mpl$_{a}nimationmanagerDocumentation$

Release latest

May 17, 2017

Contents

CHAPTER 1

NOTE: Documentation is curently in development!!!

Matplotlib animation manager (GUI) 1.0a1

It is a small convenient tool which allows to setup and save the gif/mp4-animations using the PyQt based GUI built on top of the matplotlib animation module. Program can deal with both 2D and 3D animation. For 3D axes manager can add additional rotation of the view point resulting in both object modification and rotation animation. Also animation manager can be easily integrated in your larger PyQt project as a dialog. For more details see the Quitckstart section.

- Git-hub repo: https://github.com/luchko/mpl_animationmanager
- Documentation: https://mpl-animationmanager.readthedocs.io
- Free software: MIT license

Tool is compatible with Python 2.7 or Python 3.3+ and PyQt4 4.6+ or PyQt5 5.2+.

Features

- `mpl_animationmanager` library contains two classes `AnimationManager` and `QDialogAnimManager` with the same input arguments (*see API*).
- `QDialogAnimManager` is inherited from the PyQt `QDialog`. Using this class you can easily integrate animation manager as a QDialog into your larger PyQt application.
- `AnimationManager` is a small class build on top of the `QDialogAnimManager` and uses the input arguments to initialize the `QDialogAnimManager` object and run a PyQt application using `run()` function.
- After passing the required arguments to the manager, user can setup animation properties such as: dpi, fps (frames per second), modification period.

- For 3D animation user can also setup the rotation period, elevation and initilal azimut angles. The resulting duration of the animation equals the least common multiple of modification and rotaion periods if both are provided.
- Animation can be saved in gif or mp4 format by picking one of the preinstalled movie writers used by matplotlib (imagemagick, ffmpeg etc.).

Dependencies

- **Python** 2.7 or 3.3+
- **PyQt4** 4.6+ or **PyQt5** 5.2+ : PyQt4 is recommended.
- **Matplotlib**

Important note: *Most dependencies listed above are installed automatically, however in some cases you might need to istall them separately (see next section).*

Install PyQt4 or PyQt5

- in case you use conda type: `$ conda install pyqt=4` (or 5)
- otherwise follow the links PyQt4 or PyQt5.

Installation and usage

This section explains how to install and use the latest stable release of the Matplotlib animation manager in one of the cross-platform ways listed bellow. If you prefer just to have a taste of the tool you may *jump to the example section*.

Installation using `conda` scientific package manager (recommended way)

PROJECT IS NOT RELEASED YET

Type in your command prompt:

`$ pip install conda` (if `conda` is not installed yet)

```
$ conda install mpl_animationmanager
```

Note: *All dependencies are installed by* `conda` *automatically.*

Installation using `pip` package manager from PyPI

PROJECT IS NOT RELEASED YET

Type in your command prompt:

```
$ pip install mpl_animationmanager
```

Important note: *This also installs all dependencies except PyQt4 or PyQt5. Those have to be installed separately after installing Python.*

Running from source

It is possible to use animation manager without installing it.

1. Make sure that PyQt4 or PyQt5 package is installed.
2. Download a source of the last stable package version.
3. Copy the `./mpl_animationmanager/` package source folder into the root directiory of your script.
4. Now you can import `mpl_animationmanager` module to your sript the same way as it would have been installed.

You may want to do this for fixing bugs, adding new features, integrating the widget into your own PyQt project, learning how the tool works or just getting a taste of it.

API

Both `AnimationManager` and `QDialogAnimManager` classes take the same input arguments

```
class AnimationManager(object):

    def __init__(self, ax, fAnim=None, fargs=None, numFramesModif=None, *args,␣
↪**kwargs):
        '''
        Parameters
        ----------
        ax : 2D or 3D matplotlib axes object binded to the figure
            provides control over animated figure
        fAnim : function
            fAnim(i, ax, fargs) - modifies the "ax" at each "i" step
        fargs : any
            arguments used by the "fAnim" function during the "ax" modification
        numFramesModif : int
            number of modification frames
        '''
```

Small example

Code below produces the same animation as one shown at the main demo above.

```
"""script runs a small example of the animation manager usage"""

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from mpl_animationmanager import AnimationManager

def fAnim(j, ax, lineColl):
    '''define the modification animation function'''
    ax.collections = [] # clean axes
    ax.add_collection3d(lineColl[j]) # add new artist

# create figure
fig = plt.figure('3D wireframe example')
ax = fig.gca(projection='3d')
ax.set_axis_off()
```

```
# generate modification frames (passed as fargs)
numFrames = 300
X, Y, Z = axes3d.get_test_data(0.05)
for j in range(numFrames):
    ax.plot_wireframe(X, Y, Z*np.cos(2*np.pi/numFrames*j), rstride=5, cstride=5)
fargs = ax.collections
ax.collections = []

# pass figure to the animation manager
mng = AnimationManager(ax, fAnim, fargs, numFrames)
mng.run()
```

More examples

More examples with gif demo are included in the `./mpl_animationmanager/examples/` folder (link). You might run them as a Python script after instalation `mpl_animationmanager` package.

Second option is to run the python script `run_examples.py` located in the root project directory after downloading the source code. In this script you can also pick the examples you would like to run.

Following gif-animation examples were created with the Matplotlib animation manager:

You might also have a look at the larger PyQt project Lattice graph designer where `QDialogAnimManager` is integrated for exporting the rotating visualization of 3D model.

Running `unittest`

After making any changes in the source code you can run `unitittest` to make sure that nothing is broken by typing in your command prompt:

```
$ python setup.py test
```

Contacts

About the feature extension or bugs report you can create the issue or feature request or feel free to contact me directly by e-mail:

Ivan Luchko - luchko.ivan@gmail.com

Matplotlib animation manager (GUI) 1.0a1

It is a small convenient tool which allows to setup and save the gif/mp4-animations using the PyQt based GUI built on top of the matplotlib animation module. Program can deal with both 2D and 3D animation. For 3D axes manager can add additional rotation of the view point resulting in both object modification and rotation animation. Also animation manager can be easily integrated in your larger PyQt project as a dialog. For more details see the Quitckstart section.

- Git-hub repo: https://github.com/luchko/mpl_animationmanager
- Documentation: https://mpl-animationmanager.readthedocs.io
- Free software: MIT license

Tool is compatible with Python 2.7 or Python 3.3+ and PyQt4 4.6+ or PyQt5 5.2+.

CHAPTER 2

Features

- `mpl_animationmanager` library contains two classes `AnimationManager` and `QDialogAnimManager` with the same input arguments (*see API*).
- `QDialogAnimManager` is inherited from the PyQt `QDialog`. Using this class you can easily integrate animation manager as a QDialog into your larger PyQt application.
- `AnimationManager` is a small class build on top of the `QDialogAnimManager` and uses the input arguments to initialize the `QDialogAnimManager` object and run a PyQt application using `run()` function.
- After passing the required arguments to the manager, user can setup animation properties such as: dpi, fps (frames per second), modification period.
- For 3D animation user can also setup the rotation period, elevation and initilal azimut angles. The resulting duration of the animation equals the least common multiple of modification and rotaion periods if both are provided.
- Animation can be saved in gif or mp4 format by picking one of the preinstalled movie writers used by matplotlib (imagemagick, ffmpeg etc.).

CHAPTER 3

Dependencies

- **Python** 2.7 or 3.3+
- **PyQt4** 4.6+ or **PyQt5** 5.2+ : PyQt4 is recommended.
- **Matplotlib**

Important note: *Most dependencies listed above are installed automatically, however in some cases you might need to istall them separately (see next section).*

Install PyQt4 or PyQt5

- in case you use conda type: `$ conda install pyqt=4` (or 5)
- otherwise follow the links PyQt4 or PyQt5.

CHAPTER 4

Installation and usage

This section explains how to install and use the latest stable release of the Matplotlib animation manager in one of the cross-platform ways listed bellow. If you prefer just to have a taste of the tool you may *jump to the example section*.

Installation using `conda` scientific package manager (recommended way)

PROJECT IS NOT RELEASED YET

Type in your command prompt:

`$ pip install conda` (if `conda` is not installed yet)

```
$ conda install mpl_animationmanager
```

Note: *All dependencies are installed by* `conda` *automatically.*

Installation using `pip` package manager from PyPI

PROJECT IS NOT RELEASED YET

Type in your command prompt:

```
$ pip install mpl_animationmanager
```

Important note: *This also installs all dependencies except PyQt4 or PyQt5. Those have to be installed separately after installing Python.*

Running from source

It is possible to use animation manager without installing it.

1. Make sure that PyQt4 or PyQt5 package is installed.
2. Download a source of the last stable package version.
3. Copy the `./mpl_animationmanager/` package source folder into the root directiory of your script.
4. Now you can import `mpl_animationmanager` module to your sript the same way as it would have been installed.

You may want to do this for fixing bugs, adding new features, integrating the widget into your own PyQt project, learning how the tool works or just getting a taste of it.

API

Both `AnimationManager` and `QDialogAnimManager` classes take the same input arguments

```
class AnimationManager(object):

    def __init__(self, ax, fAnim=None, fargs=None, numFramesModif=None, *args,␣
↪**kwargs):
        '''
        Parameters
        ----------
        ax : 2D or 3D matplotlib axes object binded to the figure
            provides control over animated figure
        fAnim : function
            fAnim(i, ax, fargs) - modifies the "ax" at each "i" step
        fargs : any
            arguments used by the "fAnim" function during the "ax" modification
        numFramesModif : int
            number of modification frames
        '''
```

Small example

Code below produces the same animation as one shown at the main demo above.

```
"""script runs a small example of the animation manager usage"""

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from mpl_animationmanager import AnimationManager

def fAnim(j, ax, lineColl):
    '''define the modification animation function'''
    ax.collections = [] # clean axes
    ax.add_collection3d(lineColl[j]) # add new artist

# create figure
fig = plt.figure('3D wireframe example')
ax = fig.gca(projection='3d')
ax.set_axis_off()

# generate modification frames (passed as fargs)
```

```
numFrames = 300
X, Y, Z = axes3d.get_test_data(0.05)
for j in range(numFrames):
    ax.plot_wireframe(X, Y, Z*np.cos(2*np.pi/numFrames*j), rstride=5, cstride=5)
fargs = ax.collections
ax.collections = []

# pass figure to the animation manager
mng = AnimationManager(ax, fAnim, fargs, numFrames)
mng.run()
```

More examples

More examples with gif demo are included in the `./mpl_animationmanager/examples/` folder (link). You might run them as a Python script after instalation `mpl_animationmanager` package.

Second option is to run the python script `run_examples.py` located in the root project directory after downloading the source code. In this script you can also pick the examples you would like to run.

Following gif-animation examples were created with the Matplotlib animation manager:

You might also have a look at the larger PyQt project Lattice graph designer where `QDialogAnimManager` is integrated for exporting the rotating visualization of 3D model.

Running `unittest`

After making any changes in the source code you can run `unitittest` to make sure that nothing is broken by typing in your command prompt:

```
$ python setup.py test
```

CHAPTER 5

Contacts

About the feature extension or bugs report you can create the issue or feature request or feel free to contact me directly by e-mail:

Ivan Luchko - luchko.ivan@gmail.com