Generic Mesh Structure in Java

Sebastien Jourdain

I. Goal

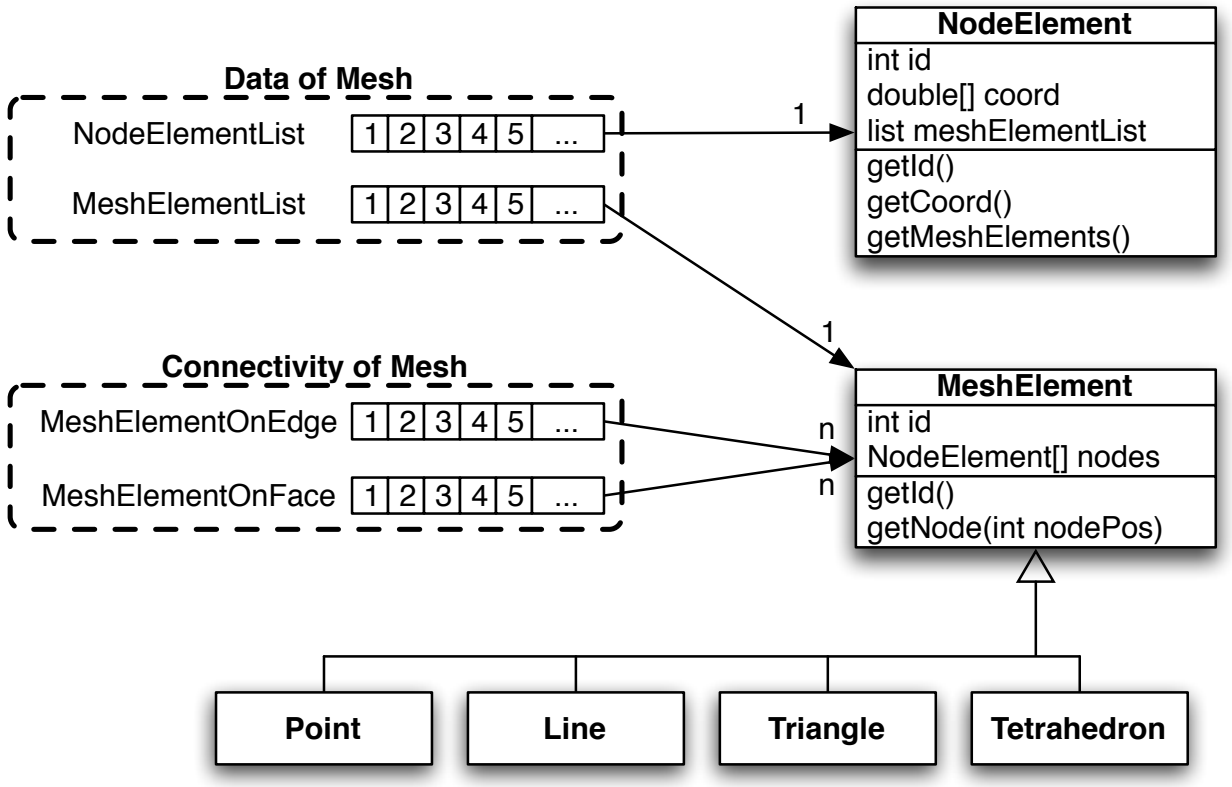
The goal of this library is to provide an high performance light weight mesh structure which provide the whole set of connectivity for simple and quick manipulation in case of mesh based dataset.

II. Principle

We do not provides the set of each mesh element type with their connectivity but instead we provide a mesh object which keep track of the mesh elements connectivity in the most effective way in term of memory cost and access time.

1. Mesh structure

Each specific mesh element provides its implicit connection with its local methods and the global mesh object provides a set of method in order to provide direct manipulation of mesh objects. By this way, the end user is not aware of the real implementation of the mesh.



2. Implementation

Code sample of a Tetrahedron method which use of implicit connectivity of the mesh.

```
/**
 * edge 0: nodes 0 1
 *      1:       1 2
 *      2:       2 0
 *      3:       3 0
 *      4:       3 2
 *      5:       3 1
 */
public Edge[] getEdgeOnNode(int nodePos) {
    Edge[] result = new Edge[3];
    switch (nodePos) {
        case -1:
            return new Edge[]{};
        case 0:
            result[0] = getEdge(0);
            result[1] = getEdge(2);
            result[2] = getEdge(3);
            break;
        case 1:
            result[0] = getEdge(0);
            result[1] = getEdge(1);
            result[2] = getEdge(5);
            break;
        case 2:
            result[0] = getEdge(1);
            result[1] = getEdge(2);
            result[2] = getEdge(4);
            break;
        case 3:
            result[0] = getEdge(3);
            result[1] = getEdge(4);
            result[2] = getEdge(5);
            break;
        default:
            throw new RuntimeException("Node position outside the range");
    }
    return result;
}
```

III. Result and conclusion

1. Memory cost

The memory cost depends of the topology of your mesh, but with classical mesh of point-edge-triangle-tetrahedron, the mesh structure is less than 1Ko by tetrahedron.

With 600MB of memory you are able to load a mesh structure of 98 000 nodes and 589 000 tetrahedron.

2. Navigation time

The connectivity is not fully stored and part of it might be calculated at each step. Thus it can take some time to list the whole face that belong to a node of our Mesh. But this time in most of case is still allowed regarding the memory cost of the mesh.

The test made on a P4 at 3Ghz in the worst case with any mesh size we can list the whole ascending connection in less than 16 ms.

3. Test made on several Gmsh mesh file

All the test have been made on the same computer which runs under Windows with a P4 at 3Ghz with 1GB of RAM.

Schema of :

- memory cost

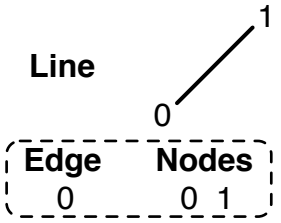
- time to load
- getConnectionOf...

IV.Annex

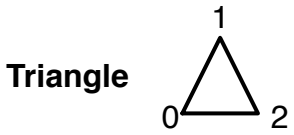
1. Result set

All the test have been made on the same computer. The computer is an AMD 64 at 2,8Ghz with 1GB of RAM.

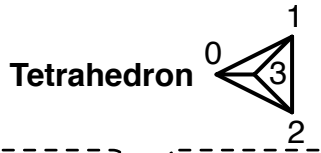
2. Gmsh implicit connectivity



Edge	Nodes
0	0 1



Edge	Nodes
0	0 1
1	1 2
2	2 0



Edge	Nodes
0	0 1
1	1 2
2	2 0
3	3 0
4	3 2
5	3 1

Face	Nodes
0	0 1 3
1	0 2 1
2	0 3 2
3	1 2 3