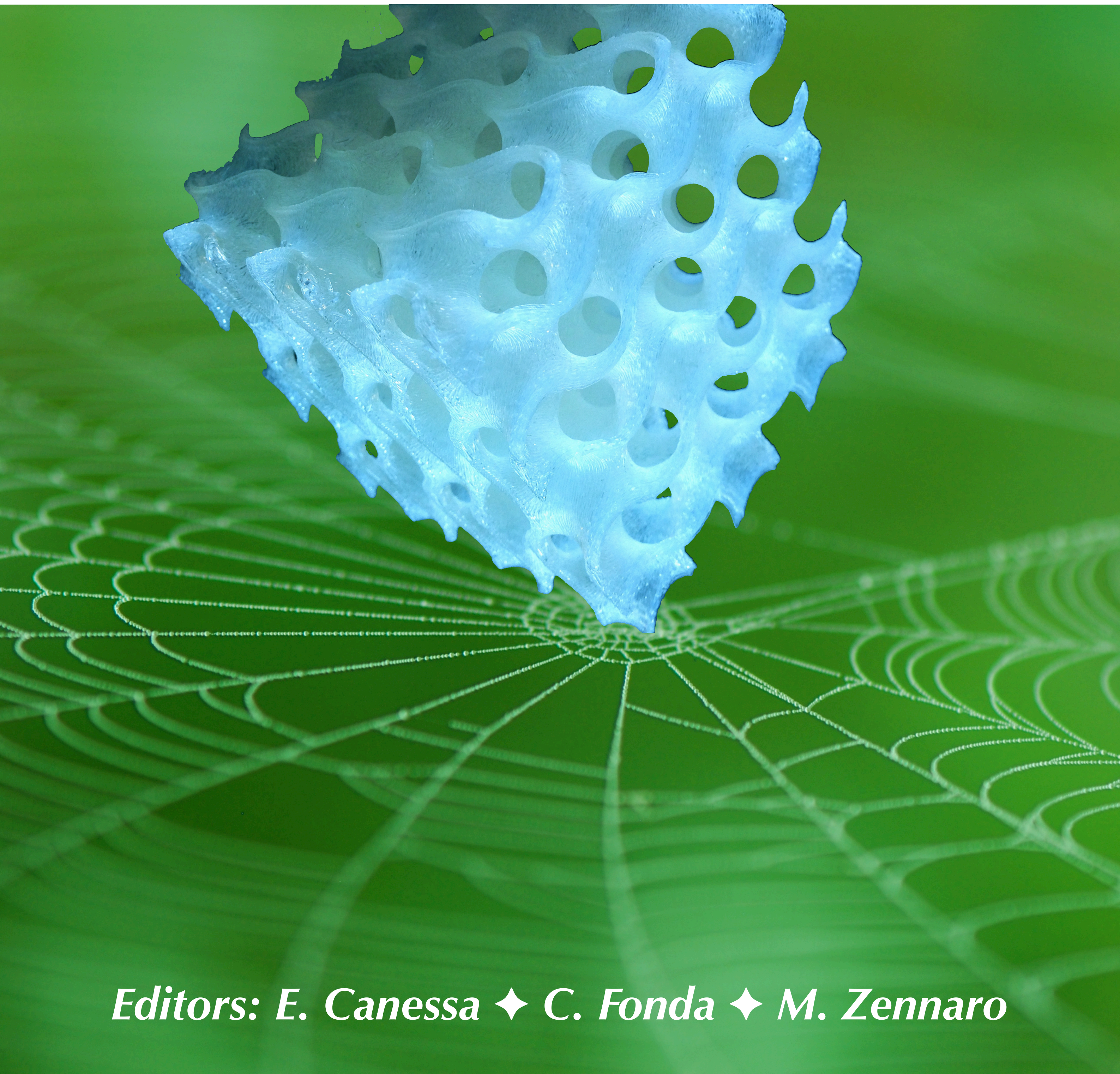
LOW-COST 3D PRINTING

FOR SCIENCE, EDUCATION & SUSTAINABLE DEVELOPMENT



Editors: E. Canessa ✦ C. Fonda ✦ M. Zennaro

Low-cost 3D Printing

for Science, Education & Sustainable Development

For more information visit us at http://sdu.ictp.it/3D/

Editors: Enrique Canessa, Carlo Fonda and Marco Zennaro

Publisher

ICTP—The Abdus Salam International Centre for Theoretical Physics

2013 ICTP Science Dissemination Unit, e-mail: sdu@ictp.it

Printing history: May 2013, First Edition

ISBN 92-95003-48-9

About this Book

Credits

This book was prepared for the First International Workshop on **"Low-cost 3D Printing for Science, Education & Sustainable Development"** held in Trieste, Italy in 2013, organized by the ICTP Science Dissemination Unit (SDU). The purpose of this activity was to discuss and create awareness on the new 3D printing through demonstrations on a number of available competing technologies, as well as presentations of ongoing research into new applications. Special focus was given to the applicability of 3D printing to promote appropriate technologies for sustainable development, scientific research and education.

Editors

Enrique Canessa is a PhD Physicist working as Coordinator of ICTP-SDU. His main areas of research are in the fields of condensed matter and scientific software applications, with particular focus on disseminating science to, and within, developing countries using open source, rich-media, mobile technologies and also 3D printing.

Carlo Fonda works for the ICTP-SDU. He is also involved in training and projects on low-cost wireless telecommunications for education, research and development. His interests include, among others, computer programming, rich-media and webcasting for science, use of mobile devices for education, and also 3D printing.

Marco Zennaro received his Engineering degree in Electronics from the University of Trieste, Italy and his PhD from KTH-Royal Institute of Technology, Stockholm, Sweden. He is currently working at ICTP in projects involving wireless sensor networks, multimedia, open access and Arduino-based technologies. His research interests are also on information and communication Technologies for development (ICT4D).

Funding

Development and publication of this book have been made possible with funding support from The Abdus Salam International Centre for Theoretical Physics, Trieste, Italy.

Special Thanks

Our sincere thanks go to the many authors of the contributions written for the book. We have given proper attribution to the author(s) of each section and/or chapter included in this work. We would also like to acknowledge the contribution and support of Professor Fernando Quevedo (ICTP Director), Ms. Mary Ann Williams, Ms. Lisa Iannitti, Mr. Jonathan Gatti, Ms. Gaya Fior and the Multimedia Publications & Printing Services of ICTP, and everyone else who has made this project possible.

ICTP The Abdus Salam International Centre for Theoretical Physics

UNESCO United Nations Educational, Scientific and Cultural Organization

IAEA International Atomic Energy Agency

ICTP Science Dissemination Unit

3D Modeling with OpenSCAD - *Part 1*

Sebastian Büttrich

pITLab, IT University of Copenhagen, Denmark

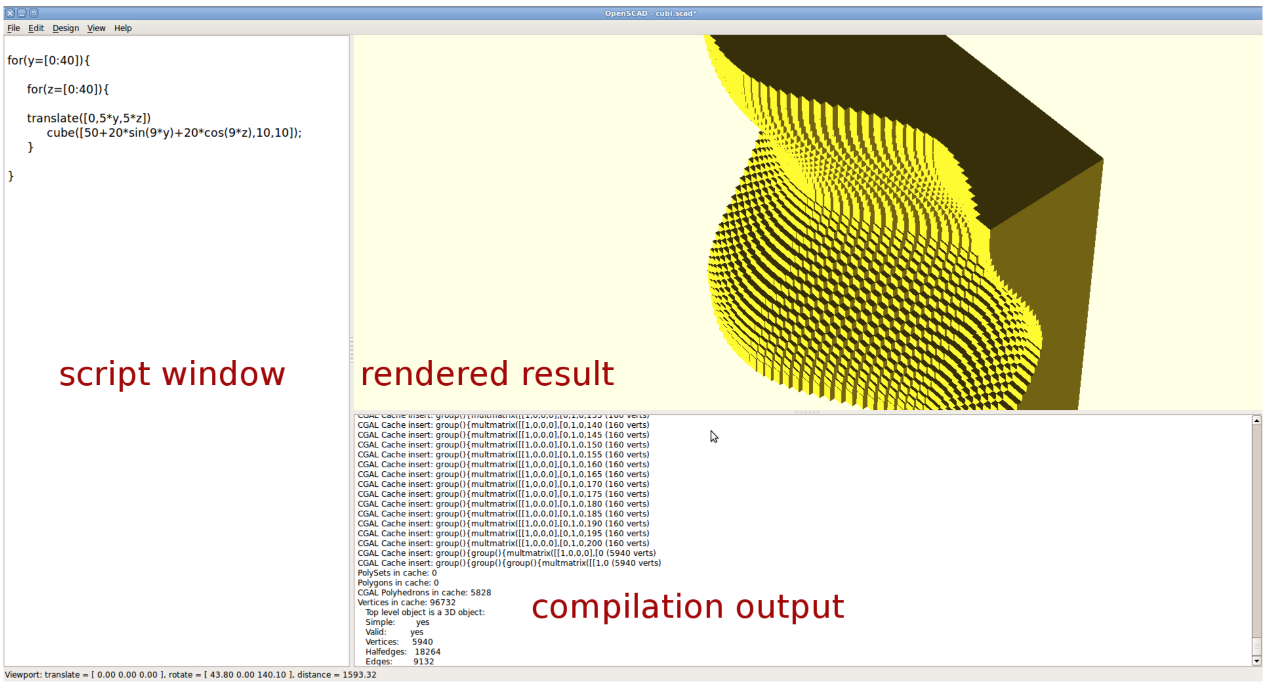
sebastian@itu.dk

On the way from idea to finished 3D print, there are a number of different steps to perform. Starting with the design of a CAD file or the capture of an existing object, followed by the conversion to an STL file, possibly some post-processing/repair work, and finally to the conversion to a printer-executable gcode file.

Your first steps in 3D printing might be based on 3D designs found on the internet, but when you are getting serious, you will want to design your own, or improve existing designs, rather than just replicating the work of others. We will focus on the design step here –*i.e.*, the production of 3D models and export of STL files.

There are many software tools available, and the following two URLs are good starting points for learning about them:

- http://www.reprap.org/wiki/Useful_Software_Packages
- https://en.wikipedia.org/wiki/Comparison_of_3D_computer_graphics_software



```
for(y=[0:40]){

    for(z=[0:40]){

    translate([0,5*y,5*z])
        cube([50+20*sin(9*y)+20*cos(9*z),10,10]);
    }

}
```

```
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,140 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,145 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,150 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,155 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,160 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,165 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,170 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,175 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,180 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,185 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,190 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,195 (160 verts)
CGAL Cache insert: group(){multmatrix([[1,0,0,0],[0,1,0,200 (160 verts)
CGAL Cache insert: group(){group(){multmatrix([[1,0,0,0],[0 (5940 verts)
CGAL Cache insert: group(){group(){group(){multmatrix([[1,0 (5940 verts)
PolySets in cache: 0
Polygons in cache: 0
CGAL Polyhedrons in cache: 5828
Vertices in cache: 96732
  Top level object is a 3D object:
  Simple:      yes
  Valid:       yes
  Vertices:    5940
  Halfedges:   18264
  Edges:       9132
```

```
Viewport: translate = [ 0.00 0.00 0.00 ], rotate = [ 43.80 0.00 140.10 ], distance = 1593.32
```

Designing in OpenSCAD

The most popular free and open source software are Blender, POV-Ray, Wings3d and OpenSCAD. OpenSCAD is suitable for anything which may be calculated and generated by code and logic rather than by freehand, mouse moves or light tracing. For the latter ones, Blender or POV-Ray might be your choice.

So when the task is to design objects of which you know the precise measures, or objects that would be cumbersome or impossible to draw, but are readily described by formulas, parameters or series, OpenSCAD is the right tool for you. Its approach to 3D design is based on mathematics and programming.

Quoting from its website http://openscad.org:

> *"Unlike most free software for creating 3D models (such as the famous application Blender) it does not focus on the artistic aspects of 3D modeling but instead on the CAD aspects. Thus it might be the application you are looking for when you are planning to create 3D models of machine parts but pretty sure is not what you are looking for when you are more interested in creating computer-animated movies."*

OpenSCAD is free software, available for Linux/UNIX, MS Windows and Mac OS X, under a GNU GENERAL PUBLIC LICENSE Version 2.

In OpenSCAD, there are two basic modeling techniques:

1. Constructive solid geometry (CSG) is the construction of full 3-dimensional objects, element by element, from script.

2. Extrusion of 2D outlines on the other hand takes existing 2-dimensional shapes, *e.g.* in the form of a DXF file or a simple 2-dimensional shape, and derives the 3D object from this, for example by rotation or elevation.

The resulting 3D file may then be exported in file formats STL or OFF.

STL stands for STereoLithography. It is a format available for export in most CAD programs. An STL file represents an object that you may call "watertight": an object without holes or singularities. While more adventurous objects of course can be imagined and drawn, only a "watertight" object, an object that can be filled with matter, can be printed in real life.

It should be mentioned that exporting to STL can be a problematic process, and it is always a good idea to check results by means of a post-processing and repair tool like Meshlab.

The basic syntax elements of OpenSCAD are variables, modules, functions, inclusions and requirements.

Variables are declared like

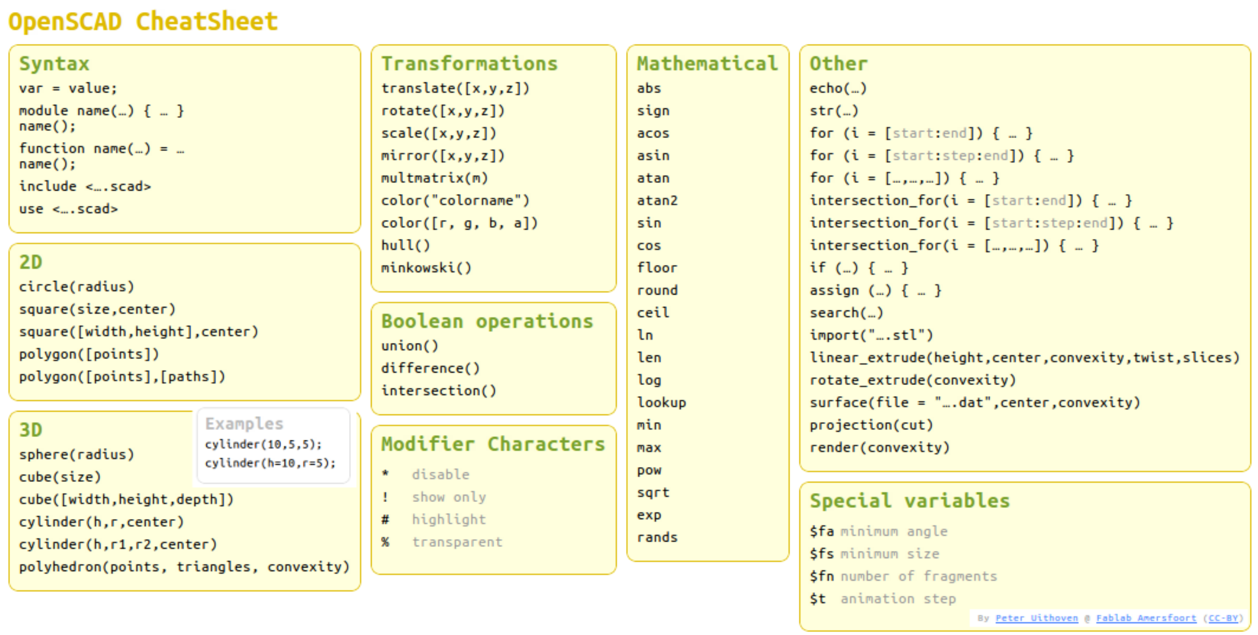
```
myVar = 5+4
```

and may be grouped into vectors/points like this:

```
myVector = [5,4,8];
```

Variables are set once at compile time, and will not change at runtime.

OpenSCAD knows scalar arithmetic operators, relational operators, boolean logic operators and a long list of common mathematical functions. You can create 2D (circle square, polygon) and 3D (cubes, spheres, cylinders) primitives, all of which take parameters like the points introduced above as input, often complemented with resolution/facet parameters and additional instructions.



OpenSCAD CheatSheet

Syntax
```
var = value;
module name(…) { … }
name();
function name(…) = …
name();
include <….scad>
use <….scad>
```

2D
```
circle(radius)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
```

3D
```
sphere(radius)
cube(size)
cube([width,height,depth])
cylinder(h,r,center)
cylinder(h,r1,r2,center)
polyhedron(points, triangles, convexity)
```

Examples
```
cylinder(10,5,5);
cylinder(h=10,r=5);
```

Transformations
```
translate([x,y,z])
rotate([x,y,z])
scale([x,y,z])
mirror([x,y,z])
multmatrix(m)
color("colorname")
color([r, g, b, a])
hull()
minkowski()
```

Boolean operations
```
union()
difference()
intersection()
```

Modifier Characters
```
*  disable
!  show only
#  highlight
%  transparent
```

Mathematical
```
abs
sign
acos
asin
atan
atan2
sin
cos
floor
round
ceil
ln
len
log
lookup
min
max
pow
sqrt
exp
rands
```

Other
```
echo(…)
str(…)
for (i = [start:end]) { … }
for (i = [start:step:end]) { … }
for (i = […,…,…]) { … }
intersection_for(i = [start:end]) { … }
intersection_for(i = [start:step:end]) { … }
intersection_for(i = […,…,…]) { … }
if (…) { … }
assign (…) { … }
search(…)
import("….stl")
linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(convexity)
surface(file = "….dat",center,convexity)
projection(cut)
render(convexity)
```

Special variables
```
$fa minimum angle
$fs minimum size
$fn number of fragments
$t  animation step
```

By Peter Uithoven @ Fablab Amersfoort (CC-BY)

The OpenSCAD cheatsheet at http://www.openscad.org/cheatsheet/ gives a good summary of all OpenSCAD language elements

The following example code shows the translate transformation and the three basic boolean operations:

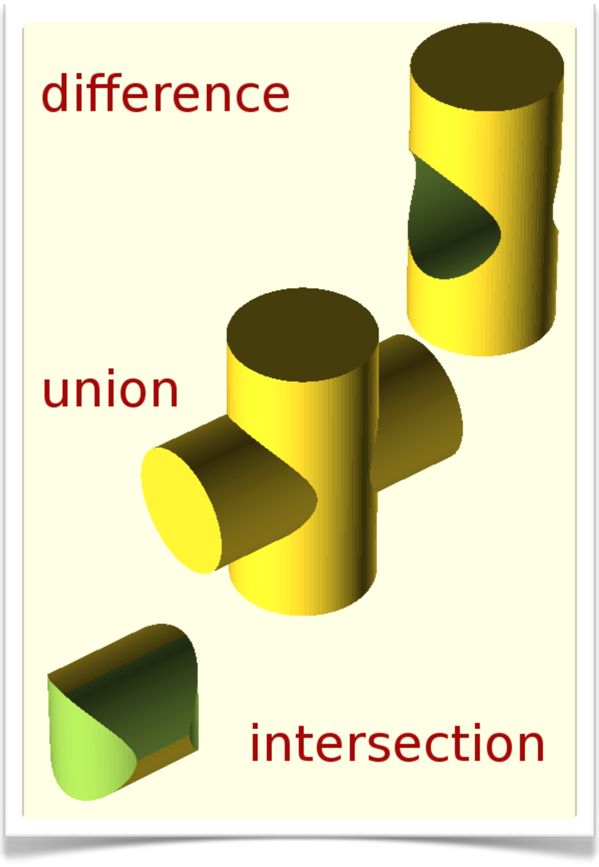
```
union (objects joined together),
difference (cutting one object out of the other);
intersection (the shared space of two objects).
```

Example code of 3 basic transformations:

```
union() {
        cylinder (h = 4, r=1, center = true, $fn=100);
        rotate ([90,0,0]) cylinder (h = 4, r=0.9, center = true, $fn=100);
}

translate([0,3,3])
difference() {
        cylinder (h = 4, r=1, center = true, $fn=100);
        rotate ([90,0,0]) cylinder (h = 4, r=0.9, center = true, $fn=100);
}

translate([0,-3,-3])
intersection() {
        cylinder (h = 4, r=1, center = true, $fn=100);
        rotate ([90,0,0]) cylinder (h = 4, r=0.9, center = true, $fn=100);
}
```



Basic boolean operations

A detailed OpenSCAD User Manual is hosted on wikibooks:

http://en.wikibooks.org/wiki/OpenSCAD_User_Manual

and it supplies all the information you will need in order to design complex objects.

Note: *All URLs in this article visited April 2013.*

3D Modeling with OpenSCAD - *Part 2*

Marius Kintel

OpenSCAD developer, Austria

marius@kintel.net

Some words from the author

OpenSCAD grew out of the RepRap community, more exactly out of the 3D printing activities at the Metalab (http://metalab.at), a hackerspace in Vienna, Austria.

The idea of OpenSCAD was born because we lacked a free software design tool for rapidly and iteratively creating mechanical parts. The existing tools at the time were too time-consuming to use and changing details often required full remodeling. Commercial CAD tools which solve these problems do exist. However, apart from them being prohibitively expensive, they weren't Open Source and we felt that the world needed a better Open Source design tool.The basic idea of OpenSCAD was to allow people to describe their 3D models beginning with basic building blocks, and iteratively build from there. Additionally we wanted it to be possible to parametrically describe shapes and positions in order to facilitate customizations and adaptations without having to go through time consuming and boring remodeling tasks.

Early on, we realized that OpenSCAD would have severe limitations in terms of creating geometric shapes, so we decided to enable users to model more complex building blocks in their software of choice. OpenSCAD can then and import these files for further modeling, while you at any time can go back and change the basic geometry without having to redo the work already done in OpenSCAD. Keeping source code as the user interface also has an important emergent property in that people are enticed to share their designs, as well as their design intentions. This also makes it possible to change, reuse, or in other ways build on the existing ideas and designs of other people.

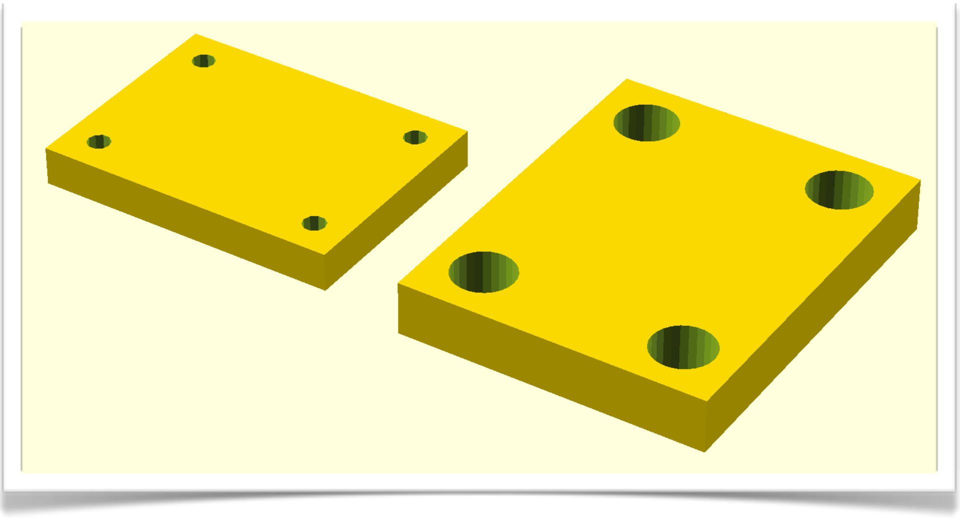
Parametric designs

One of the primary strengths of OpenSCAD is that it supports parametric designs. Parametric in this context means that you can create logical building blocks, which take certain parameters and in return create a 3D component satisfying those parameters. Examples of parameters can be *Object sizes*, *Nut and bolt holes*, *Object descriptors* (*e.g., number of teeth in a gear*) or *Design elements* (text to emboss onto a design).

In OpenSCAD, building blocks are called modules. A module is a type of template, which is defined once and can then be used multiple times with different parameters. The following code defines a module, *TopPlate*, which describes a parametric plate with four screwholes. The module parameters are plate dimensions and screw size. The *TopPlate* module is then instantiated twice:

```
module TopPlate(size, screwsize)
{
  margin = 3;
  offset = screwsize/2 + margin;
  difference() {
    cube(size);
    for (xoffset = [offset, size[0] - offset], yoffset = [offset, size[1] - offset])    {
      translate([xoffset, yoffset, -0.5]) cylinder(r=screwsize/2, h=size[2]+1);
      }
    }
}

TopPlate(size=[40,50,3], screwsize=8);
translate([50,0,0]) TopPlate(size=[40,30,5], screwsize=3);
```



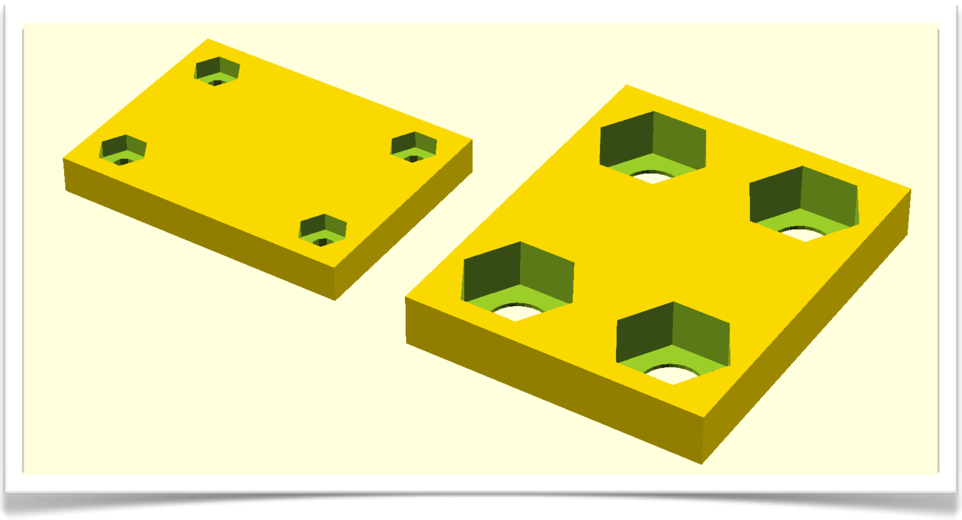
Libraries

A lot of modeling tasks, especially when creating mechanical parts or assemblies, consist of repetitive use of standard elements like fasteners, holes, slots etc. In addition to defining modules that facilitate reuse of component within one design it is also possible to use external *libraries*. OpenSCAD includes a collection of common components in a library called MCAD.

The following example builds on the previous one by adding captive nuts to the existing screw holes, and uses the MCAD library's **nuts_and_bolts** module to get the correct dimensions of the nut corresponding to the screw size:

```
include<MCAD/nuts_and_bolts.scad>
module screwhole(screw, depth)
{
  translate([0,0,-0.5]) cylinder(r=screw/2, h=depth+1);
  translate([0,0,-0.01]) nutHole(size=screw);
}
module TopPlate(size, screwsize)
{
  margin = 2;
  offset = METRIC_NUT_AC_WIDTHS[screwsize]/2 + margin;
  difference() {
    cube(size);
    for (xoffset = [offset, size[0] - offset], yoffset = [offset, size[1] - offset]) {
      translate([xoffset, yoffset, 0]) screwhole(screwsize, size[2]);
    }
  }
}

TopPlate(size=[40,50,7], screwsize=8);
translate([50,0,0]) TopPlate(size=[40,30,5], screwsize=3);
```



As a result of the nature of OpenSCAD designs, libraries are shared simply by sharing the source code of your modules. Many modelers have created component libraries, and have shared them online. There is a number of OpenSCAD libraries on Thingiverse:

http://www.thingiverse.com/search?q=openscad+library

Usage examples

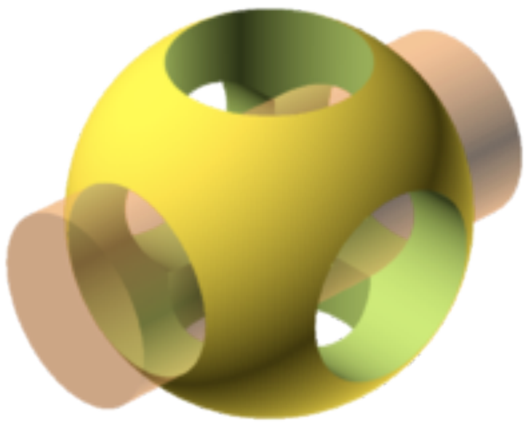
Since OpenSCAD grew out of the early 3D printing and RepRap movement, the user base is still by far the strongest within these communities. As a result, some of the most prominent examples of OpenSCAD usage is the design of 3D printers themselves.

Some examples are:

- RepRap Prusa iteration 3:
  https://github.com/josefprusa/Prusa3
- Lulzbot AO-100 (partially):
  http://download.lulzbot.com/AO-100/hardware/printed_parts/source/
- Lulzbot TK-0:
  https://github.com/mswillia/TK-0
- RepRap Mendel90:
  http://hydraraptor.blogspot.co.uk/2012/12/mendel90-updates.html

To find OpenSCAD designs online, the largest repository is Thingiverse:

http://www.thingiverse.com/tag:openscad



From Math to Jewel: an Example

Gaya Fior

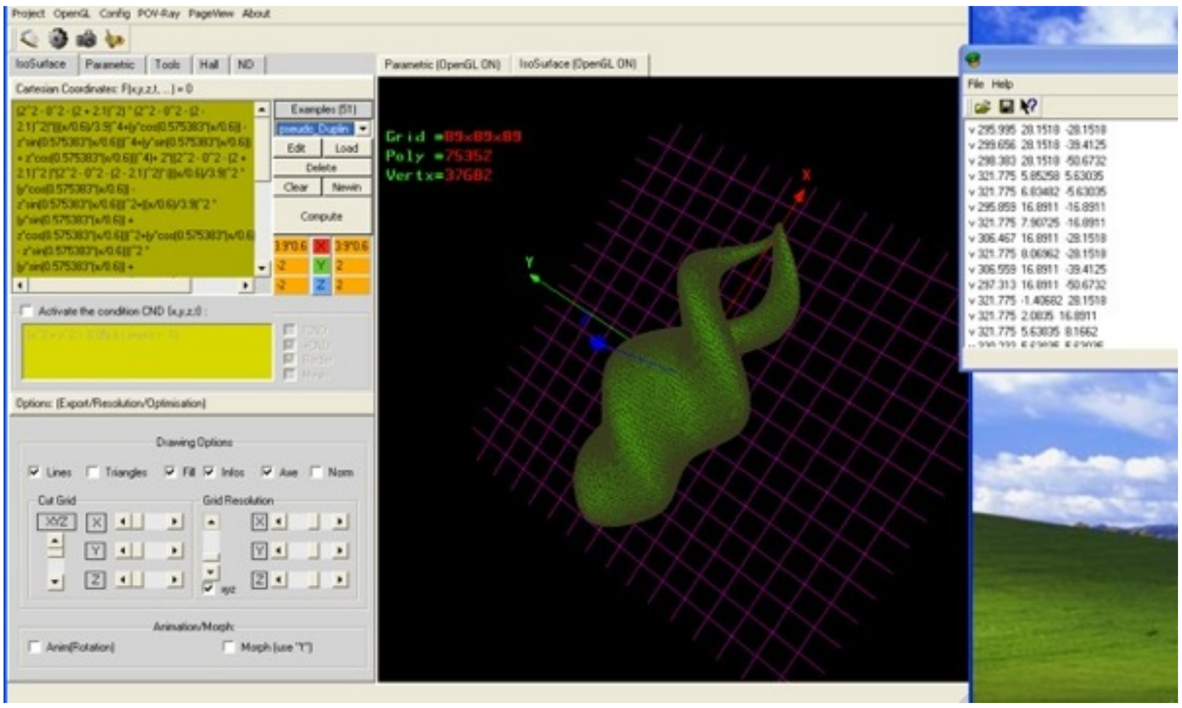
ICTP Science Dissemination Unit collaborator
and 32b.it, Trieste, Italy
gfiorfior@gmail.com

3D printing gives the possibility to transform what you can imagine into a tangible object that then can be also worn and showed off.

We will see how using just free tools available on the web we can transform a mathematical isosurface into an object that can be then used for instructional or decorative purposes.

The first step is to download a software that lets us visualize and manipulate mathematical surfaces in three dimensions. A good choice is K3DSurf[1], a free tool that works on multiple platforms and supports parametric equations and isosurfaces.

The software comes with more than 50 built-in examples, so you can start modifying the parameters in the provided equations to study the effects on the rendering result.


Project OpenGL Config POV-Ray PageView About
IsoSurface Parametric Tools Hall ND
Parametric (OpenGL ON) IsoSurface (OpenGL ON)
Examples (51)
pseudo_Duplin
Edit Load Delete Clear Newin Compute
File Help

Cartesian Coordinates: F(x,y,z,t, ...) = 0

```
(2^2 - 0^2 - (2 + 2.1)^2) * (2^2 - 0^2 - (2 -
2.1)^2)*((x/0.6)/3.9)^4+(y*cos(0.575383*(x/0.6)) -
z*sin(0.575383*(x/0.6)))^4+(y*sin(0.575383*(x/0.6))
+ z*cos(0.575383*(x/0.6)))^4)+ 2*((2^2 - 0^2 - (2 +
2.1)^2 )*(2^2 - 0^2 - (2 - 2.1)^2)*((x/0.6)/3.9)^2 *
(y*cos(0.575383*(x/0.6)) -
z*sin(0.575383*(x/0.6)))^2+((x/0.6)/3.9)^2 *
(y*sin(0.575383*(x/0.6)) +
z*cos(0.575383*(x/0.6)))^2+(y*cos(0.575383*(x/0.6)
 - z*sin(0.575383*(x/0.6)))^2 *
(y*sin(0.575383*(x/0.6)) +
```

X: -3.9*0.6 to 3.9*0.6; Y: -2 to 2; Z: -2 to 2

Activate the condition CND (x,y,z,t) :

Options: (Export/Resolution/Optimisation)

Drawing Options: [✓ Lines] Triangles [✓ Fill] [✓ Infos] [✓ Axe] Norm

Cut Grid: XYZ X Y Z — Grid Resolution: X Y Z

Animation/Morph: Anim(Rotation) Morph (use "t")

```
Grid =89x89x89
Poly =75352
Vertx=37682
```

```
v 295.995 28.1518 -28.1518
v 299.656 28.1518 -39.4125
v 298.383 28.1518 -50.6732
v 321.775 5.85258 5.63035
v 321.775 6.03402 -5.63035
v 295.859 16.8911 -16.8911
v 321.775 7.90725 -16.8911
v 306.467 16.8911 -28.1518
v 321.775 8.06962 -28.1518
v 306.559 16.8911 -39.4125
v 297.313 16.8911 -50.6732
v 321.775 -1.40682 28.1518
v 321.775 2.0835 16.8911
v 321.775 5.63035 8.1662
```

The K3DSurf interface

After playing around with the examples, you can start inserting your own equations in the text field, keeping in mind that the software requires the right-hand-side of the equation to be zero.

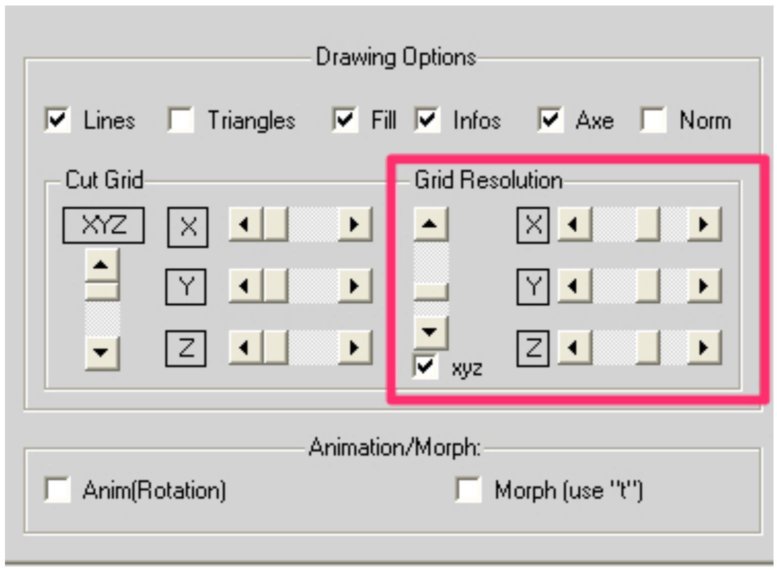
Some websites with interesting surface formulas are:

- Implicit Algebraic Surfaces
  http://www-sop.inria.fr/galaad/surface/
- University of Turin
  http://www.dm.unito.it/modelli/index.html
- Geometry, surfaces, curves and polyhedra by Paul Bourke
  http://paulbourke.net/geometry/
- Virtual mathematics museum
  http://virtualmathmuseum.org
- Java based tool that gives you the possibility to modify all parameters and visually see the result; you can then copy the corresponding equation on the formula tab
  http://www.javaview.de/demo/PaSurface.html
- Java based tool to calculate singular algebraic curves and surfaces
  http://www.singsurf.org/singsurf/SingSurf.html
- Isosurface Tutorial by Mike Williams
  http://www.econym.demon.co.uk/isotut/

For the following example we are going to use one of the build in examples provided with the software; an isosurface called pseudo-Duplin that was chosen because it looked interesting while still providing all the characteristics necessary for giving good printing results on a low-cost 3D printer.

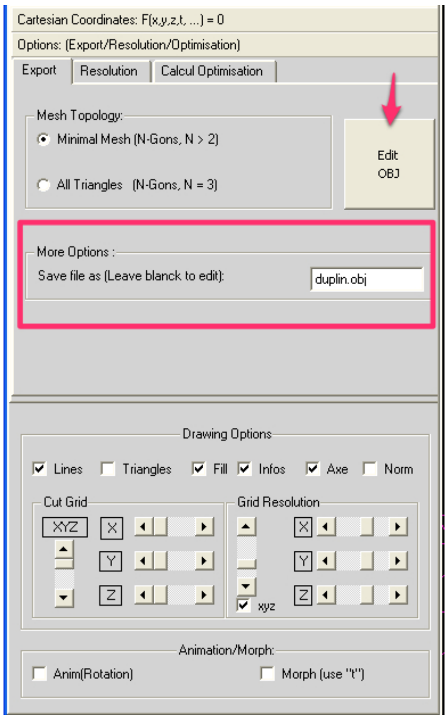
```
(2^2 - 0^2 - (2 + 2.1)^2) * (2^2 - 0^2 - (2 - 2.1)^2)*(((x/0.6)/3.9)^4+(y*cos(0.575383*(x/0.6))
- z*sin(0.575383*(x/0.6)))^4+(y*sin(0.575383*(x/0.6)) + z*cos(0.575383*(x/0.6)))^4)+
2*((2^2 - 0^2 - (2 + 2.1)^2 )*(2^2 - 0^2 - (2 - 2.1)^2)* (((x/0.6)/3.9)^2 * (y*cos(0.575383*(x/
0.6)) - z*sin(0.575383*(x/0.6)))^2+((x/0.6)/3.9)^2 * (y*sin(0.575383*(x/0.6)) +
z*cos(0.575383*(x/0.6)))^2+(y*cos(0.575383*(x/0.6)) - z*sin(0.575383*(x/0.6)))^2 *
(y*sin(0.575383*(x/0.6)) + z*cos(0.575383*(x/0.6)))^2))+2* 2^2 *((-0^2-2^2+2^2+2.1^2)* (2
*((x/0.6)/3.9) *2+2* (y*cos(0.575383*(x/0.6)) - z*sin(0.575383*(x/0.6)))* 0-2^2)-4*0 *2.1^2
*(y*cos(0.575383*(x/0.6)) - z*sin(0.575383*(x/0.6))))*(((x/0.6)/3.9)^2+(y*cos(0.575383*(x/
0.6)) - z*sin(0.575383*(x/0.6)))^2+(y*sin(0.575383*(x/0.6)) + z*cos(0.575383*(x/0.6)))^2)+
4 * 2^4 * (2 *((x/0.6)/3.9)+0 *(y*cos(0.575383*(x/0.6)) - z*sin(0.575383*(x/0.6))))* (-2^2+0 *
(y*cos(0.575383*(x/0.6)) - z*sin(0.575383*(x/0.6)))+2 * ((x/0.6)/3.9))+4* 2^4 * 2.1^2 *
(y*cos(0.575383*(x/0.6)) - z*sin(0.575383*(x/0.6)))^2+2^8
```

After selecting the appropriate object you will want to check the grid resolution on the bottom of the window: choosing a very sparse grid will result in sharp edges and a "pixelated" appearance.



The K3DSurf drawing options tab

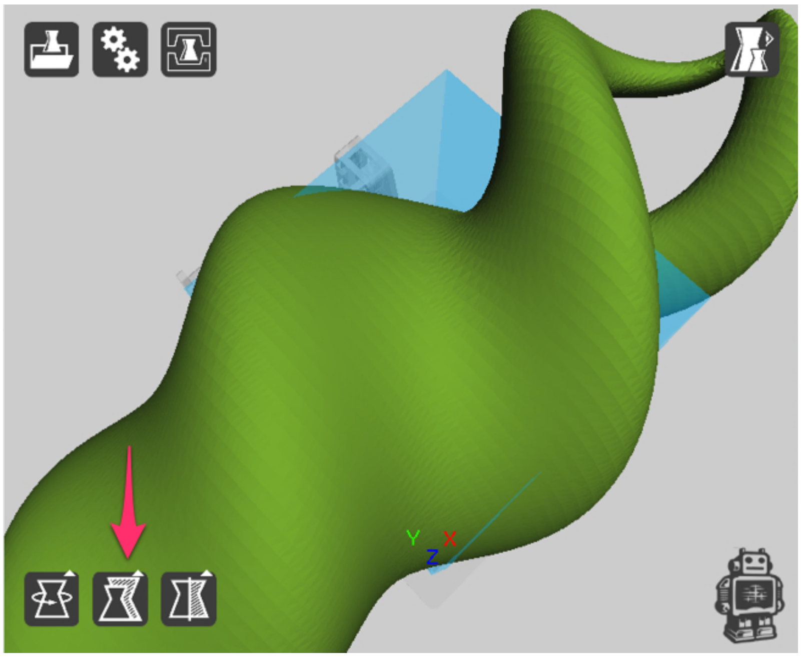
When you are satisfied with the result you can then export a OBJ file on the tab "Options: (Export/Resolution/Optimisation)".


Cartesian Coordinates: F(x,y,z,t, ...) = 0
Options: (Export/Resolution/Optimisation)
Export
Resolution
Calcul Optimisation
Mesh Topology:
[✓ Minimal Mesh (N-Gons, N > 2)]
All Triangles (N-Gons, N = 3)
Edit OBJ
More Options :
Save file as (Leave blank to edit): duplin.obj
Drawing Options
[✓ Lines] Triangles [✓ Fill] [✓ Infos] [✓ Axe] Norm
Cut Grid
XYZ
X
Y
Z
Grid Resolution
X
Y
Z
[✓ xyz]
Animation/Morph:
Anim(Rotation)
Morph (use "t")

The K3DSurf Export tab

Here you will insert the chosen name for your project (mind you have to add the *.obj* manually) and press "Edit OBJ".

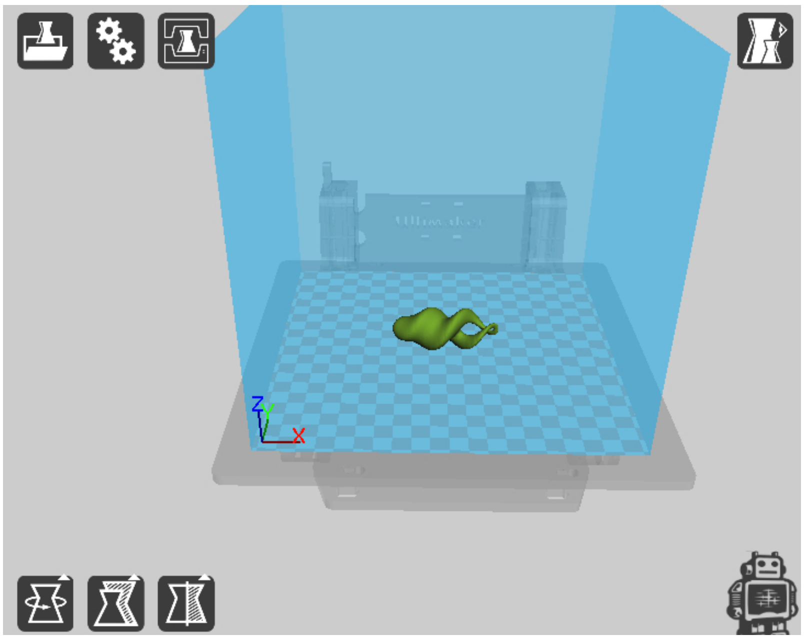
The resulting file can then opened with Netfabb Basic[2] to detect and repair errors in the triangulated mesh, if needed, and convert it to STL, the file format most commonly used by the various 3D printing software. The following steps will be shown using Cura[3], a 3D printing software developed by Ultimaker[4] that avoids the need to convert the OBJ file as it accepts also this file format.



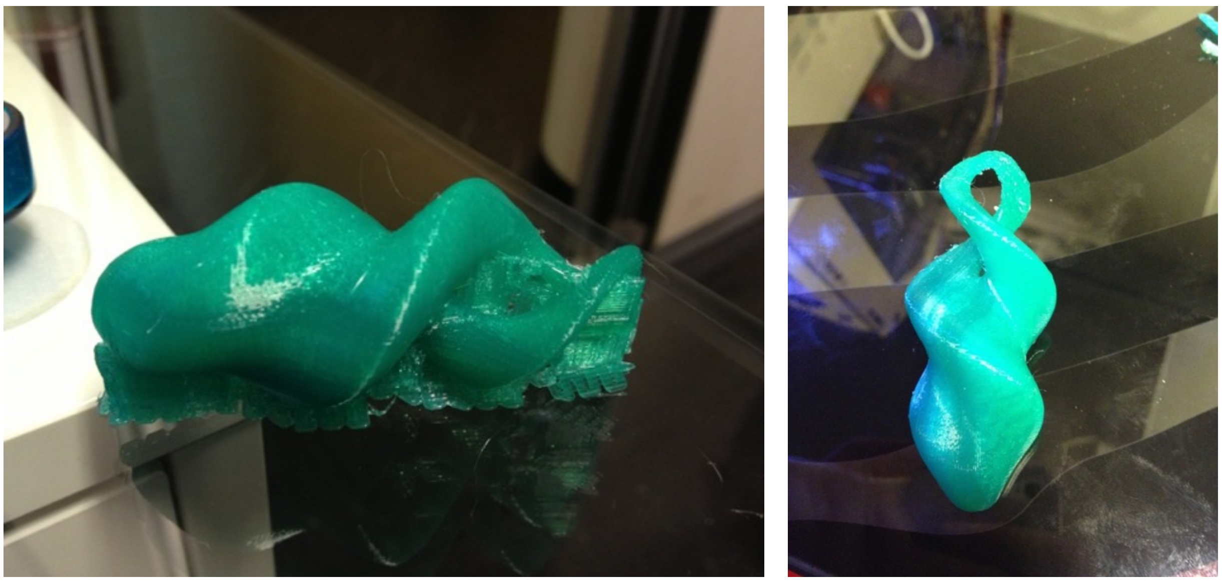
Cura's scaling option highlighted

With your slicing software you must first of all scale your object down to a reasonable size for printing, as K3DSurf will not give you the option of choosing the objects final dimensions. Cura will let you do it easily with the "scale" button, while Netfabb Basic has a "scale parts" command with which you can obtain the same result.

The object can now be sliced, keeping in mind the adequate parameters for the chosen printer, and sent to your 3D printer. Bear in mind that most mathematical isosurfaces have a curved base and empty portions and will need a raft and/or a support structure to give printable results on a low-cost 3D printer.



pseudo-Duplin scaled for printing


The chosen object printed on the Ultimaker before and after separating the support structure

Some mathematical surfaces are particularly suited for jewelry for their shape and characteristics. For instance in this example there is no need to add any kind of ring to hang the object from a chain, while in other cases you might want to manipulate the .OBJ file in a 3D modeling software to manually add a mean of hanging it.

After the test with PLA on a standard 3D printer you can then decide to send the file to a print service to have it printed out in a different material like metal or ceramics.

This example was then printed in gold plated stainless steel by i.materialise[5].

Abderrahman Taha, the developer of K3DSurf, states on the website that *"Mathematics can be so fun!"* and *"an image is worth 1000 words"*.

While both these statements are certainly true I personally think that they can also be enforced stating that *"Mathematics can be so fashionable!"* and *"a 3D object is worth 1000 images"*. Math and art traveled side by side since the ancient Egyptians started incorporating the golden ratio in their monuments[6] and today we see mathematical principles applied to everything, including fashion. A low-cost 3D printer gives us the possibility to fill the gap between imagination and creation and have in our hands a mathematical structure to study, display or wear. This can then be used ad a prototype for the next step to a professional printing service or be appreciated by itself.

Acknowledgments

http://metamagics.wordpress.com

http://k3dsurf.s4.bizhat.com

Reference

[1] http://k3dsurf.sourceforge.net

[2] http://www.netfabb.com/basic.php

[3] http://daid.github.com/Cura/

[4] http://www.ultimaker.com

[5] http://i.materialise.com

[6] http://en.wikipedia.org/wiki/Mathematics_and_art

The Abdus Salam
International Centre for Theoretical Physics

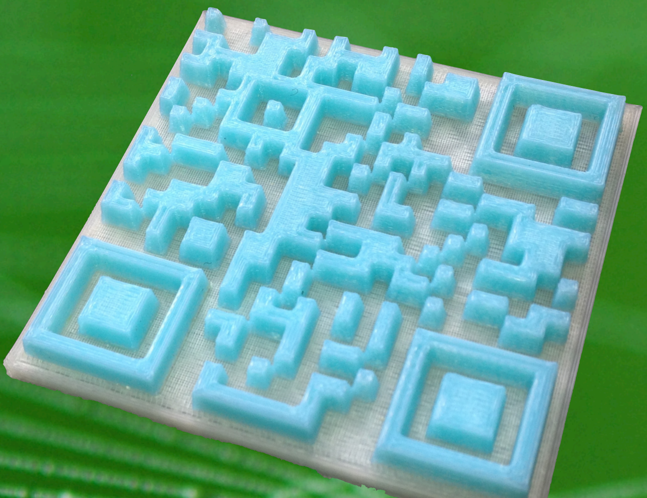
United Nations Educational, Scientific and Cultural Organization

IAEA International Atomic Energy Agency

Low-cost 3D Printing for Science, Education & Sustainable Development

Low-cost, three-dimensional (3D) desktop printing, although still in its infancy, is rapidly maturing, with seemingly unlimited potential. The hope is that this cutting-edge 3D technology will open new dimensions to science and education, and will make a marked impact in developing countries.

This book gives a reasonable, first overview of current research on 3D printing. It aims to inspire curiosity and understanding in young scholars and new generations of scientists to motivate them to start building up their own 3D printing experiences and to explore the huge potential this technology provides –*with the final goal of putting learning literally in their hands*.



Graphic design by C. Fonda.
Cover photo courtesy of G. Fior.
Published by the ICTP, © 2013.

ISBN 92-95003-48-9

9 789295 003484 >