Exercise 9: Mesh parameterization

The objective of the exercise is to implement a harmonic parameterization of a mesh with linear preservation property. Several steps have to be done in order to obtain a fully functional software. For simplicity, we assume that on the input, there is a mesh topologically equivalent to a disc, i.e. having a single continuous border.

Determining the border of the mesh

It is necessary to identify which vertices are located on the border of the mesh, so that their positions can be fixed in the linear system. In order to do that, the data structure has to be extended by a pair of functions `isOnBorder(int v)` and `nextVertexOnBorder(int v)`. The `isOnBorder()` function determines for a given vertex whether or not it lies on the border of the mesh, while the `nextVertexOnBorder(int v)` provides a subsequent border vertex for a given border vertex. It is important to have the border vertices provided in the order in which they appear on the border, so that their positions can be correctly determined. The implementation of these functions can be easily derived from the VV routine that supports meshes with border.

For the purposes of further processing, it is also necessary to build some helper data structures that allow distinguishing between known (fixed) and unknown vertices. For that purpose, two `List`s are created, one containing the indices of known (i.e. border) vertices (`knowns`, K elements), the other containing indices of unknown (inner) vertices (`unknowns`, U elements). Additionally, a dictionary of indices `unknownIndices` is created where for each unknown vertex its location in the respective list is stored.

Building a system of equations and righthandsides

It is necessary to build a sparse matrix of size UxU. Each row represents the condition for one unknown vertex, in terms of “spring stiffness”. Simplest way to compute the stiffness is

$$D_{ij} = \frac{1}{n}$$

In order to achieve linear preservation, the spring stiffness can be computed as

$$D_{ij} = \frac{1}{2}(\cot(\alpha) + \cot(\beta))$$

Having the stiffness computed for each edge, it is possible to compute the coefficients lambda for the matrix as

$$\lambda_{ij} = \frac{D_{ij}}{\sum_k D_{ik}}$$

The values on the diagonal of the matrix should be set to -1.

Coefficients lambda of unknown vertices form the matrix of the system, while coefficients of known vertices form the righthandside vectors, together with the known UV coordinates.

Solving the system (5 points)

In order to solve the system, use the `meta.numerics` library. It provides classes for representing the sparse matrix (`SparseSquareMatrix`) and a vector (`ColumnVector`). The `SparseSquareMatrix` provides the method for solving the system `Solve(ColumnVector)`. The system has to be solved twice, once for the U coordinates and once for the V coordinates.

Finally, the U and V coordinates should be retrieved for all vertices using the helper data structures.