`kleinian`; a GLUT program for visualizing Kleinian groups

Danny Calegari

March 4, 2014

Contents

1 Summary

This manual describes the GLUT program `kleinian` and the format of the data files it takes. This program uses the GLUT interface to display hyperbolic polyhedra (in the Poincaré ball model) which are invariant under Kleinian groups.

Figure 1 gives an example of a domain invariant under a rank 3 Schottky group. The quotient is a genus 3 handlebody with boundary decomposed into totally geodesic subsurfaces which meet at right angles.

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/kleinian`.

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 LaTeXfile), and a makefile.

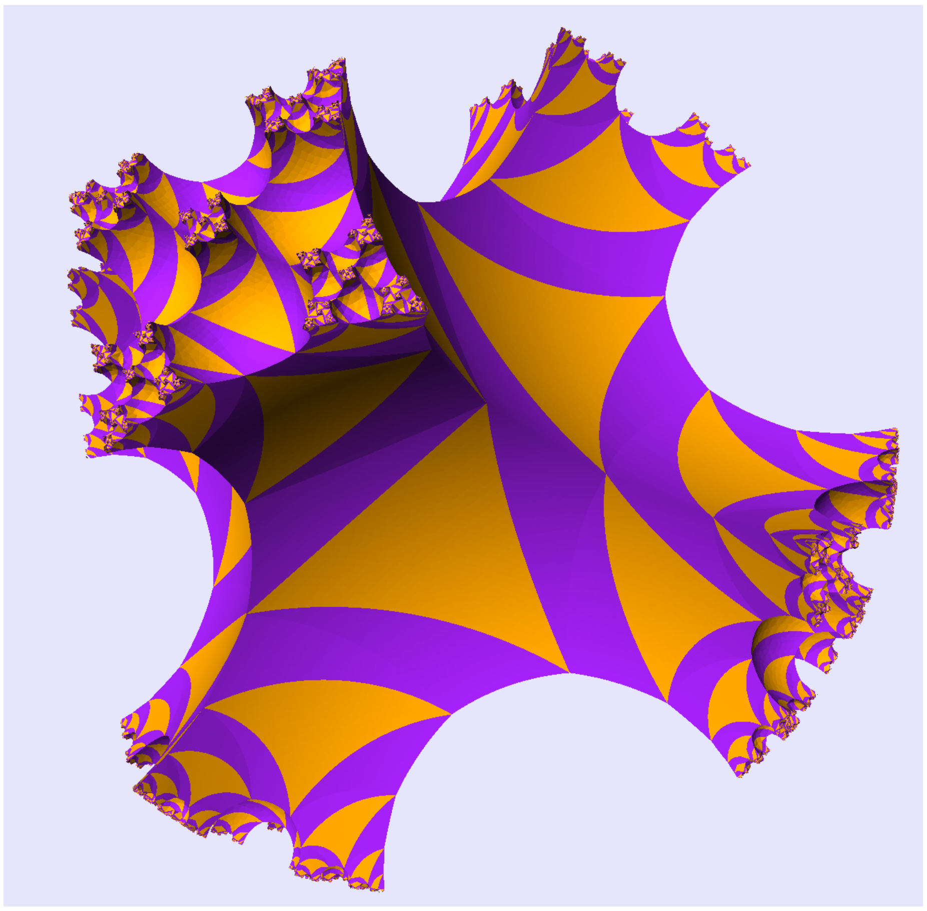

Figure 1: A domain invariant under a rank 3 Schottky group

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`, and the GLUT framework is available. Edit these values if necessary.

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

The program has been built successfully on Macintosh computers running OS X. It has also been successfully built on an Ubuntu Linux distribution; for this, the following modifications were necessary:

1. changed `-fast` to `-Ofast` and added linker options `-lglut -lGLU -lGL` in makefile.

2. installed package `freeglut3-dev`.

3 Data formats

The program is run from the command line with one of the following commands:

1. `./kleinian -t` *filename.tri* to read a precomputed triangle mesh from a file
2. `./kleinian -g` *filename.grp* to read semigroup generators and triangle orbits from a file
3. `./kleinian -e` to run one of three hardcoded examples.

These options are explained below.

3.1 Reading a precomputed triangle mesh

A data file called *filename.tri* in *triangle format* may be read with the command `./kleinian -t` *filename.tri*. This data file defines a triangular mesh, which is displayed with GLUT. Note that these triangles are Euclidean triangles; i.e. the result of subdividing Poincaré triangles to the mesh size to make the result appear curvilinear. Thus, if the mesh size is small, or if there were many Poincaré triangles in the original mesh, these files can be very large.

Triangle format is very simple:

1. The first line is the number N of triangles.
2. The next $5N$ lines each have three numbers on them. In each group of 5 lines, the first 3 lines are the xyz coordinates of the triangle vertex, the next line are the xyz coordinates of the (unit) normal to the triangle, and the last line is the rgb color of the triangle.
3. The last line is the rgb background color of the scene.

Here is a simple commented example:

```
60                                                60 triangles
-0.904988        0.000000        0.000000        triangle 0 vertex 0
-0.719152        0.549384        0.000000        triangle 0 vertex 1
-0.719152        0.169769        0.522495        triangle 0 vertex 2
-0.922602        0.312081        0.22674         normal to triangle 0
0.913725         0.588235        0.47843         rgb color of triangle 0
...
0.9              0.9             0.99            rgb background color
```

Producing a data file in triangle format by hand is probably quite laborious; note that all the program does with such a file is to display it using GLUT primitives, and allow the user to rotate the result. Thus there is no requirement that the mesh have anything to do with Kleinian groups or hyperbolic geometry, and this feature of the program could be used to visualize triangular meshes of any origin.

When `kleinian` is run in some other mode, the resulting triangular mesh (which can take a long time to compute if it is very complicated or has very small mesh size) can be saved as a file in triangle format, which can then be loaded and displayed quickly. This is the main function of this mode and format.

3.2 Reading semigroup generators and triangle orbits

A data file called *filename.grp* in *group format* may be read with the command `./kleinian -g` *filename.grp*. This data file defines a finite list of semigroup generators, and a finite number of (Poincaré) triangle orbits. The program will read the generators, generate the Cayley graph out to some depth, translate the triangle orbits about by list of generated matrix elements, project the Poincaré triangles to curvilinear Euclidean triangles in the Poincaré ball model, and approximate the result by a mesh of small Euclidean triangles which are then rendered in GLUT mode.

Group format has the following structure:

1. The first line is the number N of semigroup generators.
2. The second line is either the character `n` (for *numerical* format) or `k` (for *algebraic* format).
    - In numerical format, the next $4N$ lines are the matrix entries of the semigroup generators; these are matrices in $\mathrm{O}(3,1)$, and each of the four lines corresponding to a generators is a row of the matrix.
    - In algebraic format, for each of the N generators $G(i)$, there is an integer $m(i)$ (the number of *elementary* generators whose product is the $G(i)$), and then for $j = 0$ to $m(i) - 1$, a triple of the form a_j, b_j, l_j where a_j and b_j are distinct integers between 0 and 3, and l_j is a real number. Then

      $$G(i) = M(a_0, b_0, l_0) * M(a_1, b_1, l_1) * \cdots * M(a_{m(i)-1}, b_{m(i)-1}, l_{m(i)-1})$$

      where $M(a, b, l)$ is a rotation through angle l in the Euclidean a–b plane (when $a, b < 3$) and translation through hyperbolic distance l along the a axis (when $a < 3$ and $b = 3$).
3. The next line is the number M of triangle orbits.
4. For each of the next $3M$ lines, each triple of lines are the coordinates of the vertices of the given triangle orbit. These are *Poincaré* triangles, whose vertices are in Lorentz 4-space. Thus, each vertex coordinate is a 4-tuple x, y, z, w satisfying $w^2 - x^2 - y^2 - z^2 = 1$.
5. If the next line is the character `c` then the next M lines are the rgb colors of the triangle orbits. Otherwise the color of all triangles defaults to white.
6. If colors were specified, then if the next line is the character `b` the next line is the rgb background color. Otherwise the background color defaults to light blue.

Here is an example in numerical format:

```
4                                                                 4 generators
n                                                                 numerical format
3.7621957       0.0000000       0.0000000       3.6268604        generator 0
0.0000000       1.0000000       0.0000000       0.0000000
0.0000000       0.0000000       1.0000000       0.0000000
3.6268604       0.0000000       0.0000000       3.7621957
...
1                                                                 1 triangle orbit
1.81343         1.1752012       0.00000         2.3810978        triangle vertex 0
1.81343         0.00000         1.1752012       2.3810978        triangle vertex 1
1.1752012       1.81343         0.00000         2.3810978        triangle vertex 2
c                                                                 triangle color
0.62745         0.12549         0.94117647                        rgb value
b                                                                 background color
0.9             0.9             0.99                              rgb value
```

Here is an example in algebraic format:

```
4                                                                 4 generators
k                                                                 algebraic format
2                                                                 generator 0 has factors
0               3               0.54930614                        translate x axis 0.549
0               2               -2.0943951                        rotate xz-plane -2.094
...
1                                                                 1 triangle orbit
1.81343         1.1752012       0.00000         2.3810978        triangle vertex 0
1.81343         0.00000         1.1752012       2.3810978        triangle vertex 1
1.1752012       1.81343         0.00000         2.3810978        triangle vertex 2
c                                                                 triangle color
0.62745         0.12549         0.94117647                        rgb value
b                                                                 background color
0.9             0.9             0.99                              rgb value
```

4 Program interface

When the program reads a file in triangle format, it loads the mesh into GLUT, and moves into GLUT mode.

When the program reads a file in group format, it begins a dialog with the user to determine parameters.

1. First, the user is prompted to enter the depth to generate the Cayley graph to. This should be a non-negative integer. Unless the group is elementary (e.g. a parabolic or finite group) or the fundamental domain is very small (e.g. a $2, 3, 7$ triangle group), this should probably be 12 or less, unless very fine detail is needed.

2. After computing the Cayley graph out to the specified depth, the user is asked whether they want fancy curvilinear triangles. The answer should be a single character, either `y` or `n`.

3. An `n` answer will make the triangles Euclidean, but their vertices will be in the correct location for the Poincaré model. This is a reasonable approximation when the triangles have small hyperbolic diameter, but will look terrible if some triangles are (almost) ideal.

4. A `y` answer will prompt the user to supply a mesh size to subdivide the triangles to. This should be a floating point number, generally between 0.02 and 0.2. If the mesh is too small, and some triangles are too big, the program might run into numerical difficulty and crash.

5. The program then computes the subdivided triangle mesh, and enters GLUT mode.

When the program is run in example mode, the user can select one of three pre-computed examples to run, and for each they might first be prompted for some parameters of the example, before starting the dialog above.

4.1 GLUT mode

In GLUT mode, the following keypress options are available:

- [arrow keys] rotate the display
- [t] saves the triangulation to *output_file.tri*
- [g] saves the generators (e.g. in example mode) to *output_file.grp*
- [e] saves the figure as an .eps file *output.eps*
- [command-q] exits the program.

5 Acknowledgments

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

Version 0.01 of `kleinian` was released February 28, 2014 and is copyright Danny Calegari.

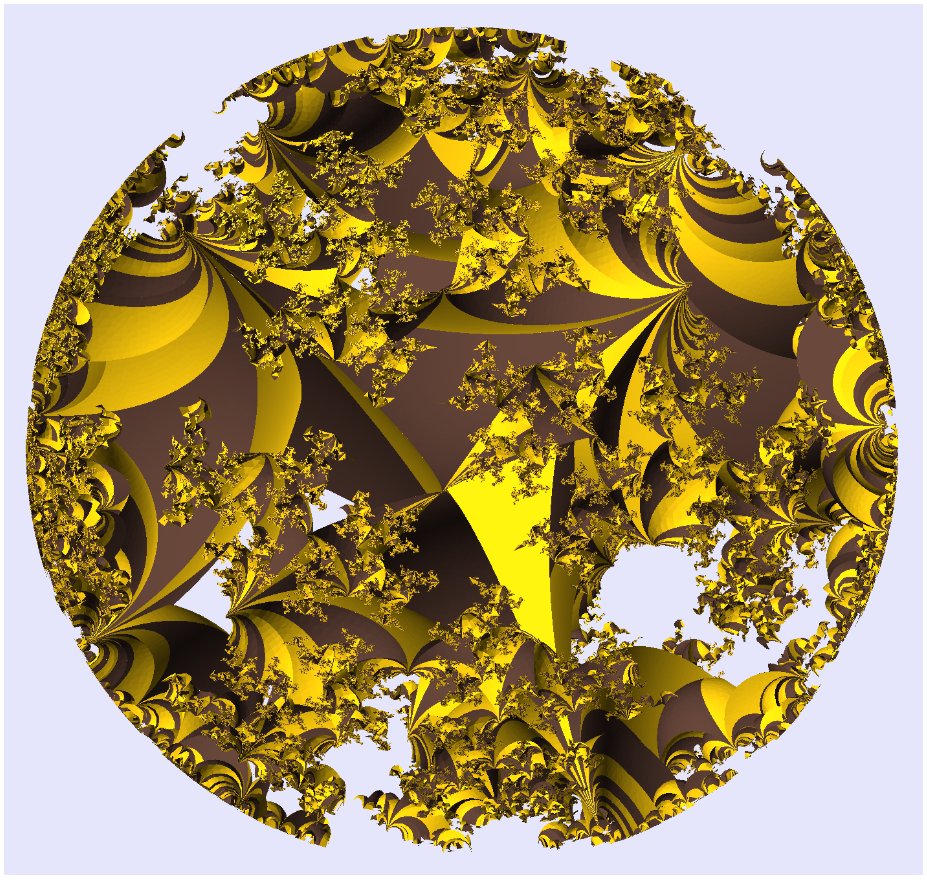

Figure 2: Figure 8 knot fiber