`wireframe`; a simple program for drawing surfaces

Danny Calegari

February 22, 2013

Contents

1 Summary

This manual describes the X-windows program `wireframe` and the format of the data files it takes. This program can be used to easily create .eps figures of rendered three-dimensional surfaces for inclusion in scientific articles.

Figure 1: A high genus surface with thin necks in `shaded` mode

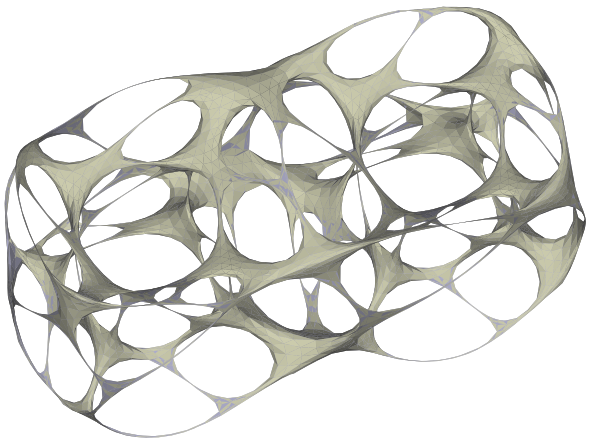
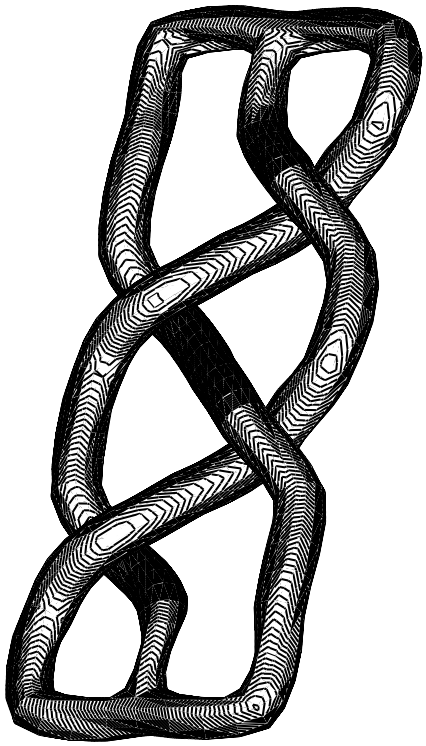


Figure 1 gives an example of a high genus surface in `shaded` mode. Figure 2 gives an example of a closed braided surface in `isobar` mode. Figure 3 gives an example of a surface with several boundary components in `shaded` mode.

Figure 2: A braided surface in `isobar` mode



2 Installing the program

The program can be built on your local machine by following these instructions:

1. Download the program files as a zip archive from github, from http://github.com/dannycalegari/wireframe.

2. Unzip the archive on your local computer; this should create a new folder containing the source files, some examples, the source for this manual (as a LaTeX file), and a makefile.

3. The makefile assumes the user has the GNU c compiler (`g++`) installed, and that the standard X-windows include files and libraries are installed in the directories `/usr/X11R6/include` and `/usr/X11R6/lib`. Edit these values if necessary.

4. Make the program from the command line with the command `make`.

3 Data formats

The program is run from the command line with the command `./wireframe -w filename` or `./wireframe -g filename`. The two options are explained below.

3.1 Reading a file in wire format

A data file called `filename` in *wire format* may be read in with the command `./wireframe -w filename`. This data file defines a triangular mesh (possibly with boundary, possibly disconnected) by specifying adjacency data and location of vertices. Here is an example of a file in wire format:

```
8
3                3 2 1          1.0 -1.0 -1.0
5                2 4 6 3 0      -1.0 -1.0 -1.0
5                3 5 4 1 0      1.0 -1.0 1.0
5                1 6 5 2 0      1.0 1.0 -1.0
5                2 5 7 6 1      -1.0 -1.0 1.0
5                3 6 7 4 2      1.0 1.0 1.0
5                4 7 5 3 1      -1.0 1.0 -1.0
3                6 4 5          -1.0 1.0 1.0
1.0 0.0 0.0
1.0 1.0 0.0
```

The format is defined abstractly as follows:

1. The first line of the file is a positive integer v which specifies the number of vertices.
2. Vertices are labeled from 0 to $(v-1)$.
3. The subsequent v lines are the data associated to vertices, in order from 0 to $(v-1)$. So the second line of the file is the data associated to vertex 0, and the $(v+1)$st line is the data associated to vertex $(v-1)$.
4. The data associated to vertex i is a triple v_i, L_i, (x_i, y_i, z_i).
5. v_i is a positive integer, which is the valence of vertex i, or the valence plus 1 if vertex i is a boundary vertex.
6. L_i is a list of v_i distinct integers. At most one entry in L_i can be -1; all other entries must be non-negative, must be between 0 and $v-1$, and must not be equal to i. Only the cyclic order of the list is important. The cyclic order of the entries corresponds to the neighbors of vertex i in anticlockwise order. A -1 entry indicates the outward normal to the boundary, so if $\cdots j, -1, k \cdots$ appears in L_i in this cyclic order, where j and k are non-negative, then vertices j, i, k are consecutive on the boundary of the mesh (with positive orientation).

7. (x_i, y_i, z_i) are real numbers, and denote the location in space of vertex i. In practice these should have absolute value at most 1.5.

8. The last two lines are the RGB color values for the outer and inner boundaries respectively.

This data should satisfy certain consistency checks. For instance, if vertex i lists vertex j as a neighbor, then vertex j should list vertex i as a neighbor. Also, the mesh should consist of triangles. If the file fails these consistency checks, the program will exit with an error.

3.2 Reading a file in graph format

A data file called `filename` in *graph format* may be read in with the command `./wireframe -g filename`. This data file defines a graph by specifying adjacency data and location of vertices. The syntax of graph format is identical to wire format *except* that a file in graph format has two additional lines coming before the RGB color values for inner and outer boundaries, each consisting of a single real number e, i (in order).

Here is an example of a file in graph format:

```
5
2                1 4         -1.0 0.0 0.0
2                2 0         -0.5 -0.5 0.0
3                1 3 4       0.0 0.0 0.0
2                -1 2        0.5 0.0 0.0
2                2 0         -0.5 0.5 0.0
0.4
0.2
0.8 0.9 0.9
1.0 1.0 0.0
```

A data file in graph format specifies a graph with straight edges. As with wire format, the first line of the file specifies the number of vertices, and subsequent lines specify, for each vertex in turn, its neighbors (taken in cyclic order) and its location. So as in wire format, the data associated to vertex i is a triple v_i, L_i, (x_i, y_i, z_i). These data must satisfy the following conditions:

1. v_i must be at least 2.

2. L_i is a list of v_i distinct integers. If one entry of L_i is equal to -1, then $v_i = 2$ and the other entry must be non-negative. Otherwise every entry of L_i must be non-negative.

The program reads in the graph, and computes from it a triangular mesh (possibly with boundary), obtained as the boundary of a tubular neighborhood of the graph. The number i is the thickness of the neighborhood near internal vertices, and e is the thickness of

"boundary" vertices. A -1 entry in a list L_i (necessarily of length 2) denotes a "boundary" vertex, which will correspond to a boundary loop in the associated triangular mesh.

The way the surface is obtained from the graph implicitly assumes that the edges are bounded away from vertical, and the cyclic order on edges at each vertex agrees with the cyclic order of their projection to the x–y plane.

4 Program interface

Running the program on a file in consistent wire or graph format will open an X-window and display the mesh, initially in wire frame outline. The x–y coordinates of the mesh and the screen agree; the z coordinate points out of the screen towards the viewer.

The following keypress options are available.

- [f] toggles between solid and wireframe. Solid can be in either shaded or isobar mode.
- [v] toggles between verbose and silent. Verbose is mainly useful for debugging.
- [i] toggles between `shaded` and `isobar` mode. `Shaded` mode paints each triangle in exterior or interior color (depending on its orientation) with an intensity proportional to the vertical component of the normal. `Isobar` mode is black and white, and draws the level sets of the z coordinate which are integer multiples of some specified value.
- [1/2] makes isobars sparser/denser. The default spacing is 0.01. Each keypress multiplies or divides the spacing by 1.1.
- [s] subdivides and smooths mesh. The subdivision adds one new vertex for each edge. The new vertex is displaced in the normal direction to make the new surface "rounder".
- [r] retriangulates to shorten edges. A quadrilateral can be triangulated in 2 ways by adding a diagonal; a 2–2 move toggles between the two different triangulations. Every non-boundary edge in a triangulation is the diagonal of some quadrilateral. This operation performs 2–2 moves which definitely shorten the total length of the 1-skeleton. Note that this changes the "combinatorial conformal structure" of the triangulation, and in combination with the curvature flow operation will tend to produce surfaces with thin necks.
- [c] flows the surface by curvature. This is a very crude implementation of (combinatorial) mean curvature flow which tries to equalize edge lengths. Actually, it is more of an energy flow, since it does not retriangulate the surface. Can be used effectively in conjunction with the retriangulation operation.
- [t] trim pyramids. A "pyramid" is a 3-valent vertex. Such vertices tend to give rise to strange artifacts when smoothed or flowed; this command trims them off.

- [arrow keys] to rotate mesh. Each keypress rotates by 0.1 radians in the x–z plane or the y–z plane.
- [+/−] to make mesh bigger/smaller. Each keypress multiplies or divides the scale by 1.05.
- [e] for .eps output. The user will be prompted for a file name to write .eps output to.
- [w] for .wire output. The user will be prompted for a file name to write .wire output to.
- [p] for .pov output. The user will be prompted for a file name to write .pov output to, which can then be imported into povray and rendered.
- [q] to quit.

5 Acknowledgments

Danny Calegari is partially supported by NSF grant DMS 1005246. The program `wireframe` is released under the terms of the GNU GPL License version 3.0.

Figure 3: A surface with boundary in `shaded` mode