Digital 3D Geometry Processing
Exercise 3 – Geometry Representations

Handout date: Monday 30.Sep.2019

Submission deadline: Thu 10.Oct.2019, 23:00 h

What to hand in

- From your modified source code, please upload **only** the file **viewer.cpp**, not in a subfolder, and not compressed.
- We run an automated script to compile and test your homework, so it is necessary for us to find the file **viewer.cpp** directly in your uploaded files.
- If there are comments on or problems in your code, put them as comments in the **viewer.cpp** file.
- Submit your solutions to Moodle before the submission deadline. Late submissions will receive 0 points!

1 Theory Exercise (Not graded)

1.1 Derive a signed distance function in 2D for a line L of the form $y = x$

1.2 Define a planar curve that has a sharp corner (discontinuity of normal vector) using only polynomials as coordinate functions. Please give a parametric representation.

1.3 Denote $\{(x(\frac{k}{N}), y(\frac{k}{N}))|k = 0,\ldots,N\}$ as a uniform sampling of a 2D curve $(x(t), y(t)), t \in [0,1]$. As the number of sampling N increases, does chord length monotonically increase (non-decrease). If yes, give a proof; if no, give a counterexample.

1.4 Find a sequence of 2D curves $\mathbf{C}_i(t), t \in [0,1]$ which satisfy:

$$\lim_{i\to\infty} \mathbf{C}_i(t) = \mathbf{L}(t), \forall t \in [0,1] \text{ where } \mathbf{L}(t) \text{ is a straight line segment in} [0,1]$$

the length of $\mathbf{C}_i(t)$ does not converge to the length of $\mathbf{L}(t)$ in $[0,1]$

2 Coding Exercise (100 pt)

If you use an IDE such as the Visual Studio 19 or Clion, open the `CMakeLists.txt` from the main directory as a project. Make sure to run the target `iso_contouring` and not the target `bin2c` which might be selected by default (in VS 19 this is done by right-clicking `iso_contouring` in the solution explorer and selecting "Set as Startup project").

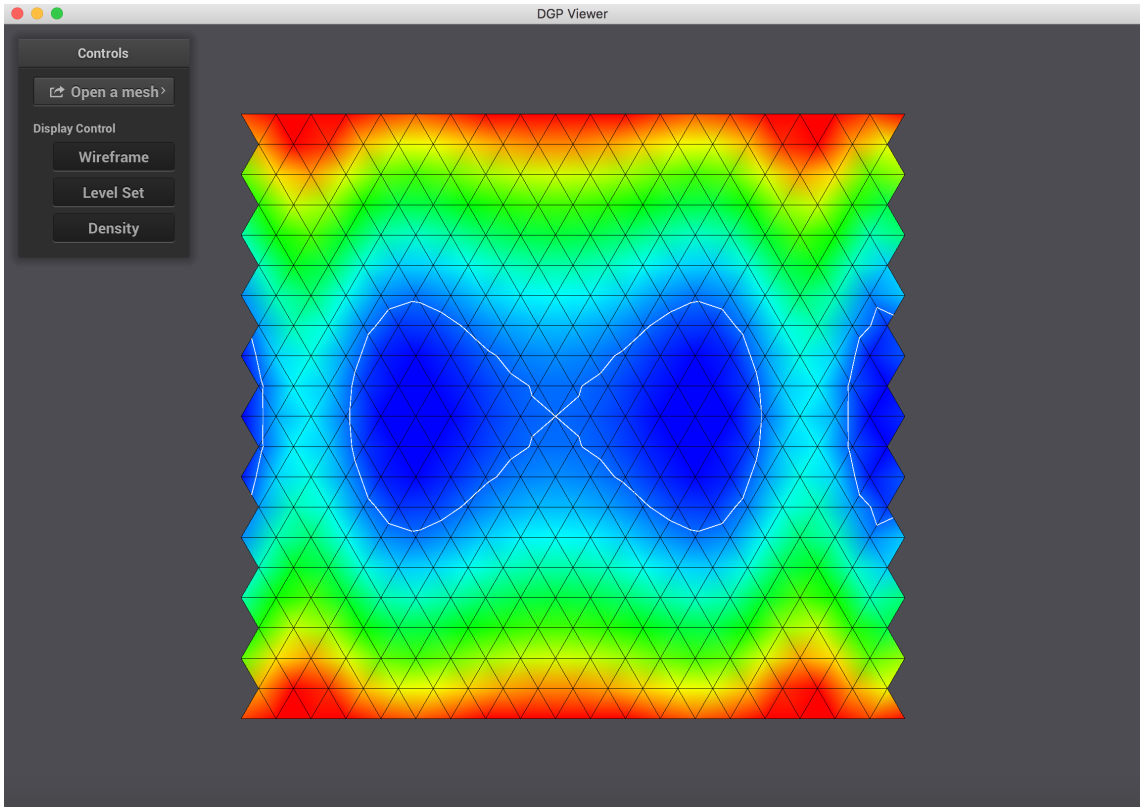

Figure 1: Final result for the *Regular Mesh 2* and function $F(x,y) = y^2 - \sin(x^2) = 0$.

Once the application is started, the mesh will be processed and an OpenGL based graphical user interface shows the resulting mesh. The simple GUI allows you to navigate the loaded mesh with the following mouse controls:

- Left-MouseMove: rotate view;
- Middle-MouseScroll: zoom in and out;
- Middle-MouseMove or Ctrl+Left-MouseMove: drag the mesh.

Use the buttons on the left side of the GUI to activate or deactivate different rendering modes.
For this exercise you will need to fill in the missing code in the file `viewer.cpp`. You are given a triangular mesh with a list of vertex coordinates stored in `v_positions` and a list of vertex indices forming each triangle stored in `triangle_ids`. For each vertex, the third coordinate is zero, since the meshes are two dimensional.
The goal of this exercise is to implement a contouring method like *marching triangles algorithm* but on triangles rather than squares. For each vertex you can compute a scalar iso-value from an implicit function. Then, for each triangle, check if the signs of the vertex iso-values are not all the same. If the signs are different use linear interpolation to compute the edge that passes through that triangle. Add two end-points of that edge in a vector `segment_points`. Here `segment_points` is a vector of points, so add the two end-points one after the other. Note, that this means that a point that appears in two edges will appear twice in this list.
Test the following implicit functions:

- $F(x,y) = \sqrt{x^2 + y^2} - 1 = 0$

- $F(x,y) = y^2 - \sin(x^2) = 0$
- $F(x,y) = \sin(2x + 2y) - \cos(4xy) + 1 = 0$
- $F(x,y) = (3x^2 - y^2)^2 y^2 - (x^2 + y^2)^4 = 0$

You can implement these test functions in `Viewer::iso_func_example(Point v_pos)`.

The marching triangles method itself needs to be implemented in `calc_iso_contouring`, which receives a function `std::function<Scalar(Point)> iso_value` on which the algorithm should be applied. In `calc_iso_contouring`, the function values at the vertices have already been computed and stored in the vector `v_iso`. Vertex indices per triangle are stored in a vector `triangle_ids`. Read the comments inside that method for further details.

To see the result on different meshes, simply choose one of the provided test meshes from drop-down menu in GUI. One correct output is shown in Figure 1. For testing in the GUI, the function from `Viewer::iso_func_example(Point v_pos)` will be passed `calc_iso_contouring`, but our automated test will pass other functions, so only use the function `std::function<Scalar(Point)> iso_value` to evaluate function values.

Unit Test

We are introducing automatic grading this year. Your homework is graded by comparing the difference between your program's output and our ground truth result. We included a reduced version of the automated test with the code. To test your code, please run the target `isotest`. If your program passes through all testing cases, `isotest` will return the following information:

```
Filters:
===============================================================================
All tests passed (9 assertions in 2 test cases)
```

Otherwise, it will give you a hint in which test case your program failed:
Note that if your code passes it is an indicator that your output format and content is correct, but we will do the testing with more and other functions.

```
Filters:
[
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
isotest is a Catch v2.7.1 host application.
Run with -? for options

-------------------------------------------------------------------------------
test case for mesh 1
  y*y - sin(x*x)
-------------------------------------------------------------------------------
/Users/ziqwang/Documents/GitHub/iso_contouring/iso_contouring/isotest.cpp:58
...............................................................................

/Users/ziqwang/Documents/GitHub/iso_contouring/iso_contouring/isotest.cpp:69: FAILED:
  REQUIRE( contour_distance(app->segment_points, correct_pts) < 1e-4 )
with expansion:
  0.7124458551 < 0.0001

-------------------------------------------------------------------------------
test case for mesh 1
  y*x - exp(x*x) + log(std::abs(x + y))
-------------------------------------------------------------------------------
/Users/ziqwang/Documents/GitHub/iso_contouring/iso_contouring/isotest.cpp:72
...............................................................................

/Users/ziqwang/Documents/GitHub/iso_contouring/iso_contouring/isotest.cpp:83: FAILED:
  REQUIRE( contour_distance(app->segment_points, correct_pts) < 1e-4 )
with expansion:
  0.5274373293 < 0.0001

===============================================================================
test cases: 1 | 1 failed
assertions: 4 | 2 passed | 2 failed
```