User's Manual for Thuban 1.0

Jonathan Coles

Jan-Oliver Wagner

Frank Koormann

User's Manual for Thuban 1.0

by Jonathan Coles, Jan-Oliver Wagner, and Frank Koormann

Revision History

Revision 1.0pre3 04-Dec-2003
Corresponds to Thuban 1.0rc1 New: I18n, right button legend menu, EPSG projectons, PostGIS support.
Revision 1.0pre2 29-Aug-2003
Corresponds to Thuban development release 0.8.1. New: chapter on extensions.
Revision 1.0pre1 08-Aug-2003
Corresponds to Thuban development release 0.8.1.

Table of Contents

List of Figures

Chapter 1. Introduction

Thuban is an interactive geographic data viewer. It has been developed because there was no simple interactive viewer for geographic information available as Free Software. Thuban is written in Python and C++ and uses the wxWindows library allowing it to run on many different platforms, including GNU/Linux and Windows.

Geographic data viewers are a necessary tool as they allow one to get a visual impression of the positional relationship of the information that may not be apparent from simple inspection of the data values themselves. Thuban allows the user to create a session that displays geographic data and then explore that data through navigation and manipulation of how it is drawn. The results can then be saved or printed.

Thuban arranges a session in a hierarchy. A session contains a map which consists of layers. Each layer represents one kind of data set. For instance, there may be a layer for roads and another layer for buildings. These layers can either be vector shapes or images.

1.1. Installation

Thuban is actively supported under Debian Testing (sarge), RedHat 7.2, and Windows 2000. Thuban depends on the following packages. These packages can also be found on the Thuban Download site (http://thuban.intevation.org/download.html).

Required:

- Python 2.2.1 (`http://www.python.org`)
- wxWindows 2.4 (`http://www.wxwindows.org`)
- wxPython 2.4 (`http://www.wxpython.org`)
- proj 4.4.5 Projection Library (`http://www.remotesensing.org/proj/`)
- SQLite 2.8.3 (`http://www.hwaci.com/sw/sqlite/`)
- PySQLite 0.4.3 (`http://pysqlite.sourceforge.net`)

Optional:

- GDAL 1.1.8 (`http://www.remotesensing.org/gdal/`)
- psycopg 1.0.8 (`http://initd.org/software/psycopg`)

Along with the source codes, the download page also offers full installation packages for Debian, Windows and RPM-based systems (Mandrake, RedHat, SuSE, etc).

1.1.1. RPM-based GNU/Linux Systems

1.1.1.1. Installing Binary Packages

The most wide-spread RPM-based GNU/Linux Systems are RedHat, Mandrake and SuSE. The documentation of these distributions should contain information about how to install third-party RPM packages. Nonetheless, a short summary is provided here.

RPM packages can be installed applying several tools. The most basic one is the command line program "rpm". The hardware architecture is identified in the name of RPM packages, eg. 'i386' for most Intel/AMD architectures. If you have a different hardware architecture, where no binary RPM packages are provided, you must rebuild binary packages from the RPM source packages first (see below). Typical rpm commands look like:

```
rpm --install Thuban-0.9.0-1.i386.rpm
```

Depending on what you already have installed on your system, you are informed that some packages are required, but not installed. You need to install them first. Either they are provided by your GNU/Linux distributor or available somewhere on the Internet. The more essential and special ones are provided together with the Thuban package.

For rpm exist some graphical user interfaces, notably kpackage, GnoRPM and xrpm.

Make yourself familiar with one of the tools and apply it to install the packages. Note, that you need to be administrator (root) for the system to do that.

1.1.1.2. Build Binaries from Source Packages

This section describes howto build RPM install-packages from RPM source-packages. This adapts and optimizes an install-package specifically to your system. This is especially helpful to resolve version conflicts of dependent packages. Furthermore, install-packages for other platforms (e.g. PowerPC) can be created.

Note: rpm must be at least version 4. Execute `rpm --version` to find out about the version.

You need to do the following preparations to be able to build the packages as a regular user. You should now perform the package buling as root since this might cause damage to your system.

- Create RPM directory structure: Choose a directory (e.g. $HOME/myrpm) and create the subdirectories BUILD, RPM, SOURCES, SPECS and SRPMS. A possible command sequence for this is:

```
mkdir $HOME/freegisrpm
cd $HOME/freegisrpm
mkdir BUILD RPMS SOURCES SPECS SRPMS
```

- Set environment variable RPM_DIR:

```
export RPM_DIR=$HOME/freegisrpm
```

- Create $HOME/.rpmmacros: This file sets general preferences and some specific settings for signing packages. If you don't have a GnuPG-key, you can skip the signature settings i.e. drop the last 4 lines. A signature becomes important when you want to give away packages to third parties.

```
%packager Name Lastname <Name.Lastname@mydomain.example>

%_topdir /home/mylogin/myrpm

%_signature gpg
%_gpg_name Name Lastname
%_pgp_path ~/.gnupg
%_pgpbin /usr/bin/gpg
```

Now you can install any RPM source-package. It's components are installed into the corresponding subdirectories of your rpm-directory. Essentially these are the sources (into directory SOURCES) and the so-called spec-file which contains all build instructions. The spec-file will go into the SPEC directory. Example: `rpm --install Thuban-0.9.0-1.src.rpm`

Create install-package: Go to the directory with the spec-files and rebuild the package:

```
cd $HOME/mypm/SPECS
rpm -bb thuban.spec
```

Next, you will find the newly created package in $HOME/myrpm/RPMS/i386. If you build the package for another architecture than i386, then the name of the directory has a corresponding name.

For documentation of RPM, either type `man rpm` or `rpm --help`. This will provide you with information on the various command line options of RPM. For more information see the homepage of RPM (http://www.rpm.org/).

1.2. Internationalization

Thuban is implemented with internationalization support. So far Thuban is translated by volunteers to the following languages (apart from its main language: English):

• French

• German

• Italian

• Russian

• Spanish

To use internationalization under POSIX systems (like GNU/Linux) you have to set the environment variable LC_ALL accordingly (e.g. LC_ALL=fr_FR for the french language support). Please check your systems documentation for details and supported settings. Specifiying LC_ALL on the command line while launching thuban allows appication specific language settings.

MS Windows users have to specify the language to be used via the control bar (which effects all applications).

1.3. The Main Window

Figure 1-1. The Main Window
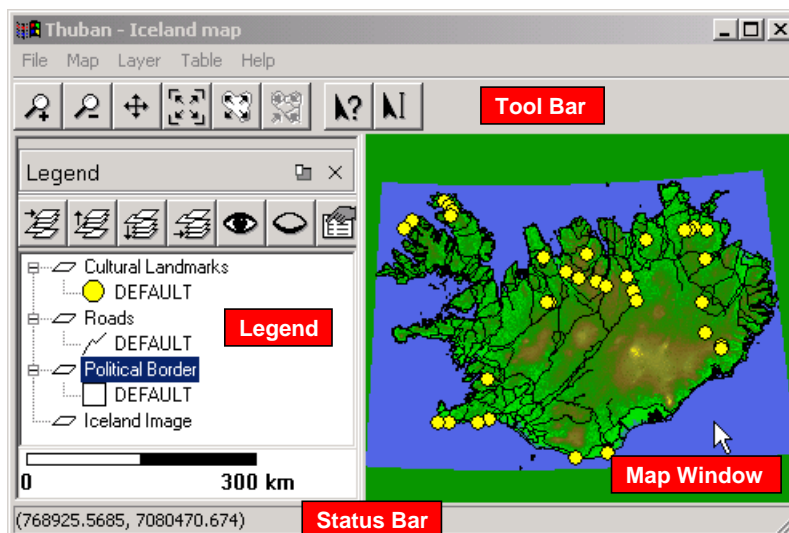
The map window shows the current state of the map and is where the user can interact with the map using the tools.

The legend on the left displays a list of the current layers and any visible classification groups. In the example, each shape layer has a default classification which specifies how the shapes in each layer are drawn. Layers that are higher in the list appear "closer" to the user. The legend can be closed by clicking on the small X in the upper right-hand region of the legend. To open it again, use Map⟶Legend. The legend is also dockable, which means that it can be detached from the main window by clicking on the small button next to the close button. It can be attached by clicking the same button again.

The status bar displays different information depending on the current context. If the user is selecting an item from the menu then the status bar will display a short help message indicating what each menu item is for. If the user has a tool selected then the position of the cursor on the map is displayed.

The tool bar provides quick access to the commonly needed tools. By hovering over each button the user can see a short messages describing what the tool does. The tools provided are Zoom In, Zoom Out, Pan, Full Extent, Full Layer Extent, Full Shape Extent, Identify, and Label. Each of the tools will be explained in further detail later in the manual.

Chapter 2. Session Management

2.1. Starting a New Session

A new session can be started from File—→New Session. If a session is already loaded and has been modified without being saved a prompt will ask if the current session should be saved. A new session consists of an empty map with no layers and no tables.

2.2. Opening a Session

A session can be opened from File—→Open Session. A dialog box will open allowing the user to browse for a Thuban Session file. Thuban session files end with `.thuban`. Selecting a file a clicking OK will load the session into Thuban. If a session is already loaded and has been modified without being saved a prompt will ask if the current session should be saved.

2.3. Saving a Session

A session can be saved from File—→Save Session. A dialog box will open allowing the user to browse the file system and select a place to save the session. Thuban sessions should be saved under a name ending in `.thuban`. If the file already exists the user will be prompted to save under a different name or overwrite the existing file.

2.4. The Session Info-Tree

Figure 2-1. Session Info Tree
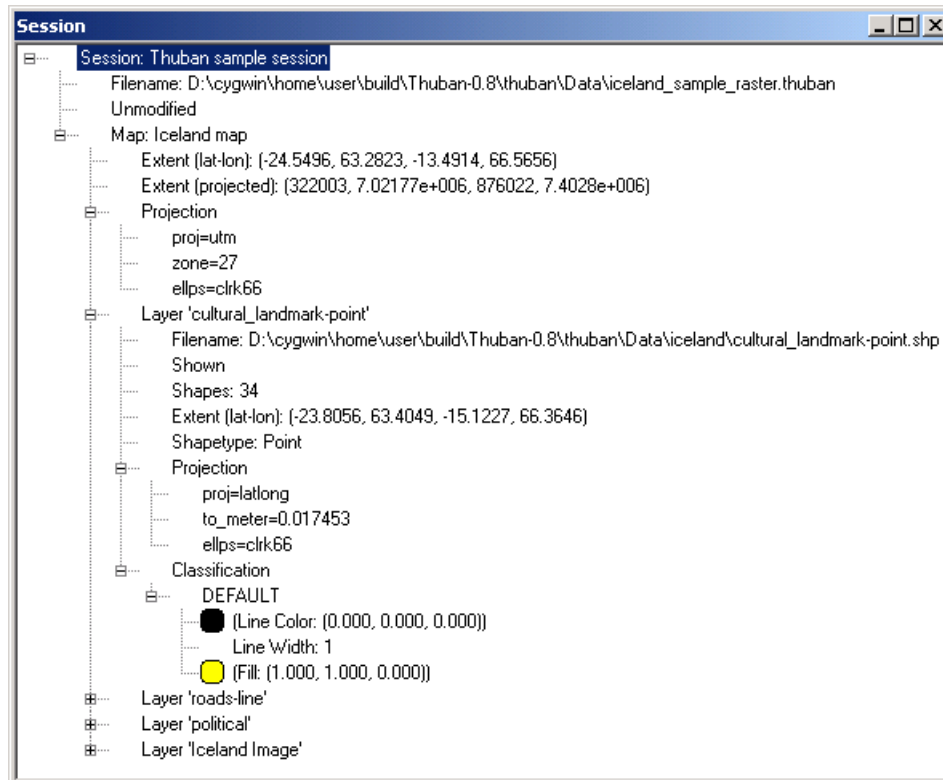
The session info-tree is primarily intended for developers working with Thuban. It displays many of the internal values for the session, map, and layers. It can be opened from File→Session Tree.

Chapter 3. Map Management

The map consists of a number of layers where each layer represents a different type of data set. By interacting with the map the user can visually explore the data.

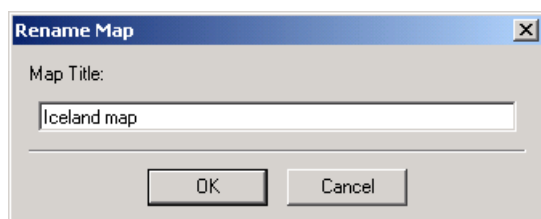The map can have a name that will appear in the Thuban title bar. The map name can be changed using Map⟶Rename.

3.1. Adding and Removing Layers

There are three types of layers that can be added to a map: Shape layers, database layers and image layers. Shape layers are stored in Shapefile format, a widely used file format for storing geographic objects. These files have the extension ".shp". Associated with the shape file is a database file which stores attributes for each shape in the Shape file. This file, in dBase format, has the extension ".dbf". Both files must have the same base name. For example, if there is a shape file named roads.shp there must also be a file roads.dbf.

- Shape layers can be added to the map with Map⟶Add Layer. Initially, only the ".shp" files are shown which is enough for the selection. However, if you switch to display all files and select one of the associated files (e.g. ".dbf"), Thuban will recognize the base name and load the corresponding Shape file.

  The file dialog for Shape files allows to select multiple files. Use the shift-button together with the left mouse button to extend the selection.

- Database layers can be added to the map with Map⟶Add Database Layer. A dialog with two is list is opened. The left list displays all database connections currently open for the session. You can retrieve a list of available layers from the selected database which is displayed on the right hand. From this list one layer can be selected, the dialog is closed afterwards.

  See appendix "Working with PostGIS" for details.

- Image layers can be added to the map with Map→Add Image Layer. It is important to select a valid image file that has geographic data associated with it. The data can be embedded in the file itself, or in another file. If geographic information cannot be found, Thuban will report an error.

3.2. Navigation

The map can be explored by using the navigation tools available on the tool bar or from the Map menu.

- The ZoomIn tool enlarges a region of the map. Clicking once on the map will double the magnification and center the map on the point that was clicked. Clicking and dragging selects a region that will be enlarged to fit the window.

- The ZoomOut tool shrinks the map so that a larger region is visible. A single click reduces the magnification by a factor of two. Clicking and dragging selects a box such that the current contents of the window will be scaled to fit into that box.

- The Pan tool allows the user to move the map around by clicking and dragging.

- The Full Extent tool rescales the viewable region so that the entire map is visible.

- The Full Layer Extent tool rescales the viewable region so that the currently selected layer fits within the window. If no layer is selected this button will be disabled.

- The Full Shape Extent tool rescales the viewable region so that the currently selected shape fits within the window. If the shape is a point, it is centered and the map is zoomed all the way in. If no shape is selected this button will be disabled. This feature is especially helpful when identifying an object related to a selected record in a tableview (see below).

3.3. Object Identification

Objects on the map can be identified using the Identify tool . Clicking on an object selects that object and opens a dialog which shows all the table attributes for that object. Any current selection is lost. Objects on the map are typically shapes and this document will often refer to objects as shapes.

3.4. Object Labeling

Objects can be labeled using the Label tool . Clicking on an object selects that object and opens a dialog which displays the table attributes for that object. An attribute can be selected to be the label on

the map. The label will be placed at the center of the shape. Clicking on an object that already has a label will remove the label.

3.5. The Legend
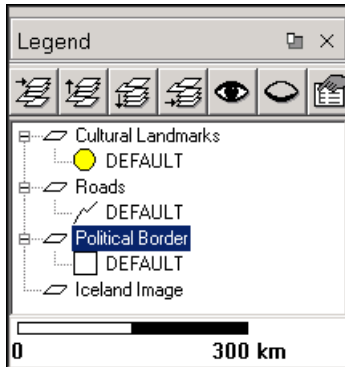
The Legend provides an overview of the layers in the map. Layers that appear higher in the legend will appear "closer" to the user. If a layer supports classification (currently, only shape layers have this feature) then the classification groups will be shown below each layer. The properties for each group are also displayed with a small graphic. Polygon layers appear as rectangles, lines appear as curved lines, and points appear as circles.

Along the top of the legend is a toolbar which allows quick access to some of the layer manipulation options under Map.

- The Move Layer to Top tool raises the selected layer to the top of the map.

- The Move Layer Up tool raises the selected layer one level.

- The Move Layer Down tool lowers the selected layer one level.

- The Move Layer to Bottom tool lowers the selected layer to the bottom of the map.

- The Visible tool shows the selected layer in the map if it was hidden.

- The Invisible tool hides the selected layer in the map.

- The Properties tool opens the layer's properties dialog box. Double-clicking on a layer or a group of a layer will open the properties dialog for that layer.

The most used layer related actions are also available from a popup menu. It is raised when a layer is clicked with the right mouse button.

Figure 3-1. Layer Popup Menu
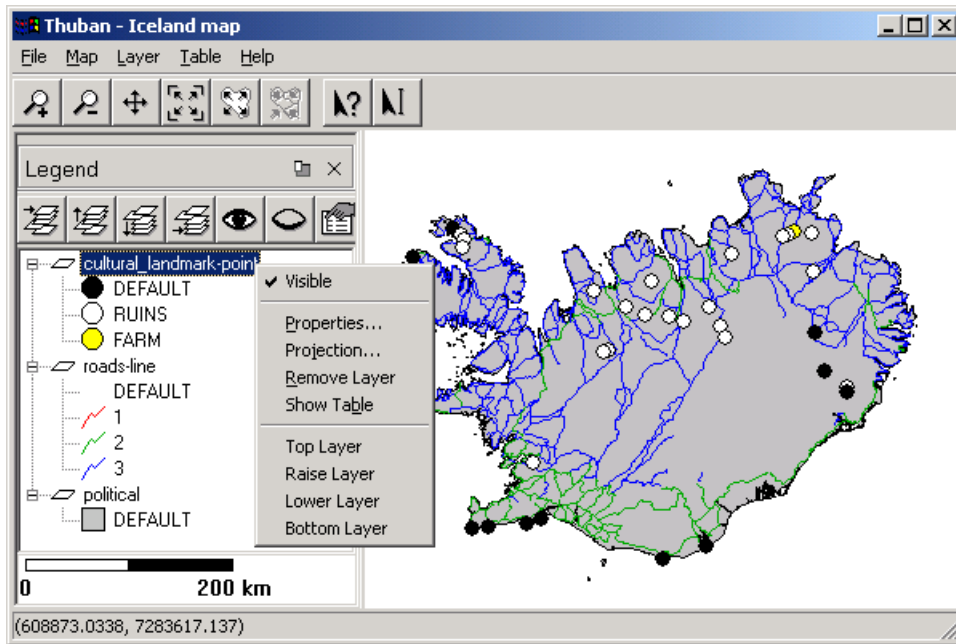
Along the bottom of the legend is the scalebar. The scalebar will be available if there are any layers and the map has a projection set.

3.6. Exporting

Under Windows, maps can be exported in Enhanced Metafile format (.wmf) from Map⟶Export for use in reports, presentations, or further modification. The current map view, legend, and, if available, scalebar are exported. Under other platforms this option is not available. Clicking this menu item open a file selection dialog that lets the user select a location to export the map.

3.7. Printing

The map can be printed using Map⟶Print. The current map view, legend, and, if available, scalebar are printed. A standard printing dialog will open allowing the user to configure the printer. This dialog will differ depending on which platform Thuban is running.

Chapter 4. Layer Management

4.1. Types of Layers

There are three types of layers supported by Thuban: shape layers, database layers and image layers. Shape layers consist of vector based shapes with geo-referenced coordinates. There are three types of supported shapes: polygons, lines (arc), and points. Database layers are similar to shape layers but loaded from a database instead of the file system. Image layers can be any image file format supported by the Geo-spatial Data Abstraction Library (GDAL). The images must have geographic coordinate data either embedded within the file or in a separate file that is in the same directory as the image file. GeoTIFF files work very well with Thuban and were designed specifically to be image layers in GIS programs.

All actions in the Layer menu act on the currently selected layer in the legend.

4.2. Properties

To view the properties for a layer it must first be selected in the legend. The menu option Layer—→Properties opens a dialog that displays a layer's properties. All layers have a title which can be modified in the text field provided. The type of layer is also shows. If the type is a type of shape (polygon, arc, point) the classification table will be shown. Image layers have no other properties other than title and type.

Figure 4-1. Properties Window
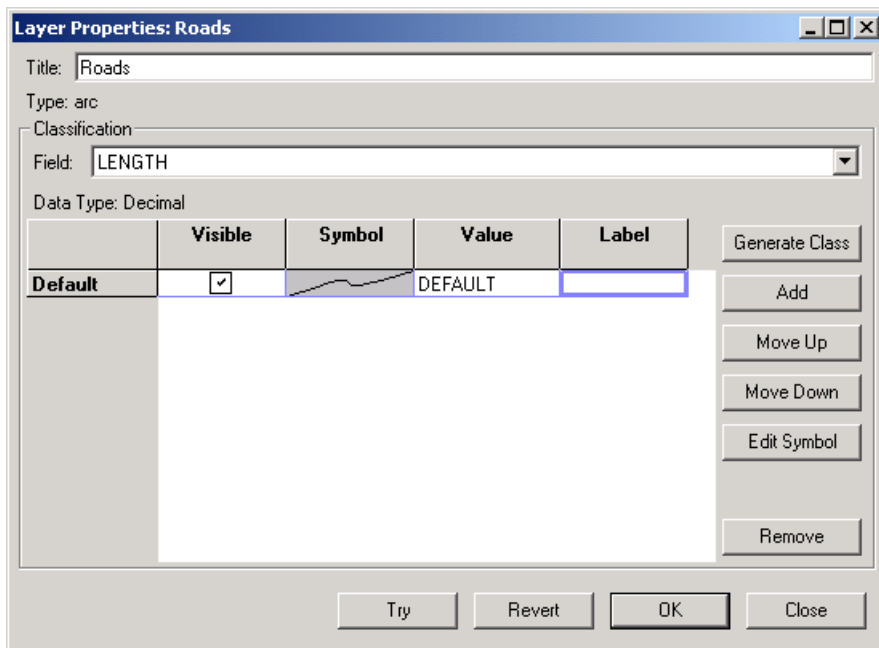
Figure 4-2. Properties Window

4.3. Visibility

Sometimes it is not desirable to view all layers at the same time. Some layers may take a long time to draw and so while navigating around the map the user may not want to wait for the map to redraw all the layers each time the map is changed. Each layer can be independently turned on or off using the Layer⟶Show or Layer⟶Hide options respectively.

4.4. Duplication

Layers and all their properties, including classifications, can be duplicated using Layer→Duplicate. Duplicating a layer is useful if the user wishes to model a layer in several different ways. Even though the layers overlap, by carefully selecting the shape properties it is possible to display several pieces of information at once. For example, one copy of a roads layer may be classified on a length property and another copy may be classified on a type property. If the length property was expressed with color and the type property expressed with line thickness then it would be possible to view both classifications by placing the type property copy over the length property copy.

Chapter 5. Layer Classifications

A layer classification is a way of assigning drawing properties to groups of shapes based on attributes stored in the layer's table. Only layer's with shapes can have a classification; image layers cannot be classified.

A classification consists of a number of groups, each group having a value or range of values to match against, and symbol properties which control how a shape is drawn on the map. The user selects which field in the table is used by the classification and when the map is drawn the value for that field for each shape is compared with each group's value. The properties of the first group to match are used to draw the shape. This allows the user to get a visual impression of not only how the data is laid out but also what kind of data lies where.

A layer always has a classification. When a new layer is added to the map, a default classification is created with the DEFAULT group. This group cannot be removed but can be hidden (see below). Every shape in the layer, regardless of its attributes, will match this group if no other group matches.

5.1. Editing Classifications

A layer's classification can be modified under the properties dialog (Layer⟶Properties). The layer's classification field can be set to None, which simply assigns a DEFAULT group to the classification. No new groups can be added to the classification if the field is None. The user must first select a field to classify on. New groups can be added to the classification with the Add button.

To apply the changes to the map the user can click either Try or OK. Try will not close the dialog box, allowing the user to see how the classification changes the map. Revert will undo the last classification applied to the map. OK will commit the changes and close the dialog. The user will be unable to undo the changes. Close simply closes the dialog box. If any changes have not been applied with Try the changes will not be applied to the map.

Figure 5-1. Properties Window
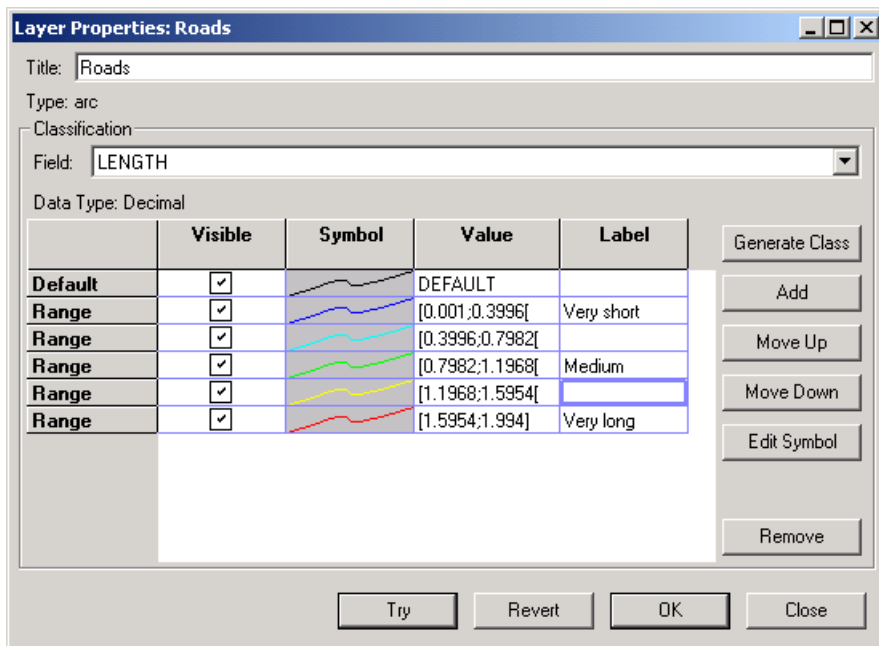
The order of the groups in the classification is significant except for the DEFAULT group, which remains at the top. When shapes are matched against groups the matching begins at the first group after the DEFAULT group so that groups higher in the list will be checked first. Matching for a given shape will stop at the first group that matches. The user can use Move Up and Move Down to change the order of the groups. The DEFAULT group will always match a shape that hasn't matched another group.

5.1.1. Visible

The Visible column has check-boxes that determine whether a classification group will be displayed in the legend. This is useful if the user knows that the groups completely cover the data set and don't want the DEFAULT group to be displayed in the legend and on a printout.

5.1.2. Symbols

Each type of shape has its own type of symbol. Thuban supports three types of shapes: polygons, lines, and points. Polygons and points have outline and fill color, while lines have only line color. Each group has associated symbol properties. To edit the symbol properties for a group the user can double click on the Symbol column or select a group and click the Edit Symbol button.

5.1.3. Value

The Value column of the classification table is the value that will be matched when the map is being drawn. The type of data that can entered into this field depends on the type of data of the classification field.

If the field is of type Text, anything entered into the field is valid. The text will be compared literally to the value of the shape attribute, including case sensitivity. If the type is Integer, then any valid integer may be entered. In addition, with special syntax, a range of values can be entered. A range from `start` to `end` inclusive is specified like this: `[start;end]`. The exclusive range is specified like this: `]start;end[`. Ranges can include infinity like this: `[-oo;oo]`. Field types can also be of type Decimal. They represent any rational number and can be used in ranges as well.

5.1.4. Label

By default, the text that is displayed for a group in the legend is the value for that group. The label can substitute a more descriptive term in the legend.

5.2. Generating Classes

Figure 5-2. Generate Class
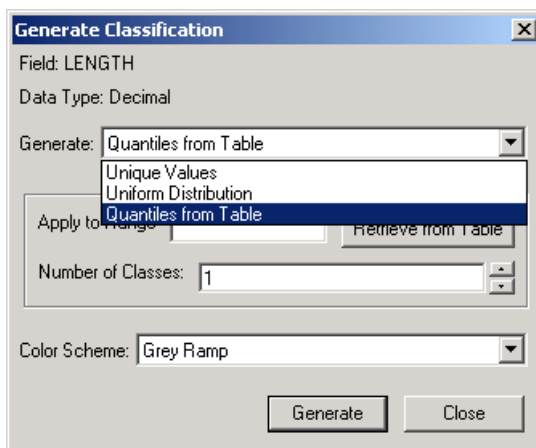
Creating a classification by hand can be tedious. Thuban, therefore, provides a means of generating an entire classification at once while still giving the user control over how it appears. Clicking Generate Class opens the `Generate Classification` dialog. Under the `Generate` pull down there are at

most three different ways to generate classifications: Unique Values, Uniform Distribution, and Quantiles. Some options may not be available if the data type for the field does not support them. For instance, `Uniform Distribution` doesn't make sense for a Text field.

For every way of generating a classification, a color scheme must be selected. Thuban provides several different color schemes that affect how the group properties change over the classification. It may be desirable that only certain properties change over the classification. If the shape type is a polygon or a point then the Fix Border Color option will be available. This allows the user to select a border color for all classification groups. It is also possible to create a custom color scheme. Selecting this option will display two symbols: the one of the left has the properties of the first group and the one on the right has the properties of the last group. Thuban will interpolate between these two properties to generate the other groups.

Figure 5-3. Custom Color Scheme
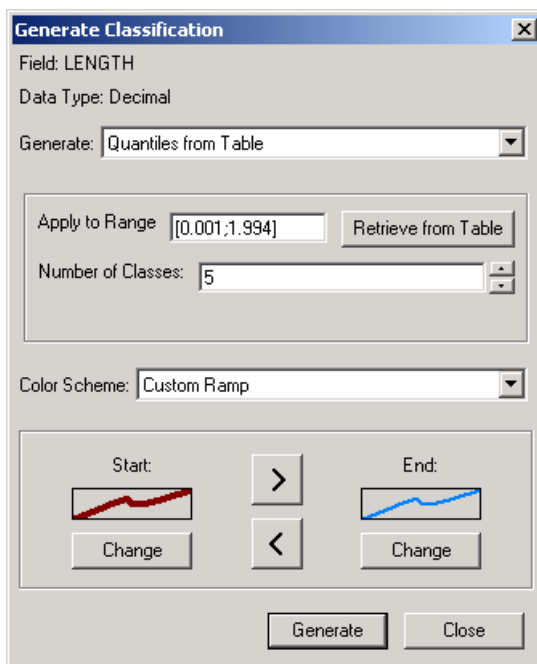
The Unique Values option lets the user select specific values that appear in the table. Clicking Retrieve From Table searches the table for all unique values and displays them in the list on the left. Items can be selected and moved to the list on the right. Each list can be sorted or reversed for easier searching. The classification that is generated will be in the same order as the list on the right.

Figure 5-4. Unique Values
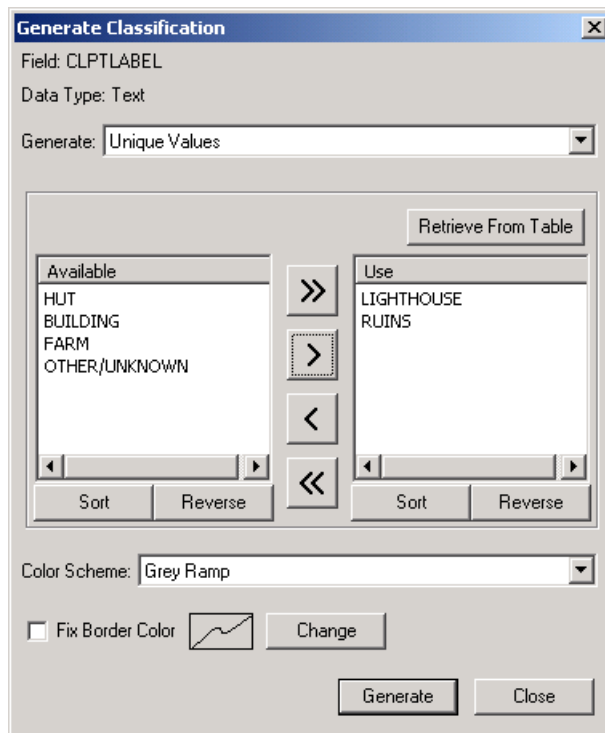
The Uniform Distribution option creates a user specified number of groups of ranges such that each range covers equal intervals. The minimum and maximum values can automatically be retrieved from the table by clicking Retrieve From Table. The stepping is how large each interval is. Adjusting this value will automatically recalculate how many groups is appropriate.

Figure 5-5. Uniform Distribution
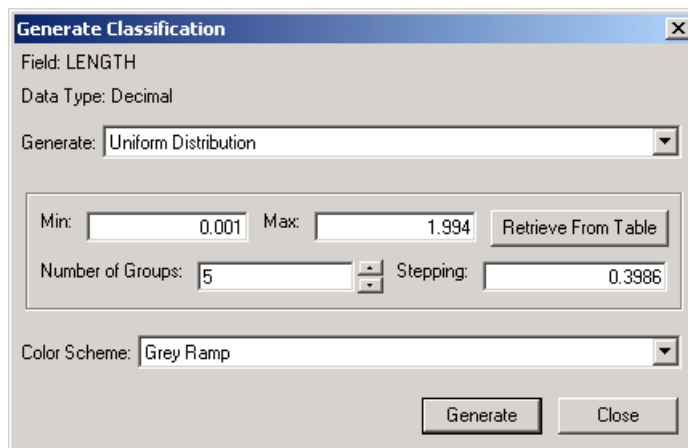
The Quantiles option generates ranges based on the number of items in the table. For example, by specifying five groups Thuban will generate five groups with appropriate ranges such that 20% of the table data is in each group. If it is impossible to generate exact groupings, Thuban will issue a warning but allow the user to continue.

Figure 5-6. Quantiles
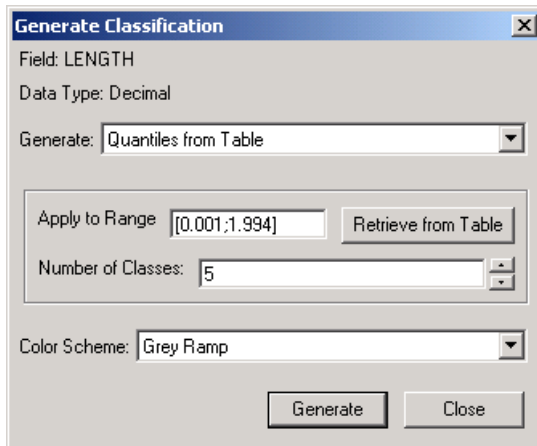
Chapter 6. Projection Management

Projections control how the geographic data is displayed on the screen. If multiple layers are loaded into Thuban where the geographic data is in a different projection system, then the user must specify a projection for each layer. The user must also tell Thuban which projection the map is in. This can be the same as the layers or a different projection in which case the layers are reprojected into that space. The map projection can be set using Map⟶Projection and the layer projection can be set using Layer⟶Projection.

Figure 6-1. Projection Window
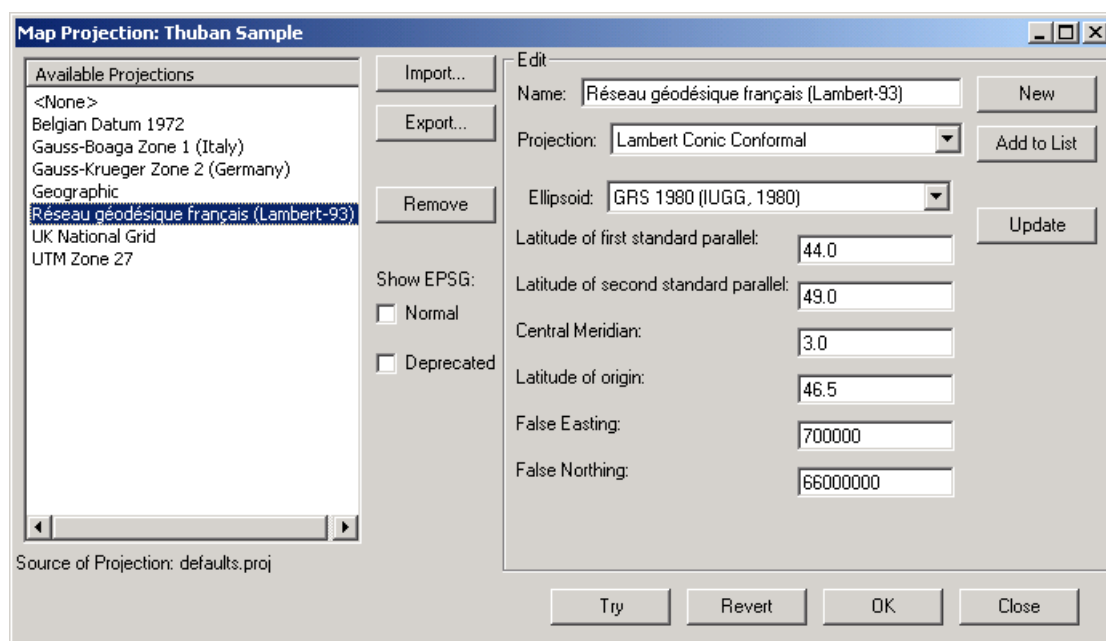
Thuban is distributed with a sample collection of projections and the set of coordinate systems as used by the EPSG (European Petroleum Survey Group). This quite large set is only displayed if activated by the according checkbox. The set falls into two parts: deprecated lists all projections which are no longer part of the EPSG data base.

The user can create new projections and make them available to all future Thuban sessions. They may also be exported and imported so that custom projections can be distributed.

6.1. Selecting a Projection

The available projections are listed on the left. If the layer or map already has a projection it will initially

be highlighted and will end with `(current)`. Selecting `<None>` will cause Thuban to use the data as it appears in the source file and will not use a projection.

6.2. Editing a Projection

Whenever a projection is selected from the list its properties are displayed on the right. These properties can be changed and the changes saved to the selected projection using Update. Only a projection that comes from a file can be updated, so if the current layer's projection is selected, Update will be disabled. Add to List adds the projection to the list of available projections as a new entry, and thus makes it available to future Thuban sessions. Clicking New will create an entirely new, empty projection. The Remove button will permanently remove a projection from the list of available projections.

To apply the selected projection to the map the user can click either Try or OK. Try will not close the dialog box, allowing the user to see how the projeciton changes the map. Revert will undo the last projection applied to the map. OK will commit the changes and close the dialog. The user will be unable to undo the changes. Close simply closes the dialog box. If no selection has been applied with Try the selection will not be applied to the map.

6.3. Importing/Exporting Projections

The projections that appear in the list of available projections can be exported to another file that the user chooses. By selecting one or more projections and clicking Export the user will be able to select a file in which to store those projections. The file can then be distributed to other Thuban users. To import a projection file the user can click Import. The imported projections are added to the list and are then available to the current session and any future Thuban sessions.

Chapter 7. Table Management

Thuban distinguishes two different types of tables: Attribute tables (which belong to a layer) and normal data tables. Both provide the same general functionality with the difference that actions on an attribute table might also effect the map display.

7.1. Table View

Figure 7-1. Table View

Thuban provides a standard dialog to display table contents, the Table View. The view has five sections: The title, selections, the table grid, export functions, and the status bar.

The title bar identifies the table with its name.

The selections box let the user perform simple analysis on the data based on comparisons: The first choice must be a field identifier of the table, the second choice determines the type of comparison. The third choice can be either a specific value (interpreted as numerical or string depending on the type of the first field) or a second field identifier. Thus you can perform analysis like selecting all records where `population > 10000` or `cars_per_inhabitant < bikes_per_inhabitant` (note that the field names are only explanatory, the dBase files allow only 11 character field names). Selections can be combined either by applying a selection only on a previously selected set of records or by adding the results of a selection to a previous set. The default is that a selection replaces earlier results.

The table grid shows the contents of the table (one record per row), with highlighted selection results. Columns and rows can be resized.

The contents of a table can be exported into a file, either dBase format (DBF) or comma separated values (CSV). The Export button raises a file dialog to specify a path and file name, the export type is determined by the file extension (either .dbf or .csv). The Export Selection button works similarly but exports only the selected records. The Close button closes the table view window. This is different from the menu item Table—→Close which unloads the table from Thuban.

The status bar displays some statistics about the table and optional selection results.

7.2. General Functionality (Menu Table)

The general functions affect all tables open in Thuban. Attribute tables are considered here as normal data tables (with the exception that they cannot be closed).

7.2.1. Open

The Table—→Open item raises a file dialog to let you select a dBase file from the file system to be loaded into Thuban read-only. On OK the selected file is loaded and a table view is opened.

7.2.2. Close

The Table—→Close item raises a dialog listing the currently open data tables (loaded via Table—→Open). Selected tables are dereferenced on confirmation. Since tables are opened read-only the contents of the tables are not affected. Any open views of the tables are closed as well. Tables used in a join cannot be closed.

7.2.3. Rename

Table—→Rename changes the table title.

7.2.4. Show

The Table—→Show item raises a list of available tables (explicitly loaded, attribute tables, results of a join). Selected tables are show in tables views on OK.

7.2.5. Join

Figure 7-2. Join Tables
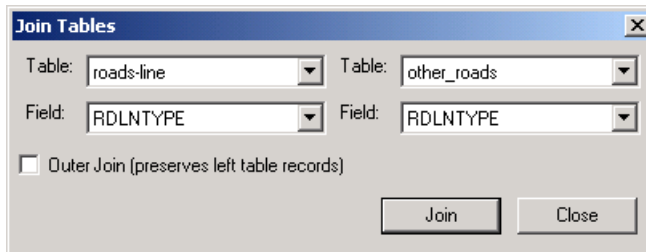
The Table—→Join item raises a dialog to specify the two tables to be joined. The join results in a new table named 'Join of "left table" and "right table"'. The dialog lets you select the two tables to be joined and the two fields the join has to be performed on. By default, the new table contains only those records which are matched by the join. If you want to preserve the records of the left table you can perform an outer join. The fields from the right table for records not matched by the join are filled with None in this case.

7.3. Attribute Tables

To clearly separate between both types of tables (data and attribute), Thuban provides functionality regarding the attribute tables under the Layer menu.

7.3.1. Show Table

Layer—→Show Table opens the attribute table of the currently active layer in a table view. In addition to the functionality described above selections affect also the map display: objects related to selected records are highlighted.

7.3.2. Join Table

Unlike the join described above, the join does not result in a new table. The attribute table of the currently active layer is the left table and other tables are joined to this table. The results of the join are available for classification. As a consequence, the join cannot result in fewer records than the source attribute table. The user is warned if the right table does not fulfill this constraint. An outer join must be used in such cases.

7.3.3. Unjoin Table

As said above, a normal table cannot be closed while it is still used in a join. While the joined table resulting from a join of normal tables can be simply closed (and thereby dereferencing the source tables), this is not possible for attribute tables. Hence joins on attribute tables must be solved explicitly. This is what the Layer—→Unjoin Table item is used for: The last join for the currently active layer is solved.

Chapter 8. Extensions

Thuban is designed to be extensible. The term Extension is used as a general term for anything that extends Thuban. This chapter introduces into some oppportunities how to add and handle extra functionality developed by your own or third parties.

8.1. Add personal extensions via thubanstart.py

After Thuban has been started for the first time, a directory .thuban is created within your home directory. There you can add a file thubanstart.py which will be imported by Thuban at start-up. It is recommended to add only import-statements to this file to keep the actual code of extensions separate.

The modules to import must either be found through the environment variable PYTHONPATH or directly be placed into the .thuban-directory.

As an example, copy the file examples/simple_extensions/hello_world.py of the Thuban source code into the .thuban-directory of your home directory. Now add add the statement import hello_world to the file thubanstart.py and run Thuban. You will notice an additional menu Extensions where the new item for the Hello-World extension is placed - select it to see the Hello-World message.

8.2. Extensions included in Thuban package

The extensions described in this section are part of the Thuban package, but not activated by default. You will find them in the Thuban installation directory under Extensions/. Activate them as personal extensions via PYTHONPATH as described in the previous section. Stable extensions will appear under the menu Extensions and extensions which are in experimental state and therefore not fully functional under Experimental.

8.2.1. Stable extensions

These extensions provide extra-functionality to Thuban that has not (yet) been integrated in the main application. They are considered to be free of bugs, but may be further polished with helpful user interactions.

8.2.1.1. gns2shp

This tool converts data of the Geospatial Names Server (GNS, see http://www.nima.mil/gns) into Shapefile format. The above web-site offer to download named places information grouped by countries

for all of the world except USA for which other data are provided.

If you download and unpack a package, you will have a text-file with suffix .txt. Selecting such a file via gns2shp will create the corresponding Shapefile with the same basename and place it in the same direcory. Afterwards it is automatically loaded into Thuban. The Shapefile will not automatically be delete afterwards.

The gns2shp.py module can also be executed on the command line for batch processing purposes.

A sample (`ls.txt` for Liechtenstein) is included in the directory `Extensions/gns2shp/test`.

8.2.2. Experimental extensions

All all of these functions have to be handled with care, since they are neither complete nor well tested. They are to be seen as a proof-of-concept and may additionally in some cases of practical help.

Any interest on further improvement of these extensions should be communicated towards the developer and user community.

8.2.2.1. importAPR

This command offer to load an ESRI®ArcView®project file (suffix .apr) and convert it for use within Thuban. After selecting a apr-file to load, a list will be presented that offers to select one of the views of the apr-file, provided there is more than one. Furthermore, the Session Info-Tree is extended with a complete representation of the parsed apr-file.

The legend of Thuban does not yet cover all of the elements as supported by the legend of ArcView® Therefore, the Thuban map will look different. Furthermore, the apr-format is a proprietary format, not openly documented. Therefore, the interpretation is partly based on reverse engeneering and good guessing.

The file-paths within the apr-file may not fit and potentially are subject to fix in the apr-file. You can do this applying any text editor. The paths are either absolute or relative from where Thuban has been started.

A sample for the Iceland data is included as `Extensions/importAPR/samples/iceland.apr`. The file-paths are relative from the Thuban main directory.

8.3. Writing simple extensions

Writing an extension for Thuban basically means to implement the extra functionality in Python with all of the Thuban classes, methods and variables available.

All classes and their methods are documented in the source code (see their doc-strings). Here is an example from Thuban/Model/layer.py that describes some of the methods of a Layer object:

```python
class BaseLayer(TitledObject, Modifiable):

    """Base class for the layers."""

    def __init__(self, title, visible = True, projection = None):
        """Initialize the layer.

        title -- the title
        visible -- boolean. If true the layer is visible.
        """
        TitledObject.__init__(self, title)
        Modifiable.__init__(self)
        self.visible = visible
        self.projection = projection

    def Visible(self):
        """Return true if layer is visible"""
        return self.visible

    def SetVisible(self, visible):
        """Set the layer's visibility."""
        self.visible = visible
        self.issue(LAYER_VISIBILITY_CHANGED, self)

    def HasClassification(self):
        """Determine if this layer support classifications."""
...
```

This example intends to give you an impression of the source-code-level documentation. You have to make yourself familiar with the Python programming language to understand some special code elements.

8.3.1. hello_world.py

Traditionally, the first example should welcome the world. Most of the code handles the frame for integrating a menu item into Thuban while the actual raising of a message is done in a single line.

```
# Copyright (C) 2003 by Intevation GmbH
# Authors:
# Jan-Oliver Wagner <jan@intevation.de>
#
# This program is free software under the GPL (>=v2)
# Read the file COPYING coming with Thuban for details.

"""
Extend Thuban with a sample Hello World to demonstrate simple
extensions.
"""


__version__ = '$Revision: 1.23 $'

# use _() already now for all strings that may later be translated
from Thuban import _

# Thuban has named commands which can be registered in the central
# instance registry.
from Thuban.UI.command import registry, Command

# The instance of the main menu of the Thuban application
# See Thuban/UI/menu.py for the API of the Menu class
from Thuban.UI.mainwindow import main_menu

def hello_world_dialog(context):
    """Just raise a simple dialog to greet the world.

    context -- The Thuban context.
    """
    context.mainwindow.RunMessageBox(_('Hello World'), _('Hello World!'))


# create a new command and register it
registry.Add(Command('hello_world', _('Hello World'), hello_world_dialog,
                     helptext = _('Welcome everyone on this planet')))

# find the extensions menu (create it anew if not found)
extensions_menu = main_menu.find_menu('extensions')
if extensions_menu is None:
    extensions_menu = main_menu.InsertMenu('extensions', _('E&xtensions'))

# finally bind the new command with an entry in the extensions menu
extensions_menu.InsertItem('hello_world')
```

8.3.2. Registering a Command

Mainly, our new function has to be registered to the Thuban framework in order to connect it to the menu. A registered command can also be connected to e.g. a toolbar button.

The instances and classes for this are imported at the beginning. Any code not inside a method or class is directly executed when the source-code module is imported. Therefore, the second part of this example consist of the plain statements to create a new Command and to add it to the menu.

By convention, it looks for a menu registered as "extensions" to insert the new command. If it does not exist yet, it gets created. It is advisable to copy this code for any of your extensions.

8.3.3. The Thuban context

A registered command that is called, always receives the Thuban context. This instance provides our method with hook references to all important components of the Thuban application.

In the example hello_world.py, our function uses the mainwindow component which offers a method to raise a message dialog. In total there are three hooks:

- application: This object is the instance of the Thuban Application class. Except maybe for loading or savinf sessions, you will not need this object for a simple extension. See Thuban/UI/application.py for the API.
- session: The instance of the current session. It manages the sessions' map and tables. You can set and remove the map or tables. In may also get the map object. However, you should know that internally it is already prepared to handle many maps. Therfore, currently you would always receive a list with exactlty one element. In the future, if there are more than one map, you will not know which one is the currently display one and therefore you should use the mainwindow as hook to find the currently displayed map. See Thuban/Model/session.py for the API.
- mainwindow: The mainwindow object is central to manage various GUI things such as the Legend sub-window. Most notably, you get access to the canvas which is the window part where the map is drawn. The canvas knows, which map it currently draws and therefore you get the current map via context.mainwindow.canvas.Map(). See Thuban/UI/mainwindow.py for the API.

Chapter 9. Trouble Shooting

Here are a few problems that users have encountered when first using Thuban.

- After adding two or more layers nothing is drawn in the map window.

  This is probably because the layers have different projections. Projections must be set on all layers and on the map itself if the layers' projections are different.

- Thuban crashes on startup with the error `NameError: global name 'False' is not defined.`

  `True` and `False` were only introduced in Python 2.2.1. Thuban depends on at least Python 2.2.1.

- After compiling Thuban, Thuban crashes with an error similar to `ImportError: /usr/local//lib/thuban/Thuban/../Lib/wxproj.so: undefined symbol: __gxx_personality_v0`

  Thuban depends on the wxWindows library. If Thuban is compiled with an incompatible version of the compiler than wxWindows was compiled with this error may occur. Try compiling with a different version of the compiler.

If an error occurs Thuban will display a dialog indicating the error before closing. The text should be copied and reported to the Intevation bugtracker (http://thuban.intevation.org/bugtracker.html). More information about the system is available from Help—→About box. This should also be included in the bug report.

Figure 9-1. Error Dialog


```
Thuban: Internal Error                                    ×

An unhandled exception occurred:

(please report to http://thuban.intevation.org/bugtracker.html)


Traceback (most recent call last):
  File "thuban.py", line 26, in ?
    Thuban.UI.main.main()
  File "Thuban\UI\main.py", line 26, in main
    app = ThubanApplication(0)
  File "C:\wx-versions\wxPython-2.4.0.6\wxPython\wx.py", line 1821, in __init__
    _wxStart(self.OnInit)
  File "Thuban\UI\application.py", line 62, in OnInit
    self.create_session()
  File "Thuban\UI\application.py", line 190, in create_session
    self.SetSession(create_empty_session())
  File "Thuban\Model\session.py", line 384, in create_empty_session

                         OK
```

Appendix A. Supported Data Sources

Shapefile

> The Shapefile format has become a standard format for saving geographic vector information. It supports polygons, lines, and points. Technical Specification. (http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf)

dBase file

> dBase files are used to store the attributes for each layer. This is closely associated with the Shapefile format. For detailed specifications on the correct format of a dBase file used with Thuban please see the Technical Specification for the Shapefile format above.

PostGIS

> PostGIS adds support for geographic objects to the PostgreSQL object-relational database. Different layer types (as for Shapefiles) are supported. PostGIS Homepage (http://postgis.refractions.net)

Raster files

> Binding the GDAL library Thuban supports numerous raster file formats, see GDAL format list (http://www.remotesensing.org/gdal/formats_list.html) for details.

> Most commonly used is the *TIFF/GeoTIFF* format: Raster maps are provided as TIFF images, with an additional "world file" storing the geographic reference (usually with an extension ".tfw").

Appendix B. Working with PostGIS

This section focusses on the use of PostGIS in the Thuban framework. For installation and maintenance of spatial databases we refer to the PostGIS Homepage (http://postgis.refractions.net). The Thuban PostGIS support requires the psycopg module (http://initd.org/software/psycopg).

Working with PostGIS Databases is seperated into two steps:

- Opening a Database Connection
- Loading a Data Layer

B.1. Opening a Database Connection

Before a data layer can be loaded from a PostGIS database a connection with the database has to be established. Session⟶Database Connections ... opens a dialog for database connection management. In the dialog new connections can be added and existing ones can be removed. Removing a database connection is not possible if the map still displays a layer provided by this database connection.
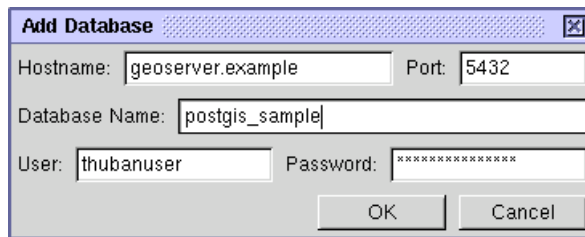
Figure B-1. Database Management Dialog
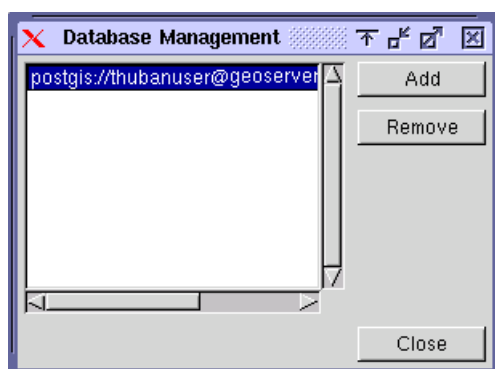
To add a new database connection to the session a dialog is opened to specify the relevant connection data. Enter all data relevant for your connection. If the connection fails the dialog remains open and provides some hints on the failure.

Figure B-2. Add Database Dialog

It is important to note that information on database connections are also stored with the session. Passwords are NOT stored. If you load a session with database connections you are asked to enter these passwords again where required.

B.2. Loading a Data Layer

Data layers as part of a map are loaded with the Map⟶Add Database Layer ... menu item. A dialog is raised displaying two choice lists. In the left list all connected databases are shown. Highlighting one of these and issuing a retrieval results in a list of available layer tables from that database. After selection of a layer the dialog is closed.

Figure B-3. Add Database Dialog
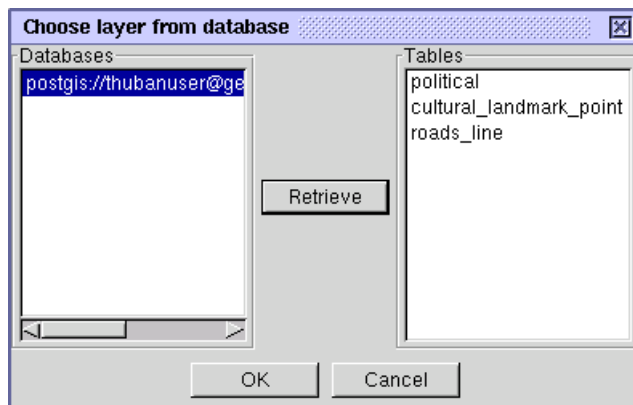
Appendix C. Supported Projections

The following types of projections are directly support by Thuban. The specific values for each are provided by the user to create custom projections. Thuban comes with predefined projections which are available through the Projections dialog.

- Geographic
  - `Ellipsoid`
  - `Source Data`: either Degrees or Radians

- Lambert Conic Conformal
  - `Ellipsoid`
  - `Latitude of 1st standard parallel`
  - `Latitude of 2nd standard parallel`
  - `Central Meridian`
  - `Latitude of Origin`
  - `False Easting` (meters)
  - `False Northing` (meters)

- Transverse Mercator
  - `Ellipsoid`
  - `Latitude` of origin
  - `Longitude` at central meridian
  - `Scale Factor` at central meridian
  - `False Easting` (meters)
  - `False Northing` (meters)

- Universal Transverse Mercator
  - `Ellipsoid`
  - `Zone` (can be guessed appling the Propose button)
  - `Southern Hemisphere` flag

Thuban comes with a sample set of map projections for various European countries. Apart from the basic projection they differ especially in their parameterization:

- Belgium Datum 1972 (Lambert Conic Conformal)

- Gauss-Boaga Zone 1 (Italy, Transverse Mercartor)

- Gauss-Krueger Zone 2 (Germany, Transverse Mercartor)

- Reseau Geodesique Francaise (France, Lambert Conic Conformal)

- UK National Grid (United Kingdom, Transverse Mercartor)

Thuban uses the comprehensive PROJ library for projections. PROJ provides more than the four commonly used projections described above. If needed Thuban can be easily extended to a new projection covered by PROJ.