pycoast Documentation

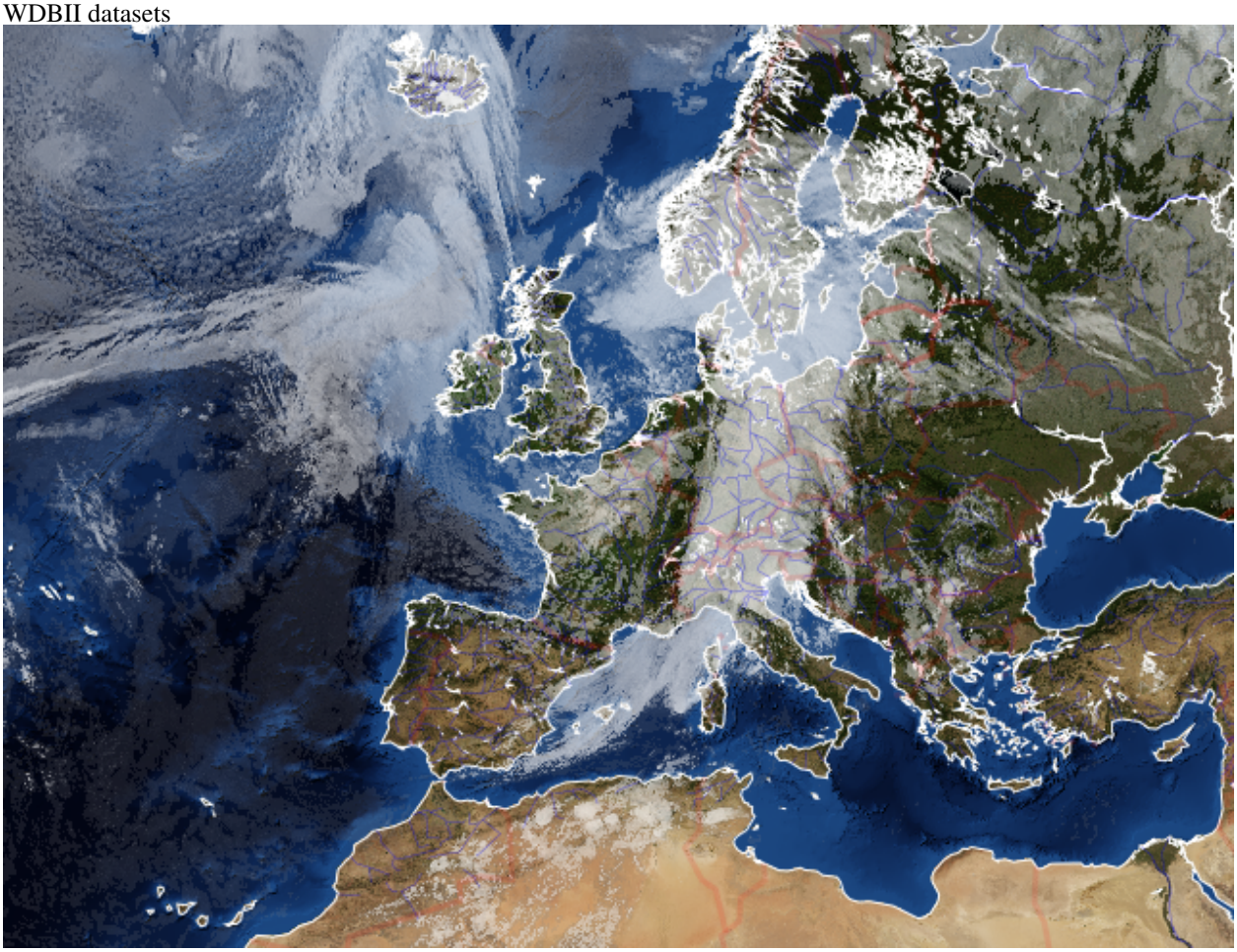
Release 0.5.2

Esben S. Nielsen

November 29, 2016

Contents

Pycoast is a Python package to add coastlines, borders and rivers to raster images using data from the GSHHS and WDBII datasets



CHAPTER 1

Installation

Pycoast depends on pyshp and PIL.

Install pycoast and dependencies.

Download the zipped GSHHS and WDBII Shapefiles from SOEST. Unzip the files to a data directory (hereafter called *GSHHS_DATA_ROOT*). The structure of *GSHHS_DATA_ROOT* should now be:

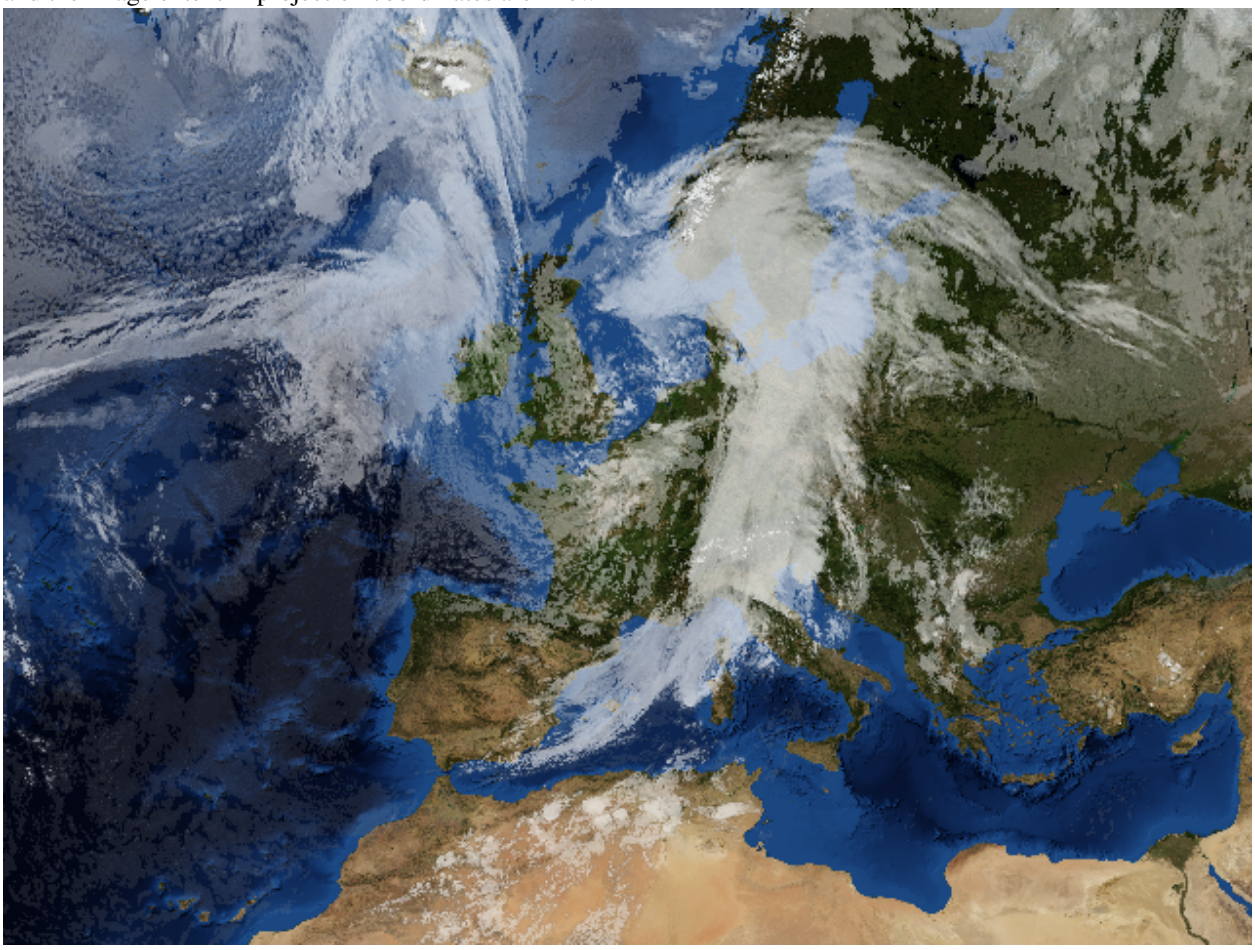
```
.
-- GSHHS_shp
|   -- c
|   -- f
|   -- h
|   -- i
|   -- l
-- WDBII_shp
    -- c
    -- f
    -- h
    -- i
    -- l
```

Where each dir on the lowest level contains Shapefiles like *GSHHS_shp/c/GSHHS_c_L1.shp*

CHAPTER 2

Usage

Pycoast can be used to add coastlines, borders and rivers to a raster image if the geographic projection of the image and the image extent in projection coordinates are known



Pycoast can add contours to either a PIL image object:

```
>>> from PIL import Image
>>> from pycoast import ContourWriter
>>> img = Image.open('BMNG_clouds_201109181715_areaT2.png')
>>> proj4_string = '+proj=stere +lon_0=8.00 +lat_0=50.00 +lat_ts=50.00 +ellps=WGS84'
>>> area_extent = (-3363403.31,-2291879.85,2630596.69,2203620.1)
>>> area_def = (proj4_string, area_extent)
```

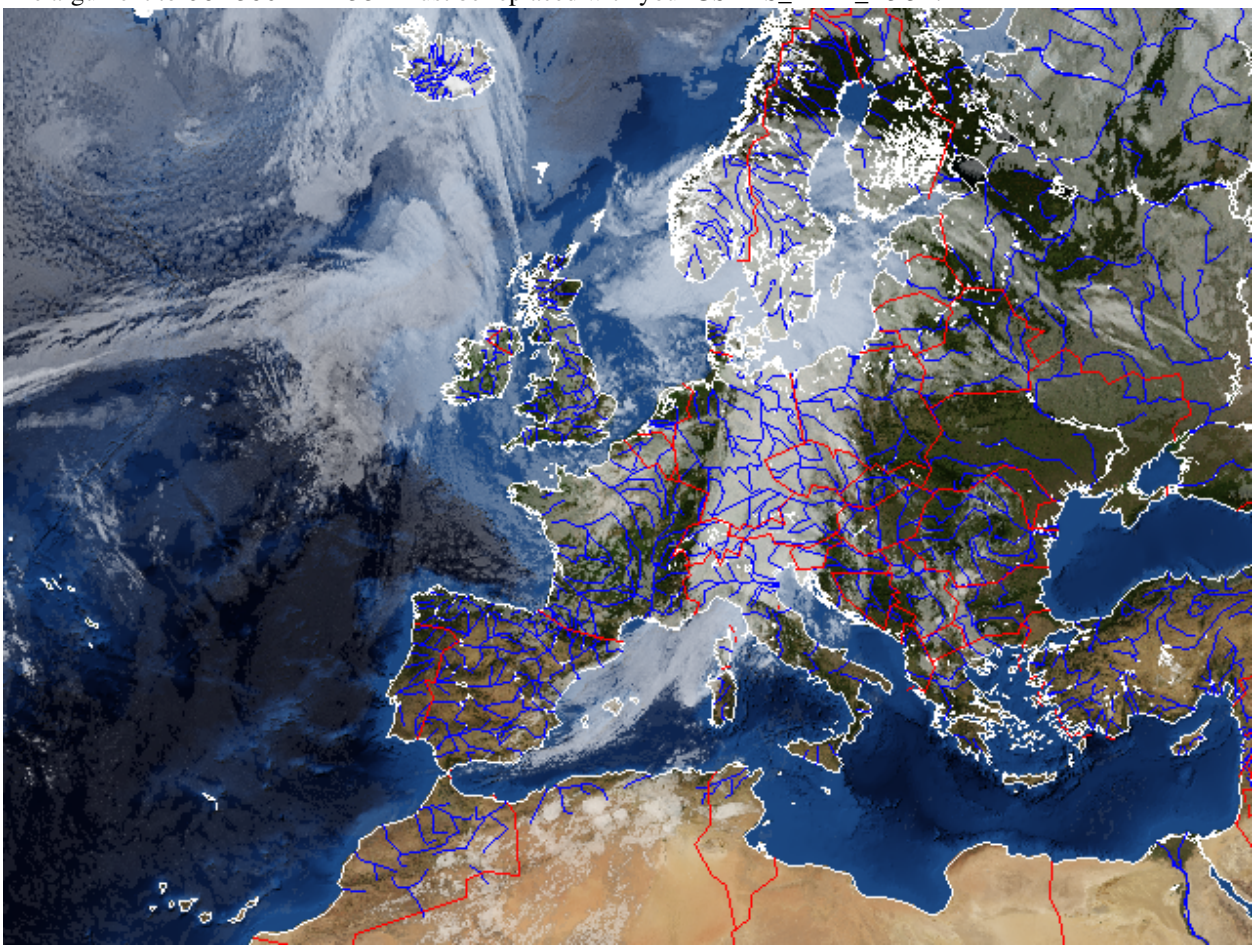
```
>>> cw = ContourWriter('/home/esn/data/gshhs')
>>> cw.add_coastlines(img, area_def, resolution='l', level=4)
>>> cw.add_rivers(img, area_def, level=5, outline='blue')
>>> cw.add_borders(img, area_def, outline=(255, 0, 0))
>>> img.show()
```

or to an image file:

```
>>> from pycoast import ContourWriter
>>> proj4_string = '+proj=stere +lon_0=8.00 +lat_0=50.00 +lat_ts=50.00 +ellps=WGS84'
>>> area_extent = (-3363403.31,-2291879.85,2630596.69,2203620.1)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriter('/home/esn/data/gshhs')
>>> cw.add_coastlines_to_file('BMNG_clouds_201109181715_areaT2.png', area_def, resolution='l', level=
>>> cw.add_rivers_to_file('BMNG_clouds_201109181715_areaT2.png', area_def, level=5, outline='blue')
>>> cw.add_borders_to_file('BMNG_clouds_201109181715_areaT2.png', area_def, outline=(255, 0, 0))
```

Where the `area_extent` is the extent of the image in projection coordinates as (x_ll, y_ll, x_ur, x_ur) measured at pixel edges.

The argument to `ContourWriter` must be replaced with your *GSHHS_DATA_ROOT*.



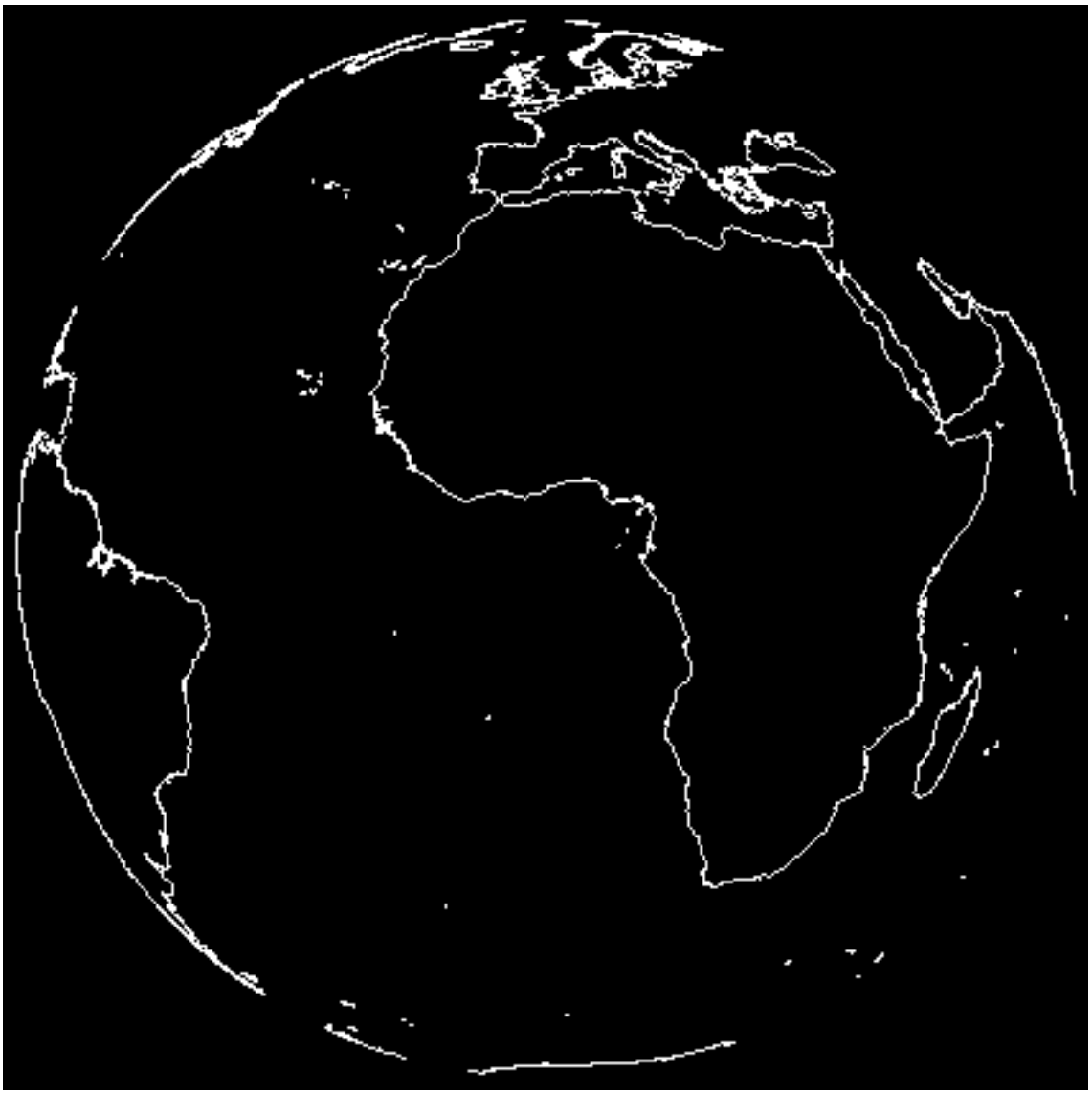
The resulting (not so pretty) image shows the effect of the various arguments. The `resolution` keyword argument controls the resolution of the dataset used. It defaults to 'c' for coarse. Increasing the resolution also increases the processing time. The `level` keyword argument controls the detail level of the dataset used. It defaults to *1* for the lowest detail level.

Instead of a tuple for `area_def` a pyresample `AreaDefinition` object can be used.

See method docstrings for information about possible argument values see method docstrings.

Creating an image with coastlines only:

```
>>> from PIL import Image
>>> from pycoast import ContourWriter
>>> img = Image.new('RGB', (425, 425))
>>> proj4_string = '+proj=geos +lon_0=0.0 +a=6378169.00 +b=6356583.80 +h=35785831.0'
>>> area_extent = (-5570248.4773392612, -5567248.074173444, 5567248.074173444, 5570248.4773392612)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriter('/home/esn/data/gshhs')
>>> cw.add_coastlines(img, area_def, resolution='l')
>>> img.show()
```



CHAPTER 3

High quality contours using AGG

The default plotting mode of pycoast uses PIL for rendering of contours. PIL does not support antialiasing and opacity. The AGG engine can be used for making high quality images using the aggdraw module.

First install the aggdraw module.

Tip: if the building of aggdraw fails with:

```
agg_array.h:523: error: cast from 'agg::int8u*' to 'unsigned int' loses precision
```

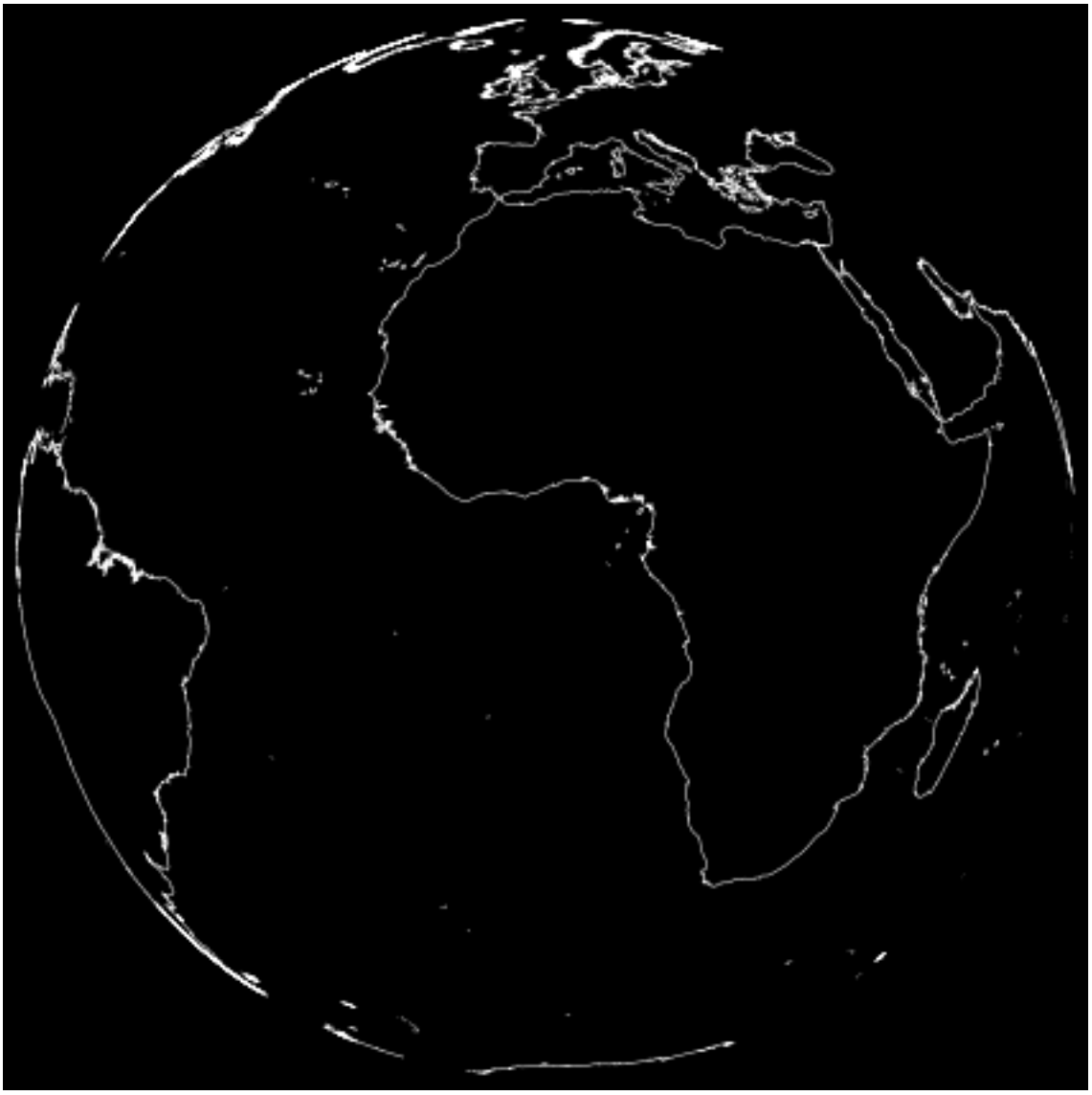
Try:

```
export CFLAGS="-fpermissive"
```

before building.

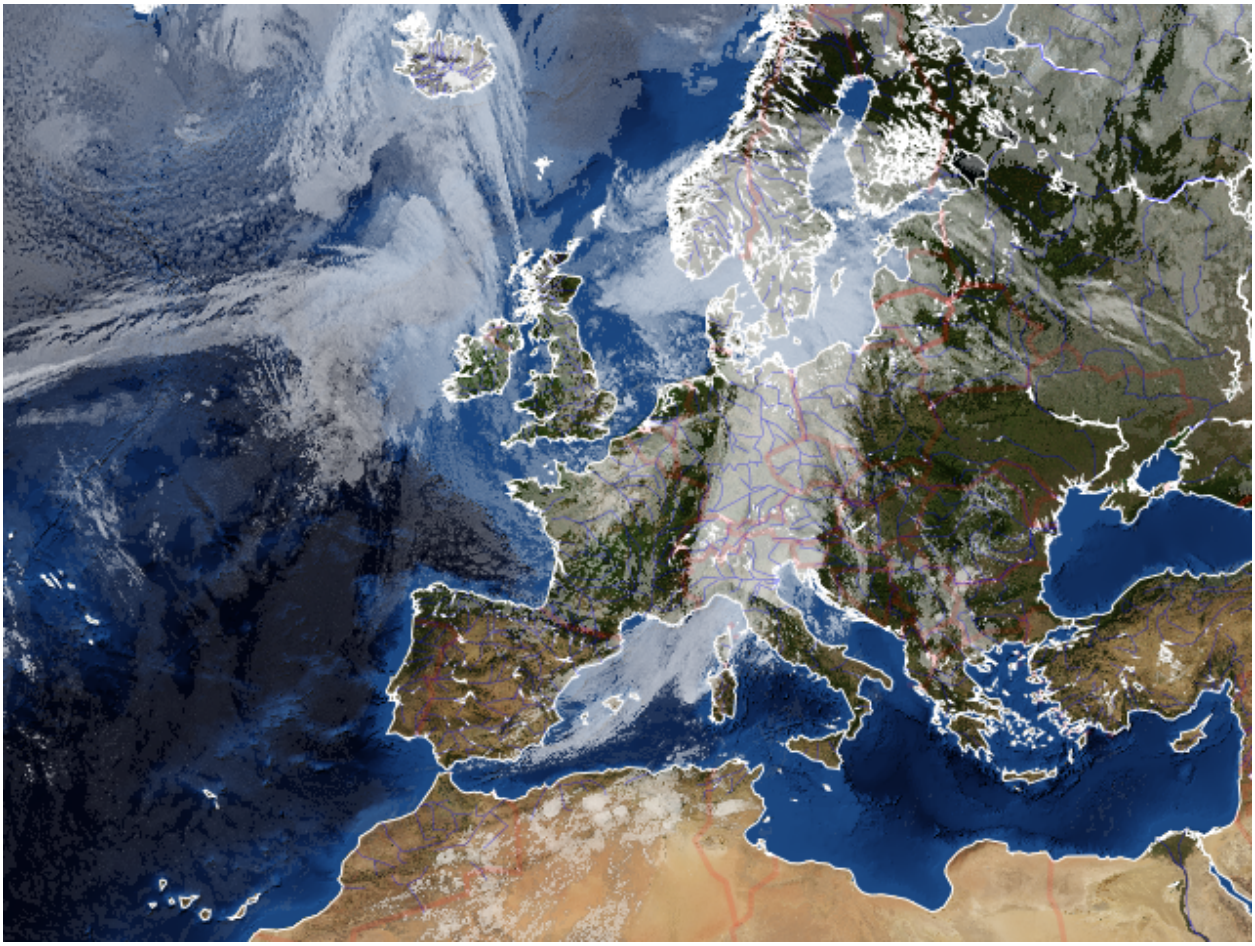
Using pycoast with AGG for making antialiased drawing:

```
>>> from PIL import Image
>>> from pycoast import ContourWriterAGG
>>> img = Image.new('RGB', (425, 425))
>>> proj4_string = '+proj=geos +lon_0=0.0 +a=6378169.00 +b=6356583.80 +h=35785831.0'
>>> area_extent = (-5570248.4773392612, -5567248.074173444, 5567248.074173444, 5570248.4773392612)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriterAGG('/home/esn/data/gshhs')
>>> cw.add_coastlines(img, (proj4_string, area_extent), resolution='l', width=0.5)
>>> img.show()
```



and making the not-so-nice image from the first example nice:

```
>>> from PIL import Image
>>> from pycoast import ContourWriterAGG
>>> img = Image.open('BMNG_clouds_201109181715_areaT2.png')
>>> proj4_string = '+proj=stere +lon_0=8.00 +lat_0=50.00 +lat_ts=50.00 +ellps=WGS84'
>>> area_extent = (-3363403.31,-2291879.85,2630596.69,2203620.1)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriterAGG('/home/esn/data/gshhs')
>>> cw.add_coastlines(img, area_def, resolution='l', level=4)
>>> cw.add_rivers(img, area_def, level=5, outline='blue', width=0.5, outline_opacity=127)
>>> cw.add_borders(img, area_def, outline=(255, 0, 0), width=3, outline_opacity=32)
>>> img.show()
```



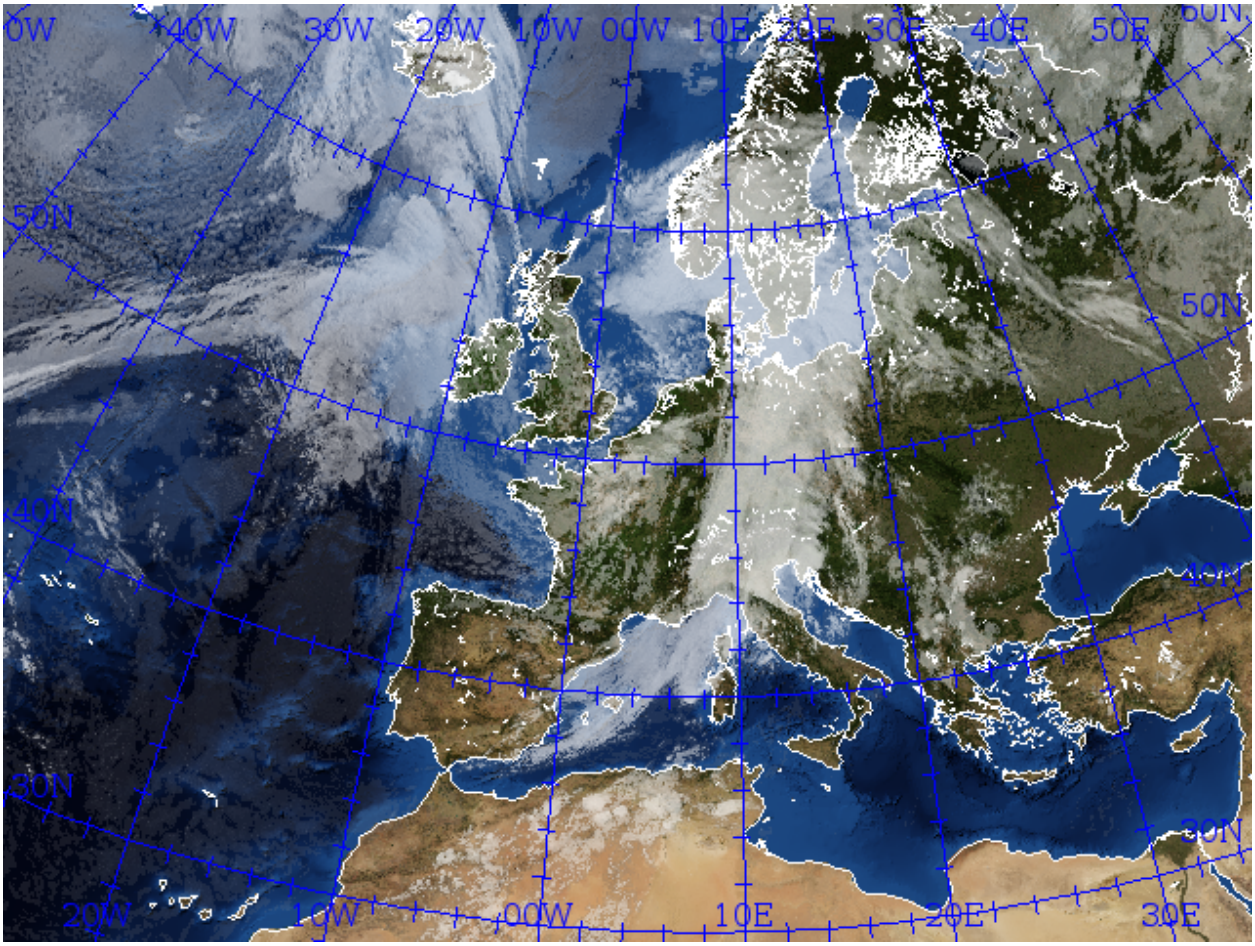
See docstrings of `ContourWriterAGG` methods for argument descriptions.

CHAPTER 4

Adding graticule to images

Pycoast can be used to add graticule to images. For PIL:

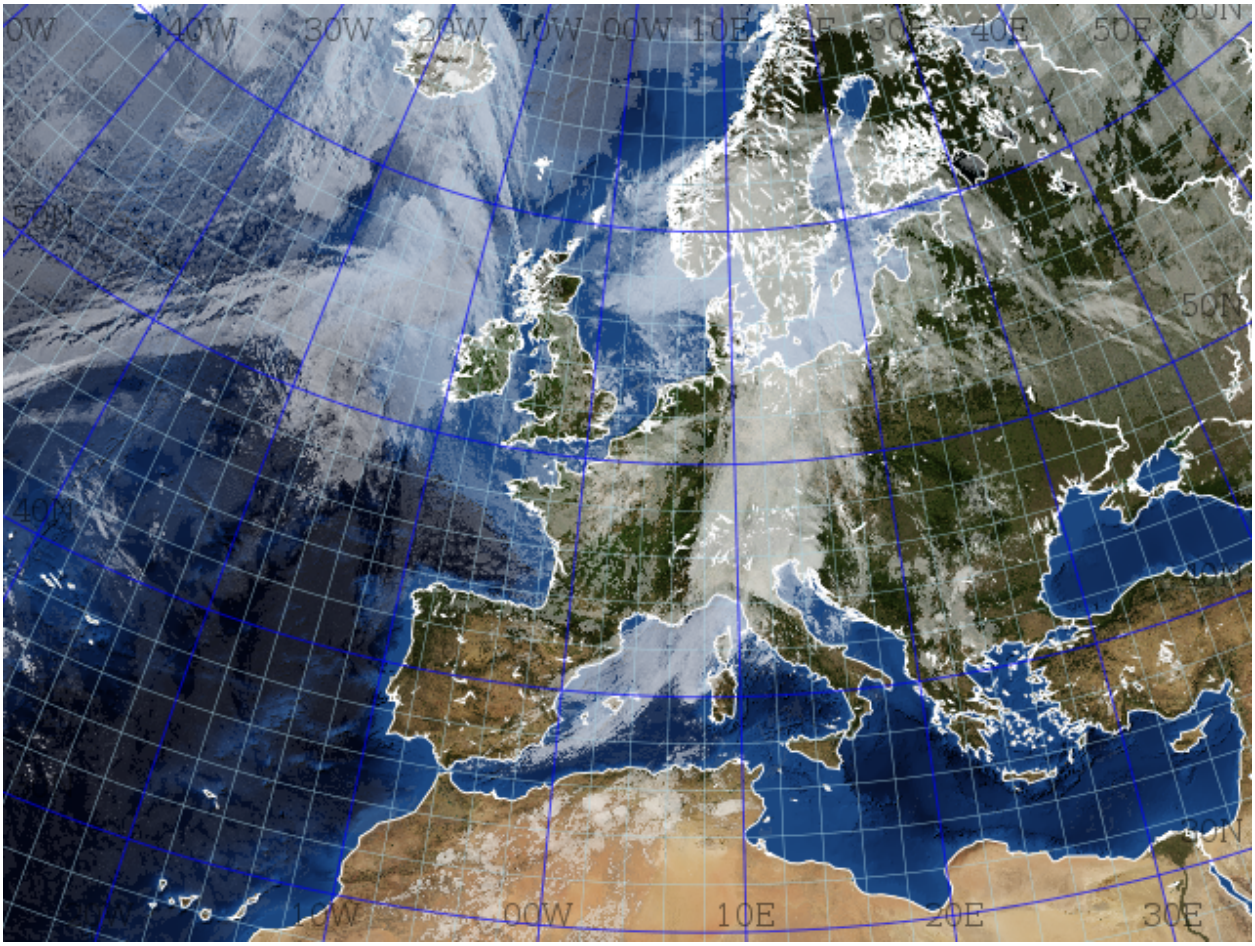
```
>>> from PIL import Image, ImageFont
>>> from pycoast import ContourWriter
>>> proj4_string = '+proj=stere +lon_0=8.00 +lat_0=50.00 +lat_ts=50.00 +ellps=WGS84'
>>> area_extent = (-3363403.31,-2291879.85,2630596.69,2203620.1)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriter('/home/esn/data/gshhs')
>>> font = ImageFont.truetype("/usr/share/fonts/truetype/ttf-dejavu/DejaVuSerif.ttf",16)
>>> img = Image.open('BMNG_clouds_201109181715_areaT2.png')
>>> cw.add_coastlines(img, area_def, resolution='l', level=4)
>>> cw.add_grid(img, area_def, (10.0,10.0),(2.0,2.0), font,fill='blue',
...             outline='blue', minor_outline='blue')
>>> img.show()
```



The font argument is optional for PIL if it is not given a default font will be used.

and for AGG:

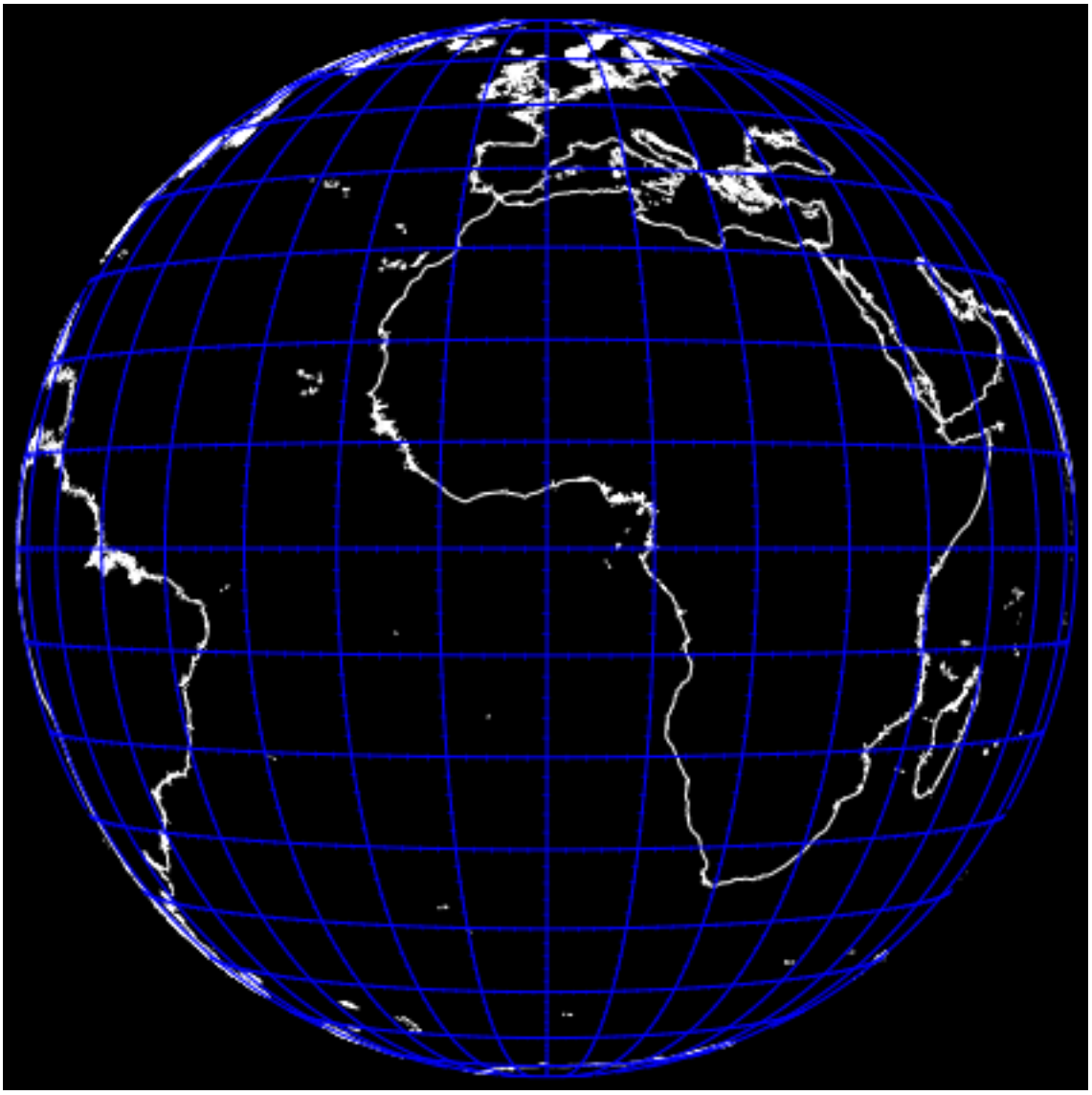
```
>>> from PIL import Image, ImageFont
>>> from pycoast import ContourWriterAGG
>>> import aggdraw
>>> proj4_string = '+proj=stere +lon_0=8.00 +lat_0=50.00 +lat_ts=50.00 +ellps=WGS84'
>>> area_extent = (-3363403.31,-2291879.85,2630596.69,2203620.1)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriterAGG('/home/esn/data/gshhs')
>>> font = aggdraw.Font('black', '/usr/share/fonts/truetype/ttf-dejavu/DejaVuSerif.ttf',
...                     opacity=127, size=16)
>>> img = Image.open('BMNG_clouds_201109181715_areaT2.png')
>>> cw.add_coastlines(img, area_def, resolution='l', level=4)
>>> cw.add_grid(img, area_def, (10.0,10.0),(2.0,2.0),font,
...             outline='blue',outline_opacity=175,width=1.0,
...             minor_outline='lightblue',minor_outline_opacity=200,minor_width=0.5,
...             minor_is_tick=False)
>>> img.show()
```



Note the difference in the optional font argument for PIL and AGG. With AGG the font argument is mandatory unless the keyword argument `write_text=False` is used.

From v0.5.0 the graticule is also usable for globe projections:

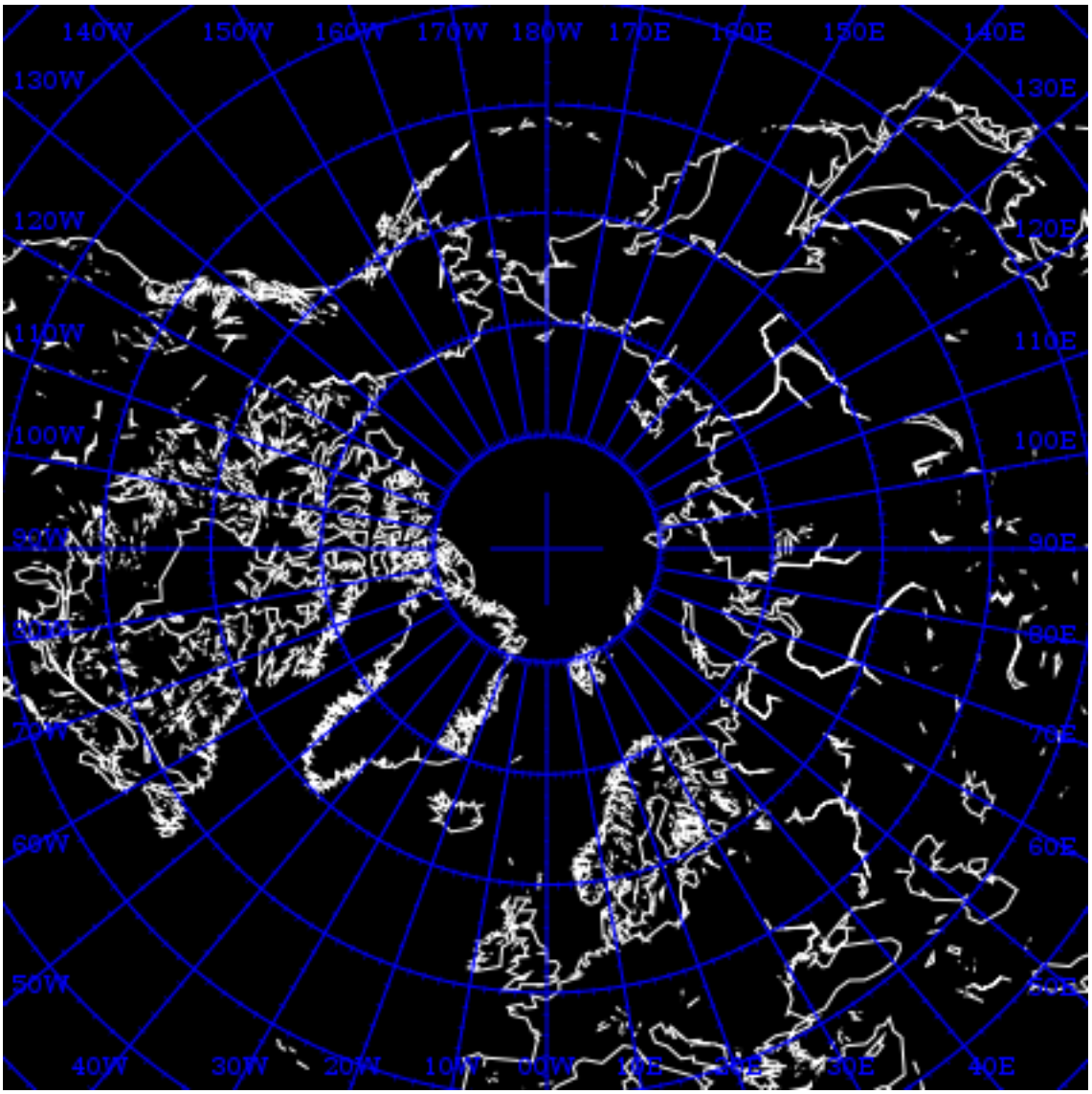
```
>>> from PIL import Image
>>> from pycoast import ContourWriterAGG
>>> img = Image.new('RGB', (425, 425))
>>> proj4_string = '+proj=geos +lon_0=0.0 +a=6378169.00 +b=6356583.80 +h=35785831.0'
>>> area_extent = (-5570248.4773392612, -5567248.074173444, 5567248.074173444, 5570248.4773392612)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriterAGG(gshhs_root_dir)
>>> cw.add_coastlines(img, area_def, resolution='l')
>>> cw.add_grid(img, area_def, (10.0,10.0),(2.0,2.0), fill='blue',
... outline='blue', minor_outline='blue', write_text=False)
>>> img.show()
```



The lon and lat labeling is shown where the lines intersect the image border. By default the lon intersections with top and bottom and the lat intersections with left and right border are displayed . The placement behaviour can be controlled with the `lon_placement` and `lat_placement` keyword variables. The placement specifier is a string containing the desired placement where 't' is top, 'b' bottom, 'l' left and 'r' right. E.g. `lon_placement='tl'` will make the lon labels display at the top and left border.

```
>>> from PIL import Image
>>> from pycoast import ContourWriterAGG
>>> import aggdraw
>>> img = Image.new('RGB', (425, 425))
>>> proj4_string = '+proj=laea +lat_0=90 +lon_0=0 +a=6371228.0 +units=m'
>>> area_extent = (-5326849.0625, -5326849.0625, 5326849.0625, 5326849.0625)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriterAGG('/home/esn/data/gshhs')
>>> cw.add_coastlines(img, area_def, resolution='c', level=4)
>>> font = aggdraw.Font('blue', '/usr/share/fonts/truetype/ttf-dejavu/DejaVuSerif.ttf', size=10)
>>> cw.add_grid(img, area_def, (10.0,10.0),(2.0,2.0), font=font, fill='blue',
...             outline='blue', minor_outline='blue',
...             lon_placement='tblr', lat_placement='')
```

```
>>> img.show()
```



Tip: If the adding graticule with AGG fails with something like:

```
Traceback (most recent call last):
    File "grid_demo_AGG.py", line 13, in <module>
        font=aggdraw.Font("blue", "/usr/share/fonts/truetype/ttf-dejavu/DejaVuSerif.ttf", size=16)
IOError: cannot load font (no text renderer)
```

make sure the FREETYPE_ROOT in setup.py of aggdraw points to the correct location e.g. set *FREETYPE_ROOT = "/usr"*

CHAPTER 5

Pycoast from a configuration file

If you want to run to avoid typing the same options over and over again, of if caching is an optimization you want, you can use a configuration file with the pycoast options you need:

```
[cache]
file=/var/run/satellit/white_overlay

[coasts]
level=1
width=0.75
outline=white
fill=yellow

[borders]
outline=white
width=0.5
```

Then, you can just call:

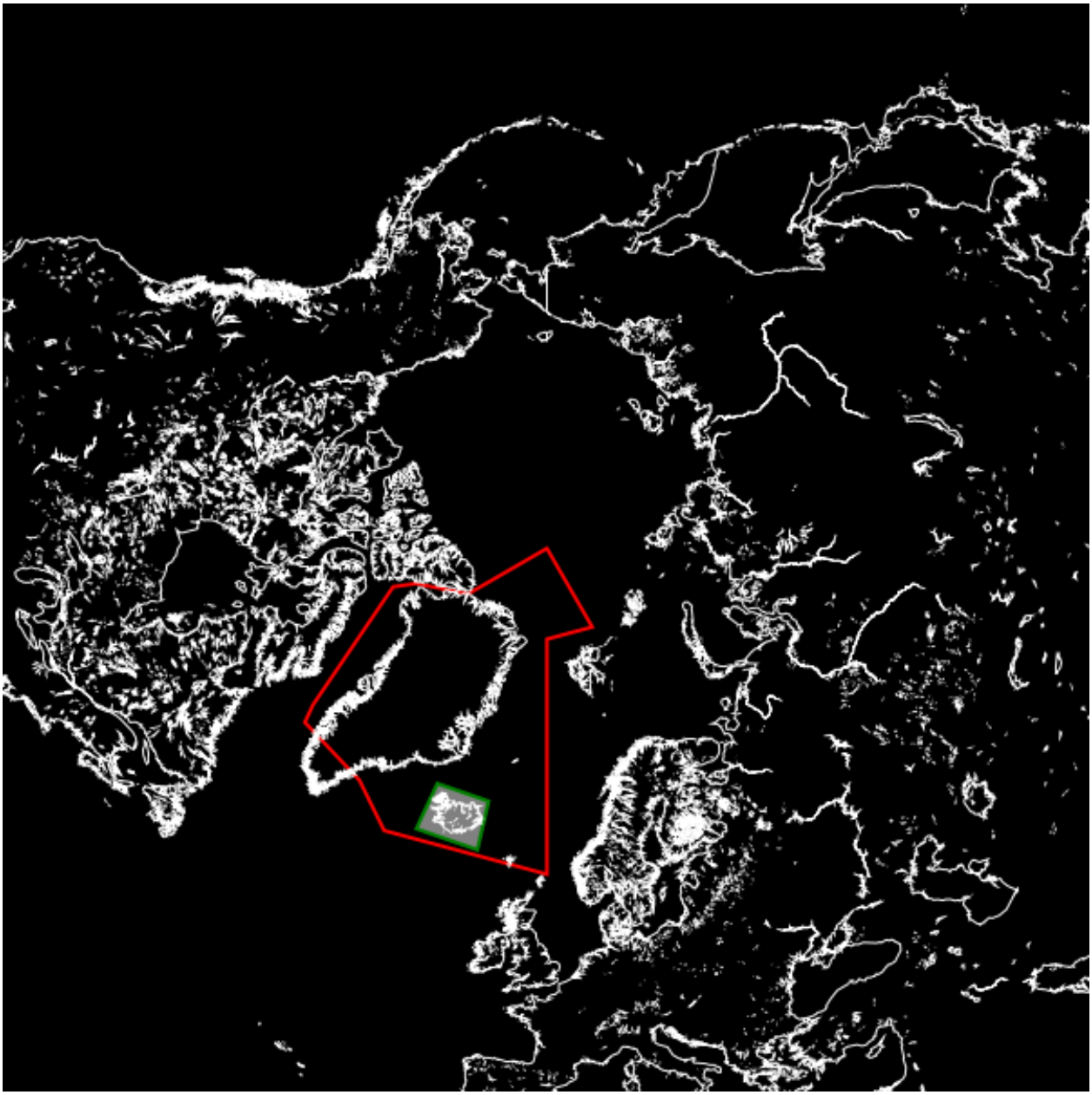
```
>>> cw.add_overlay_from_config("my_config.cfg", area_def)
```

CHAPTER 6

Custom shapes and lines

Pycoast can add custom polygons and lines, useful for outlining special target areas. The following example shows how we might use the `add_polygon` method to highlight the Reykjavik Air Traffic Control area and an additional filled box around Iceland.

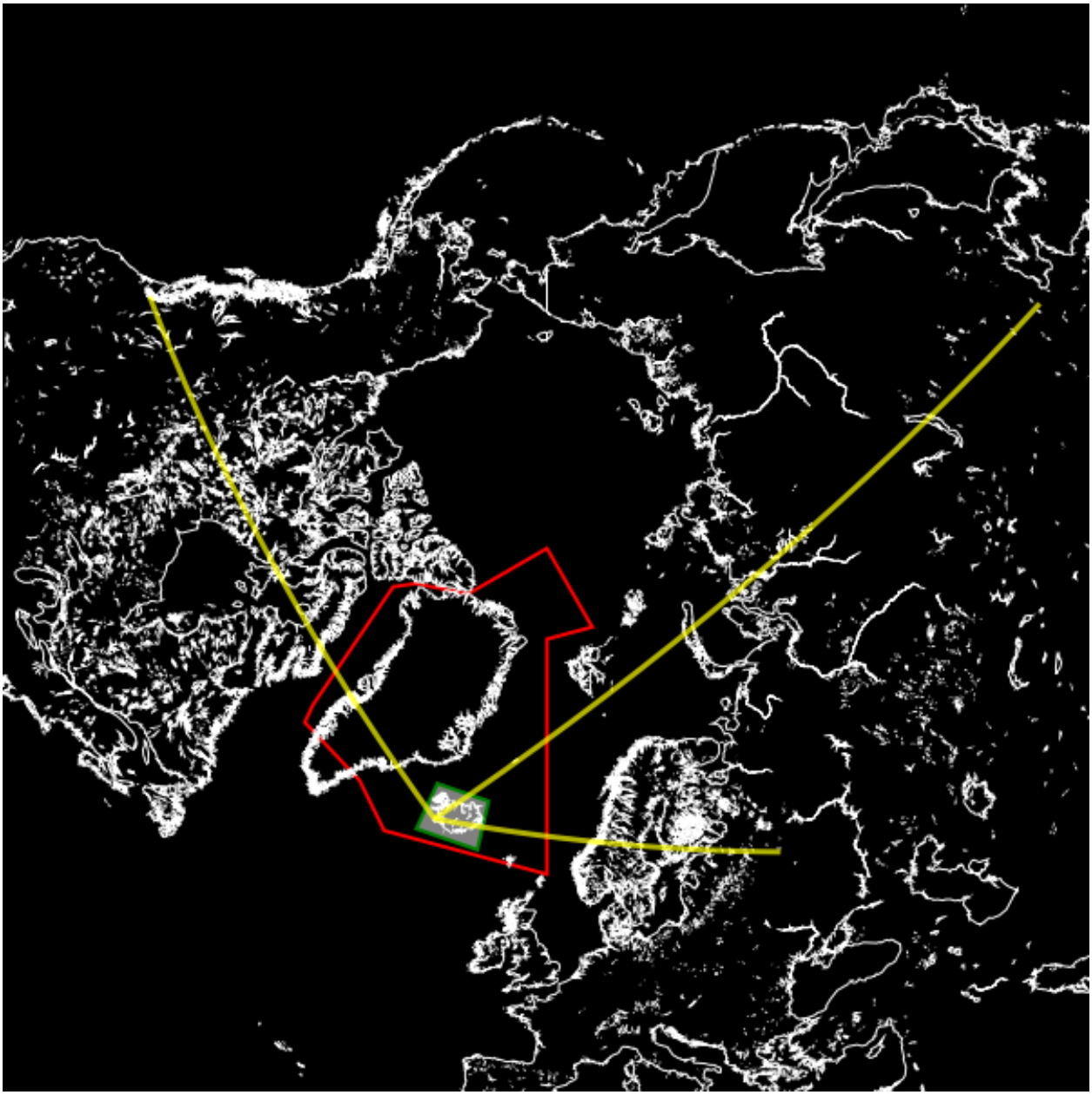
```
>>> from PIL import Image
>>> from pycoast import ContourWriterAGG
>>> img = Image.new('RGB', (600, 600))
>>> proj4_string = '+proj=laea +lat_0=90 +lon_0=0 +a=6371228.0 +units=m'
>>> area_extent = (-5326849.0625, -5326849.0625, 5326849.0625, 5326849.0625)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriterAGG('/home/esn/data/gshhs')
...
>>> REYKJAVIK_ATC = [(0.0,73.0),(0.0,61.0),(-30.0,61.0),(-39,63.5),(-55+4/6.0,63.5),(-57+45/60.0,65),
>>> ICELAND_BOX = [(-25,62.5),(-25,67),(-13,67),(-13,62.5)]
>>> cw.add_polygon(img, area_def, REYKJAVIK_ATC, outline='red',width=2)
>>> cw.add_polygon(img, area_def, ICELAND_BOX, outline='green', fill='gray', width=2)
>>> cw.add_coastlines(img, area_def, resolution='l', level=4)
>>> img.show()
```



The `add_polygon` method accepts a list of longitude, latitude pairs. An equivalent `add_line` method is also available which does not tie the first and last coordinates in the list.

Now we can plot some air traffic routes from Keflavik to Seattle, Moscow and Beijing,

```
>>> ROUTE_KEF_MOS = [(-22.6056, 63.985), (-19.046655824698217, 64.2936159845089), (-15.41883293246517
>>> ROUTE_KEF_SEA = [(-22.6056, 63.985), (-28.15308892820336, 65.36580325755281), (-34.26244035327647
>>> ROUTE_KEF_BEI = [(-22.6056, 63.985), (-17.489150553128045, 67.07686353046147), (-10.935411352029
>>> cw.add_line(img, area_def, ROUTE_KEF_MOS, outline='yellow',outline_opacity=180,width=3)
>>> cw.add_line(img, area_def, ROUTE_KEF_SEA, outline='yellow',outline_opacity=180,width=3)
>>> cw.add_line(img, area_def, ROUTE_KEF_BEI, outline='yellow',outline_opacity=180,width=3)
```

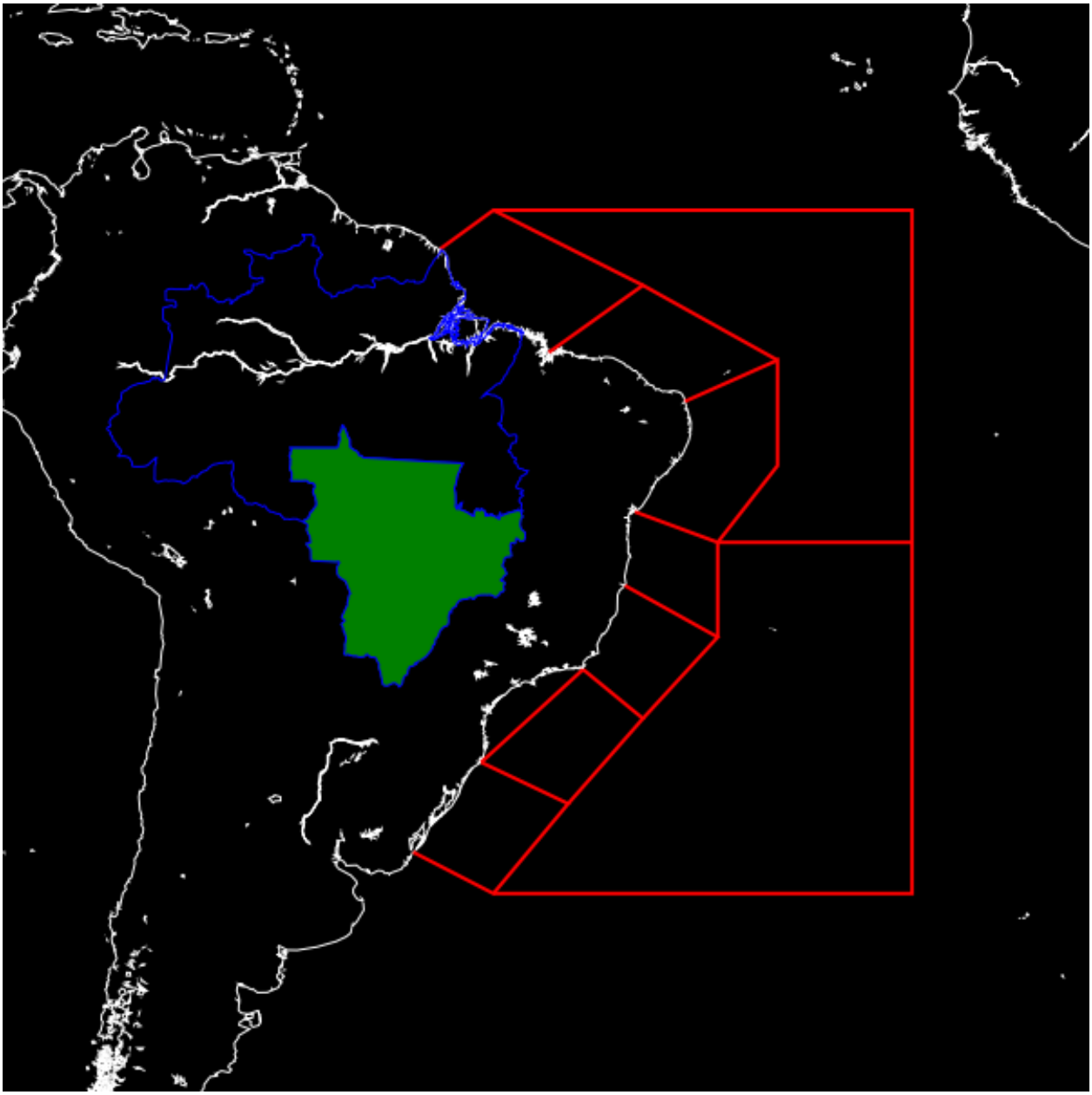


CHAPTER 7

Shapes from ESRI shape files

Pycoast supports plotting of polygons and polylines from ESRI shapefiles, currently only in lonlat coordinates format. In the following example we use `add_shapefile_shapes` method to plot all line shapes found in the file Metareas.shp. We then use the `add_shapefile_shape` (notice the singular) to plot only the 3rd and 4th shape_id within the file BR_Regioes.shp.

```
>>> from pycoast import ContourWriter
>>> img = Image.new('RGB', (600, 600))
>>> proj4_string = '+proj=merc +lon_0=-60 +lat_ts=-30.0 +a=6371228.0 +units=m'
>>> area_extent = (-2000000.0, -5000000.0, 5000000.0, 2000000.0)
>>> area_def = (proj4_string, area_extent)
>>> cw = ContourWriterAGG(gshhs_root_dir)
...
>>> cw.add_coastlines(img, area_def, resolution='l', level=4)
>>> cw.add_shapefile_shapes(img, area_def,
                            'test/test_data/shapes/Metareas.shp',
                            outline='red',width=2)
>>> cw.add_shapefile_shape(img, area_def,
                            'test/test_data/shapes/divisao_politica/BR_Regioes.shp',
                            3, outline='blue')
>>> cw.add_shapefile_shape(img, area_def,
                            'test/test_data/shapes/divisao_politica/BR_Regioes.shp',
                            4, outline='blue', fill='green')
```



If your shapefile is not in lonlat coordinates, then you can re-project your shape file using `ogr2ogr` (part of GDAL tools), e.g.

```
$ ogr2ogr original.shp  lonlat.shp  -t_srs "+proj=latlong"
```

This should work if you have all the extra meta-files original.* included with your original.shp. Please refer to the OGR documentation for more information.

CHAPTER 8

Testing

The tests can be run using nosetest:

```
$ cd <pycoast_dir>
$ nosetests tests/
```

CHAPTER 9

The pycoast API

9.1 Pycoast