

Creating (mathematical) jigsaw puzzles using jigsaw-generator

Julian Gilbey

December 2015

Abstract

The jigsaw puzzles described here are constructed from shapes such as triangles, squares and so on, with questions and answers written along their edges. The aim is to match them up correctly. Related puzzle varieties are card sorts and dominoes. This package provides a \TeX - and Python-based system designed to author such puzzles. The input is a text-based YAML file; the output can include both printable PDFs for cutting up and Markdown files for potential conversion to HTML.

Contents

1	Introduction	1
1.1	Usage overview	2
2	Data input files	3
2.1	Available layouts	4
2.2	Layout information	4
2.3	Card content	7
3	Puzzle templates	8
4	Filters	9
5	Configuration files	9
6	Some notes on YAML syntax	9

1 Introduction

For a general background to this software and its history, see my TUGboat article [2].

This documentation describes the usage of the software; some details have changed since the TUGboat article, and so all of the relevant material will be repeated here.

1.1 Usage overview

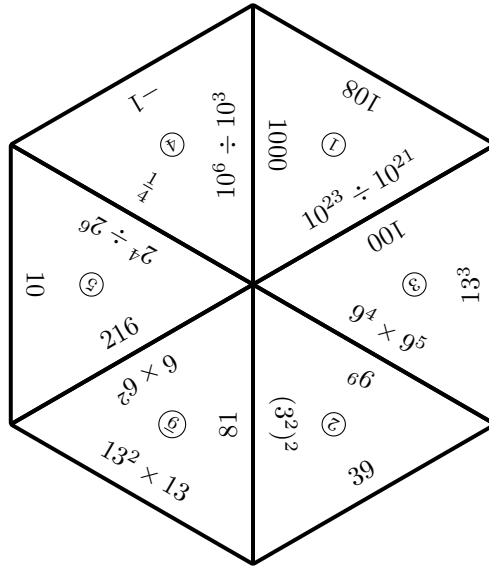
The most important files from a user's perspective are the data input files. These specify the type of puzzle (a hexagonal jigsaw or a card sort, for example), any options for the jigsaw (such as whether to number the cards or not) and the text to appear on the puzzle. Sensible defaults are provided for all of the options if they are not specified. These input files are written in YAML.

For example, here is a data file for creating a small hexagonal puzzle:

```
format: 1
type: smallhexagon
title: An example puzzle
note: 'You will have to work out the missing number shown as `?'`'

pairs:
  - ['$10^6\div10^3$', '$1000$']
  - ['$10^{23}\div10^{21}$', '$100$']
  - ['$9^4\times9^5$', '$9^9$']
  - - '$(3^2)^2$'
    - puzzletext: '?'
      solutiontext: '$81$'
  - ['$6\times6^2$', '$216$']
  - ['$2^4\div2^6$', '$\dfrac{1}{4}$']
edges:
  - '$-1$'
  - '$10$'
  - '$13^2\times13$'
  - '$39$'
  - '$13^3$'
  - '$108$'
```

The solution to the resulting puzzle looks like this:



If the puzzle is stored in the file `hexpuzzle.yaml`, then the command:

```
jigsaw-generate hexpuzzle
```

will produce several PDF files: `hexpuzzle-puzzle.pdf` (containing the puzzle), `hexpuzzle-solution.pdf` (the solution) and `hexpuzzle-table.pdf` (with the puzzle content laid out in a tabular form), along with Markdown versions `hexpuzzle-puzzle.md` and `hexpuzzle-solution.md` containing the puzzle and solution respectively; these can be used to create HTML versions of the puzzle via pandoc or some such system.

The command-line command `jigsaw-generate` offers a number of command line switches; run the command

```
jigsaw-generate --help
```

to obtain information about this.

In the following sections, the data files are described in detail, and then the other supporting files (configuration files, templates and so on) and command line options are described.

In many cases, it is adequate to simply copy and edit the example input files, which will be found in `/usr/local/share/doc/jigsaw-generator/examples` or `/usr/share/doc/jigsaw-generator/examples`, depending on your specific installation of the package.

2 Data input files

The data input files consist of layout information (the type of puzzle, whether to produce a solution and so forth) and content information. Being a YAML

mapping, these can appear in any order in the data file; will will describe them in these two separate sections for simplicity.

2.1 Available layouts

There are four jigsaw puzzle layouts currently available:

- `smallhexagon`, which consists of 6 triangles as shown above;
- `hexagon`, which consists of 24 triangles;
- `triangle`, which consists of 16 smaller triangles;
- `parquet`, which consists of 4 squares and 8 triangles.

There are also three card sort layouts currently available:

- `cards`, which consists of an arbitrary number of rectangular cards; no solution is produced, and this is intended for classification activities or suchlike, where there is no “correct” answer expected; these cards are not shuffled by default;
- `cardsort`, which is like cards, except that there is a correct order, so the cards are shuffled and a solution is produced;
- `dominoes`, which consists of an arbitrary number of domino cards; a solution is produced.

More details are given in the following sections.

2.2 Layout information

Each of these options must be followed by a colon and a space, followed by the option value, as demonstrated in the example above. All of these options are optional unless stated otherwise. The most common options are given first in each of the following three lists.

Options applicable to all puzzle types

- **format:** The version of the file format. It is currently ignored, but the format version of the current software is 1. Future changes to the software will use the format version where relevant to produce backward-compatible behaviour for older format files.
- **type:** (required) The type of jigsaw being created. The current recognised standard types are those listed above, namely `smallhexagon`, and so on.
- **title:** The title of the activity, printed at the top of the page.
- **note:** A note printed at the top of the page.

- **produceSolution**: Whether to produce a solution PDF and Markdown file; the default is **true** except for cards, where the default is **false**.
- **numberCards**: Do we number the cards for ease of identification? (Default: **true**)
- **textSize**: How large the text should be on the cards; this is a number from 0 to 9, with 0 being the smallest. (Default: 5)
- **makepdf**: Whether to produce PDF output files. (Default: **True**)
- **makemd**: Whether to produce Markdown output files. (Default: **True**)
- **latex**: Which \LaTeX engine to use to produce the PDF files. (Default: **pdflatex**)
- **texfilter**: If specified, this is a filter to pass the \LaTeX files through prior to running \LaTeX . See the later section on filters (4) for more information on filters.
- **mdfilter**: If specified, this is a filter to pass the Markdown files through to produce the final Markdown files.
- **clean**: If true, then intermediate files will be deleted. This includes the \LaTeX files and related aux files and so on. (Default: **true**)
- **puzzleHeaderTeX**: This specifies the header file to use to produce the \LaTeX puzzle file. See the later section on templates (3) for more information on header and template files.
- **puzzleTemplateTeX**: This specifies the template file to use to produce the \LaTeX puzzle file.
- **solutionHeaderTeX**: This specifies the header file to use to produce the \LaTeX solution file.
- **solutionTemplateTeX**: This specifies the template file to use to produce the \LaTeX solution file.
- **tableHeaderTeX**: This specifies the header file to use to produce the \LaTeX table file.
- **tableTemplateTeX**: This specifies the template file to use to produce the \LaTeX table file.
- **puzzleHeaderMarkdown**: This specifies the header file to use to produce the Markdown puzzle file.
- **puzzleTemplateMarkdown**: This specifies the template file to use to produce the Markdown puzzle file.

- **solutionHeaderMarkdown:** This specifies the header file to use to produce the Markdown solution file.
- **solutionTemplateMarkdown:** This specifies the template file to use to produce the Markdown solution file.
- **hiddennote:** If any entries are hidden, the note to include at the top of the PDF solution output file. (Default: 'Entries that are hidden in the puzzle are highlighted in yellow.')
- **hiddennotemd:** If any entries are hidden, the note to include at the top of the Markdown solution output file. (Default: 'Entries that are hidden in the puzzle are indicated with (*).')
- **hiddennotetable:** If any entries are hidden, the note to include at the top of the PDF table output file. (Default: 'Entries that are hidden in the puzzle are indicated with (*).')

Options applicable to card sorts and dominoes only

- **rows:** (required) How many rows of cards to produce per page.
- **columns:** (required) How many columns of cards to produce per page.
- **shuffleCards:** (card sorts only) Are the cards shuffled? This is by default **true** for cardsort, and **false** for cards.
- **shufflePairs:** (dominoes only) Are the order of the pairs shuffled from the order given in the data file? This is by default **false**.
- **flip:** (dominoes only) Are the order of the entries in each pair presented in a random order? This is by default **false**.
- **cardTitle:** The title to write on each card.
- **cardTitleSize:** How large the title should be on the cards, a number from 0 to 9 (like the **textSize**). By default, this is 1 less than the label size (see below).
- **label:** The default label to write on each card. This is like a subtitle, and appears at the bottom of the card. (This can be useful if the cards are of two or more types, and the type of card is important.) This is not used for dominoes.
- **labelSize:** How large the labels should be on the cards, again a number from 0 to 9; by default, this is 2 less than the text size. (This can be overridden for individual cards.) This is not used for dominoes (though setting it will have an impact on the size of the title if **cardTitleSize** is not set).

- **cardsep**: How much space to leave between adjacent cards. By default this is 12pt. It should be a TeX length.
- **cardsepHorizontal**: How much space to leave between horizontally adjacent cards. By default, this is **cardsep**.
- **cardsepVertical**: How much space to leave between vertically adjacent cards. By default, this is **cardsep**.
- **loop**: (dominoes only) Do we have a domino loop, with there being no start and end? The default is **true**.
- **start**: (dominoes only) If we do not have a loop, then the text to write on the start of the chain. The default is '**Start**'.
- **finish**: (dominoes only) If we do not have a loop, then the text to write on the end of the chain. The default is '**Finish**'.

Options applicable to jigsaws only

- **shufflePairs**: Are the order of the pairs shuffled from the order given in the data file? This is by default **false**.
- **flip**: Are the order of the entries in each pair presented in a random order? This is by default **true**.
- **shuffleEdges**: Are the order of the edges shuffled from the order given in the data file? This is by default **false**.

2.3 Card content

The content of the cards is given in a final one or two YAML key-value pairs, depending on the type of material being produced. The keys are **cards** (for card sorts), **pairs** (for dominoes and jigsaws) and **edges** (for jigsaws).

Each of these takes as a value a YAML sequence of card content. For **cards** and **edges**, each of these is a single entry, while for **pairs**, each one is a sequence of two entries, as in the example above. The number of cards, edges or pairs depends upon the type of puzzle being produced.

Each entry can have exactly one of the following types of content:

- A string. This will be the content of the relevant card, edge or half of a pair. This will be inserted verbatim into the Markdown and L^AT_EX output files, so should be written in L^AT_EX. (Images are handled specially; see below.)
- A YAML mapping. This uses the following keys:
 - **text**: This contains the text to be used as content for this entry (card, edge or half of a pair).

- **size**: This will be added to the **defaultsize**, so a size of 1 will produce larger text in the PDF while a size of -1 will produce smaller text.
- **hidden**: If this is set to **true**, the entry will not be shown on the puzzle, only on the solution.
- **label**: This overrides the current label.
- **puzzletext**:, **puzzlesize**: These override the **text** and **size** settings for the puzzle. Specifying **puzzletext** overrides any **hidden** setting.
- **solutiontext**:, **solutionsize**: These override the **text** and **size** settings for the solution. If **solutiontext** is specified, then **hidden** is assumed to be **true**.

Either **text** or both **puzzletext** and **solutiontext** must be present.

- Special YAML content.
 - **newpage**: **true** This produces a new page in the PDF output file for card sort activities, but has no effect on the table or Markdown outputs.
 - **newlabel**: This contains the text to be used as the default label for all following cards.
 - **newlabelsize**: This specifies the label size to be used as the default for all following cards.

3 Puzzle templates

The puzzle templates are \LaTeX files with templating marks. For the supplied templates, PGF/TikZ has been used for the graphics and to place the text items. Version 3.0.0 or higher of PGF/TikZ is required for the current templates. Alternative templates can be written if this is desired. For example, someone might prefer to use Asymptote or another graphical package, or they may wish to modify the existing templates in various ways.

There is also a layout file (written in YAML) for each puzzle type: this specifies various parameters required to create the puzzle.

TODO: Write a description of the template and layout files

In the meantime, please look at the template and layout files in the distributed package to see how they are written, and modify them appropriately if you wish to create your own. You can also email me if you have questions.

These files can be specified in the puzzle files as described above (with the template description file specified by the **type** option, and the templates defined within the layout file or overridden by the corresponding options in the puzzle file).

These files are looked for in the current directory, in the user config directory (typically `~/.config/jigsaw-generator/templates/`) and in the package

templates directory (typically `/usr/share/jigsaw-generator/templates/` or `/usr/local/share/jigsaw-generator/templates/`).

4 Filters

There may be occasions on which you need to systematically modify the \LaTeX file prior to running \LaTeX , or to modify the Markdown output in some way. Filters offer this facility. They are programs (scripts or other executables) which read in the \LaTeX or Markdown on standard input and write the filtered version to standard output. For example, you might have reason to change the standard Markdown or \LaTeX syntax for images to meet some local need.

Filters can be specified either on the command line or within the puzzle files or within the template description files.

Filters are looked for in the current directory, in the user config directory (typically `~/.config/jigsaw-generator/filters/`) and in the package filters directory (typically `/usr/share/jigsaw-generator/filters/` or `/usr/local/share/jigsaw-generator/filters/`; this assumes that all supplied filters are system-independent scripts).

5 Configuration files

The `jigsaw-generate` program reads a configuration file, found at `~/.config/jigsaw-generator/config.ini`, which is in an INI format. The default configuration file can be found in `/usr/share/jigsaw-generator/templates/default_config.ini` (or `/usr/local/share/...`, depending on where it has been installed).

The default file is effectively as follows:

```
[jigsaw-generate]
makepdf = yes
makemd = yes
latex = pdflatex
clean = yes
texfilter =
mdfilter =
```

and these correspond to the identically-named command-line options.

6 Some notes on YAML syntax

For a full, precise description of YAML syntax, see the YAML Specification [1]. What follows is a very brief summary of some of the basic parts of the specification which should be sufficient for most people's needs when using this software.

Although YAML allows unquoted strings in general, there are a number of restrictions on what is permitted in them. For this reason, it may be simplest

to single-quote all value strings. (YAML also offers double-quoted strings, but these interpolate backslash-escapes; this is probably undesirable in this context, since many \TeX expressions include backslashes.)

Within a single-quoted string, a single quote is written as a doubled quote mark (''), hence the three quote marks in a row at the end of the note field in the above example (two to indicate a quote, and the third to end the string).

For collections ('sequences' and 'mappings' in YAML's terminology, each of which consists of a number of 'entries'), YAML allows two different notations: either a flow style, which is similar to JSON notation, or a block style. With the flow style, the entries of a sequence, such as a question-answer pair or the list of edges, are enclosed in square brackets and separated by commas; for the block style, each entry appears on a new line preceded by a vertically-aligned hyphen.

Similarly, for a mapping the entries consist of key-value pairs, with the key and value separated by a colon. They can be written either using a flow style as a comma-separated list enclosed in braces, or with a block style by writing each key-value pair on an identically-indented new line.

Both of the sequence styles appear in the example here, though only the block style is used for mappings. Note also that the top-level structure of the file is itself a mapping. This means that the entries (type, title, pairs, and so on) can appear in any order. However, for the benefit of the human reader, it is wise to maintain a meaningful order to these entries.

Acknowledgements

I thank those who offered very useful ideas and feedback at the TUG conference; these have helped me to solve some of the thorny issues I had been facing.

References

- [1] Oren Ben-Kiki, Clark Evans, and Ingy döt Net. YAML Specification. Available from <http://www.yaml.org/spec/1.2/spec.html>.
- [2] Julian Gilbey. Creating (mathematical) jigsaw puzzles using \TeX and friends. *TUGboat*, 35(2):168–172, 2014.