WorkflowMaker. Simple option files format description

For WorkflowMaker version **1.0.5** and later

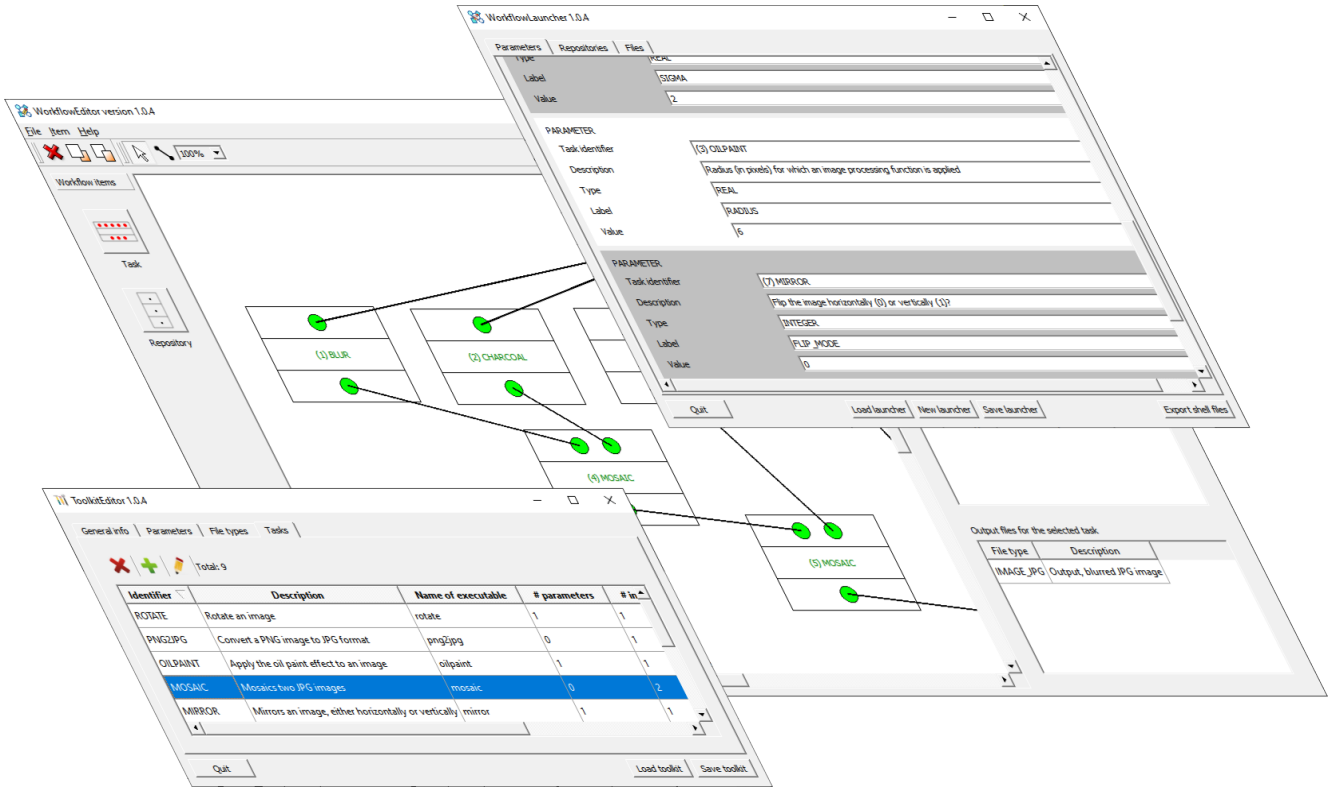

Table of contents

1 Introduction

This document describes the format for the options files that is supported by the simple_options_file_parser library. This format is compatible with the requirements of the WorkFlowMaker software suite.

The library is written in C++ and depends on no other software components. Note that this document describes the format of the supported options files, not the API (Application Programming Interface) of said library.

2 The simple options file

A simple options file is a plain text file where a list of parameters and their values is presented. The way options are specified is pretty simple and adheres to the syntax:

```
<label> = <value>
```

Where `<label>` stands for any series of nonblank characters (as `SPEED_IN_MPH` or `MINIMUM-TIME-TO-WAIT`) and `<value>` corresponds to the value that must be assigned to the aforementioned label. The following would be a valid option assignment:

```
ISOLATION_CLUSTER_SIZE = 2
```

The whitespace before, between the label and the equal sign, between this one and the value and, finally, between the line and the end of the line is simply ignored. Therefore,

```
NUMBER_OF_ITERATIONS                    =            3
```

is equivalent to

```
NUMBER_OF_ITERATIONS = 3
```

Values must be written *correctly*. This apparently obvious statement points to the fact that a value is correct – or not – depending on its *interpretation*. For instance, the string "`H_llll987`" may be perfectly valid if a tool interprets such value as a series of characters (it could be an alphanumeric identifier). On the contrary, if the same tool is expecting an integer or a real value, the aforementioned string would be clearly incorrect.

List of values (for integer and floating-point types) are also possible. For instance, the following definition would be correct (the values are interpreted as a list or array of floating-point magnitudes):

```
TNITABLE_THRESHOLDS = 0.53 0.70 0.84
```

Strings may also have several “words”, and these may be interpreted as a single value or as several ones. That is, in the following example:

```
YOUR_NAME = John F. Kennedy
```

The value will may be interpreted as “John F. Kennedy” or as “John” + “F.” + “Kennedy”. This depends on the software reading the file, since it is possible to choose between these two possibilities in the API provided to process defaults files. Therefore, users must refer to the documentation of the tool using a single options file to learn how, in that specific application, the different string-valued options are treated. That is, the way such data is interpreted depends on each tool.

When labels are used to define list of options, whatever its type (integers, doubles, strings) that include many values, it is possible to split these in several lines. To do it, use a backslash (“\”) at the end of each line. The very last line, where the list of values ends, must not include a trailing backslash, since this is the way to indicate that no more lines follow. The following is an example spanning three lines.

```
LABEL_WITH_LONG_LIST = 10 20 30 \
                       40 50 60 \
                       70 80 90
```

Comments may be introduced in options files. To do it, use the “#” character (excluding the quotes) as the first non-blank character in a line. Comments end at the end of the line. For multiple line comments, write a “#” character on each line. The following is an example of a multi-line comment.

```
# The threshold values themselves. Note that there must be as many values
# as stated in the parameter TNITABLE_N_VALUES!!! Note that these thresholds
# are used to check autocorrelation values; therefore, these must be in
# the RANGE (-1, 1) and SORTED in ASCENDING order.
```

Blank lines are simply ignored and may be thus used to format the options file making it more readable.

The following is an actual example of a simple options file.

```
# ------------------------------------------------------------------------
#
# THIS IS A LOS2HV OPTIONS FILE
#
# ------------------------------------------------------------------------
#
# OPTIONS
#
# ------------------------------------------------------------------------
#
# Options controlling the behavior of the application -------
#

#
# Spacing of the grid used to group the points.
```

```
# Use the same units than the projected coordinates.
#

GRID_SPACING          = 500

#
# The output will consist of points centered on the
# squares defined by the grid (1) or the polygons defining
# the squares themselves (2).
#

OUTPUT_MODE           = 1

# The ascending look angle (in radians).

LOOK_ANGLE_ASCENDING  = 0.5

# The descending look angle (in radians).

LOOK_ANGLE_DESCENDING = 0.5

# ------------------------------------------------------------------------
#
# FILES
#
# ------------------------------------------------------------------------

#
# Input files ------------------------------------------------------------
#

# Shapefile with points. ASCENDING.

FILE_POINTS_ASCENDING       = points_ascending.shp

# Shapefile with points. DESCENDING.

FILE_POINTS_DESCENDING      = points_descending.shp

#
# Read map describing the structure of both
# input shapefiles with points.
#

FILE_POINTS_READ_MAP        = points_read_map.op

#
# Output files -----------------------------------------------------------
#

# Output shapefile with HORIZONTAL component.

FILE_COMPONENTS_HORIZONTAL  = horizontal_components.shp

# Output shapefile with VERTICAL component.

FILE_COMPONENTS_VERTICAL    = horizontal_components.shp
```