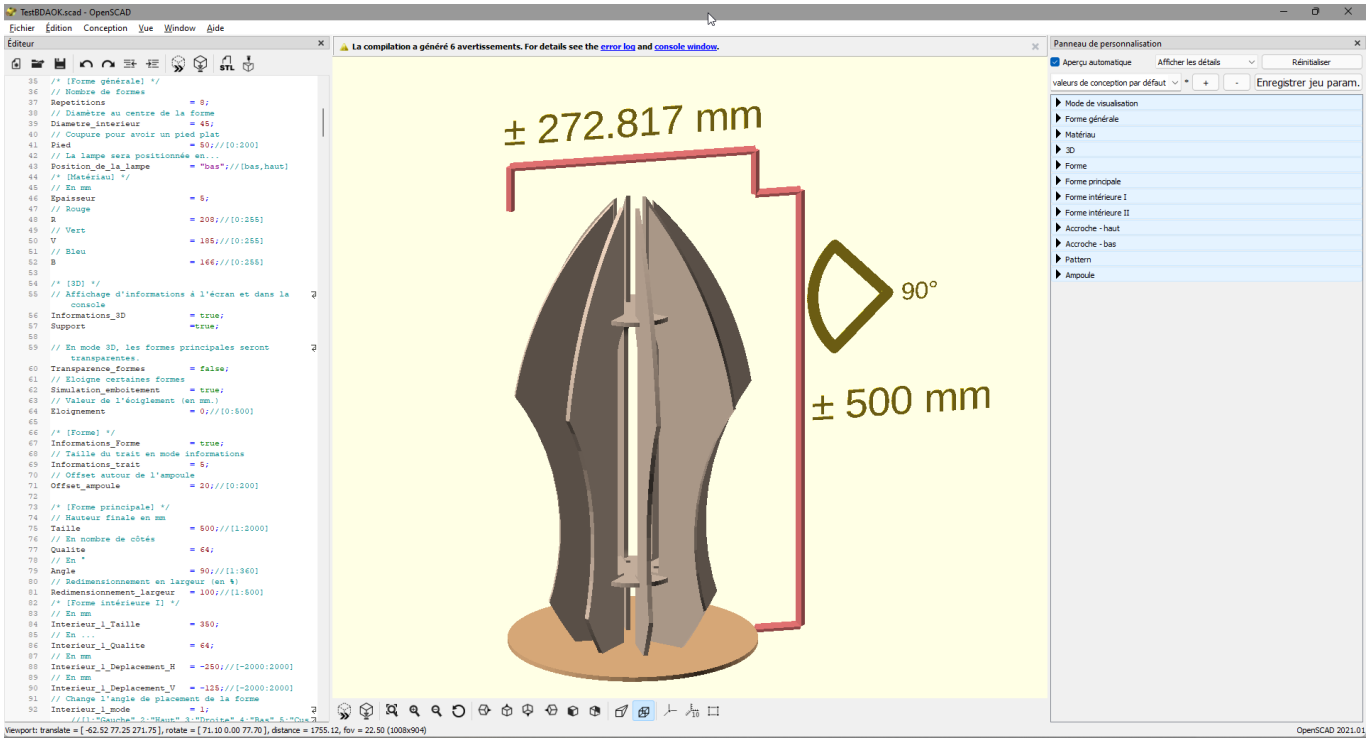
Générateur de luminaires

Ce générateur créera les plans DXF à utiliser avec une découpeuse laser afin d'obtenir le résultat voulu, prévisualisable dans le générateur.

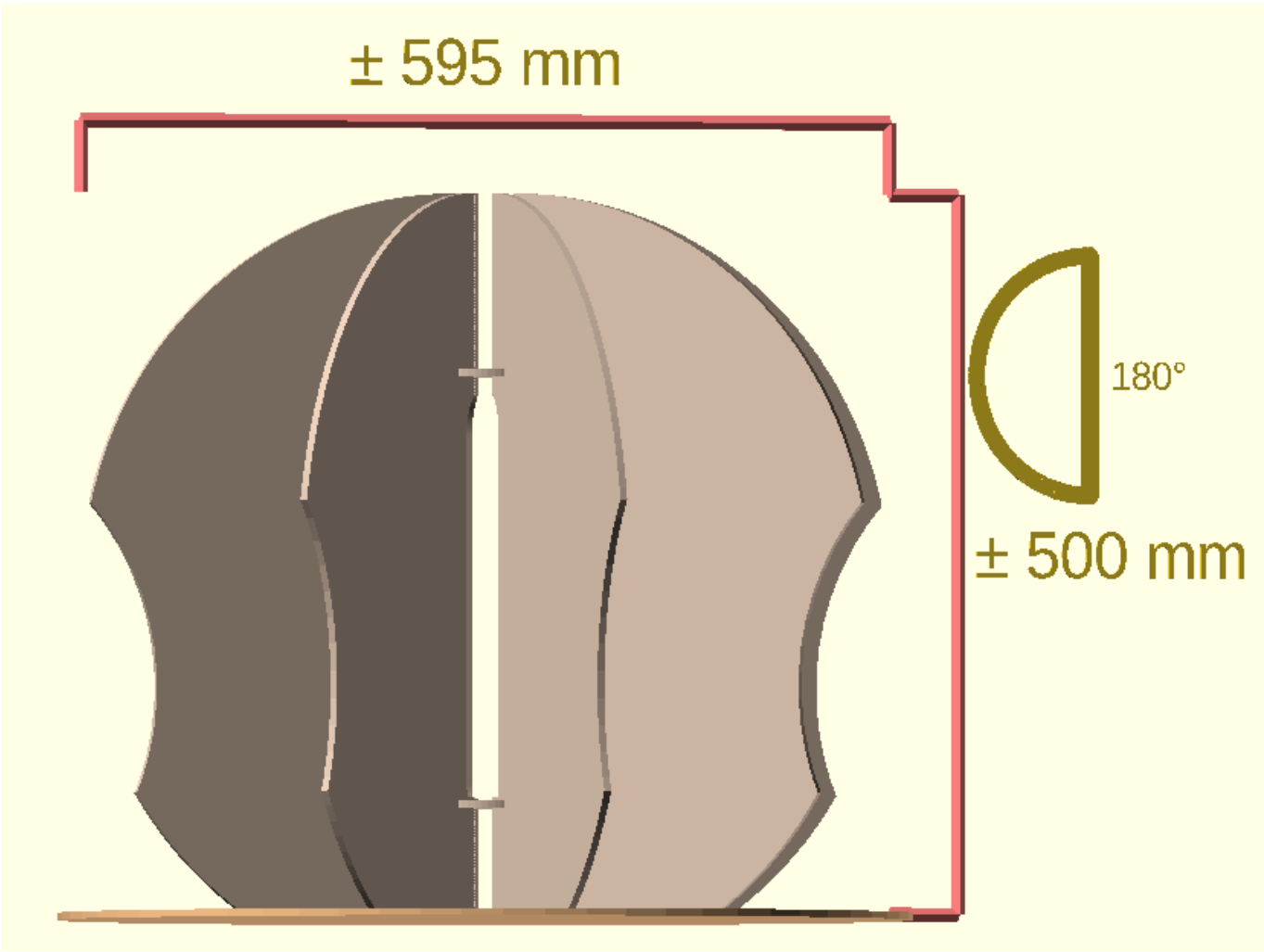
Par exemple :



Nous pouvons voir sur cette capture quatre choses :

1. L'objet en 3D.
2. La largeur totale de l'objet
3. La hauteur de l'objet
4. L'angle utilisé, indiquant la courbure de chaque élément.

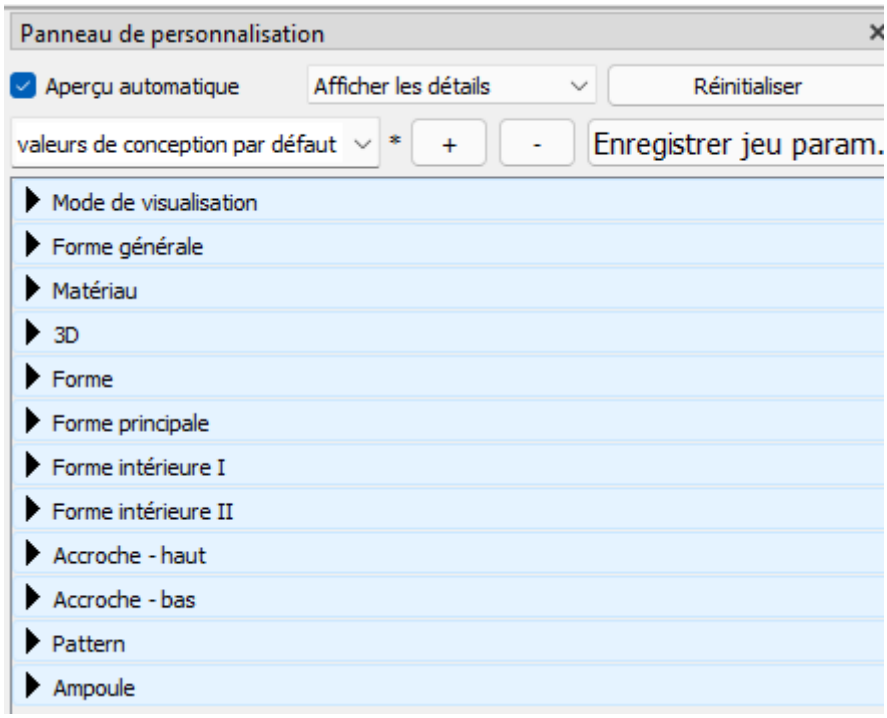
Ainsi, si vous désirez une lampe “ronde”, il suffit de mettre cet angle à 180° :



Ce générateur est complexe dans le sens où il existe énormément de variables pouvant influer sur la forme, le tout sans aucun garde-fou vous indiquant si vous faites une mauvaise utilisation de l'outil.

Merci vous référer à ce qui suit avant de me contacter ;)

Les options de customization



Mode de visualisation

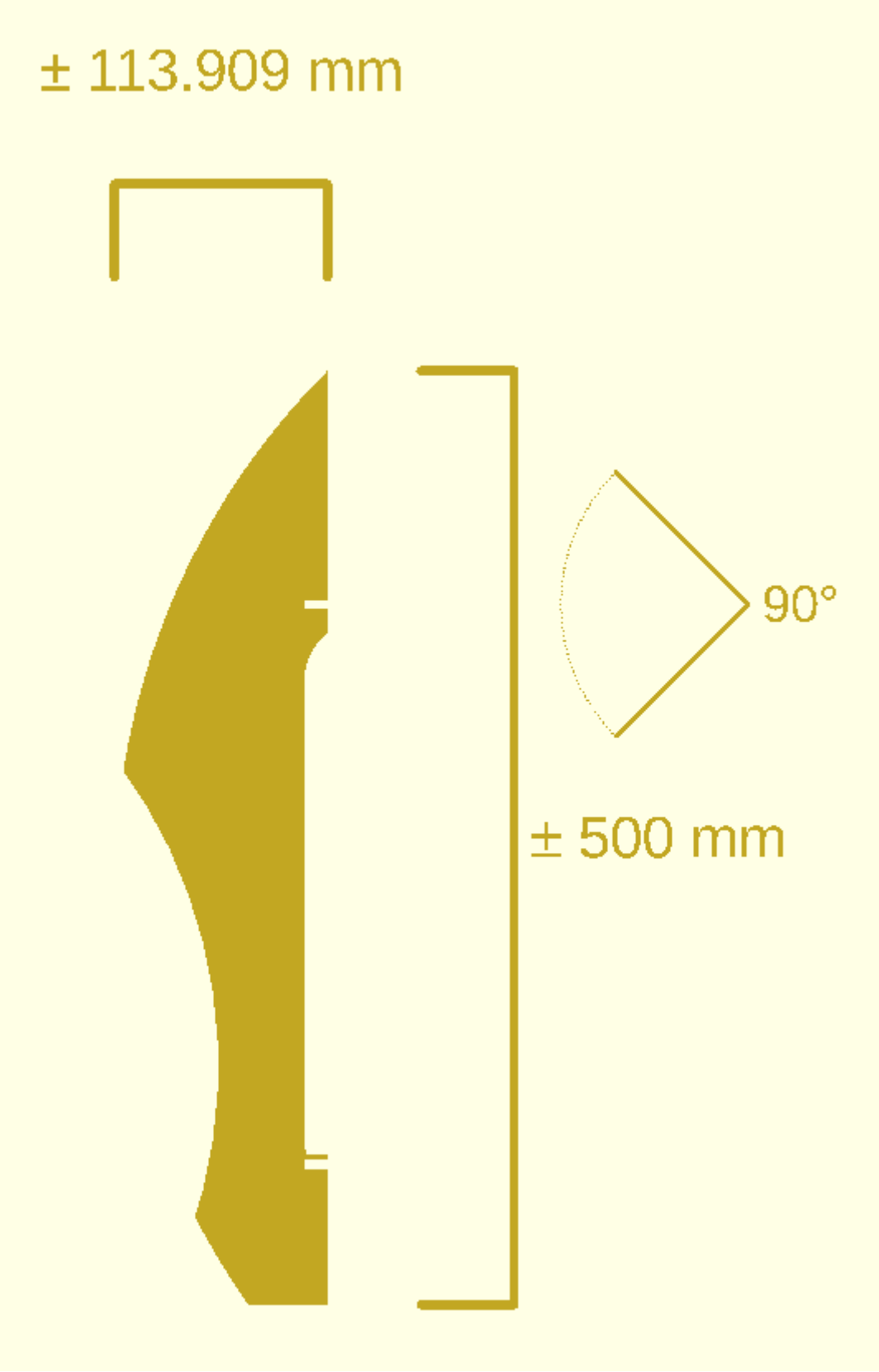
Vous aurez le choix entre :

3D

correspondant à ce que vous aurez vu plus haut.

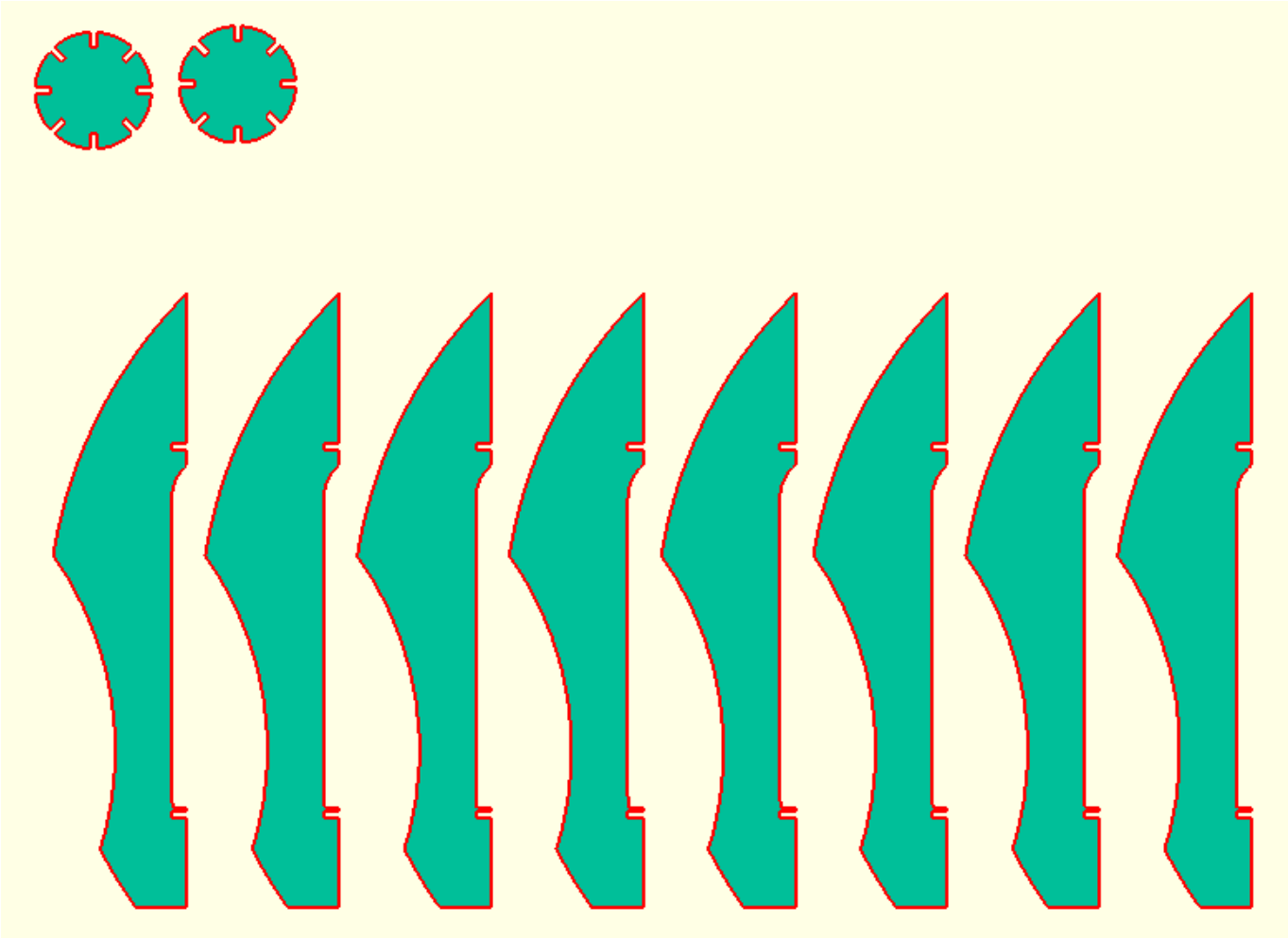
Forme

Affichant alors uniquement une des parts de la lampe :



DXF

Permettant d'exorter le DXF pour la découpe complète.



Forme générale

Répétitions

Nombre de fois que la forme sera répétée autour du luminaire.

Attention à toujours garder un moyen de mettre une ampoule!

Diamètre intérieur

Diamètre des plaques haut et bas servant d'accroches aux formes et où sera mise l'ampoule.

Position de la lampe

Simulera le fait que la lampe soit là où vous l'indiquez : en haut ou en bas.

Matériau

Epaisseur

Indique l'épaisseur du matériau que vous utiliserez.

R, V et B

indique la couleur du matériau dans la prévisualisation 3D.

3D

Informations 3D

Donne les informations de largeur, hauteur et d'angle utilisé.

Support

Affiche le support sur lequel est posé la lampe en prévisualisation 3D.

Transparence des formes

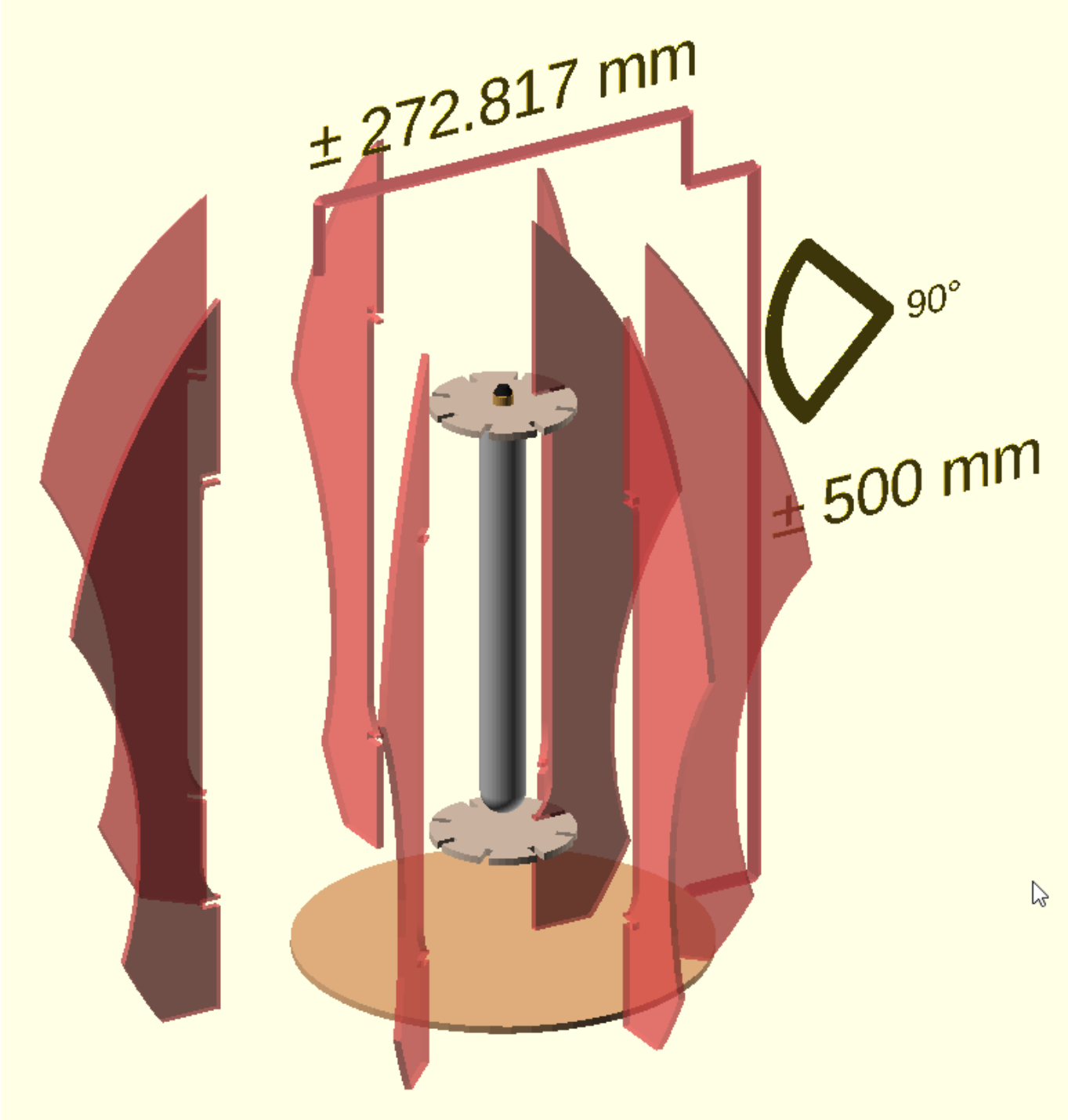
Rend les formes transparentes :

Simulation emboitement

Permet de simuler l'emboitement...

Eloignement

Eloignement que doit avoir la simulation.



Forme principale

Hauteur

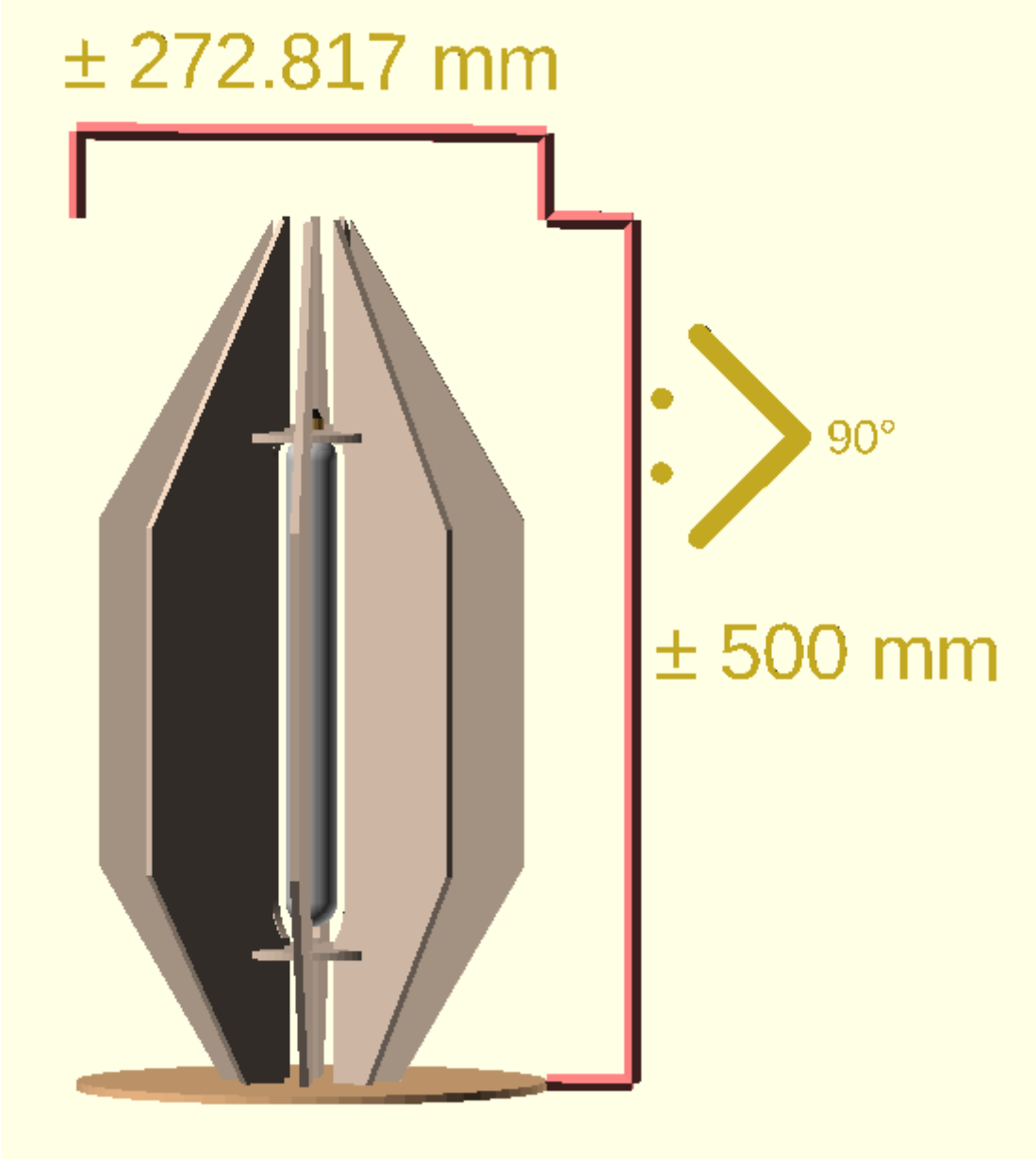
Hauteur de la lampe

Qualite

Indique le nombre de bords qu'aura la forme.

L'information est également donnée là où est indiqué l'angle.

Ici avec une qualité de 3, équivalant donc à 3 côtés :



Angle

Créé la courbure de la forme en fonction de l'angle voulu.

Redimensionnement largeur

Permet d'étirer une forme pour changer le type de courbure.

Forme intérieure 1 & 2

Deux formes intérieures peuvent etre crées pour trouer ou modifier la forme.

Intérieur taille

Indique la taille du trou

Intérieur qualité

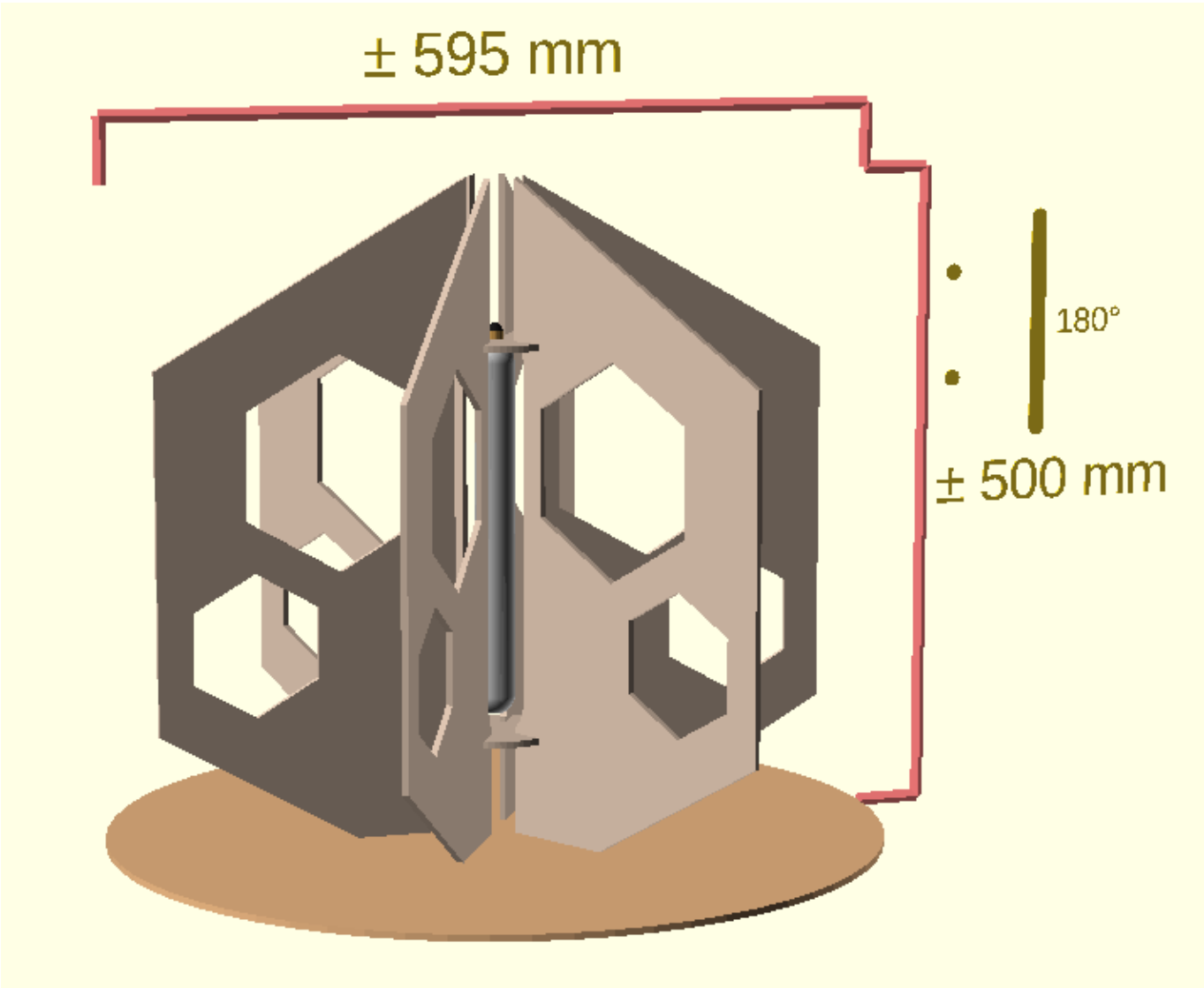
Indique le nombre de bords qu'aura ce trou

Intérieur Déplacement H et V

Déplacement horizontal et vertical

Interieur Custom rotation

Permet de faire pivoter la forme. Utile si elle a peu de bords.



Accroches haut et bas

Hauteur

Hauteur à laquelle se situe l'accorhce

Taille raccord

Taille des encoches

Trou

Créée le trou pour mettre l'ampoule

Trou diamètre

Diamètre du trou pour la lampe...

Pattern

Permet d'afficher un pattern au sein de la forme.

Application pattern

Applique ou non

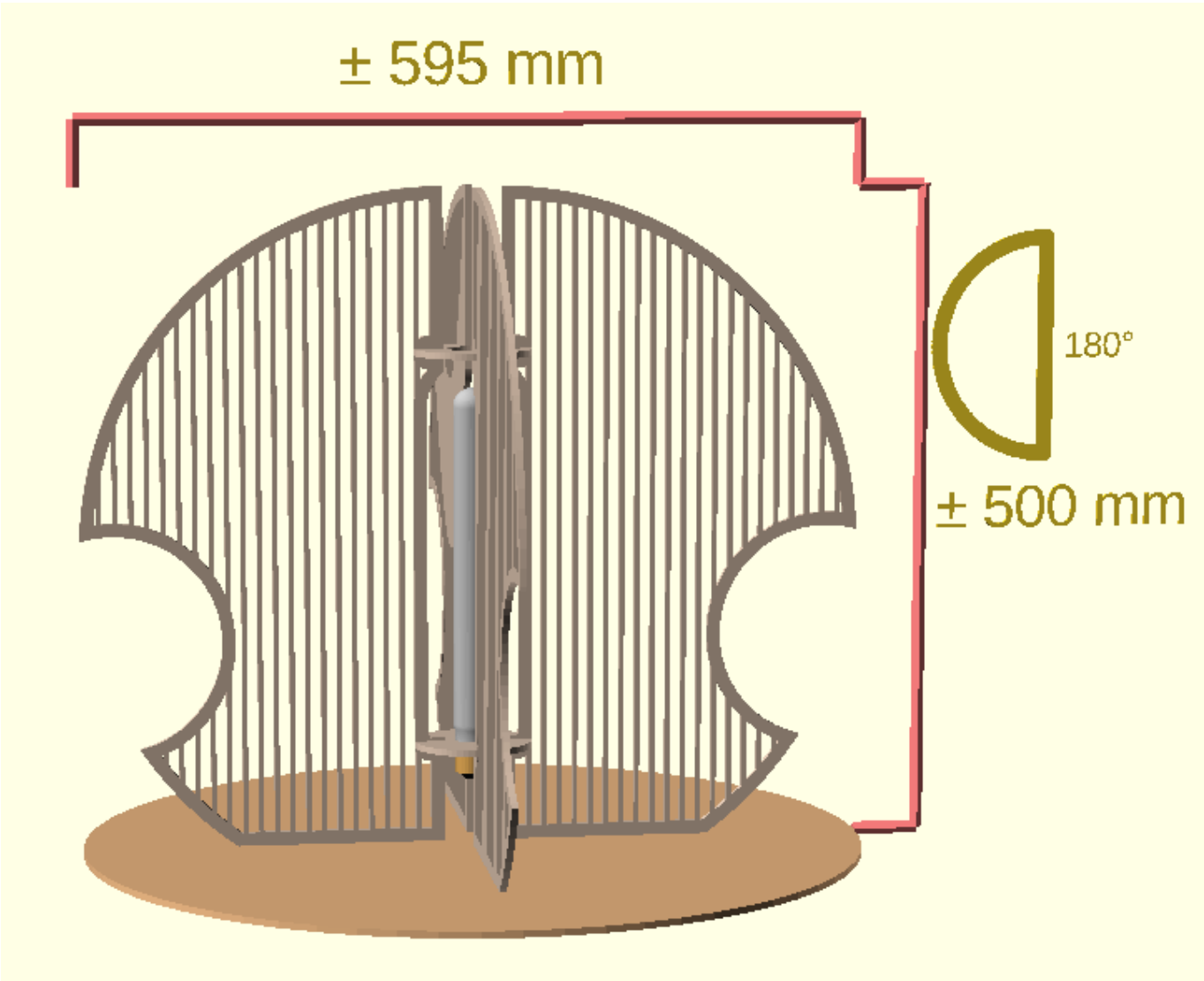
Taille bord

Taille qu'aura le bord

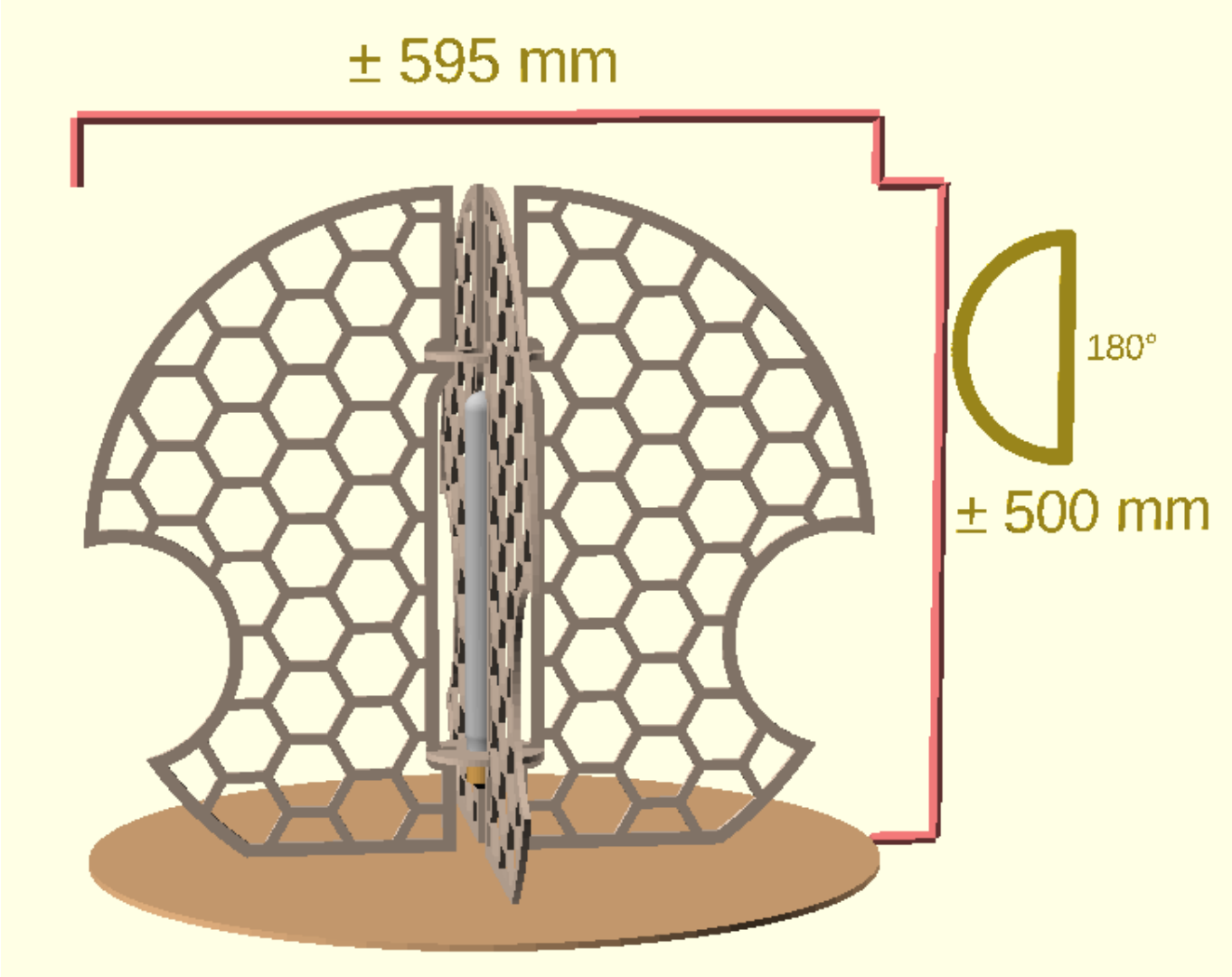
Pattern

Plusieurs patterns peuvent-être utilisés :

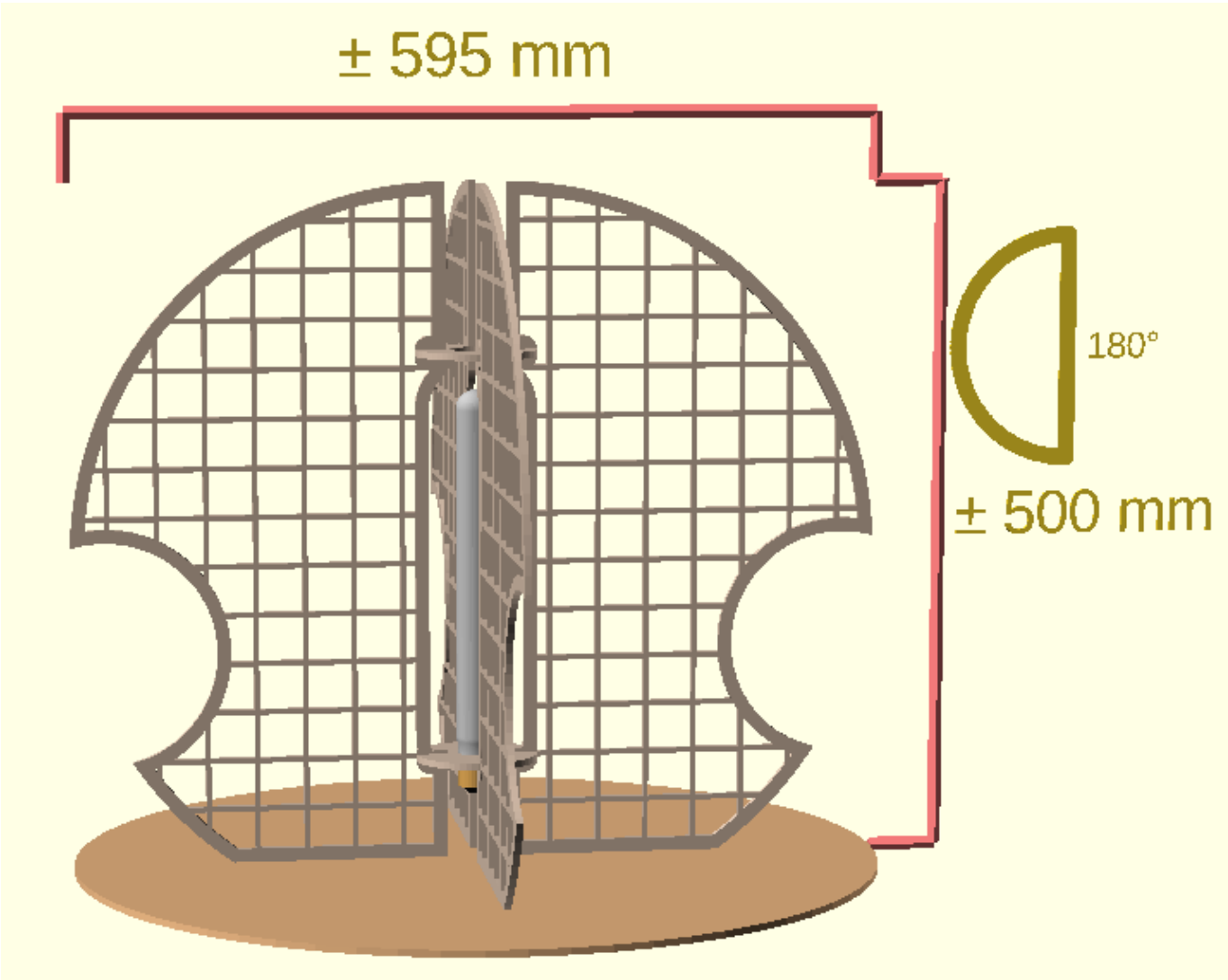
Lignes



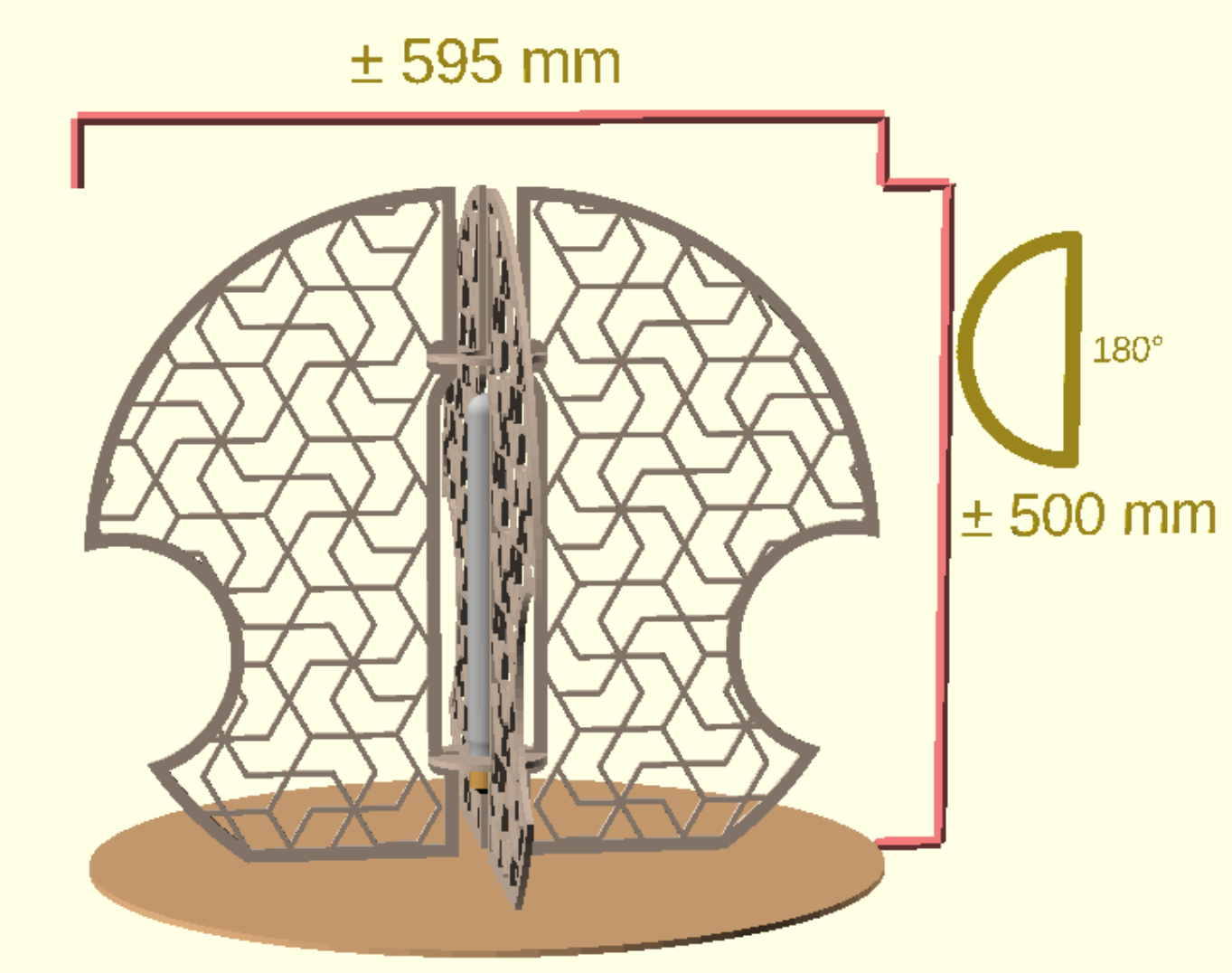
Hexagones



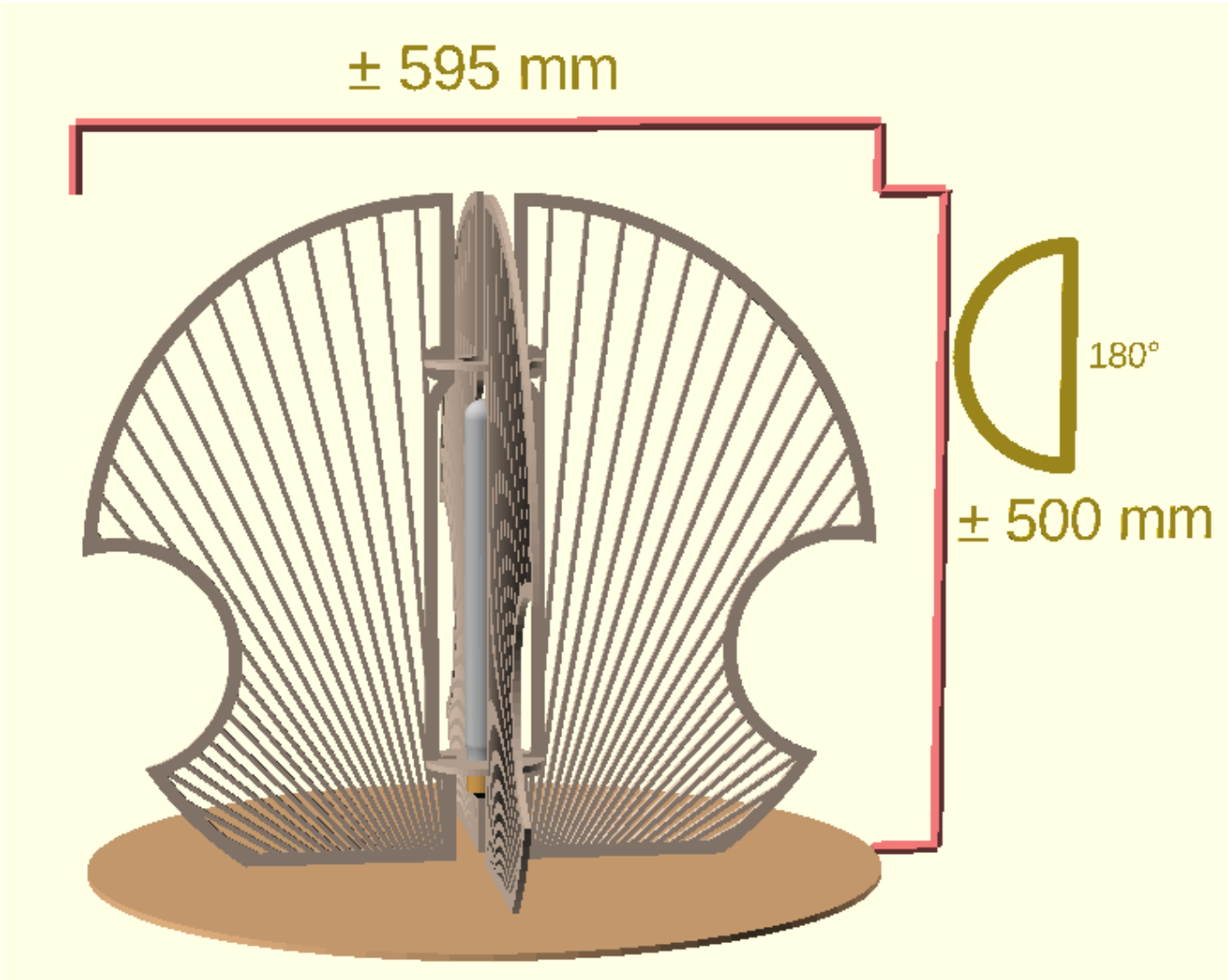
Quadrillage



Hexagones 2



Soleil



DXF / SVG

Utilisera le fichier indiqué comme pattern.

Attention à ne pas utiliser de tros gros SVG!!!!

Pattern Mod 1 à 4

Quatre modificateurs modifiant les patterns.

Chaque pattern étant fort différent l'un de l'autre, les modificateurs n'agissent pas de la même manière en fonction de chaque pattern. A vous d'essayer chacun.

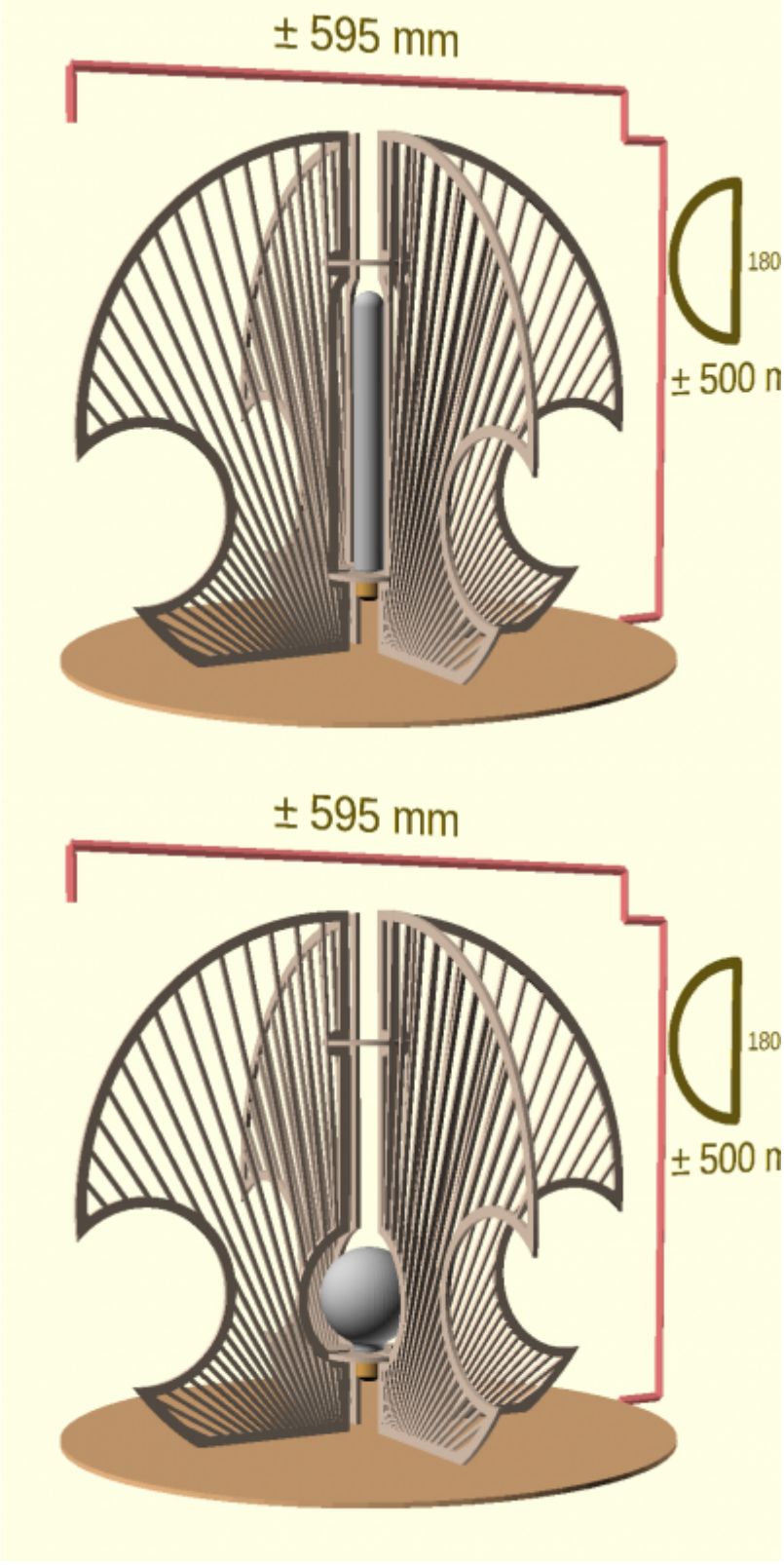
Ampoule

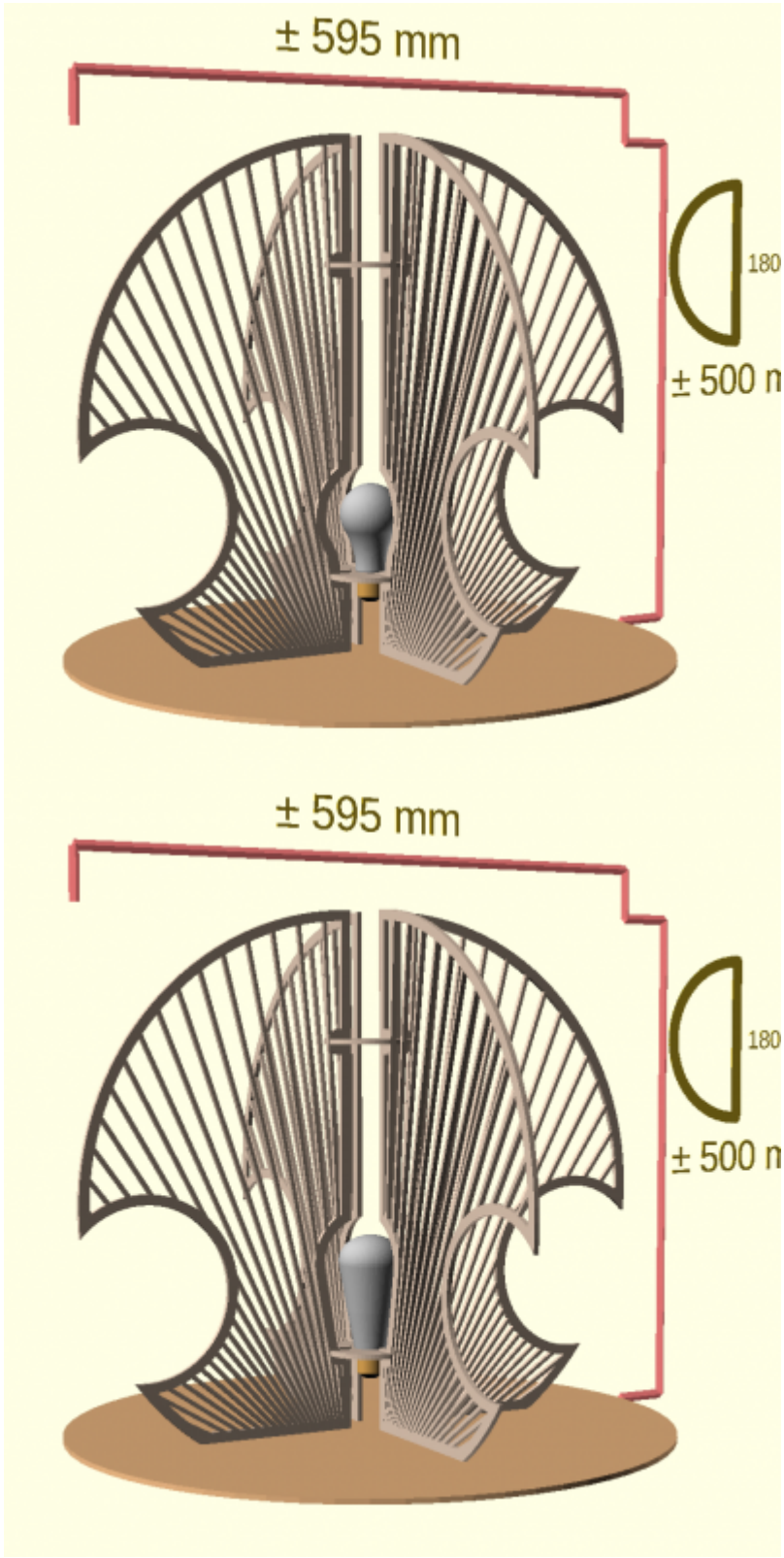
Permet de simuler la présence de l'ampoule et modifier la forme de la lampe en fonction de la lampe voulue.

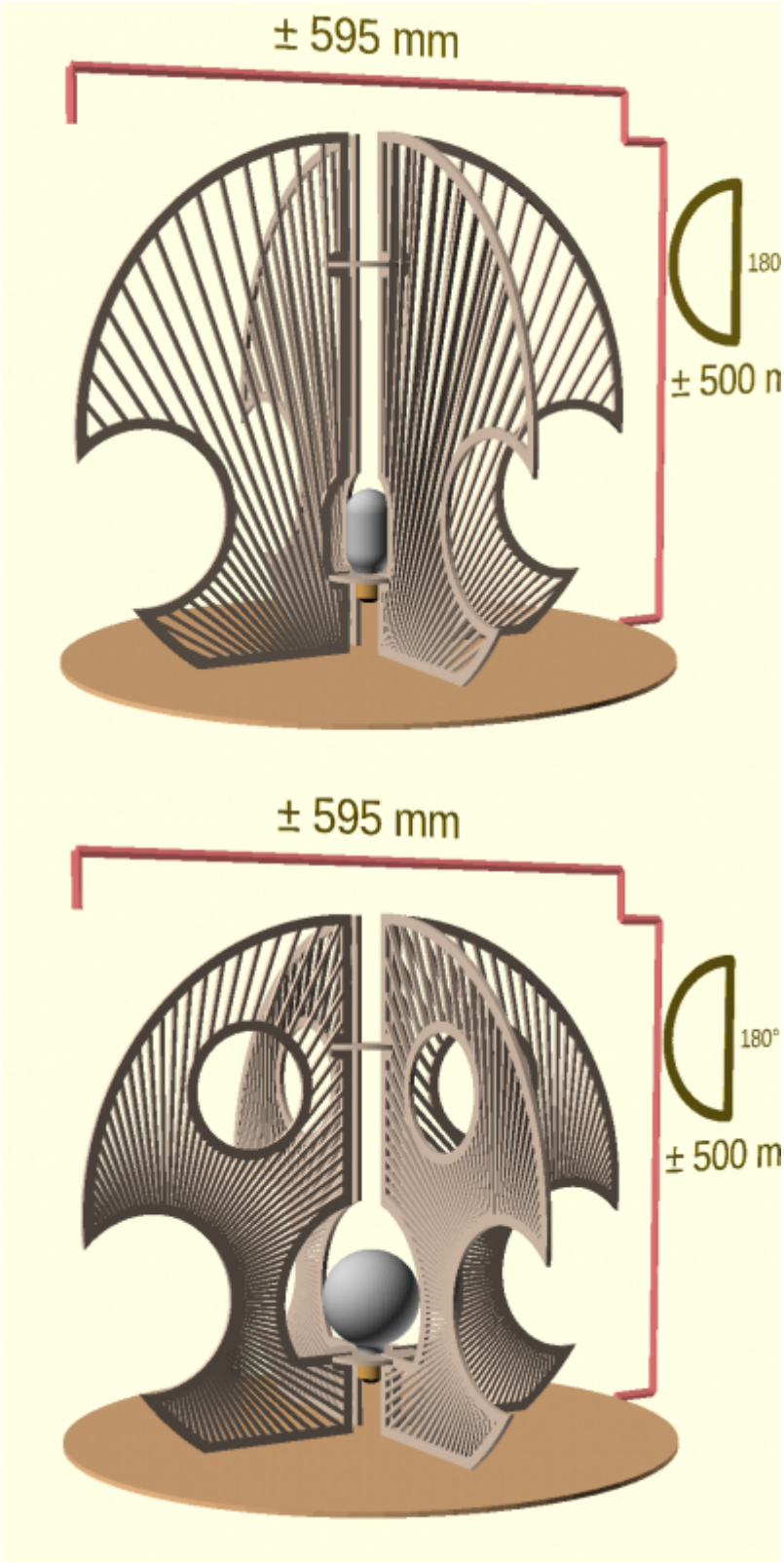
Plusieurs paramètres permettent de créer la forme de la majorité de ampoules habituelles.

Le paramètre “offset” permet d'augmenter ou diminuer l'espace entre l'ampoule et la forme de la lampe.

Il y a déjà plusieurs ampoules incluses :




± 595 mm
180
± 500 m
± 595 mm
180
± 500 m



Bugs connus

1. La hauteur indiquée est celle voulue et non celle réelle (hauteur voulue - découpe pour pied)
2. La largeur indiquée est toujours celle que l'on aurait avec un cercle parfait. La largeur d'une forme ayant peu de faces peut donc moindre que celle indiquée.

generateursupportlampes.scad

```
/*
Générateur de support à lampes.

Vanlindt Marc - marc@vanlindt.be - www.vanlindt.be

Inspiré des réalisations de ... (Merci à qui me donnera le nom)


/!\ A l'heure actuelle aucune découpe et lampe na encore été réalisée à
l'aide
v0.6.5
-------
0.6.5 :
 - offset autour de la lampe
0.6 :
 - Mode Ampoule (beta)
0.5 :
 - correction erreur des Interieur_mode quand "gauche" est utilisé.
0.4 :
 - Ajout de plusieurs "Interieur mode" afin de changer facilement ou de
manière personnalisée la rotation de la forme intérieure.
0.3 :
 - Calcul correct de la hauteur lorsque angles > à 180°
0.2 :
 - ajout variable "Pied" afin d'avoir un pied plat à l'objet.
0.1 :
 - ajout informations en 3D et Forme


*/

/* [Mode de visualisation] */
Visualisation               = "3D";//[3D,Forme,Sortie DXF,Ampoule]

/* [Forme générale] */
// Nombre de formes
Repetitions                 = 8;
// Diamètre au centre de la forme
Diametre_interieur          = 45;
// Coupure pour avoir un pied plat
Pied                        = 50;//[0:200]
// La lampe sera positionnée en...
Position_de_la_lampe        = "bas";//[bas,haut]
/* [Matériau] */
```

```
// En mm
Epaisseur                   = 5;
// Rouge
R                           = 208;//[0:255]
// Vert
V                           = 185;//[0:255]
// Bleu
B                           = 166;//[0:255]

/* [3D] */
// Affichage d'informations à l'écran et dans la console
Informations_3D             = true;
Support                     =true;

// En mode 3D, les formes principales seront transparentes.
Transparence_formes         = false;
// Eloigne certaines formes
Simulation_emboitement      = true;
// Valeur de l'éoiglement (en mm.)
Eloignement                 = 0;//[0:500]

/* [Forme] */
Informations_Forme          = true;
// Taille du trait en mode informations
Informations_trait          = 5;


/* [Forme principale] */
// Hauteur finale en mm
Taille                      = 500;//[1:2000]
// En nombre de côtés
Qualite                     = 64;
// En °
Angle                       = 90;//[1:360]
// Redimensionnement en largeur (en %)
Redimensionnement_largeur   = 100;//[1:500]
/* [Forme intérieure I] */
// En mm
Interieur_1_Taille          = 350;
// En ...
Interieur_1_Qualite         = 64;
// En mm
Interieur_1_Deplacement_H   = -250;//[-2000:2000]
// En mm
Interieur_1_Deplacement_V   = -125;//[-2000:2000]
// Change l'angle de placement de la forme
Interieur_1_mode            =
1;//[1:"Gauche",2:"Haut",3:"Droite",4:"Bas",5:"Custom"]
Interieur_1_Custom_rotation = 0;//[-180:180]

/* [Forme intérieure II] */
```

```
// En mm
Interieur_2_Taille          = 0;
// En ...
Interieur_2_Qualite         = 32;
// En mm
Interieur_2_Deplacement_H   = 0;//[-2000:2000]
// En mm
Interieur_2_Deplacement_V   = 0;//[-2000:2000]
// Change l'angle de placement de la forme
Interieur_2_mode            =
1;//[1:"Gauche",2:"Haut",3:"Droite",4:"Bas",5:"Custom"]
Interieur_2_Custom_rotation = 0;//[-180:180]

/* [Accroche - haut] */
// En % de la hauteur
Hauteur_accroche_haut       = 75;//[0:100]
// Taille de encoches en mm
Taille_raccord_haut         = 50;
// Trou pour lampe ?
Trou_haut                   = false;
// Diametre du trou en mm
Trou_haut_diametre          = 80;

/* [Accroche - bas] */
// En % de la hauteur
Hauteur_accroche_bas        = 15;//[0:100]
// Taille de encoches en mm
Taille_raccord_bas          = 50;
// Trou pour lampe ?
Trou_bas                    = false;
// Diametre du trou en mm
Trou_bas_diametre           = 80;

/* [Pattern] */
Informations_pattern        = true;
Application_pattern         = false;
// Taille du bord pour pattern.
Taille_bord                 = 10;
Pattern                     = 5;//[1:"Lignes",2:"Hexagones (Attention!
Grandes valeurs pour pattern_mod!!!)",3:"Quadrillage",4:"Hexagones II
(Attention! Grandes valeurs pour
pattern_mod!!!)",5:"Soleil!",6:"DXF/SVG - Attention, activer l'option
''SVG'' dans les préférences!"]
Pattern_mod_1               = 128;
Pattern_mod_2               = 4;
Pattern_mod_3               = 0;//[-1000:1000]
Pattern_mod_4               = -71;//[-1000:1000]
Pattern_fichier             = "pattern.svg";

/* [Ampoule] */
```

```
Montrer_ampoule             = false;
Ampoule_Hauteur             = 300;//[1:300]
Ampoule_Diametre_culot      = 27;//[14:"Petit culot - 14mm",27:"Grand
culot - 27mm"]
Ampoule_Evasement           = 50;//[0:5000]
Ampoule_Evasement_Type      = 1;//[1,2]
Ampoule_Diametre_haut       = 30;//[1:200]
Ampoule_ecrasement_haut     = 100;//[1:100]
Ampoule_Diametre_bas        = 30;//[1:60]
Ampoule_ecrasement_bas      = 100;//[1:100]
Ampoule_type                = 1;//[1:"Custom",2:"E27 - Tube long à
filament - 30 cm - 3cm",3:"E27 - Ampoule Edison Frequency à filaments -
13cm - 9.5cm",4:"E27 - Ampoule Edison cage à filaments / Spiral - 12cm
- 8cm",5:"E27 - Edison Guad - 11.5cm - 6cm",6:"E27 - Edison Squirrel -
14cm - 6.4cm",7:"E27 - Ampoule Edison Valve à filaments - 11cm -
4.5cm",8:"E14 - Ampoule Edison Oval à filaments - 10cm - 3.5cm",9:"E27
- Ampoule Edison Candle à filaments - 13cm - 5.3cm",10:"E27 - Ampoule
Edison Globe à filaments - 13cm - 5.3cm",11:"E27 - Osram Vintage 1906 -
14.5cm - 6.4cm"]
// Offset autour de l'ampoule
Offset_ampoule              = 20;//[0:200]

// Variables calculées
haut                        =
Angle>180?sin(90)*Taille:sin(Angle/2)*Taille; // Taille réelle de la
forme obtenue
hautmod                     = Taille/2/haut; // Modificateur permettant
de ramener la forme a la taille voulue
hautmod2                    = ((Taille/2)+Pied/2)/haut;
ma                          = Angle/Qualite; // Angle de chaque
division de la forme.
ep                          = Taille-(cos((Angle)/2)*Taille); //
Epaisseur réelle de la forme
ep2                         =
ep*hautmod2*Redimensionnement_largeur/100; // Epaisseur de la forme une
fois revenue à taille voulue
depl2                       = Interieur_1_Taille/2>ep ?
Interieur_1_Taille/2-ep : 0; // Modificateur de déplacement au cas où
la forme intérieure serait plus grande que la forme extérieure.
points                      = [for(i=[0:Qualite])[sin(-ma*i)*Taille ,
cos(-ma*i)*Taille]]; // Points composant ma forme de base.
cr                          = 1/255*R; // Rouge
cv                          = 1/255*V; // Vert
cb                          = 1/255*B; // Bleu

dechex                      =
[0,1/1,2,3,4,5,6,7,8,9,"A","B","C","D","E","F"];
hexaR                       = str(dechex[floor(R/16)],dechex[R%16]);
hexaV                       = str(dechex[floor(V/16)],dechex[V%16]);
hexaB                       = str(dechex[floor(B/16)],dechex[B%16]);
HexaCouleur                 = str("#",hexaR,hexaV,hexaB);
```

```
HexaRC                      = str("#",hexaR,"0000");
HexaRV                      = str("#00",hexaV,"00");
HexaRB                      = str("#0000",hexaB);

Espacement                  =
sin(360/Repetitions/2)*Diametre_interieur;// Calcul de l'espacement
entre deux formes pour savoir si l'ampoule choisie passe.




if(Visualisation=="Ampoule")
{
  MonAmpoule();
}



if(Visualisation=="3D")
{

  rotate([90,0,-90])
  color([cr,cv,cb,1])
  ObjetComplet();

  if(Support==true)
  {
        color([0.9,0.7,0.5,1])
    translate([0,0,-Epaisseur])
    cylinder(r=ep2+Diametre_interieur/2,h=Epaisseur,$fn=64);

  }


  if(Informations_3D==true)
  {

    color([1,0.5,0.5,1])
    {

      translate([0,ep2+Diametre_interieur/2+50,0])
      cylinder(d=10,h=Taille,$fn=4);

      translate([0,ep2+Diametre_interieur/2+50,0])
      rotate([90,0,0])
      cylinder(d=10,h=50,$fn=4);

      translate([0,ep2+Diametre_interieur/2+50,Taille])
      rotate([90,0,0])
```

```
      cylinder(d=10,h=50,$fn=4);

    }

    rotate([90,0,90])
    translate([ep2+Diametre_interieur+40,Taille/2,0])
    text(str("± ",Taille,"
mm"),valign="center",halign="left",size=Taille/15);


    color([1,0.5,0.5,1])
    {
      translate([0,ep2+Diametre_interieur/2,Taille])
      cylinder(d=10,h=50,$fn=4);

      translate([0,-ep2-Diametre_interieur/2,Taille])
      cylinder(d=10,h=50,$fn=4);

      translate([0,-ep2-Diametre_interieur/2,Taille+50])
      rotate([-90,0,0])
      cylinder(d=10,h=ep2*2+Diametre_interieur,$fn=4);
    }
    translate([0,0,Taille+75])
    rotate([90,0,90])
    text(str("± ",ep2*2+Diametre_interieur,"
mm"),valign="bottom",halign="center",size=Taille/15);
    translate([0,ep2+Diametre_interieur+(300*Taille/1200),Taille/4*3])
    scale([1/1200*Taille,1/1200*Taille,1/1200*Taille])
    rotate([90,0,90])
    translate([-250,-Taille/2,0])
    {
      translate([250,Taille/2])
      text(str("  ",Angle,"°  "),valign="center",halign=Angle>180 ?
"right":"left",size=Taille/25*1200/Taille);
      hull()
      {
        translate([250,Taille/2])
        circle(d=30);
        translate([sin(-90+Angle/2)*200,cos(-90+Angle/2)*200])
        translate([250,Taille/2])
        circle(d=30);
      }
      hull()
      {
        translate([250,Taille/2])
        circle(d=30);
        translate([sin(-90-Angle/2)*200,cos(-90-Angle/2)*200])
        translate([250,Taille/2])
        circle(d=30);
      }
      for(i=[0:Qualite])
```

```
      {
        translate([250,Taille/2])
        rotate([0,0,90-Angle/2])
        translate([sin(-ma*i)*200,cos(-ma*i)*200])
        circle(d=30);
      }
    }
  }

  if(Position_de_la_lampe=="bas" && Montrer_ampoule==true)
  {

    translate([0,0,Taille/100*Hauteur_accroche_bas])
    MonAmpoule();

  }

  if(Position_de_la_lampe=="haut" && Montrer_ampoule==true)
  {

      translate([0,0,Taille/100*Hauteur_accroche_haut])
      rotate([180,0,0])
      MonAmpoule();

  }
}


if(Visualisation=="Sortie DXF")
{
  for(i=[1:Repetitions])
  {
    translate([(ep2+10)*i,0])
    FormePrincipale();
translate([Diametre_interieur/2+Taille_raccord_haut/2,Taille*1.2+Diamet
re_interieur+20])
    AccrocheHaut();
translate([Diametre_interieur*2+Taille_raccord_haut+Taille_raccord_bas/
2,Taille*1.2+Diametre_interieur+Taille_raccord_bas/2])
    AccrocheBas();
  }
}

if(Visualisation=="Forme")
{
  rotate([90,0,90])
  color([cr,cv,cb,1])
  linear_extrude(Epaisseur)
  FormePrincipale();
}
```

```
module ObjetComplet()
{
  Eloignement = Simulation_emboitement==true ? Eloignement : 0;
  for(i=[0:Repetitions-1])
  {
    rotate([0,360/Repetitions*i,0])
    translate([-Diametre_interieur/2-
(Eloignement*sin(180/Repetitions*i)),0])
    if(Transparence_formes==true)
    {
      #MaFormePrincipale3D();
    }
    else
    {
      MaFormePrincipale3D();
    }
  }
  translate([0,Epaisseur/2+Taille/100*Hauteur_accroche_haut,0])
  rotate([90,0,0])
  linear_extrude(Epaisseur)
  AccrocheHaut();

  translate([0,Epaisseur/2+Taille/100*Hauteur_accroche_bas,0])
  rotate([90,0,0])
  linear_extrude(Epaisseur)
  AccrocheBas();
}

module AccrocheHaut()
{
  rotrep = Repetitions%2==1 ? 180/Repetitions : 0;
  rotate([0,0,rotrep])
  difference()
  {
    circle(d=Application_pattern==false ?
Diametre_interieur+Taille_raccord_haut:Diametre_interieur+Taille_raccor
d_haut/2+Taille_bord*2
    ,$fn=64);

    for(i=[1:Repetitions])
    {
      rotate([0,0,360/Repetitions*i])
translate([Diametre_interieur/2+Taille_raccord_haut/2+Taille_raccord_ha
ut/4,0,0])

      square([Taille_raccord_haut,Epaisseur],center=true);
    }
    if(Trou_haut==true)
    {
      circle(d=Trou_haut_diametre,$fn=64);
```

```
    }
  }
}

module AccrocheBas()
{
  rotrep = Repetitions%2==1 ? 180/Repetitions : 0;
  rotate([0,0,rotrep])
  difference()
  {
    circle(d=Application_pattern==false ?
Diametre_interieur+Taille_raccord_bas:Diametre_interieur+Taille_raccord
_bas/2+Taille_bord*2,$fn=64);
    for(i=[1:Repetitions])
    {
      rotate([0,0,360/Repetitions*i])
translate([Diametre_interieur/2+Taille_raccord_bas/2+Taille_raccord_bas
/4,0,0])

      square([Taille_raccord_bas,Epaisseur],center=true);
    }
    if(Trou_bas==true)
    {
      circle(d=Trou_bas_diametre,$fn=64);
    }
  }
}



module MaFormePrincipale3D()
{
  translate([0,0,-Epaisseur/2])
  linear_extrude(Epaisseur)
  FormePrincipale();

}


module FormePrincipale()
{

  if(Application_pattern==true)
  {

    difference()
    {
      FormeDeBase();
      offset(-Taille_bord)
      FormeDeBase();
```

```
    }
  }

  else
  {
      FormeDeBase();


  }
  if(Application_pattern==true)
  {
    intersection()
    {
      FormeDeBase();
      if(Pattern==1)Pattern1();
      if(Pattern==2)Pattern2();
      if(Pattern==3)Pattern3();
      if(Pattern==4)Pattern4();
      if(Pattern==5)Pattern5();
      if(Pattern==6)Pattern6();
    }
  }
}

module FormeDeBase()
{
  rotint = Interieur_1_mode==3 ? (180/Interieur_1_Qualite):
  Interieur_1_mode==2 ? 90+(180/Interieur_1_Qualite):
  Interieur_1_mode==4 ? -90+(180/Interieur_1_Qualite):
  Interieur_1_mode==5 ?
Interieur_1_Custom_rotation:(180/Interieur_1_Qualite)+180;

  rotint2 = Interieur_2_mode==3 ? (180/Interieur_2_Qualite):
  Interieur_2_mode==2 ? 90+(180/Interieur_2_Qualite):
  Interieur_2_mode==4 ? -90+(180/Interieur_2_Qualite):
  Interieur_2_mode==5 ?
Interieur_2_Custom_rotation:(180/Interieur_2_Qualite)+180;

  difference()
  {

    translate([0,-Pied])
    scale([hautmod2*Redimensionnement_largeur/100,hautmod2,0])
    translate([-ep,haut])

    difference()
    {

      translate([Taille,0])
      rotate([0,0,90-Angle/2])
```

```
      polygon(points);

      translate([(-
depl2+Interieur_1_Deplacement_H)*1/Redimensionnement_largeur*100,Interi
eur_1_Deplacement_V])
      scale([1/Redimensionnement_largeur*100,1,1])
      rotate([0,0,rotint])
      circle(r=Interieur_1_Taille,$fn=Interieur_1_Qualite);

      translate([(-
depl2+Interieur_2_Deplacement_H)*1/Redimensionnement_largeur*100,Interi
eur_2_Deplacement_V])
      scale([1/Redimensionnement_largeur*100,1,1])
      rotate([0,0,rotint2])
      circle(r=Interieur_2_Taille,$fn=Interieur_2_Qualite);









    }

if(Position_de_la_lampe=="bas")
      {
translate([Diametre_interieur/2,Taille/100*Hauteur_accroche_bas])
        difference()
        {
          offset(Offset_ampoule)
          AmpouleForme();
          translate([0,-5000+Epaisseur])
          square([10000,10000],center=true);
          translate([0,Taille/100*Hauteur_accroche_haut-
Taille/100*Hauteur_accroche_bas+5000-Epaisseur])
          square([10000,10000],center=true);

        }
      }


      else
      {
        difference()
        {
translate([Diametre_interieur/2,Taille/100*Hauteur_accroche_haut])
          rotate([0,0,180])
          difference()
          {
            offset(Offset_ampoule)
            AmpouleForme();
```

```
            translate([0,-5000+5])
            square([10000,10000],center=true);
 translate([0,Taille/100*Hauteur_accroche_haut-
Taille/100*Hauteur_accroche_bas+5000-Epaisseur])
          square([10000,10000],center=true);
          }
        }
      }


    translate([0,-Pied])
    square([ep2*4,Pied*2],center=true);

    translate([Taille_raccord_haut/4,Taille/100*Hauteur_accroche_haut])
    square([Taille_raccord_haut,Epaisseur],center=true);

    translate([Taille_raccord_bas/4,Taille/100*Hauteur_accroche_bas])
    square([Taille_raccord_bas,Epaisseur],center=true);

  }



  if(Visualisation=="Forme" && Informations_Forme==true)
  {
    translate([100+250*Taille/1000,Taille/4*3,0])
    scale([Taille/1000,Taille/1000,Taille/1000])
    translate([0,0])
    {
      //translate([250,Taille/2])
      text(str(" ",Angle,"° "),valign="center",halign=Angle>180 ?
"right":"left",size=40);
      hull()
      {
        //translate([250,Taille/2])
        circle(d=Informations_trait);
        translate([sin(-90+Angle/2)*200,cos(-90+Angle/2)*200])
        //translate([250,Taille/2])
        circle(d=Informations_trait);
      }
      hull()
      {
        //translate([250,Taille/2])
        circle(d=Informations_trait);
        translate([sin(-90-Angle/2)*200,cos(-90-Angle/2)*200])
        //translate([250,Taille/2])
        circle(d=Informations_trait);
      }
      for(i=[0:Qualite])
      {
        //translate([250,Taille/2])
```

```
      rotate([0,0,90-Angle/2])
      translate([sin(-ma*i)*200,cos(-ma*i)*200])
      circle(d=Informations_trait/2);
    }
  }
  hull()
  {
    translate([50,0,0])
    circle(d=Informations_trait);

    translate([100,0,0])
    circle(d=Informations_trait);
  }

  hull()
  {
    translate([50,Taille,0])
    circle(d=Informations_trait);

    translate([100,Taille,0])
    circle(d=Informations_trait);
  }

  hull()
  {
    translate([100,0,0])
    circle(d=Informations_trait);

    translate([100,Taille,0])
    circle(d=Informations_trait);
  }



  hull()
  {
    translate([0,Taille+50,0])
    circle(d=Informations_trait);
    translate([0,Taille+100,0])
    circle(d=Informations_trait);
  }

  hull()
  {
    translate([-ep2,Taille+50,0])
    circle(d=Informations_trait);
    translate([-ep2,Taille+100,0])
    circle(d=Informations_trait);
  }
```

```
    hull()
    {
      translate([-ep2,Taille+100,0])
      circle(d=Informations_trait);
      translate([0,Taille+100,0])
      circle(d=Informations_trait);
    }

    translate([0100,Taille/2,0])
    text(str(" ± ",Taille,"
mm"),valign="center",halign="left",size=Taille/22);

    translate([-ep2/2,Taille+150,0])
    text(str("± ",ep2,"
mm"),valign="bottom",halign="center",size=Taille/22);


  }
}
module Pattern1()
{
  rotate([0,0,Pattern_mod_3])
  for(i=[-Taille:Pattern_mod_1+Pattern_mod_2:Taille])
  {
    hull()
    {
    translate([i,-Taille])
    circle(d=Pattern_mod_2);
    translate([i,Taille*2])
    circle(d=Pattern_mod_2);
    }
  }
}

module Pattern2()
{
  hexa=[for(i=[0:5])[sin(60*i)*Pattern_mod_1 ,
cos(60*i)*Pattern_mod_1]];
  difference()
  {
    translate([-Taille*2,-Taille*2])
    square([Taille*4,Taille*4]);
    translate([Pattern_mod_1*2,0])
    for(j=[-(Taille/2/Pattern_mod_1)/2:(Taille/2/Pattern_mod_1)*2])
    {
      translate([0,j*cos(30)*Pattern_mod_1*2])
      for(i=[-Taille/2/Pattern_mod_1:0])
      {
        translate([Pattern_mod_1*i*3,0])
        rotate([0,0,30])
        offset(-Pattern_mod_2)
```

```
        polygon(hexa);
        translate([Pattern_mod_1*i*3+sin(30)*Pattern_mod_1*3,cos(30)*
Pattern_mod_1])
        rotate([0,0,30])
        offset(-Pattern_mod_2)
        polygon(hexa);
      }
    }
  }
}

module Pattern3()
{
  rotate([0,0,Pattern_mod_3])
  {
    for(i=[-Taille:Pattern_mod_1+Pattern_mod_2:Taille])
    {
      hull()
      {
      translate([i,-Taille])
      circle(d=Pattern_mod_2);
      translate([i,Taille*2])
      circle(d=Pattern_mod_2);
      }
    }
    for(i=[-Taille:Pattern_mod_1+Pattern_mod_2:Taille])
    {
      hull()
      {
      translate([-Taille,i])
      circle(d=Pattern_mod_2);
      translate([Taille*2,i])
      circle(d=Pattern_mod_2);
      }
    }
  }
}


module Pattern4()
{
  module p4fb(a)
  {
polygon([[0,0],[sin(120)*a,cos(120)*a],[sin(120)*a+sin(60)*a,cos(120)*a
+cos(60)*a],[sin(120)*a+sin(60)*a,cos(120)*a+cos(60)*a+a],[sin(60)*a,co
s(60)*a],[0,a]]);
  }

  module p4fb2(a,b)
  {
```

```
    rotate([0,0,30])
    for(i=[0:6])
    {
      rotate([0,0,-60*i])
      translate([0,-a])
      difference()
      {
        offset(b/2)
        p4fb(a);
        offset(-b/2)
        p4fb(a);
      }
    }
  }
  module p4fb3()
  {
    p4fb2(Pattern_mod_1,Pattern_mod_2);
    translate([sin(30)*Pattern_mod_1*6,cos(30)*Pattern_mod_1*2])
    p4fb2(Pattern_mod_1,Pattern_mod_2);
  }
  module p4fb4()
  {
    for(i=[-Taille/4/Pattern_mod_1:0])
    {
      translate([sin(30)*Pattern_mod_1*12*i,0])
      p4fb3();
    }
  }

  for(i=[-Pattern_mod_1*4:sin(60)*Pattern_mod_1*4:Taille*1.5])
  {
    translate([0,i])
    p4fb4();
  }
}

module Pattern5()
{
  for(i=[1:Pattern_mod_1])
  {
    translate([Pattern_mod_3,Pattern_mod_4])
    hull()
    {
      circle(d=Pattern_mod_2);
translate([sin(360/Pattern_mod_1*i)*Taille*4,cos(360/Pattern_mod_1*i)*T
aille*4])
      circle(d=Pattern_mod_2);
    }
  }
}
```

```
module Pattern6()
{
  translate([Pattern_mod_3,Pattern_mod_4,0])
  offset(Pattern_mod_2)
  scale([Pattern_mod_1/100,Pattern_mod_1/100,Pattern_mod_1/100])
  import(Pattern_fichier);
}


module MonAmpoule()
{

if(Ampoule_type==1){Ampoule(h=Ampoule_Hauteur,d1=Ampoule_Diametre_culot
,d2=Ampoule_Diametre_haut,d3=Ampoule_Diametre_bas,e=Ampoule_Evasement,d
e1=Ampoule_ecrasement_haut,de2=Ampoule_ecrasement_bas,et=Ampoule_Evasem
ent_Type,o=0);}

  if(Ampoule_type==2) {Ampoule(h=300 ,d1=27 ,d2=30 ,d3=30 ,e=0
,de1=100 ,de2=25  ,et=1,o=0);}
  if(Ampoule_type==3) {Ampoule(h=130 ,d1=27 ,d2=95 ,d3=30 ,e=10
,de1=100 ,de2=25  ,et=1,o=0);}
  if(Ampoule_type==4) {Ampoule(h=120 ,d1=27 ,d2=80 ,d3=30 ,e=10
,de1=100 ,de2=25  ,et=1,o=0);}
  if(Ampoule_type==5) {Ampoule(h=115 ,d1=27 ,d2=60 ,d3=30 ,e=70
,de1=100 ,de2=25  ,et=1,o=0);}
  if(Ampoule_type==6) {Ampoule(h=140 ,d1=27 ,d2=64 ,d3=33 ,e=5000
,de1=80  ,de2=10  ,et=1,o=0);}
  if(Ampoule_type==7) {Ampoule(h=110 ,d1=27 ,d2=45 ,d3=45 ,e=70
,de1=100 ,de2=100 ,et=1,o=0);}
  if(Ampoule_type==8) {Ampoule(h=100 ,d1=14 ,d2=35 ,d3=15 ,e=25
,de1=0   ,de2=30  ,et=2,o=0);}
  if(Ampoule_type==9) {Ampoule(h=130 ,d1=27 ,d2=53 ,d3=28 ,e=100
,de1=0   ,de2=0   ,et=2,o=0);}
  if(Ampoule_type==10){Ampoule(h=130 ,d1=27 ,d2=95 ,d3=27 ,e=10
,de1=100 ,de2=100 ,et=1,o=0);}
  if(Ampoule_type==11){Ampoule(h=145 ,d1=27 ,d2=64 ,d3=30 ,e=5000
,de1=80  ,de2=25  ,et=1,o=0);}
  }


module AmpouleForme()
{

if(Ampoule_type==1){Ampoule2D(h=Ampoule_Hauteur,d1=Ampoule_Diametre_cul
ot,d2=Ampoule_Diametre_haut,d3=Ampoule_Diametre_bas,e=Ampoule_Evasement
,de1=Ampoule_ecrasement_haut,de2=Ampoule_ecrasement_bas,et=Ampoule_Evas
ement_Type,o=Offset_ampoule);}

  if(Ampoule_type==2) {Ampoule2D(h=300 ,d1=27 ,d2=30 ,d3=30 ,e=0
,de1=100 ,de2=25  ,et=1,o=0);}
```

```
  if(Ampoule_type==3) {Ampoule2D(h=130 ,d1=27 ,d2=95 ,d3=30 ,e=10
,de1=100 ,de2=25  ,et=1,o=0);}
  if(Ampoule_type==4) {Ampoule2D(h=120 ,d1=27 ,d2=80 ,d3=30 ,e=10
,de1=100 ,de2=25  ,et=1,o=0);}
  if(Ampoule_type==5) {Ampoule2D(h=115 ,d1=27 ,d2=60 ,d3=30 ,e=70
,de1=100 ,de2=25  ,et=1,o=0);}
  if(Ampoule_type==6) {Ampoule2D(h=140 ,d1=27 ,d2=64 ,d3=33 ,e=5000
,de1=80  ,de2=10  ,et=1,o=0);}
  if(Ampoule_type==7) {Ampoule2D(h=110 ,d1=27 ,d2=45 ,d3=45 ,e=70
,de1=100 ,de2=100 ,et=1,o=0);}
  if(Ampoule_type==8) {Ampoule2D(h=100 ,d1=14 ,d2=35 ,d3=15 ,e=25
,de1=0   ,de2=30  ,et=2,o=0);}
  if(Ampoule_type==9) {Ampoule2D(h=130 ,d1=27 ,d2=53 ,d3=28 ,e=100
,de1=0   ,de2=0   ,et=2,o=0);}
  if(Ampoule_type==10){Ampoule2D(h=130 ,d1=27 ,d2=95 ,d3=27 ,e=10
,de1=100 ,de2=100 ,et=1,o=0);}
  if(Ampoule_type==11){Ampoule2D(h=145 ,d1=27 ,d2=64 ,d3=30 ,e=5000
,de1=80  ,de2=25  ,et=1,o=0);}
  }


echoPrez=str("<center><h1>Générateur de supports à
lampes</h1></center><b>CC-BY-SA <a href
=mailto:marc@vanlindt.be>marc@vanlindt.be</a> -
www.vanlindt.be</b><br/>Inspiré d'une création de ... présentée à la
Braderie de l'Art de Liège 2018<hr>");
echoInfos=str("
<center><h2>Informations sur l'objet</h2></center>
<table>
<tr>
<td align=right width=200><b>Hauteur de l'objet : </b></td>
<td>",Taille," mm.</td>
</tr>
<tr>
<td align=right><b>Largeur de l'objet : </b></td>
<td>",ep2*2+Diametre_interieur," mm.</td>
</tr>
<tr>
<td align=right><b>Largeur d'une forme : </b></td>
<td>",ep2," mm.</td>
</tr>
</table>
<hr>
<table>
<tr>
<td align=right width=200><b>Nombre de formes : </b></td>
<td>",Repetitions,"</td>
</tr>
</table>
<hr>
<table>
```

```
<tr>
<td align=right width=200><b>Diametre intérieur : </b></td>
<td>",Diametre_interieur," mm.</td>
</tr>
</table>
<hr>
<table>
<tr>
<td align=right width=200><b>Epaisseur matériau choisi : </b></td>
<td>",Epaisseur," mm.</td>
</tr>
</table>
<hr>
<table>
<tr>
<td align=right width=200><b>Angle utilisé : </b></td>
<td>",Angle,"°</td>
</tr>
</table>
<hr>
<table>
<tr>
<td align=right width=200><b>Hauteur raccord haut : </b></td>
<td>",Taille/100*Hauteur_accroche_haut," mm.</td>
</tr>
<tr>
<td align=right width=200><b>Hauteur raccord bas : </b></td>
<td>",Taille/100*Hauteur_accroche_bas," mm.</td>
</tr>
</table>
<hr>"
);

toto="Edison-Tube-30-3.jpg";
echo3Dt=Transparence_formes==true ? "Activée!" : "Désactivée!";
echo3D=str("<center><h2>Vue 3D</h2></center>
<small><b><i>Attention, les formes 3D générées sont des visualisations
de ce que donnera l'objet physique final, après découpe des plans
2D.<br/>Si vous tentez de créer un STL à partir de cette vue, il y aura
des erreurs dues à la juxtaposition de certains vecteurs (non-manifold
object).</i></b></small>
<br/><br/>
<b>Couleur de l'objet :</b>
<table>
  <tr>
  <td width=50 bgcolor=",HexaCouleur," rowspan=3></td>
  <td align=center width=20 bgcolor=",HexaRC,"><b></b></td>
  <td align=left>",R,"</td>
  </tr>
  <tr>
```

```
  <td align=center width=20 bgcolor=",HexaRV,"><b></b></td>
  <td align=left>",V,"</td>
  </tr>
  <tr>
  <td align=center width=20 bgcolor=",HexaRB,"><b></b></td>
  <td align=left>",B,"</td>
</table>




<b>Transparence de l'objet : </b>",echo3Dt,"
<hr>");

echoPattern=str("<center><h2>Aide pour pattern utilisé</h2></center>");


InfoPattern=["
<h3>Pattern 1 : lignes</h3>
• <b>Pattern mod 1 : </b>Espacement entre les lignes<br/>
• <b>Pattern mod 2 : </b>Epaisseur des lignes<br/>
• <b>Pattern mod 3 : </b>Rotation des lignes",
"<h3>Pattern 2 : Hexagones I</h3>
• <b>Pattern mod 1 : </b>Taille des hexagones<br/>
• <b>Pattern mod 2 : </b>Epaisseur des traits",
"<h3>Pattern 3 : Quadrillage</h3>
• <b>Pattern mod 1 : </b>Espacement entre les lignes<br/>
• <b>Pattern mod 2 : </b>Epaisseur des lignes
• <b>Pattern mod 3 : </b>Rotation du quadrillage",
"<h3>Pattern 4 : Hexagones II</h3>
• <b>Pattern mod 1 : </b>Taille des hexagones<br/>
• <b>Pattern mod 2 : </b>Epaisseur des traits",
"<h3>Pattern 5 : Soleil</h3>
• <b>Pattern mod 1 : </b>Nombre de rayons<br/>
• <b>Pattern mod 2 : </b>Epaisseur des rayons<br/>
• <b>Pattern mod 3 : </b>Déplacement horizonal<br/>
• <b>Pattern mod 4 : </b>Déplacement vertical"];

PatternInfo=InfoPattern[Pattern-1];

echoTout=str(echoPrez,echoInfos,echo3D,echoPattern,PatternInfo,"<hr>");

if(Informations_pattern==true)
{
echo(echoTout);
}

module Ampoule(h,d1,d2,d3,e,de1,de2,et,o)
{
  translate([0,0,-d1])
  {
```

```
color([0.8,0.6,0.3,1])
translate([0,0,5])
cylinder(d=d1,h=d1-5);

color([0.1,0.1,0.1,1])
cylinder(d1=d1*0.5,d2=d1*0.9,h=5);
color([0.7,0.7,0.7,1])

rotate_extrude($fn=64)
difference()
{
  offset(-e,$fn=256)
  offset(e,$fn=256)
  if(et==1)
  {
    union()
    {
      translate([0,h-((d2/2)*1/100*de1)])
      scale([1,1/100*de1])
      circle(d=d2,$fn=128);
      translate([0,d1+((d3/2)*1/100*de2)])
      scale([1,1/100*de2])
      circle(d=d3,$fn=128);
    }
    offset(-50,$fn=128)
    offset(50,$fn=128)
    {
    if(et==1)
    {
      translate([0,d1+((d3/2)*1/100*de2)])
      square([d3/2,(h-d1-+((d3/2)*1/100*de2)-((d2/2)*1/100*de1))]);
    }
    translate([0,d1-0.1])
    square([d1/2,0.1]);
  }
}
  if(et==2)
  {
    offset(-15)
    offset(15)
    {
      translate([0,h/2+d1/2+e/200])
      resize([d2,h-d1-e/100])
      circle(d=h-d1-e/100,$fn=128);

      offset(d3/10)
      offset(-d3/10)
      translate([0,d1])
      square([d3/2,d3]);
```

```
        }
      }
      translate([-d2*100,-(h+d2)*50])
      square([d2*100,(h+d2)*100]);
    }

  }
}







module Ampoule2D(h,d1,d2,d3,e,de1,de2,et,o)
{
  translate([0,-d1])
  {

    difference()
    {
      offset(-e,$fn=256)
      offset(e,$fn=256)
      if(et==1)
      {
        union()
        {
          translate([0,h-((d2/2)*1/100*de1)])
          scale([1,1/100*de1])
          circle(d=d2,$fn=128);
          translate([0,d1+((d3/2)*1/100*de2)])
          scale([1,1/100*de2])
          circle(d=d3,$fn=128);
        }
        offset(-50,$fn=128)
        offset(50,$fn=128)
        {
        if(et==1)
        {
          translate([0,d1+((d3/2)*1/100*de2)])
          square([d3/2,(h-d1-+((d3/2)*1/100*de2)-((d2/2)*1/100*de1))]);
        }
        translate([0,d1-0.1])
        square([d1/2,0.1]);
      }
    }
      if(et==2)
      {
        offset(-15)
        offset(15)
        {
```

```
        translate([0,h/2+d1/2+e/200])
        resize([d2,h-d1-e/100])
        circle(d=h-d1-e/100,$fn=128);

        offset(d3/10)
        offset(-d3/10)
        translate([0,d1])
        square([d3/2,d3]);
      }
    }
    translate([-d2*100,-(h+d2)*50])
    square([d2*100,(h+d2)*100]);
  }


  mirror()
  difference()
  {
    offset(-e,$fn=256)
    offset(e,$fn=256)
    if(et==1)
    {
      union()
      {
        translate([0,h-((d2/2)*1/100*de1)])
        scale([1,1/100*de1])
        circle(d=d2,$fn=128);
        translate([0,d1+((d3/2)*1/100*de2)])
        scale([1,1/100*de2])
        circle(d=d3,$fn=128);
      }
      offset(-50,$fn=128)
      offset(50,$fn=128)
      {
      if(et==1)
      {
        translate([0,d1+((d3/2)*1/100*de2)])
        square([d3/2,(h-d1-+((d3/2)*1/100*de2)-((d2/2)*1/100*de1))]);
      }
      translate([0,d1-0.1])
      square([d1/2,0.1]);
    }
  }
    if(et==2)
    {
      offset(-15)
      offset(15)
      {
        translate([0,h/2+d1/2+e/200])
        resize([d2,h-d1-e/100])
```

```
            circle(d=h-d1-e/100,$fn=128);

            offset(d3/10)
            offset(-d3/10)
            translate([0,d1])
            square([d3/2,d3]);
          }
        }
        translate([-d2*100,-(h+d2)*50])
        square([d2*100,(h+d2)*100]);
    }
    }
}
```