Polycircles Documentation

Release 0.1

Adam Matan

August 04, 2016

Contents

Polygonal circle approximations for KMLs and general use.

CHAPTER 1

Getting started

1.1 Installing

```
pip install polycircles
```

1.2 Your first KML circle

Generates a circle approximation readable by simpleKML.

```
from polycircles import polycircles

polycircle = polycircles.Polycircle(latitude=40.768085,
                                    longitude=-73.981885,
                                    radius=200,
                                    number_of_vertices=36)
kml = simplekml.Kml()
pol = kml.newpolygon(name="Columbus Circle, Manhattan",
                                          outerboundaryis=polycircle.to_kml())
pol.style.polystyle.color = \
        simplekml.Color.changealphaint(200, simplekml.Color.green)
kml.save("test_kml_polygon_3_manhattan.kml")
```

Note that a polygon with 36 vertices looks pretty much like a circle:

1.3 Sequence of lat-lon points

`polycircles` can simply generate a series of lat-lon tuples, for any non-KML usage.

```
   import pprint
   from polycircles import polycircles
   polycircle = polycircles.Polycircle(latitude=32.074523,
                                       longitude=34.791469,
                                       radius=20,
                                       number_of_vertices=12)
   pprint.pprint(polycircle.to_lat_lon())
   ((32.07470336197859, 34.791469),
    (32.074679198011374, 34.7915749137218),
...
```

Fig. 1.1: A Polycircle above Columbus Square, Columbus Square, Manhattan, NYC, United States
Image: Google Earth, Credit: Google

```
    (32.074613180857156, 34.79128555218445),
    (32.074679198011374, 34.791363086278196))
```

CHAPTER 2

Installing and testing

2.1 Installing

The following command:

```
pip install polycircles
```

Will install polycircles and all its dependencies.

2.2 Testing

First, clone the gir repo.

```
git clone https://github.com/adamatan/polycircles.git
```

2.2.1 Unit tests

```
python setup.py test
```

2.2.2 BDD tests

I use behave for BDD tests.

```
pip install -r test_requirements.txt
behave
```

Travis builds

The package is built and tested in travis-ci for every commit.

CHAPTER 3

KML circles

3.1 Basic KML circle

Generates a circle approximation readable by simpleKML.

```
from polycircles import polycircles

polycircle = polycircles.Polycircle(latitude=40.768085,
                                    longitude=-73.981885,
                                    radius=200,
                                    number_of_vertices=36)
kml = simplekml.Kml()
pol = kml.newpolygon(name="Columbus Circle, Manhattan",
                                          outerboundaryis=polycircle.to_kml())
pol.style.polystyle.color = \
        simplekml.Color.changealphaint(200, simplekml.Color.green)
kml.save("test_kml_polygon_3_manhattan.kml")
```

Note that a polygon with 36 vertices looks pretty much like a circle:

3.2 Donut! (Well, Torus)

Using the `innerboundaryis` of simpleKML Polygon object and two polycircles, a donut-shape can be easily created:

```
outer_polycircle = polycircles.Polycircle(latitude=40.768085,
                                          longitude=-73.981885,
                                          radius=200,
                                          number_of_vertices=36)
inner_polycircle = polycircles.Polycircle(latitude=40.768085,
                                          longitude=-73.981885,
                                          radius=180,
                                          number_of_vertices=36)

kml = simplekml.Kml()
pol = kml.newpolygon(name="Torus around Columbus Circle, Manhattan",
                                          outerboundaryis=outer_polycircle.to_kml(),
                                          innerboundaryis=inner_polycircle.to_kml())
pol.style.polystyle.color = \
        simplekml.Color.changealphaint(200, simplekml.Color.red)
kml.save("test_kml_polygon_2_torus_manhattan.kml")
```

Fig. 3.1: A Polycircle above Columbus Square, Columbus Square, Manhattan, NYC, United States
Image: Google Earth, Credit: Google

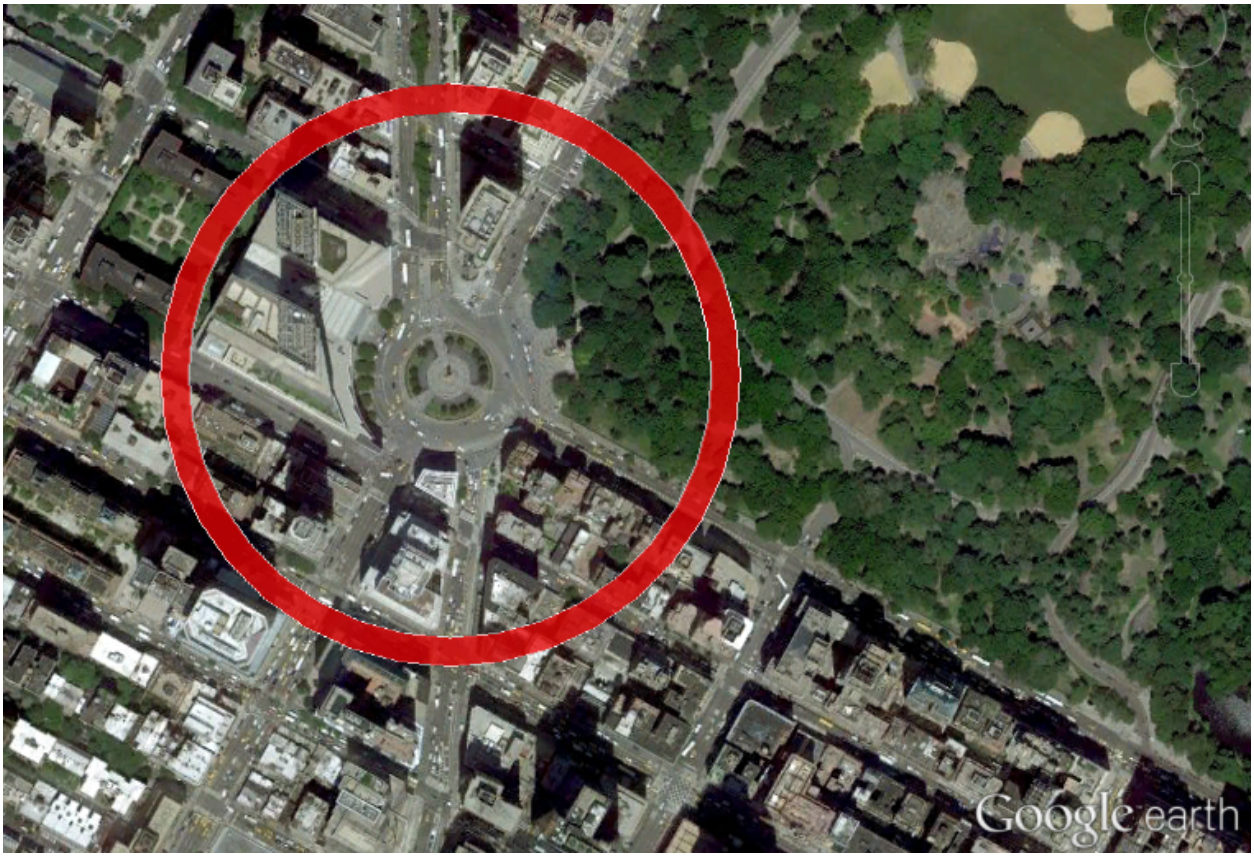

Fig. 3.2: A Torus made from two Polycircles, Columbus Square, Manhattan, NYC, United States
Image: Google Earth, Credit: Google

Or even:

Fig. 3.3: The Olympic logo made from 5 Torus Pplycircles, Copacabana beach, Rio de Janeiro, Brazil
Image: Google Earth, Credit: Google

CHAPTER 4

Accuracy

4.1 Distance accuracy

Polycircle uses Python's geographiclib for distance calculations.

Method	Accuracy
Simple trigonometry	Dozens of meters
geographiclib	10^-4 meters

The distance was measured by geopy.

4.2 How many vertices?

Raising the number of vertices makes the polygon illusion more compelling. On the other side, too many vertices make the KML file larger and Google Earth slower.

In my opinion, 36 edges are the right balance between appearances and file size.

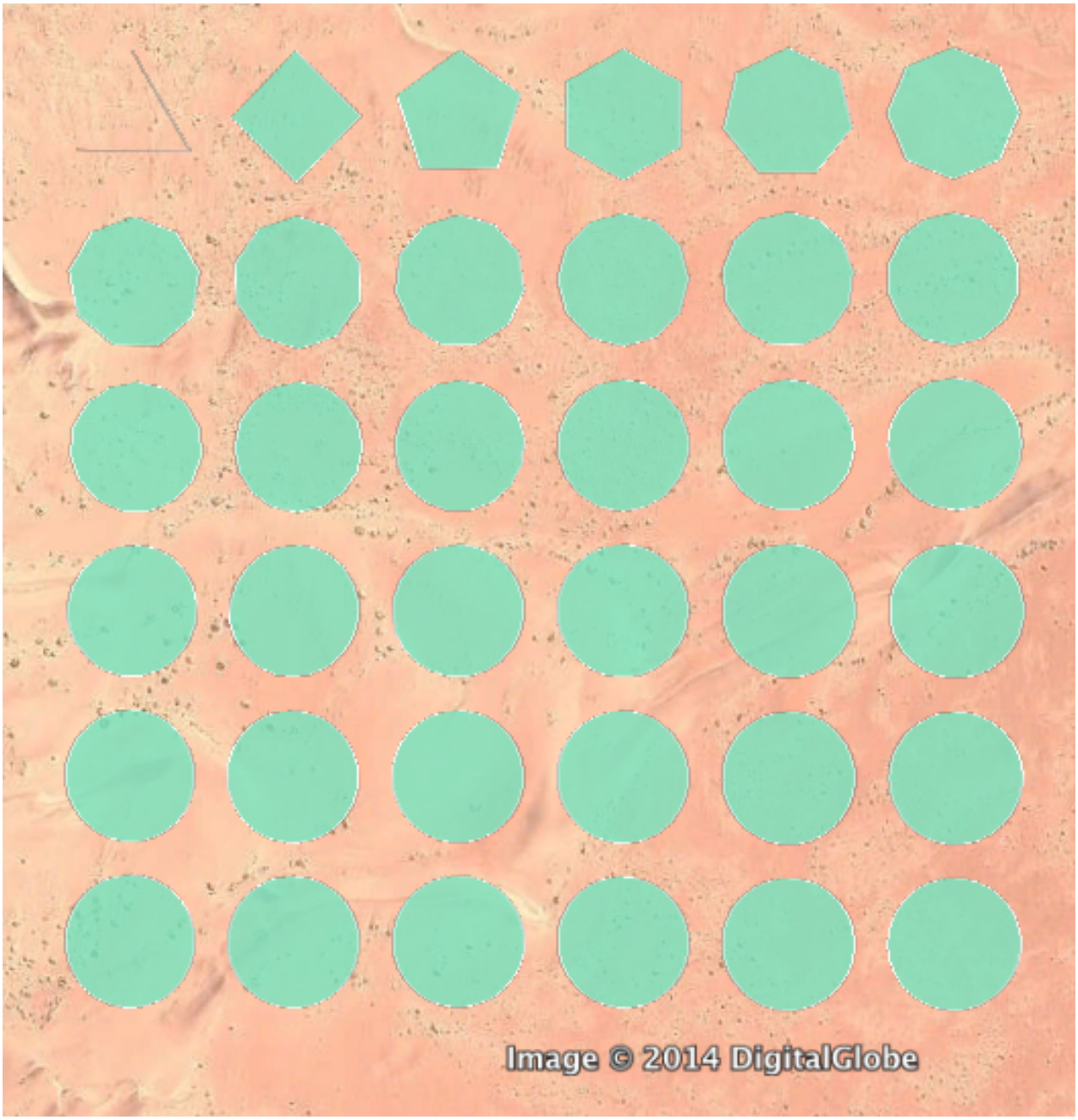

Fig. 4.1: A Polycircle above The Namibian desert, Namibia.
Image: Google Earth, Credit: Google

CHAPTER 5

Development

5.1 Changelog

5.1.1 0.1

- Circle generation
- Tests
- Documentation

5.2 TODO

- Command line script, making Polycircles useful for other languages
- Ellipses
- Simple Torus creation

5.3 Known bugs

- There is no fill color in Google Earth for triangles (polycircles with 3 vertices).

Fig. 5.1: Polygon circle in Google Earth. Map data: Google