**Documentation for the**

**Python Geographic Visualizer (Geovis)**

**Version 0.1.0**

**Author**

Karim Bahgat, karim.bahgat.norway@gmail.com

**Date**

February 21, 2014

**License**

Creative Commons Attribution-ShareAlike CC BY-SA, http://creativecommons.org/licenses/by-sa/4.0/

**Introduction**

The Python Geographic Visualizer (GeoVis) is a complete geographic visualization module intended for easy everyday-use by the general user. It has one-liners for quickly visualizing a shapefile, building and styling basic maps with multiple shapefile layers, and/or saving to imagefiles. Works as a wrapper around multiple possible rendering modules. For shapefile reading it uses a modified version of Joel Lawhead’s [PyShp module](http://code.google.com/p/pyshp/). For color magic it uses Valentin Lab’s [Colour module](https://pypi.python.org/pypi/colour),

The current version is functional, but should be considered a work in progress with potential bugs.

**Dependencies**

Technically has no external dependencies, but is highly recommended to be used with either Aggdraw, PIL or PyCairo. Default is set to Aggdraw so if using a different renderer this has to be specified for each session. If none of these are available it is still possible to use the Tkinterer Canvas as a renderer, but due to major limitations this is discouraged for larger files.

**Python Versions**

Should work on Python version 2.x. Has not yet been tested on Python 3.x.

**Installation**

No installation required, just use sys.path.append to the geovis folder location or place the geovis folder in you Python site-packages folder, and import it using “import geovis”.

**Classes and Functions**

**RENDERING OPTIONS**

***These functions let you set your preferences for how to create the visual representations of your spatial data, such as the renderer to use, and speed/quality options.***

SetRenderingOptions(\*\*renderoptions)

Sets certain rendering options that apply to all visualizations or map images.\

-renderer is a string describing which Python module will be used for rendering. This means you need to have the specified module installed. Valid renderer values are 'tkinter', 'PIL', 'aggdraw', 'pycairo'. Notes: If you have no renderers installed, then use Tkinter which comes with all Python installations, although it is significantly slow, memory-limited, and cannot be used to save images. Currently PyCairo is not very well optimized, and is particularly slow to render line shapefiles. \

-numpyspeed specifies whether to use numpy to speed up shapefile reading and coordinate-to-pixel conversion. Must be True or False.\

-reducevectors specifies whether to reduce the number of vectors to be rendered. This can speed up rendering time, but may lower the quality of the rendered image, especially for line shapefiles.

**QUICK TASKS**

***These are the functions you will be working with the most, providing visualizations of individual shapefiles with only one line of code, no setup required.***

ViewShapefile(shapefilepath, \*\*customoptions)

Quick task to visualize a shapefile and show it in a Tkinter window.\

-shapefilepath is the path string of the shapefile.\

-customoptions is any series of named arguments of how to style the shapefile visualization (optional). Valid arguments are: fillcolor, fillsize (circle size for point shapefiles, line width for line shapefiles, and no effect for polygon shapefiles), outlinecolor, outlinewidth.

SaveShapefileImage(shapefilepath, savepath, \*\*customoptions)

Quick task to save a shapefile to an image.\

-shapefilepath is the path string of the shapefile.\

-savepath is the path string of where to save the image, including the image type extension.\

-customoptions is any series of named arguments of how to style the shapefile visualization (optional). Valid arguments are: fillcolor, fillsize (circle size for point shapefiles, line width for line shapefiles, and no effect for polygon shapefiles), outlinecolor, outlinewidth.

**MAP SPECS**

***These functions allow you to specify various aspects of the next map to be visualized.***

SetMapDimensions(width, height)

Sets the width and height of the next map. At startup the width and height are set to the dimensions of the window screen.\

-width/height must be integers.

SetMapBackground(mapbackground)

Sets the mapbackground of the next map to be made. At startup the mapbackground is transparent (None).\

-mapbackground takes a hex color string, as can be created with the Color function. It can also be None for a transparent background.

**STYLE CUSTOMIZING**

***These functions allow or help you to style different aspects of a shapefile’s visual appearance.***

SetMapSymbols(\*\*optionstochange)

Sets the default symbol options that will be used to visualize shapefiles. These default options will only be used for those options of a shapefile that the user has not already specified.\

-customoptions are named arguments specifying the map’s default symbol options. Valid arguments are: fillcolor, fillsize (circle size for point shapefiles, line width for line shapefiles, and no effect for polygon shapefiles), outlinecolor, outlinewidth.

Color(basecolor="random", intensity="random", brightness="random", style=None)

Returns a hex color string of the color options specified. \

-basecolor is the human-like name of a color.\

-intensity of how strong the color should be. Must be a float between 0 and 1.\

-brightness of how light or dark the color should be. Must be a float between 0 and 1.

-style is a named style that overrides the brightness and intensity options. Valid style names are: strong, dark, matte, bright, pastelle.

ColorFeeder(\*\*coloroptions)

Returns an infinite generator of colors of a specified style. To generate a new color of the specified color options use the generator as an argument to the next() method or loop through it.\

-coloroptions is any series of named arguments to specify aspects of a color style. See the Color object documentation.

**MAP BUILDING**

***These classes and functions lets you build a more advanced map layout, displaying not only one but several shapefiles on the same map.***

NewMap()

Creates and returns a new map based on previously defined mapsettings."

AddToMap(self, shapefilepath, \*\*customoptions):

Add a shapefile to the map.\

-shapefilepath is the path string of the shapefile to add.

-customoptions are named arguments specifying the layer’s symbol options

ViewMap(self):

View the created map in a Tkinter window"

SaveMap(self, savepath):

Save the map to an image file.\

-savepath is the string path for where you wish to save the map image. Image type extension must be specified ('.png','.gif',...)"

**GENERAL UTILITIES**

***These functions are general utilities to ease or automate common tasks that are not necessarily related to mapping or visualization, such as finding and dealing with the spatial data you wish to visualize.***

ShapefileFolder(folder)

Returns a generator that will loop through a folder and all its subfolder and return information of every shapefile it finds. Information returned is a tuple with the following elements (string name of current subfolder, string name of shapefile found, string of the shapefile's file extension(will always be '.shp'))

-folder is a path string of the folder to check for shapefiles.