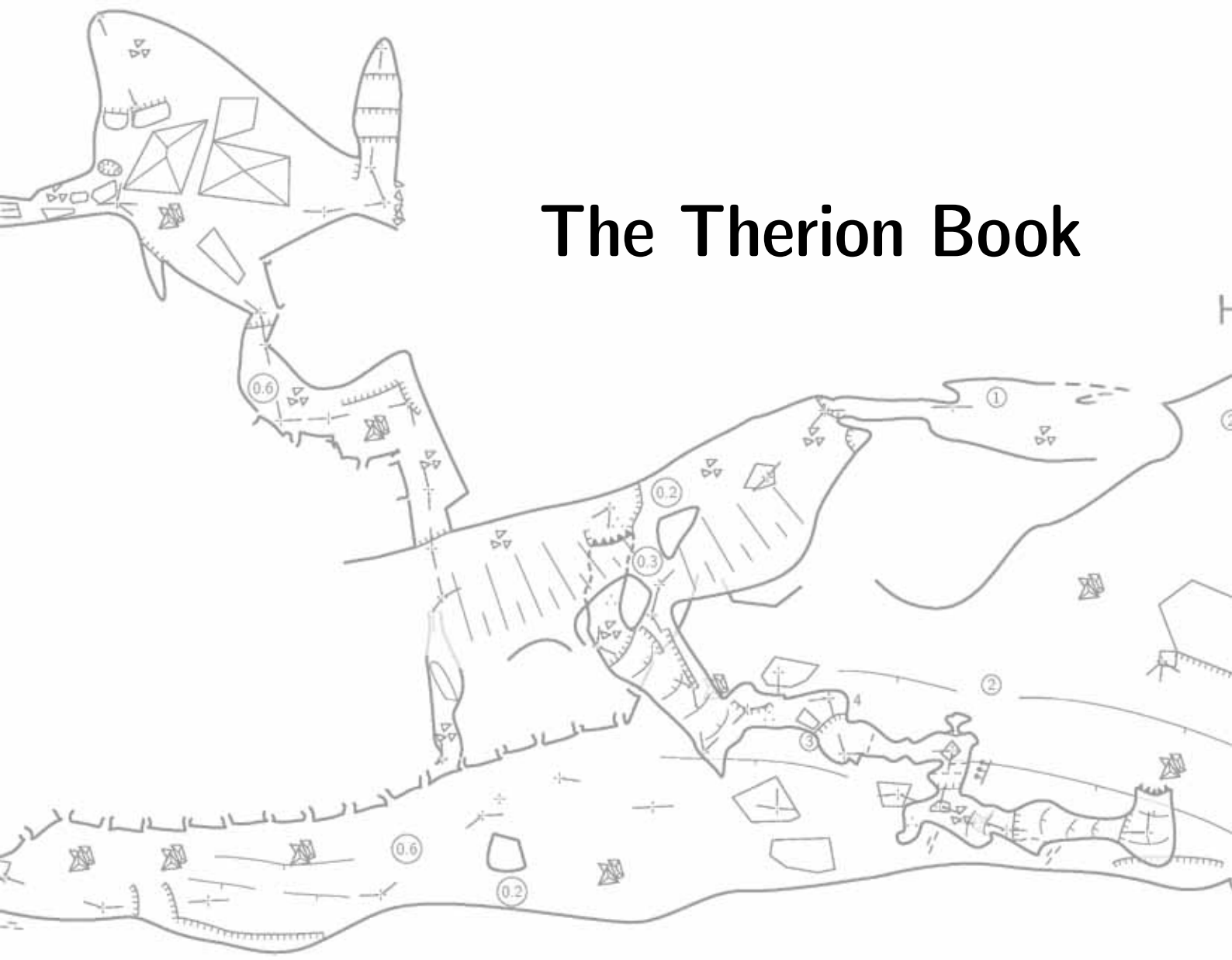
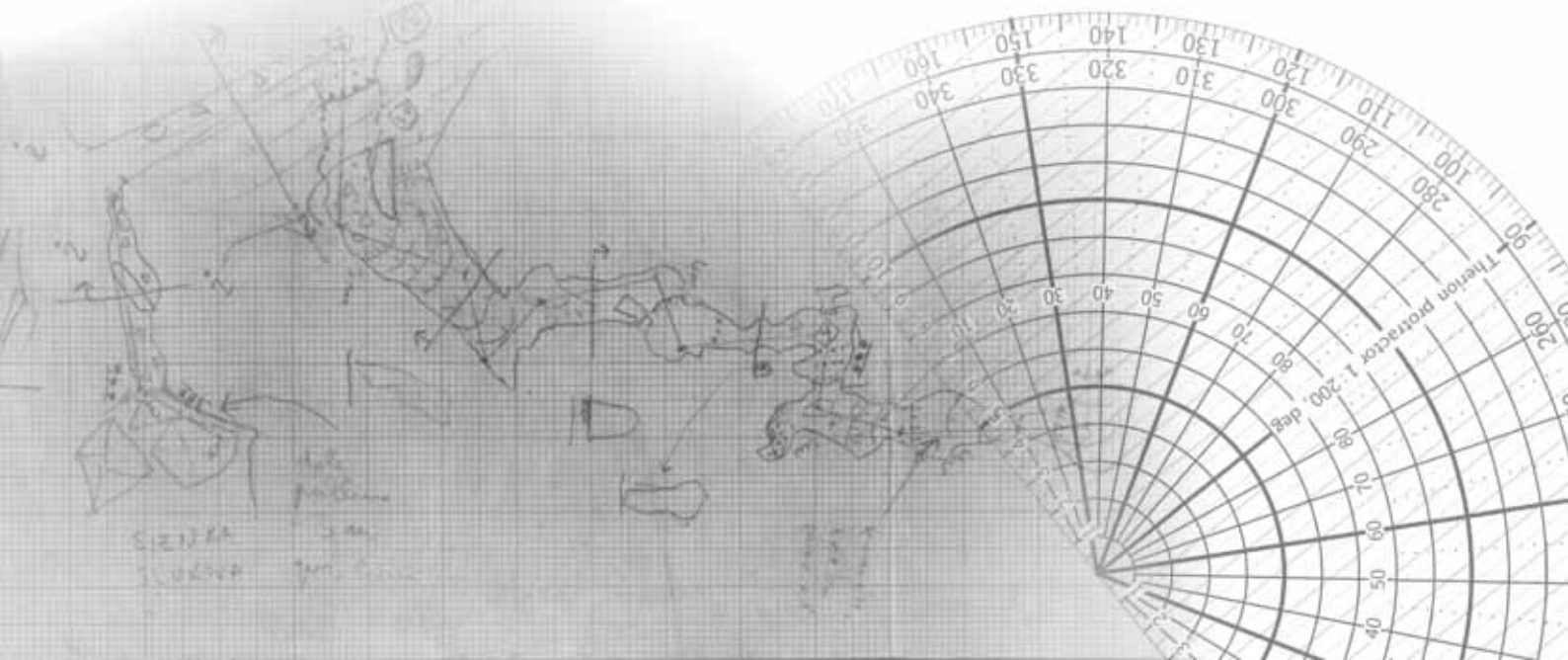
The Therion Book



Stacho Mudrák

Martin Budaj



This book describes Therion 5.4.1 (2017-04-18).

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

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

Переводы (%):

Language	*XTherion*	*Map header*	*Loch*	*Translated by*
bg	86	87	100	Alexander Yanev, Ivo Tachev, Vladimir Georgiev
cz	81	88	–	Ladislav Blažek
de	82	92	–	Roger Schuster, Georg Pacher, Benedikt Hallinger
el	85	87	–	Stelios Zacharias
en[_GB|_US]	75	93	100	Stacho Mudrák, Olly Betts
es	75	83	–	Roman Muñoz
fr	–	87	–	Eric Madelaine, Gilbert Fernandes
it	86	92	–	Marco Corvi
mi	–	91	–	Kyle Davis, Bruce Mutton
pl	–	90	–	Krzysztof Dudziński
pt[_BR|_PT]	–	83	–	Toni Cavalheiro, Rodrigo Severo
ru	81	86	–	Vasily V. Suhachev, Andrey Kozhenkov
sk	85	93	96	Stacho Mudrák
sq	85	87	–	Fatos Katallozi
zh	86	91	–	Zhang Yuan Hai, Duncan Collis

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

Содержание

LET NO ONE IGNORANT OF GEOMETRY ENTER HERE
ΑΓΕΩΜΕΤΡΗΤΟΣ ΜΗΔΕΙΣ ΕΙΣΙΤΩ
—alleged inscription over the entrance
of Plato's Academy, 4th century BC

Введение

Therion это программа для создания карт пещер. Ее цель состоит в том, чтобы помочь:

- архивировать данные съемки на компьютере в форме, максимально приближенной к оригинальным записям и зарисовкам, и обрабатывать их удобным и эффективным способом;
- рисовать красивые современные планы и разрезы;
- создавать реалистичные 3D-модели пещер.

Therion работает в операционных системах Unix, Linux, MacOS X и Win32. Исходный код и установщик для Windows доступны на веб-странице (https://therion.speleo.sk).

Therion распространяется под лицензией GNU General Public License.

Почему Therion?

В 1990-е мы активно занимались спелеологией и созданием карт. Имелось несколько компьютерных программ, которые строили нитку хода после закрытия колец и разброса ошибки. Это было большим подспорьев в работе, особенно работая над крупными и сложными пещерными системами. Мы использовали вывод одного из них (TJIKPR) в качестве фонового слоя со станциями для ручного рисования карт. После окончания огромного 166-страничного Атласа пещер мертвых летучих мышей в начале 1997 года у нас вскоре возникла проблема: мы нашли новые ходы, соединения между известными ходами. После обработки данных в TJIKPR, новые кольца повлияли на положение старых станций, большинство станций уже имели иную позицию из-за разброса невязки колец. Таким образом мы могли бы перерисовывать весь Атлас снова, или принять, что местоположение некоторых новых мест на карте было изображено не точно (в случае колец с длиной около 1 км ошибки достигали 10 м) и пытаться подогнать новые хода к старым съемкам.

Эти проблемы оставались, когда мы пытались рисовать карты с помощью некоторых программ CAD в 1998 и 1999 годах. Всегда было трудно добавить новые исследования без адаптации старых к новым рассчитанным позициям

станций во всей пещере. Мы не нашли ни одной программы, которая могла бы нарисовать современную сложную карту (т.е. не только нитку хода с LRUD), в которых старые части съемки изменялись в соответствии с новыми расчетными координатами станций.

В 1999 году мы начали думать о создании собственной программы для рисования карт. Мы знали о программах, которые идеально подходили для конкретных подзадач. Это был METAPOST— язык программирования высокого уровня для описания векторной графики, Survex — отличная программа для обработки нитки хода, и TEX — для верстки результатов. Нужно было только сложить их вместе. В рождество 1999 года мы уже имели первую версию Therion'а. Она состояла примерно из 32 КБ Perl скриптов и METAPOST макросов, но программа показала, что наши идеи были осуществимы.

В перирод 2000–2001 годов мы искали оптимальный формат входных данных, язык программирования, концепцию интерактивного редактирования карт и внутренних алгоритмов с помощью Martin Sluka (Прага) и Martin Heller (Zurich). В 2002 году мы представили первую версию Therion'а, которая отвечала нашим требованиям.

Особенности

Therion — приложение для командной строки. Он обрабатывает входные файлы в текстовом формате, в том числе 2D-карты, и создает файлы с 2D-картами или 3D-моделью в качестве вывода.

Синтаксис входных файлов подробно описан в последующих главах. Вы можете создавать эти файлы в любом текстовом редакторе, например *ed* или *vi*. Файлы содержат инструкции для Therion, такие как:

```
point 1303 1004 pillar
```

где point — комманда для символа точки, за которым следуют его координаты и специфический тип символа.

Ручное редактирование таких файлов непросто, особенно когда вы рисуете карты и вам нужно думать о пространственных (декартовых координатах). Поэтому существует специальный графический интерфейс для Therion, называемый XTherion. XTherion работает как расширенный текстовый редактор, редактор карт (где карты рисуются в полностью интерактивном режиме) и компилятор (который запускает Therion).

Это может выглядеть довольно сложно, но этот подход имеет много преимуществ:

- Строгое разделение данных и визуализации. В файлах данных указывается только то, что есть, а не то, на что это похоже. Визуальное

представление добавляется METAPOST на более поздних этапах обработки данных (это очень похоже на представление XML-данных).

Это позволяет изменять символы карты, используемые без изменения входных данных, или объединить большие карты, созданные разными людьми в разных стилях, в одну карту с едиными символами.

2D-карты адаптированы для конкретного масштаба (уровень абстракции, нелинейное масштабирование символов и текстов).

- Все данные привязываются к положениям пикетов съемки. Если координаты станций съемки изменяются в процессе закрытия колец, то все связанные данные перемещаются соответственно, поэтому карта всегда актуальна.
- Therion не зависит от конкретной операционной системы, кодировки символов или редактора входных файлов; входные файлы останутся читабельными для человека.
- Можно добавить новые форматы вывода.
- 3D-модель создается из 2D-карт, чтобы получить реалистичную трехмерную модель не вводя слишком много данных.
- Хотя поддержка WYSIWYG ограничена, вы всегда можете получить то, что хотите.

Требования к ПО

“Программа должна делать одну задачу, и должна делать это хорошо” (Кен Томпсон). Поэтому мы используем несколько внешних программ, которые обрабатывают и визуализируют данные. Therion в связке с другими программами может выполнить свою задачу намного лучше.

Therion'у необходимо:

- TEX дистрибутив. Необходимо только в том случае, если вы хотите создавать 2D-карты в формате PDF или SVG.
- Tcl/Tk с *BWidget* и опциональным расширением *tkImg*. Это требуется только для XTherion.
- LCDF Typetools, если вы хотите использовать легкую настройку для пользовательских шрифтов в PDF-картах.
- Утилиты *convert* и *identify* из дистрибутива ImageMagick, если вы хотите использовать деформирование эскизов.
- *ghostscript*, если вы хотите создавать калиброванные изображения с геопривязанными PDF-картами.

Установщик для Windows включает все необходимые пакеты, за исключением ghostscript. Прочтите *Приложение*, если вы хотите скомпилировать Therion самостоятельно.

Для отображения карт и моделей вы можете использовать любую из следующих программ:

- любой просмотрщик PDF или SVG для просмотра 2D-карт;
- любые GIS поддерживающие DXF или *shapefile* форматы для анализа карт;
- соответствующий 3D просмотрщик для моделей, экспортированных в формате отличном от стандартного;
- любой клиент базы данных SQL для обработки экспортированной базы данных.

Инсталяция

Установка из исходников (therion-5.*.tar.gz package):

Исходники — главный дистрибутив Therion. Его необходимо скомпилировать и установить в соответствии с инструкциями в *Приложении*.

Установка в Windows:

Запустите программу установки и следуйте инструкциям. Он устанавливает все необходимые материалы и создает ярлыки для XTherion и Therion Book.

Настройка среды

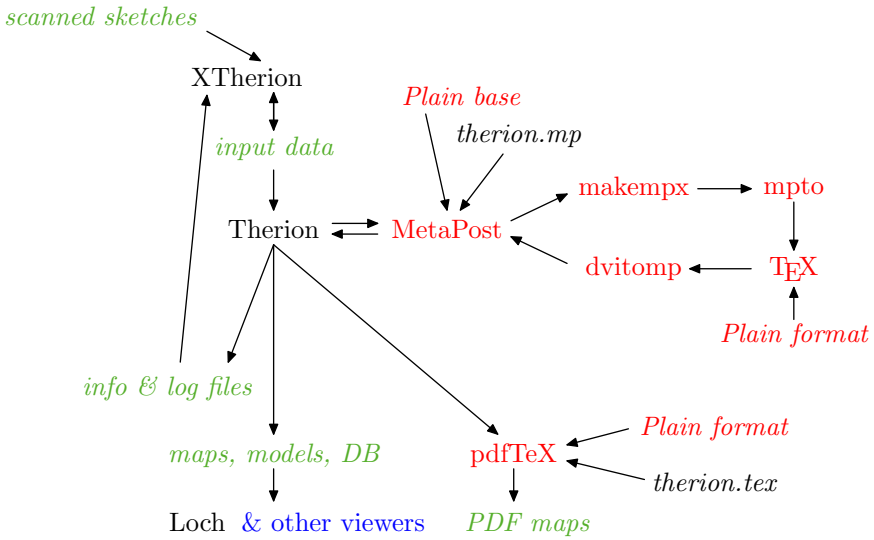
Therion считывает настройки из файла инициализации. Настройки по умолчанию должны работать отлично для пользователей использующих только латинские символы[1], стандартные TEX и METAPOST.

Если вы хотите использовать собственные шрифты для латинских или нелатинских символов в PDF-картах, отредактируйте файл инициализации. Инструкции о том, как это сделать, приведены в *Приложении*.

Как это работает?

Итак, теперь ясно, что нужно Therion'у, давайте посмотрим как он взаимодействует со всеми этими программами:

[1] На PDF-картах Therion отображает большинство акцентированных символов как сочетание акцента и базового символа. Некоторые неявные акценты могут быть опущены. Предустановленные буквы с акцентом включены для словацкого и чешского языков.



НЕ ПАНИКУЙТЕ! Когда ваша система настроена правильно, большинство из файлов скрыто от пользователя, и все необходимые программы автоматически запускаются Therion'ом.

Для работы с Therion'ом достаточно знать, что вам нужно создавать входные данные (лучше всего делать это в XTherion), запускать Therion и отображать выходные файлы (3D-модель, карта, лог-файл) в соответствующей программе.

Для тех, кто хочет больше узнать об этом, кратко изложим приведенную выше блок-схему. Названия программ отображены прямым шрифтом, а файлы данных выделены курсивом. Стрелки показывают поток данных между программами. Временные файлы данных не показаны. Значение цветов:

- черный — программы и макросы Therion'а (XTherion написан на Tcl/Tk, поэтому для него требуется этот интерпретатор);
- красный — TEX пакет;
- зеленый — входные файлы, созданные пользователем и выходные файлы, созданные Therion'ом.

Сам Therion выполняет главную задачу. Он считывает входные файлы, интерпретирует их, находит замкнутые кольца и раскидывает ошибки. Затем он преобразует все другие данные (например 2D-карты) в соответствии с позицией новых станций. Therion экспортирует данные для 2D-карт в формате METAPOST. METAPOST дает фактическую форму абстрактным символам карты в соответствии с определениями символов карты; он создает много файлов PostScript с небольшими фрагментами пещеры. Они считываются и преобразуются в PDF формат, который формирует входные данные для pdfTEX. PdfTEX собирает все фрагменты и создает PDF-файл карты пещеры.

Therion также экспортирует трехмерную модель (полную или нитку хода) в различных форматах.

Нитка хода может быть экспортирована для дальнейшей обработки в любую базу данных SQL.

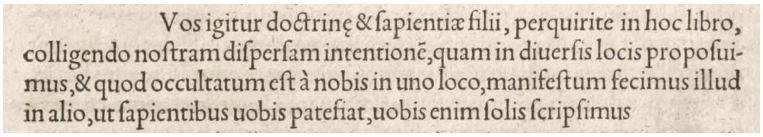
Первый запуск

После объяснения основных принципов работы Therion'а давайте попробуем его на примерах реальных данных.

- Скачайте примеры данных с сайта Therion'а и распакуйте их на жесткий диск.
- Запустите XTherion (под Unix и MacOS X введя в командной строке 'xtherion', под Windows ярлык в меню *Старт*). Откройте файл 'thconfig' из каталога примеров данных в окне 'therion компилятор'.
- Нажмите 'F9' или 'Компилировать' в меню для запуска Therion'а — вы получите несколько сообщений от Therion'а, METAPOST и TEX. PDF-карты и 3D-модель создаются в каталоге с данными.

Кроме того, вы можете открыть файлы данных съемки (*.th) в окне 'therion текстовый редактор' и файлы абрисов карт (*.th2) в окне 'therion редактор карт'. Наличие различных форматов данных может выглядеть запутанным по началу, но все они будут разъяснены в следующих главах.

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

Создание файлов данных

Основы

Входные файлы для Therion'а имеют текстовый формат. Существует несколько правил о том, как должен выглядеть такой файл:

- Есть два типа команд. Однострочные команды и многострочные команды.
- Однострочная команда завершается символом конца строки. Их синтаксис

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

  где *arg1 ... argN* являются обязательными аргументами, а пары *-option value* являются параметрами, которые вы можете свободно пропустить. Какие аргументы и опции доступны, зависит от конкретной команды. Примером может служить

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

  с тремя обязательными аргументами и одной дополнительной парой опция/значение. Иногда параметров нет или может быть несколько значений.
- Многострочные команды начинаются аналогично однострочным, но продолжаются на последующих строках до явного завершения команды. Эти строки могут содержать либо данные, либо параметры, которые применяются к последующим данным. Если строка данных начинается со слова, зарезервированного для опции, вам нужно вставить '!' перед ней. Синтаксис

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

  Опять же, для лучшей иллюстрации приведем пример:

```
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
```

Эта команда line имеет один обязательный аргумент, тип линии (коренная стена в данном случае), за которой следует одина опция. Следующие две строки содержат данные (координаты кривых Безье). Следующая строка ("smooth off") указывает параметр, который применяется к последующим данным (т.е. не для всей строки, в отличие от опции -id в первой строке), и последняя строка содержит еще несколько данных.

- Если значение параметра или аргумента содержит пробелы, вы должны заключить это значение в " " или [ ]. Если вы хотите поместить двойную кавычку " в текст в " " вам нужно вставить его дважды. Кавычки используются для строк; скобки для числовых значений и ключевых слов.
- Каждая строка, заканчивающаяся обратным слэшем (\), считается продолженной на следующей строке, как будто не было ни разрыва строки, ни зазора.
- Все, что следует за # и до конца строки, даже внутри команды, считается комментарием и игнорируется.

5.4 • Многострочные комментарии внутри comment ... endcomment блока разрешены в файлах данных и конфигурационных файлах.

Типы данных

Therion использует следующие типы данных:

- *keyword* ▷ последовательность A-Z, a-z, 0-9 и _-/ символов (не начинающиеся с '-').
- *ext_keyword* ▷ слово, которое также может содержать +*.,' символы, но не в первой позиции.
- *date* ▷ спецификация даты (или временного интервала) в формате YYYY.MM.DD@HH:MM:SS.SS - YYYY.MM.DD@HH:MM:SS.SS или '-' чтобы указать неопределенную дату.
- *person* ▷ имя и фамилия человека, разделенные пробельными символами. Используйте '/' чтобы отделить имя и фамилию, если есть несколько имен.

5.3 • *string* ▷ последовательность любых символов. Строки могут содержать специальный тег <lang:XX> для разделения переводов. В многоязычных строках только текст между <lang:XX> (где XX - это язык, выбранный в

файле инициализации или конфигурации) и следующим тегом <lang:YY> отображается на выходе. Если совпадение не найдено, все до появления тега <lang:ZZ> отображается.

- *units* ▷ поддерживаемые единицы длины: meter[s], centimeter[s], inch[es], feet[s], yard[s] (можно сокращать m, cm, in, ft, yd). Поддерживаемые угловые единицы: degree[s], minute[s] (можно сокращать deg, min), grad[s], mil[s], percent[age] (только для угла наклона). Значение градуса может быть введено в десятичной системе ($x.y$) или в специальной нотации для градусов, минут и секунд (deg[:min[:sec]]).

Системы координат

Therion поддерживает преобразования координат в геодезические системы координат. Вы можете указать опцию cs в объектах centreline, surface, import и layout и ввести XY в выбранной системе координат. Вы также можете указать вывод cs в конфигурационном файле.

Если вы не указали какой-либо cs в вашем наборе данных, то предполагается, что вы работаете в local системе координат, и никакие преобразования не выполняются. Если вы укажете cs в любом месте данных, то вы должны указать его для всех данных местоположения (fix, origin и layout и т. д.).

cs применяется ко всем последующим данным местоположения, пока другие cs не будут указаны или до конца текущего объекта, в зависимости от того, что наступит раньше.

Поддерживаются следующие системы координат:

- UTM1 – UTM60 ▷ Универсальная поперечная проекция Меркатора (Universal Transverse Mercator) в северном полушарии и заданной зоне, WGS84.
- UTM1N – UTM60N ▷ то же, что и UTM1 – UTM60
- UTM1S – UTM60S ▷ UTM в южном полушарии, WGS84.
- lat-long, long-lat ▷ широта (N положительная, S отрицательная) и долгота (E положительная, W отрицательная) в заданном порядке в градусах (разрешено deg[:min[:sec]]), WGS84. По умолчанию не поддерживается на выходе.
- EPSG:<number> ▷ Большинство систем координат EPSG. Почти каждая система координат, используемая во всем мире, имеет собственный номер EPSG. Чтобы найти номер вашей системы, см. extern/proj4/nad/epsg файл в дистрибутиве исходников.
- ESRI:<number> ▷ Аналогично EPSG, но стандарт ESRI.
- JTSK, iJTSK ▷ Чехословацкая система S-JTSK, используемая с 1920-х годов с южной и западной осью (JTSK) и ее модифицированной версией с осью,

указывающей восток и север отрицательными числми (iJTSK). JTSK не поддерживается на выходе (как и iJTSK).

- JTSK03, iJTSK03 ▷ новая реализация S-JTSK, введенная в Словакии в 2011 году.

5.4 - OSGB:<H, N, O, S или T><A-Z исключая I> ▷ Британская Национальная Сеть.

- S-MERC ▷ сферическая проекция Меркатора, используемая различными сайтами онлайн-сопоставления.

Магнитное склонение

Therion содержит встроенный IGRF[2] Модель геомагнитного поля Земли, дей-
5.4 ствительная для периода 1900–2020 гг.. Он автоматически используется, если пещера находится в пространстве с использованием любой из поддерживаемых геодезических систем координат, и никакое склонение не определяется пользователем. Вычисленное склонение указано в файле LOG для информации.

Формат данных

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.

[2] См. https://www.ngdc.noaa.gov/IAGA/vmod/

‘encoding’

Description:

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

?????????:

encoding <encoding-name>

Context:

It should be the very first command in the file.

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.

?????????:

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>.[3]

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.

?????????:

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

Context:

none, survey

Arguments:

- <id> ▷ идентификатор

Options:

- namespace <on/off> ▷ specify whether survey creates namespace (on by default)
- 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

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

- 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.

?????????:

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-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: station, length,

tape, [back]compass, [back]bearing, [back]clino, [back]gradient, counter, depth, station, position, notes, pictures, pics, instruments (insts), assistant (dog).

- 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 $\pm 90°$ 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

$$true\ bearing = measured\ bearing + declination.$$

  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). These are excluded from length calculations.

  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. *5.3*

  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, tape/length, [back]compass/[back]bearing, [back]clino/[back]gradient, depth, fromdepth, todepth, depthchange, counter, fromcount, tocount, northing, easting, altitude, up/ceiling[4], down/floor, left, right, ignore, ignoreall.

  See Survex manual for details.

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

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.

5.3 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 passeges are horizontal (inclination < vthreshold), LR is perpendicular to the shot and UD is vertical.

  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 modelling of verticals.
- extend <spec> [<station> [<station>]] ▷ control how the centerline is extended. <spec> is one of the following

  normal/reverse ▷ extend given and following stations to the same/reverse direction as previous station. If two stations are given—direction is applied only to given 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

  If no stations are specified, <spec> is valid for following shots specified.
- 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.[5] 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.

[5] 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.

??????????: 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.[6]

- 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.

[6] 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
- 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.

?????????:

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: station[7], section[8], dimensions[9];

  labels: label, remark, altitude[10], height[11], passage-height[12], station-name[13], date;

[7] 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.

[8] `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 with 'none' projection specified. You can specify it through the `-scrap` option.

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

[10] 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

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

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

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

symbolic passage fills:[14] bedrock, sand, raft, clay, pebbles, debris, blocks, water, ice, guano, snow;

speleothems: flowstone, moonmilk, stalactite, stalagmite, pillar, curtain, helictite, soda-straw, crystal, wall-calcite, popcorn, disk, gypsum, gypsum-flower, aragonite, cave-pearl, 5.4 rimstone-pool, rimstone-dam, anastomosis, karren, scallop, flute, raft-cone, clay-tree;

equipment: anchor, rope, fixed-ladder, rope-ladder, steps, bridge, traverse, camp, no-equipment;

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

others: dig, archeo-material, paleo-material, vegetable-debris, root, water-flow, spring[15], 5.4 sink, ice-stalactite, ice-stalagmite, ice-pillar, gradient, air-draught[16], map-connection[17], extra[18], u[19].

Options:

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

  station:[20] 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.[21]

  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.

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

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

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

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

[18] Additional morphing point.

[19] For user defined point symbols.

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

[21] E.g. `station:fixed`

- scale ▷ symbol scale, can be: tiny (xs), small (s), normal (m), large (l), huge (xl) or a numeric value. Normal is default. 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.
- 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.
- dist <distance> ▷ valid for extra points, 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> ▷ valid for extra points, specifies reference station.
- visibility <on/off> ▷ displays/hides the symbol.
- 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>.[22]
- id <ext_keyword> ▷ ID of the symbol.

Type-specific options:

- name <reference> ▷ if the point type is station, this option gives the reference to the real survey station.
- extend [prev[ious] <station>] ▷ if the point type is station and scrap projection is extended elevation, you can adjust the extension of the centreline using this option.
- 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.
- text ▷ text of the label, remark or continuation. It may contain following formatting keywords:[23]

 ▷ line break

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

[22] 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.

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

<thsp> ▷ thin space

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

<rtl> and </rtl> ▷ marks beginning and end of a right-to-left written text 5.3

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

- value ▷ value of height, passage-height or altitude label or point dimensions

  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]).

  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).

  altitude: the value specified is the altitude difference from the nearest station. 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.

'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.

?????????:

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[24], floor-step, pit, ceiling-step, chimney, overhang, ceiling-meander, floor-meander;

passage fills: flowstone, moonmilk, rock-border[25], rock-edge[26], water-flow;

labels: label;

special: border, arrow, section[27], survey[28], map-connection[29], u[30].

Command-like options:

- subtype <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[31], flowstone, moonmilk; *5.4*

  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.[32]

  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*).

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

- close <on/off/auto> ▷ determines whether a line is closed or not

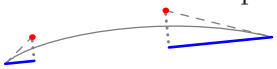
- mark <keyword> ▷ is used to mark the point on the line (see join command).

[24] 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.

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

[26] Inner edges of large boulders.

[27] Line showing cross-section position. If both control points (red dots) of a Bezier 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. *5.3*



[28] Survey line is automatically drawn by Therion.

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

[30] For user defined line symbols.

[31] Usually open to surface.

[32] E.g. `border:invisible`

- orientation/orient <number> ▷ orientation of the symbols on the line. If not specified, it's perpendicular to the line on its left side. $0 \leq$ number < 360.
- 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.
- size <number> ▷ line width (left and right sizes are set to one half of this value)
- r-size <number> ▷ size of the line to the right
- l-size <number> ▷ same to the left. Required for slope type.
- smooth <on/off/auto> ▷ whether the line is smooth at the given point. Auto is default.
- adjust <horizontal/vertical> ▷ shifts the line point to be aligned horizontally/vertically with the previous point (or next point if there is no previous point). The result is horizontal/vertical line segment). If all line points have this option, they are aligned to the average y or x coordinate, respectively. This option is not allowed in the plan projection.
- place <bottom/default/top> ▷ changes displaying order in the map.
- clip <on/off> ▷ specify whether a symbol is clipped by the scrap border.
- visibility <on/off> ▷ displays/hides the symbol.
- 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 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].
- border <on/off> ▷ this option can be specified only with the 'slope' symbol type. It switches on/off the border line of the slope.
- 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. None is default.
- 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).

- head <begin/end/both/none> ▷ can be used only with the arrow type and indicates where to put an arrow head. End is default.
- text <string> ▷ valid only for label lines.
- height <value> ▷ height of pit or wall:pit; available in METAPOST as a numeric variable ATTR__height. 5.4

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.

?????????:

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[33], u[34].

Command-like options:

- the data lines consist of border line references (IDs)
- place <bottom/default/top> ▷ changes displaying order in the map.
- clip <on/off> ▷ specify whether a symbol is clipped by the scrap border.
- visibility <on/off> ▷ displays/hides the symbol.
- 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>.

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

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

Options:

- id <ext_keyword> ▷ ID of the symbol.

'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.[35]

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.[36]

?????????:

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.

?????????:

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

[36] 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.

equate <station list>

Context:

none, survey

'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.

?????????:

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
- 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). 5.4

'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[37]) or surface grid – digital elevation model (in 3D model only).

?????????:

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

[37] 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

'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 Survex' 3D file, stations are inserted in survey hierarchy, if there exists identical hierarchy to that in 3D file.

?????????:

import <file-name> [OPTIONS]

Context:

survey / all[38]

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.

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

'grade'

Description:

This command is used to store predefined precisions of centreline data. See sd option description for centreline command.

?????????: : 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.

?????????:

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).

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 export depending on 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 user in symbols definition macros.

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

XTherion

XTherion is a GUI (Graphical User Interface) for Therion. It helps a lot with creating input data files. Currently it works in three main modes: text editor, map editor and compiler.[39]

It is not necessary for Therion itself—you may edit input files in your favourite text editor and run Therion from the command line. XTherion is also not the only GUI which may be used with Therion. It is possible to write a better one, which would be more user friendly, more WYSIWYG, faster, more robust and easier to use. Any volunteers?

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 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.[40]

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 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 map 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

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

[40] 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.

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 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.[41]

After creating a new file in the map editor, you may load one or more **images**—scanned survey sketches from the cave[42]—as a background for the drawing. Click on the *Insert* button in *Background images* menu. Unfortunately, as a limitation of Tcl/Tk language, 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 of background images using *Open XP* menu.[43]

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 map, which is 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

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

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

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

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 is 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 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 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 Bezier 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. Than 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[44] 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 red arrow will appear (by default, they'll be in the lower corners 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 mouse pointer to 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.

[44] CAUTION! The command `text` is not a Therion command! It's only a nickname for a block of an 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.

TIP: Entering symbols of the same type at once saves you a lot of time because you need not change symbol type and fill options for each new symbol. *Options* box preserves the old value and it's enough to change a few characters.[45] It is a good idea to start with drawing all survey stations (don't forget to give them names according to 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 geo- *5.3*
referencig data included in a PDF map[46].

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

If the PDF file is used directly, you have to set DPI and output format before automatic conversion[47] to a raster format.

PocketTopo data exported in Therion format[48] from PocketTopo application can *5.3*
be imported in text editor as well as in map editor (*File* → *Import* → *PocketTopo therion export* and *Backgroud Images* → *Insert* → *PocketTopo therion export*). The same file is used for both imports. Importing 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 object in the listbox using keyboard: switch using 'Tab' into desired listbox; move with underlined cursor to desired object; press 'Space'
- PageUp/PageDown ▷ scroll up/down in the side panel

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

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

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

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

- Shift+PageUp/PageDown ▷ scroll up/down in file commands window

Drawing area and background images

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

Inserting scrap

- Ctrl+R ▷ insert scrap

Inserting line

- Crtl+L ▷ insert new line and enter an 'insert line point' mode
- LeftClick ▷ insert line point (without control points)
- Ctrl+LeftClick ▷ insert line point very close to 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 previous control point
- LeftClick + drag on the control point ▷ move its position
- RightClick on one of the previous points ▷ selects the previous point while in insert mode (useful if you want to change also the direction of 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 line insertion

Editing line

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

Adding 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 line point

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

Splitting line

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

Inserting point

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

Editing point

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

Inserting area

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

Editing area

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

Selecting an existing object

- LeftClick ▷ select object on the top
- RightClick ▷ select 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 tips and hints.

How to enter 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 data in more files and separate centreline data from map data.

We usually use one *.th file containing 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.[49] The points will than be referenced using @ 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 given directory and contains equate commands to define connections between surveys.

How to draw maps?

The most important thing is to devise 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 centreline from one survey trip. The reason is that connections between scraps should be as simple as possible. Scraps in general are independent on centreline hierarchy so try to forget the survey hierarchy when drawing maps and choose best scrap joins.

We usually insert maps in the last-but-one level in survey hierarchy.[50] Each scrap may than contain arbitrary part of any survey in the last level of 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

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

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

complete d survey. The survey stations names will be referenced using @ symbol (e.g. 1@a) in the scraps.[51]

Scraps are usually stored in *.th2 files. Each file may contain more scraps. To keep data well organized, we have 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 had been defined.

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

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 scrap outline is correct:

- there is no METAPOST warning "scrap outline intersects itself"
- when you set passage fill to any color (color map-fg <number> option in layout), you may see what Therion considers to be inside of 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 exactly work. It gives also 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 -visibility option set on (all symbols by default),
- it hasn't been hidden by the -symbol-hide option in layout,

[51] 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 side you have than use longer names in stations references, like `3@dno.katakomby.jmn.dumbier`

- if its -context option is set, the corresponding symbol hasn't been hidden by the -symbol-hide option in layout.

Only visible symbols are exported.

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

- *point symbols:* symbolic passage fills (bedrock. . . guano),
- *line symbols:* all line symbols which don't have -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 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 -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 of each group follows the order of commands in the input file[52]: 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[53] is filled with color preview-below
- FOR each level[54]: 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

[52] Or *File commands* menu in XTherion

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

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

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 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.[55] 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'

??????????:

- language <xx_[YY]>

5.3 Sets the output language for translatable texts.

'cs'

??????????:

- cs <coordinate system>

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

5.3 Outside of layout command specifies the coordinate system for output. It is not possible to specify more coordinate systems for different outputs (the last occurence 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).

'sketch-warp'

?????????:

- 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.

?????????:

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, centreline from all surveys is exported in the map.

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

?????????:

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.

?????????:

unselect <object> [OPTIONS]

Arguments:

The same as the select command.

Options:

The same as the select command.

'text'

Description:

Specifies translation of any default therion text in output.

?????????:

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'.

?????????:

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> <x> <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> 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/off> scale-bar <length> <units> survey-level <N/all> language <xx[_YY]> colour/color <item> <colour> debug <on/all/first/second/scrap-names/station-names/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)

5.4 CCNP (Carlsbad Caverns National Park)

5.4 NZSS (New Zealand Symbol Set)

  SKBB (Speleoklub Banska 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)

5.3 Group may be one of the following: all, centerline, sections, water, speleothems,
5.4 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. *5.4*

  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.[56] (M, A) *5.3*

- 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. *5.4.1*

- 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 -size option. *5.4.1*

  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

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

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> <x> <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 character. (M, A; default: 0 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 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 <on/off> ▷ turn on/off legend of map-fg colours when map-fg is set to altitude, scrap or map (M, A)
- 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
- 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: off) 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 <item> <colour> ▷ customize colour for special map items (map-fg, map-bg, preview-above, preview-below, label). Colour range is 0–100 for grayscale, [0–100 0–100 0–100] triplet for RGB colours.

  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.

- 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:

- 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

'setup3d'

?????????:

- setup3d <value>

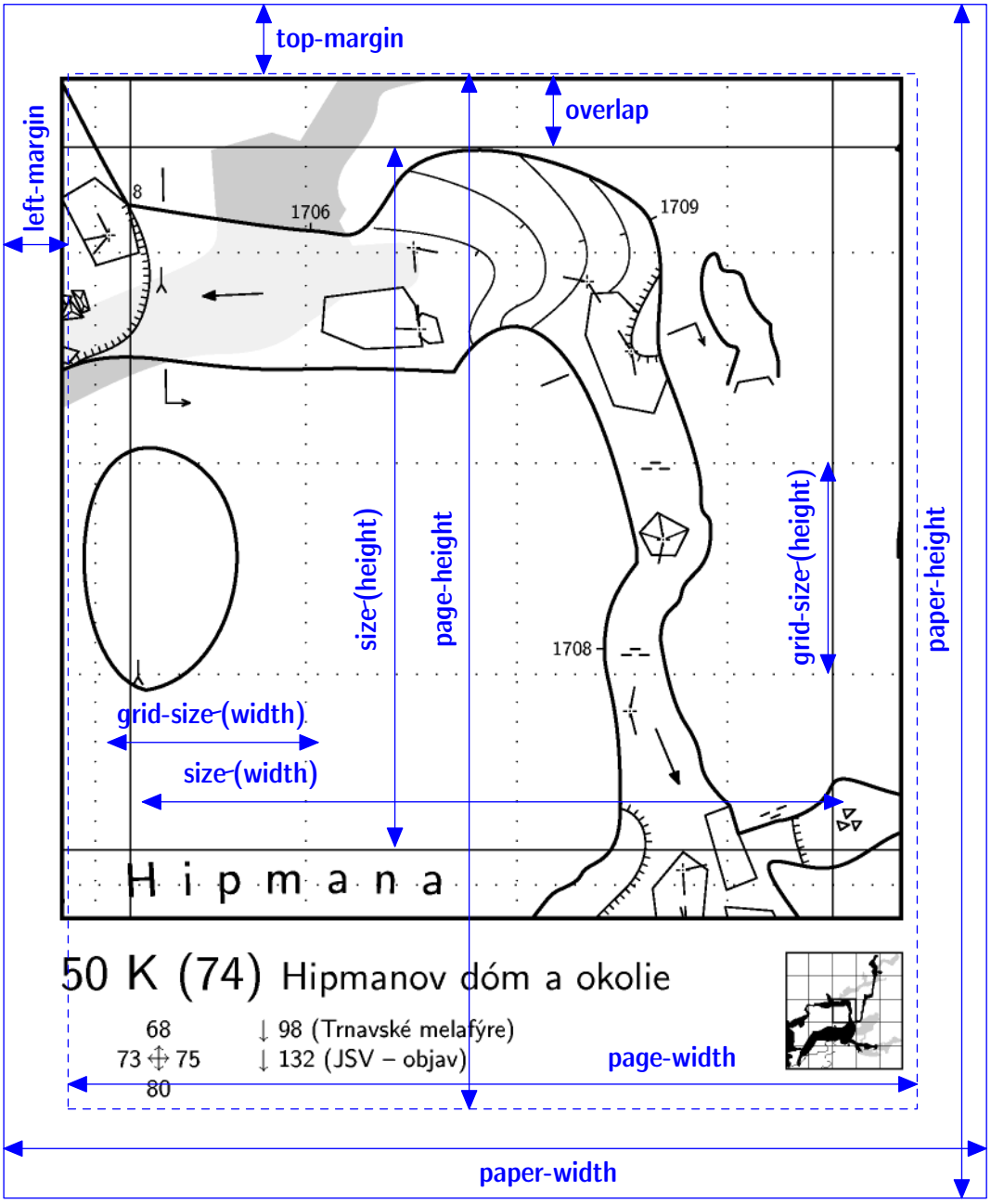
5.3 Temporary hack to set sampling distance in meters when generating piecewise linear 3d model from passage walls made of Bezier curves.

'sketch-colors'

?????????:

- sketch-colors <number-of-colors>

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



'export'

Description:

Exports selected surveys or maps.

?????????:

- 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> ▷ set source data for passage wall modeling.

map/atlas:

- format/fmt <format> ▷ set map format. Currently pdf, svg, xhtml[57], survex, dxf, esri[58], kml (Google Earth), xvi[59] and bbox[60] for map; pdf for atlas are supported.

5.3

- 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.

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

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

[59] 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.

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

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[61] and dbf. *5.4*

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. *5.3*

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 ststistics is also displayed. *5.3*

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	*5.3*
atlas	pdf	
database	sql, csv	*5.4*
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

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

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]

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.
- -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 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 and scrap distortions.

Absolute loop error is $\sqrt{'x^2 + 'y^2 + 'z^2}$, where $'x$ 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. Percentual loop error is calculated as *absolute error / loop length*. Average error is 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.

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 centreline. It is a text file starting with 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 a 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 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.

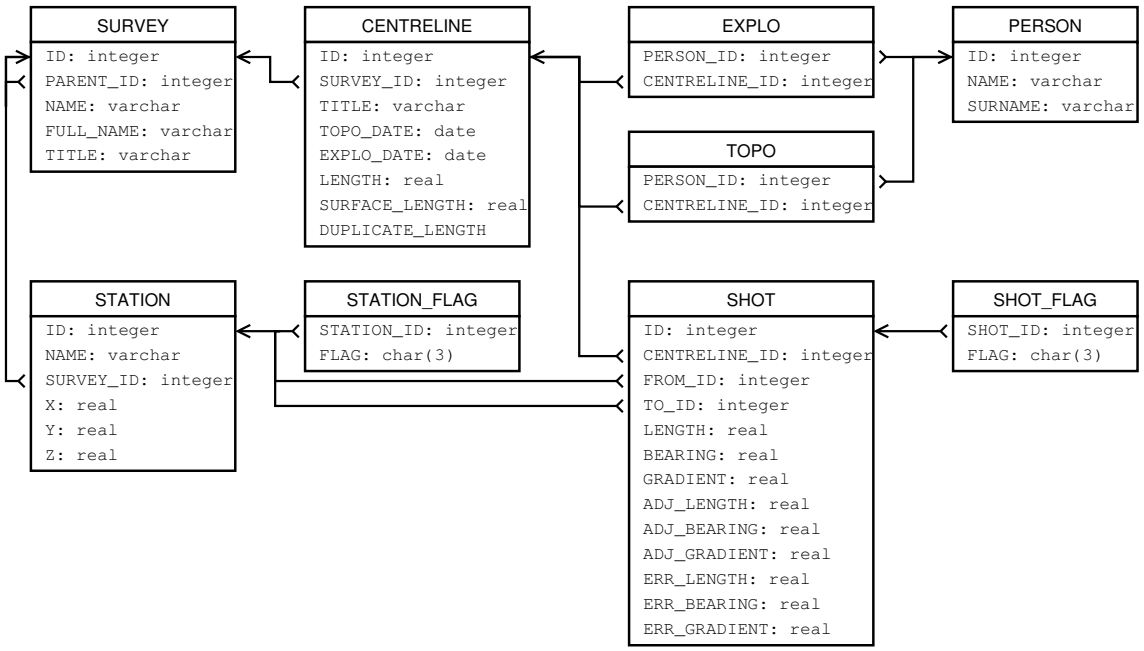


Table and column names are self-explaining; for undefined or non-existing values `NULL` is used. `FLAG` in `SHOT_FLAG` table is `dpl` or `srf` for duplicated or surface shots; in `STATION_FLAG` table `ent`, `con`, `fix`, `spr`, `sin`, `dol`, `dig`, `air`, `ove`, `arc` for stations with entrance, continuation,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

Examples of simple queries follow:

List of survey team members with an information how much has each of them 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 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[62] 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.

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

Especially useful are the numbers of neighbouring pages in N, S, E and W directions, as well as in upper and lower levels. 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 than referenced on appropriate pages. Therion uses most advanced PDF features like transparency and layers.

[62] using `station` attribute for centreline point and `point continuation` in scraps

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). 5.3

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 (and optionally morphed sketches) and can be imported in XTherion to serve as a background for digitization.

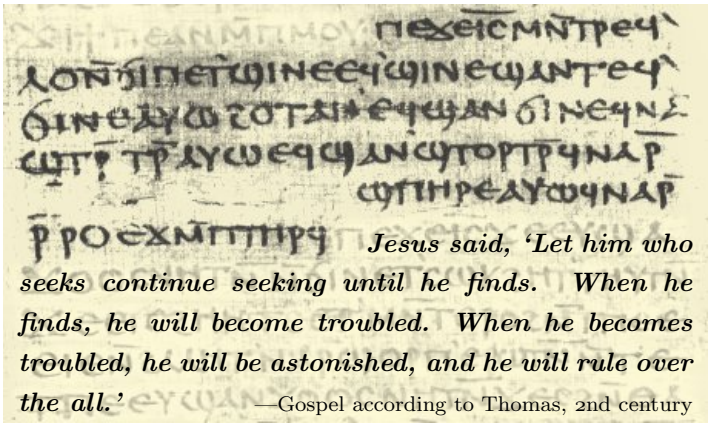
Map in Survex format is intended for a quick preview in Aven.

3D models

Therion may export 3D model 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 overcomplex.

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

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

The TEX macros are used inside of 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 desired page layout. Without this redefinition you'll get the 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 a 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 ▷ name of the current map according to 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 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,
- \color[#1 #2 #3] for colour changes (RGB values in the range 0–100), and
- \rm, \it, \bf, ?, \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\poi
\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 map pages contains atlas additional items: title page, basic facts about the cave, legend with used map symbols etc.

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

- \explotitle, \topotitle, \cartotitle ▷ translated titles

- \exploteam, \topoteam, \cartoteam ▷ participating members (according to team, exploteam 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 exported map
- \cavelength, \cavedepth ▷ approximate length and depth of displayed map
- \cavelengthtitle, \cavedepthtitle ▷ translated labels

5.4
- \cavemaxz, \caveminz ▷ altitude max/min value

5.4
- \thversion ▷ current therion version

5.4
- \currentdate ▷ current date

5.4
- \outcscode, \outcsname ▷ output coordinat system code and name

5.4
- \northdir ▷ 'true' or 'grid'

5.4
- \magdecl ▷ magnetic declination in degrees

5.4
- \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[63] columns if legend option is set on

North arrow 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 scalebar is not hidden
- \northarrow ▷ PDF form with the north arrow
- \scalebar ▷ PDF form with the scale bar

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

There is a general-purpose macro for typesetting in multiple columns[64]:

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

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

code tex-atlas \atlastitlepages

\insertmaps

\formattedlegend

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.

A simple example is

\def\maplayout{ \legendbox{0}{100}{NW}{\northarrow} }

which displays north arrow in the upper-left corner of the map sheet.

For 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}}).

Width of the above text may be adjusted by \legendwidth dimen register (its default value is set by legend-width layout option). The color and size of texts in the preformatted legend can be easily changed using \legendtextcolor, \legendtextsize, \legendtextsectionsize and \legendtextheadersize token registers, e.g. for large blue text:

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

code tex-map \legendwidth=20cm \legendtextcolor={\color[0 0 100]} \legendtextsize={\size[20]}
\legendtextheadersize={\size[60]}

It is possible to display the whole map framed by setting the \framethickness dimen register to 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
> http://cm.bell-labs.com/cm/cs/cstr/162.ps.gz

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 how to do this).

Each symbol has to have a unique name, which consists of 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 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: *5.4*

- 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

Point symbols

Point symbols are defined as macros using def ... enddef; commands. Majority of point symbol definitions has 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 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 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 take automatically 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.

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 an uniform or randomised grid in a temporary picture (having dimensions bbox path) with some point symbols; clip it according to 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 (without 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 pattern definitions.

Example on 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

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
- GNU make
- Perl
- Python 2.7 or 3
- 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/)
- ImageMagick distribution with *convert* and *identify* utilities, if you want to use warping of survey sketches.
- *ghostscript* if you want to create calibrated images from georeferenced PDF maps.

To compile Loch, you need

- freetype 2 and newer; freetype-config must work
- wxWidgets 2.6 and newer; wx-config must work
- VTK 5.0 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 MSYS (http://www.mingw.org). It's a distribution of GNU utilities with GNU make and GCC. (BTW, why not to use precompiled Windows version?)

Quick start

- unpack the source distribution therion-5.*.tar.gz
- cd therion
- make config-macosx or make config-win32, if you use MacOS X or Windows, respectively
- make
- sudo make install

Hacker's guide

Make parameters

Therion's *makefile* may take some optional parameters.

- config-linux, config-macosx, config-win32 ▷ configure Therion for a specific platform. Linux is a default.
- config-release, config-oxygen, config-ozone ▷ set optimization level for C++ compiler (none, -O2 and -O3)
- config-debug ▷ useful before debugging the program
- install ▷ install Therion
- clean ▷ delete all temporary files

Cross-compilation for Windows *5.4*

Therion supports compilation of Win32 executables in Linux using MXE cross compiler (http://mxe.cc).

- install the following static/win32 packages (i686-w64-mingw32.static-*) to the directory /usr/lib/mxe/: binutils, bzip2, expat, freetype-bootstrap, gcc, gettext, glib, harfbuzz, jpeg, libiconv, libpng, tiff, vtk, wxwidgets, xz, zlib.
- modify PATH: export PATH=/usr/lib/mxe/usr/bin:$PATH
- cd therion
- make config-win32cross
- make

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.[65] 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 mpost and tex directories it's necessary to run Perl scripts genmpost.pl and gentex.pl, which generate C++ header files, and compile Therion executable again.

5.3

[65] 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 thPID, 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 ("pdfetex" is the default).

- identify-path <file-path>

  Set the full path to ImageMagick's identify executable if Therion can't find it ("identify" is the default).

- convert-path <file-path>

  Set the full path to ImageMagick's convert executable if Therion can't find it ("convert" 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.

5.4 - cs-def <id> <proj4def>

Define a new coordinate system <id> using Proj4 syntax.

5.3 • 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> 5.3

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.

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 ▷ $\approx 2.8 \times 2.8$ m in the output scale (METAPOST limit)
- page size ▷

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

  SVG map: unlimited
- scraps count ▷ approx. 500–6000, depending on frequency of cross-sections

  current METAPOST limit: $4(scraps + sections) < 4096$ (may be arbitrarily increased)

  pdfTeX limit: $2 \times pages + images + patterns + 6(scraps + sections) < 32500$

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 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.

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)