ephyviewer Documentation

Release 1.0.0

Samuel Garcia

Jan 14, 2021

Contents

1 Overview **3**

2 Installation **5**

3 Examples **7**

3.1 Simple signal viewer 8
3.2 Signal viewer with data source 9
3.3 Signal viewer with marker on signal 11
3.4 timefreqquency viewer 13
3.5 Spikes viewer 15
3.6 Epoch viewer 16
3.7 mixed viewer 18

If you have a fat neural-electrophysiological dataset ? You want a real examination of raw signal and event before analysing ? Good shot. Ephyviewer is your friend.

ephysviwer is both a standalone application or module to script your visualisation.

CHAPTER 1

Overview

ephyviewer is a python module for building custuimized viewer for electrophysiological signals and related stuff (video/events/time frequency...)

It can be used as a stabndalone application just by lauching from console, then use the open menu (need neo>0.6):

```
ephyviewer
```

Launch it from console with filename (and optional format) (need neo>0.6):

```
ephyviewer File_axon_1.abf
ephyviewer File_axon_1.abf -f Axon
```

But th real goal is to design cutumized viewer for your needs with simple python scripts:

```
import ephyviewer
import numpy as np

app = ephyviewer.mkQApp()

#signals
sigs = np.random.rand(100000,16)
sample_rate = 1000.
t_start = 0.
view1 = ephyviewer.TraceViewer.from_numpy(sigs, sample_rate, t_start, 'Signals')

win = ephyviewer.MainViewer(debug=True, show_auto_scale=True)
win.add_view(view1)
win.show()

app.exec_()
```

Have a look at all examples to create from simple to sophisticated viewers.

CHAPTER 2

Installation

ephyviewer should work with python 2.7, 3.4, 3.5, 3.6.

ephyviewer should work either with Qt4 or Qt5 so theses depency are not mandatory for installation.

Mandatory Dependencies:

- numpy
- scipy
- matplotlib>=2.0
- pyqtgraph>=0.10.0
- PyQt4 or PyQt5

For video viewer:

- pyav

To get the last release:

```
pip install ephyviewer
```

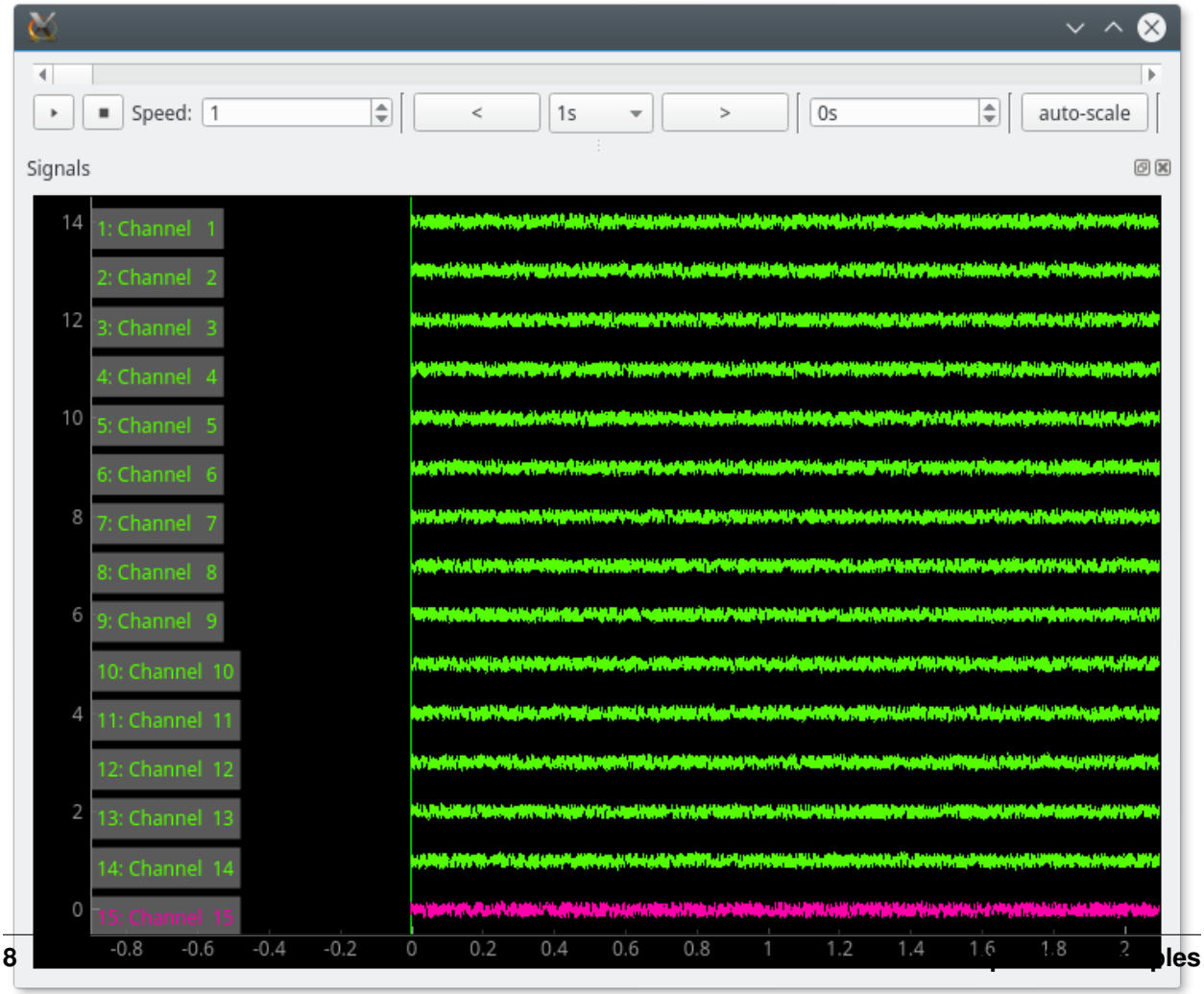
To get the last commit not already released:

```
pip install https://github.com/NeuralEnsemble/ephyviewer/archive/master.zip
```

CHAPTER 3

Examples

3.1 Simple signal viewer



trace_viewer.py

```
from ephyviewer import mkQApp, MainViewer, TraceViewer
import numpy as np

#you must first create a main Qt application (for event loop)
app = mkQApp()

#create fake 16 signals with 100000 at 10kHz
sigs = np.random.rand(100000,16)
sample_rate = 1000.
t_start = 0.

#Create the main window that can contain several viewers
win = MainViewer(debug=True, show_auto_scale=True)

#create a viewer for signal with TraceViewer
# TraceViewer normally accept a AnalogSignalSource but
# TraceViewer.from_numpy is facitilty function to bypass that
view1 = TraceViewer.from_numpy(sigs, sample_rate, t_start, 'Signals')

#Parameters can be set in script
view1.params['scale_mode'] = 'same_for_all'
view1.params['display_labels'] = True

#And also parameters for each channel
view1.by_channel_params['ch0', 'visible'] = False
view1.by_channel_params['ch15', 'color'] = '#FF00AA'

#This is needed when scale_mode='same_for_all'
#to recompute the gain
#this avoid to push auto_scale button
view1.auto_scale()

#put this veiwer in the main window
win.add_view(view1)

#show main window and run Qapp
win.show()


app.exec_()
```

3.2 Signal viewer with data source

trace_viewer_datasource.py

```
from ephyviewer import mkQApp, MainViewer, TraceViewer
from ephyviewer import InMemoryAnalogSignalSource
import ephyviewer
import numpy as np



#you must first create a main Qt application (for event loop)
app = mkQApp()
```

(continues on next page)

(continued from previous page)

```
#create fake 16 signals with 100000 at 10kHz
sigs = np.random.rand(100000,16)
sample_rate = 1000.
t_start = 0.

#Create the main window that can contain several viewers
win = MainViewer(debug=True, show_auto_scale=True)

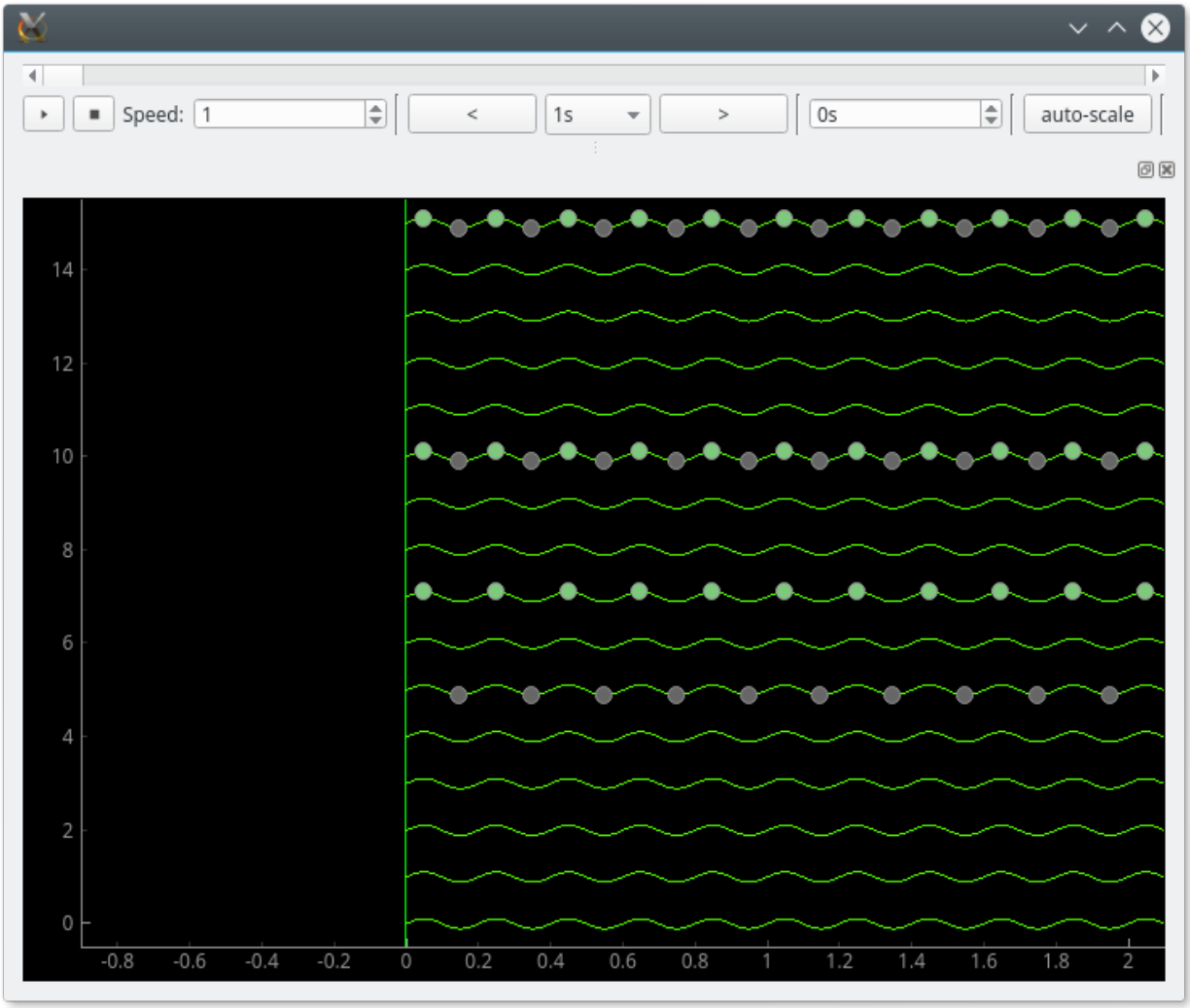
#Create a datasource for the viewer
# here we use InMemoryAnalogSignalSource but
# you can alose use your custum datasource by inheritance
source = InMemoryAnalogSignalSource(sigs, sample_rate, t_start)

#create a viewer for signal with TraceViewer
# TraceViewer normally accept a AnalogSignalSource but
# TraceViewer.from_numpy is facitilty function to bypass that
view1 = TraceViewer(source=source)


#put this veiwer in the main window
win.add_view(view1)

#show main window and run Qapp
win.show()
app.exec_()
```

3.3 Signal viewer with marker on signal



trace_viewer_with_marker.py

```
from ephyviewer import mkQApp, MainViewer, TraceViewer
from ephyviewer import AnalogSignalSourceWithScatter
import ephyviewer
import numpy as np

#you must first create a main Qt application (for event loop)
app = mkQApp()

#create 16 signals with 100000 at 10kHz
sigs = np.random.rand(100000,16)
sample_rate = 1000.
t_start = 0.


#create fake 16 signals with sinus
sample_rate = 1000.
```

(continues on next page)

(continued from previous page)

```
t_start = 0.
times = np.arange(1000000)/sample_rate
signals = np.sin(times*2*np.pi*5)[:, None]
signals = np.tile(signals, (1, 16))

#detect some crossing zeros
s0 = signals[:-2, 0]
s1 = signals[1:-1,0]
s2 = signals[2:,0]
peaks0,  = np.nonzero((s0<s1) & (s2<s1))
peaks1,  = np.nonzero((s0>s1) & (s2>s1))

#create 2 famillly scatters from theses 2 indexes
scatter_indexes = {0: peaks0, 1: peaks1}
#and asign them to some channels each
scatter_channels = {0: [0, 5, 8], 1: [0, 5, 10]}
source = AnalogSignalSourceWithScatter(signals, sample_rate, t_start, scatter_indexes,
↪ scatter_channels)


#Create the main window that can contain several viewers
win = MainViewer(debug=True, show_auto_scale=True)

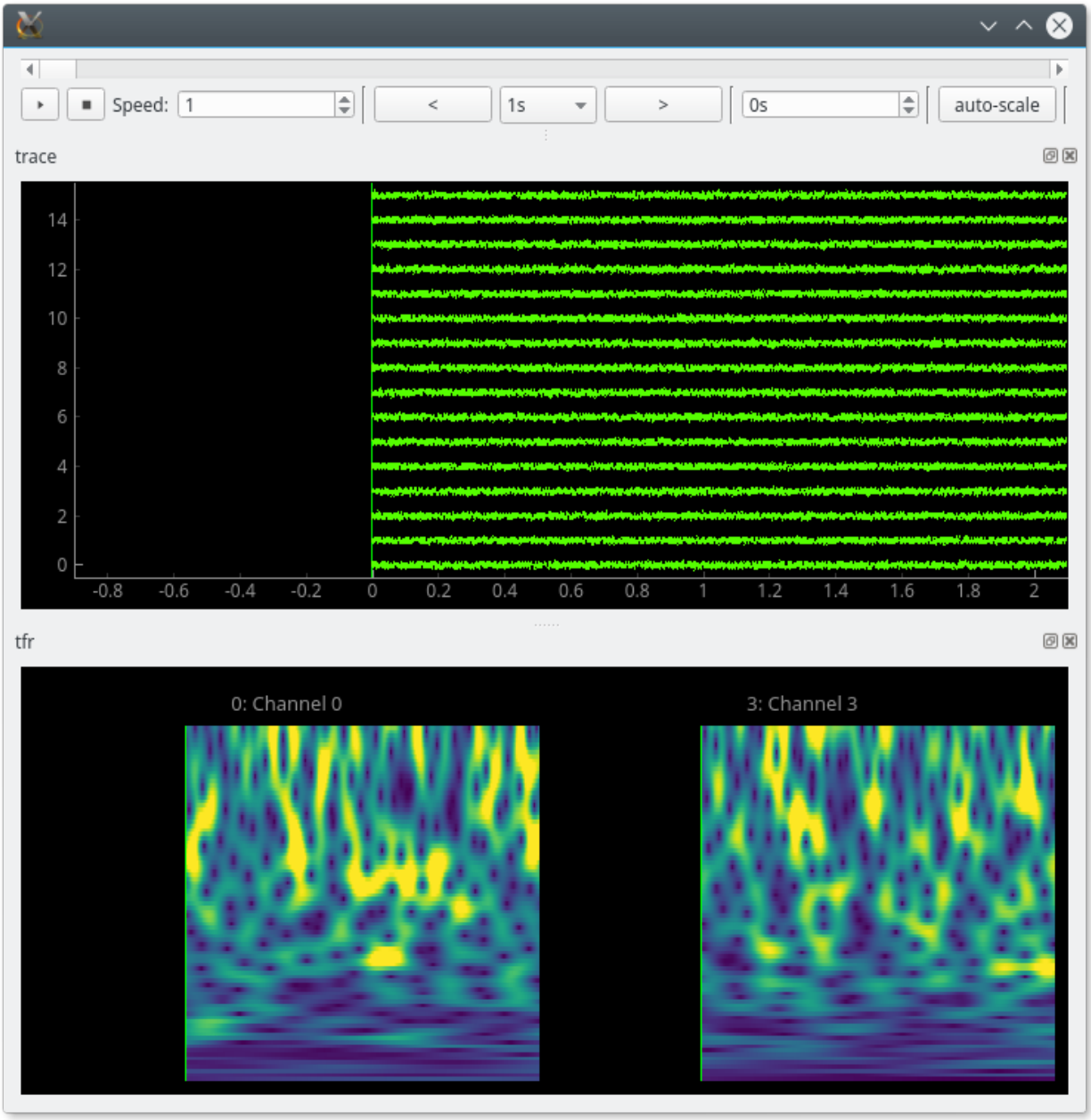
#create a viewer for signal with TraceViewer
#connected to the signal source
view1 = TraceViewer(source=source)

view1.params['scale_mode'] = 'same_for_all'
view1.auto_scale()

#put this veiwer in the main window
win.add_view(view1)

#show main window and run Qapp
win.show()
app.exec_()
```

3.4 timefreqquency viewer



timefreq_viewer.py

```
from ephyviewer import mkQApp, MainViewer, TraceViewer, TimeFreqViewer
from ephyviewer import InMemoryAnalogSignalSource
import ephyviewer
import numpy as np



#you must first create a main Qt application (for event loop)
```

(continues on next page)

(continued from previous page)

```
app = mkQApp()

#create fake 16 signals with 100000 at 10kHz
sigs = np.random.rand(100000,16)
sample_rate = 1000.
t_start = 0.

#Create the main window that can contain several viewers
win = MainViewer(debug=True, show_auto_scale=True)

#Create a datasource for the viewer
# here we use InMemoryAnalogSignalSource but
# you can alose use your custum datasource by inheritance
source = InMemoryAnalogSignalSource(sigs, sample_rate, t_start)

#create a viewer for signal with TraceViewer
view1 = TraceViewer(source=source, name='trace')
view1.params['scale_mode'] = 'same_for_all'
view1.auto_scale()

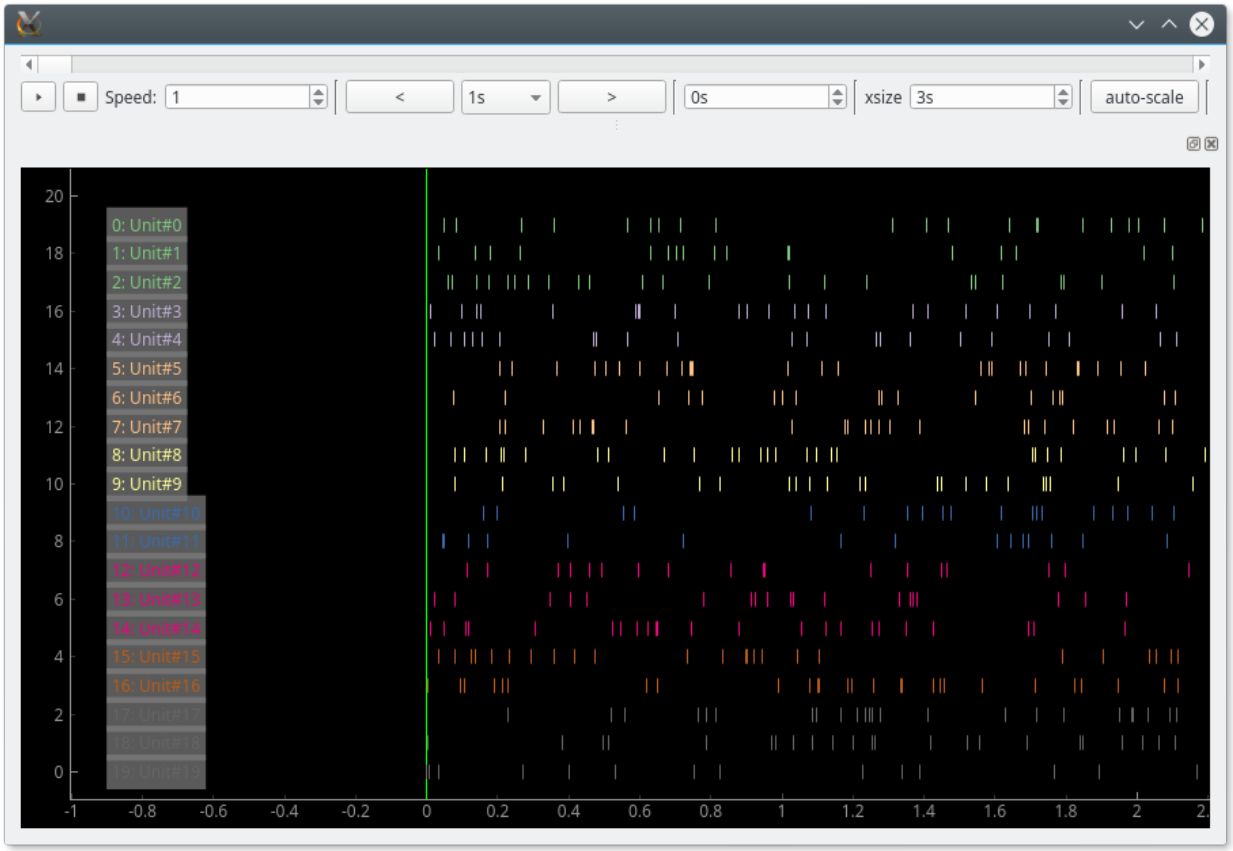
#create a time freq viewer conencted to the same source
view2 = TimeFreqViewer(source=source, name='tfr')

view2.params['show_axis'] = False
view2.params['timefreq', 'deltafreq'] = 1
view2.by_channel_params['ch3', 'visible'] = True


#add them to mainwindow
win.add_view(view1)
win.add_view(view2)


#show main window and run Qapp
win.show()
app.exec_()
```

3.5 Spikes viewer



spikes_viewer.py

```
from ephyviewer import mkQApp, MainViewer, SpikeTrainViewer
from ephyviewer import InMemorySpikeSource

import numpy as np

#you must first create a main Qt application (for event loop)
app = mkQApp()

#create fake 20 fake units with random firing
#put them in a da ta source
all_spikes =[]
for c in range(20):
    spike_times = np.random.rand(1000)*100.
    spike_times = np.sort(spike_times)
    all_spikes.append({ 'time':spike_times, 'name':'Unit#{}'.format(c) })
source = InMemorySpikeSource(all_spikes=all_spikes)

#Create the main window that can contain several viewers
win = MainViewer(debug=True, show_auto_scale=True)
```

(continues on next page)

(continued from previous page)

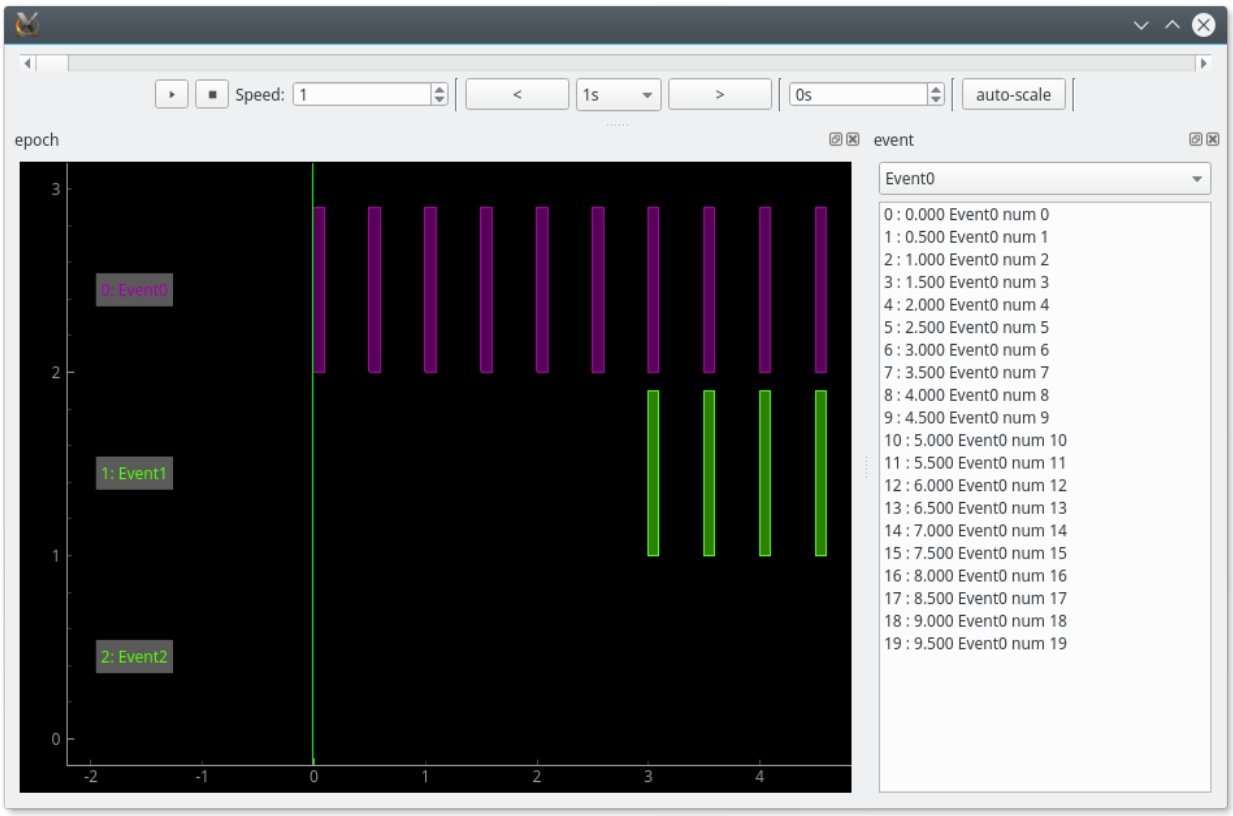
```
view1 = SpikeTrainViewer(source=source)


#put this veiwer in the main window
win.add_view(view1)

#show main window and run Qapp
win.show()


app.exec_()
```

3.6 Epoch viewer



event_epoch_viewer.py

```
from ephyviewer import mkQApp, MainViewer, EpochViewer,EventList
from ephyviewer import InMemoryEventSource, InMemoryEpochSource
import ephyviewer
import numpy as np



#Create one data source with 3 event channel
all_events = []
```

(continues on next page)

(continued from previous page)

```
for c in range(3):
    ev_times = np.arange(0, 10., .5) + c*3
    ev_labels = np.array(['Event{} num {}'.format(c, i) for i in range(ev_times.
↪size)], dtype='U')
    all_events.append({ 'time':ev_times, 'label':ev_labels, 'name':'Event{}'.
↪format(c) })
source_ev = InMemoryEventSource(all_events=all_events)


#Create one data source with 2 epoch channel
all_epochs = []
for c in range(3):
    ep_times = np.arange(0, 10., .5) + c*3
    ep_durations = np.ones(ep_times.shape) * .1
    ep_labels = np.array(['Event{} num {}'.format(c, i) for i in range(ep_times.
↪size)], dtype='U')
    all_epochs.append({ 'time':ep_times, 'duration':ep_durations, 'label':ep_labels,
↪'name':'Event{}'.format(c) })
source_ep = ephyviewer.InMemoryEpochSource(all_epochs=all_epochs)




#you must first create a main Qt application (for event loop)
app = mkQApp()


#Create the main window that can contain several viewers
win = MainViewer(debug=True, show_auto_scale=True)

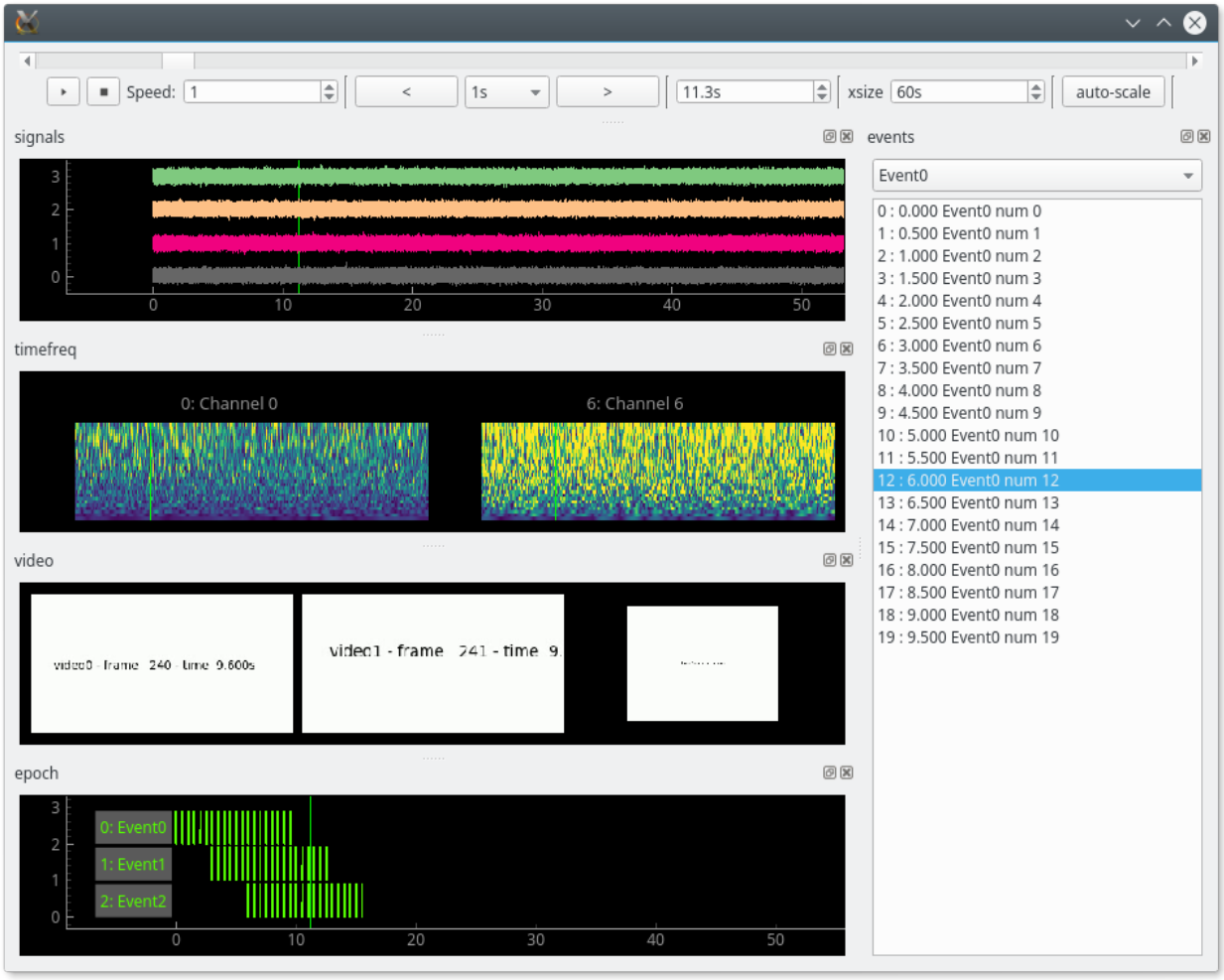

view1 = EpochViewer(source=source_ep, name='epoch')
view1.by_channel_params['ch0', 'color'] = '#AA00AA'
view1.params['xsize'] = 6.5

view2 = EventList(source=source_ev, name='event')


#add them to mainwindow
win.add_view(view1)
win.add_view(view2, location='bottom',  orientation='horizontal')


#show main window and run Qapp
win.show()
app.exec_()
```

3.7 mixed viewer



mixed_viewer.py

```
import ephyviewer

#for this example we use fake source construct by theses function
from  ephyviewer.tests.testing_tools import make_fake_video_source
from  ephyviewer.tests.testing_tools import make_fake_signals
from  ephyviewer.tests.testing_tools import make_fake_event_source
from  ephyviewer.tests.testing_tools import make_fake_epoch_source

sig_source = make_fake_signals()
event_source = make_fake_event_source()
epoch_source = make_fake_epoch_source()
video_source = make_fake_video_source()



app = ephyviewer.mkQApp()
view1 = ephyviewer.TraceViewer(source=sig_source, name='signals')
view2 = ephyviewer.VideoViewer(source=video_source, name='video')
```

(continues on next page)

(continued from previous page)

```
view3 = ephyviewer.EventList(source=event_source, name='events')
view4 = ephyviewer.EpochViewer(source=epoch_source, name='epoch')
view5 = ephyviewer.TimeFreqViewer(source=sig_source, name='timefreq')


win = ephyviewer.MainViewer(debug=True, settings_name='test1', show_global_xsize=True,
↪ show_auto_scale=True)

win.add_view(view1)
win.add_view(view5, split_with='signals')
win.add_view(view2)
win.add_view(view4)
win.add_view(view3, location='bottom',  orientation='horizontal')

win.show()

win.auto_scale()

app.exec_()
```