aggdraw Documentation

Release 1.3.8

aggdraw Developers

Apr 27, 2026

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 developer group. The official repository can be found on GitHub:

https://github.com/pytroll/aggdraw

The original documentation by effbot is still available but is out of date with the current version of the library.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, macOS, 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

class aggdraw.**Brush**(*color*, *opacity=255*)

Bases: object

Creates a brush object.

The brush color can be an RGB tuple (e.g. *(255, 255, 255)*), a CSS-style color name, or a color integer (0xAARRGGBB).

Parameters

- **color** – The brush color.
- **opacity** (*int, optional*) – The opacity of the brush (from 0 to 255). Defaults to a solid brush.

class aggdraw.**Draw**(*image_or_mode*, *size=None*, *color='white'*)

Bases: object

Creates a drawing interface object.

The constructor can either take a PIL Image object, or mode and size specifiers.

Examples::

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

Parameters

- **image_or_mode** – A PIL image or a mode string. The following modes are supported: "L", "RGB", "RGBA", "BGR", "BGRA".
- **size** (*tuple, optional*) – The size of the image (width, height).
- **color** (*optional*) – An optional background color. If omitted, defaults to white with full alpha.

arc(*xy*, *start*, *end*, *pen=None*)

Draws an arc.

Parameters

- **xy** – A 4-element Python sequence (x, y, x, y) with the upper-left corner given first.
- **start** (*float*) – The start angle of the arc.
- **end** (*float*) – The end angle of the arc.
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing the arc.

chord(*xy*, *start*, *end*, *pen=None*, *brush=None*)

Draws a chord.

If a brush is given, it is used to fill the chord. If a pen is given, it is used to draw an outline around the chord. Either one (or both) can be left out.

Parameters

- **xy** – A 4-element Python sequence (x, y, x, y) with the upper-left corner given first.
- **start** (*float*) – The start angle of the chord.
- **end** (*float*) – The end angle of the chord.
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing an outline around the chord.
- **brush** (*aggdraw.Brush*, optional) – A brush to use for filling the chord.

ellipse(*xy*, *pen=None*, *brush=None*)

Draws an ellipse.

If a brush is given, it is used to fill the ellipse. If a pen is given, it is used to draw an outline around the ellipse. Either one (or both) can be left out.

To draw a circle, make sure the coordinates form a square.

Parameters

- **xy** – A bounding rectangle as a 4-element Python sequence (x, y, x, y), with the upper-left corner given first.
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing an outline around the ellipse.
- **brush** (*aggdraw.Brush*, optional) – A brush to use for filling the ellipse.

flush()

Updates the associated image.

If the drawing area is attached to a PIL Image object, this method must be called to make sure that the image updated.

frombytes(*data*)

Copies data from a bytes object to the drawing area.

Parameters

data (*bytes*) – Packed image data compatible with PIL's tobytes method.

line(*xy*, *pen=None*)

Draws a line.

If the sequence contains multiple x/y pairs, multiple connected lines will be drawn.

Parameters

- **xy** – A Python sequence in the format (x, y, x, y, ...)
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing the line.

property mode

The mode of the draw surface (e.g. "RGBA").

Type

str

path(*xy*, *path*, *pen=None*, *brush=None*)

Draws a path at the given positions.

If a brush is given, it is used to fill the path. If a pen is given, it is used to draw an outline around the path. Either one (or both) can be left out.

Parameters

- **xy** – A Python sequence in the format (x, y, x, y, ...)
- **path** (*aggdraw.Path*) – The Path object to draw.
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing an outline around the path.
- **brush** (*aggdraw.Brush*, optional) – A brush to use for filling the path.

pieslice(*xy*, *start*, *end*, *pen=None*, *brush=None*)

Draws a pie slice.

If a brush is given, it is used to fill the pie slice. If a pen is given, it is used to draw an outline around the pie slice. Either one (or both) can be left out.

Parameters

- **xy** – A 4-element Python sequence (x, y, x, y) with the upper-left corner given first.
- **start** (*float*) – The start angle of the pie slice.
- **end** (*float*) – The end angle of the pie slice.
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing an outline around the pie slice.
- **brush** (*aggdraw.Brush*, optional) – A brush to use for filling the pie slice.

polygon(*xy*, *pen=None*, *brush=None*)

Draws a polygon.

If a brush is given, it is used to fill the polygon. If a pen is given, it is used to draw an outline around the polygon. Either one (or both) can be left out.

Parameters

- **xy** – A Python sequence (x, y, x, y, ...).
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing an outline around the polygon.
- **brush** (*aggdraw.Brush*, optional) – A brush to use for filling the polygon.

rectangle(*xy*, *pen=None*, *brush=None*)

Draws a rectangle.

If a brush is given, it is used to fill the rectangle. If a pen is given, it is used to draw an outline around the rectangle. Either one (or both) can be left out.

Parameters

- **xy** – A 4-element Python sequence (x, y, x, y), with the upper left corner given first.
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing an outline around the rectangle.
- **brush** (*aggdraw.Brush*, optional) – A brush to use for filling the rectangle.

rounded_rectangle(*xy*, *radius*, *pen=None*, *brush=None*)

Draws a rounded rectangle.

If a brush is given, it is used to fill the rectangle. If a pen is given, it is used to draw an outline around the rectangle. Either one (or both) can be left out.

Parameters

- **xy** – A 4-element Python sequence (x, y, x, y), with the upper left corner given first.
- **radius** (*float*) – The corner radius.
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing an outline around the rectangle.
- **brush** (*aggdraw.Brush*, optional) – A brush to use for filling the rectangle.

setantialias(*flag*)

Controls anti-aliasing.

Parameters
flag (*bool*) – True to enable anti-aliasing, False to disable it.

settransform(*transform=None*)

Replaces the current drawing transform.

The transform must either be a (dx, dy) translation tuple, or a PIL-style (a, b, c, d, e, f) affine transform tuple. If the transform is omitted, it is reset.

Example::
draw.settransform((dx, dy))

Parameters
transform (*tuple, optional*) – The new transform, or None to reset.

property size

The size in pixels of the draw surface (width, height).

Type
tuple

symbol(*xy*, *symbol*, *pen=None*, *brush=None*)

Draws a symbol at the given positions.

If a brush is given, it is used to fill the symbol. If a pen is given, it is used to draw an outline around the symbol. Either one (or both) can be left out.

Parameters

- **xy** – A Python sequence in the format (x, y, x, y, ...)
- **symbol** (*aggdraw.Symbol*) – The Symbol object to draw.
- **pen** (*aggdraw.Pen*, optional) – A pen to use for drawing an outline around the symbol.
- **brush** (*aggdraw.Brush*, optional) – A brush to use for filling the symbol.

text(*xy*, *text*, *font*)

Draws a text string at a given position using a given font.

Example::
font = aggdraw.Font(black, times) draw.text((100, 100), "hello, world", font)

Parameters

- **xy** – A 2-element Python sequence (x, y).
- **text** (*str*) – A string of text to render.
- **font** (*aggdraw.Font*) – The font object to render with.

Returns
A (width, height) tuple.

Return type
tuple

textsize(*text*, *font*)

Determines the size of a text string.

Parameters

- **text** (*str*) – A string of text to measure.
- **font** (*aggdraw.Font*) – The font object to render with.

Returns
A (width, height) tuple.

Return type
tuple

tobytes()

Copies data from the drawing area to a bytes object.

Returns
Packed image data compatible with PIL's frombytes method.

Return type
bytes

class aggdraw.**Font**(*color*, *file*, *size=12*, *opacity=255*)

Bases: object

Creates a font object.

This creates a font object for use with *aggdraw.Draw.text()* and *aggdraw.Draw.textsize()* from a TrueType font file.

The font color can be a color tuple (e.g. *(255, 255, 255)*), a CSS-style color name, or a color integer (0xAARRGGBB).

Parameters

- **color** – The font color.
- **file** – Path to a valid TrueType font file.
- **size** (*optional*) – The font size (in pixels). Defaults to 12.
- **opacity** (*int, optional*) – The opacity of the font (from 0 to 255). Defaults to solid.

class aggdraw.**Path**(*path=None*)

Bases: object

Path factory.

This creates a path object for use with *aggdraw.Draw.path()*.

close()

Closes the current path.

coords()

Returns the coordinates for the path.

Curves are flattened before being returned.

Returns
A sequence in (x, y, x, y, ...) format.

Return type
list

curveto(*x1*, *y1*, *x2*, *y2*, *x*, *y*)

Adds a bezier curve segment to the path.

lineto(*x*, *y*)

Adds a line segment to the path.

moveto(*x*, *y*)

Moves the path pointer to the given location.

rcurveto(*x1*, *y1*, *x2*, *y2*, *x*, *y*)

Adds a bezier curve segment to the path using relative coordinates.

Same as *curveto()*, but the coordinates are relative to the current position.

rlineto(*x*, *y*)

Adds a line segment to the path using relative coordinates.

Same as *lineto()*, but the coordinates are relative to the current position.

rmoveto(*x*, *y*)

Moves the path pointer relative to the current position.

class aggdraw.**Pen**(*color*, *width=1*, *opacity=255*)

Bases: object

Creates a pen object.

The pen color can be a color tuple (e.g. *(255, 255, 255)*), a CSS-style color name, or a color integer (0xAARRGGBB).

Parameters

- **color** – The pen color.
- **width** (*int, optional*) – The width of the pen.
- **opacity** (*int, optional*) – The opacity of the pen (from 0 to 255). Defaults to a solid pen.

class aggdraw.**Symbol**(*path*, *scale=1.0*)

Bases: object

Symbol factory.

This creates a symbol object from an SVG-style path descriptor for use with *aggdraw.Draw.symbol()*.

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)
- Z (close path)

Use lower-case operators for relative coordinates, upper-case for absolute coordinates.

Parameters
path (*str*) – An SVG-style path descriptor.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX