Exercise 8: Mesh simplification

The objective of the exercise is to implement the quadrics based edge collapsing, also known as QSlim algorithm. In order to simplify the implementation, there is a simplification framework available that handles the priority queue of edges that are considered for collapsing. You can use it by simply adding the `Simplification` project to your solution (after copying it to the appropriate location). You can invoke the simplification from the client program using following code:

```
BasicSimplificator simp = new BasicSimplificator(mesh, double.MaxValue, 1000);
mesh = simp.Simplify();
```

There are only two missing parts of the algorithm that must be implemented: the edge cost function, and the positioning of the new vertex. Both of these functions are already implemented in some trivial manner in the BasciSimplificator class. Your task is to change the implementation and observe the impact on the results.

Positioning a new vertex (3 points)

The new vertex is placed to the minimum point of a quadric related to an edge <i,j>.

Generally, it is necessary to keep a list of quadrics that are relevant for each vertex. At the beginning of the algorithm, this list is initialized with quadrics of incident triangles for each vertex. After each edge collapse step, the corresponding lists of quadrics are to be merged as well (keep in mind that each quadric should appear in the union only once).

Having a triangle $t =< v_1, v_2, v_3 >$, it is possible to compute its normal as $n = \frac{(v_2 - v_1) \times (v_3 - v_1)}{\|(v_2 - v_1) \times (v_3 - v_1)\|}$. The quadric can be computed from a vector $q = [n_x, n_y, n_z, -n.v_1]$ as $Q = qq^T$. For an edge <i,j>, we have a set of quadrics $Q_k, k \in n(i) \cup n(j)$, where $n(i)$ is a set of quadrics relevant to a vertex i. We can compute the aggregate quadric $Q_a = \sum_k Q_k$. The task is to find a point p, for which $p^T Q_a p$ is smallest. This is done by solving following system of linear equations

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} p = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Use either the `meta.numerics` library or the `MathNet` library to build the matrix and find the solution.

The overall task is to modify the function `SetNewVertexPosition(int srcVertex, int destVertex)` of the `BasicSimplificator` class, so that the correct new position of the vertex is computed. In order to do that, it is necessary to initialize and the array of lists of relevant quadrics for each vertex. The `HashSet<int>[]` data structure is a good candidate for the representation. The lists should probably reference an array of quadrics, one for each triangle, which should be initialized before the simplification starts.

Finally, the calling of the `SetNewVertexPosition` indicates that an edge has been collapsed. Therefore it is necessary to merge the relevant lists of quadrics, compute the optimum and update the lists.

Cost function (2 points)

The cost function is evaluated for each edge in the mesh and it determines the order in which the collapses are performed. Generally, there are many choices of cost functions, based on curvature, normal, volumes, etc. At the other hand, it seems natural to use the quadric residual, since the quadrics are also going to be used for placing the new vertex. The residual is computed as $r = p^T Q p$, where p is the optimal point as computed in the previous section. It might be a good idea to save the solution for the case that the edge is later indeed selected for collapsing.

The task should be implemented in a function that has the same header as the `CostFunctionEuclid` function in the BasicSimplificator class. The `BasicSimplificator` constructor should be adjusted accordingly.