polygone version 3 documentation

Ian Thompson
Department of Mathematical Sciences,
Loughborough University
i.thompson@lboro.ac.uk
www-staff.lboro.ac.uk/~mait/

December 2, 2011

1 Introduction

Polygone is a program that repairs certain defective postscript graphics files. It has nothing at all to do with a dead parrot[1]. Some applications that create postscript graphics shade regions using enormous numbers of very small polygons, when a few larger one would do instead. This leads to files that are so large as to be unusable for many purposes. Polygone parses postscript code looking for this defect, and merges polygons that share one or more common edges and are shaded in the same colour. Inexplicably, applications that generate oversized postscript files in this way often fail to draw the outlines of the polygons, leading to the appearance of an ugly, unwanted mesh overlaying the shaded areas. Polygone can fix this too, although you can easily do that yourself if you prefer (see appendix A).

Graphics saved in pdf can also suffer from these defects, but most software that creates pdf files can also create postscript, and the files generated by polygone can be converted to pdf using epstopdf or a similar utility.

2 New in version 3

- Polygone 3 has an improved pattern matching algorithm, and should be able to detect groups of coloured polygons in any 'reasonably' structured postscript code. In particular, any number of postscript operators (`closepath`, `fill`, `newpath`, etc. can be used to separate polygons (see the description of the `-s` option below).
- Polygone 3 has a much more intelligent algorithm for finding potential mergers. It is significantly faster than earlier versions.

[1] beautiful plumage, though.

- A bug which caused earlier versions to miss a small percentage of possible mergers has been eliminated.
- A bug which sometimes caused polygons in the processed postscript code to overlap slightly has been eliminated.
- A bug which caused long lines of postscript code to be truncated in the output has been eliminated.
- The test for colinear points is more accurate than in earlier versions.
- The structure of the code has been improved in a number of areas.

3 Compiling

Polygone is written in Fortran[2] 2003. At the time of writing, not all compilers support enough 2003 features to compile the source code. It has been successfully compiled under OS X 10.7.2 using the following software:

- NAG Fortran Compiler Release 5.3(836) (use the `-f2003` flag for a clean compile),
- Intel Fortran Compiler 12.1.0.038 Build 20110811.

Contact the vendor or your system administrator if your compiler rejects the code. The author cannot help you to resolve this issue.

4 Running

If polygone is executed with no arguments, it will run in interactive mode, issuing a small number of self-explanatory prompts. When it is run with arguments, it expects one of these to be the name of the postscript file to be processed. The original postscript file is always opened read-only. If processing is successful, a new file will be created, and its name will be displayed.

There are several options that can be selected in interactive mode, or passed in unix-style from the command line.

`-mj` (`j` a positive integer). Consider shapes with more than `j` vertices to be ineligible for merger. For example, running with `-m3` allows polygone to merge triangles into quadrilaterals, but it will not attempt to merge polygons with four or more vertices.

[2]The author is a mathematician, and doesn't know any better. Consider yourself lucky that it is not written in `FORTRAN`.

Repeatedly running earlier versions of polygone with successively increasing values of `j` led to improved performance in some circumstances, but this is no longer the case. The default setting is to impose no such restriction, and there are now few, if any, reasons to use a different setting.

`-r` remove mesh. When this option is used, polygone will try to define a new procedure that draws the outlines of shapes in the same colour used to fill the inside. The new file will use this to fill in shaded areas; any mesh that was visible in the old file should disappear. The default is to use the fill procedure provided by the old file. The process of merging polygons often causes the mesh to become invisible in any case.

`-sj` (`j` a positive integer). Require that a minimum of `j` postscript operators and/or procedures be used to separate consecutive polygons. The default value is `2`. In principle, a postscript code could define a procedure along the lines of

```
/sep { closepath fill newpath } bind def
```

and use this alone to separate polygons. In this case `-s1` is needed in order to enable polygone to correctly interpret the code. In practice this is rarely done, and setting `s=1` weakens the pattern matching algorithm somewhat. The default is correct in most cases.

`-v` verbose mode. This option causes polygone to display more information about its doings.

5 Troubleshooting

Polygone should never freeze or cause a memory error such as a segmentation fault, but there are a few things that can go wrong.

5.1 Display problems

Polygone uses an i/o trick to display counters, to show that slow-running loops have not frozen. This should work on all proper (i.e. unix) computers. If there is a problem with the display when running polygone, change the line which reads

```
logical , parameter :: ansi_terminal = .true.
```

to

```
logical , parameter :: ansi_terminal = .false.
```

and compile the code again.

5.2 Processing fails

Polygone can fail to process a file for a number of reasons. Processing will stop immediately if an error occurs, and any new postscript file created is likely to be unusable. Details of polygone's error messages are provided below.

- `iostat error IOS has occurred.` (`IOS` a positive integer.)

  A problem has occurred when reading the postscript code. The compiler documentation will explain the precise meaning for each possible value of `IOS`. This is most likely to indicate a fault in the postscript file; it is unlikely to be a bug in polygone.

- `collapsed path!`

  A serious problem has occurred while attempting to merge polygons. It is probable that something is wrong with the postscript file being processed, but this error may also indicate the presence of a bug in polygone itself.

- `unable to determine a safe pgf command.`

  When polygone is run with the `-r` option, it may need to define a new procedure that fills the current polygon and draws its outline in the same colour. Polygone can detect procedures that it inserted on an earlier run, and will reuse these provided their definition is not edited, or moved. If it needs to define a new procedure, polygone must first find a name that is not already in use. Initially, it will try `pgf`, and if this is already in use it will try `pgfx`, then `pgfxx`, etc. After `100` failed attempts, polygone will give up. Repeatedly processing with polygone and some other utility that meddles with postscript code could generate this error, but it should never occur under 'normal' circumstances.

- `no drawing operators detected.`

  The most likely cause of this is that the postscript file does not actually contain any groups of shaded polygons. In order to contain such groups, the file must be a true postscript graphic, i.e. a sequence of geometrical instructions. However, postscript code can also contain bitmaps (collections of coloured dots), which polygone passes unchanged into the output file. In extreme cases, a postscript code may just be a wrapper around a bitmap. This type of sham is usually the result of a crude attempt to convert a jpeg or png file into postscript. To a human, bitmaps look like long, random sequences of characters in the postscript code. Another way to spot them is to zoom in close in a viewer; bitmaps will pixelate, true postscript graphics will not.

  If the postscript file does contain groups of shaded polygons, then this error indicates a bug (of sorts). The pattern matching algorithm that polygone uses to detect them is powerful, but probably not infallible. It should be able to detect groups of coloured polygons in any 'reasonably' structured postscript code, but there is nothing to stop software from producing unreasonable code. Contact the author, but note that it may not be possible to fix this problem.

5.3 Processing appears to be successful, but the new postscript file is unusable

The probable cause of this is a bug in the pattern matching algorithm that is used to detect groups of shaded polygons. Contact the author.

A Manually removing an unwanted mesh

Open the postscript code with your favourite text editor. You need to find the instruction that is used to fill polygons. Usually this will be an abbreviation for the `fill` operator (some applications use `eofill` in place of `fill`), located near the top of the file. Different applications will do this in different ways but the definition will probably look something like this

```
/F { fill bind def }
```

or perhaps

```
/F fill load def
```

Change `fill` to

```
gsave fill grestore stroke
```

so that `F` now fills the current path *and* draws its outline. The save and restore instructions are required because postscript objects are usually discarded once they have been used. Save the file and open it in a viewer. The mesh should be gone. If it is not, try some experimentation; perhaps a different procedure needs to be redefined. Otherwise, consult a postscript manual[3], or let polygone do the work for you.

[3] e.g. from `http://www-cdf.fnal.gov/offline/`