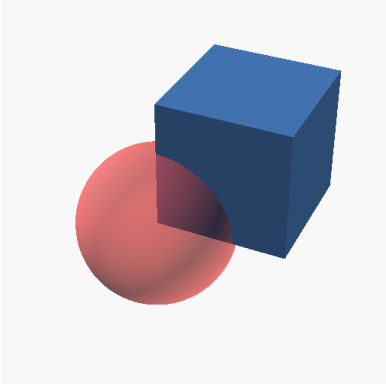

OpenScad examples

Table of Contents

Document information

Links to Document

Document online

Document Source

Document PDF

Experimental repo for building openscad files into different outputs.
This is still work in progress but can already build a png and stl of each scad file in the opescad directories.
See the online or pdf versions for the images as the readme is realy only the source and right now is not WYSIWYG!

Output

Amplifier - Project

VacuumPreAmplifierBase - 3D Object

housing a retro vacuum tube china preamp.
Have a nice wooden box that is looking for some use as a housing The pre-amp is a cheap vaccum tube type sourced from aliexpress https://a.aliexpress.com/_B0MVMZ
I've created an openscad model of the Box based on some measurments with a calliper. The model is designed to help asses where to drill holes and to print a guide to drill the holes. The preamp has a power input (12v~) an in and an output (headphone jack) and a Volume potentiometer. Also the housing is to expose the Vacuum Tubes to the interested viewer. Since the lid is hinged and the relative position of the tubes to the lid, when opening, is difficult to eyeball the model was created to try out different Hole placments as well as providing a template for Drill guides.

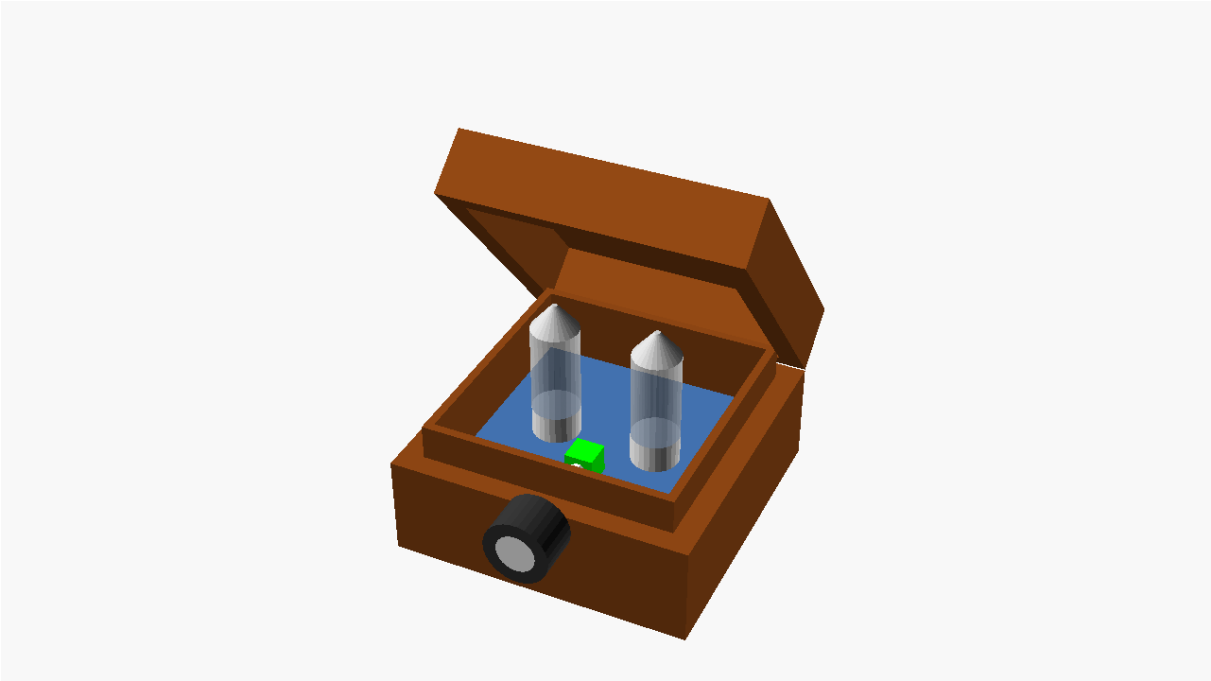

Figure 1. image

Listing 1. Openscad source

```
box1X=105.5;
box1Y=106;
box1Z=36;
box1BaseH=3;

box2X=89;
box2Y=91;
box2Z=45.5;

box3X=83;
box3Y=85;
box3Z=50;

lidX=box1X;
lidY=box1Y;
lidZ=23;
lidDepth=20.3;
lidStampR=20;
lidHingeAngle=50;
lidAnimZ=0;

preampBoardX=77;
preampBoardY=66;
preampBoardZ=1.5;
preampTubeR=17/2;
preampTubeH=42;
preampTubeBaseH=10;
preampTubeTipH=51;
```

```
preampTubeC=[200/255,200/255,200/255];
preampKnobR=11.5;
preampKnobH=16;
preampAxleH=29;

brown=[139/255,69/255,19/255];
gold=[255/255,215/255,0/255];
Blue=[0/255,0/255,200/255];

module box(){
    color(brown)
    difference(){
        union(){
            cube([box1X,box1Y,box1Z]);
            translate([box1X/2-box2X/2,box1Y/2-box2Y/2,box1BaseH])
cube([box2X,box2Y,box2Z]);
        }
        translate([box1X/2-box3X/2,box1Y/2-box3Y/2,3])
cube([box3X,box3Y,box3Z]);
        // star the next line to see inside the box
        *translate([-.5,-.5,-.5]) cube([box1X+1,box1Y*.85+1,box1Z/2+1]);
    }
}

//lid
module lid(){
    color(gold) translate([(box1X/2),(box1Y/2),lidZ+.0001])
cylinder(h=1,r1=lidStampR,r2=lidStampR);
    color(brown) translate([0,0,0])
        difference(){
            translate([0,0,.001]) cube([lidX,lidY,lidZ]);
            translate([box1X/2-box2X/2,box1Y/2-box2Y/2,0])
cube([box2X,box2Y,lidDepth]);
    }
}

module tube () {
    union(){
        color(preampTubeC,.5) translate([0,0,preampTubeBaseH])cylinder(h=42-
preampTubeBaseH,r1=preampTubeR,r2=preampTubeR);
        color([1,1,1])cylinder(h=preampTubeBaseH,r=preampTubeR);
        translate([0,0,preampTubeH]) color(preampTubeC)
cylinder(h=preampTubeTipH-preampTubeH,r1=preampTubeR,r2=1);
    }
}

translate([(box1X-box3X)/2,((box1Y-box3Y)/2)+(box3Y-preampBoardY)-
1,box1BaseH+21])
union() {
    //board
```

```
    cube([preampBoardX,preampBoardY,preampBoardZ]);
    //tubes
    translate([15+preampTubeR,15+preampTubeR,preampBoardZ]) tube();
    translate([52+preampTubeR,15+preampTubeR,preampBoardZ]) tube();
    //Volume Knob Base
    translate([38,0,preampBoardZ]) color([0,1,0]) cube([10,10,10]);
    //volume knob
    translate([43,-(preampKnobH+preampAxleH),preampBoardZ+5]) rotate([270,0,0])
    union(){
        difference() {
            color([50/255,50/255,50/255]) cylinder(h=preampKnobH,r=preampKnobR);
            translate([0,0,-.001]) cylinder(h=1,r=(preampKnobR/100)*60);
        }
        color([255/255,255/255,255/255])cylinder(h=1,r=(preampKnobR/100)*60);
        //knob axle
        translate([0,0,preampKnobH]) color([1,1,1])cylinder(h=preampAxleH,r=3);
    }
}
//draft base
translate([15,box1Y-((box1Y-box3Y)/2)-1,box1BaseH])
color([0,0,0])cube([8,1,21]);
//enclosure
box();
translate([box1X,box1Y,(box1Z)+lidAnimZ+.5]) rotate([lidHingeAngle,0,180])
lid();
```

vacuumRadioCase - 3D Object

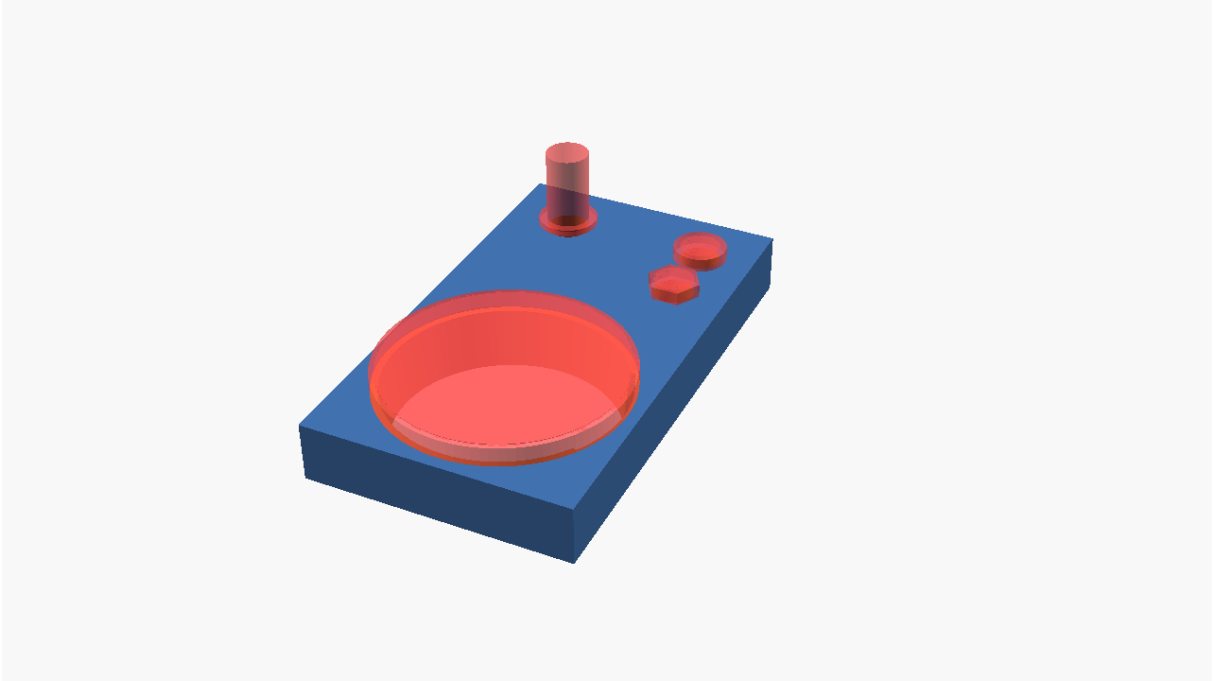

Figure 2. image

Listing 2. Openscad source

```
$fn=100;
baseH = 32;
baseW = 165 ;
baseD = 116.5 ;

innerH = 44 ;
innerW = 155 ;
innerD = 104 ;

ringW = 7.3 ;
ringH = 3 ;

xOff = (baseW - innerW) /2 ;
yOff = (baseD - innerD) /2 ;
zOff = 6 ;

module lid(){
    translate([0,0,ringH]) difference(){
        cube([baseW,baseD,baseH]);
        translate([xOff,yOff,zOff]) cube([innerW,innerD,innerH]);
    }
    translate ([ringW,ringW,0]) cube([baseW-2*(ringW),baseD-2*(ringW),ringH]);
}


module pot(text) {
    x=11;y=10;z=5;
    cylD=6.5;cylH=17;
    zWiggle=.1;
    translate([0,0,z/2-zWiggle]) cube([x,y,z+zWiggle],center=true);
    translate([0,0,z]) {
        translate([0,0,-zWiggle]) cylinder(h=cylH+zWiggle,d=cylD);
        translate([0,0,5]) cylinder(h=1,d=9);
    }
}

module speaker() {
    zWiggle = .1 ;
    outerD  = 36.5  ; innerD1 = 32.5 ; innerD2 = 35 ;
    outerH  = 3     ; innerH = 12 ;
    connectorD = 35.5 ;
    translate([0,0,9]) cylinder(h=outerH,d=outerD);
    cylinder(h=innerH + zWiggle, d1=innerD1, d2=innerD2);
    intersection() {
        color([1,1,1]) cylinder(h=innerH,d=connectorD);
        translate([0,-outerD/4,0]) cube([outerD,outerD/2,innerH]);
    }
}
```

```
module headJack(text) {
    upperD = 6 ; upperH = 4.8 ;
    lowerD = 8.5 ; lowerH = 11 ;
    nutH   = 2 ; nutD   = 8 ;
    translate([0,0,lowerH + upperH - nutH]) cylinder(h=nutH , d=nutD);
    cylinder(h=upperH + lowerH, d=upperD);
    cylinder(h=lowerH, d=lowerD);
}

module toggleSwitch(text) {
    upperD = 6 ; upperH = 4.8 ;
    lowerD = 8.5 ; lowerH = 11 ;
    nutH   = 2 ; nutD   = 8 ;
    translate([0,0,lowerH + upperH - nutH]) cylinder(h=nutH , d=nutD,$fn=6);
    cylinder(h=upperH + lowerH, d=upperD);
    cylinder(h=lowerH, d=lowerD);
}

//tests
difference() {
    zWiggle = .1 ;
    rigX = 40 ; rigY = 70 ;
    translate ([-rigX/2,-rigY,0]) cube([rigX,rigY,zOff+ringH]);
    translate([0,0,-zWiggle]) union(){
        #translate ([-10,-10,0]) pot("test");
        #translate ([0,-50,0]) speaker();
        #translate ([12,-10,-4.5]) headJack("test");
        #translate ([12,-20,-4.5]) toggleSwitch("test");
    }
}

//normal print
*difference() {
    zWiggle=.1;
    lid();
    //pot 1
    translate([20,baseD/2,0]) pot("pot 1");
    //pot 2
    translate([50,baseD/2,0]) pot("pot 2");
    //speaker
    translate([100,baseD/2,-zWiggle -2]) speaker();

}
```

Axle - Project

Axle - 3D Object

required a screw and didn't have one.

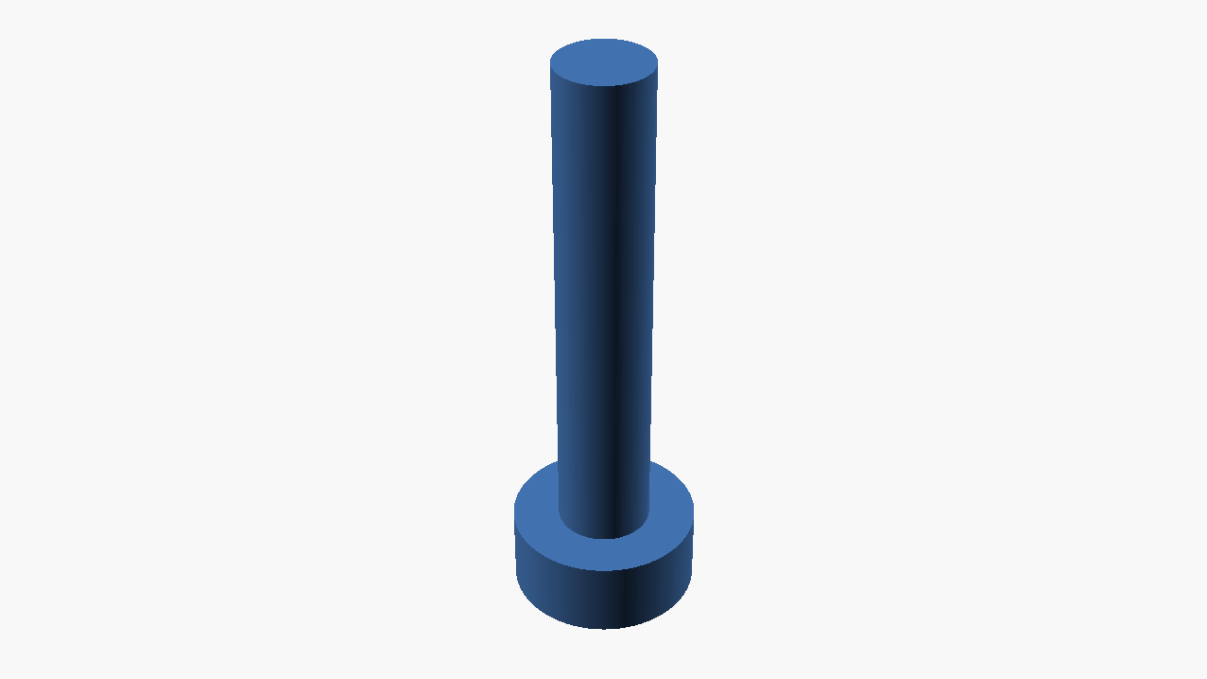

Figure 3. image

Listing 3. Openscad source

```
// axle for bearing for filament roller
// had no screw printed one ...
// the free end can be melted when the axle has been inserted so that no
fastener is required
$fn=360;
cylinder (h=22,d=3.5);
cylinder (h=3,d=7);
```

Howth-cieling-spots - Project

The spots for the ceiling in Howth needed replacing and the replacement spots had a few millimeters too little radius so that they didn't cover the hole properly in all places.
The lamp ring was a quick fix that turned out quite well and looks good.

lampRing - 3D Object

This object is not very complex and it's really just a few cylinders laid on top of one another with a few holes cut out for the lamp.

Figure 4. image

Listing 4. Openscad source

```
//for LED lamps in ceiling in Howth
// the originals are wider and therefore the new ones need a spacer to cover the
hole
//colour is white
//led lamps are 105mm Diameter (4 lamps)

lampD=105;
lampH=2;
holeD=99;
coverD=125;
coverInD=99;
coverH=2;
coverRidgeW=5;
$fn=100;

//lamp
*color("white")
    translate([0,0,coverH])
        cylinder(h=2,d=lampD);

color("white") union(){
    difference(){
        cylinder(h=coverH,d=coverD);
        translate([0,0,-.1]) cylinder(h=coverH+.2,d=coverInD);
    }
    translate([0,0,coverH])
    difference(){
        cylinder(h=lampH,d=lampD+coverRidgeW);
```

```
        translate([0,0,-.1]) cylinder(h=lampH+.2,d=lampD+1);
    }
}
```

Ikea-Repair - Project

ikeabung - 3D Object

This was a replacement foot for an IKEA shelf.

The actual foot was screwed in with a bolt on the underside.

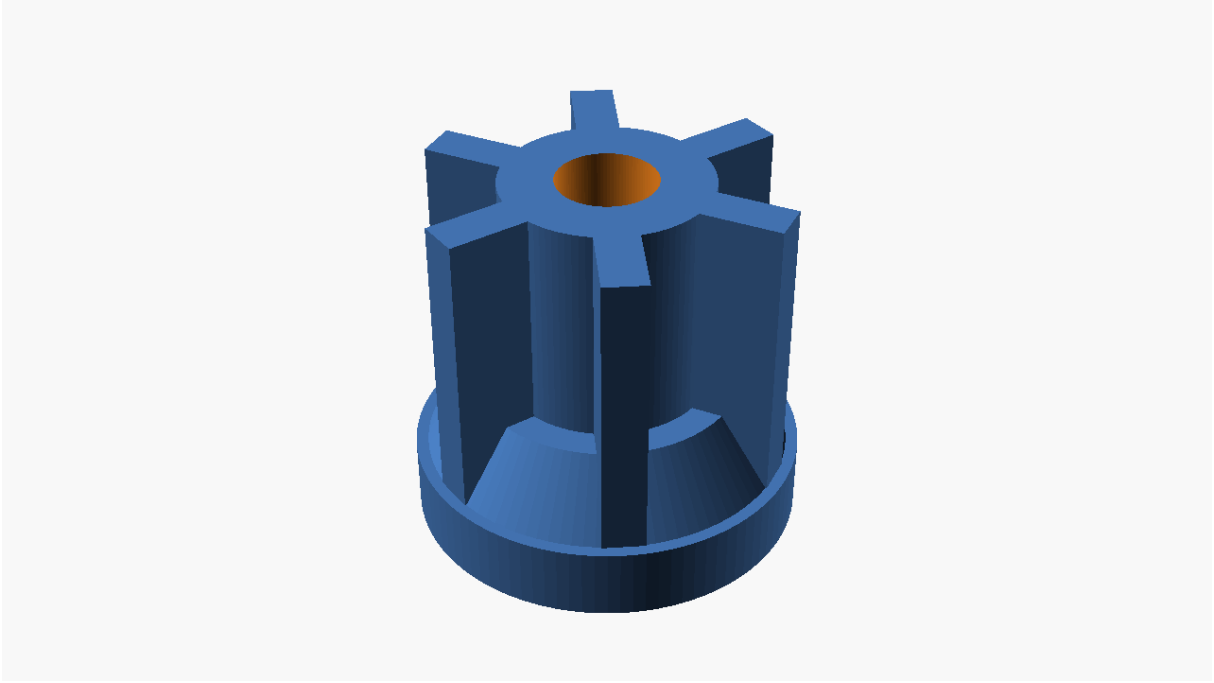

Figure 5. image

Listing 5. Openscad source

```
$fn=100;
totH=30;
baseH=6;
baseW=32;
wingW=3.5;
wingD=8;
centreD=17;

for (i = [0:360/6:360]) {
rotate([0,0,i]) translate([((baseW-2)/2)-wingD,-wingW/2,baseH])
cube([wingD,wingW,totH-baseH]);
}

difference(){
```

```
    union(){
        cylinder(h=totH,d=centreD);
        cylinder(h=6,d=32);
        translate([0,0,baseH]) cylinder(h=6,d1=baseW-2,d2=22);
        }
    translate([0,0,-.1]) cylinder(h=totH+.2,d=8.2);

    translate([0,0,-.1]) cylinder(h=8.1,d=15,$fn=6);
}
```

Jetson - Project

geekworm - 3D Object

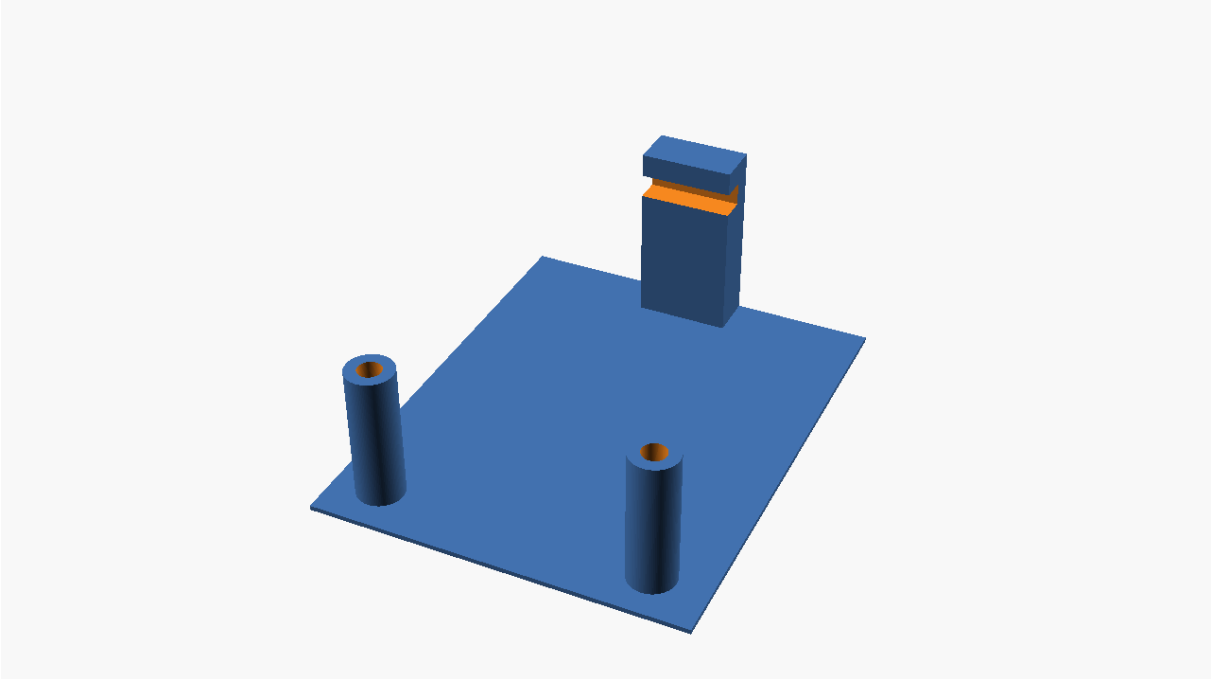

Figure 6. image

Listing 6. Openscad source

```
//casing
$fn=100;
pillarH = 30 ;
pillarD = 10 ;
pillarID = 5 ;
pillarSpacing = 58 ;

battLidX = 55 ;
battLidY = 55 ;
battLidZ = 2 ;
```

```
module pillar() {
    difference() {
        cylinder(h=pillarH, d=pillarD);
        translate([0,0,-.1]) cylinder(h=pillarH+.2, d=pillarID);
    }
}


translate ([10,10,0]) union() {
    //both pillars
    translate([0,0,0]) pillar();
    translate([pillarSpacing,0,0]) pillar();
    //The rear mount
    translate([20,80,0])
    union(){
        difference(){
            cube([20,10,40]);
            translate([-.5,-.5,30])cube([21,6,5]);
        }
    }
}

//battLid
*cube ([battLidX,battLidY,battLidZ]);

//casing base
cube ([80,100,1]);
```

power-ring - 3D Object

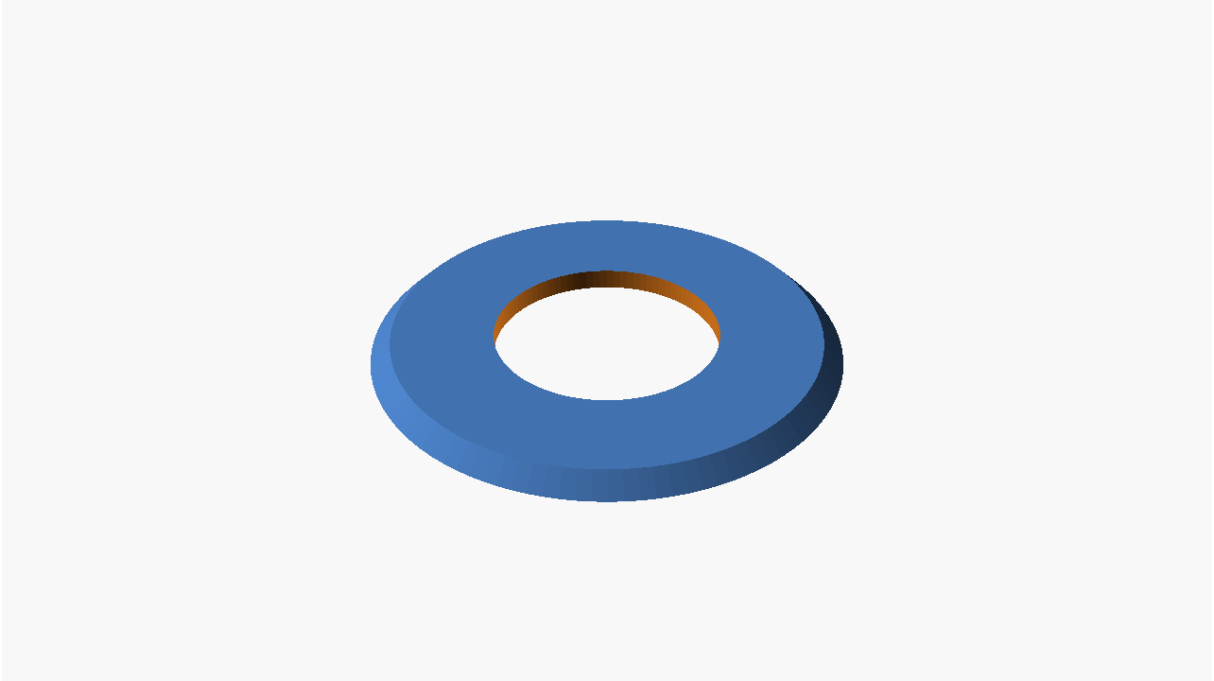

Figure 7. image

Listing 7. Openscad source

```
//since my jetson nano waveshare case has a reset and power button
// but they're swapped AND labeled.. this is to cover the reset label

$fn=100;

difference() {
    cylinder(h=1,d1=23,d2=21);
    translate([0,0,-.1]) cylinder(h=1.2,d=11);
}
```

Liebherr - Project

fridgeDoorInterimHandle - 3D Object

Figure 8. image

Listing 8. Openscad source

```
$fn=360;
Height=100;
Diameter=18;
HolePos=(Height/2);
HoleDiam=3;
HoleDepth=10;
difference () {
    hull() {
        translate([0,0,0])
            cylinder(h=1,d2=Diameter,d1=Diameter-2);
```

```
        translate([0,0,Height])
            cylinder(h=1,d1=Diameter,d2=Diameter-2);
    }
    translate([0,0,HolePos])
        rotate([90,0,0])
            cylinder(h=HoleDepth,d=HoleDiam);
}
```

Model-Plane - Project

mountingplateaircraftmotor - 3D Object

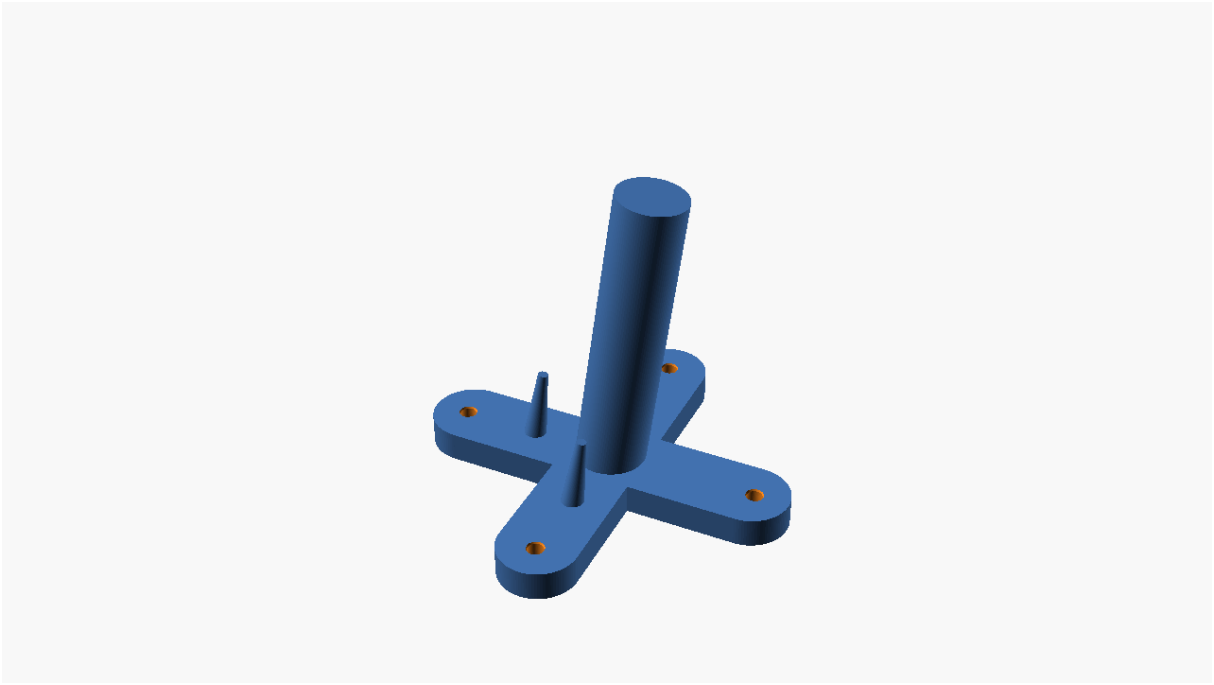

Figure 9. image

Listing 9. Openscad source

```
//second attempt with rotated arms so as to save on cutting out.
//set number of faces higher so that the cylinder doesn't lok like a pentagon
$fn=100;
//global vars
//text font Arial and so far size under 1.8 didn't show in print.....
font = "Arial";
letter_size = 2.5;
letter_height = 1.5;
line1="Ser#1";
line2="VSR";
line3="V6";
//pin depth into airframe of the wooden original was 30 test prints 20 is enough
//also of note is that I had to rotate the blasted pin by 45 and 3° as I drilled
the hole wrong. guess might be 5 or 6 (angle2) and one or two (angle1) to the
```

```
side
pinheight=30;
pinangle1=2;
pinangle2=7;
//radius of the mounting holes
screwhole=.81;
holeOffset=15.5;
//radius of screw hole opening
flare=1;

module letter(l) {
    // Use linear_extrude() to make the letters 3D objects as they
    // are only 2D shapes when only using text()
    linear_extrude(height = letter_height) {
        text(l, size = letter_size, font = font);
    }
}

//this is the 4 armed mount with drill holes
module mount()
    {
    union(){
        for ( arm = [0:90:360]){
            rotate([0,0,arm])
                //arm with drill hole
                difference(){
                    union(){
                        //arm
                        translate ([0,-4,0]) cube([15,8,3] );
                        //rounded tip
                        translate ([15,0,0]) cylinder(h=3, r1=4, r2=4);
                        }
                    //  subtract drill hole plus additional
                    translate([holeOffset,0,0]) cylinder(h=3, r1=screwhole,
r2=screwhole);
                    //flaring
                    translate([holeOffset,0,2.8]) cylinder(h=.5, r1=screwhole,
r2=flare);
                    translate([holeOffset,0,0]) cylinder(h=.25, r1=flare,
r2=screwhole);
                    //central mounting hole for spindle
                    cylinder(h=1.5, r1=2.5, r2=2.1);
                }
            }
        }
    }


module mountpluspin()
{
```

```
//the mount and the pin for insertion into the aircraft body
    union(){
        mount();
        //central mounting rod should be 30 long as measured but shorter    for
test prints
        //aslo of note is that I had to rotate the blasted pin by 45 and 3° as I
drilled the hole wrong. guess is 3° might be 5 or 6
        rotate ([pinangle1,pinangle2,45]) translate ([0,0,2])
cylinder(h=pinheight, r1=3.5, r2=3.5);
        *translate ([4,1,2.2])  letter (line1);
        *rotate(90,90,90) translate ([4,-1,2.2])  letter (line2);
        *translate ([4,-3,2.2])  letter (line3);
        }
}
//add some antispin pegs.
union (){
    mountpluspin();
    translate ([0,-8,2]) rotate ([pinangle1,pinangle2,45]) cylinder(h=8, r1=1.2,
r2=.5);
    translate ([-8,0,2]) rotate ([pinangle1,pinangle2,45]) cylinder(h=8, r1=1.2,
r2=.5);
}
```

RollHolder - Project

rollholder - 3D Object

Figure 10. image

Listing 10. Openscad source

```
$fn=360;
plateH = 2.5 ;
topInsideH = 6 ;
topInsideD = 18.35 ;
topInsideR = topInsideD/2 ;
insideD = 17.5 ;
insideH = 40 ;
lipH = .09 ;
lipW = .5 ;
lipOffH = 2.4 ;
headH = 2 ;
headD = 26.5 ;
plateHoleD = 19.22 ;
plateHoleH = 3 ;
//diameter of top inside
color("grey") translate([0,0,-plateH]) cylinder(h=6+plateH,d=topInsideD);
//diameter of inside
color("grey") translate([0,0,5]) cylinder(h=insideH,d=insideD);
//Lip
color("red") translate([0,0,lipOffH])
cylinder(h=lipW,r1=topInsideR+lipH,r2=topInsideR);
color("red") translate([0,0,lipOffH-lipW])
cylinder(h=lipW,r2=topInsideR+lipH,r1=topInsideR);
//mount ring
color("grey") translate([0,0,-plateHoleH+.1])
cylinder(h=plateHoleH+.1,d=plateHoleD);

//head
color("grey") translate([0,0,-(headH+plateH)]) cylinder(h=headH,d=headD);
//plate
*translate([0,0,-2.5]) difference(){
    translate([-20,-20.0])cube([40,40,plateH]);
    translate([0,0,-.1]) cylinder(h=plateHoleH,d=plateHoleD);
}
```

stopfen - 3D Object

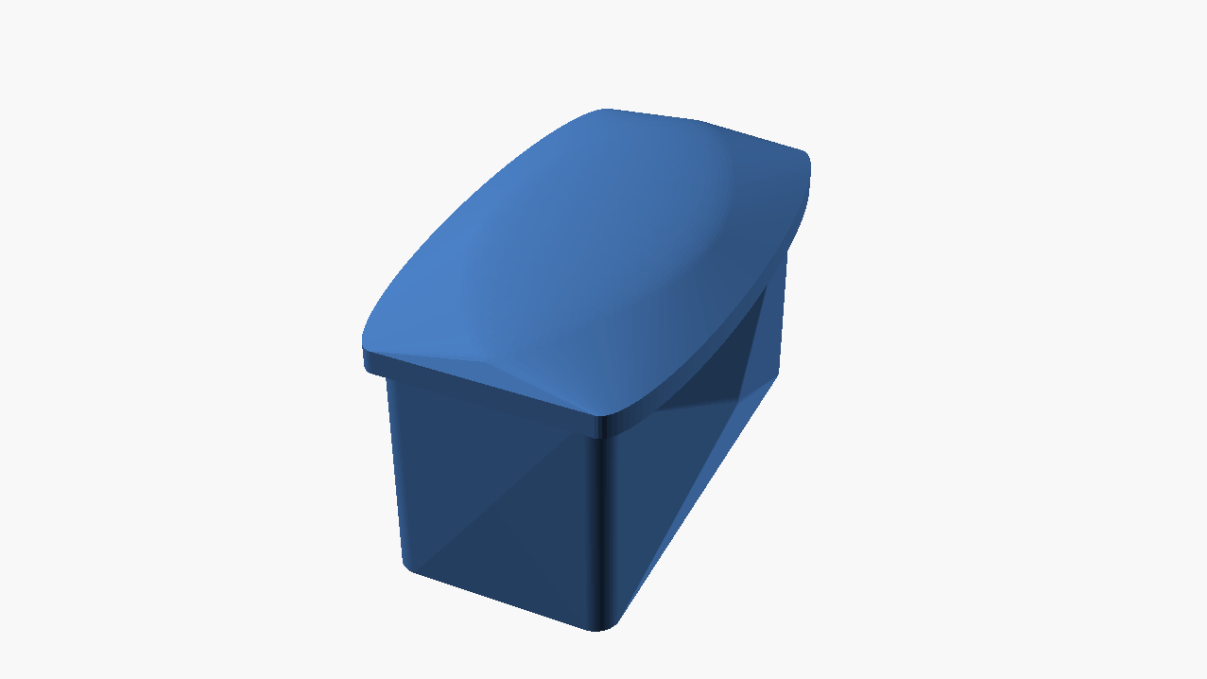

Figure 11. image

Listing 11. Openscad source

```
$fn=100;

//Variables
baseX = 9.5 /2 ;
baseY = 19.6 /2 ;
cornerR = 1 ;
bottomExtrude = 10 ;
topExtrude = 1 ;
bumpTB = .15 ;
bumpRL = 1.25 ;
//Composites
    baseRight = [+baseX, 0, 0] + [-cornerR, 0, 0] ;
    baseLeft  = [-baseX, 0, 0] + [+cornerR, 0, 0] ;
    baseBottom  = [0, -baseY, 0] + [0, +cornerR, 0] ;
    baseTop = [0, +baseY, 0] + [0, -cornerR, 0] ;
    bumpRad = cornerR ;

// Temp handle
* cylinder(h=20,d=3);

hull() {
    //Corners
    translate(baseTop + baseRight) cylinder(h=bottomExtrude, r=cornerR, center =
true) ;
    translate(baseBottom + baseRight) cylinder(h=bottomExtrude, r=cornerR,
center = true) ;
    translate(baseBottom + baseLeft) cylinder(h=bottomExtrude, r=cornerR, center
= true) ;
```

```
    translate(baseTop + baseLeft) cylinder(h=bottomExtrude, r=cornerR, center =
true) ;
    //bumps
    translate(baseTop    + [0,+bumpTB,0]) sphere(r=bumpRad);
    translate(baseRight  + [+bumpRL,0,0]) sphere(r=bumpRad);
    translate(baseBottom + [0,-bumpTB,0]) sphere(r=bumpRad);
    translate(baseLeft   + [-bumpRL,0,0]) sphere(r=bumpRad);
}


translate([0,0,bottomExtrude/2+topExtrude/2]) hull() {
    lidSideR = 2 ;
    lidX = baseX + .5 ;
    lidRight = [+lidX, 0, 0] + [-lidSideR, 0, 0] ;
    lidLeft  = [-lidX, 0, 0] + [+lidSideR, 0, 0] ;
    //bumps
    translate(lidRight  + [+bumpRL,0,0]) scale([1,5.2,1]) cylinder(h=topExtrude,
r=lidSideR, center = true) ;
    translate(lidLeft   + [-bumpRL,0,0]) scale([1,5.2,1]) cylinder(h=topExtrude,
r=lidSideR, center = true) ;
    //top blob
    translate([0, 0, 1]) scale([5,10,1]) sphere(r=bumpRad);
}
```

SWD - Project

swd-cube - 3D Object

Figure 12. image

Listing 12. Openscad source

```
font = "Liberation Sans";
cube_size = 70;
letter_size = 50;
letter_height = 5;
o = cube_size / 2 - letter_height / 2;

module letter(letter) {
    linear_extrude(height = letter_height) {
        text(letter, size = letter_size, font = font, halign = "center", valign
= "center", $fn = 100);
    }
}

union() {
    color("gray") cube(cube_size, center = true);
    translate([0, -o, 0]) rotate([90, 0, 0]) letter("W");
    translate([o, 0, 0]) rotate([90, 0, 90]) letter("D");
  translate([0, 0, o])  letter("S");
}
```

Test - Project

Test - 3D Object

Just a simple example and also used as the logo for this repo.

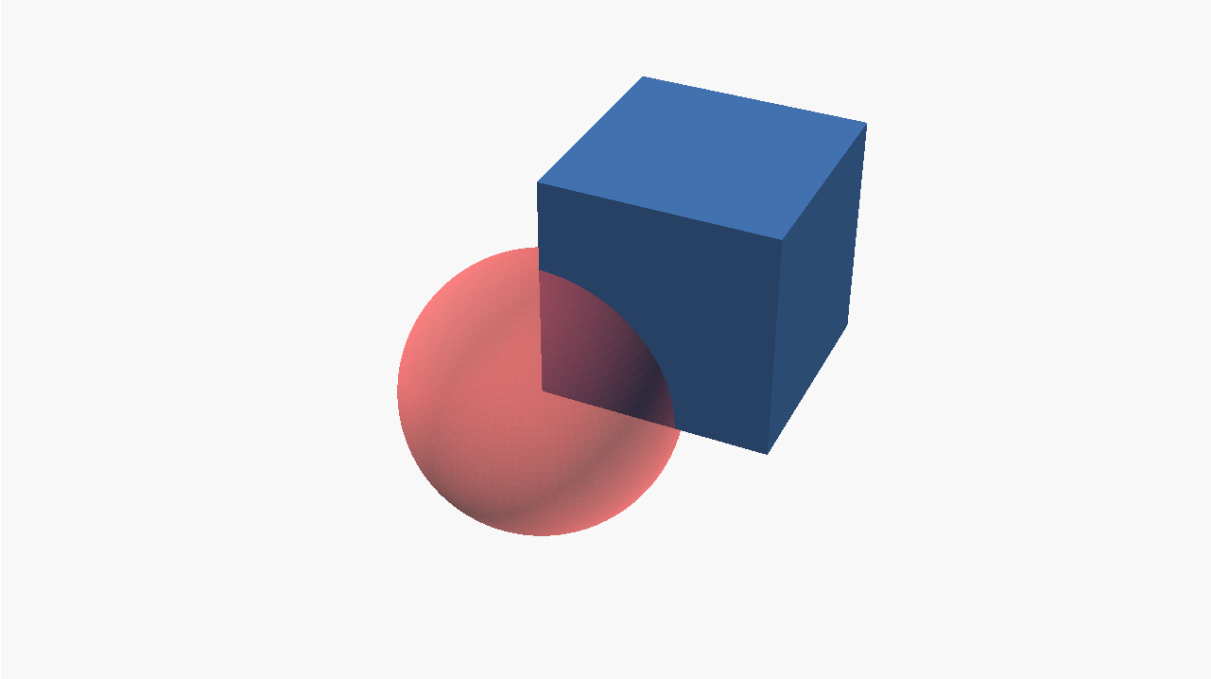

Figure 13. image

Listing 13. Openscad source

```
//just a simple test drawing
$fn=100;
cube([10,10,10]);
#sphere(d=12);
```

Thinkpad_logo - 3D Object

Figure 14. image

Listing 14. Openscad source

```
tlColor = [1,1,1] ;
tlH = 1 ;

//logo
color(tlColor) linear_extrude(height=tlH) import("ThinkPad_Logo.svg",center =
true);

//base
cube([30,10,.5], center = true) ;
```

vectorbox-test - 3D Object

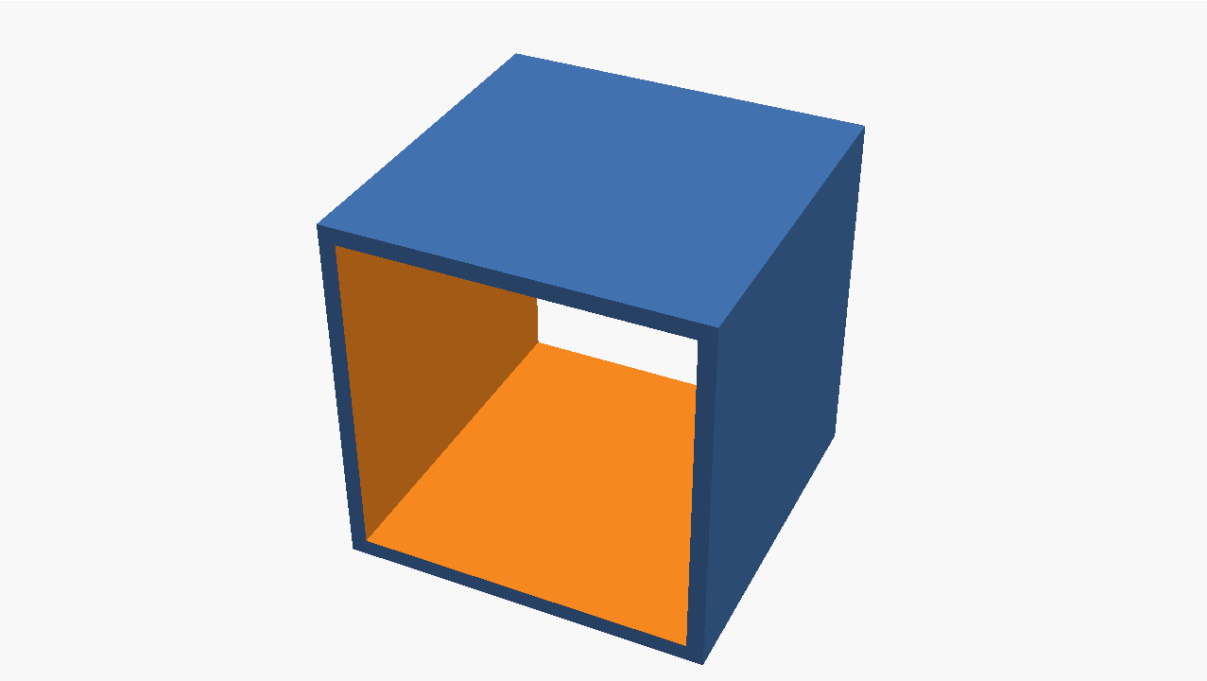

Figure 15. image

Listing 15. Openscad source

```
module vectorBox(){
    //just a small demonstrator for vector subtraction and addition
    // as things can get complicated with single unit variables the below shows
how it can be abstracted and simplified

    //inside side length
    inSize = 10 ;
    //wall thickness
    wallThick = .5;

    //the vectors predefined
    inSizeA = [inSize,inSize,inSize] ;
    inSizeX = [inSize,0,0] ;
    inSizeY = [0,inSize,0] ;
    inSizeZ = [0,0,inSize] ;
    offset = wallThick * 2 ;
    offsetA = [offset,offset,offset] ;
    offsetX = [offset,0,0] ;
    offsetY = [0,offset,0] ;
    offsetZ = [0,0,offset] ;
    offsetZY = offsetZ + offsetY ;
    offsetZX = offsetZ + offsetX ;
    offsetXY = offsetX + offsetY ;
    diffWiggle = .1 ;
    diffWiggleY = [0,diffWiggle,0] ;

    difference() {
        //total volume cube
```

```
        cube(inSizeA + offsetA);
        //inside volume cube
        translate((offsetZX)/2 - diffWiggleY/2)
            cube(inSizeA + offsetY + diffWiggleY);
    }

}
vectorBox();
```

ThingsByOthers - Project

SQ11-15 - 3D Object

Figure 16. image

Listing 16. Openscad source

```
cam_width=23.7;
cam_depth=23.1;
cam_height=23.1+0.3;

corner_radius=2;

lip_width = 5.1; // metal radiator & separator

shell_thickness = 1;

screw_cylinder_diameter = 8;

lens_width_height = 19;
```

```
lens_corner_radius = 5;

ridge_from_lens = 3.7+0.5;
ridge_thickness = 1 - 0.2;

button_width = 14.6+1;
button_depth = 4.8;
button_height = shell_thickness + 1; // false, just for the hole

button_from_lens = 11.3-4.5+0.1;

sd_depth = 3.1;
sd_height = 18.9;
sd_width = 12;
sd_from_lens = 10.75;

screw_from_lens = 23;
battery_length = 65+10+5;

spring_diameter = 8;

battery_height = battery_length + screw_cylinder_diameter;

screw_cynlinder_inner_length = cam_width - shell_thickness * 2;

bay = battery_height;
sil = screw_cynlinder_inner_length;

cx = cam_width; // SD & USB
cy = cam_depth+bay; // lens & screw
cz = cam_height; // buttons
cr = corner_radius;
lw = lip_width;
st = shell_thickness;
sc = screw_cylinder_diameter;

ix = cx - st * 2;
iy = cy - st * 2;
iz = cz - st * 2;

lxy = lens_width_height;
lc = lens_corner_radius;

ry = ridge_from_lens+0.1;
rt = ridge_thickness;

bx = button_width;
by = button_depth;
bz = button_height;

bl = button_from_lens;
```

```
sd = sd_depth;
sh = sd_height;
sl = sd_from_lens;
sw = sd_width;

sfl = screw_from_lens;

spd = spring_diameter;

*%cube([cx,cy,cz]); //general camera shape

$fn=30;

rotate([0,180,0]) // <------------------ comment this
intersection()
{
  translate([0,0,cx/2]) // <------------ comment this
  cube([cz,cy*2,cx/2]);

  translate([cz,0,0])
  rotate([0,-90,0])
  {
    difference()
    {
      hull_from_cube(cx,cy,cz,cr);

      translate([st,st,st])
        hull_from_cube(ix,iy,iz,cr);

      translate([(cx-lw)/2,0,0])
        cube([lw,cy-bay,cz]); // middle cut

      translate([cx/2,0,cz/2])
        translate([-lxy/2,0,-lxy/2])
          translate([0,st/2,0])
            sensor_hull();

      translate([cx/2,bl+by/2,cz-st/2-0.01])
        button_hole();

      translate([-st/2,sl,(cz-sh)/2])
        sd_hole();

      translate([0,sfl,sc/2+st])
        rotate([0,90,0])
          full_screw_hole();

      translate([0,sfl,cz-(sc/2+st)])
        rotate([0,90,0])
          full_screw_hole();
```

```
    translate([0,cy-(sc/2+st),sc/2+st])
      rotate([0,90,0])
        full_screw_hole();

    translate([0,cy-(sc/2+st),cz-(sc/2+st)])
      rotate([0,90,0])
        full_screw_hole();

  }

  translate([0,sfl,sc/2+st])
    rotate([0,90,0])
      difference()
      {
        translate([0,0,st])
        union()
        {
          cylinder(d=sc,h=sil);
          translate([0,-sc/2,0])
            cube([sc/2,sc,sil]);
        }
        translate([sc/2-1,-sc/2-sc+3.6,st+sil/2-lw/2])
            cube([sc/2,sc,lw]);

        full_screw_hole();
      }

  translate([0,sfl,cz-(sc/2+st)])
    rotate([0,90,0])
      difference()
      {
        translate([0,0,st])
        union()
        {
          cylinder(d=sc,h=sil);
          translate([-sc/2,-sc/2,0])
            cube([sc/2,sc,sil]);
        }
        translate([-sc+1,-sc/2-sc+3.6,st+(sil)/2-lw/2])
            cube([sc/2,sc,lw]);

        full_screw_hole();
      }

  translate([0,cy-(sc/2+st),sc/2+st])
    rotate([0,90,0])
      difference()
      {
        translate([0,0,st])
        union()
```

```
        {
          cylinder(d=sc,h=sil);
          translate([0,-sc/2,0])
            cube([sc/2,sc,sil]);
        }

        full_screw_hole();
      }

  translate([0,cy-(sc/2+st),cz-(sc/2+st)])
    rotate([0,90,0])
      difference()
      {
        translate([0,0,st])
        union()
        {
          cylinder(d=sc,h=sil);
          translate([-sc/2,-sc/2,0])
            cube([sc/2,sc,sil]);
        }

        full_screw_hole();
      }

  ridge();

  translate([0,sfl+sc/2+1,0])
    difference()
    {
      translate([st,0,st])
        cube([sil,rt,cz-st*2]);

      translate([st+(sil)/2,st,st+(cz-st*2)/2])
        rotate([90,0,0])
          cylinder(d=spd,h=rt*2);

      translate([st,-rt/2,(sil-2)/2])
        cube([cz-st*2,rt*2,4]);
    }

  translate([0,cy-(sc+st+rt+1),0])
    difference()
    {
      translate([st,0,st])
        cube([sil,rt,cz-st*2]);

      translate([st+(sil)/2,st,st+(cz-st*2)/2])
        rotate([90,0,0])
          cylinder(d=spd,h=rt*2);
```

```
        translate([st,-rt/2,(sil-2)/2])
          cube([cz-st*2,rt*2,4]);
      }

  }
}

module hull_from_cube(x,y,z,r)
{
  hull()
  {
    translate([r,r,r]) sphere(r=r);
    translate([x-r,r,r]) sphere(r=r);
    translate([r,y-r,r]) sphere(r=r);
    translate([x-r,y-r,r]) sphere(r=r);

    translate([r,r,z-r]) sphere(r=r);
    translate([x-r,r,z-r]) sphere(r=r);
    translate([r,y-r,z-r]) sphere(r=r);
    translate([x-r,y-r,z-r]) sphere(r=r);
  }
}

module sensor_hull()
{
  hull()
  {
    translate([0,st,0])
    {
      translate([lc,0,lc]) sphere(r=lc);
      translate([lxy-lc,0,lc]) sphere(r=lc);
      translate([lxy-lc,0,lxy-lc]) sphere(r=lc);
      translate([lc,0,lxy-lc]) sphere(r=lc);
    }

    translate([0,-st,0])
    {
      translate([lc,0,lc]) sphere(r=lc);
      translate([lxy-lc,0,lc]) sphere(r=lc);
      translate([lxy-lc,0,lxy-lc]) sphere(r=lc);
      translate([lc,0,lxy-lc]) sphere(r=lc);
    }
  }
}

module button_hole()
{
  difference()
  {
    cube([bx,by,bz],center=true);
```

```
    difference()
    {
      cube([bx+1,by-st,bz+1],center=true);

      cube([lw+1,by-st+1,bz+2],center=true);
    }
  }
}

module sd_hole()
{
  union()
  {
    difference()
    {
      cube([st*2,sd,sh]);
      translate([-0.5,-0.5,sh-sw+0.01])
        cube([st*2+1,sd+1,sw+1]);
    }
    translate([0,0.5,sh-sw])
        cube([st*2,sd+1,sw]);
  }
}

module ridge()
{
  translate([0,ry,0])
    difference()
    {
      translate([st,0,st])
        cube([sil,rt,cz-st*2]);

      translate([(cx-lw)/2,-ry,0])
        cube([lw,ry*2,cz]);

      translate([-1,-1,cr])
        cube([cx+2,rt+2,cz-cr*2]);
    }
}

module full_screw_hole()
{
  translate([0,0,cx])
    rotate([0,180,0])
      screwhole(cx/2);

  rotate([0,0,30])
    translate([0,0,cx/2+2])
      rotate([0,180,0])
```

```
        nuthole(cx/2);
}

module screwhole(wz)
{
  cylinder(d=3.2,h=wz+1);

  translate([0,0,-0.01])
    cylinder(d=6,h=3.5);
}

module nuthole(wz)
{
  union()
  {
    translate([0,0,-0.5])
      cylinder(d=3.2,h=wz+1);

    nd = 5.5+0.15;
    nh = 2.4+3;

    translate([0,0,wz-nh+0.1+3])
      cylinder(d=nd+0.9,h=nh,$fn=6);
  }
}
```

notmine1 - 3D Object

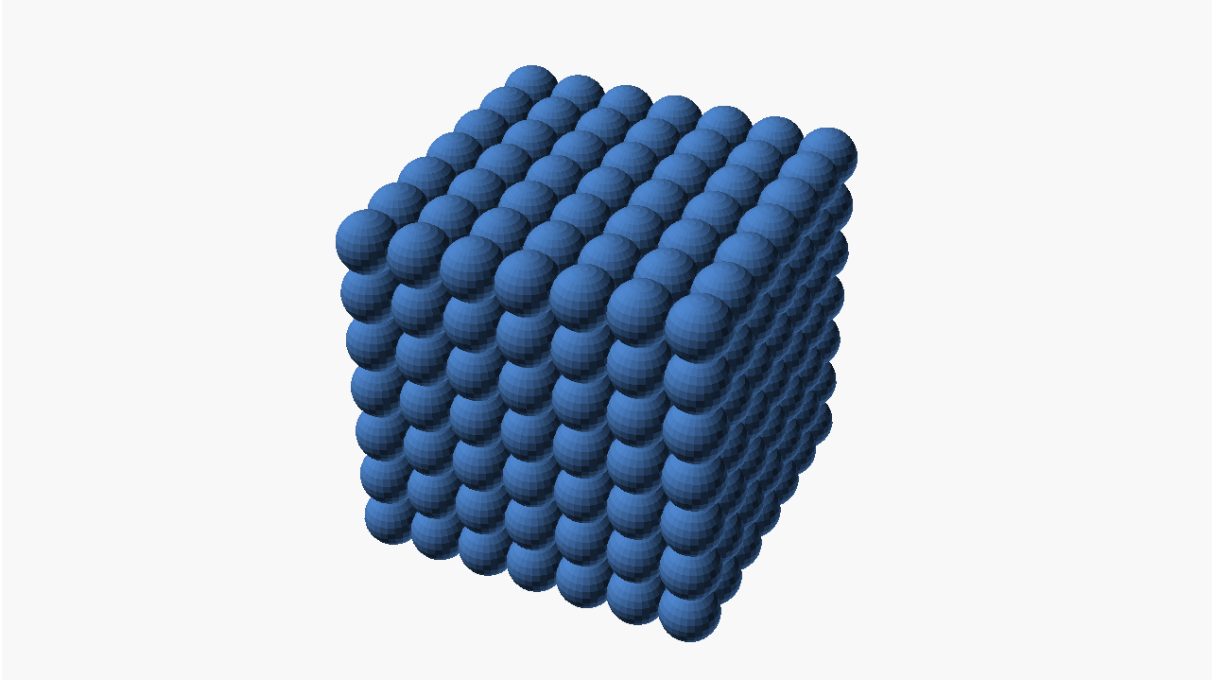

Figure 17. image

Listing 17. Openscad source

```
module manyballs(n)
{
    $fn=25;
    delta = 45;
    for(i=[0:1:n-1]) {
       x = i*delta;
       for(j=[0:1:n-1]) {
          y = j*delta;
          for(k=[0:1:n-1]) {
             z = k*delta;
             translate([x,y,z])sphere(25);
          }
       }
    }
}

manyballs(n=7);
```

velcro - 3D Object

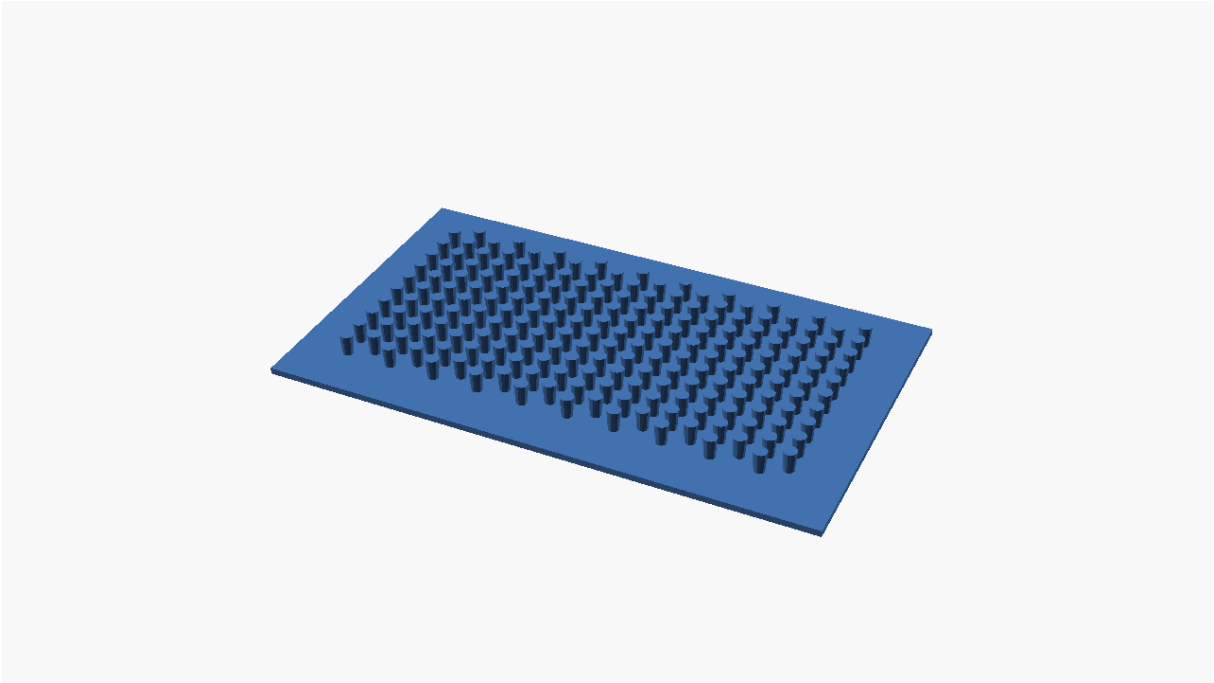

Figure 18. image

Listing 18. Openscad source

```
// Nameplate OpenSCAD example modified by Michael Laws, to create a parametric
version of 'Printable Velcro' by MM Printing:
https://www.printables.com/model/543802-printable-velcro , itself a remix of
'Printable VELCRO' by eried: https://www.printables.com/model/33302-printable-
velcro
```

```
//
// Original license:
//
// Written by Amarjeet Singh Kapoor <amarjeet.kapoor1@gmail.com>
//
// To the extent possible under law, the author(s) have dedicated all
// copyright and related and neighboring rights to this software to the
// public domain worldwide. This software is distributed without any
// warranty.
//
// You should have received a copy of the CC0 Public Domain
// Dedication along with this software.
// If not, see <http://creativecommons.org/publicdomain/zero/1.0/>.

/*[ Velcro tower elements]*/

//The diameter of the lowest (and narrowest) part of each tower in mm. A base
diameter of 1.0, will yield a top diameter of 1.3, and a tower height of 2.0mm.
Base_diameter = 1.0;//[0.5:0.1:10]

function Top_diameter() = Base_diameter*1.3;

//The height of each tower in mm.
function Height() = Base_diameter*2;

//How tight the fit is. This is an arbitrary value, not a measurement. -10 is
tightest, 10 is loosest, 0 matches the original MM Printing model. As you scale
up the base diameter, a tighter interference is needed.
Interference = 0;//[-10 : 10]

/*[ Pattern ] */
// Pattern is twice as wide horizontally as vertically. For a square pattern,
the vertical value should be twice the horizontal.

//Horizontal tower sets in pattern.
Horizontal = 10; //[1:1:1000]

//Vertical tower sets in pattern.
Vertical = 10; //[1:1:1000]

/*[ Base plate ] */

//Thickness of the baseplate.
Thickness = 0.6;//[0.2:0.1:5]

// Horiziontal offset of base plate border from towers in mm.
BorderH = 5;//[1:1:100]

// Vertical offset of base plate border from towers in mm.
BorderV = 5;//[1:1:100]
```

```
/*[Misc] */

//Number of fragments in 360 degrees. Higher value produces a more detailed
output at the expense of file size and processing time.
Resolution = 12;//[5:100]

$fn = Resolution;

function factor() = Base_diameter/12.5;

function spacing()=Base_diameter*4.4+Interference*factor();

module tower(){
        cylinder(h = Height(), r1 = Base_diameter/2, r2 = Top_diameter()/2,
center = false);
}

module single_array(){
    render(){
        h=spacing();
        tower();
        translate([h/2,h/4,0])tower();
    }
}

module horizontal_array(){
    render(){
        h=spacing();
        for(dx=[0:h:h*Horizontal-1]){
           translate([dx,0,0])single_array();
        }
    }
}

module large_array(){
    v=spacing()/2;
    for(dy=[0:v:v*Vertical-1]){
       translate([0,dy,0])horizontal_array();
    }
}

module base_plate(){
    h=spacing();
    v=h/2;
    translate([-1*BorderH,-1*BorderV,-
1*Thickness])cube([(h*Horizontal+BorderH*2-h/2),v*Vertical+BorderV*2-
v/2,Thickness]);
}

union(){
    large_array();
```

```
    base_plate();
}
```

VSAGCRD-Logo - Project

vsagcrd - 3D Object

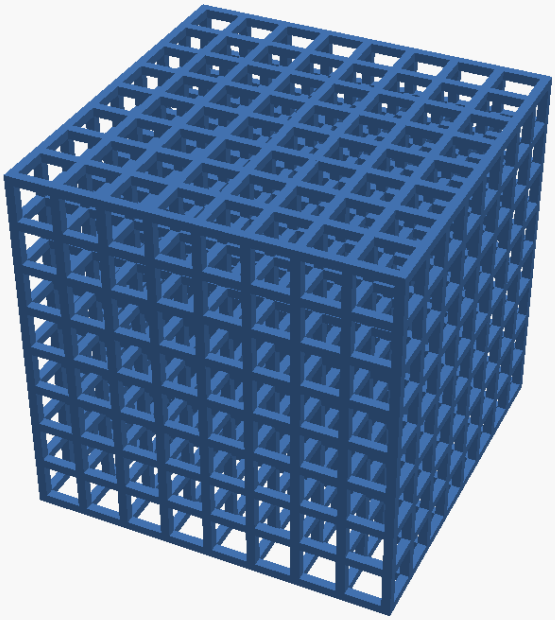

Figure 19. image

Listing 19. Openscad source

```
//Script to create a wire mesh cube with 8x8x8 empty spaces
//Virtual Space and
//Global communications research department
//logo base object with 8.2 cm side length
//consisting of the multiplied basic primitives of
//an x,y,and z axis beam iterate in one dimension in loops
//also includes the inner primitive (in 3 flavours)
$fn=100;
module x_beam(){ //just a cube with parameters in one place
    cube([82,2,2]);
}
module y_beam(){ //just a cube with parameters in one place
    cube([2,82,2]);
}
module z_beam(){ //just a cube with parameters in one place
    cube([2,2,82]);
}
module x_block(){ //primitive used for 8 points in the 3d grid
    *color([0,1,0]) translate ([02,02,02]) cube([8,8,8]);
```

```
    translate ([6,6,6]) sphere (r=1);
}
module vsr_cube(){ //loops for the xyz forests of beams
    union(){
        //xbeam
        for (xj=[0:10:80]){
            for (xi=[0:10:80]){
                translate([0,xi,xj])
                x_beam();
            }
        }
        //ybeam
        for (yj=[0:10:80]){
            for (yi=[0:10:80]){
                translate([yi,0,yj])
                y_beam();
            }
        }
        //zbeam
        for (zj=[0:10:80]){
            for (zi=[0:10:80]){
                translate([zi,zj,0])
                z_beam();
            }
        }
    }
}
module inner_vsr_cube(){ //the manual inner cube
    union(){
        //first set
        hull() {
            translate ([30,50,70]) x_block();
            translate ([10,10,60]) x_block();
        }
        hull() {
            translate ([70,40,50]) x_block();
            translate ([50,00,40]) x_block();
        }
        hull() {
            translate ([20,70,30]) x_block();
            translate ([00,30,20]) x_block();
        }
        hull() {
            translate ([60,60,10]) x_block();
            translate ([40,20,00]) x_block();
        }
        //second set
        hull() {
            translate ([30,50,70]) x_block();
            translate ([20,70,30]) x_block();
```

```
        }
        hull() {
            translate ([10,10,60]) x_block();
            translate ([00,30,20]) x_block();
        }
        hull() {
            translate ([70,40,50]) x_block();
            translate ([60,60,10]) x_block();
        }
        hull() {
            translate ([50,00,40]) x_block();
            translate ([40,20,00]) x_block();
        }
        //third set
        hull() {
            translate ([30,50,70]) x_block();
            translate ([70,40,50]) x_block();
        }
        hull() {
            translate ([10,10,60]) x_block();
            translate ([50,00,40]) x_block();
        }
        hull() {
            translate ([20,70,30]) x_block();
            translate ([60,60,10]) x_block();
        }
        hull() {
            translate ([00,30,20]) x_block();
            translate ([40,20,00]) x_block();
        }
    }
    //fourth full set as option
    *hull() { //hull over all 8 points in 3d space
        translate ([30,50,70]) x_block();
        translate ([10,10,60]) x_block();
        translate ([70,40,50]) x_block();
        translate ([50,00,40]) x_block();
        translate ([20,70,30]) x_block();
        translate ([00,30,20]) x_block();
        translate ([60,60,10]) x_block();
        translate ([40,20,00]) x_block();
    }
}
//paint the outer framed cube 8x8x8
vsr_cube();
//paint the inner cube either as wire or solid or points
*inner_vsr_cube();
```

Vacuum-rig-adapter - Project

Hose_Adaptor - 3D Object

Not one of mine:
Created by Paul Tibble - 18/7/19
* https://www.thingiverse.com/Paul_Tibble/about

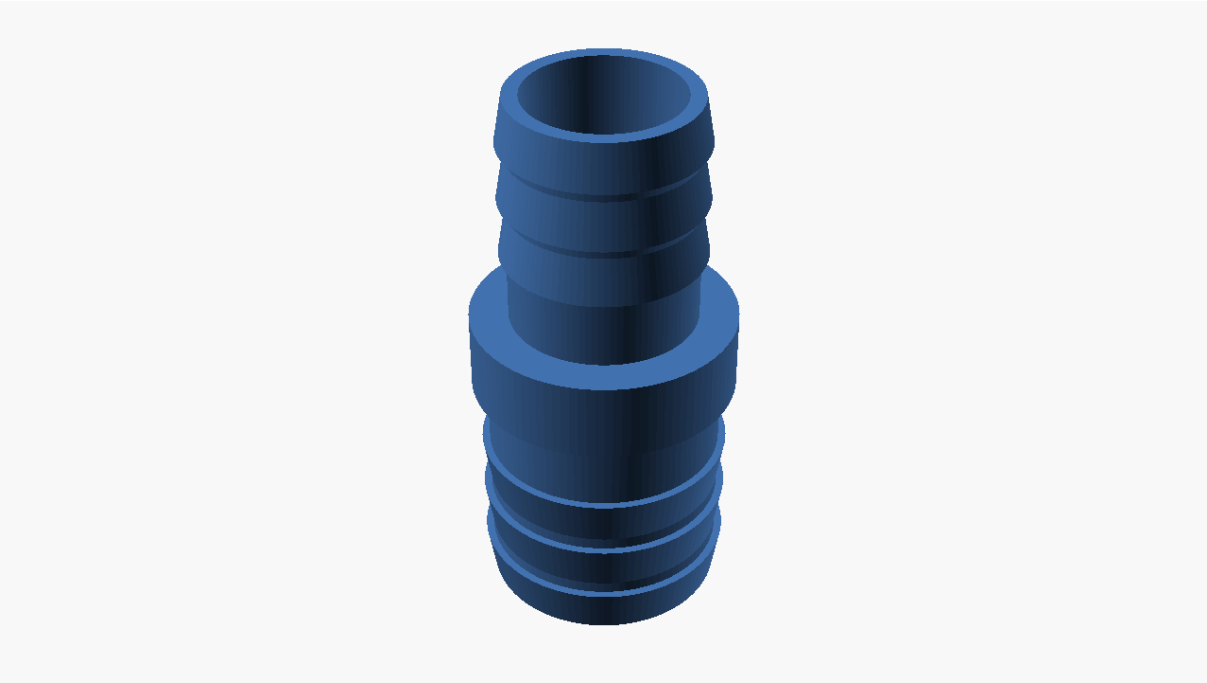

Figure 20. image

Listing 20. Openscad source

```
////////////////////////////////////////////////////
// Created by Paul Tibble - 18/7/19                //
// https://www.thingiverse.com/Paul_Tibble/about   //
// Please consider tipping, if you find this useful.  //
////////////////////////////////////////////////////

$fn = 100*1;

// Outer Diameter (bottom)
outer_diameter_1 = 15;
// Wall Thickness (bottom)
wall_thickness_1 = 2;
// Rib Thickness (bottom), set to Zero to remove
barb_size_1 = 0.5;
// Length (bottom)
length_1 = 15;
// Outer Diameter (top), should be smaller than or equal to Outer Diameter
(bottom)
outer_diameter_2 = 12;
```

```
// Wall Thickness (top)
wall_thickness_2 = 1;
// Rib Thickness (top), set to Zero to remove
barb_size_2 = 0.5;
// Length (top)
length_2 = 15;
// Middle Diameter
mid_diameter = 17;
// Middle Length
mid_length = 5;


//do not change these
inner_diameter_1 = outer_diameter_1 - (wall_thickness_1*2);
inner_diameter_2 = outer_diameter_2 - (wall_thickness_2*2);

module create_profile() {
    ////////
    // Middle
    ///////

polygon(points=[[inner_diameter_1/2,length_1],[mid_diameter/2,length_1],[mid_dia
meter/2,length_1+mid_length],[inner_diameter_2/2,length_1+mid_length]]);
    //////
    //length 1
    /////
    translate([inner_diameter_1/2,0,0])square([wall_thickness_1,length_1]);
    //barb 1

translate([outer_diameter_1/2,0,0])polygon(points=[[0,0],[0,(length_1/5)],[barb_
size_1,(length_1/5)]]);
    //barb 2

translate([outer_diameter_1/2,length_1*0.25,0])polygon(points=[[0,0],[0,(length_
1/5)],[barb_size_1,(length_1/5)]]);
    //barb 3

translate([outer_diameter_1/2,length_1*0.5,0])polygon(points=[[0,0],[0,(length_1
/5)],[barb_size_1,(length_1/5)]]);
    //////
    //length 2
    /////

translate([inner_diameter_2/2,length_1+mid_length,0])square([wall_thickness_2,le
ngth_2]);
    //rib 1

translate([outer_diameter_2/2,(length_1+mid_length+length_2),0])polygon(points=[
[0,0],[0,-1*(length_2/5)],[barb_size_2,-1*(length_2/5)]]);
    //rib 2
```

```
    translate([outer_diameter_2/2,(length_1+mid_length+length_2)-
length_2*0.25,0])polygon(points=[[0,0],[0,-1*(length_2/5)],[barb_size_2,-
1*(length_2/5)]]);
    //rib 3
    translate([outer_diameter_2/2,(length_1+mid_length+length_2)-
length_2*0.5,0])polygon(points=[[0,0],[0,-1*(length_2/5)],[barb_size_2,-
1*(length_2/5)]]);
}

rotate_extrude(angle = 360, convexity = 10) create_profile();
//create_profile();
```

balcony-storage - Project

Solar-equip-cover - 3D Object

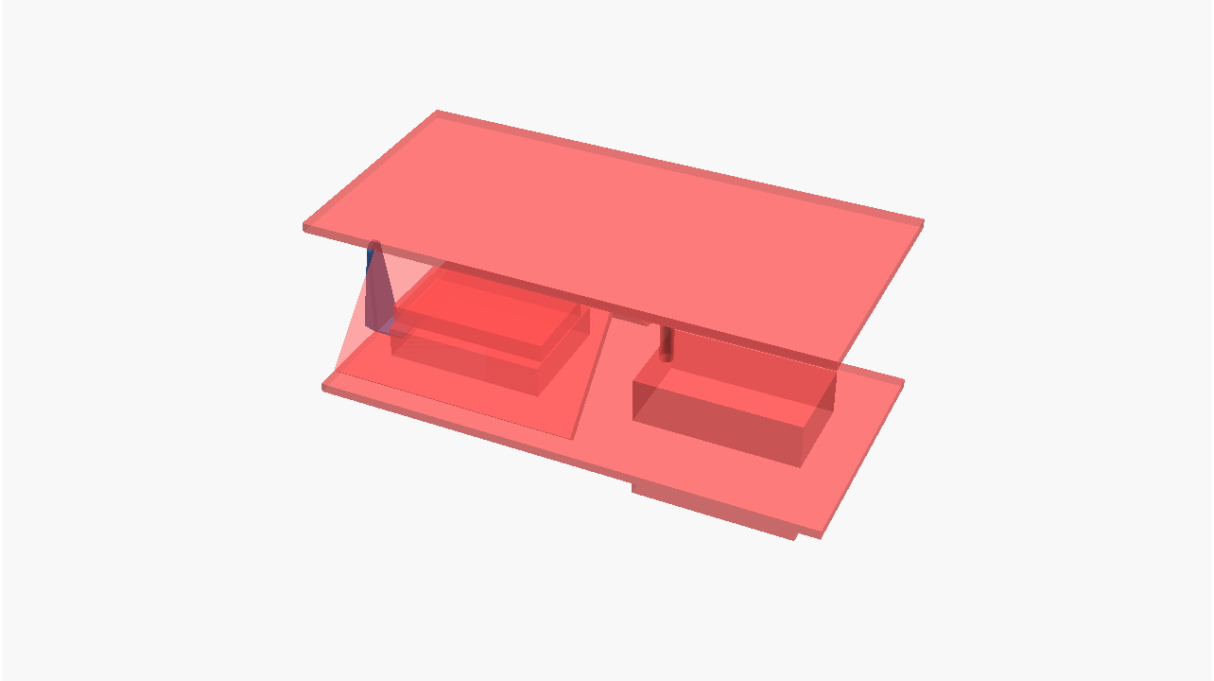

Figure 21. image

Listing 21. Openscad source

```
$fn=100;
doorX = 550 ; doorY = 290 ; doorZ = 21 ;
coverX = 497 ; coverY = 350 ; coverZ = 35 ;

module door (x,y,z) {
    Xwiggle = [.1,0,0] ;
    holeH = 15 ; holeD = 7 ;
    holeOff = [0,19,13] ;
    difference(){
        cube ([x,y,z]);
```

```
        translate([0,y,0] - [0,holeOff.y,-holeOff.z] - Xwiggle) rotate([0,90,0])
cylinder(h=holeH + 2*Xwiggle.x ,d=holeD);
    }
    translate([x,y,0] - [0,holeOff.y,-holeOff.z] - Xwiggle) rotate([0,90,0])
cylinder(h=holeH + 2*Xwiggle.x ,d=holeD);
}

module cover () {
    x = coverX ;
    y = coverY ;
    z = coverZ ;
    mount = [3,30,115] ;
    cube([x,y,z]);
    difference(){
        union() {
            translate([71,0,0]) cube(mount);
            translate([410,0,0]) cube(mount);
        }
        hull(){
            translate ([0,mount.y/2,100]) rotate([0,90,0])
cylinder(h=coverX,d=10);
            translate ([0,mount.y/2,85]) rotate([0,90,0])
cylinder(h=coverX,d=10);
        }
    }
}

//shelf
module shelf() {
    //bottom of shelf
    translate([0,0,-20]) cube([1100,530,20]);
    //top of shelf
    translate([0,0,370]) cube([1100,530,20]);
}
module components(){
        union(){
        //zendure
        translate([70,100,0]) cube([350,250,70]);
        //deye
        translate([70,120,75]) cube([350,185,40]);
        //battery
        battOffZ = [0,0,100] ;
        translate([630,110,0]+battOffZ) union(){
            batt = [360,200,290] ;
            translate([0,0,-batt.z]) cube(batt);
            translate([30,batt.y/2,0]) cylinder(h=110,d=30);
        }
    }
    //left side
    translate([0,0,0]){
```

```
        translate([0,0,0]) rotate([45,0,0]) door(doorX, doorY, doorZ);
        translate([(doorX-coverX)/2,doorY-100,240]) rotate([-45,0,0]) cover();
    }
    //right side
    *translate([555,0,0]){
        translate([(doorX-coverX)/2,doorY-100,240]) rotate([-45,0,0]) cover();
        translate([0,0,0]) rotate([45,0,0]) door(doorX, doorY, doorZ);
    }
}


//holder draft
module armL() {
    H = 200 ;
    holeD = 6 ;
    armD = 40 ;
    baseX = 80;
    difference () {
        hull(){
            translate([0,20,H]) rotate([0,90,0]) cylinder(h=10,d=armD);
            cube ([10,baseX,10]);
            translate ([-30,10,0]) cube ([30,60,10]);
        }
        translate([0, armD/2, H]) translate([-10,0,0]) rotate([0,90,0])
cylinder(h=50, d=6);
    }
    cube([70,baseX,5]);
}

#shelf();
#translate([11,50,0]) components();

translate([0,212,0]) armL();

*cover();
```

microshed - 3D Object

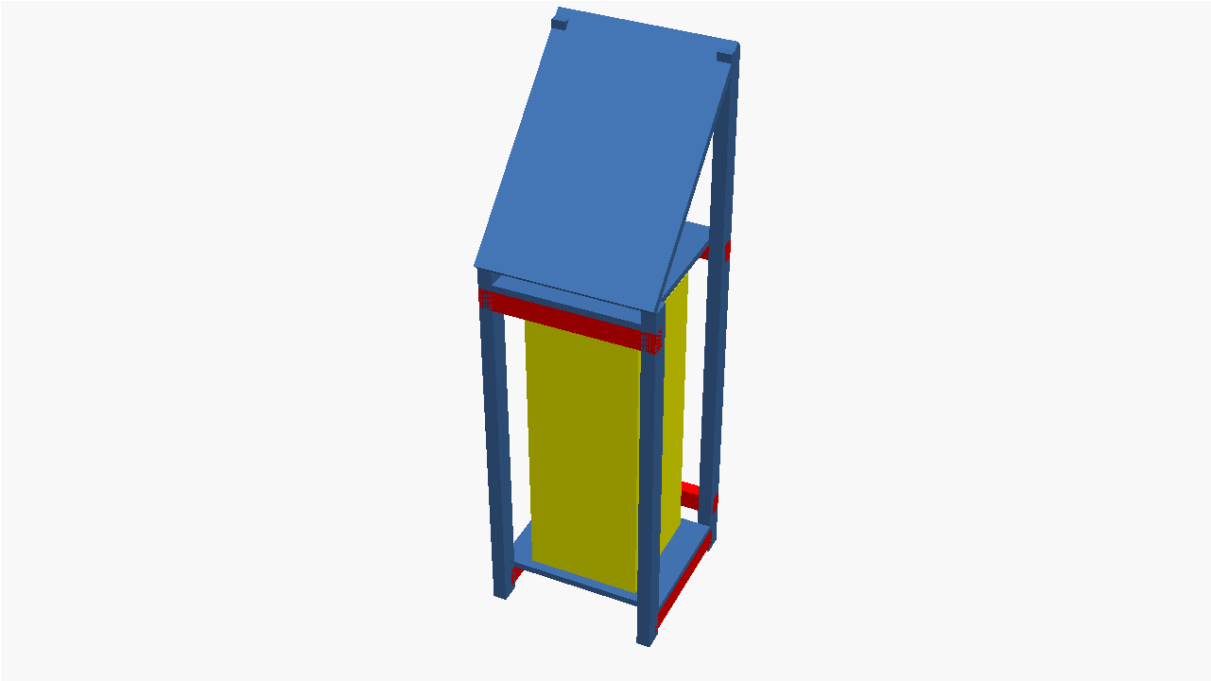

Figure 22. image

Listing 22. Openscad source

```
// A little cabinet for outside with a solar roof

// Variables
uprightX=40;
uprightY=60;
uprightIndent=uprightY/3;
IuprightIndent=uprightY-uprightIndent;
floorH=20; //thickness of board
floorOffset=uprightY+floorH+20; //height from ground the kaercher sits at
solarAngle=45; //front angle of solar roof
solarX=500; //panel Width
solarY=715; //panel Height
solarZ=25; //panel Depth
kaercherH=900; //approximate Kaercher bounding Height
kaercherX=330; //approximate Kaercher bounding Height
kaercherY=330;
kaercherHeadRoom=uprightY+20;
solarHpos=floorOffset+kaercherH+kaercherHeadRoom+uprightY;
//triangle sides
opposite = sin(solarAngle) * solarY;
adjacent = cos(solarAngle) * solarY;

// PARAMETRIC PART
//upright FL
cube([uprightX,uprightY,solarHpos]);
//upright FR
translate([solarX-uprightX,0,0]) cube([uprightX,uprightY,solarHpos]);
echo("2x front upright lengths= ",solarHpos);
```

```
// Panel / roof
translate([0,0,solarHpos])rotate([solarAngle,0,0])cube([solarX,solarY,solarZ]);

// Kaercher bounding box
translate ([solarX/2-kaercherX/2,adjacent/2-kaercherY/2,floorOffset])
color([1,1,0])cube([kaercherX,kaercherY,kaercherH]);

//upright BL
translate([0,adjacent-uprightY,0]) cube([uprightX,uprightY,solarHpos+opposite]);
//upright BR
translate([solarX-uprightX,adjacent-uprightY,0])
cube([uprightX,uprightY,solarHpos+opposite]);
echo("2x back upright lengths= ",solarHpos+opposite);

//floor
//sides
echo("FOR bottom shelf - cut in BL/BR/FR/FL at height of ", floorOffset-floorH-
uprightY," cut to depth of ", uprightIndent, " and length of ",uprightY," at the
long INSIDE side");
color([1,0,0]) translate([0,IuprightIndent,floorOffset-floorH-uprightY])
cube([uprightX,adjacent-(2*uprightY)+(2*uprightIndent),uprightY]);
color([1,0,0]) translate([solarX-uprightX,IuprightIndent,floorOffset-floorH-
uprightY]) cube([uprightX,adjacent-(2*uprightY)+(2*uprightIndent),uprightY]);
echo("2x bottom sides length= ",adjacent-(2*uprightY)+(2*uprightIndent));
//bottom floor board
translate([0,uprightY,floorOffset-floorH]) cube([solarX,adjacent-
(2*uprightY),floorH]);
echo("1x floor board XxY= ",solarX,adjacent-(2*uprightY));

//bottom back
echo("FOR bottom back - cut in BL/BR at height of ", floorOffset-
uprightY+(1.5*uprightY)," cut to depth of ", uprightX, " and length of
",uprightY," at the long BACK side");
color([1,0,0]) translate([0,adjacent-uprightX,floorOffset-
uprightY+(1.5*uprightY)]) cube([solarX,uprightX,uprightY]);
//top back
echo("FOR top shelf - cut in BL/BR/FR/FL at height of ",
floorOffset+kaercherH+kaercherHeadRoom-uprightY," cut to depth of ", uprightX, "
and length of ",uprightY," at the long OUTSIDE side");
color([1,0,0]) translate([0,adjacent-
uprightX,floorOffset+kaercherH+kaercherHeadRoom-uprightY])
cube([solarX,uprightX,uprightY]);
//top front
color([1,0,0]) translate([0,0,floorOffset+kaercherH+kaercherHeadRoom-uprightY])
cube([solarX,uprightX,uprightY]);
echo("3x back and front length= ",solarX);

//top shelf board
translate([uprightX,0,floorOffset+kaercherH+kaercherHeadRoom]) cube([solarX-
```

```
(2*uprightX),adjacent,floorH]);
echo("1x shelf board XxY= ",solarX-(2*uprightX),adjacent);
```

bcn3d - Project

bcn3d-cam-mount - 3D Object

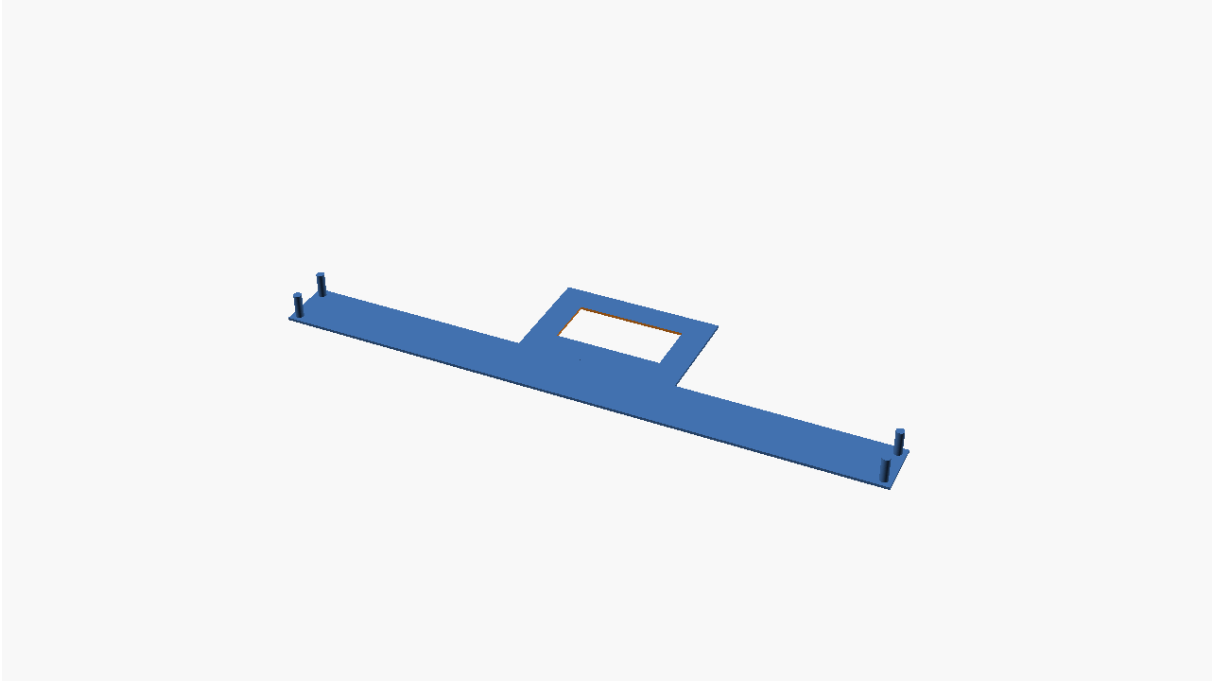

Figure 23. image

Listing 23. Openscad source

```
$fn = 100 ;
width = 220 ;
height = 15 ;
thickness = 1 ;
pinH = 10 ;
pinD = 3 ;
clipW = 40 ;
clipWout = 60 ;
clipH = 20 ;
clipHout = 40 ;
totWidth = width+2*pinD ;
totHeight = height+2*pinD ;

module pin() {
    cylinder(h=pinH,d=pinD);
}

translate([0, 0, 0]) cube([totWidth, totHeight, thickness]);
translate([pinD, pinD, 0]) {
```

```
    //move all pins at once while retaining their relative pos
    translate([0, 0, 0]) pin();
    translate([0, height, 0]) pin();
    translate([width, 0, 0]) pin();
    translate([width, height, 0]) pin();
}
difference() {
    translate([totWidth/2 - clipWout/2, totHeight, 0]) cube([clipWout, clipHout,
thickness]);
    translate([totWidth/2 - clipW/2, totHeight + clipH/2, -.1]) cube([clipW,
clipH, thickness + .2]);
}
```

bins - Project

draft-holder - 3D Object

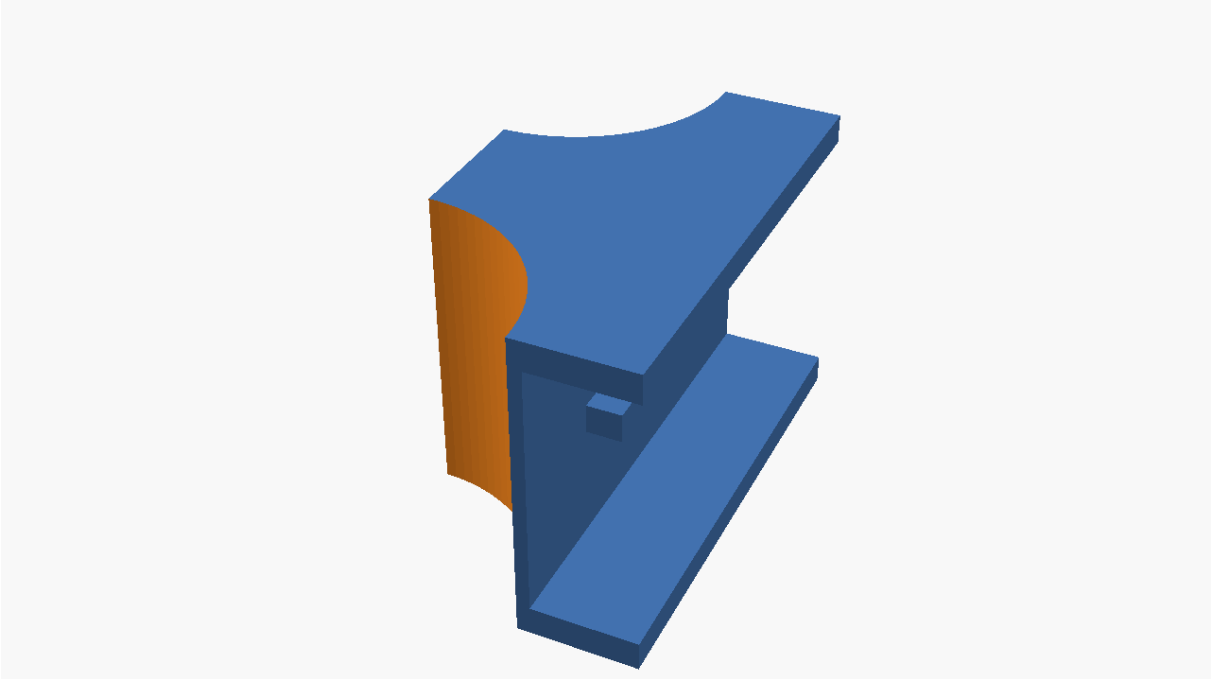

Figure 24. image

Listing 24. Openscad source

```
// a draft holder is work in progress

$fn = 100 ;
width = 40 ;
length = 15 ;
height = 27 ;
holderH = 2.5 ;
pinH = 3 ;
pinW = 20 ;
```

```
holderDepth = 9 ;
bridgeW = 12.3 ;
insertDepth = 10 ;
cylD = (width - bridgeW) ;

module bracket() {
    difference(){
        union(){
            cube([length,width,height]);
            translate([length,0,0]) cube([holderDepth,width,holderH]);
            translate([length,0,height-holderH])
cube([holderDepth,width,holderH]);
            translate([length,(width-pinW)/2,(height-holderH)/2])
cube([pinH,pinW,holderH]);
        }
        translate ([0,0,-.1]) cylinder(h=height+.2,d=cylD);
        translate ([0,width,-.1]) cylinder(h=height+.2,d=cylD);
        translate([-.1,(cylD+2*holderH)/2,holderH]) cube([insertDepth,bridgeW-
2*holderH,height-2*holderH]);
    }
}

bracket ();
```

strutinsert - 3D Object

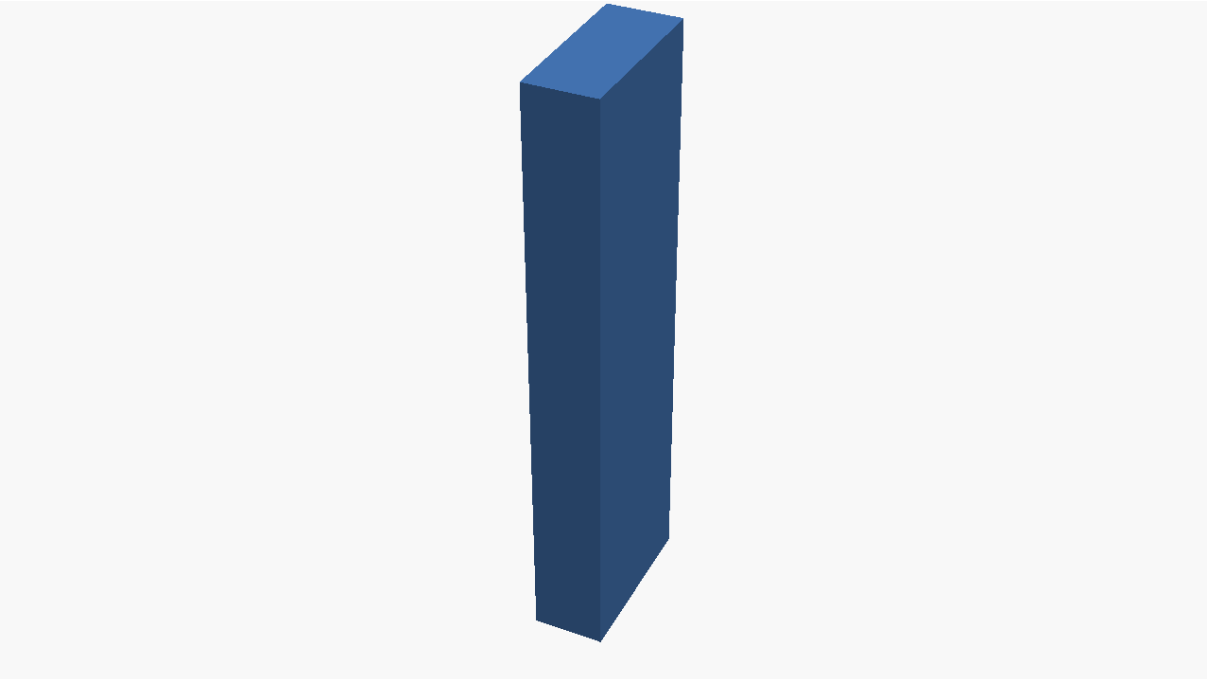

Figure 25. image

Listing 25. Openscad source

```
// A polygon extruded as a piece to repair a strut
```

```
linear_extrude(60) polygon(points=[[0,0],[7.2,0],[7.3,17],[-.1,17]]);
```

cmount - Project

2cmount - 3D Object

Figure 26. image

Listing 26. Openscad source

```
//
//  Double C-mount adapter
//  This is the thing required for my AmScope to get the
//  same focus plane for the camera as for the eyepiece
//
//
//

// you can tune this to change the focal plane
dist = 7.5;

depth = 4.0; // depth of threads

taper = 2.5;
ring = 28.0; // diameter of ring
knurls = 64;
kdia = 1.0;

include <threads.scad>
```

```
intersection
//union
() {
difference () {
    union () {
        english_thread (0.99, 32, (depth*2 + dist)/25.4);
        translate([0,0,depth]) cylinder(d1 = 25.4 + 0.3, d2 = ring, h=taper,
$fn=12*8);
        translate([0,0,depth+taper]) {
            cylinder(d = ring, h=dist-taper, $fn=12*8);
            for (n = [0:knurls-1]) rotate([0,0,(360/knurls)*n]) hull () {
                translate([ring/2,0,kdia/2]) sphere(d = kdia, h=dist-taper,
$fn=6);
                translate([ring/2,0,dist-taper-kdia/2]) sphere(d = kdia, h=dist-
taper, $fn=6);
            }
        }
    }
    translate([0,0,-1]) cylinder(d = 21.5, h=depth*2 + dist + 2, $fn=128);
}
translate ([0,0,-0.01]) cylinder(d1 = 25.4 - 2.0, d2 = 25.4 - 2.0 +
2*(depth*2+dist), h=depth*2+dist, $fn=12*8);
}
```

cookie-press - Project

star - 3D Object

Just a test to see if one can print shapes for a cookie press.

Figure 27. image

Listing 27. Openscad source

```
$fn=100;

points=8;
innerR=6;
outerR=12;
fittingD=51;
fittingH=3;

module Star(p=5, r1=6, r2=12) {
    s = [for(i=[0:p*2]) [(i % 2 == 0 ? r1 : r2)*cos(180*i/p), (i % 2 == 0 ? r1 :
r2)*sin(180*i/p)]];
    polygon(s);
}

difference() {
    cylinder(h=fittingH,d=fittingD);
    translate([0,0,-.1]) linear_extrude(fittingH+.2) Star(points, innerR,
outerR);
}
```

coords - Project

koord-rund - 3D Object

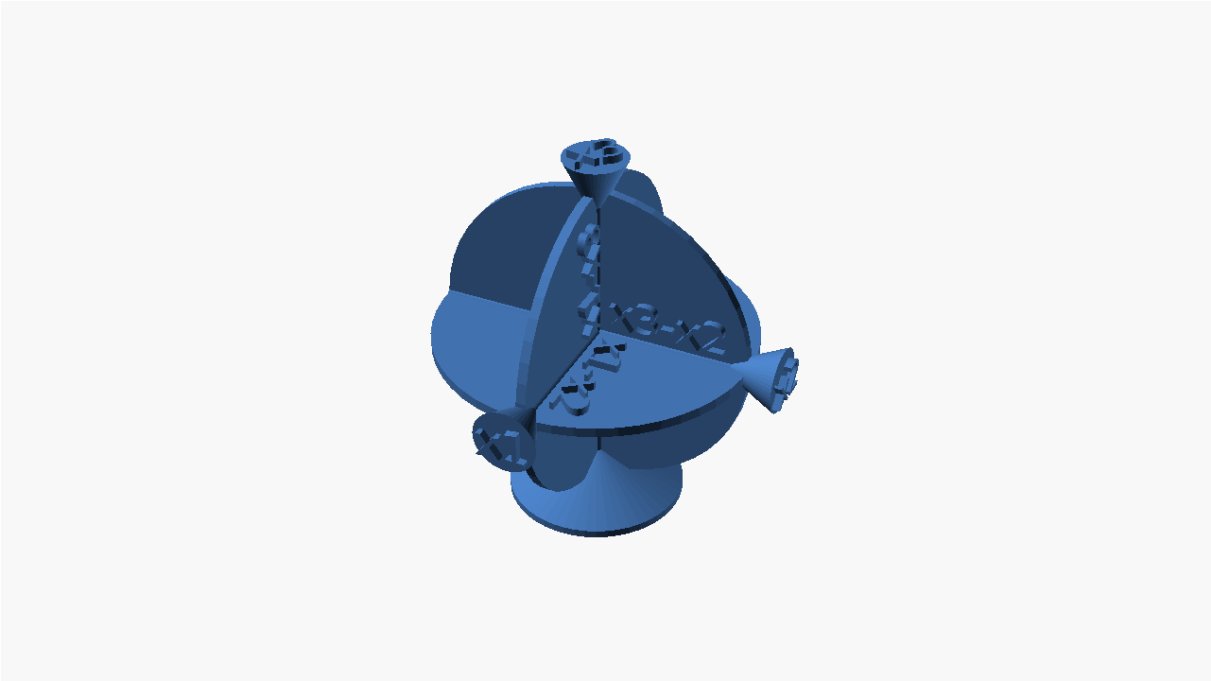

Figure 28. image

Listing 28. Openscad source

```
//Koordinatensytem
$fn=50;


//variablen
axisRadius=25;
discH=1.5;
axisD=2.7;
axisLabelD=10;
labelExt=1.2;
baseH=6;
baseD2=axisD;
baseD1=axisRadius/.9;

//ebene
module ebene(text1,text2,text1Pos) {
    cylinder(h=axisRadius*2,d=axisD,center=true);
    translate([0,0,axisRadius])
cylinder(h=axisLabelD,d2=axisLabelD,d1=.5,center=true);
    rotate([0,90,0]) cylinder(h=discH,r=axisRadius,center=true);
    translate([0,0,axisRadius+axisLabelD/2])
rotate([0,0,text1Pos])linear_extrude(labelExt)text(text1,halign="center",valign=
"center",axisLabelD/2);
    translate([discH/2,axisD,axisD]) rotate([90,0,90]) linear_extrude(labelExt)
text(text2,axisLabelD/2);
}

//ebenen ausgabe
```

```
rotate([90,0,0]) ebene("x1","x1-x3",0);
rotate([0,270,180]) ebene("x2","x1-x2",3*90);
rotate([0,0,270]) ebene("x3","x3-x2",1.5*90);

//Sockel
translate([0,0,-axisRadius]) cylinder(h=baseH,d1=baseD1,d2=baseD2,center=true);
difference() {
    translate([0,0,-axisRadius-baseH/2-labelExt/2])
cylinder(h=labelExt,d=baseD1,center=true);
    translate([0,0,-axisRadius-baseH/2]) rotate([180,0,0])
linear_extrude(labelExt)text("Fynn",halign="center",valign="center",8);
}
```

koordinaten - 3D Object

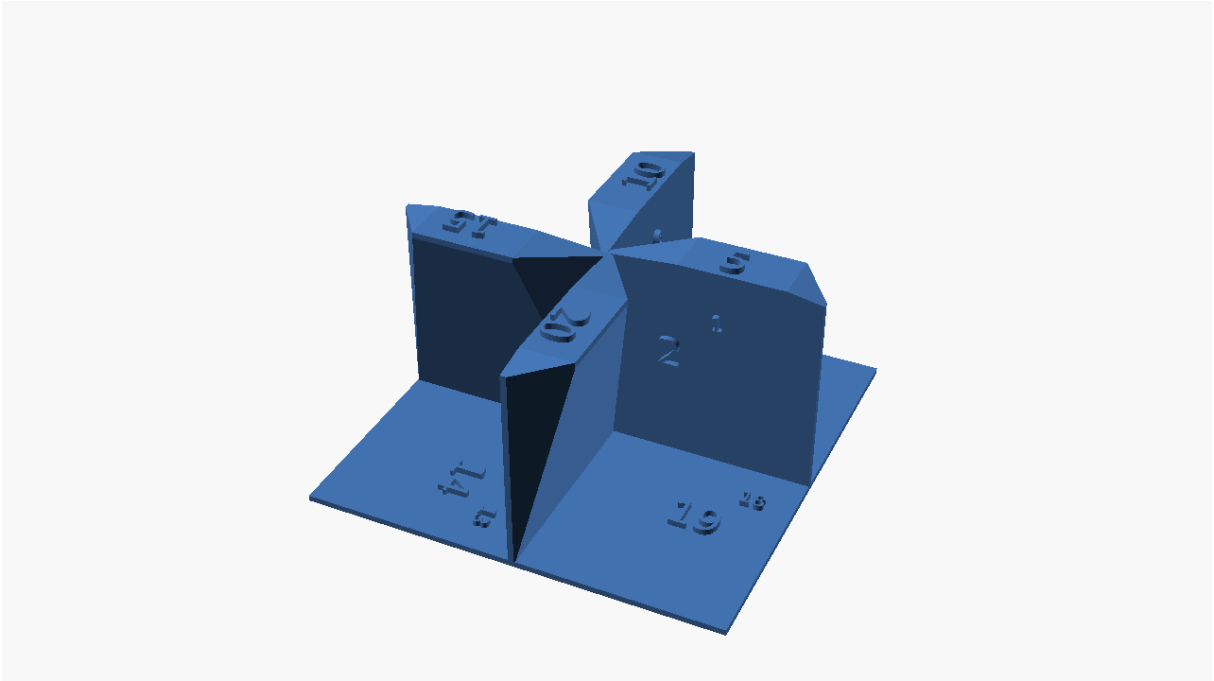

Figure 29. image

Listing 29. Openscad source

```
//koordinaten system
hoehe=70;
breite=70;
tiefe=70;
dicke=2;

//ein viertel
module viertel(text1,text2,text3,text4,text5) {
    cube([breite,tiefe,dicke]);
    hull(){
    cube([breite,dicke,tiefe]);
    translate([breite/3,0,hoehe])cube([breite/2,tiefe/4,dicke]);
```

```
    }
    //text1
    translate([breite/2,dicke,hoehe/2+hoehe/4]) rotate([90,0,0])
linear_extrude(dicke*2) scale([.5,.5,.5])text(text1);
    //text2
    translate([breite/4,dicke,hoehe/2]) rotate([90,0,0]) linear_extrude(dicke*2)
text(text2);
    //text3
    translate([breite/4,tiefe/2+tiefe/4,0]) rotate([0,0,90])
linear_extrude(dicke*2) scale([.5,.5,.5]) text(text3);
    //text4
    translate([tiefe/2,tiefe/2,0])rotate([0,0,90]) linear_extrude(dicke*2)
text(text4);
    //text5
    translate([breite/2,dicke,hoehe])rotate([0,0,0]) linear_extrude(dicke*2)
text(text5);
}
viertel("1","2","3","4","5");
rotate([0,0,90]) viertel("6","7","8","9","10");
rotate([0,0,180]) viertel("11","12","13","14","15");
rotate([0,0,270]) viertel("16","17","18","19","20");
```

desk - Project

fastener - 3D Object

This is a fastener for a writing Desk.
The idea is to add a magnet to hold it up and to print it so that it does not require a bearing.

- V1 is the first prototype for a first print test and fitting test
  - fits well and axle didn't print free so need update
  - The reason was that the axle required supports winside the part to build
  - Changed the axle to 45 degree angles to negate the need for supports
- V2 added a better axle but didn't get printed
- V3 added a better cutout and is printed
  - The cutout is currently a dummy pending getting the axle to work to try it out with magnets taped into place
  - axle prints freely so moving on to screw holes, magnets, and covers
- V4 Added final OCD logo and screw caps etc.
  - Mounted and working.

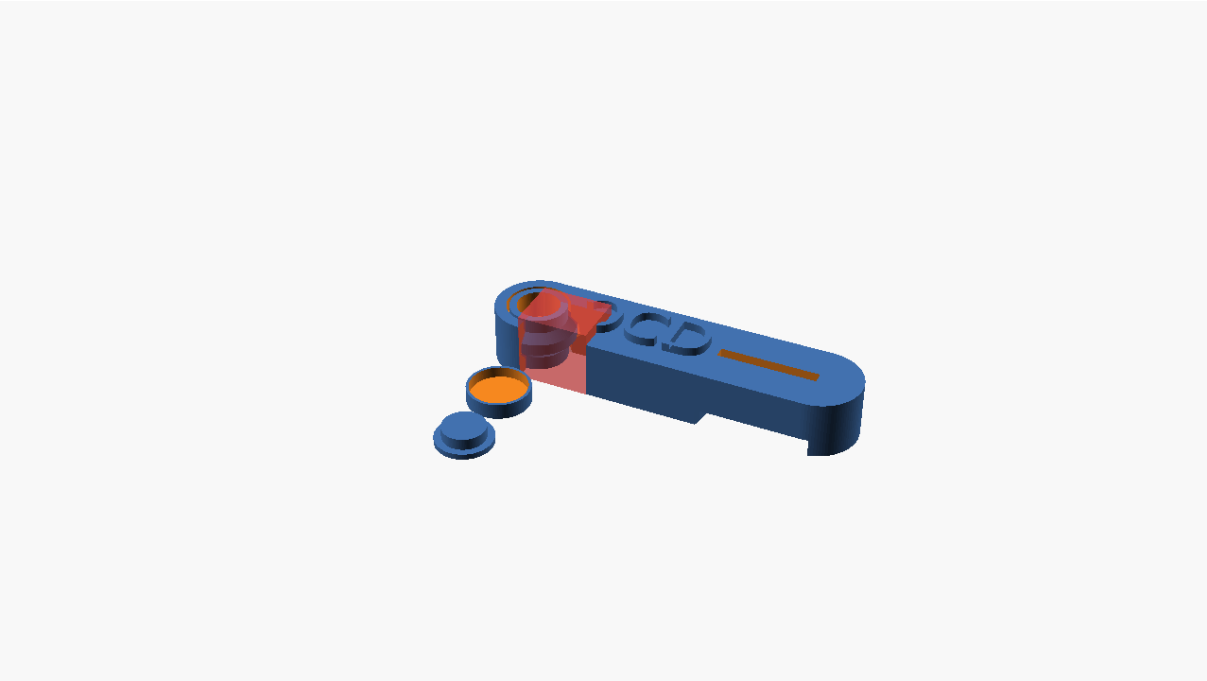

Figure 30. image

Listing 30. Openscad source

```
$fn=100;
mainLength=50;
mainD=15;
mainH=10;
axleD=10;
axleDout=axleD+3;
ringH=2;

magnetX=17;
magnetY=5;
magnetZ=2;

module axle(xxlX,xxlY) {
    translate([0,0,-xxlY/2])cylinder(h=mainH+xxlY,d=axleD+xxlX);
    translate([0,0,((mainH-ringH)/2)]) cylinder(h=ringH,d=axleDout+xxlX);
    translate([0,0,(mainH/2)-((axleDout-axleD)/2+ringH/2)])
cylinder(h=(axleDout-axleD)/2,d1=axleD+xxlX,d2=axleDout+xxlX);
    translate([0,0,(mainH/2)+(ringH/2)]) cylinder(h=((axleDout-
axleD)/2),d2=axleD+xxlX,d1=axleDout+xxlX);
}
module clip() {
    difference() {
        union(){
            hull(){
                cylinder(d=mainD,h=mainH);
                translate([mainLength,0,0]) cylinder(d=mainD,h=mainH);
            }
            translate([7,-3.5,mainH]) linear_extrude (height=1.5) {
```

```
                text("OCD",size=8);
            }
        }
        //magnet
        translate([mainLength-magnetX,-magnetZ/2,(mainH-magnetY)/2+1])
cube([magnetX,magnetZ,magnetY+10]);
        //holder
        holderW=19;
        holderRin=33;
        holderRout=holderRin+holderW;
        difference(){
            translate([0,0,-.1]) cylinder(h=3+.1,r=holderRout);
            translate([0,0,-.11]) cylinder(h=3+.22,r=holderRin);
        }
    }
}
module magnetCap(){
    //magnet cap
    difference(){
        cylinder(h=2.8,d=11);
        translate([0,0,-.1]) cylinder(h=2,d=10);
    }
}
module screwCap() {
    //screwcap axle
    cylinder(h=2,d=7.5);
    translate([0,0,2]) cylinder(h=1,d=axleD);
}

//add the clip
difference () {
    clip();
    axle(1,1);
    //REMOVE FOR PRINT!!!! JUST A CUTOUT FOR DEMO
    #translate ([0,-mainD/2-.1,-.01]) cube([mainLength/4,mainH,12]);
}

//add the axle and drill a hole in it for a srew
difference(){
    axle(0,0);
    translate([0,0,-.05]) cylinder(h=mainH+.1,d=4);
    translate([0,0,mainH/2]) cylinder(h=(mainH/2)+.1,d=7.5);
    //next two lines just a visual
    //#translate([0,0,mainH+2]) screwCap();
    //#translate([42,0,-.5]) magnetCap();
}
translate([0,-27,3]) rotate([0,180,0]) screwCap();
translate([0,-15,3]) rotate([0,180,0]) magnetCap();
```

fritzring - Project

Freez-ring - 3D Object

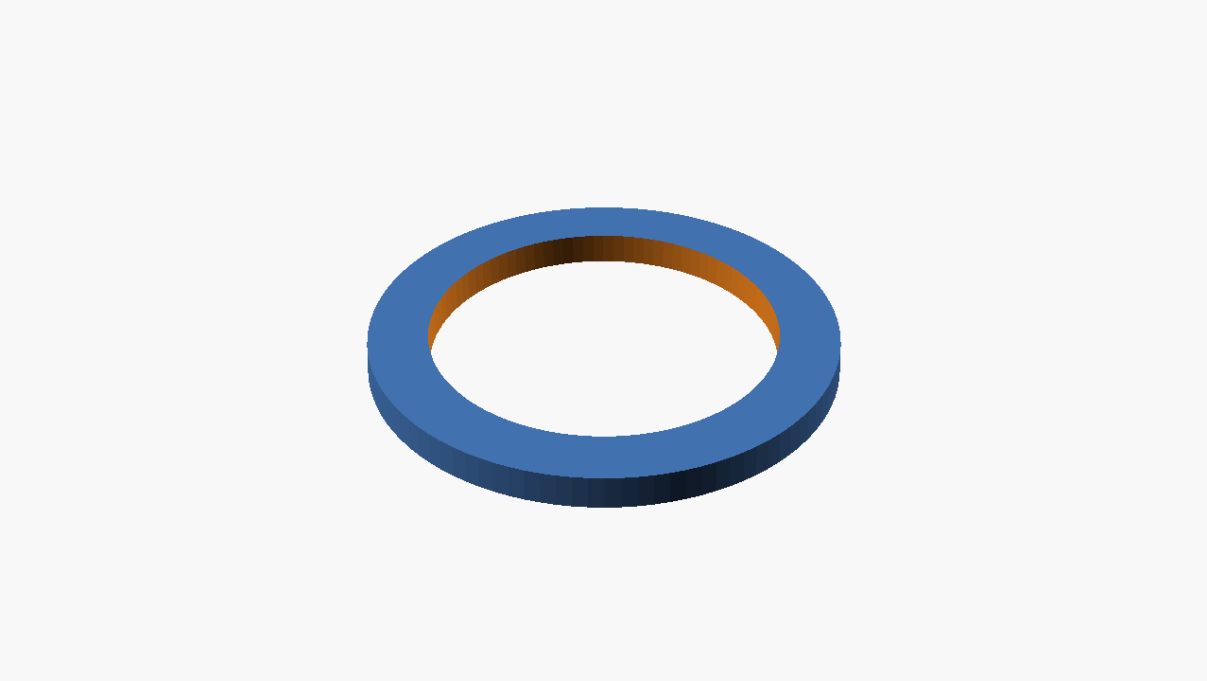

Figure 31. image

Listing 31. Openscad source

```
$fn=100;
height=5;
innerD=54 ;
upperOuterD=73;
lowerOuterD=72.5;
module flange () {
    difference(){
        translate([0,0,.5]) cylinder(d2=lowerOuterD,d1=upperOuterD,h=height);
        cylinder(d=innerD,h=height+1);

    }
}
flange();
```

fritzcolaadapter - 3D Object

Figure 32. image

Listing 32. Openscad source

```
$fn=300;
//35mm measured fromn the 3 initial rings plus 2 mm
height=35;
//following testing only
height=5;
innerD=61.8 ;
// outerD=66 is good enough for the upper 1/3 part of the zoe can holder
// outerD=65 is good enough for the middle 1/3 part of the zoe can holder
// inner D 61 sticks just a slight bit on the fritzcola bottles but works
// turns out that while innerD 61.2 is fine for the one bottle I tested with the
others are wider
// testing with .5mm extra
// and exchange upper and lower D to make it grow thinnner upwards
// should probably add three indents to make it fit even better but hey....
iterative :-)
// 61.7 sticks just slightly so adding.1mm
lowerOuterD=64;
upperOuterD=66.5;
module flange () {
    difference(){
        translate([0,0,.5]) cylinder(d2=lowerOuterD,d1=upperOuterD,h=height);
        cylinder(d=innerD,h=height+1);

    }
}
flange();
```

geo-test - Project

geoTest - 3D Object

This is not one of mine and I just kiked it as a good example.
I did correct some mistakes in it though.

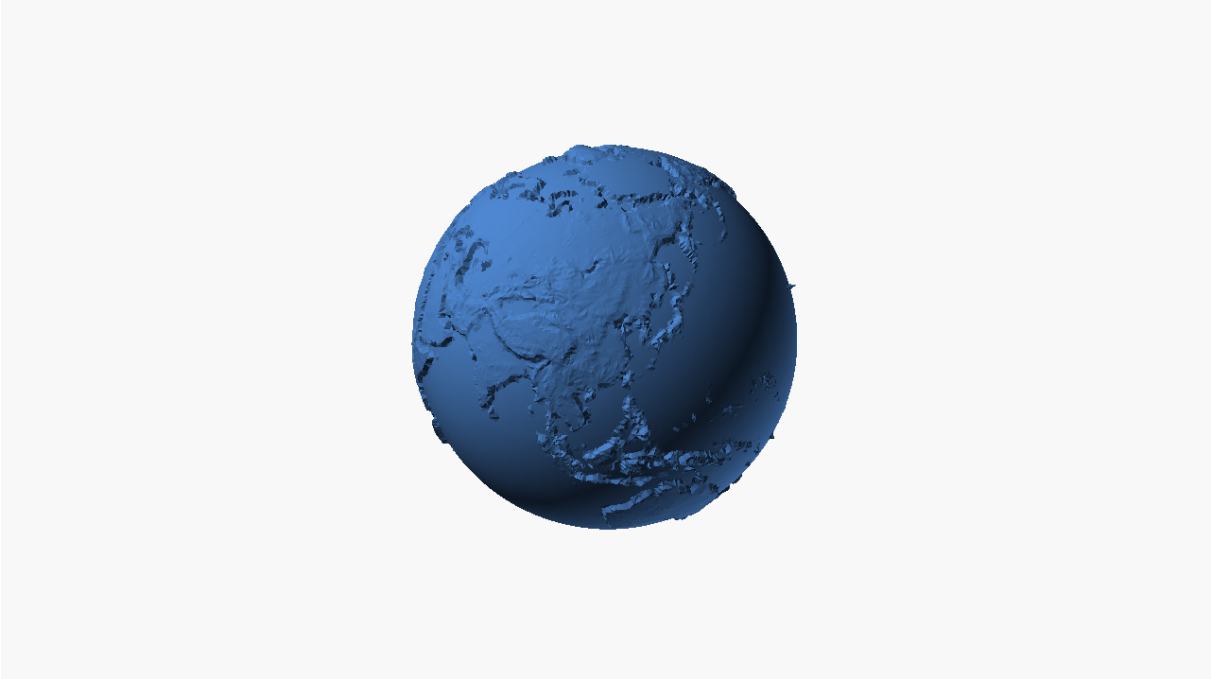

Figure 33. image

Listing 33. Openscad source

```
// Geody Planet 1 - SCAD
// Geody - https://www.geody.com/
// OpenSCAD - http://www.openscad.org/

wwrad=40; // Radius of the Planet
wrad=wwrad/20; // Radius of the Spot
wradp=wwrad-wrad/2; // Distance of the Spot from the center of the Planet
wres=50; // Resolution of the Spot

latx=48.782345; lonx=9.180819;

rotate(a=[0,0,270]) { import("geody_earthmap.stl", convexity=4); }
// download from https://www.geody.com/geody_earthmap.stl
// sphere(r=wwrad, $fn=wres); // Test Planet

translate([(-wradp)*cos(latx)*cos(lonx),(-
wradp)*cos(latx)*sin(lonx),wradp*sin(latx)]){sphere(r=wrad, $fn=wres);}
```

kitchen-door - Project

KitchenDoorHoleStopper - 3D Object

Figure 34. image

Listing 34. Openscad source

```
//plug for door hinge mounting hole (WHITE)
// door replaced by sliding glass door 27/11/2021
totDepth=15;
insertDiameter=7;
lidDiameter=14;
lidHeight=1;
$fn=100;
color ([1,1,1]) {
    cylinder(h=totDepth,d=insertDiameter);
    cylinder(h=lidHeight,d2=lidDiameter,d1=lidDiameter-lidHeight);
}
```

skirtingpatch - 3D Object

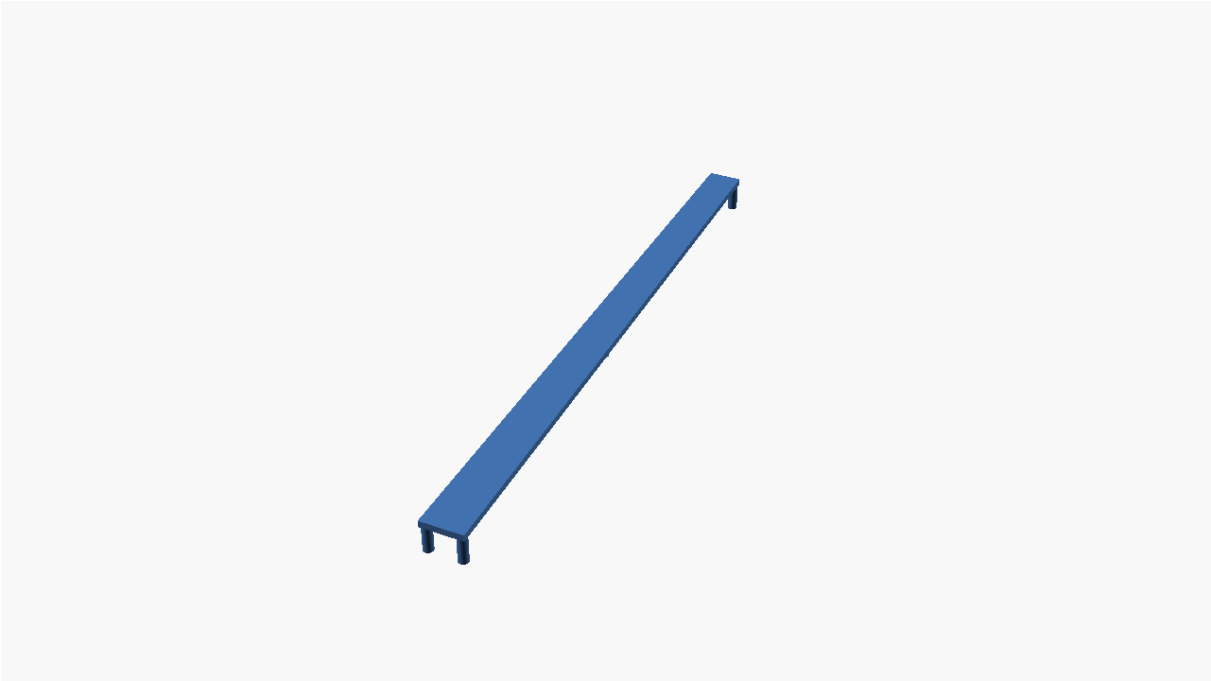

Figure 35. image

Listing 35. Openscad source

```
//approximation of the ends
$fn=100;
length=191.5;
height=10;
topDepth=2;
end1W=13;
end2W=10;
wiggle=.1;
legD=3;
legR=legD/2;
translate([0,0,height-topDepth]) hull(){
    rotate([90,0,0]) cube([end1W,topDepth,wiggle]);
    translate([0,length-wiggle,0]) rotate([90,0,0])
cube([end2W,topDepth,wiggle]);
}

//end1
translate([legR,legR,0]) cylinder(h=height,r=legR);
translate([end1W-legR,legR,0]) cylinder(h=height,r=legR);

//end2
translate([legR,length-legR-wiggle,0]) cylinder(h=height,r=legR);
translate([end2W-legR,length-legR-wiggle,0]) cylinder(h=height,r=legR);

//middle
translate([((end2W+end1W)/2)/2,length/2,0]) cylinder(h=height,r=legR);
```

strikeplate - 3D Object

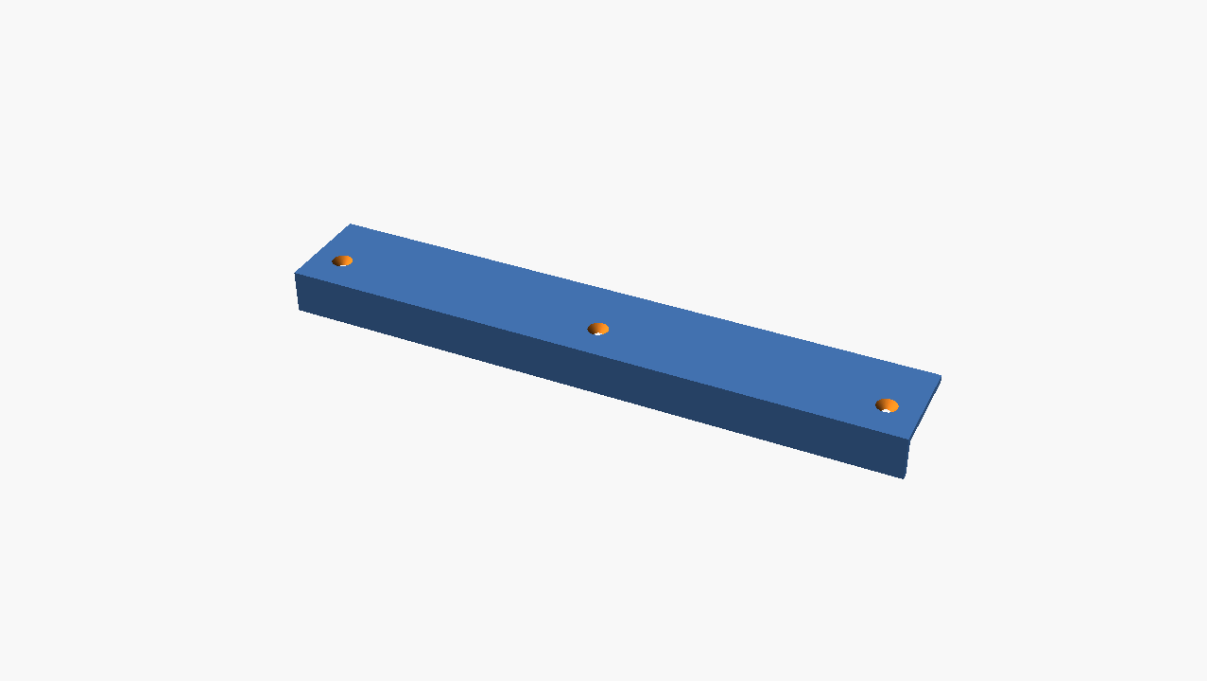

Figure 36. image

Listing 36. Openscad source

```
$fn=100;
SPlength=170;
SPwidth=28;
SPmaterialStrength=2;

module strikePlate () {
    cube ([SPlength,SPwidth,SPmaterialStrength]);
    translate ([0,0,-10])
        cube ([SPlength,SPmaterialStrength,10]);
}

module screw () {
    *cylinder(h=8,d=3);
    cylinder(h=3,d1=2,d2=7);
}

difference(){
    strikePlate();
    translate([8.5,10.5,-0.1]) screw();
    translate([85,10.5,-0.1]) screw();
    translate([SPlength-8.5,10.5,-0.1]) screw();
}
```

led-Casing - Project

CasingLED - 3D Object

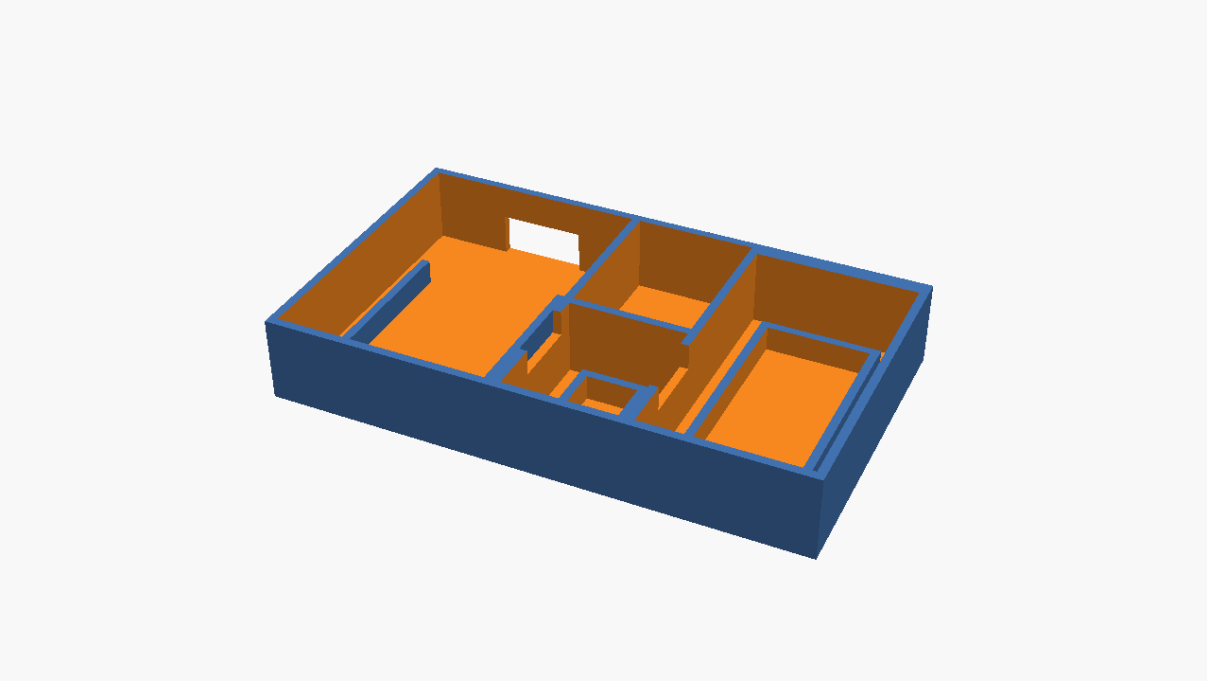

Figure 37. image

Listing 37. Openscad source

```
// OPENSCA Model for encloser for Tine's table
// Curently 3 devices Waatuino, 3.3v to 5v and esp wemos d1 mini
$fn=100;
wiggle=0.01;
module wemos(){
    difference(){
        union(){
            //wemos d1 mini 26mmx35mm h7mm
            cube([26,35,10]);
            translate([9,32,0]) cube([10,5,5]);
        }
        translate([3,5,-wiggle]) cube([1,20,3+wiggle]);
        translate([21,5,-wiggle]) cube([1,20,3+wiggle]);
    }
}
module v5v3(){
    union(){
        difference(){
            //volatege level shifter 5v 3v 14mmx16mm h7mm
            cube([14,16,10]);
            translate([3,3,-wiggle]) cube([8,10,3+wiggle]);
        }
        translate([4,4,0]) cube([6,8,3+wiggle]);
    }
}
```

```
module blanker(){
    //volatege level shifter 5v 3v 14mmx16mm h7mm
    cube([14,18,10]);
}
module wattuino(){
    //wattuino arduino 5v clone 22mmx32mm h7mm
    union(){
        difference(){
            cube([19,34,10]);
            translate([2.5,4,-wiggle]) cube([14,26,3+wiggle]);
        }
        translate([3.5,5,0]) cube([12,24,3+wiggle]);
    }
}
module casing(){
    //outer casing
    cube([63,37,10]);
}
module cabling(){
    //cabling boom
    cube([50,8,3.01]);
}
module enclosure(){
    //outer casing and substract
    difference(){
        casing();
        translate([1,1,1]) wemos();
        translate([28,1,1]) v5v3();
        translate([43,1,1]) wattuino();
        translate([28,18,1]) blanker();
        translate([7,7,7]) cabling();
        }
}

enclosure();
```

midleton - Project

internal-volume - 3D Object

the internal Volume of a presentation box to test ideas on.

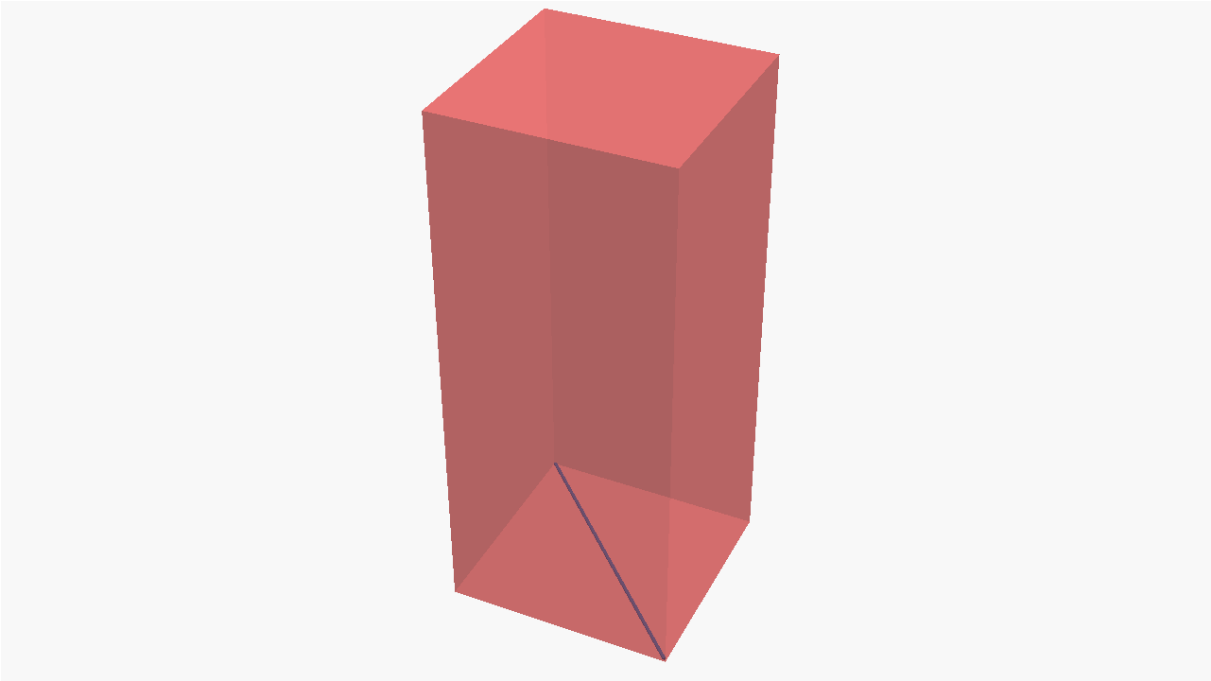

Figure 38. image

Listing 38. Openscad source

```
//inside of midleton wooden box with double doors
totHeight = 260 ;
totWidth = 111.1 ;
totDepth = 108.5 ;
dimensions = [ totWidth, totDepth, totHeight] ;
wiggle = [1.25, 1, 1] ;
volume = dimensions - wiggle ;

//inner structure definitions
structDivs = 20 ;
structStrength = 1 ;
structH = totHeight/structDivs ;
post = [structStrength, structStrength, structH] ;
block = [structStrength, structStrength, structStrength] ;

//Volume Visual
#cube( dimensions );

FL = [0, 0, 0];
FR = [volume.x - post.x, 0, 0];
BR = [volume.x - post.x, volume.y - post.y, 0];
BL = [0, volume.y - post.y, 0];

i=0;
translate(wiggle/2) *union() {
    translate ([0,0,i]) {
        //left and right sides
        hull() {
```

```
            translate( FR ) cube( post );
            translate( BR ) cube( post );
        }
        hull() {
            translate( FL ) cube( post );
            translate( BL ) cube( post );
        }
        //Base plate
        hull() {
            cube([volume.x, post.y, post.y]) ;
            translate ([0, 0, 0] + BL) cube([volume.x, post.y, post.y]) ;
        }
    }
}

translate(wiggle/2)
hull(){
    translate(FR) cube(block);
    translate(BL) cube(block);
}
```

midletoninset - 3D Object

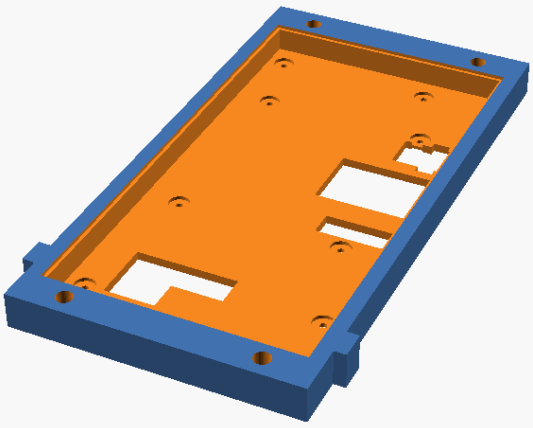

Figure 39. image

Listing 39. Openscad source

```
//This is an inlet for a whiskey presentation box from Midleton
$fn=50;
//Lower Notches
LowerNotchDepth=3.5;
```

```
LowerNotchLength=8;
LowerRNotchLengthOffset=15;
LowerLNotchLengthOffset=14.3;
module LLnotch(LowerLNotchLengthOffset){
    //Lower Left Notch
    translate([-LowerNotchDepth,LowerLNotchLengthOffset,0])
        cube([LowerNotchDepth,LowerNotchLength,BoxHeight]);
}
module LRnotch(LowerRNotchLengthOffset){
    //Lower Right Notch
    translate([BoxWidth,LowerRNotchLengthOffset,0])
        cube([LowerNotchDepth,LowerNotchLength,BoxHeight]);
}

//Variables for screen
ScreenTopY=75;
ScreenTopX=141;
ScreenTopZ=1;
ScreenEdge=1;
ScreenMaxDepth=7;
module waveshareHDMIscreen(wiggle){
    //full hd screen top face
    //and yes it has rounded corners but let's just start simple.
    union(){
        //Screen dimensions
        cube([ScreenTopY,ScreenTopX,ScreenTopZ+wiggle]);
        translate([ScreenEdge,ScreenEdge,-ScreenMaxDepth])
            cube([ScreenTopY-(2*ScreenEdge),ScreenTopX-
(2*ScreenEdge),ScreenMaxDepth+wiggle]);
        //connecting cable at the edge.
        translate([57,0,-ScreenMaxDepth]) cube([7,7,ScreenMaxDepth]);
        //USB for touch with offseted connector - wiggle through
        translate([12,19,-ScreenMaxDepth-10])
            cube([30,9,ScreenMaxDepth+10]);
        translate([12,10,-ScreenMaxDepth-10])
            cube([15,12,ScreenMaxDepth+10]);
        //USB for power - wriggle through
        translate([65,95,-ScreenMaxDepth-10])
            cube([5,15,ScreenMaxDepth+10]);
        translate([57,97,-ScreenMaxDepth-10])
            cube([17,11,ScreenMaxDepth+10]);
        //HDMI connector - Wriggle through might not work... might have to make
hole larger
        translate([44,72,-ScreenMaxDepth-10])
            cube([30,20,ScreenMaxDepth+10]);
        //Audio?
        translate([51,56.75,-ScreenMaxDepth-4])
            cube([23,7.5,ScreenMaxDepth+4]);
        //The screw holes
        Standoffs();
```

```
        //The mounting holes for the displaycover
        translate([0+10,0-5,-ScreenMaxDepth-6])
            cylinder(h=20,d=4.8);
        translate([0+10,ScreenTopX+5,-ScreenMaxDepth-6])
            cylinder(h=20,d=4.8);
        translate([ScreenTopY-10,0-5,-ScreenMaxDepth-6])
            cylinder(h=20,d=4.8);
        translate([ScreenTopY-10,ScreenTopX+5,-ScreenMaxDepth-6])
            cylinder(h=20,d=4.8);
    }
}
StandoffDepth=9;
StandoffSpace=1;
StandoffScrewHead=2;
module HolePeg(offset1){
    //standoff
    translate([0,0,-StandoffDepth+1]+offset1)
        cylinder(h=StandoffDepth-1,r=3.05);
    //screwwshaft
    translate([0,0,-StandoffDepth-StandoffSpace+1]+offset1)
        cylinder(h=StandoffDepth+StandoffSpace-1,r=1);
    //Screw head
    translate([0,0,-StandoffDepth-StandoffSpace-StandoffScrewHead+1]+offset1)
        cylinder(h=StandoffScrewHead,r=3);
}
module Standoffs(){
    //Outside holes
    //one
    *HolePeg([6,9,0]);
    HolePeg([6.5,9.75,0]);
    //the rest
    HolePeg([69,22,0]);
    HolePeg([6,132.5,0]);
    HolePeg([53,132.5,0]);

    //inside holes
    HolePeg([11.5,52.5,0]);
    HolePeg([60.5,52.5,0]);
    HolePeg([60.5,110.5,0]);
    HolePeg([11.5,110.5,0]);
}

// Midleton box measurements
//Real total Height
//BoxHeight=61;
//Display inset Height
BoxHeight=10.5;
//testprint
//BoxHeight=8.5;
BoxWidth=83.8;
```

```
LowerPartLength=162.5;
//testing value
//LowerPartLength=50;
LowerPartWallThickness=1.5;
LowerPartFloorThickness=1.5;
module Displaymodule() {
    //Lower part of the box
    difference(){
        //Outercube
        cube([BoxWidth,LowerPartLength,BoxHeight]);
        //subtract for inner space

*translate([LowerPartWallThickness,LowerPartWallThickness,LowerPartFloorThicknes
s])
            cube([BoxWidth-2*LowerPartWallThickness,LowerPartLength,BoxHeight-
(2*LowerPartFloorThickness)]);
    }
    LLnotch(LowerLNotchLengthOffset);
    LRnotch(LowerRNotchLengthOffset);
}
//Displaymodule();
//Standoffs();
//waveshareHDMIscreen();

// put it all together
difference(){
    Displaymodule();
    //Screen
    translate([(BoxWidth-ScreenTopY)/2,(BoxWidth-ScreenTopY)/2+6,BoxHeight-
ScreenTopZ])
        waveshareHDMIscreen(.1);
    //for testprint only
    *translate([-10,10,2.5])cube([100,130,15]);
    *translate([10,-10,2.5])cube([65,160,15]);
}
//remove for print... only for animation
*translate([((BoxWidth-ScreenTopY)/2),(BoxWidth-ScreenTopY)/2+6,(BoxHeight-
ScreenTopZ)+30*(1-$t)]) waveshareHDMIscreen(0);
```

test - 3D Object

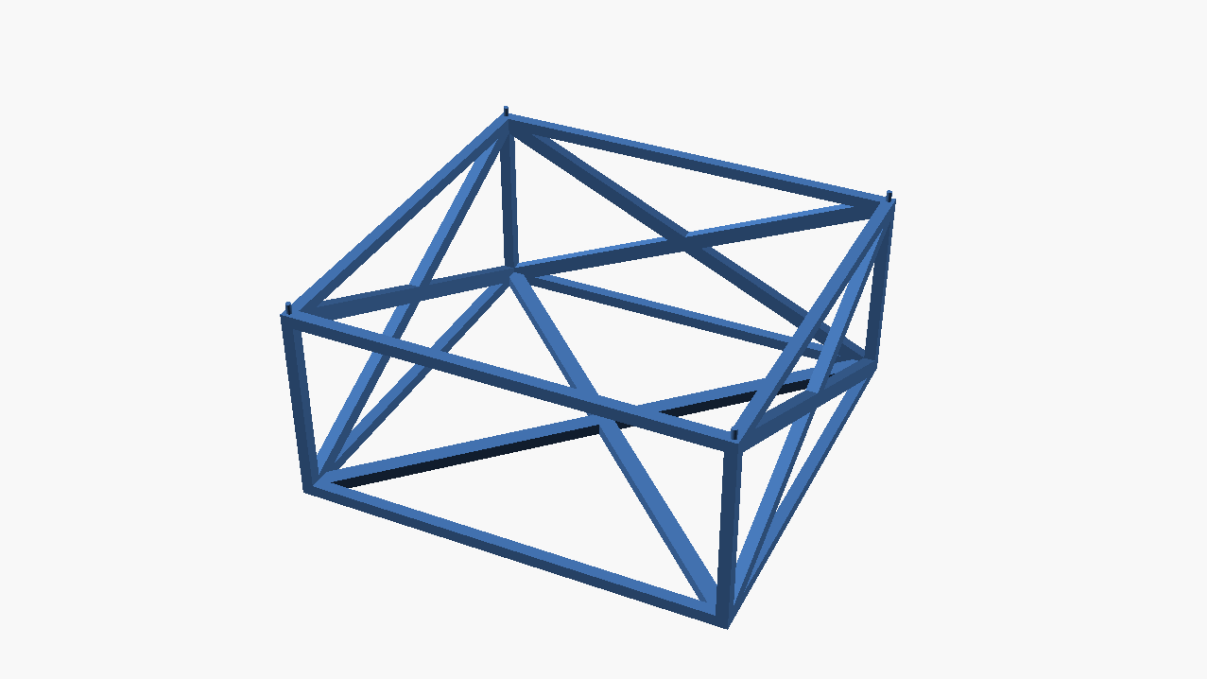

Figure 40. image

Listing 40. Openscad source

```
// This scad script creates a cubic structure based on cube dimensions
// based on uprights and cross connexions from a list of lists of corners
$fn=100;
//inside of midleton wooden box with double doors
totHeight = 260 ;
totWidth = 111.1 ;
totDepth = 108.5 ;
dimensions = [ totWidth, totDepth, totHeight] ;
wiggle = [1.25, 1, 1] ;
volume = dimensions - wiggle ;

// Corner strength
Blob = [3,3,3] ;
// Outside dimensions
//DIM = [100, 200, 200] ;
DIM = volume - [0,0,4*(volume.z/5)];

module qubicStruct(Blob, DIM) {
    // relative coordinates of the 8 corners minus the corner strength
    // eg. Bottom Back Left= BBL, or TFR= Top Front Right
    // This makes reading the list of lists easier
    BFL = [0, 0, 0] ;
    BFR = [DIM.x - Blob.x, 0, 0] ;
    BBR = [DIM.x - Blob.x, DIM.y -Blob.y, 0] ;
    BBL = [0, DIM.y -Blob.y, 0] ;
    TOP = [0, 0, DIM.z -Blob.z] ;
    TFL = BFL + TOP ;
    TFR = BFR + TOP ;
```

```
TBR = BBR + TOP ;
TBL = BBL + TOP ;

// List of top corners
topCorners = [
    TFL,
    TFR,
    TBL,
    TBR
];
bottomCorners = [
    BFL,
    BFR,
    BBL,
    BBR
];
// List of list of pairs of 3D coordinates with comment
coordinatePairs = [
    [BFL, BFR, "bottom"],
    [BFR, BBR, "bottom"],
    [BBR, BBL, "bottom"],
    [BBL, BFL, "bottom"],
    [BBL, BFR, "bottom cross"],
    [BFL, BBR, "bottom cross"],
    [BFR, TFR, "upright"],
    [BFL, TFL, "upright"],
    [BBR, TBR, "upright"],
    [BBL, TBL, "upright"],
    // top frame (ONLY needed if TOP structure to stabilize
    [TFL, TFR, "top"],
    [TFR, TBR, "top"],
    [TBR, TBL, "top"],
    [TBL, TFL, "top"],
    [BFL, TBL, "left cross"],
    [TFL, BBL, "left cross"],
    [BFR, TBR, "right cross"],
    [TFR, BBR, "right cross"],
    [BBR, TBL, "back cross"],
    [TBR, BBL, "back cross"],
];

module createHull(coord1, coord2) {
    hull() {
        translate(coord1) cube(Blob);
        translate(coord2) cube(Blob);
    }
}
difference() {
    // Create hulls around pairs of 3D coordinates
    for (pair = coordinatePairs) {
```

```
            createHull(pair[0], pair[1]);
        }
        // add mounting post hole
        for (corner = bottomCorners) {
            translate(corner + [Blob.x/2,Blob.y/2,-.1]) cylinder(h = Blob.z, d =
Blob.x - 1.4);
        }

    }
    // add mounting posts
    for (corner = topCorners) {
        translate(corner + [Blob.x / 2, Blob.y / 2, Blob.z]) cylinder(h = Blob.z
- .5, d = Blob.x - 1.6);
    }
}

qubicStruct(Blob, DIM);
```

modelTruckRepair - Project

steeringaxle - 3D Object

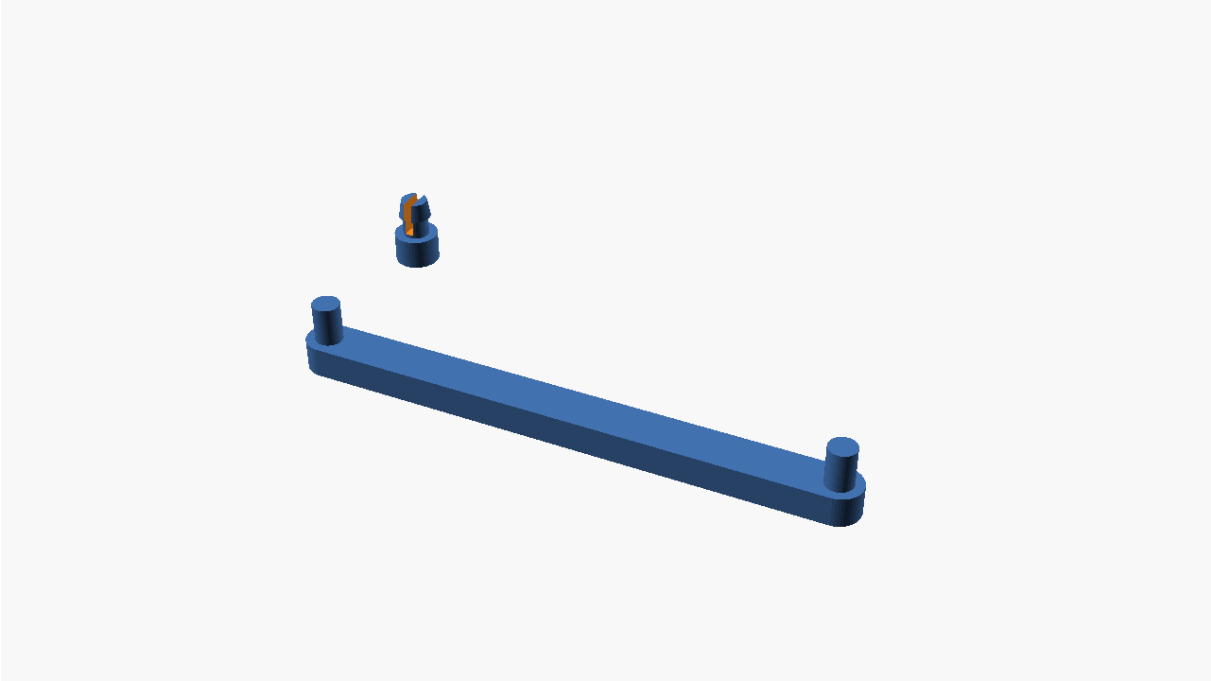

Figure 41. image

Listing 41. Openscad source

```
$fn=50;

module tabbedCylinder(){
    difference(){
```

```
        union (){
            cylinder(h=2,d1=3,d2=3);
            cylinder(h=4.6,d1=1.8,d2=1.8);
            translate ([0,0,3.4]) cylinder(h=1.2,d1=2.2,d2=1.8);
        }
        translate ([0,0,3.5]) cube([.6,2.5,2.5],center=true);
    }
}
module EndCylinder(){
        union (){
            cylinder(h=2,d1=3,d2=3);
            cylinder(h=4.6,d1=1.8,d2=1.8);
        }
}

module steeringAxle(){
    //axis
    translate ([1.5,0,0]) cube([34,3,2]);
    //connector
    translate ([1.5,1.5,0]) EndCylinder();
    //connector
    translate ([35.5,1.5,0]) EndCylinder();
}

steeringAxle();
translate([0,15,0]) tabbedCylinder();
```

monitor - Project

buttonBack - 3D Object

Button backing for the monitor.

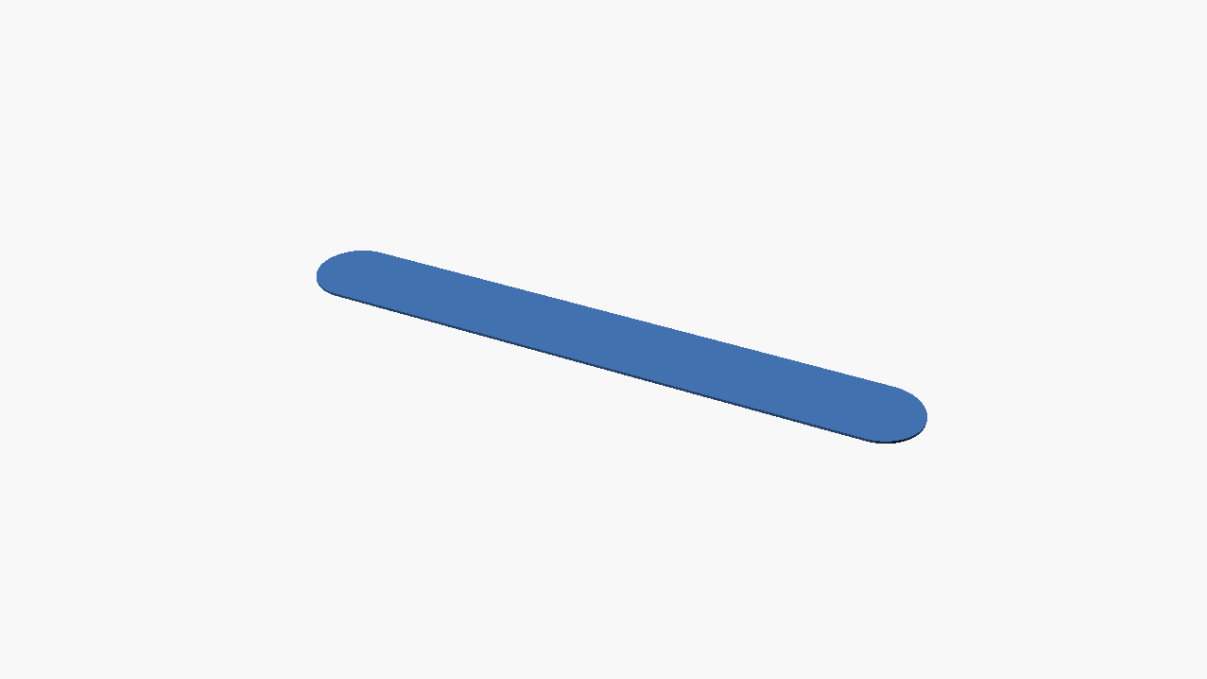

Figure 42. image

Listing 42. Openscad source

```
$fn=100;
holeEndD=16.1;
holeLength=120.1;
buttonHolderHeight=.5;

hull(){
    cylinder(h=buttonHolderHeight,d=holeEndD);
    translate([holeLength-holeEndD,0,0])
cylinder(h=buttonHolderHeight,d=holeEndD);
}
```

housing - 3D Object

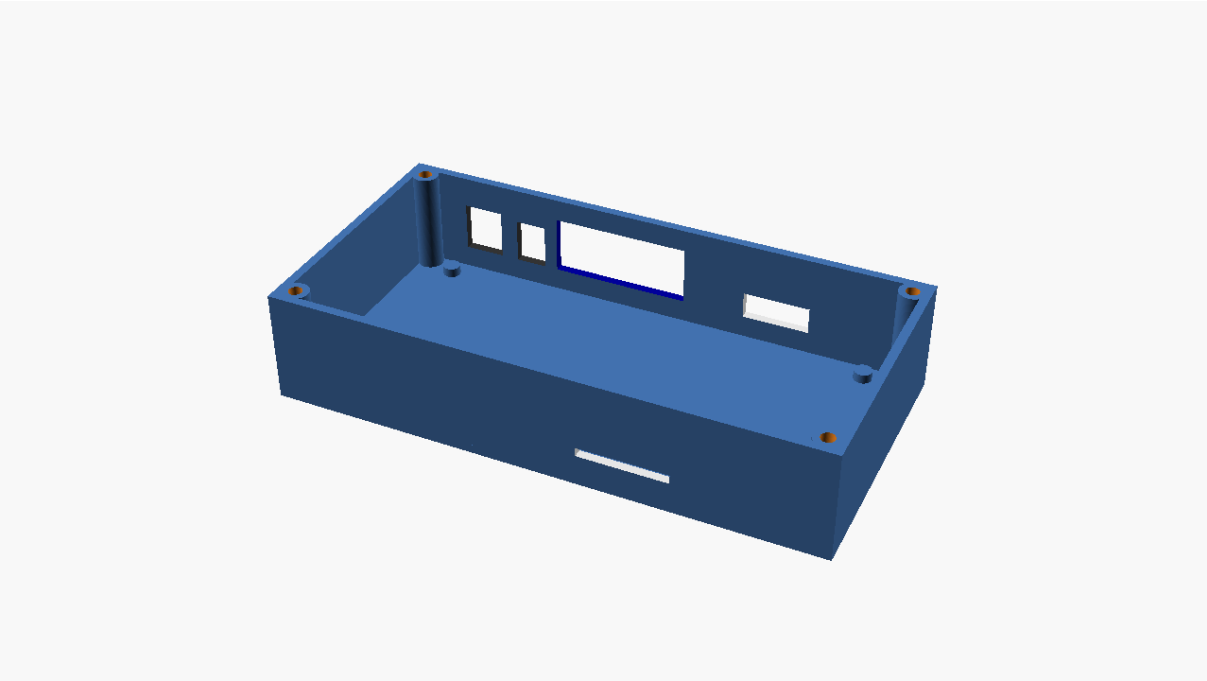

Figure 43. image

Listing 43. Openscad source

```
boardThick=2;
boardWidth=107;
boardDepth=55;
holeRad=2;

module hdmiBoard(ext) {
    Extrude=ext; //extrude the ports or set to 0 for real board
    difference(){
        $fn=100;
        union(){
            color([0,.5,0]) cube([boardWidth,boardDepth,boardThick]);
            color([.2,.2,.2]) translate([6,boardDepth-15,boardThick])
cube([9,15+Extrude,11]); //power
            color([.2,.2,.2]) translate([19,boardDepth-15,boardThick])
cube([7,15+Extrude,10]); //audio
            color([0,0,0.6]) translate([29,boardDepth-9,boardThick])
cube([31,15+Extrude,13]); //vga
            color([.9,.9,.9]) translate([74,boardDepth-9.5,boardThick])
cube([15,11+Extrude,6]); //hdmi
            color([.9,.9,.9]) translate([62,0-Extrude,boardThick])
cube([20,7+Extrude,2]); //display cable 30 pin
        }
        translate([2+holeRad,5+holeRad,-.01])
cylinder(h=boardThick+.1,r=holeRad);
        translate([2+holeRad,boardDepth-2.5-holeRad,-.01])
cylinder(h=boardThick+.1,r=holeRad);
        translate([boardWidth-holeRad-2,boardDepth-holeRad-2,-.01])
```

```
cylinder(h=boardThick+.1,r=holeRad);
        translate([boardWidth-holeRad-3,11+holeRad,-.01])
cylinder(h=boardThick+.1,r=holeRad);
    }
}

module pillars(ext) {
    $fn=100;
    translate([2+holeRad,5+holeRad,-.01]) cylinder(h=boardThick+.1,r=holeRad);
    translate([2+holeRad,boardDepth-2.5-holeRad,-.01])
cylinder(h=boardThick+.1,r=holeRad);
    translate([boardWidth-holeRad-2,boardDepth-holeRad-2,-.01])
cylinder(h=boardThick+.1,r=holeRad);
    translate([boardWidth-holeRad-3,11+holeRad,-.01])
cylinder(h=boardThick+.1,r=holeRad);
}
module plate(width,depth,height,inRad,Off) {
    //blanking plate
    plateWidth=width;
    plateDepth=depth;
    plateHeight=height;
    screwHoleRad=inRad;
    screwEdgeOff=Off;
    difference(){
        //plate
        cube([plateWidth,plateDepth,plateHeight]);
        //screwholes
        $fn=100;
        translate([screwEdgeOff,screwEdgeOff,-.5])
cylinder(h=plateHeight+1,r=screwHoleRad);
        translate([plateWidth-screwEdgeOff,screwEdgeOff,-.5])
cylinder(h=plateHeight+1,r=screwHoleRad);
        translate([screwEdgeOff,plateDepth-screwEdgeOff,-.5])
cylinder(h=plateHeight+1,r=screwHoleRad);
        translate([plateWidth-screwEdgeOff,plateDepth-screwEdgeOff,-.5])
cylinder(h=plateHeight+1,r=screwHoleRad);
    }
}
module pillar(height,inRad,outRad) {
    $fn=100;
    difference(){
        cylinder(h=height,r=outRad);
        translate([0,0,-.05]) cylinder(h=height+.1,r=inRad);
        }
}
module housing(width,depth,height,wallThick,floorThick,inRad,pillarThick,off) {
    //housing
    boxWidth=width;
    boxDepth=depth;
    boxFloorHeight=floorThick;
```

```
    boxWallThick=wallThick;
    boxWallHeight=height;
    screwHoleRad=inRad;
    screwPillarRad=inRad+pillarThick;
    screwEdgeOff=off;
    union(){
        //plate
        cube([boxWidth,boxDepth,boxFloorHeight]);
        echo("Dimensions floor plate",boxWidth,boxDepth,boxFloorHeight);
        translate([screwEdgeOff,screwEdgeOff,0])
pillar(boxWallHeight,screwHoleRad,screwPillarRad);
        translate([boxWidth-screwEdgeOff,screwEdgeOff,0])
pillar(boxWallHeight,screwHoleRad,screwPillarRad);
        translate([screwEdgeOff,boxDepth-screwEdgeOff,0])
pillar(boxWallHeight,screwHoleRad,screwPillarRad);
        translate([boxWidth-screwEdgeOff,boxDepth-screwEdgeOff,0])
pillar(boxWallHeight,screwHoleRad,screwPillarRad);
    }
    //color([1,0,0])
    translate([0,boxWallThick,boxFloorHeight]) cube([boxWallThick,boxDepth-
boxWallThick,boxWallHeight-boxFloorHeight]);
    echo("Dimensions left wall", boxWallThick,boxDepth-
boxWallThick,boxWallHeight-boxFloorHeight);
    //color([0,1,0])
    translate([0,0,boxFloorHeight]) cube([boxWidth-
boxWallThick,boxWallThick,boxWallHeight-boxFloorHeight]);
    echo("Dimensions front wall", boxWidth-
boxWallThick,boxWallThick,boxWallHeight-boxFloorHeight);
    //color([1,0,0])
    translate([boxWidth-boxWallThick,0,boxFloorHeight])
cube([boxWallThick,boxDepth-boxWallThick,boxWallHeight-boxFloorHeight]);
    echo("Dimensions right wall", boxWallThick,boxDepth-
boxWallThick,boxWallHeight-boxFloorHeight);
    //color([0,1,0])
    translate([boxWallThick,boxDepth-boxWallThick,boxFloorHeight])
cube([boxWidth-boxWallThick,boxWallThick,boxWallHeight-boxFloorHeight]);
    echo("Dimensions rear wall", boxWidth-
boxWallThick,boxWallThick,boxWallHeight-boxFloorHeight);
}
//width,depth,height,wallThick,floorThick,inRad,pillarThick,off
//housing(20,20,20,2,2,1.5,1.5,4);

difference() {
    housing(122,61,25,2,2,1.5,1.5,4);
//width,depth,height,wallThick,floorThick,inRad,pillarThick,off
    translate([7,3,5]) hdmiBoard(20);
}
translate([7,4,2]) pillars();
*translate([7,4,5]) hdmiBoard(0);
```

```
*translate([0,0,70]) plate(122,61,2,1.5,4); //width,depth,height,inRad,Off
```

screen_mounting_tabs - 3D Object

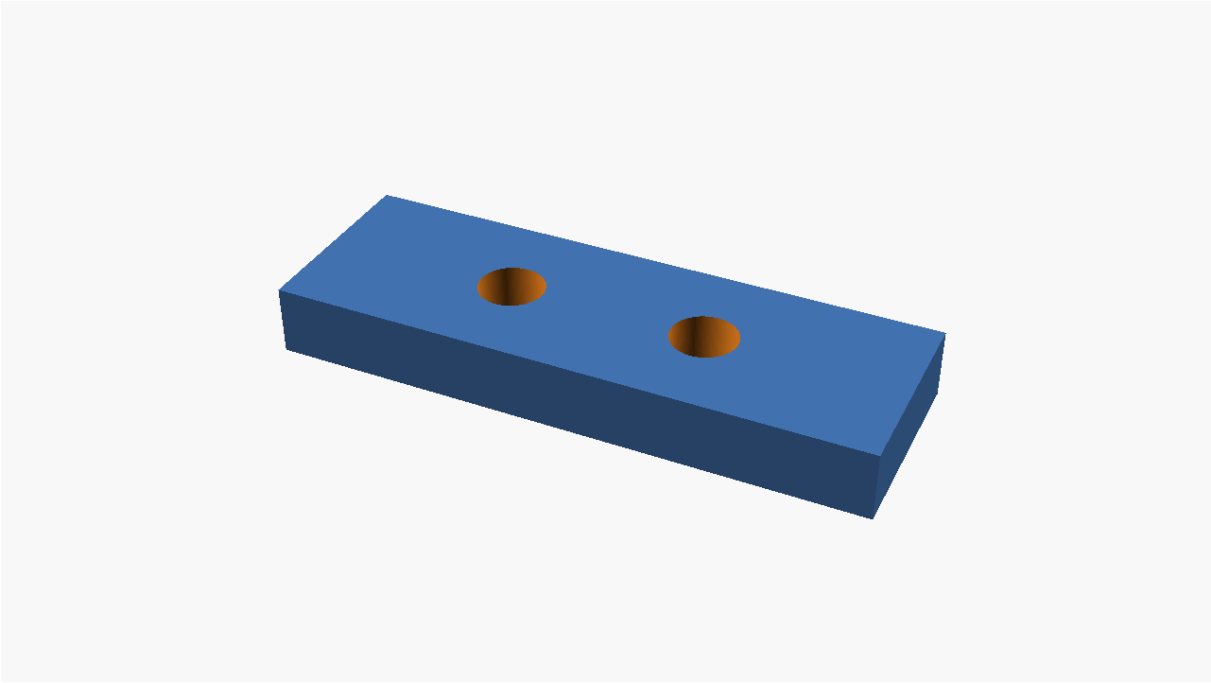

Figure 44. image

Listing 44. Openscad source

```
$fn=100;
tab_height=.3;
tab2bottom=2.4;

plus=.1; // this is to make parts larger than the hole they are to make
plusH=plus/2;

hole_d=2;
tab1_hole_spacing=8;
tab2_hole_spacing=6;
shim_height=tab2bottom-tab_height;
shim_depth=6;
shim_width=18;

module tab1(spacing){
    difference(){
        cube([shim_width,shim_depth,shim_height]);
        translate([shim_width/2-spacing/2,(shim_depth/2),-plusH])
            union() {
                cylinder(h=shim_height+plus,d=hole_d);
                translate([spacing,0,0]) cylinder(h=shim_height+plus,d=hole_d);
            }
    }
```

```
}

//for the bottom tabs
//tab1(tab1_hole_spacing);

//for the top tabs
tab1(tab2_hole_spacing);
```

odroid-case - Project

Library-container - 3D Object

This is the Library for the case.
I sourced the methods out so as to be able to re-use them better.

Figure 45. image

Listing 45. Openscad source

```
    module nibLeft(x,y,z,nibR) {
        translate([x,y/10,0]) cylinder(h=z,r=nibR);
        translate([x,9*(y/10),0]) cylinder(h=z,r=nibR);
    }
    module nibRight(x,y,z,nibR) {
        translate([0,y/10,0]) cylinder(h=z,r=nibR);
        translate([0,9*(y/10),0]) cylinder(h=z,r=nibR);
    }
    module nibBottom(x,y,z,nibR) {
        translate([x/10,0,0]) cylinder(h=z,r=nibR);
        translate([9*(x/10),0,0]) cylinder(h=z,r=nibR);
```

```
}
module nibTop(x,y,z,nibR) {
    translate([x/10,y,0]) cylinder(h=z,r=nibR);
    translate([9*(x/10),y,0]) cylinder(h=z,r=nibR);
}
module containerOpenLid(x,y,z,rimThick,bottomThick,nibYN,nibR) {
rimR=rimThick/2;
//all 8 corners defined first
//corners should be AROUND the contained cube defined by x y z
corner000=[0,0,0];
corner0=[-rimR,-rimR,-(rimR+bottomThick)];
corner0x=[x+rimR,-rimR,-(rimR+bottomThick)];
corner0y=[-rimR,y+rimR,-(rimR+bottomThick)];
corner0xy=[x+rimR,y+rimR,-(rimR+bottomThick)];
corner0z=[-rimR,-rimR,z];
corner0xz=[x+rimR,-rimR,z];
corner0yz=[-rimR,y+rimR,z];
corner0xyz=[x+rimR,y+rimR,z];   //draw the debug contents
translate([0,0,-bottomThick]) cube([x,y,bottomThick]);
module corner0() {
    translate(corner0) sphere(r=rimR);
}
module corner0x() {
    translate(corner0x) sphere(r=rimR);
}
module corner0y() {
    translate(corner0y) sphere(r=rimR);
}
module corner0xy() {
    translate(corner0xy) sphere(r=rimR);
}
module corner0z() {
    translate(corner0z) sphere(r=rimR);
}
module corner0xz() {
    translate(corner0xz) sphere(r=rimR);
}
module corner0yz() {
    translate(corner0yz) sphere(r=rimR);
}
module corner0xyz() {
    translate(corner0xyz) sphere(r=rimR);
}
if (nibYN=="nibY") {
    nibBottom(x,y,z,nibR);
    nibLeft(x,y,z,nibR);
    nibRight(x,y,z,nibR);
    nibTop(x,y,z,nibR);
}
union(){
```

```
        //floor
        hull(){
            corner0();
            corner0x();
            corner0y();
            corner0xy();
        }
        //left
        hull(){
            corner0();
            corner0z();
            corner0y();
            corner0yz();
        }
        //right
        hull(){
            corner0x();
            corner0xy();
            corner0xyz();
            corner0xz();
        }
        //top
        hull(){
            corner0y();
            corner0yz();
            corner0xyz();
            corner0xy();
        }
        //bottom
        hull(){
            corner0();
            corner0z();
            corner0x();
            corner0xz();
        }
    }
}

// example
//caseRim=1.5;
//$fn=100;
//odH=10;
//odW=156;
//odD=73;
//odJSH=6;
//#containerOpenLid(odW,odD,odH,caseRim,odJSH-caseRim,"nibY",.6);
//cube([odW,odD,odH]);

module containerVertSlot(x,y,z,rimThick,bottomThick,nibYN,nibR) {
    rimR=rimThick/2;
```

```
    //all 8 corners defined first
    //corners should be AROUND the contained cube defined by x y z
    corner000=[0,0,0];
    corner0=[-rimR,-rimR,-(rimR+bottomThick)];
    corner0x=[x+rimR,-rimR,-(rimR+bottomThick)];
    corner0y=[-rimR,y+rimR,-(rimR+bottomThick)];
    corner0xy=[x+rimR,y+rimR,-(rimR+bottomThick)];
    corner0z=[-rimR,-rimR,z];
    corner0xz=[x+rimR,-rimR,z];
    corner0yz=[-rimR,y+rimR,z];
    corner0xyz=[x+rimR,y+rimR,z];   //draw the debug contents
    translate([0,0,-bottomThick]) cube([x,y,bottomThick]);
    module corner0() {
        translate(corner0) sphere(r=rimR);
    }
    module corner0x() {
        translate(corner0x) sphere(r=rimR);
    }
    module corner0y() {
        translate(corner0y) sphere(r=rimR);
    }
    module corner0xy() {
        translate(corner0xy) sphere(r=rimR);
    }
    module corner0z() {
        translate(corner0z) sphere(r=rimR);
    }
    module corner0xz() {
        translate(corner0xz) sphere(r=rimR);
    }
    module corner0yz() {
        translate(corner0yz) sphere(r=rimR);
    }
    module corner0xyz() {
        translate(corner0xyz) sphere(r=rimR);
    }
    if (nibYN=="nibY") {
        nibLeft(x,y,z,nibR);
        nibRight(x,y,z,nibR);
    }
    union(){
        //floor
        hull(){
            corner0();
            corner0x();
            corner0y();
            corner0xy();
        }
        //left
        hull(){
```

```
            corner0();
            corner0z();
            corner0y();
            corner0yz();
        }
        //right
        hull(){
            corner0x();
            corner0xy();
            corner0xyz();
            corner0xz();
        }
    }
}
// example
//caseRim=1.5;
//$fn=100;
//odH=10;
//odW=15;
//odD=73;
//odJSH=6;
//containerVertSlot(odW,odD,odH,caseRim,odJSH-caseRim,"nibY",.6);
//cube([odW,odD,odH]);

if (library) {} else  {
    echo("trying to compile a library!");
    linear_extrude(height = 4) {
        text("trying to compile a library!");
        }
    }
```

case - 3D Object

A case for an odroid handgeld console and accessories.

The edges are rounded and there are cutouts for the parts that protrude from the console.

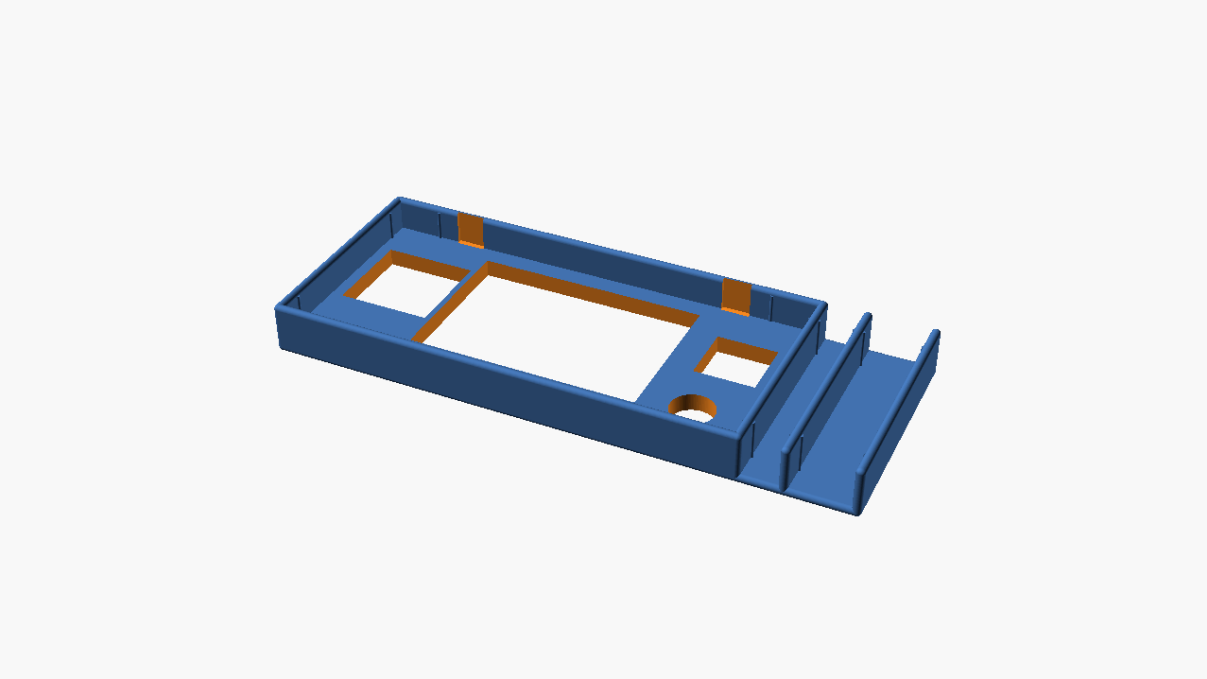

Figure 46. image

Listing 46. Openscad source

```
include <Library-container.scad>
caseRim=3;
holdersR=.7;
$fn=100;
kbD=82;
kbW=210;
kbH=7;
library=true;

odH=10;
odW=156;
odD=73;

odJSW=28-13;
odJSR=odJSW/2;
odJSoffX=13;
odJSoffY=11;

odJSH=6;
odBRW=118-38;
odBRD=12-5;
odBRoffX=38;
odBRoffY=5;
odCRW=28-7;
odCRD=58-37;
odCRoffX=7;
odCRoffY=37;
odTBW=152-120;
```

```
odTBD=61-27;
odTBoffX=120;
odTBoffY=27;
odTLoffX=23;
odTLoffY=odD;
odTLW=33-23;
odTLD=2;
odTH=20;
odTRoffX=123;
odTRoffY=odD;
odTRW=odTLW;
odTRD=odTLD;
odDPW=odBRW;
odDPD=67-14;
odDPoffY=14;
odDPoffX=odBRoffX;

//the odroid travel case with cutouts for buttons etc
difference(){
    //the container itself
    translate([caseRim/2,caseRim/2,odJSH])
containerOpenLid(odW,odD,odH,caseRim,odJSH-caseRim,"nibY",.6);
    offset=.01;
    //the cutouts
    translate([1.5,.75,-offset/2]) union() {
        translate([odW-odJSR*2-odJSoffX,odJSoffY,0]+[odJSR,odJSR,0])
cylinder(h=odJSH+offset,r=odJSR);
        translate([odW-odBRW-odBRoffX,odBRoffY,0])
cube([odBRW,odBRD,odJSH+offset]);
        translate([odW-odCRW-odCRoffX,odCRoffY,0])
cube([odCRW,odCRW,odJSH+offset]);
        translate([odW-odTBW-odTBoffX,odTBoffY,0])
cube([odTBW,odTBW,odJSH+offset]);
        translate([odW-odTLW-odTLoffX,odTLoffY,odJSH-.1])
cube([odTLW,odTLD,odTH+offset]);
        translate([odW-odTRW-odTRoffX,odTRoffY,odJSH-.1])
cube([odTRW,odTRD,odTH+offset]);
        translate([odW-odDPW-odDPoffX,odDPoffY,0])
cube([odDPW,odDPD,odJSH+offset]);
    }
}
//add on some slots for peripherals
floorDepth=0;
//microuter slot
translate([caseRim/2+caseRim+odW,caseRim/2,floorDepth])
    containerVertSlot(12,odD,odH+odJSH,caseRim,floorDepth-caseRim,"nibY",.6);
//micro USB 3 Port Hub
translate([caseRim/2+2*caseRim+odW+12,caseRim/2,floorDepth])
    containerVertSlot(19.5,odD,odH+odJSH,caseRim,floorDepth-caseRim,"nibY",.6);
```

openai - Project

I decided to have a go to see where openAI's chatGPT has its limits.
And it does have limits.
It's intersting to see that while it seems to know what an object might consist of it has a very hard time making the onject in any way accurate.
In any case it's Interesting.

esp8266case-chatgpt - 3D Object

This one is also interesting in that it does some difference etc but it's really not a model of a case.
It looks like it get's the basic premise of subtracting one thing from another etc.
On the other hand it doesn't seem to have a clue about dimensions.
Either way an interesting experimt which shows up the limitations of GPT models.

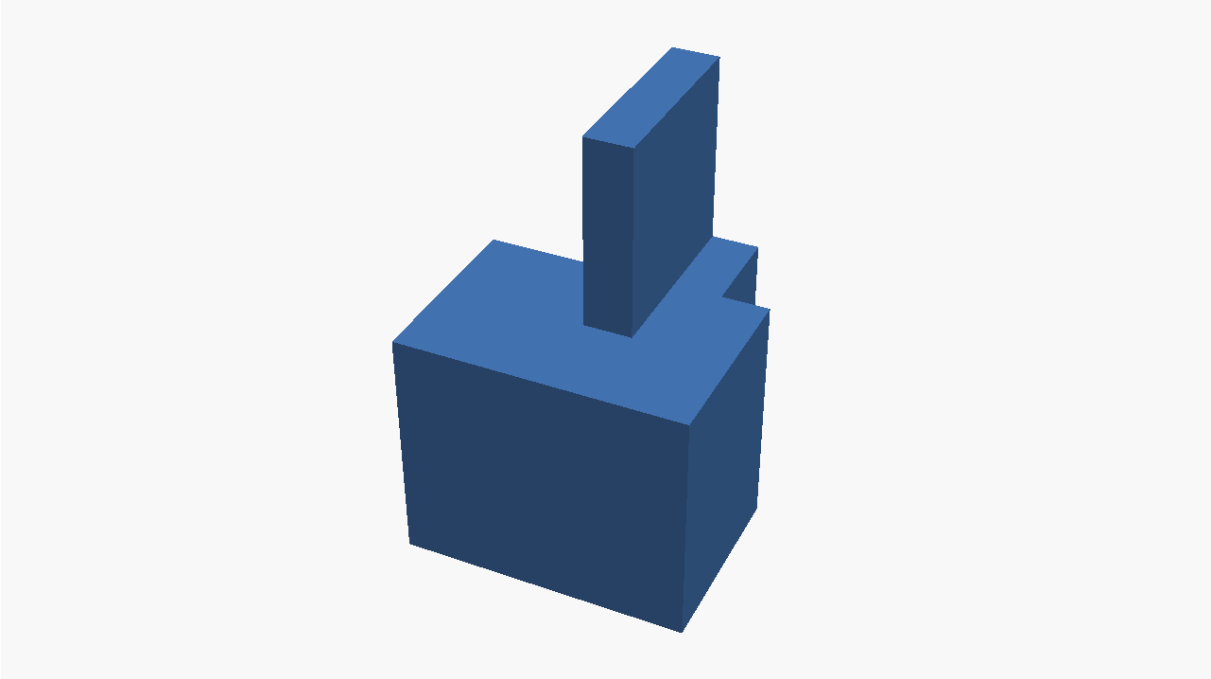

Figure 47. image

Listing 47. Openscad source

```
// Create the base of the case
translate([0, 0, 5]) {
  cube([30, 20, 10]);
}


// Create the sides of the case
translate([0, 0, 15]) {
  cube([30, 20, 10]);
  translate([15, 10, 15]) {
```

```
    rotate([0, 90, 0]) {
      cube([30, 20, 10]);
    }
  }
}

// Create the lid of the case
translate([0, 0, 25]) {
  cube([30, 20, 5]);
  translate([15, 10, 25]) {
    rotate([0, 90, 0]) {
      cube([30, 20, 5]);
    }
  }
}

// Create the opening for the ESP8266
translate([15, 10, 30]) {
  rotate([0, 90, 0]) {
    cube([20, 15, 5]);
  }
}
```

solar-generator-chatgpt - 3D Object

This is also an interesting experiment.
I asked chatGPT to make a solar generator openscad model and this came out.
On the other hand it seems to be good at figuring out what should be part of a solar generator.
I'll have to take closer look of what it did here and how hard it is to fix it.
This could be promising in total but right now it more or less seems to have hit a brick wall that seems to relate to dimensions etc.
In other words it's good at creating a structure but not so good at understanding the physics of what it made.

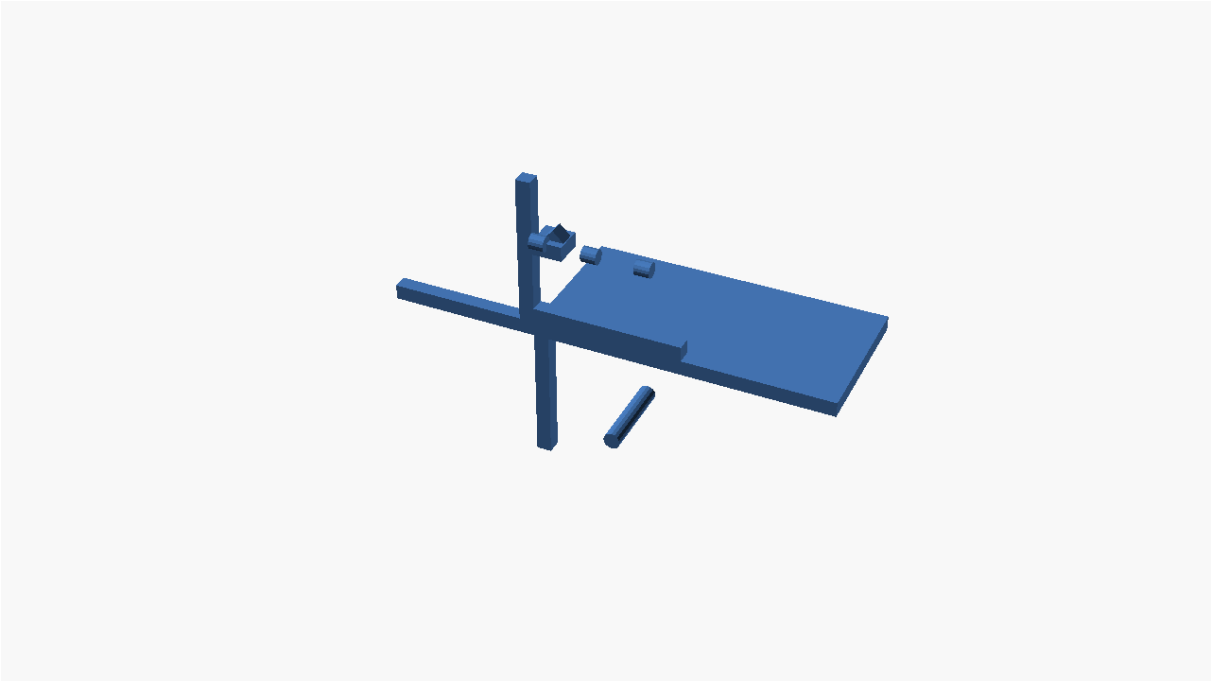

Figure 48. image

Listing 48. Openscad source

```
// Create the base of the solar generator
translate([0, 0, 5]) {
  cube([200, 100, 10]);
}

// Create the solar panels of the solar generator
translate([0, 0, 15]) {
  for (i = [0:3]) {
    rotate([0, i * 90, 0]) {
      cube([100, 10, 10]);
    }
  }
}

// Create the handle of the solar generator
translate([100, -50, 25]) {
  rotate([90, 0, 0]) {
    cylinder(r=5, h=50);
  }
}

// Create the control panel of the solar generator
translate([0, 0, 65]) {
  cube([20, 20, 10]);
  translate([5, 5, 10]) {
    rotate([0, 45, 0]) {
      cube([10, 10, 10]);
    }
```

```
  }
}

// Create the outlets of the solar generator
translate([0, 0, 75]) {
  for (i = [0:2]) {
    translate([i * 35, 0, 0]) {
      rotate([0, 90, 0]) {
        cylinder(r=5, h=10);
      }
    }
  }
}
```

teacup-chatgpt - 3D Object

What chatGPT thinks a teacup could look like in openscad.
It's interesting to see that it's sructured and syntacticly more or less correct but is also not a tea cup.

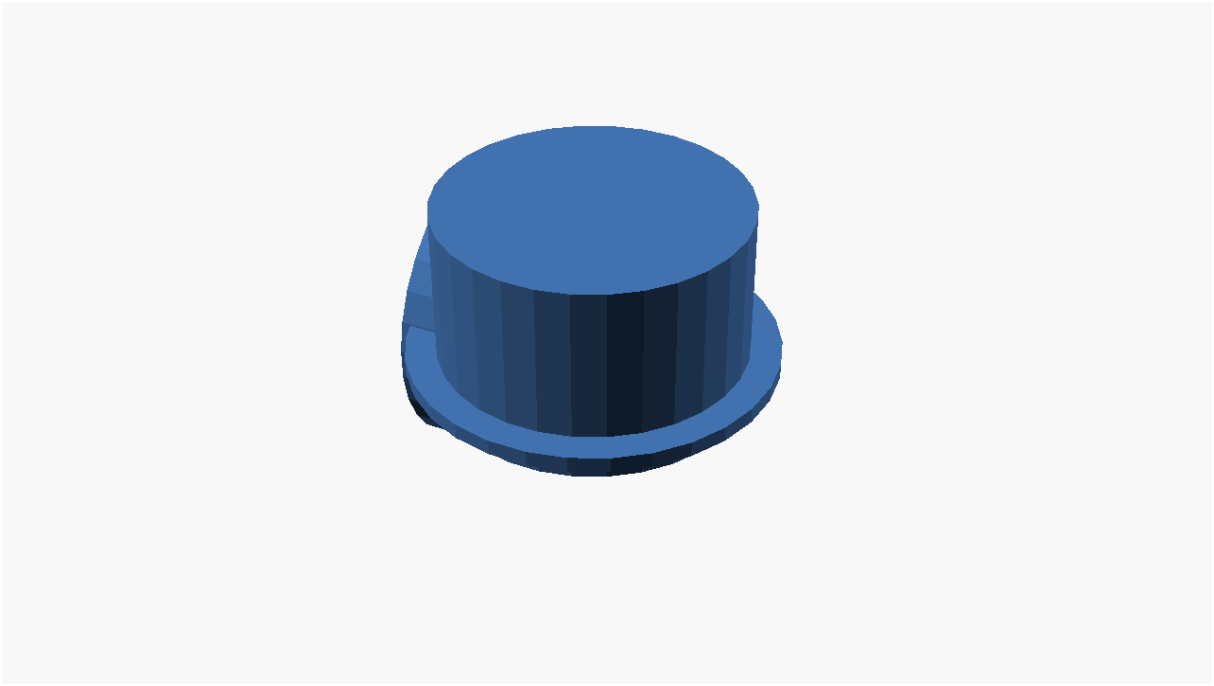

Figure 49. image

Listing 49. Openscad source

```
// Create the base of the tea cup
cylinder(r=50, h=10);

// Create the handle of the tea cup
translate([-60, 30, 0]) {
  rotate([0, 90, 0]) {
```

```
    cylinder(r=10, h=30);
  }
}

// Create the rim of the tea cup
translate([0, 0, 10]) {
  cylinder(r=60, h=5);
}

// Create the body of the tea cup
translate([0, 0, 15]) {
  cylinder(r=50, h=50);
}

// Cut out the handle from the body of the tea cup
translate([-60, 30, 0]) {
  rotate([0, 90, 0]) {
    difference() {
      cylinder(r=50, h=50);
      cylinder(r=10, h=30);
      }
      }
      }
```

piZero - Project

- Required for a rpi zero cluster
- need to add a few extra parts
  - a holder for a USB hub
  - A holder for the PSU
  - some extra stuff

RPI_zero_Cluster_mounting_bracket_power - 3D Object

Figure 50. image

Listing 50. Openscad source

```
//This WILL e the power base and possibly usb hub holdewr for the cluster
//needs the mounting posts to sit the cluster on top of
//requires a base for the power brick and a holder for the USB hub whic can sit
upright next to the cluster

$fn = 50;

module stack_joins()
{
    mount_h = 20;
    Ro=4;
    Ri=2.5;
    Rp=Ri-0.4;
    difference() {
        union() {
            translate([-39, 21.5, mount_h/2]) { cylinder(h=mount_h, r=Ro,
center=true); }
            translate([ 39,-21.5, mount_h/2]) { cylinder(h=mount_h, r=Ro,
center=true); }
            translate([-39,-21.5, mount_h/2]) { cylinder(h=mount_h, r=Ro,
center=true); }
            translate([ 39, 21.5, mount_h/2]) { cylinder(h=mount_h, r=Ro,
center=true); }

            translate([ 39, 21.5, mount_h/2 + mount_h * 0.35]) {
cylinder(h=mount_h, r=Rp, center=true); }
            translate([-39, 21.5, mount_h/2 + mount_h * 0.35]) {
cylinder(h=mount_h, r=Rp, center=true); }
```

```
            translate([-39,-21.5, mount_h/2 + mount_h * 0.35]) {
cylinder(h=mount_h, r=Rp, center=true); }
            translate([ 39,-21.5, mount_h/2 + mount_h * 0.35]) {
cylinder(h=mount_h, r=Rp, center=true); }

            translate([ 39, 21.5, mount_h + mount_h * 0.35]) { sphere(r=Rp); }
            translate([-39, 21.5, mount_h + mount_h * 0.35]) { sphere(r=Rp); }
            translate([-39,-21.5, mount_h + mount_h * 0.35]) { sphere(r=Rp); }
            translate([ 39,-21.5, mount_h + mount_h * 0.35]) { sphere(r=Rp); }
        }
        union() {
            translate([ 39, 21.5, mount_h/2 - mount_h * 0.15]) {
cylinder(h=mount_h, r=Ri, center=true); }
            translate([-39, 21.5, mount_h/2 - mount_h * 0.15]) {
cylinder(h=mount_h, r=Ri, center=true); }
            translate([-39,-21.5, mount_h/2 - mount_h * 0.15]) {
cylinder(h=mount_h, r=Ri, center=true); }
            translate([ 39,-21.5, mount_h/2 - mount_h * 0.15]) {
cylinder(h=mount_h, r=Ri, center=true); }
        }
    }
}
//parameters for the hub and the brick
BRKx=130; BRKy=67; BRKz=32.5;BRKd=8;
HUBx=12.5; HUBy=32.5; HUBz=75;

module power_brk(x,y,z,d) {
    translate ([-x/2,-y/2,-z/2]) cube([x,y,z]);
}
module power_brick(x,y,z,d) {
    union() {
        BTR=[x/2+d/2,y/2+d/2,z/2+d/2];
        FTR=[x/2+d/2,-y/2-d/2,z/2+d/2];
        BTL=[-x/2-d/2,y/2+d/2,z/2+d/2];
        FTL=[-x/2-d/2,-y/2-d/2,z/2+d/2];
        BR=[x/2+d/2,y/2+d/2,-z/2];
        FR=[x/2+d/2,-y/2-d/2,-z/2];
        BL=[-x/2-d/2,y/2+d/2,-z/2];
        FL=[-x/2-d/2,-y/2-d/2,-z/2];
        hull() {
            translate(BTR) sphere(d=d);
            translate(FTR) sphere(d=d);
            translate(BTL) sphere(d=d);
            translate(FTL) sphere(d=d);
        }
        translate(BR) cylinder(h=y/2+d/4,d=d);
        translate(FR) cylinder(h=y/2+d/4,d=d);
        translate(BL) cylinder(h=y/2+d/4,d=d);
        translate(FL) cylinder(h=y/2+d/4,d=d);
    }
```

```
}
module usb_hub(x,y,z,d) {
    *cube([x,y,z]);
    FBL=[-d/2,-d/2,0];
    FBR=[x+d/2,-d/2,0];
    FTL=[-d/2,-d/2,z+d/2];
    FTR=[x+d/2,-d/2,z];
    BBL=[-d/2,+d/2+y,0];
    BBR=[x+d/2,+d/2+y,0];
    BTL=[-d/2,+d/2+y,z];
    BTR=[x+d/2,+d/2+y,z+d/2];
    //xleft
    hull () {
        translate(FBL) sphere(d=d);
        translate(BTL) sphere(d=d); }
    hull () {
        translate(FTL) sphere(d=d);
        translate(BBL) sphere(d=d); }
    //xright
    hull () {
        translate(FBR) sphere(d=d);
        translate(BTR) sphere(d=d); }
    hull () {
        translate(FTR) sphere(d=d);
        translate(BBR) sphere(d=d); }
    //top front2back
    hull () {
        translate(FTL) sphere(d=d);
        translate(BTL) sphere(d=d); }
    hull () {
        translate(FTR) sphere(d=d);
        translate(BTR) sphere(d=d); }
    //top left2right
    hull () {
        translate(BTL) sphere(d=d);
        translate(BTR) sphere(d=d); }
    hull () {
        translate(FTL) sphere(d=d);
        translate(FTR) sphere(d=d); }
    //bottom front2back
    hull () {
        translate(FBL) sphere(d=d);
        translate(BBL) sphere(d=d); }
    hull () {
        translate(FBR) sphere(d=d);
        translate(BBR) sphere(d=d); }
    //bottom left2right
    hull () {
        translate(BBL) sphere(d=d);
        translate(BBR) sphere(d=d); }
```

```
    hull () {
        translate(FBL) sphere(d=d);
        translate(FBR) sphere(d=d); }
}

//display the stuff
//case
translate([0,0,-5]) {
    translate([((BRKx/2)-(HUBx/2)+(BRKd/4)-BRKd),-HUBy/2,BRKz/2+BRKd])
usb_hub(HUBx,HUBy,HUBz,BRKd);
    power_brick(BRKx,BRKy,BRKz,BRKd);
}
//the stack pins
difference() {
    stack_joins();
    translate([0,0,-5]) power_brk(BRKx,BRKy,BRKz,BRKd);
}
//just the hub
*usb_hub(HUBx,HUBy,HUBz,BRKd);
```

RPI_zero_Cluster_mounting_bracket_v2 - 3D Object

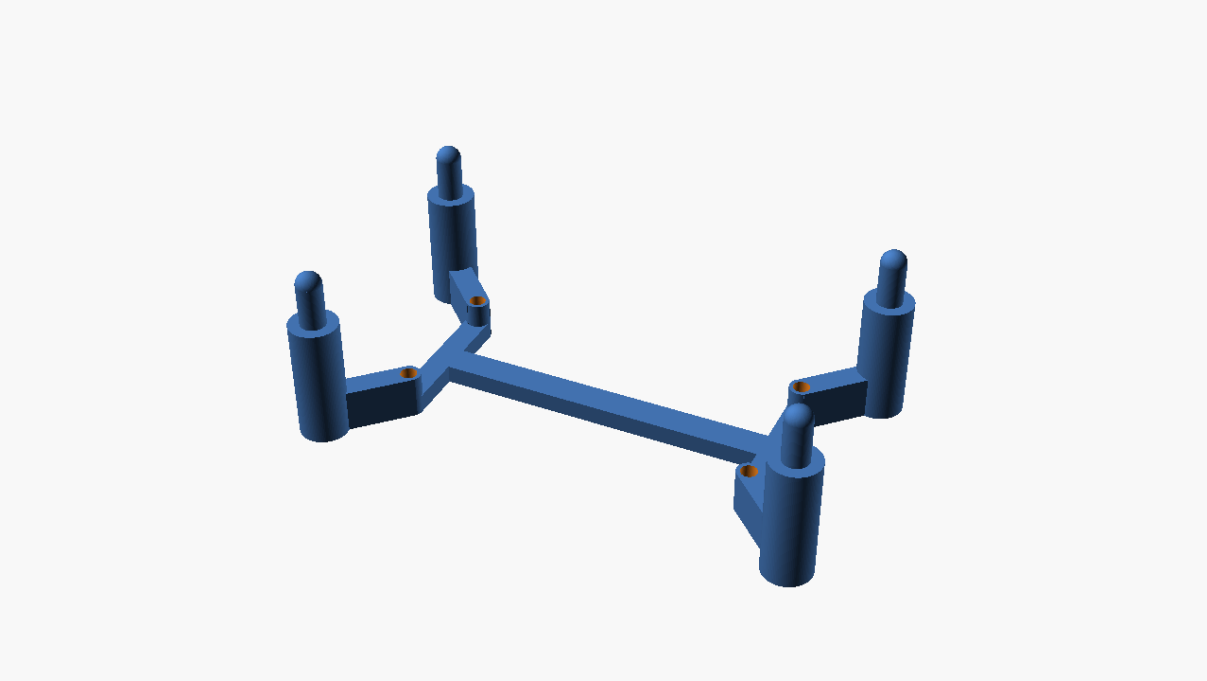

Figure 51. image

Listing 51. Openscad source

```
$fn = 100;

module mount(x, y, z)
{
    mount_h = 7;
```

```
    mount_h2 = 3;

    difference()
    {
        union()
        {
            translate([-29, 11.5, mount_h/2]) { cylinder(h=mount_h, r=2,
center=true); }
            translate([ 29,-11.5, mount_h/2]) { cylinder(h=mount_h, r=2,
center=true); }
            translate([-29,-11.5, mount_h/2]) { cylinder(h=mount_h, r=2,
center=true); }
            translate([ 29, 11.5, mount_h/2]) { cylinder(h=mount_h, r=2,
center=true); }

            translate([x+29, y, z + 3/2]) { cube([4, 23.0, mount_h2],
center=true); }
            translate([x-29, y, z + 3/2]) { cube([4, 23.0, mount_h2],
center=true); }

            translate([x, y, z + mount_h2/2]) { cube([58, 5.0, mount_h2],
center=true); }


            delta=2.1;
            translate([x+35-delta, y-17.5+delta, z + mount_h/2]) rotate([0, 0,
45]) {{ cube([4, 12.0, mount_h], center=true); }}
            translate([x+35-delta, y+17.5-delta, z + mount_h/2]) rotate([0, 0,-
45]) {{ cube([4, 12.0, mount_h], center=true); }}


            translate([x-35+delta, y-17.5+delta, z + mount_h/2]) rotate([0, 0,-
45]) {{ cube([4, 12.0, mount_h], center=true); }}
            translate([x-35+delta, y+17.5-delta, z + mount_h/2]) rotate([0, 0,
45]) {{ cube([4, 12.0, mount_h], center=true); }}

        }
        union()
        {
            translate([ 29, 11.5, mount_h/2 + mount_h * 0.15]) {
cylinder(h=mount_h, r=2.70/2, center=true); }
            translate([-29, 11.5, mount_h/2 + mount_h * 0.15]) {
cylinder(h=mount_h, r=2.70/2, center=true); }
            translate([-29,-11.5, mount_h/2 + mount_h * 0.15]) {
cylinder(h=mount_h, r=2.70/2, center=true); }
            translate([ 29,-11.5, mount_h/2 + mount_h * 0.15]) {
cylinder(h=mount_h, r=2.70/2, center=true); }
        }
    }
```

```
}

module stack_joins(x, y, z)
{
    mount_h = 20;
    Ro=4;
    Ri=2.5;
    Rp=Ri-0.4;

    difference()
    {
        union()
        {
            translate([-39, 21.5, mount_h/2]) { cylinder(h=mount_h, r=Ro,
center=true); }
            translate([ 39,-21.5, mount_h/2]) { cylinder(h=mount_h, r=Ro,
center=true); }
            translate([-39,-21.5, mount_h/2]) { cylinder(h=mount_h, r=Ro,
center=true); }
            translate([ 39, 21.5, mount_h/2]) { cylinder(h=mount_h, r=Ro,
center=true); }

            translate([ 39, 21.5, mount_h/2 + mount_h * 0.35]) {
cylinder(h=mount_h, r=Rp, center=true); }
            translate([-39, 21.5, mount_h/2 + mount_h * 0.35]) {
cylinder(h=mount_h, r=Rp, center=true); }
            translate([-39,-21.5, mount_h/2 + mount_h * 0.35]) {
cylinder(h=mount_h, r=Rp, center=true); }
            translate([ 39,-21.5, mount_h/2 + mount_h * 0.35]) {
cylinder(h=mount_h, r=Rp, center=true); }

            translate([ 39, 21.5, mount_h + mount_h * 0.35]) { sphere(r=Rp); }
            translate([-39, 21.5, mount_h + mount_h * 0.35]) { sphere(r=Rp); }
            translate([-39,-21.5, mount_h + mount_h * 0.35]) { sphere(r=Rp); }
            translate([ 39,-21.5, mount_h + mount_h * 0.35]) { sphere(r=Rp); }
        }
        union()
        {
            translate([ 39, 21.5, mount_h/2 - mount_h * 0.15]) {
cylinder(h=mount_h, r=Ri, center=true); }
            translate([-39, 21.5, mount_h/2 - mount_h * 0.15]) {
cylinder(h=mount_h, r=Ri, center=true); }
            translate([-39,-21.5, mount_h/2 - mount_h * 0.15]) {
cylinder(h=mount_h, r=Ri, center=true); }
            translate([ 39,-21.5, mount_h/2 - mount_h * 0.15]) {
cylinder(h=mount_h, r=Ri, center=true); }
        }
    }
}
```

```
module model()
{
    mount(0, 0, 0);
    stack_joins(0, 0, 0);
}

model();
```

RPi_zero_mount - 3D Object

This also is NOT one of mine but I've cleaned it up a bit as it wasn't displaying correctly. I needed it for a pi cluster and as it's quite good I didn't reinvent the wheel here.

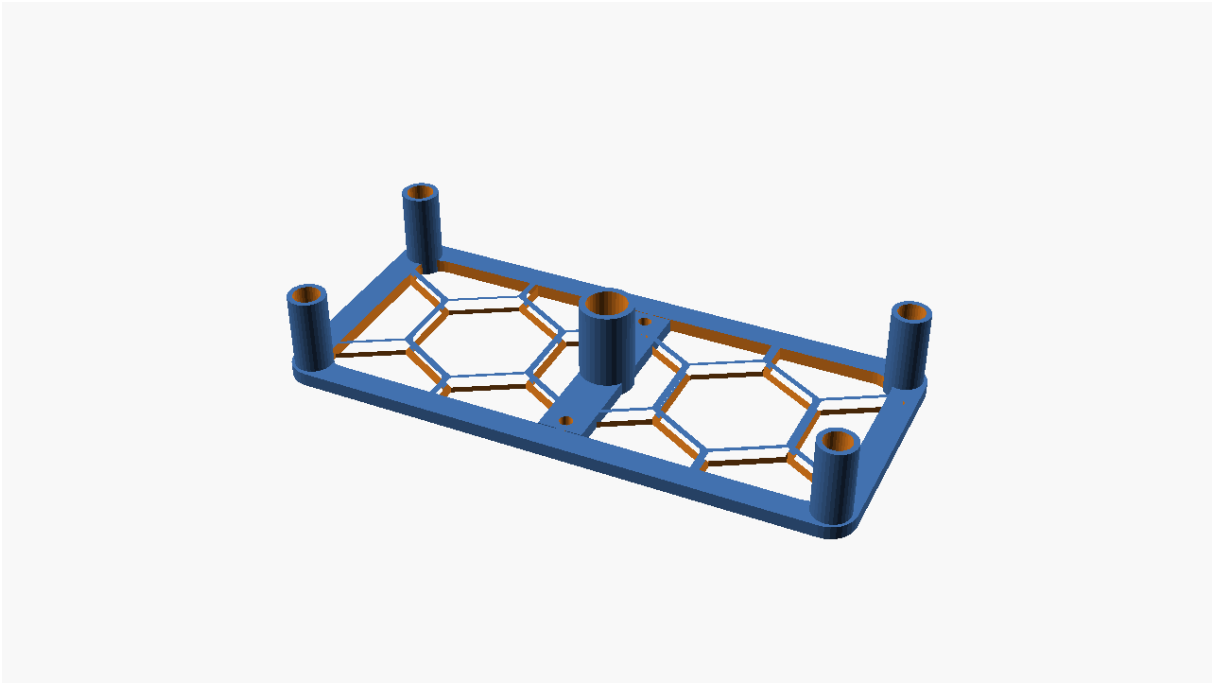

Figure 52. image

Listing 52. Openscad source

```
/* [Base] */
//type = 1; //[1:"Hexagon Grid",2:"Skeleton"]

/* [Hidden] */
$fn = 32;
zero_x = 64;
zero_y = 29;
zero_z = 1.5;

mounts_z = 8.5;
mounts_radius = 2.1;
screwholes = 2.6;
screwholes_radius = 1.5;
screwholes_depth = 10.7;
```

```
base_x = zero_x - 2*3.0;
base_y  = zero_y  - 2*3.0;
base_z = zero_z;
mount_x = zero_x/2 - screwholes;
mount_y = zero_y/2 - screwholes;
mount_z = zero_z + mounts_z;
screwhole_base_z = mount_z - screwholes_depth;

module baseplate(){
    translate([-zero_x/2+3,-zero_y/2+3,0])
        minkowski(){
            cube([base_x,base_y,base_z/2]);
            cylinder(r=3.0,h=base_z/2);
    }
}

module mounts(){
    translate([0,0,0]) cylinder(r=3.0,h=mount_z);
    translate([-mount_x, -mount_y, 0]) cylinder(r=mounts_radius,h=mount_z);
    translate([-mount_x, +mount_y, 0]) cylinder(r=mounts_radius,h=mount_z);
    translate([+mount_x, -mount_y, 0]) cylinder(r=mounts_radius,h=mount_z);
    translate([+mount_x, +mount_y, 0]) cylinder(r=mounts_radius,h=mount_z);
}

module hexagon (radius=8,latticeWidth=8,latticeLength=16,spacing=1,height=2){
    linear_extrude(height) {
        for(j = [0:latticeWidth-1]) {

translate([((sqrt(3)*radius)+spacing)/2*(j%2),sqrt((pow(((sqrt(3)*radius)+spacin
g),2))-(pow((((sqrt(3)*radius)+spacing))/2,2)))*j,0]) {
                for(i = [0:latticeLength-1]) {
                    translate([(sqrt(3)*radius*i)+spacing*i,0,0]) {
                        rotate([0,0,30]) {
                            circle(radius, $fn = 6);
                        }
                    }
                }
            }
        }
    }
}

module hex_border(){
    difference(){
        baseplate();
        holes();
        translate([0,0,-.01]) scale([0.9,0.8,1.02]) baseplate();
    }
}
```

```
module holes(){
    translate([0,0,screwhole_base_z+0.4]) {
        translate([0,0,0]) cylinder(r=screwholes_radius*1.5,h=screwholes_depth);
        translate([-mount_x,-mount_y,0])
cylinder(r=screwholes_radius,h=screwholes_depth);
        translate([-mount_x,+mount_y,0])
cylinder(r=screwholes_radius,h=screwholes_depth);
        translate([+mount_x,-mount_y,0])
cylinder(r=screwholes_radius,h=screwholes_depth);
        translate([+mount_x,+mount_y,0])
cylinder(r=screwholes_radius,h=screwholes_depth);
    };
}

module result(){
    difference(){
        translate([-2.5,-base_y/2,0]) cube([5,base_y,base_z]);
        translate([0,10,-3]) cylinder(d=1.5,h=10);
        translate([0,-10,-3]) cylinder(d=1.5,h=10);
        holes();
    }
    translate([0,0,0])
    hex_border();
    difference(){
        translate([0,0,0]) cylinder(r=3.0,h=mount_z);
        holes();
    }
    difference(){
        mounts();
        holes();
    }
    difference(){
        baseplate();
        holes();
        translate([-zero_x/2-5,-zero_y/2+1.5,-0.1]) hexagon();
    }
}

difference(){
    result();
    translate([0,10,-3]) cylinder(d=1.5,h=10);
    translate([0,-10,-3]) cylinder(d=1.5,h=10);
}
```

piZeroCluster-power - 3D Object

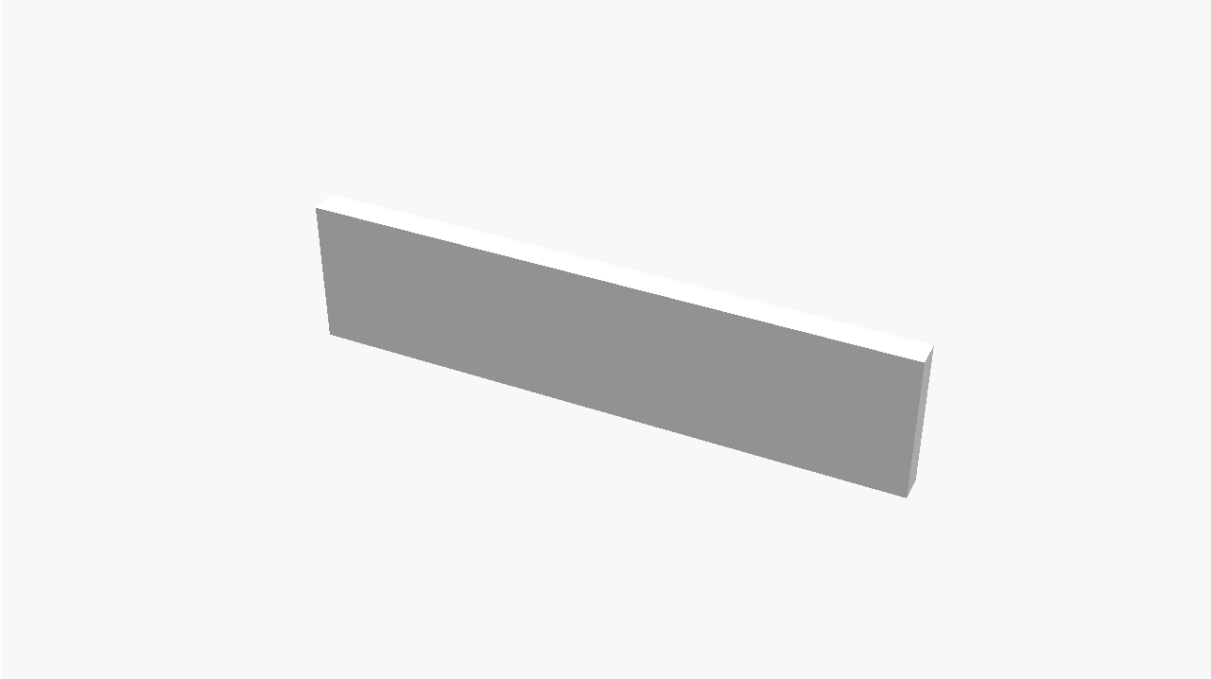

Figure 53. image

Listing 53. Openscad source

```
//measurements for the USB "iLEPO" power supply
powerW=130;
powerD=6;
powerH=32;
color([1,1,1]) cube([powerW,powerD,powerH]);

//need to create a structure to place the USB hub and pi cluster on this power
hub
// first build structure over this.
// this has rounded corners take that into account by building with cylinders
and hull
// then build USB hub structure that fits on to this
// then add mounting pins to top of overall strucute for the piZero mounting
brackets
// potentially add hood for cluster to make it less of a dust collector
(optional)
// print in white as USB power supply is white.
// make it a super structure that goes over as opposed to a case.
// potentially use algorithmic cutouts
```

powercover - Project

corner-powercover - 3D Object

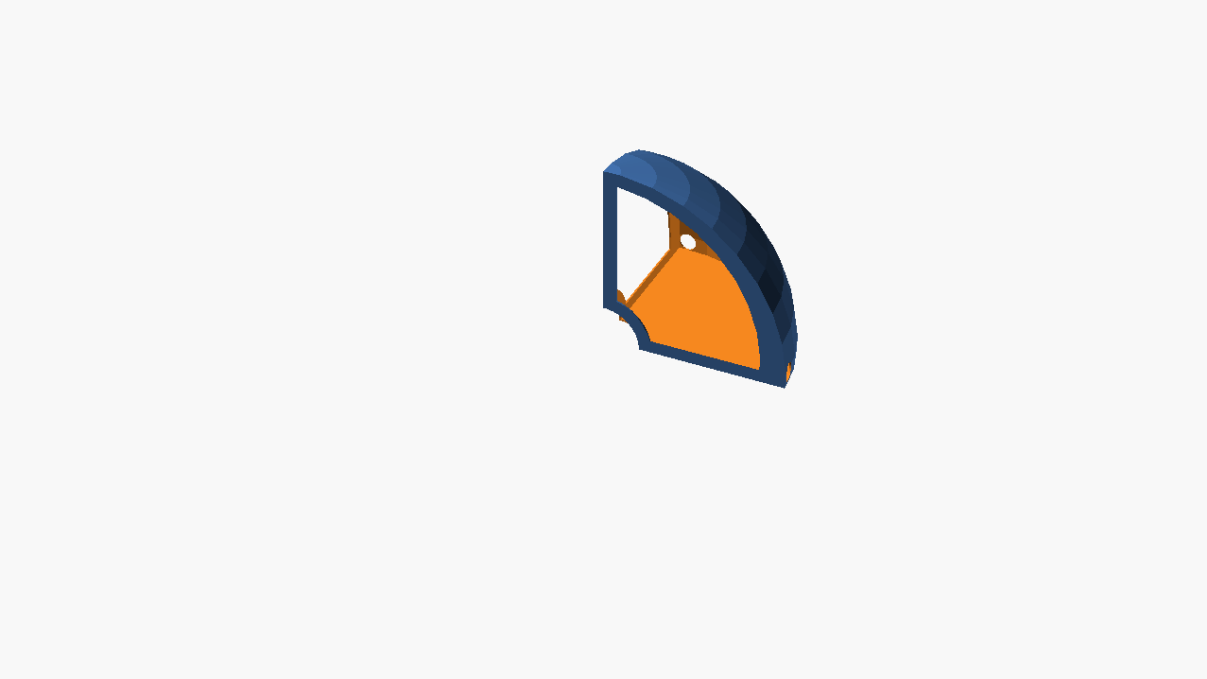

Figure 54. image

Listing 54. Openscad source

```
//in a corner there is a blasted power cable that needs a cover
$fn=36;
coverR = 100 ;
cableD = 10 ;
shellT = 1 ;
cornerR = 20 ;
sideH = 7;
//the shell to fit in the corner of oscar's room
difference(){
    difference() {
        //Whole 1/8 sphere
        intersection() {
            cube([coverR,coverR,coverR]);
            sphere(r=coverR);
        }
        translate([shellT,shellT,shellT]) intersection() {
            cube([coverR,coverR,coverR]);
            sphere(r=coverR-2*shellT);
        }
    }
    //cable passthrough
    translate([shellT+cableD/2,0,cableD/2+shellT]) rotate([-
90,90,0])cylinder(h=coverR+shellT,d=cableD);
    translate([0,cableD/2+shellT,cableD/2+shellT])
rotate([0,90,0])cylinder(h=coverR+shellT,d=cableD);
    //corner
    sphere(r=cornerR);
```

```
    //the inner inner volume for subtraction
    difference() {
        translate([0,0,sideH])intersection() {
            sphere(coverR-2*sideH);
            translate([-coverR,-coverR,0]) cube([2*coverR,2*coverR,coverR]);
        }
        translate([-sideH,-sideH,0])cube([2*sideH,2*sideH,coverR]);
        sphere(r=cornerR+sideH);
    }
}
```

projector - Project

lid - 3D Object

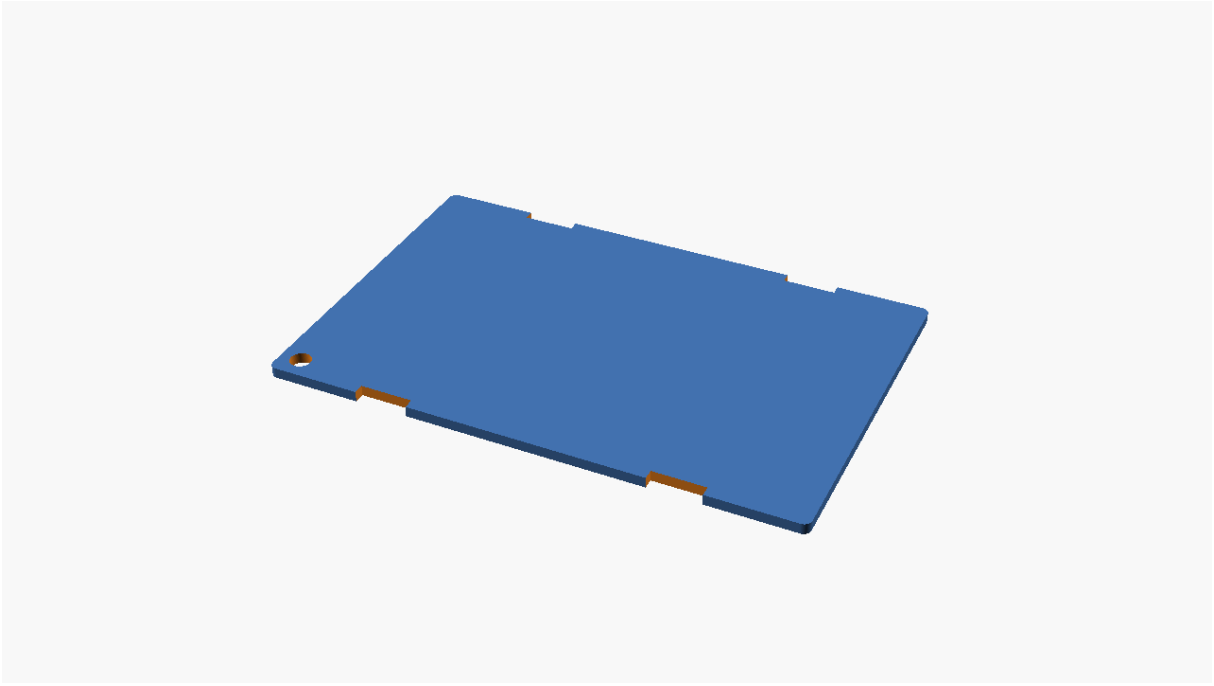

Figure 55. image

Listing 55. Openscad source

```
$fn=100;
screwX=91.5;
screwY=56.5;
screwD=4;

//inside of midleton wooden box with double doors
totHeight = 2 ;
totWidth = 101 ;
totDepth = 66.8 ;
dimensions = [ totWidth, totDepth, totHeight] ;
wiggle = [0, 0, 0] ;
```

```
volume = dimensions + wiggle ;
*cube(volume);
// Corner strength
cornerD = 3 ;
Blob=[cornerD,cornerD,volume.z];
DIM = volume ;

FL = [0, 0, 0] ;
FR = [DIM.x - Blob.x, 0, 0] ;
BR = [DIM.x - Blob.x, DIM.y -Blob.y, 0] ;
BL = [0, DIM.y -Blob.y, 0] ;

// List of corners
Corners = [FL,FR,BL,BR];

difference() {
    translate([Blob.x/2,Blob.y/2,0]) hull() {
        for (corner = Corners) {
            echo(corner);
            translate(corner) cylinder(h=DIM.z,d=Blob.x,center=true);
        }
    }
    translate([4.1,4.1,0]) cylinder(h=DIM.z+.1,d=screwD,center=true);
    notchW=10;
    notchOFF=18;
    translate ([notchOFF + (notchW/2), 0, 0]) cube([notchW, 2*2, DIM.z + .1],
center=true);
    translate ([DIM.x - notchOFF - (notchW/2), 0, 0]) cube([notchW, 2*2, DIM.z +
.1], center=true);
    translate ([notchOFF + (notchW/2), DIM.y, 0]) cube([notchW, 2*2, DIM.z +
.1], center=true);
    translate ([DIM.x - notchOFF - (notchW/2), DIM.y, 0]) cube([notchW, 2*2,
DIM.z + .1], center=true);


}
```

rack - Project

example-rack - 3D Object

This is an openscad library and some examples used to display racks.

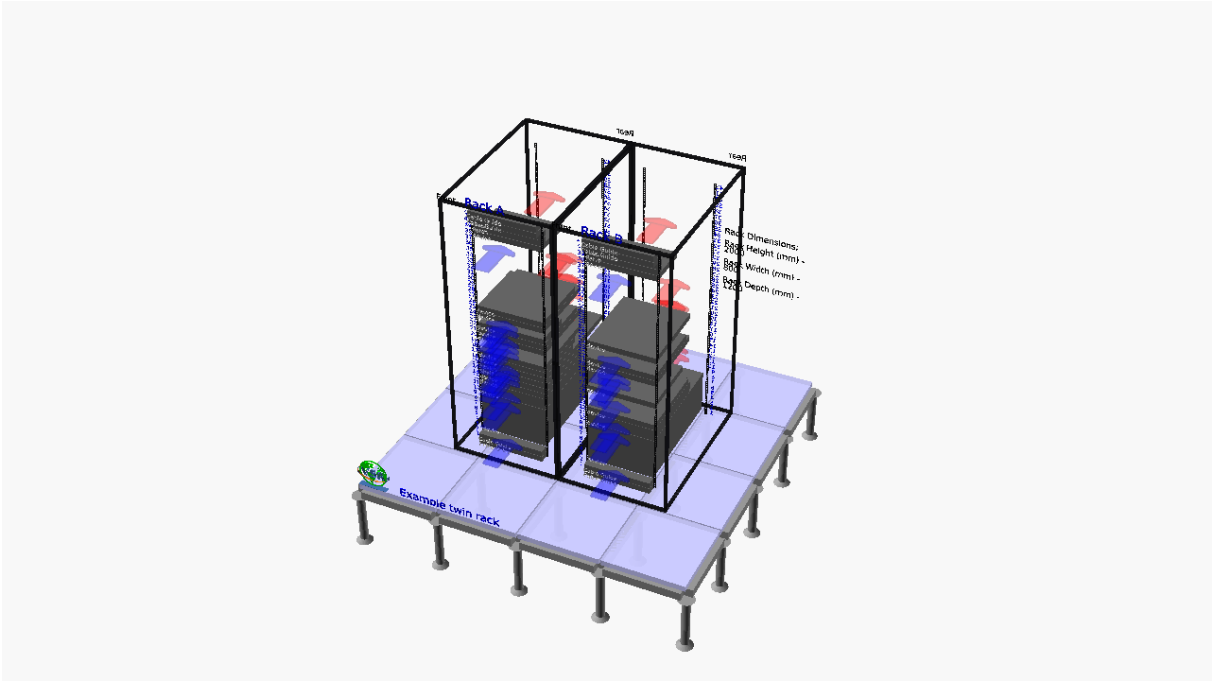

Figure 56. image

Listing 56. Openscad source

```
// Defintions for a populated Rack
// Requires the include files to be present in order to work
include <include-Settings-rack-42RU-twin-80x120x200.scad>;

// Title
floorLevelTitle = "Example twin rack";
rackTopTitle1 = "Rack A";
rackTopTitle2 = "Rack B";
useAirFlowYN = "true";
useRaisedFloorYN="true";

// Generate Devices RACK 1
translate(Rack1)
placeInRack(1,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"C
able Guide","none","none","front-outside");
translate(Rack1)
placeInRack(2,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"C
able Guide","none","none","front-outside");
translate(Rack1)
placeInRack(3,StandardRackUnitWidth,7*StandardRackUnitHeight,842,RackUnitColor,"
device","front","back","front-inside");
translate(Rack1)
placeInRack(10,StandardRackUnitWidth,2*StandardRackUnitHeight,566,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(12,StandardRackUnitWidth,1*StandardRackUnitHeight,571,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
```

```
placeInRack(13,StandardRackUnitWidth,1*StandardRackUnitHeight,560,RackUnitColor,
"device","Front","Back","front-inside");
translate(Rack1)
placeInRack(14,StandardRackUnitWidth,1*StandardRackUnitHeight,900,RackUnitColor,
"Shelf","nan","nan","front inside");
translate(Rack1)
placeInRack(15,StandardRackUnitWidth,1*StandardRackUnitHeight,429,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(16,StandardRackUnitWidth,1*StandardRackUnitHeight,457,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(17,StandardRackUnitWidth,1*StandardRackUnitHeight,571,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(20,StandardRackUnitWidth,1*StandardRackUnitHeight,566,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(21,StandardRackUnitWidth,1*StandardRackUnitHeight,502,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(22,StandardRackUnitWidth,1*StandardRackUnitHeight,269,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(24,StandardRackUnitWidth,1*StandardRackUnitHeight,566,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(25,StandardRackUnitWidth,1*StandardRackUnitHeight,467,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(38,StandardRackUnitWidth,1*StandardRackUnitHeight,305,RackUnitColor,
"device","front","back","front-inside");
translate(Rack1)
placeInRack(39,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"
Patch","none","none","front-inside");
translate(Rack1)
placeInRack(40,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"
Patch","none","none","front-inside");
translate(Rack1)
placeInRack(41,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"
Cable Guide","none","none","front-outside");
translate(Rack1)
placeInRack(42,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"
Cable Guide","none","none","front-outside");

// Generate Devices RACK 2
translate(Rack2)
placeInRack(1,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"C
able Guide","none","none","front-outside");
translate(Rack2)
```

```
placeInRack(2,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"C
able Guide","none","none","front-outside");
translate(Rack2)
placeInRack(3,StandardRackUnitWidth,7*StandardRackUnitHeight,842,RackUnitColor,"
device","front","back","front-inside");
translate(Rack2)
placeInRack(10,StandardRackUnitWidth,2*StandardRackUnitHeight,600,RackUnitColor,
"device","front","back","front-inside");
translate(Rack2)
placeInRack(12,StandardRackUnitWidth,1*StandardRackUnitHeight,571,RackUnitColor,
"device","front","back","front-inside");
translate(Rack2)
placeInRack(16,StandardRackUnitWidth,1*StandardRackUnitHeight,457,RackUnitColor,
"device","front","back","front-inside");
translate(Rack2)
placeInRack(20,StandardRackUnitWidth,1*StandardRackUnitHeight,566,RackUnitColor,
"device","front","back","front-inside");
translate(Rack2)
placeInRack(21,StandardRackUnitWidth,1*StandardRackUnitHeight,502,RackUnitColor,
"device","front","back","front-inside");
translate(Rack2)
placeInRack(24,StandardRackUnitWidth,1*StandardRackUnitHeight,566,RackUnitColor,
"device","front","back","front-inside");
translate(Rack2)
placeInRack(38,StandardRackUnitWidth,1*StandardRackUnitHeight,305,RackUnitColor,
"device","front","back","front-inside");
translate(Rack2)
placeInRack(39,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"
Patch","none","none","front-inside");
translate(Rack2)
placeInRack(40,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"
Patch","none","none","front-inside");
translate(Rack2)
placeInRack(41,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"
Cable Guide","none","none","front-outside");
translate(Rack2)
placeInRack(42,StandardRackUnitWidth,1*StandardRackUnitHeight,65,RackUnitColor,"
Cable Guide","none","none","front-outside");
```

include-Modules-v1 - 3D Object

Figure 57. image

Listing 57. Openscad source

```
//Script to create a DC RU Rack
//Author Sean Donnellan
//VERSION 0.0.1

//
// Variables
//
// Global
//  an inch is 2.54cm
//  Srandard 19" rack has 1.75 inches per RU - convert to mm
Factor=2.54 * 10;
StandardRackUnitHeight=(1.75 * Factor);
// Floor covering and tiles
//
FloorColor=[200/255, 200/255, 255/255];
FloorTileHeight=40;
FloorTileGap=10;
FloorTileXOffset=600;
FloorTileYOffset=600;
FloorTile=[FloorTileXOffset - FloorTileGap,FloorTileYOffset -
FloorTileGap,FloorTileHeight];
//DC Floor supports
//
RaisedFloorHeight=500;
RaisedFloorTransparency=0.8;
RaisedFloorColor=[150/255, 150/255, 150/255];
FSd=60;
```

```
RaisedFloorStrutDiameter=60;
FloorCarrierDiameter=60;
// Rack
//
StandardRackUnitWidth=19 * Factor;
//NumRackUnits=42;
//RackRackUnitDepth=650;
//RackWidth=800;
//RackDepth=1200;
//RackHeight=2000;
////first Rail mount hole offset from floor (normally one RU is enough for
visuals Exaxt amount can also be entered)
//RailHeightOffset=StandardRackUnitHeight;
SpecsIndent=100;
RackFrameThickness=20;
//RAL9005 14,14,16
RackColor=[14/255, 14/255, 16/255];
//Variables Are hard coded as they are standard and do not vary
RailWidth=15.875;
RailDepth=5;
CageNutWidth=9;
RackNutPos1=6.35;
RackNutPos2=(RackNutPos1+15.875);
RackNutPos3=(RackNutPos2+15.875);
// Devices
//
//Cosmetic gap
RackUnitGap=3;
RackUnitColor=[100/255, 100/255, 100/255];
RackUnitColorRed=[200/255, 100/255, 100/255];
RackUnitColorGreen=[100/255, 200/255, 100/255];
RackUnitColorBlue=[100/255, 100/255, 200/255];
RackUnitColorYellow=[200/255, 200/255, 100/255];
Rotation=[0,0,0];
Translation=[0,0,0];
// Title
//
FloatLabelColor=[1,1,1];
FloatLabelColorTitle=[0,0,1];
//Label transparency - 1=solid 0=invisible 0.5=half transparent
LabelT=1;
// Air flow
//
afTransp=.23;
afCold=[0,0,1];
afHot=[1,0,0];
afArrowSize=300;

//
// Modules
```

```
//

// Rack mounted Devices
//
//
module
placeInRack(RUp,RackUnitWidth,RackUnitHeight,RackUnitDepth,RackUnitColor,Label,A
FIn,AFOut,Side){
    XWiggle=RackWidth - StandardRackUnitWidth;
    ZWiggle=RackDepth - RackRackUnitDepth;
    OffsetInRack=[ XWiggle/2 , ZWiggle/2 , ((RUp * StandardRackUnitHeight)-
(StandardRackUnitHeight))+(RailHeightOffset)];

    translate(OffsetInRack)

multiRackUnit(RackUnitWidth,RackUnitHeight,RackUnitDepth,RackUnitColor,Label,AFI
n,AFOut,Side);
}

module multiRackUnit
(RackUnitWidth,RackUnitHeight,RackUnitDepth,RackUnitColor,Label,afin,afout,Side)
{
    Rotation=
        Side=="front-inside" ? [0,0,0] :
        Side=="right-inside" ? [0,0,90] :
        Side=="rear-inside" ? [0,0,180] :
        Side=="left-inside" ? [0,0,270] :
        Side=="front-inside-reverse" ? [0,0,180] :
        Side=="right-inside-reverse" ? [0,0,270] :
        Side=="rear-inside-reverse" ? [0,0,0] :
        Side=="left-inside-reverse" ? [0,0,90] :
        Side=="front-outside" ? [0,0,0] :
        Side=="right-outside" ? [0,0,90] :
        Side=="rear-outside" ? [0,0,180] :
        Side=="left-outside" ? [0,0,270] :
        Side=="front-outside-reverse" ? [0,0,180] :
        Side=="right-outside-reverse" ? [0,0,270] :
        Side=="rear-outside-reverse" ? [0,0,0] :
        Side=="left-outside-reverse" ? [0,0,90] :
        Side=="frp-inside" ? [0,0,0] :
        Side=="frp-outside" ? [0,0,0] :
        Side=="flp-inside" ? [0,0,0] :
        Side=="flp-outside" ? [0,0,0] :
        Side=="rrp-inside" ? [0,0,0] :
        Side=="rrp-outside" ? [0,0,0] :
        Side=="rlp-inside" ? [0,0,0] :
        Side=="rlp-outside" ? [0,0,0] :
        Side=="shelf-c" ? [0,0,0] :
        Side=="shelf-r" ? [0,0,0] :
        Side=="shelf-l" ? [0,0,0] :
```

```
        [0,0,0] ;

    Translation=
        Side=="front-inside" ? [(StandardRackUnitWidth/2)-(RackUnitWidth/2),0,0]
:
        Side=="right-inside" ? [StandardRackUnitWidth,(RackRackUnitDepth/2)-
(RackUnitWidth/2),0] :
        Side=="rear-inside" ?
[(StandardRackUnitWidth/2)+(RackUnitWidth/2),RackRackUnitDepth,0] :
        Side=="left-inside" ? [0,(RackRackUnitDepth/2)+(RackUnitWidth/2),0] :
        Side=="front-inside-reverse" ?
[(StandardRackUnitWidth/2)+(RackUnitWidth/2),RackUnitDepth,0] :
        Side=="right-inside-reverse" ? [StandardRackUnitWidth-
RackUnitDepth,(RackRackUnitDepth/2)+(RackUnitWidth/2),0] :
        Side=="rear-inside-reverse" ? [(StandardRackUnitWidth/2)-
(RackUnitWidth/2),RackRackUnitDepth-RackUnitDepth,0] :
        Side=="left-inside-reverse" ? [RackUnitDepth,(RackRackUnitDepth/2)-
(RackUnitWidth/2),0] :
        Side=="front-outside" ? [(StandardRackUnitWidth/2)-(RackUnitWidth/2),-
RackUnitDepth,0] :
        Side=="right-outside" ?
[StandardRackUnitWidth+RackUnitDepth,(RackRackUnitDepth/2)-(RackUnitWidth/2),0]
:
        Side=="rear-outside" ?
[(StandardRackUnitWidth/2)+(RackUnitWidth/2),RackRackUnitDepth+RackUnitDepth,0]
:
        Side=="left-outside" ? [-
RackUnitDepth,(RackRackUnitDepth/2)+(RackUnitWidth/2),0] :
        Side=="front-outside-reverse" ?
[(StandardRackUnitWidth/2)+(RackUnitWidth/2),0,0] :
        Side=="right-outside-reverse" ?
[StandardRackUnitWidth,(RackRackUnitDepth/2)+(RackUnitWidth/2),0] :
        Side=="rear-outside-reverse" ? [(StandardRackUnitWidth/2)-
(RackUnitWidth/2),RackRackUnitDepth,0] :
        Side=="left-outside-reverse" ? [0,(RackRackUnitDepth/2)-
(RackUnitWidth/2),0] :
        Side=="frp-inside" ? [0,0,0] :
        Side=="frp-outside" ? [0,0,0] :
        Side=="flp-inside" ? [0,0,0] :
        Side=="flp-outside" ? [0,0,0] :
        Side=="rrp-inside" ? [0,0,0] :
        Side=="rrp-outside" ? [0,0,0] :
        Side=="rlp-inside" ? [0,0,0] :
        Side=="rlp-outside" ? [0,0,0] :
        Side=="shelf-c" ? [(StandardRackUnitWidth/2)-
(RackUnitWidth/2),(RackRackUnitDepth/2)-(RackUnitDepth/2),0] :
        Side=="shelf-r" ? [(StandardRackUnitWidth-
RackUnitWidth),(RackRackUnitDepth/2)-(RackUnitDepth/2),0] :
        Side=="shelf-l" ? [0,(RackRackUnitDepth/2)-(RackUnitDepth/2),0] :
        [0,0,0] ;
```

```
    translate(Translation) rotate(Rotation) {
        difference() {
            rackUnitSolid(RackUnitWidth,RackUnitHeight-
RackUnitGap,RackUnitDepth,RackUnitColor);
            //render the label embossed
            *embossLabel(Label);
        }
        floatLabel(Label,FloatLabelColor,30,LabelT);
        afOffset=(afArrowSize+(afArrowSize/3))/2;
        front=[RackUnitWidth/2,-afOffset,0];
        back=[RackUnitWidth/2,RackUnitDepth+afOffset,0];
        left=[-afOffset,RackUnitDepth/2,0];
        right=[RackUnitWidth+afOffset,RackUnitDepth/2,0];
        outAdd=[0,0,21];
        if(useAirFlowYN=="true"){
            if(afin=="front"){
                translate(front) rotate([0,0,0])
afArrow(afCold,afTransp,afArrowSize);
            }else if(afin=="back"){
                translate(back) rotate([0,0,180])
afArrow(afCold,afTransp,afArrowSize);
            }else if(afin=="left"){
                translate(left) rotate([0,0,270])
afArrow(afCold,afTransp,afArrowSize);
            }else if(afin=="right"){
                translate(right) rotate([0,0,90])
afArrow(afCold,afTransp,afArrowSize);
            }
            if(afout=="front"){
                translate(front+outAdd) rotate([0,0,180])
afArrow(afHot,afTransp,afArrowSize);
            }else if(afout=="back"){
                translate(back+outAdd) rotate([0,0,0])
afArrow(afHot,afTransp,afArrowSize);
            }else if(afout=="left"){
                translate(left+outAdd) rotate([0,0,90])
afArrow(afHot,afTransp,afArrowSize);
            }else if(afout=="right"){
                translate(right+outAdd) rotate([0,0,270])
afArrow(afHot,afTransp,afArrowSize);
            }
        }
    }
}

module afArrow(temp,transp,Size){
    Height=20;
    Radius=Height/2;
    Width=Size/3;
```

```
    HeadHeight=(Size/10)*8;

    translate([-Width,-Size/2,0]) color(temp,transp) union(){
        hull(){
            //bottom of arrow base
            translate([Width/2,0,Radius]) sphere(r=Radius);
            translate([Width+Width/2,0,Radius]) sphere(r=Radius);
            //top of arrow base
            translate([Width/2,HeadHeight-Radius*3,Radius]) sphere(r=Radius);
            translate([Width+Width/2,HeadHeight-Radius*3,Radius])
sphere(r=Radius);
        }
        hull(){
            //tip
            translate([Width,Size,Radius]) sphere(r=Radius-5);
            //base of tip
            translate([0,HeadHeight,Radius]) sphere(r=Radius+7);
            translate([Width*2,HeadHeight,Radius]) sphere(r=Radius+7);
        }
    }
}

module rackUnitSolid(RUx,RUy,RUz,RUc){
    color(RUc) cube([RUx,RUz,RUy]);
}

module embossLabel(Label){
    TextDepth=20; //how deep to extrude the text so positioning in device and
extruding out
    translate([10,TextDepth,10]) rotate([90,0,0])
        linear_extrude(height=TextDepth){text(Label, size=30);}
    ;
}

// DC Floor tiles
//
//
module DCfloor(NumTilesX,NumTilesY,Color,Transparency,RaisedFloorHeight,RFTrue){
    union(){
        for (yp=[1:FloorTileYOffset:NumTilesY * FloorTileYOffset]){
            for (xp=[1:FloorTileXOffset:NumTilesX * FloorTileXOffset]){
                translate([xp,yp,0]){
                    translate([0,0,-
FloorTileHeight]){floorTile(FloorTile,Color,Transparency);}
                    //comment the next line to hide the raised floor details
                    if(RFTrue=="true"){

raisedFloor(RaisedFloorColor,RaisedFloorHeight,RaisedFloorStrutDiameter,FloorCar
rierDiameter,FloorTileXOffset,FloorTileYOffset,FloorTileHeight);
                    }
```

```
                }
            }
        }
    }
}

module floorTile(Ft,Ftc,Ftt){
    color(Ftc,Ftt) cube(Ft);
}

// DC Raised floor
//
module raisedFloor(FSc,RFh,FSd,FCd,FTOx,FTOy,FTh){
    FloorCarrierAndTileHeight=FCd+FTh;
    SupportHeight=RFh-FloorCarrierAndTileHeight;
    SupportOffset=SupportHeight+FloorCarrierAndTileHeight;
    union(){
        translate([0,0,-SupportOffset]) color(FSc)
floorSupports(FSc,SupportHeight,FSd,FTOx,FTOy);
        translate([0,0,-FTh]) color(FSc) floorTileCarriers(FCd,FTOx,FTOy);
    }
}

module floorCarrier(FCd,FCl){
    translate([FCd/2,-FCd/2,-FCd]) cube([FCl-FCd,FCd,FCd]);
}

module floorTileCarriers(FTCd,FTOx,FTOy){
    floorCarrier(FTCd,FTOx);
    translate([0,FTOy,0]) floorCarrier(FTCd,FTOx);
    rotate([0,0,90]) floorCarrier(FTCd,FTOy);
    translate([FTOx,0,0]) rotate([0,0,90]) floorCarrier(FTCd,FTOy);
}


module floorSupports(FSc,FSh,FSd,FTOx,FTOy){
    color(FSc){
        floorSupport(FSh,FSd);
        translate([FTOx,0,0]) floorSupport(FSh,FSd);
        translate([0,FTOy,0]) floorSupport(FSh,FSd);
        translate([FTOx,FTOy,0]) floorSupport(FSh,FSd);
    }
}

module floorSupport(FSh,FSd) {
    union(){
        cylinder(h=FSh,d=FSd);
        cylinder(h=50, r1=FSd, r2=0);
        translate([0,0,FSh-50]) cylinder(h=50, r1=0, r2=FSd);
    }
}
```

```
// Titles
//
//
module floatLabel(Label,Color,Size,LabelT){
    TextDepth=Size/10; //how deep to extrude the text
    translate ([10,-30,10]) rotate([90,0,0]) color(Color,LabelT)
linear_extrude(height=TextDepth){text(Label, size=Size);};
}

module StaticLabel(Label,Color,Size,LabelT){
    TextDepth=Size/10; //how deep to extrude the text
    translate ([0,0,0]) rotate([90,0,0]) color(Color,LabelT)
linear_extrude(height=TextDepth){text(Label, size=Size);};
}
// Utilities
//
//
//power("Power Rail A",FloatLabelColorTitle,LabelT,0,2*600+500,2790,20*600,0.1);
//
module power(Label,Labelc,LabelT,Xo,Yo,Height,Width,Rt){
    translate([Xo,Yo,Height]){
        color([200/255,200/255,200/255],Rt) cube([Width,50,200]);
        translate([1100,0,100]) floatLabel(Label,Labelc,100,LabelT);
    }
}
//
//lighting("Lighting Row
A",FloatLabelColorTitle,LabelT,0,2*600+500,2560,20*600,0.1);
//
module lighting(Label,Labelc,LabelT,Xo,Yo,Height,Width,Rt){
    translate([Xo,Yo,Height]){
        color([200/255,200/255,200/255],Rt) cube([Width,200,50]);
        translate([80,0,100]) floatLabel(Label,Labelc,100,LabelT);
    }
}
//
//sprinkler("Sprinkler Row
A",FloatLabelColorTitle,LabelT,0,2*600+400,2860,20*600,0.1);
//
module sprinkler(Label,Labelc,LabelT,Xo,Yo,Height,Width,Rt){
    translate([Xo,2*600+400,2860]){
        union(){
            color([200/255,200/255,200/255],Rt){
                rotate([0,90,0])
                    cylinder(h=Width,d=50)
                ;
                translate([1000,0,0])
                    rotate([0,0,90])
                    cylinder(h=400,d=50)
```

```
                ;
            }
        }
    //there is a 2520mm high pipe linking the sprinler rows.
    translate([500,0,0]) floatLabel(Label,Labelc,100,LabelT);
    }
}
// Rack Stuff
module
positionRack(FloorOffset,RackWidth,RackHeight,RackDepth,RackFrameThickness,RackC
olor,Label,TagsYN){
    translate(FloorOffset)
rackFrame(RackWidth,RackHeight,RackDepth,RackFrameThickness,RackColor);
    translate(FloorOffset+[0,0,RackHeight+10])
floatLabel("Front",[0,0,0],40,LabelT);
    translate(FloorOffset+[RackWidth/4,0,RackHeight+10])
floatLabel(Label,[0,0,1],60,LabelT);
    translate(FloorOffset+[RackWidth,RackDepth,RackHeight+10]) rotate([0,0,180])
floatLabel("Rear",[0,0,0],40,LabelT);
    if(TagsYN=="true"){
        translate(FloorOffset+[RackWidth+SpecsIndent,RackDepth/2,RackHeight-
3*StandardRackUnitHeight]) floatLabel("Rack Dimensions:",[0,0,0],40,LabelT);
        translate(FloorOffset+[RackWidth+SpecsIndent,RackDepth/2,RackHeight-
5*StandardRackUnitHeight]) floatLabel("Rack Height (mm) -",[0,0,0],40,LabelT);
        translate(FloorOffset+[RackWidth+SpecsIndent,RackDepth/2,RackHeight-
6*StandardRackUnitHeight]) floatLabel(str(RackHeight),[0,0,0],40,LabelT);
        translate(FloorOffset+[RackWidth+SpecsIndent,RackDepth/2,RackHeight-
8*StandardRackUnitHeight]) floatLabel("Rack Width (mm) -",[0,0,0],40,LabelT);
        translate(FloorOffset+[RackWidth+SpecsIndent,RackDepth/2,RackHeight-
9*StandardRackUnitHeight]) floatLabel(str(RackWidth),[0,0,0],40,LabelT);
        translate(FloorOffset+[RackWidth+SpecsIndent,RackDepth/2,RackHeight-
11*StandardRackUnitHeight]) floatLabel("Rack Depth (mm) -",[0,0,0],40,LabelT);
        translate(FloorOffset+[RackWidth+SpecsIndent,RackDepth/2,RackHeight-
12*StandardRackUnitHeight]) floatLabel(str(RackDepth),[0,0,0],40,LabelT);
    }
    XWiggle=RackWidth - StandardRackUnitWidth;
    ZWiggle=RackDepth - RackRackUnitDepth;
    OffsetInRack=[ XWiggle/2 , ZWiggle/2 , RailHeightOffset];
    echo(OffsetInRack);
    translate(FloorOffset) translate(OffsetInRack) rails();
}
module rackFrame(Rx,Ry,Rz,St,Rc){
    //Rx RackWidth, Ry RackHeight, Rz RackDepth, St StrutThickness, Rc RackColor
    union(){
        //front and back Struts
        translate ([0,0,0]) rackStrutX(St,Rx,Rc);
        translate ([0,Rz-St,0]) rackStrutX(St,Rx,Rc);
        translate ([0,Rz-St,Ry-St]) rackStrutX(St,Rx,Rc);
        translate ([0,0,Ry-St]) rackStrutX(St,Rx,Rc);
        //Side Struts
```

```
        translate ([0,0,0]) rackStrutZ(St,Rz,Rc);
        translate ([Rx-St,0,0]) rackStrutZ(St,Rz,Rc);
        translate ([Rx-St,0,Ry-St]) rackStrutZ(St,Rz,Rc);
        translate ([0,0,Ry-St]) rackStrutZ(St,Rz,Rc);
        //Uprights
        translate ([0,0,0]) rackStrutY(St,Ry,Rc);
        translate ([0,Rz-St,0]) rackStrutY(St,Ry,Rc);
        translate ([Rx-St,Rz-St,0]) rackStrutY(St,Ry,Rc);
        translate ([Rx-St,0,0]) rackStrutY(St,Ry,Rc);
    }
}

module rackStrutX(RackFrameThickness,RackWidth,Color){
    //all measurements in mm
    color(Color,0.8) cube([RackWidth,RackFrameThickness,RackFrameThickness]);
}
module rackStrutZ(RackFrameThickness,RackDepth,Color){
    //all measurements in mm
    color(Color,0.8) cube([RackFrameThickness,RackDepth,RackFrameThickness]);
}
module rackStrutY(RackFrameThickness,RackHeight,Color){
    //all measurements in mm
    color(Color,0.8) cube([RackFrameThickness,RackFrameThickness,RackHeight]);
}

module rails() {
    posts=[["FR",[StandardRackUnitWidth,0,0],"false"],["FL",[-
RailWidth,0,0],"true"],["BR",[-
RailWidth,RackRackUnitDepth,0],"false"],["BL",[StandardRackUnitWidth,RackRackUni
tDepth,0],"true"]];
    for (outl=[0:len(posts)-1]) {
        for(i=[1:1:NumRackUnits]) {
            var=posts[outl];
            FBLR=var[0];
            Translate=var[1];
            LabelYN=var[2];
            translate(Translate+[0,0,(i*StandardRackUnitHeight)-
StandardRackUnitHeight]) railSection(LabelYN,"true",RackColor,str(i),FBLR);
        }
    }
}
module railSection (LabelYN,RailYN,Color,RU,RailFBLR) {
    //module creates a rack rail section for exactly one RU
    //Front right or left and rear right or left post plus RU numbering
    Translation=
        RailFBLR=="FL" ? [-50,4,10] :
        RailFBLR=="FR" ? [20,4,10] :
        RailFBLR=="BL" ? [65,0,10] :
        RailFBLR=="BR" ? [0,0,10] :
        [0,0,0] ;
```

```
    Rotation=
        RailFBLR=="FL" ? [0,0,0] :
        RailFBLR=="FR" ? [0,0,0] :
        RailFBLR=="BL" ? [0,0,180] :
        RailFBLR=="BR" ? [0,0,180] :
        [0,0,0] ;
    if(RailYN=="true"){
        color(Color) difference(){
            cube([RailWidth,RailDepth,StandardRackUnitHeight]);
            translate([(RailWidth-CageNutWidth)/2,-.1,RackNutPos1])
translate([0,0,-CageNutWidth/2]) cube([CageNutWidth,CageNutWidth,CageNutWidth]);
            translate([(RailWidth-CageNutWidth)/2,-.1,RackNutPos2])
translate([0,0,-CageNutWidth/2]) cube([CageNutWidth,CageNutWidth,CageNutWidth]);
            translate([(RailWidth-CageNutWidth)/2,-.1,RackNutPos3])
translate([0,0,-CageNutWidth/2]) cube([CageNutWidth,CageNutWidth,CageNutWidth]);
        }
    }
    if(LabelYN=="true"){
        translate(Translation) rotate(Rotation)
StaticLabel(RU,FloatLabelColorTitle,30,1);
    }
}

if (library) {} else  {
    echo("trying to compile a library!");
    linear_extrude(height = 4) {
        text("trying to compile a library!");
        }
    }

// User Data
// after this
```

include-Settings-rack-42RU-twin-80x120x200 - 3D Object

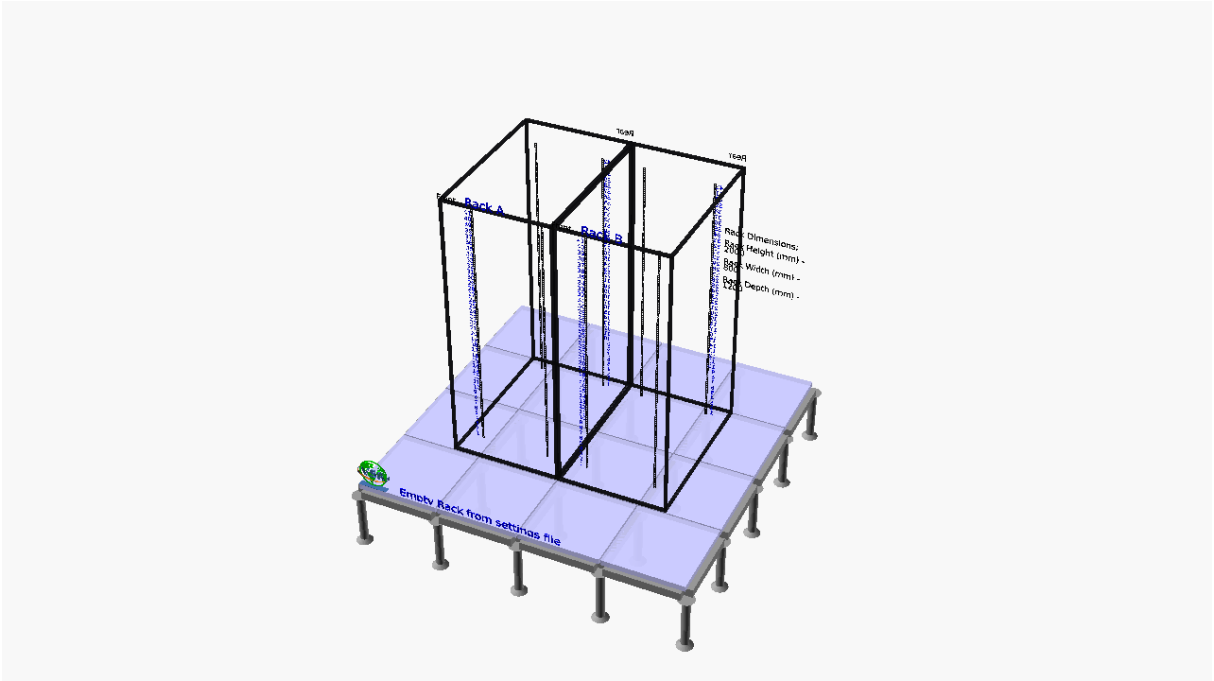

Figure 58. image

Listing 58. Openscad source

```
//42 RU twinrack 80x120x200
// Requires the include files to be present in order to work
include <include-Modules-v1.scad>;
include <logo-VSR.scad>;
library="true"; //set this to remove the warning when compiling the library on
its own
//Rack Dimension Variables
NumRackUnits=42;
RackRackUnitDepth=900;
RackWidth=800;
RackDepth=1200;
RackHeight=2000;

//default titles
floorLevelTitle = "Empty Rack from settings file";
rackTopTitle1 = "Rack A";
rackTopTitle2 = "Rack B";

//first Rail mount hole offset from floor (normally one RU is enough for visuals
Exact amount can also be entered)
RailHeightOffset=StandardRackUnitHeight;

// Generate DC Floor (each tile is 600x600 unless otherwise specified in the
main modules)
useRaisedFloorYN="true";
DCfloor(4,4,FloorColor,RaisedFloorTransparency,RaisedFloorHeight,useRaisedFloorY
N);
```

```
// Define Positions on DC Floor
RowOffset0=[400,600,0];
Rack1=[RackWidth*0,0,0]+RowOffset0;
Rack2=[RackWidth*1,0,0]+RowOffset0;
//invisible rack for optional animation (pop out)
Rack1A=[RackWidth*0,-RackDepth*$t,0]+RowOffset0;
Rack2A=[RackWidth*1,-RackDepth*$t,0]+RowOffset0;
//invisible rack for optional animation (pop in)
Rack1B=[RackWidth*0,(RackDepth*($t))-RackDepth,0]+RowOffset0;
Rack2B=[RackWidth*0,(RackDepth*($t))-RackDepth,0]+RowOffset0;

// Title
translate([0,0,-.5]) logo();
translate([300,50,-10])
floatLabel(floorLevelTitle,FloatLabelColorTitle,60,LabelT);

// Don't modify these. Instead USE them in the section to generate the racks
below
RackBSidebarInfoON="true";
RackASidebarInfoON="false";

//Generate Racks
positionRack(Rack1,RackWidth,RackHeight,RackDepth,RackFrameThickness,RackColor,r
ackTopTitle1,RackASidebarInfoON);
positionRack(Rack2,RackWidth,RackHeight,RackDepth,RackFrameThickness,RackColor,r
ackTopTitle2,RackBSidebarInfoON);
```

logo-VSR - 3D Object

Figure 59. image

Listing 59. Openscad source

```
//VSR logo
module outline_text (size,text) {
    $fn=100;
    font = "DejaVu Sans:style=Bold";
    letter_size = size;
    height = 10;
    string = text;
    textlen = len(string);
    linear_extrude(height) {
        difference() {
            offset(r=-1) {
                text(string, size = letter_size, font = font, halign = "center",
valign = "center", $fn = 64);
            }

            offset(r=-5) {
                text(string, size = letter_size, font = font, halign = "center",
valign = "center", $fn = 64);
            }
        }
    }
}

//create an approxiumated orbit from an elipsoid
module orbit(size) {

    difference(){
        scale([1,.5,1]) linear_extrude(height=12)circle(d=size+30);
        translate([0,0,-.1])scale([1,.5,1])
linear_extrude(height=12.2)circle(d=size+20);
    }
}

module tri(size,rot,height){

rotate([0,0,rot])linear_extrude(height=height)polygon([[0,0],[size*2,0],[size,si
ze*2]]);
}

module orbiter(imgW) {
    intersection() {
        difference(){
            linear_extrude(height=20) circle(d=imgW);
            translate([-imgW/2,0,-.1])cube([ imgW, imgW/2, 20 + .2]);
        }
        orbit(imgW);
    }
    difference() {
```

```
        size=imgW;
        orbit(size);
        translate([0,0,-.1]) linear_extrude(height=12.2)circle(d=size);
        translate([0,0,-.1]) cube([imgW,imgW,20]);
    }
}

module logo() {
    $fn=100;
    imgW=212;
    textH=60;
    depth=20;
    translate([imgW/2,depth*1.5,imgW/2]) rotate([90,0,0]) {
        color([0,1,0])linear_extrude(height=4) import("logo-earth.svg",center =
true);
        outline_text(textH,"VSR");
        translate([imgW/2+3,0,0])tri(10,-depth,16);
        orbiter(imgW);
    }
    translate([0,0,0]) cube([imgW,depth*2,2]);
}

if (library) {} else  {
    logo();
    }
```

rePhone - Project

- rephone components and case design
- Stl files added from thingiverse
- Scad file also as holder for stls from thingiverse
- Reference material only for now

RePhone_ALL - 3D Object

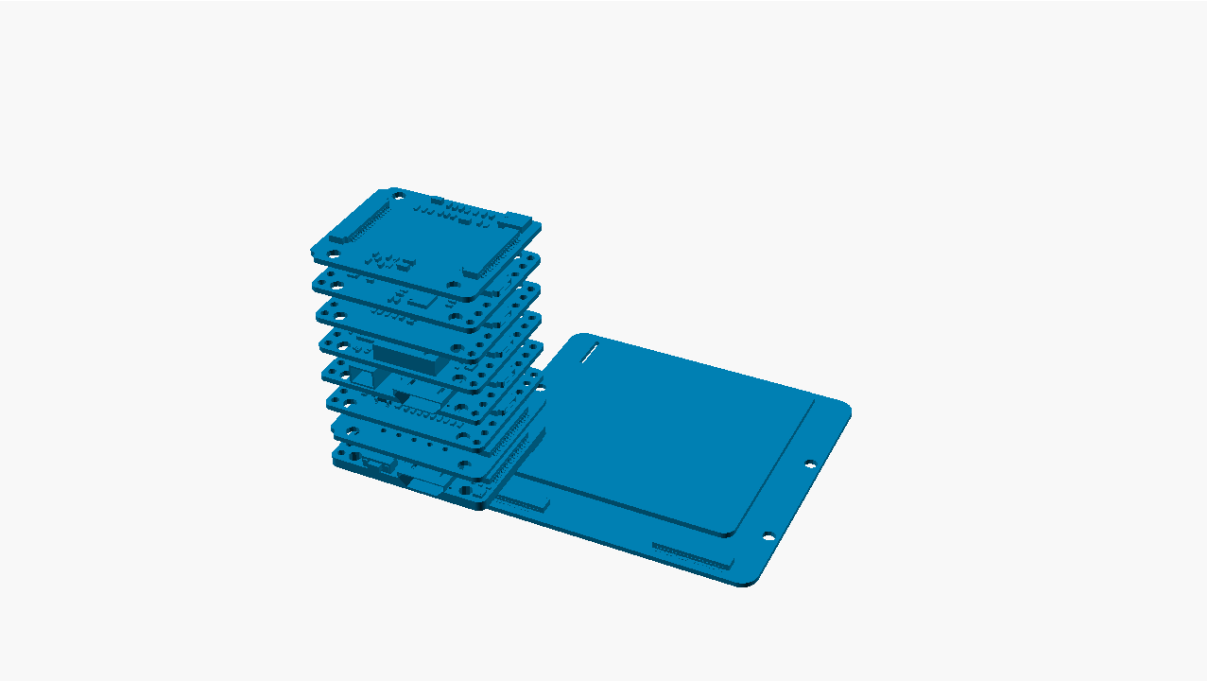

Figure 60. image

Listing 60. Openscad source

```
cube([1,1,1]);
// The Rephone modules as modules and aligned over mounting holes

color_def = [0,0.5,0.7];

//------------------------------------
// The modules
module GSM_BLE(color=color_def) {
    color(color)
    translate([0.65,-0.61,5])
        import("Xadow_GSM_BLE_v1_collapsed.stl");
}
module GSM_Breakout(color=color_def) {
    color(color)
    translate([0,0,10])
        import("Xadow___GSM_Breakout_v1_collapsed.stl");
}
module Basic_Sensors(color=color_def) {
    color(color)
    translate([-11.4,-22.8,15])
        import("Xadow_Basic_Sensors_v1_collapsed.stl");
}
module Duino(color=color_def) {
    color(color)
    translate([43.85,9.55,20])
    rotate([0,0,180])
        import("Xadow_Duino_v1_collapsed.stl");
}
```

```
module GPS(color=color_def) {
    color(color)
    translate([-12.05,-10.77,25])
        import("Xadow_GPS_v2_collapsed.stl");
}
module LED_5x7(color=color_def) {
    color(color)
    translate([-3.28,-15.98,30])
        import("Xadow_LED_5x7_v1_collapsed.stl");
}
module NFC(color=color_def) {
    color(color)
    translate([-12.05,-10.77,35])
        import("Xadow_NFC_v2_collapsed.stl");
}
module 1_54_Touhscreen(color=color_def) {
    color(color)
    translate([-50,0,-5])
        import("Xadow_1_54_Touhscreen_collapsed.stl");
}
module Audio(color=color_def) {
    color(color)
    translate([-91.42,-45.06,40])
        import("Xadow_Audio_v1.stl");
}


//-----------------------------
// Line them up :)

GSM_BLE();
GSM_Breakout();
Basic_Sensors();
Duino();
GPS();
LED_5x7();
NFC();
Audio();
1_54_Touhscreen();
```

RePhone_handset - 3D Object

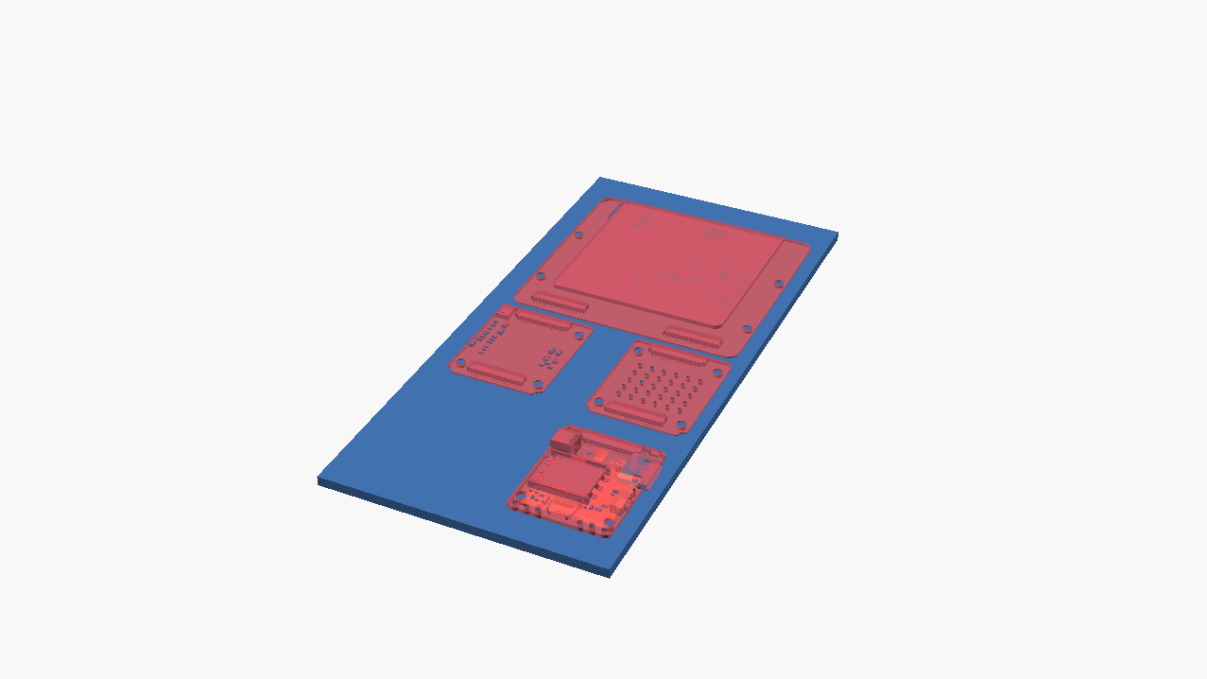

Figure 61. image

Listing 61. Openscad source

```
// The Rephone modules as modules and aligned over mounting holes

color_def = [0,0.5,0.7];

//------------------------------------
// The modules
module 1_54_Touhscreen(color=color_def) {
    color(color)
        translate([-50,0,0])
            import("Xadow_1_54_Touhscreen_collapsed.stl");
}
module GSM_BLE(color=color_def) {
    color(color)
        translate([0.65,-0.61,0])
            rotate([0,0,90])
                import("Xadow_GSM_BLE_v1_collapsed.stl");
}
module GSM_Breakout(color=color_def) {
    color(color)
        rotate([0,0,90])
            translate([0,0,0])
                import("Xadow___GSM_Breakout_v1_collapsed.stl");
}
module Audio(color=color_def) {
    color(color)
        rotate([0,0,90])
            translate([-91.42,-45.06,0])
                import("Xadow_Audio_v1.stl");
```

```
}
difference(){
    translate([-5,-65,-2]) cube([58,120,2]);
    #union(){
        1_54_Touhscreen();
        translate ([39.5,-15,0]) GSM_Breakout();
        translate ([9.5,-15,0]) Audio();
        translate ([39,-45,0]) GSM_BLE();
    }
}
```

xadow - 3D Object

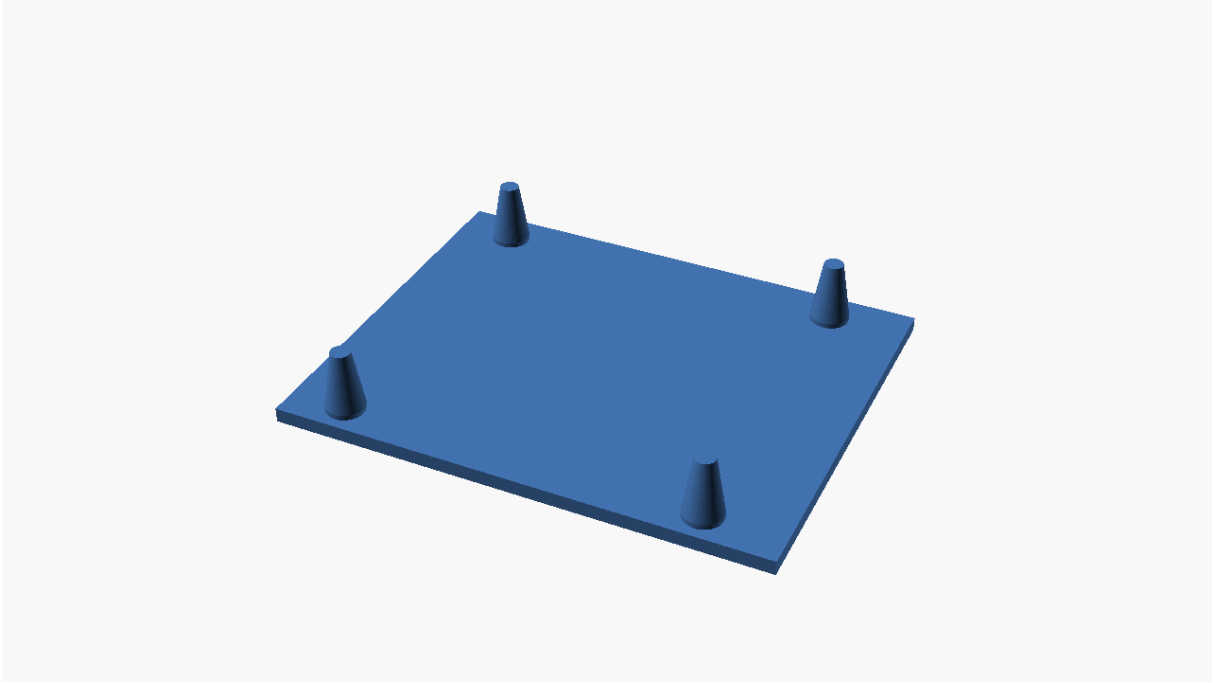

Figure 62. image

Listing 62. Openscad source

```
$fn=100;
module xadow_pin(){
    union(){
        translate([0,0,0]) cylinder(h=1,r1=1,r2=1);
        translate([0,0,1]) cylinder(h=3,r1=1,r2=.5);
    }
}
module xadow_gsm(){
    difference(){
    union(){
            //Xadow madule
            //turns out the GSM module has exactly  25.37mm X 20.30mm / 1'' X
0.8''
```

```
            //approx 2mm hole 17.5mm x18mm
            cube([25.4,20.3,.75]);
            translate([3,1.5,0]) xadow_pin();
            translate([21.4,1.5,0]) xadow_pin();
            translate([3,18.5,0]) xadow_pin();
            translate([21.4,18.5,0]) xadow_pin();
            }
            *translate([25.4,20.3,0]) cylinder(h=1,r1=1,r2=1);
        }
}

xadow_gsm();
```

reolink - Project

doorbell-mount - 3D Object

Created to mount a reolink doorbell and make it more recognizable.

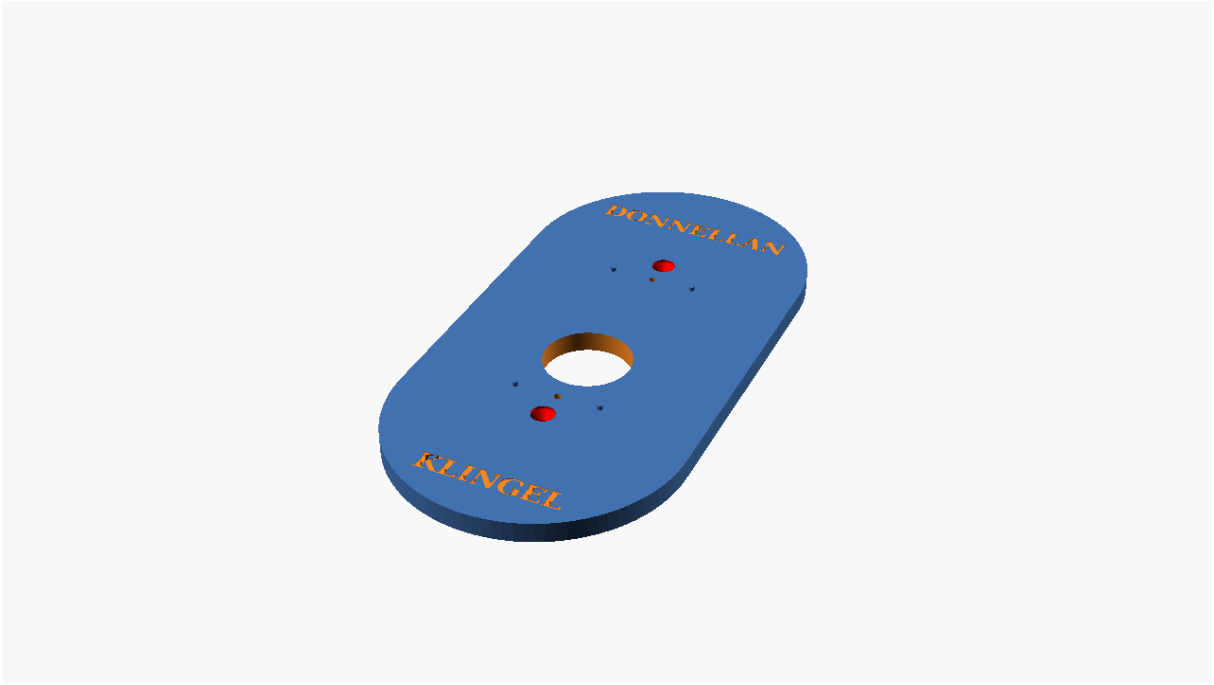

Figure 63. image

Listing 63. Openscad source

```
$fn=100;
baseH=7;
baseD=100;
baseL=200;
holeSpacing=95;
screwSpacing=75;
mountD=2.5;
```

```
flaringD1=3.5;
flaringH=3;
flaringD2=8;
textTop="DONNELLAN";
textBottom="KLINGEL";
module BLOCKTEXT (content,pos) {
    translate(pos)
        linear_extrude(height = 2)
            text(content, size = 7, direction = "ltr", spacing = 1,
valign="center",halign="center", font = "DejaVu Serif:style=bold");
}
module ScrewHole (topH,topD,flaringH,shaftH,shaftD) {
    //screw is relative to the top of the flaring
    //this is a subtractive component
    wiggle=0.001;
    color([1,0,0]) union() {
        //top clearance
        translate([0,0,-wiggle]) cylinder(h=topH+wiggle,d=topD);
        //flaring
        translate([0,0,-flaringH]) cylinder(h=flaringH,d2=topD,d1=shaftD);
        //bottom shaft
        translate([0,0,-shaftH-flaringH]) cylinder(h=baseH,d=flaringD1);
    }
}
//mounting plate
module doorbell () {
    difference () {
        //plate
        translate([0,baseD/2,0]) hull(){
            cylinder(h=baseH,d=baseD);
            translate([0,baseL-baseD,0]) cylinder(h=baseH,d=baseD);
        }
        BLOCKTEXT(textBottom,[0,baseD/6,baseH-.25]);
        BLOCKTEXT(textTop,[0,baseL-baseD/4,baseH-.25]);
        //screwholes
        translate([0,0,-1]+[0,baseL/2-holeSpacing/2,baseH])
ScrewHole(10,flaringD2,flaringH,5,flaringD1);
        translate([0,0,-1]+[0,baseL/2+holeSpacing/2,baseH])
ScrewHole(10,flaringD2,flaringH,5,flaringD1);
        //mount holes
        translate([1.7,baseL/2-screwSpacing/2,0]) rotate([0,-12.5,0])
translate([0,0,-1])cylinder(h=baseH+2,d=mountD);
        translate([1.7,baseL/2+screwSpacing/2,0]) rotate([0,-12.5,0])
translate([0,0,-1])cylinder(h=baseH+2,d=mountD);
        //cable truss
        translate([0,baseL/2-15,-.1]) cylinder(h=baseH+.2,d=30);
    }
    //bottom mount alignment tabs
    translate([-15,baseL/2-screwSpacing/2,baseH]) cylinder(h=1.5,d=2);
    translate([+15,baseL/2-screwSpacing/2,baseH]) cylinder(h=1.5,d=2);
```

```
    //top mount alignment tabs
    translate([-15,baseL/2+screwSpacing/2,baseH]) cylinder(h=1.5,d=2);
    translate([+15,baseL/2+screwSpacing/2,baseH]) cylinder(h=1.5,d=2);
}

intersection() {
    doorbell();
    *translate([0,52.5,0]) cylinder(h=20,d=15);
    *translate([-5,42.5,0]) cube([10,10,20]);
}
*translate([0,60,baseH]) color([0,0,0])
hull(){
    cylinder(h=20,d=50);
    translate([0,130-50,0]) cylinder(h=20,d=50);
}
```

schuko - Project

plug2 - 3D Object

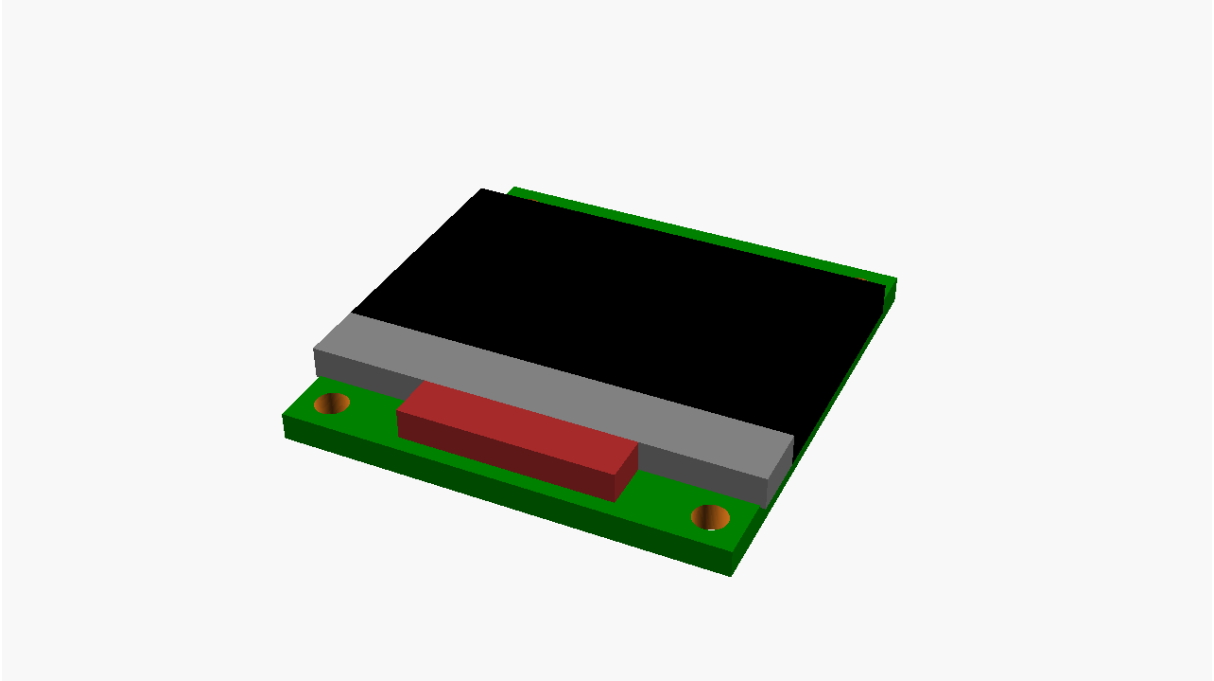

Figure 64. image

Listing 64. Openscad source

```
// mental gymnastics for centered and parametric 3d object

$fn=100;

//Variable for subtractions so as to be slightly above borders
diffWiggle = .2;
```

```
//PCB dimensions
PCB1306holeD = 2;
PCB1306holeOff = [2,2,0] ;
    PCB1306Z = 1.7 ;
    PCB1306X = 26.9 ;
    PCB1306Y = 27.9 ;
PCB1306 = [PCB1306X, PCB1306Y, PCB1306Z] ;
//1306 Top components
    LCDmaskY = 4; //how much to cover up at the bottom
    LCDX = 27.5 ; // left to right
    LCDY = 20 ; // topR to bottomR
    LCDZ = 2 ; // height from PCB
    LCDflexX = 13 ; //flex cable width
    LCDflexY = 3 ; //flex cable length from LCD to edge
LCD = [LCDX, LCDY, LCDZ] ;
LCDpos = [0, 0, PCB1306.z] ; //sits on top of the PCB and is centered
LCDviewPos = [0, LCDmaskY/2, PCB1306.z] ; //on top of the PCB above the masked
part
LCDmask = [LCD.x, LCDmaskY, LCD.z] ; // is part of the LCD so shares X and Z
LCDmaskPos = [0, -LCD.y/2 +LCDmask.y/2, LCDpos.z] ; // sits below the viewport
of the LCD
LCDview = [LCD.x, LCD.y - LCDmask.y, LCD.z]; // is part of the LCD just without
the masked part
LCDflex = [LCDflexX, LCDflexY, LCD.z] ; //is considered as high as the LCD
LCDflexPos = [0, -LCD.y/2 -LCDflex.y/2, PCB1306.z] ; //sits bellow the LCD
//1306 bottom clearance items

//array - put the parts together
extrudeFalse = false ; extrudeTrue = true ;
object = [
    [PCB1306, [0, 0, 0], "green", extrudeFalse],
    [LCDview, LCDviewPos, "black", extrudeTrue],
    [LCDflex, LCDflexPos, "brown", extrudeTrue],
    [LCDmask, LCDmaskPos, "grey", extrudeTrue]
];

module pegs(XYZ,offset,holeD) {
    //mounting holes - no need to zdiff as centered
    //relative positions
        H = XYZ.z ;
        XY = [XYZ.x, XYZ.y, 0] ;
        TR= [ [+1, 0, 0], [0, +1, 0], [0, 0, 0] ];
        TL= [ [-1, 0, 0], [0, +1, 0], [0, 0, 0] ];
        BR= [ [+1, 0, 0], [0, -1, 0], [0, 0, 0] ];
        BL= [ [-1, 0, 0], [0, -1, 0], [0, 0, 0] ];
        // move to TR then move back towards BL by offset etc
        posTR = (TR * XY/2) + (offset * BL) ;
        posTL = (TL * XY/2) + (offset * BR) ;
        posBR = (BR * XY/2) + (offset * TL) ;
```

```
        posBL = (BL * XY/2) + (offset * TR) ;
    translate (posTR) cylinder(h = H, d = holeD, center = true);
    translate (posTL) cylinder(h = H, d = holeD, center = true);
    translate (posBR) cylinder(h = H, d = holeD, center = true);
    translate (posBL) cylinder(h = H, d = holeD, center = true);
}

module brickLayer(array) {
    module blocks(list) {
        translate (list[1]) color(list[2]) cube(list[0], center=true);
    }
    for ( i = [0 : len(array) - 1] ) { blocks(array[i]); }
}
//OUTPUT
difference(){
    brickLayer(object);
    pegs(PCB1306 + [0, 0, diffWiggle], PCB1306holeOff, PCB1306holeD);
}
```

schuko-plug - 3D Object

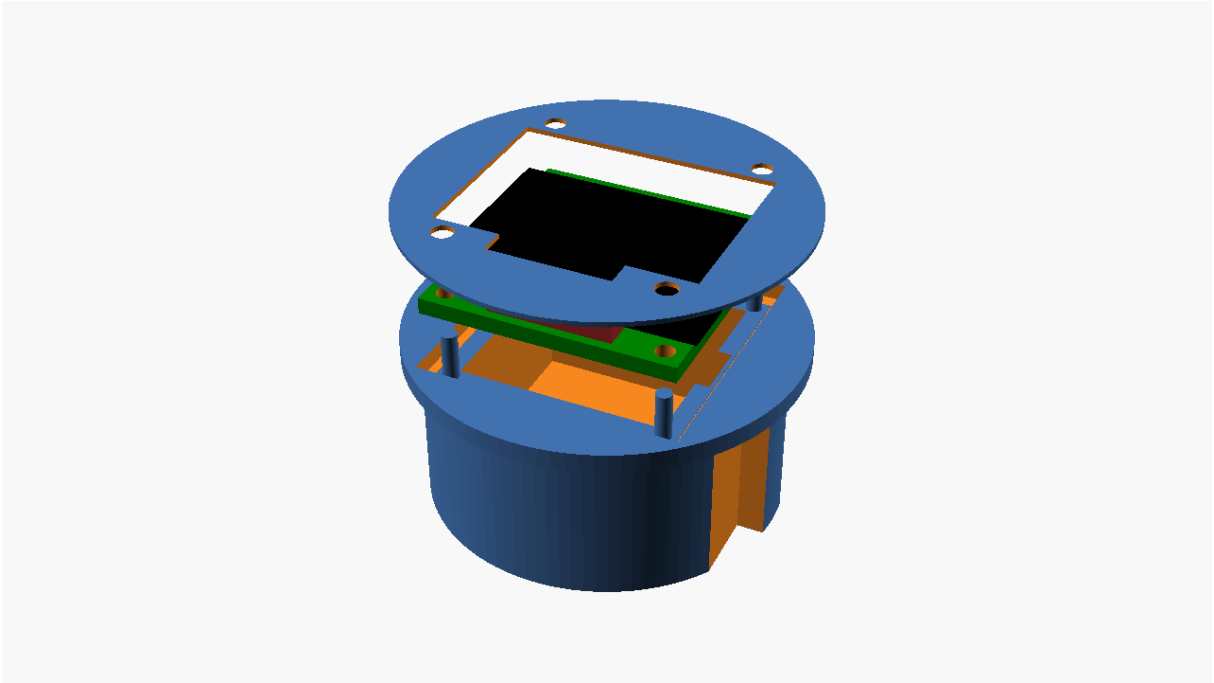

Figure 65. image

Listing 65. Openscad source

```
// Shuko plug
$fn = 100;
height = 19; //total height to top rim
plugTopD = 42; //diameter of top cover
plugTopH = 2; //2mm rim height of top cover
plugTopOff = 16.8 ; //from bottom to top rim
```

```
diffWiggle = .2;
diffWiggleA = [diffWiggle, diffWiggle, diffWiggle];
diffWiggleX = [diffWiggle, 0, 0];
diffWiggleY = [0, diffWiggle, 0];
diffWiggleZ = [0, 0, diffWiggle];
plugBottomD = 38;
plugBottomH = plugTopOff;
plugSideCutH = 3;
plugSideCutW = 5;
zdiff = [0,0,-diffWiggle/2];
cutCube = 8;
cubeXY = plugBottomD-(plugSideCutH+9);
cubeFloor = 2;
pinR = 9.5 ; //The distance from the center that the 220v power pins should be
at

//ssd1306 variables
ssd1306X = 26.9 ;
ssd1306Y = 27.9 ;
ssd1306off = [2,2,0] ;
ssd1306XY = [ssd1306X,ssd1306Y,0] ;
ssd1306PCBH = 1.7 ;
ssd1306PCBZ = [0, 0, ssd1306PCBH] ;
ssd1306PCBdim = ssd1306PCBZ + ssd1306XY ;
ssd1306mountD = 2 ;
LCDX = 27.5 ; // left to right
LCDY = 20 ; // topR to bottomR
LCDZ = 2 ; // height from PCB
LCDflexW = 13 ; //flex cable width
LCDflexH = 3 ; //flex cable length from LCD to edge
LCDmask = 4; //how much to cover up at the bottom
LCDdim = [LCDX,LCDY,LCDZ]; //Dimensions
FLEXdim = [LCDflexW,LCDflexH,LCDdim.z]; //Dimensions
LCDdimXY = [LCDX, LCDY, 0]; //XY Dimensions only without Z

module pegs(XYdimensions,offset,height,diameter) {
    //mounting holes - no need to zdiff as centered
    //relative positions
        TR= [ [+1, 0, 0], [0, +1, 0], [0, 0, 0] ];
        TL= [ [-1, 0, 0], [0, +1, 0], [0, 0, 0] ];
        BR= [ [+1, 0, 0], [0, -1, 0], [0, 0, 0] ];
        BL= [ [-1, 0, 0], [0, -1, 0], [0, 0, 0] ];
        // move to TR then move back towards BL by offset etc
        mPosTR = (TR * XYdimensions/2) + (offset * BL) ;
        mPosTL = (TL * XYdimensions/2) + (offset * BR) ;
        mPosBR = (BR * XYdimensions/2) + (offset * TL) ;
        mPosBL = (BL * XYdimensions/2) + (offset * TR) ;
    translate (mPosTR) cylinder(h = height, d = diameter, center = true);
    translate (mPosTL) cylinder(h = height, d = diameter, center = true);
    translate (mPosBR) cylinder(h = height, d = diameter, center = true);
```

```
    translate (mPosBL) cylinder(h = height, d = diameter, center = true);
}

module ssd1306(PCBdim,LCDdim,FLEXdim,PCBwiggle,LCDwiggle) {
    difference() {
        union() {
            //PCB
            translate( [0, 0, PCBdim.z/2] )  color("green") cube(PCBdim +
PCBwiggle, center = true);
            //LCD
            translate( [0, 0, PCBdim.z + LCDdim.z/2] ) color("black")
cube(LCDdim + LCDwiggle, center = true);
            //FLEX
            translate( [0, -LCDdim.y/2 - FLEXdim.y/2, PCBdim.z + FLEXdim.z/2] )
color("brown") cube(FLEXdim, center = true);
        }
        translate( [0, 0, PCBdim.z/2] ) pegs(ssd1306XY, ssd1306off, PCBdim.z +
diffWiggle, 2);
    }
}


module PCB(resize) {
    difference() {
        //ssd1306 PCB
        cube( ssd1306XY + ssd1306PCBZ + resize, center = true );
        //holes only needed for initial tests to see if alligned
        *pegs(ssd1306XY,ssd1306off,ssd1306PCBH+diffWiggle,ssd1306mountD);
    }
}

//ssd1306 mounting harness
module ssd1306Harness(resize) {
    pegD = 1.7 ;
    pegH = 5 ;
    pegZ = [0, 0, pegH] ;
    difference() {
        PCB(resize);
        cube([22,22,diffWiggle] + ssd1306PCBZ, center=true);
        translate([0, 12, 0]) cube([15, 3, diffWiggle] + ssd1306PCBZ,
center=true);
        translate([0, 0, 0]) cube([25, 6, diffWiggle] + ssd1306PCBZ,
center=true);
    }
    //add mounting pegs
    translate( pegZ/2 + ssd1306PCBZ/2 ) pegs(ssd1306XY,ssd1306off,pegH,pegD);
}

module cover() {
    //cover
```

```
    coverThick = .5 ;
    rimH = 1.5 ;
    vieportThick = .5 ;
    rimThick = 1;
    union() {
        translate([0,0,+coverThick/2]) difference() {
            //top cover
            cylinder(h=coverThick, d=plugTopD, center=true);
            //LCD assumed to be dead center
            cube(LCDdimXY + [0, 0, coverThick + diffWiggle], center=true);
            //flex cable
            translate([0, -LCDY/2 - LCDflexH/2 + diffWiggle, 0])
                cube([LCDflexW, LCDflexH + diffWiggle, coverThick + diffWiggle],
center=true);
            //subtract mounting holes
            pegs(ssd1306XY,ssd1306off,coverThick + diffWiggle,ssd1306mountD+.3);
        }
    }
}

module plug() {
    //plug inset
    difference () {
        union() {
            difference() {
                //Plug
                cylinder(h=plugBottomH,d=plugBottomD);
                //Cut the guide left and right
                    cutOffTR=[(plugBottomD/2)-plugSideCutH,plugSideCutW/2,0];
                    cutOffTL=[-
((plugBottomD/2)+plugSideCutH)+plugSideCutH,plugSideCutW/2,0];
                    cutOffBR=[(plugBottomD/2)-plugSideCutH,-(plugSideCutW/2)-
cutCube,0];
                    cutOffBL=[-((plugBottomD/2)+plugSideCutH)+plugSideCutH,-
plugSideCutW/2-cutCube,0];
                    cutCube=[plugSideCutH,cutCube,plugBottomH+diffWiggle];
                translate(cutOffTR+zdiff)cube(cutCube);
                translate(cutOffTL+zdiff)cube(cutCube);
                translate(cutOffBR+zdiff)cube(cutCube);
                translate(cutOffBL+zdiff)cube(cutCube);
            }
            // add a top rim
            translate([0,0,plugTopOff]) cylinder(h=plugTopH, d=plugTopD);
        }
        //cube cutout for inner volume
        translate([0, 0, height/2 + cubeFloor]) cube([cubeXY, cubeXY, height],
center = true);
        //punch holes for cabling where 220v power pins should be
        translate([pinR,0,0])translate(zdiff)
cylinder(h=cubeFloor+diffWiggle,d=6);
```

```
        translate([-pinR,0,0])translate(zdiff)
cylinder(h=cubeFloor+diffWiggle,d=6);
        //make room for the PCB
        translate([0,0,plugTopOff+1]) PCB([1, 1, 0]);
        translate([0,0,plugTopOff+2]) PCB([1, 1, 0]);
    }
    //add in the harness
    translate([0,0,plugTopOff-.7]) ssd1306Harness([-1, -1, 0]);
}
//
// OUTPUT
//

//plug
    plug();
//SSD1306 LCD
    translate([0,0,25]) ssd1306 ( ssd1306PCBdim, LCDdim, FLEXdim, [0, 0, 0], [0,
0, 0] );
//top cover
    translate([0,0,33]) cover();
```

schuko - 3D Object

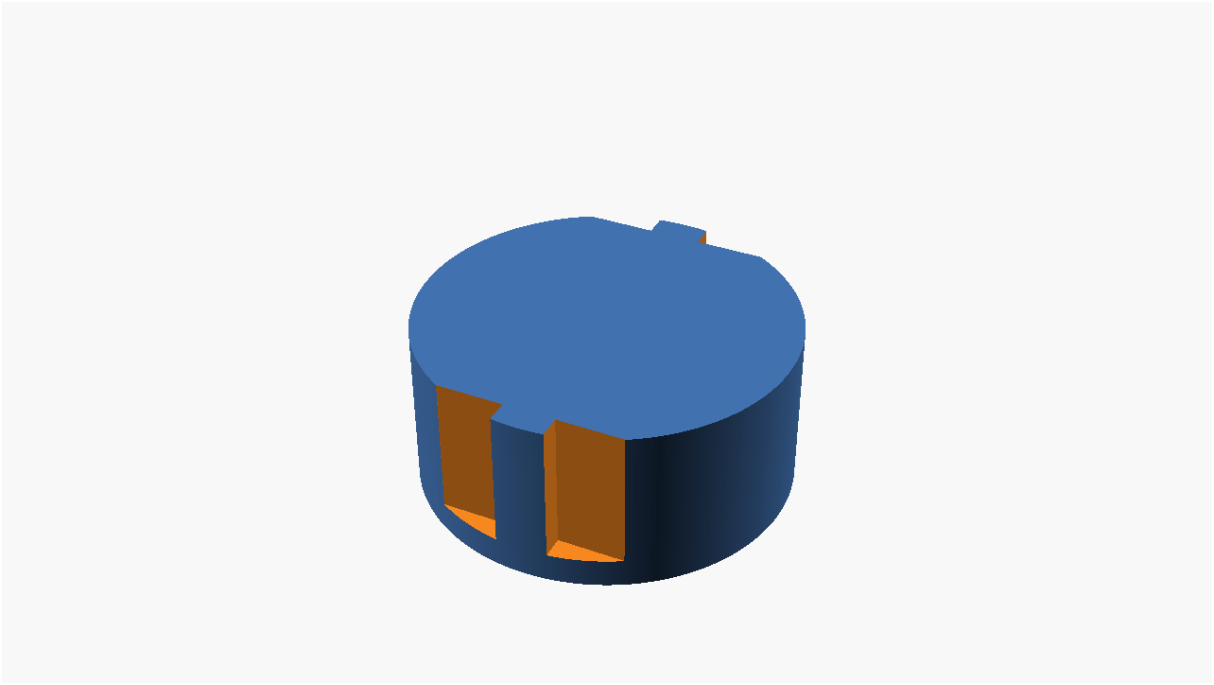

Figure 66. image

Listing 66. Openscad source

```
/*
 Parametric Schuko CEE 7/3 socket

 Copyright 2017 Anders Hammarquist <iko@iko.pp.se>
```

```
 Licensed under Creative Commons - Attribution - Share Alike

 Made using a negative "profile punch" that can be extracted
 and used to "punch" a schuko socket into any sufficiently large solid.
*/

// Diameter of cover
coverdiameter = 50; // [50:100]

// Thickness of cover
coverthickness = 4.8; // [2:0.2:15]

// Center screw offset (extreme values disables screw hole)
screwoffset = 0; // [-11:0.5:11]
// This is the socket punch. Includes cut-out for
// earthing contacts and holes for pins and center screw.
// Maximum screw offset from center is 10mm (use a larger
// value to remove the hole for the screw).
module schuko(screwoffset=0, screwdia=3.5, screwhead=6.5, screwsink=3)
{
    module earthing()
    {
        intersection() {
            union() {
                translate([-22,-2,3])
                    cube([6,4,20]);
                translate([-19,-2,17.5])
                    rotate([0,-30,0])
                        cube([15, 4, 4]);
            }
            translate([-22,-3,3])
                cube([22,6,20]);
        }
    }

    difference() {
        union() {
            translate([0,0,-1])
                cylinder(r=39/2, $fn=300, h=18.5);

            // Earthing cutouts
            color([1,1,1]) {
                earthing();

                rotate([0,0,180])
                    earthing();

            }

            // Power pins
```

```
            translate([0,10,0])
                cylinder(r=7/2, $fn=300, h=30);
            translate([0,-10,0])
                cylinder(r=7/2, $fn=300, h=30);

            if (abs(screwoffset) <= 10) {
                // Center screw
                translate([screwoffset,0,0])
                    cylinder(r=screwdia/2, $fn=300, h=30);
                translate([screwoffset,0,0])
                    cylinder(r=screwhead/2, $fn=300, h=17.5+screwsink);
            }
        }

        // Side key profile
        translate([5.4/2,16.9,3])
            cube([7,3,20]);
        translate([-5.4/2-7,16.9,3])
            cube([7,3,20]);
        translate([5.4/2,-20.4,3])
            cube([7,3.5,20]);
        translate([-5.4/2-7,-20.4,3])
            cube([7,3.5,20]);
    }
}
difference () {

difference () {
    cylinder(r=39/2, $fn=300, h=17.5);
translate ([-27.3/2,-27.8/2,0]) cube([27.3,27.8,10]);
rotate([0,0,0]){
    difference(){
        union() {
            translate([0,0,0])
                cylinder(r=44/2, $fn=300, h=21.5);
            // Lip
            rotate_extrude($fn=100) {
                polygon(points=[[0,0], [coverdiameter/2,0],
                [coverdiameter/2+0.2*coverthickness,coverthickness],
                [0,coverthickness]]);
            }

            // Pin guard: 9.5 x 28.5 x 3mm (rounded ends)
            translate([-4.75,-14.25,21.5])
                cube([9.5, 28.5, 3]);

            // center screw standoff: 6 x 2.5 (above pin guard) x 2 - 3
            // ( 8mm inside, 14 - 12.2 mm outside)
            translate([-7.25, -3, 21.5])
                cube([2.5, 6, 5.5]);
```

```
            translate([4.75, -3, 21.5])
                cube([2.5, 6, 5.5]);

        }
        schuko(screwoffset=screwoffset);
    }
}
}

}
```

shutterholders - Project

shutterholder - 3D Object

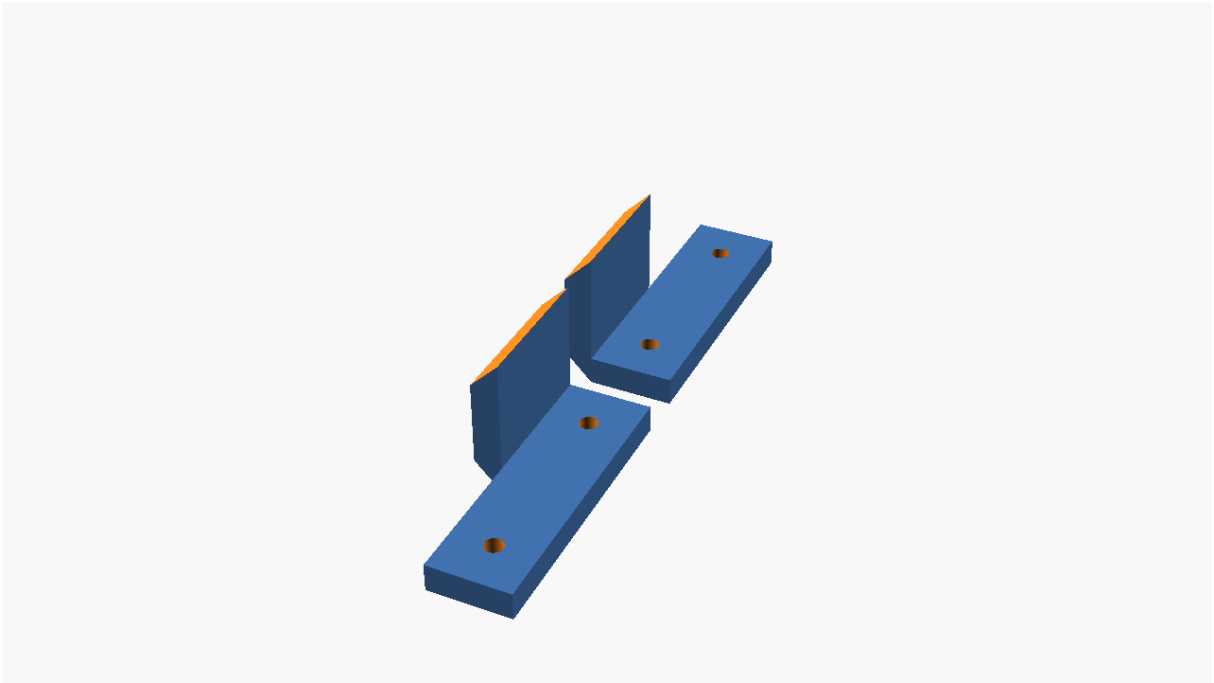

Figure 67. image

Listing 67. Openscad source

```
$fn= 368;
basedepth = 30;
basewidth = 9;
baseheight = 3;

holderheight = 15;
holderwidth = 3;
holderdepth = 15;

module holderleft() {
```

```
difference() {

cube([basewidth, basedepth, baseheight]);
translate([4.5, 25, -0.05]) cylinder(3.1, 1);
translate([4.5, 5, -0.05]) cylinder(3.1, 1);

    }

    difference() {

    translate([-3, 15, 0])
    cube([holderwidth, holderdepth, holderheight]);
    translate([-3, 14.95, 12]) rotate([0, 315, 0]) cube([5, 15.1, 3]);
    translate([0, 14.95, 0]) rotate([0, 225, 0]) cube([5, 15.1, 3]);

        }

}

module holderright() {

difference() {

cube([basewidth, basedepth, baseheight]);
translate([4.5, 25, -0.05]) cylinder(3.1, 1);
translate([4.5, 5, -0.05]) cylinder(3.1, 1);

    }

    difference() {

    translate([-3, 0, 0])
    cube([holderwidth, holderdepth, holderheight]);
    translate([-3, -0.05, 12]) rotate([0, 315, 0]) cube([5, 15.1, 3]);
    translate([0, -0.05, 0]) rotate([0, 225, 0]) cube([5, 15.1, 3]);

        }

}

holderleft();
translate([0, 35, 0]) holderright();
```

solar - Project

balcony - 3D Object

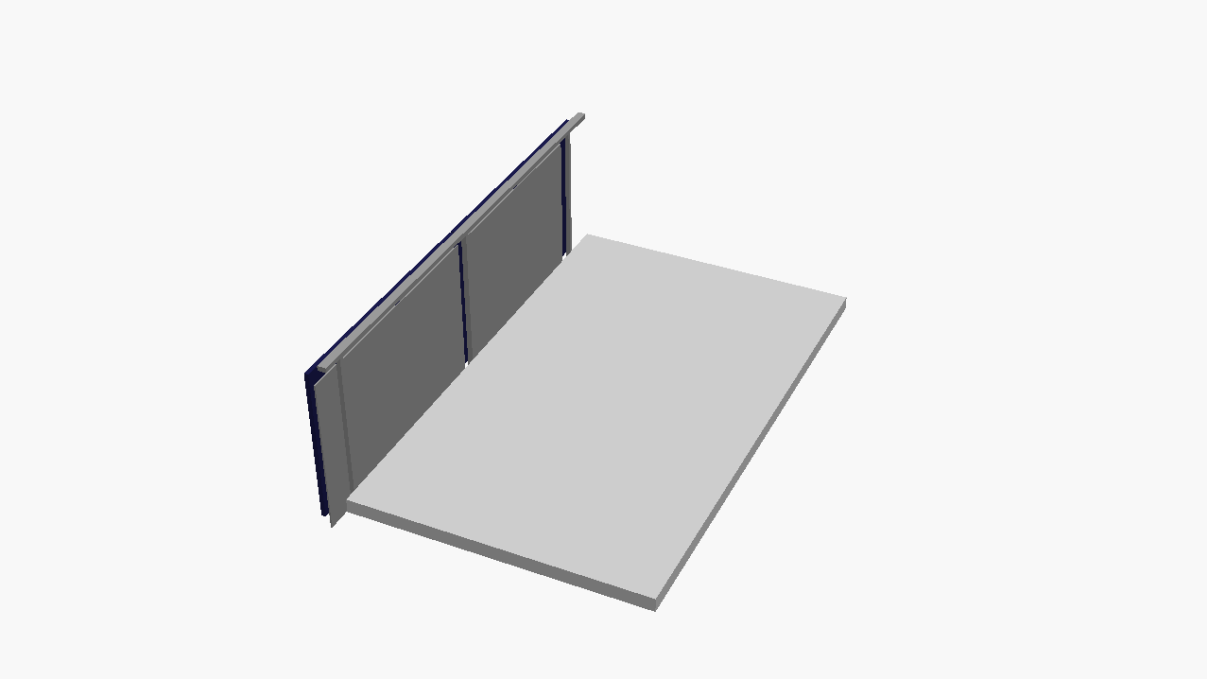

Figure 68. image

Listing 68. Openscad source

```
//handrail
handRailH=35;
handRailD=60;
handRailL=3.52*10*100;
//Post(s)
postX=30;
postY=30;
postH=1.17*10*100;
postOff=handRailD-postY;
post1Pos=15*10;
post2Pos=1.695*10*100;
post3Pos=3.235*10*100;
//Balcony
balconyD=2.08*10*100;
balconyW=3.45*10*100;
railOff=[-100,0,95*10];
blech1W=1.73*10*100;
blech1H=1.10*10*100;
blech1OffX=-2.5*10;//inside post1pos
blech1OffY=-8.5*10;//inside RailH
blech2W=1.48*10*100;
blech2H=1.10*10*100;
blech2OffX=-2.5*10;//inside post1pos
blech2OffY=blech1OffY;
//solar panel
panelW=1.755*10*100;//m
panelH=1.10*10*100;//m
```

```
panelD=30;//mm

module HandRail() {
    color([.6,.6,.6])
        translate (railOff)
            cube([handRailD,handRailL,handRailH]);
}

module HandRail() {
    color([.6,.6,.6])
        translate (railOff)
            cube([handRailD,handRailL,handRailH]);
}
module Post(pos) {
    color([.6,.6,.6])
        translate (railOff)
            translate([postOff,pos,-postH])
                cube([postX,postY,postH]);
}
module Blech(pos,w,h) {
    color([.6,.6,.6])
        translate (railOff)
            translate([0,pos,0])
                translate ([-10,-w,-h + blech1OffY])
                    cube([10,w,h]);
}
module Panel(x,y,z) {
    color([.1,.1,.3])
        translate([x,y,z])
            translate([0,0,0])
                cube([panelD,panelW,panelH]);
}
//balcony
color([.8,.8,.8]) translate ([0,0,-100]) cube([balconyD,balconyW,100]);

HandRail();
Post(post1Pos);
Post(post2Pos);
Post(post3Pos);
Blech(post2Pos-postX-blech1OffX,blech1W,blech1H);
Blech(post3Pos-postX-blech2OffX,blech2W,blech2H);

Panel(-200,0,-200);
Panel(-200,1.75*1000+20,-200);
```

smallpv - 3D Object

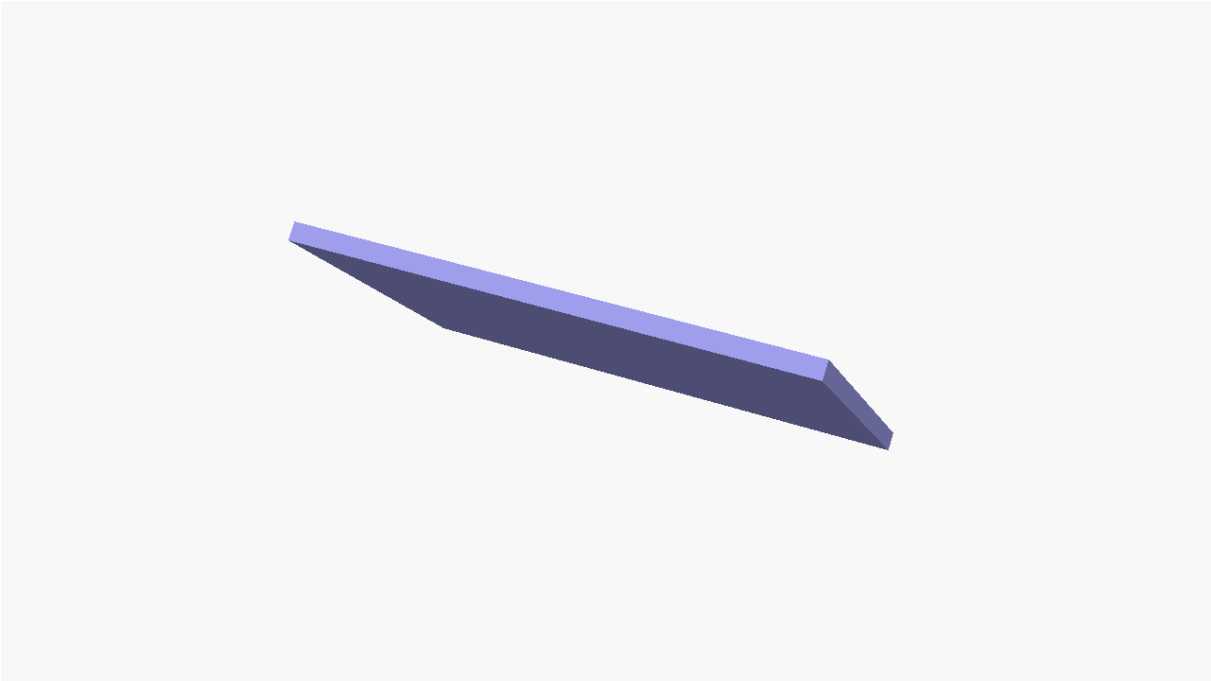

Figure 69. image

Listing 69. Openscad source

```
module pvsmall() {
    color([.6,.6,.9])
        cube([700,25,500]);
}
rotate([45,0,0])pvsmall();
```

spool-holder - Project

spool - 3D Object

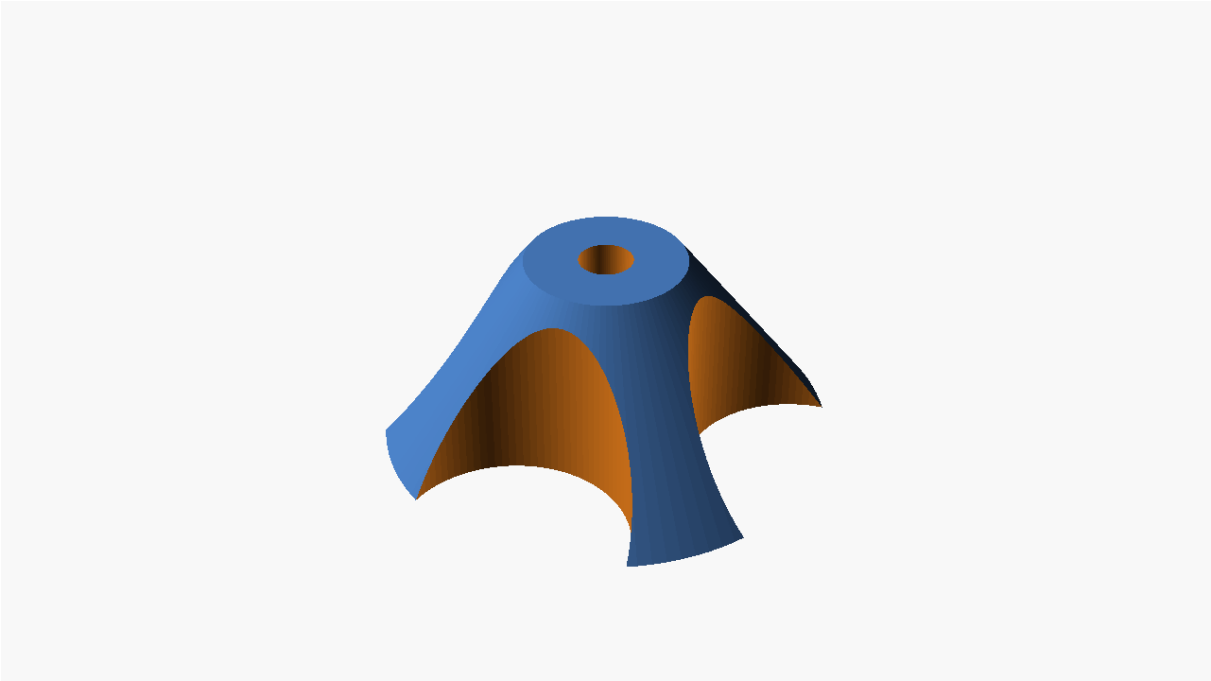

Figure 70. image

Listing 70. Openscad source

```
coneH=30;     //height of the cone
coneDin=25;   //smallest diameter of the cone
coneDout=70;  //widest diameter of cone
axleD=8;      //axle diameter of the axle for the 608 bearing - we;ll add for
printer tolerance
$fn=100;      //make things round
bearingH=7;   //608 skateboard bearing height
bearingD=22;  //608 skateboard bearing diameter we'll add amillimeter or two
later to account for the fitting ring
fittingD=bearingD+7;  //outer diameter of the fitting ring for the bearing
nubAngle=360/8;  //the fitting nubs for the bearing at x degree rotation
printerRadTol=.2; //add this value to the radius
nubRad=.5;    //the nub radius for the bearing fitting ring

module cone(height,inD,outD) {
    cylinder(h=coneH , r2=(inD/2) , r1=(outD/2) );
    }

module axle(height,diameter,tol) {
    translate([0,0,-.1]) cylinder(h=height,r=(diameter/2)+tol); //axle
}
module bearing(height,diameter,tol) {
    translate([0,0,-.1]) cylinder(h=height+.1,r=(diameter/2)+tol); //bearing
}

//subtract for quicker print
module removeCyls(bearingD,coneDout,coneH){
    translate([-((bearingD/2)+(coneDout/4)+4),0,-.1])
```

```
cylinder(h=coneH,r=coneDout/4);
    translate([+((bearingD/2)+(coneDout/4)+4),0,-.1])
cylinder(h=coneH,r=coneDout/4);
    translate([0,+((bearingD/2)+(coneDout/4)+4),-.1])
cylinder(h=coneH,r=coneDout/4);
    translate([0,-((bearingD/2)+(coneDout/4)+4),-.1])
cylinder(h=coneH,r=coneDout/4);
    }

module ring(inRad,outRad,height,tol) {
  difference(){
      cylinder(h=height,r=outRad+tol);
      translate([0,0,-.1]) cylinder(h=height+.2,r=inRad+tol);
    }
}

module fittingNubsCircle(nubRad,height,inRad,angle,tol) {
    rad=inRad+nubRad+tol;
    for (pos=[0:angle:360]) {
        *echo(pos);
        rotate ([0,0,pos]) translate([rad,0,0]) cylinder(h=height,r=nubRad);
    }
}
//
difference(){
    union(){
        difference(){
            cone(coneH,coneDin,coneDout);
            translate([0,0,-.1]) bearing(bearingH+.1,fittingD,printerRadTol);
            translate([0,0,-.1]) axle(coneH+.5,axleD,printerRadTol);
        }//
        ring( (bearingD/2)+nubRad, (fittingD/2) , bearingH , printerRadTol );
        fittingNubsCircle( nubRad , bearingH , bearingD/2 , nubAngle ,
printerRadTol );
    }//
    removeCyls(bearingD,coneDout,coneH);
}
```

sword - Project

blade - 3D Object

Figure 71. image

Listing 71. Openscad source

```
// Generated by inkscape 0.0 + inkscape-paths2openscad 0.27
// Sun Oct 29 23:55:14 2023 from "halloween.tiff.svg"

// Module names are of the form poly_<inkscape-path-id>().  As a result,
// you can associate a polygon in this OpenSCAD program with the corresponding
// SVG element in the Inkscape document by looking for the XML element with
// the attribute id="inkscape-path-id".

// fudge value is used to ensure that subtracted solids are a tad taller
// in the z dimension than the polygon being subtracted from.  This helps
// keep the resulting .stl file manifold.
fudge = 0.1;
user_unit_scale_x = 1.0;
user_unit_scale_y = 1.0;
custom_scale_x = 1;
custom_scale_y = 1;
zsize = 5.00;
line_fn = 4;
min_line_width = 1.0;
line_width_scale = 1.0;
function min_line_mm(w) = max(min_line_width, w * line_width_scale) * 1;


path7_0_center = [0.000000,0.000000];
path7_0_points = [[-6.211141,56.491387],[-11.773241,54.678287],[-
16.131421,53.140387],[-17.333261,49.249287],[-18.299491,45.710305],[-
19.090436,41.619609],[-20.747921,28.048917],[-21.487982,19.183951],[-
21.300618,17.106103],[-20.740921,15.239217],[-19.967180,11.613437],[-
```

```
20.066761,5.835937],[-20.158250,-0.312855],[-19.853401,-6.034583],[-19.575395,-
10.170069],[-19.822201,-12.128343],[-20.087007,-12.859070],[-20.000895,-
13.729269],[-19.642141,-14.457659],[-19.089021,-14.762963],[-18.709377,-
15.028199],[-18.551471,-15.665893],[-18.711741,-16.303595],[-19.097071,-
16.568833],[-19.390442,-16.920491],[-19.329611,-17.765973],[-18.951794,-
18.514595],[-18.483031,-18.633383],[-18.104140,-18.897215],[-17.942381,-
19.994553],[-16.743468,-26.595121],[-14.007776,-37.906939],[-10.993804,-
49.021308],[-8.960051,-55.029526],[-7.253416,-55.835790],[-3.894304,-
56.397304],[0.134142,-56.610912],[3.848779,-56.373461],[12.138859,-
55.090539],[18.608549,-53.569603],[20.475935,-52.443671],[21.040405,-
51.630469],[21.384400,-50.510610],[21.487982,-46.931430],[20.940719,-
40.867143],[20.252429,-31.190993],[19.962419,-20.481553],[19.567900,-
6.316918],[17.860199,9.917267],[15.895121,23.253799],[14.752035,27.853816],[13.0
86309,33.092607],[11.144568,39.376384],[10.157389,43.495277],[9.526349,46.149821
],[8.440839,48.912897],[6.695168,51.486579],[3.937319,53.258157],[0.024599,55.46
1627],[-1.120688,56.026963],[-2.820621,56.433081],[-4.656934,56.610912],[-
6.211361,56.491387],[-6.211141,56.491387]];
module poly_path7(h, w, s, res=line_fn)
{
  scale([custom_scale_x, -custom_scale_y, 1]) union()
  {
    translate (path7_0_center) linear_extrude(height=h, convexity=10,
scale=0.01*s)
      translate (-path7_0_center) polygon(path7_0_points);
  }
}

module halloween_final(h)
{
  difference()
  {
    union()
    {
      translate ([0,0,0]) poly_path7(h, min_line_mm(0.601957), 100.0);
    }
    union()
    {
    }
  }
}

halloween_final(zsize);
```

crossguard - 3D Object

This is for a plastic play "Katana" sword to make it more like a "One Piece" "Gryphon" sword.
The tiff files are the original sword cross guard from both sides so as to extrude the mounting holes for the handle and the sword.
The idea is to go from scanner to tiff.

Then from tiff to SVG.
Then from SVG to STL.
Then in openscad to subtract te shapes from the crossguard and to send it off to Cura and then to octoprint.

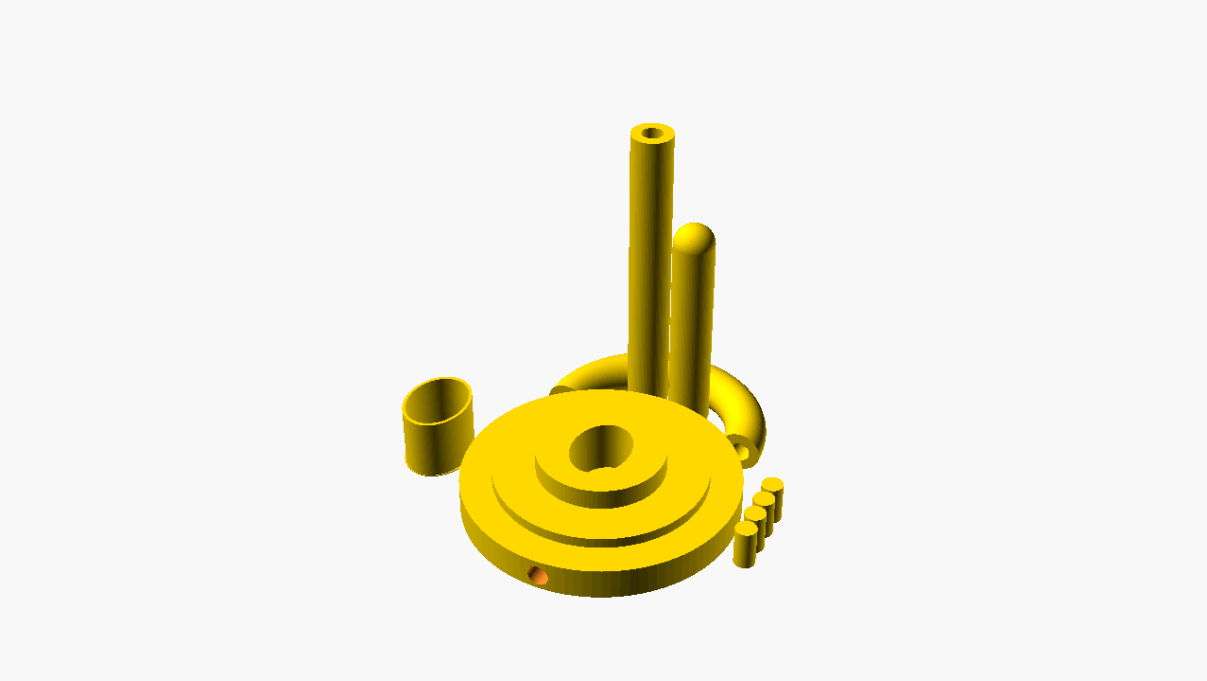

Figure 72. image

Listing 72. Openscad source

```
$fn=100;
crossGuardD=130;
crossGuardH=15;
crossGuardMidD=100;
crossGuardMidH=20;
crossGuardInnerD=60;
crossGuardInnerH=30;
eps = 0.01;
handleR = 10;
handleTR = 45;
handleL = 250;

module crossGuard() {
    cylinder(h=crossGuardH,d=crossGuardD);
    cylinder(h=crossGuardMidH,d=crossGuardMidD);
    cylinder(h=crossGuardInnerH,d=crossGuardInnerD);
}

//
// HILT
//
fudge = 0.1;
```

```
custom_scale_x = 0.2645833279742765;
custom_scale_y = 0.2645833279742765;
line_fn = 4;
function min_line_mm(w) = max(1.0, w) * 0.264583;
hilt_points = [[4.712305,111.150545],[-1.094011,110.499592],[-
2.972932,110.135122],[-3.462188,109.608767],[-3.773607,108.774219],[-
4.436979,107.552054],[-5.432227,107.011144],[-9.721137,106.591800],[-
10.735377,106.203812],[-11.314670,105.621713],[-11.834643,105.005774],[-
11.984051,105.527257],[-12.119685,105.883297],[-12.540277,105.918767],[-
14.389428,104.994043],[-17.553047,103.514177],[-19.092556,102.880464],[-
21.534606,101.319091],[-24.151081,99.339710],[-26.213867,97.451972],[-
33.501068,91.312376],[-37.392389,88.156629],[-39.010903,86.486779],[-
41.321477,83.011799],[-45.215736,77.789987],[-48.481218,72.604179],[-
50.974430,67.705653],[-52.551875,63.345689],[-56.678330,51.259617],[-
59.337010,43.394768],[-60.735652,38.462584],[-61.650982,34.907832],[-
62.305677,29.646387],[-62.760126,25.882525],[-63.514729,23.223299],[-
64.337219,21.072645],[-64.954891,18.531501],[-65.818815,14.645864],[-
66.319039,7.866276],[-66.537764,-5.264370],[-66.447944,-19.090090],[-66.022534,-
27.954901],[-64.949247,-40.097012],[-64.576605,-43.563098],[-63.777363,-
47.924908],[-61.337496,-57.784557],[-58.506478,-66.573669],[-57.218296,-
69.597308],[-56.161139,-71.189954],[-55.394537,-72.009377],[-54.917991,-
73.092386],[-54.557512,-77.066229],[-54.447780,-79.616616],[-54.130244,-
81.294058],[-53.449893,-82.554996],[-52.251719,-83.855870],[-50.317564,-
85.526970],[-48.786417,-86.499607],[-48.069093,-86.977131],[-47.610319,-
87.764921],[-47.463608,-90.285070],[-45.208186,-93.007362],[-39.659449,-
98.460952],[-34.263785,-103.293508],[-29.781396,-106.835337],[-26.310565,-
109.015727],[-23.949574,-109.763965],[-20.750313,-110.582106],[-18.395319,-
111.244120],[-15.748063,-111.591923],[-13.496602,-111.578542],[-12.328996,-
111.157007],[-11.471814,-110.670208],[-9.997877,-110.449507],[-7.811355,-
110.105197],[-5.240945,-109.321219],[-1.137638,-108.242977],[-0.350486,-
108.090624],[-0.226663,-107.697860],[0.081186,-107.047867],[1.233025,-
106.325002],[2.616337,-105.824776],[3.618607,-105.842699],[4.284269,-
105.783643],[5.074276,-105.140787],[5.982442,-104.393155],[6.932191,-
104.082193],[7.623874,-103.885627],[7.911564,-103.413031],[10.384891,-
100.265445],[16.331402,-93.637517],[23.178797,-85.744273],[27.818109,-
79.554542],[31.053211,-74.891176],[34.524573,-70.667709],[40.158967,-
64.396349],[41.226734,-62.889282],[41.592088,-61.891968],[42.074945,-
60.702015],[43.007263,-59.398799],[44.483993,-56.886157],[46.275763,-
52.788436],[49.602441,-44.503764],[50.347327,-43.216277],[50.901943,-
42.932913],[51.921409,-41.678411],[53.416649,-38.661761],[54.898401,-
34.983307],[55.877400,-31.743391],[56.416766,-30.111548],[57.023362,-
29.432817],[57.485562,-29.201633],[57.677808,-28.645804],[57.364555,-
28.210189],[56.611407,-28.268022],[55.878931,-28.375238],[55.601784,-
27.856289],[56.377303,-24.331447],[57.238303,-22.352970],[57.692615,-
21.937680],[58.154666,-21.916077],[58.790553,-21.967479],[59.078626,-
21.636472],[58.977659,-21.111651],[58.446422,-20.581610],[58.251394,-
19.738290],[58.750639,-18.229081],[59.534159,-15.386057],[60.073343,-
11.287286],[60.655429,-7.139257],[61.542096,-4.177822],[62.322794,-
2.295269],[62.650064,-
0.741156],[62.906663,0.113753],[63.563431,0.230253],[63.991905,0.313739],[64.254
```

```
280,1.045455],[64.365593,4.953078],[64.382272,8.065028],[64.459069,8.646005],[64
.572444,8.425084],[64.873195,7.449324],[65.294587,6.929440],[65.821792,6.874420]
,[66.439977,7.293248],[66.537764,7.903900],[66.115942,8.715141],[65.726622,9.388
975],[65.902530,9.669241],[65.981683,9.930291],[65.359738,10.557922],[64.473900,
12.816744],[63.765883,16.866181],[63.482095,20.738189],[63.576289,21.992644],[63
.868945,22.464726],[64.221009,24.604724],[64.277064,29.751929],[64.324272,34.821
498],[64.494546,35.764667],[64.760298,35.618793],[65.293780,34.844801],[65.44826
7,36.420426],[65.180519,38.242275],[64.466664,39.500067],[63.605585,41.079630],[
62.983941,43.675054],[62.238836,46.958797],[61.080981,50.193188],[59.015746,55.5
00659],[56.706660,62.429673],[54.809911,68.891064],[53.981690,72.795668],[53.758
845,74.546542],[53.351215,75.469437],[52.521354,77.054138],[51.283366,80.396158]
,[48.953085,85.522824],[45.496989,91.398313],[42.093111,96.299677],[38.945897,10
0.293305],[35.913504,103.476813],[32.854088,105.947815],[29.625807,107.803925],[
26.086817,109.142759],[22.095276,110.061931],[17.509341,110.659056],[8.977984,11
1.591923],[4.712305,111.150545],[4.712305,111.150545]];
module poly_hilt(h, w, s, res=line_fn)
{
  scale([custom_scale_x, -custom_scale_y, 1]) union() {
    linear_extrude(height=h, convexity=10, scale=0.01*s) polygon(hilt_points);
    }
}
module hilt(h) {
  poly_hilt(h, min_line_mm(0.1881051549233745), 100.0);
  }
//
// BLADE
//
blade_points = [[-6.211141,56.491387],[-11.773241,54.678287],[-
16.131421,53.140387],[-17.333261,49.249287],[-18.299491,45.710305],[-
19.090436,41.619609],[-20.747921,28.048917],[-21.487982,19.183951],[-
21.300618,17.106103],[-20.740921,15.239217],[-19.967180,11.613437],[-
20.066761,5.835937],[-20.158250,-0.312855],[-19.853401,-6.034583],[-19.575395,-
10.170069],[-19.822201,-12.128343],[-20.087007,-12.859070],[-20.000895,-
13.729269],[-19.642141,-14.457659],[-19.089021,-14.762963],[-18.709377,-
15.028199],[-18.551471,-15.665893],[-18.711741,-16.303595],[-19.097071,-
16.568833],[-19.390442,-16.920491],[-19.329611,-17.765973],[-18.951794,-
18.514595],[-18.483031,-18.633383],[-18.104140,-18.897215],[-17.942381,-
19.994553],[-16.743468,-26.595121],[-14.007776,-37.906939],[-10.993804,-
49.021308],[-8.960051,-55.029526],[-7.253416,-55.835790],[-3.894304,-
56.397304],[0.134142,-56.610912],[3.848779,-56.373461],[12.138859,-
55.090539],[18.608549,-53.569603],[20.475935,-52.443671],[21.040405,-
51.630469],[21.384400,-50.510610],[21.487982,-46.931430],[20.940719,-
40.867143],[20.252429,-31.190993],[19.962419,-20.481553],[19.567900,-
6.316918],[17.860199,9.917267],[15.895121,23.253799],[14.752035,27.853816],[13.0
86309,33.092607],[11.144568,39.376384],[10.157389,43.495277],[9.526349,46.149821
],[8.440839,48.912897],[6.695168,51.486579],[3.937319,53.258157],[0.024599,55.46
1627],[-1.120688,56.026963],[-2.820621,56.433081],[-4.656934,56.610912],[-
6.211361,56.491387],[-6.211141,56.491387]];
module poly_blade(h, w, s, res=line_fn)
{
```

```
  scale([custom_scale_x, -custom_scale_y, 1]) union() {
    linear_extrude(height=h, convexity=10, scale=0.01*s) polygon(blade_points);
    }
}
module blade(h) {
  poly_blade(h, min_line_mm(0.601957), 100.0);
  }

//handle part
module handleTube(cylH) {
    color("gold") difference(){
        cylinder(h=cylH,r=handleR);
        translate([0,0,-eps]) cylinder(h=10,d=10);
        translate([0,0,cylH-10+eps]) cylinder(h=10,d=10);
    }
}

module hilt2(height,diameter,waist){
    //hilt measurments
    //somewhat rough as hilt is not prefect elipse
    //diameter=36;
    //waist=26;
    //height=10;
    scale([waist/diameter,1]) color("gold") cylinder(h=height,d=diameter);
}

//handle bend
module handleBend(turnR,handleR){
    bound=turnR+handleR+1;
    color("gold") difference() {
        intersection() {

translate([0,0,0])rotate_extrude()translate([turnR,0,0])circle(r=handleR);
            translate([0,0,-handleR]) cube([bound,bound,handleR*2]);
        }
        //two holes for mounting
        translate([-eps,handleTR,0]) rotate([0,90,0]) cylinder(h=10,d=10);
        translate([handleTR,10-eps,0]) rotate([90,0,0]) cylinder(h=10,d=10);
    }
}

difference() {
    //test fitting only
    //cylinder(h=30,d=34);
    //real crossguard - uncomment when the test verison is commented
    color("gold") crossGuard();
    translate([0,0,-0.1]) rotate([0,0,6]) blade(5.11);
    //hiltS=.55; //WRONG
    //scale([hiltS,hiltS,1]) translate([0,0,5-.1]) rotate([0,0,-189]) hilt(30);
    translate([0,0,5]) hilt2(30.01,36,27.5);
```

```
    translate([0,-130/2+20-.1,(crossGuardH-6)/2+2.5]) rotate([90,90,0])
cylinder(h=20.1,d=(10));
}

//approximation of hilt for scale
*color("black")translate([0,0,5]) hilt2(290.01,36,27.5);
//empty hilt shell for testing
translate([-85,0,0]) color("gold") union(){
    difference(){
        scale([1.1,1.1]) hilt2(30.01,36,27.5);
        translate([0,0,-.1]) hilt2(30.2,36,27.5);
    }
    difference(){
        hilt2(.5,40,32);
        translate([0,0,-0.1]) rotate([0,0,6]) blade(5.11);
    }
}


//two bends
translate([0,60,0]) handleBend(handleTR,handleR);
translate ([-2,60,0]) rotate([0,0,90]) handleBend(handleTR,handleR);

//long handle tube
translate([-11,78,0]) handleTube(handleL-(2*handleTR+handleR));

//mounting pegs
pegD=10;
pegH=19;
translate([75,0,0]) color("gold") cylinder(h=pegH,d=pegD);
translate([75,11,0]) color("gold") cylinder(h=pegH,d=pegD);
translate([75,22,0]) color("gold") cylinder(h=pegH,d=pegD);
translate([75,-11,0]) color("gold") cylinder(h=pegH,d=pegD);

//cross handle tube
translate([11,78,0]) color("gold") union() {
    crosshandleH=130*.75;
    handleTube(crosshandleH);
    translate([0,0,crosshandleH]) sphere(r=handleR);
}
```

hilt - 3D Object

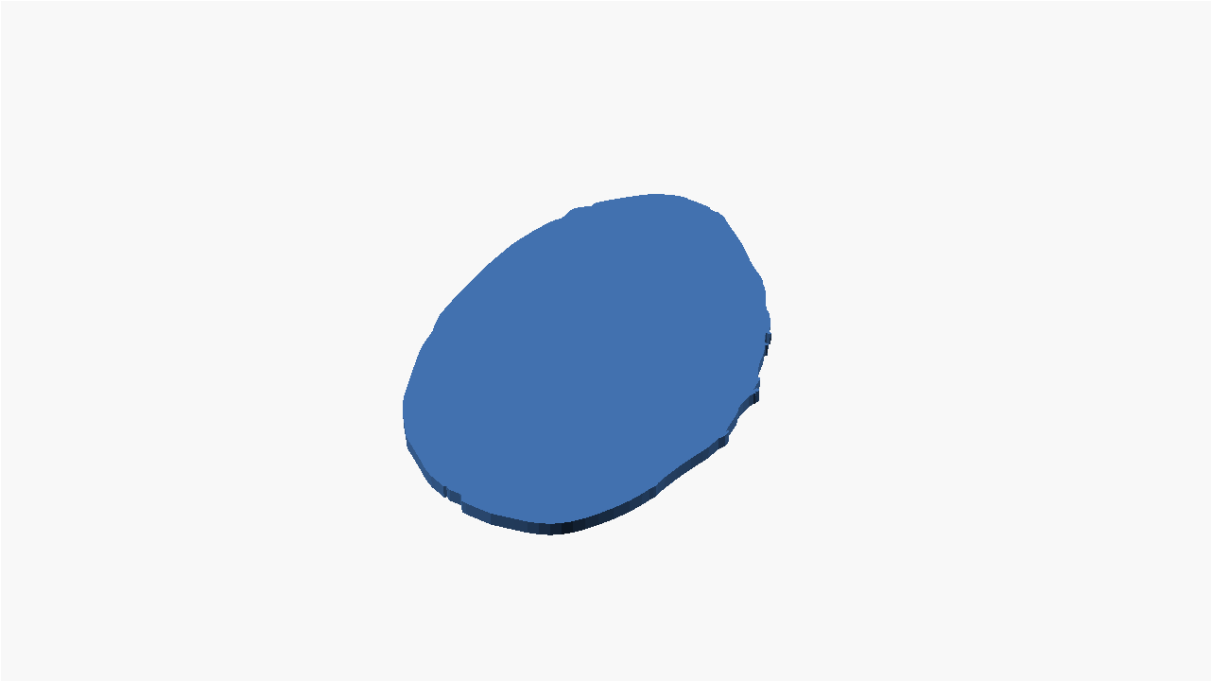

Figure 73. image

Listing 73. Openscad source

```
// Generated by inkscape None + inkscape-paths2openscad 0.27
// Sun Oct 29 23:48:49 2023 from "halloweeneeeee3.tiff.svg"

// Module names are of the form poly_<inkscape-path-id>().  As a result,
// you can associate a polygon in this OpenSCAD program with the corresponding
// SVG element in the Inkscape document by looking for the XML element with
// the attribute id="inkscape-path-id".

// fudge value is used to ensure that subtracted solids are a tad taller
// in the z dimension than the polygon being subtracted from.  This helps
// keep the resulting .stl file manifold.
fudge = 0.1;
user_unit_scale_x = 0.2645833279742765;
user_unit_scale_y = 0.2645833279742765;
custom_scale_x = 1;
custom_scale_y = 1;
zsize = 5.00;
line_fn = 4;
min_line_width = 1.0;
line_width_scale = 1.0;
function min_line_mm(w) = max(min_line_width, w * line_width_scale) * 0.264583;


path1_0_center = [0.000000,0.000000];
path1_0_points = [[4.712305,111.150545],[-1.094011,110.499592],[-
2.972932,110.135122],[-3.462188,109.608767],[-3.773607,108.774219],[-
4.436979,107.552054],[-5.432227,107.011144],[-9.721137,106.591800],[-
10.735377,106.203812],[-11.314670,105.621713],[-11.834643,105.005774],[-
```

```
11.984051,105.527257],[-12.119685,105.883297],[-12.540277,105.918767],[-
14.389428,104.994043],[-17.553047,103.514177],[-19.092556,102.880464],[-
21.534606,101.319091],[-24.151081,99.339710],[-26.213867,97.451972],[-
33.501068,91.312376],[-37.392389,88.156629],[-39.010903,86.486779],[-
41.321477,83.011799],[-45.215736,77.789987],[-48.481218,72.604179],[-
50.974430,67.705653],[-52.551875,63.345689],[-56.678330,51.259617],[-
59.337010,43.394768],[-60.735652,38.462584],[-61.650982,34.907832],[-
62.305677,29.646387],[-62.760126,25.882525],[-63.514729,23.223299],[-
64.337219,21.072645],[-64.954891,18.531501],[-65.818815,14.645864],[-
66.319039,7.866276],[-66.537764,-5.264370],[-66.447944,-19.090090],[-66.022534,-
27.954901],[-64.949247,-40.097012],[-64.576605,-43.563098],[-63.777363,-
47.924908],[-61.337496,-57.784557],[-58.506478,-66.573669],[-57.218296,-
69.597308],[-56.161139,-71.189954],[-55.394537,-72.009377],[-54.917991,-
73.092386],[-54.557512,-77.066229],[-54.447780,-79.616616],[-54.130244,-
81.294058],[-53.449893,-82.554996],[-52.251719,-83.855870],[-50.317564,-
85.526970],[-48.786417,-86.499607],[-48.069093,-86.977131],[-47.610319,-
87.764921],[-47.463608,-90.285070],[-45.208186,-93.007362],[-39.659449,-
98.460952],[-34.263785,-103.293508],[-29.781396,-106.835337],[-26.310565,-
109.015727],[-23.949574,-109.763965],[-20.750313,-110.582106],[-18.395319,-
111.244120],[-15.748063,-111.591923],[-13.496602,-111.578542],[-12.328996,-
111.157007],[-11.471814,-110.670208],[-9.997877,-110.449507],[-7.811355,-
110.105197],[-5.240945,-109.321219],[-1.137638,-108.242977],[-0.350486,-
108.090624],[-0.226663,-107.697860],[0.081186,-107.047867],[1.233025,-
106.325002],[2.616337,-105.824776],[3.618607,-105.842699],[4.284269,-
105.783643],[5.074276,-105.140787],[5.982442,-104.393155],[6.932191,-
104.082193],[7.623874,-103.885627],[7.911564,-103.413031],[10.384891,-
100.265445],[16.331402,-93.637517],[23.178797,-85.744273],[27.818109,-
79.554542],[31.053211,-74.891176],[34.524573,-70.667709],[40.158967,-
64.396349],[41.226734,-62.889282],[41.592088,-61.891968],[42.074945,-
60.702015],[43.007263,-59.398799],[44.483993,-56.886157],[46.275763,-
52.788436],[49.602441,-44.503764],[50.347327,-43.216277],[50.901943,-
42.932913],[51.921409,-41.678411],[53.416649,-38.661761],[54.898401,-
34.983307],[55.877400,-31.743391],[56.416766,-30.111548],[57.023362,-
29.432817],[57.485562,-29.201633],[57.677808,-28.645804],[57.364555,-
28.210189],[56.611407,-28.268022],[55.878931,-28.375238],[55.601784,-
27.856289],[56.377303,-24.331447],[57.238303,-22.352970],[57.692615,-
21.937680],[58.154666,-21.916077],[58.790553,-21.967479],[59.078626,-
21.636472],[58.977659,-21.111651],[58.446422,-20.581610],[58.251394,-
19.738290],[58.750639,-18.229081],[59.534159,-15.386057],[60.073343,-
11.287286],[60.655429,-7.139257],[61.542096,-4.177822],[62.322794,-
2.295269],[62.650064,-
0.741156],[62.906663,0.113753],[63.563431,0.230253],[63.991905,0.313739],[64.254
280,1.045455],[64.365593,4.953078],[64.382272,8.065028],[64.459069,8.646005],[64
.572444,8.425084],[64.873195,7.449324],[65.294587,6.929440],[65.821792,6.874420]
,[66.439977,7.293248],[66.537764,7.903900],[66.115942,8.715141],[65.726622,9.388
975],[65.902530,9.669241],[65.981683,9.930291],[65.359738,10.557922],[64.473900,
12.816744],[63.765883,16.866181],[63.482095,20.738189],[63.576289,21.992644],[63
.868945,22.464726],[64.221009,24.604724],[64.277064,29.751929],[64.324272,34.821
498],[64.494546,35.764667],[64.760298,35.618793],[65.293780,34.844801],[65.44826
7,36.420426],[65.180519,38.242275],[64.466664,39.500067],[63.605585,41.079630],[
```

```
62.983941,43.675054],[62.238836,46.958797],[61.080981,50.193188],[59.015746,55.5
00659],[56.706660,62.429673],[54.809911,68.891064],[53.981690,72.795668],[53.758
845,74.546542],[53.351215,75.469437],[52.521354,77.054138],[51.283366,80.396158]
,[48.953085,85.522824],[45.496989,91.398313],[42.093111,96.299677],[38.945897,10
0.293305],[35.913504,103.476813],[32.854088,105.947815],[29.625807,107.803925],[
26.086817,109.142759],[22.095276,110.061931],[17.509341,110.659056],[8.977984,11
1.591923],[4.712305,111.150545],[4.712305,111.150545]];
module poly_path1(h, w, s, res=line_fn)
{
  scale([custom_scale_x, -custom_scale_y, 1]) union()
  {
    translate (path1_0_center) linear_extrude(height=h, convexity=10,
scale=0.01*s)
      translate (-path1_0_center) polygon(path1_0_points);
  }
}

module halloween2_final(h)
{
  difference()
  {
    union()
    {
      translate ([0,0,0]) poly_path1(h, min_line_mm(0.1881051549233745), 100.0);
    }
    union()
    {
    }
  }
}

halloween2_final(zsize);
```

tesa - Project

tesa - 3D Object

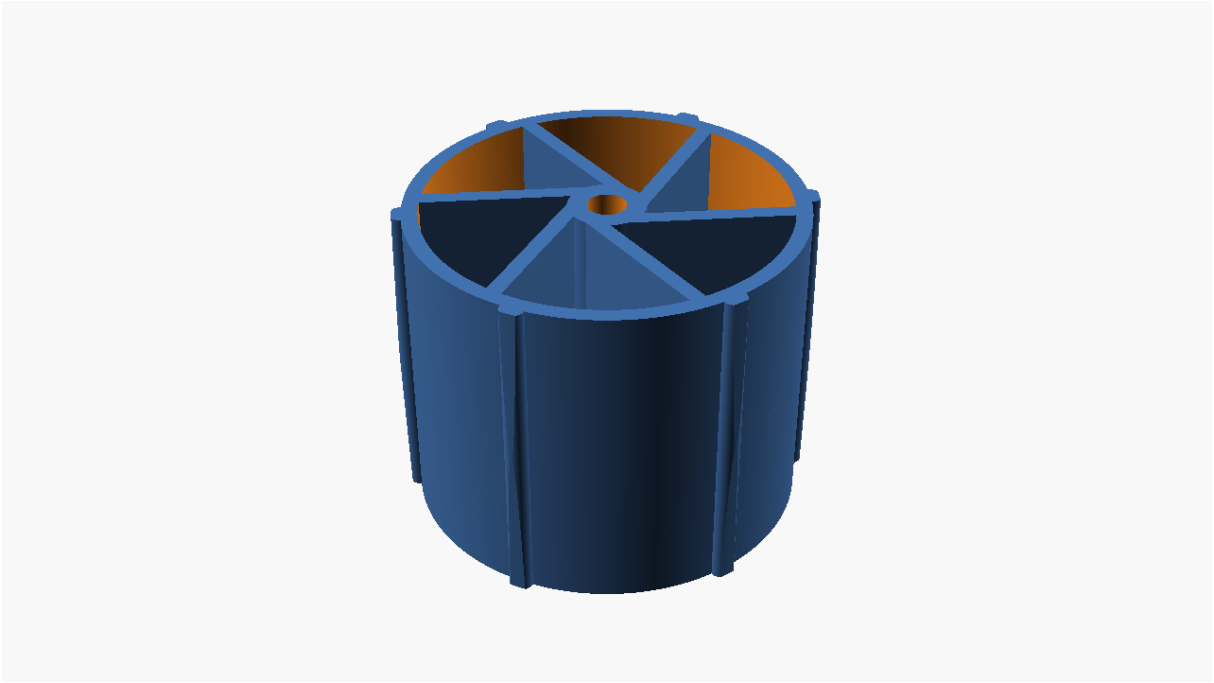

Figure 74. image

Listing 74. Openscad source

```
//Tesa roller ersatzroller
//celotape roller

$fn=360;
height=20;
outsideD=24.5;
outsideDepth=2;
axleD=2.4;
hubD=axleD*2;
nubR=.75;
module taper(){
difference(){
union(){
    translate([0,0,-.1]) cylinder(h=height+.2,d=outsideD+2+nubR);
}
union(){
    translate([0,0,-
.11])cylinder(h=height/2+.22,d1=outsideD+1.5*nubR,d2=outsideD+2*nubR);
    translate([0,0,height/2+.1])
cylinder(h=height/2+.1,d1=outsideD+2*nubR,d2=outsideD+1.5*nubR);
}
}
}
difference(){
union(){
    //outside
    difference(){
```

```
        cylinder(h=height,d=outsideD);
        translate([0,0,-.1])cylinder(h=height+.2,d=outsideD-outsideDepth);
    }
    //HUB
    difference(){
        cylinder(h=height,d=hubD);
        translate([0,0,-.1])cylinder(h=height+.2,d=axleD);
    }
    //nubs
    for (i = [0:5]) {
        translate([sin(360*i/6)*outsideD/2, cos(360*i/6)*outsideD/2, 0 ])
        rotate([0,0,0])cylinder(h = height/2, r=nubR);
    }
    for (i = [0:5]) {
        translate([sin(360*i/6)*outsideD/2, cos(360*i/6)*outsideD/2, height/2 ])
        cylinder(h = height/2, r=nubR);
    }
    //spokes
    for (i = [0:360/6:360]) {
        rotate([0,0,i])translate([1.2,0,0])cube([1,(outsideD/2)-
(axleD/2.4),height]);
    }
}
taper();
}
```

thumb-screw - Project

Knurl - 3D Object

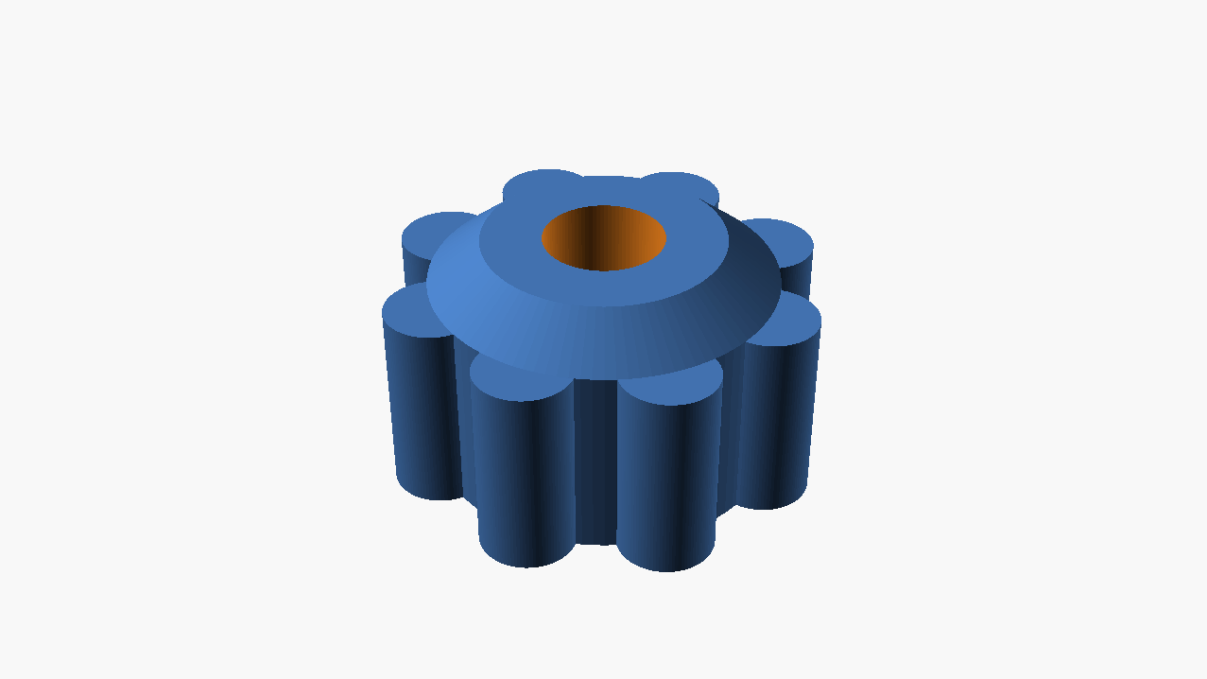

Figure 75. image

Listing 75. Openscad source

```
diam=9;
diamOut=14.4;
holeDiam=5;
height=8.2;
$fn=100;
knurlNum=8;
knurlInc=360/knurlNum;
knurlDiam=4;
insetHeight=5;
insetDiam=5;
totalHeight=10;


module knob() {
difference(){
    union(){
        cylinder(h=height, d=diamOut);
        translate([0,0,height]){
        cylinder(h=totalHeight-height,r1=7.2,r2=5);
        }
    }
    translate([0,0,-.5])
        cylinder(h=totalHeight+1, d=holeDiam);
    translate([0,0,-.5]){
        cylinder(h=totalHeight-insetHeight+.5,d1=holeDiam+5,d2=holeDiam);
    }
    }
```

```
for(i=[0: knurlInc: 360]){
   rotate([0,0,i])
        translate([diamOut/2,0,0])
            cylinder(h=height, d=knurlDiam);
}
    }


knob();
```

Appendix A: To do

Right now the github source is not perfect as the readme does not display the images when viewed in github.

- ☐ Add a readme in the directories or fix asciidoc to deal with asciidoc not showing in github
- ☐ Split the build scruipts to handle scad/plantuml/asciidoc in separate steps depending on changes
  - ☑ The scad build already ony builds scad files that exist and is fed by find
    - ☐ Feed with a list of changed scad files instead of the more simple find
- ☐ Add further process steps for the images like meshlabserver to do furhter processing:
  - Stuff (glass rendering, wireframe, mesh magic, stats, etc.)
- ☑ Need to add subdirectories to pool projects together
  - ☑ build index for subdirs
  - ☑ need to make the scad build fit to the subdir model
  - ☑ add check for empty directory (no scad files)
    - ☑ only build scad files and only run container with scad files (empties ignored)
- ☐ Add further optional scad steps
  - ☐ Need to add view parameters as options
  - ☐ Need to add animation options
- ☑ Need to add text display option for each item
- ☐ Allow adding photos to the objects or projects to show makes
- ☐ investigate including a js or similar stl viewer for html