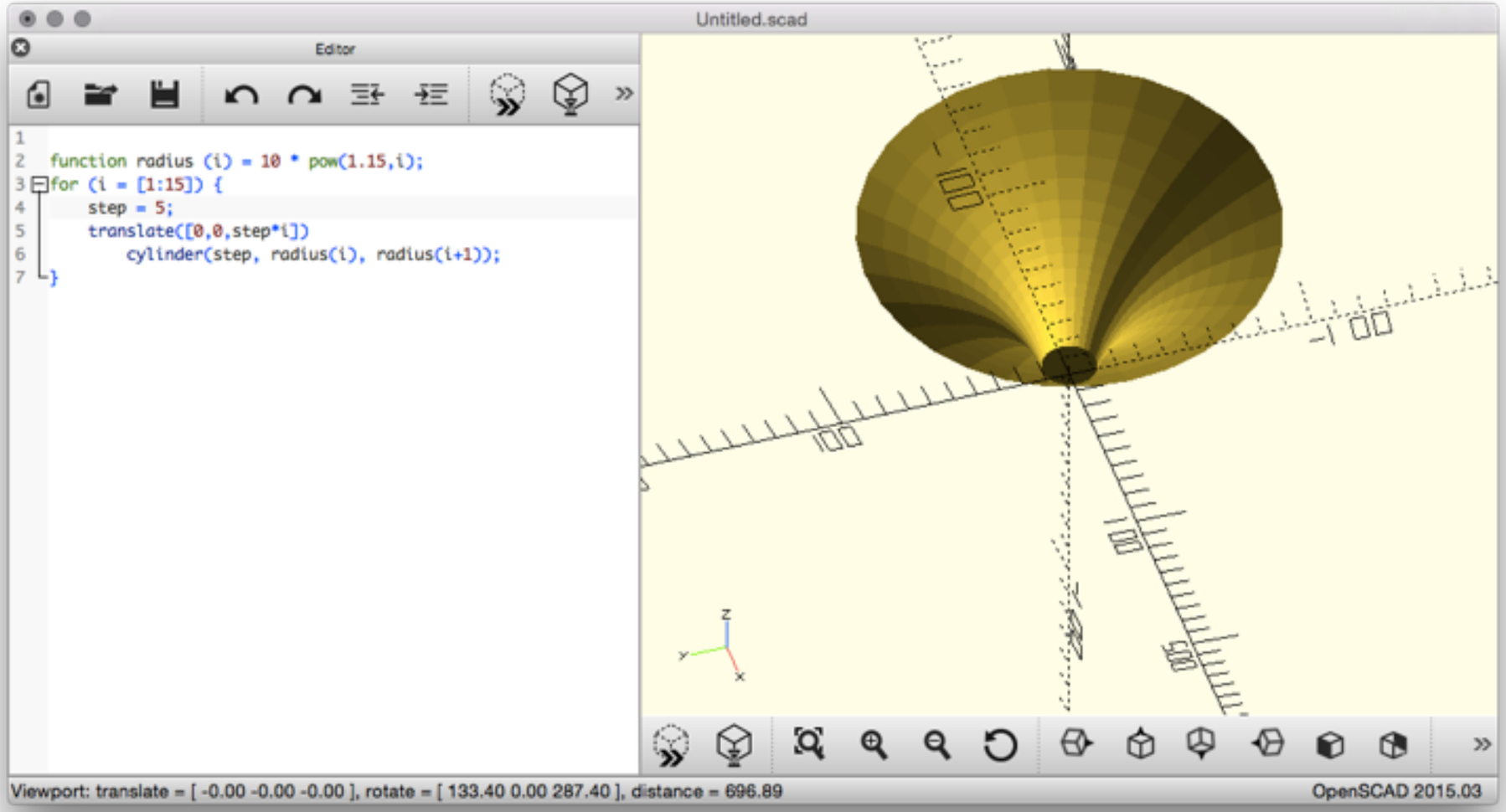
I am currently installing some new wheel and gear leg fairings on the plane which involves fabrication of new intersection fairings to cover the leg-to-wheel and leg-to-fuselage transition areas.

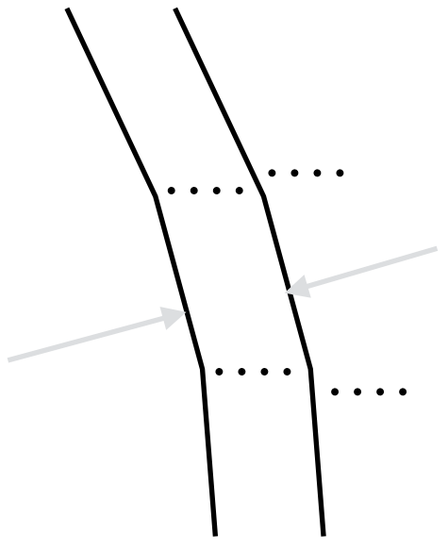
I was planning to make the intersection fairings out of formed sheet metal but it might be possible to 3-D print them if you are up for a challenge. They are commonly made in fiberglass because of the complexity of the shape. They would be the right size for your printer and have thin cross sections so not much material would be involved.



I don’t know how to model or to print this shape. That makes it interesting.

I’ve been using OpenSCAD which assembles models with unions and intersections of cubes and cylinders (or truncated cones). It has some algorithmic capability but mostly for step and repeat. It can also form a convex hull but your shape is concave. Small experiment attached.

The printer expects inputs as STL files, whatever those are. The printer will design removable scaffolding to support overhangs greater than 45 degrees.



What negative adjustment would I make to the top and bottom radius of a second cylinder to be removed from each step so that the volume thickness would be small and uniform along a surface normal?

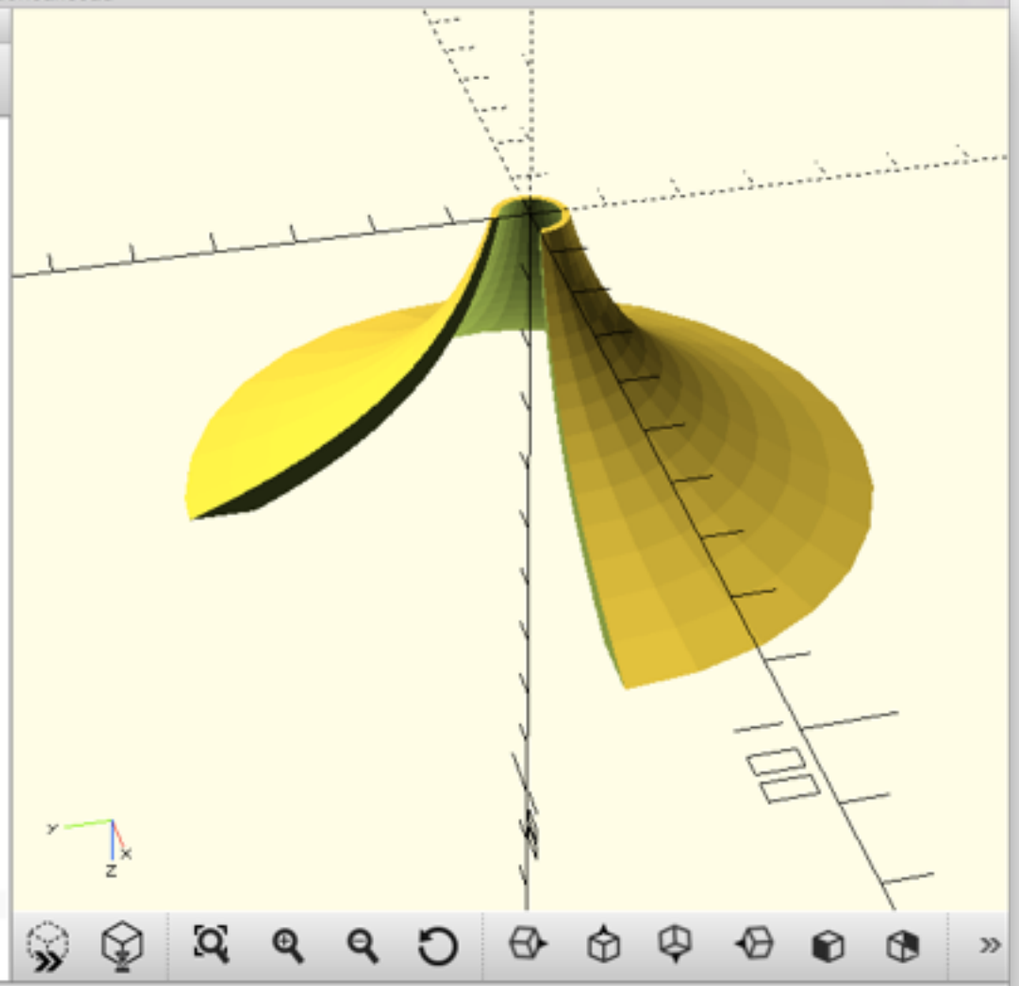
This calculation is complicated by the vertical expansion required of the negative cylinder so as to not create ambiguous planes that might or might not be removed.

cylinderical.scad

Editor

STL

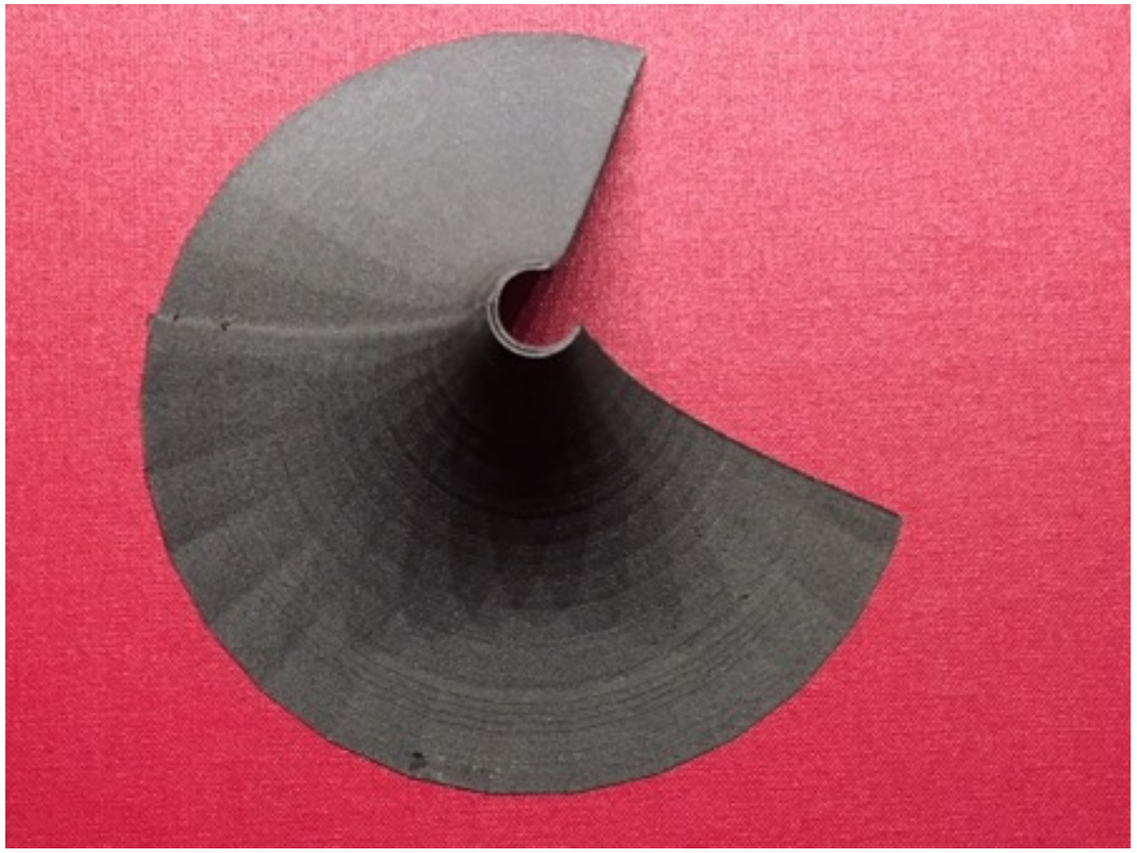
```
1
2  function radius (i) = 10 * pow(1.15,i);
3
4  module outside (h, i) {
5      cylinder(h, radius(i), radius(i+1));
6  }
7
8  module inside (h, i) {
9      extension = .01;
10     thickness = 1.5;
11     translate([0,0,-extension/2])
12         cylinder(h+extension, radius(i-thickness), radius(i+1-thickness));
13 }
14
15 module fairing () {
16     for (i = [0:15]) {
17         step = 5;
18         translate([0,0,step*i])
19             difference() {
20                 outside(step, i);
21                 inside(step, i);
22             }
23     }
24 }
25
26 difference () {
27     fairing ();
28     translate ([0,0,-1]) cube(100,100,100);
29 }
```

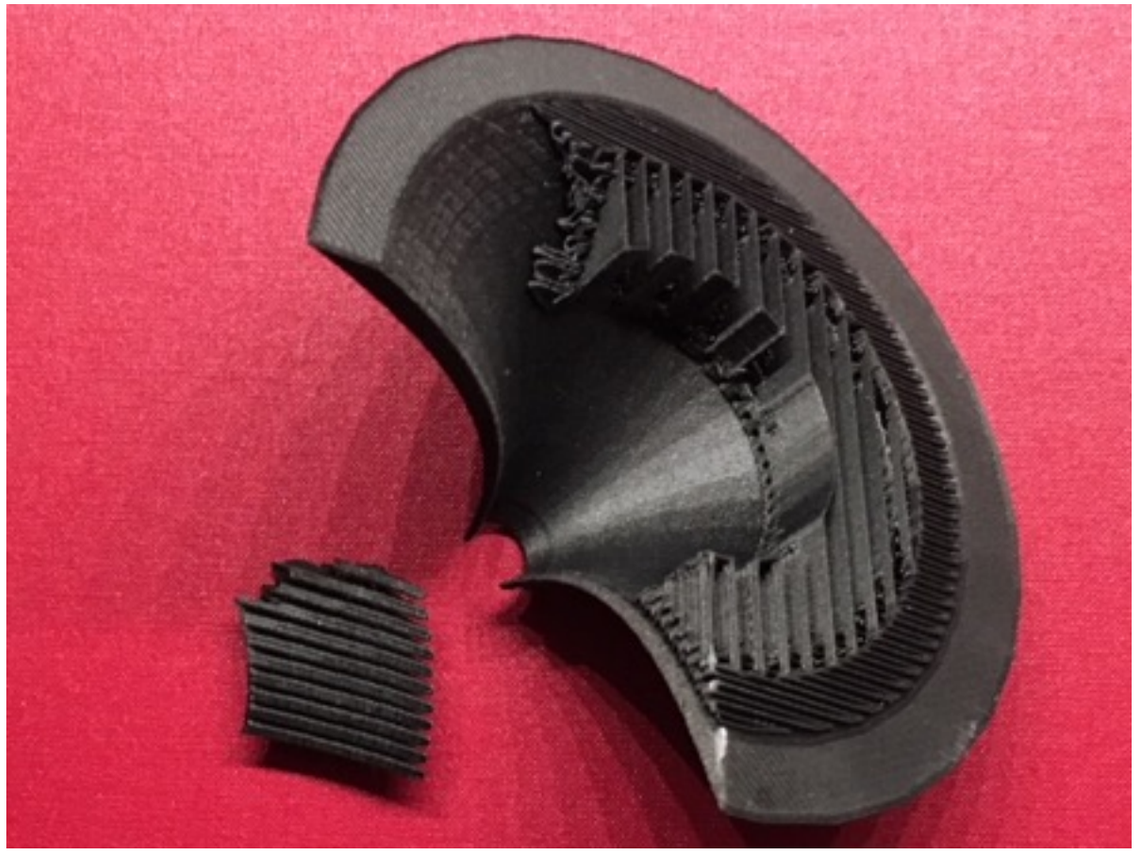


Viewport: translate = [17.27 6.90 25.06], rotate = [234.90 0.00 283.90], distance = 286.60

OpenSCAD 2015.03

I neglect thickness uniformity and try a print.





Some support material removed.

Creating these objects seem like a job for a 3D scanner, another tool that I know little about. Perusing the numerous pages on the topic it looks like one could be constructed using parts produced on your printer but that is a project in itself. Also we would need parts to scan.

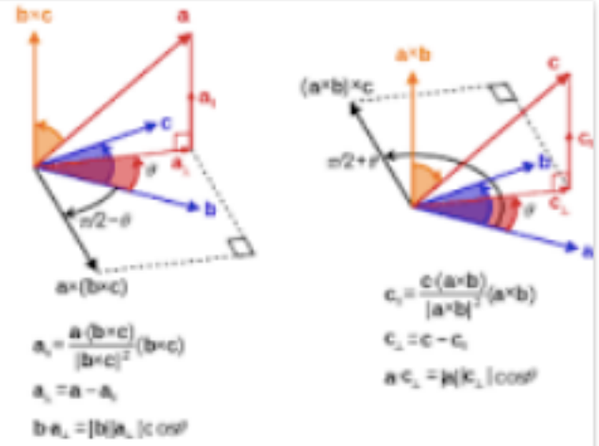
Regarding modeling, I was wondering if there would be a solution based on aerodynamic first principles.

The problem with intersection fairings is that they exist to cover up the intersections of two completely different shapes, so even if those shapes were perfectly formed airfoils fitting some mathematical description, the blend region between them would still need to be constructed using some kind of interpolation rules. Most people just lay on some fiberglass until it looks about right and then add resin.

Interpolation sounds like a nice approach. Maybe trace the perimeter top and bottom, connect corresponding points to make a volume, then suck air out using some formula for elasticity.

Strategy: Assemble polyhedron slabs based on top and bottom perimeters. Use cross product of slab edges to find additional points to give the slab thickness.

The **cross product** a × b is defined as a **vector** c that is **perpendicular (orthogonal) to both** a and b, with a direction given by the right-hand rule and a magnitude equal to the area of the parallelogram that the **vectors** span.

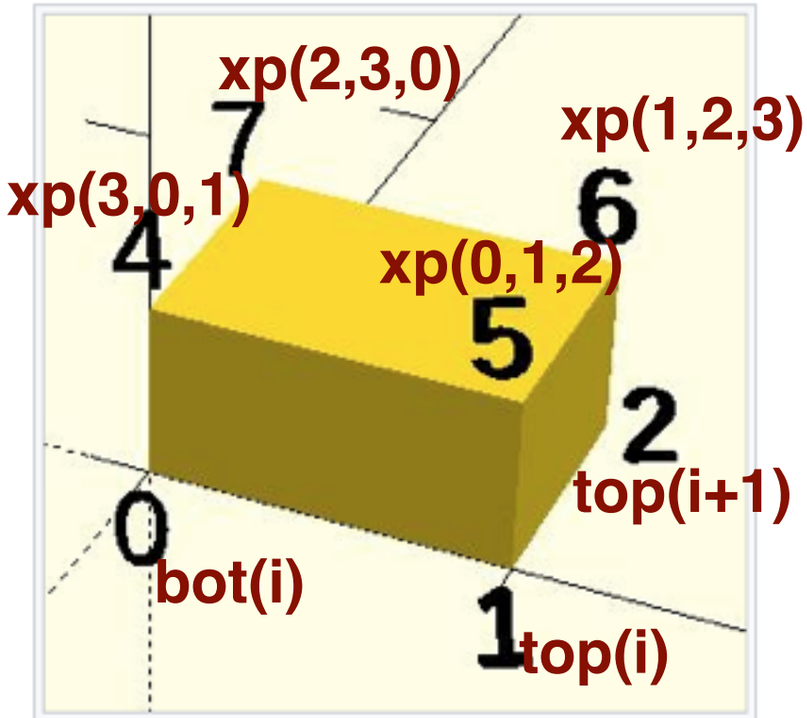


Cross product - Wikipedia
https://en.wikipedia.org/wiki/Cross_product

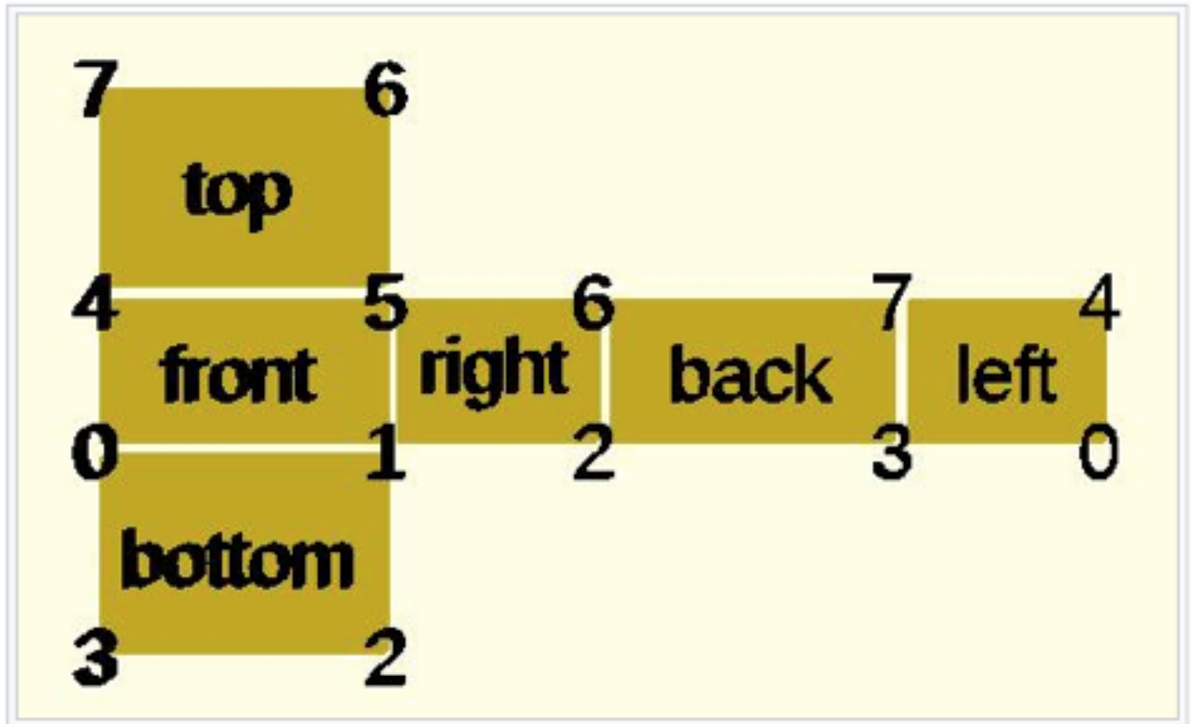
Discovery: polyhedron requires points be traced in a specific order. It is easy to get this wrong even when carefully following the example.

Discovery: Offset perimeter point selection will twist the construction but leads to polyhedron with non-planer corners. Better to bend slabs by interpolating intermediate corners as planed.

Example 1 Using polyhedron to generate cube([10, 7, 5]);



point numbers for cube



unfolded cube faces

```
CubePoints = [
  [  0,  0,  0 ],  //0
  [ 10,  0,  0 ],  //1
  [ 10,  7,  0 ],  //2
  [  0,  7,  0 ],  //3
  [  0,  0,  5 ],  //4
  [ 10,  0,  5 ],  //5
  [ 10,  7,  5 ],  //6
  [  0,  7,  5 ]]; //7

CubeFaces = [
  [0,1,2,3],  // bottom
  [4,5,1,0],  // front
  [7,6,5,4],  // top
  [5,6,2,1],  // right
  [6,7,3,2],  // back
  [7,4,0,3]]; // left

polyhedron( CubePoints, CubeFaces );
```

// outside (bottom)
o = [bot(i), top(i), top(i+1), bot(i+1)]

// inside (top) from cross product
i = [
  [xp(o(3), o(0), o(1))],
  [xp(o(0), o(1), o(2))],
  [xp(o(1), o(2), o(3))],
  [xp(o(2), o(3), o(0))]
]

CubePoints = concat(o,i)

twist.scad

Editor

```
1  n = 32;
2
3  function psin (i) = sin(360*i/n);
4  function pcos (i) = 1.5*cos(360*i/n);
5
6  function twist (i) = (i + n/4) % n;
7  function top (i) = [10*psin(twist(i)), 10*pcos(twist(i)), 20];
8  function bot (i) = [10*psin(i), 10*pcos(i), 0];
9
10 function xp (a, b, c) =
11     let (w = cross(a-b, c-b))
12     3 *  w / norm(w) + b;
13
14 module slab(i) {
15     o0 = bot(i);   o1 = top(i);
16     o2 = top(i+1); o3 = bot(i+1);
17     CubePoints = [
18         xp(o3, o0, o1),
19         xp(o0, o1, o2),
20         xp(o1, o2, o3),
21         xp(o2, o3, o0),
22         o0, o1, o2, o3];
23     CubeFaces = [
24         [0,1,2,3],  // bottom
25         [4,5,1,0],  // front
26         [7,6,5,4],  // top
27         [5,6,2,1],  // right
28         [6,7,3,2],  // back
29         [7,4,0,3]]; // left
30     polyhedron( CubePoints, CubeFaces, 3 );
31 }
32
33 module fairing () {
34     for (i = [0:n]) {
35         slab(i);
36     }
37 }
38
39 difference() {
40     fairing();
41     translate([0,0,10]) cube(25);
42     translate([-20,-20,-40]) cube(40);
43 }
```

Viewport: translate = [-3.50 -1.62 9.65], rotate = [113.80 0.00 128.90], distance = 98.36

OpenSCAD 2015.03