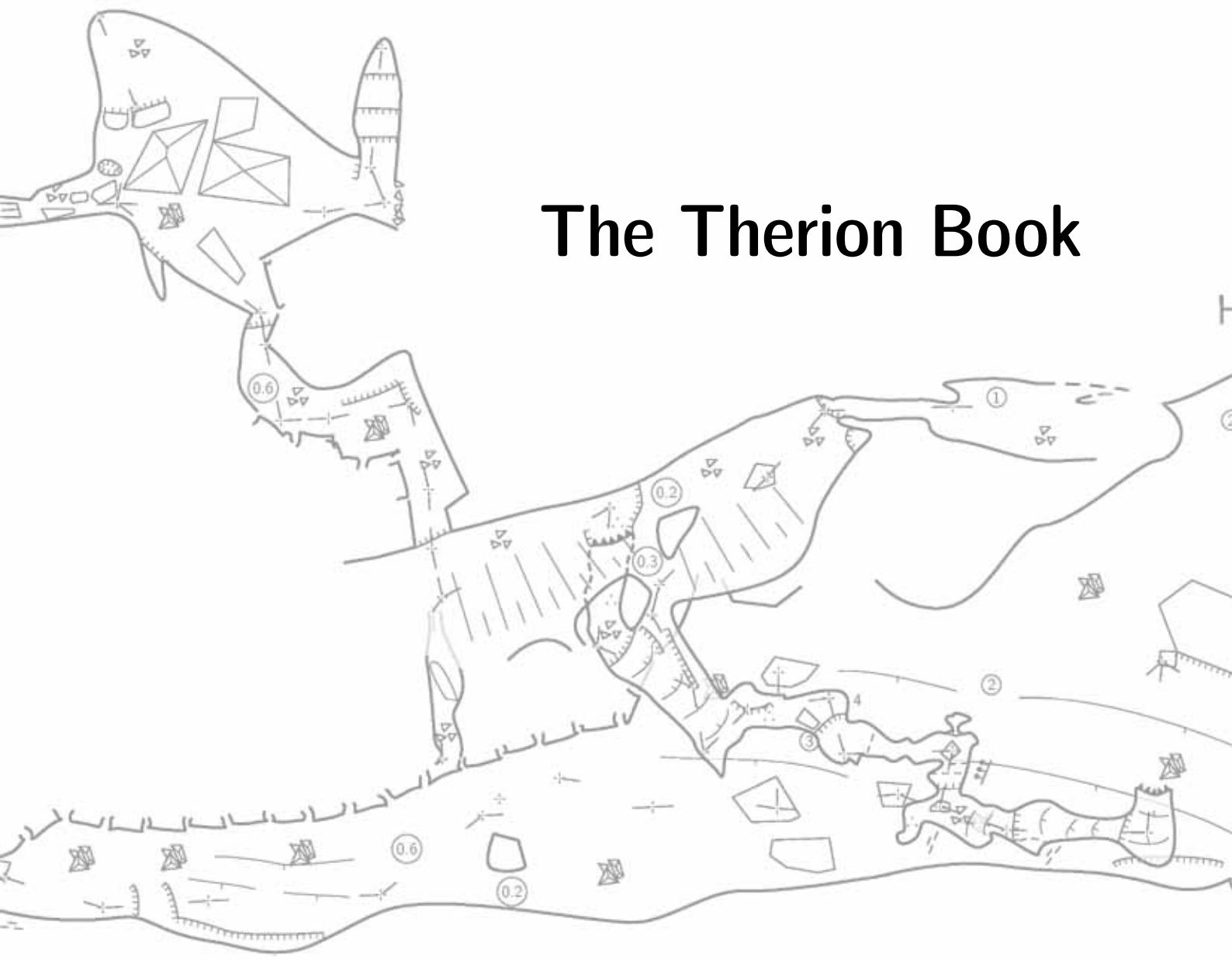
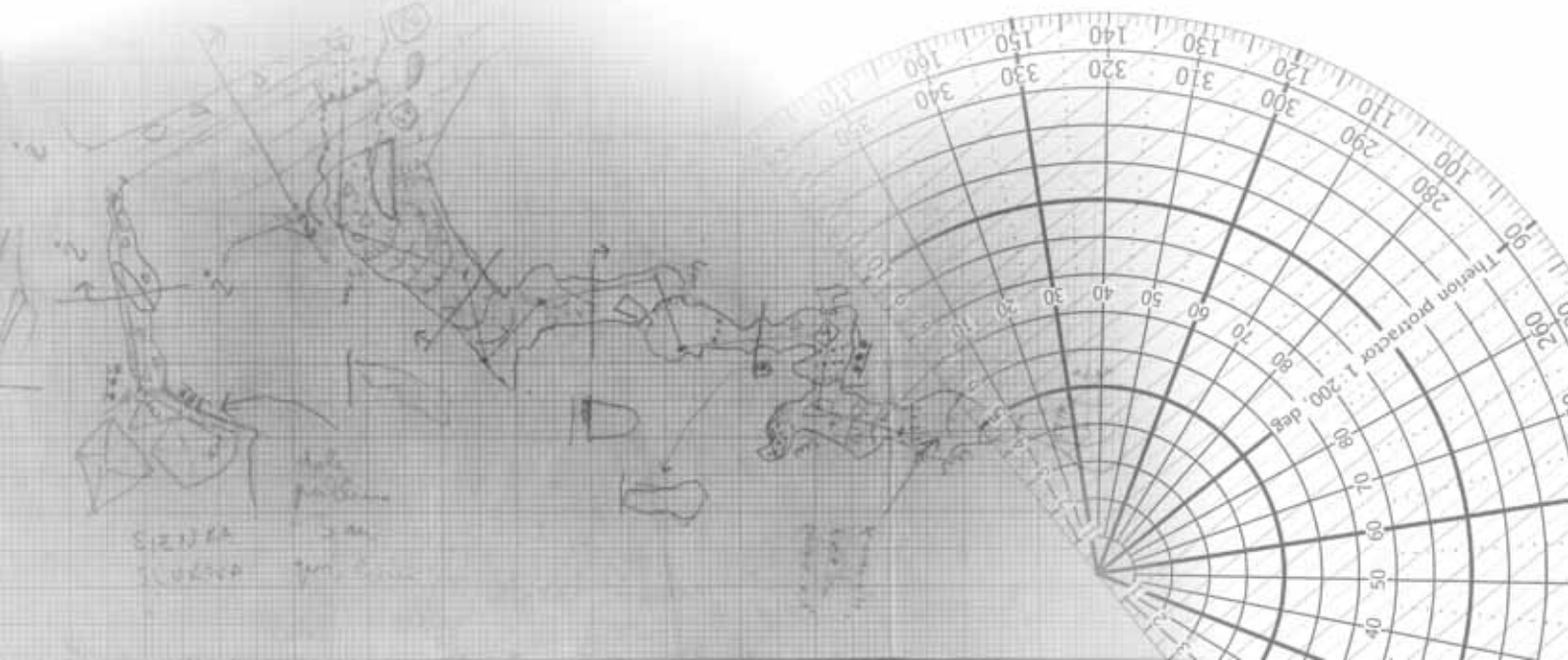
The Therion Book



Stacho Mudrák

Martin Budaj



Therion is copyrighted software. Distributed under the GNU General Public License.

Copyright © 1999–2025 Stacho Mudrák, Martin Budaj

This book describes Therion 6.4.1 (2026-11-03).

Code contributions by *Matěj Plch*, *Olly Betts*, *Marco Corvi*, *Vladimir Georgiev*, *Georg Pacher* and *Dimitrios Zachariadis*.

We owe thanks to *Martin Sluka*, *Ladislav Blažek*, *Martin Heller*, *Wookey*, *Olly Betts* and all users for their feedback, support and suggestions.

Translations (%):

Language	*XTherion*	*Map header*	*Loch*	*Translated by*
bg	81	77	100	Alexander Yanev, Ivo Tachev, Vladimir Georgiev
ca	–	80	–	Evaristo Quiroga
cs	83	75	–	Ladislav Blažek
de	97	74	–	Roger Schuster, Georg Pacher, Benedikt Hallinger
el	80	69	–	Stelios Zacharias
en[_GB|_US]	72	100	100	Stacho Mudrák, Olly Betts, Rodrigo Severo
es	100	79	–	Roman Muñoz, Evaristo Quiroga
fr	99	79	–	Eric Madelaine, Gilbert Fernandes
it	100	99	100	Marco Corvi, Francesco Bellamoli
mi	–	71	–	Kyle Davis, Bruce Mutton
pl	–	71	–	Krzysztof Dudziński
pt[_BR|_PT]	100	100	–	Rodrigo Severo, Toni Cavalheiro
rs	–	78	–	Ivana Miskovic
ru	97	77	–	Vasily V. Suhachev, Andrey Kozhenkov
sk	96	78	96	Stacho Mudrák
sl	100	80	96	Marko Zakrajsek
sq	80	69	–	Fatos Katallozi
zh	81	71	–	Zhang Yuan Hai, Duncan Collis

The cover picture shows survey sketch of *Hrozný kameňolom* Chamber in the Cave of Dead Bats in Slovakia and the map of it produced by Therion.

Table of Contents

Introduction 7
Why Therion? 7
Features 8
Software requirements 9
Installation 9
Setting-up an environment 10
How does it work? 10
First run 11
Creating data files 12
Basics 12
Data types 13
Coordinate systems 14
Magnetic declination 15
Data format 15
'encoding' 15
'input' 16
'survey' 16
'centreline' 17
'scrap' 23
'point' 25
'line' 29
'area' 33
'join' 34
'equate' 34
'map' 35
'surface' 36
'scan' 37
'import' 38
'grade' 39
'revise' 39
'require' 40
Custom attributes 40
Graphical interfaces 40
XTherion 40
XTherion—text editor 41
XTherion—map editor 41
Additional tools 44
Keyboard and mouse shortcuts in the Map editor 45
Thinking in Therion 47
How to enter a centreline 47
How to draw maps 48
How to create models 49
Therion in depth 49
How the map is put together 49
Processing data 52
Configuration file 52
'system' 52
'encoding' 52
'language' 52

'cs' . . . 52
'scrap-sort' . . . 53
'sketch-warp' . . . 53
'input' . . . 53
'source' . . . 53
'select' . . . 54
'unselect' . . . 54
'maps' . . . 55
'maps-offset' . . . 55
'log' . . . 55
'text' . . . 55
'layout' . . . 55
'lookup' . . . 63
'setup3d' . . . 64
'sketch-colors' . . . 65
'export' . . . 65
Running Therion . . . 67
XTherion—compiler . . . 69
What do we get . . . 70
Information files . . . 70
Log file . . . 70
XTherion . . . 70
SQL export . . . 70
Lists—caves, surveys, continuations . . . 72
2D maps . . . 72
Maps for printing . . . 72
Maps for GIS . . . 73
Special-purpose maps . . . 73
3D models . . . 73
Loch . . . 73
Changing layout of PDF maps . . . 74
Page layout in the atlas mode . . . 74
Page layout in the map mode . . . 80
Customizing text labels . . . 81
New map symbols . . . 81
Point symbols . . . 83
Line symbols . . . 83
Area symbols . . . 84
Special symbols . . . 85
Appendix . . . 86
Compilation . . . 86
Installing the dependencies . . . 86
Using CMake . . . 87
Hacker's guide . . . 88
Environment variables . . . 90
Initialization files . . . 90
Therion . . . 90
XTherion . . . 94
Limitations . . . 94
Example data . . . 95
History . . . 97
Future . . . 98
General . . . 98
2D maps . . . 98

3D models . . . 99
XTherion . . . 99
Loch . . . 99
Labyrinth . . . 99
Case studies . . . 100
Drawing maps in therion . . . 100
Listing caves . . . 100
Area symbols . . . 102
Sketch morphing . . . 102
Extended elevation control . . . 105
Stations in extended elevation scraps . . . 108
Displaying overlaying maps in offset . . . 109
Importing survex .3d files . . . 111
Using surveys specified in .th files . . . 111
Creating non-existing surveys . . . 112
Ignoring station prefixes . . . 113
Managing large projects . . . 114
Conclusion . . . 116
Question marks handling . . . 116
Question marks in centerline . . . 116
Question marks in maps . . . 117
Exporting question mark lists . . . 118
Using user defined symbol types . . . 119

LET NO ONE IGNORANT OF GEOMETRY ENTER HERE

ΑΓΕΩΜΕΤΡΗΤΟΣ ΜΗΔΕΙΣ ΕΙΣΙΤΩ

—alleged inscription over the entrance
of Plato's Academy, 4th century BC

Introduction

Therion is a tool for cave surveying. Its purpose is to help

- archive survey data on a computer in a form as close to the original notes and sketches as possible and retrieve them in a flexible and efficient way;
- draw a nice up-to-date plan or elevation map;
- create a realistic 3D model of the cave.

It runs on Unix, Linux, MacOS X and Win32 operating systems. Source code and Windows installer are available on the Therion web page (`https://therion.speleo.sk`).

Therion is distributed under the GNU General Public License.

Why Therion?

In the 1990s we did a lot of caving and cave surveying. Some computer programs existed which displayed survey shots and stations after loop closure and error elimination. These were a great help, especially for large and complicated cave systems. We used the output of one of them—TJIKPR—as a background layer with survey stations for hand-drawn maps. After finishing a huge 166-page Atlas of the Cave of Dead Bats in early 1997, we soon had a problem: we found new passages connecting between known passages and surveyed them. After processing in TJIKPR, the new loops influenced the position of the old surveys; most survey stations now had a slightly different position from before due to the changed error distribution. So we could either draw the whole Atlas again, or accept that the location of some places was not accurate—in the case of loops with a length of approximately 1 km there were sometimes errors of about 10 m—and try to distort the new passages to fit to old ones.

These problems remained when we tried to draw maps using some CAD programs in 1998 and 1999. It was always hard to add new surveys without adapting the old ones to the newly calculated positions of survey stations in the whole cave. We found no program that was able to draw an up-to-date complex map (i.e. not just survey shots with LRUD envelope), in which the old parts are modified according to the most recent known coordinates of survey stations.

In 1999 we began to think about creating our own program for map drawing. We knew about programs which were perfectly suited for particular sub-tasks. There was METAPOST, a high level programming language for vector graphics description, Survex for

excellent processing of survey shots, and TEX for typesetting the results. We only had to glue them together. By Xmas 1999 we had a minimalistic version of Therion working for the first time. This consisted only of about 32 kB of Perl scripts and METAPOST macros but served the purpose of showing that our ideas were implementable.

During 2000–2001 we searched for the optimal format of the input data, programming language, concept of interactive map editor and internal algorithms with the help of Martin Sluka (Prague) and Martin Heller (Zürich). In 2002 we were able to introduce the first really usable version of Therion, which met our requirements.

Features

Therion is a command-line application. It processes input files, which are—including 2D maps—in text format, and creates files with 2D maps or 3D model as the output.

The syntax of input files is described in detail in later chapters. You may create these files in an arbitrary plain text editor like *ed* or *vi*. They contain instructions for Therion, e.g.

```
point 1303 1004 pillar
```

where `point` is a keyword for point symbol followed by its coordinates and a symbol type specification.

Hand-editing of such files is not easy—especially when you draw maps, you need to think in spatial (Cartesian coordinate) terms. Thus there is a special GUI for Therion called XTherion. XTherion works as an advanced text editor, map editor (where maps are drawn fully interactively) and compiler (which runs Therion on the data).

It may look quite complicated, but this approach has a lot of advantages:

- There is strict separation of data and visualization. The data files specify only where the objects are, not what they look like. The visual representation is added by METAPOST in later phases of data processing. (It's a very similar concept to XML data representation.)

  This makes it possible to change map symbols used without changing the input data, or merge more maps created by different people in different styles into one map with unified map symbols set.

  2D maps are adapted for particular output scale (level of abstraction, non-linear scaling of symbols and texts)

- All data is relative to survey station positions. If the coordinates of survey stations are changed in the process of loop closure, then all relevant data is moved correspondingly, so the map is always up-to-date.

- Therion is not dependent on particular operating system, character encoding or input files editor; input files will remain human readable

- It's possible to add new output formats
- 3D model is generated from 2D maps to get a realistic 3D model without entering too much data
- although the support for WYSIWYG is limited, you get what you want

Software requirements

"A program should do one thing, and do it well." (Ken Thompson) Therefore we use some valuable external programs, which are related to the problems of typesetting and data visualization. Therion can then do its task much better than if it were a standalone application in which you could calibrate your printer or scanner and, with one click, send e-mail with your data.

Therion needs:

- PROJ library.
- TEX distribution. Necessary only if you want to create 2D maps in PDF or SVG format.
- Tcl/Tk with *BWidget* and optionally *tkImg* extension. It is only required for XTherion.
- LCDF Typetools if you want to use easy setup for custom fonts in PDF maps
- *ghostscript* if you want to create calibrated images from georeferenced PDF maps.

Windows installer includes all required packages with the exception of ghostscript. Read the *Appendix* if you want to compile Therion yourself.

For displaying maps and models you may use any of the following programs:

- any PDF or SVG viewer displaying 2D maps;
- any GIS supporting DXF or *shapefile* formats for analyzing the maps;
- appropriate 3D viewer for models exported in other than default format;
- any SQL database client to process the exported database.

Installation

Installation from sources (therion-6.*.*.tar.gz package):

The source code is a primary Therion distribution. It needs to be compiled and installed according to the instructions in the *Appendix*.

Installation on Windows:

Run the setup program and follow the instructions. It installs all the required dependencies and creates shortcuts to XTherion and Therion Book.

Windows 7 and 8 users may need to install the Universal CRT libraries[1] to run Therion, unless they have already been installed through system updates.

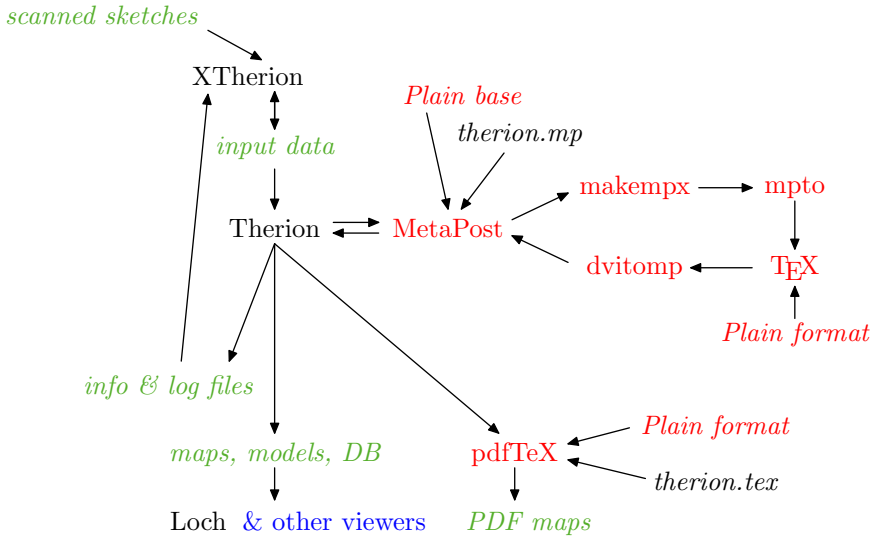
Setting-up an environment

Therion reads settings from the initialization file. Default settings should work fine for users using just latin characters[2], standard TEX and METAPOST.

If you want to use your own fonts for latin or non-latin characters in PDF maps, edit the initialization file. Instructions on how to do this are in the *Appendix*.

How does it work?

So, now it's clear what Therion needs, let's have a look at the way it interacts with all these programs:



DON'T PANIC! When your system is set-up correctly the majority of this is hidden from the user and all necessary programs are run automatically by Therion.

For working with Therion it is enough to know that you have to create input data (best done with XTherion), run Therion, and display the output files (3D model, map, log file) in the appropriate program.

[1] https://learn.microsoft.com/en-us/cpp/windows/universal-crt-deployment

[2] In the PDF map Therion renders most of the accented characters as a combination of accent and a base character. Some obscure accents might be omitted. Precomposed accented letters are included for Slovak and Czech languages.

For those who want to understand more about it, here is a brief explanation of the above flowchart. Program names are in roman font, data files in italics. Arrows show data flow between programs. Temporary data files are not shown. The meaning of colours:

- black—Therion programs and macros (XTherion is written in Tcl/Tk, so it needs this interpreter to run)
- red—TEX package
- green—input files created by the user and output files created by Therion

Therion itself does the main task. It reads the input files, interprets them, finds closed loops and distributes errors. Next it transforms all other data (e.g. 2D maps) according to new stations position. Therion exports data for 2D maps in METAPOST format. METAPOST gives the actual shape to abstract map symbols according to map symbol definitions; it creates a lot of PostScript files with small fragments of the cave. These are read back and converted to a PDF-like format, which forms input data for pdfTEX. PdfTEX does all the typesetting and creates a PDF file of the cave map.

Therion also exports 3D models (full or centreline) in various formats.

Centreline may be exported for further processing in any SQL database.

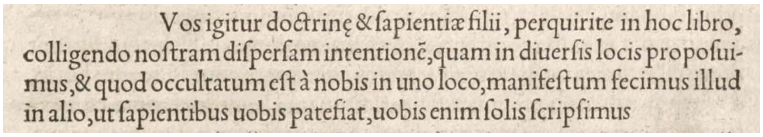
First run

After explaining the basic principles of Therion it's a good idea to try it on the example data.

- Download the sample data from Therion web page and unpack it somewhere on your computer's hard drive.
- Run XTherion (under Unix and MacOS X by typing 'xtherion' in the command line, under Windows there is a shortcut in the *Start* menu).
- Open the file 'thconfig' from the sample data directory in the 'Compiler' window of XTherion
- Press 'F9' or 'compile' in the menu to run Therion on the data—you'll get some messages from Therion, METAPOST and TEX.
- PDF maps and 3D models are created in the data directory.

Additionally, you may open survey data files (*.th) in the 'Text editor' window and map data files (*.th2) in the 'Map editor' window of XTherion. Although the data format may look confusing at first, it will be explained in the following chapters.

Only for you, children of doctrine and learning, have we written this work. Examine this book, ponder the meaning we have dispersed in various places and gathered again; what we have concealed in one place we have disclosed in another, that it may be understood by your wisdom.


Vos igitur doctrinę & ſapientiæ filii, perquirite in hoc libro, colligendo noſtram diſperſam intentionē, quam in diuerſis locis propoſuimus, & quod occultatum eſt à nobis in uno loco, manifeſtum fecimus illud in alio, ut ſapientibus uobis patefiat, uobis enim ſolis ſcripſimus

—Henricus C. Agrippa ab Nettesheym, 1533

Creating data files

Basics

The input files for Therion are in text format. There are a few rules about how such a file should look:

- There are two kinds of commands. One-line commands and multi-line commands.
- A one-line command is terminated by an end of line character. The syntax of these is

  ```
  command arg1 ... argN [-option1 value1 -option2 value2 ...]
  ```

  where *arg1 ... argN* are obligatory arguments, and pairs *-option value* are options, which you may freely omit. Which arguments and options are available depends on the particular command. An example may be

  ```
  point 643.5 505.0 gradient -orientation 144.7
  ```

  with three obligatory arguments and one optional option/value pair. Sometimes options have no or multiple values.

- Multi-line commands begin similarly to one line commands, but continue on subsequent lines until explicit command termination. These lines may contain either data or options, which are applied to subsequent data. If a data line starts with a word reserved for an option, you have to insert '`!`' in front of it. The syntax is

  ```
  command arg1 ... argN [-option1 value1 -option2 value2 ...]
    ...
    optionX valueX
    data
    ...
  endcommand
  ```

  Again, for better illustration, a real example follows:

  ```
  line wall -id walltobereferenced
    1174.0 744.5
    1194.0 756.5 1192.5 757.5 1176.0 791.0
    smooth off
    1205.5 788.0 1195.5 832.5 1173.5 879.0
  ```

`endline`

This command `line` has one obligatory argument, a line type (passage wall in this case), followed by one option. The next two lines contain data (coordinates of Bézier curves to be drawn). The next line ("`smooth off`") specifies an option which applies to subsequent data (i.e. not for the whole line, unlike the option `-id` in the first line) and the last line contains some more data.

- if the value of an option or argument contains spaces, you should enclose this value in `" "` or `[ ]`. If you want to put a double-quote `"` into text in `" "` you need to insert it twice. Quotes are used for strings; brackets for numerical values and keywords.
- each line ending with a backslash (`\`) is considered to continue on the next line, as if there was neither line-break nor backlash.
- everything that follows `#`, until the end of line—even inside a command—is considered to be a comment, and is ignored.
- multiline comments inside `comment` ... `endcomment` block are allowed in data and configuration files

Data types

Therion uses following data types:

- *keyword* ▷ a sequence of `A-Z`, `a-z`, `0-9` and `_-/` characters (not starting with '`-`').
- *ext_keyword* ▷ keyword that can also contain `+*.,'` characters, but not on the first position.
- *restr_keyword* ▷ keyword limited to `A-Z`, `a-z` and `_` characters because of METAPOST limitations
- *date* ▷ a date (or a time interval) specification in the format `YYYY[.MM[.DD[@HH[:MM[:SS[.SS]]]]]] [- YYYY[.MM[.DD[@HH[:MM[:SS[.SS]]]]]]]` or '`-`' to leave a date unspecified.
- *person* ▷ a person's first name and surname separated by whitespace characters. Use '`/`' to separate first name and surname if there are more names.
- *string* ▷ a sequence of any characters. Strings may contain special tag `<lang:XX>` to separate translations. In multilingual strings only the text between `<lang:XX>` (where `XX` is the language selected in initialization or configuration file) and the next `<lang:YY>` tag is displayed on the output. If no match is found, everything before any occurrence of `<lang:ZZ>` tag is displayed.
- *units* ▷ length units supported: meter[s], centimeter[s], inch[es], feet[s], yard[s] (also m, cm, in, ft, yd). Angle units supported: degree[s], minute[s] (also deg, min), grad[s], mil[s], percent[age] (clino only). A degree value may be entered in decimal notation ($x.y$) or in a special notation for degrees, minutes and seconds (deg[:min[:sec]]).

Coordinate systems

Therion supports coordinate transformations in geodetic coordinate systems. You can specify `cs` option in `centreline`, `surface`, `import` and `layout` objects and then enter XY data in given system. You can also specify output `cs` in configuration file.

If you do not specify any `cs` in your dataset, it is assumed you are working in `local` coordinate system, and no conversions are done. If you specify `cs` anywhere in the data, you have to specify it for all location data (`fix`, `origin` in `layout` etc.).

`cs` applies to all subsequent location data until other `cs` is specified or until the end of the current object, whichever comes first.

Following coordinate systems are supported:

- `UTM1` – `UTM60` ▷ Universal Transverse Mercator in northern hemisphere and given zone, WGS84 datum. Equivalent to EPSG:32601–EPSG:32660.
- `UTM1N` – `UTM60N` ▷ same as `UTM1` – `UTM60`
- `UTM1S` – `UTM60S` ▷ UTM in southern hemisphere, WGS84 datum. Equivalent to EPSG:32701–EPSG:32760.
- `lat-long`, `long-lat` ▷ latitude (N positive, S negative) and longitude (E positive, W negative) in given order in degrees (*deg*[:*min*[:*sec*]] allowed), WGS84 datum. Not supported on output. Equivalent to EPSG:4326.
- `EPSG:<number>` ▷ Most of EPSG coordinate systems. Almost every coordinate system used worldwide has its own EPSG number. To find the number of your system, see `https://epsg.org`.
- `ESRI:<number>` ▷ Similar to EPSG, but ESRI standard.
- `ETRS` ▷ European Terrestrial Reference System 1989 (ETRS89); long-lat order, not supported on output. Equivalent to EPSG:4258.
- `ETRS28` – `ETRS37` ▷ ETRS89 zones in UTM projection; east-north order. Equivalent to EPSG:25828–EPSG:25837.
- `JTSK`, `iJTSK` ▷ Czechoslovak S-JTSK system used since 1920s with south and west axis (JTSK) and its modified version with axis pointing east and north and negative numbers (iJTSK). JTSK is not supported on output (iJTSK is).
- `JTSK03`, `iJTSK03` ▷ new S-JTSK realisation introduced in Slovakia in 2011.
- `OSGB:<H, N, O, S or T><A-Z except I>` ▷ British Ordnance Survey National Grid.
- `S-MERC` ▷ the spherical Mercator projection, as used by various online mapping sites. Equivalent to EPSG:3857.

Magnetic declination

Therion contains built-in IGRF[3] Earth geomagnetic field model valid for period 1900–2025. It is automatically used if the cave is located in space with a `fix` station using any of the supported geodetic coordinate systems and in time with the centerlines `date` command. The computed declination is listed in the LOG file for information.

If the user specified a `declination` in the centerline, that value takes precedence over the automatic calculation.

Data format

The syntax of input files is explained in the description of individual commands. Studying the example files distributed with Therion will help you understand. See also an example in the *Appendix*.

Each of the following sections describes one Therion command using the following structure:

Description: notes concerning this command.

Syntax: schematic syntax description.

Context: specifies the context in which is this command allowed. The *survey* context means that the command must be enclosed by `survey ... endsurvey` pair. The *scrap* context means that the command must be enclosed within `scrap ... endscrap` pair. Context *all* means that the command may be used anywhere.

Arguments: a list of the obligatory arguments with explanations.

Options: a list of the available options.

Command-like options: options for multi-line commands, which can be specified among the data lines.

'encoding'

Description: sets the encoding of input file. This allows the use of non-ASCII characters in input files.

Syntax: `encoding <encoding-name>`

Context: It should be the very first command in the file.

[3] See https://www.ngdc.noaa.gov/IAGA/vmod/

Arguments:

- `<encoding-name>` ▷ to see a list of all the supported encoding names, run Therion with `--print-encodings` option. 'UTF-8' (Unicode) and 'ASCII' (7 bit) encodings are always supported.

'input'

Description: inserts the contents of a file in place of the command. Default extension is '.th' and may be omitted. For greatest portability use relative paths and Unix slashes '/', not Windows backslashes '\', as directory separators.

Syntax: `input <file-name>`

Context: all

Arguments:

- `<file-name>`

'survey'

Description: Survey is the main data structure. Surveys may be nested—this allows a hierarchical structure to be built. Usually some level of this hierarchical structure survey represents caves, higher levels karst areas and lower levels e.g. passages.

Each survey has its own namespace specified by its `<id>` argument. Objects (like survey stations or scraps; see below) which belong to a subsurvey of the current survey are referenced as

`<object-id>@<subsurvey-id>`,

or, if there are more nesting levels

`<object-id>@<subsubsurvey-id>.<subsurvey-id>`.[4]

This means, that object identifiers must be unique only in the scope of one survey. For instance, survey stations names can be the same if they are in different surveys. This allows stations to be numbered from 0 in each survey or the joining of two caves into one cave system without renaming survey stations.

Syntax:
```
survey <id> [OPTIONS]
... other therion objects ...
endsurvey [<id>]
```

Context: none, survey

Arguments:

- `<id>` ▷ survey identifier

[4] Note: it's not possible to reference any object from the higher-level surveys.

Options:

- `namespace <on/off>` ▷ specify whether survey creates namespace (default: `on`)
- `declination <specification>` ▷ set the default declination for all data objects in this survey (which can be overridden by declination definitions in subsurveys). The `<specification>` has three forms:

  1. `[]` an empty string. This will reset the declination definition.

  2. `[<value> <units>]` will set a single value (also for undated surveys).

  3. `[<date1> <value1> [<date2> <value2> ... ] <units>]` will set declination for several dates. Then the declination of each shot will be set according to the date specification of the data object. If you want to explicitly set the declination for undated survey data, use '`-`' instead of date.

  If no declination is specified and some geodetic coordinate system is defined, the declination is automatically computed using built-in geomagnetic model.

  N.B.: The declination is positive when the magnetic north is east of true north.
- `person-rename <old name> <new name>` ▷ rename a person whose name has been changed
- `title <string>` ▷ description of the object
- `entrance <station-name>` ▷ specifies the main entrance to the cave represented by this survey. If not specified and there is exactly one station marked entrance in this survey, it is considered to represent a cave also. This information is used for `cave-list` export.

'centreline'

Description: Survey data (centreline) specification. The syntax is borrowed from Survex with minor modifications; the Survex manual may be useful as an additional reference for the user. A synonym term 'centerline' may be used.

Syntax: `centreline [OPTIONS]`

```
date <date>
team <person> [<roles>]
explo-date <date>
explo-team <person>
instrument <quantity list> <description>
calibrate <quantity list> <zero error> [<scale>]
units <quantity list> [<factor>] <units>
sd <quantity list> <value> <units>
grade <grade list>
declination <value> <units>
```

```
    grid-angle <value> <units>
    infer <what> <on/off>
    mark <type>
    flags <shot flags>
    station <station> <comment> [<flags>]
    cs <coordinate system>
    fix <station> [<x> <y> <z> [<std x> <std y> <std z>]]
    equate <station list>
    data <style> <readings order>
    break
    group
    endgroup
    walls <auto/on/off>
    vthreshold <number> <units>
    extend <spec> [<station> [<station> [<station>]]]
    station-names <prefix> <suffix>
    ...
    [SURVEY DATA]
    ...
  endcentreline
```

Context: none, survey

Options:

- `id <ext_keyword>` ▷ id of the object
- `author <date> <person>` ▷ author of the data and its creation date
- `copyright <date> <string>` ▷ copyright date and name
- `title <string>` ▷ description of the object

Command-like options:

- `date <date>` ▷ survey date. If multiple dates are specified, a time interval is created.
- `explo-date <date>` ▷ discovery date. If multiple dates are specified, a time interval is created.
- `team <person> [<roles>]` ▷ a survey team member. The first argument is his/her name, the others describe the roles of the person in the team (optional—currently not used). The following role keywords are supported:

role	*alias*	*intended meaning*
tape	length	Measured leg lengths
compass	bearing	Measured bearings
clino	gradient	Measured vertical angles
backtape	backlength	Like `tape` but for backsights

backcompass	backbearing	Like `compass` but for backsights	
backclino	backgradient	Like `clino` but for backsights	
instruments	insts	All instruments: both compass and clino; use for all-in-one instruments such as DistoX.	
counter	count	Topofil length measurements	
depth		Measured differences in height between stations, e.g. underwater with a diver's depth gauge, or above water with a manometer	
station		Added markers at stations	
position	gps	Recorded absolute positions of stations (e.g. fixed surface stations with a GPS)	*6.3.5*
notes	notebook	Recorded instrument readings	
pictures	pics	Drew sketches	
assistant	dog	General helper (e.g. held the end of the tape on stations)	
altitude	dz	Recorded the altitudes of stations (e.g. with an altimeter)	
dimensions		Measured all passage dimensions	
left		Measured `left` passage dimension	
right		Measured `right` passage dimension	
up	ceiling	Measured `up` passage dimension	
down	floor	Measured `down` passage dimension	
explorer		Explored the area of cave being surveyed	

- `explo-team <person>` ▷ a discovery team member.
- `instrument <quantity list> <description>` ▷ description of the instrument that was used to survey the given quantities (same keywords as team person's role)
- `infer <what> <on/off>` ▷ '`infer plumbs on`' tells the program to interpret gradients $\pm 90^\circ$ as UP/DOWN (this means no clino corrections are applied). '`infer equates on`' will case program to interpret shots with 0 length as equate commands (which means that no tape corrections are applied)
- `declination <value> <units>` ▷ sets the declination for subsequent shots

$$\textit{true bearing} = \textit{measured bearing} + \textit{declination}.$$

The declination is positive when the magnetic north is east of true north. If no declination is specified, or the declination is reset (`-`), then a valid declination specification is searched for in all surveys the data object is in. See declination option of survey command.

- `grid-angle <value> <units>` ▷ specifies the magnetic grid angle (declination against grid north).

- `sd <quantity list> <value> <units>` ▷ sets the standard deviation for the given measurements. The Quantity list can contain the following keywords: length, tape, bearing, compass, gradient, clino, counter, depth, x, y, z, position, easting, dx, northing, dy, altitude, dz.
- `grade <grade list>` ▷ sets standard deviations according to the survey grade specification (see grade command). All previously specified standard deviations or grades are lost. If you want to change an SD, use the sd option after this command. If multiple grades are specified, only the last one applies. You can specify grades only for position or only for surveys. If you want to combine them, you must use them in one grade line.
- `units <quantity list> [<factor>] <units>` ▷ set the units for given measurements (same quantities as for sd).
- `calibrate <quantity list> <zero error> [<scale>]` ▷ set the instrument calibration. The measured value is calculated using the following formula: $measured\ value = (read\ value - zero\ error) \times scale$. The supported quantities are the same as sd.
- `break` ▷ can be used with interleaved data to separate two traverses
- `mark [<station list>] <type>` ▷ set the type of named stations. `<type>` is one of: fixed, painted and temporary (default). If there is no station list, all subsequent stations are marked.
- `flags <shot flags>` ▷ set flags for following shots. The supported flags are: `surface` (for surface measurements), `duplicate` (for duplicated surveys), `splay` (for short side legs that are hidden in maps and models by default) and `artificial` (for man-made passages, such as dug or blasted connections). These are excluded from length calculations. *6.4.1*

  `artificial` shots are drawn and modelled like any other shot, and they take part in loop closure. Only the length statistics ignore them. A shot that already carries `surface`, `duplicate` or `splay` keeps being counted with that flag; adding `artificial` to it changes nothing.

  All shots having one of the stations named either '`.`' or '`-`' are `splay` shots by default (see also `data` command).

  If flag is set to `approx[imate]`, it is included to total length calculations, but also displayed separately in survey statistics. It should be used for shots, that were not surveyed properly and need to be resurveyed.

  Also "`not`" is allowed before a flag.
- `station <station> <comment> [<flags>]` ▷ set the station comment and its flags. If `""` is specified as a comment, it is ignored.

  Supported flags: `entrance`, `continuation`, `air-draught[:winter/summer]`, `sink`, `spring`, `doline`, `dig`, `arch`, `overhang`. Also `not` is allowed before a flag, to remove previously added flag.

You can also specify custom attributes to the station using `attr` flag followed by attribute name and value. Example:
`station 4 "pit to explore" continuation attr code "V"`

If there is a passage, that was explored, but not surveyed yet, estimated explored length of this passage can be added to the station with `continuation` flag. Just add `explored <explored-length>` to the station flags. Explored lengths are a part of survey/cave statistics, displayed separately. Example:
`station 40 "ugly crollway" continuation explored 100m`

- `cs <coordinate system>` ▷ coordinate system for stations with fixed coordinates
- `fix <station> [<x> <y> <z> [<std x> <std y> <std z>]]` ▷ fix station coordinates (with specified errors—only the units transformation, not calibration, is applied to them).
- `equate <station list>` ▷ set points that are equivalent
- `data <style> <readings order>` ▷ set data style (normal, topofil, diving, cartesian, cylpolar, dimensions, nosurvey) and readings order. Reading is one of the following keywords: station, from, to, [back]tape/[back]length, [back]compass/[back]bearing, [back]clino/[back]gradient, depth, fromdepth, todepth, depthchange, counter, fromcount, tocount, northing, easting, altitude, up/ceiling[5], down/floor, left, right, ignore, ignoreall.

  See Survex manual for details.

  For interleaved data both newline and direction keywords are supported. If backward and forward compass or clino reading are given, the average of them is computed.

  If one of the shot stations is named either '`.`' or '`-`', the shot has `splay` attribute set. *Dot* should be used for shots ending on features inside passage, *dash* for shots ending on passage walls, floor or ceiling. Although Therion makes no distinction between them yet, it should be used to improve 3D modeling in the future.
- `group`
- `endgroup` ▷ `group`/`endgroup` pair enables the user to make temporary changes in almost any setting (calibrate, units, sd, data, flags...)
- `walls <auto/on/off>` ▷ turn on/off passage shape generation from LRUD data for subsequent shots. If set `auto`, passage is generated only if there is no scrap referencing given centreline.
- `vthreshold <number> <units>` ▷ threshold for interpreting LRUD readings as left-right-front-back reading perpendicular to the shot.

  If passages are horizontal (`inclination < vthreshold`), LR is perpendicular to the shot and UD is vertical.

[5] dimension may be specified as a pair `[<from> <to>]`, meaning the size at the beginning and end of the shot

If passages are more or less vertical (`inclination > vthreshold`), even UD becomes perpendicular to the shot – otherwise passages would not look very good. In the case of vertical shots, UD is interpreted as north-south dimension from the station to allow tube-like modeling of verticals.

- `extend <spec> [<station> [<station> [<station>]]]` ▷ control how the centerline is extended. `<spec>` may be one of the following keywords

  `normal/reverse` ▷ extend given and following stations to the same/reverse direction as previous station. If station names are given, direction is applied only to the specified station or shot.

  `left/right` ▷ same as above, but direction is specified explicitly.

  `vertical` ▷ do not move station (shot) in X direction, use only Z component of the shot

  `start` ▷ specify starting station (shot)

  `ignore` ▷ ignore specified station (shot), continue extended elevation with other station (shot) if possible

  `hide` ▷ do not show specified station (shot) in extended elevation

  or a percentage in the range `0` to `200`. This sets the extend ratio, i.e. the factor used for the horizontal component of a leg in extended elevation. Therion keeps the vertical component unchanged and multiplies only the horizontal displacement by the ratio. Thus `100` means the normal width of the extended leg, `50` compresses it to one half, `200` stretches it to double width, and `0` suppresses any horizontal displacement (effectively the same result as `vertical` for that leg).

  The optional station arguments are interpreted as follows:

  `extend <spec>` ▷ set the default extend behaviour for subsequent shots in the current centreline.

  `extend <spec> <from>` ▷ apply the extend setting to a station.

  `extend <spec> <from> <to>` ▷ apply the extend setting to a specific shot.

  `extend <spec> <before> <from> <to>` ▷ apply the extend setting to a specific shot, with an additional `before` station condition. This form is primarily useful with `ignore`, where the shot is ignored only if the extended-elevation traversal reaches `<from>` from `<before>`.

  Station names use the normal Therion object reference syntax, so `station@survey` may be used.

- `station-names <prefix> <suffix>` ▷ adds given prefix/suffix to all survey stations in the current centreline. Saves some typing.

'scrap'

Description: Scrap is a piece of 2D map, which doesn't contain overlapping passages (i.e. all the passages may be drawn on the paper without overlapping). For small and simple caves, the whole cave may belong to one scrap. In complicated systems, a scrap is usually one chamber or one passage. Ideally, a scrap contains about 100 m of the cave.[6] Each scrap is processed separately by METAPOST; scraps which are too large may exceed METAPOST's memory and cause errors.

Scrap consists of point, line and area map symbols. See chapter *How the map is put together* for explanation how and in which order are they displayed.

Scrap border consists of lines with the `-outline out` or `-outline in` options (passage walls have `-outline out` by default). These lines shouldn't intersect—otherwise Therion (METAPOST) can't determine the interior of the scrap and METAPOST issues a warning message "`scrap outline intersects itself`".

Each scrap has its own local cartesian coordinate system, which usually corresponds with the millimeter paper (if you measure the coordinates of map symbols by hand) or pixels of the scanned image (if you use XTherion). Therion does the transformation from this local coordinate system to the real coordinates using the positions of survey stations, which are specified both in the scrap as point map symbols and in centreline data. If the scrap doesn't contain at least two survey stations with the `-name` reference, you have to use the `-scale` option for calibrating the scrap. (This is usual for cross sections.)

The transformation consists of the following steps:

- Linear transformation (shifting, scaling and rotation) which 'best' fits stations drawn in the scrap to real ones. 'Best' means that the sum of squared distances between corresponding stations before and after transformation is minimal. The result is displayed red if `debug` option of the `layout` command is set `on`.
- Non-linear transformation of the scrap which (1) moves survey stations to their correct position, (2) is continuous. Displayed blue in the `debug` mode.
- Non-linear transformation of the scrap which (1) moves joined points together, (2) doesn't move survey stations, (3) is continuous. Finally the position of curves' control points is adjusted to preserve smoothness. The result is final map.

[6] If necessary, scraps may be much smaller—just to display a few meters of the cave. *When deciding about scrap size please take into account the following:* Using small scraps may take more time for cartographer to optimize scrap joins. On the other hand smaller scraps will probably be less distorted by map warping algorithms than larger scraps. Using too large scraps may exhaust METAPOST's memory if passage fills are frequently used and the map editor in XTherion is much less responsive when editing huge scraps.

Syntax: `scrap <id> [OPTIONS]`
`... point, line and area commands ...`
`endscrap [<id>]`

Context: none, survey

Arguments:

- `<id>` ▷ scrap identifier

Options:

- `projection <specification>` ▷ specifies the drawing projection. Each projection is identified by a type and optionally by an index in the form `type[:index]`. The index can be any keyword. The following projection types are supported:

  1. `none` ▷ no projection, used for cross sections or maps that are independent of survey data (e.g. digitization of old maps where no centreline data are available). No index is allowed for this projection.

  2. `plan` ▷ basic plan projection (default).

  3. `elevation` ▷ orthogonal projection (a.k.a. projected profile) which optionally takes a view direction as an argument (e.g. `[elevation 10]` or `[elevation 10 deg]`).

  4. `extended` ▷ extended elevation (a.k.a. extended profile).

- `scale <specification>` ▷ is used to pre-scale (convert coordinates from pixels to meters) the scrap data. If scrap projection is none, this is the only transformation that is done with coordinates. The `<specification>` has four forms:

  1. `<number>` ▷ `<number>` meters per drawing unit.

  2. `[<number> <length units>]` ▷ `<number> <length units>` per drawing unit.

  3. `[<num1> <num2> <length units>]` ▷ `<num1>` drawing units corresponds to `<num2> <length units>` in reality.

  4. `[<num1> ... <num8> [<length units>]]` ▷ this is the most general format, where you specify, in order, the x and y coordinates of two points in the scrap and two points in reality. Optionally, you can also specify units for the coordinates of the 'points in reality'. This form allows you to apply both scaling and rotation to the scrap.

- `cs <coordinate system>` ▷ assumes that (calibrated) local scrap coordinates are given in specified coordinate system. It is useful for absolute placing of imported sketches where no survey stations are specified.[7]

- `stations <list of station names>` ▷ stations you want to plot to the scrap, but which are not used for scrap transformation. You don't have to specify (draw) them with the `point station` command.

[7] If there are some survey stations in the scrap, the `cs` specification is ignored.

- `sketch <filename> <x> <y>` ▷ underlying sketch bitmap specification (lower left corner coordinates).
- `walls <on/off/auto>` ▷ specify if the scrap should be used in 3D model reconstruction
- `flip none/horizontal/vertical` ▷ flips the scrap after scale transformation (default: `none`).
- `station-names <prefix> <suffix>` ▷ adds given prefix/suffix to all survey stations in the current scrap. Saves some typing.
- `author <date> <person>` ▷ author of the data and its creation date
- `copyright <date> <string>` ▷ copyright date and name
- `title <string>` ▷ description of the object

'point'

Description: Point is a command for drawing a point map symbol.

Syntax: `point <x> <y> <type> [OPTIONS]`

Context: scrap

Arguments:

- `<x>` and `<y>` are the drawing coordinates of an object.
- `<type>` determines the type of an object. The following types are supported:

  special objects: `dimensions`[8], `section`[9], `station`[10];

  labels: `altitude`[11], `date`[12], `height`[13], `label`, `passage-height`[14], `remark`, `station-name`[15];

  symbolic passage fills:[16] `bedrock`, `blocks`, `clay`, `debris`, `guano`, `ice`, `mudcrack`, `mud`, `pebbles`, `raft`, `sand`, `snow`, `water`;

[8] Use `-value` option to specify passage dimensions above/below centerline plane used while creating 3D model.

[9] `section` is an anchor for placing the cross-section at this point. This symbol has no visual representation. The cross section must be in the separate scrap. You can specify it through the `-scrap` option.

[10] Survey station. For each scrap (with the exception of scraps in 'none' projection) at least one station with station reference (`-name` option) has to be specified.

[11] General altitude label. All altitudes are exported as a difference against grid Z origin (which is 0 by default). To display altitude on the passage wall, use `altitude` option for any line point of the passage wall

[12] Set date's value with the `-value` option

[13] Height of formations inside of the passage (like pit etc.); see below for details.

[14] Height of the passage; see below for details.

[15] If no text is specified, the name of the nearest station is used.

[16] Unlike other point symbols, these are clipped by the scrap border. See the chapter *How the map is put together*.

speleothems: anastomosis, aragonite, cave-pearl, clay-tree, crystal, curtains, curtain, disc-pillar, disc-stalactite, disc-stalagmite, disc-pillars, disc-stalactites, disc-stalagmites, disk, flowstone, flute, gypsum-flower, gypsum, helictites, helictite, karren, moonmilk, pendant, pillar-with-curtains, pillars-with-curtains, pillar, popcorn, raft-cone, rimstone-dam, rimstone-pool, scallop, soda-straw, stalactite-stalagmite, stalactites-stalagmites, stalactite, stalactites, stalagmite, stalagmites, volcano, wall-calcite;

equipment: anchor, bridge, camp, fixed-ladder, gate, handrail, masonry, name-plate, no-equipment, no-wheelchair, rope-ladder, rope, steps, traverse, via-ferrata, walkway, wheelchair;

passage ends: breakdown-choke, clay-choke, continuation, entrance, flowstone-choke, low-end, narrow-end;

others: air-draught[17], altar, archeo-excavation, archeo-material, audio, bat, bones, borehole, danger, dig, electric-light, ex-voto, extra[18], gradient, human-bones, ice-pillar, ice-stalactite, ice-stalagmite, map-connection[19], paleo-material, photo, root, seed-germination, sink, spring[20], tree-trunk, u[21], vegetable-debris, water-drip, water-flow.

Options:

- subtype <restr_keyword> ▷ determines the object's subtype. The following subtypes for given types are supported:

  station:[22] temporary (default), painted, natural, fixed;

  air-draught: winter, summer, undefined (default);

  water-flow: permanent (default), intermittent, paleo.

  The subtype may be specified also directly in <type> specification using ':' as a separator.[23]

  Any subtype specification can be used with user defined type (u). In this case you need also to define corresponding metapost symbol (see the chapter *New map symbols*).

- orientation/orient <number> ▷ defines the orientation of the symbol. If not specified, it's oriented to north. $0 \leq$ number < 360.

- align ▷ alignment of the symbol or text. The following values are accepted: center, c, top, t, bottom, b, left, l, right, r, top-left, tl, top-right, tr, bottom-left, bl, bottom-right, br (default: center).

[17] Number of ticks is set according to -scale option

[18] Additional morphing point. See -dist and -value options.

[19] Virtual point, used to indicate connection between shifted maps (extended elevation, map offset).

[20] Always use spring and sink symbols with a water-flow arrow.

[21] For user defined point symbols.

[22] if station subtype is not specified, Therion reads it from centreline, if it's specified there

[23] E.g. station:fixed

- `scale` ▷ symbol scale, can be: tiny (xs), small (s), normal (m), large (l), huge (xl) or a numeric value (default: `normal`). Named sizes scale by $\sqrt{2}$, so that $xs \equiv 0.5$, $s \equiv 0.707$, $m \equiv 1.0$, $l \equiv 1.414$ and $xl \equiv 2.0$.
- `place <bottom/default/top>` ▷ changes displaying order in the map (default: `default`).
- `clip <on/off>` ▷ specify whether a symbol is clipped by the scrap border. You cannot specify this option for the following symbols: station, station-name, label, remark, date, altitude, height, passage-height (default: `on`).
- `visibility <on/off>` ▷ displays/hides the symbol (default: `on`).
- `context <point/line/area> <symbol-type>` ▷ (to be used with `symbol-hide` and `symbol-show` layout options) symbol will be hidden/shown according to rules for specified `<symbol-type>`.[24]
- `id <ext_keyword>` ▷ ID of the symbol.

Type-specific options:

- `dist <distance>` ▷ if the point type is extra, specifies the distance to the nearest station (or station specified using `-from` option. If not specified, appropriate value from LRUD data is used.
- `from <station>` ▷ specifies a reference station. This option is primarily intended for points of type `extra`. If omitted, Therion uses the nearest station, and if `-dist` is not specified, an appropriate value from LRUD data is used.

  If the point type is `station` and the scrap projection is extended elevation, this option specifies the neighbouring survey station that should be used as origin (from station) when placing the station point on the extended centreline. This is useful at junctions, where the same survey station may be reached by more than one shot and the cartographer needs to choose which branch the point should follow. If `-from` is not given, Therion falls back to the nearest matching branch in the scrap.

  Therion accepts this option also for other point types, but it has no defined effect there.
- `name <reference>` ▷ if the point type is station, this option gives the reference to the real survey station.
- `scrap <reference>` ▷ if the point type is section, this is a reference to a cross-section scrap.
- `explored <length>` ▷ if the point type is continuation, you can specify length of passages explored but not surveyed yet. This value is afterwards displayed in survey/cave statistics.

[24] Example: if you specify `-context point air-draught` to a label which displays the observation date, the `symbol-hide point air-draught` command would hide both air-draught arrow and the corresponding label.

- `text` ▷ text of the label, remark or continuation. It may contain following formatting keywords:[25]

  `<br>` ▷ line break

  `<center>`/`<centre>`, `<left>`, `<right>` ▷ line alignment for multi-line labels. Ignored if there is no `<br>` tag.

  `<thsp>` ▷ thin space

  `<rm>`, `<it>`, `<bf>`, `<ss>`, `<si>` ▷ font switches

  `<rtl>` and `</rtl>` ▷ marks beginning and end of a right-to-left written text

  `<lang:XX>` ▷ creates multilingual label (see `string` type for detailed description)

  `<size:N>` ▷ specify the font size in points; `N` should be an integer between 1 and 127. *6.1.0*

  `<size:N%>` ▷ specify the font size as a percentage of the native font size of the given label; `N` should be between 1 and 999.[26] *6.1.1*

  `<size:S>` ▷ specify the font size using predefined scales; `S` can be one of `xs`, `s`, `m`, `l`, `xl`. *6.1.1*

- `value` ▷ value of height, passage-height, altitude, dimensions or date

  height: according to the sign of the value (positive, negative or unsigned), this type of symbol represents chimney height, pit depth or step height in general. The numeric value can be optionally followed by '`?`', if the value is presumed and units can be added (e.g. `-value [40? ft]`). Values '`+?`' and '`-?`' are also accepted, meaning unknown height or depth.

  passage-height: the following four forms of value are supported: `+<number>` (the height of the ceiling), `-<number>` (the depth of the floor or water depth), `<number>` (the distance between floor and ceiling) and `[+<number> -<number>]` (the distance to ceiling and distance to floor). All four forms can be optionally followed by a unit. If there is more than one piece of information - two numbers, number and unit or else - the value must be enclosed in square brackets.

  altitude: the value specified is the altitude difference from the nearest station. The value will be set to 0 if defined as '`-`', '`.`', '`nan`', '`NAN`' or '`NaN`'. If the altitude value is prefixed by '`fix`' (e.g. `-value [fix 1300]`), this value is used as an absolute altitude. The value can optionally be followed by length units.

  dimensions: `-value [<above> <below> [<units>]]` specifies passage dimensions above/below centerline plane used in 3D model.

  date: `-value <date>` sets the date for the date point.

[25] For SVG output, only `<br>`, `<thsp>`, `<it>`, `<bf>`, `<rm>` and `<lang:XX>` keywords are taken into account; all others are silently ignored.

[26] For practical reasons, the values are currently used in the increments of 10, so both 46 and 53 are interpreted as 50 % size.

'line'

Description: Line is a command for drawing a line symbol on the map. Each line symbol is oriented and its visualization may depend on its orientation (e.g. pitch edge ticks). The general rule is that the free space is on the left, rock on the right. Examples: the lower side of a pitch, higher side of a chimney and interior of a passage are on the left side of pitch, chimney or wall symbols, respectively.

Syntax:
```
line <type> [OPTIONS]
     [OPTIONS]
     ...
     [LINE DATA]
     ...
     [OPTIONS]
     ...
     [LINE DATA]
     ...
endline
```

Context: scrap

Arguments:

- `<type>` is a keyword that determines the type of line. The following types are supported:

  passages: `wall`, `contour`, `slope`[27], `floor-step`, `pit`, `pitch` (synonym of pit), `ceiling-step`, `chimney`, `overhang`, `ceiling-meander`, `floor-meander`, `low-ceiling`, `pit-chimney`;

  passage fills: `flowstone`, `moonmilk`, `rock-border`[28], `rock-edge`[29], `water-flow`, `abyss-entrance`, `dripline`, `fault`, `gradient`, `joint`, `rimstone-dam`, `rimstone-pool`;

  equipment: `fixed-ladder`, `handrail`, `rope`, `rope-ladder`, `steps`, `via-ferrata`, `walkway`;

  labels: `label`;

[27] Slope line marks upper border of the sloping area. It's necessary to specify `l-size` in at least one point. Gradient lines length and orientation is an average of specified `l-size`s and `orientation`s in the nearest points. If there is no orientation specification, gradient marks are perpendicular to the slope line.

[28] Outer outline of large boulders. If the line is closed, it is filled with the background colour.

[29] Inner edges of large boulders.

special: border, arrow, section[30], survey[31], map-connection[32], u[33].

Command-like options:

Most of these options are accepted both as a line command option and as a [LINE DATA] option despite representing a 'whole line' option. The last one appearing prevails. The exceptions are adjust, altitude, l-size, mark, orientation, size and smooth that are only accepted as a [LINE DATA] option.

The direction and gradient options are only accepted as a [LINE DATA] command option when set to point. Otherwise they are accepted as a line command option.

- subtype <restr_keyword> ▷ determines line subtype. The following subtypes are supported for given types:

  wall: invisible, bedrock (default), sand, clay, pebbles, debris, blocks, ice, underlying, overlying, unsurveyed, presumed, pit[34], flowstone, moonmilk;

  border: visible (default), invisible, temporary, presumed;

  water-flow: permanent (default), conjectural, intermittent;

  survey: cave (default), surface (default if centreline has surface flag).

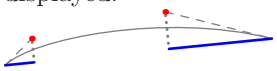
  The subtype may be specified also directly in <type> specification using ':' as a separator.[35]

  Any subtype specification can be used with user defined type (u). In this case you need also to define corresponding metapost symbol (see the chapter *New map symbols*).

  subtype can be set either as a line option or as a line-point option (inside [LINE DATA]). When used as a line option (either as TYPE:SUBTYPE or -subtype SUBTYPE), it defines the subtype starting at the beginning of the line and stays in effect until it is overridden by another subtype specified at a line point or until the end of the line, whichever comes first. As a line point option (inside [LINE DATA]) it can be used multiple times.

- [LINE DATA] specify either the coordinates of a line segment <x> <y>, or coordinates of a Bézier curve arc <c1x> <c1y> <c2x> <c2y> <x> <y>, where c indicates the control point.

[30] Line showing cross-section position. If both control points (red dots) of a Bézier curve (grey line) are given then the section line (blue) is drawn up to the perpendicular projection (dotted) of the first control point and from the projection (dotted) of the section control point. No section curve is displayed.



[31] Survey line is automatically drawn by Therion.

[32] Used to indicate connection between maps (in offset, or the same points in extended elevation).

[33] For user defined line symbols.

[34] Usually open to surface.

[35] E.g. border:invisible

- `close <on/off/auto>` ▷ determines whether a line is closed or not (default: `auto`).
- `mark <keyword>` ▷ is used to mark the point on the line (see `join` command).
- `outline <in/out/none>` ▷ determines whether the line serves as a border line for a scrap. Default value is '`out`' for walls, '`none`' for all other lines. Use `-outline in` for large pillars etc.
- `reverse <on/off>` ▷ whether points are given in reverse order (default: `off`).
- `smooth <on/off/auto>` ▷ whether the line is smooth at the given point. Deafult is `auto`. This setting only has any effect if both segments that passes through the point affected by the smooth setting are Bezier Curves (the one arriving at it and the one leaving it). On means that the line connection at this point will be smoothed, both by therion and by xTherion. Off means that smoothness at the point is not guaranteded. Auto means that, if both segments that passes through the point are Bezier Curves, Therion will automatically detected if they are smooth and, if they are detected as smooth, Therion will keep them smooth in it's transformations. xTherion always creates Bezier Curves as smooth, even if the previous segment was a straight one.
- `adjust <horizontal/vertical>` ▷ shifts the line point to be aligned horizontally/vertically with the previous point. It can't be set on the first point. The result is a horizontal/vertical line segment. When unset (the default situation) the line point isn't automatically aligned with the previous line point. This option is not allowed in the `plan` projection.
- `place <bottom/default/top>` ▷ changes displaying order in the map. Default is `top` for `ceiling-step` and `chimney`, `default` for all other types.
- `clip <on/off>` ▷ specify whether a symbol is clipped by the scrap border (default: `on`).
- `visibility <on/off>` ▷ displays/hides the symbol (default: `on`).
- `context <point/line/area> <symbol-type>` ▷ (to be used with `symbol-hide` and `symbol-show` layout options) symbol will be hidden/shown according to rules for specified `<symbol-type>`.

  Type-specific options:
- `altitude <value>` ▷ can be specified only with the wall type. This option creates an altitude label on the wall. All altitudes are exported as a difference against grid Z origin (which is 0 by default). If the value is specified, it gives the altitude difference of the point on the wall relative to the nearest station. The value will be set to 0 if defined as "-", ".", "nan", "NAN" or "NaN". The value can be prefixed by a keyword "`fix`", then no nearest station is taken into consideration; the absolute given value is used instead. Units can follow the value. Examples: `+4`, `[+4 m]`, `[fix 1510 m]`.

- anchors <on/off> ▷ this option can be specified only with the 'rope' line type (default: on).
- border <on/off> ▷ this option can be specified only with the 'slope' symbol type. It switches on/off the border line of the slope (default: on).
- direction <begin/end/both/none/point> ▷ can be used only with the section type. It indicates where to put a direction arrow on the section line (default: none). The point option must be used inside [LINE DATA]. The others can (and should) be used as a line option.
- gradient <none/center/point> ▷ can be used only with the contour type and indicates where to put a gradient mark on the contour line. If there is no gradient specification, behaviour is symbol-set dependent (e.g. no tick in UIS, tick in the middle in SKBB). The point option must be used inside [LINE DATA]. The others can (and should) be used as a line option (default: none).
- head <begin/end/both/none> ▷ can be used only with the arrow type and indicates where to put an arrow head (default: end).
- l-size <number> ▷ Size of the line (to the left). Only valid on and required for slope type.
- orientation/orient <number> ▷ orientation of the symbols on the line. Can be used only with slope type. If not specified, it's perpendicular to the line on its left side. $0 \leq$ number < 360.
- rebelays <on/off> ▷ this option can be specified only with the 'rope' line type (default: on).
- scale ▷ scale affects only the text on label lines, can be: tiny (xs), small (s), normal (m), large (l), huge (xl) or a numeric value (default: normal). Named sizes scale by $\sqrt{2}$, so that $xs \equiv 0.5$, $s \equiv 0.707$, $m \equiv 1.0$, $l \equiv 1.414$ and $xl \equiv 2.0$. Absolute font sizes (in points) can be assigned to named sizes using fonts-setup in the layout configuration section.
- size <number> ▷ synonym of l-size
- text <string> ▷ valid only for label lines.
- height <value> ▷ height of pit or wall:pit; available in METAPOST as a numeric variable ATTR__height.

Options:

- id <ext_keyword> ▷ ID of the symbol.

'area'

Description: Area is specified by surrounding border lines. They may be of any type, but must be listed in order and each pair of consecutive lines must intersect. In order to be sure that lines intersect even after scrap transformation you may e.g. continue a lake border 1 cm behind a passage wall—these overlaps will be automatically clipped by scrap border. You may use invisible border to achieve this inside of the passage. When defined by more than one line, the actual area will be the intersection of the areas defined by each separate line.

Syntax:
```
area <type>
  place <bottom/default/top>
  clip <on/off>
  visibility <on/off>
... border line references ...
endarea
```

Context: scrap

Arguments:

- `<type>` is one of following: `water`, `sump`, `sand`, `debris`, `blocks`, `flowstone`, `moonmilk`, `snow`, `ice`, `clay`, `pebbles`, `bedrock`[36], `u`[37], `mudcrack`, `pillar`, `pillar-with-curtains`, `pillars`, `pillars-with-curtains`, `stalactite`, `stalactite-stalagmite`, `stalagmite`.

Command-like options:

All options can appear as command-like options or as `area` options, i.e., on the same line as the `area` command. The last one appearing prevails.

- the data lines consist of border line references (IDs)
- `place <bottom/default/top>` ▷ changes displaying order in the map (default: `top`).
- `clip <on/off>` ▷ specify whether a symbol is clipped by the scrap border (default: `on`).
- `visibility <on/off>` ▷ displays/hides the symbol (default: `on`).
- `context <point/line/area> <symbol-type>` ▷ (to be used with `symbol-hide` and `symbol-show` layout options) symbol will be hidden/shown according to rules for specified `<symbol-type>`.

Options:

- `id <ext_keyword>` ▷ ID of the symbol.

[36] An empty area which can be used to clean the background.

[37] For user defined area symbols, may be followed by arbitrary subtype.

'join'

Description: Join works in two modes: it joins either two scraps or two or more points or lines in a map together.

When joining more than two points or lines, use one join command for all of them, not a sequence of join commands for pairs.[38]

When joining scraps, only passage walls are joined. It's a good idea to place a scrap join in the passage which is as simple as possible, otherwise you have to specify join for each pair of objects which should be joined.[39]

When joining more than two scraps at the same scrap border, a manual join must be performed where the connection points must be entered in one join statement.[40]

Syntax: `join <point1> <point2> ... <pointN> [OPTIONS]`

Context: none, scrap, survey

Arguments:

- `<pointX>` can be an ID of a point or line symbol, optionally followed by a line point mark `<id>:<mark>` (e.g. `podangl_l31@podangl:mark1`). `<mark>` can be also '`end`' (end of the line) or line point index (where 0 is the first point).

  A special case is when `<point1>` and `<point2>` are scrap IDs—than the closest scrap ends are joined together.

Options:

- `smooth <on/off>` indicates whether two lines are to be connected smoothly.
- `count <N>` (when used with scraps) ▷ Therion will try to join scraps which connect in `N` locations/passages.

'equate'

Description: Sets the survey stations equivalence.

Syntax: `equate <station list>`

Context: none, survey

[38] E.g. use `join a b c`, not `join a b` followed by `join b c`.

[39] If you want some object which is clipped by a scrap boundary to continue to a neighbouring scrap, use `-clip off` option for that object.

[40] Like `join origScrapLineWest:end upperScrapLineWest:0 lowerScrapLineWest:0` and another similar join command for the three east wall lines.

'map'

Description: A map is a collection of either scraps or other maps of the same projection type. It's possible to include survey in the map—this will display centreline in the map. Map object simplifies the data management when selecting data for output. See the chapter *How the map is put together* for more thorough explanation.

(Note: `break` only changes level of maps of scraps and has no function when used with maps of maps, as they will cause a `break` implicitly)

Syntax:
```
map <id> [OPTIONS]
    ... scrap, survey or other map references ...
    break
    ... next level scrap, survey or other map references ...
    preview <above/below> <other map id>
endmap
```

Context: none, survey

Arguments:

- `<id>` ▷ scrap identifier

Command-like options:

- the data lines consist of scrap or map references. Note that you can not mix them together.
- if you refer to map, you can specify offset at which this sub-map will be displayed together with preview type of its original position. Syntax is following:
  `<map reference> [<offset X> <offset Y> <units>] <above/below/none>`
- scraps following the `break` will be placed on another level (only applies to maps consisting of scraps)
- `preview <above/below> <other map id>` will put the outline of the other map in the specified preview position relative to the current map.

  Preview is displayed only if the map is in the `map-level` level as specified by the `select` command.

  Use the revise command if you want to add maps from higher levels to the preview.
- `colo[u]r <color>` ▷ set the map colour; this option overrides the automatic choice when the layout specifies `colour map-fg [map]`.

Options:

- `projection/proj <plan/elevation/extended/none>` ▷ required if the map contains survey.

- `title <string>` ▷ description of the object
- `survey <id>` ▷ associate a survey with map (e.g. all surveying statistics from this survey will be used when this map is selected for output).

'surface'

Description: Surface (terrain) specification. It is possible to display it in two ways: as a scanned topographical map (both in 2D map and 3D model[41]) or surface grid – digital elevation model (in 3D model only).

Syntax:
```
surface [<name>]
   cs <coordinate system>
   bitmap <filename> <calibration>
   grid-units <units>
   grid <origin x> <origin y> <x spacing> <y spacing> <x count> <y count>
   grid-flip (none)/vertical/horizontal
   [grid data]
endsurface
```

Context: none, survey

Command-like options:

- `cs <coordinate system>` ▷ coordinate system for bitmap calibration and grid origin specification
- `bitmap <filename> <calibration>` ▷ scanned topographical map.

  `calibration` may have two forms:

  1. `[X1 Y1 x1 y1 X2 Y2 x2 y2 [units]]`, where upper case X/Y variables are picture coordinates (pixels; lower-left corner is `0 0`), lower-case x/y variables are real coordinates. Optional units apply to real coordinates (metres by default).

  2. `[X1 Y1 station1 X2 Y2 station2]`, where upper case X/Y variables are picture coordinates and `station1` and `station2` are survey stations names.
- `grid-units <units>` ▷ units in which grid is specified. Metres by default.
- `grid <origin x> <origin y> <x spacing> <y spacing> <x count> <y count>`

  `<origin x> <origin y>` ▷ specify coordinates of the lower-left (S-W) corner of the grid

  `<x spacing> <y spacing>` ▷ distance between grid nodes in W-E and S-N directions

[41] You need to enter elevation data in order to display the topographical map in 3D model. Currently only JPEG maps are supported in 3D.

<x count> <y count> ▷ number of nodes in the row and number of rows which form the grid (see below).

- [grid data] ▷ a stream of numbers giving the altitude a.s.l. in grid nodes. It starts in the grid-origin and fills the grid in rows (in the row from W to E; rows from S to N).
- grid-flip (none)/vertical/horizontal ▷ useful if your grid (exported from other program) needs to be flipped

'scan'

Description: The scan command imports a 3D triangulated mesh into the cave model. It is intended for meshes measured outside Therion, for example by laser scanning or photogrammetry. The mesh is tied to the survey centreline by calibration lines: each calibration line gives the coordinates, in the mesh coordinate system, of one survey station already known from the centreline.

The command contributes geometry to 3D model exports. It does not create 2D map symbols and does not replace scraps used for map drawing.

Syntax:
```
scan
   file <filename>
   units <units>
   axes <x-spec> <y-spec> <z-spec>
   <station> <x> <y> <z>
   ...
endscan
```

Context: none, survey

Arguments:

The scan command itself has no obligatory arguments. Data lines inside the command have the following form:

- <station> ▷ name of an existing survey station. If the scan command is inside a survey, normal survey name-space rules apply. Use station@survey if a station from another visible survey must be referenced.
- <x> <y> <z> ▷ coordinates of <station> in the mesh coordinate system. These numbers are interpreted using the current units command-like option of the scan object.

Command-like options:

- file <file-name> ▷ mesh file to import. This command-like option is required. The current implementation reads STL meshes. Relative file names are resolved relative to the file that contains the scan command.
- units <length units> ▷ length unit used by coordinates in the mesh file and in the station calibration lines (default: m).

- `axes <x-axis> <y-axis> <z-axis>` ▷ orientation of the mesh coordinates before the station calibration is applied. Each argument is one of `x`, `y`, `z`, `-x`, `-y` or `-z`, and the three arguments must contain all three coordinate names. The default is `x y z`. The first argument defines Therion's local `x` coordinate, the second its local `y` coordinate, and the third its local `z` coordinate. For example, `axes -z -x y` transforms a mesh point `(x,y,z)` to `(-z,-x,y)`.

Special notes:

Calibration lines should refer to stations whose 3D positions are already known from the centreline. A single station can only shift the mesh. With two stations Therion can determine horizontal rotation and scale; additional well-distributed stations give a better local morphing of the mesh. Avoid using stations that are very close to each other or nearly vertical above one another as the only calibration points.

The `units` and `axes` command-like options are applied both to vertices read from the mesh file and to the station coordinates given in the `scan` block. If an imported mesh is mirrored or rotated, change `axes` rather than changing the station calibration coordinates.

'import'

Description: Reads survey data in different formats (currently processed centreline in *.3d, *.plt, *.xyz formats). Survey stations may be referenced in scraps etc. When importing a Survex 3D file, stations are inserted in the survey hierarchy if there exists an identical hierarchy to that in 3D file.

Syntax: `import <file-name> [OPTIONS]`

Context: survey / all[42]

Options:

- `filter <prefix>` ▷ if specified, only stations with given prefix and shots between them will be imported. Prefix will be removed from station names.
- `surveys (create)/use/ignore` ▷ specifies how to import survey structure (works only with .3d files).

  `create` ▷ split stations into subsurveys, if subsurveys do not exist, create them

  `use` ▷ split stations into existing subsurveys

  `ignore` ▷ do not split stations into sub-surveys
- `cs <coordinate system>` ▷ coordinate system for stations with fixed coordinates
- `calibrate [<x> <y> <z> <X> <Y> <Z>]` ▷ coordinates in the imported file are shifted from lower-case coordinates to upper-case coordinates.

[42] only with .3d files, where survey structure is specified

'grade'

Description: This command is used to store predefined precisions of centreline data. Built in grades are: BCRA[43] and UISv1[44].

See `sd` option description for `centreline` command to define your own grades.

Syntax: :
```
grade <id>
...
[<quantity list> <value> <units>]
...
endgrade
```

Context: all

'revise'

Description: This command is used to set or change properties of an already existing object.

Syntax: The syntax of this command for object created with "single line" command is

```
revise id [-option1 value1 -option2 value2 ...]
```

For objects created with "multi line" commands is syntax following

```
revise id [-option1 value1 -option2 value2 ...]
  ...
  optionX valueX
  data
  ...
endrevise
```

Context: all

Arguments:

The id stands for object identifier (the id of an object you want to revise must always be specified).

[43] see `http://bcra.org.uk/surveying/`; syntax is: `BCRA`n, where n may be `3` or `5`

[44] see `http://www.uisic.uis-speleo.org/UISmappingGrades.pdf`; syntax is: `UISv1_`n, where n is `-1` to `6` or `X`; whereas `-1` to `2` are only declaratory and `X` requires `sd` data in `centerline`)

'require'

Description: Checks that the version of Therion in use is at least `<version>` and stops with an error if not.

Syntax: `require <version>`

Context: all

Arguments:

- `<version>` ▷ The minimum supported Therion version in the form `major[.minor[.patch]]`, e.g. `6.3`.

Custom attributes

Objects *survey*, *centreline*, *scrap*, *point*, *line*, *area*, *map* and *surface* can contain user-defined attributes in a form `-attr <name> <value>`. `<name>` may contain alphanumeric characters, `<value>` is a string.

The custom attributes are used in map exports depending on the output format:

- in *shapefile* export they are written directly to the associated dbf file,
- in maps generated using METAPOST (PDF, SVG) the attributes are written in the METAPOST source file as strings (named like `ATTR_<name>`) and can be evaluated and used by the user to define symbols in macros.

  You can test presence of such a variable using `if known ATTR_<name>: ... fi`.

Graphical interfaces

To make creating and modifying data files easier, you can use a GUI (Graphical User Interface) for Therion. You can, in fact, edit input files in your favourite text editor and run Therion from the command line, so any GUI is completely optional.

XTherion is a GUI that is included in Therion distributions. There are also alternative editors, like wtherion[45], and others in development.

XTherion

XTherion works in three main modes: text editor, map editor and compiler.[46]

[45] wtherion is a graphical editor for TH2 files available on the web. See https://github.com/daem-on/wtherion

[46] Here we're concerned with creating data, so only the first two modes are described in this section. For compiler features see the chapter *Processing data*.

This manual does not describe such familiar things as 'if you want to save a file, go to menu File and select Save, or press Ctrl-s'. Browse the top menu for a minute to get the feeling of XTherion.

For each mode of operation, there is an additional right or left menu. The submenus may be packed; you may unpack them by clicking on the menu button. For most of the menus and buttons, there is a short (translated) description in the status line, so it's not hard to guess the meaning of each one. The order of submenus on the side may be customized by the user. Right-click on the menu button and select in the menu which of the other menus it should be swapped with.

XTherion—text editor

XTherion's text editor offers some interesting features which may help with creating text input files: support for Unicode encoding and ability to open multiple files.[47]

To make entering data easy, it supports table formatting of centreline data. There is a menu *Data table* for typing the data. It may be customized to the user's data order by pressing a *Scan data format* button when the cursor is below the data order specification ('data' option in the 'centreline' command).

XTherion—map editor

Map editor allows you to draw and edit maps fully interactively. But don't expect too much. XTherion is not a truly WYSIWYG editor. It displays only the position, not the actual shape, of drawn point or line symbols. Visually there is no difference between a helictite and a text label—both are rendered as simple dots. The type and other attributes of any object are specified only in the *Point control* and *Line control* menus.

Exercise: Find two substantial reasons, why the map drawn in XTherion can't be identical with Therion output. (If you answer this, you'll know, why XTherion will never be a true WYSIWYG editor. Authors' laziness is not the correct answer.)

Let's begin by describing typical use of the map editor. First, you have to decide which part of the cave (which scrap) you'll draw.[48]



After creating a new file in the map editor, you may load one or more **images**—scanned survey sketches from the cave[49]—as a background for the drawing. Click on the *Insert* button in *Background images* menu. Unfortunately, as a limitation of Tcl/Tk language,

[47] File encoding is specified on the first line of the file. This line is hidden by XTherion and may be accessed only indirectly using the right-hand menu.

[48] It's possible to draw more than one scrap in each file, in which case all inactive scraps are rendered yellow.

[49] XTherion can't scale nor rotate individual images, so use the same orientation, scale and DPI for all images used in the same scrap.

only GIF, PNM and PPM (plus PNG and JPEG if you installed tkImg extension) images are supported. Additionally XTherion supports XVI (XTherion vector image) format, which displays centreline and LRUD information on the background, and PocketTopo data exported in Therion format (see below). All opened images are placed in the upper-left corner of the working area. Move them by double clicking and dragging with the right mouse button or through a menu. For better performance on slower computers, it's possible to temporarily unload a currently unused image from memory by unchecking its *Visibility* check-box. It's possible to open an existing file without loading background images using *Open XP* menu.[50]

The size and zoom setting of the **drawing area** is adjusted in the corresponding menu. *Auto adjust* calculates optimal size of the working area according to the sizes and positions of loaded background images.

After these preparation steps, you're ready for drawing, or, more precisely, for **creating a map data file**. It's important to remember, that you're actually creating a text file which should conform to the syntax described in the chapter *Data format*. Actually, only a subset of the Therion commands are used in the Map editor: multi-line `scrap ... endscrap` commands which may contain `point`, `line` and `area` commands. (Cf. chapter *Data format*). This corresponds with a section of hand-drawn maps, which are built up from points, lines and filled areas.

So, the first step is defining the **scrap** by a `scrap ... endscrap` multi-line command. In the *File commands* menu click on the *Action* submenu and select *Insert scrap*. This changes the *Action button* to *Insert scrap* if it had any other value. After pressing this button a new scrap will be inserted in the beginning of the file. You should see lines

```
scrap - scrap1
endscrap
end of file
```

in the preview window above the *Insert scrap* button. This window is a simplified outline of the text file, which will be saved by XTherion. Only the command (`scrap`, `point`, `line`, `text`—why text see below) and its type (for `point` and `line`) or ID (for `scrap`) are shown.

The full contents of any command are displayed in the *Command preview* menu.

For modifying previously-created commands, there are additional menus—e.g. *Scrap control* for the `scrap` command. Here you can change the ID (very important!) and other options. For details see chapter *Data format*.

Now it's possible to insert some **point symbols**. As with scrap insertion, go to the *File commands* menu, click on the *Action* submenu and select *Insert point*; than press newly renamed *Insert point* button. A shortcut for all this is Ctrl-p. Than click on the

[50] *Note:* Therion doesn't use background images in any way unless you assign them to some scrap using `-sketch` option.

desired spot in the working area and you'll see a blue dot representing a point symbol. Its attributes can be adjusted in the *Point control* menu. You'll stay in 'insert' mode—each click on the working area adds a new point symbol. Take care not to click twice on the same place—you would insert two point symbols in the same place! To escape from 'insert' to 'select' mode, press *Esc* key on the keyboard or *Select* button in the *File commands* menu.

What will be the order of commands in the output file? Exactly the same as in the outline in the *File commands* menu. Newly created point, line and text objects are added before the currently marked line in the outline. It is possible to change the order by selecting a line and pressing *Move down*, *Move up* or *Move to* buttons in the *File commands* menu. This way you can also move objects between scraps.

Drawing lines is similar to drawing in other vector editing programs, which work with Bézier curves. (Guess how to enter the line insertion mode, other than using the shortcut Ctrl-l.) Click where the first point should be, then drag the mouse with pressed left button and release it where the first control point should be. Then click somewhere else (this point will be the second point of the curve) and drag the mouse (adjusting the second control point of the previous arc and the first control point of the next one simultaneously.) If this explanation sounds too obscure, you can get some practise working in some of the standard vector editors with comprehensive documentation. The line will be finished after escaping from the insertion mode. Beginning and orientation of the line is marked by a small orange tick to the left at the first point.

For line symbols, there are two control menus: *Line control* and *Line point control*. First one sets attributes for the whole curve, like type or name. The check-box *reverse* is important: Therion requires oriented curves and it is not unusual that you begin to draw from the wrong end. The *Line point control* menu enables you to adjust the attributes of any selected point on the line, such as the curve being smooth at this point (which is on by default), or the presence of neighbouring control points ('`<<`' and '`>>`' check-boxes).

Areas are specified by their surrounding lines. Click on *Insert area* and then click on the lines surrounding the desired area. They are automatically inserted in the *Area control* and named (if not already named). An alternate way is to insert them as a `text`[51] command, the contents of which (entered in the *Text editor* menu of the Map editor) is usual `area ... endarea` multi-line command (see the chapter *Data format*.)

If you draw some scraps with `none` projection, it's necessary to **calibrate** the drawing area. The scale can be defined only one way in XTherion—using coordinates of two points (specified both in the picture coordinate system and in the 'real' coordinate system).

After selecting a scrap (click on its header in the *File commands* menu) two small red squares connected by a red arrow will appear (by default, they'll be in the lower corners

[51] CAUTION! The command `text` is not a Therion command! It's only a nickname for a block of arbitrary text in XTherion. In the file saved by XTherion, there'll only be whatever you type into the *Text editor* or see in the *Command preview*. It may be an area definition or whatever you want, such as a comment beginning with a '`#`' character.

of drawing area). You have to drag them to points with known coordinates—usually intersections of mm grid lines on the scanned drawing. If you can not see these points, you can either

- press *Scale* button in the *Scraps* menu and click on two different places on the image where the endpoints of calibration arrow should be, or
- move the mouse pointer to the desired position, read pointer coordinates from the status bar and enter these coordinates into *picture scale points* boxes in the *Scraps* control. After filling X1,Y1 and X2,Y2 coordinate pairs the calibration arrow will be moved correspondingly.

Then you have to enter real coordinates of these points (X1, Y1, X2, Y2).

In the **selection mode** you can select existing line or point objects and set their attributes in the corresponding menus, move them, or delete them (Ctrl-d or *Action button* in *File commands menu* after setting *Action* to *Delete*).

There is a *Search and select* menu which makes it easy to switch between objects and visualize things you can't see at the first look at the picture. For example, if you enter expression 'station' and press *Show All*, all stations on the picture will become red.

XTherion doesn't do any syntax checking; it only writes drawn objects with their attributes to a text file. Any errors are detected only when you process these files with Therion.

TIP: Entering symbols of the same type at once saves you a lot of time because you need not to change the symbol type and fill options for each new symbol. *Options* box preserves the old value and it's enough to change a few characters.[52] It is a good idea to start with drawing all survey stations (don't forget to give them names according to the real names in the centreline command), than all passage walls followed by all other point symbols, lines and areas. Finally, draw cross-sections.

Additional tools

Help/Calibrate bitmap produces OziExplorer-compatible MAP file based on georeferencing data included in PDF maps produced by Therion versions 5.3–6.2.[53] [54] *6.3*

If the map in PDF format has been converted to raster using an external program, the converter uses raster image *and* pdf map with the same base name located in the same directory to calculate the calibration data.

[52] In the case of survey stations, XTherion automatically increases the station number for the next symbol inserted.

[53] Calibration information for nine distinct points is present if the centreline contains station(s) fixed using geodetic coordinate system(s).

[54] Therion 6.3+ uses standard geospatial PDF files where such a calibration is not needed.

If the PDF file is used directly, you have to set the DPI and output format before automatic conversion[55] to a raster format.

PocketTopo data exported in Therion format[56] from PocketTopo application can be imported in the text editor as well as in the map editor (*File* → *Import* → *PocketTopo Therion export* and *Background Images* → *Insert* → *PocketTopo Therion export*). The same file is used for both imports. Importing the sketch does not create scrap data directly. The drawing is just displayed on the background like scanned bitmaps and should be digitized manually.

Keyboard and mouse shortcuts in the Map editor

General

- Ctrl+Z ▷ undo
- Ctrl+Y ▷ redo
- F9 ▷ compile current project
- to select an object in the listbox using the keyboard: switch using 'Tab' into the desired listbox; move with the underlined cursor to the desired object; press 'Space'
- PageUp/PageDown ▷ scroll up/down in the side panel
- Shift+PageUp/PageDown ▷ scroll up/down in the file commands window

Drawing area and background images

- RightClick ▷ scroll the drawing area
- Double RightClick on the image ▷ move the image

Inserting a scrap

- Ctrl+R ▷ insert scrap

Inserting a line

- Crtl+L ▷ insert a new line and enter an 'insert line point' mode
- LeftClick ▷ insert a line point (without control points)
- Ctrl+LeftClick ▷ insert a line point very close to the existing point (normally it's inserted right above closest existing point)
- LeftClick + drag ▷ insert line point (with control points)
- hold Ctrl while dragging ▷ fix the distance of the previous control point

[55] `ghostscript` and `convert` should be installed on your system. Note, that Windows installation does not include `ghostscript`.

[56] This is a special text format which needs to be imported using XTherion and can not be processed by Therion directly.

- LeftClick + drag on the control point ▷ move its position
- RightClick on one of the previous points ▷ select the previous point while in insert mode (useful if you want to change also the direction of the previous control point)
- Esc or LeftClick on the last point ▷ end the line insertion
- LeftClick on the first line point ▷ close the line and end the line insertion

Editing a line

- LeftClick + drag ▷ move the line point
- Ctrl+LeftClick + drag ▷ move the line point close to the existing point (normally it is moved right above the closest existing point)
- LeftClick on control point + drag ▷ move the control point

Adding a line point

- select the point before which you want to insert points; insert required points; press Esc or left-click on the point you selected at the beginning

Deleting a line point

- select the point you want to delete; press *Edit line* → *Delete point* in the *Line control* panel

Splitting a line

- select the point at which you want to split the line; press *Edit line* → *Split line* in the *Line control* panel

Inserting a point

- Ctrl+P ▷ switch to 'insert point' mode
- LeftClick ▷ insert point at a given position
- Ctrl+LeftClick ▷ insert point very close to the existing point (normally it will be inserted right above the closest point)
- Esc ▷ escape from the 'inset point' mode

Editing a point

- LeftClick + drag ▷ move the point
- Ctrl+LeftClick + drag ▷ move the point close to the existing point (normally it is moved right above the closest existing point)
- LeftClick + drag on the point arrow ▷ change point orientation or size (according to the given switches in the *Point control* panel)

Inserting an area

- press Ctrl+A or *File commands* → *Insert* → *area* to switch to the 'insert area border' mode
- RightClick on the lines, that surround the desired area
- Esc to finish the area border lines insertion

Editing an area

- select the area you want to edit
- press 'Insert' in the *Area control* to insert other border lines at the current cursor position
- press 'Insert ID' to insert a border with a given ID at the current cursor position
- press 'Delete' to remove the selected area border line

Selecting an existing object

- LeftClick ▷ select the object at the top
- RightClick ▷ select the object right below the top object (useful when several points lie above each other)

Thinking in Therion

Although everything (well, almost everything) about Therion input files has been explained, this chapter offers some additional hints and tips.

How to enter a centreline

The basic building block is the `centreline` command. If the cave is larger than a few meters it's a good idea to split the data into more files and separate the centreline data from the map data.

We usually use one `*.th` file containing a centreline per survey trip. It's handy to start with an empty template file as shown below, where dots will be replaced with appropriate texts.

```
encoding ISO8859-1
survey ... -title "..."
  centreline
    team "..."
    team "..."
    date ...
    units clino compass grad
    data normal from to compass clino length
```

```
      ... ... ... ... ...
  endcentreline
endsurvey
```

To create a unique namespace the `centreline` command is enclosed in `survey` ... `endsurvey` command. It's useful when the survey has the same name as the file which contains it.[57] The points will then be referenced using the `@` character—see the `survey` command description.

For really large caves it's possible to build a hierarchical structure of directories. In such a case we create one special file called `INDEX.th` which includes all other `*.th` files from a given directory and contains `equate` commands to define connections between surveys.

How to draw maps

The most important thing is to devise a division of the cave into scraps. Scrap is the basic building block of the map. It's almost always a *bad* idea to try to fit each scrap to corresponding `*.th` file with a centreline from one survey trip. The reason is that the connections between scraps should be as simple as possible. Scraps in general are independent on the centreline hierarchy so try to forget the survey hierarchy when drawing maps and choose the best scrap joins.

We usually insert maps in the last-but-one level in survey hierarchy.[58] Each scrap may then contain arbitrary parts of any survey in the last level of the hierarchy. For example, there is a survey `main` which contains surveys `a`, `b`, `c` and `d`. Surveys `a` – `d` contain centreline data from four survey trips and each of them is in a separate file. There is a map `main_map` which contains scraps `s1` and `s2`. If the `main_map` is located in the `main` survey, scrap `s1` may cover part of the centreline from survey `a`, complete survey `b` and part of `c`; `s2` will cover part of the `a` and `c` surveys and a complete `d` survey. The survey stations names will be referenced using the `@` symbol (e.g. `1@a`) in the scraps.[59]

Scraps are usually stored in `*.th2` files. Each file may contain more scraps. To keep the data well organized, there are some naming conventions: in the file `foo.th2` all scraps are named `foo_si`, where `i` is `1`, `2` an so on. Cross-sections are named `foo_ci`, lines `foo_li` etc. This helps a lot with large cave systems: if some scrap is referenced, you immediately know in which file it has been defined.

Similar to `*.th` files, there may be one file `INDEX.th2` per directory which includes all the `*.th2` files, and defines scrap joins and maps.

[57] E.g. `survey entrance` in the file `entrance.th`.

[58] Remember that surveys create namespaces, so you may reference only the objects in the given survey and all subsurveys.

[59] If you include maps in the top-level survey, you may reference any survey station in any scrap, which is very flexible. On the other hand you then have to use longer names in station references, like `3@dno.katakomby.jmn.dumbier`.

When drawing scraps you should check if the outline is properly defined: all lines creating the outer border should have `-outline out` option; all lines surrounding inner pillars `-outline in` option. Scrap outlines can't intersect themselves—otherwise the inner side of the scrap can't be determined. There are two simple tests that the scrap outline is correct:

- there is no METAPOST warning "`scrap outline intersects itself`"
- when you set a passage fill to any color (`color map-fg <number>` option in `layout`), you may see what Therion considers to be inside the scrap.

How to create models

The model is created from scrap outlines. The height and depth of the passage are computed from `passage-height` and `dimensions` point map symbols.

Therion in depth

How the map is put together

This chapter explains how `-clip`, `-place`, `-visibility` and `-context` options of `point`, `line` and `area` commands work exactly. It also gives an explanation of `color`, `transparency`, `symbol-hide` and `symbol-show` options of the `layout` command.

While exporting the map, Therion has to determine three attributes for each point, line or area symbol: visibility, clipping and ordering.

(1) Symbol is visible if all of the following is true:

- it has the `-visibility` option set `on` (all symbols by default),
- it hasn't been hidden by the `-symbol-hide` option in the layout,
- if its `-context` option is set, the corresponding symbol hasn't been hidden by the `-symbol-hide` option in the layout.

Only the visible symbols are exported.

(2) Some symbols are clipped by the scrap outline. These are by default all of the following:

- *point symbols:* symbolic passage fills (bedrock. . . guano),
- *line symbols:* all line symbols which don't have the `-outline` option set with the exception of `section`, `arrow`, `label`, `gradient` and `water-flow`
- *area symbols:* all.

The default setting may be changed using the `-clip option`, if this is allowed for a particular symbol. All other symbols are not clipped by the scrap boundary.

(3) Ordering: Each symbol belongs to one of the following groups which are drawn consecutively:

- bottom ▷ all symbols with the `-place bottom` option set
- default-bottom ▷ all `area` symbols by default
- default ▷ symbols which don't belong to any other group
- default-top ▷ `ceiling-step` and `chimney` by default
- top ▷ all symbols with `-place top` option set

Ordering of symbols inside each group follows the order of commands in the input file[60]: symbols which come first are drawn last (i.e. they are displayed at the top of each group).

Now we are ready to describe how the map (or atlas chapter) is constructed:

- map area is filled with `color map-bg`
- surface bitmaps are displayed if `surface` is set `bottom`
- FOR each scrap: outline is filled white
- grid is displayed if `grid` is set `bottom`
- preview below[61] is filled with `color preview-below`
- FOR each level[62]:

  BEGIN of transparency

    FOR each scrap: outline is filled with `color map-fg`

    FOR each scrap: `area` symbols are filled and clipped to scrap boundary

  END of transparency

  BEGIN of clipping by text labels (for all labels in this and upper levels)

    FOR each scrap:

      draw all symbols to be clipped (with the exception of `line survey`)

        ordered from bottom to top

      draw `line survey` symbols

      clip to scrap boundary

    FOR each scrap:

      draw all symbols not to be clipped (with the exception of `point station`

[60] Or *File commands* menu in XTherion.

[61] As specified using the `preview` option in the `map` command.

[62] Level is a collection of scraps not separated by a `break` in the `map` command.

and all labels) ordered from bottom to top

draw `point station` symbols

END of clipping by text labels

FOR each scrap: draw all (point and line) labels (including `wall-altitude`)

- preview above is drawn with `color preview-above`
- surface bitmaps are displayed if `surface` is set `top`
- grid is displayed if `grid` is set `top`

We both step and do not step in the same rivers.
Ποταμοῖς τοῖς αὐτοῖς ἐμβαίνομέν τε καὶ οὐκ ἐμβαίνομεν.

—Heraclitus of Ephesus, 6th/5th century BC

Processing data

Besides data files, which contain survey data, Therion uses a configuration file, which contains instructions on how the data should be presented.

Configuration file

The configuration filename can be given as an argument to Therion. By default Therion searches for file named `thconfig` in the current working directory. It is read like any other Therion file (i.e. one command per line; empty lines or lines starting with '`#`' are ignored; lines ended with a backslash continue on the next line.) A list of currently supported commands follow.

'system'

Allows to execute system commands during Therion compilation.[63] Normally Therion waits until the subprocess is finished. If you want to continue compilation without break, use `<command> &` syntax on Linux and `start <command>` syntax on Windows.

'encoding'

Works like the `encoding` command in data files—specifies character sets.

'language'

Syntax:

- `language <xx_[YY]>`

Sets the output language for translatable texts.

'cs'

Syntax:

- `cs <coordinate system>`

[63] E.g. to open or refresh external PDF viewer.

Outside of `layout` command specifies the coordinate system for output. It is not possible to specify more coordinate systems for different outputs (the last occurrence of `cs` is used for all output files).

If no `cs` is defined in the configuration file, the first `cs` Therion encounters in the data files is used as an output `cs`.

Inside the `layout` specifies coordinate system for subsequent location data (`origin`, `grid-origin`).

'scrap-sort'

Description: Sort scraps by altitude midpoint (default). If turned off, scraps keep their input order. Only applies if no maps are defined.

Syntax: `scrap-sort <on/off>`

'sketch-warp'

Syntax:

- `sketch-warp <algorithm>`

Specifies which scrap warping (morphing) algorithm to use. Possible algorithms are `line`—the default; `plaquette`—invented by Marco Corvi.

'input'

Works like `input` command in data files—includes other files.

'source'

Description: Specifies which source (data) files Therion should read. You can specify several files here; one per line. You can also specify them using the `-s` command line option (see below).

It is also possible to type (some small snippets of) code directly in configuration file using the multi-line syntax.

Syntax:
`source <file-name>`
or
`source`
... therion commands...
`endsource`

Arguments:

- `<file-name>`

'select'

Description: selects objects (surveys and maps) for export. By default, all survey objects are selected. If there is no map selected, all scraps belonging to selected surveys are selected by default for map export.

If there are no scraps or maps in the data, centerline from all surveys is exported in the map.

When exporting maps in different projections, you need to select them for each projection separately.

`select` does not only affect subsequent `<export>` commands but instead also `<export>` commands preceding the `select` command in the configuration file.

Syntax: `select <object> [OPTIONS]`

Arguments:

- `<object>` ▷ any survey or map, identified by its ID.

Options:

- `recursive <on/off>` ▷ valid only when a survey is selected. If set on (by default) all subsurveys of the given survey are recursively selected/unselected.
- `map-level <number>` ▷ valid only when a map is selected. Determines the level at which map expansion for atlas export is stopped. By default 0 is used; if 'basic' is specified, expansion is done up to the basic maps. *Note:* Map previews are displayed only as specified in maps in the current `map-level`.
- `chapter-level <number>` ▷ valid only when a map is selected. Determines the level at which chapter expansion for atlas export is stopped. By default 0 is used, if '`-`' or '`.`' is specified, no chapter is exported for this map. If `title-pages` option in `layout` is on, each chapter starts with a title page.

'unselect'

Description: Unselects objects from export.

Syntax: `unselect <object> [OPTIONS]`

Arguments:

The same as the `select` command.

Options:

The same as the `select` command.

'maps'

Description: Turns processing of maps on (default) or off. If you turn if off, all scraps from selected surveys will be used in the output, no map definitions are taken into account. Useful for debugging map definitions.

Syntax: `maps <on/off>`

'maps-offset'

Description: Turns drawing maps in offset on (default) or off. If you turn if off, all cave passages will be displayed in detail in their actual position. All offset specifications will be completely ignored.

Syntax: `maps-offset <on/off>`

'log'

Description: Turn on logging of various info. Currently only extended elevation processing log is supported.

Syntax: `log extend`

'text'

Description: Specifies translation of any default Therion text in output.

Syntax: `text <language ID> <therion text> <my text>`

Arguments:

- `<language ID>` ▷ standard ISO language identifier (e.g. `en` or `en_GB`)
- `<therion text>` ▷ therion text to translate. For list of therion texts and available translations, see `thlang/texts.txt` file.

'layout'

Description: Specifies layout for 2D maps. Settings which apply to atlas mode are marked 'A'; map mode 'M'.

Syntax: `layout <id> [OPTIONS]`

```
    copy <source layout id>
    cs <coordinate system>
    north <true/grid>
```

```
scale <picture length> <real length>
base-scale <picture length> <real length>
units <metric/imperial>
rotate <number>
symbol-set <symbol-set>
symbol-assign <point/line/area/group/special> <symbol-type> \
                                              <symbol-set>
symbol-hide <point/line/area/group/special> <symbol-type>
symbol-show <point/line/area/group/special> <symbol-type>
symbol-colour <point/line/area/group/special> <symbol-type> <colour>
min-symbol-scale <scale>
fonts-setup <tinysize> <smallsize> <normalsize> <largesize> <hugesize>
size <width> <height> <units>
overlap <value> <units>
page-setup <dimensions> <units>
page-numbers <on/off>
exclude-pages <on/off> <list>
title-pages <on/off>
nav-factor <factor>
nav-size <x-size> <y-size>
transparency <on/off>
opacity <value>
surface <top/bottom/off>
surface-opacity <value>
sketches <on/off>
layers <on/off>
grid <off/top/bottom>
grid-origin <x> <y> <z> <units>
grid-size <x> <y> <z> <units>
grid-coords <off/border/all>
origin <x> <y> <z> <units>
origin-label <x-label> <y-label>
own-pages <number>
page-grid <on/off>
legend <on/off/all>
legend-columns <number>
legend-width <n> <units>
colour-legend <smooth/discrete/off/on>
map-comment <string>
map-header <x> <y> <off/n/s/e/w/ne/nw/se/sw/center>
map-header-bg <on/off>
map-image <x> <y> <n/s/e/w/ne/nw/se/sw/center> <filename>
```

```
    statistics <explo/topo/carto/copyright all/off/number>
               <explo/topo-length on/hide/off>
               <carto/copyright-count on/hide/off>
    scale-bar <length> <units>
    survey-level <N/all>
    language <xx[_YY]>
    colo[u]r-model <cmyk/rgb/grayscale>
    colo[u]r <item> <colour>
    smooth-shading <off/quick>
    debug <on/all/first/second/scrap-names/station-names/off>
    geospatial <on/off>
    doc-author <string>
    doc-keywords <string>
    doc-subject <string>
    doc-title <string>
    code <metapost/tex-map/tex-atlas>
    endcode
endlayout
```

Arguments:

`<id>` ▷ layout identifier (to be used in the `export` command)

Command-like options:

- `copy <source layout id>` ▷ set properties here that are not modified based on the given source layout.

  map presentation-related:

- `scale <picture length> <real length>` ▷ set scale of output map or map atlas (M, A; default: `1 200`)
- `base-scale <picture length> <real length>` ▷ if set, Therion will optically scale the map by a (`scale/base-scale`) factor. This has the same effect as if the map printed in `base-scale` would be photo-reduced to the `scale`. (M, A)
- `rotate <value>` ▷ rotates the map (M, A; default: `0`)
- `units <metric/imperial>` ▷ set output units (M, A; default: `metric`)
- `symbol-set <symbol-set>` ▷ use `symbol-set` for all map symbols, if available. Be aware, that symbol set name is case sensitive. (M, A)

  Therion uses following predefined symbol sets:

  UIS (International Union of Speleology)

  ASF (Australian Speleological Federation)

  AUT (Austrian Speleological Association)

BCRA (British Cave Research Association)

NSS (National Speleological Society/USA)

NZSS (New Zealand Symbol Set)

SBE (Brazilian Speleological Society -*Sociedade Brasileira de Espeleologia*)

SKBB (Speleoklub Banská Bystrica)

- `symbol-assign <point/line/area/group/special> <symbol-type> <symbol-set>` ▷ display a particular symbol in the given symbol-set. This option overrides `symbol-set` option.

  If the symbol has a subtype, `<symbol-type>` argument may have one of the following forms: `type:subtype` or simply `type`, which assigns new symbol set to all subtypes of a given symbol.

  Following symbols may not be used with this option: point *section* (which isn't rendered at all) and all point and line labels (*label*, *remark*, *altitude*, *height*, *passage-height*, *station-name*, *date*). See the chapter *Changing layout/Customizing text labels* for details how to change labels' appearance. (M, A)

  Group may be one of the following: all, centerline, sections, water, speleothems, passage-fills, ice, sediments, equipment.

  There are two special symbols: north-arrow, scale-bar.

- `symbol-hide <point/line/area/group/special> <symbol-type>` ▷ don't display particular symbol or group of symbols.

  You may use `group cave-centerline`, `group surface-centerline`, `point cave-station`, `point surface-station` and `group text` in `symbol-hide` and `symbol-show` commands.

  Use `flag:<entrance/continuation/sink/spring/doline/dig>` as a `<symbol-type>` to hide stations with particular flags (e.g. `symbol-hide point flag:entrance`).

  May be combined with `symbol-show`.(M, A)

- `symbol-show <point/line/area/group/special> <symbol-type>` ▷ display particular symbol or group of symbols. May be combined with `symbol-hide`. (M, A)

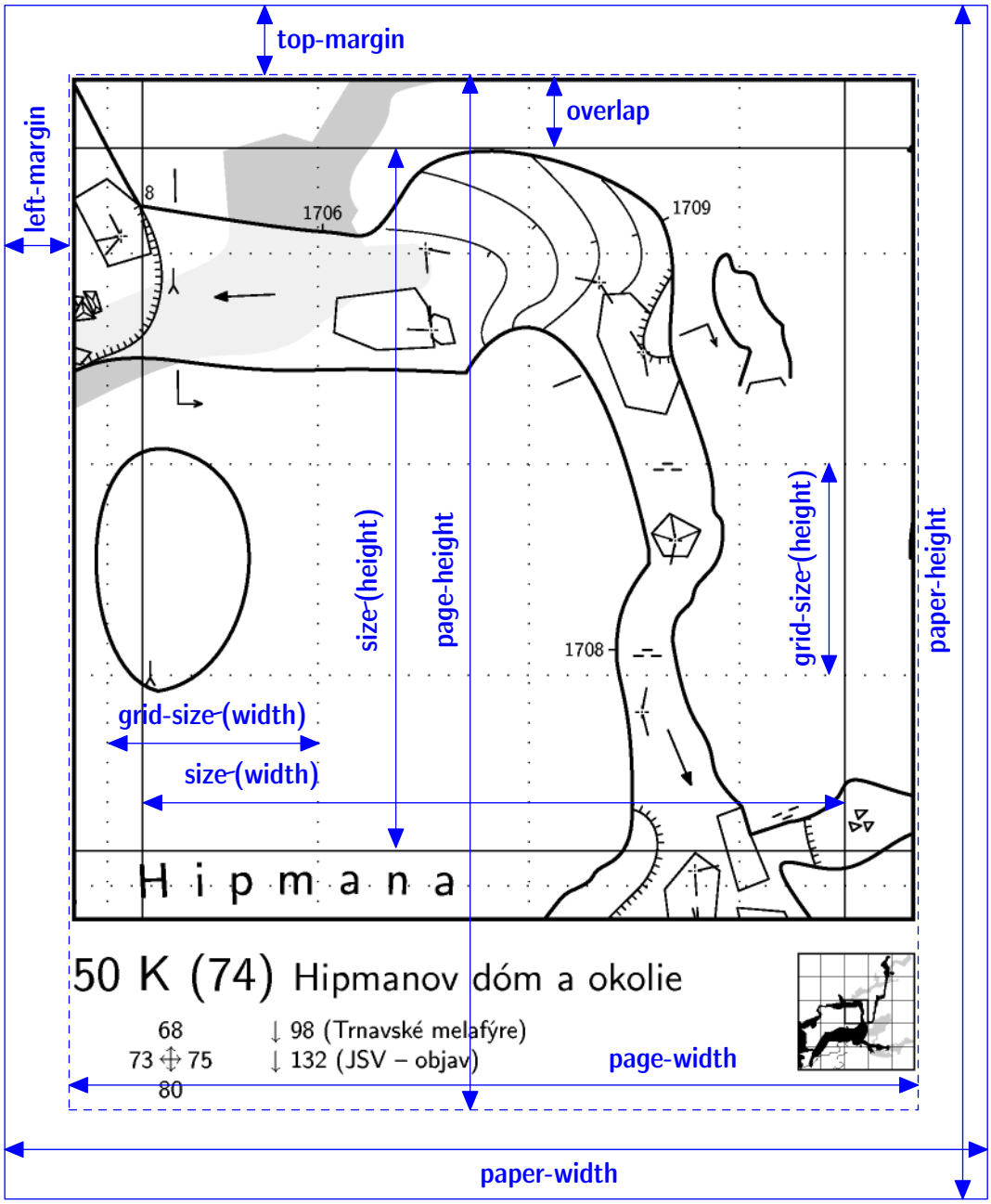
- `symbol-colo[u]r <point/line/area/group/special> <symbol-type> <colour>` ▷ change colour of particular symbol or group of symbols.[64] (M, A)

- `min-symbol-scale <scale>` ▷ define minimal `<scale>`, from which points and lines are displayed on the map. E.g. for `min-symbol-scale M`, no points or lines scaled `S` and `XS` will be shown on the map. `<scale>` has the same format, as `scale` option for points and lines.

[64] Note: colour change currently applies to pattern fills only if (1) output format is PDF and (2) METAPOST version is at least 1.000

- `fonts-setup <tinysize> <smallsize> <normalsize> <largesize> <hugesize>` ▷ specify size of the text in points. `<normalsize>` applies to point label, `<smallsize>` applies to remark and all other point labels. Each of them may apply to line label according to its `-scale` option.

  The defaults are `8 10 12 16 24` for scales upto 1:100; `7 8 10 14 20` for scales upto 1:200; `6 7 8 10 14` for scales upto 1:500 and `5 6 7 8 10` for scales smaller than 1:500.



page layout related:

- `size <width> <height> <units>` ▷ set map size in the atlas mode. If not specified, it will be calculated from `page-setup` and `overlap`. In map mode applies iff `page-grid` is `on` (M, A; default: `18 22.2 cm`)

- `overlap <value> <units>` ▷ set overlap size in paper units in the atlas mode or map margin in the map mode (M, A; default: `1 cm`)
- `page-setup <dimensions> <units>` ▷ set page dimensions in this order: paper-width, paper-height, page-width, page-height, left-margin and top-margin. If not specified, it will be computed from `size` and `overlap` (A; default: `21 29.7 20 28.7 0.5 0.5 cm`)
- `page-numbers <on/off>` ▷ turn on/off page numbering (A; default: `true`)
- `exclude-pages <on/off> <list>` ▷ exclude specified pages from cave atlas. The list may contain page numbers separated by a comma or dash (for intervals) e.g. `2,4-7,9,23` means, that pages 2, 4, 5, 6, 7, 9 and 23 should be omitted. Only the map pages should be counted. (Set `own-pages 0` and `title-pages off` to get the correct page numbers to be excluded.) Changes of `own-pages` or `title-pages` options don't affect page excluding. (A)
- `title-pages <on/off>` ▷ turn on/off title pages before each atlas chapter (A; default: `off`)
- `nav-factor <factor>` ▷ set atlas navigator zoom factor (A; default: `30`)
- `nav-size <x-size> <y-size>` ▷ set number of atlas pages in both directions of navigator (A; default: `2 2`)
- `transparency <on/off>` ▷ set transparency for the passages (underlying passages are also visible) (M, A; default: `on`)
- `opacity <value>` ▷ set opacity value (used if `transparency` is `on`). Value range is 0–100. (M, A; default: `70`)
- `surface-opacity <value>` ▷ set the surface bitmap opacity (used if `transparency` is `on`). Value range is 0–100. (M, A; default: `70`)
- `surface <top/bottom/off>` ▷ set the position of the surface bitmap above/below the map. (M, A; default: `off`)
- `sketches <on/off>` ▷ turn on/off displaying of morphed sketch bitmaps. (M, A; default: `off`)
- `layers <on/off>` ▷ enable/disable PDF 1.5 layers (M, A; default: `on`)
- `grid <off/bottom/top>` ▷ enable/disable grid (optionally coordinates' values may be also displayed) (M, A; default: `off`)
- `cs <coordinate system>` ▷ coordinate system for `origin` and `grid-origin`
- `north <true/grid>` ▷ specify default orientation of the map. By default, true (astronomical) north is used. It is ignored when used with local coordinate system.
- `grid-origin <x> <y> <z> <units>` ▷ set coordinates of grid origin (M, A)
- `grid-size <x> <y> <z> <units>` ▷ set grid size in real units (M, A; default is equal to scalebar size)

- `grid-coords <off/border/all>` ▷ specify where to label grid with coordinates. (M, A; default: `off`)
- `origin <x> <y> <z> <units>` ▷ set origin of atlas pages (M, A)
- `origin-label <x-label> <y-label>` ▷ set label for atlas page which has the lower left corner at the given origin coordinates. May be either a number or a string.[65] (M, A; default: `0 0`) *6.1.0*
- `own-pages <number>` ▷ set number of own pages added before the first page of automatically generated pages in atlas mode (currently required for correct page numbering) (A; default: `0`)
- `page-grid <on/off>` ▷ show pages key plan (M; default: `off`)

map legend related:

- `map-header <x> <y> <off/n/s/e/w/ne/nw/se/sw/center>` ▷ print map header at location specified by `<x> <y>`. Predefined map header contains some basic information about cave: name, scale, north arrow, list of surveyors etc. It is fully customizable (see the chapter *Changing layout* for details). `<x>` is easting (left-right on page). `<y>` is northing (up/down page). Ranges for `<x>` and `<y>` are -100–200. Lower-left corner of the map is `0 0`, upper-right corner is `100 100`. The header is aligned with the specified corner or side to this anchor point. (M; default: `0 100 nw`)
- `map-header-bg <on/off>` ▷ when on, background of map header is filled with background color (e.g. to hide map grid). (M; default: off)
- `map-image <x> <y> <n/s/e/w/ne/nw/se/sw/center> <filename>` ▷ include image[66] specified by `<filename>` into map at location specified by `<x> <y>`. For coordinates and alignment details, see `map-header` specification.
- `legend-width <n> <units>` ▷ legend width (M, A; default: `14 cm`)
- `legend <on/off/all>` ▷ display list of used map symbols in the map header. If set to `all`, all symbols from the current symbol set are displayed. (M, A; default: `off`)
- `colo[u]r-legend <smooth/discrete/off/on>` ▷ set type of map-fg colours legend when map-fg is set to altitude, scrap or map. For compatibility reasons, `on` is equivalent to `smooth`. (M, A; default: `smooth` if applicable)
- `legend-columns <number>` ▷ adjusts the number of legend columns (M, A; default: `2`)
- `map-comment <string>` ▷ optional comment displayed at the map header (M)
- `statistics <explo/topo/carto/copyright all/off/number>` or

[65] String labels form the following sequence, either in lower or upper case: A, B, ..., Z, AA, AB, ...

[66] Note that you can include PDF too, which may be used to combine plan and extended elevation into one nice looking PDF file.

- `statistics <explo/topo-length on/hide/off>` ▷ display some basic statistics; if set to off, team members are sorted alphabetically; otherwise according to their contribution to exploration and surveying (M, A; default: `hide`)
- `statistics <carto/copyright-count on/hide/off>` ▷ display number of scraps for given author/copyright string; if set to off, items are sorted alphabetically; otherwise by number of scraps (M, A; default: `hide`) *5.5.4*
- `scale-bar <length> <units>` ▷ set the length of the scale-bar (M, A)
- `language <xx[_YY]>` ▷ set output language. Available languages are listed on the copyright page. See the *Appendix* if you want to add or customize translations. (M, A)
- `colo[u]r-model <cmyk/rgb/grayscale>` ▷ select the output colour model (M, A; default: `cmyk`).

  The CMYK colour model is intended for printing. Black overprint is used for the best appearance of black lines and texts. The hue of other colours may vary depending on the type or settings of the RIP or printer driver.

  The RGB colour model is intended for screen or multimedia projector presentation. If used for printing, the result will depend on the colour management settings and also on particular colour, as not all RGB colours can be represented on printer.

  The grayscale colour model is meant for printing on B&W printers.

  If input colours (`lookup`, `colour` in the `layout`) do not match the output colour model (e.g. only RGB is defined in the `lookup` and CMYK selected as the `colour-model`), they will be converted to the output model.
- `colo[u]r-profile <cmyk/rgb/grayscale> <filename>` ▷ assign an ICC profile to colours specified in the given colour model. ICC profiles are applied only to drawings, not to embedded raster images like sketches and surface bitmaps. (M, A)
- `colo[u]r <item> <colour>` ▷ customize colour for special map items (`map-fg`, `map-bg`, `preview-above`, `preview-below`, `labels`). Colour range is 0–100 for grayscale, [0–100 0–100 0–100] triplet for RGB colours and [0–100 0–100 0–100 0–100] quadruplet for CMYK colours.[67] RGB colours can be specified in hexadecimal format (e.g. ffeeaa).

  For `map-fg`, you can use `altitude`, `scrap` or `map` as colours. In this case the map is coloured according to altitude, scraps or maps.

  For `map-bg`, you can use `transparent` to omit page background completely.

  For `labels`, you can switch colour `on/off`. If `on`, labels are coloured using the colour of associated scrap.

[67] Note, that not all colour combinations are valid; e.g. no printer will print CMYK [100 100 100 100]. The maximum ink coverage or limit ($C+M+Y+K$) may be around 240 or 300, depending on the printer.

- `smooth-shading <off/quick>` ▷ set the mode of smooth scrap background shading. By default, altitude and depth colour is interpolated across the scrap the `quick` way. Some issues are present if transparent symbol colours are used.[68] More precise modes should be added in the future. If `off`, scrap is filled with single colour. *6.0*
- `debug <on/all/first/second/scrap-names/station-names/off>` ▷ draw scrap in different stages of transformation in different colours to see how Therion distorts map data. See the description of `scrap` command for details. The points with distance changed most during transformation are displayed orange. If `scrap-names` is specified, scrap names are shown for each scrap, `station-names` displays name of each survey station.
- `survey-level <N/all>` ▷ `N` is the number of survey levels displayed next to the station name (M, A; default: `0`).

  PDF related:
- `geospatial <on/off>` ▷ enable/disable geospatial PDF features. When enabled, compatible applications can use this information (e.g. Acrobat Reader for length measurements and finding coordinates; GIS applications like GDAL[69] and QGIS for importing and processing georeferenced data). This feature should be used with `layers on` setting. (M; default: `on`) *6.3*
- `doc-author <string>` ▷ set document author (M, A)
- `doc-keywords <string>` ▷ set document keywords (M, A)
- `doc-subject <string>` ▷ set document subject (M, A)
- `doc-title <string>` ▷ set document title (M, A)

  customization:
- `code <metapost/tex-map/tex-atlas>` ▷ Add/redefine TEX and METAPOST macros here. This allows user to configure various things (like user defined symbols, map and atlas layout at one place &c.) See the chapter *Changing layout* for details.
- `endcode` ▷ should end the TeX and METAPOST sections

'lookup'

Description: Allows to define lookup maps. They are used for defining custom coloring of maps.[70]

Syntax: `lookup <type>[:<index>] [-title "custom title text"]`

[68] Coloured scrap background is visible underneath the semitransparent areas, although only lower-scrap-drawings should be visible.

[69] Raster GDAL tools like `gdal_translate` work fine, while vector tools like `ogr2ogr` support Therion maps starting from version 3.10.0. Note that QGIS uses GDAL library to import PDF data.

[70] see `colour` command.

```
  <parameter(s)> [colour] ["text in legend"]
  ...
endlookup
```

Arguments:

- `<type>` ▷ May be `altitude`, `explo-date`, `topo-date`, `map` or `scrap`.
- `<index>` ▷ the index is a user defined string that allows to distinguish different lookup maps for the same `type`.
- `<parameter>` ▷ the thing, you want to set the color for, e.g. an altitude with type `altitude`. May be distinct values or (if appropriate) bands.
- `[colour]` ▷ grayscale value, RGB triplet, CMYK quadruplet or a combination of eight values (RGB, grayscale, CMYK). Use empty brackets (`[]`) if you want Therion to use its default colour palette together with a text label.
- `[text in legend]` ▷ some optional text to show in the legend.

Example: Banded altitudes. It should generate red-blue scale with desired values.

```
lookup altitude -title "Altitude legend"
   700 [100 0 0] "700 m a.s.l."
   680
   660
   640
   620
   600 [0 0 100] "below 600 m"
 endlookup
```

Example: Using banded altitude specification (with index banded). Notice that if the text is omitted, then the default will display the range, ie 1600 m - 1500 m

```
lookup altitude:banded
   [1500 1600] []          # <- displays "1600 m - 1500 m"
   [1800 1900] [] "cave floor 2"
 endlookup
```

'setup3d'

Syntax:

- `setup3d <value>`

Temporary hack to set sampling distance in meters when generating piecewise linear 3d model from passage walls made of Bézier curves.

'sketch-colors'

Syntax:

- `sketch-colors <number-of-colors>`

This option can be used to reduce size of sketch bitmap images in maps.

'export'

Description: Exports selected surveys or maps.

Syntax:

- `export <type> [OPTIONS]`

Arguments:

- `<type>` ▷ The following export types are supported:

  `model` ▷ 3D model of the cave

  `map` ▷ one page 2D map

  `atlas` ▷ 2D atlas in more pages

  `cave-list` ▷ summary table of caves

  `survey-list` ▷ summary table of surveys

  `continuation-list` ▷ list of possible continuations

  `database` ▷ SQL database with centreline

Options:

common:

- `encoding/enc <encoding>` ▷ set output encoding
- `output/o <file>` ▷ set output file name. If no file name is given the prefix "`cave.`" is used with an extension corresponding to output format.

  If the output filename is given and no output format is specified, the format is determined from the filename extension.

  model:

- `format/fmt <format>` ▷ set model output format. Currently the following output formats are supported: `loch` (native format; default), `compass` (plt file), `survex` (3d file), `dxf`, `esri` (3d shapefiles), `vrml`, `3dmf` and `kml` (Google Earth).
- `enable <walls/[cave/surface-]centerline/splay-shots/surface/all>` and

- `disable <walls/[cave/surface-]centerline/splay-shots/surface/all>` ▷ selects which features to export, if the format supports it. Surface is currently exported in `therion` format only.
- `wall-source <maps/centerline/all/splays>` ▷ set source data for passage wall modeling.

  map/atlas:
- `format/fmt <format>` ▷ set map format. Currently `pdf`, `svg`, `xhtml`[71], `survex`, `dxf`, `esri`[72], `kml` (Google Earth), `xvi`[73] and `bbox`[74] for map; `pdf` for atlas are supported.
- `projection <id>` ▷ unique identifier that specifies the map projection type. (See the `scrap` command for details.)

  If there is no map defined, all scraps in the given projection are exported.

  If there are no scraps with the specified projection then Therion will display centreline from selected surveys.
- `layout <id>` ▷ use predefined map or atlas layout.
- `layout-xxx` ▷ where `xxx` stands for other layout options. Using this you can change some layout properties directly within the export command.
- `encoding/enc <encoding>` ▷ set output encoding

  common for lists:
- `format/fmt <format>` ▷ set continuation output format. Currently the following output formats are supported: `html` (default), `txt`, `kml`[75] and `dbf`.

  continuation-list:
- `attributes <(on)/off>` ▷ set whether to export user defined attributes in continuation list table.
- `filter <(on)/off>` ▷ set whether continuations without comment/text should be filtered out.

  cave-list:
- `location <on/(off)>` ▷ set whether to export coordinates of cave entrances in the table.
- `surveys (on)/off` ▷ exports raw list of caves when set `off`. Otherwise survey structure with aggregated statistics is also displayed.

[71] SVG embedded in XHTML file which contains also legend

[72] ESRI shapefiles. Multiple files are written to a directory with the specified filename.

[73] Xtherion vector image. XVI images may be used in xtherion to draw in-scale maps. Scale (100 DPI image resolution is assumed) and grid-size from layout are used in export.

[74] Text file containing geographic coordinates of lower-left and upper-right corners of the map area.

[75] For cave-list and continuation-list.

database:

- `format/fmt <format>` ▷ currently `sql` and `csv`
- `encoding/enc <encoding>` ▷ set output encoding

File formats summary:

export type	*available formats*
model	loch, dxf, esri, compass, survex, vrml, 3dmf, kml
map	pdf, svg, xhtml, dxf, esri, survex, xvi, kml, bbox
atlas	pdf
database	sql, csv
lists	html, txt, kml, dbf

Running Therion

Now, after mastering data and configuration files, we're ready to run Therion. Usually this is done from the command line in the data directory by typing

```
therion
```

The full syntax is

```
therion [-q] [-L] [-l <log-file>]
        [-s <source-file>] [-p <search-path>]
        [-b/--bezier]
        [-d] [-x] [--use-extern-libs] [<cfg-file>]
```

or

```
therion [-h/--help]
        [-v/--version]
        [--print-encodings]
        [--print-environment]
        [--print-init-file]
        [--print-library-src]
        [--print-symbols]
        [--print-tex-encodings]
        [--print-xtherion-src]
        [--reproducible-output]
        [--generate-output-crc]
        [--verify-output-crc]
```

Arguments:

`<cfg-file>` Therion takes only one optional argument: the name of a configuration file. If no name is specified `thconfig` in the current directory is used. If there is no `thconfig` file (e.g. current directory is not a data directory), Therion exits with an error message.

Options:

- `-d` ▷ Turn on debugging mode. The current implementation creates a temporary directory named `thTMPDIR` (in your system temporary directory) and does not delete any temporary files.
- `-h, --help` ▷ Display short help.
- `-L` ▷ Do not create a log-file. Normally therion writes all the messages into a therion.log file in the current directory.
- `-l <log-file>` ▷ Change the name of the log file.
- `-p <search-path>` ▷ This option is used to set the search path (or list of colon-separated paths) which therion uses to find its source files (if it doesn't find them in the working directory).
- `-q` ▷ Run therion in quiet mode. It will print only warning and error messages to STDERR.
- `--print-encodings` ▷ Print a list of all supported input encodings.
- `--print-tex-encodings` ▷ Print a list of all supported encodings for PDF output.
- `--print-init-file` ▷ Print a default initialization file. For more details see the *Initialization* section in the *Appendix*.
- `--print-environment` ▷ Print environment settings for therion.
- `--print-symbols` ▷ Print a list of all therion supported map symbols in `symbols.xhtml` file.
- `--reproducible-output` ▷ Create reproducible PDF and SVG files. No volatile information is included (e.g. creation date or software version is omitted from the metadata). TEX variables `\thversion` and `\currentdate` are set to predefined constants. This option enforces the use of Therion loop closure.

  To get the same output files on different platforms, it's necessary to install the same set of basic fonts used by Therion[76] and use reasonably recent TEX distribution on all platforms.
- `--generate-output-crc` ▷ Generate a '.crc' file with CRC32 checksum for each output file. Implies `--reproducible-output`.

[76] CM, CS and/or CMCYR font families; the '.pfb' font files have to be exactly the same on all platforms as they are embedded into PDF files.

- `--verify-output-crc` ▷ Verify that the output file has not been changed. Generate reproducible output, calculate CRC32 checksum and check it against one saved in '.crc' file.
- `-s <source-file>` ▷ Set the name of the source file.
- `--use-extern-libs` ▷ Don't copy TEX and METAPOST macros to working directory. TEX and METAPOST should search for them on their own. Use with caution.
- `-v, --version` ▷ Display version information.
- `-x` ▷ Generate file '.xtherion.dat' with additional information for XTherion.

XTherion—compiler

XTherion makes it easier to run Therion especially on systems without a command line prompt. Compiler window is the default window of XTherion. To run Therion it's enough to open a configuration file and press 'F9' or 'Compile' button.

XTherion displays messages from Therion in the lower part of the screen. Each error message is highlighted and is hyperlinked to the source file where the error occurred.

After a first run there are activated additional menus *Survey structure* and *Map structure*. User may comfortably select a survey or map for export by double clicking on some of the items in the tree. Simple click in the *Survey structure* tree displays some basic information about the survey in the *Survey info* menu.

What do we get

Information files

Log file

Besides the messages from Therion and other programs used, the log file contains information about computed values of magnetic declination and meridian convergence, loop errors, scrap distortions and transformations between coordinate reference systems chosen by the Proj library.

Absolute loop error is $\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$, where Δx is the difference between the identical start and end points of the loop before the error distribution measured along the x coordinate axis; similarly for y and z. Percentage loop error is calculated as *absolute error / loop length*. Average error is the simple arithmetic average of all loop errors.

Scrap distortion is calculated using the distortion measure defined for all pairs of points (point symbols, points and control points of line symbols) in the scrap. The measure is calculated as $\frac{|d_a - d_b|}{d_b}$, where d_b is the distance of points before warping and d_a is the distance of points after warping. The maximal and average scrap distortions are calculated as a maximum or average of such measures applied to all pairs of points.

The log file also provides information about transformations between coordinate systems, either user-specified pipelines or the transformations chosen by the Proj library.

XTherion

Therion provides some basic facts about each survey (length, vertical range, N–S range, E–W range, number of shots and stations) if `-x` option is given. This information is displayed in XTherion, *Compiler* window, *Survey info* menu, when some survey from the *Survey structure* menu is selected.

SQL export

SQL export makes it easy to get very detailed and subtle information about the centreline. It is a text file starting with a tables declaration (where '?' stands in the following listing for a maximal value required by the column data)

```
create table SURVEY (ID integer, PARENT_ID integer,
  NAME varchar(?), FULL_NAME varchar(?), TITLE varchar(?));
create table CENTRELINE (ID integer, SURVEY_ID integer,
  TITLE varchar(?), TOPO_DATE date, EXPLO_DATE date,
  LENGTH real, SURFACE_LENGTH real, DUPLICATE_LENGTH real);
create table PERSON (ID integer, NAME varchar(?), SURNAME varchar(?));
create table EXPLO (PERSON_ID integer, CENTRELINE_ID integer);
create table TOPO (PERSON_ID integer, CENTRELINE_ID integer);
create table STATION (ID integer, NAME varchar(?),
  SURVEY_ID integer, X real, Y real, Z real);
create table STATION_FLAG (STATION_ID integer, FLAG char(3));
create table SHOT (ID integer, FROM_ID integer, TO_ID integer,
  CENTRELINE_ID integer, LENGTH real, BEARING real, GRADIENT real,
  ADJ_LENGTH real, ADJ_BEARING real, ADJ_GRADIENT real,
  ERR_LENGTH real, ERR_BEARING real, ERR_GRADIENT real);
create table SHOT_FLAG (SHOT_ID integer, FLAG char(3));
```

which is followed by a mass of SQL insert commands. This file may be loaded into any SQL database (after some database-dependent initialization, which may include running an SQL server and connecting to it, creating a database and connecting to it. A good idea is to start a transaction before loading this file, if the database doesn't start a transaction automatically.) It's important to set-up database encoding to match the one specified in Therion `export database` command.

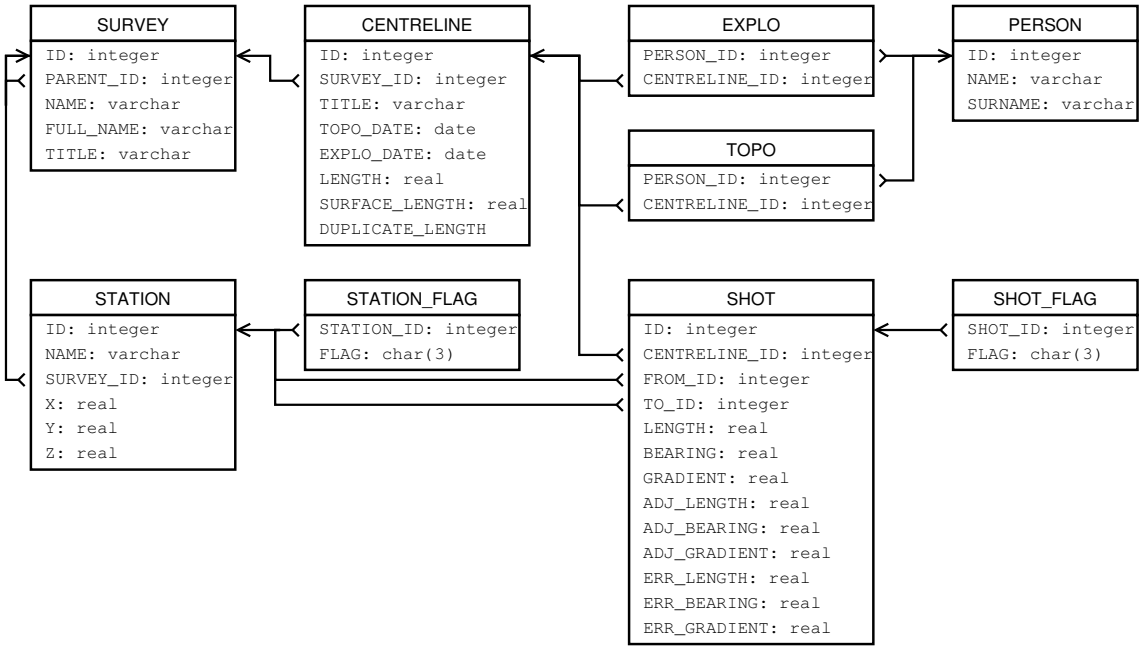


Table and column names are self-explanatory; for undefined or non-existing values `NULL` is used. `FLAG` in `SHOT_FLAG` table is `dpl`, `srf`, `apx`, `spl` or `art` for duplicated, surface, approximate, splay or artificial shots; in `STATION_FLAG` table `ent`, `con`, `fix`, `spr`, `sin`, `dol`, `dig`, `air`, `ove`, `arc` for stations with entrance, continuation, fixed, spring, sink, doline, dig, air-draught, overhang or arch attributes, respectively.

Examples of simple queries follow:

List of survey team members with information about how much each of them has surveyed:

```
select sum(LENGTH), sum(SURFACE_LENGTH), NAME, SURNAME
  from CENTRELINE, TOPO, PERSON
  where CENTRELINE.ID = TOPO.CENTRELINE_ID and PERSON.ID = PERSON_ID
  group by NAME, SURNAME order by 1 desc, 4 asc;
```

Which parts of the cave were surveyed in the year 1998?

```
select TITLE from SURVEY where ID in
  (select SURVEY_ID from CENTRELINE
  where TOPO_DATE between '1998-01-01' and '1998-12-31');
```

How long are the passages lying between 1500 and 1550 m a.s.l.?

```
select sum(LENGTH) from SHOT, STATION S1, STATION S2
  where (S1.Z+S2.Z)/2 between 1500 and 1550 and
  SHOT.FROM_ID = S1.ID and SHOT.TO_ID = S2.ID;
```

Lists—caves, surveys, continuations

Using `export continuation-list` you get an overview of all points in the centreline and scraps marked[77] as a possible continuation.

`export cave-list` gives you a tabular information about surveyed caves (you need to specify `entrance` flags in your data) including length, depth and entrance(s) location.

Detailed information about each sub-survey gives `export survey-list` command. The length includes shots with `approximate` flags, but not `explored`, `duplicate` or `surface`.

2D maps

Maps for printing

Maps are produced in PDF and SVG formats, which may be viewed or printed in a wide variety of viewers. Be sure to uncheck *Fit page to paper* or similar option if you want to print in the exact scale.

[77] Using `station` attribute for centreline points and `point continuation` in scraps.

In atlas mode some additional information is put on each page: page number, map name, and page label.

The numbers of neighbouring pages in N, S, E and W directions, as well as in upper and lower levels are especially useful. There are also hyperlinks at the border of the map if the cave continues on the next page and on the appropriate cells of the Navigator.

PDF files are highly optimized—scraps are stored in XObject forms only once in the document and then referenced on appropriate pages. Therion uses advanced PDF features like transparency and layers.

Created PDF files may be optionally post-processed in applications like pdfTEX or Adobe Acrobat—it's possible to extract or change some pages, add comments or encryption, etc.

If the map was produced using georeferenced data then it also contains georeferencing information. This can be extracted by XTherion to produce georeferenced raster images (see *XTherion/Additional tools* for details).

Maps for GIS

Maps produced in DXF, ESRI or KML formats may be further processed in appropriate software. These maps do not contain visualized map symbols

Special-purpose maps

Map in XVI format contains centreline with LRUD information (and optionally morphed sketches) and can be imported in XTherion to serve as a background for digitization.

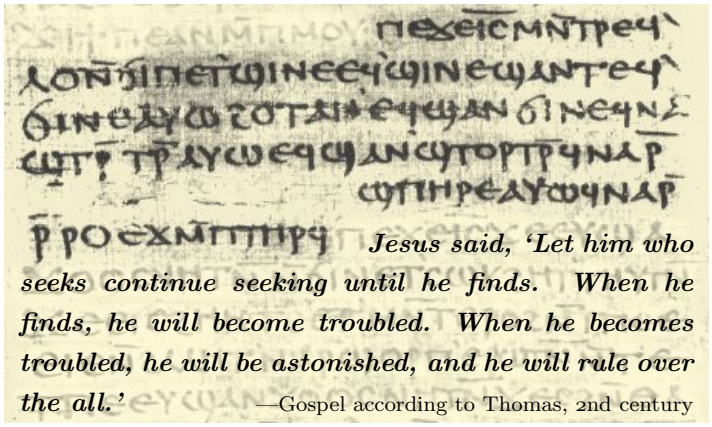
Map in Survex format is intended for a quick preview in Aven.

3D models

Therion may export 3D models in various formats besides its native format. These may be loaded in appropriate viewing, editing or raytracing programs to be printed or further processed. If the format doesn't support arbitrary passage shape definition, only the centreline is included.

Loch

Loch is a 3D model viewer included in the Therion distribution. It supports e.g. high-resolution rendering to file and stereo view using 3D-glasses.



Jesus said, 'Let him who seeks continue seeking until he finds. When he finds, he will become troubled. When he becomes troubled, he will be astonished, and he will rule over the all.'

—Gospel according to Thomas, 2nd century

Changing layout of PDF maps

This chapter is extremely useful if you're not satisfied with the predefined layout of map symbols and maps provided, and want to adapt them to your needs. However, you need to know how to write plain TEX and METAPOST macros to do this.

Page layout in the atlas mode

The `layout` command allows basic page setup in the atlas mode. This is done through its options such as `page-setup` or `overlap`. But there are no options which would specify the position of map, navigator and other elements inside the area defined by `page-width` and `page-height` dimensions; e.g., why is the navigator below the map and not on its right or left side?

There are many possible arrangements for a page. Rather than offer even more options for the `layout` command, Therion uses the TEX language to describe other page layouts. This approach has the advantage that the user has direct access to the advanced typesetting engine without making the language of Therion overly complex.

Therion uses pdfTEX with the *plain* format for typesetting. So you should be familiar with the plain TEX if you wish to define new layouts.

The ultimate reference for plain TEX is

Knuth, D. E.: *The TEXbook*, Reading, Massachusetts, Addison-Wesley [1]1984

For pdfTEX's extensions there is a short manual

Thành, H. T.—Rahtz, S.—Hagen, H.: *The pdfTEX user manual*, available at `http://www.pdftex.org`

The TEX macros are used inside the `code tex-atlas` part of the `layout` command (see the chapter *Processing data* for details). The basic one predefined by Therion is the

\dopage

macro. The idea is simple: for each page Therion defines TEX variables (count, token, and box registers) which contain the page elements (map, navigator, page name etc.). At the end of each page macro \dopage is invoked. This defines the position of each element on the page. By redefining this macro you'll get the desired page layout. Without this redefinition you'll get a standard layout.

Here is the list of variables defined for each page:

Boxes:

- \mapbox ▷ The box containing the map. Its width (height) is set according to the size and overlap options of the layout command to

  size_width + 2*overlap or

  size_height + 2*overlap, respectively

- \navbox ▷ The box containing the navigator, with dimensions

  size_width * (2*nav_size_x+1) / nav_factor or

  size_height * (2*nav_size_y+1) / nav_factor, respectively

  Both \mapbox and \navbox also contain hyperlinks.

Count registers:

- \pointerE, \pointerW, \pointerN, \pointerS contain the page number of the neighbouring pages in the E, W, N and S directions. If there is no such page its page number is set to 0.

- \pagenum current page number

Token registers:

- \pointerU, \pointerD contain information about pages above and below the current page. It consists of one or more concatenated records. Each record has a special format

  page-name|page-number|destination||

  If there are no such pages, the value is set to notdef.

  See the description of the \processpointeritem macro below for how to extract and use this information.

- \pagename ▷ the name of the current map according to the options of the map command.

- \pagelabel ▷ the page label as specified by origin and origin-label options of the layout command.

The following variables are set at the beginning of the document:

- `\hsize`, `\vsize` ▷ TEX page dimensions, set according to `page-width` and `page-height` parameters of the `page-setup` option of the `layout` command. They determine our playground when defining the page layout using the `\dopage` macro.
- `\ifpagenumbering` ▷ This conditional is set true or false according to the `page-numbers` option of the `layout` command.

There are also some predefined macros which help with the processing of `\pointer*` variables:

- `\showpointer` with one of the `\pointerE`, `\pointerW`, `\pointerN` or `\pointerS` as an argument displays the value of the argument. If the value is 0 it doesn't display anything. This is useful because the zero value (no neighbouring page) shouldn't be displayed.
- `\showpointerlist` with one of the `\pointerU` or `\pointerD` as an argument presents the content of this argument. (Which contains `\pointerU` or `\pointerD`, see above.) For each record it calls the macro `\processpointeritem`, which is responsible for data formatting.

  Macro `\showpointerlist` should be used without redefinition in the place where you want to display the content of its argument; for custom data formatting redefine `\processpointeritem` macro.
- `\processpointeritem` has three arguments (page-name, page-number, destination) and visualizes these data. The arguments are delimited as follows

  ```
  \def\processpointeritem#1|#2|#3\endarg{...}
  ```

  An example definition may be

  ```
  \def\processpointeritem#1|#2|#3\endarg{%
    \hbox{\pdfstartlink attr {/Border [0 0 0]}%
          goto name {#3} #2 (#1)\pdfendlink}%
  }
  ```

  (note how to use the *destination* argument), or much simpler (if we don't need hyperlink features):

  ```
  \def\processpointeritem#1|#2|#3\endarg{%
    \hbox{#2 (#1)}%
  }
  ```

For font management there are macros

- `\size[#1]` for size changes,
- `\cmykcolor[#1 #2 #3 #4]`, `\[rgb]color[#1 #2 #3]` and `\graycolor[#1]` for text colour changes (the values should be in the range 0–100),
- `\black` which selects the black color in the appropriate colour model, and
- `\rm`, `\it`, `\bf`, `\ss`, `\si` for type face switching.

See below for a list of predefined texts which may be used in the atlas.

There is also a `\framed` macro which takes a box as an argument and displays the box framed. The frame style can be customized by redefining the `\linestyle` macro which defaults to `1 J 1 j 1.5 w`.

Now we're ready to define the `\dopage` macro. You may choose which of the predefined elements to use. A very simple example would be

```
layout my_layout
  scale 1 200
  page-setup 29.7 21 27.7 19 1 1 cm
  size 26.7 18 cm
  overlap 0.5 cm
  code tex-atlas
    \def\dopage{\box\mapbox}
    \insertmaps
endlayout
```

which defines the landscape A4 layout without the navigator nor any texts. There is only a map on the page.

Note the `\insertmaps` macro. Map pages are inserted at its position. This is not done automatically because you may wish to insert some other pages before the first map page.

More advanced is the default definition of the `\dopage` macro:

```
\def\dopage{%
 \vbox{\centerline{\framed{\mapbox}}
  \bigskip
  \line{%
    \vbox to \ht\navbox{
      \hbox{\size[20]\the\pagelabel
        \ifpagenumbering\space(\the\pagenum)\fi
        \space\size[16]\the\pagename}
      \ifpagenumbering
        \medskip
        \hbox{\qquad\qquad
          \vtop{%
            \hbox to 0pt{\hss\showpointer\pointerN\hss}
            \hbox to 0pt{\llap{\showpointer\pointerW\hskip0.7em}%
              \raise1pt\hbox to 0pt{\hss$\updownarrow$\hss}%
              \raise1pt\hbox to 0pt{\hss$\leftrightarrow$\hss}%
              \rlap{\hskip0.7em\showpointer\pointerE}}
              \hbox to 0pt{\hss\showpointer\pointerS\hss}
          }\qquad\qquad
          \vtop{
```

```
            \def\arr{$\uparrow$}
            \showpointerlist\pointerU
            \def\arr{$\downarrow$}
            \showpointerlist\pointerD
          }
        }
      \fi
      \vss
      \scalebar
    }\hss
    \box\navbox
  }
 }
}
```

Using other plain TEX macros or TEX primitives it's possible to add other features, e.g. a different layout for odd and even pages; headers and footers; or adding a logo to each page.

In addition to the map pages, the atlas contains additional items: title page, basic facts about the cave, legend with used map symbols etc.

Therion automatically generates a list of used map symbols and lists of people who have discovered, surveyed and drawn the selected part of the cave. Following token registers may be used (according to the user's requirements before or after the `\insertmaps` macro):

- `\explotitle`, `\topotitle`, `\cartotitle` ▷ translated titles
- `\exploteam`, `\topoteam`, `\cartoteam` ▷ participating members (according to `team`, `explo-team` options for `centreline` and `author` option of scraps)
- `\explodate`, `\topodate`, `\cartodate` ▷ corresponding dates
- `\comment` ▷ is set according to `map-comment` option of the `layout` command
- `\copyrights` ▷ is set according to copyright options for surveys and other objects
- `\cavename` ▷ name of the exported map; set according to `-title` option of the exported map
- `\cavelength`, `\cavedepth` ▷ approximate length and depth of the displayed map
- `\cavelengthtitle`, `\cavedepthtitle` ▷ translated labels
- `\cavemaxz`, `\caveminz` ▷ altitude max/min value
- `\thversion` ▷ current Therion version
- `\currentdate` ▷ current date
- `\outcscode`, `\outcsname` ▷ output coordinate system code and name

- `\northdir` ▷ 'true' or 'grid'
- `\magdecl` ▷ magnetic declination in degrees
- `\gridconv` ▷ grid meridian convergence in degrees

There is a macro `\atlastitlepages` which combines most of the token registers mentioned above to get simple preformatted atlas introductory pages.

For legend displaying there are

- `\iflegend` ▷ conditional; true iff `legend` option of the `layout` command was set to `on` or `all` values
- `\legendtitle` ▷ token register containing translated legend title
- `\insertlegend` ▷ macro for inserting legend symbols pictures with translated descriptions in the specified number of columns (according to `legend-columns` layout option)
- `\formattedlegend` ▷ combines all three above commands to get preformatted legend with header and symbols typeset in two[78] columns if `legend` option is set `on`

North arrow, altitude bar and scale bar may be displayed using

- `\ifnortharrow` ▷ conditional; true if map projection is plan and symbol north-arrow is not hidden in `layout`
- `\ifscalebar` ▷ conditional; true if the scale bar is not hidden
- `\ifaltitudebar` ▷ conditional; true if the altitude bar is not hidden
- `\northarrow` ▷ macro containing PDF form with the north arrow
- `\scalebar` ▷ macro containing PDF form with the scale bar
- `\altitudebar` ▷ macro containing PDF form with the altitude bar

There is a general-purpose macro for typesetting in multiple columns[79]:

- `\begmulti <i>`, `\endmulti` ▷ text between these macros is typeset in `<i>` columns

An example of how to create an atlas with lists of surveyors etc. followed by map pages and with legend at the end:

```
code tex-atlas
  \atlastitlepages
  \insertmaps
  \formattedlegend
```

To use a relative path when including another TEX file use the `\inputrel` macro instead of `\input`.

[78] Default; adjust the `legend-columns` layout option to get more or less of them.

[79] Not to be used with map legend, where multiple columns are to be adjusted by `legend-columns` layout option

Page layout in the map mode

In the map mode it's possible to use a lot of predefined variables which are described in the previous chapter:

`\cavename`, `\comment`, `\copyrights`, `\explotitle`, `\topotitle`, `\cartotitle`, `\exploteam`, `\topoteam`, `\cartoteam`, `\explodate`, `\topodate`, `\cartodate`, `\cavelength`, `\cavedepth`, `\cavelengthtitle`, `\cavedepthtitle`, `\cavemaxz`, `\caveminz`, `\thversion`, `\currentdate`, `\outcscode`, `\outcsname`, `\northdir`, `\magdecl`, `\gridconv`, `\ifnortharrow`, `\ifscalebar`, `\northarrow`, `\scalebar`, `\iflegend`, `\legendtitle`, `\insertlegend`, `\begmulti <i>`, `\endmulti`, `\formattedlegend`, `\legendcolumns`.

In order to place them somewhere on the map page, you have to define `\maplayout` macro in the `code tex-map` section of the `layout` command. It should contain one or more `\legendbox` invocations. The `\legendbox` macro has four parameters: coordinates ranging 0–100, alignment specification (N, E, S, W, NE, SE, SW, NW or C) and the content to be displayed. To show the content of variables marked as *count registers* or *token registers* in the previous chapter, you need to prefix them with `\the`.

A simple example is

```
\def\maplayout{
  \legendbox{0}{100}{NW}{\northarrow}
  \legendbox{100}{100}{W}{\the\thversion}
}
```

which displays north arrow in the upper-left corner of the map sheet and the version of Therion in the upper-right corner.

The fourth parameter may also be a `\loadpicture` macro, which includes a picture in PDF, JPEG or PNG format. Although it accepts a relative or absolute path, in some cases[80] you need to convert a relative path to absolute and save it for further use:

```
\savepath{mypath1}{../data/picture.png}
\def\maplayout{
  \legendbox{100}{80}{NW}{\loadpicture{mypath1}}
}
```

For the user's convenience, there is `\legendcontent` token register. It contains preformatted cave name, north arrow, scale bar, explo/topo/carto teams, comment, copyrights and legend. (The `\legendcontent` is also used in the default map layout definition: `\def\maplayout{\legendbox{0}{100}{NW}{\the\legendcontent}}`).

The width of the above text may be adjusted by `\legendwidth` dimen register (its default value is set by `legend-width` layout option). The colour and size of texts in the

[80] If you combine multiple layouts stored in different directories.

preformatted legend can be easily changed using \legendtextcolor, \legendtextsize, \legendtextsectionsize and \legendtextheadersize token registers, e.g. for large blue text:

```
code tex-map
  \legendwidth=20cm
  \legendtextcolor={\cmykcolor[0 30 50 50]}
  \legendtextsize={\size[20]}
  \legendtextheadersize={\size[60]}
```

It is possible to display the whole map framed by setting the \framethickness dimen register to a positive value, e.g. 0.5mm.

Customizing text labels

Starting with the release 5.4.1 you can use fonts-setup layout option instead of the METAPOST macro fonts_setup().

New map symbols

Therion's layout command makes it easy to switch among various predefined map symbol sets. If there is no such symbol or symbol set you want, it's possible to design new map symbols.

However, this requires knowledge of the METAPOST language, which is used for map visualization. It's described in

Hobby, J. D.: *A User's Manual for MetaPost*, available at
https://www.tug.org/docs/metapost/mpman.pdf

The user may also benefit from comprehensive reference to the METAFONT language, which is quite similar to METAPOST:

Knuth, D. E.: *The METAFONTbook*, Reading, Massachusetts, Addison-Wesley [1]1986

New symbols may be defined in the code metapost section of the layout command. This makes it easy to add new symbols at the run-time. It is also possible to add symbols permanently by compiling them into Therion executable (see the *Appendix* for instructions on how to do this).

Each symbol has to have a unique name, which consists of the following items:

- one of the letters 'p', 'l', 'a', 's' for point, line, area or special symbols, respectively;
- underscore character;
- symbol type as listed in the chapter *Data format* with all dashes removed;
- if the symbol has a subtype, add an underscore character and subtype;

- underscore character;
- symbol set identifier in uppercase

Example: standard name for a point 'water-flow' symbol with a 'permanent' subtype in the 'MY' set is `p_waterflow_permanent_MY`. Standard name for user-defined symbol types should not include symbol set identifier, e.g. `p_u_bat`.

Each new symbol has to be registered by a macro call

```
initsymbol("<standard-name>");
```

unless it's compiled into Therion executable.

There are four predefined pens *PenA* (thickest) ... *PenD* (thinnest), which should be used for all drawings. For drawing and filling use `thdraw` and `thfill` commands instead of METAPOST's `draw` and `fill`.

The following variables are also available:[81]

- boolean `ATTR__shotflag_splay`, `ATTR__shotflag_duplicate`, `ATTR__shotflag_approx` ▷ set for line survey
- boolean `ATTR__stationflag_splay` ▷ set true for endstations of splay shots
- boolean `ATTR__scrap_centerline` ▷ set true for scraps created from centreline
- boolean `ATTR__elevation` ▷ true for (extended) elevation, false for plan projection
- numeric `ATTR__height` ▷ height of a pit or wall:pit
- string `ATTR__id` ▷ contains current object ID
- string `ATTR__survey` ▷ contains current survey name
- string `ATTR__scrap` ▷ contains current scrap name
- picture `ATTR__text` ▷ contains typeset text e.g. for point continuation
- string `NorthDir` ▷ 'true' or 'grid'
- numeric `MagDecl` ▷ magnetic declination in degrees
- numeric `GridConv` ▷ grid meridian convergence in degrees

If you need to include some METAPOST definitions from a file specified by a relative path, you need to use the macro `inputrel("relative/path.mp")`.

[81] If names clash with Therion commands (like `color`), you can add an exclamation mark '`!`' to prevent Therion parsing the line: `! color myNewColorDef;`

Point symbols

Point symbols are defined as macros using `def ... enddef;` commands. The majority of point symbol definitions have four arguments: position (pair), rotation (numeric), scale (numeric) and alignment (pair). Exceptions are *section* which has no visual representation; all *labels*, which require special treatment as described in the previous chapter, and *station* which takes only one argument: position (pair).

All point symbols are drawn in local coordinates with the length unit u. Recommended ranges are $\langle -0.5u, 0.5u \rangle$ in both axes. The symbol should be centered at the coordinates' origin. For the final map, all drawings are transformed as specified in the T transformation variable, so it's necessary to set this variable before drawing.

This is usually done in two steps (assume that the four arguments are P, R, S, A):

- set the U pair variable to $\left(\frac{width}{2}, \frac{height}{2}\right)$ of the symbol for the correct alignment. The alignment argument A is a pair representing ratios $\left(\frac{shift_x}{U_x}\right)$ and $\left(\frac{shift_y}{U_y}\right)$.

  (Hence `aligned A` means `shifted (xpart A * xpart U, ypart A * ypart U)`.)

- set the T transformation variable

  ```
  T:=identity aligned A rotated R scaled S shifted P;
  ```

For drawing and filling use `thdraw` and `thfill` commands instead of METAPOST's `draw` and `fill`. These automatically take care of T transformation.

An example definition may be

```
def p_entrance_UIS (expr P,R,S,A)=
  U:=(.2u,.5u);
  T:=identity aligned A rotated R scaled S shifted P;
  thfill (-.2u,-.5u)--(0,.5u)--(.2u,-.5u)--cycle;
enddef;
initsymbol("p_entrance_UIS");
```

Line symbols

Line symbols differ from point symbols in respect that there is no local coordinate system. Each line symbol gets the *path* in absolute coordinates as the first argument. Therefore it's necessary to set T variable to `identity` before drawing.

The following symbols take additional arguments:

- arrow ▷ numeric: 0 is no arrows, 1 arrow at the end, 2 begin, 3 both ends
- contour ▷ text: list of points which get the tick or one of -1, -2 or -3 to mark undefined tick, tick in the middle or no tick, respectively

- section ▷ text: list of points which get the orientation arrow or −1 to indicate no arrows
- slope ▷ numeric: 0 no border, 1 border; text: list of (point,direction,length) triplets

Usage example:

```
def l_wall_bedrock_UIS (expr P) =
  T:=identity;
  pickup PenA;
  thdraw P;
enddef;
initsymbol("l_wall_bedrock_UIS");
```

Area symbols

Areas are similar to lines: they take only one argument – *path* in absolute coordinates.

You may fill them in three ways:

- fill a uniform or randomised grid in a temporary picture (having dimensions `bbox path`) with some point symbols; clip it according to the `path` and add to the `currentpicture`
- fill `path` with a solid colour
- fill `path` with a predefined pattern using a `withpattern` keyword.

Patterns are defined using the same user interface (except the `patterncolor` macro) as described in the article

> Bolek, P.: "METAPOST and patterns," *TUGboat*, 3, XIX (1998), pp. 276–283, available online at https://www.tug.org/TUGboat/Articles/tb19-3/tb60bolek.pdf

You may use standard METAPOST `draw` and similar macros without setting of T variable in the pattern definitions. In PDF output, all patterns are *uncolored*—this means that any colour information specified in the pattern definition is ignored and the colour is assigned later, when the pattern is used (using `symbol-colour` layout option).

An example of how to define and use patterns:

```
beginpattern(pattern_water_UIS);
  draw origin--10up withpen pensquare scaled (0.02u);
  patternxstep(.18u);
  patterntransform(identity rotated 45);
endpattern;

def a_water_UIS (expr p) =
  T:=identity;
  thclean p;
  thfill p withpattern pattern_water_UIS;
enddef;
initsymbol("a_water_UIS");
```

Special symbols

There are currently two special symbols: scale bar and north arrow. Both are experimental and subject to change.

1. *When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.*
2. *The only way of discovering the limits of the possible is to venture a little way past them into the impossible.*
3. *Any sufficiently advanced technology is indistinguishable from magic.*

—Arthur C. Clarke, 1973

Appendix

Compilation

Installing the dependencies

If you want to compile Therion from source code and run it, you need (first three are required only during compilation):

- GNU C/C++ compiler or Clang
- CMake
- Perl
- Python 3
- PROJ library (`https://proj.org/`). Supported versions are 6.3.1 and newer (except 7.0.0).
- fmt (`https://github.com/fmtlib/fmt`).
- Tcl/Tk 8.4.3 and newer (`https://www.tcl.tk`) with *BWidget* widget set (`https://sourceforge.net/projects/tcllib/`) and optionally *tkImg* extension (`https://sourceforge.net/projects/tkimg/`).
- TEX distribution with at least TEX with Plain format, recent pdfTEX, and METAPOST (`https://www.tug.org`).
- LCDF Typetools package (`https://www.lcdf.org/type/`)
- *ghostscript* if you want to create calibrated images from georeferenced PDF maps.

To compile Loch, you need

- freetype 2 and newer; `pkg-config freetype2` must work
- wxWidgets 3 and newer; `wx-config` must work
- VTK 7.0 and newer
- libjpeg, libpng, zlib

All programs (with the exception of BWidget and tkImg package) are usually included in Linux, Unix or MacOS X distributions. For Windows consider using MinGW and MSYS2 (https://www.msys2.org/). It's a distribution of GNU utilities with GNU make and GCC. (BTW, why not to use precompiled Windows version?)

Installing dependencies on **Ubuntu 26.04**:

```
sudo apt install bwidget catch2 cmake gcc ghostscript imagemagick lcdf-
typetools libfmt-dev libfreetype-dev libjpeg-dev libpng-dev libproj-dev
libtk-img-dev libvtk9-dev libwxgtk3.2-dev tcl-dev texlive-binaries
texlive-metapost zlib1g-dev
```

Installing dependencies in **Fedora 44**:

```
sudo dnf install brotli-devel bwidget catch2-devel cmake fmt-devel g++
ImageMagick-c++-devel jbigkit-devel krb5-devel libidn2-devel libjpeg-turbo-
devel libnghttp2 libnghttp2-devel libpsl-devel libssh-devel libwebp-devel
openldap-devel proj-devel texlive-metapost texlive-pdftex-quiet
texlive-scheme-basic tkimg vtk-devel wxGTK-devel
```

Installing dependencies in **termux** on **Android**:

```
pkg install clang cmake fmt ninja perl proj tcl texlive-bin
```

Then run `termux-install-tl` to install additional texlive packages and `termux-setup-storage` to make Android data directories accessible from termux.

Check also `therion/.github/workflows/` for complete configurations for Ubuntu, Mac OS X and Windows.

Using CMake

Unpack the source distribution `therion-6.*.tar.gz` and create a separate directory for the build, e.g.: *6.0*

- `cd therion && mkdir build && cd build`
- `cmake [parameters] ..`
- `make -j4`

CMake parameters

Here is a selection of parameters which can be used with cmake:

- `-G <generator>` ▷ specify the generator, e.g. `-G Ninja` to use *Ninja* build system instead of make, or `-G "MSYS Makefiles"` to build using *make* under MSYS2
- `-DUSE_BUNDLED_SHAPELIB=OFF` ▷ use the system Shapelib library
- `-DUSE_BUNDLED_CATCH2=ON` ▷ use the bundled version of Catch2 library
- `-DECM_ENABLE_SANITIZERS=<option>` ▷ use runtime sanitizers, relevant options:

`address` – detects invalid memory accesses, use-after-free, double free, memory leaks, useful for debugging

`undefined` – detects undefined behavior, for example use of uninitialized values

- `-DCMAKE_BUILD_TYPE=<option>` ▷ build type, relevant options:

  `Debug` – compile with debug symbols and asserts

  `Release` – compile with optimizations

  `RelWithDebInfo` – compile with debug symbols and optimizations, useful for profiling and debugging

- `-DTHBOOK_FORMAT=<option>` ▷ set the output size of the Therion book (most of the images are omitted in smaller sizes); relevant options:

  `0` – A4 portrait

  `1` – small screen portrait (some lines and images don't fit)

  `2` – small screen landscape

  `3` – ebook reader optimized

The following CMake components can be used to selectively install a part of the package: th-runtime, loch-runtime, th-docs, loch-docs.

Hacker's guide

Compilation on Android *6.3*

Install the dependencies as listed above and run the following in the therion source directory:

- `mkdir build && cd build`
- `cmake -DUSE_BUNDLED_CATCH2=ON -DBUILD_LOCH=OFF -G Ninja ..`
- `ninja`

Adding new translations

Therion supports translation of map labels. Suppose you want to add a new language `xx`.

- run '`perl process.pl export xx`' in the 'thlang' Therion source subdirectory. This creates a file `texts_xx.txt`. This file is UTF-8 encoded.
- edit the `texts_xx.txt` file. Add your translations at lines beginning with '`xx:`'.
- run `make update`
- compile Therion

Adding new encodings

Although UTF-8 Unicode encoding covers all characters which Therion is able to process, it may be inconvenient to use it. In that case it's possible to add support for any 8-bit encoding for text input files. Copy a translation file to the `thchencdata` directory; add its name to 'ifiles' hash in the beginning of the Perl script `generate.pl`; run it and recompile Therion.

The translation file should contain two hexadecimal values of a character (first one in the 8-bit encoding, second one in Unicode) in each line. Possible comments follow the '`#`' character.

Adding new TEX encodings

It's easy to add new encodings for 2D map output.[82] Copy an appropriate encoding mapping file with an `*.enc` extension to the `texenc/encodings`, run the Perl script `mktexenc.pl` located in the `texenc` directory and compile Therion.

Therion uses the same encoding files as `afm2tfm` program from the TEX distribution, which has the same format as an encoding vector in a PostScript font. You may find more details in the chapter *6.3.1.5 Encoding file format* in the documentation to Dvips program.

Generating new TEX and METAPOST *headers*

Therion uses TEX and METAPOST for 2D map visualization and typesetting. Predefined macros are compiled into the Therion executable and are copied to the working directory just before running METAPOST and TEX (unless the `--use-extern-libs` option is used). Layout command makes it possible to modify some macros in the configuration file at the run-time.

However, it's possible to make permanent changes to the macro files. After modifying the files in the `src/therion-mpost` and `src/therion-tex` directories it's necessary to run Therion executable compilation again, which will also generate C++ header files.

Updating the geomagnetic model

Therion uses the IGRF model to calculate the magnetic declination. Download the model in a txt format from `https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html` and save it in the `geomag/` directory (e.g. `igrfXYcoeffs.txt`). Run `./igrf2c.py igrfXYcoeffs.txt` which creates `thgeomagdata.h` in the Therion source directory and recompile Therion.

To test the model, extract the file `sample_out_IGRFXY.txt`, which is included in the Geomag distribution available on the same web page. Put it into the `geomag/test/` directory, run `./build.sh sample_out_IGRFXY.txt` and check the lines with an exclamation mark in the output.

[82] This section applies to old-style font selection using `tex-fonts` command in the initialization file and is obsolete when using `pdf-fonts` command.

Environment variables

Therion reads following environment variables:

- `THERION` ▷ [not required] search path for (x)therion.ini file(s)
- `HOME` (`HOMEDRIVE` + `HOMEPATH` on WinXP) ▷ [not required, but usually present on your system] search path for (x)therion.ini file(s)
- `TEMP`, `TMP` ▷ system temporary directory, where Therion stores temporary files (in a directory named `th$PID$`, where `$PID$` is a process ID), unless `tmp-path` is specified in the initialization file.

Consult the documentation of your OS how to set them.

Initialization files

Therion's and XTherion's system dependent settings are specified in the file therion.ini or xtherion.ini, respectively. They are searched for in the following directories:

- on UNIX: `.`, `$THERION`, `$HOME/.therion`, `/etc`, `/usr/etc`, `/usr/local/etc`
- on Windows: `.`, `$THERION`, `$HOME\.therion`, `<Therion-installation-directory>`, `C:\WINDOWS`, `C:\WINNT`, `C:\Program Files\Therion`

Therion

If no file is found Therion uses its default settings. If you want to list them, use `--print-init-file` option. The initialization file is read like any other therion file. (Empty lines or lines starting with '`#`' are ignored; lines ending with a backslash continue on next line.) Currently supported initialization commands follow.

- `loop-closure <therion/survex>`

  By default, survex is used if present, otherwise therion.

- `encoding-default <encoding-name>`

  Set the default output encoding (currently unused).

- `encoding-sql <encoding-name>`

  Set the default output encoding for SQL export.

- `language <xx[_YY]>`

  Default output language. See the copyright page for the list of available languages.

- `units <metric/imperial>`

  Set default units.

- `mpost-path <file-path>`

  Set the full path to a METAPOST executable if Therion can't find it ("`mpost`" is the default).

- `mpost-options <string>`

  Set METAPOST options.

- `pdftex-path <file-path>`

  Set the full path to a pdfTEX executable if Therion can't find it ("`pdftex`" is the default).

- `source-path <directory>`

  Path to data and configuration files. Used mostly for system-wide grades and layout definitions.

- `tmp-path <directory>`

  Path where temporary directory should be created.

- `tmp-remove <OS command>`

  System command to delete files from the temporary directory.

- `tex-env <on/off>`

  [Works on Windows only.] When set to `off` (default), Therion temporarily clears all environment variables related to TEX. Useful if there is other TEXdistribution installed on your system which had set-up any environment variables, which could confuse TEX and METAPOST programs supplied in Therion for Windows distribution.

  Set to `on` if you use other TEX distribution for maps processing.

- `text <language ID> <therion text> <my text>`

  Using this option you can change any default therion text translation in output. For list of therion texts and available translations, see `thlang/texts.txt` file.

- `cs-def <id> <proj_def>`

  Define a new coordinate system `<id>` using Proj syntax. It is recommended to specify coordinate systems using the EPSG codes directly (e.g. EPSG:4258), or using Well-known text (WKT) definition.

- `cs-trans <cs1> <cs2> <proj-pipeline>` *6.1.0*

  Define a transformation pipeline between two coordinate systems.[83] Both `cs1` and `cs2` can be lists of aliases enclosed in brackets.

[83] See `https://proj.org/usage/transformation.html` for details of pipelines definition.

Therion contains a built-in database of transformation pipelines.[84] You can override any built-in definition either by redefining the pipeline using `cs-trans` or make it ignored by using an empty string in the `cs-trans` definition.

If the pipeline references transformations grids which are not installed locally, Therion attempts to download them from `cdn.proj.org` if `proj-missing-grid` option is set to `download` (otherwise it prints an error message and stops).

This option is ignored if Therion is using PROJ v6.

Example: `cs-trans [jtsk epsg:5513] [etrs34 epsg:25834] "+proj=pipeline +step +inv +proj=krovak +axis=wsu +lat_0=49.5 +lon_0=24.8333333333333 +alpha=30.2881397527778 +k=0.9999 +x_0=0 +y_0=0 +ellps=bessel +step +inv +proj=hgridshift +grids=sk_gku_JTSK03_to_JTSK.tif +step +proj=push +v_3 +step +proj=cart +ellps=bessel +step +proj=helmert +x=485.021 +y=169.465 +z=483.839 +rx=-7.786342 +ry=-4.397554 +rz=-4.102655 +s=0 +convention=coordinate_frame +step +inv +proj=cart +ellps=GRS80 +step +proj=pop +v_3 +step +proj=utm +zone=34 +ellps=GRS80"`

- `proj-missing-grid <ignore/warn/fail/cache/download>` *5.5.1*

  Set missing transformation grids handling if `proj-auto` is `on` or a custom transformation pipeline in `cs-trans` references such grids. The grids are used to achieve a better transformation precision between some coordinate systems.

  For `cs-trans` pipelines, only `download` tries to download the grid; all other options are equivalent to `fail`. If `proj-auto` is used to find the best transformation, the following applies:

  `ignore` silently lets Proj to choose other transformation which doesn't use the missing grid(s); this usually leads to decreased transformation accuracy (say metres instead of centimetres). See the `log` file for a list of the used transformations and their precisions.

  `warn` behaves like `ignore`, but prints warnings about missing grids (the download links are usually displayed as well).[85]

  `fail` stops after the first missing grid is detected and displays the download link.

  `cache` enables the network connectivity and lets Proj to download the missing parts of grids from the Internet. The information is stored in a local cache.[86] As only parts of grids covering the current area are downloaded, it's potentially faster and less space consuming then `download`. The downside is that the local cache is not used by Proj if the network is disabled (in Therion you have to use the `cache` mode to use the local cache).

[84] Mostly for JTSK to ETRS89 transformations. The definitions are in the `thcsdata.tcl` file in the source code, `proj_transformations` variable.

[85] See `https://proj.org/resource_files.html` for the instructions where to put the downloaded grids.

[86] See `https://proj.org/usage/network.html`

`download` temporarily enables the network connectivity and lets Proj to download the missing grids from the Internet. The grids are used in *subsequent* runs (in any mode).

The default setting is `download`. If Proj v6 is used by Therion, the settings `cache` and `download` are equivalent to `fail`, as the network connectivity is a feature of Proj v7 and later. *6.1.0*

- `pdf-fonts <rm> <it> <bf> <ss> <si>`

  Set-up fonts to be used in PDF maps. The command has to be followed by paths specifying where regular, italic, bold, sans-serif and sans-serif oblique fonts are located in your system. TrueType and OpenType fonts are supported.

  Therion requires LCDF Typetools to be installed on your system to use this command. Example:

```
pdf-fonts  "/usr/share/fonts/Serif.ttf" \
           "/usr/share/fonts/Serif-Italic.ttf" \
           "/usr/share/fonts/Serif-Bold.ttf" \
           "/usr/share/fonts/Sans.ttf" \
           "/usr/share/fonts/Sans-Oblique.ttf"
```

- `otf2pfb <on/off>`

  When set to `on` (default), OpenType fonts used in `pdf-fonts` are converted to PFB fonts, if they are PostScript-based. Some information is lost in the PFB format, but there is advantage that pdfTEX can embed subset of PFB fonts (in contrast with OpenType fonts which must be fully embedded).

- `tex-fonts <encoding> <rm> <it> <bf> <ss> <si>`

  Original and more complicated way to set-up fonts for PDF maps. You need to explicitly specify encoding (maximum 256 characters from the font that will be used). The list of currently supported encodings gives the `--print-tex-encodings` command line option. The same encoding must be used while generating TEX metrics (`*.tfm` files) for those fonts (e.g. with the afm2tfm program) and this encoding must be explicitly given also in the pdfTEX's map file. The only exception is the base set of Computer Modern fonts, which use 'raw' encoding. This encoding doesn't need to be specified in the pdfTEX's map file.

  Encoding has to be followed by five font specifications for regular, italic, bold, sans-serif and sans-serif oblique styles. Default setting is `tex-fonts raw cmr10 cmti10 cmbx10 cmss10 cmssi10`

  Example how to use other fonts (e.g. TrueType Palatino in xl2 (an encoding derived from ISO8859-2) encoding). Run:

```
ttf2afm -e xl2.enc -o palatino.afm palatino.ttf
```

```
afm2tfm palatino.afm -u -v vpalatino -T xl2.enc
```

`vptovf vpalatino.vpl vpalatino.vf vpalatino.tfm`

You get files `vpalatino.vf`, `vpalatino.tfm` and `palatino.tfm`. Add the line

`palatino <xl2.enc <palatino.ttf`

to the pdfTEX's map file. The same should be done for the italic and bold faces and corresponding sans-serif and sans-serif-oblique fonts. If you're lazy try

`tex-fonts xl2 palatino palatino palatino palatino palatino`

(We should use actually virtual font `vpalatino` instead of `palatino`, which contains no kerning or ligatures, but pdfTEX doesn't support `\pdfincludechars` command on virtual fonts. To be improved.)

If you want to add some unsupported encodings, read the chapter *Compilation / Hacker's guide*.

- `tex-fonts-optional <encoding> <rm> <it> <bf> <ss> <si>`

  Similar to `tex-fonts`, but tests if the TEX fonts are installed in the system. It does nothing if any of the specified fonts is not present.

  This setting is used by default for Czech/Slovak and cyrillic fonts to avoid METAPOST errors on systems without these fonts present.

  As the test takes some time (pdfTeX instance is run), you might disable the default behaviour completely by setting `tex-fonts` in the INI file.

- `tex-refs-registers <on/off>`

  Switch between using count registers and macros to store references to graphical objects in TEX. Each approach has some advantages, see the section *Limitations*.

XTherion

Initialization file for XTherion is actually a Tcl script evaluated when XTherion starts. The file is commented; see the comments for details.

Limitations

- scrap size ▷

  METAPOST in the default ('scaled') mode: $\approx 2.8 \times 2.8$ m in the output scale

  METAPOST in the 'double' mode:[87] practically no limit[88]

[87] To run METAPOST in the 'double' mode, set `mpost-options "-numbersystem=double"` in the initialization file. It's not recommended to use arbitrary-precision modes 'decimal' and 'binary', as there are still bugs in their implementation and they are much slower than the 'double' mode. You need to use METAPOST newer than 2.00 to use this mode without issues.

[88] It's high enough to be reached.

- page size ▷

  PDF map or atlas: $\approx 5 \times 5$ m (pdfTEX limit)

  SVG map: limits depend on the viewing application

- scraps count ▷

  METAPOST in the scaled mode: $4(scraps + sections) < 4000$

  METAPOST in the 'double' mode: practically no limit

  TEX limit in registers mode:[89] $2 \times pages + images + patterns + 6(scraps + sections) < 32\,500$ when using pdfTEX (or approximately 65 000 when using LuaTEX[90])

  TEX limit in macro mode:[91] limited only by memory available to TEX

- TeX memory ▷

  pdfTEX: static memory limits are set in the `texmf.cnf` configuration file in your TEX distribution; note that the total memory available is quite limited, regardless of the RAM available

  LuaTEX: the memory is dynamically allocated

Example data

Following simple example illustrates basic usage of Therion commands:

```
encoding  utf-8

survey main -title "Test cave"

  survey first
    centreline
      units compass grad
      data normal from to compass clino length
                  1    2  100     -5    10
    endcentreline
  endsurvey

  survey second -declination [3 deg]
    centreline
      calibrate length 0 0.96
```

[89] This is the default approach, in which the count registers are used to store references to the graphic objects.

[90] To use LuaTEX, set `pdftex-path "<path-to-luatex>"` in the initialization file.

[91] Instead of using count registers, each reference is stored in a separate macro. This mode is activated by setting `tex-refs-registers off` in the initialization file.

```
    data normal from to compass length clino
                1    2  0       10     +10
  endcentreline
endsurvey

centreline
  equate 2@first 1@second
endcentreline

# scraps are usually in separate *.th2 files
scrap s1 -author 2004 "Therion team"

  point 763 746 station -name 2@second
  point 702 430 station -name 2@first
  point 352 469 station -name 1@first
  point 675 585 air-draught -orientation 240 -scale large

  line wall -close on
    287 475
    281 354 687 331 755 367
    981 486 846 879 683 739
    476 561 293 611 287 475
  endline

endscrap

map m1 -title "Test map"
  s1
endmap

endsurvey
```

Corresponding configuration file could be:

```
encoding  utf-8
source test

layout l1
  scale 1 100
  layers off
endlayout

select m1@main
```

```
export model -fmt survex
export map -layout l1
```

If you save data file as 'test.th' and configuration file as 'thconfig' you may process them with Therion.

History

- **1999**

  Oct: first concrete ideas

  Nov: start of programming (Perl scripts and METAPOST macros)

  Dec 27: Therion compiles simple map in PostScript format for the first time (32 kB of Perl and 7 kB of METAPOST and TEX source code). The map warping model was substantially different from the current one (positions of features were relative to a particular survey shot, not to positions of all stations in a scrap). This version already included some interesting features such as *transformation functions* which allowed user specification of the input format for survey data, or splitting large maps to multiple sheets.

  Dec 30: the first web page (with data examples but without source code)

- **2000**

  Jan: xthedit (Tcl/Tk), a graphical front-end for Therion

  Feb 18: start of reprogramming (Perl)

  Apr 1: the first hyperlinked PDF cave map / atlas

  Aug: experiments with PDF, pdfTEX and METAPOST

- **2001**

  Nov: start of reimplementation from scratch: Therion (C++ with some Perl scripts inherited from the previous version); notion of a scrap; interactive 2D map editor ThEdit as a replacement of xthedit (Delphi)

  Dec: ThEdit exports simple map for the first time

- **2002**

  Mar: Therion 0.1 — Therion is able to process survey data (centreline) of the Cave of Dead Bats. XTherion, text editor designed for Therion (Tcl/Tk).

  Jul 27: Therion 0.2 — Therion compiles simple map (consisting of two scraps) for the first time (800 kB of source code)

  Aug: XTherion extended to 2D map editor (as a replacement of ThEdit)

  Sep: Therion compiles first real and complex map of a cave. XTherion extended to compiler.

- **2003**

  Mar: the first version of The Therion Book finished

  Apr: Therion included in Debian GNU/Linux

  Jun: all Perl scripts rewritten in C++, Therion is one executable program now (although using Survex and TEX)

- **2004**

  Mar: Therion 0.3 — Therion exports 3D model created from 2D maps. Loop closure algorithm included into Therion.

- **2006**

  Oct: Therion 0.4 — New 3D viewer (Loch).

- **2007**

  Feb: Therion 0.5 — Support for bitmap sketches morphing.

- **2016**

  Dec: GitHub repository.

- **2021**

  Jul: Therion 6.0.0 — Improved graphics (smooth colour transitions, coulour management).

- **2025**

  May 8: experimental support for 3D meshes.

Future

Although Therion is already used for map production, there are a lot of new features to be implemented:

General

- loop closure information in SQL

2D maps

- complete the predefined symbol sets
- generate registers for atlas
- use MPlib instead of METAPOST

3D models

- improve passage walls modeling

XTherion

- improve 2D editing capabilities

Loch

- colour schemes
- survey tree for selecting sub-surveys to display
- spatial filtering (e.g. clipping by planes)
- support for multiple surfaces

Labyrinth

- completely new GUI in the far future (see https://labyrinth.speleo.sk)

There is no royal road to geometry.
Μὴ εἶναι βασιλικὴν ἀτραπὸν ἐπὶ γεωμετρίαν.

—Euclid, 3rd century BC

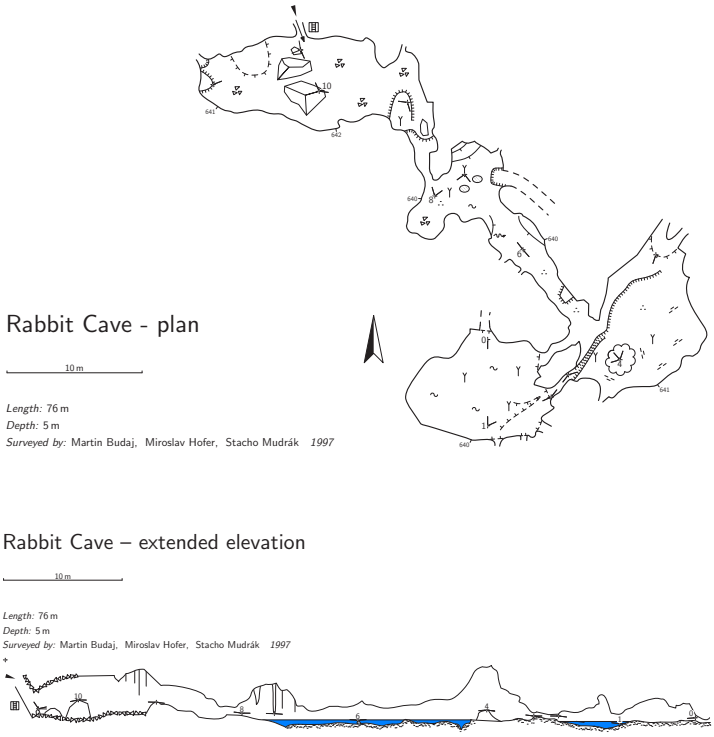
There is no royal road, but there is a road.

—Frederick P. Brooks, 1987

Case studies

This chapter is automatically generated from the samples directory in the Therion source code.

Drawing maps in therion



Listing caves

When surveying caves in some large area, you often need to generate an up to date list of all caves in that area. In therion dataset, "cave" can be defined in several ways, depending on the level of cave survey.

If you have just surveyed entrance station (e.g. using GPS or standard surveying methods), all you need to do is to assign `entrance` flag using

```
station cave-1 "Unsurveyed cave" entrance
```

If there is some explored passage behind this entrance, but you did not survey it properly, you should add also `continuation` flag and `explored` flag followed by explored length. Explored but unsurveyed passage lengths are also part of summary statistics.

```
station cave-2 "Unsurveyed but explored" entrance \
        continuation explored 80m
```

If you have properly surveyed some cave, its data are usually in some `survey` structure. Survey becomes a cave, if it has main entrance station specified using `-entrance` option. Example:

```
survey small_cave -title "Tunnel cave" -entrance 0
```

If survey contains survey data, but it has no main entrance specified, it is not considered as a cave, just as data organization unit and thus not present in the cave list. Only surveys containing some caves (i.e. entrance stations or surveys with main entrance defined) are present in the final cave list table.

After caves in your data have been properly defined, you can generate table using

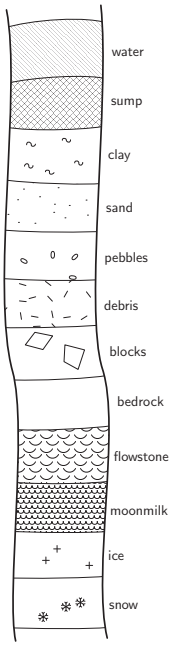
```
export cave-list -o caves.html
```

Resulting table looks like this

Title	**Length**	**Depth**	**Explored**	**Altitude**
Tunnel cave	25	7		1244.000
Unsurveyed cave				1234.000
Unsurveyed but explored			80	1256.000

Area symbols

Therion area types



Sketch morphing

Original survey sketch:

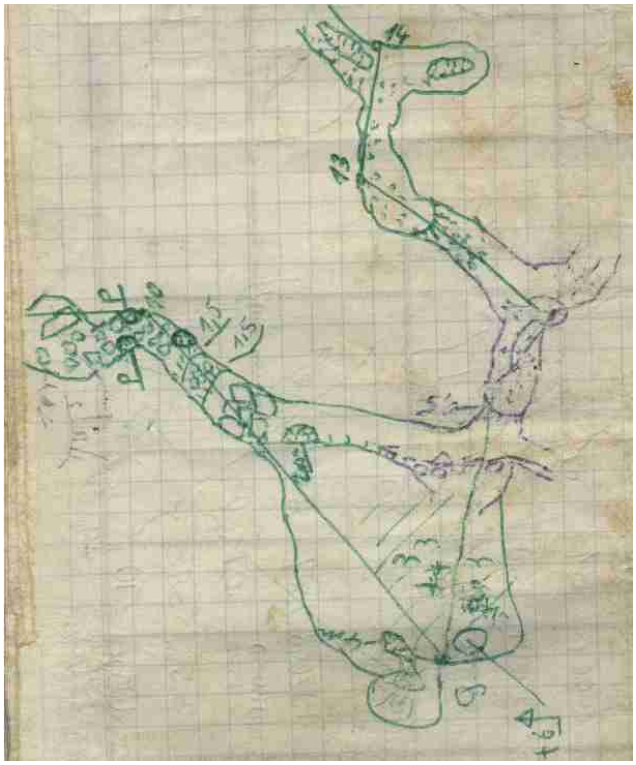
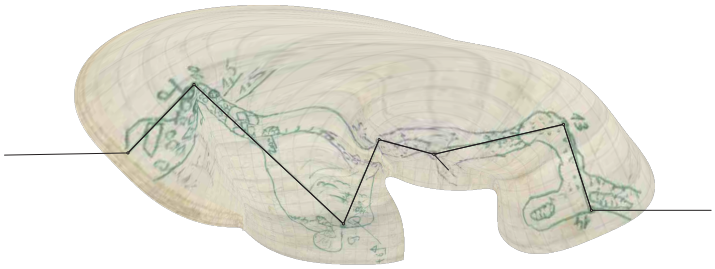
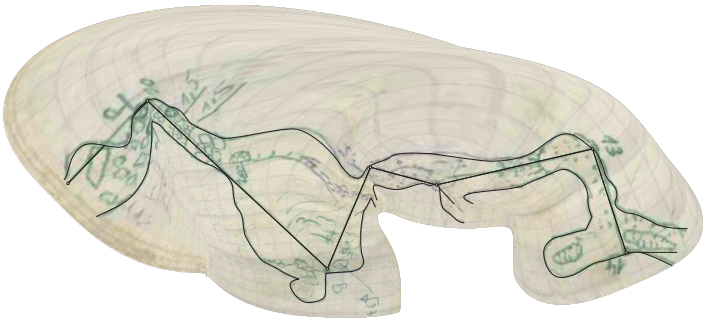


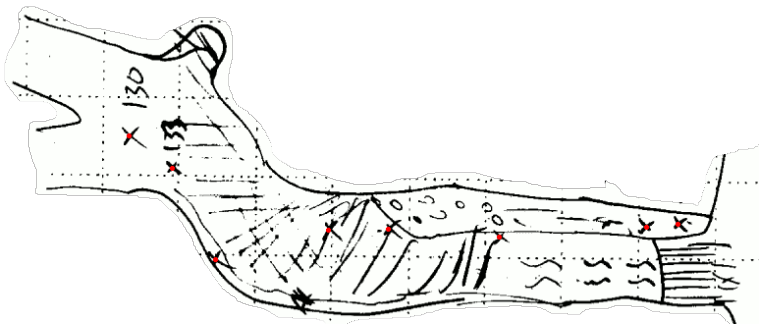
Image after morphing:



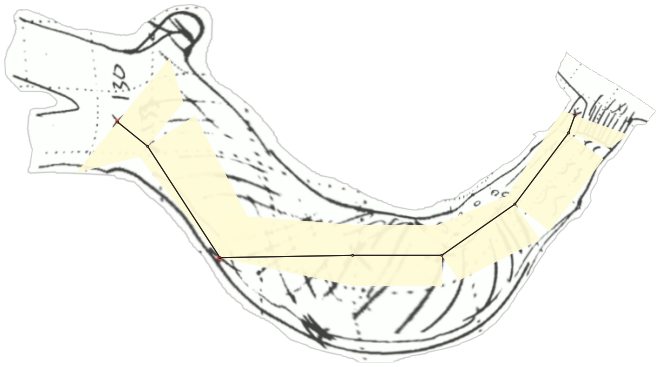
Walls after morphing:



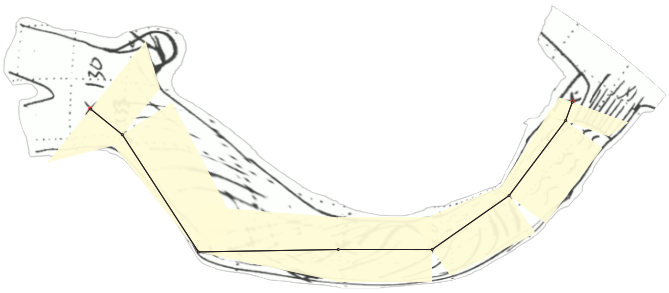
Another survey sketch:



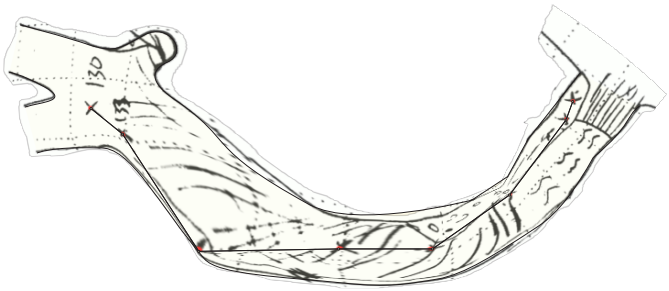
This image after morphing:



Morphed sketch after *extra* points insertion:

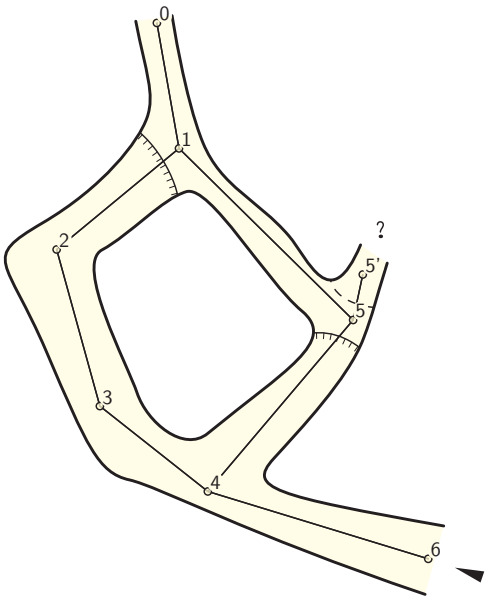


And the morphed walls:

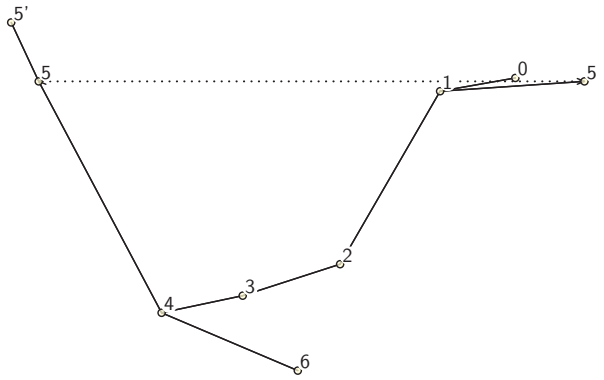


Extended elevation control

Assume following situation (in plan view), when there is a loop in centerline between stations 1 and 4, small chimney near station 5 and entrance to the cave is at station 6.



By default, centerline extended elevation of this situation looks like this:



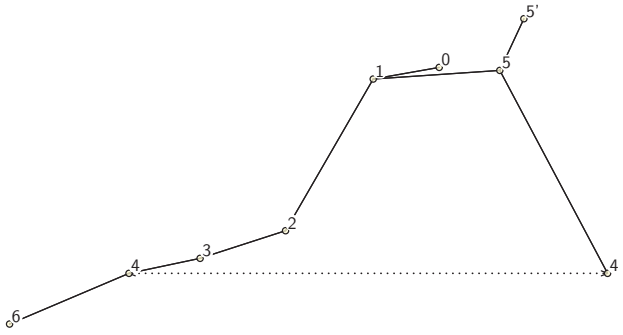
This is obviously not what we would like to get.

To control process of extended elevation, there is a special `extend` option in centerline command.

First of all, we would like to start our extended elevation at station 6 (where the entrance is). This can be done by specifying

```
extend start 6
```

in the centerline. Now it looks better,



but there is still a problem with a branch containing station 5.

If we would like to start this branch from station 4, we need to forbid therion to extend station 5 from 1. This can be done by specifying

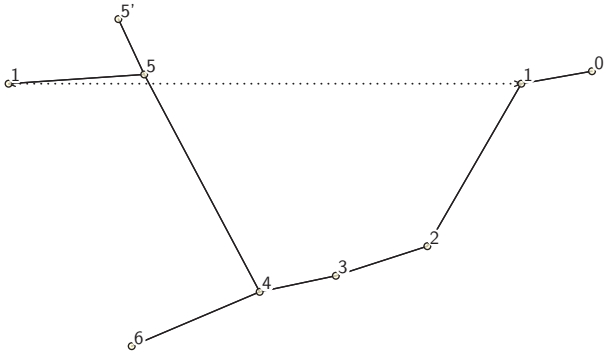
```
extend ignore 1 5
```

This means, that shot from 1 to 5 will be ignored, when extended elevation is generated.

If we would like to extend branch starting with station 5 to the left, we need to specify also

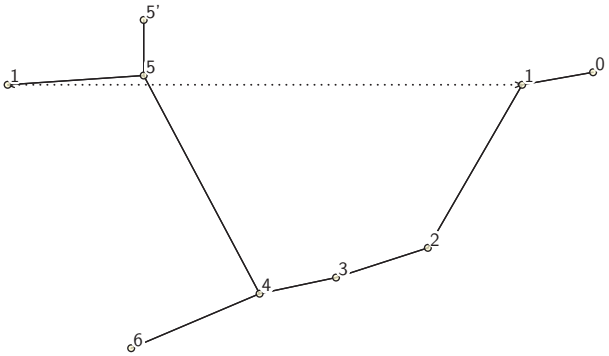
```
extend left 5
```

This will extend all stations from station 5 to the left.



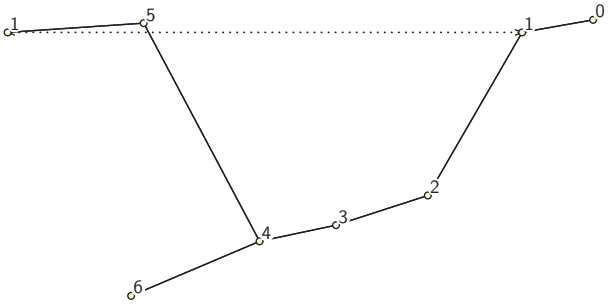
As we have mentioned before, there is a small chimney above station 5. In this case, it is much more natural to draw this shot as vertical (because it is a chimney). To specify this, use

```
extend vertical 5 5'
```



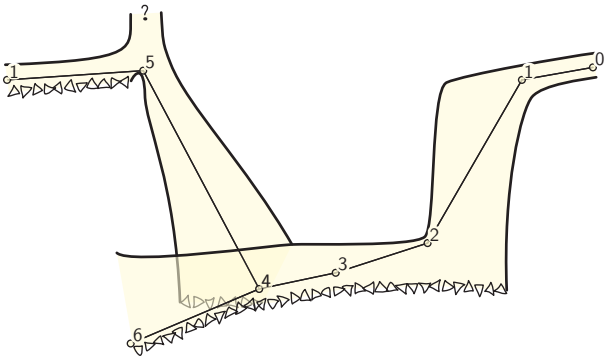
Or you can completely hide this leg from extended centerline using:

```
extend hide 5 5'
```



Stations in extended elevation scraps

Even if the station 1 is present more then one time in the map, therion automatically detects correct position of this station in each scrap and they are usually drawn correctly.



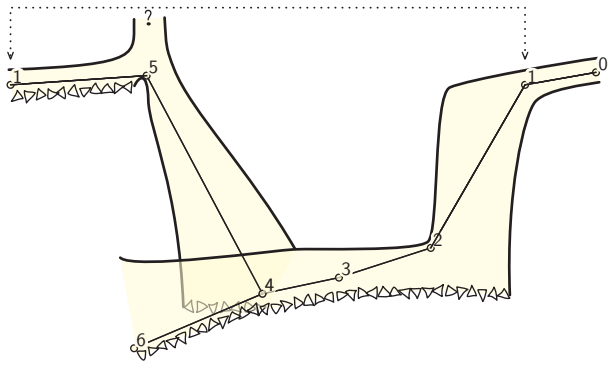
The only thing that is missing is connection line between stations 1 in these two branches. Therion does not automatically generate these lines because their shape usually depends on particilar map.

To draw such a line all you need to do is to create a simple scrap with this line. Here is an example:

```
scrap sc -proj extended
  point 0 0 station -name 1 -from 5 -visibility off
  point 100 0 station -name 1 -from 2 -visibility off
  line map-connection
      0   5
      0  15
    100  15
```

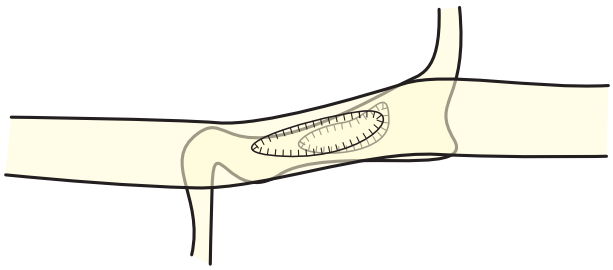
```
    100   5
  endline
endscrap
```

As you can see, even there are two stations with same name, they are distinguished by `-from` option, which specifies previous station in extended elevation. Using this scrap you receive the final map:



Displaying overlaying maps in offset

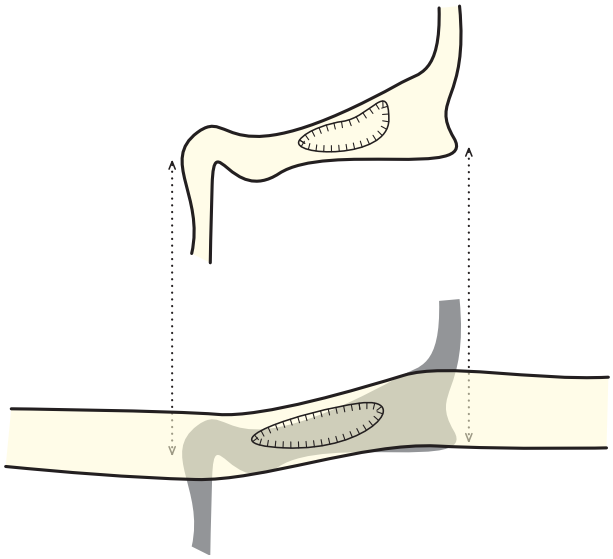
Assume that there are two maps *m1* and *m2* (*m1* is above *m2*). By default therion displays *m1* overlying *m2*.



To display map *m2* in offset you need to create a map containing *m2* and specify offset and preview type for *m2* in this map. Example:

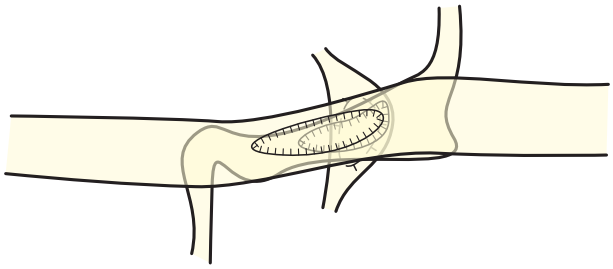
```
map m12
  m1
  m2 [0 8 m] below
endmap
```

After selecting the *m12* in configuration file you receive:



Joining arrows are created from *map-connection* points specified in scraps that are moved. Otherwise these points are ignored.

If the situation is more complicated, e.g. there is *m3* below *m2*

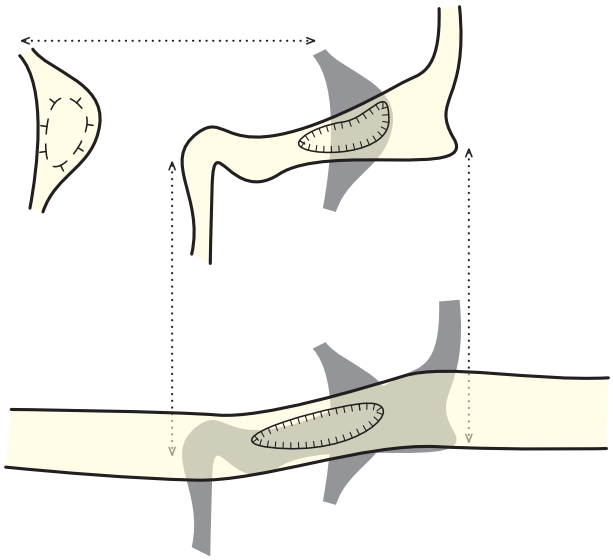


you need to create more complicated map structure to handle this.

```
map m23
  m2
  m3 [-8 0 m] below
endmap

map m123
  m1
  m23 [0 8 m] below
endmap
```

Then *m123* looks like this:



Importing survex .3d files

Therion surveys nearly correspond to station prefixes defined by `*begin`/`*end` pairs in survex source files. Usually there does not exist one-to-one mapping between them. So if you want to keep your centerline in survex files you need to solve the problem how to match survex and therion data structures. This sample shows three different ways, how to deal with station prefixes when importing survex .3d files to therion.

Using surveys specified in .th files

If you will import .3d file with `-surveys use` switch, then therion tries to find the match between *survex prefix* and *therion survey name*. If this match is found, stations are inserted into survey found. Otherwise prefix is left with station names. Example code:

```
import use.3d -surveys use
input use-out.th2
survey use
  input use-in.th2
endsurvey
```

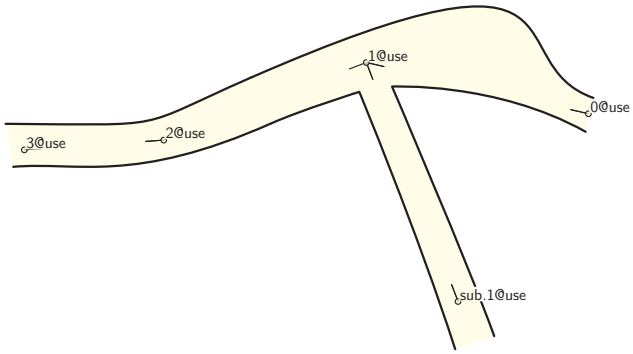
In this case you should take care where you input your .th2 files containing scraps. In *use-out.th2* file you should refer to station names using:

```
point 165.744 176.58 station -name 2@use
```

but in the *use-in.th* file using

```
point 321.454 236.22 station -name 1
```

Map with station names in this case looks like this:



Station names are before '@' symbol, survey names follow this symbol.

Creating non-existing surveys

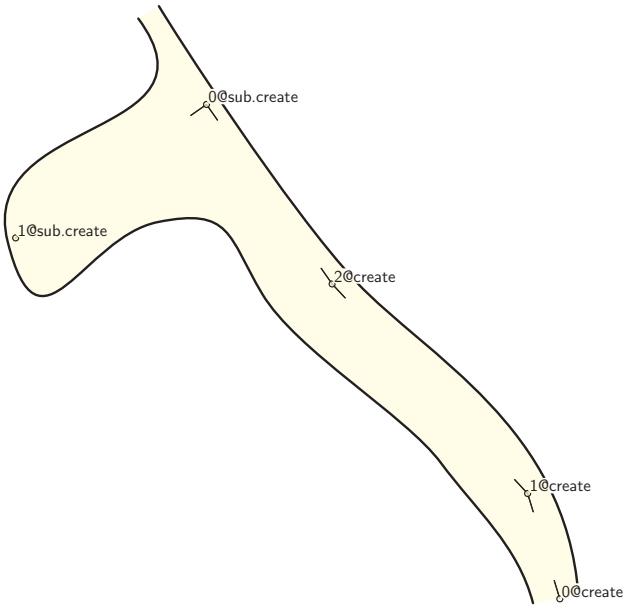
If you import your .3d file using

```
import create.3d -surveys create
```

all survex prefixes are taken into account and if surveys with corresponding names do not exist, they are created. In this case, if you do

```
input create.th2
```

right after `import` command, you refer to stations in .th2 file with names shown on the map.

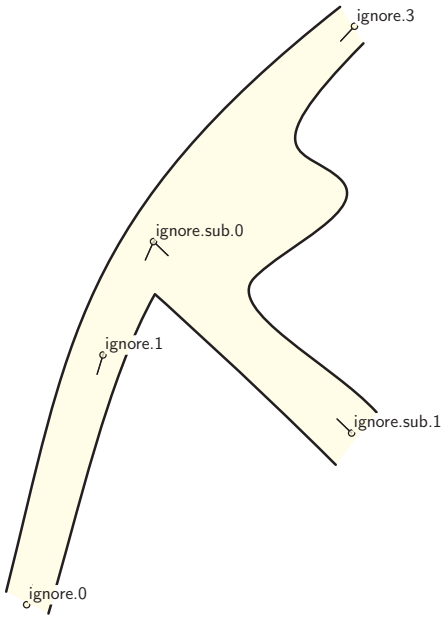


Ignoring station prefixes

Last possibility is to import your .3d file using

```
import ignore.3d -surveys ignore
```

Also in this case, station references in corresponding .th2 file are same as station names on the next map.



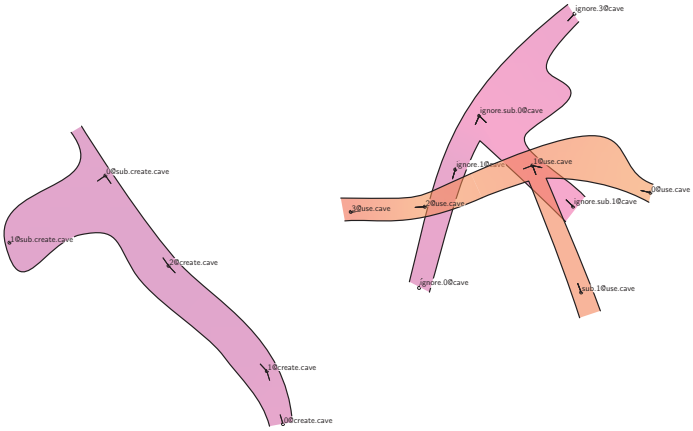
Note that no surveys are created in this case. Station names are taken from .3d files without change.

Managing large projects

Assume a situation when you want to join these three small maps within single large project. Assume that coordinates of cave entrances are specified in the top-level cave.3d file. If your joining code will be

```
import cave.3d -surveys use
survey cave
  input use/use.th
  input create/create.th
  input ignore/ignore.th
endsurvey
```

not all stations are replaced correctly. The “created” series is placed wrong on the map.

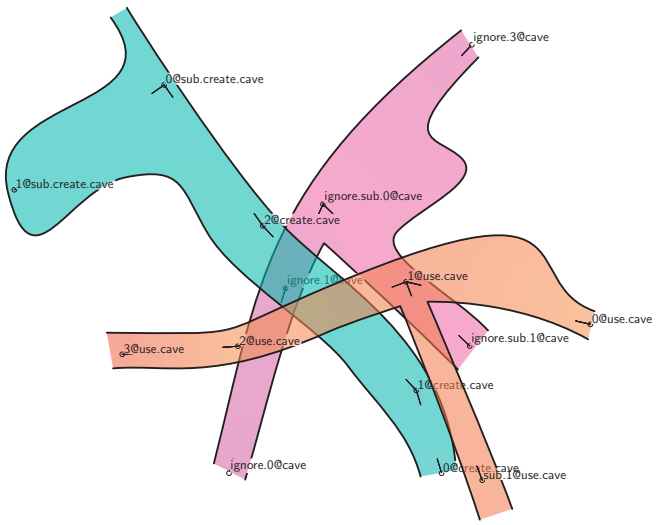


This is because in top level import command, survey *create* is imported from file `create/create.3d` with wrong coordinates, and we import top-level .3d file with `-surveys use` switch. This means, cave.create.1 will be imported here as create.1@cave and not 1@create.cave.

To solve this problem, we need to re-import stations from top-level cave.3d file once more with `-surveys create` switch.

```
import cave.3d -surveys use
import cave.3d -surveys create
survey cave
  input use/use.th
  input create/create.th
  input ignore/ignore.th
endsurvey
```

After this additional `import` final looks as expected.



Conclusion

Even if there are several possibilities how to map survex prefix structure to therion survey structure, the most clean solution is to create survey structure to desired depth in .th files using empty `survey`/`endsurvey` pairs and always use `-surveys use` switch when importing .3d files.

Question marks handling

Possible continuations are treated special way in therion in both centerline and map files. You may associate text description, explored length (behind this continuation) or any other atribute to it (like Code if you have your own coding standard for continuations).

Question marks in centerline

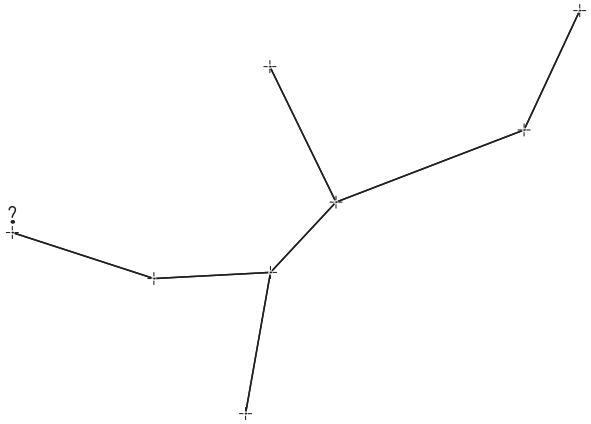
In the centerline object, you may add a special flags to the station, where continuation is possible. Just use following syntax

```
station 5 "pit" continuation attr Code V explored 20m
```

When you export map and use

```
symbol-show point flag:continuation
```

in your layout, station with continuation flag specified is marked by question mark (continuation symbol is shown above the station).



You may redefine continuation symbol to show also continuation description (stored in `_text` attribute) using following layout code

```
code metapost
  def p_continuation(expr pos,theta,sc,al) =

    % draw default continuation symbol
    p_continuation_UIS(pos,theta,sc,al);

    % if text attribute is set
    if known(ATTR__text) and picture(ATTR__text):

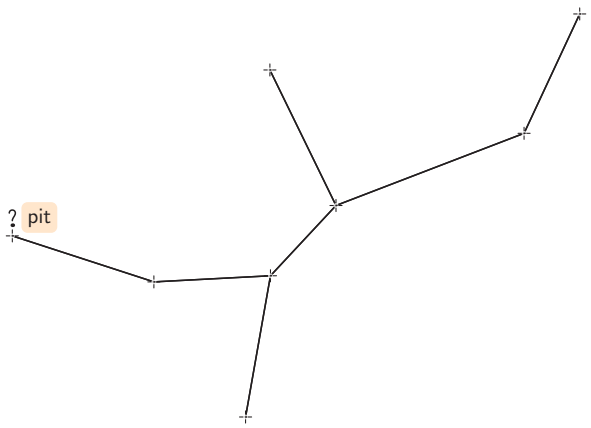
      % set labeling color to light orange
      push_label_fill_color(1.0, 0.9, 0.8);

      % draw filled label with text next to ?
      p_label.urt(ATTR__text,(.5u,-.25u) transformed T,0.0,8);

      % restore original labeling color
      pop_label_fill_color;

    fi;
  enddef;
endcode
```

Then also continuation description is displayed in the map.

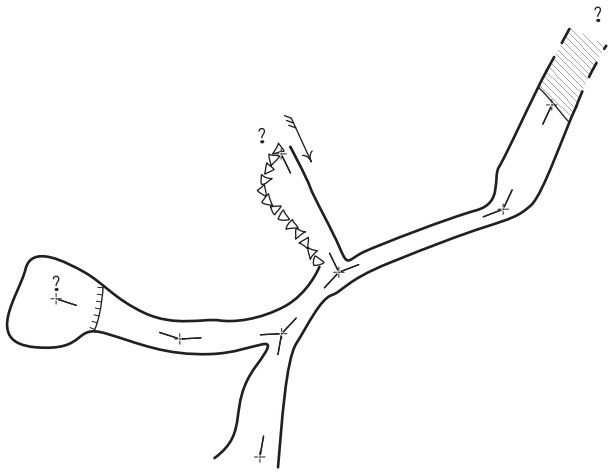


Question marks in maps

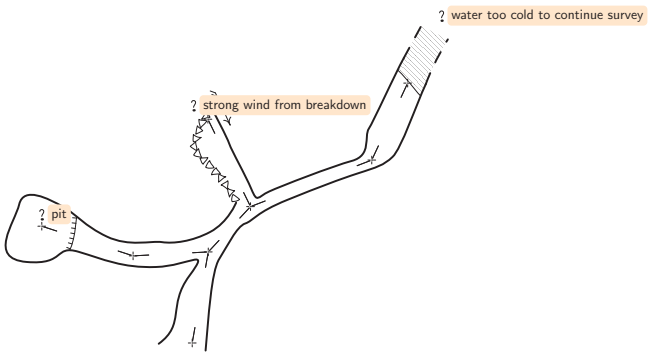
In the scrap (.th2 file), you can use `-text`, `-code` and `-explored` options to continuation symbol.

```
point 796.0 676.0 continuation -attr Code A -explored 50m \
    -text "water too cold to continue survey"
```

In the map, also the question marks only are displayed by default.



When you redefine continuation symbol as mentioned above, you can show also continuation codes and descriptions.



Exporting question mark lists

You may also export a list of all continuations from your project project using

```
export continuation-list -o questions.html
```

File questions.html then contains following list:

Comment	Explored	Survey	Station	Code
water too cold to continue survey	50.0	Sample cave	7	A
strong wind from breakdown		Sample cave	3	B
pit	20.0	Sample cave	5	V

Using user defined symbol types

If therion does not offer any appropriate symbol for the feature you want to draw to the map, you can use user defined symbol type (type `u`, valid for point, line and area objects).

The syntax is very simple. Assume you want to create "bat" point, line and area. You just use `u:bat` as a type (bat is in fact subtype) of u type. So your code will be like this:

```
point 555.0 480.0 u:bat
```

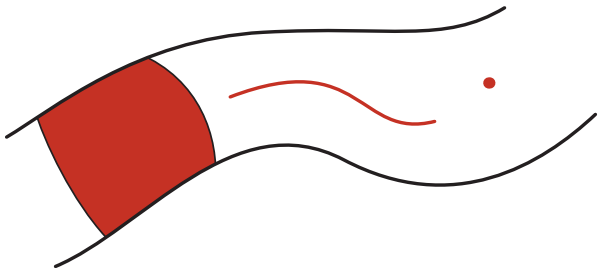
or

```
line u:bat
```

or

```
area u:bat
```

When you export map without defining the symbols in METAPOST, user defined symbols are highlighted in red without any graphical representation.



To display them correctly you need to define symbols for them in METAPOST languge the same way as ordinary symbols are usually redefined.

Firstly point symbol. In the

```
code metapost
```

section of your layout define point u:bat symbol like this

```
def p_u_bat(expr pos, theta, sc, al) =
  T := identity shifted pos;
  thfill (bat_path scaled 2.0);
enddef;
```

similarly the line u:bat symbol

```
def l_u_bat(expr P) =
  T:=identity;
  cas := 0;
  dlzka := arclength P;
  mojkrok:=adjust_step(dlzka, 1.0u);
  pickup PenD;
  forever:
    t := arctime cas of P;
    thfill bat_path scaled 0.5 shifted (point t of P);
    cas := cas + mojkrok;
    exitif cas > dlzka + (mojkrok / 3); % for rounding errors
  endfor;
enddef;
```

and finally the area u:bat symbol (pattern in this case)

```
% bat pattern
beginpattern(pattern_bat);
    fill bat_path;
endpattern;

% bat area symbol
def a_u_bat (expr Path) =
  T:=identity;
  thclean Path;
  thfill Path withpattern pattern_bat;
enddef;
```

These symbols will be included also in the legend. To change the way how they are drawn there just define appropriate macro. Its name should be symbol macro name with `_legend` suffix.

```
def l_u_bat_legend =
  l_u_bat(((.2,.2) -- (.8,.8)) inscale)
enddef;
```

Finally, add description of your new symbols that will be shown in the legend using `text` command anywhere in the configuration file.

```
text en "point u:bat" "bat"
text en "line u:bat" "bat path"
text en "area u:bat" "lot of bats"
```

After all these definitions you receive bat point, line and area symbols with proper graphical representation and legend boxes.

