UFR de mathématique et d'informatique

Université de Strasbourg

# Exa-MA WP1 - Terrain

Giulio CARPI LAPI

August 23, 2024

# Introduction

- Internship took place at **Cemosis - IRMA**
- **PEPR Numpex** through **Exa-MA** project
- Part of the **HiDALGO2** project
- **Ktirio Urban Building application**
- Supervised by **Vincent Chabannes**


Figure: cemosis logo [1]


Figure: Numpex logo [2]


Figure: HiDALGO2 logo [3]

# Introduction: Main objectives

- **Exa-MA WP1 - Terrain**

- **Exa-MA WP1 - Vegetation**
- **Exa-MA WP1 - Urban Building LOD-1**
- **Exa-MA WP1 - Urban Building LOD-2**

- Develop a **detailed 3D model** of the urban landscape

# Introduction: Main objectives

- Source elevation data from **Mapbox**
- **Reduce mesh density** with **contour line constraints**
- **Parallelization**

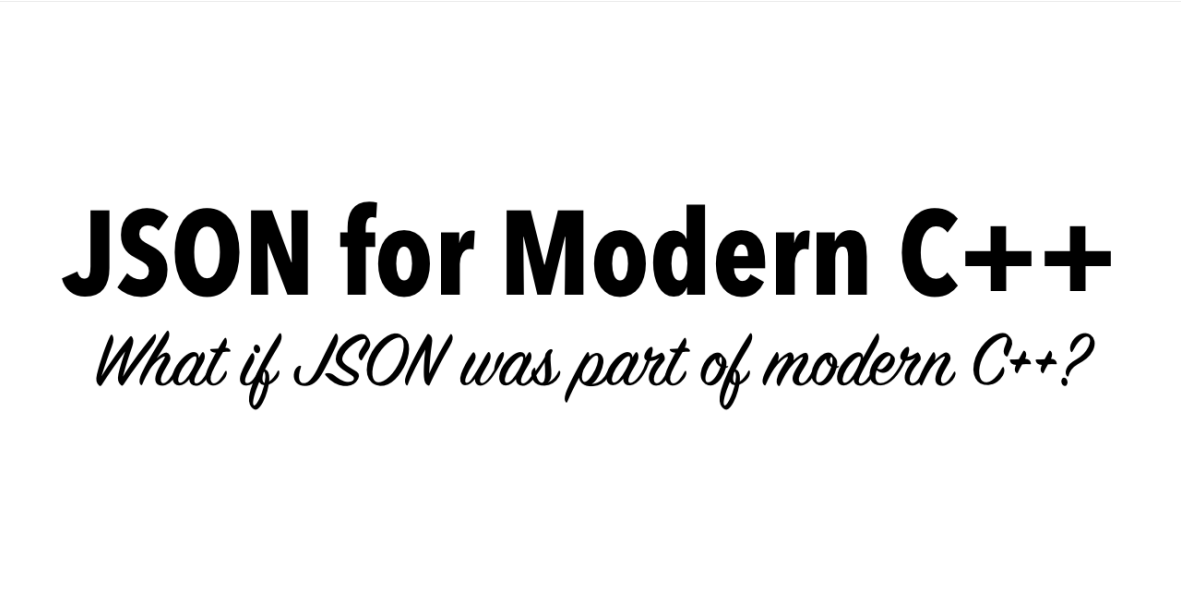


Figure: json nlohmann logo[4]

C++ library for **JSON** parsing

Figure: curl logo[5]

**URL** transfer library

# Introduction: Software and libraries

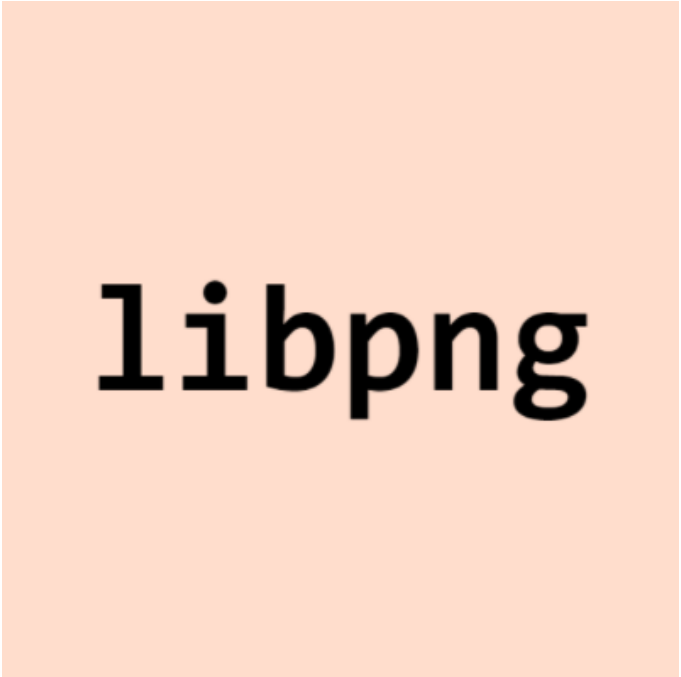


Figure: libpng logo[6]

**PNG** reference library

# Introduction: Software and libraries

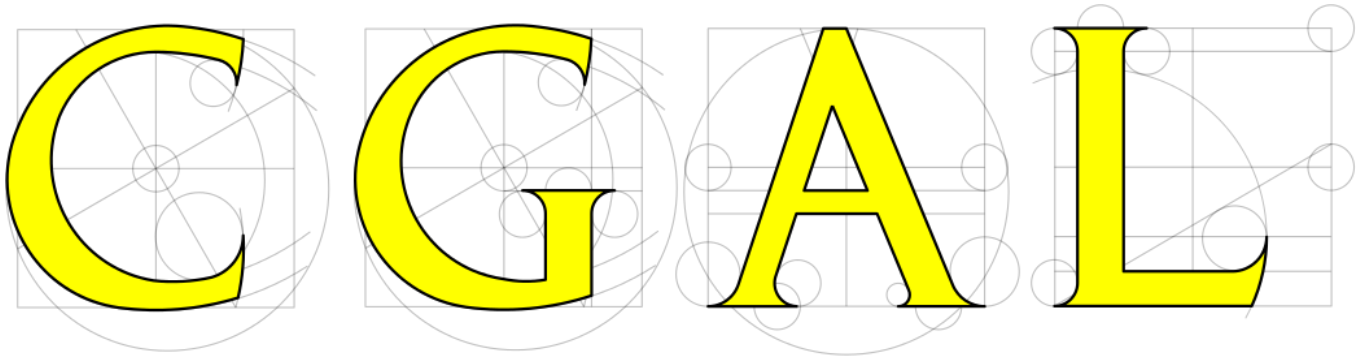


Figure: CGAL logo[7]

Open source software library for **computational geometry algorithms**

# Methodology: Pre-existing codebase

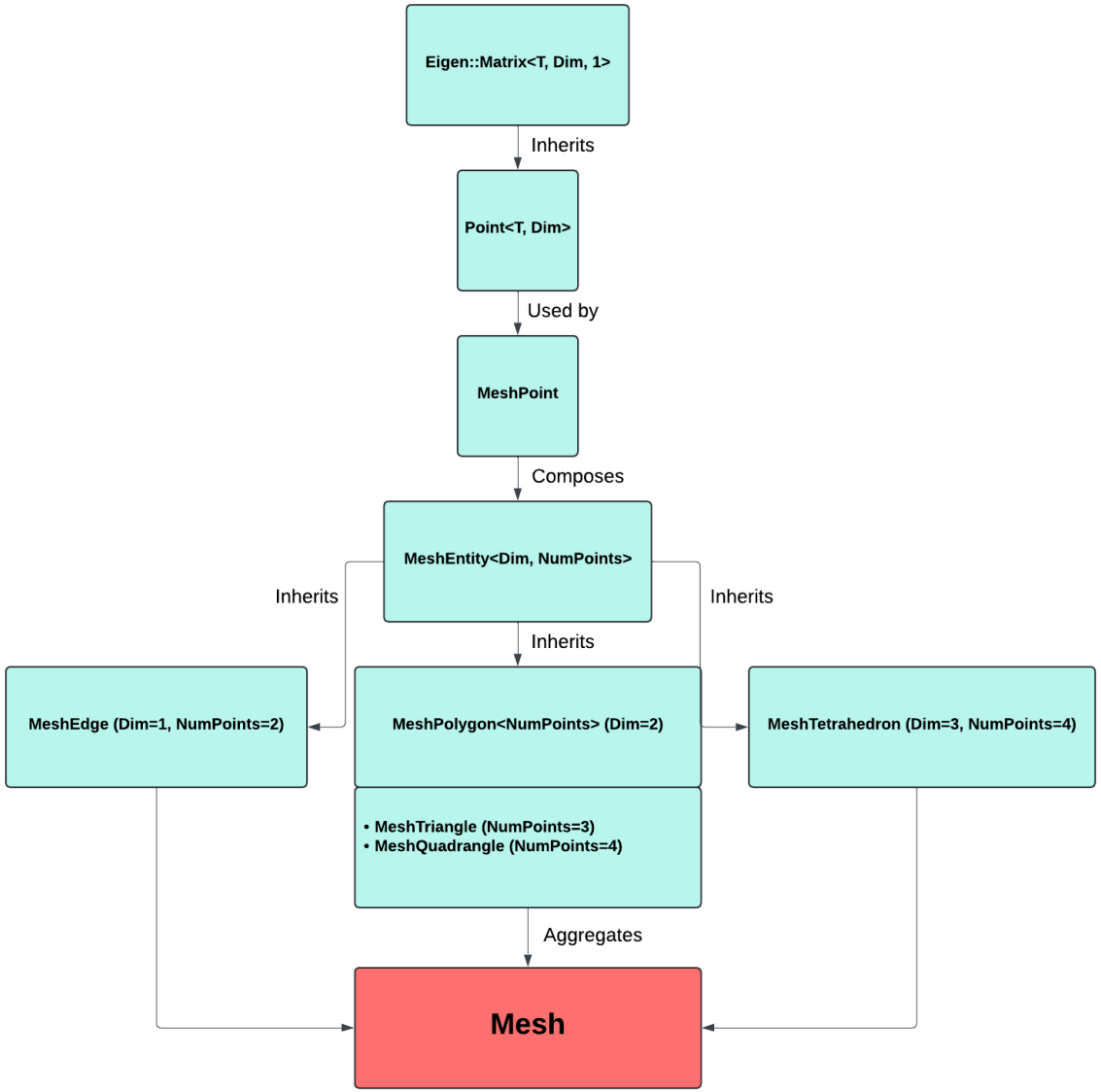


Figure: UML diagram of the codebase structure 1

# Methodology: Pre-existing codebase

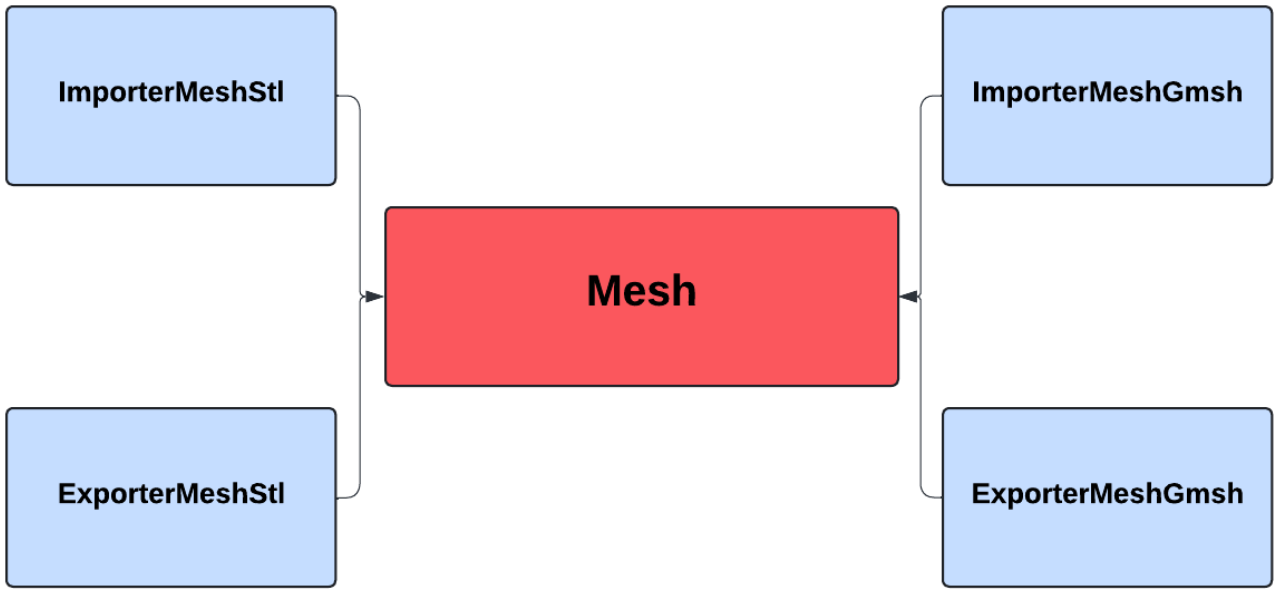


Figure: UML diagram of the codebase structure 2

# Methodology: Configuration

```
{
  "useGPS": true,
  "precision": 0.5,
  "longitude": 5.7232,
  "latitude": 45.1835,
  "zoom": 16,
  "api_key": "[YOUR_API_KEY_HERE]"
}

```

# Methodology: Latitude and longitude

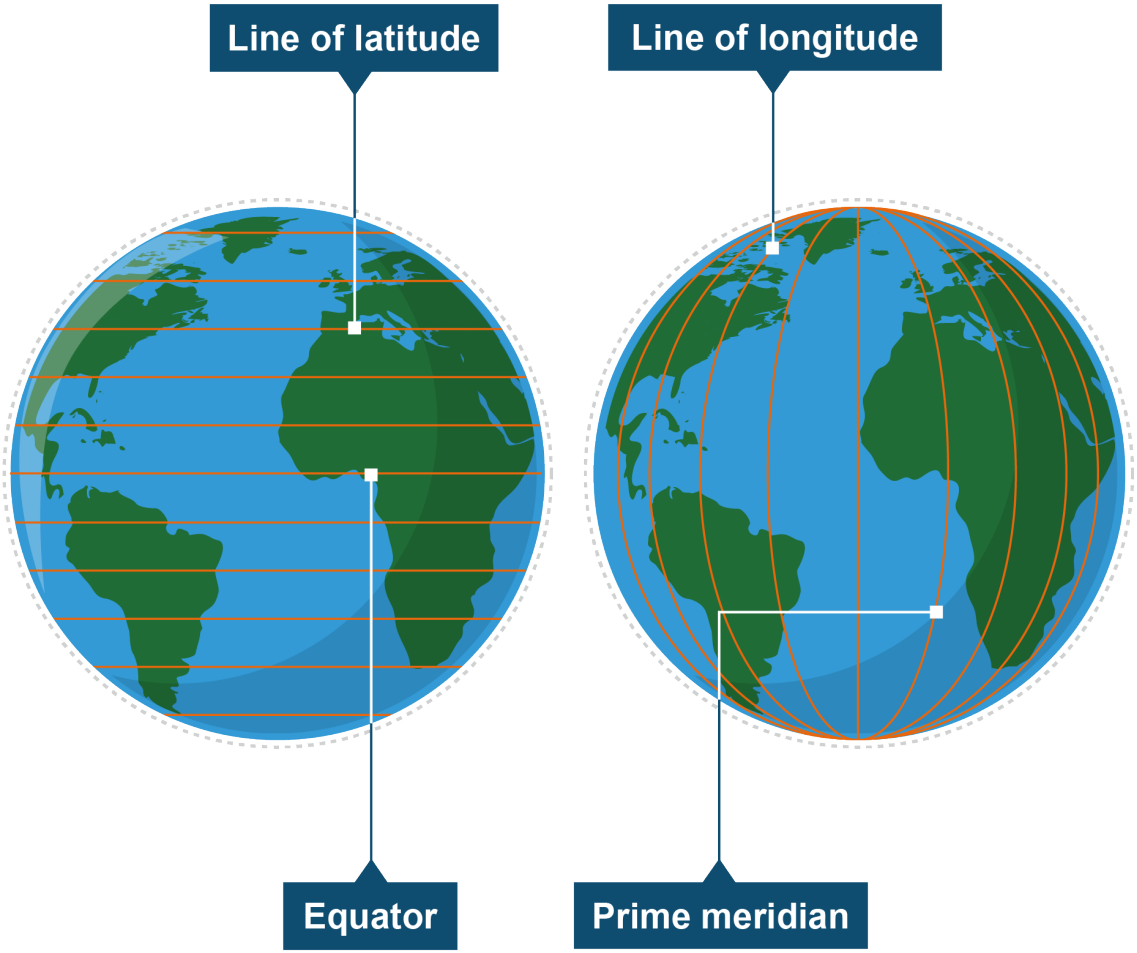


Figure: Latitude and longitude [8]

# Methodology: Mercator projection

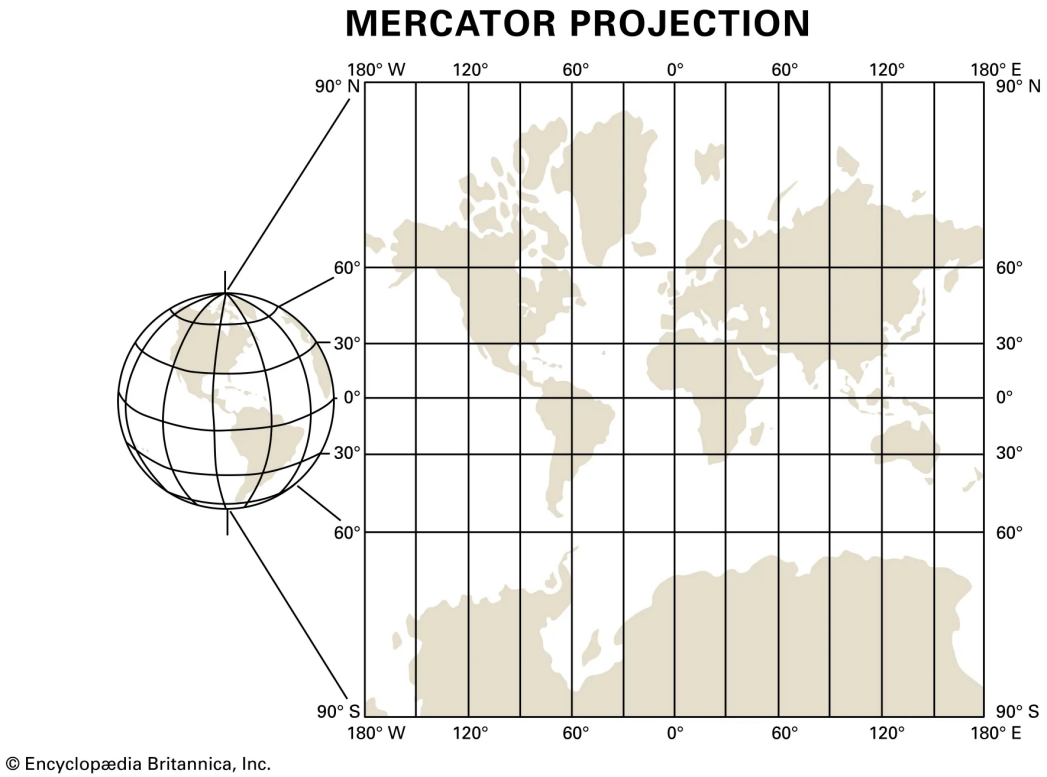


Figure: Mercator projection [9]

# Methodology: Tiled web map

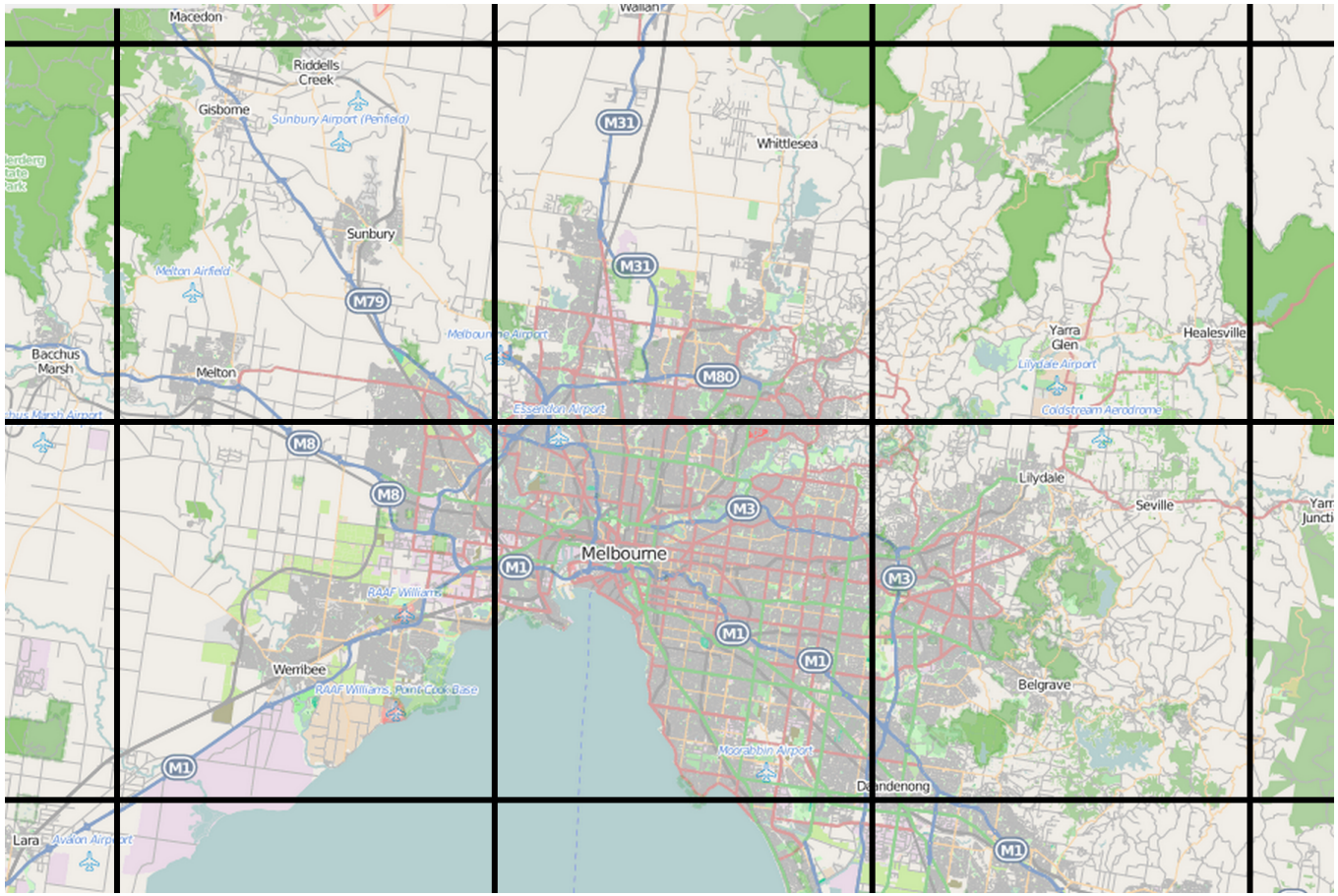


Figure: Tiled web map [10]

# Methodology: Tiles coordinates

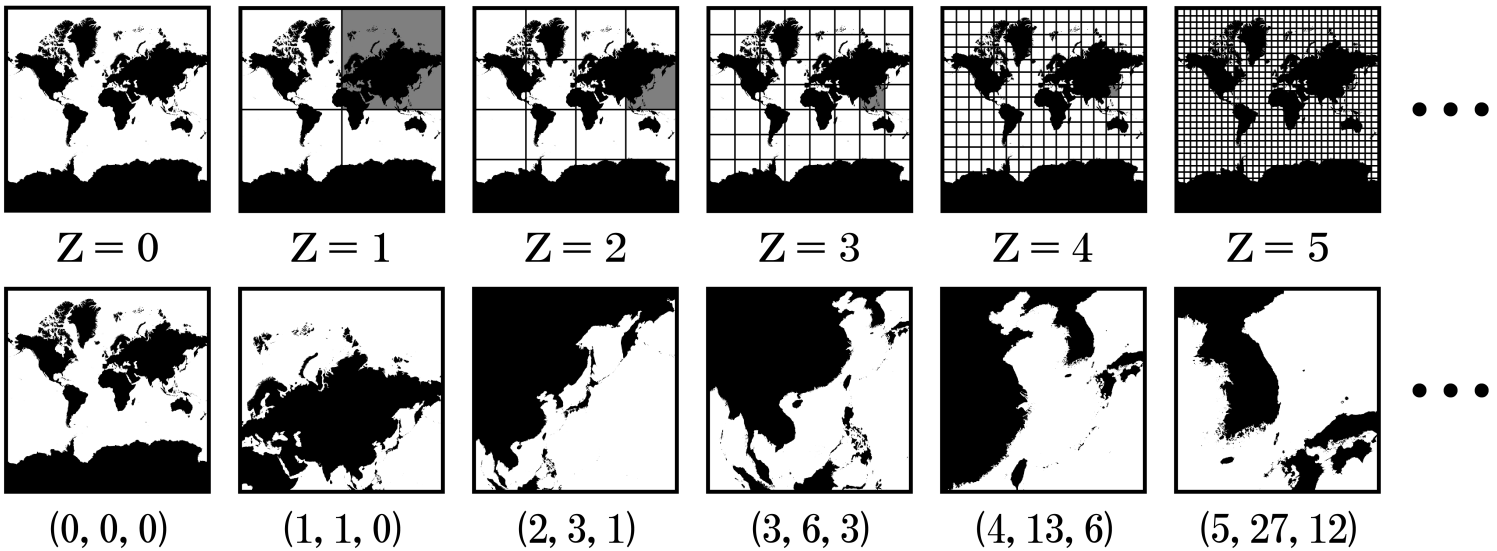


Figure: Tiles coordinates [11]

# Methodology: Data acquisition

URL structure for **Mapbox Terrain-RGB v1** API:

```
1 "https://api.mapbox.com/v4/mapbox.terrain-rgb/" + std::
      to_string(zoom) + "/" + std::to_string(x) + "/" + std::
      to_string(y) + ".png?access_token=" + api_key;
2
```

# Methodology: Geographic coordinates to tile coordinate

$$x = \left\lfloor \frac{\text{lon} + 180}{360} \times 2^{\text{zoom}} \right\rfloor$$

$$y = \left\lfloor \left( 1 - \frac{\log\left( \tan\left( \frac{\text{lat} \times \pi}{180} \right) + \frac{1}{\cos\left( \text{lat} \times \frac{\pi}{180} \right)} \right)}{\pi} \right) \times 2^{\text{zoom}} \right\rfloor$$

## Function to read PNG data:

```
1  std::vector<std::vector<std::tuple<int, int, int>>> readPNG(
       const std::vector<char> &data)
2  {
3    std::vector<std::vector<std::tuple<int, int, int>>> image;
4    std::vector<std::tuple<int, int, int>> row;
5    for (unsigned x = 0; x < image_struct.width; ++x)
6    {
7        int r = buffer[3 * (y * image_struct.width + x) + 0];
8        int g = buffer[3 * (y * image_struct.width + x) + 1];
9        int b = buffer[3 * (y * image_struct.width + x) + 2];
10       row.emplace_back(r, g, b);
11   }
12     return image;
13 }
14
```

# Methodology: Mapbox Terrain-RGB v1

$$height = -10000 + ((R \times 256 \times 256 + G \times 256 + B) \times 0.1)$$

# Methodology: Delaunay triangulation

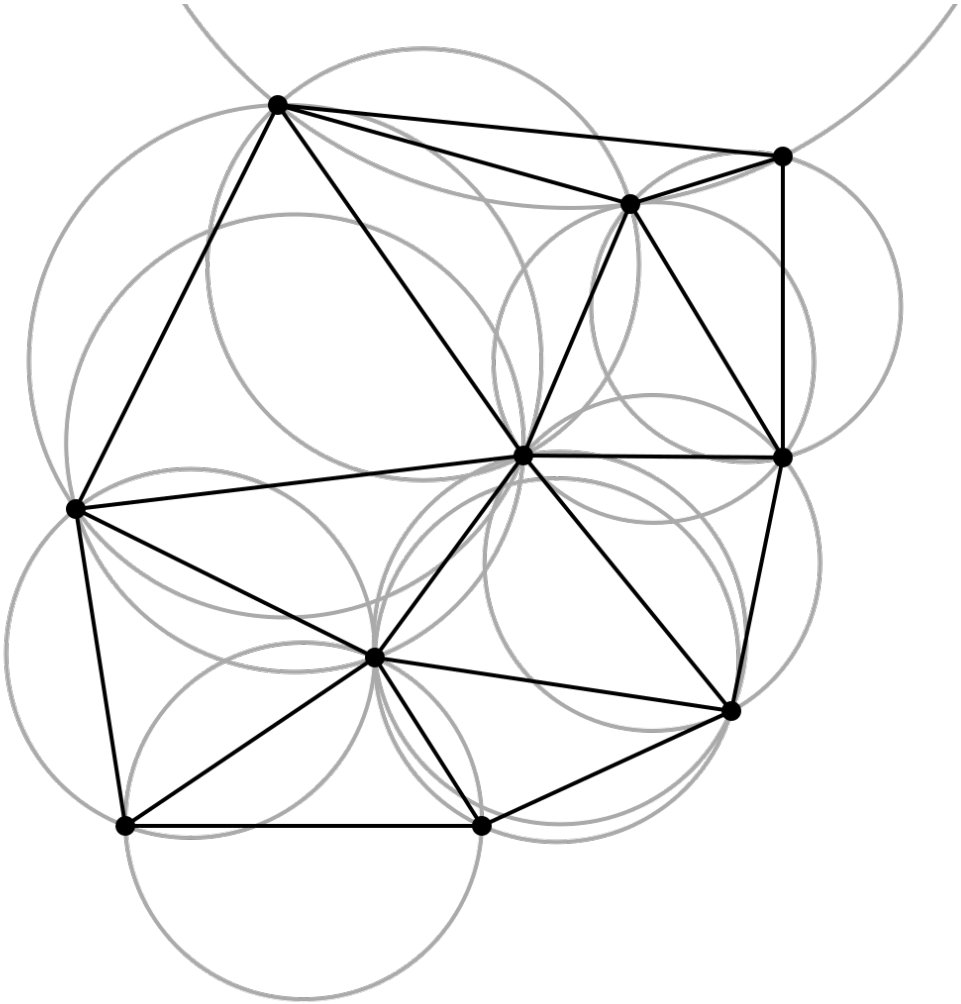

Figure: Delaunay triangulation [12]

# Methodology: Constrained Delaunay triangulation

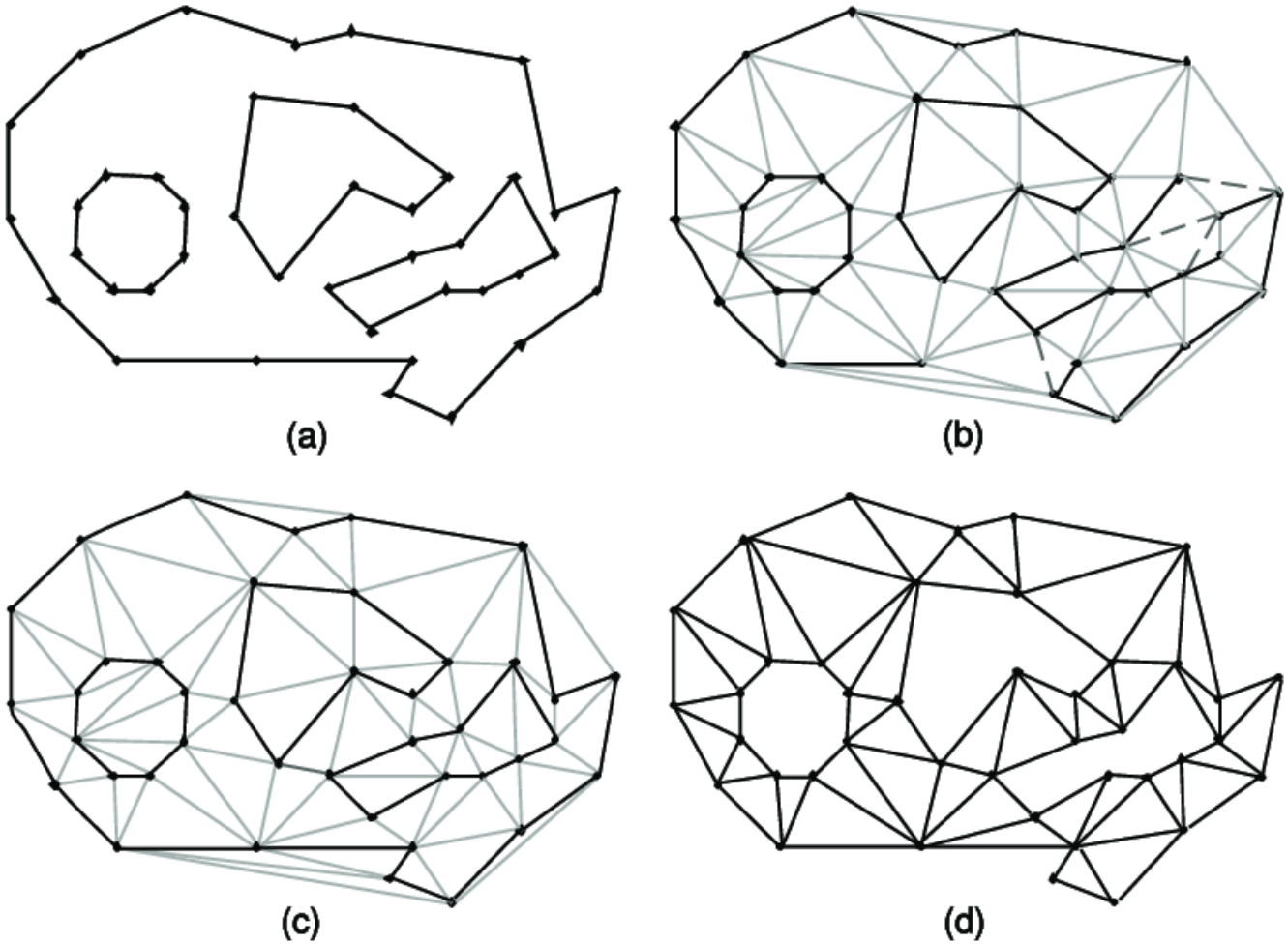


Figure: Constrained Delaunay triangulation [13]

# Implementation: Project structure

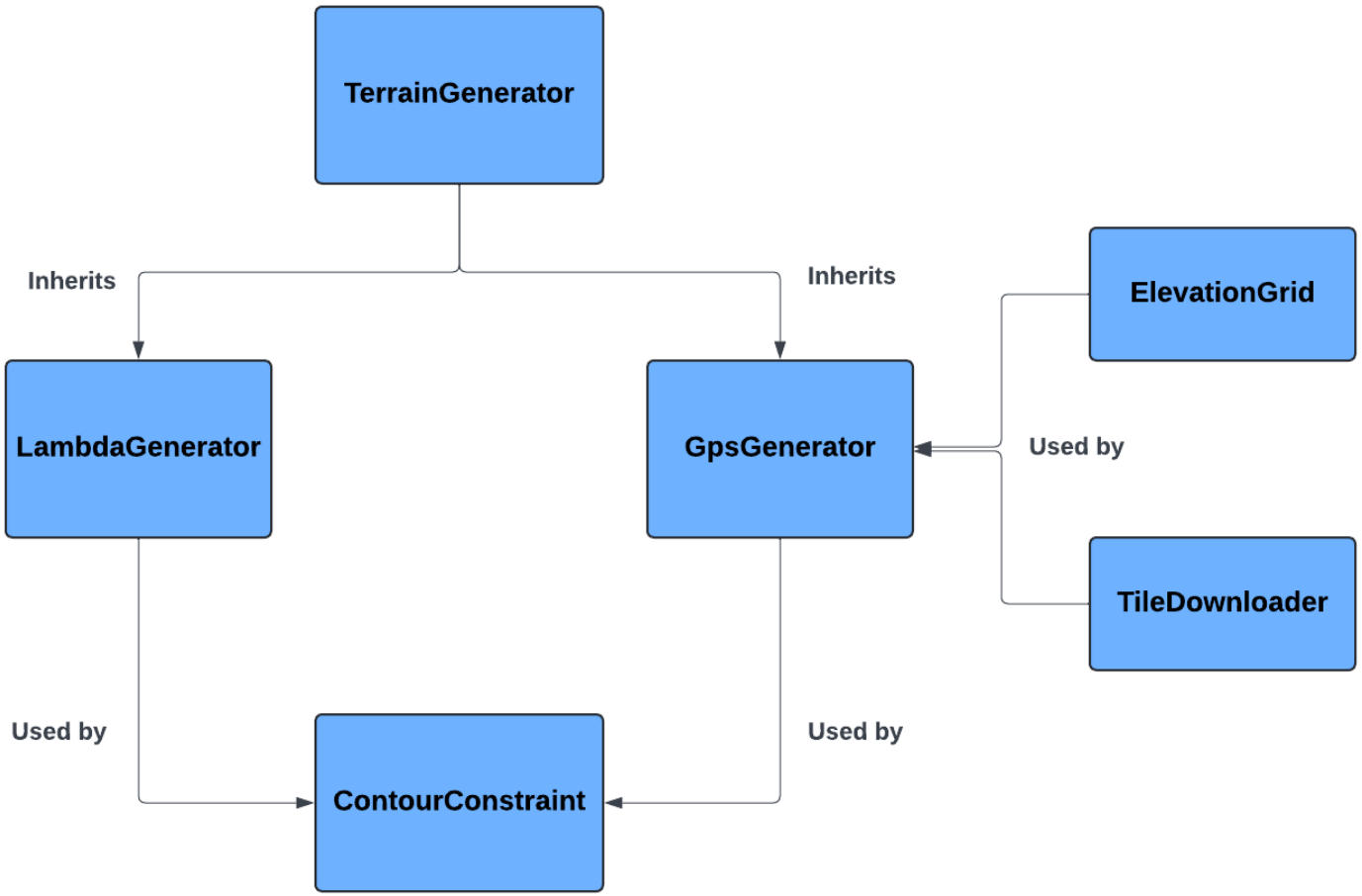


Figure: UML diagram of the project structure

# Implementation: Mesh generation

**TerrainGenerator** class definition:

```
1  class TerrainGenerator
2  {
3  public:
4      using mesh_type = Mesh;
5      virtual std::unique_ptr<mesh_type> generate() = 0;
6      virtual ~TerrainGenerator() {};
7  };
8
```

**LambdaGenerator** class definition:

```
1  class LambdaGenerator : public TerrainGenerator
2  {
3  public:
4      using fn_type = std::function<double(double, double)>;
5      LambdaGenerator(int meshSize, fn_type const &fn);
6      virtual ~LambdaGenerator();
7
8      std::unique_ptr<mesh_type> generate() override;
9
10 private:
11     int M_meshSize;
12     fn_type M_fn;
13 };
14
```

# Implementation: Mesh generation

$$z(x, y) = \sqrt{r^2 - (x - x_0)^2 - (y - y_0)^2}$$

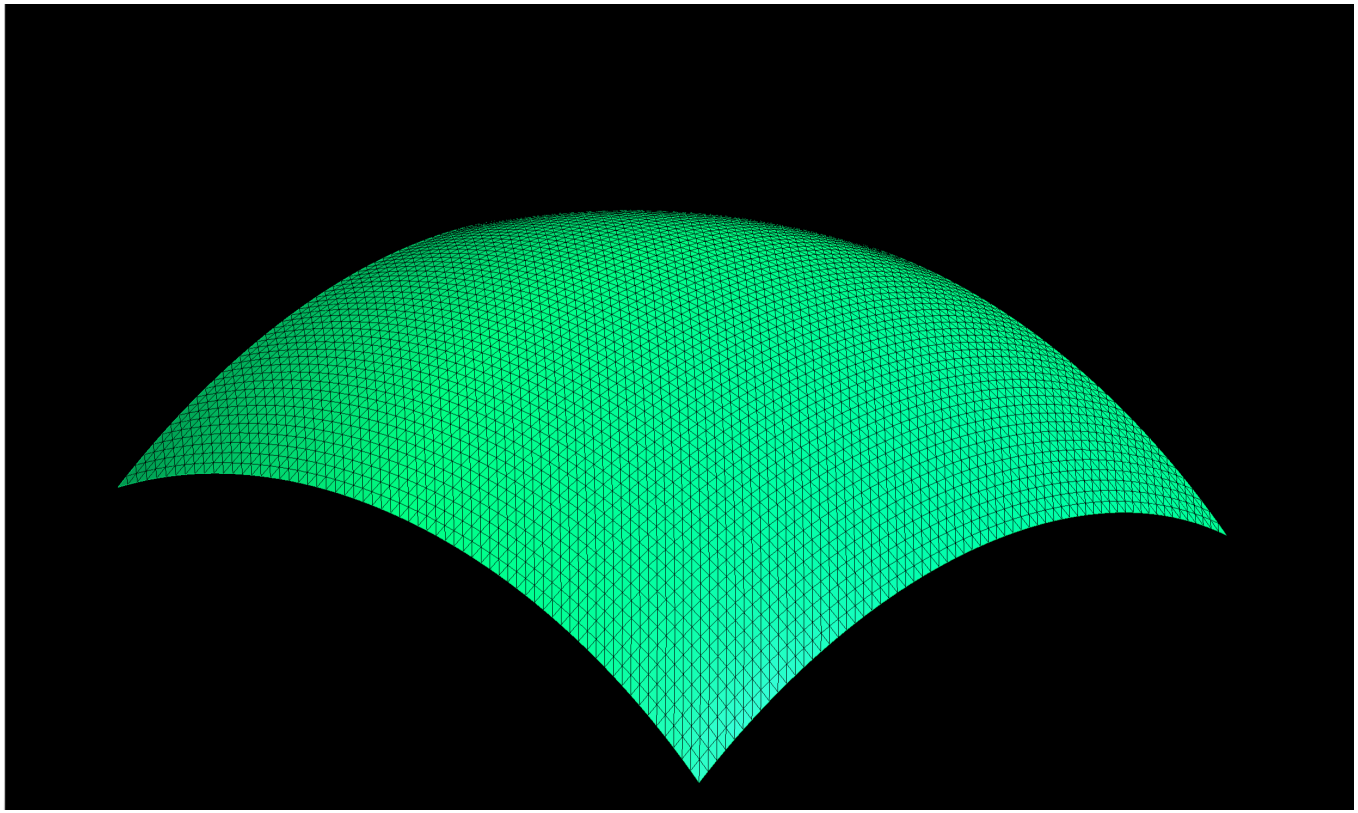

Figure: 3D mesh **LambdaGenerator** 1

# Implementation: Mesh generation

$$z(x, y) = sin(x) + cos(y)$$

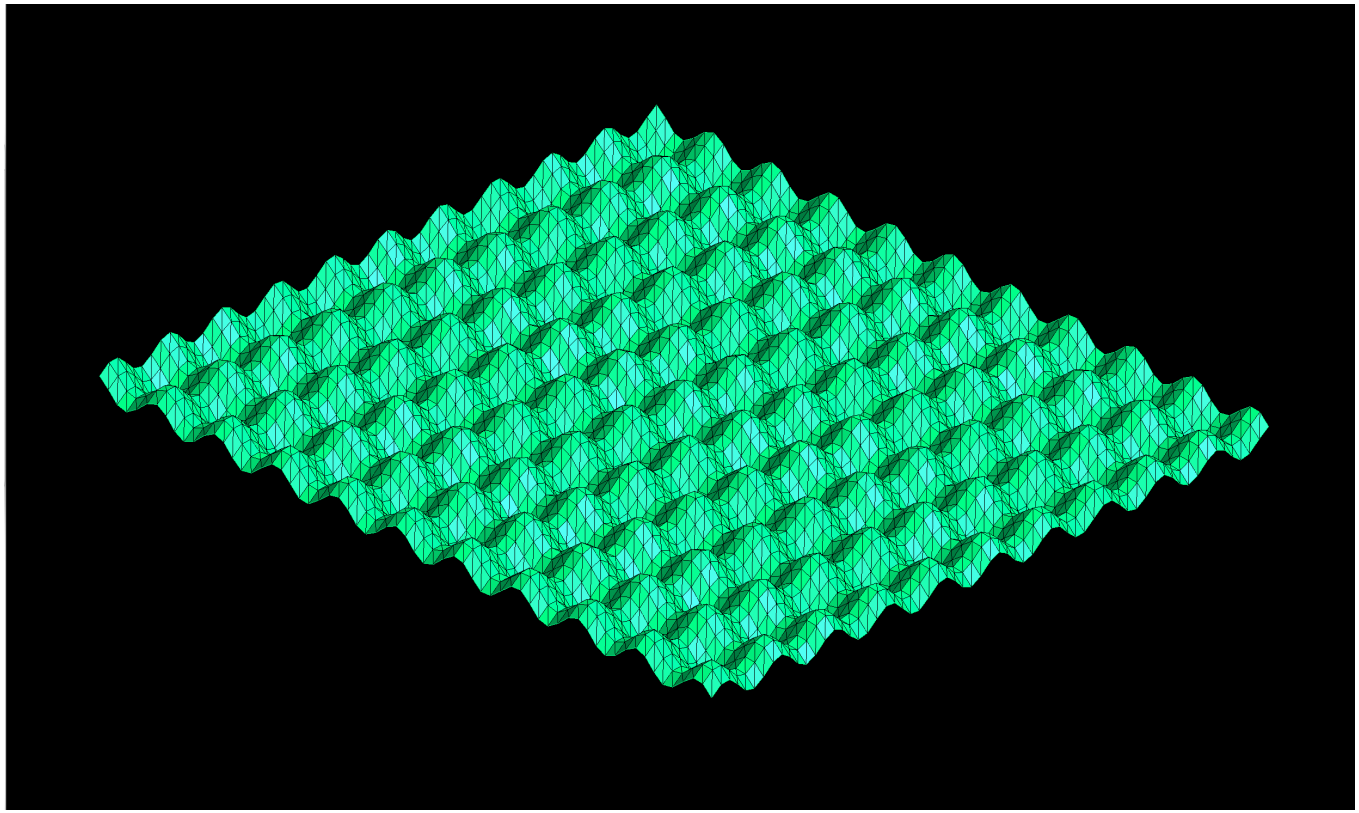

Figure: 3D mesh **LambdaGenerator** 2

## **GpsGenerator** class definition:

```
1  class GpsGenerator : public TerrainGenerator
2  {
3  public:
4      GpsGenerator(int meshSize, double latitude, double
       longitude, unsigned zoomLevel, std::string const &apiKey
       );
5      virtual ~GpsGenerator();
6
7      std::unique_ptr<mesh_type> generate() override;
8
9  private:
10     int M_meshSize;
11     double M_latitude;
12     double M_longitude;
13     unsigned M_zoomLevel;
14     std::string M_apiKey;
15 };
16
```

# Implementation: Mesh generation

$(x, y) = (33809, 23527)$ at zoom level 16

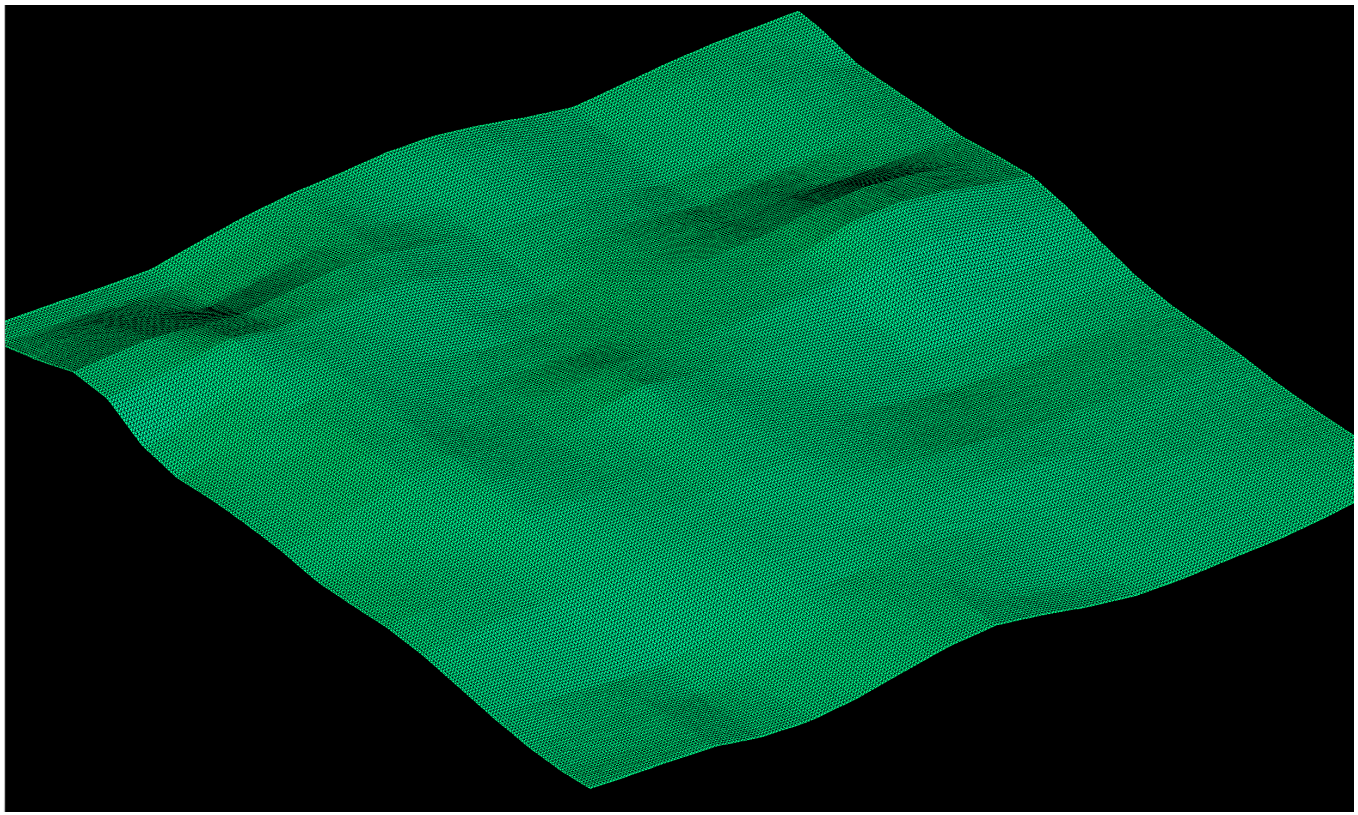

Figure: 3D mesh **GPS**

# Implementation: Contour lines generation

**ContourConstraint** class:

- **computeAllContoursMeshes** function
- **getMinMaxZ** function
- **computeContourMesh** function

# Implementation: Contour lines generation

$$z(x, y) = \sqrt{r^2 - (x - x_0)^2 - (y - y_0)^2}$$

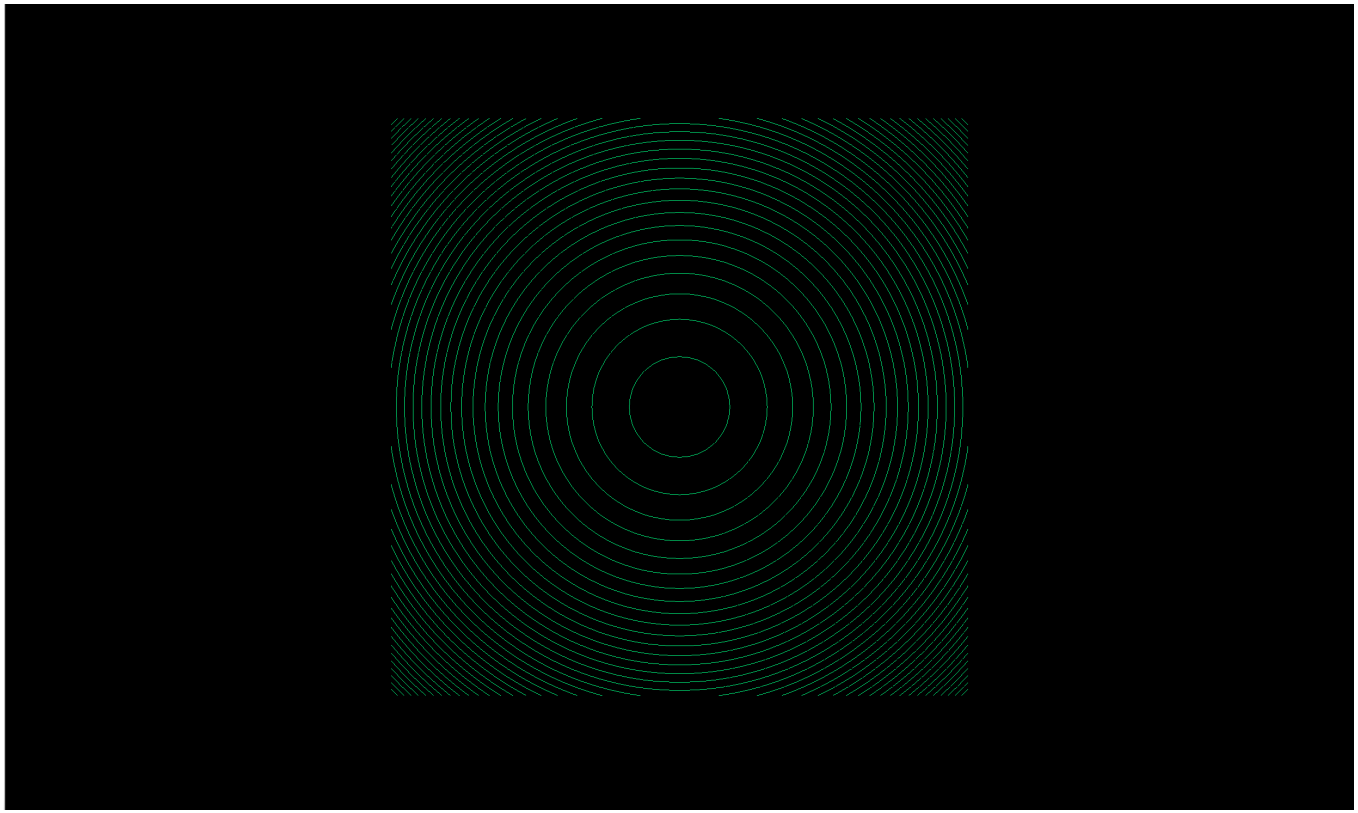

Figure: 3D mesh **contour lines** 1

# Implementation: Contour lines generation

$$z(x, y) = sin(x) + cos(y)$$

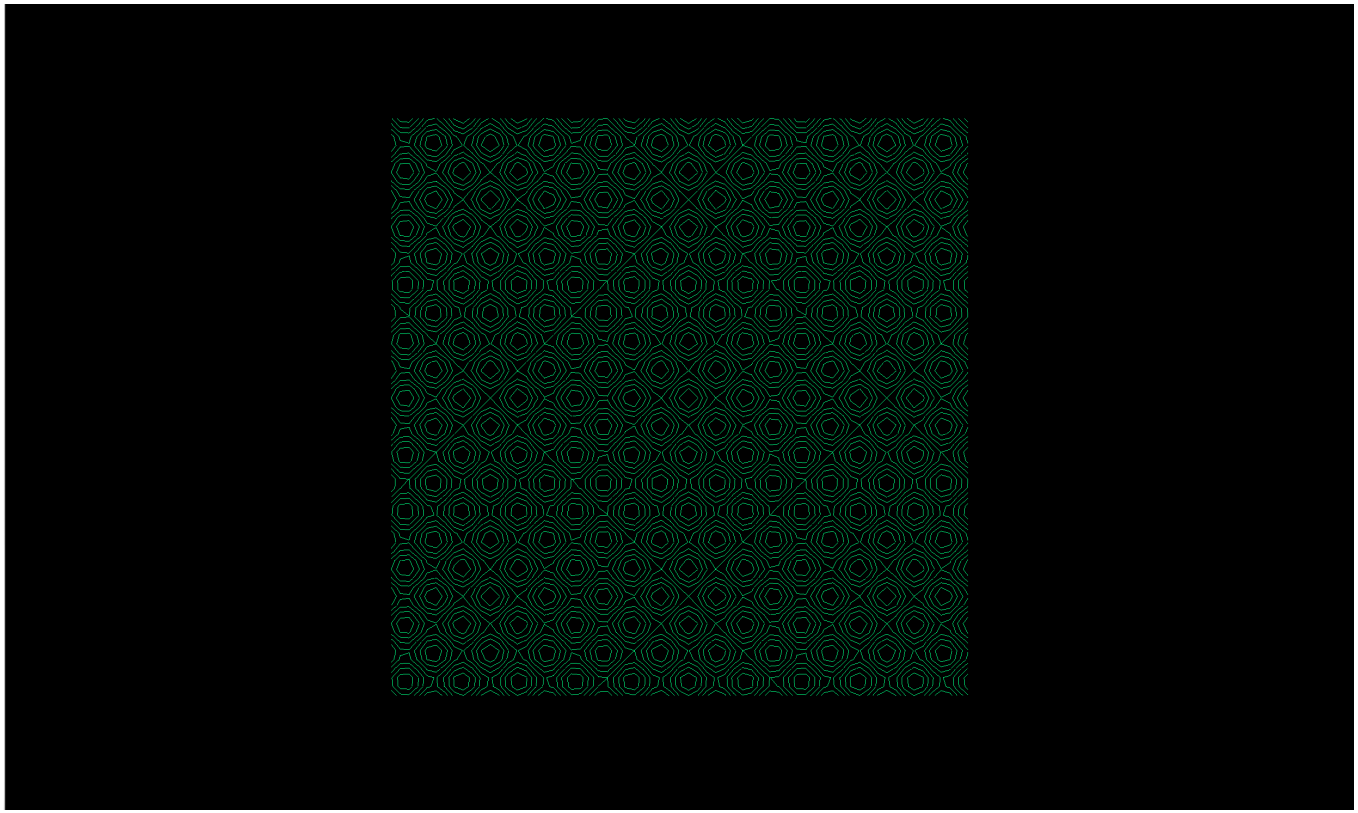

Figure: 3D mesh **contour lines** 2

# Implementation: Contour lines generation

$(x, y) = (33809, 23527)$ at zoom level 16

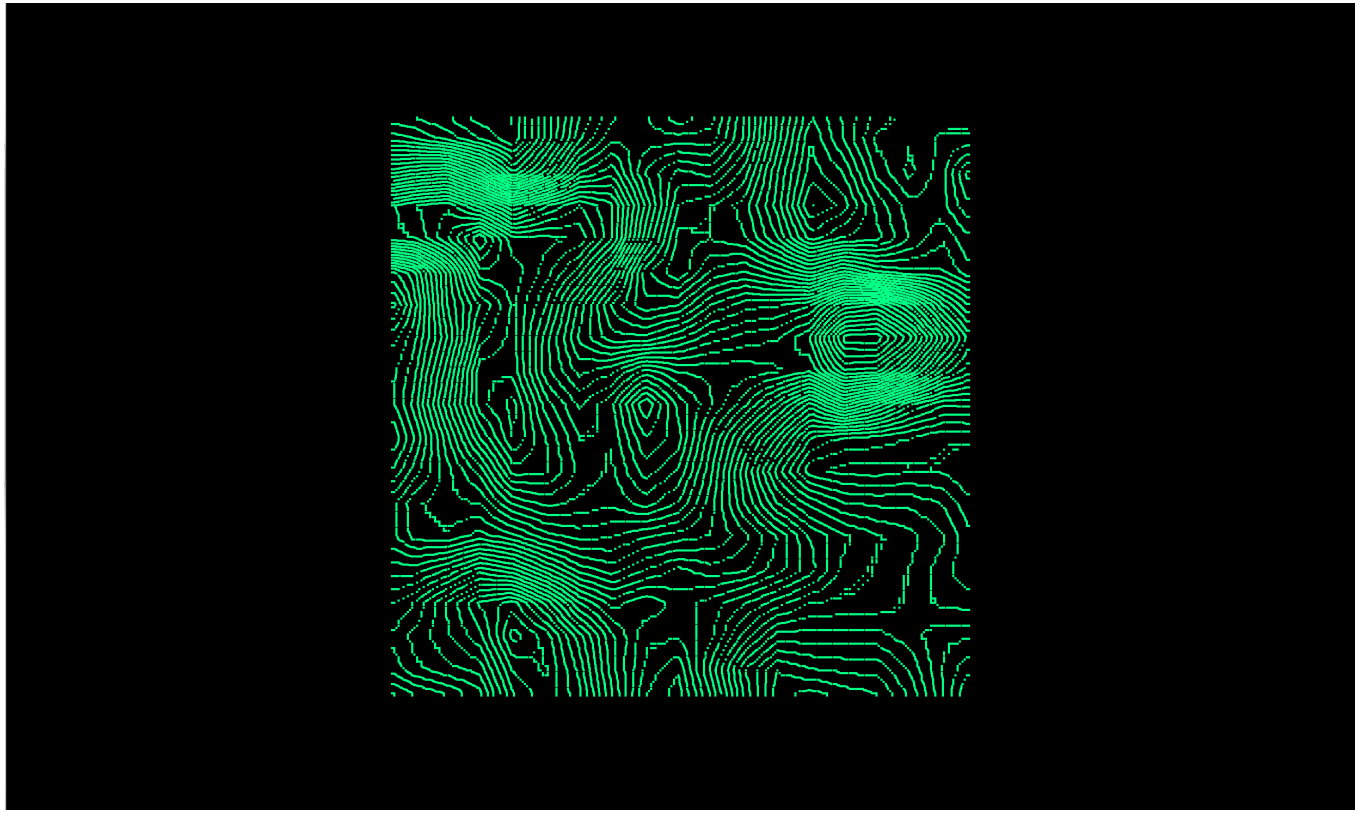

Figure: 3D mesh **contour lines GPS**

# Implementation: Re-triangulation

**triangulateAssembledMesh** function:

- **Inserting constraints** into **Constrained_Delaunay_triangulation_2**
- **CGAL::make_conforming_Delaunay_2**
- **Generating** the **final mesh**

# Results

$$z(x, y) = \sqrt{r^2 - (x - x_0)^2 - (y - y_0)^2}$$

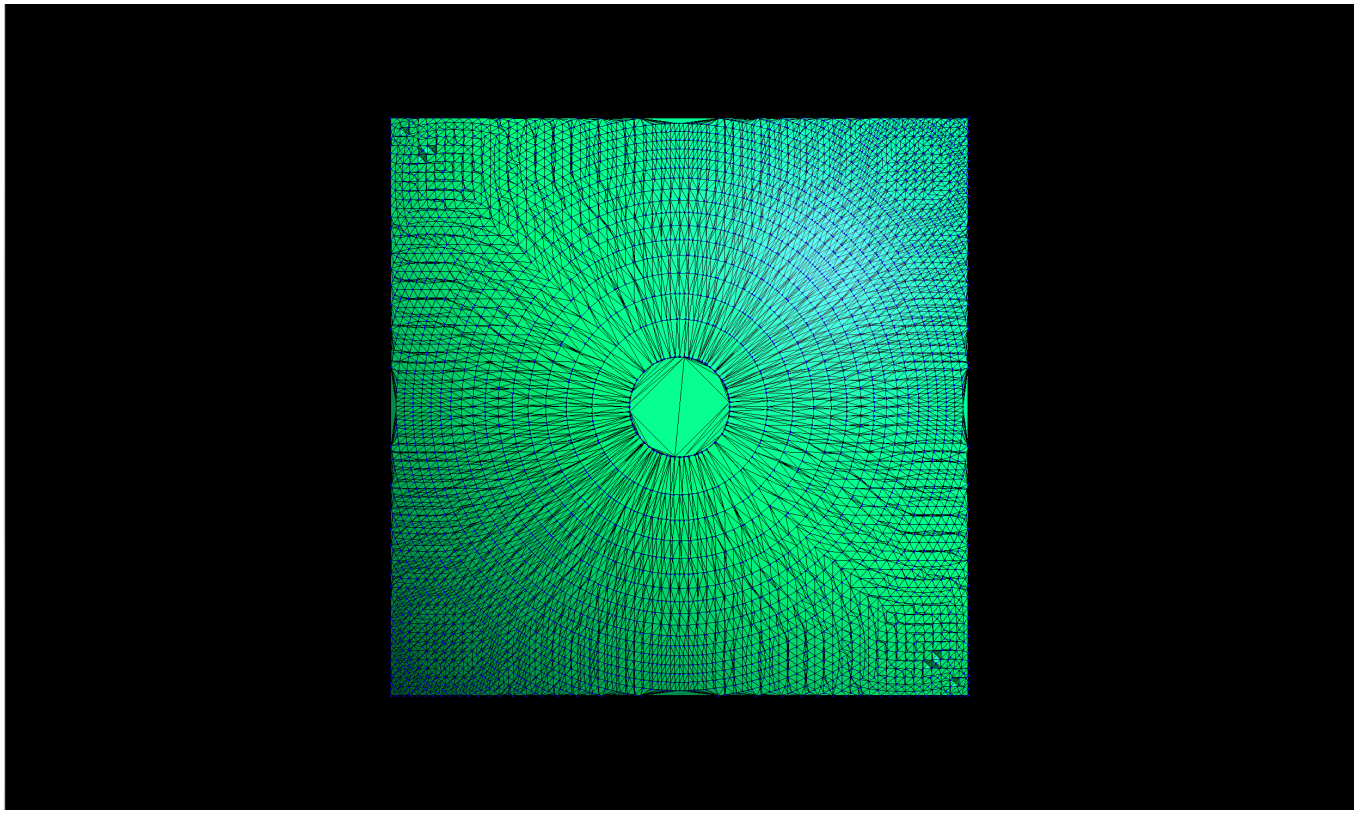

Figure: 3D mesh after re-triangulation (lambda function 1)

# Results

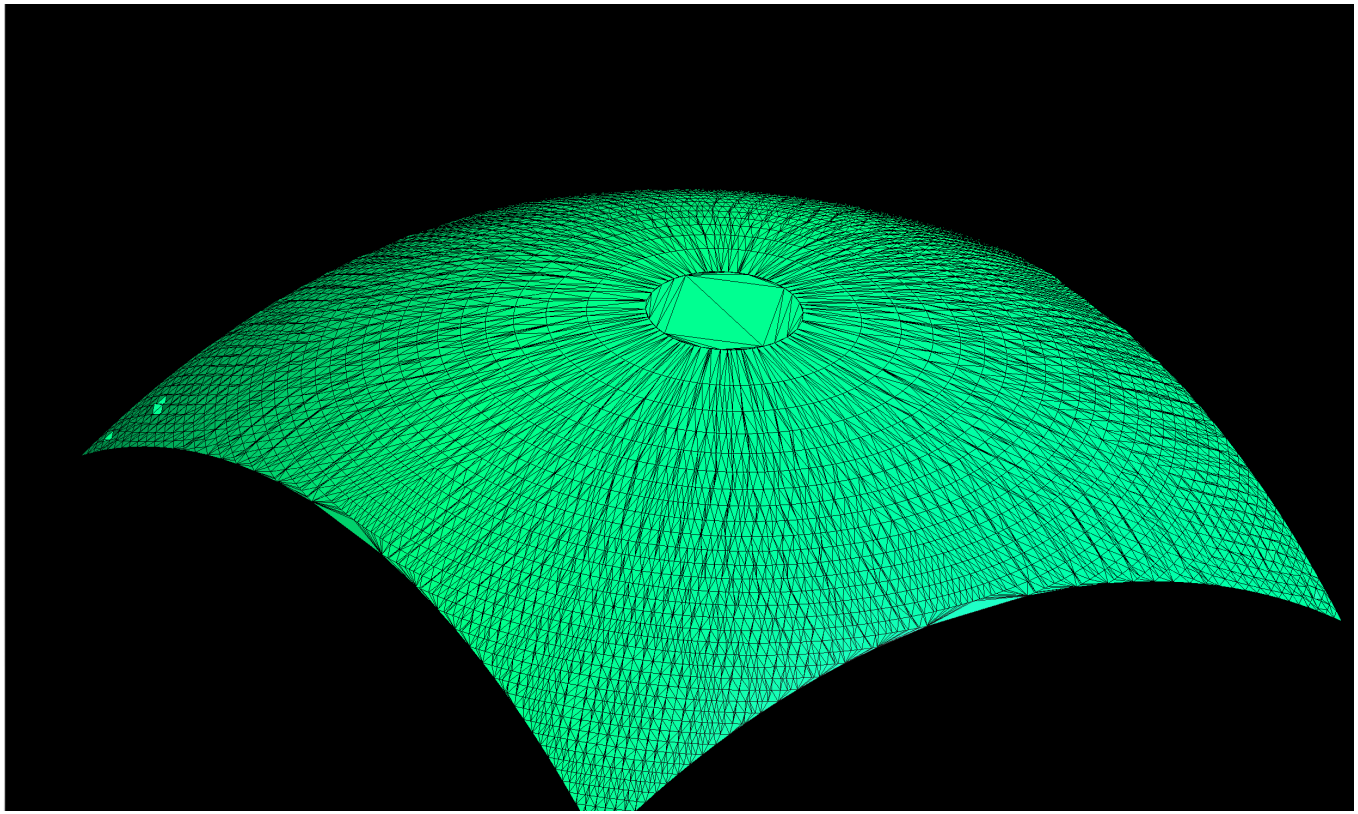

Figure: 3D mesh after re-triangulation (lambda function 1) side view

# Results

$$z(x, y) = sin(x) + cos(y)$$

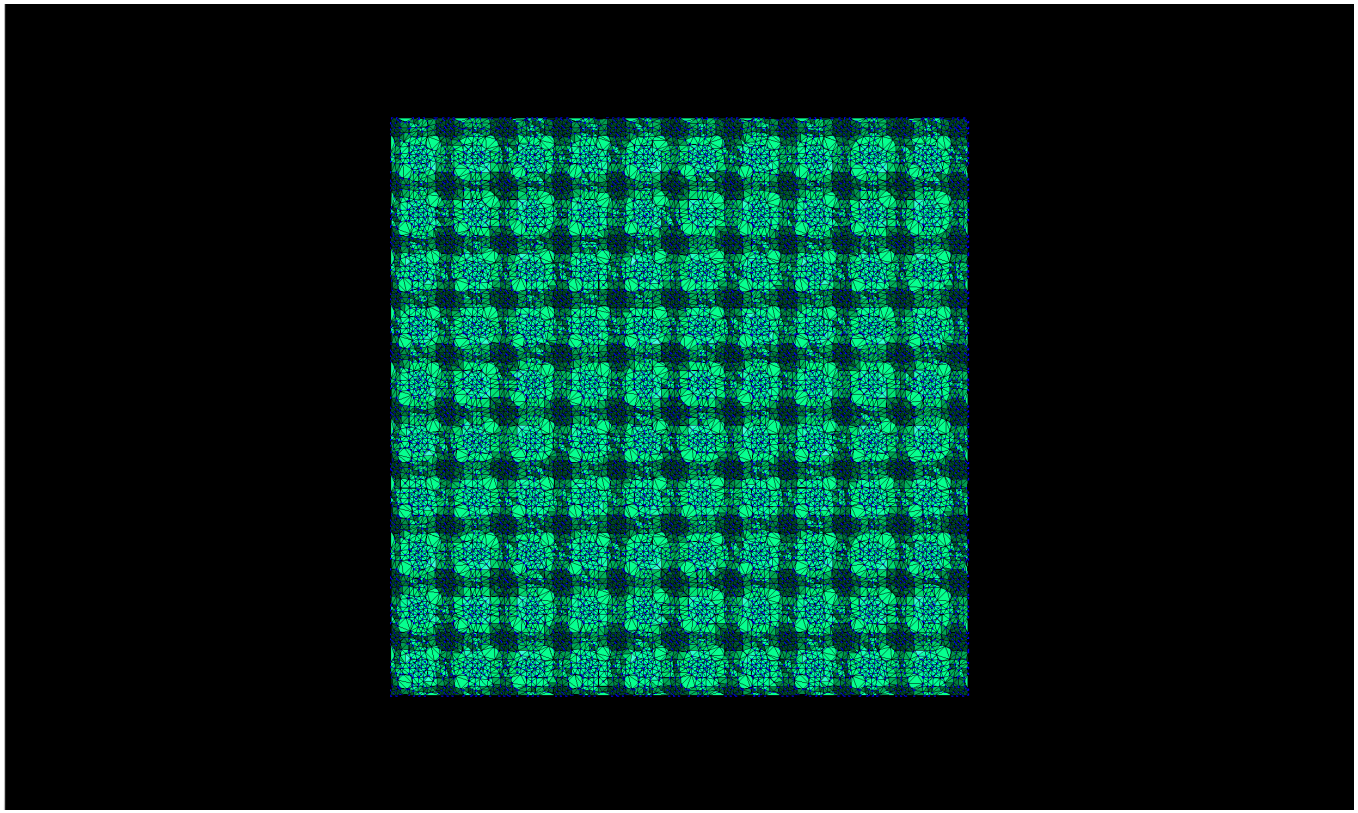

Figure: 3D mesh after re-triangulation (lambda function 2)

# Results

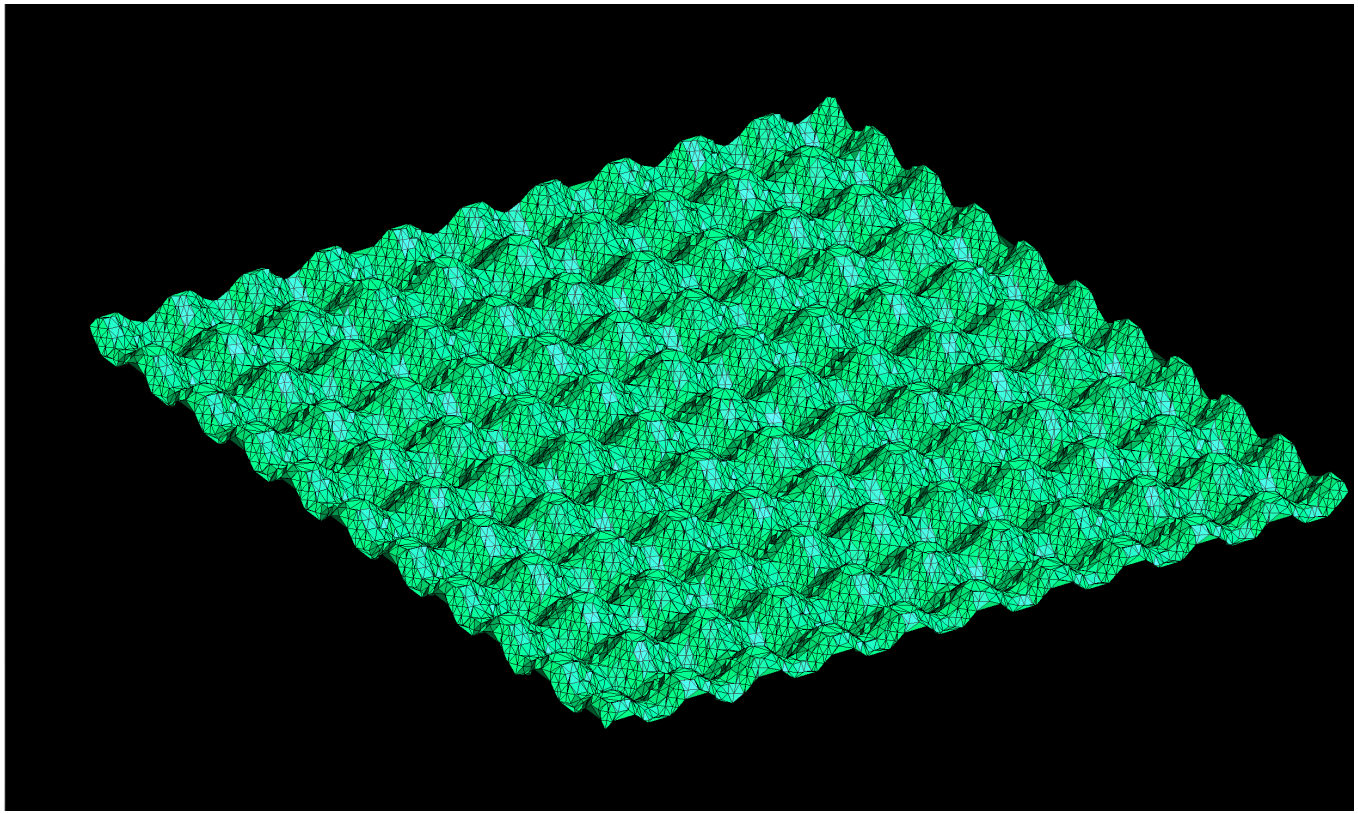

Figure: 3D mesh after re-triangulation (lambda function 2) side view

# Results

$(x, y) = (33809, 23527)$ at zoom level 16

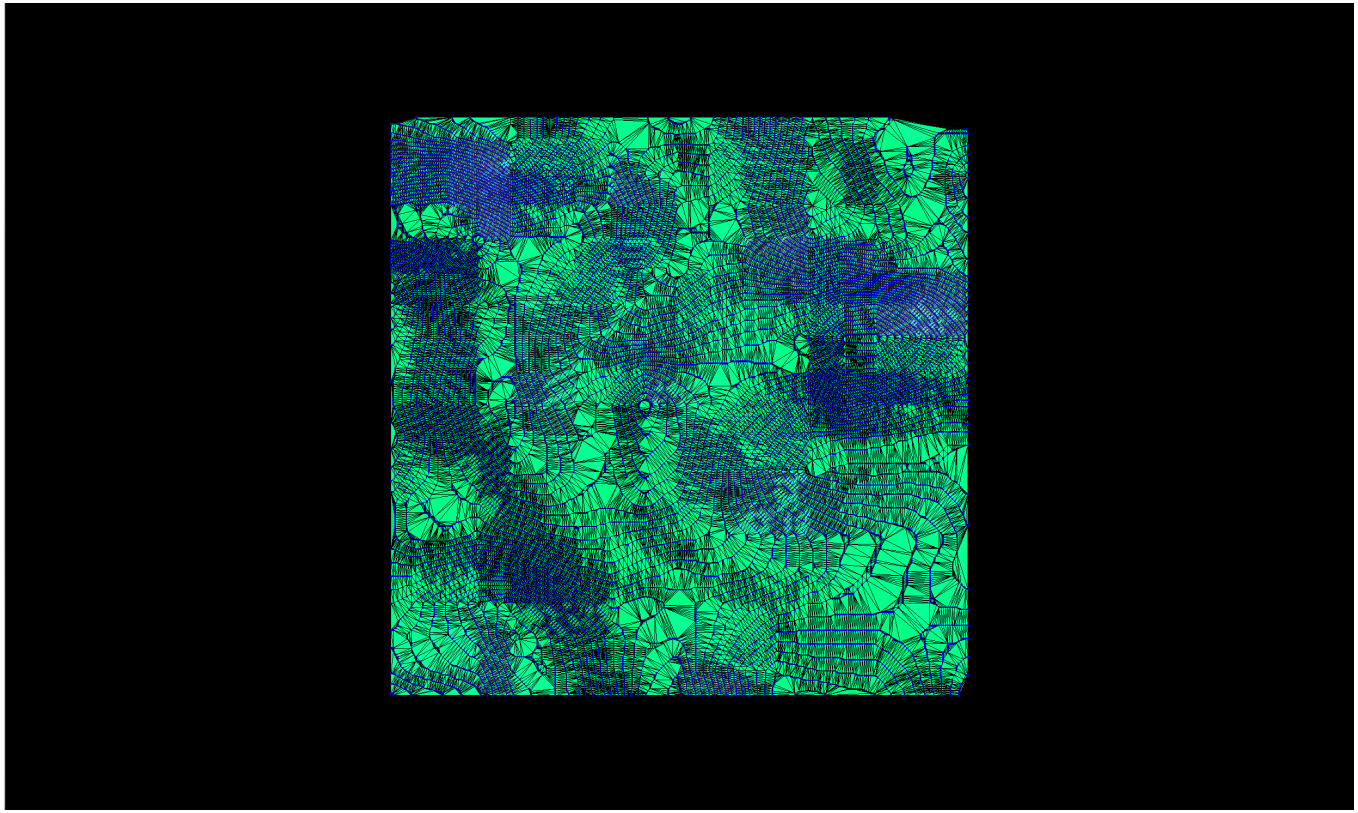

Figure: 3D mesh after re-triangulation (GPS data zoom 16)

# Results

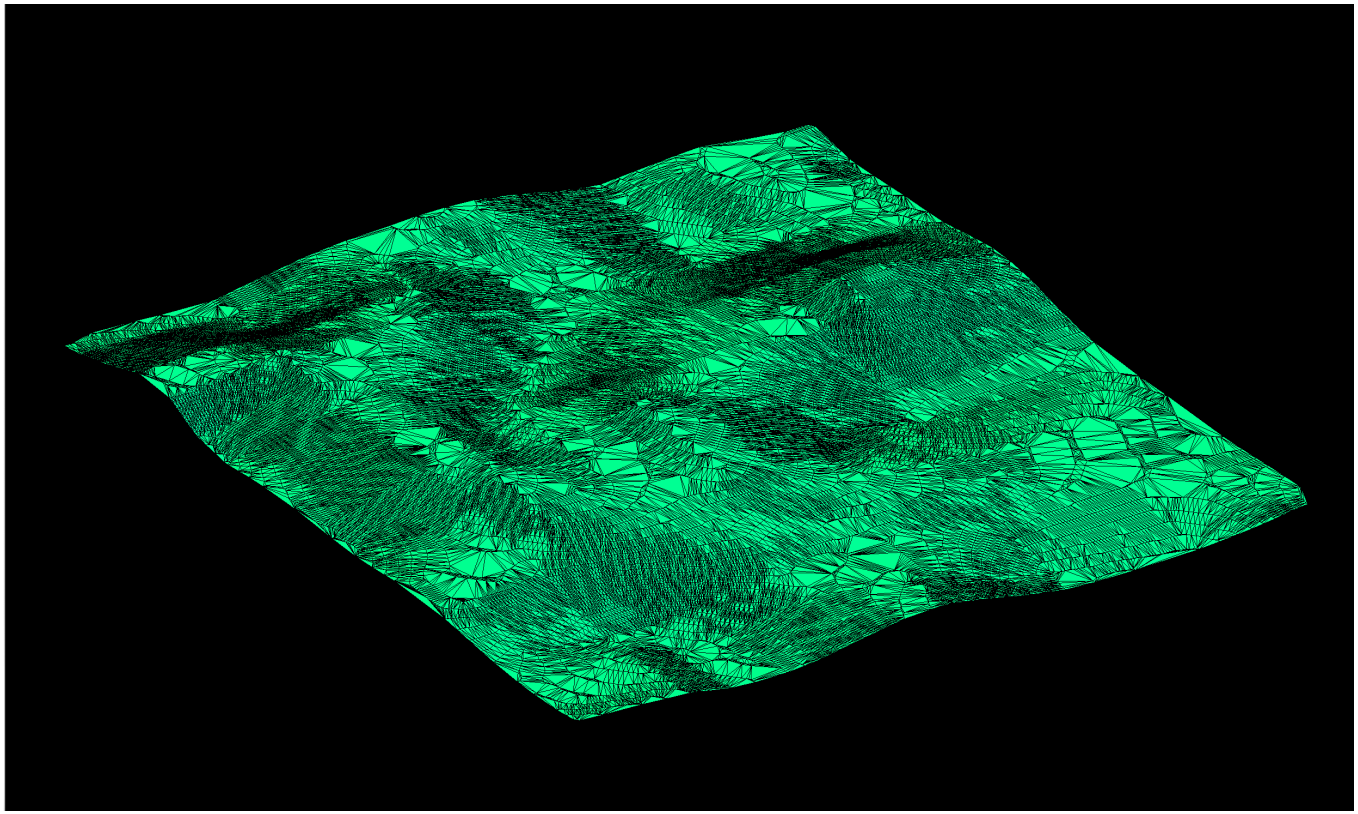

Figure: 3D mesh after re-triangulation (GPS data zoom 16) side view

# Results

$(x, y) = (16904, 11763)$ at zoom level 15

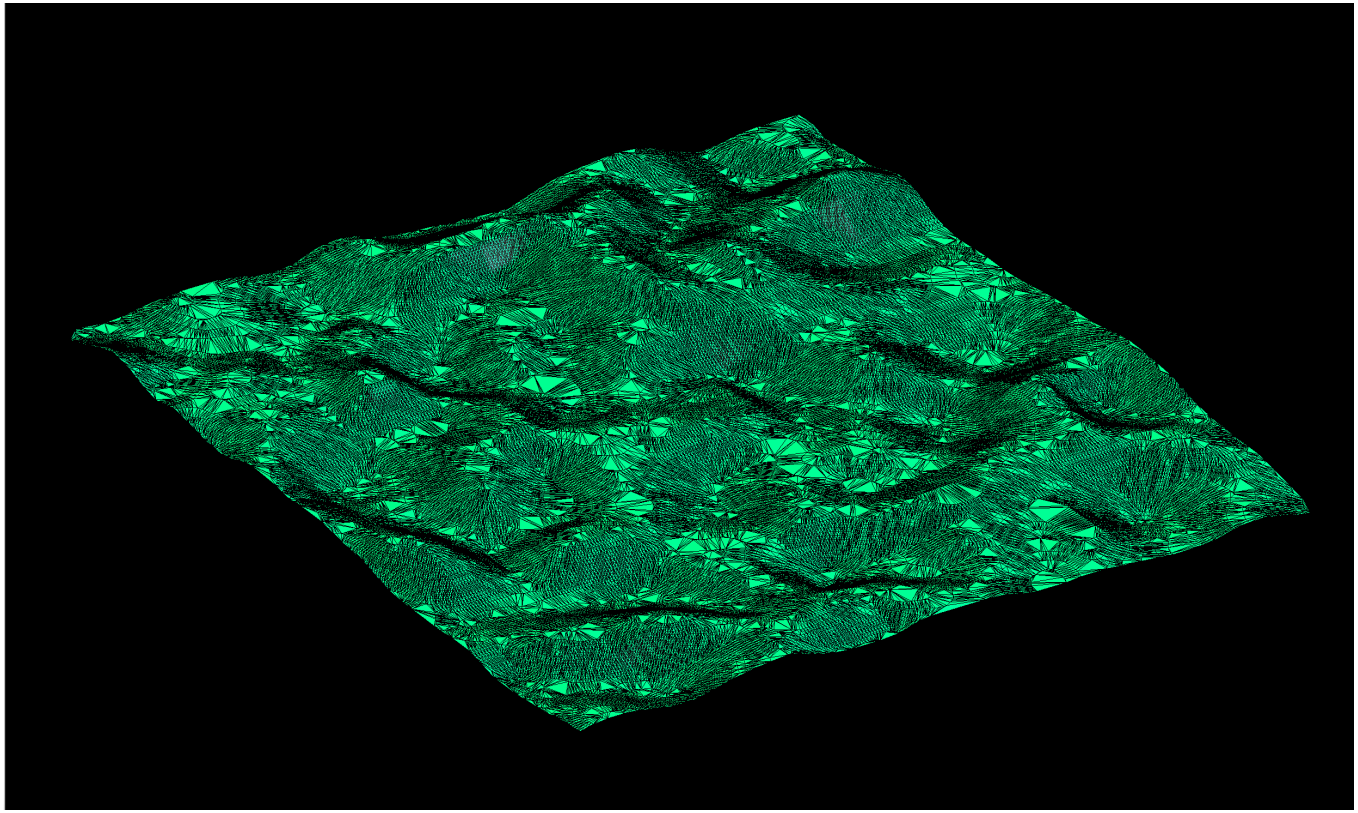

Figure: 3D mesh after re-triangulation (GPS data zoom 15)

# Results

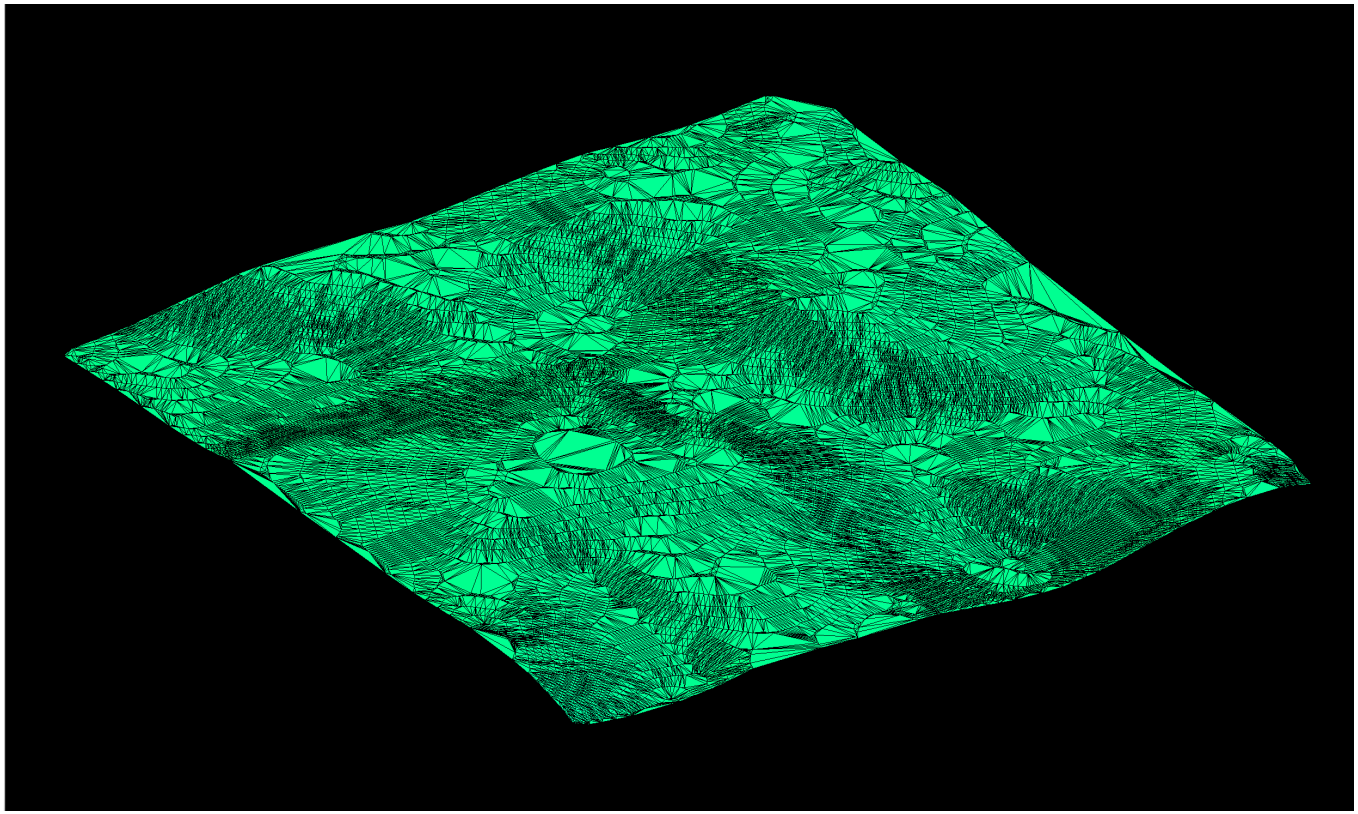

Figure: 3D mesh after re-triangulation (GPS data Strasbourg zoom 16)

# Results

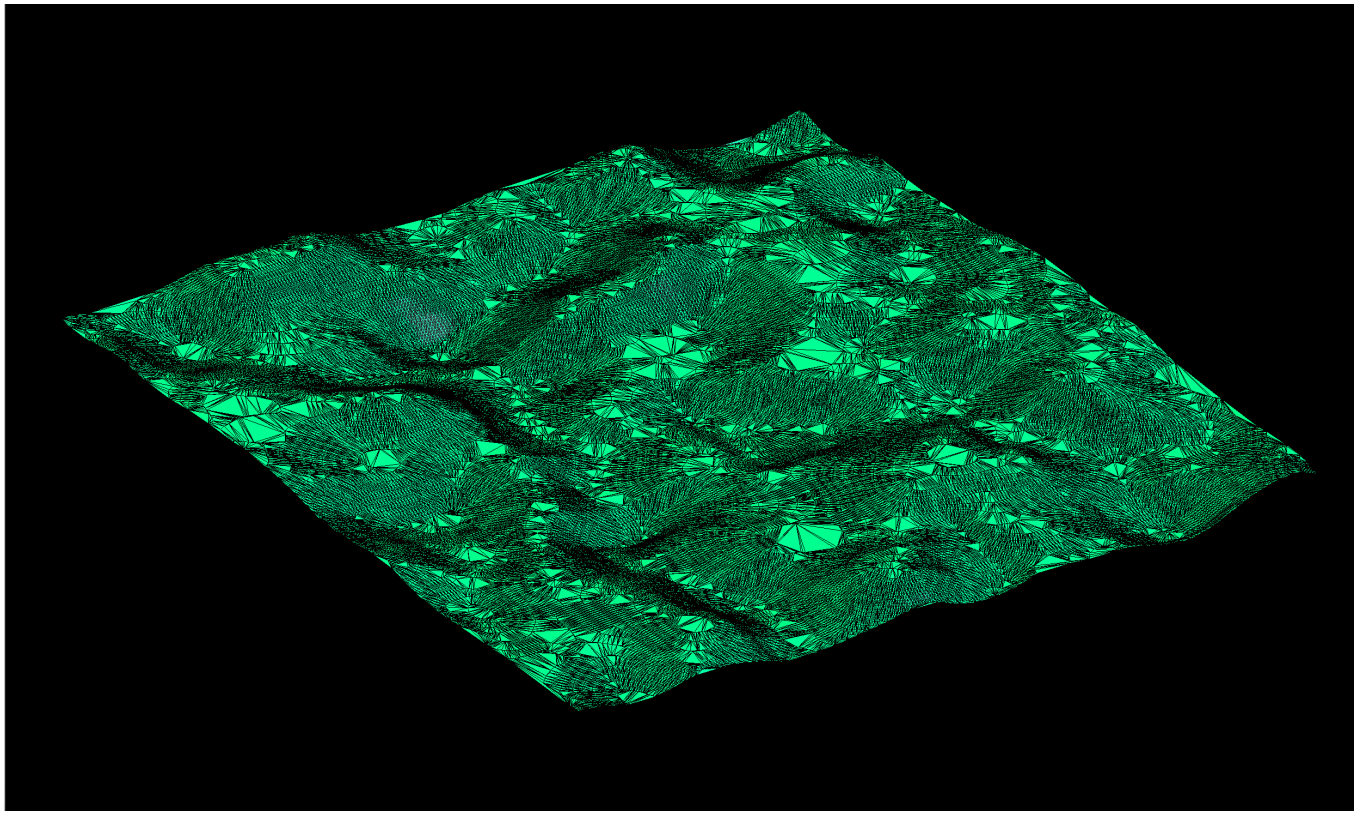

Figure: 3D mesh after re-triangulation (GPS data Strasbourg zoom 15)

# Prospects: Handling and merging multiple tiles

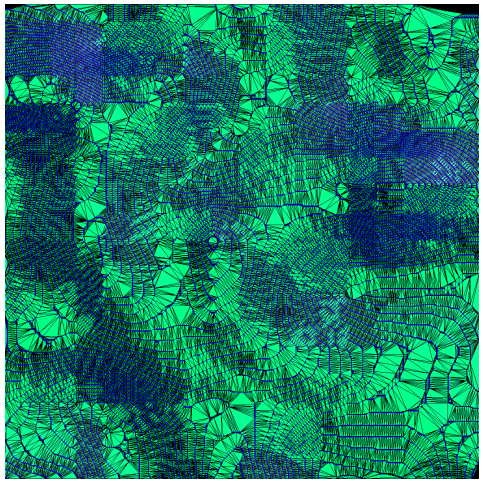

Figure: (33809, 23527)

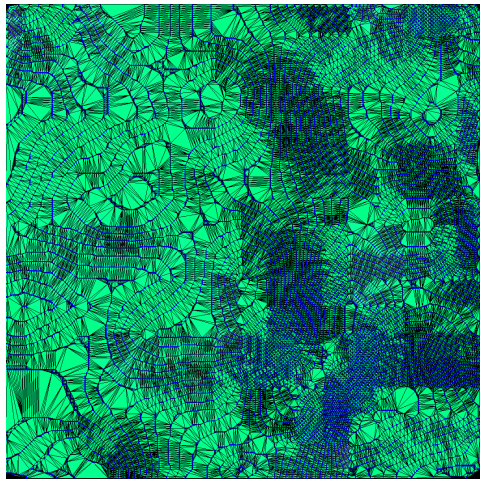

Figure: (33810, 23527)

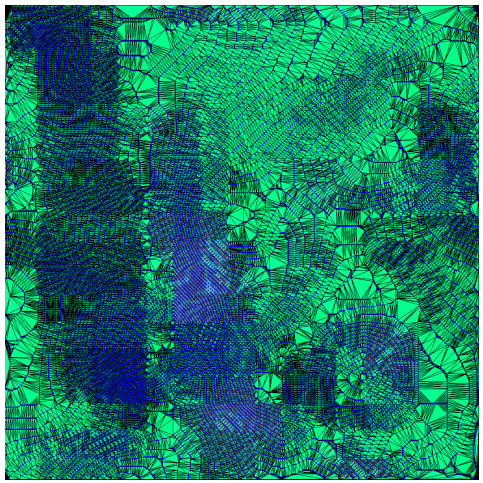

Figure: (33809, 23528)

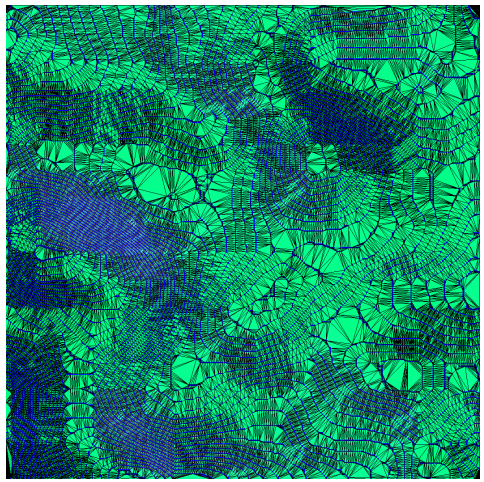

Figure: (33810, 23528)

# Prospects: Handling and merging multiple tiles

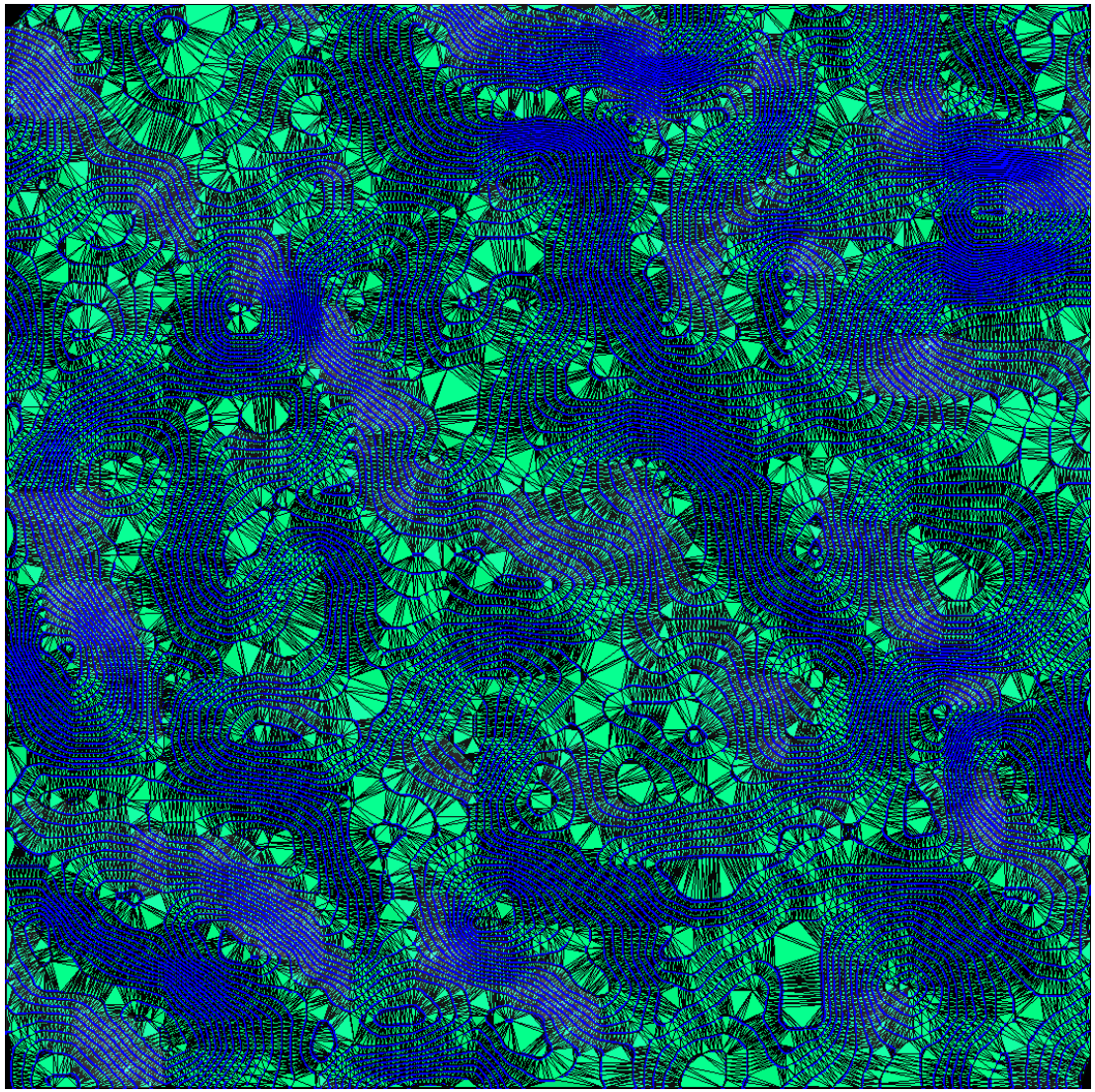

Figure: Resultig merged tile

# Prospects: Mesh refinement

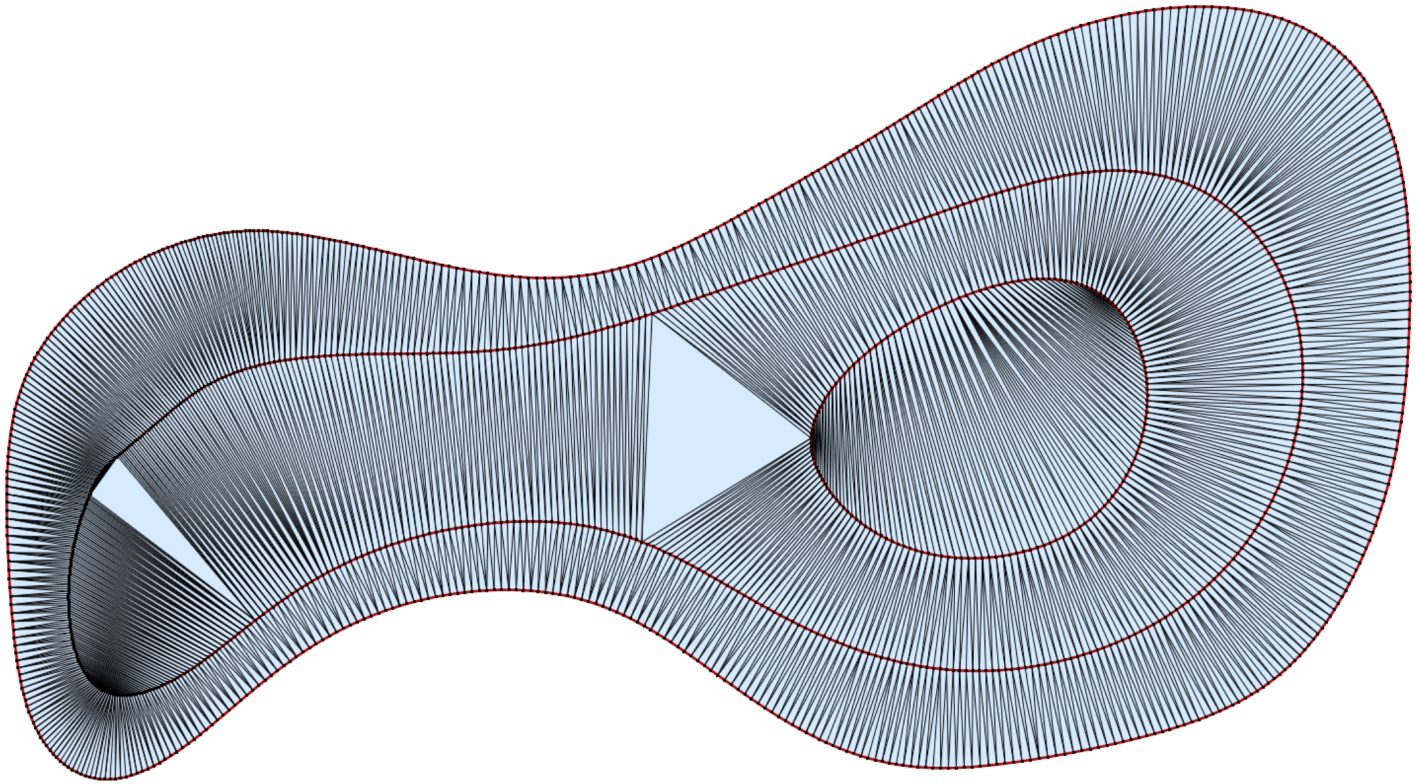

Figure: Mesh constrained to contour lines

# Prospects: Mesh refinement

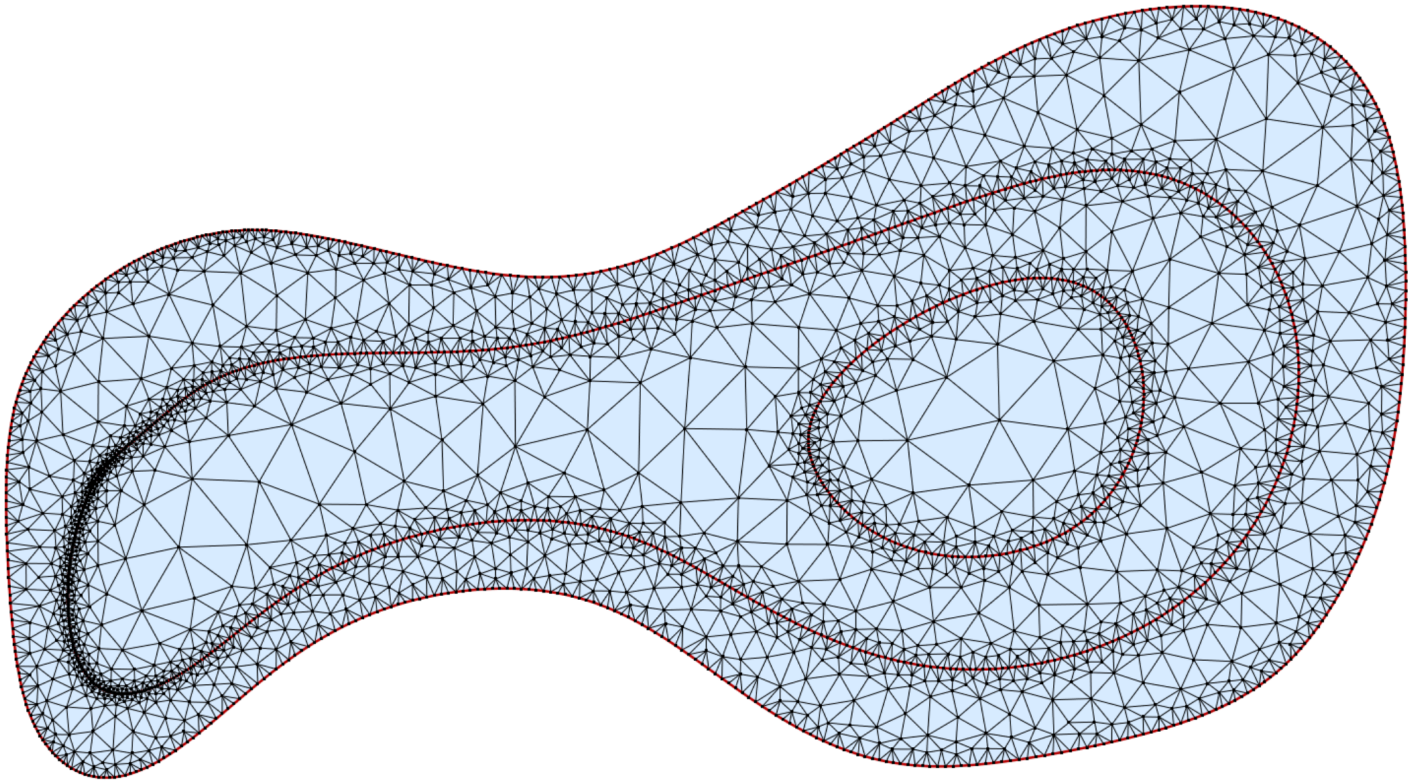

Figure: Mesh refined

# Prospects: Parallelization

- **Parallel processing** of **multiple tiles**
- **Parallelizing mesh generation**
- **Parallel tile merging**

# Prospects: Urban elements integration

- **Integrating urban** elements with **terrain models**

- **Detailed 3D model** of the **urban environment**

# Conclusion

Flexible **terrain** generation:

- **LambdaGenerator**
- **GpsGenerator**

# Conclusion

**Contour lines** generation:

- **ContourConstraint**

# Conclusion

**Triangulation**:

- **triangulateAssembledMesh**

# Conclusion

Framework developed provides foundation:

- **Future developments** and **applications**
- **Sustainable urban development** and **environmental conservation**

# The end

**Thank you all for your attention!**


Cemosis.
Cemosis.
https://www.cemosis.fr, 2024.


PEPR Numpex.
Priority Research Program and Equipment for Numerical Exascale computing.
https://numpex.org/numpex-program, 2024.


CoE Hidalgo2.
HPC and big data technologies for global challenges.
https://www.hidalgo2.eu/about, 2024.


Lohmann, Niels.
JSON for Modern C++.
https://json.nlohmann.me, 2024.


Stenberg, Daniel.
cURL: A Command Line Tool and Library for Transferring Data with URLs.
https://curl.se, 2024.


libpng: The Official PNG Reference Library.

http://www.libpng.org/pub/png/libpng.html, 2024.


CGAL: The Computational Geometry Algorithms Library.
https://www.cgal.org, 2024.


BBC Bitesize.
Latitude and Longitude.
https://www.bbc.co.uk/bitesize/guides/ztqtyrd/revision/1, 2024.


Britannica.
Mercator projection.
https://www.britannica.com/science/Mercator-projection/images-videos, 2024.


Wikipedia contributors.
Tiled Web Map.
https://en.wikipedia.org/wiki/Tiled_web_map#/media/File:Tiled_web_map_Stevage.png, 2024.


Wikipedia contributors.
XYZ Tiles coordinate numbers.
https://en.wikipedia.org/wiki/File:XYZ_Tiles.png, 2024.


Wikipedia contributors.
Delaunay triangulation.
https://en.wikipedia.org/wiki/Delaunay_triangulation#/media/File:Delaunay_circumcircles_vectorial.svg, 2024.


Dilo Arta.
Storage and Manipulation of Vague Spatial Objects Using Existing GIS Functionality.
https://www.researchgate.net/publication/225854833_Storage_and_Manipulation_of_Vague_Spatial_Objects_Using_Existing_GIS_Functionality, 2024.


IRMA.
IRMA.
https://irma.math.unistra.fr, 2024.


University of Strasbourg.
University of Strasbourg.
https://en.unistra.fr, 2024.


Exa-MA.
Methods and Algorithms for Exascale.

https://numpex.org/exama-methods-and-algorithms-for-exascale, 2024.


Prud'homme, Christophe.
Expanding horizons: Ktirio and the urban building vision in Hidalgo2.
https://github.com/orgs/feelpp/discussions/2167, 2024. October 2023.


CoE Hidalgo2.
The Urban Building Model.
https://www.hidalgo2.eu/urban-building-model, 2024.


Inria.
National Institute for Research in Digital Science and Technology.
https://www.inria.fr/en, 2024.


European Environment Agency.
Greenhouse gas emissions from energy use in buildings in Europe.
https://www.eea.europa.eu/en/analysis/indicators/greenhouse-gas-emissions-from-energy?activeAccordion=546a7c35-9188-4d23-94ee-005d97c26f2b, 2024. October 2023.


European Environment Agency.
Accelerating the energy efficiency renovation of residential buildings — a behavioural approach.
https://www.eea.europa.eu/publications/accelerating-the-energy-efficiency, 2024.
June 2023.


European Commission.
2050 long-term strategy.
https://climate.ec.europa.eu/eu-action/climate-strategies-targets/2050-long-term-strategy_en#:~:text=Striving%20to%20become%20the%20world's%20first%20climate%2Dneutral%20continent%20by%202050.&text=The%20EU%20aims%20to%20be,to%20the%20European%20Climate%20Law%20., 2024.


European Commission.
The European Green Deal.
https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal_en, 2024.


European Commission.
European Climate Law.

https://climate.ec.europa.eu/eu-action/european-climate-law_en, 2024.


Prud'homme, Christophe.
CoE Hidalgo2 Urban Building Pilot at NumPEx workshop on Discretization@Exascale.
https://github.com/orgs/feelpp/discussions/2188, 2024. November 2023.


EuroHPC JU.
The European High Performance Computing Joint Undertaking.
https://eurohpc-ju.europa.eu/index_en, 2024.


Wikipedia contributors.
Mapbox.
https://en.wikipedia.org/wiki/Mapbox, 2024. August 2024.


Mapbox.
Mapbox Terrain-RGB v1.
https://docs.mapbox.com/data/tilesets/reference/mapbox-terrain-rgb-v1, 2024.


Mapbox.

Mapbox Raster Tiles API.
https://docs.mapbox.com/api/maps/raster-tiles, 2024.


STL (STereoLithography) File Format Specification.
https://www.fabbers.com/tech/STL_Format, 2024.


Geuzaine Christophe, and Jean-François Remacle.
Gmsh: A 3D Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities.
https://gmsh.info, 2024.
Version 4.10.


*MSH: The Gmsh Mesh File Format*, 2024.


Wikipedia contributors.
World Geodetic System.
https://en.wikipedia.org/wiki/World_Geodetic_System, 2024.


Wikipedia contributors.
Mercator projection.
https://en.wikipedia.org/wiki/Mercator_projection, 2024.


Wikipedia contributors.
Web Mercator projection.

https://en.wikipedia.org/wiki/Web_Mercator_projection, 2024.


Wikipedia contributors.
Constrained Delaunay triangulation.
https://en.wikipedia.org/wiki/Constrained_Delaunay_triangulation, 2024.


Chew, L. Paul.
Constrained Delaunay Triangulations.
https://www.cs.jhu.edu/~misha/Spring16/Chew87.pdf, 1987.

Dartmouth College.


Wikipedia contributors.
Delaunay triangulation.
https://en.wikipedia.org/wiki/Voronoi_diagram, 2024.


Wikipedia contributors.
Voronoi diagram, 2024.


Wikipedia contributors.
Tiled web map.
https://en.wikipedia.org/wiki/Tiled_web_map, 2024.


Wikipedia contributors.
Delaunay triangulation with all the circumcircles and their centers.
https://en.wikipedia.org/wiki/Delaunay_triangulation#/media/File:Delaunay_circumcircles_centers.svg, 2024.


Wikipedia contributors.
Delaunay triangulation and its Voronoi diagram.
https://en.wikipedia.org/wiki/Delaunay_triangulation#/media/File:Delaunay_Voronoi.svg, 2024.


Alliez, Pierre.
Constrained mesh image, 2024.
Image provided during personal communication.


Alliez, Pierre.
Constrained refined mesh image, 2024.
Image provided during personal communication.