Python MSS Documentation

Release latest

Tiger-222

October 06, 2016

Contents

An ultra fast cross-platform multiple screenshots module in pure python using ctypes.

Very basic, it will grab one screen shot by monitor or a screen shot of all monitors and save it to a PNG file, Python 2.6/3.5 compatible & **PEP 8** compliant. It could be easily embedded into games and other softwares which require fast and plateforme optimized methods to grab screenshots.

MSS stands for Multiple ScreenShots.

It's under MIT license.

Contents:

CHAPTER 1

Installation

1.1 Recommended way

Quite simple:

```
pip install mss
```

1.2 From sources

Alternatively, you can get a copy of the module from GitHub:

```
git clone https://github.com/BoboTiG/python-mss.git
cd python-mss
```

And then:

```
sudo python setup.py install
```

CHAPTER 2

Usage

2.1 Instance the good class

So MSS can be used as simply as:

```
from mss import mss

with mss() as screenshotter:
    # ...
```

Or import the good one (choose one):

```
# MacOS X
from mss.darwin import MSS

# GNU/Linux
from mss.linux import MSS

# Microsoft Windows
from mss.windows import MSS

with MSS() as screenshotter:
    # ...
```

Of course, you can use it the old way:

```
from mss import mss
# or from mss.linux import MSS as mss

screenshotter = mss()
```

2.2 Examples

One screenshot per monitor:

```
for filename in screenshotter.save():
    print(filename)
```

Screenshot of the monitor 1:

```
print(next(screenshotter.save(mon=1)))
```

Screenshot of the monitor 1, with callback:

```
def on_exists(fname):
    ''' Callback example when we try to overwrite an existing
        screenshot.
    '''

    from os import rename
    from os.path import isfile

    if isfile(fname):
        newfile = fname + '.old'
        print('{0} -> {1}'.format(fname, newfile))
        rename(fname, newfile)
    return True

print(next(screenshotter.save(mon=1, callback=on_exists)))
```

A screenshot to grab them all:

```
print(next(screenshotter.save(mon=-1, output='fullscreen.png')))
```

2.3 Into the Python's console

```
>>> from mss import mss
>>> sct = mss(display=':0')

# Retrieve monitors informations
>>> displays = sct.enum_display_monitors()
>>> displays
[{'width': 1920, 'top': 0, 'height': 1080, 'left': 0}, {'width': 1920, 'top': 0, 'height': 1080, 'le
# You can access monitors list via `monitors`:
>>> sct.monitors
[{'width': 1920, 'top': 0, 'height': 1080, 'left': 0}, {'width': 1920, 'top': 0, 'height': 1080, 'le

# Retrieve pixels from the first monitor
>>> pixels = sct.get_pixels(displays[1])
>>> pixels
<ctypes.c_char_Array_6220800 object at 0x7fe82e9007a0>
# You can access pixels data via `image`:
>>> sct.image
<ctypes.c_char_Array_6220800 object at 0x7fe82e9007a0>

# Save pixels to a PNG file: option 1
>>> files = sct.save(mon=1)
>>> next(files)
'monitor-1.png'
>>> next(files)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
StopIteration

# Save pixels to a PNG file: option 2
>>> mon = displays[1]
```

```
>>> sct.to_png(data=pixels, width=mon['width'], height=mon['height'], output='monitor-1.png')
```

CHAPTER 3

Examples

3.1 GNU/Linux

Usage example with a specific display:

```
from mss.linux import MSS


display = ':0.0'
print('Screenshot of display "{0}"'.format(display))
output = 'monitor{0}-%d.png'.format(display)

with MSS(display=display) as screenshotter:
    for filename in screenshotter.save(output=output):
        print(filename)
```

3.2 Using PIL

You can use the Python Image Library (aka Pillow) to do whatever you want with raw pixels. This is an example using frombytes():

```
from mss import mss
from PIL import Image


with mss() as screenshotter:
    # We retrieve monitors informations:
    monitors = screenshotter.enum_display_monitors()

    # Get rid of the first, as it represents the "All in One" monitor:
    for num, monitor in enumerate(monitors[1:], 1):
        # Get raw pixels from the screen.
        # This method will store screen size into `width` and `height`
        # and raw pixels into `image`.
        screenshotter.get_pixels(monitor)

        # Create an Image:
        img = Image.frombytes('RGB',
                              (screenshotter.width, screenshotter.height),
                              screenshotter.image)
```

```
        # And save it!
        img.save('monitor-{0}.jpg'.format(num))
```

CHAPTER 4

Support

Python	GNU/Linux	MacOS X	Windows
3.6	yes	yes	yes
3.5	Yes	Yes	Yes
3.4	yes	yes	yes
3.3	yes	???	yes
3.2	yes	???	yes
3.1	yes	???	yes
3.0	yes	???	yes
2.7	Yes	Yes	Yes
2.6	yes	yes	yes

Feel free to try MSS on a system we had not tested, and let report us by creating an issue.

CHAPTER 5

MSS API

5.1 Classes

`MSSBase` is the parent's class for every OS implementation.

class `mss.linux.`**MSS**

__init__(*display=None*)

Param str display: display to use.

GNU/Linux initializations.

5.2 Methods

class `mss.base.`**MSSBase**

bgra_to_rgb(*raw*) → bytes

Parameters **raw** (*mixed*) – raw data containing BGRA values.

It converts pixels values from BGRA to RGB. This is the method called to populate *image* into *get_pixels*.

enum_display_monitors(*force=False) -> list(dict*)

Parameters **force** (*bool*) – force rescan of monitors informations.

Raises **NotImplementedError** – Subclasses need to implement this.

Get positions of one or more monitors. If the monitor has rotation, you have to deal with it inside this method.

This method has to fill *monitors* with all informations and use it as a cache:

- `monitors[0]` is a dict of all monitors together;
- `monitors[N]` is a dict of the monitor N (with N > 0).

Each monitor is a dict with:

```
{
    'left':   the x-coordinate of the upper-left corner,
    'top':    the y-coordinate of the upper-left corner,
    'width':  the width,
    'height': the height
}
```

get_pixels(*monitor*) → bytes

Parameters **monitor** (*dict*) – monitor's informations generated by *enum_display_monitors()*.

Raises **NotImplementedError** – Subclasses need to implement this.

Retrieve screen pixels for a given monitor. This method has to define *width* and *height*. It stocks pixels data into *image* (RGB) and returns it.

save(*mon=0*, *output='monitor-%d.png'*, *callback=lambda *x: True*) → generator

Parameters

- **mon** (*int*) – the monitor's number.
- **output** (*str*) – the output's file name. `%d`, if present, will be replaced by the monitor number.
- **callback** (*callable*) – callback called before saving the screenshot to a file. Takes the `output` argument as parameter.

Grab a screenshot and save it to a file. This is a generator which returns created files.

to_png(*data*, *output*) → None

Parameters

- **data** (*bytes*) – raw pixels (RGBRGB...RGB) fom *get_pixels()*.
- **output** (*str*) – output's file name.

Raises **ScreenshotError** – On error when writing `data` to `output`.

Dump data to the image file. Pure Python PNG implementation.

5.3 Attributes

class `mss.base.`**MSSBase**

image

Getter Raw pixels of a monitor.

Setter See *get_pixels*.

Type bytes

monitors

Getter The list of all monitors.

Setter See *enum_display_monitors()*.

Type list(dict)

width

Getter Width of a monitor.

Setter See *get_pixels()*.

Type int

height

Getter Height of a monitor.

Setter See *get_pixels()*.

Type int

5.4 Exception

exception mss.exception.**ScreenshotError**
Base class for MSS exceptions.

5.5 Factory

mss.**mss**() → MSSBase
Factory function to instance the appropriate MSS class.

CHAPTER 6

Who uses it?

This is a non exhaustive list where MSS is integrated or has inspired:

- Pombo;
- Automation Framework, a Batmans utility;
- Flexx Python UI toolkit;
- NativeShot (Mozilla Firefox module);
- and you pherhaps?

CHAPTER 7

Indices and tables

- genindex
- search

m

Symbols

B

E

G

H

I

M

P

S

T

W