



VYM – View Your Mind

Version 2.9.0

Usermanual

©Uwe Drechsel

March 9, 2023

Contents

1	Introduction	7
1.1	What is a VYM map?	7
1.2	Why should I use <i>maps</i> ? Time, Space and your Brain.	7
1.3	Where could I use a <i>map</i> ?	8
1.4	What you shouldn't do with a <i>map</i>	9
1.5	Internet Ressources	9
2	The Concept of the VYM application	10
2.1	The Mainwindow and its satellites	10
2.2	Dockable windows	11
2.3	Toolbars	12
2.4	Menus and Context menus	12
2.5	Maps	12
3	Mapeditor	13
3.1	Start a new map	13
3.2	Navigate through a map	13
3.3	Modify and move branches	16
3.4	Colours and Images - Using the right side of your brain	17
3.5	Design of map background and connecting links	20
3.6	Links to other documents and webpages	21
3.6.1	How to create an URL	21
3.6.2	How to create a vmlink	22
3.7	Multiple maps	22
3.8	Brainstorming	22
4	Noteeditor	23
4.1	States	23
4.2	Import and export notes	24
4.3	Edit and print note	24
4.4	RichText: Colours, paragraphs and formatted text	24
4.5	Fonts and how to switch them quickly	24

4.6	Find text	24
4.7	Paste text into note editor	25
5	Task editor	25
5.1	Creating tasks	25
5.2	Reminders - let a task sleep for a while	26
5.3	Priority of tasks	26
5.4	Filter tasks - keeping the overview	27
6	Slideeditor - presentations	27
7	Hello world - vym and other applications	28
7.1	Import	28
7.1.1	Mozilla Firefox bookmarks	28
7.1.2	Freemind and Freeplane	28
7.1.3	Mind Manager	29
7.1.4	Directory structure	29
7.2	Export	29
7.2.1	Last used format	29
7.2.2	Image	29
7.2.3	PDF	29
7.2.4	SVG	30
7.2.5	Open Office	30
7.2.6	HTML (Webpages)	31
7.2.7	A & O – Achievements and Objectives	31
7.2.8	ASCII	31
7.2.9	CSV	31
7.2.10	Taskjuggler	31
7.2.11	L ^A T _E X	31
7.2.12	Markdown	32
7.2.13	OrgMode	32
7.2.14	XML	32
7.2.15	Export a part of a map	32
7.3	Connect vym to the cloud	32

7.3.1	Confluence	33
7.3.2	JIRA	33
7.4	Connect vym using DBUS	33
8	Advanced usage	33
8.1	Quickly sorting branches or postponing actions	33
8.2	Properties of an object	34
8.3	Changing the history: Undo and Redo	35
8.4	Macros	35
8.5	Bookmarks	36
8.6	Associating images with a branch	36
8.7	Modifier Modes	37
8.8	Hide links of unselected objects	37
8.9	XLinks	38
8.10	Adding and removing branches	39
8.11	Adding a whole map or a part of a map	39
9	VYM on Mac OS X	39
9.1	Contextmenu and special keys	39
9.2	Viewing external links	40
A	VYM initialisation process and configuration	41
A.1	Settings menu	41
A.2	Configuration file	43
A.3	Path to ressources	43
A.4	Command line options	43
B	Scripts	44
B.1	Overview	44
B.2	Example scripts	45
B.2.1	Macro to create a rounded rectangle frame	45
B.2.2	Batch script to export all maps as images	45
B.2.3	Full scripting using ruby and DBUS	45
B.3	Available commands	46

C	Contributing to VYM	67
C.1	Getting help	67
C.2	How to report bugs	67
C.3	Compiling from the sources	68
C.3.1	Getting the sources	68
C.3.2	The Qt toolkit	68
C.3.3	Compiling VYM	68
C.4	VYM file format	69
C.5	New features	69
C.6	New languages support	69
C.7	New export/import filters	70

Credits

Many people have sent me their feedback and ideas, and all of that has helped a lot to make VYM better. Thanks to all of you!

For this manual I would like to send some special thanks to

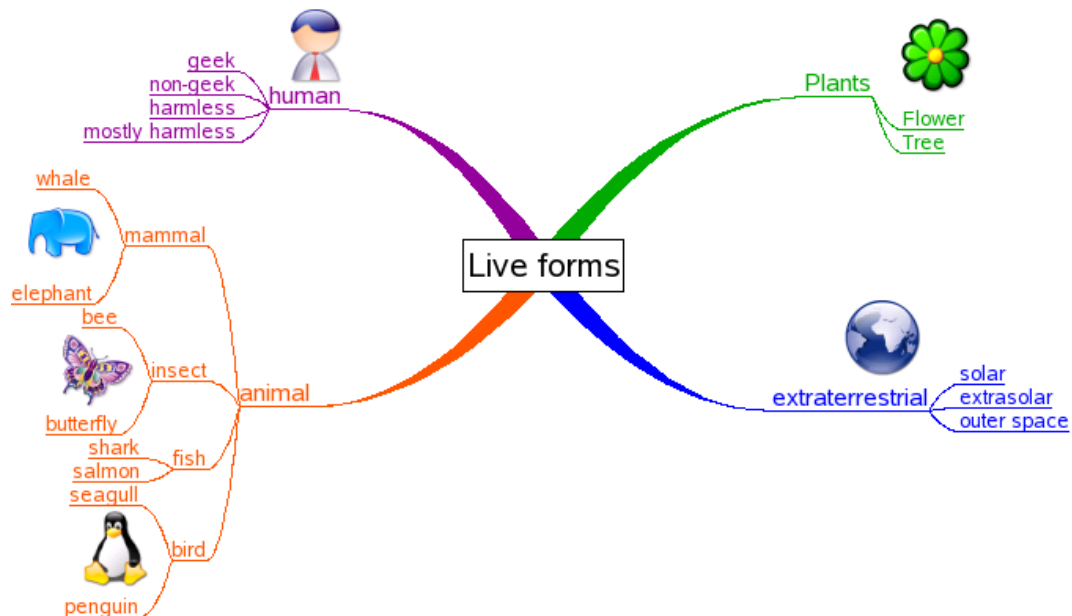
- *Peter Adamson* for lots of feedback and proofreading of my far from perfect english
- The team of *AClibre (Academia y Conocimiento Libre)* in Colombia for their initial translation of the manual to spanish:

Encargado	Actividad
<ul style="list-style-type: none">– Vanessa Carolina Gutiérrez Sanchez– Erika Tatiana Luque Melo– Jeffrey Steve Borbón Sanabria– John Edisson Ortiz Román	<ul style="list-style-type: none">– Traducciónl– Revisión y correcciones varias– Estructuración y exporte– Revisión y correcciones varias

1 Introduction

1.1 What is a VYM map?

A VYM map (abbreviated below as *map*) is a tree-like structure:



Such maps can be drawn by hand on a sheet of paper or flip chart and help to structure your thoughts. While a tree like structure like the illustration above can be drawn manually VYM offers much more features to work with such maps. VYM is not just another drawing software application, but a tool to store and modify information in an intuitive way. For example you can reorder parts of the map by pressing a key or add various pieces of information like a complete email by a simple mouse click.

Once you have finished collecting and organising your ideas, you can easily generate a variety of outputs including for example a presentation in Open Office based on a *map*.

Hint: You find the map shown above and others by clicking

Help → Open vym examples

in the menu bar.

!

1.2 Why should I use *maps*? Time, Space and your Brain.

Space

A *map* can concentrate very complex content in a small space such as a piece of paper. It helps to use both sides of your brain: the logical side and also your creative side (e.g. by using pictures, colours and keywords in a map, often called *anchors*). It is a technique to help organize the way you think and stimulate your creativity: It can help you by developing, sorting and helping to memorise your ideas.

Time

Because you just use keywords and drawings, it is much faster than good old fashioned 'notes'. Your brain memorizes things by associating them with other things – a *map* makes use of those connections and stimulates new associations.

Your Brain

In 1960 Prof. ROGER SPERRY discovered that both hemispheres of the human brain undertake different tasks (of course both of them basically *can* do the same):

Left side	Right side
<ul style="list-style-type: none">• verbal speech and writing• numbers• logical thinking• analysing and details• science• linear thinking• concept of time	<ul style="list-style-type: none">• body language• visual thinking, day dreams• intuition and emotion• overview of things• creativity• art, music, dancing• non-linear thinking, connecting things• spatial awareness

In our science oriented western society we have learned to mainly rely on our left side of the brain, the "rational" one. In other cultures, such as the native americans and other "old" cultures, the right side is much more important. *Map* are just one way to stimulate the other side and make use of additional resources we all have.

1.3 Where could I use a *map*?

Here are some examples, how you can use those *maps*

- to prepare articles, papers, books, talks, ...
- to sort complex data
- to memorize facts, peoples names, vocabulary, ...
- to sort emails, files and bookmarks on your computer
- to moderate conferences

- to brainstorm solutions to problems
- to record the tasks when planning a project

1.4 What you shouldn't do with a *map*...

A *map* drawn by somebody shows the way that the author thinks. There is no question of right or wrong in the way it is drawn, so there is no way to criticise it. "It is, what it is" (F. LEHMANN). The tool will be of considerable use to the author and only very limited use to anyone else.

However, when groups share in creating a *map* all of the group will benefit from its use. An example of such use is when a Tutor develops a *map* with a group of students during instruction. Another group use is when a Project leader gathers a group of specialists to help *map* the tasks that will be required to deliver a project.

1.5 Internet Ressources

There are a few tutorial vides on youtube:

- Youtube: <https://www.youtube.com/c/ViewYourMind>

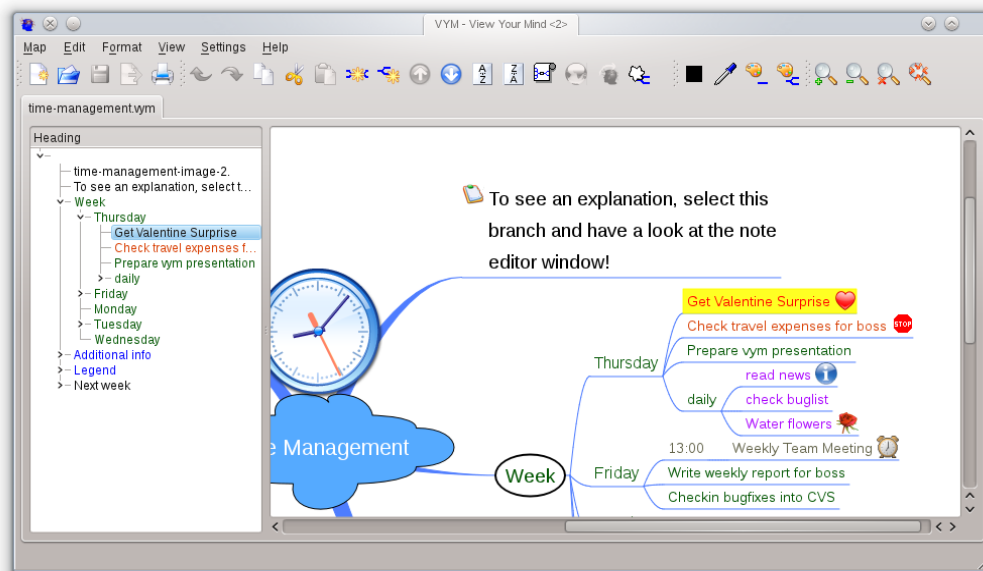
A good starting point to learn more about Mindmaps in general is Wikipedia:

- English: http://en.wikipedia.org/wiki/Mind_map
- German: <http://de.wikipedia.org/wiki/Mindmap>

2 The Concept of the VYM application

2.1 The Mainwindow and its satellites

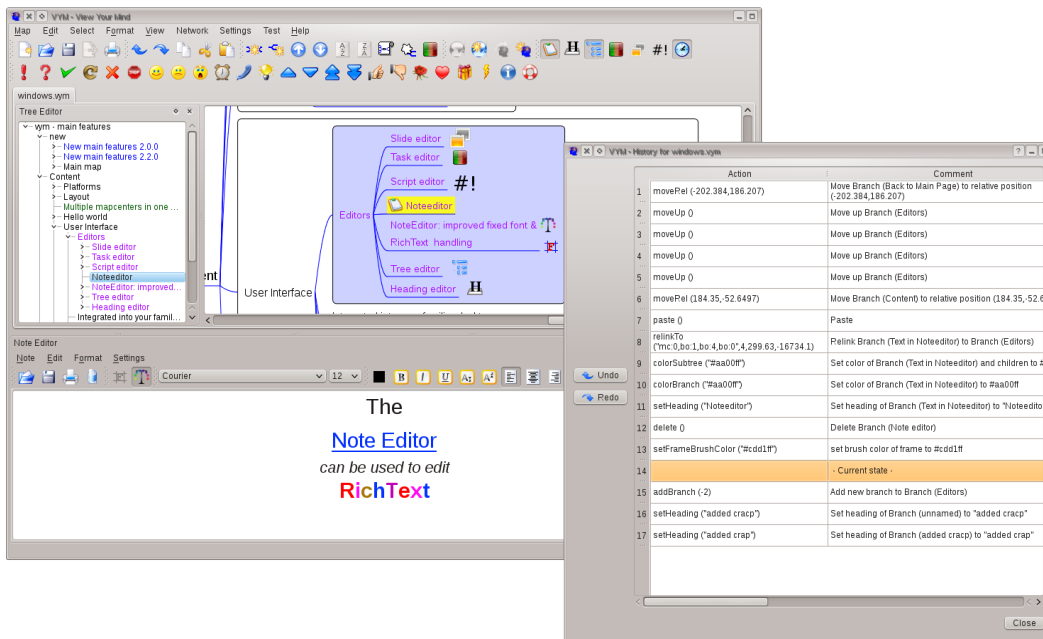
VYM comes with several windows, the central one is called *mainwindow*. It contains one or more tabs, each of these tabs has a *mapeditor* and optionally a *tree editor*, representing a different view of the same data:



The (currently visible) main areas of the mainwindow are the *tree editor* to the left and the *mapeditor* to the right. Note that both editors represent the same data and share their selection: Both show the *heading* "Get valentine surprise".








More windows, each having a special purpose, can be opened and arranged around or even in the mainwindow. These extra windows are called *satellites*¹. The next image below shows the *mainwindow* together with the *history window*. The mainwindow itself currently has three parts: *treeeditor*, *mapeditor* and at the bottom the *noteeditor*

¹The advantage of having separate window instead of integrating them in a combined workspace is flexibility in arranging the windows. For example I usually have the *noteeditor* "behind" the *mapeditor*. On Linux my windowmanager (KDE) allows me to enter text into a small visible corner of the *noteeditor* without clicking the mouse button to move focus to the window, so that it receives input from keyboard. I just push the mouse around to set the window focus, a concept which is useful also working with <http://www.gimp.org>.



Most of the time you will work in the *mapeditor* by just adding new branches, moving around and reordering them. The various ways to do this will be explained in 3. You can store additional information e.g. the content of a email easily in a *branch*: Just type or copy&paste it into the *noteeditor*. Working with notes is explained in 4

Here is a list of the available satellite windows, the *dockable* feature is explained in next section 2.2:

- Branch Property Window (see section 8.2)
-  Find window to search for text (dockable)
-  Heading editor (dockable, features are the same as in the noteeditor, see section 4)
-  Historywindow (see section 8.3)
-  Noteeditor (dockable, see section 4)
-  Scripteditor (dockable, see section B.1)
-  Slideeditor (dockable, see section 6)
-  Taskeditor (dockable, see section 5)

2.2 Dockable windows

Beginning with VYM 1.13.0 some of the windows may be docked and become a part of an existing window. Or the other way round: Parts of a window may be "undocked" and become their own independent windows or can be placed in a different position within the mainwindow. Example: Toolbar above the map with flags can be placed below or besides of the map, or even float independantly.

2.3 Toolbars

The toolbars in the mainwindows give quick access to many functions and also display the state of selected objects in the map. For example a branch may show certain *flags*, the corresponding flags are also set in the toolbar.

Hint: Toolbars are *dockable* (see also section 2.2): You can reposition all toolbars by simply grabbing and dragging them with the toolbar handle to a new position. For example you can move the flags-toolbar from its original horizontal position on top of the mapeditor to a vertical position on the right side. Or just insert it again at its original position. Also hiding some of the toolbars is possible by right-clicking on the toolbar handle.

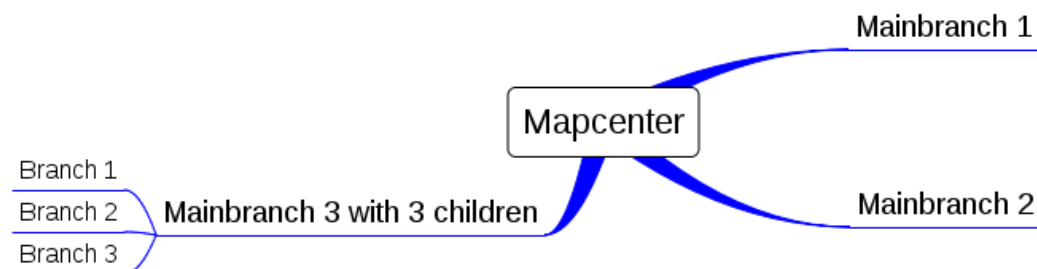
!

2.4 Menus and Context menus

At the top of each window you will find the menubar. The options provided there are similar to those you are probably used to from other applications. Note that many (and even more) options are available via *context menus*. Those are available if you right-click onto an object in a map (on Mac OS X Command-Click).

2.5 Maps

The *map* has one or more *mapcenters*. Each mapcenter has *branches* radiating out from the centre just like the trunk of a tree. Each branch in turn may have branches again.



We will call a branch directly connected to the mapcenter a *mainbranch*, because it determines the position of all its child branches.

If you want to have more than one mapcenter, open the context menu by right-clicking onto the background of a map and select "Add mapcenter". Or just press [C]. The mapcenter and the branches all have a *heading*. This is the text you see in the mapeditor. Usually it should just be one or a few key words, so that one can easily keep track of the whole map. There are several ways to quickly edit the heading, see 3.3.

In the toolbar above the mapeditor you see various symbols.



These are called *flags* and can be used to mark branches in the *map*, e.g. if something is important or questionable. There are also more flags set by VYM automatically to show additional information, e.g. when a note is attached to a particular branch.

By default some of these flags are set exclusively e.g. when the "thumb-up" flag is set, then the "thumb down" is reset and vice versa. You can change this default behaviour in the settings menu (see [A.1](#)).

3 Mapeditor

3.1 Start a new map

After VYM is started you will see the *mainwindow* with two parts: the *mapeditor* and the *tree editor*. Usually you will work in both windows, but at the moment we will just need the mapeditor.

Select the mapcenter "New map" in the middle of the mapeditor by left-clicking with the mouse. It will be highlighted yellow to show that is selected. There are several ways to add a new branch to the center:

- Using the mouse: Open the context menu by clicking with the right mouse button (CTRL-Click on Mac) onto the mapcenter and choose Add → Add branch as child
- Press [Ins] or [A]

A new branch will appear and you will be able to type the heading of the branch. Finish adding the new branch by pressing [Enter]. Sometimes it comes in handy to be able to add a new branch above or below the current one.

- Use [Shift-A] to add a branch above the selected one or...
- [Ctrl-A] to add one below.

It is also possible to add a branch in such a way, that the current selection becomes the child of the new branch, which is like inserting it *before* the selection. This can be done using the context menu.

Hint: To delete a branch press [CTRL-X]. If enabled in the Settings menu (see A.1), you can also use the [Del] key.

!

3.2 Navigate through a map

Select branches

To select branches you can use the left button of your mouse or also the arrow keys. Depending on the *orientation* of a branch tap [←] or [→] to move nearer to the mapcenter or deeper down into the branches. Within a set of branches, let's call them a *subtree*, you

can use [PgUp] and [PgDn] to go up and down. You can also use [Home] and [End] to select the first and last branch.

You can also select branches in the *tree editor*. Once this editor is active, you can also use [↑] and [↓], which have a slightly different meaning than in mapeditor: don't forget to click back into mapeditor to continue working there.

There is also a *selection history*: Use [CTRL-I] and [CTRL-O] to go back to previous or later selections.

Panning the view of a map

While adding more and more branches the size of the map may become larger than the mapeditor window. You can use the scrollbars on the right and the bottom of your mapeditor window to scroll the view up or down or left or right. It is easier to just scroll using the left mouse button: Click anywhere on the *canvas* itself. Choose an empty space somewhere between the branches. The mouse pointer will change from an arrow to a hand, now move or drag the visible map to show the desired part.

If you select branches using the arrow keys, the map will scroll to ensure that the selected branch is always visible.

Zooming the view of a map

Working with huge maps, the *zoom*-function comes in handy: You can use

- from the menu: View →Zoom in, View →Zoom out, View →reset Zoom.
- the toolbar buttons

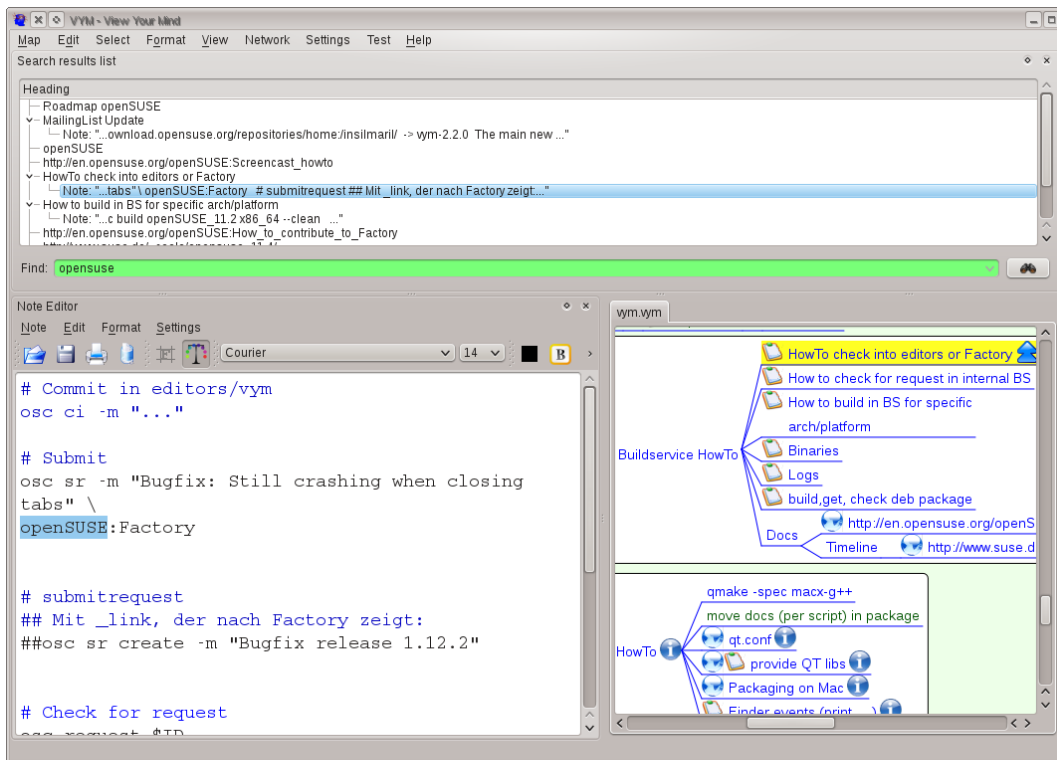


Clicking the crossed magnifying lens icon will reset the zoomed view to its original size. The keyboard shortcuts for zooming are [+], [-]. [,] resets the zoom and [.] centers view on selection.

Alternatively you can *zoom using the mouse*: Press [CTRL] while using the scrollwheel. With [CTRL] and middle-mouse you reset the zoom.

Find Function

Choose Edit →Find or just press [CTRL+F] to open the Findwidget. The image below shows the findwidget above the notepad and the mapeditor:



The text you enter here will be searched in all the branch headings and also in the associated notes. Just click on one of the results shown to select the found heading or an occurrence in a note. In above example we searched for "opensuse", which had a number of occurrences in this map, e.g. in the note of "HowTo check into in editors on Factory" branch.

Press [CTRL+F] again to hide the widget once it's no longer needed. Or you could also undock it (see 2.2).

Keep the overview – scroll a part of the map

A very big subtree of a map e.g. a branch with hundreds of child branches would make it very hard to keep an overview over the whole map. You can hide all the children of a branch by *scrolling* it – in other software also called *folding*. Think of the whole subtree as painted onto a broadsheet newspaper. You can scroll or fold the paper to a small roll, leaving just the headline visible.

To scroll or unscroll a branch and its children,

- press the [S]
- press the middle-mouse button or
- choose the scroll icon from the toolbar.

If you select parts of a scrolled branch e.g. using the find function or by using the arrow-keys, it will unscroll temporary. This is shown as a scroll with a little hour glass. If the temporary unscrolled part is no longer needed, it will be hidden again automatically. It is also possible to unscroll all branches using "Edit→Unscroll all scrolled branches".

You can also hide parts of the map while exporting it e.g. to a webpage or a presentation, see 7.2 for details.

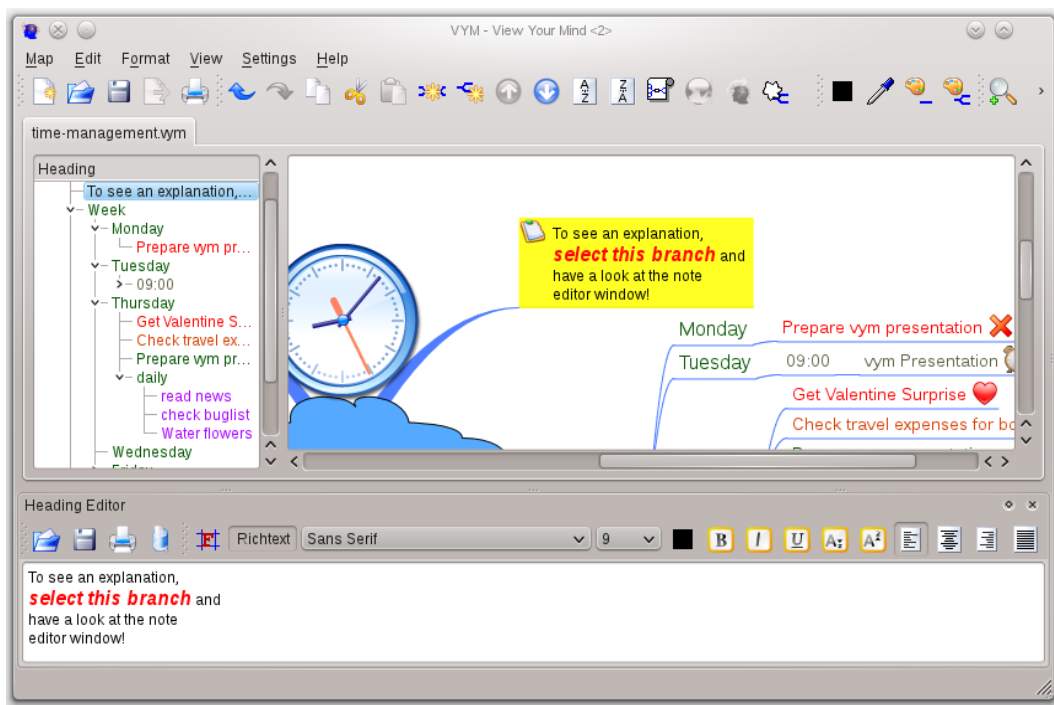
3.3 Modify and move branches

Modify the heading

You can edit the heading by selecting the branch and then

- pressing [Enter]
- double-clicking with left mouse.

Just type the new heading (or edit the old one) and press [Enter]. You can also open the *heading editor* by pressing [E] or clicking



in the toolbar. The *heading editor* allows you to enter richtext, just like the *note editor* does for notes (see section 4).

Move a branch

The easiest way to move a branch is to select it with left-mouse and drag it to the destination while keeping the mouse button pressed. Depending on the branch it will be

- moved to the destination or
- *linked* to a new *parent* (mapcenter or branch)

If you drag the branch over another one or over the mapcenter, you will notice that the link connecting it to the old parent will be changed to lead to the new parent which is now under your mousepointer. If you release the button now, the branch will be relinked.

If you release the button in the middle of nowhere, the result will depend on the type of branch you are releasing:

- A mainbranch is directly connected to the mapcenter. It will stay on its new position.
- An ordinary branch will "snap" back to its original position ².

Thus you can easily rearrange the layout of the mainbranches to avoid overlapping of their subtrees. There is another convenient way to move branches, especially if you want to *reorder* a subtree: You can move a branch up or down in a subtree by

- pressing [↑] and [↓]
- selecting Edit → Move branch
- clicking on the toolbar buttons:



There is yet another way to move branches: If you press [Shift] or [Ctrl] while moving with the mouse, the branch will be added above or below the one the mouse pointer is over. This can also be used to reorder branches in a map.

Another special way to move or select branches are *targets*, see 8.1. Branches with targets use this flag:



3.4 Colours and Images - Using the right side of your brain

Change colour of a heading

You can also use colours to add more information to a map, e.g. use red, green and more colours to prioritize tasks. Again you can

- use the menu and choose e.g Format → Set Color
- use the toolbar



²In newer versions of vym the "snap back" is animated. Animation can be configured, see section A.1

The first button (black in the graphic above) shows the current colour. Clicking on it let's you choose another colour. You can also "pick" another colour by selecting a branch with the desired colour and using the "pick colour" button. Both of the icons showing a palette actually apply the current colour to the selected branch. While the first one just colours the heading of the selection, the last one also colours all the children of the selected branch.

A very useful function is the *pick color modifier* of the mouse: Select the branch which should get the new colour, then press [Shift] and simultaneously click with left-mouse on another branch to copy its colour to the currently selected subtree. If you only want to color the branch, not the whole subtree, press [Shift]+[Ctrl] simultaneously.

Use flags

VYM provides various flags. They are usually displayed in the toolbar on top of the mapeditor window. (Note: Like all toolbars you can also move them to the left or the right side of the window or even detach them. Just grab the very left "dotted" part of the toolbar with your left-mouse button.)



If you have a branch selected, you can set any number of flags by clicking them in the toolbar. The toolbar buttons change their state and always reflect the flags set in the selected branch. So, to remove a flag from a branch, select the branch and then click the highlighted flag on the toolbar.

VYM uses three kinds of flags:

- *System Flags* are set by VYM to indicate e.g. that there is additional information in a note (more on this in [4](#)) or that there is a task associated (see the [5](#))
- *Standard Flags*
- *User Flags*

Standard flags can be toggled by clicking in the toolbar



User flags are similar, but can be added to their own toolbar when clicking the edit icon

Images

The easiest way to add an image to a branch is by dragging it e.g. from a webbrowser to the mapeditor while a branch is selected there.

You can also add an image to a branch by opening the context menu of the branch. Right click the selected branch, choose "Add Image". A dialog window enables you choose the

image to load. ³ While an image is selected in the dialog, a preview of the image is displayed. It is also possible to select multiple images.

You can position the image anywhere you want, just drag it with left mouse. To relink it to another branch, press [Shift] while moving it. To delete it, press [Del].

If you right-click onto an image, a context menu will open which let's you first choose one of several image formats. Then a file dialog opens to save the image.

Hint: This is used to "export" the image as separate file. As part of the map it always will be saved anyway in the map itself! You can also cut and copy images.

!

Note that it is not possible to add objects to an image, which would make working with the map much more complex if e.g. images could be linked to images.

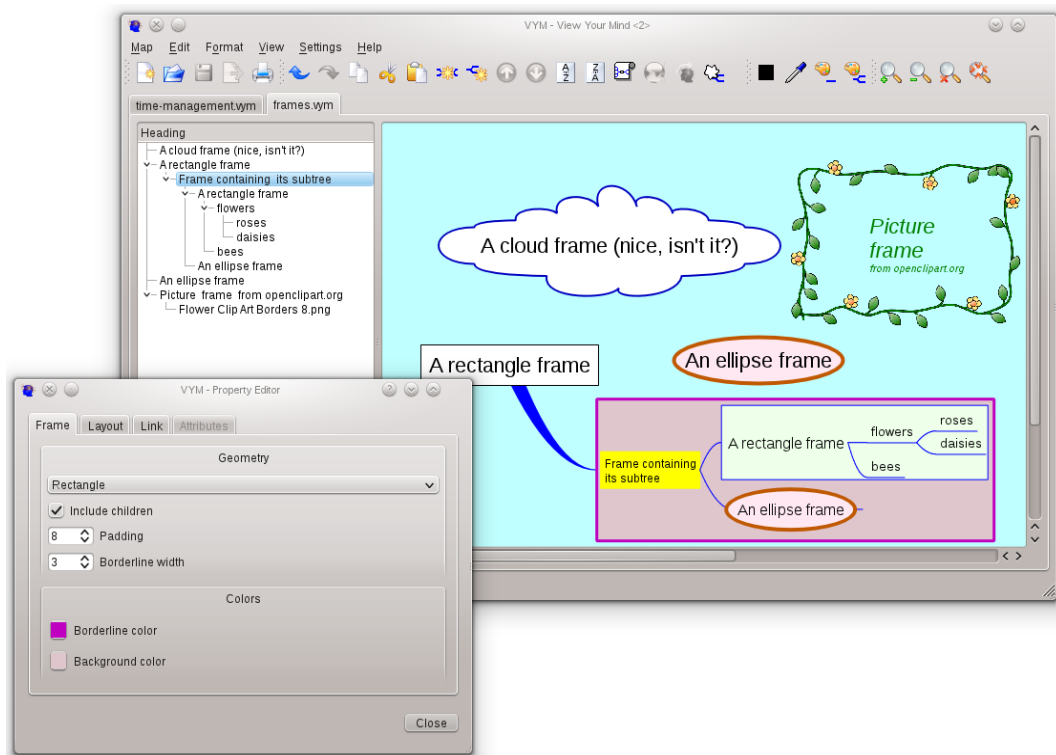
The option "**Use for export**" controls the output of exports e.g. to HTML: If set to no, the image won't appear in the *text* part of the output. This is useful for large images or if images are used as a kind of frame e.g. the famous cloud symbol around a part of the map. Those shouldn't appear in the middle of the text.

Resizing images has been added to /vym in version 1.13.20. Later versions will include more functionality like changing its z-value (put it into background) etc.

Frames

Various types of frame can be added to a branch in the *property window* (see 8.2), e.g. ellipses, rectangle and cloud. The frames can optionally also include the children of the the framed branch, thus allowing even to "put boxes of contents into other boxes". Alternatively, you can use use images as frames:

³Supported image types are: PNG, BMP, XBM, XPM and PNM. It may also support JPEG, MNG and GIF, if specially configured during compilation (as done when VYM is part of SUSE LINUX).



3.5 Design of map background and connecting links

The design of the background of a map and also of the links connecting various parts of the map can be changed by

- Selecting Format from the menu
- Right clicking on the canvas, which will open a context menu

Background

The colour is set (and also displayed) as "Set background colour". Alternatively you can set a background image, though this is not recommended in general. Working on the map becomes slow and the image currently cannot be positioned freely.

Link colour

Links connecting branches can be coloured in one of two ways:

- use the same colour for the heading and for the branch link line.
- use *one* colour for all links and choose different colours for the branch headings text. The default colour for branch link lines is blue.

The latter can be set with "Set link colour". Check or uncheck the "Use colour of heading for link" option to toggle between the two designs for your map.

Link style

VYM offers four different styles for the appearances of links:

- Line
- Parabel
- Thick Line
- Thick Parabel

The "thick" styles only apply to links starting at the mapcenter, link lines for the rest of the map are always painted "thin".

3.6 Links to other documents and webpages

VYM supports two kind of external links:

- Documents, viewed with an external webbrowser. Examples are `.pdf` and `.html` files. References are called *URLs* and marked with the globe flag:



- VYM maps, viewed in VYM itself. References are called a *vymlinks* and marked with the VYM flag:



3.6.1 How to create an URL

Use one of the following:

- Drag and drop URL from a webbrowser
- Use the "URLs and vymlinks toolbar:



- Keyboard shortcut: Press [U] or right-click onto a branch to open the contextmenu then choose "References→Edit URL". If you want to use a file dialog to conveniently choose a local file you can use [U]. Also drag and drop from a browser can be used to create a new branch with an URL.
- Context menu: Right click onto branch, in the context menu there is also an option to open all URLs found in the selected subtree of the map. That's useful to simultaneously open a collection of URLs in the webbrowser, especially if the browser can open them in tabs.

After an URL was entered, a little globe will appear in the branch. By clicking on the globe in the toolbar or the context menu an external browser⁴ will be launched. For more

⁴The browser can be changed in the Settings Menu (see A.1).

information on working with bookmarks and webbrowsers see section 8.5.

If your VYM installation supports accessing external tools like JIRA or Confluence, VYM might update the branch after retrieving information from the tool, e.g. replacing the complete URL of a Confluence by its concrete page name.

3.6.2 How to create a vmlink

Creation is possible both from the "URLs and vmlinks" toolbar or from the context menu of a branch:



A file dialog opens where you can choose the map. Clicking this flag beside the branch heading, in the toolbar or in the context menu of a branch will open the map in another tab (see 3.7 for working with multiple maps). To delete an existing link, just right click the branch and select "Delete VYM link".

Hint: Open a linked map in background by pressing [Ctrl] while clicking on the icon in the map

!

In the context menu there is also an option to open all vmlinks found in the selected subtree of the map. That's useful to simultaneously open a collection of related maps⁵.

3.7 Multiple maps

You can work on multiple maps at the same time. Each new map is opened in another *tab*. The available tabs are shown just above the mapeditor. You can use the normal cut/copy/paste functions to copy data from one map to another.

3.8 Brainstorming

When brainstorming you collect quickly a number of thoughts or ideas, without sorting, discussing or otherwise spending any amount of time on a particular idea.

VYM helps you to quickly create mapcenter, either by pressing [C] or clicking the



If you want to place the new mapcenter at a specific position, you could also open the context menu by right-clicking on the background and selecting "Add mapcenter".

Once you are done adding new items, you can start to sort them and arrange them by moving and relinking them to create a new map.

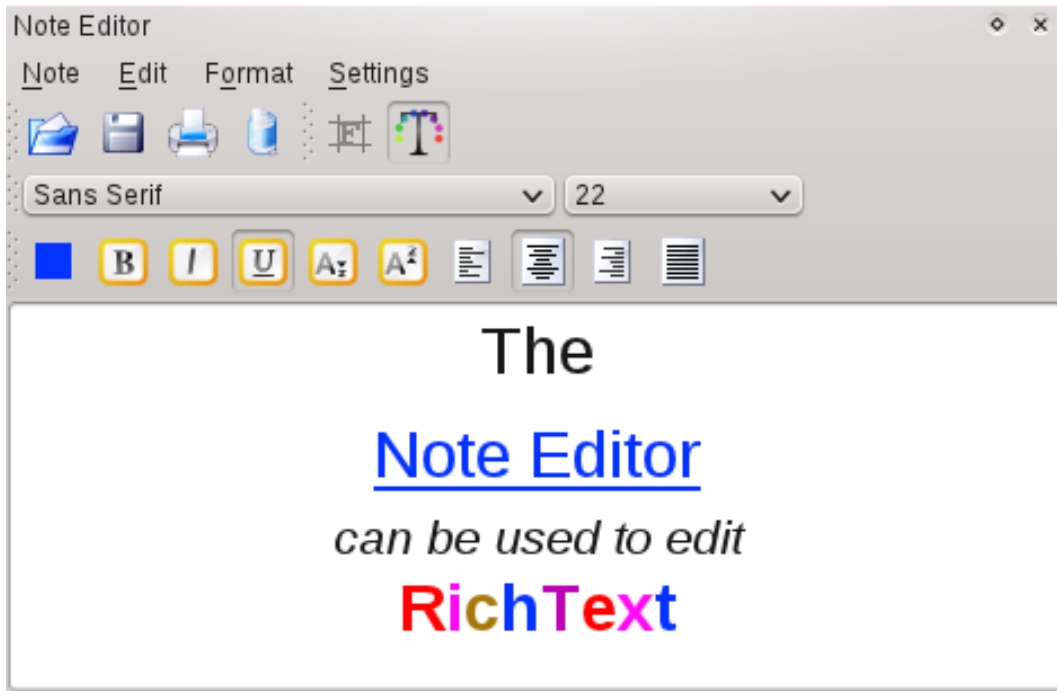
⁵Technical note: Internally VYM uses absolute paths, to avoid opening several tabs containing the same map. When a map is saved, this path is converted to a relative one (e.g. `/home/user/vym.map` might become `./vym.map`). This makes it fairly easy to use multiple maps on different computers or export them to HTML in future.

Hint: If you have enabled "Automatic Layout" in the Settings menu, the new parts will move around to avoid overlapping

!

4 Noteeditor

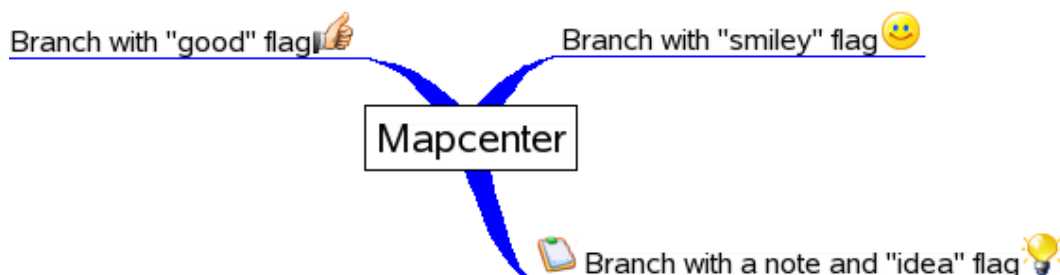
If you want to attach more text to a branch e.g. a complete email, a cooking recipe, or the whole source code of a software project, you can use the noteeditor. (The *Headingeditor* has the same features.)



This editor displays text associated with a branch selected in the mapeditor. The noteeditor shows different background colours depending on whether text is associated with a selected branch.

4.1 States

Before you can type or paste text into it, you have to select a branch in the mapeditor. In the mapeditor a little "notepad" flag will appear next to the heading of the branch, once you have entered some text in the noteeditor. This is illustrated in the lower branch on the right hand side:



4.2 Import and export notes

The note is always saved automatically within the VYM map itself. Nevertheless sometimes it is nice to import a note from an external file or write it. In the Note Editor use "File→ Import" and "File→ Export" to do so.

4.3 Edit and print note

Editing works like in any simple texteditor, including undo and redo functions. You can delete the complete note by clicking the trashcan. Only the note itself is printed by clicking the printer icon.

4.4 RichText: Colours, paragraphs and formatted text

Notes and also the headings of branches can either use a default font or all text attributes available in RichText, like bold, italic, colors, etc. To enable the latter, click the RichText button:



4.5 Fonts and how to switch them quickly

If you just want to do quick notes and don't need fully formatted RichText as mentioned above, you still can select either a fixed font width font or a variable width font by clicking



The fixed font is usually used for emails, source code etc. while the variable font is used for simple notes, where one doesn't need fixed character widths.

In the Settings menu both fonts can be set. The default font can also be toggled between the fixed and variable font by selecting or deselecting the "fixed font is default" menu item.

<p>Hint: Additionally to the default fonts any font installed on your system can be used. Please note, that the chosen font also will be used for HTML exports, so if your VYM mind map should be exported to a web page you should only use fonts which are available generally.</p>
--

!

4.6 Find text

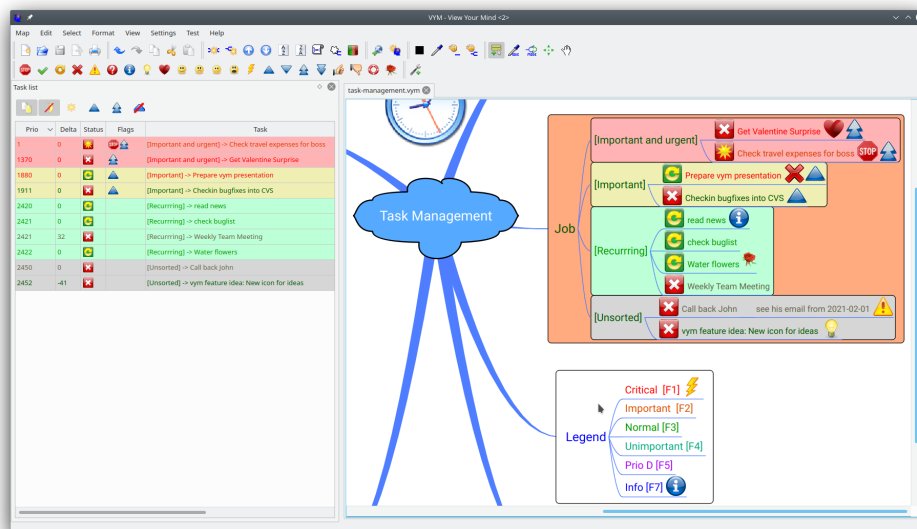
The noteeditor itself has no Find function, use Find in the mapeditor, which will also display occurrences of text in notes (see 3.2).

4.7 Paste text into note editor

Often you will paste text into the editor from another application e.g. an email.

5 Task editor

Tasks are used to easily create and maintain a "Todo-list". The taskeditor is visible on the left side in the image below.



The default columns in the taskeditor list are:

- Delta: This value can be manually added to the priority to change the automated prioritization
- Status: Shows the state and if the task is currently sleeping
- Flags: Lists flags, which affect the priority (see above)
- The name of task (identical to heading of branch in the map)

Priorities can be adjusted manually in the taskeditor, either by entering a delta value or by dragging the task to a new position (more on priorities below).

5.1 Creating tasks

To create a task press [Shift + W]. The branch will get an additional flag and its name will be visible in the taskeditor.

A task can have different states:


- New task or just awakened task 🌟

- Not started 
- Work in progress 
- Finished 

You can cycle these states by pressing [W].

5.2 Reminders - let a task sleep for a while

A task may be set to "sleep", which can be used to get a reminder after a certain amount of time. The flag will change to one of these:

- Not started - sleeping 
- Work in progress - sleeping 

You can set the sleep time of a task should right clicking on the task flag in the mapeditor or the task in the taskeditor. "Reset sleep" will wake up a currently sleeping task.

Alternatively you can press [Shift-Q] and manually enter the sleep time, here are some examples:


Input	Sleep
1	1 day, postpone until tomorrow morning
365	365 days
1w	1 week
3h	3 hours
10s	10 seconds
18:00	Postpone until 6pm
24.12.2024	Postpone until Dec 24 in 2024
2024-12-24	Postpone until Dec 24 in 2024
2038-12-24T21:34	Postpone until December 24 in 2038, 9:34 pm (ISO format)

5.3 Priority of tasks

The tasks visible in the taskeditor are ordered by their *priorities*, listed on the left side of the editor. The lower the priority number, the more important is a task: The most important task always has priority 1.






There are several principles that influence the priority of a task:

- Old tasks tend to bubble up, meaning they get a higher priority over time. Rationale: If you have a task named "water the flowers" and you have postponed it now for 187 days, better delete the task. (And create a new task to dump the remains of the flowers.)

- Colors in the map: If you use the Function keys to assign the colors red, amber, green, etc. the tasks will get a different priority accordingly
- The stopsign flag  will increase the priority. Use it for your shipstoppers. . .
- The arrow up flags also move tasks up in the list
- Sleeping tasks and finished tasks will tend to fall to the bottom of your task list.
- The freshly woken up "morning tasks" will tend to pop up right at the top, so that they cannot be missed. Remove the "morning" state by pressing [W] once.

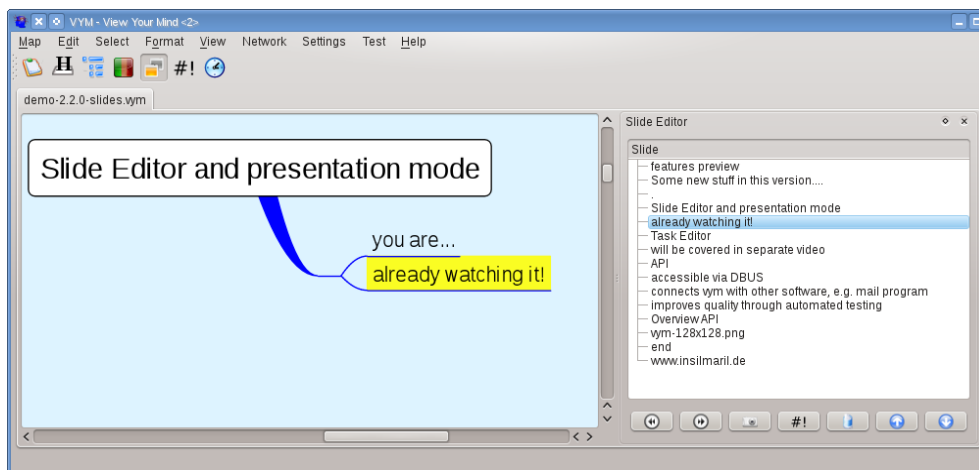
5.4 Filter tasks - keeping the overview

The taskeditor has filters:



Icon	Filter
	Current map only, hide tasks from other maps
	Active tasks only, hide finished nor sleeping
	New tasks only
	Tasks with "arrow up" flags only
	Only tasks without "arrow up" flags





6 Slideeditor - presentations

VYM can be used to do animated presentations using the slideeditor as seen on the right side of the image below:



The slideeditor can be opened by pressing [S] or from the *View* menu. The (currently) available action are:

-  Select previous slide
-  Select next slide

-  Create a new slide by snapshotting the current selection. The exact set of actions performed when selecting the new snapshot defined in a script called `slideeditor-snapshot.vys`. The script is one of the VYM macros, see 8.4.
- **#!** Open the scripteditor to manually edit the current slide (More on scripting in appendix B).
-  Delete the current slide
-  Move current slide up in slidedeck
-  Move current slide down in slidedeck

In the *View-menu* or it's toolbar you can also toggle the presentation mode to hide most of the toolbars and buttons

7 Hello world - vym and other applications

This section is about how VYM can interact with other applications. Many applications can now read and write their data using XML, the eXtensible Markup Language. VYM also uses XML to save its maps, see C.4 for a more detailed description.

So if you make use of another application that understands XML, chances are good that someone could write import/export filters for VYM . Volunteers are always welcome ;-)

7.1 Import

7.1.1 Mozilla Firefox bookmarks

Currently VYM supports an experimental import of Firefox bookmarks: Firefox can backup bookmmarks in a file in JSON format. This file can be imported into an existing VYM map.

Future VYM versions might be able to export this bookmark map again to JSON, so that it could be restored in Firefox.

7.1.2 Freemind and Freeplane

Freemind is no longer actively developed, the project is continued in the Freeplane project, see also <https://www.freeplane.org/>.

VYM supports reading the general structure of a Freeplane map and some of its flags. Also notes are read.

7.1.3 Mind Manager

VYM has currently a very basic import filter to convert maps created by *Mind Manager*⁶ into VYM maps. Notes and pictures are not converted at the moment. You can import files with

- File → Import → Mind Manager

7.1.4 Directory structure

VYM can read a directory structure. This is mainly for testing VYM e.g. to easily create huge maps used for benchmarks (yes, there is still room to optimize VYM ;-)

7.2 Export

Often you may not want to export the whole map, but just parts of it. For example you may have additional info you want to talk about in a presentation, while those parts should not be visible to the audience. To achieve this you can "hide" parts of the map during exports by setting the "hide in export" flag.



You can toggle this flag in the toolbar or by pressing [H]. Note that there is a global option in the settings menu (A.1) to toggle the use of this flag. By default the flag is enabled.

7.2.1 Last used format

Repeats the last export action without further dialogs like asking for directories. The associated export type and filepaths are stored within the map and thus map specific. Note: Not all export types support this feature yet.

7.2.2 Image

VYM supports all image formats which are natively supported by the QT toolkit: BMP, JPEG, PBM, PGM, PNG, PPN, XPM, and XBM. For use in websites and for sending images by email PNG is a good recommendation regarding quality and size of the image. VYM uses QTs default options for compressing the images.

7.2.3 PDF

Exports to Portable Document Format.

⁶Mind Manager is a commercial i.e. non free, software application by Mindjet for Windows and the Mac. Both names are registered trademarks by Mindjet. For more information see their website at <http://mindjet.com>

7.2.4 SVG

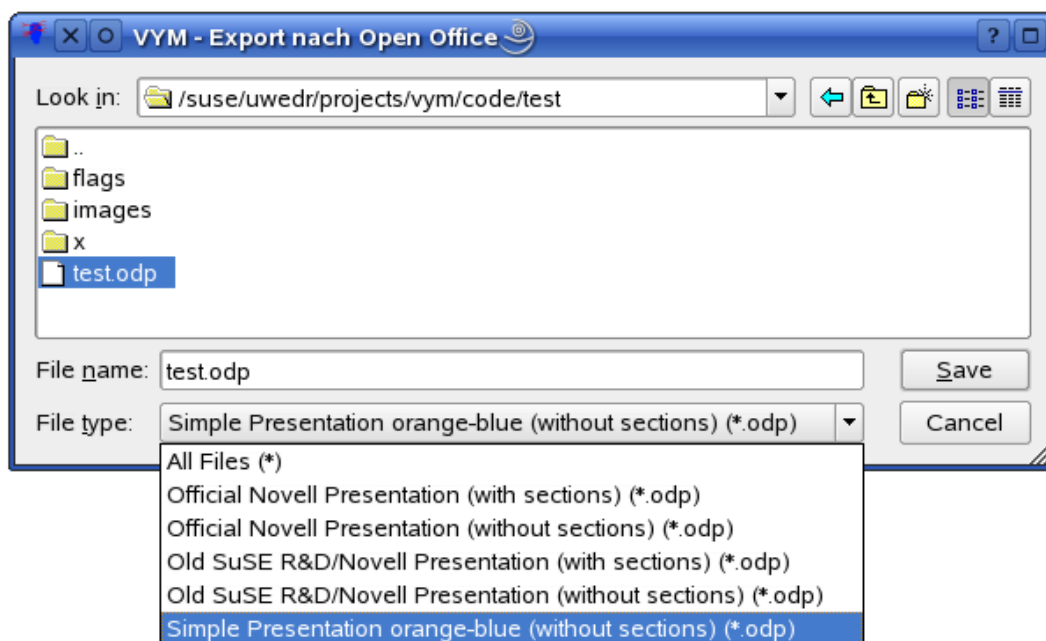
Exports to Scalable Vector Graphics.

7.2.5 Open Office

Open Office beginning with version 2 uses the so called "Open Document Format", which can be written by VYM . The options are currently limited, but it possible to export presentations which can be opened in Open Office Impress. By selecting

- File →Export→Open Office

you get a file dialogue where you can choose the output file and the file type:



The file types represent various templates, which can be created with some manual work from an existing Open Office document. The structure of VYM map is then inserted into a template. There are some limitations at the moment:

- VYM can't take care of page lengths, so you have to check and probably reedit in Open Office to avoid text running over the end of a page
- Images and flags are not used at the moment
- Notes are just written as plain text, without RichText
- The full range of templates are not available in all distributions.

Some of the templates make use of *sections* i.e sections insert the headings of mainbranches as chapters for sections into the presentation.

7.2.6 HTML (Webpages)

This is the format to use if you wish to create a webpage. To see an example visit the VYM homepage: www.InSilmaril.de/vym A dialog allows the user to set various options:

- **Include image:** If set, VYM will create an image map at the top of the HTML output. Clicking on a branch in the map will jump to the corresponding section in the output.
- **Colored headings:** If set to yes, VYM will colour the headings in the text part with the same colours used in the VYM map.
- **Save settings:** If set to yes, VYM will save above settings in the map.

7.2.7 A & O – Achievements and Objectives

A specialized form of ASCII export (see next section), which is used for workreports. Currently it is considered experimental.

7.2.8 ASCII

Exporting a map as text is somewhat experimental at the moment. Later this will probably be done using stylesheets. So the output may change in future versions of VYM

7.2.9 CSV

Exports map into a Comma Separated Value file, which can be used to import into all kinds spreadsheet software.

7.2.10 Taskjuggler

Used to export to Taskjuggler project management software. Currently considered experimental.

7.2.11 L^AT_EX

VYM can generate an input file for L^AT_EX. Currently this is considered as experimental, there are no options (yet). By selecting

- File → Export → L^AT_EX

you will be asked in a file dialog for the name of the output file. This file may then be included in a L^AT_EX document using command:

```
\include{inputfile.tex}
```

New in version 2.2.0: You can configure the names of the sections in the vym config file, depending on your platform e.g. in

```
$HOME/.config/InSilmaril/vym.conf
```

Just add e.g. these entries;

```
/export/latex/sectionName-0=chapter
/export/latex/sectionName-1=section
/export/latex/sectionName-2=subsection
/export/latex/sectionName-3=subsubsection
/export/latex/sectionName-4=paragraph
```

7.2.12 Markdown

Used to export to Markdown, see also <https://daringfireball.net/projects/markdown/> Currently considered experimental.

7.2.13 OrgMode

Used to export to Emacs OrgMode, see also <https://orgmode.org/> Currently considered experimental.

7.2.14 XML

The map is written into a directory both as an image and as an XML file. The directory is set in a file dialog. If the directory is not empty, you will be warned and offered choices if you are at risk of overwriting existing contents.

It is possible to export different maps into the same directory. Each file generated will have the map's name as prefix, e.g. `todo.vym` becomes `todo.xml`, `todo.png`, `todo-image-1.png` and so on. This is useful if, for example, a website comprises several combined maps that have to be stored in the same directory.

7.2.15 Export a part of a map

Select a branch you want to export together with its children, then open the context menu and choose *Save Selection*. This will create a file with the suffix `.vyp`, which is an abbreviation for "vym part".

7.3 Connect vym to the cloud

Starting with VYM 2.8.16 some exchange with cloud applications is possible. So far this is limited to cloud applications from the company Atlassian⁷. You find the related features and settings in the "Connect" settings in the menubar on top of the main window.

⁷Atlassian, Confluence and JIRA are registered trademarks

7.3.1 Confluence

VYM so far can

- Get Confluence user name and insert it into map. (This will create a link to the user profile during export to Confluence)
- Export a map to a Confluence page
- Get the name of a Confluence page and the space name and use it as heading when pasting an URL. (Happens automatically, if Confluence is configured)

7.3.2 JIRA

- Get description of a JIRA ticket and use it as heading
- Get additional information, e.g. color branch if ticket is already closed

7.4 Connect vym using DBUS

If you don't know what DBUS is, you probably want to skip this section, this is about remote controlling VYM using the DBUS protocol on Linux.

Currently this is used to

- Run the development tests, see also <https://github.com/insilmaril/vym/blob/develop/test/vym-test.rb>
- Add content from other applications, e.g. paste an email from the mutt client, see also <https://github.com/insilmaril/vym/blob/develop/scripts/vym-addmail.rb>

For more details see also [B.2.3](#).

8 Advanced usage

8.1 Quickly sorting branches or postponing actions

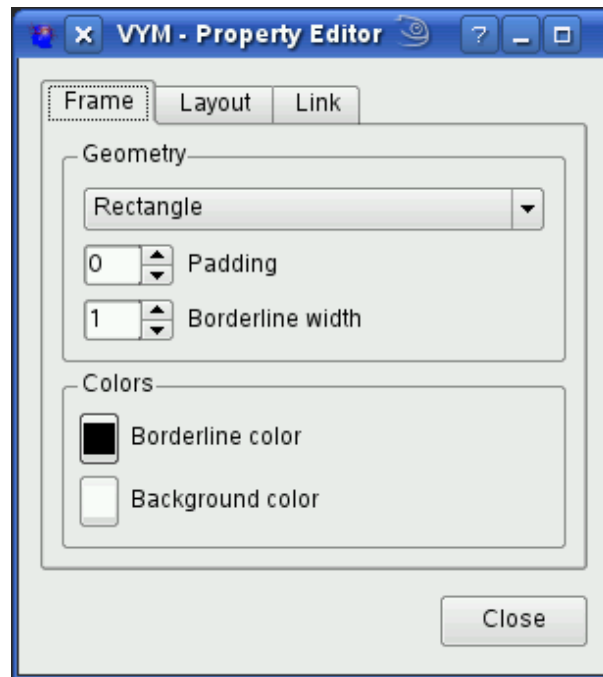
Sometimes it's very handy to have a small number of shortcuts to quickly select a certain branch or move something somewhere and proceed with next item. This can be used for quick sorting to a number of destination. For example if used in timeplanning, you could quickly move something to "next tuesday". The destinations are called *targets* and are marked with



The target flag is toggled with [T]. If you want to "Goto" a target, press [G]. Similar if you want to "Move" a branch, press [M]. A context menu will open and lets you select the target.

8.2 Properties of an object

For any branch you can open a satellite window (see 2.1): the *property window*:



Frame

sets the appearance of the frame of a branch. Currently there are

- No frame
- Rectangle
- Ellipse
- Cloud

The "Include Children" checkmark is used to, well, include the children of the branch. "Padding" sets the distance between both neighbouring branches and frame and also the frame and branch itself. The "width" sets the thickness of the frame.

Two colors can be assigned to background of the frame and frame itself.

Layout

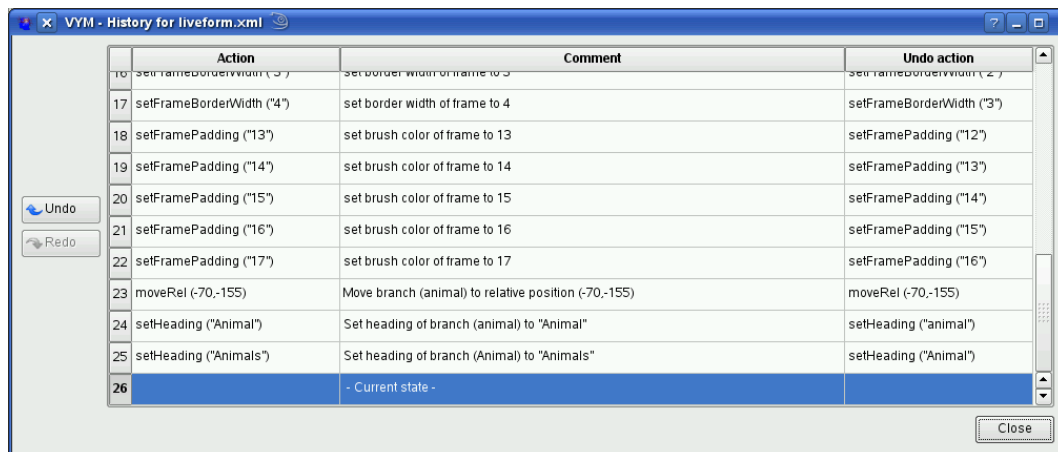
The images belonging to a branch can use different layouts, e.g. floating freely alongside or being included *within* the branch. For details and illustration see 8.6).

Link

The advantage of hiding a link, which is the connection between a branch (or image) and its parent, is to make the branch itself appear a mapcenter of its own. Details are in 8.8).

8.3 Changing the history: Undo and Redo

VYM keeps track of all changes done in a map. The default number of changes which can be undone is 75. The complete history can be seen in the *historywindow*:



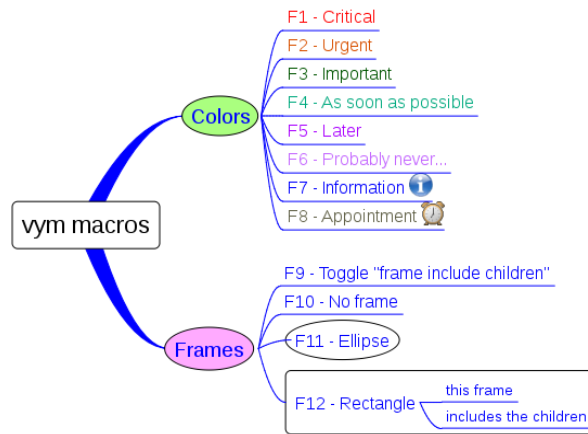
A single step back be undone or redone with [CTRL-Z] or [CTRL-Y], or by using the buttons in the toolbar or the *historywindow*. Inside the *historywindow*, you can click on a line to unwind all actions done until that point in time – or redo all changes by clicking on the last line.

Hint: You can "paste from the past": Go back in time by e.g. with [CTRL-Z], then copy to clipboard by pressing [CTRL-C]. Now do all actions again, e.g. by [CTRL-Y] or clicking on the last action in *historywindow*. Now paste from the past with [CTRL-V].

!

8.4 Macros

Each function key [F1] to [F12] holds a macro, which is executed on the current selection if the key is pressed. The default macros change the colour of a subtree or set the frame of a branch:



Each macro is a VYM script, which is executed when the associated key is pressed. The default location of the scripts can be changed in the Settings menu. More information on using scripts in VYM is found in appendix B.

8.5 Bookmarks

Open new tabs instead of new windows

If you use konqueror as your browser, VYM will remember the konqueror session which was opened first by VYM . You can also press [Ctrl] and click to open the link in a new tab.

VYM can also open a new tab in Mozilla or Firefox using the remote command⁸ of these browsers.

Drag and Drop

If you want to keep bookmarks in a map, select a branch where you want to add the bookmark, then simply drag the URL from your browser to the map. Also you could use an existing heading as URL: Right click onto the branch and select "Use heading for URL".

Directly access bookmark lists of a browser

Please see the sections 7.1 and 7.2 about Import and Export filters.

8.6 Associating images with a branch

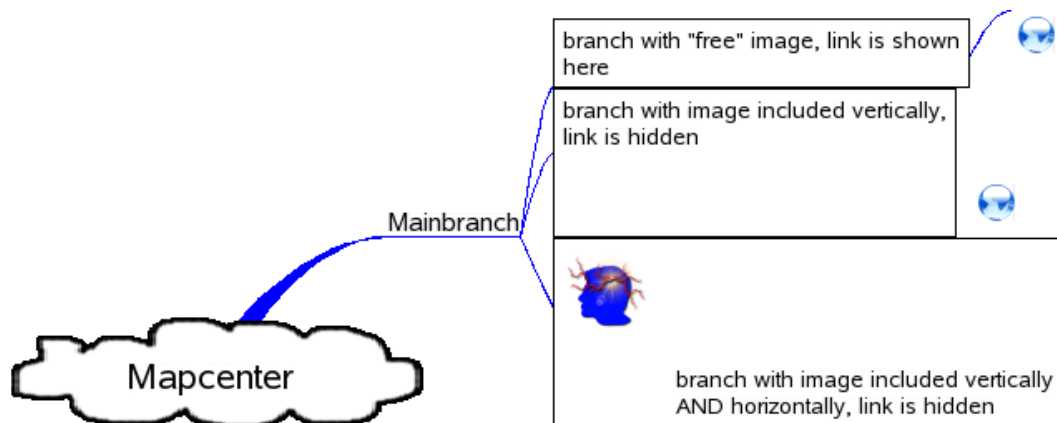
The default setting for an image is for it to float "freely". Images can be positioned anywhere on the canvas, but may end up in the same place as other parts of the map obscuring that part of the map.

⁸<http://www.mozilla.org/unix/remote.html>

The solution is to insert or include them "into" a branch. This can be done via the property window (see 8.2):

- Include images horizontally
- Include images vertically

The image is still positioned relative to its parent branch, but the heading and border of the branch frame adapt to the floating image, see below:



8.7 Modifier Modes

The modifier mode can be selected in the toolbar

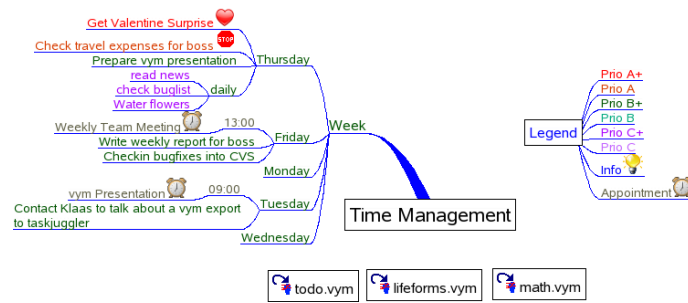


or also using the adjacent keys [L], [K] and [L]. The selected mode influences mouse behaviour when the [Shift]-modifier is used:

- Multiple selection: Press Shift and click to select multiple objects
- Colorpicker: Pick from another branch and apply to currently selected branch
- XLink: Draw a connecting XLink between two branches. See also 8.9
- Move object: Move an object, but when releasing it over another one, do not relink. (Useful for positioning branches in other branches for presentations.)
- Move view: Only move view without selecting an object

8.8 Hide links of unselected objects

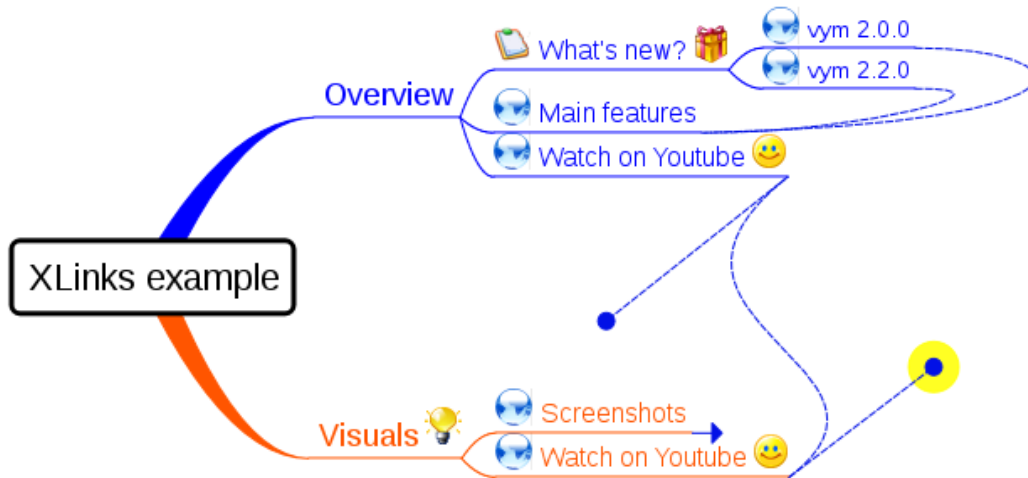
Sometimes it would be useful to position a branch freely, just like a mainbranch or an image. This is possible for all branches, you can use a mainbranch and hide its connecting link to the mapcenter or hide the link between a child branch and its parent. This can be used e.g. for legends or a collection of vymLinks pointing to other maps:



To hide the link between a branch and its parent open the 8.2 and check "Hide link if object is not selected" on "Link" tab.

8.9 XLinks

So far all the data in the vym map has been treelike. Using xLinks you can link one branch to any other, just like attaching a rope between two branches in a real tree. This is especially useful in complex maps, where you want to have crossreferences which can not be displayed on the same visible area of the *mapeditor* window. The following example map still fits on one screen, but shows how data can be crosslinked. In the graphics there is a link from a task (prepare a presentation) to general information:



Note that a xLink which points to a branch, that is not visible (because it is scrolled), is just shown as a little horizontal arrow. In the image above have a look at the "Screenshot" branch.

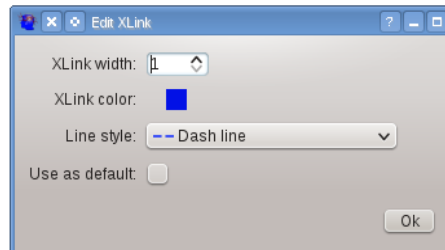
Create a xLink

Choose the link mode from the modifier toolbar (by clicking the toolbar icon or pressing [L]). Select the branch, where the xLink should start. Press the modifier key [Shift] drag the mouse pointer to the branch where the link should end. (The link is drawn to follow the mouse pointer). When you release the mouse over a branch the xLink becomes permanent.

Modify or delete a xLink

First select the link, either in the tree editor or by clicking the xLink itself in the mapeditor. A dialogue opens, where you can set colour, width and also delete the xLink.

You can also move the control points of the link to change the curve and change the appearance of the xlink by right clicking on one of the control points:



Follow a xLink

In a complex VYM map it sometimes comes in handy to be able to jump to the other end of a xLink. You can do this by opening the context menu of the branch and clicking on "Goto xLink" and selecting the xLink you want to follow. Even easier is to click on the lower right end of a branch – a popup menu will show up with all xLinked branches. Click one of them to jump to it.

8.10 Adding and removing branches

The context menu of a branch shows some more ways to add and delete data e.g. you can delete a branch while keeping its children. The children become linked to the parent of the previously removed branch. Similar branches can be inserted into existing maps. For keyboard shortcuts also have a look at the context menu.

8.11 Adding a whole map or a part of a map

Select a branch where you want to add a previously saved map (.vym) or a part of a map (.vyp), then open the context menu and choose *Add* → *Add Map (Insert)*. For the import you can choose between *Add Map (Insert)* and *Add Map (Replace)*: The imported data will be added after the selected branch.

9 VYM on Mac OS X

9.1 Contextmenu and special keys

Most Macs unfortunately just have a single mouse button. In order to show the context menu which usually would be opened with the right mouse button, you can click while pressing the [kommand]-key.

Especially on Laptops some of the keys usually used on PC keyboards seem to be missing. The Qt-Mac Edition of VYM has its own keyboard shortcuts. To find the shortcuts just have a look at all the menu entries, the shortcut is visible next to an entry. Toolbar buttons also may have shortcuts, just position the mouse pointer over a button and wait for the little help window to appear.

9.2 Viewing external links

VYM on Mac uses the system call `/usr/bin/open` to view links. Mac OS determines automatically if the link is a pdf or www page and opens the right browser.

A VYM initialisation process and configuration

A.1 Settings menu

The *Settings* menu allows to configure VYM to your needs:

Set application to open PDF files

Choose a PDF viewer like `acrobat` or `konqueror` which is installed on your system.

Set application to open external links

Choose your favourite application, this usually depends on your platform, e.g.

1. Windows: `explorer`
2. Linux: `xdg-open` or `mimeopen`
3. Mac: `/usr/bin/open`

Defaults should be set by VYM automatically.

Set path for macros

Set the default search path for macros, which will be executed when you press one of the function keys. Each key corresponds to a file (`macro-1.vys`..`macro12.vys`) in the search path.

Set number of undo levels

Sets the number of undo/redo levels. The default setting is 75 levels.

Autosave and autosave time

Automatic saving of modified maps can be toggled on or off. The autosave time is entered in seconds.

Write backup on save

When saving a map called `example.vym`, VYM will rename the existing file to `example.vym~` before writing the `example.vym` itself.

Edit branch after adding it

If set, the heading of a new branch will be edited immediatly after adding the branch.

Select branch after adding it

If set, a new branch will be selected immediatly after adding it. When you "brainstorm" on a given keyword, you don't want to go deeper and deeper into details, but keep the focus on the keyword. So the default setting here is to *not* select the freshly added branch.

Select existing heading

If set and you begin to edit the heading of a branch, the heading text in the dialog will be selected. Usefully to copy&paste to other applications.

Delete key

If set, the [Delete] is enabled to, well, delete objects. This can be switched off to avoid confusing with the nearby [Insert]-key on PC keyboards.

Exclusive flags

If set, some of the standard flags can only be used exclusively, e.g. the smileys.

Use hide flags

If set, every branch which also has the hide flag set (see [7.2](#)) will be hidden in exports.

Note editor is dockable

If set (default), the note editor can be docked into the main widget. Changing this setting needs a restart of VYM . Details see [2.2](#).

Animation

If set (default), some animation will be used, e.g. for "snapping back" of released branches.

Autolayout

If set (not on default), VYM will try to autolayout mainbranches. Currently considered experimental and only working under certain circumstances.

A.2 Configuration file

On startup VYM will look for a configuration for user specific settings like window positions, toolbars etc. If this file does not already exist, it will be created. The file is located in the users home directory. The exact position depends on the platform:

Platform	Configuration file
Linux	<code>~/.config/InSilmaril/vym.conf</code>
Mac OS X	<code>/Users/NAME/Library/Preferences/com.insilmaril.vym.plist</code>
Windows (registry)	<code>HKEY_Current_User/Software/InSilmaril/vym</code>

The file can be edited manually, or on Mac OS X with Property List Editor (installed with xtools). On windows you can use `regedit.exe`.

A.3 Path to ressources

VYM will try to find its ressources (images, stylesheets, filters, etc.) in the following places:

1. Path given by the environment variable `VYMHOME`.
2. If called with the local option (see [A.4](#) below), VYM will look for its data in the current directory.
3. `/usr/share/vym`
4. `/usr/local/share/vym`

A.4 Command line options

Usage: `vym [OPTION]... [FILE]...`
Open FILES with vym

<code>-b</code>	<code>batch</code>	batch mode: hide windows
<code>-c</code>	<code>commands</code>	List all available commands
<code>-cl</code>	<code>commandslatex</code>	List commands in LaTeX format
<code>-d</code>	<code>debug</code>	Show debugging output
<code>-h</code>	<code>help</code>	Show this help text
<code>-L</code>	<code>load</code>	Load script
<code>-l</code>	<code>local</code>	Run with ressources in current directory
<code>—locale</code>	<code>locale</code>	Override system locale setting to select language
<code>-n STRING</code>	<code>name</code>	Set name of instance for Dbus access
<code>-q</code>	<code>quit</code>	Quit immediatly after start for benchmarking
<code>-R FILE</code>	<code>run</code>	Run script
<code>-r</code>	<code>restore</code>	Restore last session
<code>—recover</code>	<code>recover</code>	Delete lockfiles during initial loading of files

-s	shortcuts	Show Keyboard shortcuts on start
-t	testmode	Test mode, e.g. no autosave and changing of its setting
-v	version	Show vym version

You can also give several filenames at the commandline to let VYM open several maps at once.

B Scripts

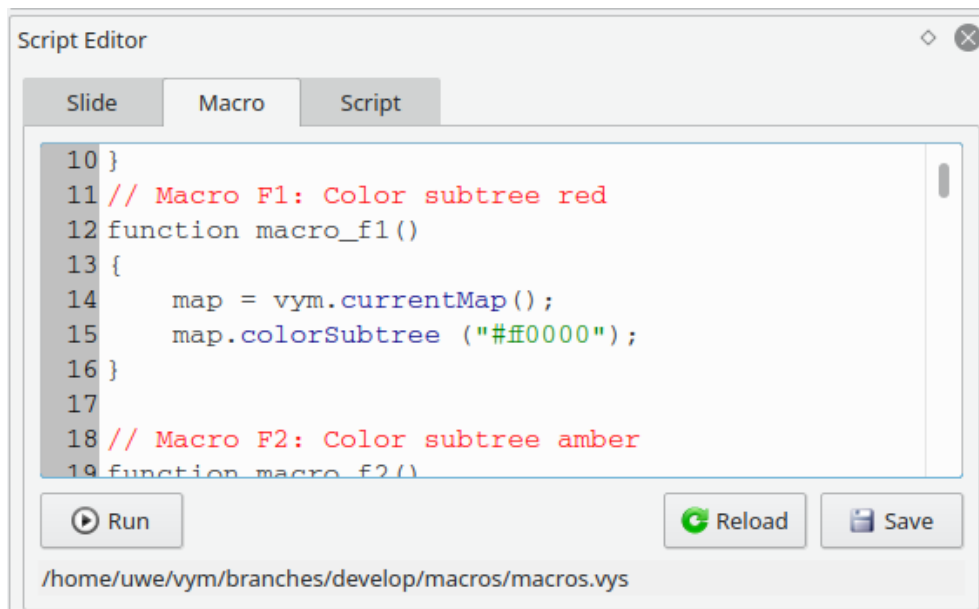
B.1 Overview

Beginning with version 2.7.0 VYM is fully scriptable, though the scripting support is still considered a *technical preview*. Some parts still might change and improve in later versions. Scripts are internally used for

- Undo and Redo
- Macros on function keys
- Slideshow

In addition to the internal scriptengine, which is using QScript, you can also use external ruby scripts, which communicate with VYM via DBUS. Please note that the latter is currently only possible on Linux. See also the examples in [B.2](#).

The scripts within VYM are edited using the *script editor*:



Open the scripteditor by pressing [Alt + S] or from the *View*-menu. The output of of scripts can be seen in the script output window with [Alt + Shift + S]

B.2 Example scripts

A set of example scripts is installed together with VYM , see the installation directory and the subfolder `demos/scripts/` and the macros in the macro tab of the script editor.

B.2.1 Macro to create a rounded rectangle frame

```
// Macro Shift + F1: Frame background light red
function macro_shift_f1()
{
    map = vym.currentMap();
    status = "Background off";
    if (map.getFrameType() == "NoFrame") {
        status = "Background light red";
    }
    toggle_frame ( map );
    map.setFrameBrushColor("#ffb3b4");
    statusMessage(status);
}
```

B.2.2 Batch script to export all maps as images

This script can be used to export all maps in a directory automatically. If the script is named `export-image.vys`, call VYM with

```
$ vym --quit --run export-image.vys *.vym
```

B.2.3 Full scripting using ruby and DBUS

Nearly every action in VYM can be controlled via DBUS (on Linux machines). You can have several VYM instances running at the same time, e.g. for production and development. Before controlling one, you need to give it a name, here "test" is used:

```
vym -n test
```

You can now access VYM via DBUS, if you have Qt installed, try `qdbusviewer`. In the `scripts` directory, which is part of VYM , you'll find the script `vym-ruby.rb`. This rubyscript provides two classes to manage and control VYM instances. A short script to set the heading of a branch might be:

```
#!/usr/bin/env ruby

require "#{ENV['PWD']}/scripts/vym-ruby"

vym_mgr=VymManager.new
vym=Vym.new(vym_mgr.find('test') )
```

```
vym.select "mc:0"  
vym.setHeading "This is a new heading!"
```

An full example how this is used is in the automated VYM testing:

```
test/vym-test.rb
```

B.3 Available commands

Start vym with the "command" option to get a listing of available commands:

```
vym --commands --quit
```

The currently available commands are:

- addBranch
 SelectionType: Branch
 Return Type: Void
 Parameters: 1
 Parameter: 1:
 Comment: Index of new branch
 Type: Int
 Optional: yes
- addBranchBefore
 SelectionType: Branch
 Return Type: Void
 Parameters: 0
- addMapCenter
 SelectionType: Any
 Return Type: Void
 Parameters: 2
 Parameter: 1:
 Comment: Position x
 Type: Double
 Optional: No
 Parameter: 2:
 Comment: Position y
 Type: Double
 Optional: No
- addMapInsert

SelectionType: Any
Return Type: Void
Parameters: 3
Parameter: 1:
Comment: Filename of map to load
Type: String
Optional: No
Parameter: 2:
Comment: Index where map is inserted
Type: Int
Optional: yes
Parameter: 3:
Comment: Content filter
Type: Int
Optional: yes

- addMapReplace
SelectionType: Branch
Return Type: Void
Parameters: 1
Parameter: 1:
Comment: Filename of map to load
Type: String
Optional: No
- addSlide
SelectionType: Branch
Return Type: Void
Parameters: 0
- addXLink

- SelectionType: BranchLike
 - Return Type: Void
 - Parameters: 5
 - Parameter: 1:
 - Comment: Begin of XLink
 - Type: String
 - Optional: No
 - Parameter: 2:
 - Comment: End of XLink
 - Type: String
 - Optional: No
 - Parameter: 3:
 - Comment: Width of XLink
 - Type: Int
 - Optional: yes
 - Parameter: 4:
 - Comment: Color of XLink
 - Type: Color
 - Optional: yes
 - Parameter: 5:
 - Comment: Penstyle of XLink
 - Type: String
 - Optional: yes
- branchCount
 - SelectionType: Any
 - Return Type: Int
 - Parameters: 0
- centerCount
 - SelectionType: BranchLike
 - Return Type: Int
 - Parameters: 0
- centerOnID
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: UUID of object to center on
 - Type: String
 - Optional: No
- clearFlags
 - SelectionType: BranchLike
 - Return Type: Void
 - Parameters: 0
- colorBranch

- SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: New color
 - Type: Color
 - Optional: yes
- colorSubtree
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: New color
 - Type: Color
 - Optional: yes
- copy
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 0
- cut
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 0
- cycleTask
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: True, if cycling in reverse order
 - Type: Bool
 - Optional: yes
- depth
 - SelectionType: BranchOrImage
 - Return Type: Int
 - Parameters: 0
- exportMap
 - SelectionType: Any
 - Return Type: Bool
 - Parameters: 1
 - Parameter: 1:
 - Comment: Format (AO, ASCII, CONFLUENCE, CSV, HTML, Image, Impress, Last, La
 - Type: String
 - Optional: No
- getDestPath
 - SelectionType: Any
 - Return Type: String
 - Parameters: 0

- getFileDir
 - SelectionType: Any
 - Return Type: String
 - Parameters: 0
- getFileName
 - SelectionType: Any
 - Return Type: String
 - Parameters: 0
- getFrameType
 - SelectionType: Branch
 - Return Type: String
 - Parameters: 0
- getHeadingPlainText
 - SelectionType: TreeItem
 - Return Type: String
 - Parameters: 0
- getHeadingXML
 - SelectionType: TreeItem
 - Return Type: String
 - Parameters: 0
- getMapAuthor
 - SelectionType: Any
 - Return Type: String
 - Parameters: 0
- getMapComment
 - SelectionType: Any
 - Return Type: String
 - Parameters: 0
- getMapTitle
 - SelectionType: Any
 - Return Type: String
 - Parameters: 0
- getNotePlainText
 - SelectionType: TreeItem
 - Return Type: String
 - Parameters: 0
- getNoteXML
 - SelectionType: TreeItem
 - Return Type: String
 - Parameters: 0
- getSelectionString
 - SelectionType: TreeItem
 - Return Type: String
 - Parameters: 0

- getTaskPriorityDelta
 - SelectionType: Branch
 - Return Type: Int
 - Parameters: 0
- getTaskSleep
 - SelectionType: Branch
 - Return Type: String
 - Parameters: 0
- getTaskSleepDays
 - SelectionType: Branch
 - Return Type: Int
 - Parameters: 0
- getURL
 - SelectionType: TreeItem
 - Return Type: String
 - Parameters: 0
- getVymLink
 - SelectionType: Branch
 - Return Type: String
 - Parameters: 0
- getXLinkColor
 - SelectionType: XLink
 - Return Type: String
 - Parameters: 0
- getXLinkWidth
 - SelectionType: XLink
 - Return Type: Int
 - Parameters: 0
- getXLinkPenStyle
 - SelectionType: XLink
 - Return Type: String
 - Parameters: 0
- getXLinkStyleBegin
 - SelectionType: XLink
 - Return Type: String
 - Parameters: 0
- getXLinkStyleEnd
 - SelectionType: XLink
 - Return Type: String
 - Parameters: 0

- hasActiveFlag
 - SelectionType: TreeItem
 - Return Type: Bool
 - Parameters: 1
 - Parameter: 1:
 - Comment: Name of flag
 - Type: String
 - Optional: No
- hasNote
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- hasRichTextNote
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- hasTask
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- importDir
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Directory name to import
 - Type: String
 - Optional: No
- initIterator
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 2
 - Parameter: 1:
 - Comment: Name of iterator
 - Type: String
 - Optional: No
 - Parameter: 2:
 - Comment: Flag to go deep levels first
 - Type: Bool
 - Optional: yes
- isScrolled
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0

- loadImage
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Filename of image
 - Type: String
 - Optional: No
- loadNote
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Filename of note
 - Type: String
 - Optional: No
- moveDown
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- moveUp
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- moveSlideDown
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- moveSlideUp
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- move
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 2
 - Parameter: 1:
 - Comment: Position x
 - Type: Double
 - Optional: No
 - Parameter: 2:
 - Comment: Position y
 - Type: Double
 - Optional: No

- moveRel
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 2
 - Parameter: 1:
 - Comment: Position x
 - Type: Double
 - Optional: No
 - Parameter: 2:
 - Comment: Position y
 - Type: Double
 - Optional: No
- nextIterator
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 1
 - Parameter: 1:
 - Comment: Name of iterator
 - Type: String
 - Optional: No
- nop
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- note2URLs
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- parseVymText
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 1
 - Parameter: 1:
 - Comment: parse XML of VymText, e.g for Heading or VymNote
 - Type: String
 - Optional: No
- paste
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- redo
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- relinkTo

- SelectionType: TreeItem
 - Return Type: Bool
 - Parameters: 4
 - Parameter: 1:
 - Comment: Selection string of parent
 - Type: String
 - Optional: No
 - Parameter: 2:
 - Comment: Index position
 - Type: Int
 - Optional: No
 - Parameter: 3:
 - Comment: Position x
 - Type: Double
 - Optional: yes
 - Parameter: 4:
 - Comment: Position y
 - Type: Double
 - Optional: yes
- remove
 - SelectionType: TreeItem
 - Return Type: Void
 - Parameters: 0
- removeChildren
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- removeKeepChildren
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- removeSlide
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Index of slide to remove
 - Type: Int
 - Optional: No
- repeatLastCommand
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- saveImage

- SelectionType: Image
 - Return Type: Void
 - Parameters: 2
 - Parameter: 1:
 - Comment: Filename of image to save
 - Type: String
 - Optional: No
 - Parameter: 2:
 - Comment: Format of image to save
 - Type: String
 - Optional: No
- saveNote
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Filename of note to save
 - Type: String
 - Optional: No
- scroll
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- select
 - SelectionType: Any
 - Return Type: Bool
 - Parameters: 1
 - Parameter: 1:
 - Comment: Selection string
 - Type: String
 - Optional: No
- selectFirstBranch
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- selectFirstChildBranch
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- selectID
 - SelectionType: Any
 - Return Type: Bool
 - Parameters: 1
 - Parameter: 1:
 - Comment: Unique ID
 - Type: String
 - Optional: No

- selectLastBranch
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- selectLastChildBranch
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- selectLastImage
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- selectLatestAdded
 - SelectionType: Any
 - Return Type: Bool
 - Parameters: 0
- selectParent
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- selectToggle
 - SelectionType: BranchOrImage
 - Return Type: Bool
 - Parameters: 0
- setFlagByName
 - SelectionType: TreeItem
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Name of flag
 - Type: String
 - Optional: No
- setTaskPriorityDelta
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Manually add value to priority of task
 - Type: String
 - Optional: No
- setTaskSleep

- SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Days to sleep
 - Type: String
 - Optional: No
- setFrameIncludeChildren
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Include or don't include children in frame
 - Type: Bool
 - Optional: No
- setFrameType
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Type of frame
 - Type: String
 - Optional: No
- setFramePenColor
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Color of frame border line
 - Type: Color
 - Optional: No
- setFrameBrushColor
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Color of frame background
 - Type: Color
 - Optional: No
- setFramePadding
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Padding around frame
 - Type: Int
 - Optional: No

- setFrameBorderWidth
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Width of frame borderline
 - Type: Int
 - Optional: No

- setHeadingConfluencePageName
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0

- setHeadingPlainText
 - SelectionType: TreeItem
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: New heading
 - Type: String
 - Optional: No

- setHideExport
 - SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Set if item should be visible in export
 - Type: Bool
 - Optional: No

- setIncludeImagesHorizontally
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Set if images should be included horizontally in parent branch
 - Type: Bool
 - Optional: No

- setIncludeImagesVertically
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Set if images should be included vertically in parent branch
 - Type: Bool
 - Optional: No

- setHideLinksUnselected

- SelectionType: BranchOrImage
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Set if links of items should be visible for unselected items
 - Type: Bool
 - Optional: No
- setMapAnimCurve
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: EasingCurve used in animation in MapEditor
 - Type: Int
 - Optional: No
- setMapAuthor
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment:
 - Type: String
 - Optional: No
- setMapAnimDuration
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Duration of animation in MapEditor in milliseconds
 - Type: Int
 - Optional: No
- setMapBackgroundColor
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Color of map background
 - Type: Color
 - Optional: No
- setMapComment
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment:
 - Type: String
 - Optional: No

- setMapTitle
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment:
 - Type: String
 - Optional: No

- setMapDefLinkColor
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Default color of links
 - Type: Color
 - Optional: No

- setMapLinkStyle
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Link style in map
 - Type: String
 - Optional: No

- setMapRotation
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Rotation of map
 - Type: Double
 - Optional: No

- setMapTitle
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment:
 - Type: String
 - Optional: No

- setMapZoom

- SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Zoomfactor of map
 - Type: Double
 - Optional: No
- setNotePlainText
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Note of branch
 - Type: String
 - Optional: No
- setScaleFactor
 - SelectionType: Image
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Scale image by factor f
 - Type: Double
 - Optional: No
- setSelectionColor
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Color of selection box
 - Type: Color
 - Optional: No
- setTaskPriority
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Priority of task
 - Type: Int
 - Optional: No
- setTaskSleep
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 1
 - Parameter: 1:
 - Comment: Sleep time of task
 - Type: String
 - Optional: No

- setURL
 - SelectionType: TreeItem
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: URL of TreeItem
 - Type: String
 - Optional: No

- setVymLink
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Vymlink of branch
 - Type: String
 - Optional: No

- setXLinkColor
 - SelectionType: XLink
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Color of xlink
 - Type: String
 - Optional: No

- setXLinkStyle
 - SelectionType: XLink
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Style of xlink
 - Type: String
 - Optional: No

- setXLinkStyleBegin
 - SelectionType: XLink
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Style of xlink begin
 - Type: String
 - Optional: No

- setXLinkStyleEnd

- SelectionType: XLink
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Style of xlink end
 - Type: String
 - Optional: No
- setXLinkWidth
 - SelectionType: XLink
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Width of xlink
 - Type: Int
 - Optional: No
- sleep
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Sleep (seconds)
 - Type: Int
 - Optional: No
- sortChildren
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Sort children of branch in revers order if set
 - Type: Bool
 - Optional: yes
- toggleFlagByUid
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Uid of flag to toggle
 - Type: String
 - Optional: No
- toggleFlagByName
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Name of flag to toggle
 - Type: String
 - Optional: No

- toggleFrameIncludeChildren
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- toggleScroll
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- toggleTarget
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- toggleTask
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- undo
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- unscroll
 - SelectionType: Branch
 - Return Type: Bool
 - Parameters: 0
- unscrollChildren
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- unselectAll
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- unsetFlagByName
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Name of flag to unset
 - Type: String
 - Optional: No
- clearConsole
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0

- currentMap
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- currentMapIndex
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- editHeading
 - SelectionType: Branch
 - Return Type: Void
 - Parameters: 0
- loadMap
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Path to map
 - Type: String
 - Optional: No
- mapCount
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0
- selectMap
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Index of map
 - Type: Int
 - Optional: No
- selectQuickColor
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 1
 - Parameter: 1:
 - Comment: Index of quick color [0..6]
 - Type: Int
 - Optional: No
- currentColor
 - SelectionType: Any
 - Return Type: Void
 - Parameters: 0

- `toggleTreeEditor`
 SelectionType: Any
 Return Type: Void
 Parameters: 0
- `version`
 SelectionType: Any
 Return Type: Void
 Parameters: 0

C Contributing to VYM

So far I'd say I have written 98% of the code on my own. No surprise, that VYM exactly fits my own needs. Nevertheless I would like to encourage all users of VYM to contribute. Maybe not only with feature requests, but also with code, new import/export filters, translations etc. In this appendix I'll try to show how easy it is to expand the things you can do already with VYM . I really look forward to hear from you!

C.1 Getting help

Frequently asked questions

Please refer to the FAQ available on the VYM website:

<http://www.InSilmaril.de/vym/faq.html>

Mailinglists

There are several mailinglists: `vym-forum` is the VYM users forum to discuss various questions, while `vym-devel` is intended for people interested in contributing to VYM . The third list is `vym-trans` to coordinate translations. You can view the archives and subscribe at

https://sourceforge.net/mail/?group_id=127802

Contacting the author

Especially for support questions please try the mailinglists first. If everything else fails you can contact the author Uwe Drechsel at

vym@InSilmaril.de

C.2 How to report bugs

Please file bugs and issues in github:

<https://github.com/insilmaril/vym/issues>

I build VYM regularly for openSUSE and Windows, so you may report it against a recent version there, even if you use another Operating System. Please don't forget to tell me what you are using:

- the exact steps needed to reproduce the bug
- the version and build date of VYM (see the Help →About VYM)
- hardware and Operating System

C.3 Compiling from the sources

C.3.1 Getting the sources

You find the latest code of VYM in the `develop` branch on Github:

<https://github.com/insilmaril/vym>

C.3.2 The Qt toolkit

Qt is C++ toolkit for multiplatform GUI and application development. It provides single-source portability across MS Windows, Mac OS X, Linux and many more. Qt is also available for embedded devices. Qt is a product of the Qt Company. For more information see

<https://www.qt.io>

C.3.3 Compiling VYM

Make sure you have installed your Qt environment properly, see the Qt documentation for details. You need to have the Qt command `qmake` in your PATH-environment, then run

```
$ qmake
$ make
$ make install
```

The last command `make install` needs root-permissions. Of course it may be omitted, if you just want to test VYM . For testing you should also use the `"-l"` (local) option on startup (see A.4).

C.4 VYM file format

VYM maps usually have the suffix ".vym" and represent a compressed archive of data. Actually the widely used "zip" format is used⁹.

If you want to have a closer look into the data structure map called "mapname.vym", just edit the XML data by using the `vivym` script or by uncompressing manually. In a shell on Linux or Mac OS you can do this by calling

```
$ unzip mapname.vym
```

This will create directories named `images` and `flags` in your current directory and also the map itself, usually named `mapname.xml`. The XML structure of VYM is pretty self explaining, just have a look at `mapname.xml`.

This XML file can be loaded directly into VYM, it does not have to be compressed. If you want to compress all the data yourself, use

```
$ zip -r mapname.vym .
```

to compress all data in your current directory.

C.5 New features

There are lots of features which might find their way into VYM. Together with VYM you should have received a directory with several example maps. You find them by clicking Help → Open vym example maps. There you will find the map `vym-projectplan.vym`. It lists quite a lot of things to be done in future. If you have more ideas, open enhancement tickets team at <https://github.com/insilmaril/vym/issues>

C.6 New languages support

In order to add a new language to VYM you need the sources (see C.3.1) and an installation of Trolltechs QT. A part of QT are the development tools, from those tools especially the translation tool "Linguist" is needed.

In some Linux distributions the development tools are in an extra package, e.g. on SUSE LINUX you should have installed:

```
libqt5-devel.rpm  
libqt5-devel-doc.rpm  
libqt5-devel-tools.rpm
```

If you don't have Qt in your system, you can get it from <https://qt.nio> You can translate the text in vym itself by performing the following steps:

⁹zip is meanwhile available on all major platforms, including Windows 10: either in the file explorer or on the command line e.g. as part of the "tar" command

- Let's assume now your encoding is "NEW" instead of for example "de" for german or "en" for english
- Copy the file `lang/vym_en.ts` to `lang/vym_NEW.ts` (The code itself contains the english version.)
- Add `lang/vym_NEW.ts` to the TRANSLATIONS section of `vym.pro`
- Run Linguist on `vym_NEW.ts` and do the translation
- Run `lrelease` to create `vym_NEW.qm`
- Do a `make install` to install the new vym and check your translation

If you feel brave, you can also translate the manual. It is written in LaTeX, you just have to change the file `tex/vym.tex`. (Linguist and Qt are not needed, but it is useful to know how to work with LaTeX and esp. `pdflatex` to create the PDF.)

Please mail me every translation you have done. I can also give you a developer access to the project, if you want to provide translations regularly.

C.7 New export/import filters

VYM supports various kinds of filters. Data can be written directly, inserted into templates or it can be written as XML data and then processed by XSL transformations.

Most of the import/export functionality is available in the classes `ImportBase` and `ExportBase` and subclasses. All of them can be found in `imports.h` and `exports.h`.

Direct import/export

An example for a direct export is the XML export. This method touches the implementation of nearly every object of VYM, so whenever possible you should better use a XSL transformation instead.

If you still want to know how it is done, start looking at `MapEditor::saveToDir` in `mapeditor.cpp`.

Templates

Templates have been introduced to export to opendoc format used e.g. by Open Office. While I read the spec (> 500 pages) about the format¹⁰ I had the feeling that I did not want to write the export from scratch. It would be too complex to adapt the styles to your own wishes, e.g. the layout.

Instead I analyzed existing Open Office documents. I found out that there are lots of redundant bits of information in a standard presentation, for example each list item is contained in its own list. In the end I came up with the default presentation style, which still could be simplified, just in case you have free time...

¹⁰<http://www.oasis-open.org/>

The existing templates are still work in progress, before you spend too much time developing your own style, please contact me. Basically the following steps are needed to build your own style:

1. Create an example in Open Office. Use a title, authors name, page heading etc. which you can easily grep for in the output file.
2. Unzip the Open Office document into a directory.
3. The main file is called `content.xml`. All data is in one single line. You can split the XML tags using the script `scripts/niceXML`, which is part of the VYM distribution.
4. Copy the output of `niceXML` to `content-template.xml`.
5. Looking closer you will find lots of unused definitions, for example of styles. You can delete or simply ignore them.
6. Try to find your title, authors name. VYM will replace the following strings while exporting:

```
<!-- INSERT TITLE -->    title of map
<!-- INSERT AUTHOR-->    author
<!-- INSERT COMMENT -->  comment
<!-- INSERT PAGES-->     content of map
```

The content itself is generated in a similar way by inserting lists into `page-template`. Here the following substitutions are made:

```
<!-- INSERT PAGE HEADING-->  heading of a page (mainbranch or child
                               of mainbranch, depending on the use of
                               sections)
<!-- INSERT LIST -->         all children of the branch above
```

Currently images are exported and notes just will appear as text without formatting and colours.