Geographic measures in Boost Geometry: length, area and beyond

Vissarion Fisikopoulos

What is Boost.Geometry?

- Part of Boost C++ Libraries
- Header-only
- C++03 (conditionally C++11)
- Metaprogramming, Tags dispatching
- Primitives, Algorithms, Spatial Index
- Standards: OGC SFA

How to Get Started?

- Documentation: `www.boost.org/libs/geometry`
- Mailing list: `lists.boost.org/geometry`
- GitHub: `github.com/boostorg/geometry`

Who is Boost.Geometry?

- Boost.Geometry is an open source project (as any other Boost library)
- Anybody can, and is welcome, to contribute
- Core development team:
  - Barend Gehrels
  - Bruno Lalande
  - Mateusz Loskot
  - Adam Wulkiewicz
  - Menelaos Karavelas
  - Vissarion Fysikopoulos
- Contributions from about a dozen of other developers
- See Boost.Geometry website for credits and GitHub repositorys history

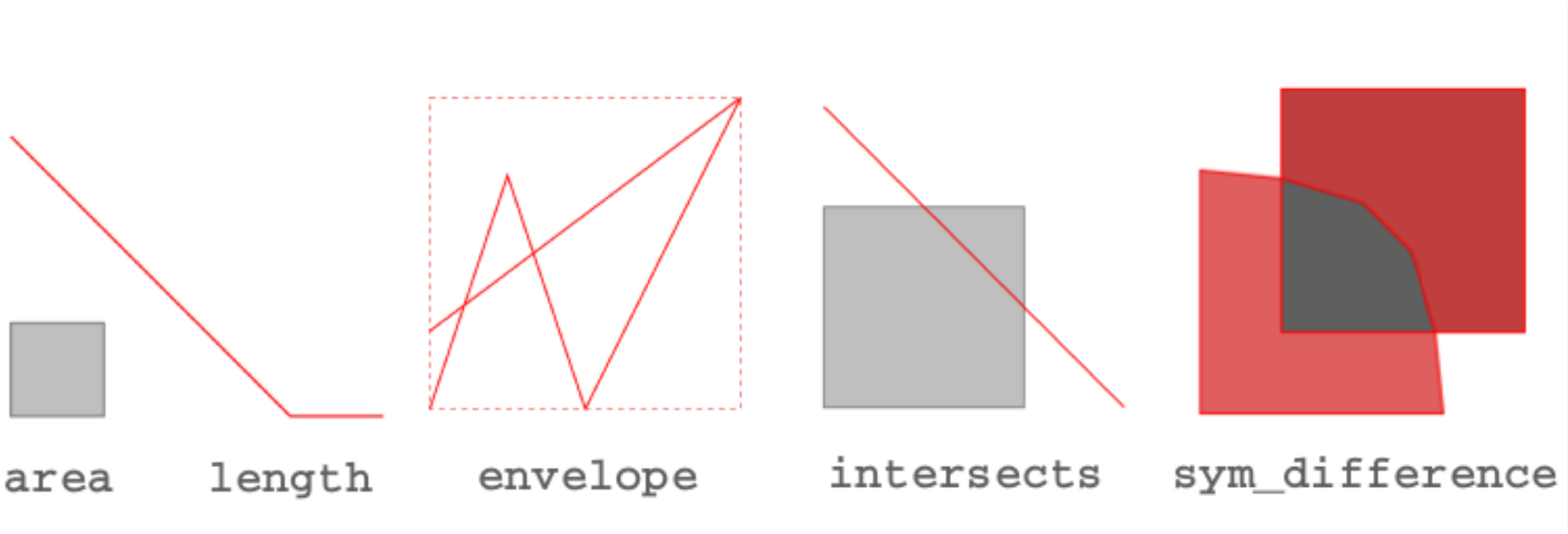
Boost.Geometry & MySQL

- MySQL (since 5.7) relies on Boost geometry for GIS support (geographic support since 8)
- no homegrown set of GIS functions for MySQL
- both aim in OGC standard compliance
- compatible licences
- MySQL benefit from BG open source community (maintenance, bug fixing, gsoc)
- BG is C++/header only $\rightarrow$ no problems with versions of a shared library on different platforms for MySQL

Algorithms in Boost.Geometry

- Creation/modification of geometries
  `append, make, clear, correct, unique, reverse, expand, assign`
- Geometric properties
  `area, centroid, length, num points, num segments, num interior rings, perimeter`
- Set operations
  `difference, intersection, sym difference, union`
- Relational operations and predicates (OGC)
  `crosses, covered by, disjoint, equals, intersects, overlaps, touches, within, relate, is empty, is valid, is simple`
- Other algorithms
  `envelope, convex hull, simplify, distance, buffer, for each, convert, transform`

Examples of algorithms

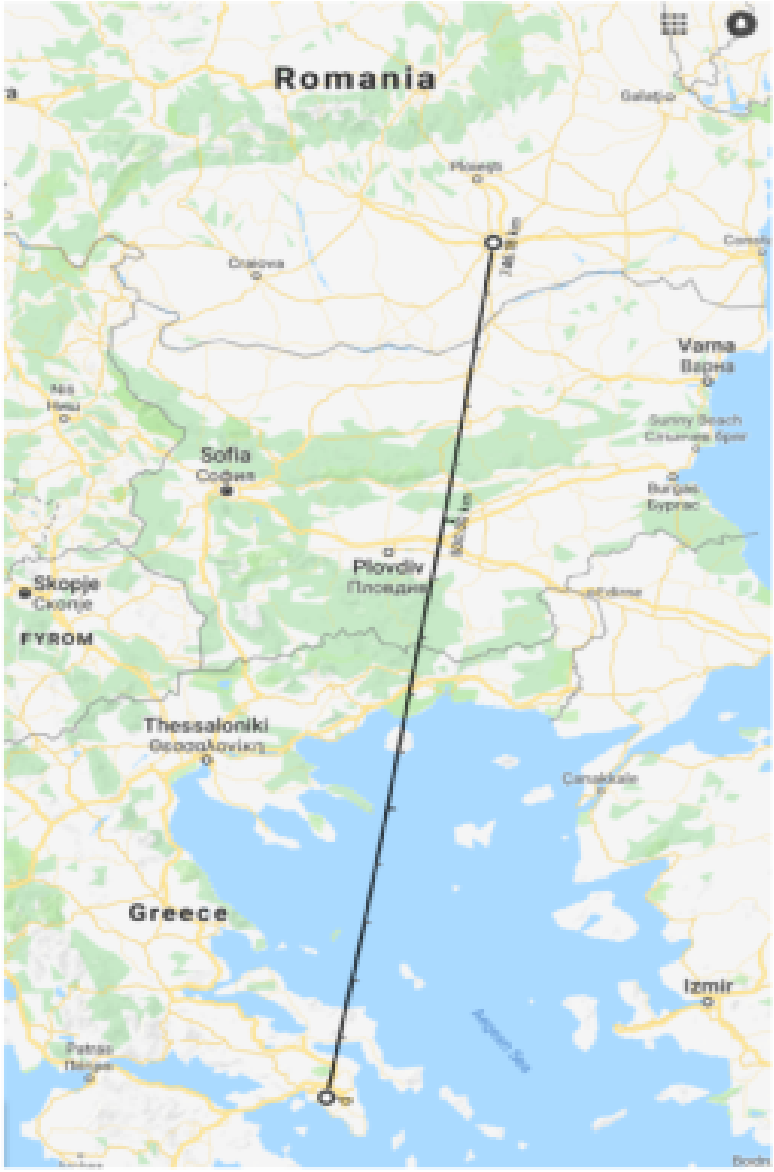


Hello, world!

```cpp
#include <boost/geometry.hpp>
#include <boost/geometry/geometries/geometries.hpp>
#include <iostream>

namespace bg = boost::geometry;

int main() {
  using point_geo =
    bg::model::point<double, 2, bg::cs::geographic<bg::degree>>;

  //Athens, Acropolis
  point_geo athens(23.725750, 37.971536);
  //Bucharest, FOSS4G
  point_geo bucharest(26.102347, 44.43555);

  std::cout << bg::distance(athens, bucharest) << std::endl;
}
```

Hello World! result

```
Output:  744954 m
```



maps.google.com

Boost Geometry's 3-layer design

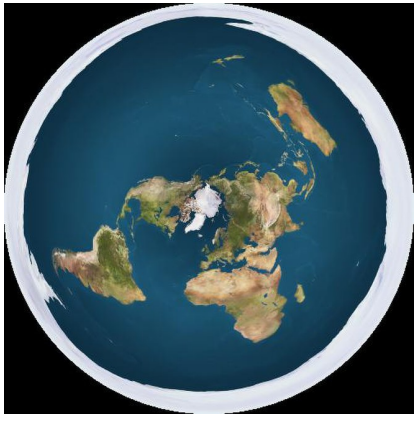
- BG exploits namespaces:
  - `boost::geometry`
  - `boost::geometry::dispatch`
  - `boost::geometry::detail`
- Namespace `boost::geometry` contains available algorithms and some utilities
- Namespace `boost::geometry::dispatch` contains the basic tag-dispatching classes
- Namespace `boost::geometry::detail` has the implementation details

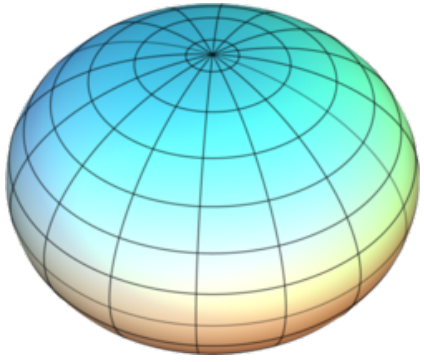
Algorithms & strategies

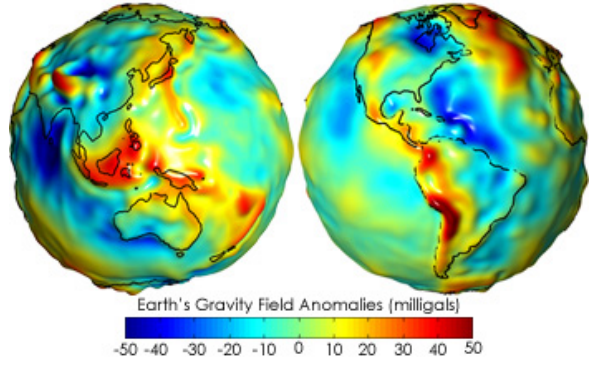
- All algorithms are template free functions; input is geometries and possibly strategies
- Each algorithm performs concept checking on input; then calls dispatch class; dispatching based on tags
- Strategies are coordinate system specific algorithms

Models of the Earth & coordinate systems

- Flat
  `boost::geometry::cs::cartesian`
- Sphere (*Widely used e.g.* `google.maps`)
  `boost::geometry::cs::spherical_equatorial<bg::degree>`
  `boost::geometry::cs::spherical_equatorial<bg::radian>`
- Ellipsoid of revolution (*geographic GIS state-of-the-art*)
  `boost::geometry::cs::geogrphic<bg::degree>`
  `boost::geometry::cs::geogrphic<bg::radian>`
- Geoid (*Special applications, geophysics etc*)







Distance computation

- Vincenty (state-of-the-art iterative method - geographic strategy)
- Thomas (series approximation order 2 - geographic strategy)
- Andoyer (series approximation order 1 - geographic strategy)
- Higher series approximation (up-to order 8 - geographic strategy) *
- Haversine (trigonometric approx. - spherical_equatorial strategy)
- Projection + cartesian strategy

* Pull request: https://github.com/boostorg/geometry/pull/431

Distance computation with strategies

```
#include <boost/geometry.hpp>
#include <boost/geometry/geometries/geometries.hpp>
#include <iostream>

namespace bg = boost::geometry;

int main() {
  using point_geo =
    bg::model::point<double, 2, bg::cs::geographic<bg::degree>>;

  //Athens, Acropolis
  point_geo athens(23.725750, 37.971536);
  //Bucharest, FOSS4G
  point_geo bucharest(26.102347, 44.43555);

  std::cout << bg::distance(athens, bucharest,
                            bg::strategy::distance::vincenty<>() )
            << std::endl;
}
```

Output: 744954 m

Output (with vincenty strategy): 744947 m (7m difference)

Distance between polygons

`maps.google.com`

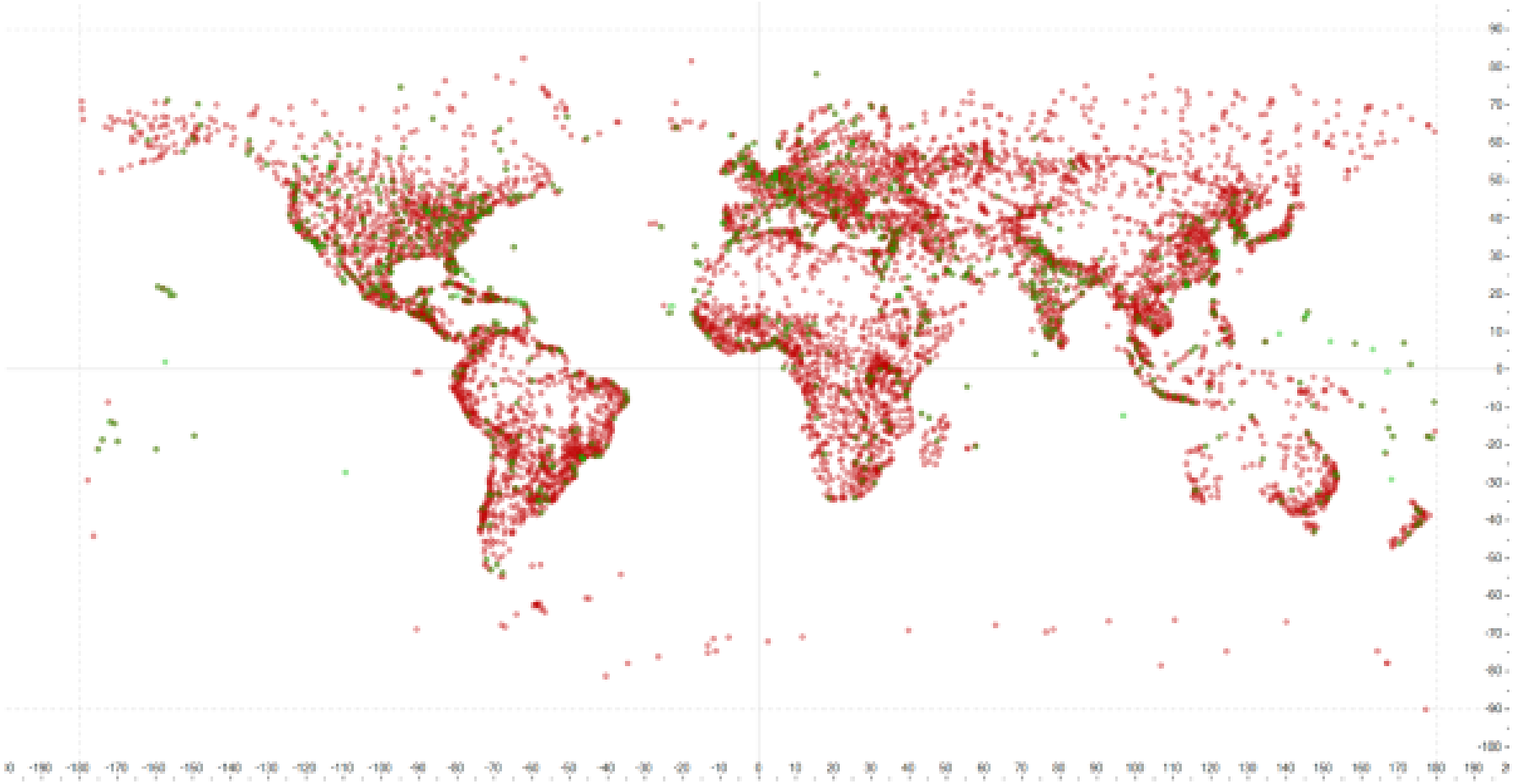
Distance between polygons

```
namespace bg = boost::geometry;

using point_geo = bg::model::point<double, 2, bg::cs::geographic<bg::degree>>;
using polygon_geo = bg::model::polygon<point_geo>;

polygon_geo gr, ro;

// fusiontables.google.com/data?docid=1zn8cjdD6qlAFI7ALMEnwn89g50weLi1D bAGSZw
bg::read_wkt("POLYGON((23.648809225 38.355682203, ..., 23.648809225 38.355682203))",
             gr);
bg::read_wkt("POLYGON((22.681436175 44.224700043, ..., 22.681436175 44.224700043))",
             ro);

bg::strategy::distance::geographic_cross_track< bg::strategy::vincenty > str;

std::cout << bg::distance(gr, ro, str) << std::endl;
```

Output: 218937 m

Output (with vincenty strategy): 218935 m (2m difference)

Distance point to multi-point and between multi-points



Distance point to multi-point

```
namespace bg = boost::geometry;

using point_geo = bg::model::point<double, 2,
                                   bg::cs::geographic<bg::degree>>
using multi_point_geo = bg::model::multi_point<point_geo>;

// www.naturalearthdata.com/downloads/10mculturalvectors
std::ifstream ifs2("ne_10m_airports.shp", std::ifstream::binary);

multi_point_geo airports;

bg::read_shapefile(ifs2, airports);

std::cout << bg::distance(point_geo(26.094497, 44.441609),
                          airports) << std::endl;
```

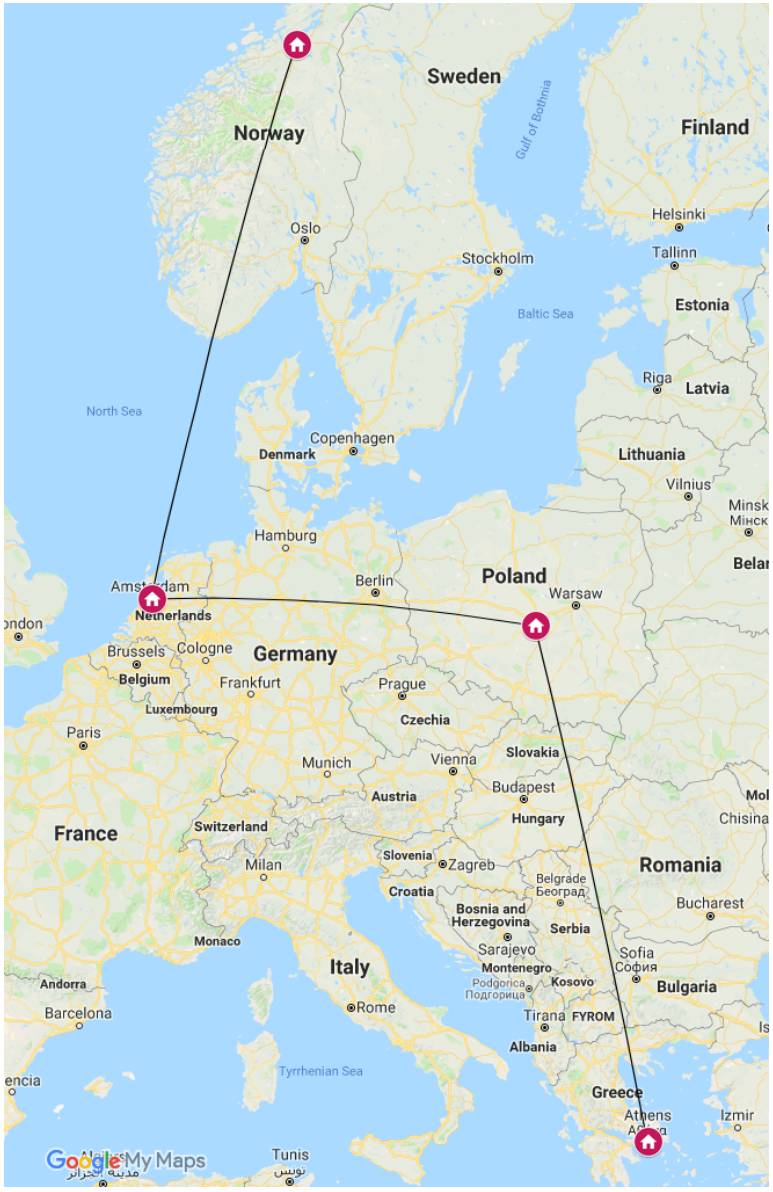
Output: 6199.21 m

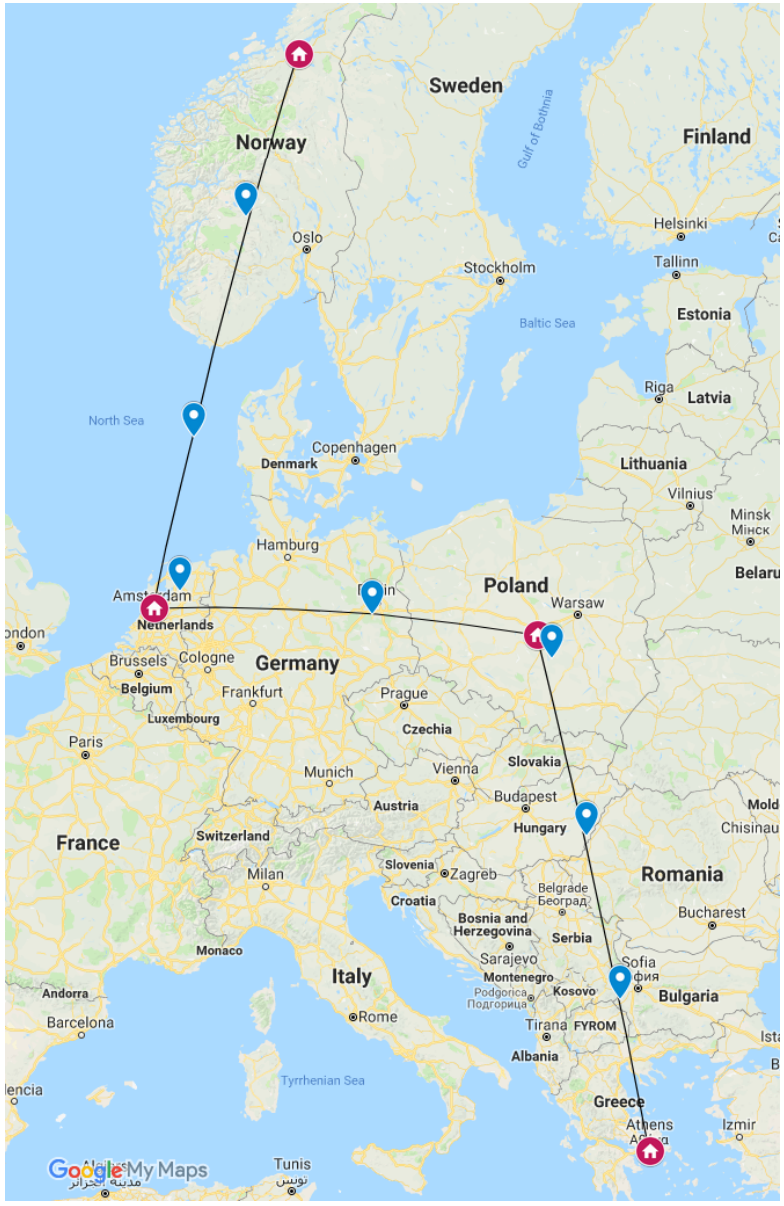
Distance between multi-points

```
namespace bg = boost::geometry;
using point_geo = bg::model::point<double, 2,
                                   bg::cs::geographic<bg::degree>>;
using multi_point_geo = bg::model::multi_point<point_geo>;

// www.naturalearthdata.com/downloads/10mculturalvectors
std::ifstream ifs1("ne_10m_populated_places.shp", std::ifstream::binary);

std::ifstream ifs2("ne_10m_airports.shp", std::ifstream::binary);
multi_point_geo populated_places, airports;

bg::read_shapefile(ifs1, populated_places);
bg::read_shapefile(ifs2, airports);

std::cout << bg::distance(populated_places, airports) << std::endl;
```

Output: 427.955 m

Line interpolate points





Line interpolate points

```
namespace bg = boost::geometry;
typedef bg::model::point<double, 2,
                         bg::cs::geographic<bg::degree> > point;

using linestring_type = bg::model::linestring<point>;
using multipoint_type = bg::model::multi_point<point>;

linestring_type l {{23.7275, 37.9838}, //Athens
                   {19.4560, 51.7592}, //Lodz
                   {4.9036, 52.3680}, //Amsterdam
                   {10.3951, 63.4305}}; //Trondheim
multipoint_type mp;

bg::line_interpolate_point(l,500000.0,mp);
std::cout << wkt(mp) << std::endl;
```

Output: MULTIPOINT((22.57502513 42.39930172),(21.33184367

46.81043626),(19.97378916 51.21703093),(13.19702028

52.24398028),(5.88051868 52.7698247),(6.416288309

56.16967713),(8.386069808 60.53816825))

Area

Belgium

Example 4: Area

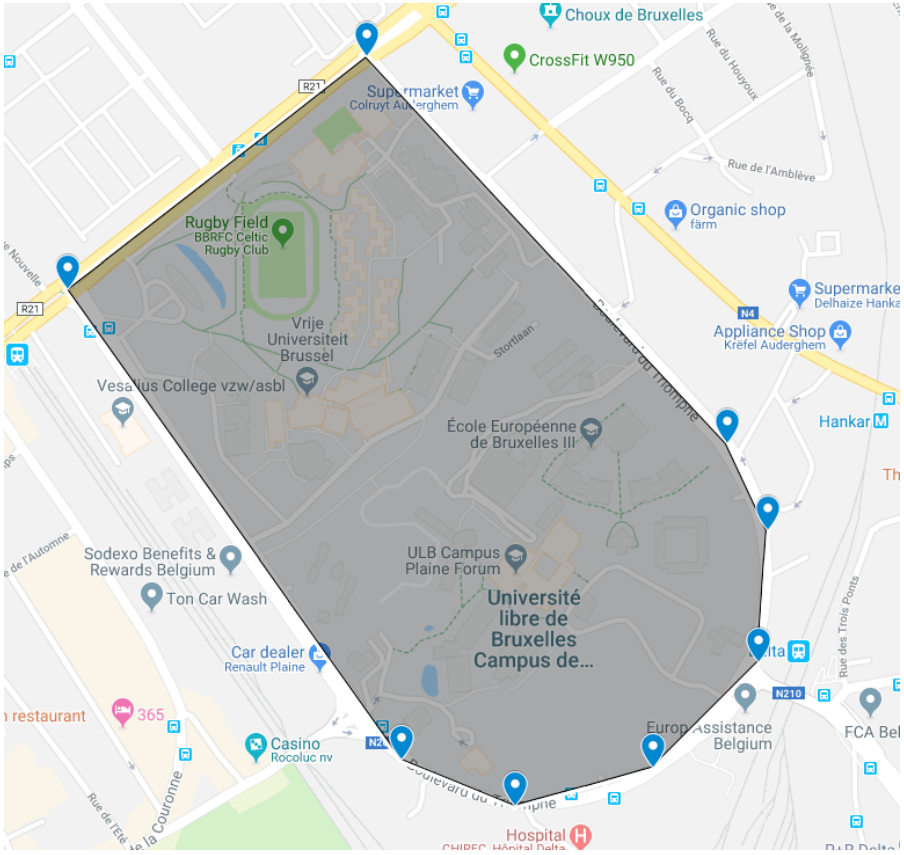
Belgium

```
namespace bg = boost::geometry;
typedef bg::model::point<double, 2,
                         bg::cs::geographic<bg::degree> > point;
bg::model::polygon<point> poly;

bg::read_wkt("POLYGON((2.54166319863 51.0911090428 3.37086325679 51.373854107 \
                3.43986325156 51.2447821855 3.89541801404 51.2056910007 \
                4.23890019286 51.3504270229 4.32770008701 51.290127276 \
                4.25236303814 51.3751450878 5.03847307305 51.4869450919 \
                5.23897322671 51.2622820907 5.84713612726 51.1531911176 \
                5.63881817249 50.8488820987 6.01180012435 50.7572730935 \
                6.27041821549 50.6198541375 6.39820016212 50.3231729857 \
                6.13440926324 50.1278451369 5.9730542739 50.170000075 \
                5.74777319107 49.9074911044 5.89916310333 49.6627732262 \
                5.80788233348 49.5450452048 5.47278222549 49.508882154 \
                4.86847332403 49.8022182559 4.82472716541 50.1675641263 \
                4.51055415514 49.9474912228 4.1492361245 49.9783731717 \
                4.16500013748 50.2830541795 3.29708205679 50.5243001057 \
                3.15857311809 50.7843642428 2.65055417947 50.8161090248 \
                2.54166319863 51.0911090428))", poly);
std::cout << bg::area(poly, bg::strategy::area::geographic
                           <
                              bg::strategy::vincenty
                           >()) << std::endl;
```

andoyer	30822.6163 km^2
thomas	30819.5846 km^2
vincenty	30819.5844 km^2
series	30875.8950 km^2

Area

ULB


Choux de Bruxelles
CrossFit W950
Rue du Houyoux
Rue du Bocq
Rue de l'Amblève
Organic shop
ferm
Supermarke
Delhaize Hanka
Appliance Shop
Krëfel Auderghem
Hankar
Rugby Field
BBRFC Celtic Rugby Club
Vrije Universiteit Brussel
Stortlaan
Vesalius College vzw/asbl
École Européenne de Bruxelles III
Sodexo Benefits & Rewards Belgium
ULB Campus Plaine Forum
Ton Car Wash
Université libre de Bruxelles Campus de...
Car dealer
Renault Plaine
restaurant
365
Casino
Rocoluc nv
Europ Assistance Belgium
FCA Bel
Rue des Trois Ponts
Hospital
R21
N4
N210
N2

Area

ULB

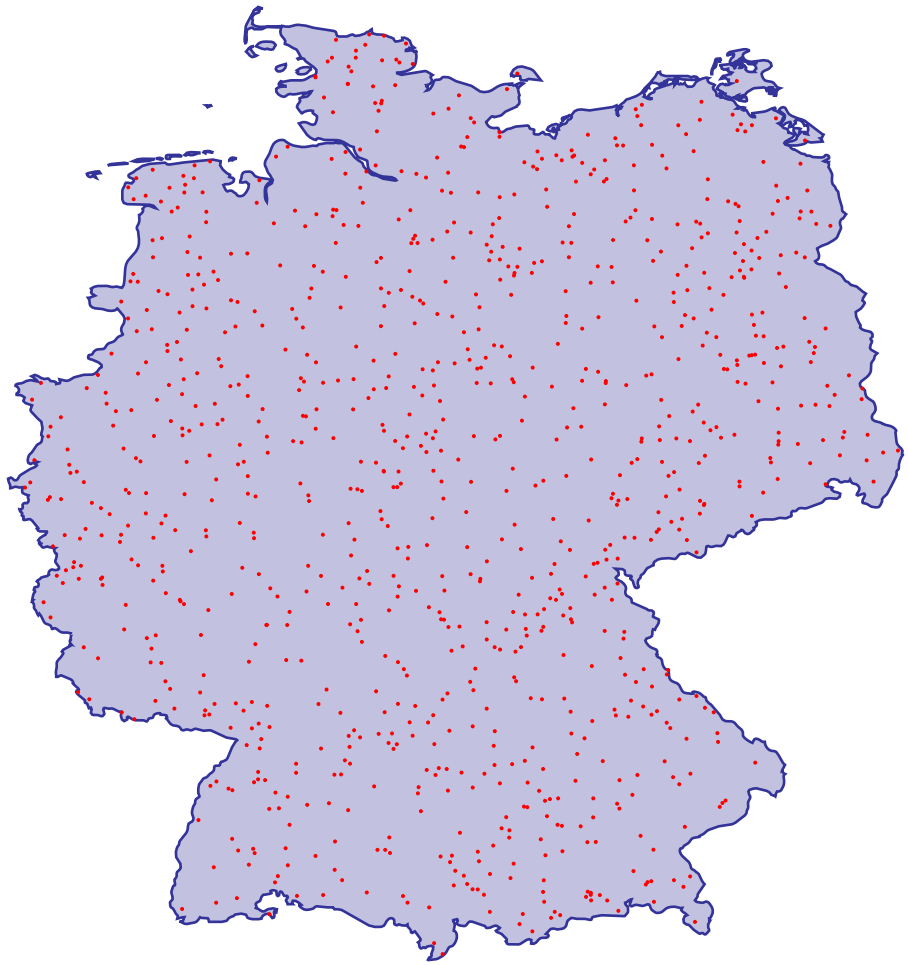
```
namespace bg = boost::geometry;
typedef bg::model::point<double, 2,
                         bg::cs::geographic<bg::degree> > point;

bg::model::polygon<point> poly;
bg::read_wkt("POLYGON((4.395971 50.8256969\
                4.4026444 50.8211695\
                4.4033954 50.8201528\
                4.4032237 50.8186074\
                4.4012711 50.8173737\
                4.3987176 50.8169264\
                4.3966148 50.8174686\
                4.3904349 50.8229724\
                4.395971 50.8256969))", poly);
  std::cout << bg::area(poly, bg::strategy::area::geographic
                                <
                                  bg::strategy::vincenty
                                >()) << std::endl;
```

andoyer	493001 m^2
thomas	495767 m^2
vincenty	495800 m^2
series	494520 m^2

Random point generators & Voronoi diagrams

- GSOC 2019 project
- student: Tinko Bartels, Germany
- mentors: V.F., Adam Wulkiewicz

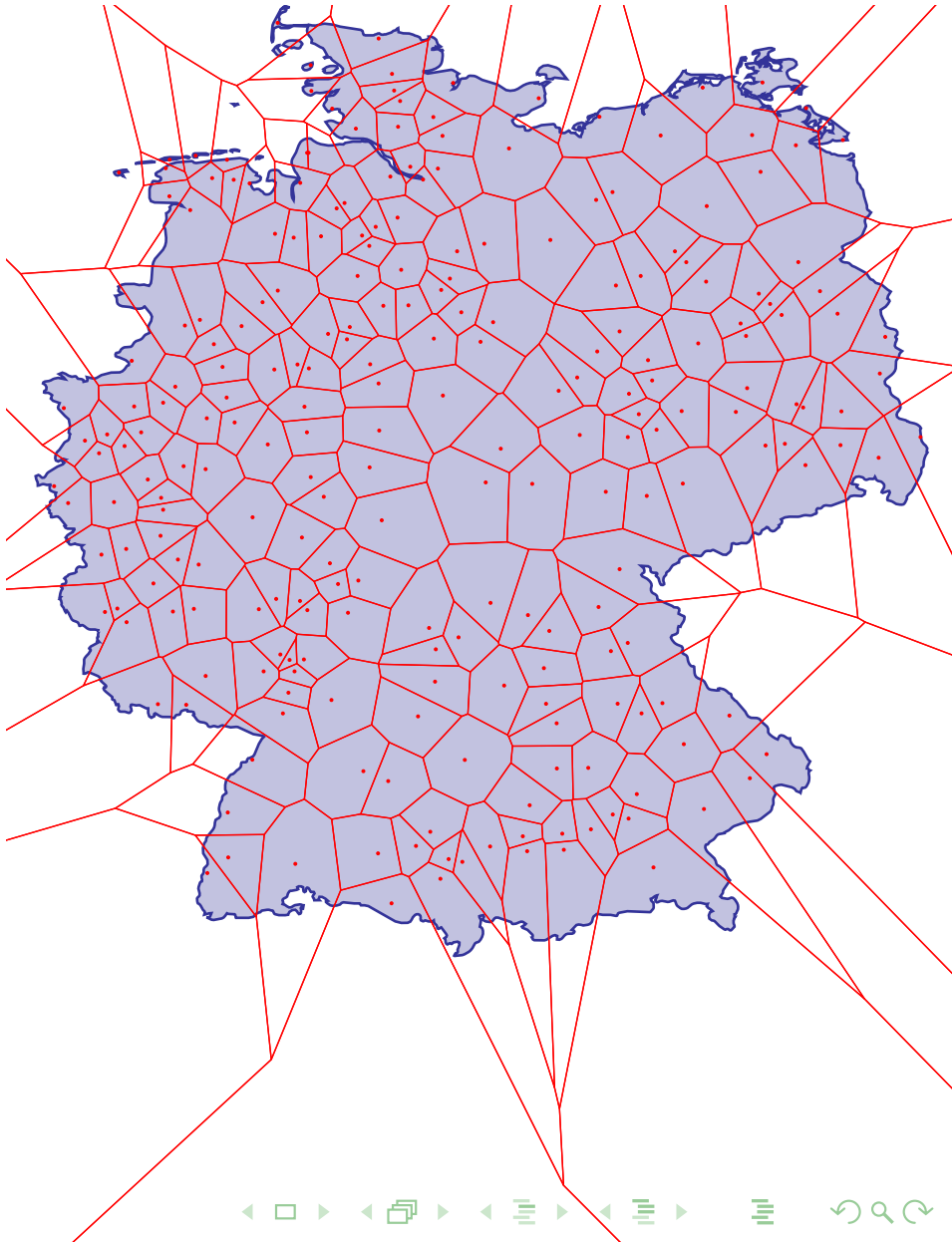
Google Summer of Code



Random point generators & Voronoi diagrams

- GSOC 2019 project
- student: Tinko Bartels, Germany
- mentors: V.F., Adam Wulkiewicz
- dataset: Germany airports `https://github.com/jpatokal/openflights/blob/master/data/airports.dat`

Google Summer of Code



Thank you!

Questions?