      translate([0,0,d1/2])
      rotate([0,a,0])
      cylinder(d=d1,h=d1/2);
    }
    union(){
      translate([0,0,-1])
      cylinder(d=d2,h=d1/2+1);
      translate([0,0,d1/2])
      sphere(d=d2);
      translate([0,0,d1/2])
      rotate([0,a,0])
      cylinder(d=d2,h=d1/2+1);
```

```
    }
  }
}

module roundcube  (s,b,t,center,q){
  s=s==undef?[50,40,30]:s;
  b=b==undef?[5,5,5,5]:b;
  t=t==undef?[20,20,20,20]:t;
  q=q==undef?16:q;
  center=center==undef?false:center;
  translate([center==true?-s[0]/2:0,center==true?-
s[1]/2:0,center==true?-s[2]/2:0])
  hull(){
    translate([b[0]/2,b[0]/2,b[0]/2]) sphere(d=b[0],$fn=q);
    translate([s[0]-b[1]/2,b[1]/2,b[1]/2]) sphere(d=b[1],$fn=q);
    translate([s[0]-b[2]/2,s[1]-b[2]/2,b[2]/2]) sphere(d=b[2],$fn=q);
    translate([b[3]/2,s[1]-b[3]/2,b[3]/2]) sphere(d=b[3],$fn=q);
    translate([t[0]/2,t[0]/2,s[2]-t[0]/2]) sphere(d=t[0],$fn=q);
    translate([s[0]-t[1]/2,t[1]/2,s[2]-t[1]/2]) sphere(d=t[1],$fn=q);
    translate([s[0]-t[2]/2,s[1]-t[2]/2,s[2]-t[2]/2])
sphere(d=t[2],$fn=q);
    translate([t[3]/2,s[1]-t[3]/2,s[2]-t[3]/2]) sphere(d=t[3],$fn=q);
  }
}

module bone       (h,d1,d2,c,q){
  h    = h    == undef ? 50       : h;
  d1   = d1   == undef ? 20       : d1;
  d2   = d2   == undef ? 14.14214 : d2;
  c    = c    == undef ? 40       : c;
  q    = q    == undef ? 128      : q;

  rotate_extrude(){
    rotate_extrude_correct(){
      cnc((c)*1,fn=q)
      {
        translate([0,h,0]) circle(d=d2,$fn=q);
        translate([0,0])
        circle(d=d1,$fn=q);
      }
    }
  }
}

module rock(d,c,seed){
  c=c==undef?3:c;
    seed=seed==undef?1:seed;
  intersection_for(i=[0:c]){
a=[random(360,s=i*seed),random(360,s=i*seed*2),random(360,s=i*seed*3)];
    rotate(a)
    cube([d,d,d*10],center=true);
```

```
  }
}

module pie        (d,p,pct,i=1,a=0){
  pct=topct(p);
  echo(pct);
  rotate([0,0,a*360])
  linear_extrude(i)
  piepart(d=d,p=p[i-1]/sum(p)+0.01);
  if(i<len(p)){
    pie(d=d,p=p,i=i+1,a=a+pct[i-1]);
  }
}
/*═══════════════════════════╗
│  Nouvelles fonctions   \
╞═══════════════════════════╝
╞═══════════════════╗
│  Sur nombres (a)  │
╘═══════════════════*/
function random    (n,s,pos)       = rands(pos==undef?0:pos==true?0:-
n,n,1,s==undef?n:s)[0];
function fibonacci (n,a=0,b=1,c=1) =
c<n+1?fibonacci(a=b,b=a+b,c=c+1,n=n):a;
function hypo      (a,b)           = sqrt((a*a)+(b*b));
function real      (a,b,c)         = ((a*a)+(b*b))+c;
function imaginary (a,b,c)         = (2*a*b)+c;
function pair      (a)             = a%2==0?true:false;
function normal    (a)             = a/(sqrt(a[0]*a[0]+a[1]*a[1]));
/*═══════════════════════════════╗
│  Sur tables [a,b,c, ...] │
╘═══════════════════════════════*/
function sum     (a,b=0,c=0,n)       =
b<(n==undef?len(a):n)?sum(a=a,b=b+1,c=c+a[b],n=n):c;
function topct   (a)               = a/sum(a);
function moyenne  (a,b=0,c=0)       =
b<len(a)?sum(a=a,b=b+1,c=c+a[b])/len(a):c;
function invert  (a)               = let(b=[for(i=[0:len(a)-1])
a[(len(a)-1)-i]])b;
function sort    (a,invert=false)  = len(a) == 0 ? [] : let (
      b=floor(len(a)/2),
      c=[for(i=a) if (i<a[b]) i],
      d=[for(i=a) if (i>a[b]) i],
      e=[for(i=a) if (i==a[b]) i]
    )
invert==false?concat(sort(c),e,sort(d)):invert(concat(sort(c),e,sort(d)
));
/*═══════════════════════════════════╗
│  Sur vecteurs [[a,b],[c,d]] │
╘═══════════════════════════════════*/
function length        (a,b)           = sqrt(((b[0]-a[0])*(b[0]-
a[0]))+((b[1]-a[1])*(b[1]-a[1])));
function divide         (a,b,c)         = [a[0]+(b[0]-a[0])*c,
```

```
a[1]+(b[1]-a[1])*c];
function myangle        (a,b)           = atan2(b[0]-a[0],b[1]-a[1]);
function join           (a,c=0,t=[])    = let
(u=concat(t,a[c]))c==len(a)?t:join(a=a,c=c+1,t=u);
// NIGHTLY BUILDS ONLY
function join2(aa) = [for(i=[0:len(aa)-1] ) each aa[i]];
function clean(a) = [for(i=[0:len(a)-1]) each
(a[i]==a[i+1]?"":a[i][0]==undef?"":[a[i]])];
function koch           (a,angle,maxit,it)       =  let(
a=a[0]==a[len(a)-1]?a:concat(a,[a[0]]),
b = [ for ( i = [ 0 : len(a)-2  ] ) [
      a[i],
      divide(a[i],a[i+1],1/3),
      divide(a[i],a[i+1],1/3) + [
        sin(myangle(a[i],a[i+1])-
(angle==undef?60:angle<=60?60:angle>=180?180:angle))*length(a[i],
a[i+1])/3,
        cos(myangle(a[i],a[i+1])-
(angle==undef?60:angle<=60?60:angle>=180?180:angle))*length(a[i],a[i+1]
)/3],

  divide(a[i],a[i+1],2/3)+[sin(-90+myangle(a[i],a[i+1])-((90+(90-
(angle<=60?60:angle>=180?180:angle)))))*length(a[i],a[i+1])/3,cos(-90+m
yangle(a[i],a[i+1])-(90+(90-
(angle<=60?60:angle>=180?180:angle))))*length(a[i],a[i+1])/3],
  divide(a[i],a[i+1],2/3),
  a[i+1]
  ]],
  maxit=maxit==undef?0:maxit,
  it=it==undef?0:it
    )
it==maxit?clean(join2(b)):koch(a=clean(join2(b)),angle=angle,maxit=maxi
t,it=it+1);

function fract(a,angle,in,maxit,it,close)= let(
    close=close==undef?true:close,
    a=close==true?a[0]==a[len(a)-1]?a:concat(a,[a[0]]):a,
  maxit=maxit==undef?3:maxit==0?1:maxit,
    it=it==undef?0:it,
        inside=in==undef?1:in==true?1:-1,
    angle=angle==undef?60:angle,
  b = [ for ( i = [ 0 : len(a)-2  ] ) [
      a[i],
      divide(a[i],a[i+1],angle/180),
      divide(a[i],a[i+1],angle/180) +
            [
        sin(myangle(a[i],a[i+1]) + angle*inside) *
(length(a[i],a[i+1])/3) ,
        cos(myangle(a[i],a[i+1]) + angle*inside) *
(length(a[i],a[i+1])/3)
            ],
```

```
      divide(a[i],a[i+1],1-(angle/180)),
      a[i+1]]
    ]
  )
it+1==maxit?clean(join2(b)):fract(a=clean(join2(b)),angle=angle,in=in,m
axit=maxit,it=it+1,close=close);

/*
    function fract           (a,angle,in,maxit,it)             = let(
  a=a[0]==a[len(a)-1]?a:concat(a,[a[0]]),
  maxit=maxit==undef?3:maxit==0?1:maxit,
    it=it==undef?0:it,
    angle=angle==undef?60:angle,

  b = [ for ( i = [ 0 : len(a)-2  ] ) [
      a[i],

      divide(a[i],a[i+1],angle/180),

      divide(a[i],a[i+1],1/2) +
            [
        sin(myangle(a[i],a[i+1]) + (in==false?-90:90)) *
(length(a[i],a[i+1])/3) ,

        cos(myangle(a[i],a[i+1]) + (in==false?-90:90)) *
(length(a[i],a[i+1])/3)
            ],

      divide(a[i],a[i+1],1-(angle/180)),

      a[i+1]]
    ]
  )

it+1==maxit?clean(join2(b)):fract(a=clean(join2(b)),angle=angle,in=in,m
axit=maxit,it=it+1);
*/


function curve(table,fn) = let(
    fn = fn == undef ? 8 : fn,
    c = [ for ( i = [0:(fn)] ) each
[divide(table[0],table[1],1/(fn)*i)]],
    d = [ for ( i = [0:(fn)] ) each
[divide(table[1],table[2],1/(fn)*i)]],
    e = [ for ( i = [0:(fn)] ) each [divide(c[i],d[i],1/(fn)*i)]])
  e;

function doublevector(table,f,it=0) = let(
    f=f==undef?0:f,
    aa = [for (i=[0:len(table)-1]) each
```

```
[table[i],divide(table[i],table[i+1],0.5)]]
      )
    it==f?clean(aa):doublevector(clean(aa),f=f,it=it+1);

function ngon(d,fn,inside) = let (
d=d==undef?10:inside==undef?d:inside==true?d:d*((d/2)/(cos(360/fn/2)*d/
2)),
    fn=fn==undef?4:fn,
    aa=[for(i=[0:fn])[sin(360/fn*i)*d/2,cos(360/fn*i)*d/2]]
    )
  aa;

function square(d,center)=let(
    center=center==undef?false:center,
    d=d==undef?[10,10]:d,
    c1  = center  ==  true ? [-d[0]/2,-d[1]/2]  :  [    0,    0],
    c2  = center  ==  true ? [-d[0]/2, d[1]/2]  :  [    0, d[1]],
    c3  = center  ==  true ? [ d[0]/2, d[1]/2]  :  [ d[0], d[1]],
    c4  = center  ==  true ? [ d[0]/2,-d[1]/2]  :  [ d[0], 0],
    aa=[c1,c2,c3,c4,c1]
  )
  aa;

function ellipse(s,fn) = let (
    fn=fn==undef?16:fn,
    s=s==undef?[10,10*aphi]:s,
    aa=[for(i =[0:fn] ) [sin(360/fn*i)*s[0]/2,cos(360/fn*i)*s[1]/2]]
  )
  aa;

function losange(s) = let (
    s=s==undef?[10,10*aphi]:s,
    aa=[for(i =[0:4] ) [sin(360/fn*i)*s[0]/2,cos(360/fn*i)*s[1]/2]]
  )
  aa;

function circle(d,r,fn) = let (
    fn=fn==undef?16:fn,
    r=r==undef?d==undef?5:d/2:r,
    aa=ngon(d=r*2,fn=fn)
  )
  aa;

function star(d1,d2,fn) = let (
    d1=d1==undef?10:d1/2,
    d2=d2==undef?5:d2/2,
    fn=fn==undef?7:fn,
aa=[for(i=[0:2*(fn)])[sin(360/(2*fn)*i)*(pair(i)==true?d1:d2),cos(360/(
2*fn)*i)*(pair(i)==true?d1:d2)]]
  )
  aa;
```

```
function roundsquare(s,d,fn) = let (
  fn = fn == undef ? 8:fn,
  s = s == undef ? [15,20] : s,
  d = d == undef ? [3,6,3,6] : len(d) == 1 ?
[d[0]/2,d[0]/2,d[0]/2,d[0]/2]:len(d)==2?[d[0]/2,d[1]/2,d[1]/2,d[1]/2]:l
en(d)==3?[d[0]/2,d[1]/2,d[2]/2,d[2]/2]:d[0]==undef?[d/2,d/2,d/2,d/2]:d/
2,

  p1 = [0,0],
  p2 = [0,d[0]],
  p3 = [0,s[1]-d[1]],
  p4 = [0,s[1]],
  p5 = [d[1],s[1]],
  p6 = [s[0]-d[2],s[1]],
  p7 = [s[0],s[1]],
  p8 = [s[0],s[1]-d[2]],
  p9= [s[0],d[3]],
  p10= [s[0],0],
  p11= [s[0]-d[3],0],
  p12= [d[0],0],

  c1=curve([p3,p4,p5],fn=fn),
  c2=curve([p6,p7,p8],fn=fn),
  c3=curve([p9,p10,p11],fn=fn),
  c4=curve([p12,p1,p2],fn=fn),

  aa=clean(join2([c1,c2,c3,c4,[p3]]))

)

  aa;

function piepart(d,a,p) = let (
  d=d==undef?10:d/2,
  a=a==undef?p==undef?90:p>=1?360*1/p:360*p:a,
  aa=concat([[0,0]],[for(i=[0:a])[-sin(-90+i)*d,cos(-90+i)*d]])
)
  aa;

function triangle(w,h)= let (
  h=h==undef?cos(30)*w:h,
  aa=[[-w/2,0],[0,h],[w/2,0],[-w/2,0]]
)
  aa;



function fractshape(d,fn,inside,maxit)= let(
  d=d==undef?10:d/2,
  fn=fn==undef?5:fn,
```

```
  maxit=maxit==undef?3:maxit,
  inside=inside==undef?true:inside,
  angle=fn==3?60:fn==4?89:360/fn,
  points=fract(ngon(d=d*2,fn=fn),maxit=maxit,angle=angle,in=inside)
)
points;

function teardrop(d,a,fn)=let (
  d=d==undef?10:d,
  a=a==undef?30:a,
  h=d*tan(90-a),
  fn=fn==undef?16:fn,
  courbe= [for(i=[0:fn]) [sin(90-a+(360-(90-a)*2)/fn*i)*d/2,cos(90-
a+(360-(90-a)*2)/fn*i)*d/2]],
  aa=concat(courbe,[[0,(cos(90-a)*d/2)+h*sin(90-a)/2]],[[sin(90-
a)*d/2,cos(90-a)*d/2]])
)
aa;


function fractalize(table,force,maxit,seed)= let (
  force=force==undef?1:force,
    maxit=maxit==undef?3:maxit,
    seed=seed==undef?1:seed,
    aa=
    [
      for(i=[0:len(table)-2], ab =
doublevector([[table[i][0],table[i][1]],[table[i+1][0],table[i+1][1]]],
maxit=maxit))
        for(j=ab[0]) each
clean([[ab][0],[ab][1]]+[[(random(pos=false,n=force,s=sin(seed)*sin(ab[
0])+sin(ab[1]))),(random(pos=false,n=force,s=cos(seed)*cos(ab[0])-
sin(ab[1])))],[(random(n=force,s=tan(seed)+sin(ab[0])+2*cos(ab[1]))),(r
andom(n=force,s=cos(seed)+cos(ab[0])-3*cos(ab[1])))]])
    ]
 )
 clean(aa);



function pointgrid(dim,n,seed)
=[for(i=[0:n-1])[random(n=dim[0],s=sin(i/n/(seed==undef?1:seed))),rando
m(n=dim[1],s=cos(i*2/n/(seed==undef?1:seed)))]];


function rescale(a,s) = [for (i=a) i*s];
function retranslate(a,t) = [for (i=a) i+t];
function 2Drot(object,angle) = [for(i=[0:len(object)-1])
[sin(myangle([0,0],object[i])+angle)*length([0,0],object[i]),cos(myangl
e([0,0],object[i])+angle)*length([0,0],object[i])]];
```

```
function to3D(a,b,h,bottom,top) = let (
  bottom=bottom==undef?true:bottom,
  top=top==undef?true:top,

  aa=[    for(i=[0:len(a)])
each[[a[i][0],a[i][1],0],[b[i][0],b[i][1],h]]  ],
  bb=[    for(i=[0:1:len(aa)]) each [[i,i+1,i+2],[i+1,i+3,i+2]]    ],

  cc=bottom==true?[    for(i=[0:2:len(aa)]) each [i]   ]:[],
//  dd=top==true?[    for(i=[0:2:len(aa)]) each [len(aa)-1-i]   ]:[],
dd=top==true?[    for(i=[0:2:len(aa)]) each [len(aa)-1-i]   ]:[],

  ee=concat(bb,[cc],[dd])
  )
[clean(aa),clean(ee)];

function simple3D(a,b,h,bottom,top,angle,correct) = let (
    angle=angle==undef?0:angle,
    correct=correct==undef?0:correct,
    bottom=true,
    top=true,
c=2Drot(interpolate(a,b,maxstep=1,step=0,correct=correct,q=1),angle),
d=2Drot(interpolate(a,b,maxstep=1,step=1,correct=correct,q=1),angle),
    aa=[    for(i=[0:len(c)])
each[[c[i][0],c[i][1],0],[d[i][0],d[i][1],h]]  ],
    bb=[    for(i=[0:1:len(aa)]) each [[i,i+1,i+2],[i+1,i+3,i+2]]    ],
    cc=bottom==true?[    for(i=[0:2:len(aa)]) each [i]   ]:[],
    dd=top==true?[    for(i=[0:2:len(aa)]) each [len(aa)-1-i]   ]:[],
    ee=concat(bb,[cc],[dd])
  )
[clean(aa),clean(ee)];

function vectranslate(a,n,it)=let(
  n=n==undef?0:n,
  it=it==undef?0:it,
  aa=n==0?a:[for(i=[0:len(a)-1])  i==0?a[len(a)-2]:a[i-1]]
  )
  it==n+len(a)-2?aa:vectranslate(a=aa,n=n,it=it+1);



module 2Dto3D(a,b,h,segment,correct,quality,rotation){
angle=rotation==undef?0:rotation/segment;
quality=quality==undef?1:quality;
he=h==undef?64:h;
mm=segment==undef?16:segment;
aabc=a==undef?ngon(d=50,fn=3):a;
adef=b==undef?chaincurve(koch(ngon(d=50,fn=3),maxit=1),fn=4):b;
correct=correct==undef?0:correct;

  union(){
```

```
    for(i=[0:mm-1]){
      my3Dobject=to3D(
2Drot(interpolate(aabc,adef,maxstep=mm,step=i,correct=correct,q=quality
),i*angle),
2Drot(interpolate(aabc,adef,maxstep=mm,step=i+1,correct=correct,q=quali
ty),(i+1)*angle),
        h=he/mm,
        top=i==mm-1?true:true,
        bottom=i==0?true:true);

      translate([0,0,i*he/mm])
      color([1/mm*i,1-(1/mm*i),1,1])
      union()
      {
        polyhedron(my3Dobject[0],my3Dobject[1]);
        polyhedron(my3Dobject[0],my3Dobject[1]);
      }
    }
  }
}


function interpolate(a,b,step,maxstep,correct,q)= let(
pp=ppcm(len(a)-1,len(b)-1,q)-[1,1],
correct=correct==undef?0:correct,
abc=vectranslate(multiplyfaces(a,pp[0]),n=correct==undef?0:correct),
  def=multiplyfaces(b,pp[1]),
  aa=[for(i=[0:len(def)]) each [divide(abc[i],def[i],step/(maxstep))]])
  clean(aa);

function ppcm(a,b,q)=let(
  aa=[for(i=[0:max(a,b)]) each [i*a*(q==undef?1:q)]],
  bb=[for(i=[0:max(a,b)]) each [i*b*(q==undef?1:q)]],
  cc=[for(i=[0:max(a,b)]) each [for(j=[1:100]) each
aa[i]==bb[j]?bb[j]:""]])
  [cc[0]/a,cc[0]/b];


function multiplyfaces(object,n)=let(
n=n==undef?1:n==0?0:n,
aa=n==0?object:[for(i=[0:len(object)-2]) each
addpoints(object[i],object[i+1],n)
])
concat(clean(aa),[object[0]]);

function addpoints(c1,c2,n)=[for(i=[0:n]) each
[divide(c1,c2,i/(n+1))]];

function chaincurve(table,fn,closed,detail) = let (
  detail=detail==undef?1:detail==0?1:detail*sign(detail)*2+1,
  closed=closed==undef?true:closed,
```

```
//
table=table[0]==table[len(table)-1]?table:concat([for(i=[0:len(table)])
table[i]],table[len(table)-1]),
  //totaltab=table,
totaltab=concat(table,[table[0]],closed==true?[table[1]]:"",closed==tru
e?[table[2]]:""),
  tab = multiplyfaces(totaltab,detail),
  d = [for (i=[(closed==true?4:2):2:len(tab)-(closed==false?4:2)]) each
curve([tab[(i)-1],tab[i],tab[i+1]],fn)],
  b =table[0],
  c = table[len(table)-1],
  a = d
  )
  clean(a);


function RVB(a,b,c,d)= let( a=a==undef?0.5:1/255*a,
b=b==undef?0.5:1/255*b,
c=c==undef?0.5:1/255*c,
d=d==undef?1:1/255*d,
aa=[a,b,c,d]
)aa;

function mirror(a,x,y)=let(
  xx=x==undef?1:x==true?-1:1,
  yy=y==undef?1:y==true?-1:1,
  aa=[
    for(i=[0:len(a)-1])
      each
      [
        [a[i][0]*xx,a[i][1]*yy]
      ]
  ]
)aa;


function center(a,center)=let(
    LMax=sort([for(i=[0:len(a)-1])a[i][0]])[0],
    LMin=sort([for(i=[0:len(a)-1])a[i][0]],invert=true)[0],
    HMax=sort([for(i=[0:len(a)-1])a[i][1]])[0],
    HMin=sort([for(i=[0:len(a)-1])a[i][1]],invert=true)[0],
    LargMax=LMax*sign(LMax),
    LargMin=LMin*sign(LMin),
    HautMax=HMax*sign(HMax),
    HautMin=HMin*sign(HMin),
    Largeur=LargMax<=LargMin?LargMin-LargMax:LargMax-LargMin,
    Hauteur=HautMax<=HautMin?HautMin-HautMax:HautMax-HautMin,
    aa=retranslate(a,[-LMax-(center==false?0:Largeur/2),-HMax-
(center==false?0:Hauteur/2)])
)aa;
```

```
function offset(a,d,i)= let
(
  invert=i==undef?false:true,
  aa=mesangles(a),
  mesangles=orientangles(aa),
  bb=[
    for(i=[0:len(a)-2]) each [[a[i][0]+sin(90-
mesangles[i])*d,a[i][1]+cos(90-mesangles[i])*d]]
  ]
)

concat(bb,[bb[0]]);


function mesangles(a,invert)=let
(
  invert=invert==undef?false:true,
  aa=[
      for(i=[0:len(a)-2])
        let(
          a1=(invert==false?90:-90)-myangle(a[i],a[i==len(a)-2?0:i+1]),
          a2=(invert==false?90:-90)-myangle(a[i],a[i==0?len(a)-2:i-1]),

          aa1=(invert==false?90:-90)-
myangle(a[i],a[i==len(a)-2?0:i+1]),
          aa2=(invert==false?90:-90)-
myangle(a[i],a[i==0?len(a)-2:i-1]),
          a3=(a1+a2)/2+180,
                    aaa=aa[i-1]
        )
      a3
    ]
)
aa;



function orientangles(a,i)=
let
(
  i=i==undef?0:i,
  aa=[
    for(j=[0:len(a)-1])
      each (a[j]-a[j-1])>=90?a[j]-180:(a[j]-a[j-1])<=-90?a[j]+180:a[j]]
)
i==len(a)-1?aa:orientangles(a=aa,i=i+1);
```

```
function coeffdirect(a)=let //coefficient directeur
(
aa=(a[1][1]-a[0][1])/(a[1][0]-a[0][0])
)
aa;

/*
  Dans un plan cartésien, on peut trouver les coordonnées du point d’intersection de deux courbes (comme par exemple deux droites) en résolvant le système d’équations.
  Soit les droites dont les équations sont y = x – 4 et y = –2x + 5, alors : x – 4 = –2x + 5. On représente ces droites dans un plan cartésien.
  Donc : 3x = 9 et x = 3
  Puis :  y = –1
  Les coordonnées du point d’intersection de ces droites sont (3, –1).
*/




module 2D(a){
  polygon(a);
}

module 3D(a){
    if(a[1]!=undef)
    {
        polyhedron(a[0],a[1]);
    }
}

function cube(d,center)=let
  (
    d=d==undef?[10,10,10]:d,
    center=center==undef?false:center,
    mys=square([d[0],d[1]],center=center),
    aa=to3D(mys,mys,h=d[2])
  )
  center==false?aa:translate3D(aa,[0,0,-d[2]/2]);

function cylinder(d,d1,d2,h,fn,center)=let
  (
    d1=d==undef?d1==undef?10:d1:d,
    d2=d==undef?d2==undef?10:d2:d,
    h=h==undef?40:h,
    fn=fn==undef?16:fn,
    center=center==undef?false:center,
```

```
    myg1=ngon(d=d1,fn=fn),
    myg2=ngon(d=d2,fn=fn),
    aa=to3D(myg1,myg2,h=h)
  )
  center==false?aa:translate3D(aa,[0,0,-h/2]);

function translate3D(a,b)=[[for(i=[0:len(a[0])-1]) a[0][i]+b],a[1]];
function rescale3D(a,s)=  [[for(i=[0:len(a[0])-1]) a[0][i]*s],a[1]];




function gradient(a,b,c) = let
(
  c=c==undef?1:c,
  aR=a[0],      aV=a[1],      aB=a[2],      aA=a[3],
  bR=b[0],      bV=b[1],      bB=b[2],      bA=b[3],
  mR=(aR-bR)/c, mV=(aV-bV)/c, mB=(aB-bB)/c, mA=(aA-bA)/c,
  aa=[for(i=[0:c-1]) [a[0]-mR*i,a[1]-mV*i,a[2]-mB*i,a[3]-mA*i]]
)
aa;


module menger(d,maxit,it){
    let(it=it==undef?0:it)
    if(it==maxit){
        square([d,d]);
    }
    else{
union(){
        menger(d=d/3+0.001,maxit=maxit,it=it+1);
        translate([d/3,0])
        menger(d=d/3+0.001,maxit=maxit,it=it+1);
        translate([d/3*2,0])
        menger(d=d/3+0.001,maxit=maxit,it=it+1);
        translate([0,d/3])
        menger(d=d/3+0.001,maxit=maxit,it=it+1);
        translate([d/3*2,d/3])
        menger(d=d/3+0.001,maxit=maxit,it=it+1);
        translate([0,d/3*2])
        menger(d=d/3+0.001,maxit=maxit,it=it+1);
        translate([d/3,d/3*2])
        menger(d=d/3+0.001,maxit=maxit,it=it+1);
        translate([d/3*2,d/3*2])
        menger(d=d/3+0.001,maxit=maxit,it=it+1);
}}}

module menger3d(it,d,maxit){
    it=it==undef?1:it;
    if (it==maxit){
        cube([d,d,d],center=true);
```

```
    }
    if (it<=maxit){
        union(){
        for (i=[-1:1]){
            translate([d/3,d/3,d/3*i]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
            translate([-d/3,d/3,d/3*i]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
            }

        translate([0,d/3,d/3]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
        translate([0,d/3,-d/3]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);

        for (i=[-1:1]){
            translate([d/3,-d/3,d/3*i]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
            translate([-d/3,-d/3,d/3*i]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
            }
        translate([0,-d/3,d/3]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
        translate([0,-d/3,-d/3]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
        translate([d/3,0,d/3]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
        translate([d/3,0,-d/3]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
        translate([-d/3,0,d/3]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
        translate([-d/3,0,-d/3]) rotate([0,90,0])
menger3d(it=it+1,d=d*1/3,maxit=maxit);
}}}

module jcube(it,d,maxit){
    it=it==undef?1:it;
union()
    {
        if(it==maxit)
        {
            cube([d,d,d],center=true);
        }
        if(it<=maxit)
        {
            translate([d/2,d/2,d/2])
            translate([-d/2*(sqrt(2)-1),-d/2*(sqrt(2)-1),-
d/2*(sqrt(2)-1)])
            jcube(it=it+1,maxit=maxit,d=d*(sqrt(2)-1));

            translate([-d/2,d/2,d/2])
```

```
            translate([d/2*(sqrt(2)-1),-d/2*(sqrt(2)-1),-
d/2*(sqrt(2)-1)])
            jcube(it=it+1,maxit=maxit,d=d*(sqrt(2)-1));

            translate([d/2,-d/2,d/2])
            translate([-d/2*(sqrt(2)-1),d/2*(sqrt(2)-1),-
d/2*(sqrt(2)-1)])
            jcube(it=it+1,maxit=maxit,d=d*(sqrt(2)-1));

            translate([-d/2,-d/2,d/2])
            translate([d/2*(sqrt(2)-1),d/2*(sqrt(2)-1),-
d/2*(sqrt(2)-1)])
            jcube(it=it+1,maxit=maxit,d=d*(sqrt(2)-1));

            translate([d/2,d/2,-d/2])
            translate([-d/2*(sqrt(2)-1),-
d/2*(sqrt(2)-1),d/2*(sqrt(2)-1)])
            jcube(it=it+1,maxit=maxit,d=d*(sqrt(2)-1));

            translate([-d/2,d/2,-d/2])
            translate([d/2*(sqrt(2)-1),-
d/2*(sqrt(2)-1),d/2*(sqrt(2)-1)])
            jcube(it=it+1,maxit=maxit,d=d*(sqrt(2)-1));

            translate([d/2,-d/2,-d/2])
            translate([-
d/2*(sqrt(2)-1),d/2*(sqrt(2)-1),d/2*(sqrt(2)-1)])
            jcube(it=it+1,maxit=maxit,d=d*(sqrt(2)-1));

            translate([-d/2,-d/2,-d/2])
translate([d/2*(sqrt(2)-1),d/2*(sqrt(2)-1),d/2*(sqrt(2)-1)])
            jcube(it=it+1,maxit=maxit,d=d*(sqrt(2)-1));

            translate([-d/2,-d/2,0])
            translate([d/2*(1-(2*(sqrt(2)-1))),d/2*(1-
(2*(sqrt(2)-1))),0])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([-d/2,d/2,0])
            translate([d/2*(1-(2*(sqrt(2)-1))),-d/2*(1-
(2*(sqrt(2)-1))),0])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([d/2,-d/2,0])
            translate([-d/2*(1-(2*(sqrt(2)-1))),d/2*(1-
(2*(sqrt(2)-1))),0])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([d/2,d/2,0])
            translate([-d/2*(1-(2*(sqrt(2)-1))),-d/2*(1-
(2*(sqrt(2)-1))),0])
```

```
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([-d/2,0,-d/2])
            translate([d/2*(1-(2*(sqrt(2)-1))),0,d/2*(1-
(2*(sqrt(2)-1)))])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([-d/2,0,d/2])
            translate([d/2*(1-(2*(sqrt(2)-1))),0,-d/2*(1-
(2*(sqrt(2)-1)))])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([d/2,0,-d/2])
            translate([-d/2*(1-(2*(sqrt(2)-1))),0,d/2*(1-
(2*(sqrt(2)-1)))])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([d/2,0,d/2])
            translate([-d/2*(1-(2*(sqrt(2)-1))),0,-d/2*(1-
(2*(sqrt(2)-1)))])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([0,-d/2,-d/2])
            translate([0,d/2*(1-(2*(sqrt(2)-1))),d/2*(1-
(2*(sqrt(2)-1)))])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([0,-d/2,d/2])
            translate([0,d/2*(1-(2*(sqrt(2)-1))),-d/2*(1-
(2*(sqrt(2)-1)))])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([0,d/2,-d/2])
            translate([0,-d/2*(1-(2*(sqrt(2)-1))),d/2*(1-
(2*(sqrt(2)-1)))])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));

            translate([0,d/2,d/2])
            translate([0,-d/2*(1-(2*(sqrt(2)-1))),-d/2*(1-
(2*(sqrt(2)-1)))])
            jcube(it=it+1,maxit=maxit,d=d*(1-(2*(sqrt(2)-1))));
}}}
```