University of Freiburg
Faculty of Environment and Natural Resources
Module: GIS Plus
Lecturer: João Paulo Pereira, Holger Weinacker
Authors: Luka Kern, Nele Stackelberg and Felix Rentschler

July 6, 2018

Project 06: Rasterizer Function

Abstract

A rasterinzing tool creates a regular grid whithin the bounding box of a geometry collection and assigns a value to each cell depending on the presence of a geometry in that cell region. This information can either be a binary one (no-data / data) or based on the geometry attributes. For this Project, the first cell will create a randomized geometry list containing the input data. The attributes for each of the three geometries are stored in a separated attributes list.

1 Aim of the Project

The aim of this project was to create a function that generates a raster-file (.tif). This file displays the geometries from a shapfile, which was given as input to the function. The spatial resolution and name of the output-file can be defined as input to the function. For the case that no shapefile is available to run the function, a code to generate random geometries is provided.

2 Structure of the function

2.1 Load in the shapefile

To load in the shapefile, the fiona-package is used. Every geometry-object of the shapefile is stored in a GeometryCollection from the shapely-package.

2.2 Defining the bounding box

We create a buffer around the actual shape of the shapefile which will look prettier in the resulting image.

```
within_list = []
for i in range(0, len(geometry_coll)):
    if isinstance(geometry_coll[i], spg.polygon.Polygon):
        step = [pixel.within(geometry_coll[i]) for pixel in geom_pixels]
    if isinstance(geometry_coll[i], spg.point.Point):
        step = [
            (
                    (pixel.x > (geometry_coll[i].x - 0.5 * resolution)) &
                    (pixel.x <= (geometry_coll[i].x + 0.5 * resolution))
            ) &
            (
                    (pixel.y > (geometry_coll[i].y - 0.5 * resolution)) &
                    (pixel.y <= (geometry_coll[i].y + 0.5 * resolution))
```

```
        ) for pixel in geom_pixels
    ]

if isinstance(geometry_coll[i], spg.linestring.LineString):
    step = [pixel.within(geometry_coll[i].buffer(float(resolution)))
            for pixel in geom_pixels]
print('The process is running: {}% completed'.format(
    (round(100 * i / len(geometry_coll), 2))))
within_list.append(step)
```

```
class MyClass(Yourclass):
                def __init__(self, my, yours):
                bla = '5 1 2 3 4'
                print bla
```

2.3 R Code

```
#============================================================================
#                          Initial Values
#============================================================================
def rasterizer(filepath,
                                                    pixels=100,
                                                buffer=10,
outputname="output.tiff",
save=True,
preview=True):
```

3 Issues

3.1 Solved

3.2 Unsolved