polygone version 4 documentation

Ian Thompson
Department of Mathematical Sciences,
University of Liverpool
ian.thompson@liv.ac.uk
pcwww.liv.ac.uk/∼itho17/personal/polygone

July 3, 2014

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 4

- Yet more powerful pattern matching. The structure of postscript graphics generated by certain applications continues to deteriorate, and at least one such application now generates files that polygone 3 does not understand. Version 4 is harder to fool.
- Faster tokenization. The introduction of the faster processing algorithm in version 3 created a situation in which a significant proportion of runtime was used for conversion of postscript code into polygone's internal data structures. Therefore this process has been optimised.
- Better handling of line ends. Earlier versions of polygone relied on the detection of `EOR` when reading in files, which could cause problems if a postscript file created on one machine was processed on another. Version 4 is platform independent in this respect.

[1] beautiful plumage, though.

- New `-a` option. This somewhat experimental option is described in section 5.
- A bug that caused the strings '.' and '-.' to be identified as numbers has been fixed.

3 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.
- 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.

4 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 using the following software:

- NAG Fortran Compiler Release 5.3.2(958) for x86-64 Linux,
- NAG Fortran Compiler Release 5.3.1(924) for Windows 7.

Before compiling on a Windows system, change the line which reads

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

to

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

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

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

5 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.

`-aj` (`j=0, 1` or `2`). This option can be used to deal with files in which not all polygons are drawn using the same postscript operator.

- The default mode `-a0` allows for only one drawing operator, and is appropriate for most files.
- `-a1` causes polygone to allow for two different drawing operators, and treat shapes drawn with these in exactly the same way.
- `-a2` causes polygone to allow for two different drawing operators, but discard polygons drawn by the less prevalent of the two.

Generally, it is best to try processing in the default mode and then try different `-a` options if polygone detects drawing tokens but no polygon groups, or finds a large number of groups each with only a small number of polygons. This could happen if `fill` is used for some polygons and `eofill` is used for others, in which case `-a1` should correct the problem. If `-a1` causes a collapsed path error, the file probably includes some identical shapes twice, e.g. with the outline and interior drawn separately. In this case `-a2` should be used, possibly with `-r`. Postscript files for which modes other than `-a0` are appropriate seem to be rather rare, so it has not been possible to test these options extensively.

`-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. 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 postscript 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 dispappear 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,[3] 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.

6 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.

6.1 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.

6.2 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!`

  There are four ways in which this error can arise.

  - There is something wrong with the postscript file being processed.
  - A file that should be processed with the `-a2` option is being processed with the `-a1` option (see section 5).

[3] Running polygone with the `-r` option creates a file of this type.

- A false positive result from the pattern matching algorithm has caused polygone to attempt to process code that does not in fact represent a group of polygons. This can happen if the postscript code doesn't actually contain polygon groups (see the error `no drawing operators detected`, below).
- (Unlikely) There is a bug in polygone's processing algorithm.

- `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 with 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.

6.3 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 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,[4] or let polygone do the work for you.

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