UNIVERSITÄT BERN

Digital 3D Geometry Processing
Exercise 7 – Delaunay Triangulation

Handout date: 09.04.2019

Submission deadline: 16.04.2019, 13:00 h

What to hand in

A .zip compressed file renamed to `Exercise`n`-GroupMemberNames.zip` where n is the number of the current exercise sheet. It should contain:

- Hand in **only** the files you changed (headers and source). It is up to you to make sure that all files that you have changed are in the zip.
- A `readme.txt` file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems. Indicate what fraction of the total workload each project member contributed.
- Other files that are required by your `readme.txt` file. For example, if you mention some screenshot images in `readme.txt`, these images need to be submitted too.
- Submit your solutions to ILIAS before the submission deadline. Late submissions will receive 0 points! The total points of this homework is 6.

Coding Exercise (6 pts)

The goal of this exercise is to implement the Delaunay Triangulation algorithm which maximizes the minimum angle of all the angles of the triangles in the triangulation. After correct implementation, the demo will help you to build a connection between the Voronoi diagram and the triangulation in an interactive way.

- By clicking the button `Create Initial Mesh` in the `Plugin-DGPExercises`, the demo starts with an initial triangle mesh containing two triangles and four vertices. The corresponding Voronoi cells are visualized as projections of cones on the base plane in different colors. The Voronoi edges are the projections of the cone intersections on the base plane (See Figure 1). In case the view is changed, you can restore the perspective by pressing the `Set 2D View` button.

- In the demo, you need to incrementally add points to the triangle mesh. The picking mode should be activated by clicking on the arrow icon in the OpenFlipper toolbox. Then you can left click with your mouse inside the initial mesh to add a point. If the newly added point is on a mesh edge, it will perform an edge split; otherwise it will do a face split. The Voronoi diagram will be updated simultaneously.

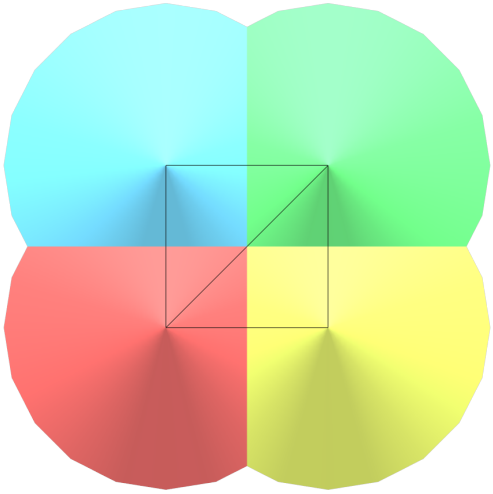

Figure 1: The initial mesh and Voronoi diagram.

- Add your code in the function `insert_point(...)` in the file `DelaunayTriangulation2D.cc`. Here you will need to implement the algorithm to check if the new triangles meet the Delaunay condition. The function `is_delaunay(...)` in the file `DelaunayTriangulation2D.cc` works as a basic operation that you will use for checking, where you need to fill in the missing code which is to test if an edge is Delaunay or not.