3D Printing for Mathematical Visualisation

HENRY SEGERMAN

This column is a place for those bits of contagious mathematics that travel from person to person in the community, because they are so elegant, surprising, or appealing that one has an urge to pass them on.

Contributions are most welcome.

➢ Please send all submissions to the Mathematical Entertainments Editor, **Ravi Vakil,** Stanford University, Department of Mathematics, Bldg. 380, Stanford, CA 94305-2125, USA
e-mail: vakil@math.stanford.edu

3D printing is quickly becoming a very affordable option for producing physical objects. The term "3D printing" covers a number of closely related technologies, all of which produce a 3-dimensional physical object from a computer model by building it up in successive layers. These technologies were developed primarily for use in rapid prototyping for industrial design; they allow a designer to convert a computer model of a prototype into a physical object quickly in comparison to most previously available technologies. In addition, there are a number of other advantages that make 3D printing attractive for creating mathematical models:

1. There is a huge amount of freedom in the geometry that may be produced. With "subtractive" manufacturing techniques, in which material must be removed from an initial solid (e.g., lathing, drilling, or carving), objects with intricate internal structures can be very difficult to produce. With 3D printing's "additive" procedure, these problems are greatly mitigated.
2. The resulting object very closely approximates the mathematical ideal of the model. The workflow is

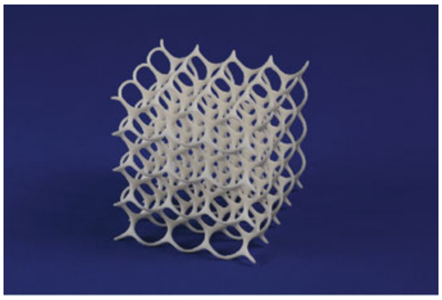
   Mathematical concept $\rightarrow$ computer model $\rightarrow$ 3D printed object

   The first arrow might involve construction "by hand" within a 3D computer-aided-design program, or the geometry might be constructed using a program (e.g., a Python script) written by the user. In either case, the geometry of the computer model can be extremely faithful to the mathematical concept. The second arrow is dealt with by a 3D printer, and these are typically accurate to a fraction of a millimetre.
3. 3D printing does not have large initial setup costs (in contrast to the process of casting from a mould for example, which must be produced first, often at great expense).

On the subject of affordability, there are now a number of web-based companies that will 3D print your models for you at very reasonable prices, so it is not necessary to own or have access to a 3D printer to be able to get started. That said, many colleges and universities have purchased 3D printers for use in their architecture or design departments. The main advantage of having direct access to a 3D printer is turnaround time: depending on the size of the object, the printing itself generally takes several hours. However, with a web-based company, it might take a week or two from the time a design is uploaded to their website until the time the 3D printed object is in one's hands.

This article is in two parts. The first shows some recent 3D printed work by a number of mathematicians and mathematical artists; it is hoped that this will inspire the reader to try the second part, which is a tutorial on

DOI 10.1007/s00283-012-9319-7


(a) A $4 \times 4 \times 4$ block of the lattice structure. Used with permission, copyright 2008, George Hart.

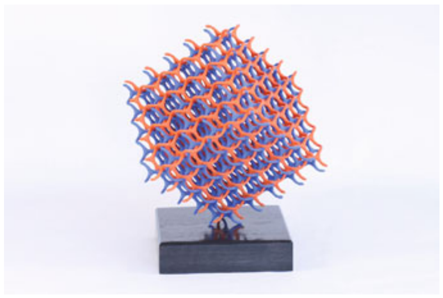

(b) Two mirror image copies of the lattice interlinked with each other. Used with permission, copyright 2012, George Hart.

Figure 1. The crystal structure (10,3)-a, 3D print design by George Hart.

producing a 3D design of a parametrically defined surface. In terms of software, the tutorial only requires access to Mathematica [15]. To go along with this article, I have also written detailed instructions for using one of the 3D printing services to fabricate your design, and I've included some notes about going further in producing objects using 3D printing. These notes are available at http://www.segerman.org/3d_printing_notes.html.

Examples of 3D Printed Visualisations

George Hart has written previously in *The Mathematical Intelligencer* about 3D printed mathematical visualisations [9], covering models of the Sierpinski Tetrahedron and the Menger Sponge, as well as various polyhedral designs and projections of 4-dimensional polytopes. Figure 1 shows two of his more recent works, based on the crystal structure (10,3)-a that was described by the chemist A. F. Wells. See [8] for more details. Bathsheba Grossman is an artist working mostly in 3D printed metal. The two examples shown in Figure 2 illustrate a technique that she developed to represent a surface with perforations. The sizes of the holes are related to the curvature of the surface. See [7]. Carlo Séquin is a computer science professor, and he often teaches courses involving computer modelling and 3D printing. Two of his recent works are shown in Figure 3, and many more can be found at [14]. Some 3D printers are able to print directly in colour, as in Figures 1b and 3b. David Bachman (Figure 4) and Burkard Polster (Figure 5, also see [12]) are both mathematicians whose 3D designs are available on shapeways.com, one of the web-based 3D printing services. I have also included two designs by myself in collaboration with Saul Schleimer in Figure 6. These objects are native to the 3-dimensional sphere, S^3, and are mapped to $\mathbb{R}^3$ via stereographic projection so they can be printed. See [13] for details. Note that the resulting 3D printed objects in Figures 1b and 6b(lower) consist of two parts that are intricately interlinked. This kind of object would be extremely difficult to manufacture by means other than 3D printing.

There are a number of other mathematical artists working with 3D printing. See also work by Vladimir Bulatov [4], whose designs involve polyhedral symmetry or 3-dimensional hyperbolic geometry, and "Nervous System" [11], who use computer simulation of natural processes, for example, reaction-diffusion.

AUTHOR

HENRY SEGERMAN is a research fellow with the Department of Mathematics and Statistics at the University of Melbourne in Australia. His mathematical research is in 3-dimensional geometry and topology, particularly involving ideal triangulations of 3-manifolds. He is also a mathematical artist, working primarily with 3D printing, often replicating topological and geometric objects. Other artistic interests include procedural generation, fractals, self reference, ambigrams, and puzzles.

Department of Mathematics and Statistics
University of Melbourne
Parkville, VIC 3010
Australia
e-mail: segerman@unimelb.edu.au

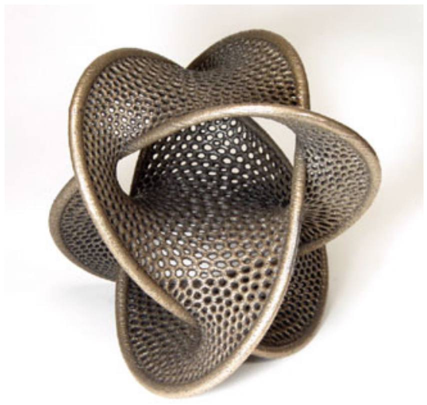
Getting Started in 3D Printing

Software

There are many more-or-less fully featured 3D design programs. These range from very expensive programs used in films and computer games (e.g., 3ds Max [2], Maya [3]) through free and open-source software (e.g., Blender [6], MeshLab [5]).[1] Many of the commercial programs include large educational discounts, although even so they can be relatively expensive. Of the free software, Blender is very full featured, but (at least at the moment) it has a somewhat nonstandard user interface and a steep learning curve. MeshLab has far fewer features, but it is useful for viewing and making small changes to mesh files (the data that is sent to a 3D printer). Both Blender and MeshLab have

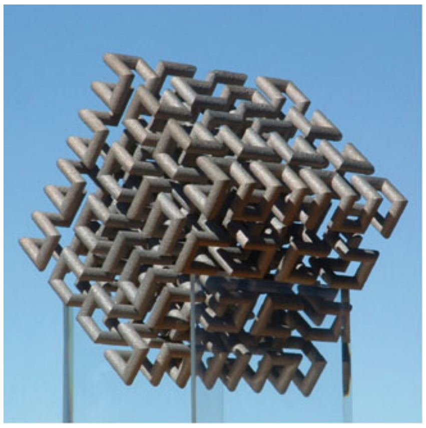
[1]Note that this is a very incomplete list.

(a) The gyroid, a triply periodic minimal surface discovered by Alan Schoen. Used with permission, copyright 2005, Bathsheba Grossman.

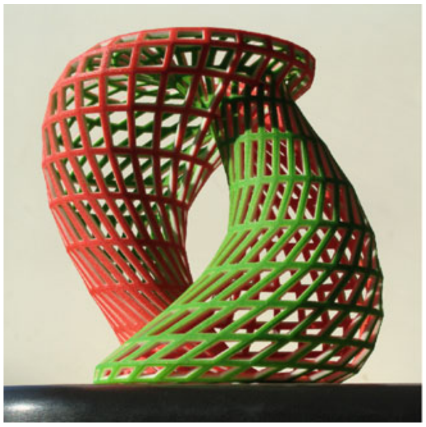


(b) A Seifert surface with boundary the Borromean rings. Used with permission, copyright 2007, Bathsheba Grossman.

Figure 2. Designs by Bathsheba Grossman.

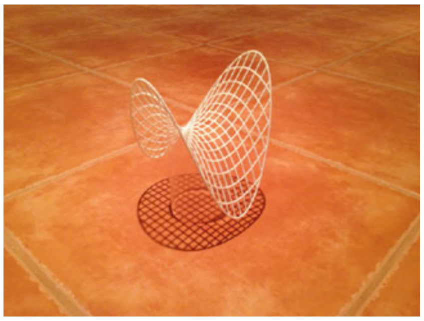


(a) "Hilbert Cube 3D", a step in the construction of a space-filling curve. Used with permission, copyright 2005, Carlo Séquin.

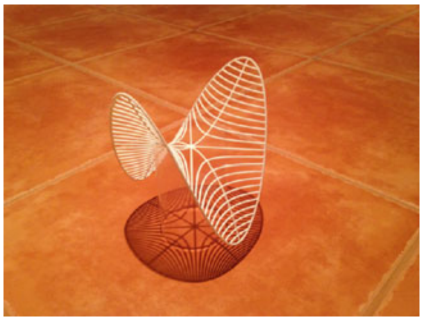


(b) "Symmetrical Half-way Point for Torus Eversion". Used with permission, copyright 2011, Carlo Séquin.

Figure 3. Designs by Carlo Séquin.



(a) A hyperbolic paraboloid, showing slices through the surface in the x and y directions. Used with permission, copyright 2010, David Bachman.



(b) A hyperbolic paraboloid, showing level curves. Used with permission, copyright 2010, David Bachman.

Figure 4. Designs by David Bachman.

versions for Windows, OS X, and Linux. I mostly use a commercial program called Rhinoceros [1] (or just Rhino), which is available for Windows, with an OS X version currently in open beta. As with many of these software packages, Rhinoceros has a scripting language (including a version based on Python), with which it is possible to

(a) A solid of constant width created by rotating a Reuleaux triangle around one of its symmetry axes. Used with permission, copyright 2009, Burkard Polster.

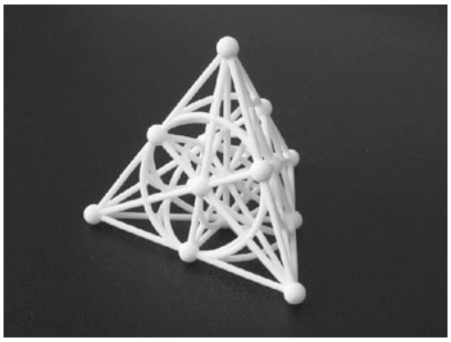
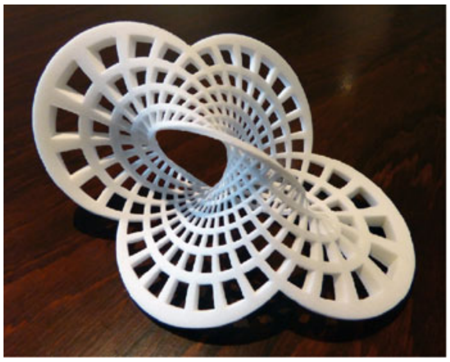

(b) A representation of $PG(3,2)$, which is the smallest finite projective space. Used with permission, copyright 2009, Burkard Polster.

Figure 5. Designs by Burkard Polster.


(a) An unusual parameterisation of a punctured Möbius strip. Used with permission, copyright 2012, Henry Segerman.

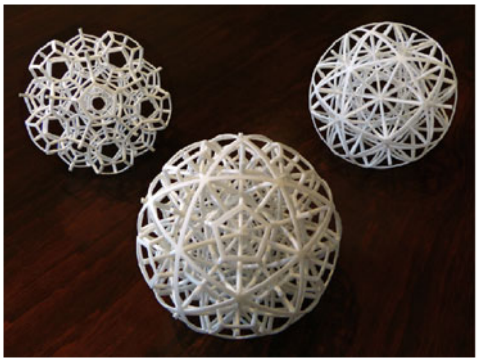

(b) Dual Half 120- and 600-cells. Used with permission, copyright 2012, Henry Segerman.

Figure 6. Designs by the author in collaboration with Saul Schleimer.

generate geometry procedurally. It is also easy to make "ruler and compass"-like constructions in 3D, and it can import 2-dimensional geometry from Adobe Illustrator. I also use MeshLab as a sanity check for any mesh files I make before sending them to a printer. Becoming familiar with one of these programs is a good idea for going further, but can get quite far using Mathematica alone. Because readers of this article very likely have access to Mathematica, we will generate an example visualisation using it.

Example: Generating a Parametric Surface Using Mathematica

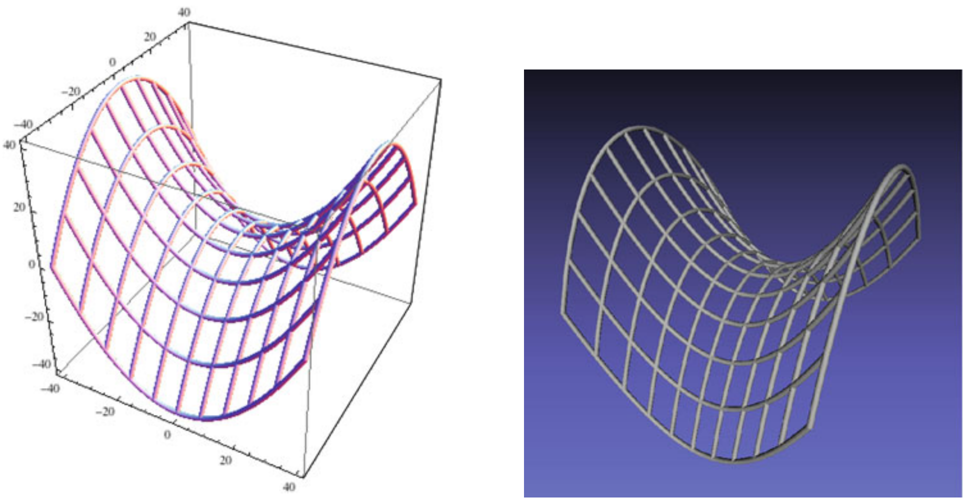
Inspired by Bachman's models in Figure 4, we will produce a gridlike representation of a parametrically defined surface, in this case a hyperbolic paraboloid. The Mathematica code[2] for this is shown in the next section. Models such as this could be very useful pedagogical tools in a multivariable calculus class, for example. The graphical output of the code is shown in Figure 7a.

In addition to this graphical output, the code also exports the graphics data as a mesh in the "STL"[3] file format. The file "MathematicaParametricSurface.stl" should appear in your root directory when the code is run. In Figure 7b, we see this file as viewed in MeshLab. The STL file format is a standard format for describing a mesh of triangles, and it is one of the formats that 3D printers take as input. The file essentially consists of a long list of triangles, each triangle given by the coordinates in 3 dimensions of its three corners. A 3D printer takes this list of triangles and interprets their union as a closed, orientable, embedded surface in $\mathbb{R}^3$, from which it can determine which voxels (3-dimensional analogues of pixels) are inside the surface (and so should be solid and filled with plastic or whatever material the printer is using) and which are outside the surface (and so should be empty). Some programs (for example MeshLab) allow you to export in an ASCII version of STL (as opposed to the usual binary version), which results in a much larger file size, but the file can also be read in any text editor, and the triangle coordinates can be read manually.[4]

[2] I have tested the code in Mathematica 7 and 8, and it may run in some earlier versions as well.

[3] "STL" stands for "Standard Tessellation Language."

[4] If one were so inclined, one could write a program to output this STL format directly, bypassing the need for any 3D software. In most cases however, this would involve the reinvention of numerous wheels.



(a) Graphical output from Mathematica. **(b)** The STL file as viewed in MeshLab.

Figure 7. Generated model of a hyperbolic paraboloid.

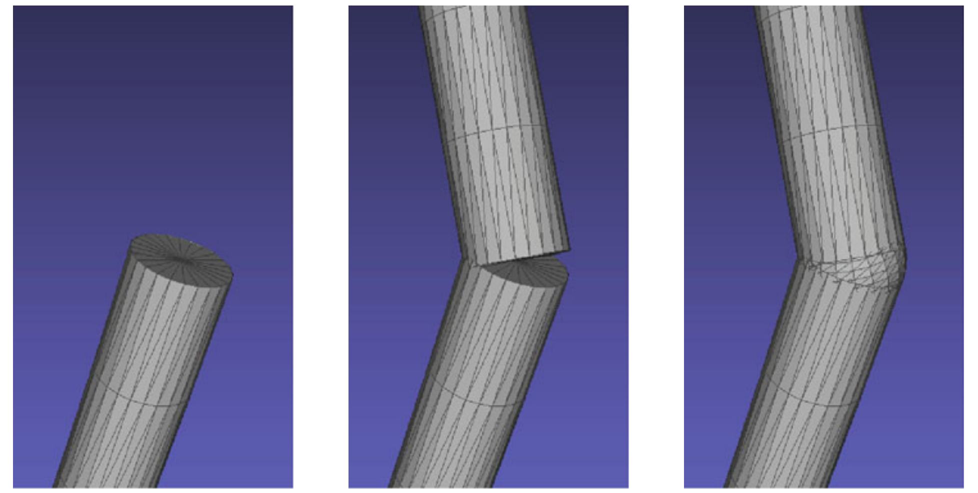
Before going into detail about how the code works (and so how to modify it!), a few words are in order regarding the choice of representation of the surface. A surface is 2-dimensional, but any real world representation of it must have some nonzero thickness. The idea used here is to draw curves on the surface given by keeping one of the two parameters constant. We "thicken up" these curves using regular tubular neighbourhoods of them. Instead, we could have thickened up the whole surface rather than just these curves, in effect filling in the rectangular holes in the surface. There are a few reasons that I tend to avoid this. First, this uses more material, and generally this means that the object will be more expensive to produce. Second, the pattern can provide a better visual sense of the shape of the object than a blank, featureless surface does, and it also communicates mathematical content in terms of the parametrisation. Third, it can be useful to be able to see partially through the surface to other features behind it.

Mathematica Code

Here is the code to generate the graphical output shown in Figure 7a, together with the STL file.

- Line 1 provides the parametric function $f : [-1,1]^2 \to \mathbb{R}^3$ that defines the parametric surface patch. This, of course, can be altered to produce any desired parametric surface.
- Line 2 defines a scaling factor, by which the geometry is scaled up in the final model. Our units will be in millimetres, and so the x-coordinate bounds of our model will be approximately -1×40 to 1×40, which is 80 millimetres, or 8 centimetres. The same is true for the y-coordinate bounds.
- Line 3 gives the radius of the tubular neighbourhoods around the curves, again in millimetres. Depending on the choice of material and printer, a diameter of 1.5 mm for a cylindrical "stick-like" part of a model seems to be safe, as long as it is well supported (i.e., not a long prong). The absolute minimum thickness for a part is around 1 mm, and if I wanted to be extra safe I would go to 2 mm in diameter.
- The generated mesh of triangles is an approximation to a tubular neighbourhood of the curves, with the cross-section a regular polygon. The variable in Line 4 specifies the number of sides this polygon has.
- Line 5 gives the number of squares in each direction on the parameterised grid.

```
1  f[u_, v_] := {u, v, u^2 - v^2};
2  scale = 40;
3  radius = 0.75;
4  numPoints = 24;
5  gridSteps = 10;
6  curvesU = Table[scale*f[u, i], {i, -1, 1, 2/gridSteps}];
7  curvesV = Table[scale*f[j, v], {j, -1, 1, 2/gridSteps}];
8  tubesU = ParametricPlot3D[curvesU, {u, -1, 1}, PlotStyle -> Tube[
       radius, PlotPoints -> numPoints], PlotRange -> All];
9  tubesV = ParametricPlot3D[curvesV, {v, -1, 1}, PlotStyle -> Tube[
       radius, PlotPoints -> numPoints], PlotRange -> All];
10 corners = Graphics3D[Table[Sphere[scale f[i, j], radius], {i, -1, 1,
       2}, {j, -1, 1, 2}], PlotPoints -> numPoints];
11 output = Show[tubesU, tubesV, corners]
12 Export["MathematicaParametricSurface.stl", output]
```



(a) The "Tube" PlotStyle directive produces closed polygonal meshes. **(b)** At a corner, these leave an ugly gap. **(c)** This can be fixed by adding a sphere at the corner.

Figure 8. Adding spheres at the corners of the parametric patch.

- Lines 6 and 7 produce lists of one-parameter functions of *u* and *v*, respectively, which give the "grid line" curves in the two directions.
- Lines 8 and 9 produce the graphics data for the tubular neighbourhoods of the curve functions from lines 6 and 7. The **PlotStyle** directive tells Mathematica to represent the curves as tubes with the given radius, the **PlotPoints** directive tells it the number of sides that the tubes are to have, and the **PlotRange** directive ensures that it doesn't crop out any parts of the geometry.
- The tubes that these lines generate are embedded, closed polygonal meshes, see Figure 8a. These meshes intersect each other at grid intersections, so the union of the tubes is not embedded.[5] At corners of the grid, our tubes leave an ugly gap as in Figure 8b, because each curve is thickened into a tube only along the perpendiculars to the direction of the curve. A true regular neighbourhood of the curve would also include a hemisphere around each endpoint of the curve. We fix this by adding in our own sphere, as in Figure 8c, which is what line 10 in the code does.
- Finally, line 11 shows all three sets of 3D graphics objects together in the same space, and line 12 exports the mesh corresponding to those objects to our STL file.

Using a 3D Printing Service

The STL file we generated in the previous section is ready to upload to a 3D printing service as is. There are a number of 3D printing services with international reach, and there are many other local companies. One either uploads the file to the company's website or emails the file to them. At some point, you will need to specify the units in which the file is measured, which is "millimetres" in our example, to be consistent with our calculations in the Mathematica code.

Some of the varieties of 3D printing technology are better suited to mathematical models than others. In particular, many involve depositing two kinds of material: the material that is to be the actual model, and a scaffolding material that is broken off by hand after printing. This doesn't work so well with an intricate model with a complicated internal structure. Other versions of the technology use a dissolvable scaffolding material, or they work by "solidifying" only the top layer of a tank of dust, so the dust in lower layers supports the object as it is built. Even given these restrictions, there is a large variety of available materials, ranging from plastics to metals, glass, and ceramics. Note however that different materials have different requirements in terms of the minimum safe thickness of walls, tubes, and so on. The 3D printing service one uses should have information available on its website or on request to help choose an appropriate technology and

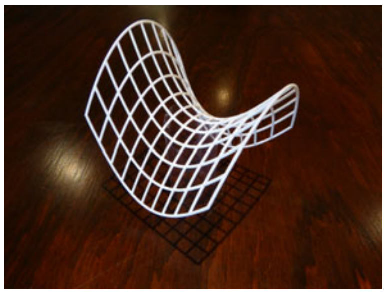


Figure 9. The resulting 3D printed object.

[5]If you use a 3D printing service, they should be able to fix this problem automatically on upload. If you are using your own 3D printer, then you may have to do some more work, taking the Boolean union of the tubes.

material in which to print. I usually use nylon plastic, which is 3D printed using a process called "selective-laser-sintering" (often referred to as "SLS"). The material is inexpensive, very strong, and is flexible rather than brittle. At the time of writing, the model generated in this article, as shown in Figure 9, cost me less than US$10 to produce using this material.

Going Further

There is an enormous number of interesting and beautiful mathematical models that can be generated with Mathematica alone. For further inspiration, George Hart has also written about the use of Mathematica to produce geometry suitable for 3D printing [10]. However, for more complicated models, it may make more sense to use a 3D design program, such as Rhinoceros, Blender, or one of the others listed previously. These programs allow a large amount of control over the 3-dimensional geometry, and they act in some ways like 3-dimensional versions of 2-dimensional vector graphics programs, such as Adobe Illustrator, Inkscape, or Xfig. One can move individual parts of a model around, copy and paste, and transform objects in various ways. The equivalent of the Mathematica "PlotStyle–>Tube" directive lets us build tubes around given curves, and one can build complicated mesh surfaces by joining together parameterised patches. For more details, see http://www.segerman.org/3d_printing_notes.html. Good luck!

REFERENCES

[1] Robert McNeel & Associates, *Rhinoceros*, http://www.rhino3d.com/.

[2] Autodesk, *3ds max*, http://usa.autodesk.com/3ds-max/.

[3] Autodesk, *Maya*, http://usa.autodesk.com/maya/.

[4] Vladimir Bulatov, *Bulatov abstract creations*, http://bulatov.org/.

[5] ISTI CNR, *MeshLab*, http://meshlab.sourceforge.net/.

[6] Blender Foundation, *Blender*, http://www.blender.org/.

[7] Bathsheba Grossman, *Bathsheba sculpture*, http://www.bathsheba.com.

[8] George W. Hart, *The (10, 3)-a network*, http://www.georgehart.com/rp/10-3.html.

[9] George W. Hart, *Creating a mathematical museum on your desk*, Mathematical Intelligencer **27** (2005), no. 4, 14–17.

[10] George W. Hart, *Procedural generation of sculptural forms*, Proceedings of the Bridges Conference, also available at http://www.georgehart.com/ProceduralGeneration/Bridges08-Hart10pages.pdf, 2008.

[11] Jesse Louis-Rosenberg and Jessica Rosenkrantz, *Nervous system*, http://n-e-r-v-o-u-s.com/.

[12] Burkard Polster and Marty Ross, *Print your own socks*, Column in *The Age*, also available at http://education.theage.com.au/cmspage.php?intid=147&intversion=78, February 2011.

[13] Saul Schleimer and Henry Segerman, *Sculptures in S^3*, Proceedings of the Bridges Conference 2012; also available at http://arxiv.org/abs/1204.4952.

[14] Carlo Séquin, http://www.cs.berkeley.edu/~sequin/.

[15] Wolfram Research, Inc., *Mathematica*, http://www.wolfram.com/mathematica/.