# Geometric Algorithms and Spatial Data Structures

Final project 2020/2021 – Trapezoidal maps

Simone Seu (60/73/65249)

Link of the repository: https://github.com/simoseu/TrapezoidalMaps

## Structure

The folder/file structure:

**Data_structure**: the folder contains the classes of all data structures implemented. In the following are described the implemented data structures:

**- dag:** The class implement a Directed Acyclic Graph as a vector of **nodes**, is possible to add new nodes to the dag, replace an existing node in the dag with a new one, access to a node given its index or obtain the whole vector of nodes, also get the number of nodes stored.

**- node:** The class implements the node of the **dag**, a node has a Node Type which is implemented by an enum (X, Y, LEAF) one for each type of node that a dag can store. The X node represents a point, the Y node represents a segment and the LEAF represents a Trapezoid.
Each node stores an index (the link to the trapezoidal map object that the node represents).
For example a node of type LEAF contains the trapezoidal map index (where the trapezoid is stored) of the trapezoid represented by the node.
It also contains the indexes of the two children of the node.

**- trapezoidalmap:** The class implement the trapezoidal map as a vector of **trapezoids,** is possible to add new trapezoid to the trapezoidal map, replace an existing trapezoid in the trapezoidal map with a new one, access to a node given its index and obtain the whole vector of trapezoids, is possible to get the number of trapezoid stored in the trapezoidal map.

**- trapezoid:** The class implements the trapezoid, a trapezoid contains a top and a bottom edge, a left and a right point, and the link to the dag leaf that represents the trapezoid.
Can also store the references to its neighbors trapezoids, storing their indexes.

Is possible to get and set all the information above mentioned and is possible to retrieve the corners of the trapezoid, used to draw the trapezoid.

**Drawables:** The folder contains the drawable objects.

**- drawable_trapezoidalmap:** The class implements the drawable version of the trapezoidal map data structures. The added information with respect to the trapezoidal map data structures are a vector of colors, that store the color of each vector of the trapezoidal map and an index that represent the highlighted trapezoid (when a query is performed the highlighted trapezoid shows visually in which trapezoid lies the query point).

**Utilis:** The folder contains some utility function.

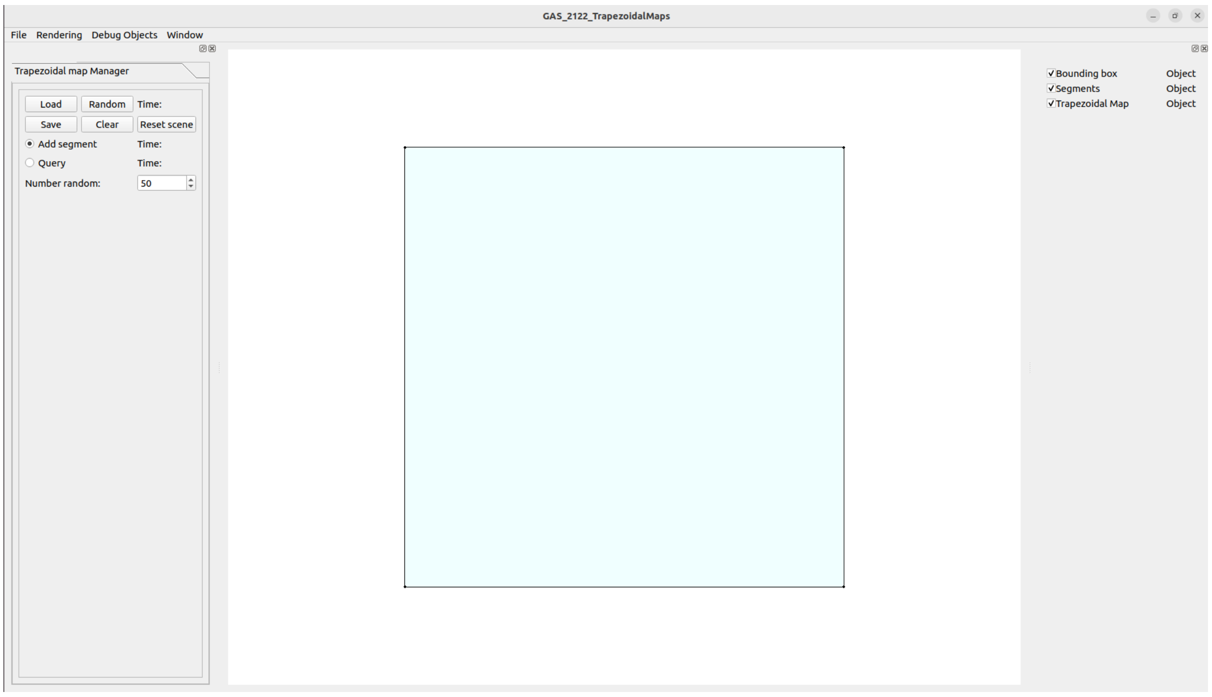
**- projectUtils:** In this file are written some functions used in the algorithms and in the drawing of a trapezoid. Like randomColor() that generates a new random color or orderSegment() that order the endpoint of a given segment by its x coordinates.

**Algorithms:** The folder contains the algorithms implemented for the project

**- algorithms:** In this file are implemented the algorithms for the incremental building of the trapezoidal map and its associated directed acyclic graph. For Example queryPoint() implements searching of a point in the trapezoidal map, while followSegment() returns all the trapezoids intersected by the inserted segment. It also contains a function that initializes the two data structures.

## Results

Some images and results:



Time:

Time:

Time:

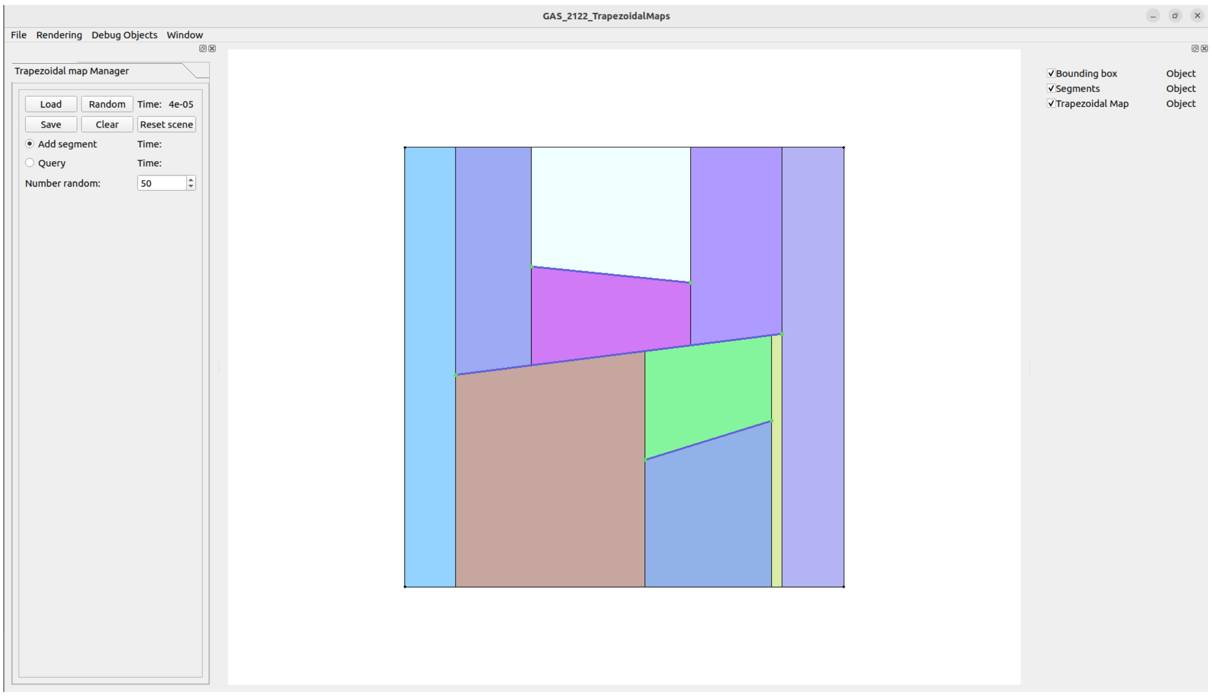
Number random: 50

✓ Bounding box Object

✓ Segments Object

✓ Trapezoidal Map Object

No segment inserted, the initial trapezoid (the bounding rectangle) is shown.
We can add by hand new segments, or we can generate them randomly or load them with files.



Time: 4e-05

Time:

Time:

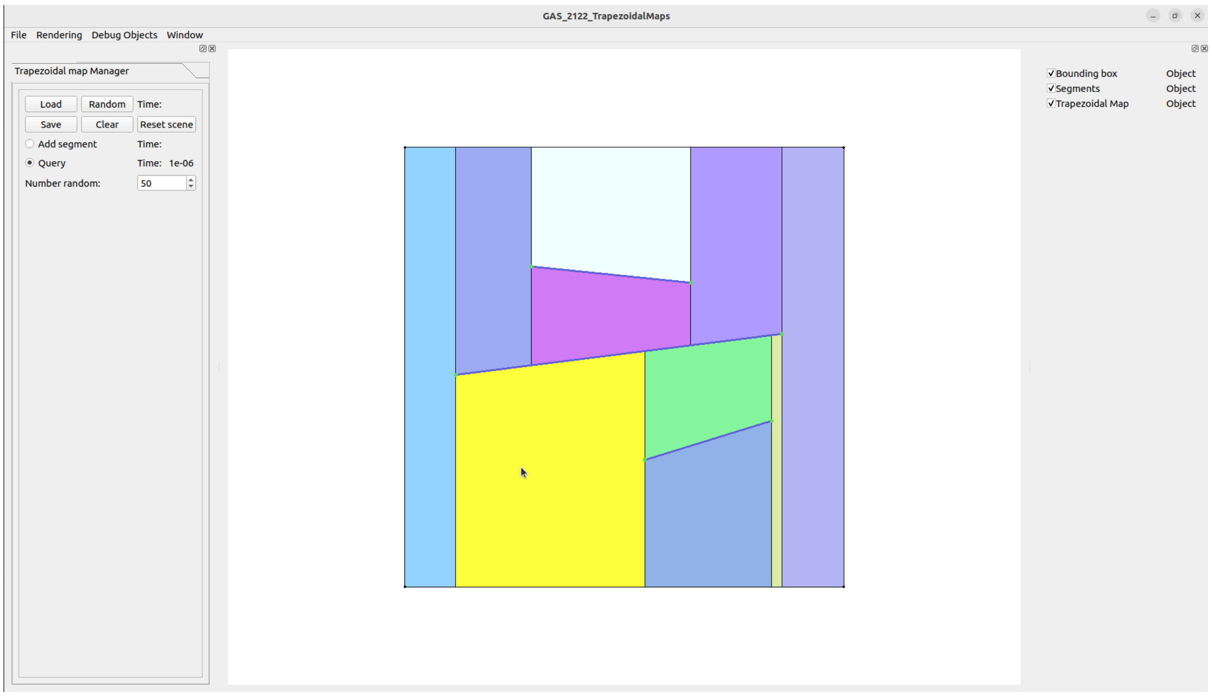
Number random: 50

✓ Bounding box Object

✓ Segments Object

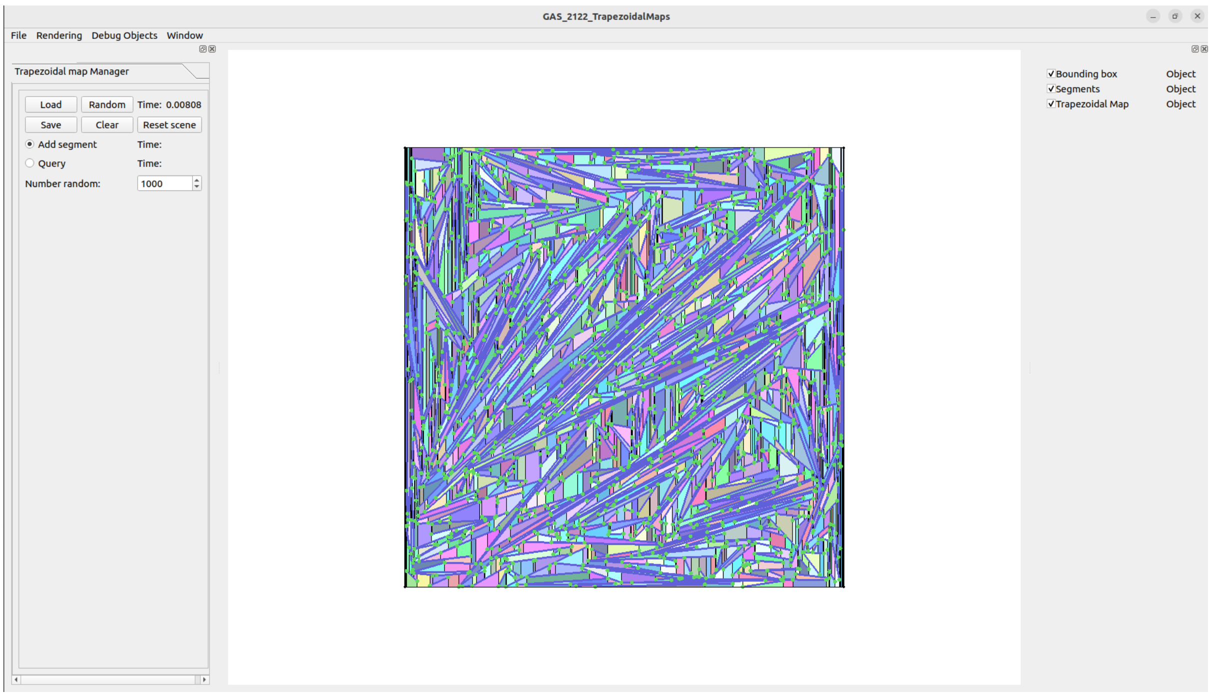
✓ Trapezoidal Map Object

The above picture shows the trapezoidal map obtained loading the slide example file.
If we query the trapezoidal map, the trapezoid query result will be highlighted with a bright yellow color.



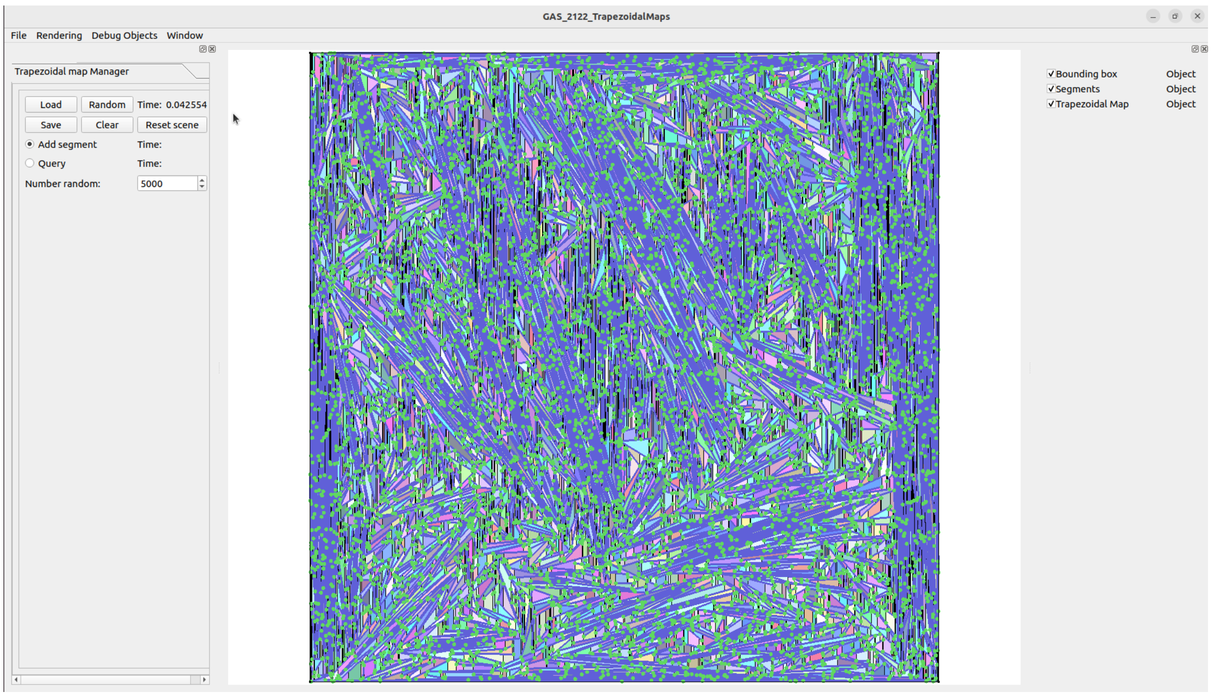
Time:
Add segment Time:
[✓ Query] Time: 1e-06
Number random: 50

The result of the query.



Time: 0.00808
[✓ Add segment] Time:
Query Time:
Number random: 1000

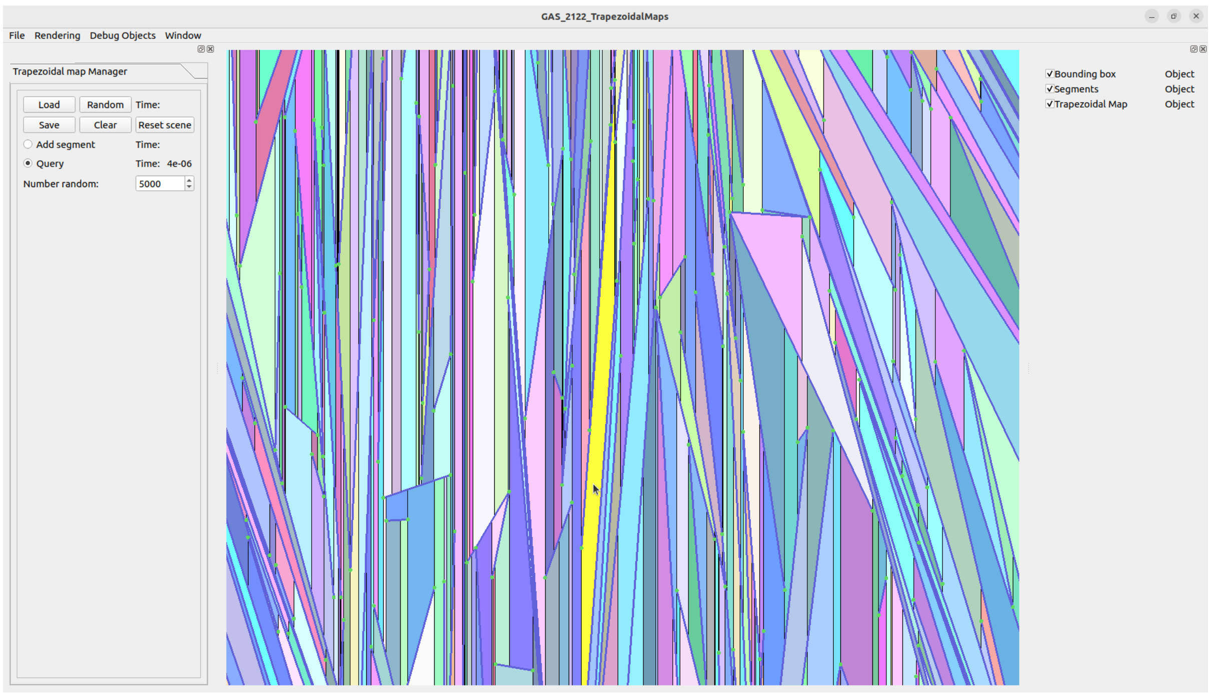
Example with 1000 trapezoids:



Time: 0.042554

Number random: 5000

Example with 5000 trapezoids.



Query Time: 4e-06

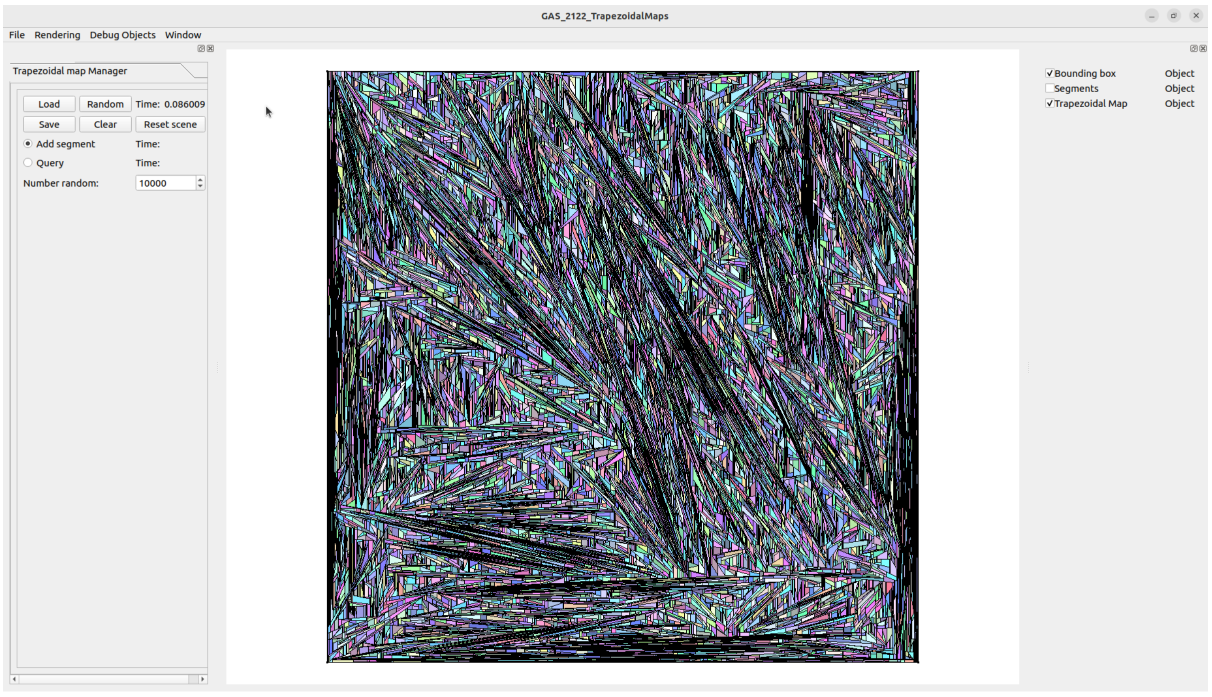
Number random: 5000

Example query with 500 trapezoid (zoomed in)

Time for the test files in the dataset:

| **File** | **Time** |
|---|---|
| random10.txt | 0.000126 |
| random20.txt | 0.000233 |
| random40.txt | 0.00032 |
| random50.txt | 0.000382 |
| random100.txt | 0.000752 |
| random500.txt | 0.003961 |
| random1000.txt | 0.007391 |
| random2000.txt | 0.016049 |
| random5000.txt | 0.041942 |
| simple_1 | 2.5e-05 |
| simple_2 | 2.8e-05 |
| simple_3 | 1.6e-05 |
| simple_4 | 3.2e-05 |
| simple_5 | 3.7e-05 |
| simple_6 | 3.8e-05 |
| simple_8 sharedpoint | 3.4e-05 |
| simple_9 | 4.5e-05 |
| simple_10_sharedpoint | 3.5e-05 |

Test with 10000 random segments: (removed segments from the visualization to make the trapezoids “more” visible)



Time: 0.086009

Time:

Time:

Number random: 10000

[✓ Bounding box]

Segments

[✓ Trapezoidal Map]