User Manual for *Survey2GIS* (Version 1.5.1)

– software for converting survey data to GIS file formats –

written by the *survey-tools.org* development team (http://www.survey-tools.org)

13th October 2019

Contents

1 Introduction

Prologue: This software is meant to process data coming from a *topographic survey*. The purpose of the latter is to capture and map geometric aspects of the real world, *as visible* on the surface. Accordingly, this software has been designed to process "geometries" (points, lines and polygons) that represent the cross-sections of real-world features, *as visible* on a (more or less horizontal) cutting plane (i. e. a natural or excavated surface). Limited support for vertical cross-sections has been added in version 1.5 of this software (see 7). *Survey2GIS* has not been designed to act as a tool for constructing or drawing *assumed* geometries. Its design philosophy is in alignment with the (prevailingly 2D) GIS concepts of *layers* and *topological quality*. If the geometric output produced by this software does not match expectations, then users are urged to review section 9 of this manual, before looking for errors in the data, processing settings or the software itself.

This user manual describes *Survey2GIS*, a flexible tool capable of processing survey data into high quality GIS vector data. The input data must be in plain text format and contain one single point *measurement* (=record) per line. Such data may be downloaded from a total station or GPS, but may also be manually created using a simple text editor. Digital surveying devices take point measurements and store them in internal memory, from where they can be downloaded, using the software supplied by the manufacturer, and saved into a plain text file suitable for processing by *Survey2GIS*.

Broadly speaking, using *Survey2GIS* is then a two-step process:

1. The user writes a simple parser definition text file that describes the structure of the survey data file(s).
2. The user runs *Survey2GIS* with the parser file and one or more survey data files as input.

The parser file syntax is described in details in section 4 of this document.

In order to produce complex shapes ("geometries"), such as line strings or polygons, from one ore more survey data files, individual point measurements (i. e. the vertices of the geometries) have to be merged in a precisely controlled manner. In addition, each geometry may be associated with an *attribute* record in the GIS output data to store descriptive data such as unique identifiers and custom object codes. To assure that all of this works flawlessly, the surveyor must take care that each measurement is attributed with a special code/tag which will later allow *Survey2GIS* to decide which records to merge into line or polygon geometries. The hardware used for surveying, the field work and the data processing must be aligned accordingly for an efficient and seamless workflow.

The role of *Survey2GIS* is to act as translator between the raw survey data and the more complex objects ("features") used in GIS-based data processing. The primary goals of this software's design are flexibility, robustness and high quality of output data. Although good data processing requires a rigorous approach to quality control, there is no need for the *Survey2GIS* user/surveyor to follow one specific surveying method. Instead, one of several basic processing modes may be chosen and the structure of the survey data may be adapted in various ways to suit individual needs.

Note: The GIS vector model, even though it may be capable of storing and visualizing 3D coordinates, is not a real 3D model. Most importantly, GIS use a basic top-down 2D model to represent vectorial layers and the topological relationships ("overlaps", "contains", "borders", etc.) between objects/features. This means that GIS are *not* well-suited for accurately representing real-world objects with pronounced vertical structuring, such as complex buildings. The intended use case for *Survey2GIS* is *topological* surveying, i. e. measuring points, linear structures and areas on the terrain surface. Some limited support for "vertical" data, to be used in 2D GIS, is available (see 7).

1.1 Provided functionality

The following functionality is currently provided by *Survey2GIS*:

- Line-wise reading of point coordinate measurements and encoded attribute data from one or more input text files.
- Multiplexing of point readings into geometric objects (line strings and polygons).
- Freely definable fields (schema) for storing attribute data in the GIS output data.
- Detailed validation and recording (protocol) of all measurements and their processing, including topological cleaning.
- Merging of all input files and export of geometry and attribute data into standard GIS file formats.
- Re-orientation and reprojection of coordinate data.

Some of the software's noteworthy features are:

- Open source and freely available. No proprietary technology, no annoying copy protection or usage restrictions.
- Runs under Linux, Mac OS X (macOS) and Windows operating systems.
- Minimal hardware requirements; ideal for use on small field laptops.
- Can be used through the command line or a graphical user interface (GUI).
- Processes and produces 2D or 3D data.
- Simple to translate into any language, including different numeric notations (decimal point and grouping characters). The software can always be switched back to "English" for use in multi-lingual environments.
- Freely definable parser schema allows to adjust the processing for any input data field structure.

- Processing of several input data files at the same time and merging of their data into a single output file.
- Complete recording of all hints, warnings and error message issued during processing; on screen and into a text file.
- Invalid input data will be discarded from the output, but processing will continue in most cases.
- Handling of missing and "no data" records.
- Topological data validation and cleaning to produce high quality GIS data suitable for spatial analysis.

1.2 Structure of input data

Input data may be supplied as one or several text files (alternatively from the operating system's standard "in" stream: see 2.1). These text files contain at least one line (record) of data, with each line storing the coordinates of exactly one point measurement, along with additional attribute data (identifiers, custom object codes, etc.). Attribute data may be preserved or discarded in the output, according to the needs of the end user. A minimal record does, however, need at least an X and a Y coordinate value.

There is no provision in *Survey2GIS* for handling various legacy text encodings, such as DOS and Windows "codepages" or extended ASCII tables. This software assumes that input text is either plain ASCII or UTF-8 encoded. The user must take care that any software used to further process the data produced by *Survey2GIS* can correctly interpret the text encoding (see 11 for more on this topic).

1.3 Installation

There is no installation routine in the form of a "setup" program. Simply locate the version for your operating system in the "bin" folder and run it. There is a "README" file for each operating system that contains further details.

Users who do not wish to use proprietary software to download data from their survey devices may find *TotalOpenStation* (`http://tops.iosa.it`), a free tool for serial communication with selected total stations and GPS models, a useful companion to *Survey2GIS*.

Windows users should also consider installing a capable text editor (such as *NotePad++*: `http://notepad-plus-plus.org`), to facilitate the manual editing and cleaning of large data and log files.

1.4 Basic usage

This software may be used from the command line or through a graphical user interface (GUI). The same capabilities are available in both modes. The GUI is described further in section 3. To use *Survey2GIS* from the Windows command line, start *cmd.exe* first, then change into the folder where the software resides (replace all items in pointed brackets "<>" with the actual path entries), by typing:

```
cd /D <drive>:\<survey2gis folder>
```

For a first test run, create a folder named "output" within the the program's folder (or another convenient location with write access). The data produced by *Survey2GIS* will be written into this folder. In addition, copy the files "parser_desc_end.txt" and "sample_data_end.dat" from the "samples" subfolder (shipped as part of the *Survey2GIS* package) into the installation folder. The software can then be run by issuing the following command:

```
survey2gis -p parser_desc_end.txt -o output -n test_data sample_data_end.dat
```

Details of the above command line:

- Option *-p* (*parser*) specifies the file that has the information about the input data's structure (schema). Details about the syntax of this file can be found in section 4.2.
- Option *-o* (*output*) specifies the folder into which all output files will be written. In this example, its name happens to be "output", as well. The default for *Survey2GIS* is to write the data in *Shapefile* format (see 10 for choices).
- Option *-n* (*name*) specifies the user-definable (base) part of the output files' names. E. g. in the case of Shapefiles, there will be several output files, with the extensions „.shp", „.shx" and „.dbf" added to the base name (see 10.1).
- The final item specified on the command line is not an option, but the name of an input file. The file „sample_data_end.dat" contains one simple survey dataset with point, line and polygon records. This is a plain text file that can be viewed and edited using any text editor. It would be possible to specify further input file names on the command lines. The contents of all input files would then be merged by *Survey2GIS* and written to just one output dataset.

The result of the above command line should be three Shapefile sets containing the point, line and polygon records from "sample_data_end.dat" respectively.

A summary of all command line options is available via the option *-h* (*help*):

```
survey2gis -h
```

A detailed discussion of all options is available in section 2.

2 Program startup and options

The software is controlled by a number of options that can either be stated on the command line (started via *cmd.exe* on Windows systems), as described in this section, or through a graphical user interface (GUI), as described in section 3. Both interfaces offer the same level of control.

On the command line, the name of the executable program ("survey2gis") is followed by the names (or abbreviations) of one or more options, each of them in turn followed by user-specified option values (=settings). Finally, a list of input files is supplied on the same line. All option and input file specifications must be separated from each other by a space character.

Starting the software with the option "-h" (for "help", note the space preceding the option name):

```
survey2gis -h
```

... will display a summary description of the software and its options on the screen (much shortened here; actual display may differ):

```
Usage: survey2gis -p FILE -o DIR -n NAME [OPTION]... [FILE]...
Read geometry and attribute descriptions from a survey protocol file
in ASCII format and convert them to common GIS and CAD output formats.

Possible OPTIONs are:
  -p, --parser=         name of file with parsing schema (required)
  -o, --output=         directory name for output file(s) (required)
  -n, --name=           base name for output file(s) (required)
  -f, --format=         output format (see list below; default: "shp")
                        "shp" (Esri Shapefile)
                        "dxf" (AutoCAD Drawing Exchange Format)
                        "geojson" (GeoJSON Object)
                        "kml" (Keyhole Markup Language)
  -L, --label=          choose field for labels (see manual for details)
      --label-mode-point= label mode for points (default: "center")
      --label-mode-line= label mode for lines (default: "center")
      --label-mode-poly= label mode for polygons (default: "center")
                        "center" (Place at center of geometry)
                        "first" (Place on first vertex)
                        "last" (Place on last vertex)
                        "none" (Do not label)
  ...
```

A minimal command line for running *Survey2GIS* to process some actual data must include:

- The specification of a parser schema file (see 4) with option "-p",
- the name of an *existing folder* in the file system, where the output data is to be stored (option "-o"),
- the base name for naming all output files, using option "-n",
- and one or more input file names.

The parser schema describes the structure of the input data in detail. It must be supplied as a structured text file. Its syntax is described in detail in section 4.

Most option names have short and long spelling variants, which are equivalent. E. g. it makes no difference whether the parser file is specified using:

```
-p parser.txt
```

or using the long option name:

```
--parser=parser.txt
```

Long option names are separated from option values by an equal sign ("=") and generally improve the legibility of command line statements.

The order in which options are given is of no importance, but input files should always be listed as the last items of the command line.

Option values and file names that include spaces (a practice that should generally be avoided), must be written in quotation marks. Otherwise they will be interpreted as separate tokens on the command line.

As an alternative, the *graphical user interface* (GUI: see 3) can be started by calling *Survey2GIS* without any options, or by adding the special option "--show-gui":

```
survey2gis --show-gui
```

2.1 Input and output files

All items specified on the command line, that follow the list of options and option values, will be interpreted as the names of input data files. All input data files will be interpreted according to the parser schema file (specified with option "-p"). The output data produced will be stored as files in the folder specified by the "-o" option. The output folder must exist and must allow write access. The current folder can be used as output folder by specifying the single dot (".") as folder name.

The output file format (see details in 10) is selected using option "-f". By default, this option's value is "shp", which causes *Survey2GIS* to produce GIS Shapefiles (see 10.1). All input data will be merged into

a common output dataset. The actual names and number of output files depends on the chosen format. For this reason, the option "-n" specifies a *base name* only, without any file extension. The base name will be automatically extended and file type extensions added as required.

Important: This also means that an existing file will be *overwritten automatically*, if its name clashes with one of those produced by *Survey2GIS*!

The minus ("-") character has a special purpose. If it is given instead of a file name, then *Survey2GIS* will wait for input data to arrive through the operating system's standard input stream. On Mac OS X (macOS) and Linux systems, this stream is known as *stdin*. By default, it is linked with the keyboard input stream, so that the user may directly type input data to be forwarded to *Survey2GIS*. To complete keyboard input, the user has to type the special sequence "^Z" on Windows systems (press keys "CTRL" and "Z" simultaneously), and "^C" ("CTRL" and "C") on Mac OS X and Linux, on an otherwise blank line.

2.2 Language settings

All messages and GUI elements of *Survey2GIS* are "internationalized". On startup, *Survey2GIS* will check the language settings of the operating system on which it is running and display messages in the system's language, provided that a matching translation is available. If no translation exists, then *Survey2GIS* will fall back to English. In any case, option "-e" will force *Survey2GIS* to run in English mode, disregarding any operating system settings.

Internationalization also includes different numerical notations. E. g. the number "tenthousandpointzero" would be written like this in a German or Dutch environment:

```
10.000,00
```

Whereas in an English speaking location, it would be:

```
10,000.00
```

The numerical notation, including representations of the decimal point and grouping character, will be set to the *English standard* by default, as this is how most hardware produces output. However, *Survey2GIS* offers settings for the specification of decimal point and grouping characters (using options "-i" and "-g", respectively).

It is of crucial importance that the numerical notation is correctly set to that of the *input data*. Many devices and programs use English notation by default. Typical symptoms of a mismatch include wrongly placed geometries in the output data, as well as stepped and deformed line segments and polygon boundaries.

2.3 List of options

The following is a complete but terse description of all options accepted by *Survey2GIS*. Links to more details on how these options influence processing are provided where appropriate. Options are listed in the same order in which they appear in the command line usage summary (invoked by running "survey2gis --help").

Option: **-p**, --parser=

Meaning: Complete path and name of the file that contains the parser schema for processing all input data (see 4).

Example:

```
-p C:\Users\Me\parser.txt
```

Option: **-o**, --output=

Meaning: Complete path and name of the folder in which to store all output data.

Example:

```
-o c:\Data
```

Option: **-n**, --name=

Meaning: Base name (no extension) for all output file(s). The files will be stored in the folder specified by "-o" (see above).

Example:

```
-n output
```

Option: **-f,** --format=

Meaning: Name of the format in which to store the output data (default: "shp").

Remarks: The help text displayed via option "-h" contains the complete list of possible formats (see also 10).

Example:

```
-f shp
```

Option: **-L,** label=

Meaning: Specify the name of a *field* (see 4.3) to use for creating a special labels layer. See section 7 for details.

Example:

```
-L level
```

Option: --label-mode-point=

Meaning: Specify where to place labels for *point* geometries. Valid choices are: "center" (default), "first", "last" and "none.

Remarks: Only "none" will have a real effect in the case of point geometries. See section 5 for details.

Example:

```
--label-mode-point=none
```

Option: --label-mode-line=

Meaning: Specify where to place labels for *line* geometries. Valid choices are: "center" (default), "first", "last" and "none. See section 5 for details.

Example:

```
--label-mode-line=center
```

Option: --label-mode-poly=

Meaning: Specify where to place labels for *polygon* geometries. Valid choices are: "center" (default), "first", "last" and "none. See section 5 for details.

Example:

```
--label-mode-poly=center
```

Option: **-O,** --orientation=

Meaning: Sets the *orientation* of the output data's coordiante axes. The default orientation mode is "world" which does not modify the coordinate data. See section 7 for details.

Example:

```
-O localxz
```

Option: **-S,** --selection=

Meaning: Specify a *selection command* to reduce the input data to only those records that pass the selection criteria. It is possible to specify one or more selections, each one via a separate "--selection=" option. The selections will be applied in the same order in which they are specified. See section 6 for details.

Remarks: Note that this option's shortcut is a *capital* "-S", while "-s" is the shortcut for the "--snapping=" option (see below).

Example:

```
-S objdesc:all:EQ:Boundary
```

Option: **-l**, --log=

Meaning: Stores all processing messages displayed by *survey2gis* in the specified file.

Example:

```
-l C:\Users\Me\protocol.txt
```

Option: **-t**, --tolerance=

Meaning: Sets the threshold for the distance of individual point measurements. Measurements that are spaced below this distance from each other will be regarded as duplicate (see 9.1.1).

Example:

```
-t 0.05
```

Option: **-s**, --snapping=

Meaning: Sets the threshold for snapping of vertices that constitute polygon boundaries. Snapped vertices will be moved so that they are placed exactly on nearby vertices off neighbouring polygon boundaries. The maximum distance up to which snapping is performed is specified by this option (see 9.1.5 for details). Setting this option to "0" (default) will disable boundary vertex snapping.

Example:

```
-s 0.05
```

Option: **-D**, --dangling=

Meaning: Sets the threshold for snapping of vertices that constitute the first and last vertices (also called "nodes") of lines. Snapped line nodes will be moved to the closest vertex or segment on a nearby line or polygon boundary. This process is also known as *overshoot* (node is located behind the segment on which it should lie) and *undershoot* (node is located before the segment on which it should lie) correction of so-called "dangles" (see 9.1.8 for details). The maximum distance up to which node snapping happens is specified by this option. Setting this option to "0" (default) will disable line node snapping.

Remarks: Note that this option's shortcut is a *capital* "-D", while "-d" is the shortcut for the "--decimal-places=" option (see below).

Example:

```
-D 0.05
```

Option: **-x**, --x-offset=

Meaning: Specifies a constant offset to be added to all *X* coordinates.

Remarks: This may be useful if e. g. UTM coordinates have been recorded without the first two digits. Please note that the attribute table of the output data will always store the original values, without the offset! If the transformed values should also be stored in the attribute table, then the appropriate GIS tools ("field calculator" or similar) must be used directly after data import to add the offset to the field values.

Example:

```
-x 100000
```

Option: **-y**, --y-offset=

Meaning: see above (-x, --x-offset=).

Example:

```
-y 100000
```

Option: **-z**, --z-offset=

Meaning: see above (-x, --x-offset=).

Example:

```
-z 100000
```

Option: **-d**, --decimal-places=

Meaning: Specifies the number of decimal places (precision) with which numerical values will be stored in the attribute fields of the output data. For survey data in meters, a setting of "2" means centimeter precision. The precision of coordinate storage is not influenced by this. The default is "3".

Example:

```
-d 3
```

Option: **-i**, --decimal-point=

Meaning: Specifies the character used for representing the decimal point in the *input data* (see 2.2).

Example:

```
-i .
```

Option: **-g**, --decimal-group=

Meaning: Specifies the grouping character used for formatting large numbers in the *input data* (see 2.2).

Example:

```
-g ,
```

Option: **-r**, --raw-data

Meaning: If this option is given, then *survey2gis* will produce an additional output dataset that preserves all individual point measurements of the original survey (i. e. before the data was multiplexed into lines and polygons).

Note: The "raw" output will *not include* vertices that have been removed during topological cleaning, because they are spaced too closely to another vertex (see 9). To also preserve such vertices, set the option "--tolerance=" (see above) to a negative value. In parser mode "None" (see 4.4), the "raw" output will be identical to the regular output.

Example:

```
-r
```

Option: **-2**, --force-2d

Meaning: Force all output data to be 2D, even if the input data is 3D. This setting only affects the storage of the coordinate values in the output file(s). Internal processing is not affected by this setting.

Example:

```
-s
```

Option: **-c**, --strict

Meaning: Switches the parser into a "strict" mode in whch it will discard records without a geometry marker instead of interpreting them as single point measurements (see 4.4).

Example:

```
-c
```

Option: **-v**, --validate

Meaning: Performs a validation of the content of the parser schema (specified with option "-p"). The schema file's content will be checked for syntactical correctness and completeness, and any errors will be reported.

Example:

```
-p parser.txt -v
```

Option: **-e**, --english

Meaning: Sets all program messages and the input data's assumed numerical notation to English.

Example:

```
-e
```

Option: **-h**, --help

Meaning: Displays a help text with a synopsis of program usage and a list of options to the screen.

Example:

```
-h
```

Option: --proj-in=

Meaning: Specify *given* spatial reference system of *input* data (internal abbreviation, EPSG code or full PROJ.4 string). Note that PROJ.4 format specifications are *case-sensitive*. See section 8 for details.

Example:

```
--proj-in=epsg:31467
```

Option: --proj-out=

Meaning: Specify *intended* spatial reference system of *output* data (internal abbreviation, EPSG code or full PROJ.4 string). Note that PROJ.4 format specifications are *case-sensitive*. See section 8 for details.

Example:

```
--proj-out=epsg:wgs84
```

Option: --proj-dx=

Meaning: Set the *X* shift factor for a datum transformation to WGS 84 from the *intended* spatial reference system of the *output* data. See section 8.2.1 for details. The default is "0" which means "no *X* shift".

Example:

```
--proj-dx=598.1
```

Option: --proj-dy=

Meaning: Set the *Y* shift factor for a datum transformation to WGS 84 from the *intended* spatial reference system of the *output* data. See section 8.2.1 for details. The default is "0" which means "no *Y* shift".

Example:

```
--proj-dy=73.7
```

Option: --proj-dz=

Meaning: Set the *Z* shift factor for a datum transformation to WGS 84 from the *intended* spatial reference system of the *output* data. See section 8.2.1 for details. The default is "0" which means "no *Y* shift".

Example:

```
--proj-dz=418.2
```

Option: --proj-rx=

Meaning: Set the *X* rotation factor for a datum transformation to WGS 84 from the *intended* spatial reference system of the *output* data. See section 8.2.1 for details. The default is "0" which means "no *X* rotation".

Example:

```
--proj-rx=0.202
```

Option: --proj-ry=

Meaning: Set the *Y* rotation factor for a datum transformation to WGS 84 from the *intended* spatial reference system of the *output* data. See section 8.2.1 for details. The default is "0" which means "no *Y* rotation".

Example:

```
--proj-ry=0.045
```

Option: --proj-rz=

Meaning: Set the *Z* rotation factor for a datum transformation to WGS 84 from the *intended* spatial reference system of the *output* data. See section 8.2.1 for details. The default is "0" which means "no *Z* rotation".

Example:

```
--proj-rz=-2
```

Option: --proj-ds=

Meaning: Set the scaling factor for a datum transformation to WGS 84 from the *intended* spatial reference system of the *output* data. See section 8.2.1 for details. The default is "1" which means "no change in scale".

Example:

```
--proj-ds=6.7
```

Option: --proj-grid=

Meaning: Set the grid file to be used for a grid-based datum transformation to WGS 84 from the *intended* spatial reference system of the *output* data. See section 8.2.1 for details. The default is to not use any grid file.

Example:

```
--proj-grid=BETA2007.gsb
```

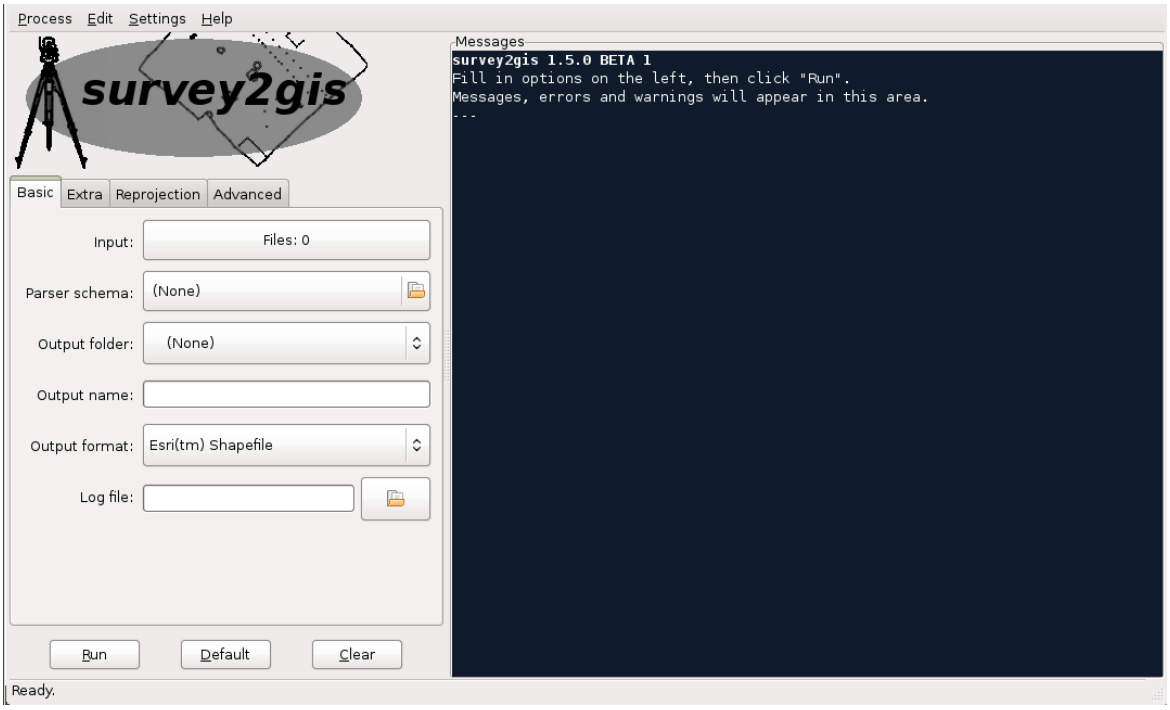
3 Graphical user interface

As an alternative to the command line usage, *Survey2GIS* can also be used via its built-in graphical user interface (GUI). There are two ways of starting the software in GUI mode:

1. Run "survey2gis" without any options: the GUI will start with default option settings.
2. Run "survey2gis" with the special option "--show-gui": the GUI will reflect any option settings given on the command line.

The GUI is divided vertically into two main areas (see below). On the left side of the GUI, you will find all options that are also available via the command line. There are four pages: The page "Basic" has all options for basic data processing; the page "Extra" affords access to additional functionality such as data selections (see 6) and labels (see 5); the page "Reprojection" allows to specify reprojection options (see 8); the page "Advanced" has options for fine-tuning the processing.

The screenshot below shows the GUI layout (GNU/Linux operating systems; graphical appearance will vary on different systems):



Once at least all required settings have been made on the "Basic" page, pressing the button "Run" will start the data processing. Any messages generated by *Survey2GIS* during data processing will be displayed in the message area on the right side of the GUI. Its content mirrors that of both the command line output and the (optional) log file.

Warnings are displayed in yellow. They do not lead to an interruption of data processing, but may contain important hints regarding input data problems.

Error messages are displayed in red and lead to termination of data processing, in which case not output data will be produced.

In order to avoid having to re-enter frequently used option values, the current settings can be saved into a file with extension ".s2g" using the "Settings ▷ Save/As" command from the main menu.

Only those option settings will be saved that have actually been specified, making it possible to save and restore partial sets of settings.

The menu item "Settings ▷ Defaults" (or the button with the same name at the bottom of the GUI) is used to restore settings from a file called "default.s2g" stored in the same folder aus *Survey2GIS* (provided such a file exists).

Warning: Any file present in the output folder will be *overwritten without warning* if a new file of the same name is produced. This is true for both output data and log files. Likewise, if a settings file is loaded then the current settings in the GUI will be replaced without warning.

3.1 Integrating the GUI into another application

The integration of the *Survey2GIS* GUI into a host application (e. g. a GIS) is assisted by *Survey2GIS* in two ways:

1. The host application can pass some settings to to *Survey2GIS* and then add the "--show-gui" option to show the GUI.
2. While *Survey2GIS* is running in GUI mode, it will output all its messages to the standard output stream with special markups. The host application can watch the output stream and react to the markup tags (see below).

If *Survey2GIS* has produced data, then the following tags will be present in the output stream:

<OUTPUT_FORMAT>*format*</OUTPUT_FORMAT>

... with "format" being the name of the output format produced, such as „Esri Shapefile".

This is followed by the names of the actual output files, separated by geometry type:

<OUTPUT_POINTS>*data*</OUTPUT_POINTS>
<OUTPUT_LINES>*data*</OUTPUT_LINES>
<OUTPUT_POLYGONS>*data*</OUTPUT_POLYGONS>

... with *data* being the file name of the respective dataset.

If the user has set the "--raw-data" option, then there will be an additional:

<OUTPUT_POINTS_RAW>*data*</OUTPUT_POINTS_RAW>

Since it is possible that the user has produced several different output formats and files during one session with *Survey2GIS*, the host application will always have to analyse the entire output stream.

4 The parser schema

The part of *Survey2GIS* that is responsible for making sense of the data in the input file (i. e. correctly *interpret* the file's contents) is called the "parser". The parser cannot automatically understand the structure of an input file. Instead, it must be supplied with a *parser schema*, which is a plain text file with a simple syntax that describes the structure of the input data. The user must create a valid parser schema file before running *Survey2GIS* on new input data. A given parser schema is then supplied to *Survey2GIS* using the *-p* (*Parser*) option (see 2.3). It can be used to process as many additional input files as required, so long as the input files all adhere to the structure specfied in the schema file. This section explains the contents of a valid parser schema file.

4.1 Syntax

To see the contents of this section in context, study the (non-working) illustrative parser description "parser.txt" in the subfolder "docs", as well as the (working) parser and data files in the subfolder "samples".

Lines that start with a "#" (hashmark) are comment lines. They will be ignored just like empty lines.

A valid parser schema must consist of two types of sections. Each section must in turn be declared using a key word in square brackets ("[]"). The two section types are called "[Parser]" and "[Field]". There must be exactly one section of type "[Parser]" that contains general settings for the parser. Following that, up to 251 sections of type "[Field]" may be added, each one describing the properties of one of the input data's elements/fields. The fields must be declared in the same order as they appear (from left to right) in the input data file(s).

Every section then contains further declarations that consist of "option name & value" pairs ("settings"), separated by an equal ("=") sign in the general form "*option=value*". Only one such pair is allowed per line of the parser schema, e. g.:

```
[Field]
Name=Index
```

The entire text to the right of "=" will be interpreted as the option's value. Empty values are not allowed. If an option is not to be set, then it should not be declared at all. Parser key words, option names and values within a schema are treated as *not case sensitive*. The names of options and their values may also be given using quotation marks. Additional whitespace characters will be ignored, so the following is just as valid as the above:

```
[Field]
‘‘name’’ = ‘‘Index’’
```

Option names must not contain any whitespace. Names that consist of more than one word may be spelled using the underscore ("_") character instead:

```
[Field]
Name = Index
empty_allowed = No
```

Some options, such as "empty_allowed" in the example above, are boolean values or "switches" that can only be toggled on or off (boolean "true" or "false"). The following are all equivalent values for turning an option "on": "Y", "Yes", "On", "1", "Enable", or "True". For turning an option "off", the following may be used: "N", "No", "Off", "0", "Disable", or "False".

A parser schema may be validated by running *Survey2GIS* with the options "-v" and "-p", followed by the name of the schema file. If the schema contains any errors, an informative message will be generated. If there are no errors then a summary description of the schema will be displayed.

The next section contains a discussion of all options that can or must be declared within the sections "[Parser]" and "[Field]".

4.2 Options for [Parser]

The following options (listed in alphabetical order) can ("optional") or must ("mandatory") be declared within the section "[Parser]". They control the general workings of the input data parser. The example file "parser.txt", contained in the subfolder "docs", contains a parser schema (not suitable for processing actual data) that illustrates all possible options with short comments.

Option: **comment_mark**

Meaning: Special characters that may be used to introduce comments in the input data. All text preceded by a comment mark will be skipped when processing the data.

Remarks: Optional. One or more comment marks may be specified. However, these must not include characters that are also used as field separators or quotation marks (see "quotation" and "separator" in 4.3).

Example:

```
comment_mark = #
comment_mark = //
```

Option: **coor_x, coor_y, coor_z**

Meaning: These three options specify the fields that hold the *X*, *Y* and *Z* coordinates of the measurements.

Remarks: The options „coor_x" und „coor_y" must always be specified and contain the names of two distinct, fully declared fields (see next sections. All fields must be numeric und of type "double". If "coor_z" is not specified, then *Z* coordinates will be assumed to be "0.0" for all measurements.

Example:

```
coor_x = xfield
coor_y = yfield
coor_z = zfield
```

Option: **geom_tag_point, geom_tag_line, geom_tag_poly**

Meaning: Specification of geometry markers ("tags") for the three basic geometry types "point", "line" and "polygon". These markers allow the parser to multiplex simple point measurements into more complex geometries. They must be part of the input data.

Remarks: The details of how these markers are used by the parser depend on the parser mode chosen (see 4.4). Setting "geom_tag_point" is optional for some parser modes. If it is not set, then *Survey2GIS* will try to recognize simple point measurements automatically. However, this will only work reliable if

certain requirements are met (see 12). For all parser modes except "None", both "geom_tag_line" and "geom_tag_poly" must be defined (and use different markers!) as part of the parser description. For parser mode "None", no geometry tag definitions need to be made; if any are found, they will be ignored. It is possible to declare more than one (up to 32) geometry tag for each geometry type. However, they must not have any characters (case matters) in common. There must also be no overlap in characters with field separators, quotation marks (see 4.3 for both) or comment marks (see "comment_mark", above). If the field used to store the tag is numeric, then all digits, plus ("+") and ("-") are also not allowed. Neither are the decimal point and thousands separator characters, if the field is of type "double".

Example:

```
geom_tag_point = "*"
geom_tag_line = "-"
geom_tag_poly = "@"
geom_tag_poly = "@_"
```

Option: **info**

Meaning: A short description of the parser schema.

Remarks: Optional.

Example:

```
info = This parser has been designed to test Survey2GIS
```

Option: **key_field**

Meaning: Name of the key field that contains the *same* value for all measurements that represent vertices of the *same* multi-vertex geometry (i. e. a line or polygon). If the parser is working in mode "End" or "Max", then all *successive* measurements with the same value in "key_field" will be merged into one geometry. Note that "successive" in this context means that measurements will be merged up to and including the next one that contains a line or polygon geometry marker (see "tag_field", below). The "key_unique" (see below) option can be used to merge multi-part geometries.

Remarks: Mandatory for parser modes „End" and „Max".

Example:

```
key_field = key
```

Option: **key_uniqe**

Meaning: Specifies whether the key field (see above) is unique across *disjunct* (multi-part) geometries.

Remarks: If this option is turned "on", *Survey2GIS* will assume that every geometry has a globally unique key value. This can be used to correctly process geometries that consist of disjunct parts (e. g. a set of spatially separate areas that should be considered parts of one polygon). In this case, all geometries with the same value in "key_field" (see above) will be stored as one multi-part geometry in the output file, with only one row of attribute data. If this option is not specified, then *Survey2GIS* cannot produce multi-part geometries and the user must instead manually assemble them in the GIS.

Example:

```
key_unique = yes
```

Option: **name**

Meaning: Parser name.

Remarks: Optional.

Example:

```
name = Survey2GIS test parser
```

Option: **no_data**

Meaning: An *integer* number that represents attribute values which are "null" (="no data"). This should be set to a value that never occurs in valid data. The chosen value will be written into the output data whenever a field actually contains no data (i. e. is empty; see "empty_allowed" in 4.3).

Remarks: Optional. See descriptions of output formats in 10 for details on how "no data" is set when there is no user-defined value for it. See also "empty_allowed" (above).

Example:

```
no_data = -99999
```

Option: **tag_field**

Meaning: Name of the field that contains the geometry markers (see 4.4). May be identical with "key_field" (see above), *except* in parser mode "Max". Must contain the name of field declared within a "[Field]" section (see 4.3).

Remarks: Must be specified for all parser modes except „None". To reduce typing work in the field, geometry markers may be part of any other attribute field (see 4.4 for details). In this case, however, the attribute values must not overlap with the characters used as geometry markers. Coordinate fields can also

not be used for this purpose (see "coor_x", "coor_y" and "coor_z", above). Note that in parser mode "Min" it will usually make sense for "tag_field" to be a separate field, since there will be only one record (line) with complete attribute data.

Example:

```
tag_field = tag
```

Option: **tagging_mode**

Meaning: Determines the principal working mode of the parser (*parser mode*). Possible settings are „min", „max", „end" or „none". Further explanations are given in 4.4.

Remarks: Mandatory.

Example:

```
tagging_mode = end
```

Option: **tag_strict**

Meaning: If this option is "on", the parser will be *strict* about geometry markers. This means that every measurement (i. e. line in the input file) must contain a geometry marker. Measurements without one will be discarded.

Remarks: Optional. Turned "off" if not specified.

Example:

```
tag_strict = yes
```

4.3 Options for [Field]

The general parser settings (see preceding section) must be followed by two or more (up to 251) field definitions, each one introduced by a

```
[Field]
```

line. In addition to coordinates, individual fields may represent any kind of attribute data recorded during the survey. For the automated reconstruction of complex line strings and polygons, the parser schema must contain at least one additional field that contains the geometry markers and key values (see descriptions of "key_field" and "tag_field" in preceding section).

All attribute data will be stored as part of the output file(s), provided that this is supported by the chosen output file format. In the case of Shapefiles, the attribute data will be stored as a *DBase* file with extension ".dbf". Further restrictions may apply to the definition of attribute fields, depending on the capabilities of specific output formats (see 10 for details).

Option: **change_case**

Meaning: This option may be set to convert the contents of a text field to all upper or lower case characters after reading the input data. Possible values are "lower" (convert to lower case) and "upper" (convert to upper case) or "none" (no conversion; this is the default).

Remarks: Optional. Only applies to fields of type "text".

Example:

```
change_case = lower
```

Option: **empty_allowed**

Meaning: Indicates whether a field is allowed to have empty content. E. g. this will be the case when two field separator characters succeed each other immediately or with only whitespace between them. In that case, a "no data" (see option "no_data", below) value will be stored if "empty_allowed" is on. Otherwise, the entire measurement (line of input data) will be considered invalid and will be discarded.

Remarks: Optional. This option is "off" by default. It cannot be turned "on" for fields that are declared to be "key_field" or "tag_field" or if "merge_separators" (see below) is "on".

Important: If the corresponding field's separator is set to a whitespace character ("tab" or "space"), then empty fields cannot be processed correctly: Since whitespace to the left or right of a field's contents ("padding") is automatically removed, the parser cannot distinguish between empty fields and field separators in this extreme case.

Example:

```
empty_allowed = yes
```

Option: **info**

Meaning: One-line description of the field's content.

Remarks: Optional (but highly recommended). For purposes of documentation.

Example:

```
info = Field with unique object index.
```

Option: **merge_separators**

Meaning: Specifies whether two adjacent field separators should be treated as one separator.

Rermarks: Optional. If turned "off" (default) then two adjacent separators will be interpreted as empty field content (see "empty_allowed" above).

Example:

```
merge_separators = yes
```

Option: **name**

Meaning: The name of the field, with a maximum length of 10 characters. Valid characters for field names are all letters of the Latin alphabet, the underscore ("_") and the digits "0" to "9". There is no distinction between upper and lower case spelling. All field names will be stored in all *lower* case automatically (note: correctly enforced since version 1.4.0).

Remarks: Mandatory. Every field must have a valid name.

Example:

```
name = index_fld
```

Option: **persistent**

Meaning: This option is only meaningful in parser mode "Min" (s. 4.4). If it is turned "on" for a field, then the parser expects a field of this type to be present in every line of the input file, even though there may be no valid attribute data. This option can be useful when processing with superfluous data. E. g. some software writes additional data into each line when downloading data from a device.

Remarks: Optional. Coordinate fields are always assumed to be persistent in parser mode „Min".

Example:

```
persistent = yes
```

Option: **quotation**

Meaning: If the content of a text field is surrounded by quotation marks and the marks themselves should not be saved as part of the field content, then the quotation mark character used must be specified with this option.

Remarks: Optional. Only one quotation mark character per field may be declared. This must not be a character that is also used as field separator (see "separator", below) or comment mark (see "comment_mark" in 4.2). Applies only to fields of type "text".

Example:

```
quotation = '
```

Option: **skip**

Meaning: When "skip" is turned "on" for a field, then the field will be processed by the parser but it will not be stored as part of the output. This is useful for fields that are part of the input data structure but are of no importance for further data processing.

Remarks: Optional. The default is store every defined field in the output file(s).

Example:

```
skip = yes
```

Option: **separator**

Meaning: It is possible to specify one or more separator characters that indicate the end of one field's content and the beginning of the next. Field separators may also include more than one character. The special declarations "tab" and "space" are used to represent tabulator and space characters, respectively.

Remarks: Must be specified for every field except the last one. Also note the option "merge_separators" (above).

Example:

```
separator = ;
separator = tab
separator = space
```

Option: **type**

Meaning: This option defines the data type of the field. The available choices are "integer" (for whole numbers: -1, 0, 100, ...), "double" (for real numbers in floating point notation: -10,05, 0.0, 1200,12, ...) and "text" (for generic text). Note that some output format will severly limit the length of text fields (see corresponding entry in 10). Correct processing of floating point data requires correct settings for decimal point and thousands separator characters (cf. 2.2).

Remarks: Must be specified for every field.

Example:

```
type = integer
```

Option: **unique**

Meaning: If this option is turned "on", then the parser will check whether the field's contents are unique *per geometry*. If that is not the case then *Survey2GIS* will issue a warning during processing. However, the affected geometries will not be modified or discarded. It is then up to the user to decide how to handle duplicate attribute values.

Remarks: Optional. Turned "off" by default. Uniqueness is checked across all input file(s).

Example:

```
unique = yes
```

Option: **value**

Meaning: The „value" option allows the user to add a "pseudo field" to the output data that is not actually present in the input data. The field will have a constant, user-defined value for all attribute table records in the output data. The field's definition is a simplified one: Only the options „info", „name", „type" and „value" will be processed. Providing any other options will produce an error message.

Remarks: The field's data type (see option „type") must match the specified "value". In case of floating point values, make sure that the language settings (see 2.2) match the value's notation.

Example (for a field of type "double"):

```
value = 1000.05
```

Option: **@** (replace)

Meaning: The "@" character signifies replacement of text field contents. This option allows the user to replace repetitive, short text (codes, abbreviations) with longer, more descriptive text for storage in the attribute table. The syntax for the replacement operation is:

```
@old=new
```

This replaces the text "old" with "new", whenever the former appears in the content of the text field *and* constitutes the *entire* contents of the field. Partial replacements will not be performed. It is possible to specify more than one replacement pair per text field. There must be no whitespace or other characters between "@" and "old".

Remarks: This option is only available for fields of type "text". The comparison of old and new field content is *not* case sensitive. The new text must not be longer than 254 ASCII characters.

Example (for a field of type „text"):

```
@ABB = Abbreviation
@P1 = Point of type 1
...
```

4.4 Geometry markers and parser modes

In order to reconstruct complex geometries, such as line strings and polygons, from the survey data, the parser needs to understand how to interleave the individual measurements that represent the vertices of complex geometries.

There are some fundamental limitations to what can be achieved:

- Even though *Survey2GIS* is capable of processing any number of input data files, it is not possible to spread vertex measurements belonging to one geometry across more than one file. However, it is possible to store the vertices of separate parts of a multi-part geometry (s. description of parser option "key_unique") in separate files.
- The list of consecutive vertex measurements within one input file may only be interrupted by comment lines or empty line. Regarding the surveying work in the field, It is thus not possible to interrupt the surveying of one feature, start and complete another feature and later finish surveying the first. Where such a working mode is unavoidable, the resulting data file must be re-ordered manually before it can be processed by *Survey2GIS*.
- The only available geometry types are "point", "line" and "polygon". Rounded shapes cannot be approximated by *Survey2GIS*. This must be done in a separate post-processing step, using external software.

Note: Further hints on the processing of complex geometries, such as polygons with holes or multi-part geometries, can be found in the section on "topological cleaning" (9).

Every geometry type is indicated to the parser via a freely definable geometry marker (see 4.2). Candidate characters for markers are all characters that are not normally used in attribute table field content, such as "@" or "$". When surveying, these markers are keyed into the device by the surveyor and stored along with the associated coordinate measurements. They may be inserted as separate data fields or at the end of an existing field (depending on the parser mode chosen).

The field that contains the geometry markers is called the "tag field" and must be declared as part of the parser schema (see option "tag_field" in 4.2). If there is no such declaration, then *Survey2GIS* will interpret every measurement/line of input data as an individual point measurement. In this case it will be impossible to build lines or polygons automatically from the input data.

In some parser modes, the value of a key field (option "key_field") is used to indicate to the parser which measurements to interleave into one geometry. The key field's content is read in a case-sensitive manner, so that "1a" and "1A" would represent two different keys. The field option "change_case" may be used to alter this behaviour.

The final measurement/vertex of a polygon does not have to be identical with the first one, i e. duplicate measurements need not be taken. Polygons will be closed automatically by *Survey2GIS*. In fact, such a duplicate vertex is considered a topological error that will be fixed automatically by *Survey2GIS* (see 9 for details).

Depending on the nature of field work, it may be useful to apply different strategies for marking geometries. For this reason, *Survey2GIS* provides alternative processing methods, called "parser modes". These can be set in the parser schema using the "tagging_mode" option:

```
[Parser]
...
tagging_mode = min
...
```

Only one mode may be specified per parser scheme. This means that *all* input data will be processed using the same mode. Possible modes are „Min", „Max", „End" and „None".

Note: While it is possible to use any text type attribute field for storing an additional geometry tag (and even to declare the same field as "tag field" and "key field" in some parser modes: see details below), this is not recommended practice. It is better to use a separate, dedicated geometry tag field. Nevertheless, if geometry markers are stored as part of other field content, then the geometry marker *must* appear either as the first or last character of the field content.

4.4.1 „Min": minimal coding work

This mode is selected by default if the user does not specify a different "tagging_mode" in the parser schema. It is useful for reducing the coding (attribute data entry) work during surveying. In this mode, the first measurement of a line or polygon *must* be tagged with the appropriate geometry marker.

The appearance of a geometry marker in the input data indicates the start of a list of vertex measurements. Only the first vertex measurement includes attribute data on the same line. All following measurements/lines are stored as *reduced records* that contain only coordinate values and unavoidable additional data that may have been inserted by the software responsible for producing the input data file(s). If such additional data exists, then the user *must* create matching field definitions and set the "persistent" option (see field options in 4.3), so that the parser will be able to separate actual coordinate data from "noise". It is not necessary to set a "key_field" in this mode, unless the data contains multi-part geometries.

This is an excerpt of input data fit for processing in mode "Min":

```
# Sample file structure for a polygon (square) in mode „Min''.
# Hashmark introduces a comment line that will not be parsed.
# Fields: index, label, geometry tag (@=polygon).
1  polygon_1  @  10.00  10.00  1.00
# The first line had all the attributes and the geometry tag.
# The following lines will only have the x/y/z vertex coordinates.
10.00  20.00  1.05
20.00  20.00  1.10
```

```
20.00  10.00  1.00
# The next occurence of a full record terminates the polygon.
2  polygon_2  @  30.00  30.00  2.50
...
```

To maintain consistency, persistent and coordinate fields have to appear in the same sequence in which they were defined, both in the first (attributed) record and in the following (reduced records). E. g., assuming that the second field ("label") in the example above has been set to be "persistent", then a valid record structure might look like this:

```
1  polygon_1  @  10.00  10.00  1.00
polygon_1  10.00  20.00  1.05
polygon_1  20.00  20.00  1.10
polygon_1  20.00  10.00  1.00
```

The minimum number of fields, $Min(f)$, in a valid reduced record in mode "Min" is therefore:

$Min(f) = number\,of\,coordinate\,fields\,(2\,or\,3) + number\,of\,persistent\,fields$".

4.4.2 „Max": maximal control

In mode "Max", each measurement *must* be tagged with a geometry marker (this also means that the parser schema must define a marker for each geometry type). This mode is intended to allow tightly controlled processing of fully edited data. Suitable data may be created using a text editor or other means and *Survey2GIS* may be used to generate consistent GIS data from it. This mode may also be useful for processing data records exported from a database management system.

A "tag_field" as well as a"key_field" must be defined as part of the parser schema. They must be separate fields. In principle, it is enough for the value of "key_field" to change between successive geometries. For the benefit of data integrity, however, "key_field" should be unique per geometry across all input data.

This is an excerpt of input data, suitable for processing in mode "Max":

```
# Sample file structure for a polygon (square) in mode „Max''.
# Hashmark introduces a comment line that will not be parsed.
# Fields: index, label, geometry tag (@=polygon).
# The second field (label) is the key field.
1  polygon_1  @  10.00  10.00  1.00
2  polygon_1  @  10.00  20.00  1.05
3  polygon_1  @  20.00  20.00  1.10
4  polygon_1  @  20.00  10.00  1.00
```

```
# A change in the value of the key field terminates the polygon.
5  point_1  @  35.00  32.00  2.60
# A change in the value of the key field terminates the first point.
6  point_2  @  33.00  32.00  2.40
# etc.
...
```

Note: The vertices of polygons and lines encoded in mode "Max" must be recorded in *direct succession*. It is not possible to e. g. intersperse single point measurements between the vertices of a polygon!

4.4.3 „End": flexible surveying

In this mode, every measurement of a line or polygon *must* be *concluded* with a geometry marker. The marker must be part of the final vertex measurement.

The advantage of mode "End" is that the decision whether to record a geometry as a polygon or line string must only be taken with the final measurement. In order to allow the parser to collect all vertices that belong to a geometry, working backward from the marked one, all related measurements must have an identical value in the "key_field". This may cause additional typing work during the survey.

Both a "key_field" and a "tag_field" *must* be defined as part of the parser schema in mode "End". However, both may be the same field. The geometry markers will not be stored as part of the output data. In principle, it is enough for the value of "key_field" to be identical within on geometry, as the geometry tag will reliably indicate the end of each geometry. For the benefit of data integrity, however, "key_field" should be unique per geometry across all input data.

This is an excerpt of input data, suitable for processing in mode "End":

```
# Sample file structure for a polygon (square) in mode „End''.
# Hashmark introduces a comment line that will not be parsed.
# Fields: index, label with embedded geometry tag (@=polygon).
# The second field (label) is also the key field.
1  polygon_1  10.00  10.00  1.00
2  polygon_1  10.00  20.00  1.05
3  polygon_1  20.00  20.00  1.10
4  polygon_1@ 20.00  10.00  1.00
# After the occurence of „@'' (geometry tag), another object follows.
5  polygon_2  30.00  30.00  2.50
...
```

4.4.4 „None": points-only surveys

In mode "None", there will be no marking of geometries. This mode is suitable for surveys that include only simple point measurements, such as the work done to survey an elevation model.

By contrast, in modes "Min" and "Max", the geometry markers *may* be left out (unless option "tag_strict" has been turned on), in which case measurements are automatically assumed to represent simple points.

Note that mode "None" also allows *Survey2GIS* to be used as a flexible tool for converting simple point lists to GIS datasets.

This is an excerpt of input data, suitable for processing in mode "None":

```
# Sample file structure for a polygon (square) in mode „None''.
# Hashmark introduces a comment line that will not be parsed.
# Fields: index, X, Y, Z.
1  10.00  10.00  1.00
2  10.00  20.00  1.05
3  20.00  20.00  1.10
4  20.00  10.00  1.00
5  30.00  30.00  2.50
...
```

5 Labels

In addition to producing output geometries for the measured geometries, it is also possible to output additional point data with *labels* (also called "annotations"). Label points are 2D points (Z data is not considered when computing positions of label points) with special attribute fields. The actual data format and attribute table schema used by the labels depends on the chosen output format. See section 10 for details.

Warning: File formats that produce additional files for labels (e. g. Shapefiles: see 10.1) will overwrite existing data. Since they also use auto-naming, this may result in unexpected consequences. Auto-naming will add "_labels" to the user-specified output base name. Check to make sure that this does not collide with any existing files that you wish to keep!

Producing a labels layer requires the specification of a field in the input data. The contents of this field will be used as label text for the generated label points. Note that it is possible to choose a label text field that has been marked "skip=true" in the parser (see 4.3). The label field can be of type text, integer or double. Its contents will be copied verbatim into the label texts (except that geometry markers will be removed, if present).

In addition, the user has control over where the label points are placed on the associated geometries: The possible placing modes are: "center" (default), "first", "last" and "none". The effect of these modes depends on the type of geometry:

- Point geometries: Placement mode is ignored, one label point is placed exactly on each point. No label points will ever be produced for raw vertex data (see option "--raw-data" in section 2).
- Line geometries: In mode "center", one label point will be placed exactly halfway along the path of each line (string); in mode "first", the label point will be placed on the first vertex of each line and in mode "last" it will be placed on the last vertex. In all three mode, label points are guaranteed to lie on the line geometries.
- Polygon geometries: In mode "center", one label point will be placed in the geometric center of each polygon; in mode "first", the label point will be placed on the first vertex of each polygon and in mode "last" it will be placed on the last vertex. When using mode "center" with polygons that contain holes or are strongly concave, it cannot be guaranteed that label points will actually lie within the polygon areas.

Mode "none" has the same effect on all geometry types and will suppress the generation of label points for the respective type.

In the case of multi-part geometries, one label point will be generated for *each* part.

Note: If the label field contains geometry tags *in addition* to other content, then *Survey2GIS* will attempt to remove all geometry tags from the output labels. This can fail in some scenarios. To ensure that labels are free of geometry tags, use a separate, dedicated field to store them (see parser definition of "tag_field" in 4.2).

6 Data selections

Sometimes it may be desirable to reduce the data to a *selected subset* of records, e. g. for separate processing and export of smaller areas within a larger survey region. For this purpose, it is possible to specify one or more *selection commands* on the command line using the "-S" option (see 6.1 for details):

```
-S seltype:geomtype:field:expression -S seltype:geomtype:field:expression [-S ...]
```

The effect of this is that only those records, whose *attribute table contents and geometry types* pass one or more selection criteria (*expressions*), will be retained in the output. The selection will take place immediately after all geometries have been assembled and before any further processing, such as topological cleaning (see 9). Selection commands can be applied to some or all geometry types (points, lines, polygons).

Complex geometries (lines and polygons) can consist of many original point records (vertices), but only one attribute table record, once assembled. Selections on the attributes of assembled geometries cannot be guaranteed to match original vertex values. E. g. the coordinate values of the individual vertices are not preserved (except for the first one's) in the attribute record of a polygon or line, so selecting a singular value of the "Z" coordinate may fail to produce the intended matches. In such cases, selecting a range of values or using a regular expression (see 6.2 for details on both) will often give better results.

6.1 Selection command syntax

Each selection command has a fixed syntax that consists of up to *four tokens*, named "seltype", "geomtype", "field" and "expression", separated by a ":" (colon), plus one or two *optional modifiers* ("*" and/or "+" or "-"; items in square brackets are not required in all cases):

```
[*]seltype[+|-]:geomtype[:field:expression]
```

1. The first token defines the *type of selection* (see 6.2) to apply, optionally including modifiers to invert ("*"), add to ("+") or subtract from ("-") an existing selection (see 6.3 for details).

2. The second token stands for the *geometry type* to include in the selection. This can be "point", "raw", "line", "poly" or "all".

3. The third token is the *name* of the *field* (see 4.3) to which the expression (see below) applies. Note that field names are treated as case-insensitive, meaning that e. g. "feature" and "FEATURE" both refer to one and the same field (see item "name" under 4.3).

4. The fourth token is the selection *expression*. In the most simple case, this expression is the text to which the field's contents should be compared. More complex expressions arise in the case of regular expression type selections (see 6.2).

The third and fourth tokens must *not* be specified if the selection type is set to "all" (i. e. select all records regardless of attribute field content).

When using text expansion/replacement on a field's content (see explanation of the "@" operator in 4.3), the selection expression will be applied to the expanded/replaced content, *not* the original data.

In order to avoid problems *on the command line* with expressions that contain spaces, the entire filter argument should be surrounded with quotation marks (see examples in this section). This is not an issue when using the graphical user interface.

6.2 Selection types

The following comparative selection types can be applied to attribute fields of all types (text and numeric). Comparisons are done against the text in the "expression" token:

eq The "equal" type (*eq*) selects only those records whose attribute field contents are equal to the given expression.

neq The "not equal" (*neq*) type selects only those records whose attribute field contents are *not* equal to the given expression.

lt The "less than" (*lt*) type selects only those records whose field contents are "less than" the given expression. In the case of numeric fields, the meaning of this is intuitive. In the case of text fields, the comparison is done lexicographically, i. e. both field contents are sorted as in a dictionary, and the one that comes first in the sorted order is "less than" the other.

gt The "greater than" (*gt*) type selects only those records whose field contents are "greater than" the given expression. In the case of numeric fields, the meaning of this is intuitive. In the case of text fields, the comparison is done lexicographically, i. e. both field contents are sorted as in a dictionary, and the one that comes second in the sorted order is "greater than" the other.

lte The "less than or equal" (*lte*) type works in the same way as the "less than" type (see above), but it also selects those records whose field contents are "equal" to the given expression.

gte The "greater than or equal" (*gte*) type works in the same way as the "greater than" type (see above), but it also selects those records whose field contents are "equal" to the given expression.

Examples (note the use of quotation marks *on the command line* to prevent problems with spaces in the expression token):

```
-S "eq:poly:objname:boundary"
-S "eq:poly:objname:boundary type A"
-S "gte:all:elevation:12.00"
```

There is one selection type that can only be applied to numeric fields:

range Selects only those records whose numeric field contents fall within a "minimum to maximum" (inclusive) range specified by the user. When applying a range selection, the "expression" token must be of the format "min;max", e. g.:

```
-S "range:point:elevation:100.0;120.0"
```

There are also some selection types that can *only* be applied to fields of type *text*:

sub Selects only those records whose field contents contain the expression as a *subset*. The example below selects "boundary", "Boundary", "boundaries", "closed boundary", etc.:

```
-S "sub:poly:objname:oundar"
```

regexp selects all records that match a *regular expression* (https://en.wikipedia.org/wiki/Regular_expression). The set of supported operators is listed below (See https://github.com/cesanta/slre):

```
^       Match beginning of field content.
$       Match end of field content.
()      Grouping and substring capturing
\s      Match whitespace.
\S      Match non-whitespace.
\d      Match decimal digit.
\n      Match new line character.
\r      Match line feed character.
\f      Match form feed character.
\v      Match vertical tab character.
\t      Match horizontal tab character.
\b      Match backspace character.
+       Match one or more times (greedy).
+?      Match one or more times (non-greedy).
*       Match zero or more times (greedy).
*?      Match zero or more times (non-greedy).
?       Match zero or once (non-greedy).
x|y     Match x or y (alternation operator).
\meta   Match one of the meta character: ^$().[]*+?|\.
\xHH    Match byte with hex value 0xHH, e.g. \x4a.
[...]   Match any character from set. Ranges like [a-z] are supported.
[^...]  Match any character but ones from set.
```

Example:

```
-S "regexp:all:objname:type 1|type 2"
```

The above selects all records that have either "type 1" or "type 2" as their "objname".

Regular expressions in *Survey2GIS* do *not* support Unicode characters.

Finally, there is one special selection type:

all An "all" type selection matches, as the name implies, all records, regardless of attribute field types and contents. Its purpose is to allow selecting geometry types without filtering by attribute values. In this case, neither the "field" nor the "expression" token must be specified. E. g. to select all "line" type records:

```
-S "all:line"
```

6.3 Selection modifiers and chains

It is possible to *invert* the effect of a selection command (including both geometry and selection type specifications) by prefixing the type token with a "*" (asterisk). E. g. the following selects those records that do *not* have the content "boundary" in their "objname" field:

```
-S "*eq:all:objname:boundary"
```

Each selection type also has an alternative name, spelled in *capital* letters:

```
EQ,LT,GT,LTE,GTE,SUB,REGEXP,RANGE,ALL
```

Using these versions on text type fields results in a *case insensitive* comparison. This has the same effect as converting both the field's content and the expression to all upper case letters before comparing them. E. g.:

```
-S "EQ:poly:objname:Boundary"
-S "EQ:poly:objname:boundary"
-S "EQ:poly:objname:BOUNDARY"
```

... will all match "Boundary", "BoUNDary", etc.

When selecting from numeric type fields, there will be no difference between the results of either version. The same is true for "ALL".

It is possible to create a *selection chain* by applying several selection commands to one or several fields. This will successively refine the selection result and is achieved using the "+" (add to current selection result) and "-" (subtract from current selection result) modifiers, which have to be suffixed to the selection type token.

In summary, the following rules apply:

1. Initially, all records are part of the selection (this is the equivalent of an "all:all" selection).
2. One or more selection commands can be specified, they will be processed in order of specification.
3. The modifier "+" adds the result of the selection command (matched records) to the current selection.
4. The modifier "-" subtracts the result of the selection command (matched records) from the current selection.
5. If neither modifier is given, then the current selection will be replaced with the result (matched records) of the selection command.
6. After all selection commands have been applied, only those records that are still part of the selection will appear in the output file(s).

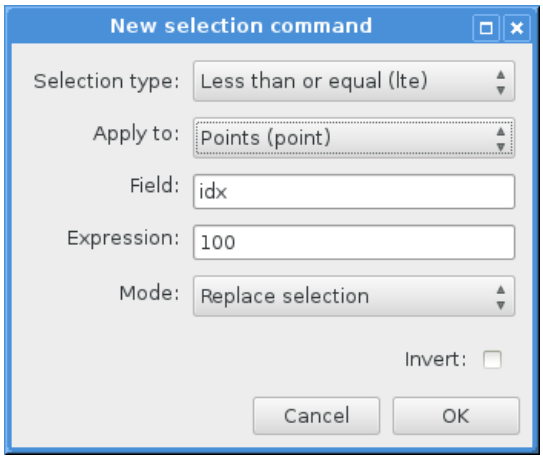
The selection commands will be applied and records added or subtracted from the selection, in the same order as they are specified by the user. E. g. the example sequence below (enter all three "-S" options on one command line) will first select all records for which the field "objname" (object name) contains "oundar", such as "Boundary"; the second command will then remove all records from the selection, for which the field "objname" is set to "closed boundary"; the third command will leave only those records in the final selection result, for which the field "objtype" is *not* set to "breakline" (all expressions are not case sensitive).

```
-S "SUB:poly:objname:oundar"
-S "EQ-:poly:objname:closed boundary"
-S "EQ-:poly:objtype:breakline"
```

Details about each selection command's effect on the data will be displayed in the form of status messages during processing.

6.4 The GUI's selection command editor

It is also possible to compose selection commands using the graphical user interface (GUI, see 3). To do so, click on the button labeled "Selection:". This will produce a list of currently defined selection commands. Click "Add" to define a new selection command or "Edit" to modify an existing one. In both cases, a form similar to the one pictured below will appear.



Selection commands can be constructed by filling in all required fields of the form. In the case of selection type "all", any input for "Field:" and "Expression" will be ignored and discarded once the form is closed with a click on "OK". Note that pressing "OK" will also trigger a syntax check of the entered selection command. This check only validates the formal correctness of the command. It does neither check whether the command will produce a non-empty selection, nor does it check for valid field names and types. Because the latter are completely data-dependent, they can only be checked once the program is run. Therefore, the status messages of *Survey2GIS* should be watched carefully to verify selection results.

Selection commands will be processed in the same order (top to bottom) in which they appear in the list of selections.

Saving the current GUI settings into a ".s2g" settings file will also save all currently defined selection commands.

Technical note: The "|" (pipe) character is used to represent the logical "or" operator in regular expressions. At the same time, it is used as a list separator in *Survey2GIS* settings files (see 3). To avoid conflicts when loading/saving regular expressions from/to settings files and at the same time retain compatibility with older versions of *Survey2GIS*, the "|" character will be stored as ASCII character #30 (RS, record separator) in settings files. This must be taken into account, should the need arise to edit settings files that contain regular expressions outside of *Survey2GIS*.

7 Orientation modes

Starting with version 1.5.0, *Survey2GIS* provides the concept of "orientation modes" as a work-around for handling vertical data ("catenas" in soil science, "profiles" in archaeology, etc.) in a traditional 2D/pseudo 3D GIS environment.

Processing 3D data is more complex than just passing through additional Z coordinate data. GIS concepts and operations, such as topology and georeferencing, do not easily translate into the third dimension.

Thus, survey data that include a strong vertical component (cross-sections of soil volumes, standing walls, etc.) are not easy to integrate into a 2D GIS workflow. *Survey2GIS* addresses this problem by providing options to *re-orientate* the input data into a new, synthetic 2D coordinate system. The resulting data can be loaded into a 2D GIS and treated as though it were regular X-Y planar 2D data. Such a transformation from 3D to 2D always incurs a loss of information.

By default, the orientation of the output data produced by *Survey2GIS* is identical to that of the input data (setting option "--orientation" to "world"), i. e. no re-orientation is performed (7.1).

If requested by the user, re-orientation of axes is performed immediately after an input file has been read and *before* further geometry processing, in particular topological cleaning, is applied (see 9). This allows operations such as snapping to work on vertical data as expected. The coordinates of label points (if present: see 5) will also be transformed.

The program option "--orientation=" (see 2.3) can be used to re-orientate the data as described below. Note that not all orientation modes may be available for all types of input data.

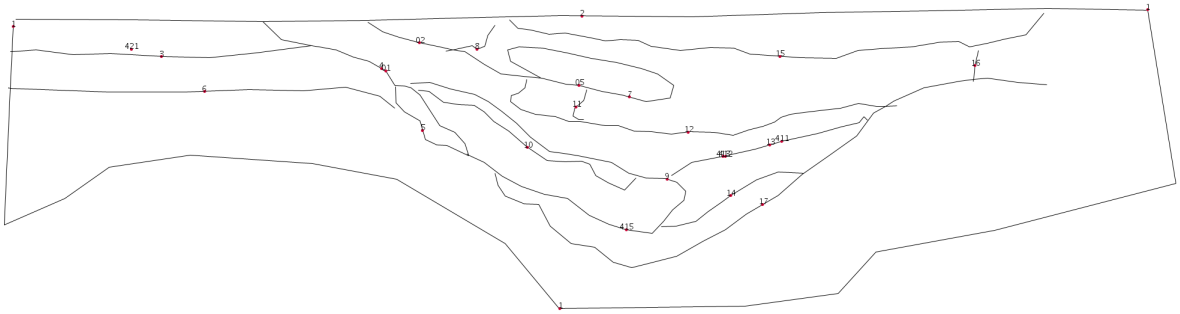
Warning: Data produced in an orientation mode other than "world" will be subject to transformations with *no guarantee of reversibility*. Therefore, all survey raw data should be archived and re-orientated data should never be considered the primary or authoritative processing result.

7.1 World orientation (default)

Orientation mode "world" is the default mode of operation in *Survey2GIS*. It does not modify the input data in any way. The corresponding option value exists merely to keep the user interface consistent.

7.2 Local X-Z orientation

The "local X-Z" orientation mode (option value "localxz") is a simple means of transforming 3D survey data to 2D cross-sections ("profiles") that can be visualized in a 2D GIS using a synthetic coordinate system. It is not suitable for large areas with many disjunct geometries, but intended to visualize a single "profile" (alternative terms: "cut", "trench wall", "catena", etc.):



The re-orientated data will have a new, synthetic coordinate system with swapped Z and Y axes and newly computed X coordinates. The fact that the transformed data's X coordinates will always start at "0" while the Z coordinates are preserved, makes it relatively easy to compare different datasets side-by-side (provided that the GIS can display several data views at once).

This transformation method is non-parametric and fully automatic. It does not require the definition of a projection plane.

The transformed data will have the following properties:

- The new Y coordinates will be the same as the original Z data.
- The coordinates of the synthetic X axis will no longer represent any original measurements.
- The smallest X coordinate value found in the input data will become the origin ($X = 0$) of the new, synthetic X axis.
- The largest X coordinate on the synthetic X axis will be identical with the greatest difference in X values between the input measurements.
- All Y coordinate data are discarded: user-defined offsets on any axis are also discarded.

This transformation method has a number of limitations:

- Input data must be 3D cartesian data. Lat/lon and 2D data cannot be transformed using this method.
- Input data should only cover a single profile's extent.
- All input data must represent points that were measured from just one location. By extension, this means that complex 3D structures with strong concavity ("recesses") will not produce satisfactory results.
- Data with strong variation along the profile's major axis (i. e. on the X-Y plane) will be strongly distorted
- Data with "local X-Z" orientation cannot be *reprojected* to another spatial reference system (see 8).

- Data that has a significantly larger extent on the *Z* axis than on the *X*-*Y* plane will more likely produce wrongly orientated results (see description of process, below).
- It is not possible to transform data from more than one data source (input file) in one program run.

Results will be as accurate as possible if all points/vertices between the first and the last measurement lie on a straight line when viewed in the *X*-*Y* plane. The more they deviate from this ideal assumption, the more distorted the result will be. Likewise, transformed profiles will be easier to interpret if the original *Z* coordinates are "reasonably orthogonal" to the *X*-*Y* plane (because the new *Y* axis retains the geographic alignment of the original *Z* axis).

The transformation works as follows:

1. Set reference point *RP* and determine profile orientation (see below).
2. For all input points/vertices ($p_{i=1,2,\ldots,n}$) in input dataset P:
   (a) Compute *X* coordinate for transformed version of p_i as straight-line distance between *X*/*Y* location of p_i and *RP*.
   (b) Set *Y* for transformed p_i to value of original *Z* coordinate.
   (c) Set new *Z* coordinate to be "0".

In order to produce usable results, two key parameters will be derived automatically from the data:

1. The overall *W*-*E* orientation of the profile, i. e. the question of where the left edge of the profile will be in the *X*-*Z* coordinate system. If this fails, then the transformed data will appear "mirrored".
2. The identity of the most "extreme" surveyed point with the "left-most" coordinate in the synthetic *X*-*Z* coordinate system. If the wrong point is identified, the left edge of the transformed profile will appear distorted.

The first parameter, "profile orientation" is determined by computing the weighted sum of differences between the *X* coordinate of a *reference point* and those of all other points. All *X* coordinate differences are inversely weighted by the difference in *Z* between them and the reference point. The effect of this is that measurements further removed on the *Z* axis from the reference points have less of an effect on the result, thereby compensating for the practical problem that the true "left" edge of the surveyed profile might receed significantly in comparison to the first measured point. If the weighted sum of *X* differences is positive, then the profile is determined to have "eastward" orientation, "westward" otherwise. Note that the latter are not accurate descriptions of the profile's orientation in the geographic *X*-*Y* plane. They merely serve to determine where the "left" and "right" edges of the transformed data will be in the synthetic *X*-*Z* coordinate system.

In the extreme case that all *X* coordinate differences sum to exactly "0" (e. g. this can theoretically happen if all measured points lie on a straight line parallel to the *S*-*N* geographic axis), profile orientation cannot be decided automatically and will be assumed to be "eastward orientation".

In practice, then, everything hinges on choosing a suitable reference point, which the program assumes to be the *first point* of the input data. This is more likely to produce accurate results if one keeps the following in mind when conducting the profile survey:

1. Measurements should start at or near the upper-left corner (as viewed from the surveying station) of the profile.
2. Successive measurements should generally proceed toward the right and bottom, ending at or near the lower-right corner of the profile.

The above assumptions do not have to be met fully. Given a sufficient number of measurements, the program will be able to compensate for e. g. a reference point that is not identical with the true top-left point in the input data.

Survey2GIS will provide information on the chosen reference point, profile orientation and other key parameters as part of its logging output.

8 Spatial reference systems and reprojections

Support for spatial reference systems and reprojections has been introduced in version 1.5.0 of *Survey2GIS*. This necessitated some significant changes in the user interface and processing logics. Most importantly, geographic latitude/longitude data is now handled explicitly, leading to modified program behaviour in the areas of topological cleaning and output formats. Please note the relevant parts of sections 9 and 10, in addition to the information given in this section.

Survey2GIS allows the user to set spatial reference system (SRS) and reprojection information for only the input or both input and output data. Setting the input SRS correctly helps ensure correct data processing. Additionally setting the output SRS will cause *Survey2GIS* to perform output data reprojection.

SRS and further reprojection parameters (concerning datum transformation for higher accuracy in certain scenarios: see 8.2.1) can be set through a number of options with the common prefix "proj-" (note thate there are no one-letter short names for these):

```
--proj-in
--proj-out
--proj-dx
--proj-dy
--proj-dz
--proj-rx
--proj-ry
--proj-rz
--proj-ds
--proj-grid
```

The first two options ("--proj-in" and "--proj-out") are used to set the SRS of the input and output data, respectively, while the remaining ones are only used in the context of a geodesic datum shift. More information on each option is given in the following subsections (note also the short summaries in 2.3).

Reprojection is performed *after* all other geometric processing is done. This means that it is possible to e. g. topologically clean UTM input data and reproject the output to a latitude/longitude system.

To compute reprojections, *Survey2GIS* uses a built-in copy of the PROJ.4 open source library. This means that all SRS definitions will be converted to PROJ.4 definitions internally, and that all reprojection capabilities are identical with those of the built-in version of PROJ.4. Read 8.2 for more information.

If the input data consists of geographic lat/lon coordinates, it will be assumed that these are provided as simple decimal degrees (see https://en.wikipedia.org/wiki/Decimal_degrees). *Survey2GIS* does not include any support for correctly parsing lat/lon data that uses "E/W" prefixes and/or "degrees, minutes, seconds" notations.

8.1 Setting input and output SRS

The user can specify the (given) SRS of the input data and the (intended) SRS of the output data. This is done via the options "--proj-in" and "--proj-out", respectively. Only one SRS can be set for all input files. It is not possible to mix input data with different SRS in one program run.

Setting *only* an *input* SRS allows *Survey2GIS* to adjust processing if needed. This is especially important in the case of latitude/longitude coordinate data which requires different mathematical treatment. The program will issue a warning or abort operations in situations where lat/lon data cannot be processed (correctly).

If *both* input and output SRS are set and are *different* from each other, then *Survey2GIS* will *reproject* the data (unless the input data's SRS is "local": see below) from the input to the output reference system. Reprojection generally involves a loss of accuracy. To minimize the latter, additional datum transformation options can be supplied by the user (see 8.2.1) for details.

Setting *only* an *output* SRS will result in a processing error.

SRS can be specified using EPSG codes, PROJ.4 strings or, in a few common cases, via convenient shorthands (read on for details).

8.1.1 Internal SRS abbreviations

For convenience, *Survey2GIS* understands a small number of shorthand names that can be used to set the SRS (capitalization is ignored):

local Setting the SRS to "local" tells *Survey2GIS* that the data uses a local, cartesian reference system with metric units (meters). It is not possible to reproject such data. The "local" abbreviation exists mainly for the sake of consistency and completeness. If the input SRS is set to "local" and the output SRS is set to any other value, the program will issue an error message and abort.

wgs84 Setting the SRS to "wgs84" is identical to setting it to "EPSG:4326" (latitude and longitude values with WGS 84 datum). Note that *Survey2GIS* assumes that all input data uses *decimal* notation (see 8.3).

web This abbreviation is used to specify the SRS used by Google Maps and similar online mapping tools. Please see 8.1.4 for details. It is identical with "EPSG:3857".

utmXXN/S It is possible to specify a UTM zone (WGS 84 datum is always assumed) by replacing "*XX*" with a one or two-digit zone number and "N"(orth) or "S"(outh) designation. E. g. specify "utm32s" for "UTM with WGS 84 datum, zone 32, north". The EPSG equivalents are codes "32601" through "32760".

dhdnX This abbreviation is used for the Gauß-Krüger national grid of Germany. Replace "*X*" with zone identifiers "2" to "5". The EPSG equivalents are codes "31466" through (including) "31469".

osgb This is the traditional (still in use) grid system of the British Ordnance Survey (OSGB). It is identical with "EPSG:27700".

8.1.2 Using EPSG codes

If no internal abbreviation (see 8.1.1) is available, the second most simple way to specify an SRS is to pass its EPSG code (in fact, all SRS abbrevations are automatically mapped to their EPSG equivalents by *Survey2GIS*).

The full EPSG SRS database is available for reference online at http://www.epsg-registry.org/.

Setting an EPSG code as input or output SRS is done by prefixing the EPSG code with "epsg:" (capitalization is ignored).

Valid examples:

```
--proj-in=EPSG:27700
--proj-out=epsg:4326
```

Note that *Survey2GIS* only supports a subset of EPSG definitions, namely those that are also supported by the included version of *PROJ.4* (see 8.1.3). Also note that PROJ.4 uses a simple, potentially imperfect, translation from EPSG codes to its own format, which may incur some loss of information/accuracy (see: http://proj4.org/faq.html#how-do-i-use-epsg-coordinate-system-codes-with-proj). If this is an issue, consider specifying PROJ.4 format paramaters directly (see 8.1.3).

8.1.3 Using PROJ.4 strings

In some scenarios, neither the use of an internal SRS abbreviation nor the specification of an EPSG code (see 8.1.1 and 8.1.2, respectively) may be sufficient to obtain accurate reprojection results.

In such cases, full PROJ.4 SRS definitions ("PROJ.4 strings") can be provided as input and/or output SRS definitions.

PROJ.4 strings consists of several *tokens*, each one prefixed with a "+" and separated by a space from the next one. Capitalization of token names does not matter to *Survey2GIS*, as it automatically takes care of converting all token names to lowercase text. The token's name is followed by its value, separated from the name using an equal sign ("=") *without* further whitespace.

For example, the PROJ.4 version of "EPSG:4326" (lat/lon data with WGS 84 datum) is defined as follows:

```
+proj=longlat +ellps=WGS84 +datum=WGS84
```

And "EPSG:32601" (UTM zone 1, north) is defined as:

```
+proj=utm +zone=1 +ellps=WGS84 +datum=WGS84 +units=m
```

Please consult the PROJ.4 online documentation at http://proj4.org/ for full details, and also mind the information in 8.2.

In addition, the following may be helpful when trying to compose valid PROJ.4 strings:

- *Survey2GIS* prints out the full PROJ.4 definitions for all user-provided SRS as part of its standard logging output.
- The website at http://www.spatialreference.org/ can be used to convert between different SRS representations, including EPGS and PROJ.4.

An additional "+no_defs" token is *always* appended to the PROJ.4 string by *Survey2GIS*. This prevents any (potentially interfering) defaults that might be provided by PROJ.4 from being applied.

Datum transformation parameters and/or a datum shift grid file (see 8.2.1) can be appended using the tokens "+towgs84" and "+nadgrid", respectively. Note that these tokens will automatically be discarded and replaced by the corresponding *Survey2GIS* option values, if the latter are provided (see also 8.2.1).

Important: PROJ.4 definition strings are case-sensitive! E. g. specifying "+ellps=wgs84" will produce an error (the correct form being "+ellps=WGS84").

Important: PROJ.4 does not understand international numeric locales! This means that it is *not* possible to use decimal separators other than "." (period) in PROJ.4 token values!

8.1.4 Web Mercator

Popular Internet navigation and visualization tools, such as Google Earth/Maps, OpenStreetMap, etc., use a perfect sphere as Earth model (i. e. the length of the equator is assumed to be exactly the same as the length of any meridian arc). This is not a realistic geodetic assumption, but it increases 3D performance. This perfectly spherical Earth model is called "Web Mercator" (see https://en.wikipedia.org/wiki/Web_Mercator). The tricky thing is that these tools will *pretend* that the coordinates displayed on a Web Mercator "ellipsoid" are *identical* with those on a geodetic WGS 84 ellipsoid (an effect of this is that length and area measurements using a 3D tool with a Web Mercator sphere will be wrong compared to the real, geodetic measurements on a WGS 84 Earth model): the coordinates seen on the 3D display are *meant* to be WGS 84.

Paradoxically, when performing a geodetically *correct* reprojection with a datum conversion between the two (see 8.2.1) *wrong* Y coordinates will result! This is due to PROJ.4 automatically compensating for the difference in *N-S radii* between the two systems' Earth models.

According to http://spatialreference.org/ref/sr-org/6864/, the error can be quite significant:

> "Relative to an ellipsoidal development errors of up to 800 meters in position and 0.7 percent in scale may arise."

According to EPSG entry:

> "Relative to WGS 84 / World Mercator (CRS code 3395) errors of 0.7 percent in scale and differences in northing of up to 43km in the map (equivalent to 21km on the ground) may arise."

To circumvent this problem and maintain the intuitive assumption that e. g. the coordinates of locations identified on a Google Earth display are the same as their counterparts on a "proper" map with WGS 84 datum, *Survey2GIS* will perform a two-stage reprojection whenever Web Mercator data is involved:

1. The Web Mercator-based coordinates are redeclared as WGS 84 *without* any reprojection, *pretending* that they are identical.
2. The "converted" WGS 84 coordinates will then be reprojected "properly" to the desired output SRS (unless "WGS 84" is the desired target system, in which case no further action is required).

Be advised however, that this is a work-around trick and that 3D tools with Web Mercator models are not suitable for applications that require geodetic correctness and accurate measurements.

This mechanism will *only* work if either "EPSG:3857" or the abbreviation "web" is used to specify the input/output SRS. If a PROJ.4 string is used, then it must include the necessary additions to handle the *Y* coordinates compensation (see http://proj4.org/faq.html#changing-ellipsoid-why-can-t-i-convert-from-wgs8 for details).

Providing any datum transformation parameters or a grid shift file (see 8.2.1) will lead to a program error if Web Mercator is the source or target SRS!

8.2 Reprojection using PROJ.4

Important: This version of *Survey2GIS* uses PROJ.4, version *4*. At the time of writing, a major revision of PROJ.4, leading to version *5* and involving numerous changes in the user interface and underlying reprojection engine, was in progress. Please make sure to refer to the correct PROJ.4 version when looking for further information online.

If both input and output SRS are provided, are valid (see 8.1), and neither is a "local" system (see 8.1.1), then *Survey2GIS* will reproject all coordinate data *from* the input SRS *to* the output SRS.

Reprojection includes all point/vertex coordinates as well as (if present) label coordinates (see 5).

Survey2GIS relies on the open source library *PROJ.4* (http://proj4.org/) for SRS and reprojection support. Therefore, its capabilities for data reprojection, geodesic datum support, etc. are identical with those provided by *PROJ.4*. Essentially, reprojection functionality is limited to 2D geographic (lat/lon) and 2D planimetric (*X*/*Y*) SRS.

Survey2GIS ships with a bundled copy of *PROJ.4*. The exact version number of the copy used by *Survey2GIS* is reported in the processing log. The bundled subfolder "proj" contains a database file with EPSG codes (called "epsg": see 8.1.2) for use by PROJ.4 and some grid files (you can add your own: see 8.2.2). This folder is essential for PROJ.4 to work correctly, and it should reside within the folder from which

Survey2GIS is launched (in the case of the Windows version, this is the location of "survey2gis.exe", *not* "survey2gis.bat"):

The "proj" folder and the "epsg" file residing inside of it are part of the default distribution of *Survey2GIS* and thus PROJ.4 should work without problems.

Alternatively, the environment variable "PROJ_LIB" can be set to contain the full path to a copy of the "proj" folder. And as a third fallback option, PROJ.4 will look for "proj" in a default, system-wide location (the latter will vary, depending on the operating system).

If the "proj" folder does not exist (in any of the locations referenced above) or is not readable, *Survey2GIS* will still run, but reprojection functionality will be limited, and attempting to perform certain reprojections (e. g. involving EPSG codes) will result in an error.

8.2.1 Datum transformations

Where SRS with different datums (such as "WGS 84" vs. "ED50") are involved, the key to minimizing loss of accuracy during reprojection is to specify the best available *datum transformation*.

A geodetic/map datum is essential information that describes the "anchor point" and orientation of an SRS. In practice, the Earth model (such as a geodetic ellipsoid) is also considered part of the (extended) map datum.

Historically, many different datums have been used for mapping around the globe. In more recent times, mapping practice has converged on a smaller number of datums used for large areas of the world. The most common map datum, which also underlies the modern UTM system, is the "World Geodetic System" datum of 1984 (WGS 84), but other datums, such as ETRS89 also remain in use.

When computing reprojections, *Survey2GIS* (more accurately: the PROJ.4 library on which it relies) uses WGS 84 as a common "pivotal datum": if datums other than WGS 84 are involved, then coordinate transformations pass from the input SRS datum through WGS 84 and into the output SRS datum. Consequentially, it is possible to specify datum transformation options *to* WGS 84 from the *input* SRS.

When using SRS abbreviations or EPSG codes for specify input/output SRS in *Survey2GIS* (see 8.1), commonly applied transformations parameters to WGS 84 are already included in the SRS definition. However, there is no guarantee that the default transformation achieves the best possible results. Therefore, datum transformations can also be provided by the user.

There are two basic datum transformation types: Three-parameter transformations provide translation (shift) factors towards the WGS 84 datum along the geographic X, Y and Z axes, while seven-paramater transformations also specify three rotational factors (again, around the geographic axes) and a scale factor.

Transformation parameters can be set using the respective options provided by *Survey2GIS*:

```
--proj-dx
--proj-dy
--proj-dz
```

```
--proj-rx
--proj-ry
--proj-rz
--proj-ds
```

The first three options ("dx", "dy" and "dz") listed above are the translations factors of a three-parameter transformation. The following three ("rx", "ry" and "rz") and the final factor ("ds") are the rotational and scale factors, respectively of a seven-parameter transformation.

Note that these options can also be specified as part of a PROJ.4 string (token "towgs84": see 8.1.3), but options passed to *Survey2GIS* will replace/overwrite those in the PROJ.4 string (if present).

Note also that it is possible to specify only some of the options above (i. e. less than would be needed for a complete three or seven parameter transformation). In such case, *Survey2GIS* will automatically complete the datum transformation parameters using default parameters where needed. The default value for the scale factor ("--proj-ds") is "1", all other parameters (translation and rotation factors) default to "0".

User-defined transformation parameters will only applied by *Survey2GIS* if at least one parameter value deviates from its default value.

Important: Not supplying any transformation options to *Survey2GIS* does not mean that reprojection will not include a datum transformation. E. g. most EPSG SRS definitions (see 8.1.2) include their own transformation parameters, which *will* be applied if there is no user-specified transformation. *Survey2GIS* will report any applied transformation parameters as part of its log output.

The following conditions will raise an error and abort the current processing job:

- Both numeric datum transformation parameters *and* a transformation grid file (see 8.2.2) have been specified (only one of the two can be applied).
- Numeric datum transformation parameters *or* a transformation grid file have been specified, but "--proj-in" or "--proj-out" are missing (the latter must both be specified to trigger a reprojection).

If the *output* SRS has *no* datum transformation (to WGS 84) definition, then a synthethic three-parameter transformation with all parameters (i. e. X, Y and Z axis shifts) set to "0" will automatically be added to it. The reason for this is that transformation parameters *must* be present in both input and output SRS if a datum transformation is required (see: http://proj4.org/faq.html#why-do-i-get-different-results-with-4-5-

Important: For the reason given in 8.1.3, it is *not* possible to use decimal separators other than "." (period) in the specification of datum transformation parameters!

8.2.2 Grid-based reprojection

Some application cases, such as high-precision engineering and cadastral mapping, may require a reprojection accuracy that cannot be achieved with even the best available parametric datum transformations (see 8.2.1)

in all scenarios. In such cases, the alternative is to use a dense set ("grid") of local correction vectors, provided specifically for reprojection to the target SRS.

A file ("grid file") containing such correction data can be passed to *Survey2GIS*, which will in turn pass the data through to the underlying PROJ.4 library. The grid file's data coverage must match the extents of the (reprojected) data in the target SRS, otherwise it will simply have no effect.

The full path and name of a grid file can be supplied to *Survey2GIS* via the corresponding option:

```
--proj-grid=
```

For more on using grid files read section "nadgrids - Grid Based Datum Adjustments" at http://proj.maptools.org/gen_parms.html.

Note that PROJ.4 allows more than one grid file to be used, but only one can be specified using the dedicated option of *Survey2GIS*. Alternatively, by specifying the PROJ.4 string directly (see 8.1.3) and using the "+nadgrids=" token, several grid shift files can be supplied (separated by commas). In the latter case, the first grid file whose extents match the input data will be used for the reprojection. When using the "+nadgrids" token in a PROJ.4 string, do not use the "--proj-grid=" option, as the latter will automatically replace the former.

Note: any relative path given in the value of the "--proj-grid=" option will be resolved to a full, absolute path automatically by *Survey2GIS*, so that PROJ.4 can correctly locate the file. When instead using the "+nadgrids" token in a PROJ.4 string, users must take care of providing complete paths themselves.

8.3 Notes on SRS and reprojection issues

Reprojection is a complex process. This section details some of the trickier issues that are involved in the process and gives advice on how to obtain the best possible results.

Shape preservation in large-area surveys Some reprojection types (notable those involving a transformation from planimetric X/Y to geographic lat/lon data) can result in a significant loss of geometric accuracy. In such cases, better reprojection results may be obtained by adding more vertices to lines and polygon boundaries during surveying.

Datum transformation quality Using the best possible datum transformation is key to reprojection accuracy where systems of different natures are involved. The default transformation parameters contained in EPSG codes are not always the best choice. Grid files should be used where available; Search the EPSG database at http://www.epsg-registry.org/ for alternative transformation parameters and accuracy assessments and read section 8.2.1 of this manual.

Converting SRS representations: The website at http://www.spatialreference.org/ provides a very helpful database of global SRS in a variety of common formats, including EPSG, PROJ.4 and WKT (Well Known Text). It is also possible to download ".prj" auxillary files for adding SRS info to existing Shapefiles.

Lat/lon data: *Survey2GIS* expects all lat/lon data ito use decimal degrees. Sufficient locational accuracy in lat/lon coordinates may require many decimal places. Helpful information and guidance on this topic is available at https://en.wikipedia.org/wiki/Decimal_degrees.

Vertical (Z) coordinate units: PROJ.4 automatically detects the correct *Z* units from the SRS definition before doing the reprojection. However, *Survey2GIS always* assumes that *Z* is in *meters* when re-orienting coordinates (such as "Local X-Z": 7.2) or performing any other 3D operations. These two assumptions may collide and produce unexpected results in rare circumstances.

9 Topological quality and cleaning

Note: The processes described here assume that there is "something wrong" with the original input data and that it needs fixing. More precisely, it is assumed that the input data deviates from the *intended* geometries significantly enough to warrant some automated adjustment that will (ideally just slightly) modify the measured points, lines and polygons to improve the quality of the outcome. Note also that *Survey2GIS* assumes that the input data represents geometries on a relatively planar surface, meaning that they are similar to their flat 2D counterparts (i. e. constant $Z = 0.0$) and have little potential for overlaps and undercuts. If this is not the case or you wish to rather preserve the data closer to those originally surveyed, then set "--tolerance=", "--snapping=" and "--dangling=" options to "0.0" to skip *most* topological cleaning actions (note that the ones described in 9.1.5 and 9.1.6 will run in any case!).

GIS-based data processing has specific requirements regarding data quality. One of the most important ones is the *topological correctness* of data. GIS use topological relations such as "contains", "intersects" or "overlaps" to query and filter spatial data. In order for such queries to return correct results, the data must be topologically correct.

To improve topological quality, *Survey2GIS* subjects *all selected input data* (by default, this means all data, but see 6 for how to reduce the set of geometries to a selection) to some essential checks and tries to automatically fix common problems. It is important to note, however, that the GIS topology model is strictly two-dimensional, and that *Survey2GIS* is also mostly limited to two-dimensional topology checks and corrections. Basically, what this means is that data is treated as though all Z coordinates were constant "0". The more planar the actual input data, the better the chances that topological corrections will work accurately and as intended.

Even though many GIS offer there own, sometimes very powerful, topological cleaning tools, it is best to let *Survey2GIS* do the basic work, as it has the information available at the level of "raw" measurements, whereas the GIS can only work on the reconstructed geometries. There are two stages at which topological errors can be detected: When each input file is being processed and after all geometries found in all input files have been built; and *Survey2GIS* will report topological issues for both stages separately.

9.1 Problems and corrections

Typical survey data suffers from a limited set of topological error sources and defects, many of which can be corrected automatically by *Survey2GIS*. This section describes the topological cleaning funtions in the same sequence in which *Survey2GIS* carries them out.

Note that it is unlikely that the results of such automated cleaning will be perfect in practice. Real-world survey data contains erroneous and extreme measurements. In addition, the geometries processed by *Survey2GIS* are weakly structured (essentially, only by the order of records in the input file/s and the geometry types and sizes), and they have no "natural" hierarchy that would allow for perfect resolution of topological problems in all cases. Therefore, the cleaned output data should always be closely examined and manually improved (e. g. using flexible editing tools in GIS) as necessary.

In some cases, processing line and polygon geometries separately with selection commands (see 6) can provide better control over the cleaning process. It may also be advisable to produce a version of the output with all high-level topological cleaning functions turned off (i. e setting program options "--tolerance=", "--snapping=" and "--dangling=" to "0") to be able to review the effects of the cleaning process.

9.1.1 Duplicate points (thinning)

Duplicate points result from measurements that are so close to each other that they should actually be considered one and the same point. They often represent redundant vertices and increase the likelihood that hard-to-spot self-intersections in complex polygon boundaries will occur (see 9.1.4). But even simple point measurements can be problematic in this regard, e. g. when duplicate elevation points lead to over-representation during surface interpolation in GIS.

Therefore, *Survey2GIS* has the option "--tolerance=" that allows to set a distance threshold below which points will be considered duplicates. In such cases, only the first point measurement will be preserved and all of its duplicates discarded. This process is also called "thinning", and its effect depends on the geometry type:

- *Points* will be thinned by removing all points that are closer than the threshold to *any* other point within the *same* input file.
- *Lines* will be thinned by removing all vertices whose distance to the next (successive) *vertex on the same line* is smaller than the threshold.
- Likewise, vertices on *polygon* boundaries will be thinned by removing all vertices whose distance to the next (successive) *vertex on the same polygon boundary* is smaller than the threshold.

In the case of 3D data, the distance threshold will be applied to the 3D straight-line distance. This helps preserve intentionally densified measurements along sharp terrain edges.

Point/vertex thinning can be turned off entirely by setting option "--tolerance=" to a value smaller than "0" to .

Warning: Large values for the thinning threshold (option "--tolerance=") can severly degrade the shapes of polygons and lines and produce unintended results. In most cases, using a threshold of a similar magnitude as the (estimated) accuracy of the survey is advisable. If point data must be thinned more aggressively, then the different geometry types can be processed separately using a selection command (see 6).

9.1.2 Splinters

Splinters are geometries that are intended to be lines but have less than two vertices, and geometries intended to be polygons with less than three vertices. Such malformed geometries will simply be discarded.

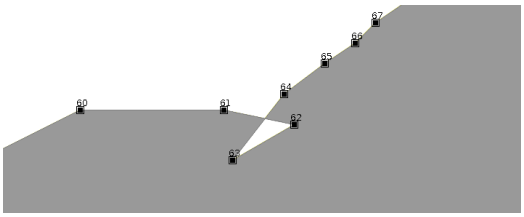
9.1.3 Multi-part geometries

While not a "topology issue" in the strict sense, the concept of multi-part geometries is an important aspect of data quality and consistency.

Sometimes physical constraints mean that the geometry that represents a single object must be recorded as separate, spatially disjunct *parts*. E. g., this may be the case if the area of a polygonal object has been exposed in two separate locations. In order for such multi-part recording to work properly, all parts of the geometry must have the same value in "key_field" and the parser option "key_unique" must be enabled (see descriptions in 4). Provided that this is the case, *Survey2GIS* will then automatically assemble all parts into one *multi-part geometry* and assign it only one attribute data record in the GIS output.

9.1.4 Self-intersections of polygon boundaries

A *self-intersection* occurs when a vertex B is placed behind its preceding vertex A in such a way that a straight line from B to its successor vertex C crosses over the existing polygon boundary. Small self-intersections can be hard to spot but will result in triangular, "inverted" areas when visualizing the affected polygon in GIS:



Self-intersections are frequent topological errors that prevent many GIS operations from working properly. Currently, *Survey2GIS* cannot automatically correct this type of error. However, the risk of self-intersections occuring can be lowered by using a "--tolerance" setting equal to the minimal wanted distance between polygon boundary vertices (see also 9.1.1).

Survey2GIS checks for self-intersections early on, when building lines and polygons from the input data. If this is the case, a warning will be issued. To facilitate manual cleaning of self-intersections, new vertices will be added to geometries at self-intersection points.

Some higher level topological cleaning actions will fail with self-intersections present. In such cases, *Survey2GIS* will skip further cleaning of the affected geometries and issue a warning. Check the status messages produced at the end of program operation for the total number of detected self-intersections

9.1.5 Shared boundaries of polygons

Note: Automatic cleaning of boundaries of adjacent polygons is a feature that is available starting with version 1.5.1 of *Survey2GIS*.

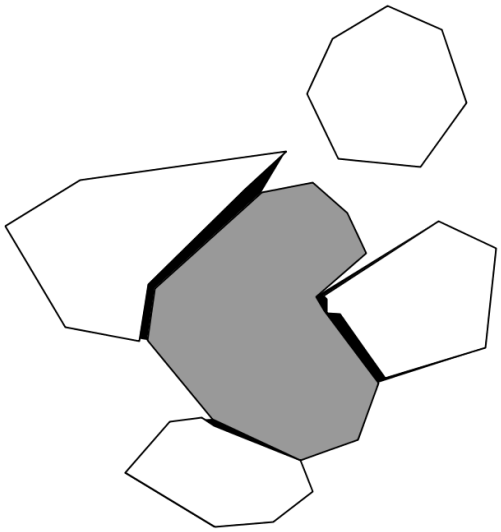
Two adjacent polygons share a common boundary along their line of contact. During surveying, however, practical restrictions require that both polygons are recorded completely, effectively recording the shared boundary twice – and most likely with some discrepancies, since it will be impossible to re-record the shared vertices in the exact same locations. Therefore, two fundamental problems will be impossible to avoid:

1. Small gaps will be left between polygons, even though they are directly adjacent.
2. The boundaries of two adjacent polygons will overlap slightly.

From the point of view of 2D GIS topology, the second case is a topological error if the two polygons are part of the same GIS layer. For the production of topologically correct output data, *Survey2GIS* will attempt to close unintended small gaps between polygon boundaries, and also to remove overlapping areas of adjacent polygons. For these purposes, it offers two tools that complement each other well:

1. Small difference in location between vertices of shared polygons can be compensated by adjusting the affected vertices so that the lie exactly on top of each other.
2. Overlapping areas of adjacent polygons will be removed automatically and additional vertices will be inserted, so that the polygons will have identical vertices along their shared boundary.

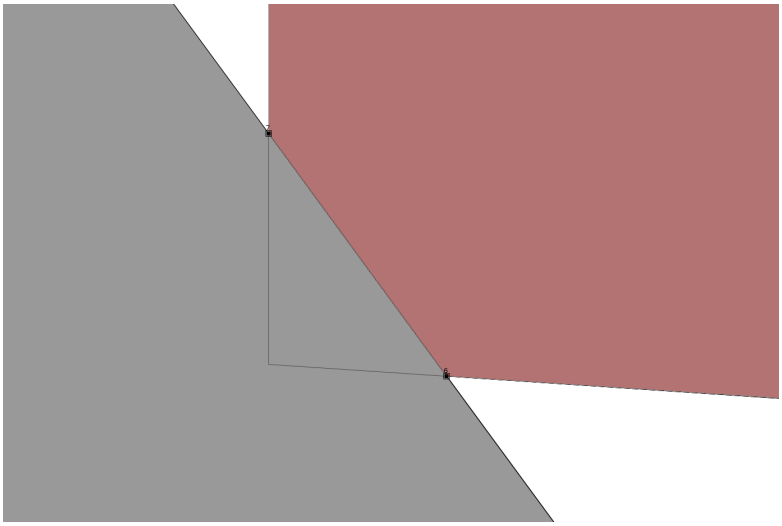
Only the first tool's action must be controlled by the user, using the "--snapping" option, which causes a vertex that is part of the *outer boundary* of polygon *B* to be moved exactly on top of ("snapped to") the closest vertex on polygon *A* (which is assumed to be its "sibling" vertex). The value set for this snapping distance threshold should reflect the locational error margin of attempting to measure the same point location more than once in practice.



The illustration above shows a (grey) polygon with four neighbouring (white) polygons. Snapping causes the vertices of the boundaries of the three nearest polygons to be moved, enlarging the polygons' areas by the amounts shown in black.

The snapping threshold should be chosen to lie well below the typical distance of separate point measurements, but above the accuracy with which an already measured point can be measured again.

The section function for cleaning the boundaries of adjacent polygons is fully automated: If (after snapping), there is still an area of overlap between two adjacent polygons, then the affected area will automatically be removed from the polygon that occurs *later* in the input data. Additional vertices will be inserted on the shared boundary where necessary:



Best results can be achieved, if:

- Larger (base) objects are surveyed first.
- Smaller, adjacent objects are surveyed later, and a small number of surveyed points are chosen intentionally so that lie a good distance (further than the threshold value for snapping, above!) *within* the area of the base object.

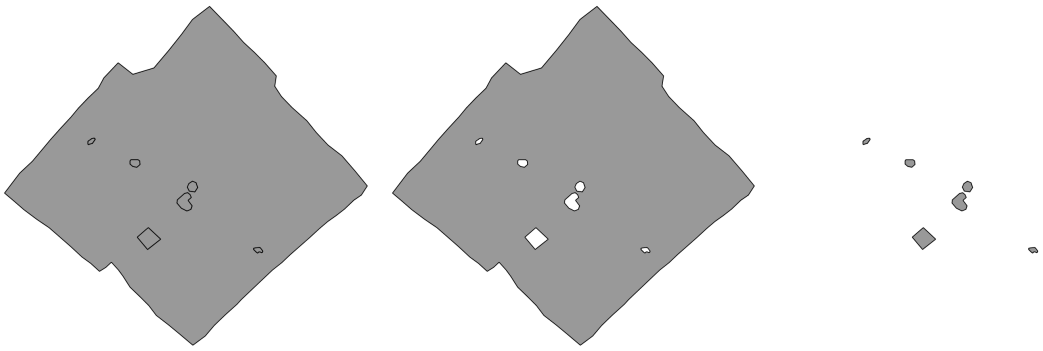
In practice, the effectiveness of automatic cleaning of shared polygon boundaries will be limited, depending on the complexity of overlaps, and the following should be taken into consideration:

- Relatively small overlap areas have the highest probability of getting fixed automatically.
- The operation is 2D only.
- Polygons with inner boundaries ("holes": see 9.1.6) cannot be adjusted, if this means that the number of holes would change (otherwise however, polygons with holes will be adjusted correctly).
- When checking for overlap, each part of a multi-part polygon (see 9.1.3) will be treated like an individual polygon, to ensure correct results.
- An extreme overlap, which would require a polygon to be cut into several disjunct parts, cannot be cleaned automatically.

- If the program option "--force-2d" (reduce all output to 2D coordinates) is *not* provided: Overlap cleaning will commonly create new vertices at boundary intersections for which Z coordinates will be interpolated linearly from neighbouring 3D vertices on the other polygon.

Also note, that the operations described above will likely fail for polygons that have other topological defects, notably self-intersctions (see 9.1.4).

9.1.6 Internal polygon boundaries (holes)



The 2D topology model of GIS demands that polygons in one and the same layer may not overlap each other. However, what if polygon B lies completely within the area of/is embedded in larger polygon A? In this case, B must be modelled as an internal boundary (hole) of A. In practice, this can be done in two different ways:

1. The polygon is recorded as a multi-part (at least two parts) geometry (see 9.1.3): First the base polygon and then any number of smaller polygons within its area. The latter will be converted to holes automatically. In order for this to work, the base polygon and every one of its parts must have the same value in "key_field" and the parser option "key_unique" must be turned "on" (see 4).

2. If the embedded polygon is a separate object, then it can be modelled as such by assigning it its own unique key value. In this case, *Survey2GIS* will automatically "punch" holes into the larger polygon and "fit" all embedded polygons into them without overlap (see illustration above).

In order to get good results from this cleaning process, it is important to record the polygons in a well-defined order (see 12). Also note that the corrections described above are likely to fail when applied to polygons that have topological errors, such as intersecting boundaries (cf. 9.1.7).

9.1.7 Intersections of lines and polygon boundaries

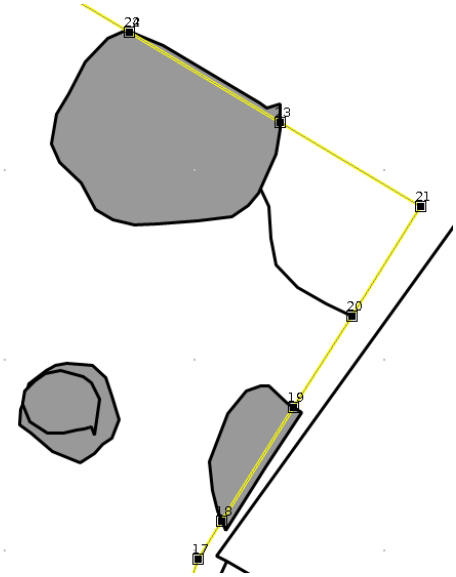
Complex surveys are likely to produce geometries that criss-cross and intersect each other. Such intersection points are often of critical importance for cleaning geometries manually, and intersecting lines are only

considered to be topologically valid if they both have a vertex at the exact point of intersection. Intersection vertices also allow for the automated removal of "dangles" (see 9.1.8).

Therefore, *Survey2GIS* marks intersection points by adding additional vertices to the output data, so that it will be easier to clean the data. There are three cases that lead to additional vertices being added at intersection points:

1. Lines crossing other lines (intersection vertices will be added to all crossed/crossing lines).
2. Lines crossing polygon boundaries (intersection vertices will only be added to crossing line segments, *not* to polygon boundaries).
3. Polygon boundaries crossing other polygon boundaries (intersection vertices will be added to all crossed/crossing boundaries).

(Note that the last case should rarely occur if boundary vertex snapping is effective: 9.1.5.)



The program will print two different intersection metrics once the data has been processed:

```
Detected line/polygon boundary intersections: 12
Added vertices at line/polygon boundary intersections: 8
```

In some cases, the number of intersection vertices added will be lower than the number of detected intersections. This happens, e. g., if a geometry is intersected by more than one other geometry in the exact same location. Such redundant intersection vertices are automatically identified and removed.

Note: Since *Survey2GIS* cannot know which geometry the user may want to correct manually later, an intersection vertex is added to all intersecting geometries of the same type.

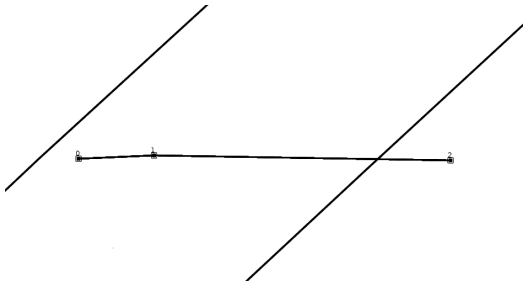
Self-intersections of lines are not considered topological errors by *Survey2GIS*, but will lead to an intersection vertex being placed on the location of the self-intersection. By contrast, self-intersecting polygon boundaries are *topological errors* (see also 9.1.4), as are intersections between polygon boundaries (see 9.1.5). If the latter are detected, warnings will be issued and topological errors will be logged, but no intersection vertices will be added.

9.1.8 Dangles (overshoots and undershoots)

In many cases, the start and end vertices of lines (also referred to as "nodes") are *intended* to be located exactly on another line or polygon boundary, but due to the limited accuracy of the survey, they actually lie at a small distance from their intended locations. Such line vertices are collectively called "dangles", and fall into two categories:

1. Line nodes that lie before their intended locations are "undershoots".
2. Line nodes that lie beyond their intended locations are "overshoots".

The illustration below shows a line that has both an undershoot (vertex "0") and an overshoot (vertex "2").

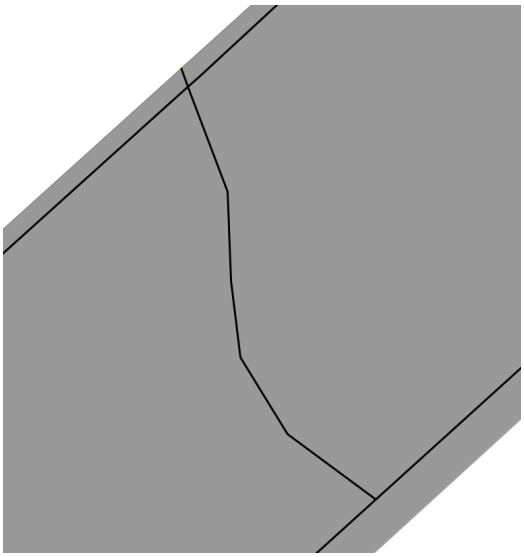


There is a separate threshold option, "--dangling=", to control the snapping of dangles to the nearest line/boundary segment. This is needed, because polygon boundary snapping (option "--snapping="; see 9.1.5) is run first and it can move the vertices of polygon boundaries so much that dangles will no longer snap. Therefore, the dangle snapping threshold should be set larger than the polygon boundary snapping threshold.

Only the first and last segments of a line qualify as dangles, and only if they are succeeded/preceded (respectively) by an *intersection vertex* (see 9.1.7). If these criteria lead to unsatisfactory cleaning results, and short chains of tiny line segments should also be removed, then consider increasing the "--threshold" option value (see 9.1.1) to reduce such chains to single vertices (note that this may introduce slight inaccuracies into the dangle snapping, as the original line bearing may be modified), or use the intersection vertices added by *Survey2GIS* (see 9.1.7) to manually remove all excess line segments.

Both overshoot and undershoot corrections will modify original line lengths and both operations can lead to counter-intuitive results if the first or last *n* vertices are all intersection vertices inserted by *Survey2GIS* (see

9.1.7). This can happen if e. g. a line segment crosses both another line segment and a polygon boundary segment. In such case, the dangling node will be cut back to the closest intersection point, which may or may not be the ideal/intended snapping position. The image below shows a line that has been snapped to a polygon boundary in the upper part and to a line segment in the lower part during dangle correction:



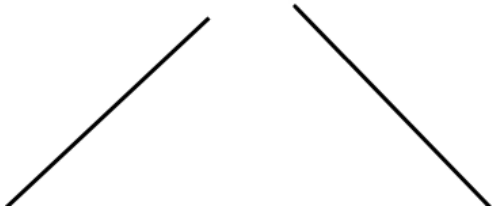
Sometimes, a dangle might occur to be an overshoot in relation to one geometry and an undershoot in relation to another. In such cases, *Survey2GIS* picks whichever dangle type is shortest (there is also an extremely small chance that both dangle types will have the exact same length, in which case the dangle will be considered an overshoot by default).

There are some limitations to the performance of automatic dangle cleaning by *Survey2GIS*:

- Undershoots within the same geometry *part* (in multi-part geometries: 9.1.3) will not be detected.
- Undershooting dangles are extruded by a 2D distance, so that they snap perfectly to the nearest line segment in a 2D GIS layer. For very long dangles and line segments with a steep Z gradient, this can introduce significant inaccuracies in the Z coordinates of snapped nodes.
- Overshoot correction of short lines that consist of only two segments can also lead to an extreme case where the entire line (part) would be deleted. In such cases, the geometry will *not* be modified. Instead, a warning message will be issued and a topological error logged.

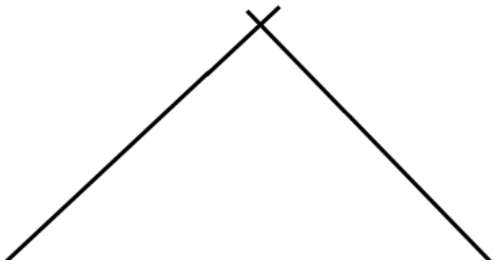
Therefore, the results of automated dangle cleaning should always be checked carefully and manual corrections applied where necessary (*Survey2GIS* automatically inserts vertices at crossing points to aid this: see 9.1.7). Use a small threshold for dangle cleaning to avoid overlay aggressive modification of line geometries.

A particularly hard problem is posed by "double undershoots" that frequently occure at the corners of line geometries and cannot be detected automatically. Since they do not yield intersection vertices, they are also difficult to fix manually:



There are two options for avoiding "double undershoots" while surveying:

- Code the geometry as a polygon instead of a line, so that it will be automatically closed.
- Cross the first segment with the last one while surveying, so that an intersection vertex will be generated and the two dangles can be removed as overshoots:



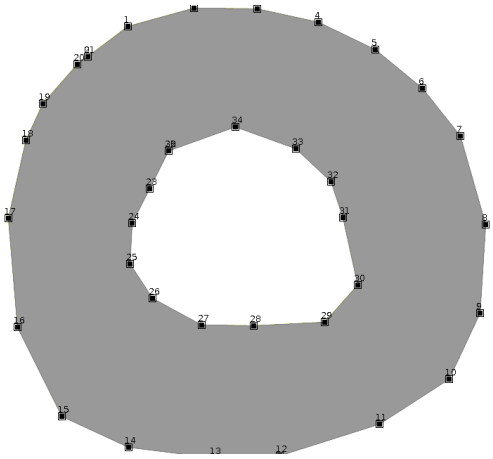
9.1.9 Order of polygon vertices

GIS data has a directional component that needs to be considered when digitizing polygonal data. In general, GIS attempt to correct the vertex order (also called "winding order") of polygons automatically, but it is safer to prevent problems from the ground up.

Visualization of 3D data, in particular, can suffer defects when data with inconsistent vertex ordering is rendered to the screen.

The following rules should be respected when surveying the vertices of polygons (see enumeration of vertices in illustration, below):

1. Record the vertices of *outer* polygon boundaries (also disjunct boundaries of multi-part geometries) in *clockwise* order.
2. Record the vertices of *inner* polygon boundaries ("holes") in *counter-clockwise* order.



When reconstructing polygons from survey data, *Survey2GIS* will attempt to enforce the above rules automatically and re-order vertices if necessary. In order to avoid problems, however, it is recommended to respect these rules during field work, when recording polygon outlines.

9.2 Practical considerations and limitations

Essentially all common GIS use a 2D topology model based on the concept of a bird's eye data view. This may cause topological errors to arise, even though the original 3D data is theoretically free of such errors. Polygon boundaries are especially problematic in GIS analysis, because from a 2D top-down perspectives, boundaries may seem to overlap even though they are free of overlap in 3D space. There is no way to automatically correct for such effects.

An alternative is to reduce the 3D input data to 2D data by dropping the definition of the Z coordinate field from the parser schema (4.4). In this case, *Survey2GIS* will assume all Z coordinates to be uniformly "0" and all topological cleaning operations are reduced to 2D operations.

In any case, complex polygons, at least, should always be inspected carefully and cleaned as needed, using the available GIS tools.

The topological cleaning functions available in *Survey2GIS* are not suitable for correcting very large or complex topological errors (e. g. multiple overlaps or "holes within holes"). The best prevention of topological errors is rigid self-control and good practice during the survey.

All functions for topological cleaning are run *after* a reprojection (if any: see 8.2), in order to avoid introducing further inaccuracies into the reprojected result. However, they run *before* a coordinate transformation to change axes orientations (if any: see 7), so that the result will be unaffected by a change of orientation in the coordinate system.

Topological processing of input data that is provided as latitude and longitude coordinates has severe limitations. Only the following cleaning actions will be performed:

- Removal of redundant vertices (see 9.1.1) and "splinters" (see 9.1.2).

- Check for self-intersections (see 9.1.4).
- Snapping for adjacent polygon boundaries (see 9.1.5).
- Re-ordering of polygon vertices (see 9.1.9).

The removal of redundant vertices and snapping actions use a planar distance measure (to be specified as decimal degrees in this case!), which reduces accuracy. The program will issue a warning message, accordingly.

Important: When combining a data selection (see 6) with topological cleaning, the result can vary from that obtained by processing the complete input data! The reason for this is that geometries not included in the selection no longer have an effect on topological operations in *Survey2GIS*. For example, polygon boundaries will not be adjusted if the overlap with boundaries of polygons that are part of the data selection. If this behaviour is not desirable, then it might be better to process the data in full and use e. g. a GIS to select data afterwards.

10 Output formats

In order to support further data processing with different applications, *Survey2GIS* can produce a variety of output formats.

For archival purposes, it is recommended to always store copies of the original raw input data files.

10.1 ESRI Shapefile

Option:

```
-f shp
```

The „ESRI Shapefile" format is the default output format for *Survey2GIS*. Although it is not a real standard in the sense that it has been openly developed and documented by an independent organisation, it is the most widely used vectorial data format in the GIS world, due to the fact that its inventor ESRI has released a detailed specification (https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf).

Technically speaking, a "Shapefile" is not a single file but a set of files (three or more) with the same base name and different extensions. The files generated by *Survey2GIS* will have the extensions „.shp", „.shx" and „.dbf". The latter is a file in DBase format that holds the table of attribute data for each geometry stored in the ".shp" file.

The Shapefile format is subject to a number of limitations, many of which arise from the use of DBase as attribute table format:

1. The maximum file size is limited by the size of the (32 bit integer) index in the ".shx" file. Typical limits are 2 or 4 GB, depending on the GIS' capabilities.
2. Field names can not be longer than 10 characters (DBase limit).
3. A stored decimal attribute value can have a maximal total length of 18 places. When storing values from very large numerical ranges, the decimal precision may have to be limited (DBase limit).
4. A stored text attribute value can not be longer than 254 characters (DBase limit).
5. The attribute table in DBase format does not store any information about the text encoding. If text fields use special characters that are not part of the ASCII set, then the user must take care that all software used to process the data uses the right encoding.
6. A single Shapefile datasaet can only store either points, lines or polygons. For this reason, *Survey2GIS* will automatically split the otuput into separate Shapefiles as required.

10.1.1 ESRI Shapefile label layers

When exporting a label layer (see 5) for Shapefile output, an additional points Shapefile will be produced with the following attributes:

labeltext This is a *string* (text) type field that contains the text to be used as a label at each point.

fonttype A *text* type field that contains the name of the font to be used for the labels. This is set to "Arial" by *Survey2GIS*.

fontstyle This field contains the font style as an *integer* code. It is set to "0" (plain) by *Survey2GIS*. Commonly used values are: "0" (plain), "1" (bold) and "2" (italics).

fontcolor The font color to use for labels is encoded using an *integer* value. The coding follow the rules of the Java class *java.awt.Color* (see https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html#getRGB() for details). The value exported by *survey2gis* is "-16777216" (black).

fontsize This is a *double* type field that contains the font size for the label text ("10.0" as set by *Survey2GIS*).

fontrotate This is a *double* type field that contains clockwise rotation of the font size ("0.0" as set by *Survey2GIS*).

geomtype This *integer* type field encodes the geometry type for which a label has been generated. Possible values are: "0" (point), "1" (line) and "2" (polygon).

Note that this attribute field structure has been chosen for compatibility with the "Annotations" functionality of *gvSIG CE* (http://gvsigce.org). In addition, a ".gva" label settings file will be produced (this can be ignored by users of other GIS). Label layers produced by *Survey2GIS* can be loaded as *annotation layers* into *gvSIG CE* and modified using that GIS' special annotation tools.

10.1.2 Null ("no data") in DBF attribute tables

The following is the default behavior if the user has *not* specified a "no data" value as part of the parser description (see option "no_data" in 4.3):

Attribute values that are "null" (no data) will be represented by setting all characters of the corresponding field to spaces in the DBF attribute table. The interpretation of this depends on the GIS. Commonly, text attributes will be read as empty strings, numeric attributes as "0".

10.2 Drawing Exchange Format (DXF)

Warning 1: Do *not* rely on DXF as exclusive output format for long-term data storage and archiving! For details, see the discussion below.

Warning 2: Data in DXF files is not suitable for spatial analysis that relies on the topological correctness of data! For details, see the discussion below.

Option:

```
-f dxf
```

DXF (Drawing Exchange Format) is a file format that can be processed by most CAD applications and (as opposed to e. g. DWG) is *relatively* simple in structure, with format documentation being available from AutoDesk (https://www.autodesk.com/techpubs/autocad/acad2000/dxf/). It allows storage of all geometry types as separate layers in a single file. However, it is not suitable for storing attribute data, since it does not support relational data structures. The DXF file itself can be used to store a handle for each drawing object (point, line or polygon) which can act as a "primary key" to join additional attribute data to the geometries in a GIS (see also 10.2.4).

The DXF output of *Survey2GIS* has the following characteristics:

- The file format version used is "AC1015" (AutoCAD Release 15/AutoCAD 2000).
- Points, "raw" points, lines and polygons are each stored in a separate layer. Raw points are labeled with their original coordinates.
- Each drawing object (except for "raw" points) has an integer type handle that corresponds to the "geometry ID" assigned to each object by *Survey2GIS* and is also stored in a separate DXF layer.
- In addition, each attribute field is written to a separate DXF layer.
- Handles and field values are plotted at point coordinates, at the centres of areas/polygons, and at the central vertices of lines (one for each part).
- Most of these layers are switched to "invisible" by default. However, not all CAD interpret this correctly and as a result switch all layers to "visible", regardless.
- Polygons are degraded to polylines, unless 2D output (option "-z") is produced (see 10.2.3).
- The topological quality aspects "correct order of boundary vertices" and "holes" are not supported (see 9).
- Full attribute data is written to a separate, simple text file (see 10.2.4). The values in the first column ("geom_id") correspond to the DXF object handles.

The DXF output option in *Survey2GIS* exists solely for the purpose of allowing CAD-based publishing workflows. The use of DXF for storing and/or processing topographical data is subject to severe limitations.

The DXF file produced by *Survey2GIS* always uses a simple point (".") as decimal separator for coordinate values, ignoring the "--decimal-point=" option, as well as any operating system settings.

10.2.1 DXF and data archiving

Regarding long-term storage/archival of data, it must be noted that DXF is (like all proprietary CAD formats) not independently standardized and is in constant flux. DXF is a registered trademark of AutoCAD producer AutoDesk. New versions of the DXF specification are released alongside each new version of AutoCAD (this is necessarily so, as DXF is a file-based, sequential representation of AutoCAD's internal, hierarchical object database). Most problematically, the format specification explicitely encourages a proprietary use of some of its elements. Programmers of "add-ons" or "plug-ins" are free to invent new classes and decide whether or not to include them into the DXF output, or even document them. All of this means that compatibility of this format with current software cannot be sustained in the long term and that DXF is not suitable for long term data archival. However, *Survey2GIS* uses only a small subset of the format and produces a relatively simple DXF *ASCII* file that allows reconstruction of the original data.

For long-term data storage, make sure to archive the original raw data and use a simple, well-documented GIS format, such as Shapefile (s. 10.1).

10.2.2 DXF and topological data

The import of DXF-stored data into a GIS for subsequent spatial data analysis is a common practice that must nevertheless be strongly discouraged. DXF is ignorant of topological data quality (s. 9) and, most importantly, does not have a simple and useful (in a GIS context) representation for areal objects (polygons). There is no reasonable way to represent properties such as holes and multi-parts in DXF-stored polygon. This will inevitably result in difficulties when attempting to link spatial data (geometries) with attribute data.

For GIS-based processing, let *Survey2GIS* create its output in a GIS format, such as Shapefile (s. 10.1).

10.2.3 DXF and planar polygons vs. polylines

Note: Planarization methods are not yet implemented in *Survey2GIS*!

As mentioned above, AutoCAD (and thus DXF) does not have a simple representation of an areal object, i. e. a polygon. Instead, such objects are modelled as hatch patterns that are set into the boundaries of polyline objects. *However*, the AutoCAD/DXF hatch pattern is defined to be a *planar* object. What this means is that all vertices on the boundary of the hatch pattern (i. e. the polyline) have to lie within the same X-Y plane. This restriction cannot be met by real-world surveying data, which will have variation in the Z coordinates, due to the geometry of the underlying natural surface. As a result, the following options are possible for DXF output:

1. Export all polygons as polylines (default behaviour).
2. Produce 2D output on export, thus flatting all Z coordinate values to "0.0".

3. Planarize the vertices of each polygon so that they lie in the same *X*-*Y* plane *and* the object's plane is parallel to the *X*-*Y* plane of the world coordinate system. The latter restriction is to make sure that AutoCAD will actually draw the hatched area when looking onto the objects from a top-down perspective.

Note that, in theory, the same problem exists for GIS and polygons with *Z* data (the mathematical definition of a polygon being that all of its vertices lie one the same plane). In practice however, non-planarity of polygons is simply ignored by 2D GIS.

More information on the subject of planarization of polygons can be found in **??**.

10.2.4 DXF and attribute data

There is no perfect way of associating attribute data records with drawing objects in a DXF file. *Survey2GIS* offers a simple work-around: An additional text file (with extension ".txt") is produced that stores one attribute table record per line.

The lines below are an example of what an attribute table text file might look like (excerpt):

```
geom_id;const1;idx;planum;type;extra;number;coorx;coory;coorz
0;123.450000;67;1;"GR";"W";0;3513037.664000;5279881.392000;399.563000
1;123.450000;1;1;"LO";"W";0;3513041.874000;5279875.482000;399.025000 2;123.450000;15;1;"LO";"
...
```

There are only a few formatting conventions that apply:

- Fields are separated by ";" (semicolon).
- The first line contains the names of the fields.
- The contents of text fields are surrounded by quotes (").

The first field is always the integer type field *geom_id*. This field contains a primary key that matches the handles for all drawing objects in the exported DXF file and can be used to associate CAD objects and table records.

No attribute data is written for "raw point" geometries (vertices).

The attribute text file produced by *Survey2GIS* always uses a point (".") as decimal separator for coordinate values, ignoring the "--decimal-point=" option, as well as any operating system settings.

10.2.5 Null ("no data") in DXF attributes

The following is the default behavior if the user has *not* specified a "no_data" option value as part of the parser description (see 4.3):

The representation of attribute values that are "null" (no data) depends on the attribute type: text fields will be written as empty strings, numeric fields as "0".

10.2.6 DXF label layers

Labels (see 5) are stored as a separate layer inside the DXF.

10.3 GeoJSON

The GeoJSON output is a single plain text file that contains a *JavaScript* style object representation of all features produced by *Survey2GIS*. GeoJSON is not very space-efficient and not a typical end-user format. It is more commonly used for exchanging data between spatial databases and WebGIS applications. The GeoJSON file produced by *Survey2GIS* follows the official specification at http://geojson.org closely (but mind the note on coordinate references, below!) and uses human-readable formatting.

Note: Similar to KML (see 10.4), GeoJSON was designed for large-area data coverage with geographic (latitude/longitude) coordinates. In fact, since its RFC 7946 revision, the GeoJSON standard no longer supports any coordinate reference system other than lat/lon data with WGS 84 datum (equivalent to EPSG 4326). However, since GeoJSON is useful for direct integration of *Survey2GIS* into a survey data processing pipeline or database backend, this software will continue to allow output of non-conforming GeoJSON with other coordinate system types, unless running in "strict" mode (see 2.3). Be aware that there is no way of storing SRS information in GeoJSON output and that strictly standard-conforming software might not be able to process such output correctly.

Features are sorted by geometry type (to minimize occlusions when importing the data as a multi-geometry GIS layer): first raw vertices, then points, then lines and finally polygons, and stored in a comma-separated list that contains plain, human-readable entries. Feature geometries are stored in the "geometry" part and attributes in "properties":

```
{ "type": "FeatureCollection",
  "features": [
    { "type": "Feature", "id": 0,
      "geometry": {
        "type": "Point",
        "coordinates": [3513040.585000, 5279881.854000, 399.102000]
      },
      "properties": {
        "geom_id": 0,
        "const1": 123.450000,
        "idx": 256,
        "level": 1,
        "type": "Gold",
        "aux": "FZ",
        "_id": 0,
```

```
            "coorx": 3513040.585000,
            "coory": 5279881.854000,
            "coorz": 399.102000
          }
        },
    ...
```

A "geom_id" property of type integer is always added and represents a key value that is unique within the scope of the GeoJSON object file.

Note that there is no explicit attribute type description. Whether a field is of type "text" (e. g. "type" in the example output above), "double" (e. g. "coorx") or "integer" (e. g. "level") is simply determined by the format of the value.

The GeoJSON file produced by *Survey2GIS* always uses a simple point (".") as decimal separator for coordinate values and numeric attribute values, ignoring the "--decimal-point" option, as well as any operating system settings.

10.3.1 GeoJSON geometry types

The GeoJSON output produced by *Survey2GIS* uses strongly typed GeoJSON geometries. What this means is that the most simple type is chosen which is capable of representing the feature produced by *Survey2GIS*:

- Points are stored as GeoJSON type "Point".
- Polygons are stored as GeoJSON type "Polygon" if the consist of only one part; they are stored as type "MultiPolygon" if they consists of more than one part.
- Lines are stored as GeoJSON type "LineString" if the consist of only one part; they are stored as type "MultiLineString" if they consists of more than one part.

Note: Since features of *Survey2GIS* types "point" and "raw point" share the same GeoJSON geometry type ("Point"), it will not be possible to differentiate between the two in the GeoJSON output file! If the latter is an issue for your workflow, then consider using a geometry type selection (see 6) and exporting to separate output files.

10.3.2 GeoJSON primary keys (field "id")

The GeoJSON standards reserves a field named "id" for use as primary key. If the input data already contains a field of that name, then it cannot be written to the GeoJSON "properties" member without clashing with the primary key.

In such case, *Survey2GIS* will attempt to resolve the problem by renaming the user-defined "id" field to "_id" (underscore plus original field name). In case a user-defined "_id" field also exists, this resolution will fail and the program will abort with an error message.

Note that, theoretically, the above should not be an issue, since the GeoJSON specification states that "id" should be part of the feature object itself, not its "properties" member. However, it seems that some GeoJSON drivers (notably http://www.gdal.org, which is used in most open source GIS) do not make this destinction (*Survey2GIS* does place "id" into the "type" member of each feature).

10.3.3 Null ("no data") in GeoJSON objects

The following is the default behavior if the user has *not* specified a "no data" value as part of the parser description (see 4.2):

The representation of attribute values that are "null" (no data) depends on the attribute type: text fields will be written as empty strings, numeric fields as "0" (integer) or "0.0" (double).

10.3.4 Label layers in GeoJSON

There is only limited support for labels in GeoJSON output.

Label properties (location, font settings, etc.: see 5) are stored as part of the "properties" (see above) section of the labelled features; labels are *not* stored as separate point features (as opposed to e. g. Shapefile output: see 10.1).

Since GeoJSON allows only one "properties" member per feature, there will also only be *one* set of label *X* and *Y* coordinates per feature. These coordinates will represent the label position of the *first* part of multi-part line and polygon geometries.

10.4 Keyhole Markup Language (Google KML)

The Keyhole Markup Language (Google KML) is an XML-based format that was popularized for use with Google Earth (https://www.google.com/earth/). Accordingly, the *sole* purpose of KML is to provide data for flexible 3D visualization in Google Earth. KML sacrificies accuracy for speed and simplicity and is therefore not a suitable format for subsequent data processing and archival.

Note: The most relevant limitation of KML is that it supports (or rather: assumes) strictly latitude/longitude coordinate data (in decimal notation) with an *assumed* WGS 84 datum (which is really Web Mercator: see 8.1.4 for details). Therefore, KML is only available for survey data that uses lat/lon coordinates (e. g. GPS data) or data that can be reprojected to lat/lon coordinates, such as UTM data (see 8.1). In the latter case, a suitable output SRS must be set (see 8.1).

The KML produced by *Survey2GIS* has been designed to keep large *survey* datasets manageable in Google Earth.

KML uses "Placemarks" to represent features (points, lines and polygons plus attributes) and these can in turn be grouped into "Folders" which can be used to collectively turn Placemarks on or off in Google Earth. The KML produced by *Survey2GIS* contains the following Folders:

1. Points
2. Lines
3. Polygons
4. Vertices
5. Labels

Most of the above should be self-explanatory. The folder "Vertices" contains the raw measurements for each point, line or polygon vertex, and exists only if raw data output is chosen (option "--raw-data", see 2.3); the contents of this folder will not be visible by default. The folder "Labels" contains user-defined labels (if chosen, see 5, also 10.4.2). The drawing order is determined by Google Earth.

Attribute data is stored using the "Extended data" feature of KML. This allows *Survey2GIS* to define a data schema and store exact representations of the attribute data. All attribute fields and their types will be visible when clicking on a Placemark in Google Earth.

Note that Google Earth has a reputation for producing display problems with complex polygons. Presumably, the reason for this is that its 3D tesselation algorithms are not sufficiently accurate and over-optimized for display speed.

The KML file produced by *Survey2GIS* always uses a simple point (".") as decimal separator for coordinate values and numeric attribute values, ignoring the "--decimal-point=" option, as well as any operating system settings.

10.4.1 KML geometry types and encoding

Coordinates are written into the KML file in this order: longitude (decimal degrees), latitude (decimal degrees), elevation (always assumed to be in meters).

The geometry types supported by KML Placemarks are:

- Polygon
- LineString
- Point

Multi-part geometries are represented by storing more than one Polygon or LineString per Placemark element.

Google Earth expects the vertices of polygons (outer boundaries as well as holes) to be written in *counter clockwise* order (GoogleEarth uses this to determine the direction of 3D faces for its artificial lighting). This order is automatically enforced by *Survey2GIS* when exporting the vertex data. Elevation (Z) data can be

interpreted in various ways by Google Earth, but *Survey2GIS* always sets the KML element <*altitudeMode*> to "absolute", meaning that elevation data is taken to be in meters above sea level.

Note that there may well be local mismatches in the accuracy between the Google Earth elevation model (DEM) and the measured Z coordinates in the survey data, which can lead to some visual artefacts in Google Earth.

10.4.2 Label layers in KML

The KML output will contain a dedicated folder "Labels" which contains the labelled Placemarks for all features, with label locations and contents defined by the user (see 5 for details).

11 Handling special (accent) characters

Support for the use of special or accented characters (i. e. those not included in the standard ASCII code table, such as German umlauts) for international text is limited to *only the following contexts* in *Survey2GIS*:

1. Names of files and folder,
2. program messages and user interface,
3. selection commands that include replacement text with special characters (cf. 6)
4. and a small subset of parser schema elements (cf. 4):
   (a) the content of "info" in the "[Parser]" section,
   (b) the content of "name" in the "[Parser]" section,
   (c) the content of "info" in a "[Field]" section,
   (d) and replacement text that follows an "@" in a "[Field]" section.

In all other contexts (e. g. option names passed on the command line, parser options, names of attribute fields, . . .) the use of non-ASCII characters is *not* supported.

Internally, *Survey2GIS* handles all international character encoding as *Unicode* version UTF-8. All ASCII characters have identical codes in UTF-8, which means that a conversion is not necessary and that pure ASCII data will always be processed correctly by *Survey2GIS*.

UTF-8 ist the most widely used encoding on the Internet and the default for many operating systems (in the rare case that a Linux or Mac OS X system does not use UTF-8, it can be adjusted to do so). The most notable exception to this is the Windows operating system, which (due to historical reasons) still handles character encoding via a number of "code pages" (the Windows code page that is closest to UTF-8 has ID 65001). Starting with version 1.3.1, *Survey2GIS* is capable of correctly processing the multi-byte encoding used by Windows for file and folder names. To correctly display the UTF-8 encoded program messages issued when running *Survey2GIS* from the Windows command line (*cmd.exe*), set the font used by *cmd.exe* to one that covers the Unicode character range, such as "Lucida Console".

When using special characters in a parser schema (where supported: see above), make sure that the text file containing the schema is produced with a program (text editor) that generates UTF-8 encoded files.

All attribute table field contents produced by *Survey2GIS* will also be encoded in UTF-8. This means that e. g. special characters used in the DBF attribute table of a Shapefile will only be displayed correctly by a GIS that supports UTF-8.

The use of special characters (as well as whitespace) in file and folder names should always be avoided, as this is very likely to cause problems at some stage (e. g. when processing the files with another software or exchanging files across different operating systems).

12 Hints on survey practice

The quality of the data produced by *Survey2GIS* is in large part determined by the design of the survey workflow. The tools for error correction built into *Survey2GIS* can only fix some of the problems that arise during field work. This section contains some important hints for a field workflow that ensures good quality result data.

The order of measurements, i. e. the order in which the records will be stored in the input data files, is of great importance:

1. The directions of lines are determined by the order of their vertices.
2. The the outher boundaries of polygons should be recorded in clockwise direction, inner boundaries counter-clockwise, (use the south to north axis of the spatial reference system for orientation). See 9.1.9 for details.
3. Input files should be forwarded to *Survey2GIS* in the same order in which they were produced during the survey.
4. Topological cleaning functions such as snapping and "hole punching" (9.1) will work best if larger "base objects" are recorded before smaller objects.

The most important principle is that of *hierarchical* surveying: first the base lines and largest objects, then adjacent smaller objects, embedded objects etc.

When recording polygons, it is not necessary to record the first vertice again to close the geometry. In so far as the output formats demands this, *Survey2GIS* will take care of closing polygons by exactly duplicating the first vertex.

An important consideration regards the encoding of point measurements. In parser modes "End" and "Min", geometry tags for points may be left out (no declaration of "geom_tag_point" has to be made in the parser schema). In this case, *Survey2GIS* will attempt to separate single point measurements automatically from vertices of lines and polygons. However, this will only work reliably if the primary key values (i. e. the contents of the field "key_field") are unique *across all geometries*. If this is not the case, the following problems may occur:

- Parser mode "End": If a point occurs directly *before* a line or polygon record, and if that point has the same key value as the vertices of the *following* line or polygon object, then it will be erroneously merged as a vertex into that object.
- Parser mode "Min": If a point follows directly *after* a line or polygon object and if that point has the same key value as the vertices of the *preceding* line or polygon object, then it will be erroneously merged as a vertex into that object.

In order to avoid these problems, the following alternative solutions are recommended:

1. The „geom_tag_point" option is declared in the parser schema and all point measurements are explicitely tagged as such during recording.

2. Separate, non-overlapping key value ranges are used for the different geometry types, e. g. 0-9999 for points, 10000-19999 for lines, 20000-29999 for polygons.

A Known problems

The following problems are known to exist in the current version of *Survey2GIS*:

- When run from Windows' *cmd.exe*, *Survey2GIS* cannot be used in GUI mode and display text on the console at the same time. Separate versions of "Survey2GIS.exe" must be run for each interface mode.
- The exchange of settings files (extension ".s2g") is subject to the following restriction: Numeric settings are saved using the current number format. If they are loaded into another instance of *Survey2GIS* that uses different language settings, then they may not be processed correctly. In such cases, please check the number formats of all numeric options and correct them if necessary.
- Support for latitude/longitude input coordinate data is currently limited. Most significantly, some topological cleaning functions will be inaccurate or unavailable. The software will issue appropriate warnings when processing lat/lon input data.
- Point records that have the same value in "key_field" as any subsequent line or polygon records will be merged with these subsequent geometries, causing e. g. selections to return wrong results. This problem only occurs if geometry markers are stored in the key field (i. e. "tag_field" and "key_field" refer to one and the same field: 4.2) and the key values are not globally unique (i. e. if a point geometry has the same key value as a line or polygon geometry). To avoid this problem, define "tag_field" and "key_field" to refer to different fields in the input data.

B License