Final Project

In the final project of this course you have to develop a 2D Interactive Delaunay Triangulation algorithm.

A bonus point is given if it is implemented the optional feature of drawing the complementar Voronoi diagram of a given triangulation.

Therefore, the maximum grade will be 31/30.

The algorithm must be implemented in C++, and a base project is given.

1. Base Project

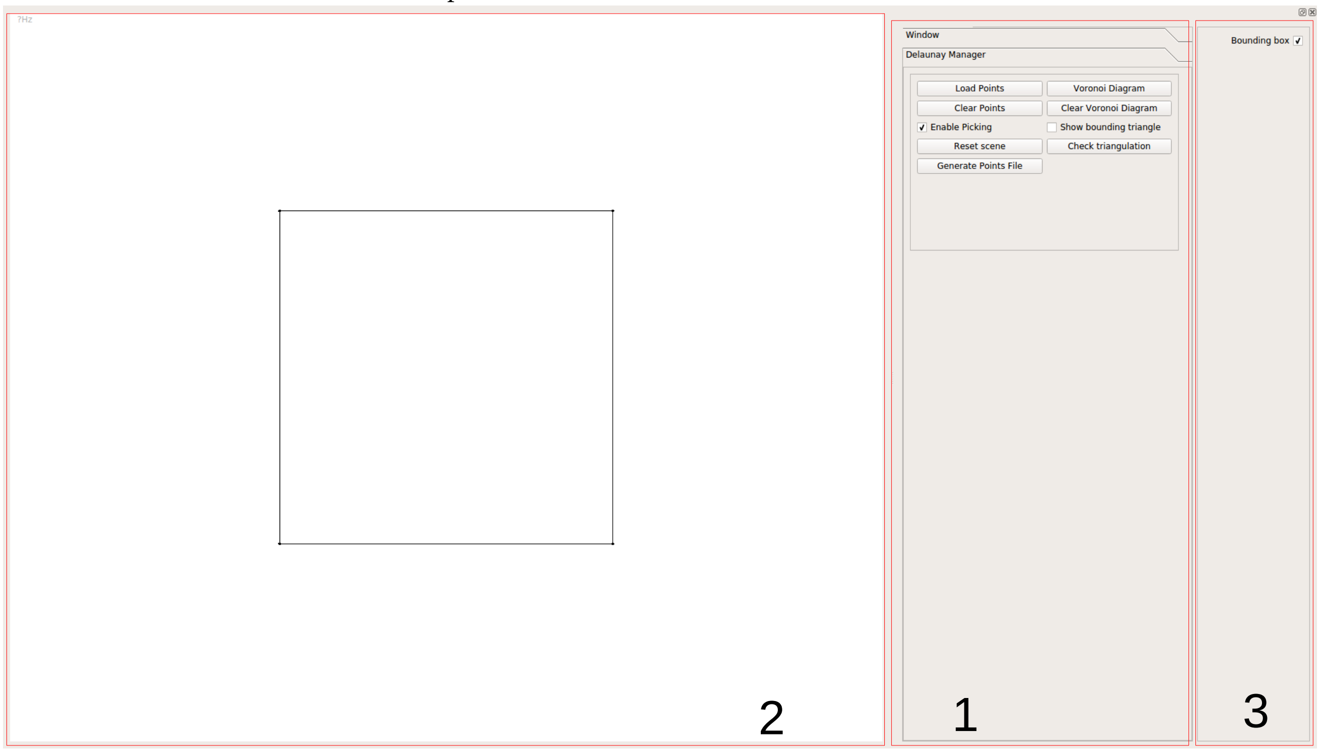
1.1. Overview

The Base Project is organized in modules. There are two main modules: the "common module" and the "viewer module". Every module is organized inside a .pri file which is included inside the main .pro file of the project. **You mustn't modify the folders or the files inside the folders associated to the modules**. All your files and folders must be organized in the main folder of the project.

In the main folder of the project you can find the two module folders, two folders called "gui" and "utils", and the main.cpp. The "gui" folder contains the DelaunayManager (will explain what it is later) and the "utils" folder contains some utility functions.

When you compile and run the project, this window, which we will call "MainWindow" will be opened.

The MainWindow is divided in three parts:



1. Here you can find all the managers. A manager is a QFrame which represent a GUI with a set of operations that allows to manage some objects. Here there are two managers: the Window Manager (which manages the entire MainWindow, you don't have to care about it) and the Delaunay Manager, which will manage your Delaunay Triangulation and the Bounding Box (that is already implemented). A bunch of buttons and checkboxes are already put inside the manager. You can see how a manager is added to the MainWindow inside the main.
2. This is the GL Canvas. It is a Canvas that allows to draw objects with opengl calls. In this project, the MainWindow will take care to draw inside the canvas all the "DrawableObjects" passed to it.

3. Here you can find a set of check boxes: every checkbox is linked to a DrawableObject that is inside the MainWindow. At the beginning, there is only a DrawableObject which is "Bounding Box", and it is drawn inside the GL Canvas.

1.2. *Drawable Objects*

How the viewer works? The Viewer can draw inside the GLCanvas every objects which implements the interface "DrawableObject" (viewer/interfaces/drawable_object.h). Every class that implements the interface "DrawableObjects" has to implement the three pure abstract member functions draw(), sceneCenter(), sceneRadius(). Inside the draw, you have to put the OpenGL code that draws the object. You can see an example of a implemented DrawableObject in the class DrawableBoundingBox2D (viewer/drawableobjects/drawableboundingbox2D.h).

You don't have to study OpenGL. Your triangulation data structure will need to draw only points and lines, and there are a few functions which do this automatically. Check the functions "drawPoint2D" and "drawLine2D" in the file "viewer/objects/objects.h".

1.3. *Managers*

Let's look at the Delaunay Manager, which inherits the QFrame class. A QFrame is composed of an header file, a source file and an ui file. Go to "Forms/gui/delaunaymanager.ui". Now you can see the Gui of the manager. When the application is running, every time a button is pressed a special member function (Qt calls them "slots") will be called. For example, if you want to modify the member function associated to the "Load Points" button, you can right click on the button, select "Go to slot", select "clicked" and click OK. You should now be inside the member function called "DelaunayManager::on_loadPointsPushButton_clicked()".

In this project, the main purpose of the DelaunayManager is to manage a Delaunay Triangulation. Therefore, you have to put your Delaunay Triangulation Data Structure as an attribute of the DelaunayManager (check the boundingBox attribute to understand how it works). If you want to draw your Delaunay Triangulation, your data structure needs to implement the "DrawableObject" interface, and then you can send your triangulation to the canvas, calling the method mainWindow.pushObj(), which takes a const pointer to a DrawableObject.

2. Specifications

Use the points BT_P1, BT_P2 and BT_P3 for the bounding triangle. These points are declared inside the file "delaunaymanager.cpp". All the points of your triangulation have to stay inside the bounding box declared inside the DelaunayManager.

Inside the DelaunayManager, you have to fill some slot member functions associated with buttons:

- on_clearPointsPushButton_clicked: here you have to clear your triangulation (delete all the triangles and clear your DAG);
- on_showBoundingTriangleCheckBox_stateChanged: here you just need to draw also the bounding triangle when arg1 is true;
- on_loadPointsPushButton_clicked: after the points are loaded from file, you need to insert all the points in the triangulation;
- point2DClicked: this function is called every time a point is clicked. Insert the point "p" inside the triangulation.
- on_checkTriangulationPushButton_clicked: here you need to extract the triangulation in two data structures: std::vector<Point2D> and Array2D<unsigned int>: in the vector there will be the set of points of the triangulation, the Array2D is a 2D matrix with n_triangles rows and 3 columns: every entry (i,j) is the index of the "i-th" triangle.
- Do not write code inside the member functions with the comment "do not write code here".

The bonus point asks to create a Voronoi Diagram using the Delaunay Triangulation. You don't have to create a proper data structure to store a voronoi diagram: you just have to create a DrawableObject

containing a set of points and lines which draws these elements.
If you find a bug on the code, or if you have questions, please write an email at muntoni.alessandro@gmail.com and cordafab@gmail.com or open a discussion on the forum on moodle.

3. How to Send the Project

This year, you are asked to send your final project through github or bitbucket, and you are asked to commit often during the development of the project. Projects with a too small number of commits (or no commits at all) will be rejected.

Access here to have a private repository assigned: https://classroom.github.com/assignment-invitations/ade7bae671e014052787740fd1e141a1

1. Access to your github account;
2. Accept the assignment in order to have a private repository;
3. Push the base project on your repository;
4. Once the project is developed, upload on on moodle a pdf file which documents how you organized the project (how data structures and algorithms are linked, how are organized your file, etc.). Put inside the pdf file the name of your github repository.

Remember that this is an *individual* project: you can collaborate in order to solve of high level problems, but projects with similar pieces of code will be not tolerated (*both* projects will be rejected).

4. Virtual Machine vs Physical Machine

We are giving to you the base project and a Virtual Machine which is ready to use.

I suggest you to avoid to use the Virtual Machine and, if it is possible, to compile the base project on your physical machine. However, it is almost impossible to compile it on Windows (you can try to link manually all the required libraries: good luck!). Therefore, if you have a PC and if you can, install a linux based (better an ubuntu-based) OS and run install.sh script in order to install all the required libraries and compile the project. If you have a Mac, please send an email to Fabrizio (cordafab@gmail.com) which will help you with the installation and linking of all the required libraries.

If you can't, use the Virtual Machine.

5. Guidelines for a Good Final Project

1. Before submitting, rename the .pro file with a name of this format: <matr>_<surname>_<name>.pro.
2. The base project compiles with zero warnings. Make sure to submit a project with zero warnings. Warnings are indicative of bad and error-prone coding. A zero-warnings code doesn't mean that is good code, but good code always produces zero warnings.
3. Separate the definition of a class (files .h) from its implementation (files .cpp), and do not use "using namespace ..." on headers files (why? http://stackoverflow.com/questions/5849457/using-namespace-in-c-headers).
4. Try to avoid the usage of global variables and other shortcuts. Try to follow the Object Oriented paradigm as best as you can (why? http://c2.com/cgi/wiki?GlobalVariablesAreBad).
5. Please organize your code following the Object Oriented Paradigm. **Keep separated algorithms from data structures**, write short (when it is possible) methods which solve standalone problems.
6. Use const keyword. If you don't know how to use const keyword, take a look at this tutorial: (http://www.cprogramming.com/tutorial/const_correctness.html).
7. Follow the guidelines given during the short C++ course!

6. TIPS

6.1. *DEBUGGING*

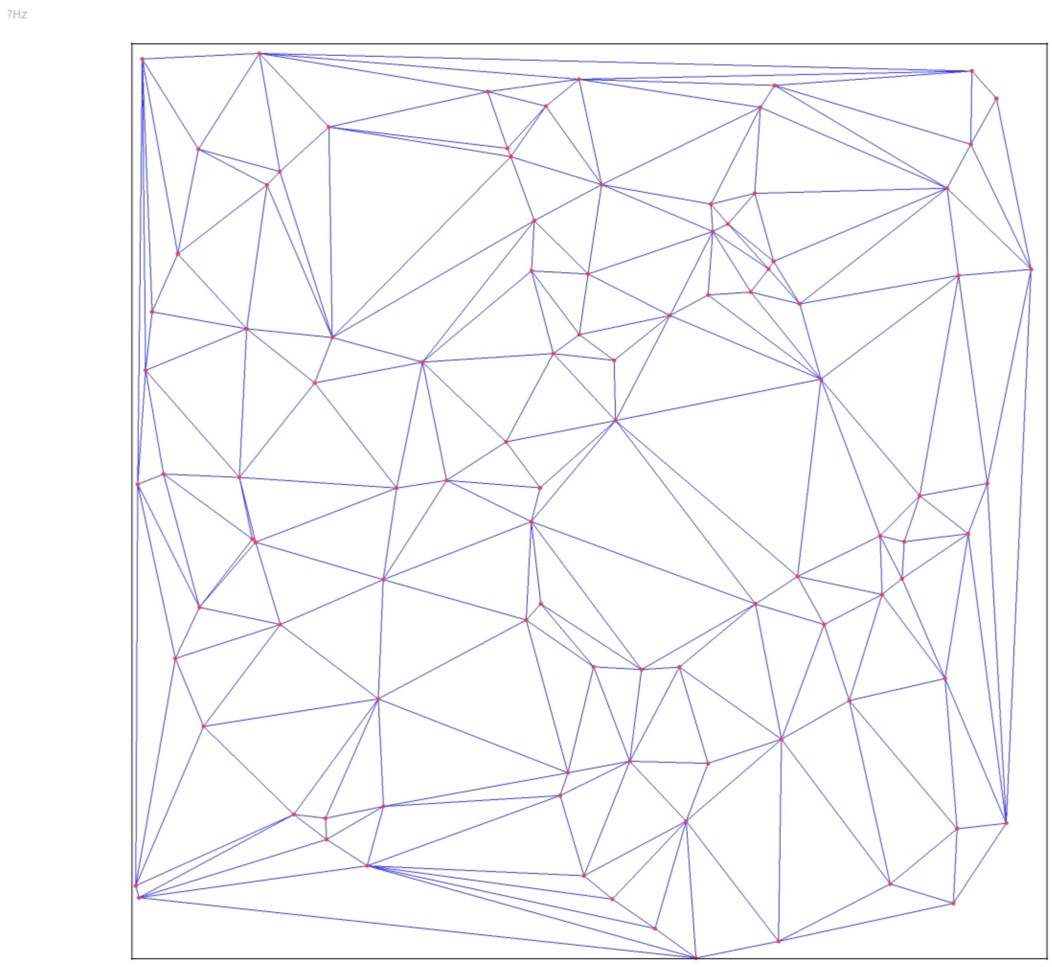
Debugger is your friend and it is a very powerful instrument. 90% of the situations putting a break point is faster (and a smarter choice) than putting std::cout (or std::cerr) everywhere.

The debugger allows you to find the exact point where your application is crashing and why, and to see the state of all the variables in every scope during the execution of your application. If you never used a debugger, this is the best time to start and learn.

6.2. *RESULTS*

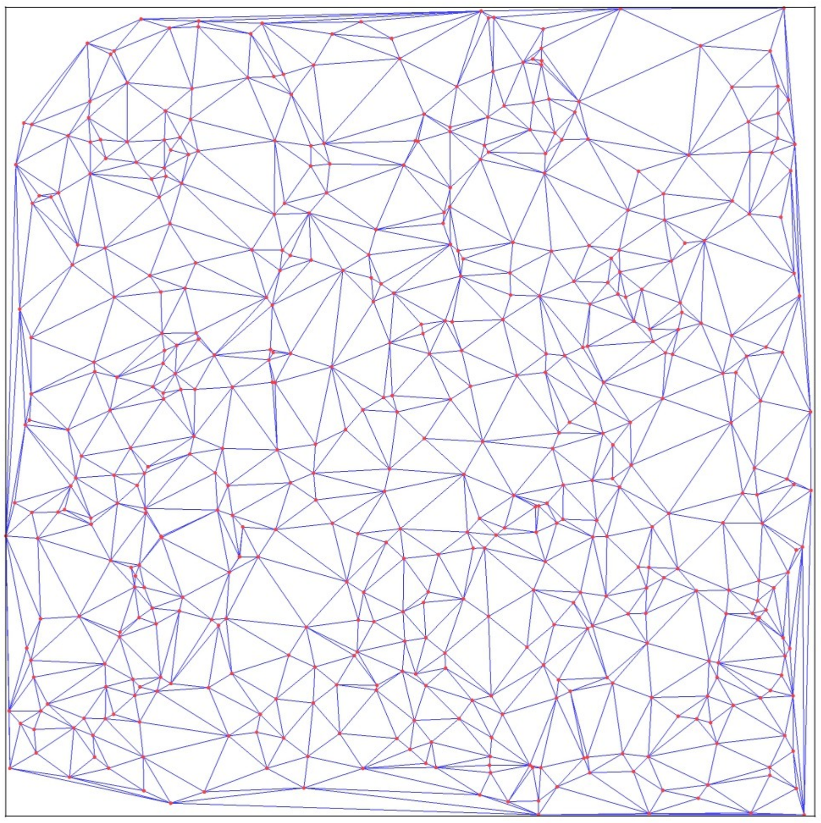
Here you can find some screenshots of the results you have to expect on the given input files:

100:



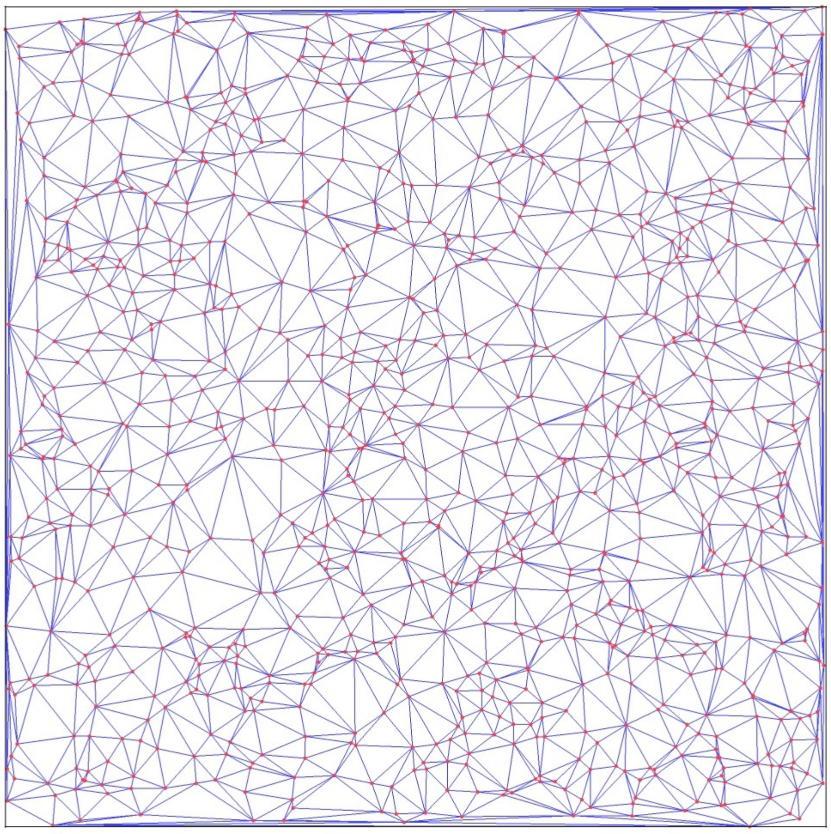
500:

0.4Hz



1000:

0.5Hz



7. Grade and Deadlines

The grade will be composed by:

- 7/30: Correctness of the project;
- 5/30: Documentation;
- 10/30: Structure of the project, code modularity and style;
- 8/30: Efficiency;
- 1/30: Bonus point.

The first deadline is for all the students that need to have the grade registered before August 20th for the ERSU scholarship. If you want your grade before August 20th, you must submit your project by August 4th at 23:59.

The deadline for the project is August 31st. If you submit your project after August 31st at 23:59, your grade will be penalized by a point every month. This is a summary table:

Submit by	**Maximum Grade**
23:59 August 4th	31/30 by August 20th
23:59 September 30th	31/30
23:59 October 31st	30/30
23:59 November 30th	29/30
23:59 December 31st	28/30
23:59 January 31st	26/30
23:59 February 28th	24/30

After March 1st, you will have to submit the project with the specifications of 2018.