aggdraw Documentation

Release 1.3.8

aggdraw Developers

Dec 03, 2025

CONTENTS

The AggDraw library provides a python interface on top of the AGG library. The library was originally developed by Fredrik Lundh (effbot), but has since been picked up by various developers. It is currently maintained by the PyTroll developmer group. The official repository can be found on GitHub:

https://github.com/pytroll/aggdraw

The original documentation by effbot is still available but may be out of date with the current version of the library. Original examples will be migrated as time is available (pull requests welcome).

CHAPTER

ONE

INSTALLATION

Aggdraw is available on Linux, OSX, and Windows. It can be installed from PyPI with pip:

```
pip install aggdraw
```

Or from conda with the conda-forge channel:

```
conda install -c conda-forge aggdraw
```

CHAPTER

TWO

FREE-THREADING SUPPORT

Basic free-threading compatibility has been enabled starting with the Python 3.14 wheels of aggdraw 1.4.0. However, only the minimum steps have been taken to let Python 3.14's free-threaded build know that aggdraw could be used without the GIL. No additional locking or redesign of the library has been done.

Aggdraw itself should have no global state, but each individual object is free to have internal state. It is up to the user to limit interactions for an aggdraw object to a single thread or to protect against concurrent access using locks. Even with this in mind, free-threading support in aggdraw is extremely unstable and experimental. If you have a use case for aggdraw in a free-threading environment please file an issue on GitHub to describe your use case and how it is going.

CHAPTER

THREE

API

Python interface to the Anti-Grain Graphics Drawing library

The aggdraw module implements the basic WCK 2D Drawing Interface on top of the AGG library. This library supports anti-aliasing and alpha compositing, but is otherwise fully compatible with the WCK renderer.

Examples

```
>>> # draw cross on top of PIL image
>>> d = aggdraw.Draw(im)
>>> p = aggdraw.Pen("black", 0.5)
>>> d.line((0, 0, 500, 500), p)
>>> d.line((0, 500, 500, 0), p)
>>> d.flush()
```

```
>>> # draw cross on internal image memory
>>> d = aggdraw.Draw("RGB", (320, 200), "white")
>>> p = aggdraw.Pen("black", 0.5)
>>> d.line((0, 0, 500, 500), p)
>>> d.line((0, 500, 500, 0), p)
>>> s = d.tobytes()
```

aggdraw.**Brush**()

Creates a brush object.

Parameters

- **color** (*tuple or str or int*) – Brush color. This can be a color tuple (R, G, B) or (R, G, B, A), a CSS-style color name, or a color integer (0xaarrggbb).
- **opacity** (*int, optional*) – Brush opacity. Default 255.

aggdraw.**Draw**()

Creates a drawing interface object.

Parameters

- **image_or_mode** (*PIL.Image.Image or str*) – A PIL Image or a mode string. The following modes are supported: “L”, “RGB”, “RGBA”, “BGR”, “BGRA”.
- **size** (*tuple*) – If a mode string was given, this argument gives the image size as a 2-element tuple.
- **color** – An optional background color specifier. If a mode string was given, this is used to initialize the image memory. If omitted, it defaults to white with full alpha.

Examples

```
>>> d = aggdraw.Draw(im)
>>> d = aggdraw.Draw("RGB", (800, 600), "white")
```

aggdraw.**Font**()

Create a font object from a truetype font file for use with *text* and *textsize*.

Parameters

- **color** (*tuple or str or int*) – Font color. This can be a color tuple (R, G, B) or (R, G, B, A), a CSS-style color name, or a color integer (0xaarrggbb).
- **file** (*str*) – Font source file.
- **size** (*int, optional*) – Font size in pixels. Default 12.
- **opacity** (*int, optional*) – Font opacity. Default 255.

aggdraw.**Path**()

Path factory (experimental).

aggdraw.**Pen**()

Creates a Pen object.

Parameters

- **color** (*tuple or str or int*) – Pen color. This can be a color tuple (R, G, B) or (R, G, B, A), a CSS-style color name, or a color integer (0xaarrggbb).
- **width** (*int, optional*) – Pen width. Default 1.
- **opacity** (*int, optional*) – Pen opacity. Default 255.

aggdraw.**Symbol**()

Create a Symbol object for use with `Draw.symbol()`.

Parameters

path (*str*) – An SVG-style path descriptor. The following operators are supported: M (move), L (line), H (horizontal line), V (vertical line), C (cubic bezier), S (smooth cubic bezier), Q (quadratic bezier), T (smooth quadratic bezier), and Z (close path). Use lower-case operators for relative coordinates, upper-case for absolute coordinates.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX