PGVisualizer User Manual

v.1.2

Nico Van Cleemput
nico.vancleemput@gmail.com

March 26, 2009

Contents

1 Introduction and terminology **1**

2 Basic working **2**

3 Walkthrough **2**

4 User interface **5**
4.1 Export 5
4.2 Edit 7
4.3 View 8
4.4 Embedder 9
4.5 Filter 10

5 History **10**

6 To Do **11**

A The PG file format **11**

1 Introduction and terminology

The PGVisualizer is a program to visualize and edit periodic graphs saved in the PG format. The PG format can save face information, but this is not mandatory. Some of the functionality for faces will be obsolete in this case.

This program and the PG format were both designed for my research on toroidal azulenoids, and, although I tried to be as general as possible, sometimes there are explicit references to this present in the user interface.

A periodic graph $PG(V, E)$ is represented in the PG format as follows. The plane is divided into equal parallelograms by two sets of equidistant parallel lines. One set is horizontal, the other set is not parallel with the first set. Such

a parallelogram is called the fundamental domain of the periodic graph. The set of vertices is embedded in the fundamental domain, thusly creating an infinite graph. The edge are described from an arbitrary fundamental domain which we give the coordinates $(0, 0)$. For each vertex the set of edges that start from that vertex is described by given the target vertex and the target fundamental domain.

The PG format gives the coordinates of the vertices in the square with corner points $(1, 1)$, $(1, -1)$, $(-1, -1)$ and $(-1, 1)$ and those coordinates are then transformed to coordinates in the parallelogram-shaped fundamental domain.

2 Basic working

When you start the PGVisualizer by running the file PG.jar or the file PGVisualizer.exe, you will be prompted for a file. Select your PG file and click open. The main window will open and the first graph in the file will be visualized. You can navigate through the file by using the toolbar at the top of the window.

While reading the PG file, PGVisualizer stores each line (i.e. each graph) in a list of strings. When it has to visualize a certain graph, it first checks whether that string is already parsed. If this is the case, it shows that graph, otherwise it parses that string to a structure which serves as a model for the visualization. You can edit that model by interacting with it through the user interface. At any point, you can revert all your changes, which will remove the structure and reparse the string in the list. You can also 'commit' the current state of the structure. This will convert the current state to a string and store this in the list. This will however not change the file on disk. Finally you can also save all the changes. This will commit all the changes that aren't already committed and then save the list of strings to a file.

It is also possible to filter the list. This filtered list then contains a subset of the original list. These is a real subset and aren't copies. Any changes made to these graphs are also made to the graphs in the original list.

Any change that is made to the structure of the graph is directly visible in all the views of that graph. This means that changes made to a graph in a filtered list or in the editor are in real-time visible in all the lists.

3 Walkthrough

This is a very short walkthrough that will reflect the type of use I expect will be most common. For a detailed working of all the functions I refer to 4.

When you open a file with PGVisualizer you get a window that looks something like Figure 1.

You can navigate through the list by using the navigation bar at the top. The outer two buttons will skip to the begin and end of the list. The buttons with the single arrows skip one graph ahead or back. When you want to jump to a specific graph directly you can type in its number and press the goto-button.

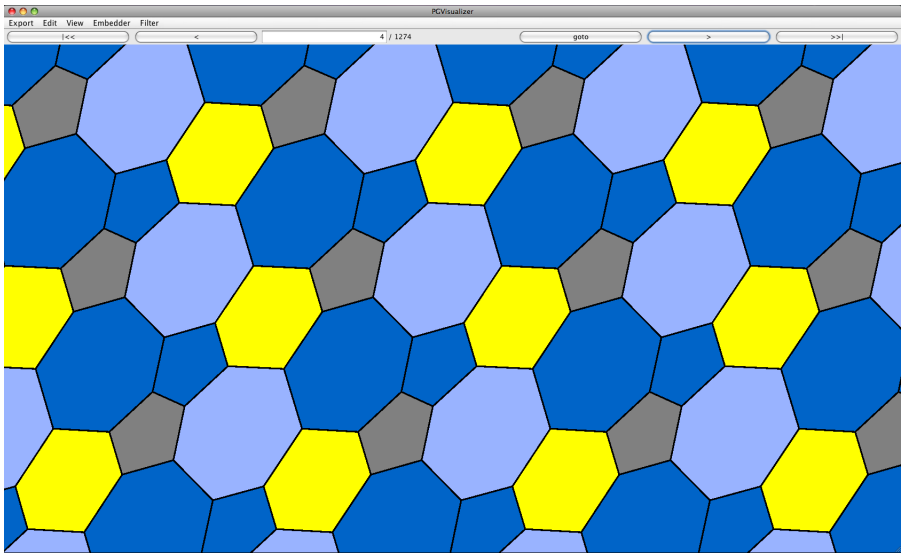


Figure 1: The main window of PGVisualizer

It is also possible to view this periodic graph as a finite structure. Select the menu item *Finite structure* from the *View* menu. This opens the dialog window shown in Figure 2.

At the bottom of this dialog window you see the periodic graph without the faces. The red lines are the fundamental domains. In the first column at the top you can fill in the number of horizontal and vertical copies you want to make of the fundamental domain. If you press the preview you can view the larger fundamental domain that will be used for the finite structure. In the second column you can give the offset used to connect the original fundamental domains that are at the edge of the larger fundamental domain, and with the overflow check box you can let these connections overflow. It is advised that you let PGVisualizer calculate the optimal shift by pressing the optimal shift button. In the dropdown box you can select which finite structure you want to see.

For instance we take 10 horizontal copies and 6 vertical copies with the optimal shift and overflow. We use this larger fundamental domain to form a torus with the X-axis as major circle. As a tiled structure this looks like Figure 3. The library jReality is used for these visualizations.

As a molecule this looks like Figure 4. For these molecular views the library Jmol is used. If the periodic graph is viewed as a molecule PGVisualizer attempts to also show the face highlighting. For the azulenoids this should always work. If you introduce extra colourings this may create conflicts when two neighbouring faces have different colours. This colouring can be turned on through the menu *PG Colorings*. The other menus offer access to the different

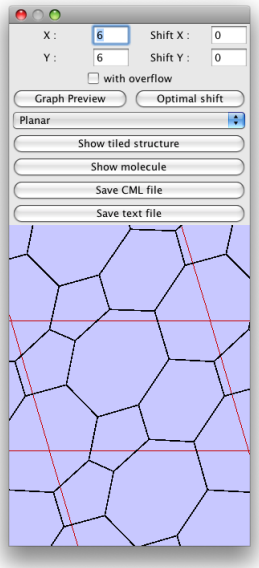


Figure 2: The *Finite structure* dialog window

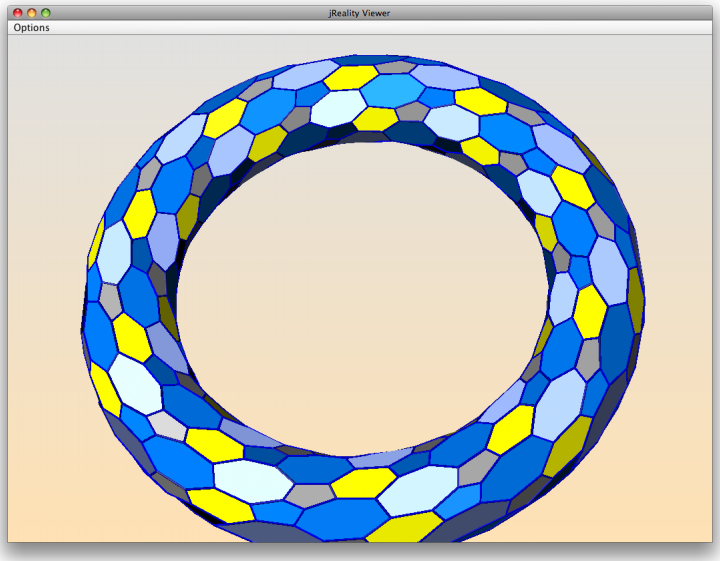


Figure 3: A periodic graph viewed as a tiled structure

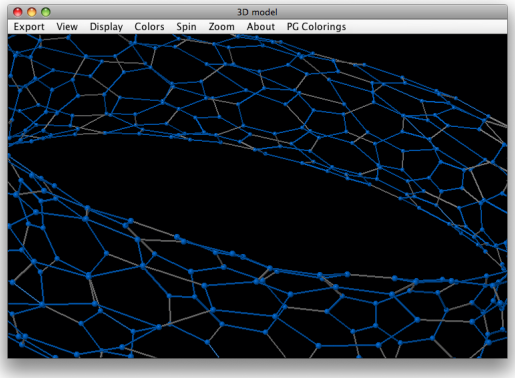


Figure 4: A periodic graph viewed as a molecule

functionalities of Jmol and we hope that they are quite self-explanatory. In case you want support for other features of Jmol, feel free to contact us.

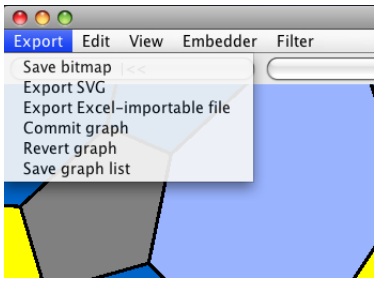
You can also filter the original list. This functionality is available through the *Filter* menu. The *Filter* dialog window is shown in Figure 5.

You simply construct the filter using the dropdown boxes or checkboxes and add them to the list by pressing add. If, for instance, you want all the periodic graphs with only one azulenoid per fundamental domain you could use a vertex filter. An azulenoid contains 10 vertices, so adding the filter "exact 10 vertices" is sufficient. Next you press the filter button and the filtered list opens up in a new PGVisualizer window.

4 User interface

The main window has five menus. Below you will find a detailed description of each menu.

4.1 Export



Save bitmap export a PNG image of the currently selected graph.

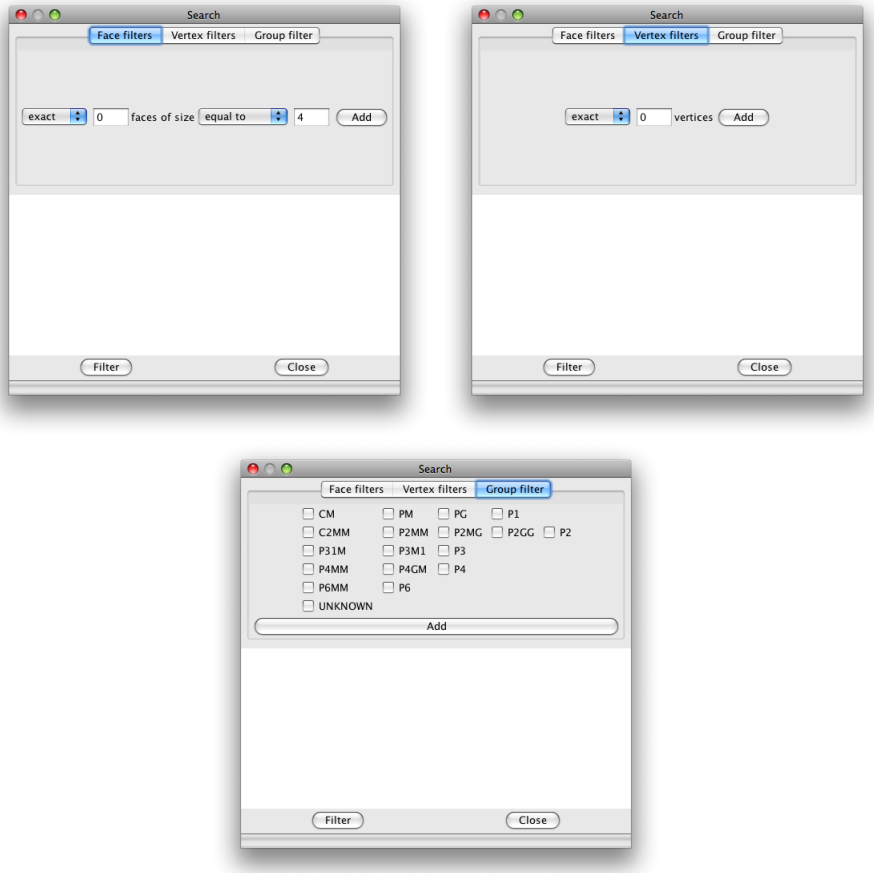


Figure 5: The *Filter* dialog window: on the left hand side the Faces tab, on the right hand side the Vertex tab and at the bottom the Group tab.

Export SVG export the currently selected graph to a SVG file[1].

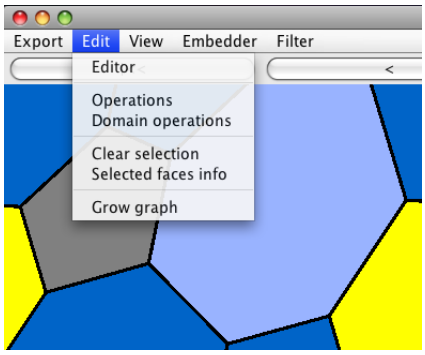
Export Excel-importable file exports a tab-separated text file with the extension xls. This file can be opened in Excel and will then be converted to an Excel file with a short wizard. The records in this file are separated with tabs and strings are denoted by ".

Commit graph commit the changes to the list of strings.

Revert graph revert to the last commited version of the graph.

Save graph list save the list of graph to a file you specify.

4.2 Edit



Editor opens the periodic graph editor with the currently selected graph. You can edit the graph by selecting vertices and/or faces and moving them by dragging. Multiple selection is realized by holding down shift while selecting the vertices or faces. You can drag them outside the fundamental domain and they will reappear on the other side.

Operations opens a dialog window from which you can perform several operations on the graph. The upper four button represent four operations you can perform on the vertices in the original square fundamental domain (see 1). The other buttons allow you to move the complete graph in the direction corresponding with the button.

Domain operations opens a dialog window form which you can perform several operations on the fundamental domain. You can change the upper left angle and the horizontal side of the parallelogram.

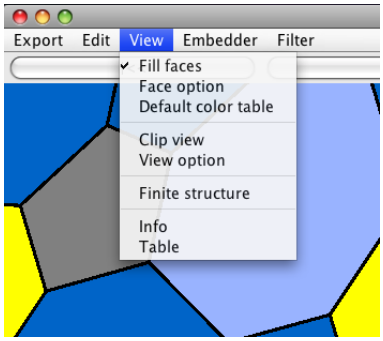
Clear selection clears the current selection.

Selected faces info gives an overview of the size and colour of the selected faces.

[1]Scalable Vector Graphics is an XML format for vector graphics. A good editor for SVG is e.g. Inkscape (http://www.inkscape.org)

Grow graph opens a dialog window in which you can enlarge the graph by concatenating several fundamental domains.

4.3 View



Fill faces enable or disable the filing of the faces.

Face option opens a dialog window with some options for faces. In this dialog window you can change the transparency of the faces and change the colour of the selected faces.

Default color table shows an overview of the colours that PGVisualizer uses to fill faces that aren't highlighted. All faces of sizes not mentioned in this table are coloured dark gray.

Clip view toggles the clipping of the view. Standard the entire window is filled with the tiling, but it is also possible to clip this to a given number of fundamental domains.

View option opens a dialog window from which you can change the number of fundamental domains that are at least shown and are used when clipping the view, and from which you can change the size of the vertices (standard this is set to 0).

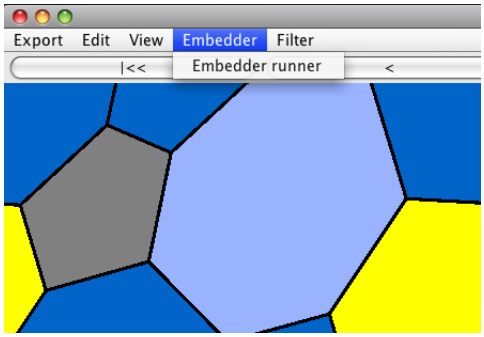
Finite structure opens the *Finite structure* dialog window. (See 3)

Info opens a dialog window with some info. This includes the number of vertices per fundamental domain (this is also the number of azulenoids per fundamental domain times 10), the catalogue number i.e. the sequence number in the file from which it was loaded — this is mostly useful when browsing a filtered list to retrieve the graph in the original list afterwards —, the wallpaper group of the original azulenoid tiling and the original Delany-Dress symbol if this was stored in the comment of the PG file (which is the case for the azulenoids).

Table opens a dialog window with an overview table of the file. This table contains the information from the info dialog, but shows an overview for

the complete file. It is also possible to select a row in this table and the viewer will jump to the corresponding periodic graph.

4.4 Embedder



This menu item opens a dialog window from which you can run several embedders on the currently selected graph. These embedders are

- Spring embedder

  Contracts or expands edges based on a constant length and has dampening.

- Spring embedder equal edges

  Contracts or expands edges based on the mean length of the edges.

- Spring embedder minimal equal edges

  Contracts or expands edges based on the length of the shortest edge.

- Spring embedder to zero

  Tries to contract all the edges to length 0.

- Spring embedder (contract faces)

  Same as 'Spring embedder' but all the vertices of a face are also pulled to the center.

- Spring embedder to zero (contract faces)

  Same as 'Spring embedder to zero' but all the vertices of a face are also pulled to the center.

- Random embedder

  Places all the vertices at a random position.

- Tutte embedder

  Fixes the position of all the vertices that have an edge that leaves the fundamental domain and places the other vertices in the center of gravity of their neighbours.

- Domain angle embedder using edge length

  Changes the angles of the fundamental domain while trying to optimize an energy function based on the deviation of the lengths of the edges from the mean length.

- Domain angle embedder using edge angles

  Changes the angles of the fundamental domain while trying to optimize an energy function based on the deviation of the angles around a vertex from the optimal angles.

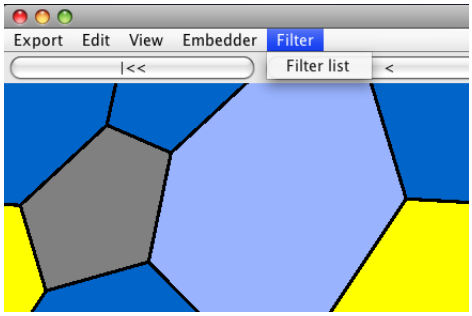
- Domain edge embedder using edge length

  Changes the length of the sides of the fundamental domain while trying to optimize an energy function based on the deviation of the lengths of the edges from the mean length.

- Domain edge embedder using edge angles

  Changes the length of the sides of the fundamental domain while trying to optimize an energy function based on the deviation of the angles around a vertex from the optimal angles.

4.5 Filter



This menu opens the filter dialog window. See 3 for information on this functionality.

5 History

20/08/2008 Version 1.0

08/12/2008 Version 1.0.1

- Disable Filter button during filtering

26/02/2009 Version 1.1

- Support `symbol` and `group` comment stream in PG format.

- Added filter based on wallpaper group of original tiling.
- Added table with overview of current list.

26/03/2009 Version 1.2

- Better support for Jmol.

6 To Do

Their is still a lot of room for improvement of the program. However I currently lack the time to make these changes. If you have ideas that are not mentioned below, or have the skills and the willingness to implement some of the features below, feel free to contact me.

- The 'finite structure' dialog deserves a slicker interface.
- Add support for composite embedder which you can construct at runtime by putting together other embedders.
- Make the default colourings editable.
- ...

A The PG file format

A PG-file has the extension pg and contains at most one periodic graph per line. A line can also consist entirely of a comment when it starts with a #. A single periodic graph looks like this:

$$\mathrm{s}_1 \mid \mathrm{s}_2 \mid \mathrm{s}_3 \mid \mathrm{s}_4 \; [\mid \mathrm{s}_5 \;] \; [\; \# \text{ info }] \; [\; \# \text{ info }] \ldots$$

It consists of four or five strings (the fifth is optional) separated by |'s (pipes). At the end there is the possibility to add additional information such as comments or face highlighting.

- s_1

  This string only contains one integer number. This is the order of the repetitive part of the graph.

- $\mathrm{s}_2 = \mathrm{s}_2^a \; \mathrm{s}_2^b \; [\; \mathrm{s}_2^c \;]$

  This string consists of two or three real numbers separated by spaces. The first is length of the horizontal side of the domain, the second is the length of the vertical side of the domain. If there is a third number present then this will be used as the upper left angle of the domain.

- $s_3 = x_1\ y_1;\ x_2\ y_2;\ x_3\ y_3;\ \ldots$

  This strings contains the coordinates of the vertices. It contains several parts separated by semicolons. The number of parts has to be equal to the order given in s_1. Each part contains two real numbers separated by a space.

- $s_4 = start_1\ end_1\ X_1\ Y_1;\ start_2\ end_2\ X_2\ Y_2;\ \ldots$

  This string contains an entry for each edge. The entries are separated by semicolons and each entry consists of four integer numbers separated by spaces. The first number gives the start vertex, the second number the end vertex and the third and fourth number are the X and Y coordinate of the domain to which the end vertex belongs.

- $s_5 = f_1;\ f_2;\ f_3;\ \ldots$

  This fifth string is optional and it contains information about the faces. It contains an entry for each face. The entries are separated by semicolons and each entry consists of a row of integer numbers separated by spaces. Each number gives the index of a vertex.

  $f_i = v_1^i;\ v_2^i;\ v_3^i;\ \ldots$

- When the first word in one of the additional info string is **facehighlight** then the rest of the string is interpreted as face highlighting information and the string should be formatted as follows

  $$\text{facehighlight}\ F_1\ c_{F_1}\ F_2\ c_{F_2}\ \ldots$$

  In this F_i stands for the sequence number of a face in the string s_5 and c_{F_i} stands for the colour of that face given as a integer in which bits 24-31 are alpha, 16-23 are red, 8-15 are green and 0-7 are blue, i.e. as defined by the function getRGB() of the class Color in Java 5.

- When the first word in one of the additional info string is **symbol** then the rest of the string is interpreted as a Delaney-Dress symbol in .ds-format.

- When the first word in one of the additional info string is **group** then the rest of the string is interpreted as the name of a wallpaper group. The possibilities are defined in the enum *be.ugent.caagt.pg.visualizer.groups.WallpaperGroup* and are P6MM, P6, P4MM, P4GM, P4, P31M, P3M1, P3, C2MM, P2MM, P2MG, P2GG, P2, CM, PM, PG, P1, UNKNOWN. Any other value will be interpreted as UNKNOWN.