gvtree

a git version tree browser

Version 1.5-0

Abstract

gvtree is a graphical git version tree browser written C++ for Linux platform using Qt libraries. The main focus is the review of repositories, rather than changing code and developing. The main functionality is to select a node in the version graph and compare it to the current HEAD version, the direct predecessors or a selected version. Additionally a comparison between the current local changes and the local HEAD version is possible. A version history of a individual file can be viewed as well.

Table of Contents

History

Date	Version	Changes
September to December 2021	gvtree-1.1-0-beta.2	Initial Document Version
30. January 2022	gvtree-1.1-0-beta.4	Added History Revision of all Chapters Update of Screenshots "Current git status" dock widget
5. February 2022	gvtree-1.1-0-beta.7	Preferences Chapter Update of Screenshots
6. February 2022	gvtree-1.1-0	Release
2. March 2022	gvtree-1.2-0-beta.1	Revision of all Chapters
12. April 2022	gvtree-1.2-0	Release
2. October 2022	gvtree-1.3-0-beta.2	Revision of all Chapters
11. October 2022	gvtree-1.3-0-beta.5	Revision of all Chapters
14. October 2022	gvtree-1.3-0	Added Chapter 9 to Chapter 8
29. October 2022	gvtree-1.4-0-beta.3	Revision of all Chapters
29. November 2022	gvtree-1.5-0	Revision of all Chapters

Credits

Thanks to Winfried Nöth and Carsten Raufuß for beta testing and hints for improvement.

References

(1) https://doc.qt.io/archives/qt-4.8/classes.html

This is the class reference of the Qt Documentation Archives.

(2) /usr/lib/qt4/examples/graphicsview/elasticnodes

The elasticnodes was a good example to get started building up node and edge structures with QGraphicsItems.

(3) https://rachel53461.wordpress.com/2014/04/20/algorithm-for-drawing-trees

Rachel Lim's Blog, Algorithm for Drawing Trees
The description to draw a tree graph without collisions is very helpful.
For *gvtree* the step to distribute the middle nodes is not used.

OS and Build Environment

For the development Debian 9.4.0 has been used. The usage of Debian 9.4.0 explains the older Qt 4.8 version. The source code can be compiled as well with Qt 5 libraries. The program has been compiled and checked with the following two environments:

Debian 11 Environment

- g++ (Debian 10.2.1-6) 10.2.1 20210110
- qt5base5-dev (...) 5.15.2+dfsg-9
- xserver-xorg 1:7.7+22
- vim-common 2:8.2.2434-3
- git 2.30.2

Debian 9.4.0 Environment

- g++ (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
- libqt4-dev 4:4.8.7+dfsg-11+deb9u1
- xorg 1:7.7+19
- vim 2:8.0.0197-4+deb9u3
- git 2.11.0

Additional Dependencies

To work with the application git should be installed and an editor capable to compare files. The default is `gvim -d [file 1] [file 2] ... [file n]` to compare files and `gvim [file]` to show/edit the current local file.

To compare other objects like images or sound or perhaps pdf documents, the mime type of a file can be mapped to an appropriate tool.

Build

After extracting the source package:

tar -vxzf gvtree-1.5-0.tar.gz

Change to the folder `gvtree-1.5-0`

cd gvtree-1.5-0

Now, just run the following command

qmake
make

To execute the program enter:

./gvtree

If you like to install it to a $PATH directory, e.g. `/usr/local/bin`, use the following commands instead:

qmake PREFIX=/usr/local
make
sudo make install

Now you can just type:

gvtree

Command Line Arguments

With command line argument **-h** the following information is printed:

```
gvtree-1.5-0

Tool to display git log graph

gvtree Copyright (C) 2021 Wolfgang Trummer

  This program comes with ABSOLUTELY NO WARRANTY
  This is free software, and you are welcome to redistribute it
  under certain conditions

  This program is licensed under
  GNU GENERAL PUBLIC LICENSE
  Version 3, 29 June 2007

--------------------------------------------------------------------------------

Parameters:

[path]

   Set a file constraint. The version tree of the file will
   be displayed.

-r [local git repository directory]

   If not specified the current path is checked for a valid repository
   or the repository used in the previous session is displayed.
   Which one is used can be controlled by the preferences setting.

--version Version string is printed to stdout

--silent true|false Silent mode.

   If true, commands are not printed to stdout. The preferences
   'print commandline to stdout' is set to this value.

--css [style sheet file]

   Load a css style sheet file.
   If not specified the last file used will be taken.
   Perhaps it is a good idea to copy gvtree.css to ~/.config/gvtree
   and run ./gvtree --css ~/.config/gvtree/gvtree.css once.

-t Testing:

   Display the test tree graph from (3).

-f [gitlog]

   Testing:
   Load a file created with
     git log --graph --decorate --pretty="#%h#%at#%an#%d#%s#"
   This has been helpful during development to import constraint and
   complex repository data.

-h This information.

--------------------------------------------------------------------------------
```

The arguments **-t** and **-f** are just for testing the rendering of the graph and the parsing the **git log** output.

With the first start `~/.config/gvtree/gvtree.ini` is created. The window state and the preferences are saved there.

It is a good idea to copy the file `css/gvtree.css` to `~/.config/gvtree/gvtree.css` as well and run

./gvtree --css ~/.config/gvtree/gvtree.css

once. The css file path is then written to `gvtree.ini` and always used.

The gvtree.css file can be customized before.

The path to the css file can be changed in Windows - Preferences - Basic Settings menu as well.

The default directory for temporary files is `/tmp` it can be changed in the preferences to a different directory, too.

Tutorial

The following sections describe a walk trough the functionality of *gvtree*.

Step 1 Sample git repository

To show the functionality of *gvtree* a sandbox repository is created with the following steps.

The directory to start with is `/home/gvtree`.

Create a subdirectory `test_repository`

mkdir test_repository

Change into the new directory and initialize a new git repository

cd test_repository

git init

Create a file `main.c`, perhaps with the following content:

```
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char* argv[])
{
  printf("Hello world!\n");
  return 0;
}
```

Add the file to the repository

git add main.c

and commit it.

git commit

Now just run *gvtree* for the first time.

gvtree -r /home/gvtree/test_repository

or, if you are already in the directory `/home/gvtree/test_repository` just start

gvtree

The result should look like this:

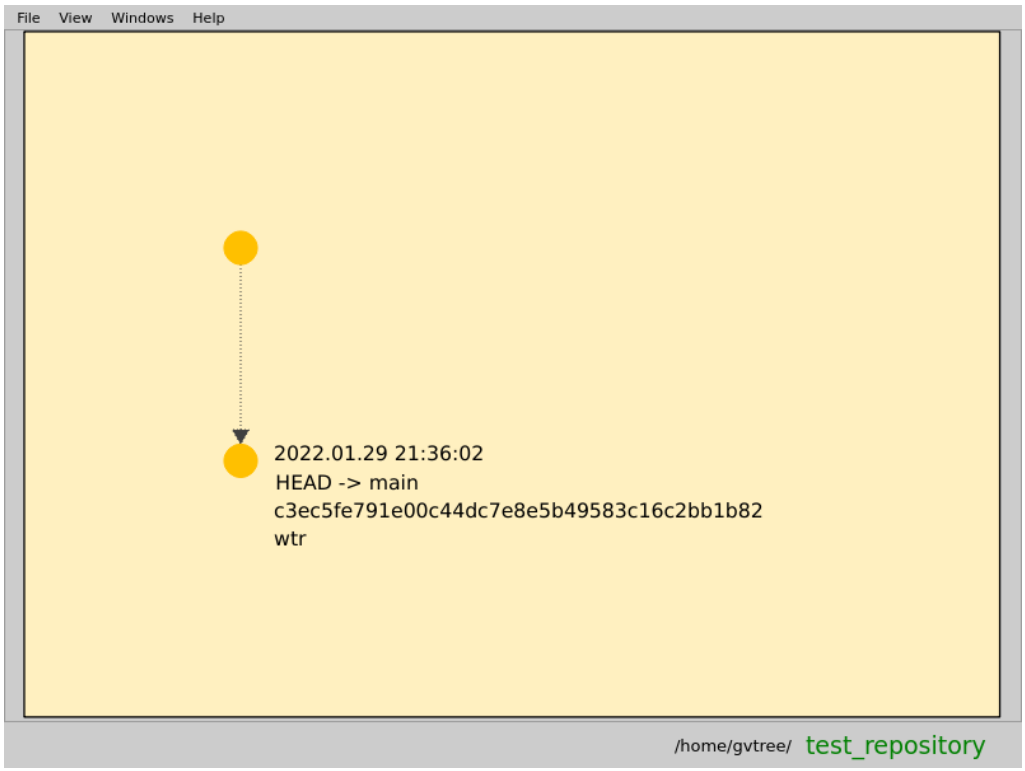

Figure 1: Initial window layout

The window's minimal size is 400x400 pixels. The default size is 800x600.

At the moment the version tree for the current local repository contains only one version. The first displayed node is the empty root node. The information attached in this example is the commit date and time, tag and branch information, the git version hash and the user name.

Now open the top menu Windows and tag all dock widgets Version Information, Current git status, Compare Versions, Search Version and Branch List.

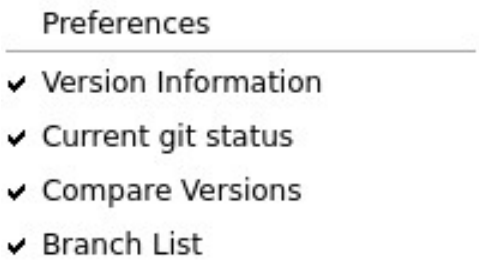

Figure 2: Windows menu

The main window should look like this, then:

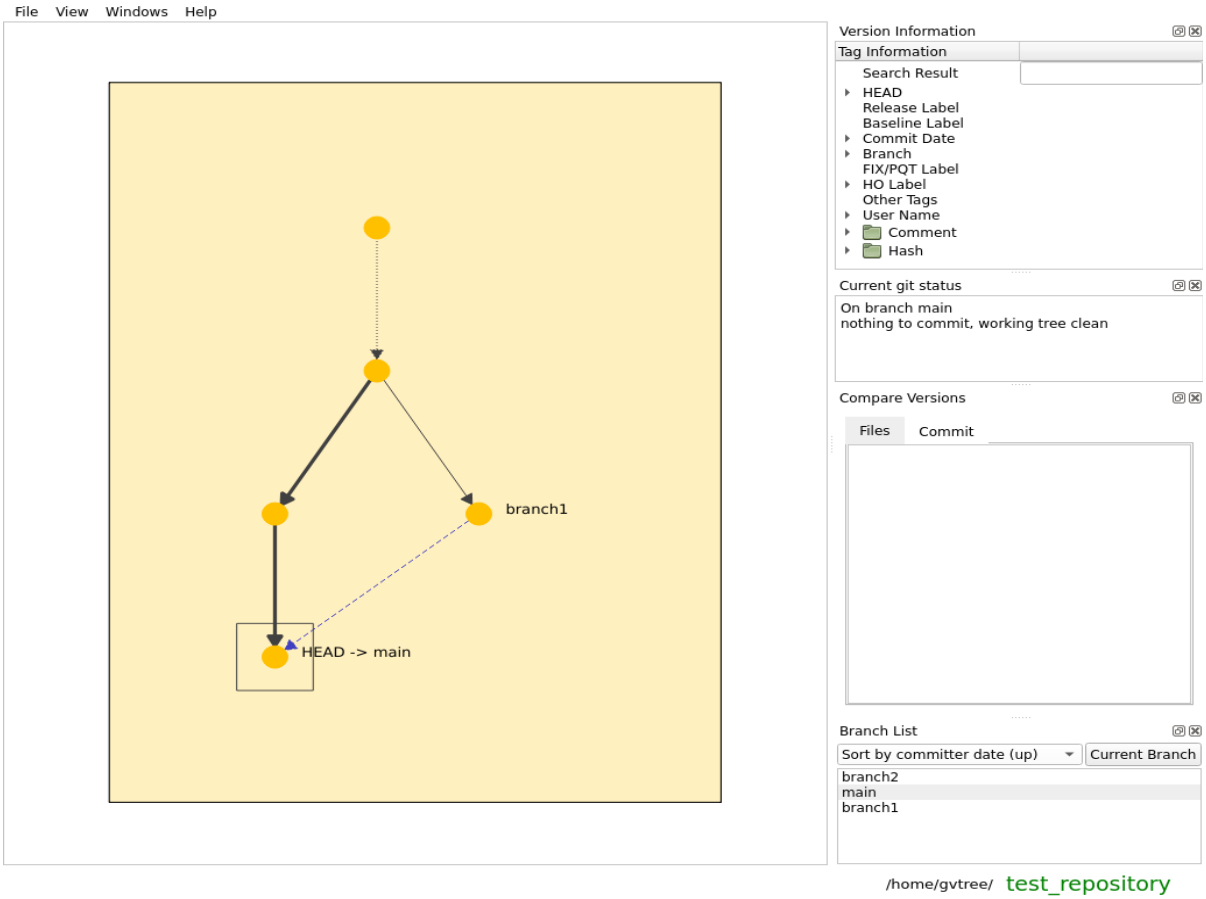

Figure 3: All dock widgets open.

- On the left side there is the graphical representation of the version graph.
- On the right side there are four dock widgets.
  - Version Information contains a tree widget with selection options for commit date, commit user, certain git tags, the git hash value and the commit commentThe search dialog is integrated into this widget, too.
  - The Current git status section just shows the output of **git status**.
  - The Compare Versions section is filled as soon as versions are compared.
  - The Branch List shows the current selected branch. If the selection is changed, the main view is adapted to the corresponding branch and the latest version of this branch is focused.

For the moment close all right dock widgets again.

Control	Keyboard	Mouse
Fit in view	1	
Focus latest version on the current visible branch. In most cases this is the local HEAD version.	h	
Focus HEAD version	o	
Zoom in	+	Wheel up
Zoom out	-	Wheel down
Pan	wasd	MMB + Move
Pan (faster)	Shift + wasd	
Pan	STRG	LMB + Move
Select version		LMB
Context menu		RMB
Search version	STRG + f	

Select the main view and press the key 1 to adjust the graph to fit into the view port.

Now create a branch

git branch branch1

and check out this branch.

git checkout branch1

Add a README file containing "YYY" in the first line.

git add README

git commit

Refresh the *gvtree* view by opening File menu and select Reload Repository

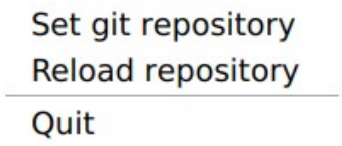

Figure 4: File menu

After the update the graph looks like this:

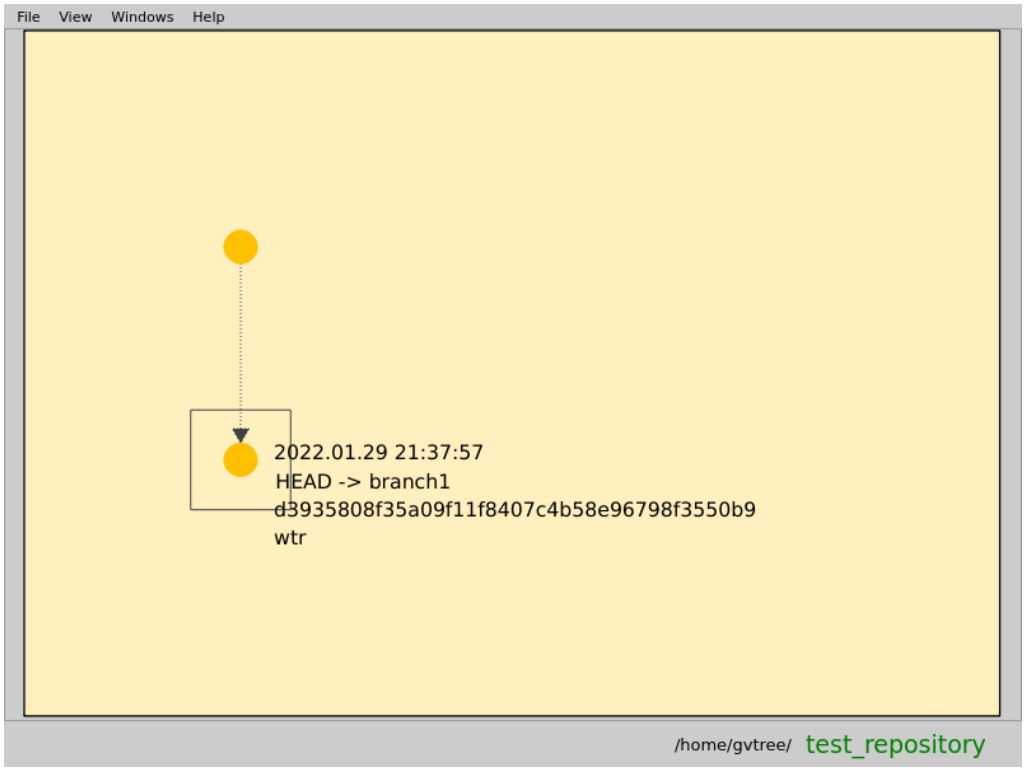

Figure 5: Update after repository change

RMB click in the box containing the version node and the following context menu will appear:

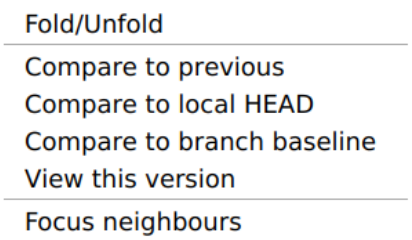

Figure 6: Version context menu

Selecting Fold/Unfold will lead to this update:

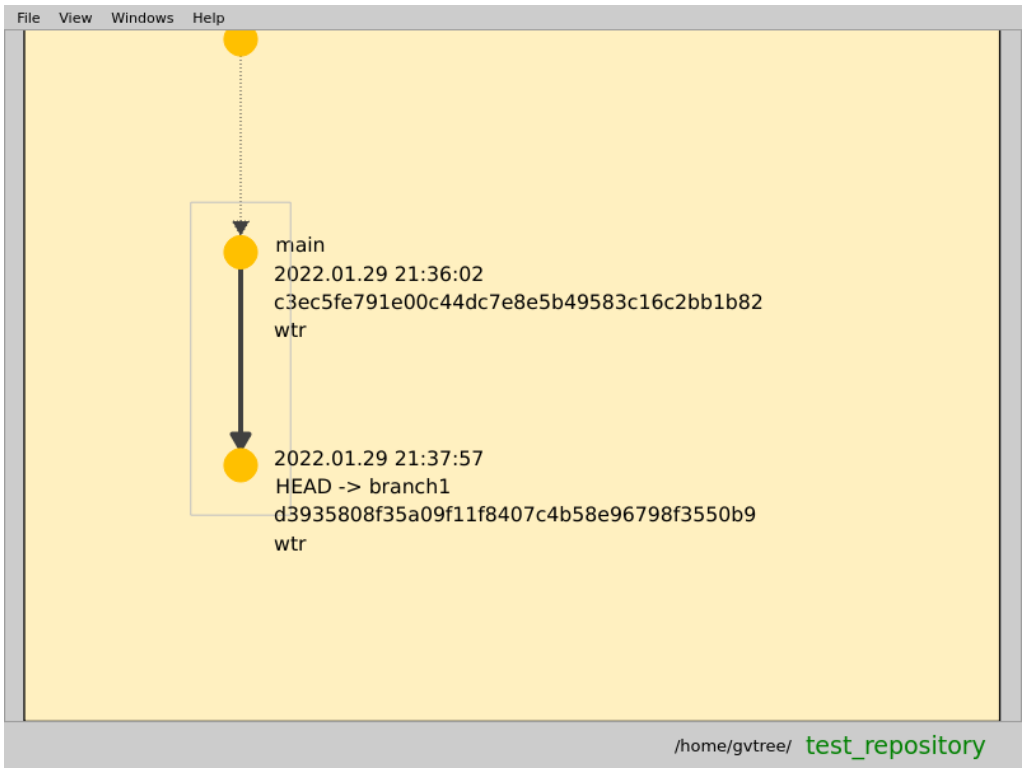

Figure 7: View after expanding a folder

All versions without incoming merges or outgoing branches are folded automatically.

Now check out main again.

git checkout main

Again, create a README file with different content "XXX".

git add README

git commit

Update the *gvtree* graph view again.

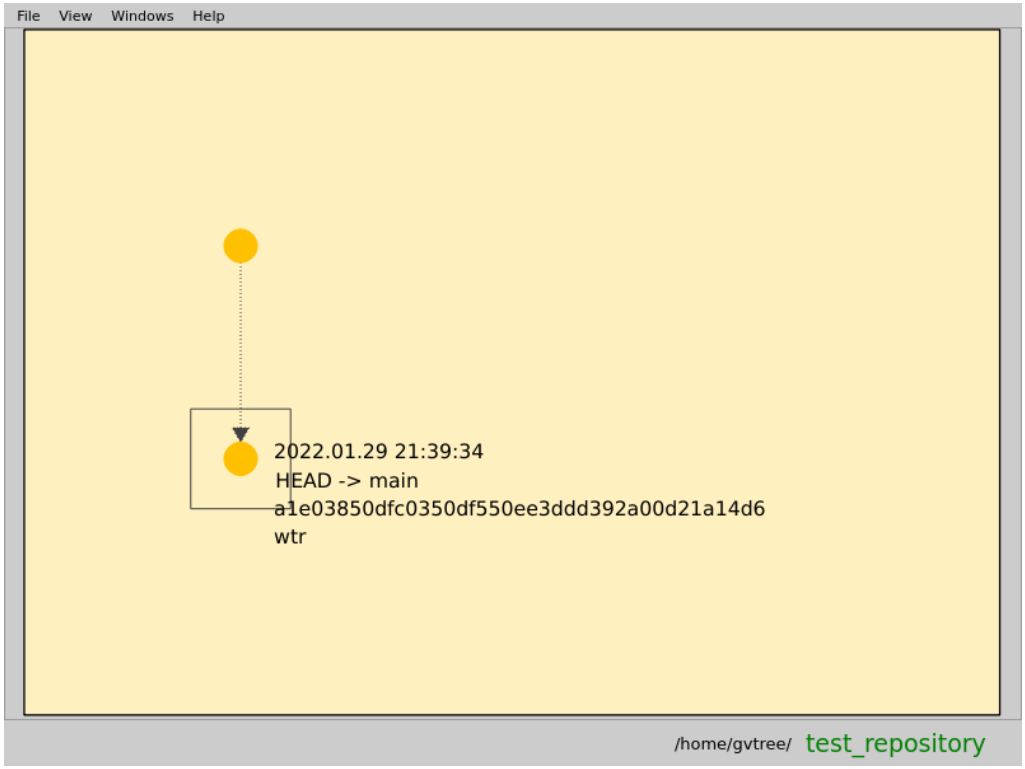

Figure 8: Changed branch back to main

Now merge branch1.

git merge branch1

Solve the merge conflict in README to have two lines "XXX" and "YYY".

git add README

git commit

Perhaps you have recognized the Reload repository button already. It appears if a change of the `.git` directory in the local repository has been recognized. Pressing it has the same effect like File menu and Reload repository.

Figure 9: Reload repository button

After the refresh, the graph should now look like this:

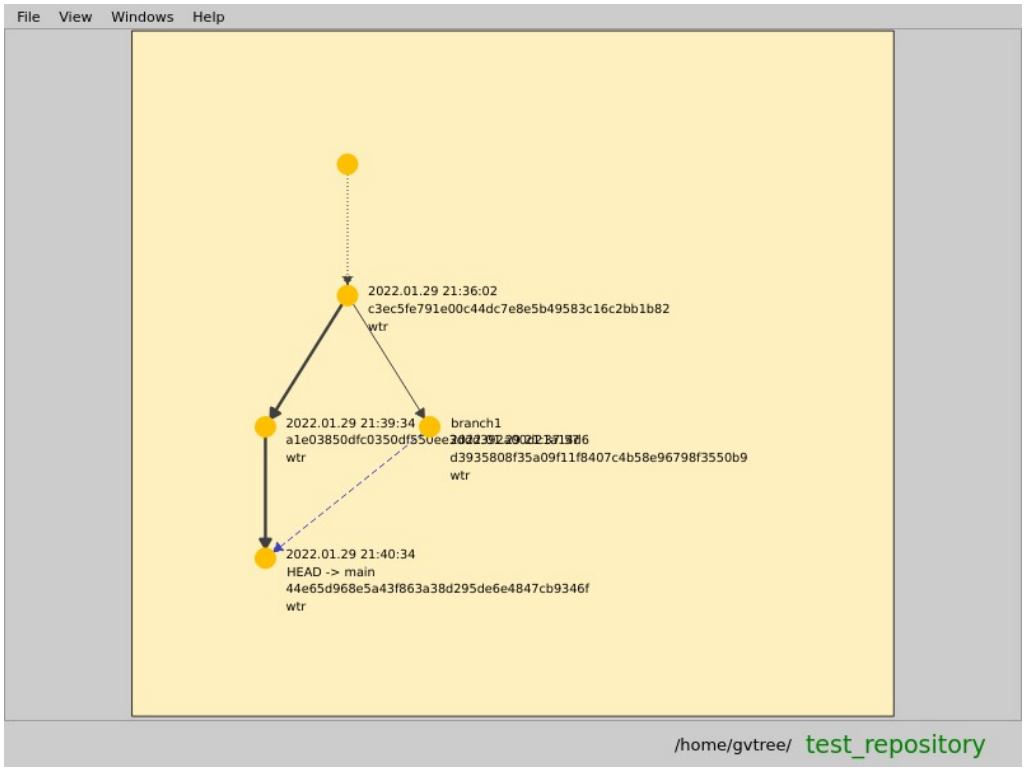

Figure 10: Version graph with merge

The edge representing the merge is displayed dashed and has a different color.

Versions without a real or not displayed predecessor are linked to the zero root node with a dotted edge.

Now improve the layout and hide the git hash value:

Open View menu and change the settings to:

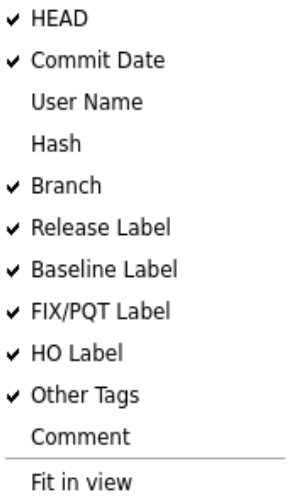

Figure 11: View menu

The hash and commit user information is not displayed anymore.

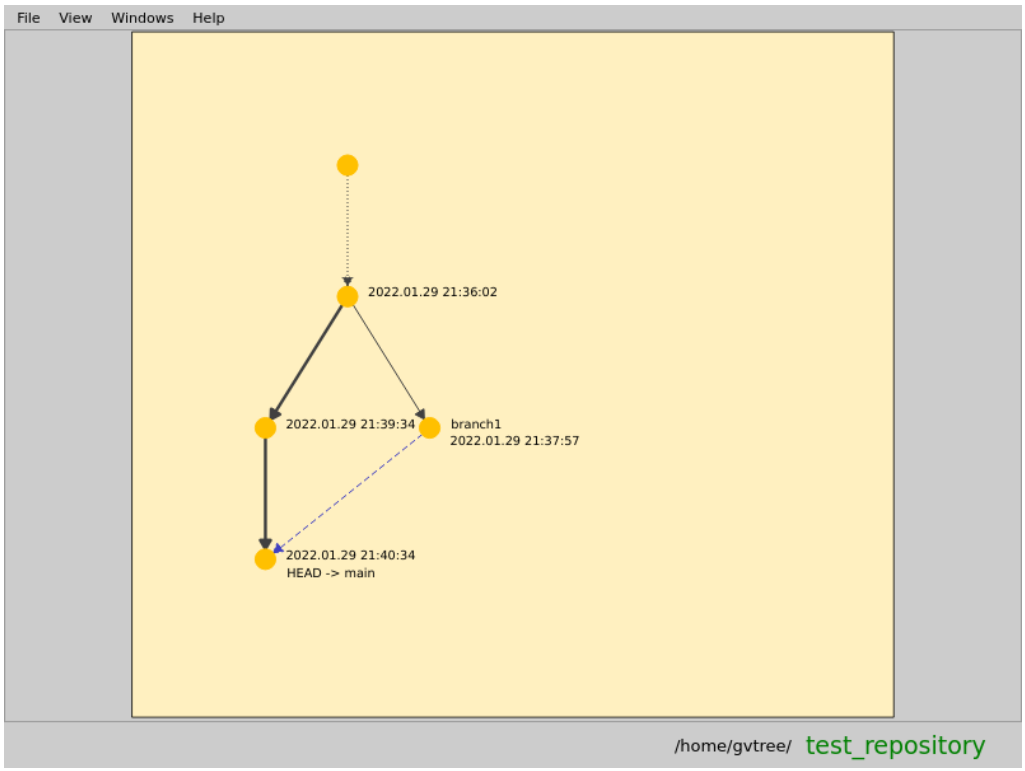

Figure 12: Hidden git hash values

The space between commit date and an other node is still very small. To correct this, open Windows menu and select Preferences.

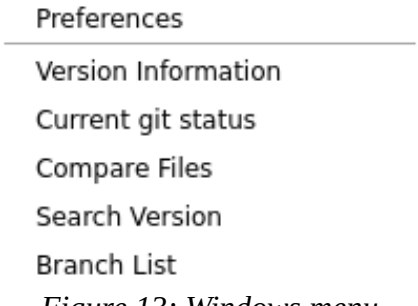

Figure 13: Windows menu

In the dialog select the tab Rendering.

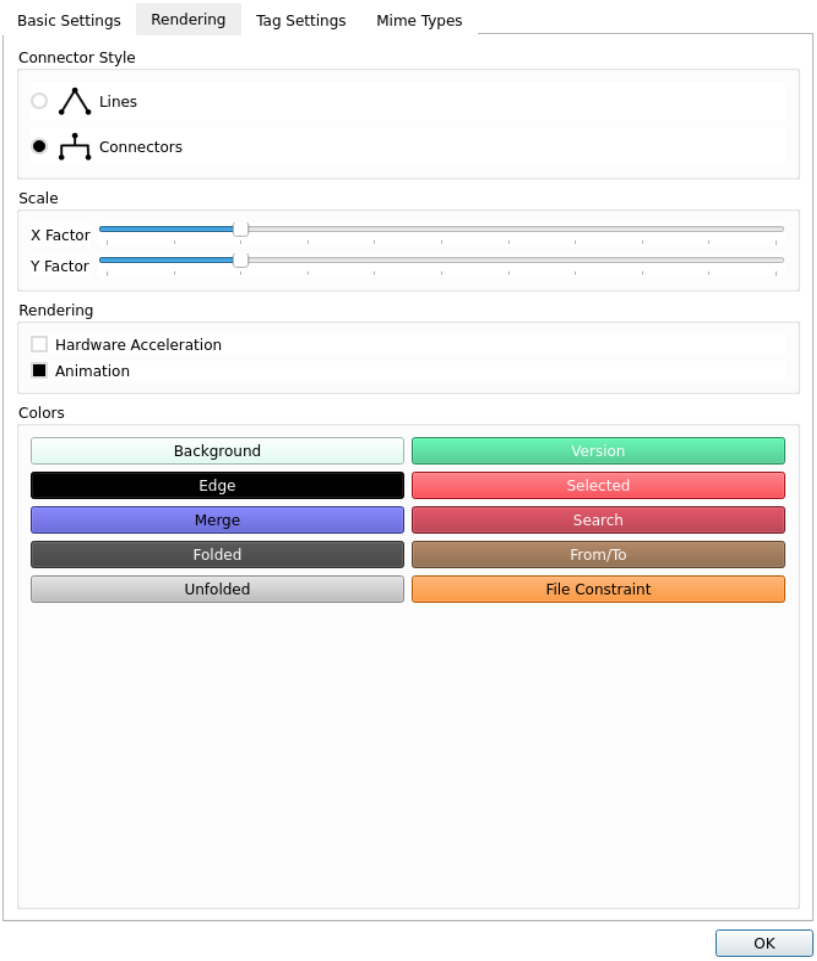

Figure 14: Preferences dialog, page Rendering

Increase the value of the two sliders X Factor and Y Factor, then press OK.
To fit the whole graph into the screen, press key 1 in the main view again.

The result should look like this:

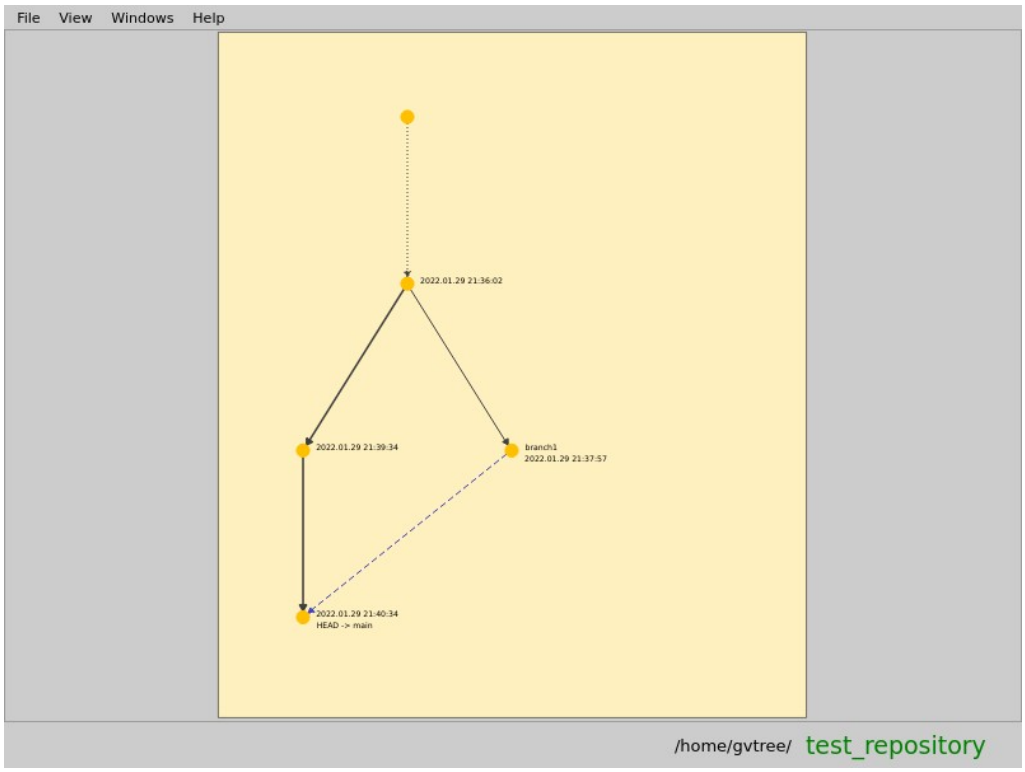

Figure 15: Scaled view

Step 2 Compare Versions

In the Windows menu, select the dock widget Compare Versions. Detach the Compare Versions dock from the main window.

Now do a RMB click on the version node with the branch1 information.

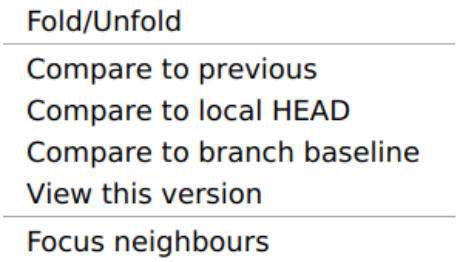

Figure 16: Context menu of a version node

In the context menu select Compare to previous.

A markup cursor appears to identify the two versions which are compared.

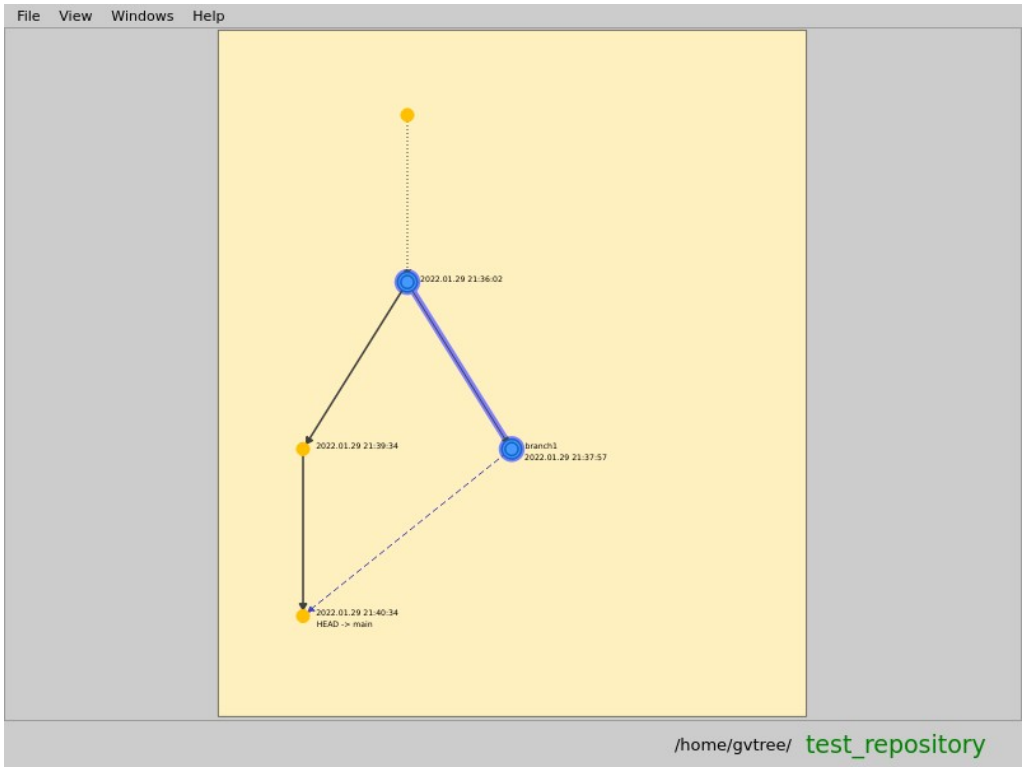

Figure 17: Compare versions markup

The Compare Versions dock should now look like this:

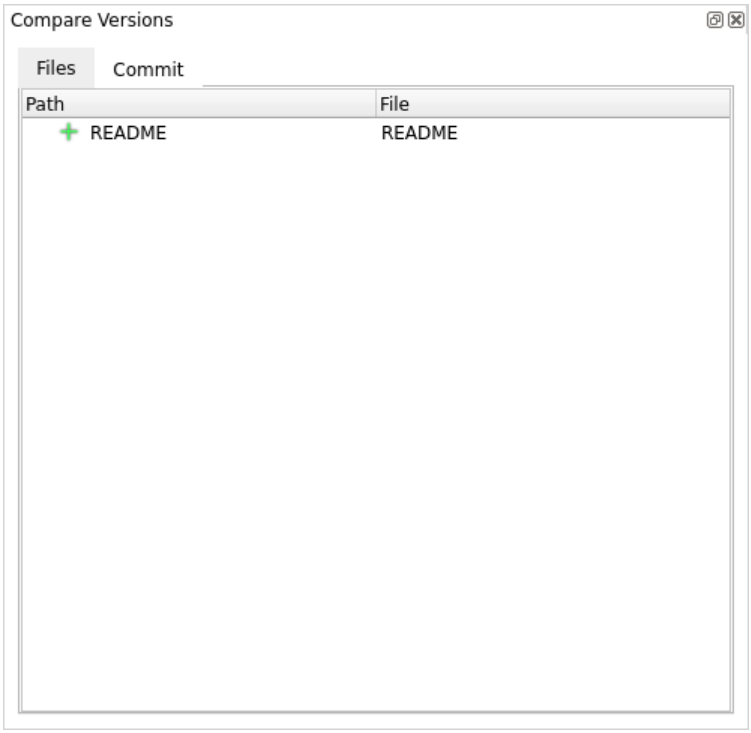

Figure 18: Files tab of the Compare Versions dock window

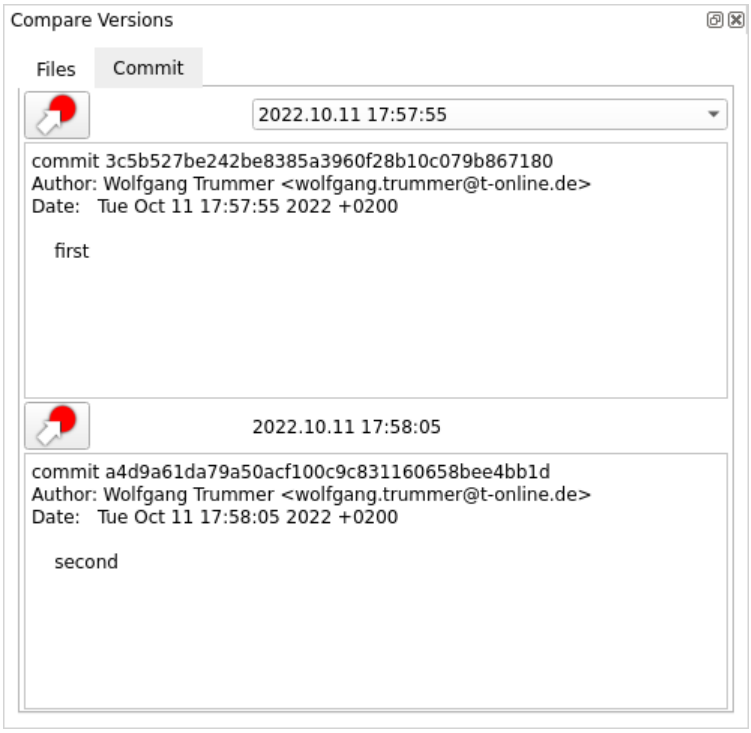

Figure 19: Commit information tab of the Compare Versions dock window

In a tree view all changed files are listed. In this example it is only the README file.

The left text browser below contains the commit information of the from-version, the right text browser contains the commit information of the to-version.

By pressing the button above the commit info, the corresponding version in the graph view is focused and is marked up.

Figure 20: Focus version

It will look like this, then:

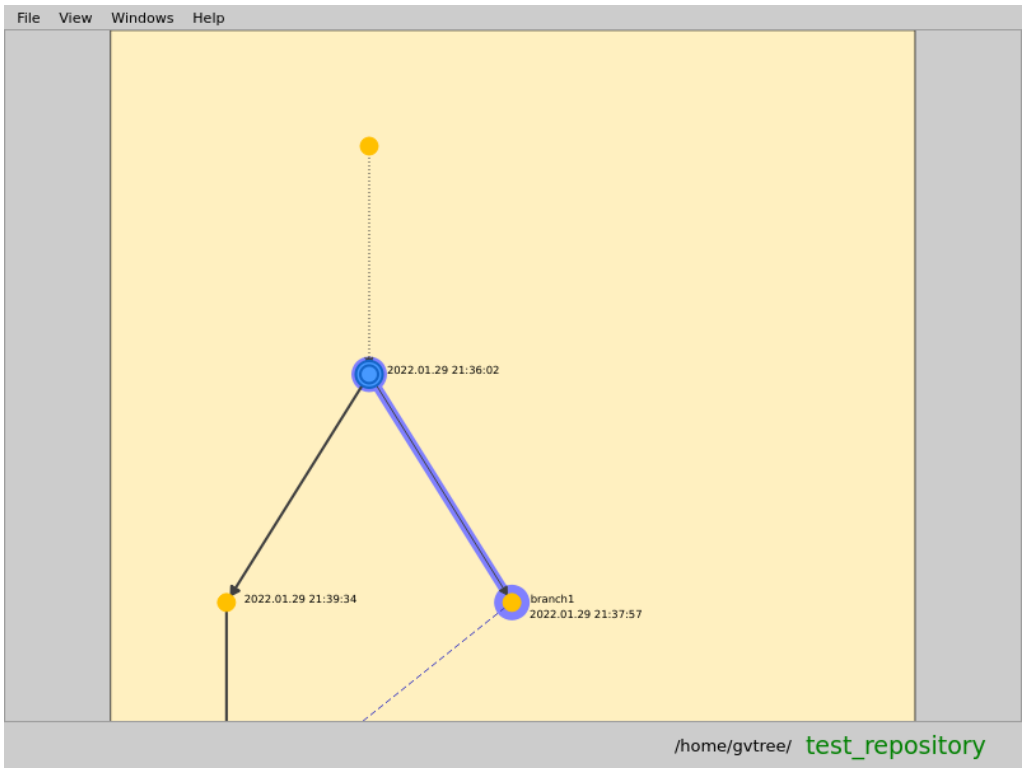

Figure 21: Focused version

Now do the same for the HEAD version.

This version has got two predecessors because of the merge.

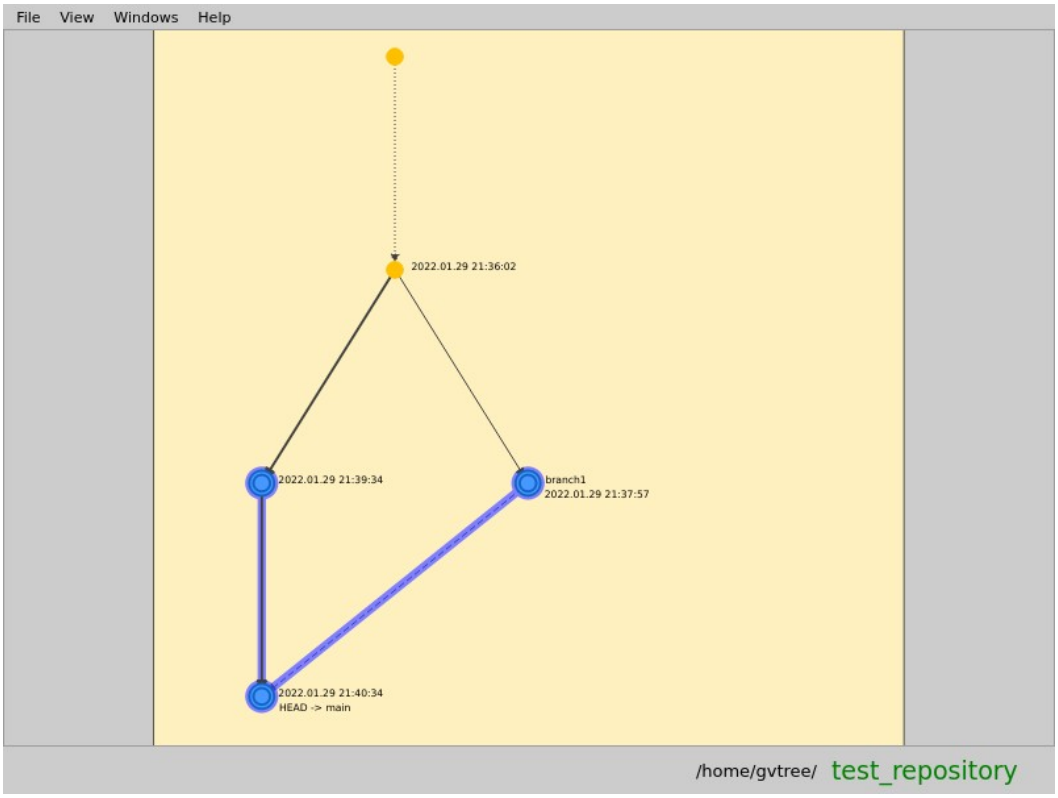

Figure 22: Compare to more than one predecessor

The Compare Versions window has changed, too:

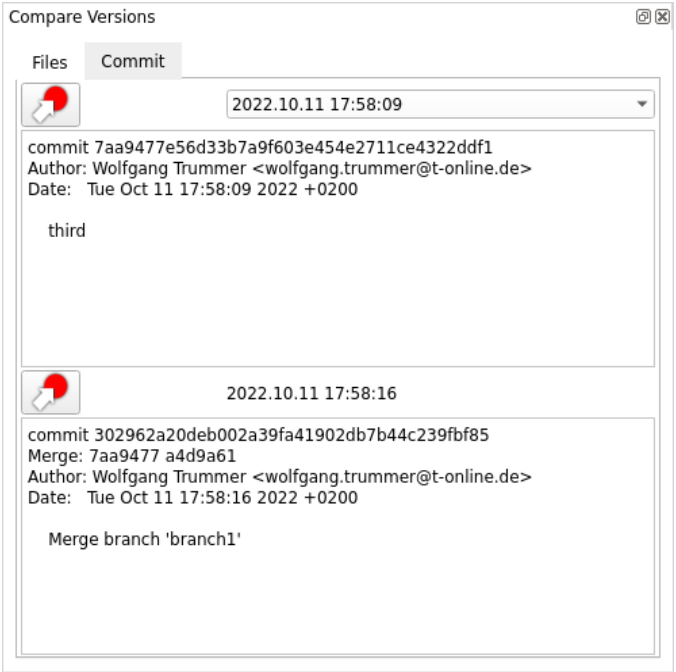

Figure 23: Updated Compare Versions window

In the Files tab of the Compare Versions window the symbol in front of README has changed, too.

	File has changed / modified
	File has been removed
	File has been added
	File has been renamed

The second difference is, that the from-version is selectable by the combo box.

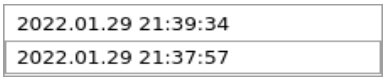

Figure 24: Combo box

The displayed commit info will change accordingly. Pressing the focus version button will focus and markup the selected version.

In the file tree view select the README file and open the context menu with a RMB click.

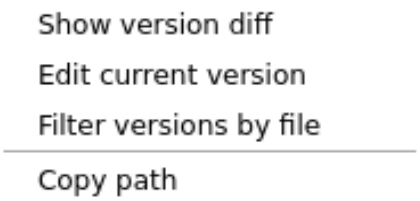

Figure 25: Context menu for a single file.

Select Show version diff.

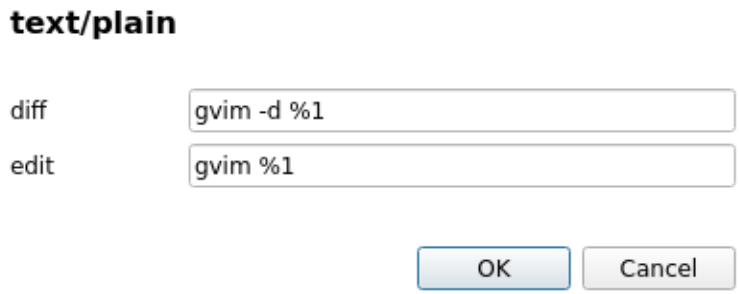

Figure 26: If mime type is unknown the tool selection dialog is opened.

In this case a dialog will open, because so far the mime type of the file `text/plain` is unknown and not linked to a viewer or an editor. The setting can be changed later in the Preferences dialog. The %1 is a placeholder for a list of file names separated by a blank.

Pressing OK now, a *gvim* opens with three columns - two from-versions and one to-version.

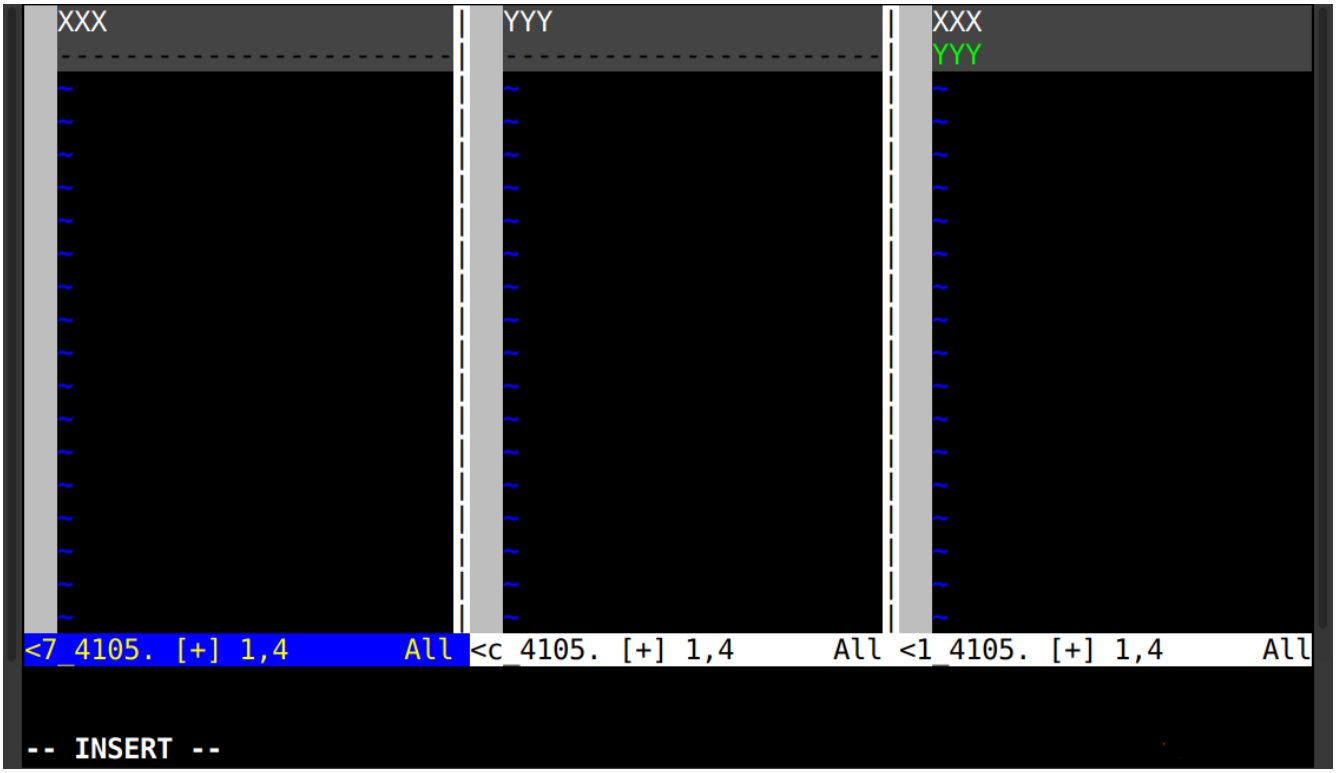

Figure 27: gvim as difftool with three columns.

In case of an image file instead of a text editor *gimp* for example can be specified, for pdf documents *evince* and perhaps for sound files *aplay*.

If you open the RMB menu of the HEAD version again and select Compare to branch baseline the current version is compared to the first direct parent with more than one child. In the example above it would be the version with the timestamp 21:36:02.

If there is no branch other than main, the branch baseline is the first version displayed after the root node.Now choose again the RMB click context menu of the README file in the tree view. Select Filter versions by file. The effect is, that all versions and edges are marked up, where this file has been changed. The file name constraint README is added to the bottom status bar.

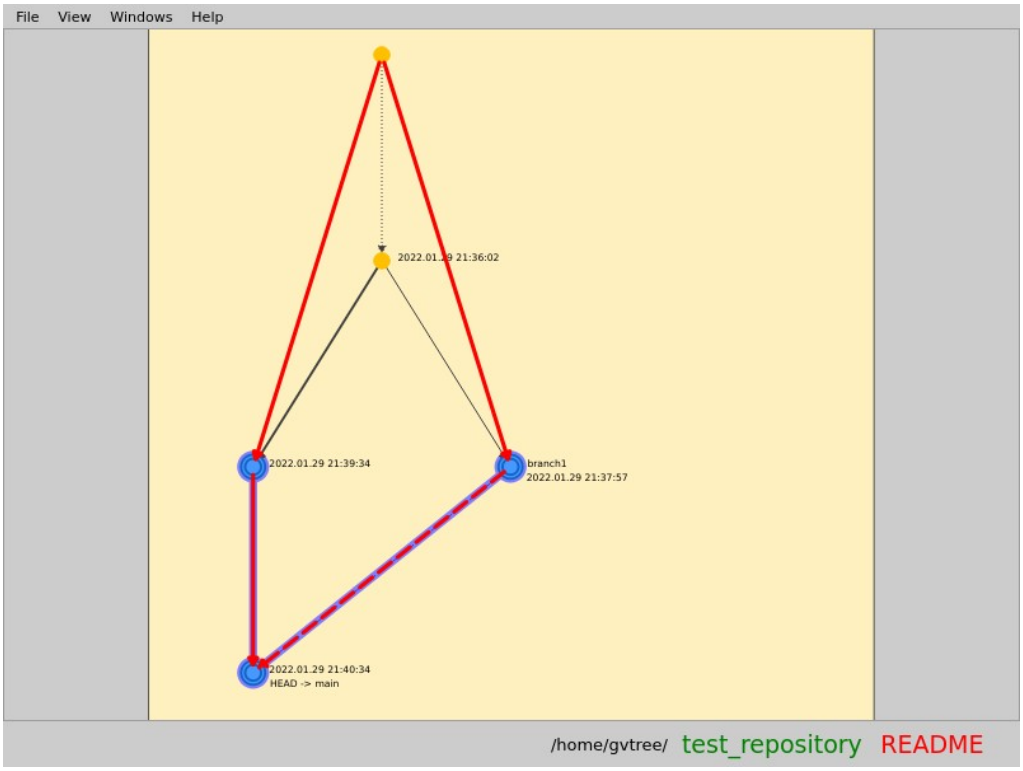

Figure 28: Version graph with file constraint

A LMB click on README in the status bar will remove the constraint again. Selecting Remove filter in the tree view context menu has got the same effect. In a larger version tree it is more simple then to find out, where a file has been altered.

The last item in the context menu Copy path just copies the file path to the desktop's clipboard.

Now exit *gvtree* by selecting Quit in the File menu.

A file constraint can be added when starting *gvtree*, too.

In a new terminal window change to `/home/gvtree/test_repository`

cd /home/gvtree/test_repository

gvtree README

Remove the file constraint like above.

The changed settings and the window state have been restored. The Compare Files dock should be detached and visible. The spacing between the version tree nodes should not have changed.

Step 3 Preferences

Now open the Preferences dialog again.

Rendering

The Rendering tab has already been visited. Feel free to change your color settings or change to the other connector style. The Hardware Acceleration will try to use a `QGLWidget` for the graph view, if possible. The Animation checkbox is set by default. When the view is re-focused it is hopefully scrolling and zooming smoothly to the version nodes of interest. If too slow, without the checkbox set, the view is changed directly.

Basic Settings

The Current Repository Path can be set and changed here. The Initial Repository Path setting defines if the current path is used to look for the git repository. The other option is to display the repository which was used in the last session.

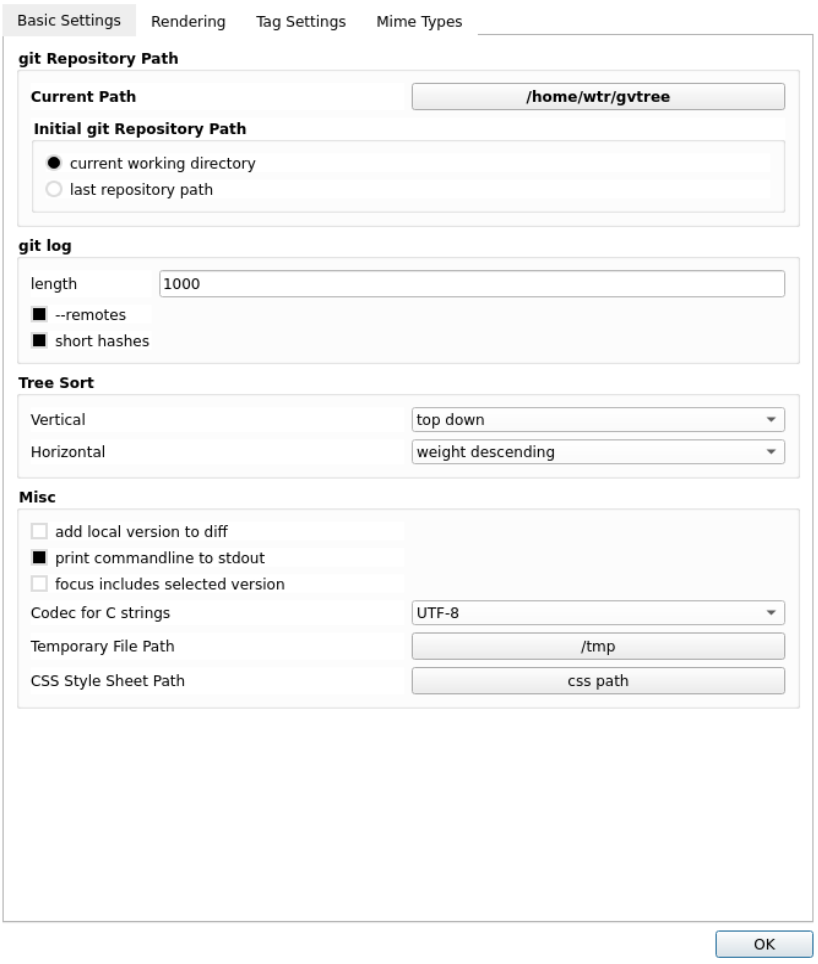

Figure 29: Preferences Basic Settings

In the next section

The tree information is imported from a git log output. This input can be truncated to the last n versions. The tree will be smaller and less complex, then. A good value for git log length is 1000. Initialization of a huge tree graph takes some seconds.

git log –remotes adds the –remotes switch to git log actions.

The short git hashes check switches between %h and %H output of git log.

In case of top down view is checked in the Vertical Tree Sort section, the HEAD version is printed on the top. The Horizontal sort takes just the git log --graph order in case of natural. The weight of the subtrees can be used as sorting criteria, or the commit date.

If add local version to diff is set the local file version is displayed if it is not equal to one of the other versions to compare.

With print command line to stdout, every command line to run git or a compare tool is printed to stdout.

The focus contains selected version will ensure that the selected version is always contained in the view when searching other versions.

The Codec for C Strings is only relevant for *gvtree* compiled with a Qt version < 5.0.

The css file located in the source tree `css/gvtree.css` is included when compiled. In case no path to a different style sheet file is set this default is used. The default can be changed by referencing a file in the CSS Style Sheet Path. (An empty file for no style sheet is allowed.)

Temporary files are created when comparing different versions. This files are erased if gvtree is quit and the location is specified in the Temporary File Path setting.

Tag Settings

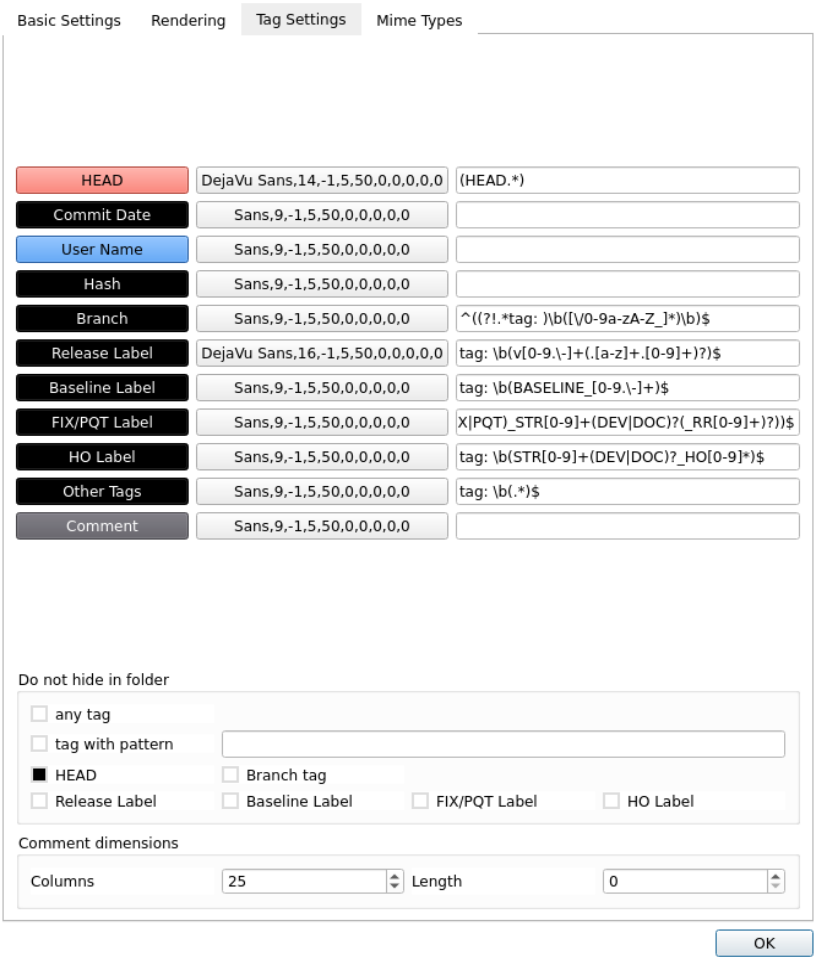

Figure 30: Preferences Tag Settings

In the upper section color and font style for certain version information is defined. For some information it is possible to edit a regular expression to match git log data. Depending on how a working flow for a project is defined it is possible to adapt baseline, release or QA label information. In this example, a Release Label looks like **R1.2-3-4** or similar. A baseline label e.g. **BASELINE_1.2-3**. Other patterns are perhaps possible, but tricky.

FIX, **PQT** (preliminary quality test) and **HO** (handoff) are perhaps project or company specific and are related to a QA work flow.

At the moment it is not possible to add more patterns by config, but in the code it is simple to add more tags (look for `gridLayout->addTagPreference(...)`).

The Do not hide in folder section controls when a version is not added to a folder.

If commit Comment information is displayed in the graph the maximum Length of the comment and the number of characters per line Columns can be defined in the section Comment dimensions.

In the local repository, add a tag named **STR1234_HO**.

git tag -a -m "STR1234_HO" STR1234_HO

Update the graph view.

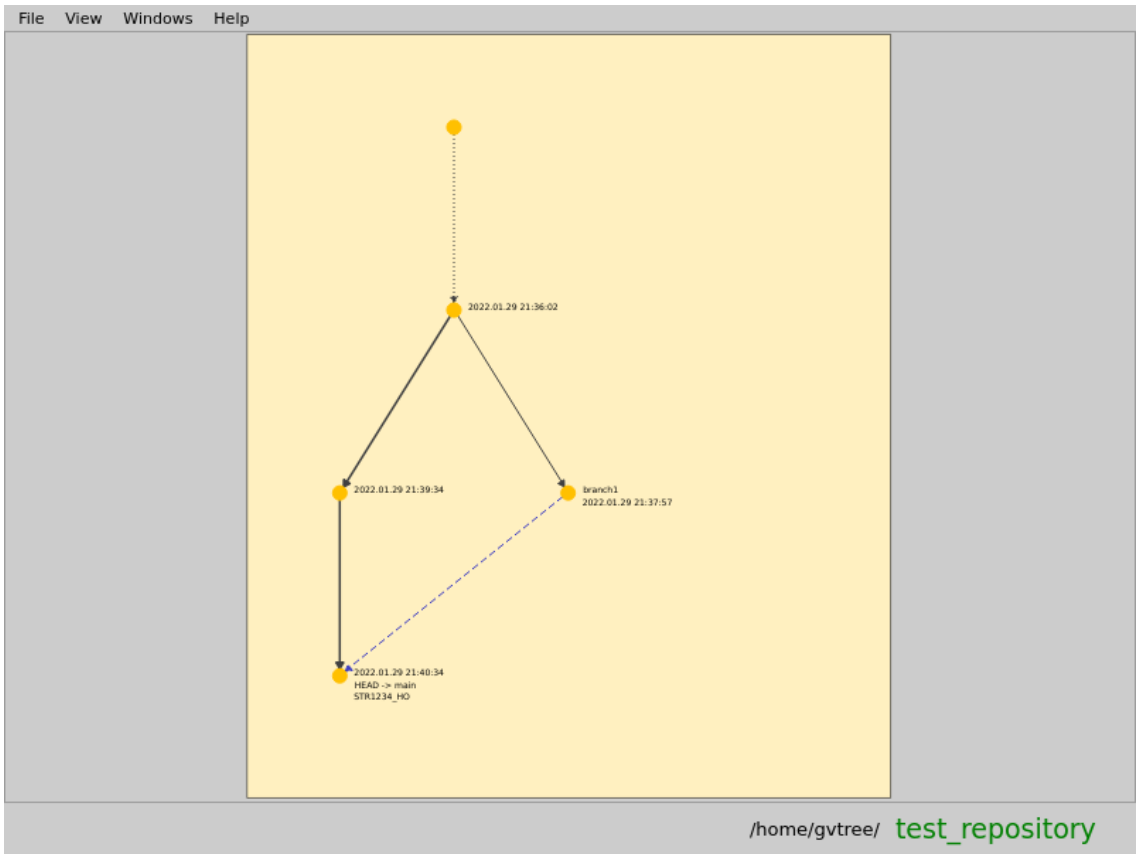

Figure 31: Handoff Tag

The visibility of the HO Label can be controlled with the View menu. Switch off HO Label.

Mime Types

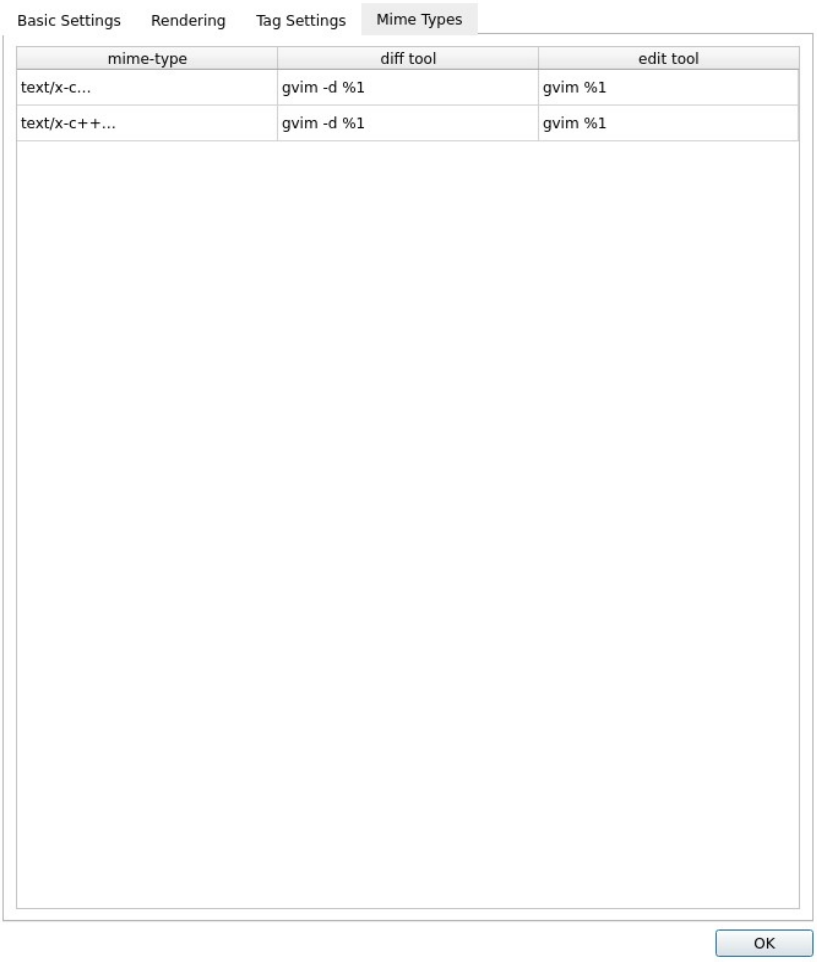

Figure 32: Preferences Mime Types

In Step 2 the diff tool and edit tool preference for `text/plain` files has been added.

The columns diff tool and edit tool can be changed here, if necessary.

Step 4 Some more features

The current local repository can be changed by:

- A file dialog which is opened by the File menu Set git repository.
- The same file dialog which appears when pressing the repository name in the status bar of the main window.
- It can be opened by pressing the Current Repository Path button in the Preferences on the page Basic Settings.
- When starting *gvtree* the local repository path can be handed over with the command line argument **-r** followed by the path.
- If the **-r** parameter is not specified when starting, depending on the Preferences Basic Settings Initial Repository Path the current path is checked for a git repository. If last repository path is selected the repository of the last session is used.

The Help menu offers three selections.

- Help will show where to find this document ($INSTALL_PATH/share/doc/gvtree/gvtree-1.5-0.pdf).
- About shows a nice splash screen with the project icon.
- License contains the HTML copy of the GPL V3.0

Step 5 Folder

In the Windows - Preferences - Basic Settings set the tag fold HEAD version.

Now just add a file TODO to the repository.

git add TODO

git commit

Refresh the graph view.

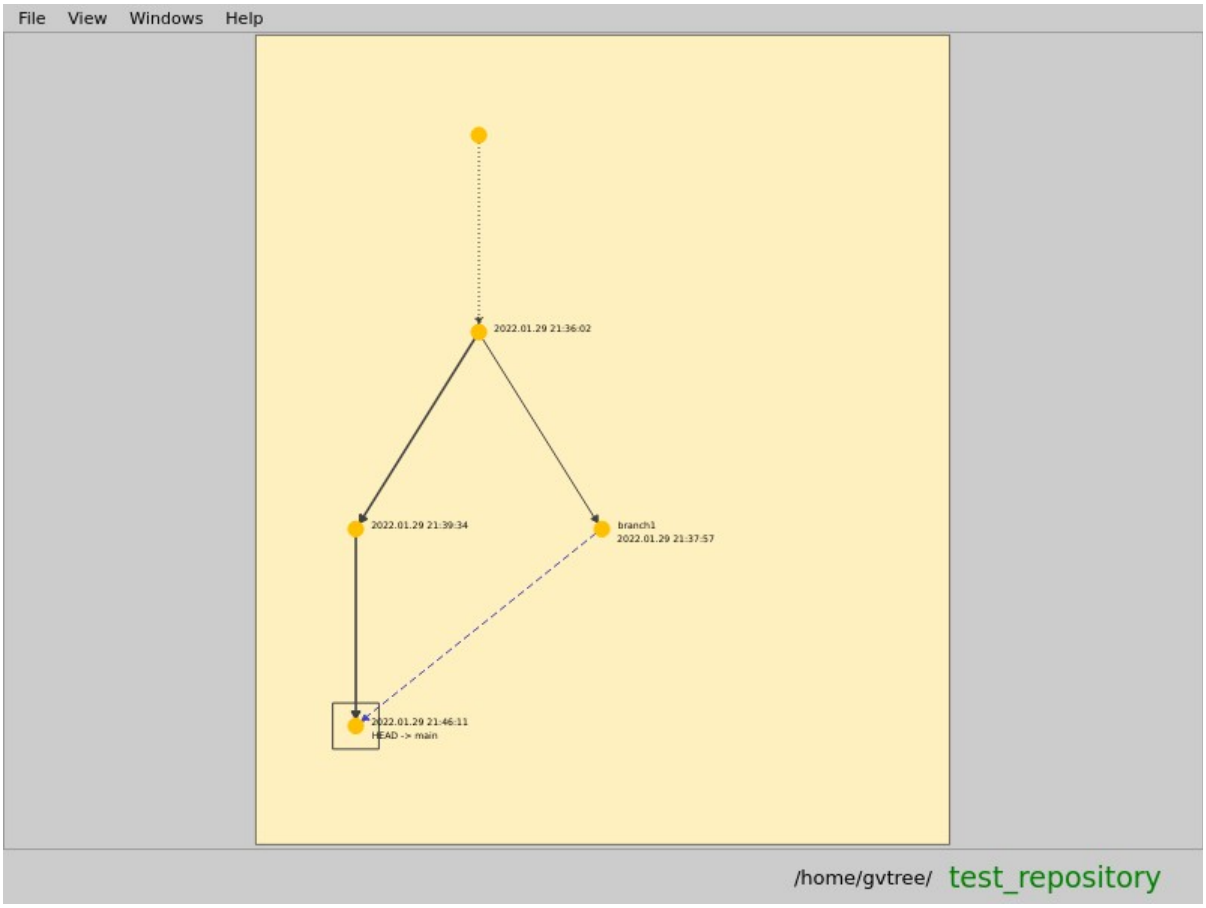

Figure 33: Folded version

The folder box now contains two versions. If there is no branch or merge, versions are folded and only the last version node of the folder is displayed.

Do a RMB click in the graph view background and open the following context menu:

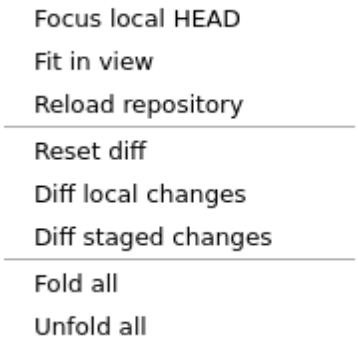

Figure 34: Global background context menu

Fold all and Unfold all are global actions in this context menu. With the context menu of a single folder the action will only affect the one element.

Focus local HEAD is helpful in case of bigger version trees. To do the test, just zoom into the view with the mouse wheel or pan with the MMB middle mouse button pressed.

Fit in view ensures visibility of the complete version tree.

The Reload repository is the same like the option in the File menu.

Open the context menu again and select Focus local HEAD.

The result should be:

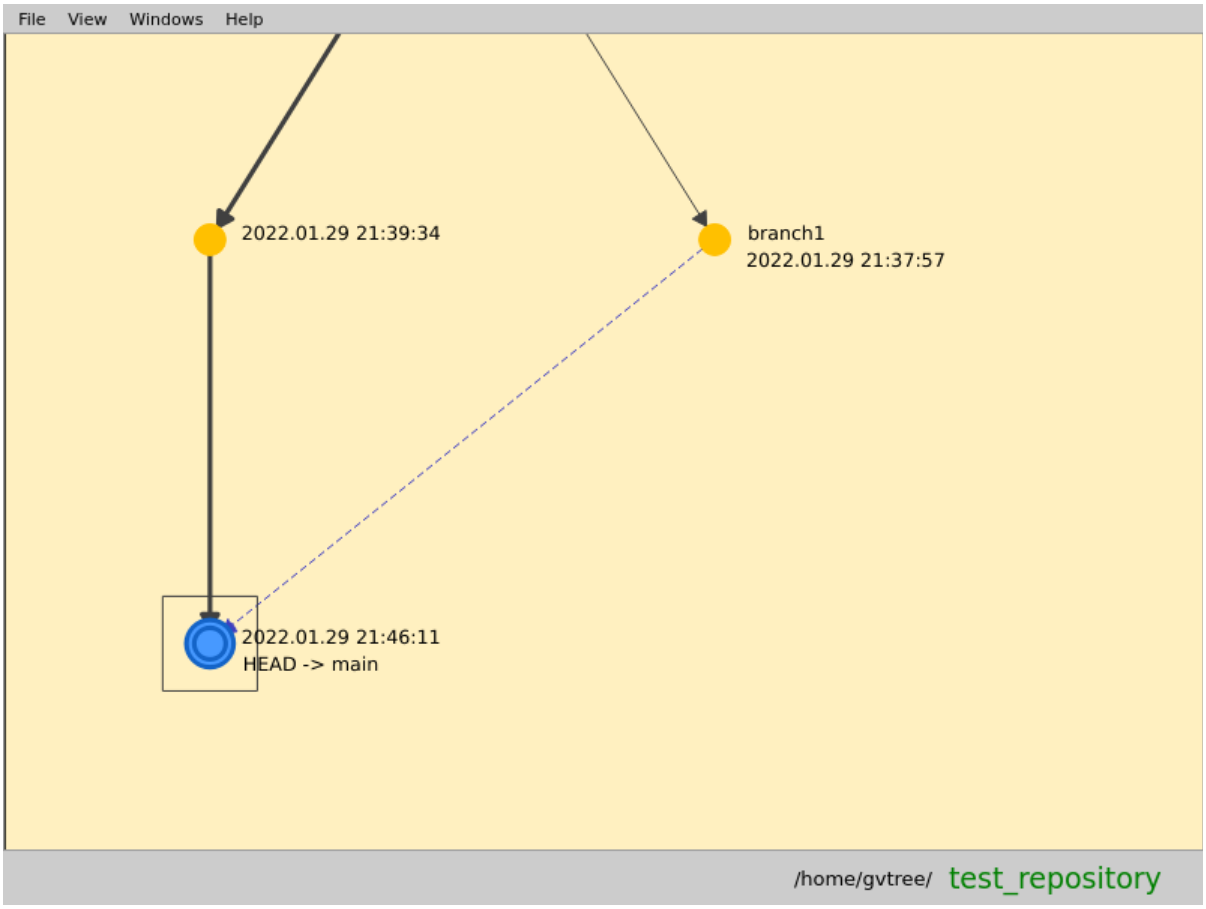

Figure 35: Focus HEAD version

To focus the HEAD version there is the additional keyboard shortcut h.

Now edit the TODO file. Add the line "This is a local change.".

In the RMB click context menu select Diff local changes. The Compare Files dock will open.

Open the context menu of the file TODO and select Show version diff. Now the local changes are compared to the current local HEAD version.

Files already staged with git add are not visible in the Diff local changes. The difference between the latest commit and the staged files can be done by the menu item Diff staged changes.

In the graph view select Reset diff in the RMB context menu. The markup of the local HEAD version disappears and the content of the Compare Files window is removed.

Step 6 Selecting a version

With a LMB click a version can be selected. The node will appear in a different color.

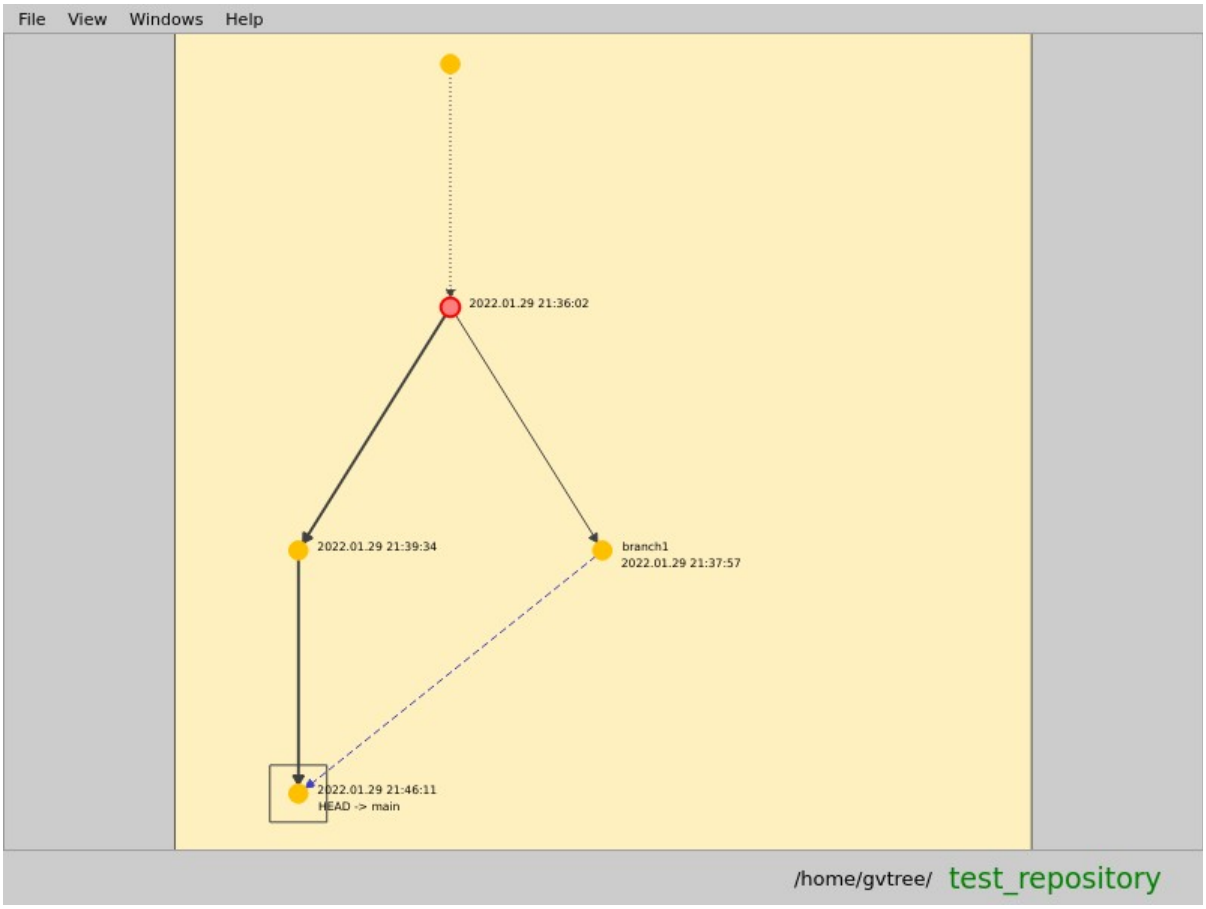

Figure 36: LMB version selection

Now open the RMB click context menu of the version with **branch1** information attached.

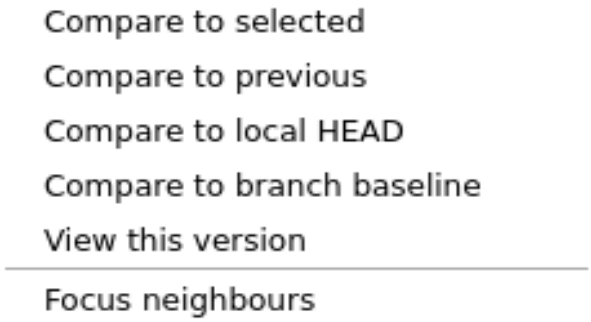

Figure 37: Compare context menu when a version is selected.

The option Compare to selected is displayed. In this case Compare to previous would have the same effect, but in a larger tree it is possible to compare more distant versions.

The option Compare to local HEAD is just a shortcut without selecting the local HEAD version before.

With Focus neighbours the visibility of all version nodes linked by normal edge is ensured.

(If right click is not performed on a leaf node, Hide subtree or Show subtree is displayed. So far, this feature is just experimental. Hiding a subtree is not persistent and a markup that there is a hidden subtree is not yet implemented.)

To change the selected version, LMB to a different version node.

To remove the selection, RMB to the main view background and select Reset Selection.

A selection is kept even if not visible after changing the displayed version tree via Branch List. This helps comparing different branch versions.

Step 7 Context menu of edges

Now move the mouse pointer over an edge. The RMB click context menu should look like this, then:

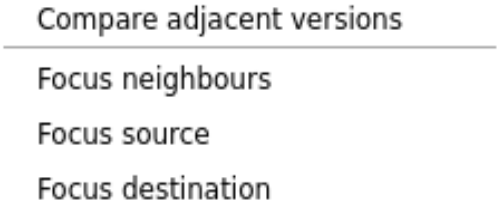

Figure 38: Context menu of a graph edge

With Compare adjacent versions the two versions connected by the edge are compared.

The three focus options are helpful especially if the versions are connected by a very long merge edge.

Step 8 Lookup version information

Open the Version Information dock window in the Windows menu.

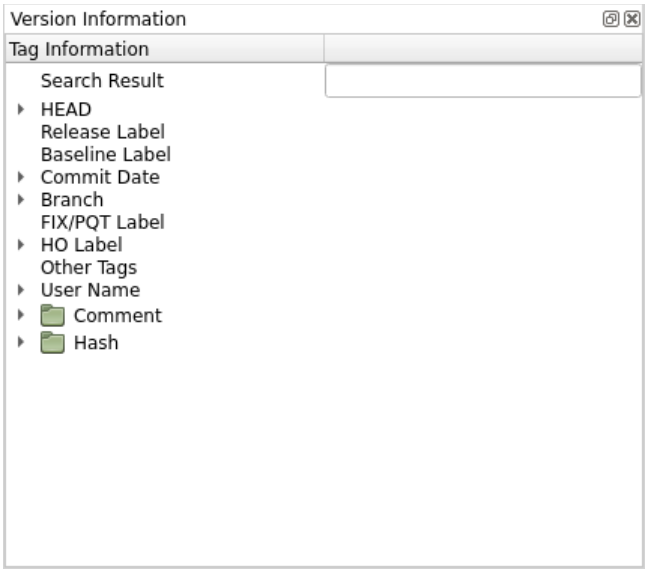

Figure 39: Version Information dock widget

In earlier releases this window contained its information in tabs and lists. This has been changed to a tree. The top level nodes have been the tabs before.

- Search Result
  In the second column a line edit is visible where a search pattern can be entered. The versions matching the search pattern will be listed under this node. When changing the search pattern the Search Result is updated on the fly.
- HEAD
  All versions matching the corresponding regular expression pattern defined in the Preferences - Tag Settings page.
- Release Label
  All versions matching ...
- Baseline Label
  All versions matching ...
- Commit Date
  The commit date is split into YYYY, MM, DD, HH:MM:SS. This item has 4 tree levels then. It is easy then to markup all versions created in a year, month or on a day.
- Branch
  All versions matching ...
- FIX/PQT Label
  All versions matching ...
- HO Label
  All versions matching ...

- User Name
- Comment
- Hash

Change to the HO Label row.

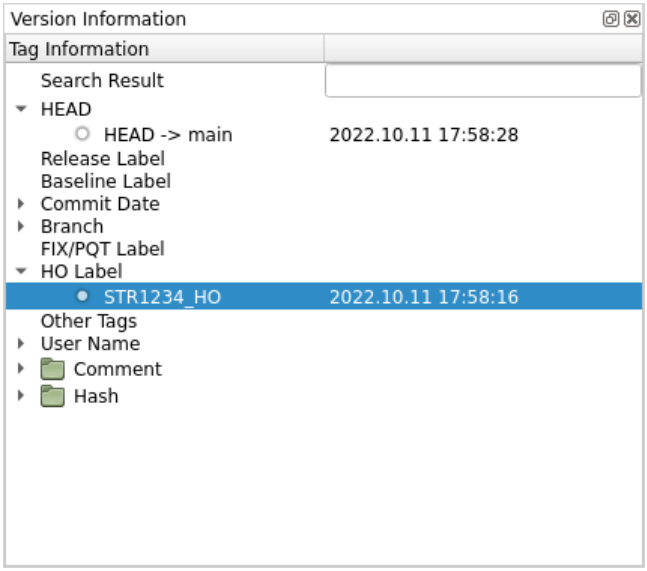

Figure 40: Version Information dock widget page HO Label

Select STR1234_HO with a LMB.
In the graph view the version with the tag STR1234_HO is focused and gets a markup. If this version is contained in a folder, it is ensured that the folder is open so that this version is visible.

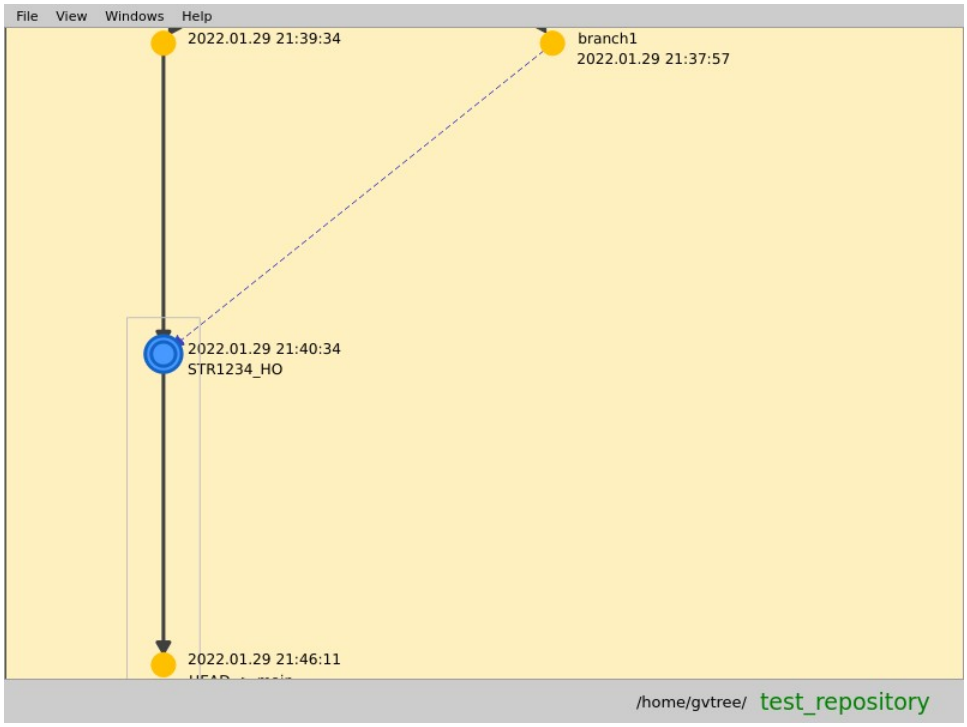

Figure 41: Search by Version Information widget.

Now press STRG + f and the cursor focus is set to the Version Information dock window to the line edit to enter a search pattern. Just type the year in the field. In this example 2022. Expressions with less than 3 characters are ignored. Regular expressions are allowed.

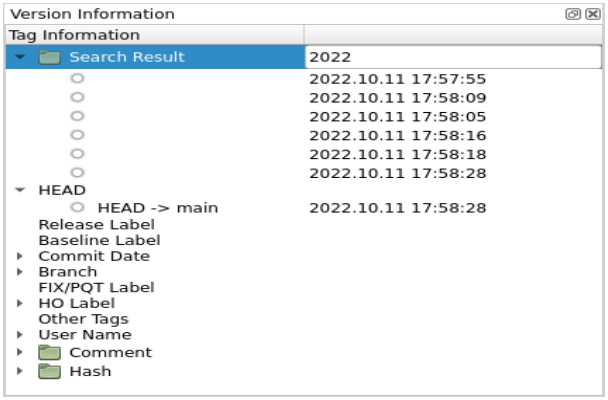

Figure 42: Simple search with results printed under the Search Result node.

The markup and focus in the graph view will look like this, then:

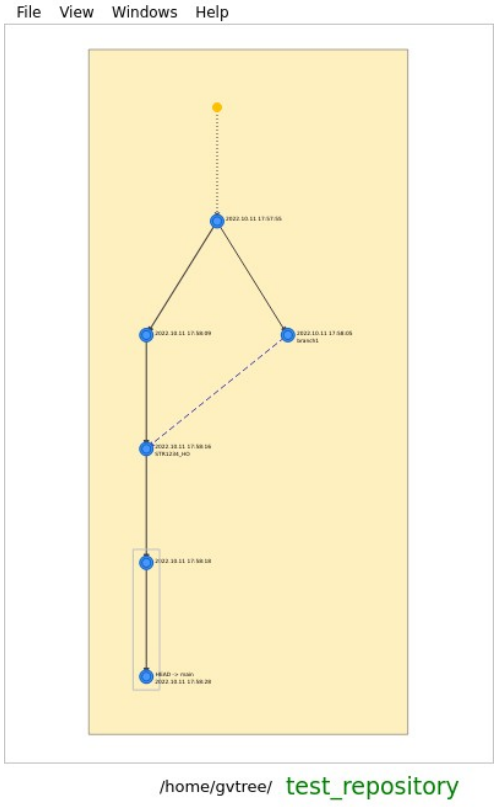

Figure 43: All matches for the search pattern 2022

All matching nodes are visible. The tag information causing the match is displayed automatically (commit date).

The Version Information window has got a tree node Search Result. Under this node all current matching versions are displayed with their commit date timestamp. If the matches are distributed all over a big version tree, it is easier to access individual versions with the Version Information window.

Step 9 The Branch List widget

First, in the menu Preferences - Basic Settings the **git log --remotes** should be switched off. Otherwise selection of a branch in the the Branch List widget will just focus to the latest branch version. The graph will not change. In the example there is only one branch **branch1** beside the **main** branch. Just add one more branch **branch2** in the following way:

git checkout STR1234_HO

git branch branch2

git checkout branch2

To create a new version, just add a ChangeLog file to the repository.

echo "ChangeLog" > ChangeLog

git add ChangeLog

git commit

git checkout main

Now run **gvtree** again or press reload and open the dock widget Branch List. In the main view, select the **HEAD→main** version.

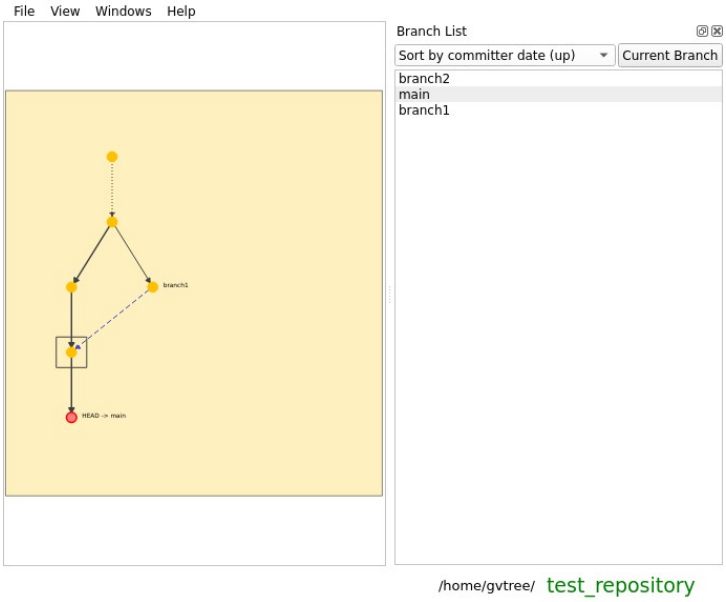

Figure 44: Branch List widget

There are now three entries visible in the list. The current checked out branch **main** is selected. With Sort by committer date (up) the latest branch is on top of the list. A sort by name is possible, too.

In the Branch List widget, select **branch2**.

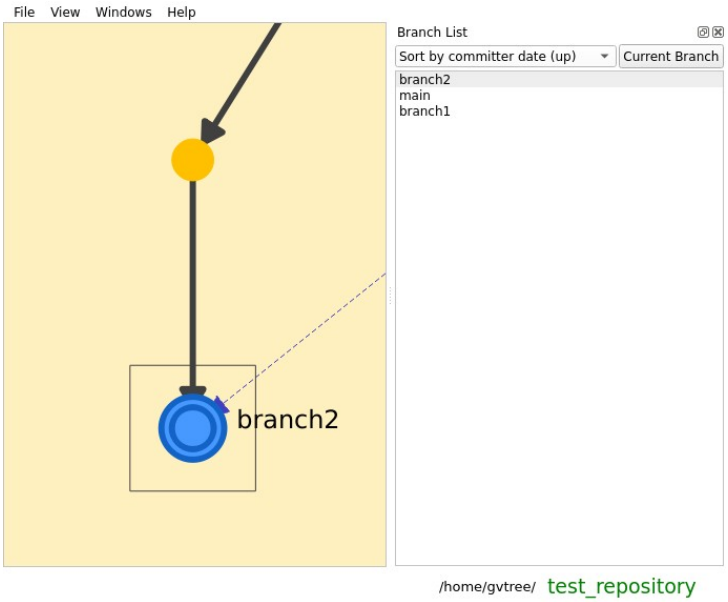

Figure 45: Selected branch2

Zoom out and perform a right click on the **branch2** version to open the context menu.

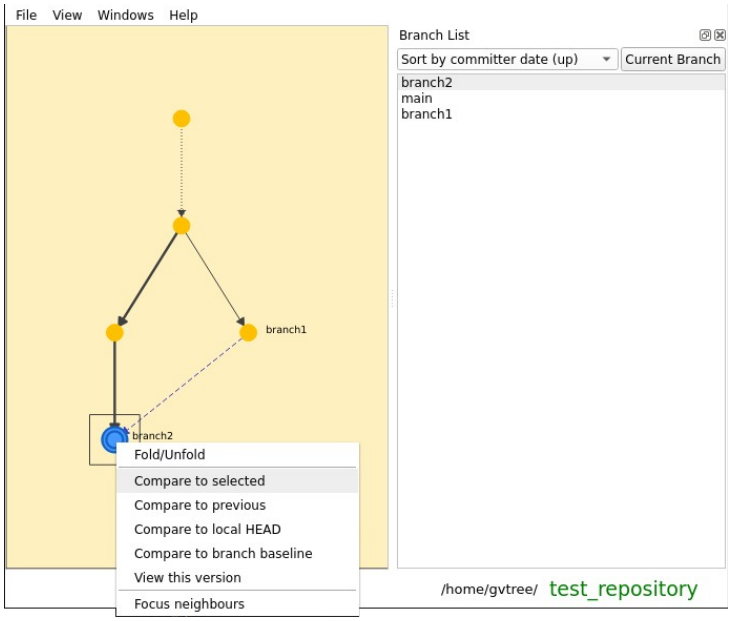

Figure 46: Compare versions between different branches

The **HEAD→main** version is still selected and the **branch2** version can now be compared to it.

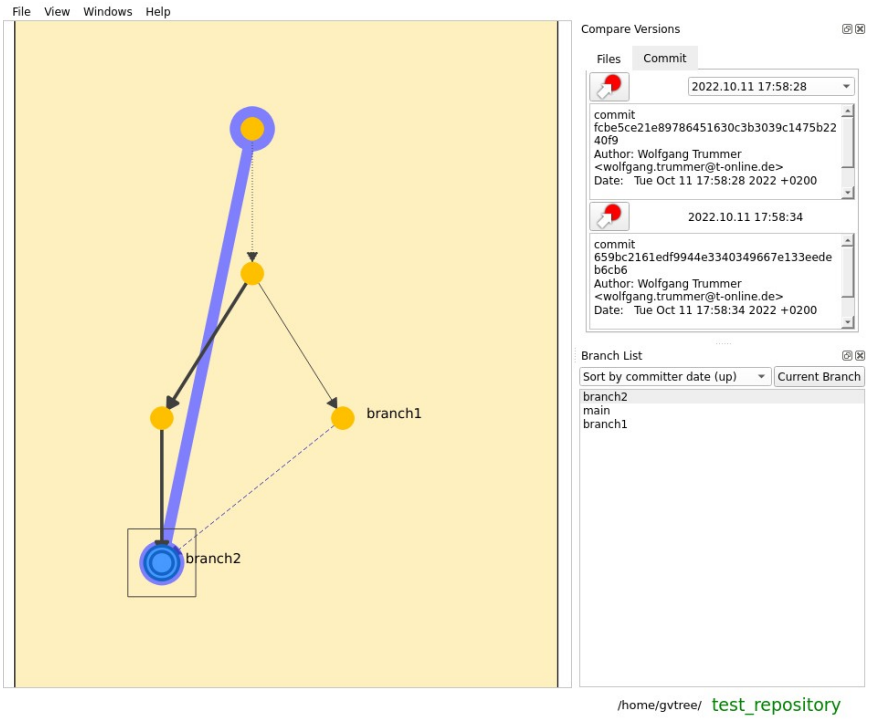

Figure 47: Compare HEAD→main and branch2

Pressing Current Branch will restore the main view and show the current checked out branch.

Appendix A License

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of

protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.
The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.
All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so

exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also

receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered

work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by

the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS