University of Stuttgart
Institute for Visualization and Interactive Systems (VIS)

Thomas Ertl, Steffen Frey *Stuttgart, 22. 05. 2020*

Scientific Visualization (Assignment 5)

Exercise 5. 1 [3 Points] Delaunay Triangulation - Edge-Flip

Derive a valid Delaunay triangulation for the given points in Figure 1 by applying the plane-sweep algorithm from left to right. Use the edge-flip algorithm on the triangulation until the mesh meets the Delaunay properties. Submit a picture of the intermediate Delaunay triangulation for every step of the sweepline. You can do this by either drawing the triangulation

Hint: You can create a picture of every step by providing the points to the *delaunay.py* script. The script takes triples of vertex indices to draw triangles. (Example of two adjacent triangles: *python delaunay.py '[[0,2,1],[0,1,3]]'*.) You can look up the associated indices in Figure 1. For achieving full points in this task usage of the drawing script is NOT required. You may also submit hand-drawn images.

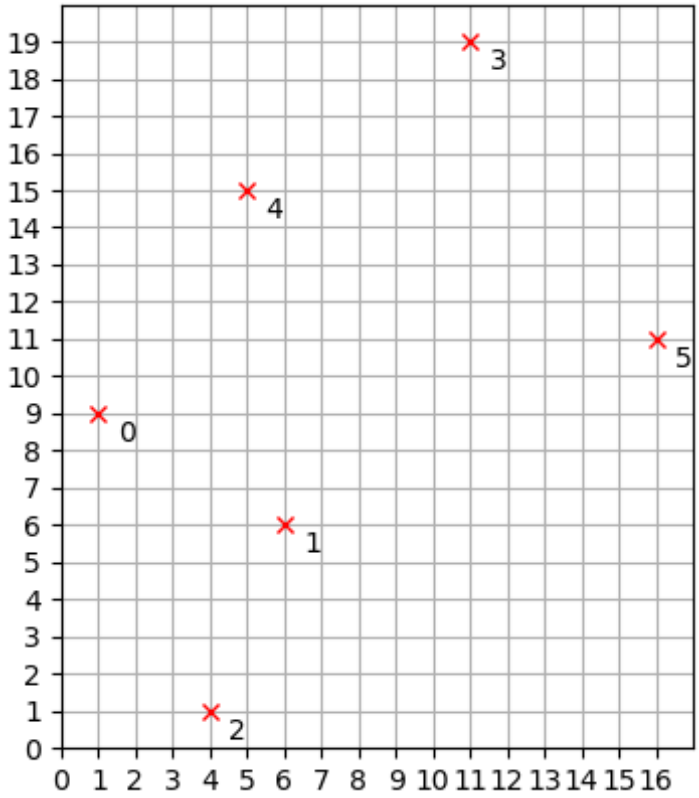

Figure 1: Delaunay-Triangulation. The numbers besides the vertices indicate the vertex index for the drawing script.

Exercise 5. 2 [3 Points] Inverse Distance Weighting

Inverse Distance Weighting can be used for interpolation within scattered data. In Figure 2 such a data set is depicted. $P_i(x, y, d)$ with $i \in \{1, 2, 3, 4, 5, 6\}$ are the given points, where x and y are the coordinates and d is the assigned value. Interpolate the value d at the points P_7 and P_8 considering only neighbors within a radius of 3 and using exponent $p = 2$ in the basis functions. For both P_7 and P_8 give all participating points' distances, their weights (evaluated basis functions) and their actual contribution. Also submit the interpolated values for P_7, P_8 respectively.

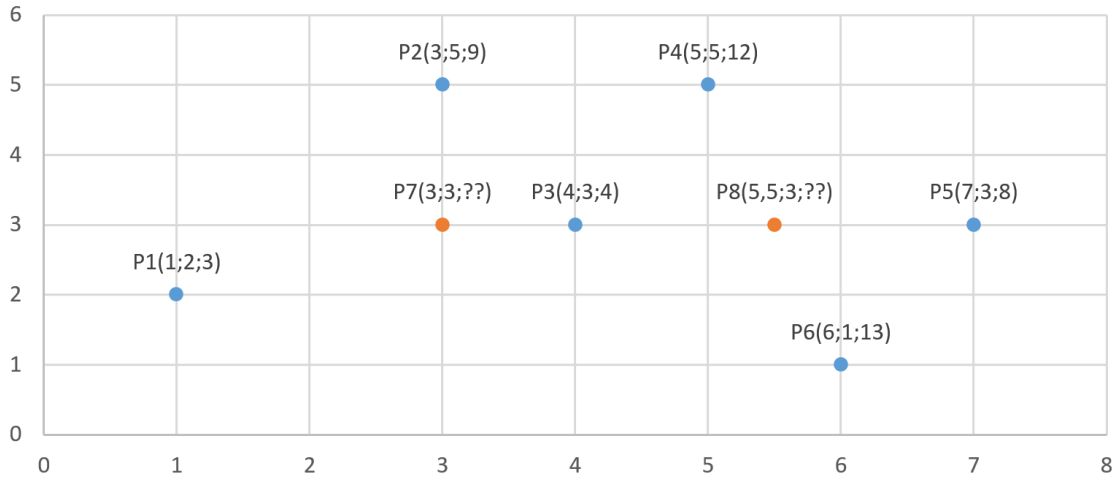

Figure 2: Plot of scattered data

Exercise 5. 3 [1 Points] Interpolation inside a prism

Take a look at the grid in Figure 3. It shows a prism with its edges at positions A to F and a point P somewhere within. Describe shortly how you would go about interpolating the value for given point P. You don't need to provide a formula, just describe a suitable method or combination thereof.

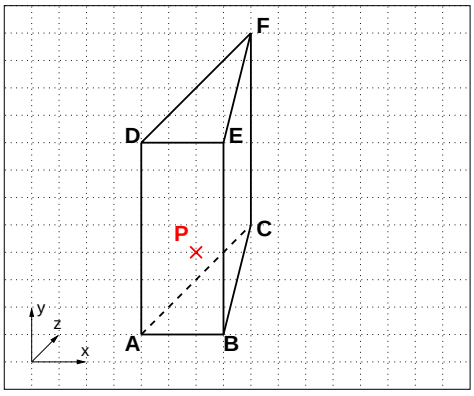

Figure 3: Prism with values defined on its edges

Exercise 5. 4 [5 Points] Paraview: Simple Gradient Plugin

In this exercise, you will write a plugin for ParaView which takes a regular grid with scalar data as input and calculates the gradient of the field using central differences. The gradient will then later be visualized using the `Glyph` filter. Since the boundaries are a special case where central differences cannot be used, you can just set the gradients there to zero. Of course you can—if you like—calculate the gradients at the boundaries using forward or backward differences. Like in the past assignments, the plugin is given inside a docker file and the setup works the same.

To calculate the gradient, you will need to fill in the missing code in `exercise_04.cxx`. The comments in the source file will help you understand the task. You will need the documentation for the `vtkImageData` object at `http://www.vtk.org/doc/nightly/html/classvtkImageData.html`.

To view the results of your calculation, first load the Rayleigh-Taylor data set `rayltayl.vti` provided in the data folder of the plugin. With the data set selected in the `Pipeline Browser`, navigate to the `SimpleGradient` filter in the `SciVis` folder of the `Filters` menu and apply it. (*Hint:* The `Information` pane gives you additional information about the extent of the data set and the arrays added, which might help you debug your code).

To visualize the gradients calculated, now add a `Glyph` filter using the gradients as input. Before you apply the filter however, you should adjust a few settings:

- Set the `Scale Array` in the `Scale` section to `Gradient`. This will cause the vectors to be scaled according to their magnitude.
- Set the `Scale Factor` to 0.0015, which is just a linear factor for the scaling.
- Set `Glyph Mode` in the section `Masking` to `Uniform Spatial Distribution`. 2000 should be a good number for the `Maximum Number Of Sample Points` value.

To complete the assignment, hand in the `exercise_04.cxx` file you modified, along with a screenshot showing the gradients visualized with arrow glyphs (e.g. something similar to the images shown in Figure 4). You can create a screenshot in ParaView via `File > Save Screenshot...` The following command will copy the required file from docker into your host systems file system:

```
$ docker cp scivis_sheet_5:/root/scivis-2020/source/plugins/
    assignment_05/modules/exercise_04/exercise_04.cxx ./
```

Submission Deadline: 2020-05-29, 23:55

please hand in your submission through the ILIAS system.

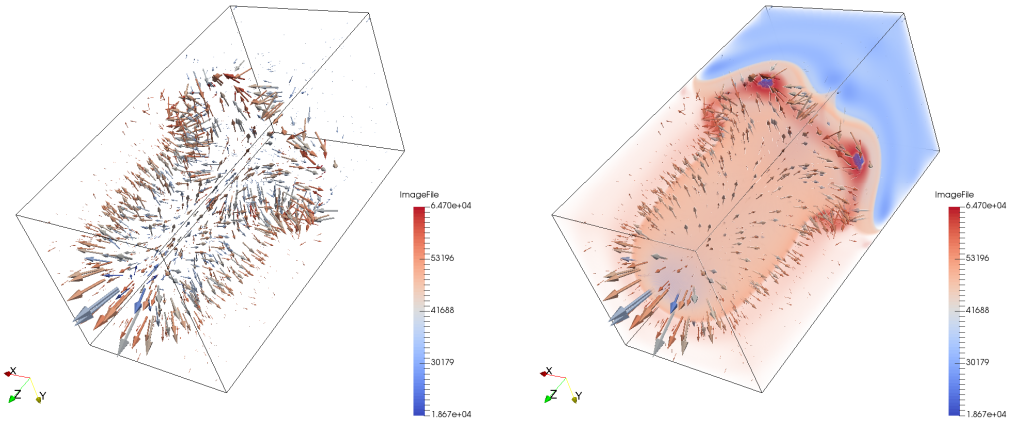

Figure 4: Left figure shows the output of the `SimpleGradient` plugin, visualized using the `Glyph` filter. The right figure adds a volume rendering for context.