
Timeline Documentation

The Timeline Authors

Aug 03, 2018

Contents

| | | |
|-----------|--|-----------|
| 1 | About Timeline | 1 |
| 2 | Screenshots | 3 |
| 3 | Installing | 9 |
| 3.1 | Windows | 9 |
| 3.2 | Fedora | 9 |
| 3.3 | Installing from Snapcraft | 10 |
| 3.4 | Installing from source | 10 |
| 4 | Support | 13 |
| 4.1 | Documentation | 13 |
| 4.2 | Mailing list | 13 |
| 4.3 | FAQ | 13 |
| 4.4 | Standard email responses | 14 |
| 5 | Project status | 17 |
| 6 | Project history | 19 |
| 6.1 | Initial mission statement | 19 |
| 6.2 | Conference 2015 | 19 |
| 6.3 | Conference 2016 | 21 |
| 7 | Articles | 23 |
| 8 | Contributing | 25 |
| 8.1 | What to contribute? | 25 |
| 8.2 | Process | 25 |
| 9 | Guidelines | 27 |
| 9.1 | How to write a problem description | 27 |
| 9.2 | Code style | 27 |
| 9.3 | How to write tests | 27 |
| 10 | Howtos for developers | 29 |
| 10.1 | Setting up a development environment | 29 |
| 10.2 | Build an exporter for Timeline | 30 |
| 10.3 | Build a dialog widget | 32 |

| | | |
|-----------|--|-----------|
| 10.4 | Run automated tests (unit tests) | 36 |
| 10.5 | How to use Timeline component in your wxPython application | 37 |
| 10.6 | Making a Timeline release | 39 |
| 10.7 | Make a string translatable | 41 |
| 10.8 | Run Timeline in a different language | 42 |
| 10.9 | Translate Timeline into another language | 42 |
| 11 | Architecture overview | 43 |
| 11.1 | GUI classes | 44 |
| 11.2 | Drawing | 45 |
| 11.3 | Startup | 46 |
| 11.4 | System documentation - API | 46 |
| 12 | Project infrastructure | 47 |
| 12.1 | SourceForge | 47 |
| 12.2 | Source control | 47 |
| 12.3 | Mailing list | 47 |
| 12.4 | Build server | 48 |
| 12.5 | Translation service | 48 |
| 13 | Changelog | 49 |
| 13.1 | Version 1.19.0 | 49 |
| 13.2 | Version 1.18.0 | 49 |
| 13.3 | Version 1.17.0 | 50 |
| 13.4 | Version 1.16.0 | 50 |
| 13.5 | Version 1.15.0 | 50 |
| 13.6 | Version 1.14.0 | 51 |
| 13.7 | Version 1.13.0 | 52 |
| 13.8 | Version 1.12.0 | 52 |
| 13.9 | Version 1.11.0 | 53 |
| 13.10 | Version 1.10.0 | 53 |
| 13.11 | Version 1.9.0 | 54 |
| 13.12 | Version 1.8.1 | 55 |
| 13.13 | Version 1.8.0 | 55 |
| 13.14 | Version 1.7.1 | 56 |
| 13.15 | Version 1.7.0 | 56 |
| 13.16 | Version 1.6.0 | 58 |
| 13.17 | Version 1.5.1 | 58 |
| 13.18 | Version 1.5.0 | 58 |
| 13.19 | Version 1.4.1 | 59 |
| 13.20 | Version 1.4.0 | 59 |
| 13.21 | Version 1.3.0 | 59 |
| 13.22 | Version 1.2.4 | 60 |
| 13.23 | Version 1.2.3 | 60 |
| 13.24 | Version 1.2.2 | 60 |
| 13.25 | Version 1.2.1 | 60 |
| 13.26 | Version 1.2.0 | 60 |
| 13.27 | Version 1.1.0 | 61 |
| 13.28 | Version 1.0.1 | 61 |
| 13.29 | Version 1.0.0 | 62 |
| 13.30 | Version 0.21.1 | 62 |
| 13.31 | Version 0.21.0 | 63 |
| 13.32 | Version 0.20.0 | 63 |
| 13.33 | Version 0.19.0 | 63 |

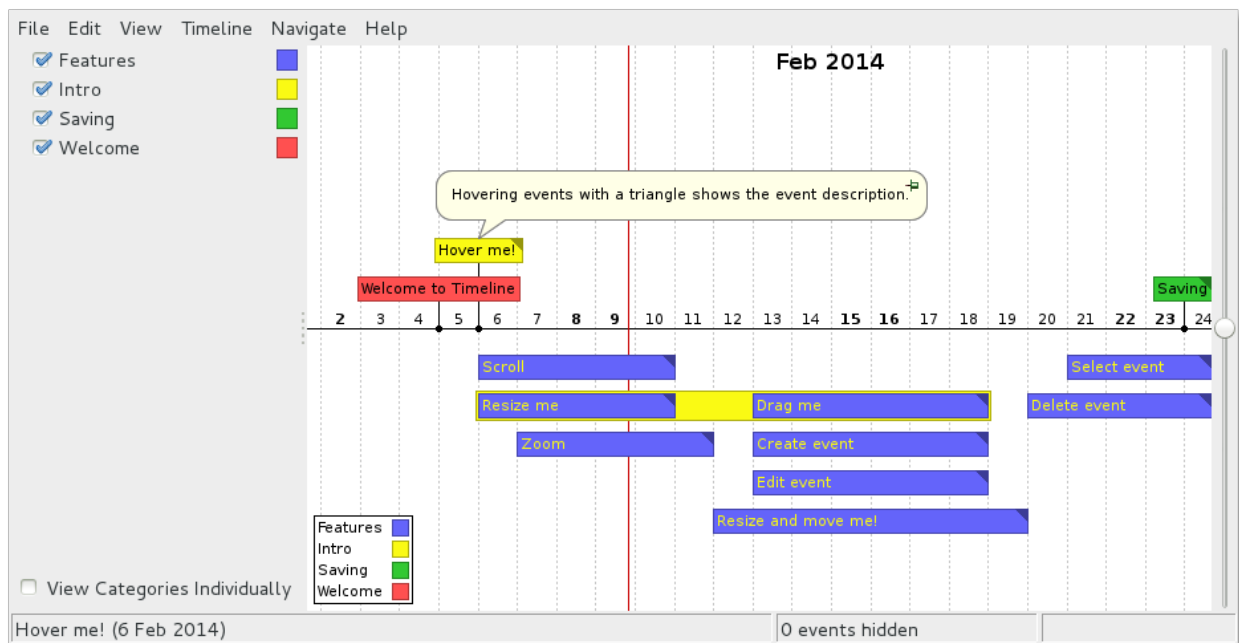
| | |
|--|----|
| 13.34 Version 0.18.0 | 64 |
| 13.35 Version 0.17.0 | 64 |
| 13.36 Version 0.16.0 | 64 |
| 13.37 Version 0.15.0 | 64 |
| 13.38 Version 0.14.0 | 65 |
| 13.39 Version 0.13.0 | 65 |
| 13.40 Version 0.12.1 | 66 |
| 13.41 Version 0.12.0 | 66 |
| 13.42 Version 0.11.1 | 66 |
| 13.43 Version 0.11.0 | 67 |
| 13.44 Version 0.10.2 | 67 |
| 13.45 Version 0.10.1 | 67 |
| 13.46 Version 0.10.0 | 67 |
| 13.47 Version 0.9.0 | 68 |
| 13.48 Version 0.8.0 | 68 |
| 13.49 Version 0.7.0 | 69 |
| 13.50 Version 0.6.0 | 69 |
| 13.51 Version 0.5.0 | 70 |
| 13.52 Version 0.4.0 | 70 |
| 13.53 Version 0.3.0 | 71 |
| 13.54 Version 0.2.0 | 71 |
| 13.55 Version 0.1.0 | 71 |
| 13.56 A note about version numbers | 72 |

CHAPTER 1

About Timeline

Timeline is a cross-platform application for displaying and navigating events on a timeline.

Timeline is [free software](#), distributed under the [GNU General Public License version 3](#).



See more [screenshots here](#).

Features:

- Scroll and zoom with mouse wheel
- Different representation depending on zoom level
- Go to a specific date

- Search events
- Organize events in hierarchical categories
- Move and resize events with the mouse
- Duplicate events
- Export to image
- Available in multiple languages ([supported languages](#))

CHAPTER 2

Screenshots

Here you can find various screenshots of Timeline and timelines created using it.

Do you have a screenshot that you would like to be included here? [Send us an email](#).

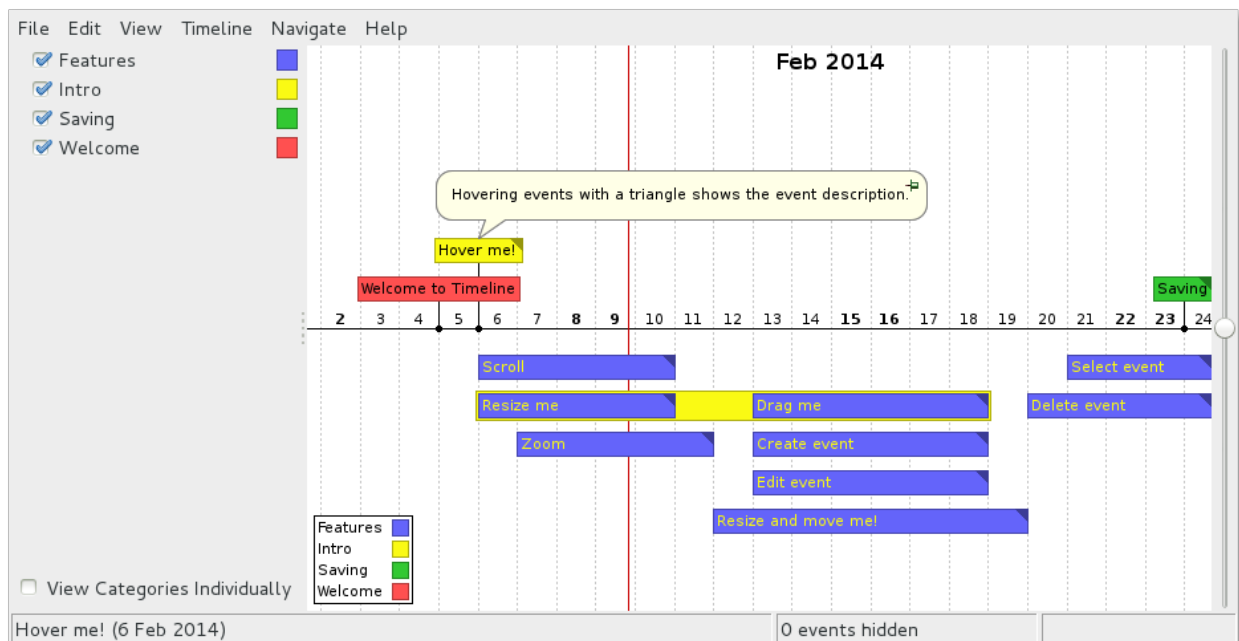



Fig. 1: The main window when the tutorial timeline is open.

Event Properties

When: 

☐ Period ☐ Show time ☐ Fuzzy ☐ Locked ☐ Ends today

Text:

Category: ▼

Container: ▼

Description Icon Alert Hyperlink

Hovering events with a triangle shows the event description.




Fig. 2: The event editor dialog.

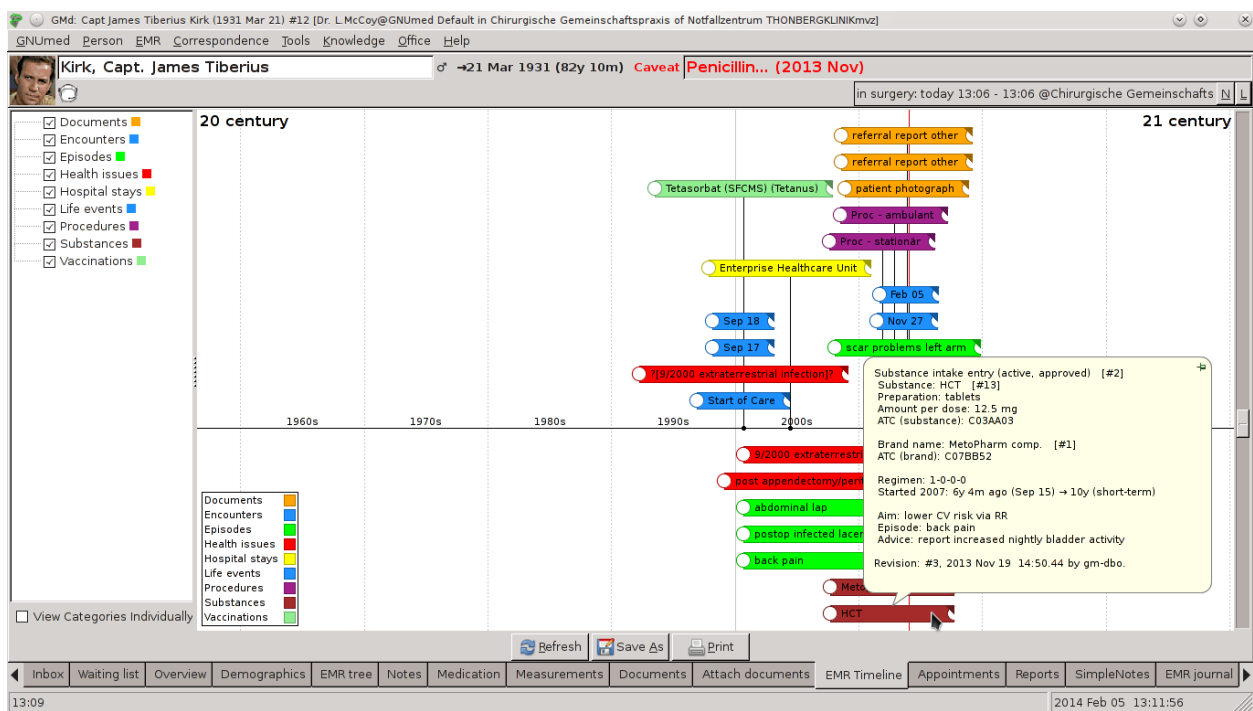


Fig. 3: We use the Timeline component to display Electronic Medical Records in a chronological way.



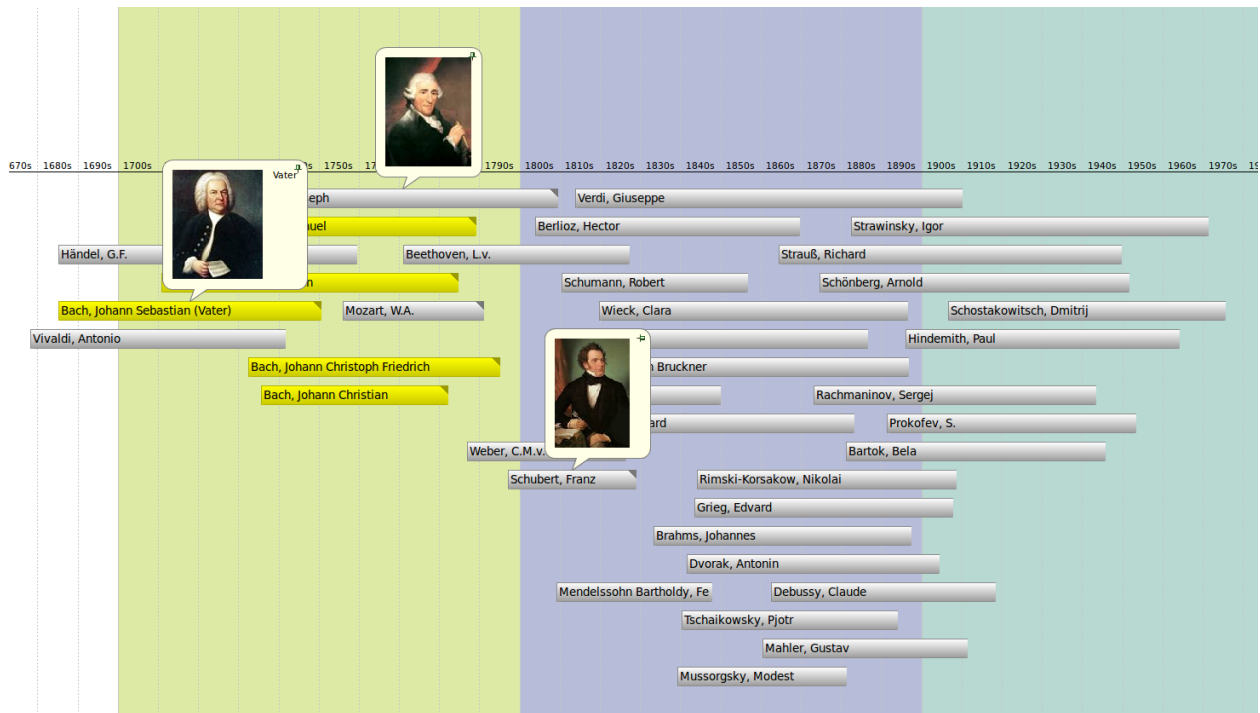


Fig. 5: An interactive list of famous composers: click on the name and you can see the portrait of the artist

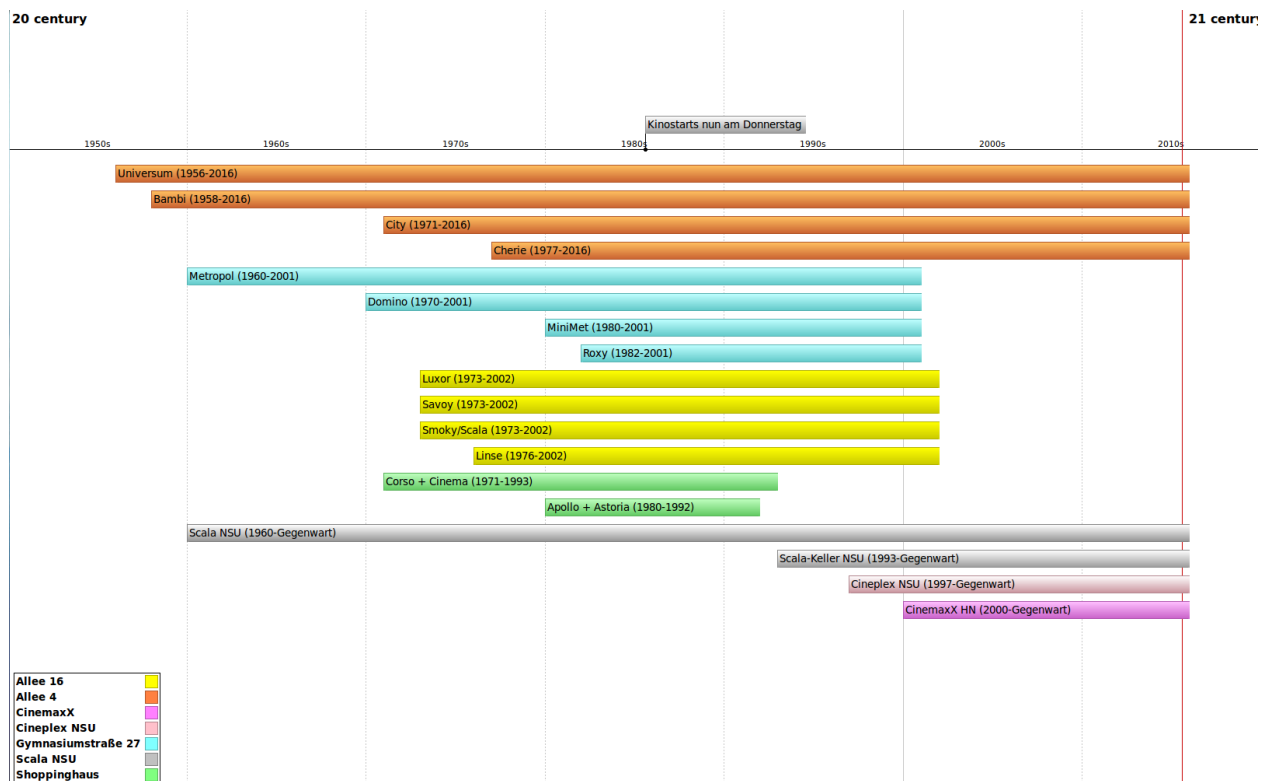


Fig. 6: An overview for all cinemas in our city for this blog.

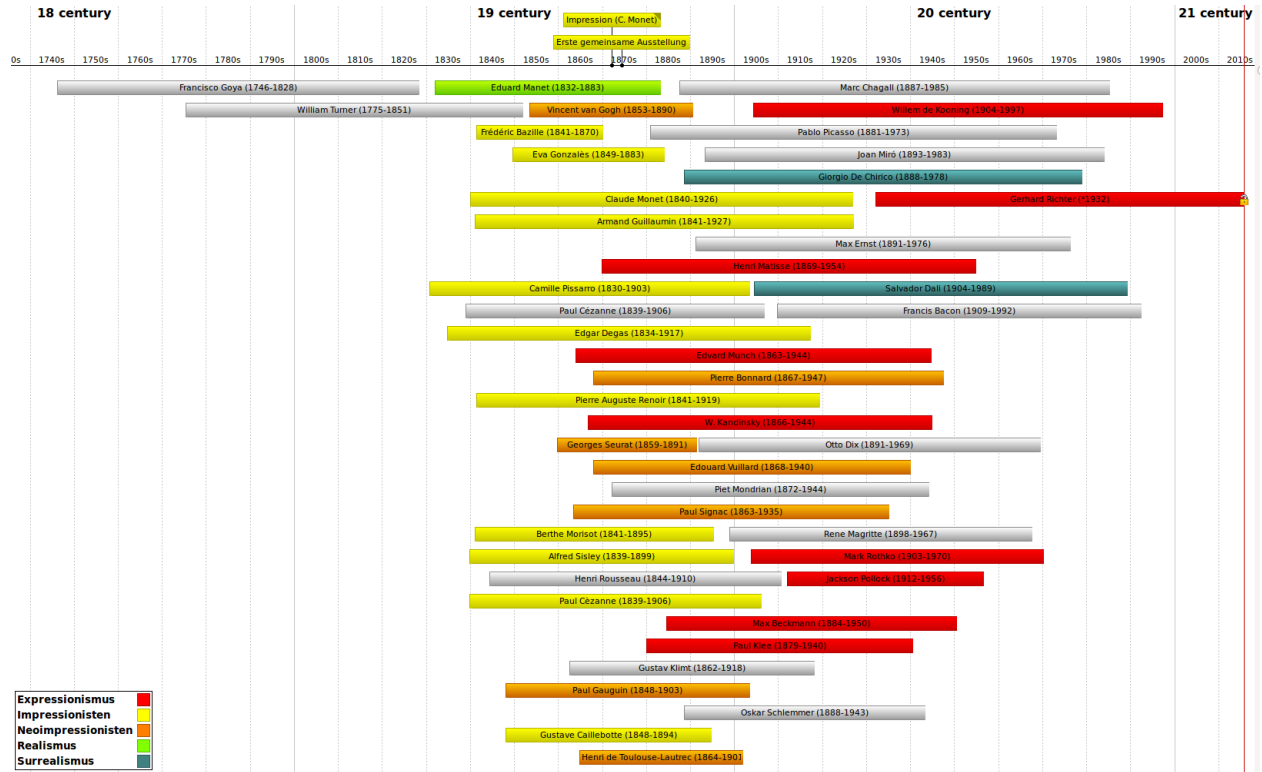


Fig. 7: Famous painters.

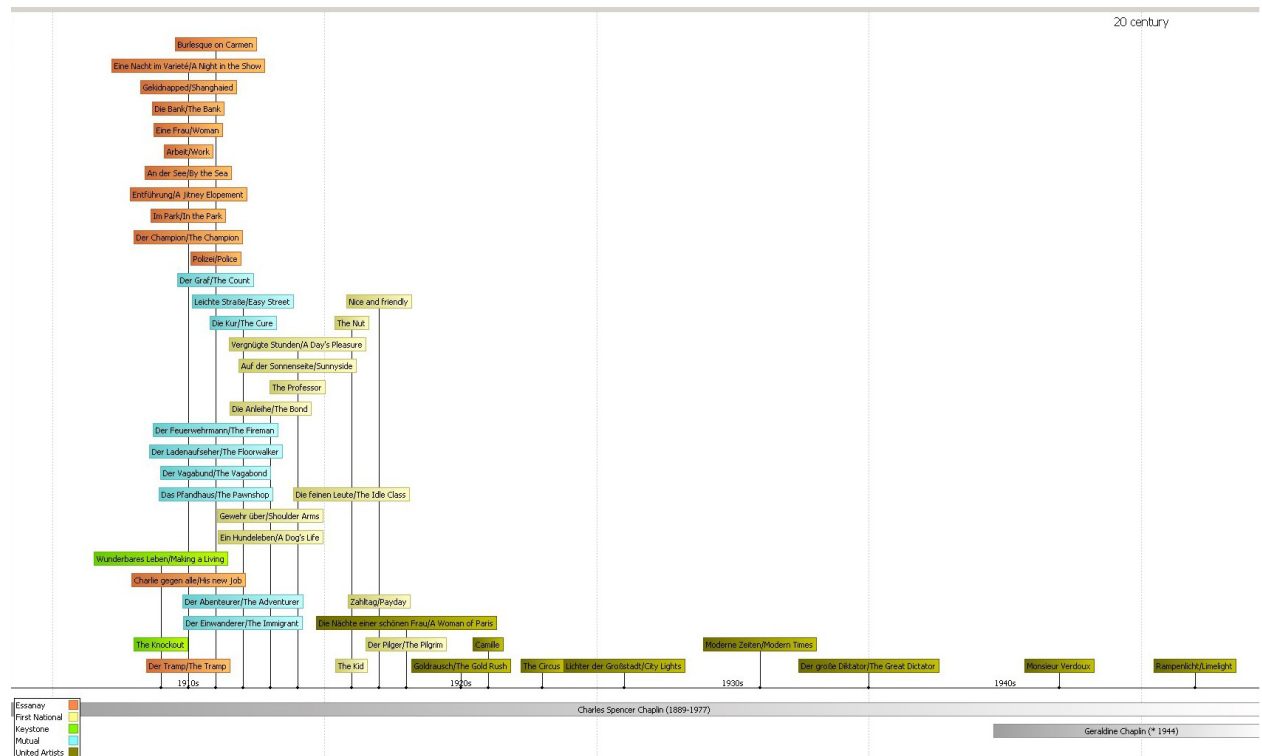


Fig. 8: The films of Charles Chaplin.

CHAPTER 3

Installing

Timeline runs on multiple platforms. If you can run [Python](#) and [wxPython](#) you should be able to run Timeline.

The recommended way to install Timeline is with a binary package or installer. If that is not available for your platform, you have to install from source.

If you have problems installing Timeline, please see the [Support](#) page for ways to get help.

3.1 Windows



We provide a binary installer for Windows. It installs an executable version of Timeline that is self contained and doesn't need any other dependencies.

Download one of the following installers:

- Latest release: [SetupTimeline1180Py2ExeWin32.exe](#)
- Beta version: [latest Windows build](#)

The beta version is for users that want to try the latest features and give feedback on them before a release.

3.2 Fedora



Timeline exists in the Fedora repositories. Install it with the following command:

```
sudo dnf install timeline
```

The version included in Fedora is often not the latest. If you want the latest version, you have to install from source.

3.3 Installing from Snapcraft

<https://snapcraft.io/timeline>

3.4 Installing from source



Download one of the following source packages:

- Latest release: [timeline-1.18.0.zip](#)
- Beta version: [latest source build](#)

The beta version is for users that want to try the latest features and give feedback on them before a release.

When you install from source, you have to install required dependencies yourself. See the next section for instructions how to do that. Once that is done, and you have extracted the zip file, you should be able to run Timeline with this command:

```
python <path-to-timeline-directory>/source/timeline.py
```

Hint: If you get an error similar to the one below, there is probably a dependency missing:

```
$ python source/timeline.py
Traceback (most recent call last):
  File "source/timeline.py", line 64, in <module>
    setup_humblewx()
  File "source/timelinelib/wxgui/setup.py", line 35, in setup_humblewx
    import humblewx
ImportError: No module named humblewx
```

3.4.1 Installing dependencies

This section describes how to install the software that Timeline depends on.

Python

<http://www.python.org>

There is most definitely a binary package or installer for your platform.

Timeline requires version 2.6 or greater.

wxPython (Python package)

<http://www.wxpython.org>

There is probably a binary package or installer for your platform.

Timeline requires version 3.0 or greater.

humblewx (Python package)

<https://github.com/thetimelineproj/humblewx>

A Python package that we extracted from the Timeline source code because it could be generally useful.

The latest version can be installed with the following command:

```
pip install --user git+https://github.com/thetimelineproj/humblewx.git
```

icalendar (Python package)

<http://pypi.python.org/pypi/icalendar>

At the moment, this is included in the Timeline repository and does not have to be installed.

markdown (Python package)

<http://pypi.python.org/pypi/Markdown>

At the moment, this is included in the Timeline repository and does not have to be installed.

pysvg (Python package)

<http://code.google.com/p/pysvg/downloads/list>

At the moment, this is included in the Timeline repository and does not have to be installed.

Do you need help with Timeline? Do you want to help others with Timeline? This is where to look.

4.1 Documentation

The documentation you are reading right now might have the answer to your question. Browse it to see if you can find it. You can also search it by typing in a search term in the “Quick search” field in sidebar to the left.

4.2 Mailing list

We use a [mailing list](#) for all kinds of discussions about Timeline. If you have a question about Timeline, you can send it to the mailing list. Everyone that is registered will receive your message and can write a response.

Before you send your question, please browse or search the mailing list archive to see if you can find your answer.

If you want to participate in the discussions on the mailing list and help answer users’ questions about Timeline, you need to register.

4.3 FAQ

4.3.1 Can’t zoom wider than 1200 years

This workaround should not be needed in versions $\geq 1.7.0$.

Timeline should not be able to zoom wider than 1200 years. But unfortunately, this happens sometimes, and a period larger than 1200 years gets written to the `.timeline` file.

When this happens, you can no longer open the file.

This is a bug in Timeline that we should fix. But there is a workaround that you can use to fix the problem yourself.

The `.timeline` file is actually just an xml file. In there, somewhere around the bottom of the file, you should find this section:

```
<displayed_period>
  <start>2013-09-28 00:10:06</start>
  <end>2013-10-09 20:34:55</end>
</displayed_period>
```

If you open the `.timeline` file in a text editor and change the start and end dates to be a period less than 1200 years, you should be able to open your file again.

Make sure that you save the file with UTF-8 encoding.

4.4 Standard email responses

Here are some standard email responses to common emails on the mailing list. Use them as a template when answering questions on the mailing list.

Crash report that has not been fixed:

```
Thanks for taking the time to report this error.

We have added the problem to our backlog:
https://sourceforge.net/p/thetimelineproj/backlog/

/The Timeline Team
```

Crash report that has been fixed in the upcoming release:

```
Thanks for taking the time to report this error.

The problem has been fixed and will be available in the next release.

Don't want to wait for it? Try the beta version!

* Source: https://jenkins.rickardlindberg.me/job/timeline-linux-source/
  ↳lastSuccessfulBuild/artifact
* Windows installer: https://jenkins.rickardlindberg.me/job/timeline-windows-exe/
  ↳lastSuccessfulBuild/artifact

/The Timeline Team
```

Crash report that has been fixed in released version:

```
Thanks for taking the time to report this error.

The problem has been fixed in the current version. It can be downloaded
here: https://sourceforge.net/projects/thetimelineproj

/The Timeline Team
```

Person interested in developing Timeline:

```
We're glad you're interested in contributing to Timeline. We want to help
as much as we can.
```

(continues on next page)

(continued from previous page)

Here are some points you can start **with**:

- * We have a backlog of problems that we want to solve (<https://sourceforge.net/p/thetimelineproj/backlog>)
- * Crash reports **in** the backlog are usually easy to fix. Although they might require some investigation.
- * Timeline **is** licensed under GPLv3. We propose a model where you (**and** everyone **else**) keep the copyright of their patches. As the patches will be GPLv3 **as** well, we can include them **in** the project.
- * Registering on this mailing list **is** a good idea.
- * The website has some more information:
<http://thetimelineproj.sourceforge.net/>
- * Some sections about developing are outdated.
- * We're here on the mailing list to help you with any questions you have :-)

/The Timeline Team

Person writes to the mailing list but we would like the discussion to happen in the forum instead:

Thanks **for** writing to the mailing list.

We are experimenting **with** using a forum **for** discussions instead of the mailing list.

Please **continue** the discussion here: <http://...>

/The Timeline Team

CHAPTER 5

Project status

Timeline is being actively developed.

Roughly every third month there is a new release of Timeline. The [Changelog](#) shows when the latest version was released as well as when we plan the next release.

Since Timeline is an open source project, anyone can participate in the development and make contributions to each release. Have a look at [Contributing](#) for more information.

News about the project can be found on the [SourceForge news page](#).



6.1 Initial mission statement

In the very beginning, Timeline was created because Rickard was frustrated with regular calendars. He found it difficult to get a sense of when events occurred in time. Especially events far apart that would not fit in a calendar view. He thought that presenting events on a timeline where he could quickly zoom in and out and move around to show different views would help.

So, the core task of Timeline is to **display and navigate events on a timeline**. It has done that since version 0.1.0 which was released on 11 April 2009. Excerpt from the README:

The Timeline Project aims to create a cross-platform application for displaying and navigating information on a timeline. The main goal is that it should be easy to quickly display different periods in time with different kinds of information on the timeline.

If you have data that you would like to display on a timeline, Timeline should be able to show it. Timeline has multiple backends to display data from different sources.

6.2 Conference 2015

The first Timeline conference took place at Södertuna castle on the 13-15 February 2015.

The Timeline team met to discuss future directions of the project.

Summary:

- We want to continue developing Timeline
- We want to get more people involved in the development of Timeline
- We worked on making it easier for new people to get into Timeline
 - We started writing guides
 - We improved the structure of the files in the repository



Fig. 1: Södertuna castle.



Fig. 2: The Timeline team.

- We experimented with a plugin architecture that will make it easier to make small modifications to Timeline
- We want to let the Timeline community decide the direction in which Timeline should develop
 - We looked at mailing list traffic from 2014 to learn what problems users face when they are using Timeline
 - We started entering those problems in a backlog
 - We documented the process we should try to follow
 - In this process, all users' problems are entered in the backlog
 - We try to solve problems from the backlog that most users have faced

6.3 Conference 2016

The second Timeline conference took place at Trosa stadshotell on the 12-14 February 2016.

Review of past year:

- We have continued to develop Timeline. We are happy with that and will continue.
- More people are involved in discussions about Timeline. We are happy about that. Although there have been no contributions of code.

We discussed if extracting the canvas component would be a good project to work on. Reasons for doing it:

- More developers might contribute code to the canvas component if they use it themselves in their own project.
- The canvas is used today but is not supported by us.

We discussed having a forum instead of the mailing list and backlog. Problems it might solve:

- The discussion happens in one place: in the forum thread instead of first on the mailing list and then transferred to the backlog.
- It is easier to see the history of discussions compared to the mailing list archive.

Refactoring goals:

- Rename timeline to db.
- Move test/specs to test/unit and test/something-else.

Future plans:

- We would like to find a conference where we can meet people interested in Timeline. Perhaps FOSDEM?
- We are interested in creating an Android app that displays timelines. It will be a totally different project, but it will use concepts from Timeline and will try to preserve the look and feel. One idea is to focus on the Canvas component and try to implement such component in Java as well. It will still be separate projects, but they can use similar concepts.

CHAPTER 7

Articles

Here are links to articles written by others about Timeline:

- [\(German\) Ein Bild sagt mehr als 1000 Worte - Zeitleisten unter Linux](#) (4 October 2015)
- [Timeline release statistics](#) (1 July 2015)
- [Analysis of Timeline emails](#) (21 June 2015)
- [TimeLine: a Python-Based Timeline Creator for Linux](#) (4 June 2015)

Here are links to sites writing about Timeline:

- [\(German\) Timeline – Das OpenSource-Programm](#)

Timeline is an open source project that is developed by a community of people. Anyone interested can participate in the development.

8.1 What to contribute?

There are many ways you can contribute to the development of Timeline. The most useful thing you can do is to just use Timeline and tell us how it works for you. Send feedback and comments to the [Mailing list](#) and participate in discussions.

Here is a blog post that might be useful that talks about ways to contribute to open source projects in general: [14 Ways to Contribute to Open Source without Being a Programming Genius or a Rock Star](#).

8.2 Process

This section describes how changes to Timeline are made.

It describes how we work.

If we find better ways of working we can change this process to reflect that.

8.2.1 General workflow

- We continuously collect problems to be solved
- We work in sprints that are around 3 months long
- In each sprint we work on the most interesting problems
- After each commit a beta release is built automatically
- At the end of each sprint a final release is made

8.2.2 Backlog

The backlog is here: <https://sourceforge.net/p/thetimelineproj/backlog>

The backlog contains problems that have been identified by users of Timeline.

Collecting problems

The old way of collecting problems looks like this:

- The main way users of Timeline interact with the project is via the *Mailing list*.
- Requests that users make are transformed into problem descriptions
- Usually a conversation is needed to find the problem description
- Crash reports are already clear problem descriptions
- Problems are stored in the backlog

The new experimental way of collecting problems looks like this:

- Users discuss Timeline on the forum
- Some discussions will result in problems being identified
- Discussions with identified problems might be tagged so that they can be easily found

Prioritization

- A problem is prioritized higher if more users have experienced it
- Anyone is allowed to comment on problems to indicate its importance to them

8.2.3 Sprints

- A developer picks an identified problem (preferably a problem with high priority) and works on it
- **To be accepted for inclusion in Timeline, each patch must**
 - follow guidelines
 - pass tests
 - have an entry in the changelog if major user visible changes was made
- A patch is delivered by pushing to the repo (if the developer has push-access)
- A patch is delivered by sending it to the mailing list (if the developer lacks push-access)

8.2.4 Release

- At the end of a sprint a developer creates release packages and uploads to SourceForge

9.1 How to write a problem description

- Write brief description of the problem in ticket title
- Write an extended description in the ticket text (not all the below items need to be entered)
 - Elaborate the description
 - Describe similar problems
 - Attach a screenshot illustrating the problem
 - Motivate why this problem is important to solve
- Add suggested solutions in comments to the ticket using the following syntax:

```
**Suggested solution**: ...
```

9.2 Code style

We try to follow PEP 8 with some exceptions: <https://www.python.org/dev/peps/pep-0008/>

Exceptions:

- We allow longer lines than 79 characters. But preferably not longer than 120
- We prefer to have double quoted strings

9.3 How to write tests

The idea of having a guideline for how to write tests is that if they all look the same, they are easier to understand.

Here are the guidelines:

- All test classes should inherit a custom test case class

10.1 Setting up a development environment

Timeline uses the [Mercurial](#) version control system. You can clone the repository with the following command:

```
hg clone http://hg.code.sf.net/p/thetimelineproj/main
```

Once you have the repository cloned, you need to install dependencies (see *Installing dependencies*) and development tools (see the next section).

Then make sure you can run all tests:

```
python tools/execute-specs.py
```

If that works, you have the basic environment set up.

10.1.1 Installing development tools

This section describes how to install developments tools. Some tools are only needed in certain situations.

mock (Python package)

<http://pypi.python.org/pypi/mock>

This is used in some tests.

At the moment, this is included in the Timeline repository and does not have to be installed.

gettext

<http://www.gnu.org/software/gettext/>

This is used when working with translations. It is also used when running tests.

Windows users: get the zipfiles `gettext-tools-0.17.zip` and `gettext-runtime-0.17-1.zip` from here <http://ftp.acc.umu.se/pub/gnome/binaries/win32/dependencies/>.

10.2 Build an exporter for Timeline

In Timeline an Exporter is used to export data from a Timeline to a file. The Exporter transforms the data into a format that is understood by another application such as a web browser or a spreadsheet program.

To be able to install an Exporter into Timeline it must be built as a Plugin. That means it must implement all the methods defined in the class `PluginBase`. To install the Exporter it is placed in the directory `timelinelib.plugin.plugins.exporters`

10.2.1 Step 1 - Inherit from PluginBase

Say that we want to build an Exporter that creates a file to be used by the fictitious application `EventViewer`. The first thing to do is to create a module with a class definition like this:

```
from timelinelib.plugin.pluginbase import PluginBase

class ExportToEventViewerFile(PluginBase):
    pass
```

10.2.2 Step 2 - Implement base class methods

Next step is to implement the methods defined in the `PluginBase` class:

```
from timelinelib.plugin.pluginbase import PluginBase
from timelinelib.plugin.factory import EXPORTER

class ExportToEventViewerFile(PluginBase):

    def displayname(self):
        return _("Export to EventViewer...")

    def service(self):
        return EXPORTER

    def run(self, timeline, parent=None):
        pass
```

Note that the `displayname()` returns a string surrounded by `_()`. The reason for this is to make it possible to internationalize the text.

The `service()` returns the constant `EXPORTER` which makes Timeline understand to create a menu alternative for the exporter under `File -> Export`.

10.2.3 Step 3 - Write the implementation

The implementation is written in the `run()` method. This method is passed a timeline object, from which information about the timeline data can be retrieved. The parent argument is the gui object from which the run function is called. For exporters, this is the mainframe window. The following timeline methods are useful for retrieving timeline data:

```

timeline.get_all_events() Return a list with all events in a timeline
timeline.get_containers() Return a list with all container events in a timeline
timeline.get_categories() Return a list with all categories in a timeline

```

Useful Event functions:

```

event.get_timeperiod() Returns the start and end of the event
event.get_text() Returns the event title
event.get_category() Returns the category associated with the event
event.get_fuzzy() Returns True if the event has a fuzzy period
event.get_locked() Returns True if the event period is locked
event.get_ends_today() Returns True if the event ends today
event.get_description() Returns the event description
event.get_icon() Returns the link to the icon
event.get_hyperlink() Returns string with semicolon separated hyperlinks.
event.get_alert() Returns alert information
event.get_progress() Return the percentage done
event.is_container() Returns True if it is a container
event.is_subevent() Returns True if the event is inside a container
container.get_subevents() Returns the events associated with the container

```

10.2.4 Step 4 - Install the Exporter

Put the module that contains the Exporter in the `timelinelib.plugin.plugins.exporters` directory. That's it! Sample code:

```

from timelinelib.plugin.pluginbase import PluginBase
from timelinelib.plugin.factory import EXPORTER

FILE_DESCRIPTION = _("Event viewer files")
FILE_EXTENSIONS = ["txt"]
PLUGIN_DISPLAYNAME = _("Export to EventViewer...")

class SampleExporter(PluginBase):

    def displayname(self):
        return PLUGIN_DISPLAYNAME

    def service(self):
        return EXPORTER

    def run(self, timeline, parent=None):
        path = self._get_save_path(self.parent, FILE_DESCRIPTION, FILE_EXTENSIONS)
        if path is not None:
            result = self._transform_data(timeline)
            self._save_result_to_file(result, path)

    def _transform_data(self, timeline):
        collector = []
        for event in timeline.get_all_events():
            collector.append(self._transform_event(event))
        return "".join(collector)

    def _transform_event(self, event):
        return "%s %s\n" % (event.get_text(), event.get_time_period().get_label())

```

10.2.5 The TimelineExporter

This class uses an input dialog to collect the export format as well as what data to export. At the moment, only export to CSV is implemented, but the idea is that this class can be extended to export other file formats as well.

The code is found at:

```
timelinelib.plugin.plugins.exporters.timelineexporter.py
```

10.3 Build a dialog widget

This howto describes how we like to build dialog widgets (`wx.Dialog`).

To get started, we have a tool that can generate boilerplate code. Let's try it:

```
python tools/dialog_template.py
```

If we enter the name `TestDialog`, the following files will be created for us:

```
source/timelinelib/wxgui/dialogs/testdialog/__init__.py
source/timelinelib/wxgui/dialogs/testdialog/testdialog.py
source/timelinelib/wxgui/dialogs/testdialog/testdialogcontroller.py
test/specs/wxgui/dialogs/testdialog/__init__.py
test/specs/wxgui/dialogs/testdialog/testdialog.py
```

Essentially, a dialog consists of 3 files: the dialog itself, the controller, and the test file.

The dialog and the controller collaborate in a pattern inspired by the [Humbe Dialog Box](#). The dialog corresponds to the view and the controller corresponds to the smart object.

What the boiler plate code has given us is a way to test our dialog. Let's try the following command:

```
python tools/execute-specs.py --halt-gui --only testdialog
```

A dialog shows up with a hello world button.

The `--halt-gui` flag ensures that the dialog stays open until we manually close it. That is not desirable when running tests automatically because it needs manual inspection, but for quickly inspecting our dialog, it's perfect.

Let's try without the `--halt-gui` flag just to ensure that it works:

```
python tools/execute-specs.py --only testdialog
```

Now let's look at how the GUI elements are created. Here is `source/timelinelib/wxgui/dialogs/testdialog/testdialog.py` without the copyright notice:

```
from timelinelib.wxgui.dialogs.testdialog.testdialogcontroller import
↳ TestDialogController
from timelinelib.wxgui.framework import Dialog

class TestDialog(Dialog):

    """
    <BoxSizerVertical>
        <Button label="$(test_text)" />
    </BoxSizerVertical>
```

(continues on next page)

(continued from previous page)

```

"""

def __init__(self, parent):
    Dialog.__init__(self, TestDialogController, parent, {
        "test_text": "Hello World",
    }, title=_("New dialog title"))
    self.controller.on_init()

```

Notice the docstring that contains XML. That XML describes the GUI elements that are present in the dialog and how they are laid out.

Let's try to change the XML to the following:

```

<BoxSizerVertical>
    <Button label="$ (test_text)" />
    <BoxSizerHorizontal>
        <TextCtrl id="text_one" />
        <TextCtrl id="text_two" />
    </BoxSizerHorizontal>
</BoxSizerVertical>

```

And run the test again:

```
python tools/execute-specs.py --halt-gui --only testdialog
```

We see that the elements are laid out as described in the XML.

Now let's implement some functionality. When we press the button we want to fill the text widgets with some text. The way this is going to work is that the dialog will send an event to the controller, the controller then calls methods on the dialog to update some part.

First, let's connect the event by changing the XML for the button:

```
<Button label="$ (test_text)" event.EVT_BUTTON="on_click" />
```

We also need to add the appropriate method in the controller. The controller (`source/timelinelib/wxgui/dialogs/testdialog/testdialogcontroller.py`) should now look like this:

```

from timelinelib.wxgui.framework import Controller

class TestDialogController(Controller):

    def on_init(self):
        pass

    def on_click(self, event):
        pass

```

If we run the test again, it will not crash, but nothing happens when we click the button. Let's change the `on_click` method to call methods on the dialog (referred to as the view):

```

def on_click(self, event):
    self.view.SetTextOne("hello")
    self.view.SetTextTwo("world")

```

And let's add the two methods in the dialog class:

```
def SetTextOne(self, text):
    self.text_one.SetValue(text)

def SetTextTwo(self, text):
    self.text_two.SetValue(text)
```

`self.text_one` and `self.text_two` were automatically created because we assigned those ids to the controls in the xml.

If we run the tests again and press the button, the texts should be updated.

One purpose for splitting a dialog into a GUI part and a controller part is to make the controller testable in isolation. The idea is to put most of the dialog logic in the controller and test that independently of the GUI.

Let's try to add a test of this kind to `test/specs/wxgui/dialogs/testdialog/testdialog.py`:

```
def test_it_populates_text_when_button_is_clicked(self):
    self.controller.on_click(wx.CommandEvent())
    self.view.SetTextOne.assert_called_with("Hello")
```

Let's run the tests again (but this time there is no need to halt the gui):

```
python tools/execute-specs.py --only testdialog
```

We see this:

```
AssertionError: Expected: (('Hello',), {})
Called with: (('hello',), {})
```

There is a mismatch between the value we set and the value we expect to be set in the first text field. We have to figure out which one is correct, fix it, and move on.

10.3.1 The control object

To make it easier to test a Dialog we follow a pattern where all business logic is placed in a class of its own, which we call the controller class. The controller is instantiated in the `__init__` function of the Dialog class like this:

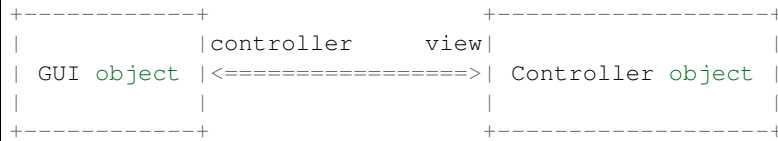
```
def __init__(self, timeline, ...):
    self.controller = MyDialogController(self, timeline)
```

Notice that the constructor takes a reference to the Dialog as first argument. The controller `__init__` function looks like this:

```
def __init__(self, view):
    self.view = view
```

The Dialog class should contain no business logic at all. It should only contain simple logic to handle the gui objects within the Dialog. For example to set a value in a TextBox the Dialog, the dialog shall provide a method `set_text(text)` that can be used by the controller. For the same reason it must provide a `get_text()` function that the controller can use to retrieve values entered by a user.

When it comes to testing the Dialog class is mocked out which means that the test don't have to deal with gui objects. It can be verified that Dialog functions are called when code in the controller is tested. We assume for example that the `Dialog.set_text()` function puts the text in the TextBox control. We can verify this by running the app. If it works once, we assume it will work the next time also. That means we can concentrate on testing the business logic in the controller.



10.3.2 The gui constructor

The gui constructor typically contains the following

- Initiation of gui superclass
- Call to the create_gui() method
- Creation of controller object
- Tell the controller to populate the dialog

Sample:

```

class MyDialog(wx.Dialog):

    def __init__(self, parent, data):
        wx.Dialog.__init__(self, parent, title="mydialog", name="my_dialog",
                           style=wx.DEFAULT_DIALOG_STYLE)

        self._create_gui()
        self.controller = MyController(self)
        self.controller.populate()

class MyController(object)

    def __init__(self, view, data)
        self.view = view
        self.data = data

    def populate(self):
        self.view.set_name(self.data.name)

```

10.3.3 Sizers and controls

We try to break the creation of the gui into small functions. Normally a sizer is passed to a function. The idea is that the function shall create some objects and place them in the sizer:

```

def _create_checkbox_add_more(self, sizer):
    label = _("Add more events after this one")
    self.chb_add_more = wx.CheckBox(self, label=label)
    sizer.Add(self.chb_add_more, flag=wx.ALL, border=BORDER)

```

Controls that need to be referenced later are added to the dialog object (self).

10.3.4 Helper methods

Some problem tends to repeat themselves. So to avoid duplicate code it's desirable to have that piece of code defined once. The place to define such code is in the wxgui.utils package. In this package the following gui helper code can

be found:

- WildcardHelper A class used to define file types in a open/save file dialog
- get_color Takes an rgb tuple as argument and returns a wx:Colour object
- set_wait_cursor Changes the cursor for the given gui window to a wait cursor
- set_default_cursor Changes the cursor for the given gui window to a default cursor
- **get_user_ack Displays a message box with a given message and returns true if the user pressed the YES button.**
The following three displays a message box with a given message.
- display_information_message
- display_warning_message
- display_error_message
- show_modal Show a modal dialog using error handling pattern

10.3.5 Module structure

The dialog class and the controller class are typically saved in separate source files and these files are placed in a module under source.timelinelib.wxgui.dialogs.

The tests for these classes are placed in a module under test.specs.wxgui.dialogs.

10.3.6 Calling updating dialogs

When a dialog is used to change data in a timeline, it's important that the mechanism to prevent two users to change a timeline at the same time, comes into play.

For that reason an updating dialog should always be opened through the helper function self_locking. Like the code where the EventEditorDialog is opened:

Sample:

```
def open_event_editor_for(parent, config, db, handle_db_error, event):
    def create_event_editor():
        if event.is_container():
            title = _("Edit Container")
            return ContainerEditorDialog(parent, title, db, event)
        else:
            return EventEditorDialog(
                parent, config, _("Edit Event"), db, event=event)
    def edit_function():
        gui_utils.show_modal(create_event_editor, handle_db_error)
    safe_locking(parent, edit_function)
```

10.4 Run automated tests (unit tests)

We like automated test. All of Timeline's automated tests can be run with a single command:

```
python tools/execute-specs.py
```

If you are working on a specific feature and only want to run a subset of the tests, you can do it like this:

```
python tools/execute-specs.py --only specs.Event. specs.Category.
```

Some tests have a bit of randomness to them. So running them multiple times might give different results. We have a script to run the tests a number of times in sequence:

```
python tools/execute-specs-repeat.py
```

It works for a subset as well:

```
python tools/execute-specs-repeat.py --only specs.Event. specs.Category.
```

10.5 How to use Timeline component in your wxPython application

Note: This is work in progress and feedback is welcome.

The core component in Timeline, the canvas where events are drawn, is a reusable component that any wxPython application can use. This page documents how to use that component.

10.5.1 Importing

Currently, the canvas component is embedded in the Timeline source code. All code related to the canvas component is located in `timelinelib.canvas`. This is not 100% true because the canvas component depends on other parts of `timelinelib`. The long term goal though is that it shouldn't so that the `timelinelib.canvas` package can be extracted to its own project.

In order to use the canvas component, we must obtain the Timeline source code and make sure that the `source/timelinelib` folder is on our Python path.

For the time being, we also need to setup the gettext translation function. The canvas currently depends on gettext, but it should not in the future. We can use this function to setup gettext:

```
def install_gettext_in_builtin_namespace():
    def _(message):
        return message
    import __builtin__
    if not "_" in __builtin__.__dict__:
        __builtin__.__dict__["_"] = _
```

Hint: If we get an error similar to the one bellow, gettext has not been properly setup:

```
Traceback (most recent call last):
..
NameError: name '_' is not defined
```

10.5.2 Example

Here is a complete example how to use the canvas component:

```
make_sure_timelinelib_can_be_imported()
install_gettext_in_built_in_namespace()

import wx

from timelinelib.canvas import TimelineCanvas

class MainFrame(wx.Frame):

    def __init__(self):
        wx.Frame.__init__(self, None, size=(800, 400))
        self._create_canvas()
        self._display_example_timeline()

    def _create_canvas(self):
        self.canvas = TimelineCanvas(self)
        self.canvas.Bind(wx.EVT_MOUSEWHEEL, self._on_mousewheel)

    def _on_mousewheel(self, event):
        self.canvas.Scroll(event.GetWheelRotation() / 1200.0)

    def _display_example_timeline(self):
        # The only way to populate the canvas at the moment is to use a
        # database object from Timeline and call its display_in_canvas method.
        from timelinelib.db import db_open
        db = db_open(":tutorial:")
        db.display_in_canvas(self.canvas)

if __name__ == "__main__":
    app = wx.App()
    frame = MainFrame()
    frame.Show()
    app.MainLoop()
```

10.5.3 The canvas

The TimelineCanvas is a wx control that takes a single argument: the parent control.

```
class timelinelib.canvas.TimelineCanvas
```

```
    __init__(parent)
```

Navigation

The following functions change the time period that is displayed in the canvas.

```
timelinelib.canvas.TimelineCanvas.Navigate (navigation_fn)
```

This is the most generic navigation function. It is used by all the other navigation functions. It takes a function that is called with one argument, the time period, and should return a new time period. The return value is the new time period that is displayed.

```
timelinelib.canvas.TimelineCanvas.Scroll (factor)
```

Exporting

`timelinelib.canvas.TimelineCanvas.SaveAsPng (path)`

`timelinelib.canvas.TimelineCanvas.SaveAsSvg (path)`

10.6 Making a Timeline release

10.6.1 Preparations on main

1. **Check that information is correct in changelog.rst**
 - (a) Check
 - (b) Commit “Updated changes”
2. **Check that information is correct in about.py and AUTHORS**
 - (a) Check developers
 - (b) Check contributors
 - (c) **Check translators**
 - i. Check in LaunchPad for contributors that has an e-mail address.
 - (d) Commit “Updated about”
3. **For Windows build - check that all plugins are imported**
 - (a) Add imports for new plugins, last in the file `releasewincmdmod2_factory_py.py`

10.6.2 Feature freeze

When all features for a major version (x.y) have been implemented in main we move the development for that version over to stable. In stable we prepare for the (x.y.0) release and then continue to do bugfix releases (x.y.1, x.y.2, ..) there.

Features for the next version (x.y+1) can continue to be developed in main.

1. **Move main repo to stable repo**
 - (a) `cd stable`
 - (b) `hg pull ../main`
 - (c) `hg update`
 - (d) `hg push`
2. **Import translations from Launchpad**
 - (a) Request download from here (login required) <http://translations.launchpad.net/thetimelineproj/trunk/+translations> Format: PO format
 - (b) Run `python tools/import-po-from-launchpad-export.py /path/to/launchpad-export.tar.gz`
 - (c) This script updates the .po files
 - (d) Commit “import translations”
3. **Update Timeline.iss**

- (a) Check that all po files found in the directory wintimelinetranslations also are mentioned in the main-releasewininnoTimeline.iss file.
 - (b) Commit “Added po info to iss-file”
4. **Check that version numbers are correct in `timelinelib/meta/version.py`**
 - (a) Check version number
 - (b) Change to “DEV = False”
 - (c) Commit “0.xx.0 Changed version for release”
5. **Check that information is correct in `changelog.rst`**
 - (a) Change Planned -> Released Update versions.timeline. (release/versions.timeline)
 - (b) Commit “Updated changes”
 - (c) Pull these changes into main.
6. **Check that information is correct in `about.py`**
 - (a) Check developers
 - (b) Check contributors
 - (c) Check translators
 - (d) Commit “Updated about”
 - (e) Pull these changes into main.
7. **Run “`python tools/execute-specs.py`” to find possible errors**
 - (a) Fix errors
 - (b) Commit
 - (c) Pull these changes into main.
8. **Run “`python release/make-source-release.py`”**
 - (a) This script creates the file timeline-x.y.0.zip file
 - (b) It also runs the tests as in section 5.
9. **Try running the unzipped release to make a basic check that it works**
 - (a) Mail copies to all developers, to let them test before publishing
10. Tag the release 1. Run “`hg tag x.y.z`”
11. Update repository 1. `hg push`

10.6.3 Launchpad

1. **Upload new pot-file (So that new texts are found)**
 - (a) Create new pot-file Run the script `tools/generate-pot-file.py`
 - (b) **Request upload from here (login required)** <http://translations.launchpad.net/thetimelineproj/trunk/+translations>

Status of upload can be checked at <https://translations.launchpad.net/thetimelineproj/trunk/+imports>

10.6.4 Work on main

1. Move stable repo to main repo

- (a) `cd main`
- (b) `hg pull ../stable`
- (c) `hg push`

2. Change versions numbers in main to denote the next version (x.y+1.0)

- (a) `version.py`
- (b) `changelog.rst`
- (c) `README`
- (d) Run “python tools/execute-specs.py” to find where else you need to modify

3. Commit and push

10.6.5 Publish

1. Upload the timeline-0.xx.0.zip file to Source Forge

2. Create windows binary package and upload to Source Forge

- (a) copy stable to new directory
- (b) Execute build script to create install-exe “stable-copyreleasewincmdbuild_install_exe.cmd”
- (c) Test the install-exe execute stable-copybinSetupTimeline9nn9Py2Exe.exe
- (d) Try running the installed Timeline to make a basic check that it works
- (e) **Upload the install file to Source Forge**
 - vi. Ensure that the exe file has “Default Download For” Windows checkbox checked and ensure that the the zip file has all the others checked

3. Make release announcement:

- (a) Post news to SF (<http://sourceforge.net/p/thetimelineproj/news/?source=navbar>) You need to login
- (b) Post news to Freecode (<https://freecode.com/projects/timeline-2>) You need to login

4. Notify developers of repo change

- (a) Send email to thetimelineproj-user@lists.sourceforge.net

10.7 Make a string translatable

If you introduce a new string in the source code that you would like to be translated into different languages, you need to do 2 things:

1. Enclose the string in `_()`
2. Upload a pot file to [Launchpad](#)

Example of enclosed string:

```
print(_("Hello!"))
```

To generate a pot file, run

```
python tools/generate-pot-file.py
```

It requires the `xgettext` program to be on the path. See [gettext](#) for instructions how to install it.

You should now have a file:

```
translations/timeline.pot
```

that contains an entry for your string looking something like this:

```
#: source/timeline.py:58
msgid "Hello!"
msgstr ""
```

Upload this pot file to Launchpad using this form: <https://translations.launchpad.net/thetimelineproj/trunk/+translations-upload>. (It requires login.)

Now translators will see your new string and can translate it.

10.8 Run Timeline in a different language

First you need to compile the translation files:

```
python tools/generate-mo-files.py
```

It requires the `msgfmt` program to be on the path. See [gettext](#) for instructions how to install it.

Then you need to change your locale before running Timeline. On a Linux system, you can do it like this:

```
LANG=sv_SE python source/timeline.py
```

10.9 Translate Timeline into another language

Timeline is translated on **Launchpad**: <http://translations.launchpad.net/thetimelineproj>.

For instructions how to **get started** translating on Launchpad, see: <http://help.launchpad.net/Translations/StartingToTranslate>.

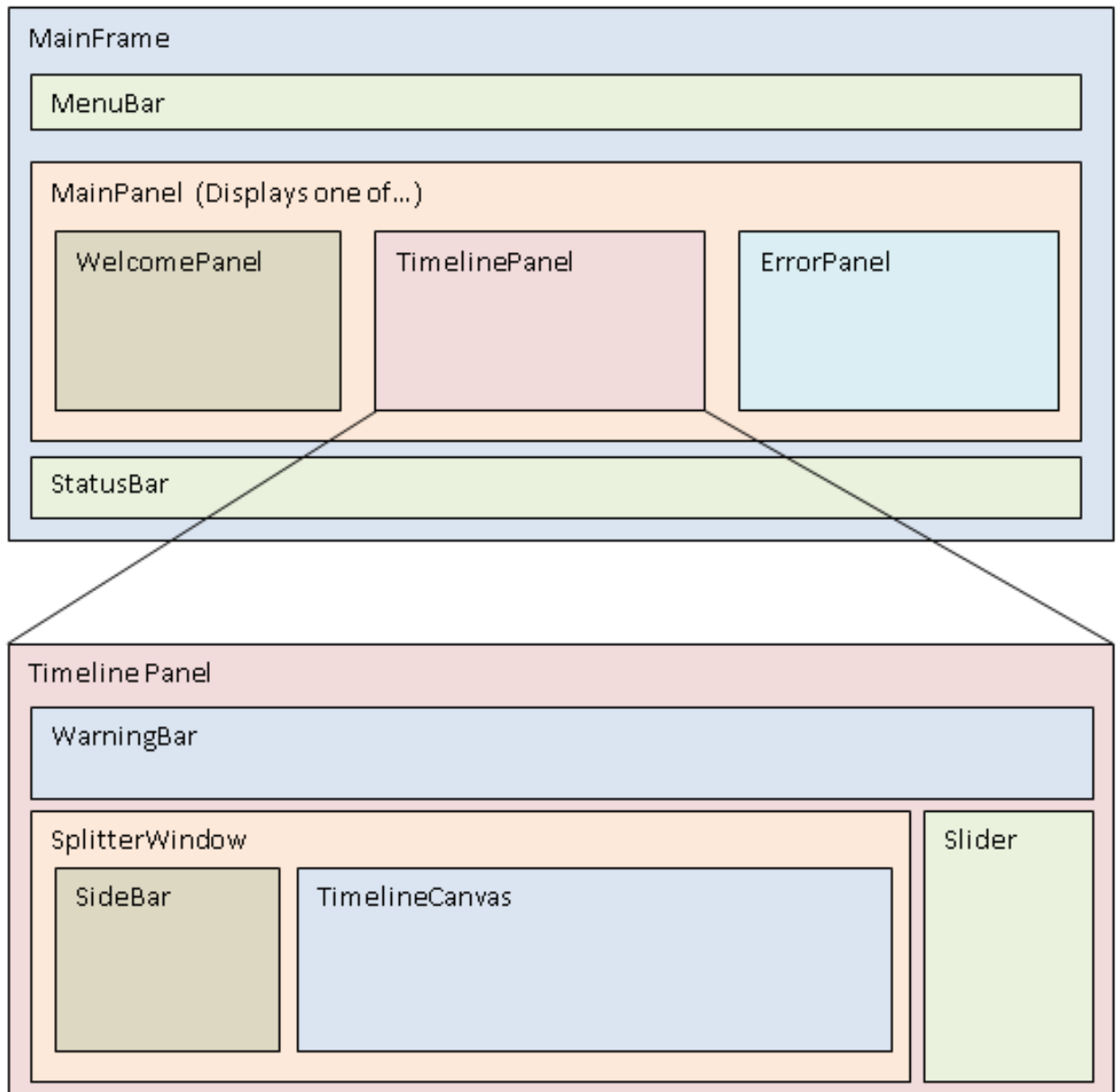
CHAPTER 11

Architecture overview

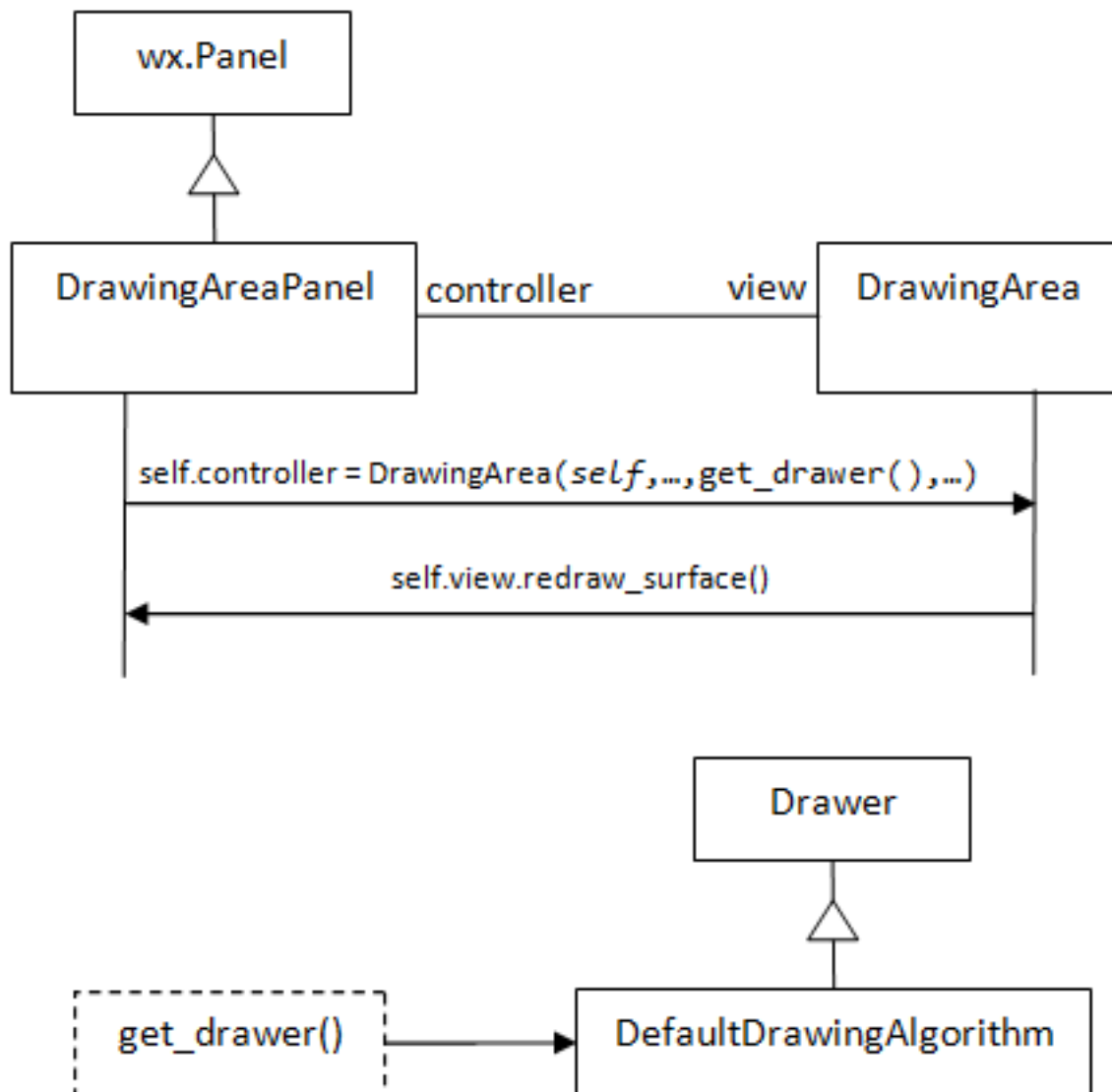
This page intends to explain the internal architecture of Timeline on a high level.

This page includes diagrams of some of the classes in Timeline. They can remind you of how classes work, but they are not intended to fully explain the code.

11.1 GUI classes



11.2 Drawing

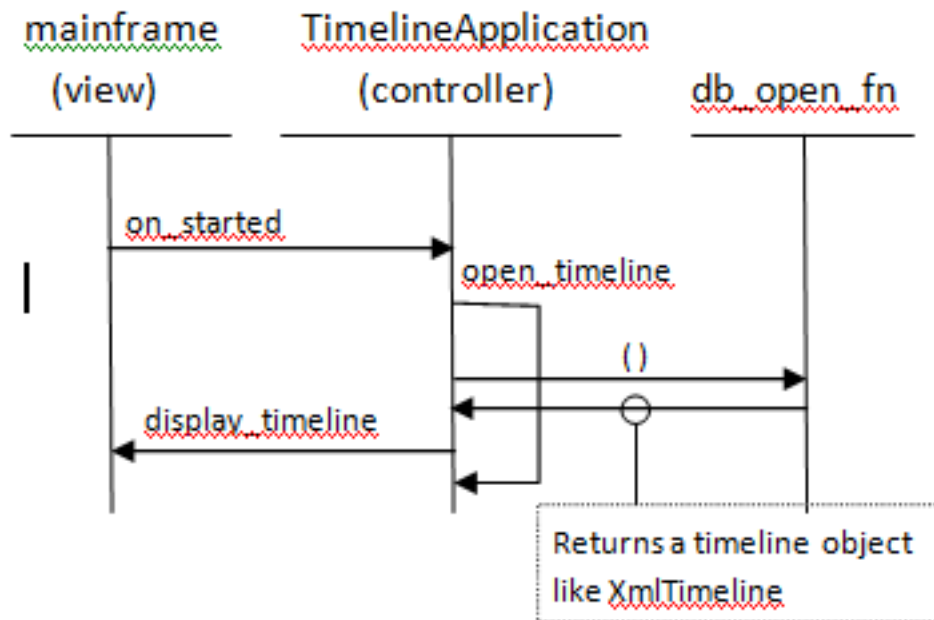


For the purpose of making testing easier we try to divide gui-components in two parts: the GUI class itself and another class, the controller, containing the 'business' logic. The goal is to have no 'business' logic at all in the GUI class. It should only have simple gui actions.

Another design decision for the purpose of making testing easier is to inject objects into the constructor of a class. This can be seen here when the `DrawingAreaPanel` creates its controller, the `DrawingArea`, it passes the drawer object in the constructor, `get_drawer()`.

11.3 Startup

Opening a timeline on application startup



11.4 System documentation - API

System documentation.

12.1 SourceForge

Timeline is mainly hosted by SourceForge.

Project URL: <http://sourceforge.net/projects/thetimelineproj>

12.2 Source control

We use Mercurial and the repository is hosted at SourceForge.

URL: <http://sourceforge.net/p/thetimelineproj/main/ci/default/tree/>

12.3 Mailing list

We use a mailing list provided by SourceForge. We use only a single list, and we use it for all kinds of communication.

- **Address:** thetimelineproj-user@lists.sourceforge.net.
 - You can send emails to this address and everyone that is registered will receive your message. You **don't** need to be registered to send emails to the mailing list.
- **Browse the archive:** [browse](#).
- **Search the archive:** [search](#).
- **Register:** [register](#).

12.4 Build server

We host our own Jenkins server to automatically test Timeline on different platforms whenever the source code changes.

If a build fails, an email will be sent to the mailing list.

URL: <http://jenkins.rickardlindberg.me>

12.5 Translation service

We use Launchpad to handle translations.

URL: <http://translations.launchpad.net/thetimelineproj>

13.1 Version 1.19.0

Planned release on 31 December 2018.

Don't want to wait for the final release? Try the beta version!

- [Download source](#).
- [Download windows installer](#).

Export: * Not all events are shown in export listbox when filtering is turned off.

 Include events without category when filtering is turned off.

13.2 Version 1.18.0

Released on 30 July 2018.

GUI:

- Added new representation of fuzzy edges when selecting view: Other Gradient Event box drawer with fuzzy edges. (#174)

Fixed crash reports and bugs:

- Wrong editor is opened when right-click and selecting edit, on a milestone. Check if event is milestone before selecting editor.
- Milestones can convert to ordinary events when a timeline is compressed. Milestones is no longer part of the compression algorithm..
- Balloons are always shown for hooverd events. Balloons are not shown if menu "View/Balloons on hover" is disabled.

- `AttributeError: 'NoneType' object has no attribute 'get_ends_today'`. Event object existence is checked before getting attribute.
- `InvalidOperationError: Circular category parent`. A circular parent is no longer possible to select. (This bug was introduced in the 1.16.0 release.)

13.3 Version 1.17.0

Released on 25 Mars 2018.

GUI:

- If the xml contains a description field for a container, it will now be displayed in a balloon, when hovered.
- Selected events are not deselected when scrolling timeline with mouse.
- Events can be selected with alt + mouse drag.
- Events exported to listbox can now be filtered by visible categories

Fixed crash reports and bugs:

- `PyAssertionError: C++ assertion "(itemid >= 0 && itemid < SHRT_MAX)"` Eliminated menu id creation by using constant values.
- `ValueError: Start time can't be after end time` This happened when ends-today flag was set, and start-time was in future.
- A change by another user is now detected when Timeline is closing.
- `PyAssertionError: C++ assertion "node" failed at ..\..\src\msw\menu.cpp(863) in wxMenu::DoRemove(): bug in wxMenu::Remove logic` This happened when context menu has been used and another timeline is opened.
- It's now possible to change the background colour again.

13.4 Version 1.16.0

Released on 13 November 2017.

GUI:

- Using context menu no longer causes toolbar menu to stop working.
- Balloon text font is now settable in preferences dialog.
- Sample text for font preferences are now coloured also.

Fixed crash reports and bugs:

- `AttributeError: 'NoneType' object has no attribute 'GetParent'` This happens when System info dialog is opened by context popup menu.

13.5 Version 1.15.0

Released on 31 July 2017.

GUI:

- Path to the configuration file is displayed in the System Info dialog.
- Date format is now displayed in the System Info dialog, as configured.
- Era rectangle is always visible, even when zooming out far.
- Text in a balloon can now be displayed besides or under an icon.

Fixed crash reports and bugs:

- `UnicodeEncodeError: 'ascii' codec can't encode character u'\u03c0' in position 0: ordinal not in range(128)` This happened when the BC label contained non-ascii characters.
- `UnicodeEncodeError: 'ascii' codec can't encode characters in position 18-21: ordinal not in range(128)` This happened when a font face name contained non-ascii characters.
- Events highlighted during search sometimes get stuck in highlighted state.
- `PyAssertionError: C++ assertion "!wxMouseCapture::stack.empty()" failed at ../../src/common/wincmn.cpp(3319) in wxWindowBase::ReleaseMouse(): Releasing mouse capture but capture stack empty?` This happens in when dragging the mouse from the calendar control.

13.6 Version 1.14.0

Released on 8 May 2017.

Calendar:

- BC years are formatted correctly in status bar.
- Decades and centuries are correctly represented around year 0 and in BC years. (Centuries are now denoted 1900s and represent the years 1900-1999.)

GUI:

- The formatting of the time duration for Gragorian time is more intuitive.
- All events can be selected with a menu command
- View selection to hide/show events done (progress = 100%).
- The limitation of number sizes has been removed in the numeric event editor.
- Now the position of the legend can be changed.

Fixed crash reports and bugs:

- Now weekends can be colorized again. (#170)
- It's no longer possible to close the milestone editor dialog with an invalid date/time. (#171)
- The event progress bar is now correctly drawn when event is partly outside of screen.
- `OverflowError: long int too large to convert to float.` (#126)
- `wx._core.PyAssertionError: C++ assertion "Assert failure" failed at ../../src/gtk/menu.cpp(1300) in GetGtkHotKey(): unknown keyboard accel.` This was caused by incorrect translations.
- `TypeError: %d format: a number is required, not TimeDelta.` This happened when trying to measure the distance between two overlapping events in a numeric timeline.

- `IndexError: list index out of range`. This happened under some circumstances when zooming out far and scrolling to the far left.
- `AttributeError: 'int' object has no attribute 'seconds'`. This happened when starting a slideshow with a numeric timeline.

13.7 Version 1.13.0

Released on 31 January 2017.

GUI:

- The naming strategy of overlapping Era's has been changed
- Major strip labels are drawn vertical when they don't fit in horizontal space.
- Balloon width is no longer dependent on the event width, so the text don't disappear too early.

Exporting:

- How to handle encoding errors, when exporting events to file, can now be selected.
- The events in a timeline can now be presented as a slideshow in a web browser.

Fixed crash reports and bugs:

- A Milestone can now have an empty text without crashing. (#165)
- Now an Era in a numeric timeline can have "ends today" without crashing. (#166)
- `NotImplementedError`: I don't believe this is in use. (#168)
- Now you can tab out of an invalid date field without crashing. (#169)

13.8 Version 1.12.0

Released on 31 October 2016.

GUI:

- Era's now have an ends-today property. (#159)

Documentation:

- Help pages updated.

Data:

- Option to switch off time for entire project. (#157)
- Sample text is displayed for fonts in the preference dialog

Export SVG:

- Eras are now drawn in the SVG image. (#144)
- Improved drawing of labels in SVG image. (#145)
- Timeline background colour is used in SVG image.

Fixed crash reports and bugs:

- Milestones are handled correctly when undoing changes.

- Duplicate categories in ics file is now handled correctly (#160)
- Invalid date and time entries, now generates error message. (#163)
- Creating exception message should not fail now. (#161)
- Duplicate dir names in directory Timeline is now handled. (#162)

13.9 Version 1.11.0

Released on 2 August 2016.

Data import:

- VTOD elements are now imported, as events, from ics files. (#142)
- Import options can now be specified when importing events, from ics files. (#141)

Data export:

- When exporting a timeline to images a merged image is also created.

Translations:

- Made label texts in 'Export to Listbox', translatable. (#147)

GUI:

- A checkmark can now be displayed in front of the event text when the event is done (100% progress). (#134)
- The duplicate event dialog can be opened from the event editor dialog (#131)
- After a search match the found event is highlighted
- The background colour can now be user defined. (#151)

Data:

- Introduced the special event type, Milestone.

Navigation:

- Now it's possible to return to the previous time period after a navigation. (#153)

Bug fixes:

- Bosparanian date format crashes.
- Timeline menu items are now disabled when no timeline is opened. (#148)
- Float division by zero when mouse moved. (#150)

13.10 Version 1.10.0

Released on 30 April 2016.

Calendar:

- Locale date formatter can now handle abbreviated month names in locale format pattern. (#133)
- The locale date format is now replaced with a user defined format

GUI:

- Users can now design and use their own icons for fuzzy, locked, and hyperlink. (#93)

- The vertical zoom (menu or Alt +/-) now zooms instead of scrolling.
- Ctrl+Shift+MouseWheel now scrolls vertically instead of zooming.
- Marking invalid dates with pink background now works correctly even in Windows.
- The date controls should now follow the locale date formatting setting.
- Weekdays can now have a colour different from the background.
- Scrolling timeline after regaining focus now works properly even in Windows. (#138)
- The vertical space between events is now a user settable preference.

Translations:

- The BC string in strips is now translatable

Fixed crash reports:

- The Timeline xml file is updated when an Era is deleted (#139)
- Import events dialog gives UnicodeEncodeError if exceptions contain unicode messages.

Import:

- Categories are now created when importing ics data (#141)

Export:

- Data in Export to Listbox can now be copied to clip board (#146)

13.11 Version 1.9.0

Released on 31 January 2016.

Calendar:

- Locale date formats correctly at start of timeline. (#116)

GUI:

- There is an optional tool bar that contains buttons for toggling some settings.
- “To time” in event editor is correctly laid out when checking “Period”.
- Images can be dragged and dropped on an event to change icon. (#103)
- A preference decides if the time checkbox is checked for new events. (#119)
- Subevents in a container can be locked if the extended container strategy is used. (#110)
- The description text in the event editor can be selected with Ctrl+A. (#115)
- The ends-today checkbox in the event editor is enabled when the editor is opened from the menu. (#114)
- The events in the exported list are sorted by start date. (#106)
- Colors can be selected for major strip lines, minor strip lines and now line. (#111)
- Overlapping eras are now displayed in a mixed color. (#108)
- Colors can now be selected for events without an associated category. (#81)
- The Ends-today property can be set on subevents if the extended container strategy is used.
- A new dialog in the help menu displays System information.

Translations:

- The wx stock items are translated correctly in the Windows binary. (#109)
- The strip text ‘Century’ is translatable. (#107)

Bug fixes:

- Edit event dialog does not crash when there is a db error. (#127)
- Application does not crash at startup if system has locale zh_CN (Chinese). (Merged from 1.5.1.)
- Application does not crash when duplicating container events. (#125)

13.12 Version 1.8.1

Released on 10 November 2015.

This is a bugfix release. It fixes a critical bug that disables editing numerical timelines.

Fixed crash reports:

- `AttributeError: 'NumTimePicker' object has no attribute 'show_time'` (#117)

13.13 Version 1.8.0

Released on 31 October 2015.

This is a periodic release.

Calendar:

- Timelines can be created using the “The Dark Eye” (Das Schwarze Auge, DSA) official calendar.

Drawing:

- When you scroll vertically by dragging, the view moves proportionally. (#88)
- Containers expand vertically when they contain overlapping events. This is an experimental feature that must be enabled. (#39)
- You can zoom out to a period longer than 1200 years. There is no longer a limit. (#90)

Exporting:

- Exporting to CSV behaves properly when there is a newline in the description of an event. (#92)

GUI:

- All dialogs have a polished and more uniform look.
- When creating a new timeline, a dialog pops up that let’s you choose what type of timeline you want to create. (#97)
- Event and eras can be created with a period longer than 1200 years. There is no longer a limit. (#98)
- When duplicating an event with period month it behaves properly in edge cases.

Fixed crash reports:

- `PyAssertionError: C++ assertion "wxAssertFailure" failed at ..\..\src\common\stockitem.cpp(166) in wxGetStockLabel(): invalid stock item ID` (#95)

- `KeyError: <bound method Font.Underlined of <timelinelib.wxgui.components.font.Font; proxy of <Swig Object of type 'wxFont *' at 0x8f240f0> >> (#83)`
- `string index out of range (#85)`
- `AttributeError: 'NoneType' object has no attribute 'julian_day' (#96)`
- `ValueError: julian_day must be >= 0 (#79)`
- `LockedException: Unable to take lock on ... (#105)`

13.14 Version 1.7.1

Released on 17 August 2015.

This is a bugfix release. It fixes a critical bug where data could be lost.

Data:

- Content of .timeline file is not erased when it is opened. This was a bug that has now been fixed.

Drawing:

- Minor strip font is only bold for weekend days. A bug made it a bit random before.

Fixed crash reports:

- `AttributeError: 'module' object has no attribute 'Color'`
- `AttributeError: 'EventEditorDialog' object has no attribute 'set_focus' (#89)`

13.15 Version 1.7.0

Released on 30 July 2015.

This is a periodic release of Timeline. It contains many solutions to problems identified by users of Timeline.

Data:

- Events can have multiple hyperlinks. (#30)
- An experimental feature allows entering dates before 4714 BC. This allows larger time periods to be created. (#51)

Drawing:

- An icon is drawn in the event box if it has hyperlinks. This makes it easier to see which events have hyperlinks. (#29)
- Period events can be configured to never be drawn above the center line. This should make it more obvious which events are period events and which are point events. (#42, #46)
- A setting exist that decides if event texts should be centered or not. (#73)
- There is no horizontal padding between events. This allows more events to fit on the screen. (#2)
- Some fonts used to draw the timeline can be customized. This should allow users to customize the look of their timelines to their taste. (#63)
- A setting can draw point events with the left box edge at the vertical line. This makes it more clear where the event starts in time. (#60)

GUI:

- A notification is shown when a shortcut is saved. (#23)
- The category editor can be opened with double click. This makes the intuitive way to open the editor possible. (#47)
- The period checkbox in the event editor remembers its value from last time. This should speed up entering of period events. (#28)
- Multiple events can be added to a category by selecting them and selecting a context menu item. This should make it more convenient to assign categories. (#67)
- The tab-order of controls in the event editor dialog can be customized. This allows users to put their most frequently used controls first. (#62)
- The divider line can be adjusted with mouse dragging. This should make it more convenient to use Timeline on a touch device. (#58)
- Events can be moved vertically by selecting them and pressing Up/Down or selecting menu items. This makes it more obvious how to move events vertically. (#45)

Exporting:

- Exporting a whole timeline to several images now preserves the vertical position of events between images. So now images can be put together and the events will align correctly. (#72)

Misc:

- Undo works after compress. This allows users to undo compress action if the result was not desirable. (#65)
- Does not fail to open Timeline files that have period wider than 1200 years. This should prevent users from having to manually edit the xml file. (#8)
- Crash reports have information about locale settings. This makes it easier to troubleshoot errors depending on locale settings. (#54)

Fixed crash reports:

- `AttributeError: 'EraEditorDialog' object has no attribute 'on_return'` (#57)
- `KeyError: '33'` (#53)
- `KeyError: 'Nov'` (#50)
- `ValueError: Invalid date.` (#55)
- `LockedException: Unable to take lock on...` (#69)
- `OverflowError: long int too large to convert to float` (#75)
- `Exception: No timeline set` (#56)
- `TypeError: unsupported operand type(s) for +: 'int' and 'TimeDelta'` (#48, #78)
- `WindowsError: [Error 32] The process cannot access the file because it is being used by another process` (#33)
- `UnicodeEncodeError: 'ascii' codec can't encode character u'\xc9' in position 0: ordinal not in range(128)` (#49)

Windows specific:

- The log file is created in a standard user temp directory. This ensures that even if Timeline is installed in a read-only location, the log file can be created. (#74)

- Broken fragments of sidebar is not shown at startup. (#52)

13.16 Version 1.6.0

Released on 30 April 2015.

Solved problems:

- Dividerline slider pos preserved between sessions
- Introduced a Gradient Event box drawer
- A new Event box drawer is added (gradient draw)
- When selecting period in event editor - end date = start date + 1 day
- Introduced background Era's
- Bitmaps used to mark fuzzy and locked edges
- Fixed crash when opening preferences dialog (wxPython 3.0.2.0)
- Fixed crash when opening hyperlink
- Fixed crash when using experimental feature locale date
- Fixed crash when entering non-ascii characters in feedback dialog subject or text
- Crash report: AttributeError: 'MainFrame' object has no attribute 'open_timeline' (#22).
- Crash report: PyAssertionError: C++ assertion "Assert failure" failed at ../src/common/sizer.cpp(1401) in DoInsert(): too many items (9 > 24) in grid sizer (maybe you should omit the number of either rows or columns?) (#21). This was only a problem with wxPython 3.
- Crash report: KeyError: '33' (#26). This happened when using experimental feature 'locale date'.
- Added export function timeline -> CSV
- Crash report: ValueError: to_julian_day only works for positive julian days, but was -32104 (#43).

13.17 Version 1.5.1

Released on 4 December 2015.

Bug fixes:

- Application does not crash at startup if system has locale zh_CN (Chinese)

13.18 Version 1.5.0

Released on 31 January 2015.

New features, enhancements:

- Made progress bar thinner to improve visibility
- Made progress- and done-colors selectable
- Deeper zooming, to one minute, enabled

- Introduced the concept of ‘Experimental features’
- Experimental feature - Mark event as done
- Experimental feature - Extend container height
- Experimental feature - Locale date formats

Bug fixes:

- Fixed: Crash report: Duplication subevent
- Fixed: Crash report: Clicking Return in datetimepicker in Event alert editor
- Fixed problem with duplication of containers
- Fixed problem with menus requiring a timeline

13.19 Version 1.4.1

Released on 12 November 2014.

Bug fixes:

- Fixed: Crash report: AttributeError: ‘MemoryDB’ object has no attribute ‘events’

13.20 Version 1.4.0

Released on 9 November 2014.

New features, enhancements:

- Added undo feature
- Added a context menu to the timeline window
- Added a notification window at the top of the screen when opening a read-only timeline or a timeline that is not saved on disk
- Expanded range of numeric time picker
- Added import dialog

Bug fixes:

- Fixed the following error when using wxPython >= 2.9: AttributeError: ‘module’ object has no attribute ‘Color’
- Fixed the following error: iCCP: known incorrect sRGB profile
- Fixed navigation problem, go to time, for numeric timeline
- Synchronizing a timeline that has been modified by someone else actually reads the modified timeline instead of ignoring it. (This bug was introduced in version 1.1.0.)

13.21 Version 1.3.0

Released on 30 June 2014.

New features, enhancements:

- Event description included in search target.
- Search result can now be presented and selected in a listbox
- CategoriesEditor is now resizeable

Bug fixes:

- Scrolling with PgUp/PgDn does not crash when it would end up on non-existing Feb 29 ([bug report](#))
- Prevent PyAssertionError when opening category editor (wxPython 3.0.0.0)
- Fit millennium does not crash if timeline is far to the left
- Some Edit menu items are disabled when there is no open Timeline

13.22 Version 1.2.4

Released on 7 April 2014.

Bug fixes:

- Exception in event editors when “Add more events after this one” is checked

13.23 Version 1.2.3

Released on 5 April 2014.

Bug fixes:

- Shortcuts dissappear when navigation menu is created

13.24 Version 1.2.2

Released on 5 April 2014.

Bug fixes:

- Uninitialized flag comes into play when opening an ics file

13.25 Version 1.2.1

Released on 5 April 2014.

Bug fixes:

- Encoding problems with navigation menus and shortcut configuration.

13.26 Version 1.2.0

Released on 5 April 2014.

New features, enhancements:

- Shortcuts can be user defined.
- Events now have a progress attribute.
- Find feature for categories with Ctrl+F when mouse in category tree.
- Event duration is displayed in the status bar
- Alert dialog appears on top and beeps when shown

Bug fixes:

- Exception when opening event editor from menu for a numeric timeline.
- Incorrect display of decades BC, fixed.
- Contents indicator is drawn even when no balloon data exists.
- End date is set to now in validate function when ends-today is checked

13.27 Version 1.1.0

Released on 28 December 2013.

New features, enhancements:

- Century labeling changed. Century 0 is now removed
- Menus for Zoom In and Zoom Out
- Menus for vertical Zoom In and vertical Zoom Out
- Numeric Timeline
- New category tree in sidebar

Bug fixes:

- SVG export can handle ampersand (&) in event text
- SVG export can handle more characters by using UTF-8 encoding
- Prevent overflow error when zooming in on wide events
- Prevent error when using up arrow to increase month in date editor
- Prevent error when fitting all events and they almost fit
- Move event vertically, can be done for events very close to each other (with different y-coordinates)
- Ics-files could load events without text which caused an exception when trying to 'Save As'
- Handle exception in dragging situation when julian day becomes < 0.

13.28 Version 1.0.1

Released on 4 October 2013.

Bug fixes:

- Events Disappearing when zooming

13.29 Version 1.0.0

Released on 30 September 2013.

After about 4.5 years in development, Timeline 1.0.0 is released. This is the first time we increment the x-component of the version number (*A note about version numbers*). The main reason for doing so is that Timeline can no longer read files produced with Timeline versions before 0.10.0 (released over 3 years ago).

The other big thing in 1.0.0 is that the experimental support for dates before year 0 is no longer experimental. We have rewritten large parts of the date handling partly to be able to support BC dates in a better way.

New features, enhancements:

- Implemented export to image for whole timeline
- Implemented vertical zooming with Alt+Mousewheel
- Implemented vertical scrolling of timeline events
- Select all, Ctrl-A implemented in event editor description
- New entries in categories tree context menu allowing parent/children check/uncheck
- New checkbox under categories tree, used to view categories individually independent on parent checked-status
- Dialog for sending feedback (available from help menu and event editor)
- Balloon size restricted to not expand over timeline border
- Help documentation updated
- Show numerical day number together with day name when zooming to week

Bug fixes:

- Fixed exception when right-clicking in CategoriesEditor
- When 'ends today' start time can't be > now, anymore
- Search bar gives no exception when searching twice or using search button

Removed features:

- Printing: Use export to image and print image instead
- Old Timeline file format: Last used in version 0.9.0

Non-visible changes:

- Adjustments made to be able to use wxPython version 2.9
- Replaced internal time type to support dates before year 0

13.30 Version 0.21.1

Released on 7 July 2013.

Bug fixes:

- Bug fix. Exception when exporting image

13.31 Version 0.21.0

Released on 30 June 2013.

New features, enhancements:

- Added feature, Set category on selected events
- Added feature, Set category on events without category
- Added ‘Import’ feature that makes it possible to merge timelines.
- Added ‘Edit Event’ menu

Bug fixes:

- Bug fix. Allow Preferences setting when no timeline exists
- Bug fix. Reset selected events list when selected events are deleted

13.32 Version 0.20.0

Released on 30 March 2013.

New features, enhancements:

- Added ‘Save As’ feature
- Strategy for allowing multiple users to use the same Timeline file.
- The timeline view regains focus when the event editor is closed.
- Enter-key works in date and time fields of the event editor
- Some help texts updated
- New version of icalender to cope with years before 1900
- TimelineComponent can explicitly clear the drawing area

Bug fixes:

- Fixed problem with Event texts starting with ‘(’- or ‘[’-character
- Delete event by context menu now works

13.33 Version 0.19.0

Released on 30 December 2012.

New features, enhancements:

- Possibility to define URL on events and execute “Goto URL” to open web browser.
- Implemented ‘fit week’ navigation function.
- Help text added, to describe vertical movement of events.

Bug fixes:

- Build script generates zip file with only LF as line endings in files
- Year 0 removed from timeline display when using extended date range

13.34 Version 0.18.0

Released on 30 September 2012.

New features, enhancements:

- Zooming with scroll wheel zooms at cursor position instead of center.

Bug fixes:

- Adding multiple events without closing event dialog, works again.
- Alert time comparison problem solved
- Fixed problem with ends-today property
- Fit millennium now works close to edges
- Fit century now works close to edges

13.35 Version 0.17.0

Released on 15 June 2012.

This is a new feature release.

New features, enhancements:

- Possibility to define alerts on events.
- Non-period events can be added to container events

Bug fixes:

- No Error when fitting month, december, when using extended timetype.

13.36 Version 0.16.0

Released on 31 January 2012.

This is a new feature release.

New features, enhancements:

- Events can be grouped in containers

Bug fixes:

- Timeline files with non-English names can be opened
- Creating new locked events does not raise exception

13.37 Version 0.15.0

Released on 30 October 2011.

This is a new feature release.

New features, enhancements:

- Custom font color for categories
- Measure distance between events
- Only break text in balloon if needed to keep balloon on screen

Bug fixes:

- SVG export can now handle text with non-english characters
- Long category names are now visible in category editor
- Timeline repaints after editing category color
- No year of out range exception in event dialog

13.38 Version 0.14.0

Released on 30 July 2011.

This is a new feature release.

New features, enhancements:

- Move all selected events
- Mark event period as fuzzy and edges will change to triangles
- Mark event period as locked and edges will be curved and the event can not be moved or resized
- Mark event as ending today and its period will be updated to end today
- Experimental support for inertial scrolling (can be enabled in preferences)
- Shows status text when zooming

Bug fixes:

- Not possible to select too large period when zooming with shift+drag
- Prevent exception (in cases when year was out of range) when scrolling with page up/down
- Show user friendly message when creating event with too long period
- Display error message in status bar if period is too long when resizing event
- No time exception when exporting to SVG
- No exception when using extended date range and exporting to SVG

13.39 Version 0.13.0

Released on 30 April 2011.

This is a new feature release.

New features, enhancements:

- Events can be moved up and down with Alt+Up/Down
- Hidden event count is shown in status bar
- Event text changes color to white if background is dark
- Timeline can be scrolled with Alt+Left/Right

- Edit category button added in categories editor
- Export to SVG

Bug fixes:

- No exception if “Fit all events” results in a period too large to display
- No error if pressing left or right in empty categories tree control

13.40 Version 0.12.1

Released on 30 January 2011.

This is a translation update and bugfix release.

Bug fixes:

- Menu items are correctly disabled if no timeline is open
- Clicking calendar button when an invalid date is entered gives error message instead of exception
- LANG environment variable is only set on Windows to prevent locale error at startup on Linux systems
- Fit all events ignores hidden events

13.41 Version 0.12.0

Released on 9 January 2011.

This is a new feature release.

New features, enhancements:

- Experimental support for extended date range (before 1 AD)

Bug fixes:

- Centuries before 10th are displayed correctly (9 instead of 90)
- Correct translations are used on Windows

New translations:

- Lithuanian
- Vietnamese

13.42 Version 0.11.1

Released on 24 October 2010.

This is a translation update and bugfix release.

Bug fixes:

- Create event through menu does not raise exception
- Time removed when saving event and ‘Show time’ not checked

13.43 Version 0.11.0

Released on 12 October 2010.

This is a new feature release.

New features, enhancements:

- New improved date and time entry control
- New navigation function: fit millennium

Bug fixes:

- Remove import of wx.lib.wordwrap that caused a crash on Ubuntu

New translations:

- Italian
- Turkish

13.44 Version 0.10.2

Released on 11 June 2010.

This is a translation update and bugfix release.

Bug fixes:

- “Add more events after this one” does not give error message when ticked in the create event dialog
- Do not write empty displayed_period tag to xml file
- Prevent application from crashing with wxPython version 2.8.11.0

13.45 Version 0.10.1

Released on 25 May 2010.

This is a translation update release.

New translations:

- Polish
- French

13.46 Version 0.10.0

Released on 9 May 2010.

This is a new feature release.

New features, enhancements:

- Switch to XML-based file format for storing timeline data
- Support hierarchical categories

- Function to duplicate events according to a pattern
- More user friendly error when application crashes
- Save window position
- More shortcuts for navigation commands
- Selected event gets highlighted line

Bug fixes:

- Application shows error message in category editor instead of crashing

13.47 Version 0.9.0

Released on 7 February 2010.

This is a new minor feature and bugfix release.

New features, enhancements:

- Timeline scrolls when creating period events, resizing events, and moving events
- Option to start weeks on Sundays
- Balloon shown shorter time after mouse out
- New navigation functions: year, month, week forward/backward
- Middle mouse click centers timeline on that spot
- Shift+Scroll moves horizontal line up/down

Bug fixes:

- Fixed issues with 'Go to Date' dialog
- Balloon now visible even if event stretches outside screen
- All keys now work in the search bar
- Prevent crash if long period events are used
- Small corrections to documentation

13.48 Version 0.8.0

Released on 1 January 2010.

This is a new minor feature release.

New features, enhancements:

- Basic search function
- Weekend day numbers are drawn in bold in month view
- Experimental read-only support for ics files
- Timeline that shows last modified dates of files in a directory
- Allow balloons to stick
- Write files in a safer way without permanent backups

- New navigation functions: find first, find last, fit century, fit decade, fit all
- New icons in help browser (Windows)
- Man page (GNU/Linux)

Bug fixes:

- Fit month and fit day now work for December and last day of month
- The same help page can now be opened again after the help browser is closed
- Recently opened list can't contain the same file twice now

New translations:

- Hebrew (Yaron Shahrabani)
- Catalan (BennyBeat)

13.49 Version 0.7.0

Released on 1 December 2009.

This is a new minor feature release.

New features, enhancements:

- Visual move and resize of events
- Snap when creating, moving, and resizing events
- Show balloons with event information on hover
- Associate icons with events (shown in balloons)
- Improved drawing of events: new selection and data indicator
- Added context menu for events

New translations:

- Russian (Sergey Sedov)

13.50 Version 0.6.0

Released on 1 November 2009.

This is a new minor feature release.

New features, enhancements:

- Added shortcuts for editing categories from the event editor dialog
- Mapped backspace key to previous page in help browser
- Added option to open most recent timeline at startup (default yes)
- Show exact time of an event in status bar
- The y position of the divider between period events and single point events can now be adjusted

Bug fixes:

- Period events with description now has correct width

- The legend is now always drawn on top of events

13.51 Version 0.5.0

Released on 1 October 2009.

This is a new feature release.

New features, enhancements:

- Added ‘Open Recent’ menu
- Replaced manual with a wiki-like help system
- Visualize description of selected events in balloons
- Improved error messages when reading or writing timeline data fails
- Added functionality for printing timeline
- Added new navigation functions: Backward/Forward
- Added welcome panel that shows if no timeline is open

New translations:

- Dutch (Koert Loret)

Bug fixes:

- Fixed problem on Windows where you could not enter dates before 1752-09-14

13.52 Version 0.4.0

Released on 1 September 2009.

This is a new feature release.

The first step in supporting additional data for events has been implemented. The file format had to be changed for this. Files written by version 0.4.0 will not be readable by previous versions, but 0.4.0 can read 0.3.0 files and will convert them automatically.

New features, enhancements:

- Translation support
- Export to Image
- Legend for categories
- Longer descriptions for events (visualization will be implemented in 0.5.0)

New translations:

- Swedish (Roger Lindberg)
- Spanish (Roman Gelbort)
- German (Nils Steinger)
- Brazilian Portuguese (Leonardo Frigo da Purificação)

13.53 Version 0.3.0

Released on 1 August 2009.

In this release the documentation has been improved and a few bugs have been fixed.

The file format has also been updated to decrease the risk of losing data. Users are therefore strongly encouraged to upgrade to this version. The file format is readable by the 0.2.0 version but it can not take advantage of the new format.

New features, enhancements:

- Changed to allow events without categories.
- Improved what's displayed in the title bar (open file name first).
- Added application icon.
- Added Help menu.
- Converted user manual to DocBook format.
- Integrated user manual with application (first step).
- Started experimenting with unit tests.
- Added copyright notes to all source files.
- Added AUTHORS, CHANGES, COPYING, and INSTALL.

Bug fixes:

- Fixed bug where application raised exceptions when scrolling to the very end or the very beginning of time (year 10 or year 9999).
- If multiple timelines were opened, the displayed period would just be saved for the last opened one. That is fixed now so it is saved for all.

13.54 Version 0.2.0

Released on 5 July 2009.

This version contains lots of improvements.

File format written by this version is not readable by previous versions.

New features, enhancements:

- Added support for showing and hiding events from certain categories.
- Added a week view in one zoom level of the timeline.
- Added navigation functions such as 'Go to Date' and 'Go to Today'.
- Improved controls for entering a date and time.

13.55 Version 0.1.0

Released on 11 April 2009.

First usable version.

13.56 A note about version numbers

Timeline uses a three-component version numbering system (X.Y.Z).

Z is only incremented when critical bugs are corrected or translations are updated. The functionality of the program is the same for all X.Y versions.

Y is incremented every time a new feature or enhancement is added.

X is incremented when the new version is no longer compatible with previous versions or when the program undergoes some big change or significant milestone.

Symbols

`__init__()` (`timelinelib.canvas.TimelineCanvas` method),
38

N

`Navigate()` (`timelinelib.canvas.TimelineCanvas` method),
38

S

`SaveAsPng()` (`timelinelib.canvas.TimelineCanvas`
method), 39

`SaveAsSvg()` (`timelinelib.canvas.TimelineCanvas`
method), 39

`Scroll()` (`timelinelib.canvas.TimelineCanvas` method), 38

T

`timelinelib.canvas.TimelineCanvas` (built-in class), 38