Assignment 3

The goal of this assignment is to calculate a path length in a mountainous area but first a number of preliminary problems have to be solved.

A In the program *assignment3A2013.py* a number of random points in 2D are generated. Using these points a Delaunay triangulation is created and visualized. Also a point *p* is given. Using *p* do a point-location, i.e., determine in which triangle *p* is located. A brute force approach is ok. You may use the python function from Thomas his presentation, it is on Nestor. High-light the edges of the located triangle.

B A standard problem in computational geometry is to determine whether and where two line-segments *s1* and *s2* in 2D, say the x,y plane, intersect. Let the endpoints of *s1* be *p1* and *p2*, and of *s2* be *p3* and *p4*. The points on *s1* are given by λ_1 *.p1 + (1-* λ_1*) .p2*, where 0<= λ_1<=1. Similarly the points on *s2* are given by λ_2 *.p3 + (1-* λ_2*) .p4*, where *0<=* λ_2*<=1*. Calculating the point of intersection means that two equations in the parameters λ_1 and λ_2 have to be solved, where the equations are given by
λ_1 *.p1 + (1-*λ_1*) .p2=* λ_2 *.p3 + (1-*λ_2*) .p4*. Only when it holds that 0<= λ_1<=1 and *0<=* λ_2*<=1* the segments intersect. Use Mathematica or Maple to solve this set of equations symbolically and implement the resulting expressions in a python function.
In the following we need yet another calculation. In 3D three points *p1, p2, p3* are given and these points define a plane P. A point p0 in the x,y plane is given. Calculate the z coordinate of the point p0' in P, where the the projection of p0' in the x,y plane is p0.
Use Mathematica or Maple to derive the relevant expressions and implement these calculations in a python function.

C In the program *assignment3C2013.py* a number of random points in 2D are generated. Using these points a Delaunay triangulation *dt* is created and visualized. Two points *p0* and *p1* are given. Let the segment *s* be defined by the endpoints *p0* and *p1*. High-light the edges of *dt* intersected by *s*. Calculate the intersection points of *s* with the edges. Visualize the intersection points. Find and print the consecutive intersection points going from p0 to p1 without using the actual coordinates of the intersection points.

Now we are done in the plane and start working in 3D.

D In the program *assignment3D2013.py* a set S of points in 3D is used. S is obtained in two ways. For test situations S consists of synthetically generated points, generated by the function *generate_points()* . For the real work another set will be used, read from the file *mntElkSampledScaled.txt* by the function *read_points()*. This file stores points in 3D, resulting from a laser scan of Mount Elk in the state of Washington in the USA. It is advisable to use, during the experimental stage, the generated points because then only 1000 points are generated while the file contains 11104 points. By only considering the x and y coordinate of the points we get points in the x,y plane. Of these points a Delaunay triangulation *dt* is created, just like before. Now this

triangulation is lifted in the z direction, giving the triangulation *dtl*. This is performed by lifting every vertex in the x,y plane to the z coordinate of its corresponding point in 3D. In this way a surface in 3D is reconstructed from sample points in 3D. *dtl* is visualized. The image can be rotated by pressing the z or x key. When points are read from file, Mount Elk is visualized.

Now the problem you have to solve. On Mount Elk we want to walk along a path P. The projection of P in the x,y plane is a straight line, starting at p0 and ending at p1, where p0 and p1 are given points in the x,y plane, see *assignment3D2013.py*

P is embedded in *dtl,* so it will consist of a series of consecutive short line segments in 3D. Calculate these segments, list them in the order going from p0 to p1, print the total length of P and visualize P. Don't forget to take into account the start and endpoint of P.

Explain why the Delaunay triangulation is very well suited for this problem.

Can you think of a way of verifying the correctness of your path-length calculation for simplified input data?