Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 5987-5998 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

A FRAMEWORK FOR TRIANGULAR MESH GENERATION AND MODIFICATION PROCESSES

Narcís Coll[a], Marité Guerrieri[a], Teresa Paradinas[a], Maria-Cecilia Rivara[b] and J. Antoni Sellarès[a]

[a]*Departament d'Informàtica i Matemàtica Aplicada, Universitat de Girona, Spain, {coll,mariteg,teresap,sellares}@ima.udg.edu*

[b]*Department of Computer Science, Universidad de Chile, Chile, mcrivara@dcc.uchile.cl*

Keywords: Mesh generation, Mesh modification, Object oriented development, Design patterns, Framework

Abstract. In this paper we present an object-oriented framework for implementing mesh generation and mesh modification processes. We have modeled the framework using Design Patterns, which exploit the benefits of reuse and provides an environment where new components can be easily developed. Examples of the application of the framework are presented: 2D Delaunay refinements, 2D Bisection Refinements, and terrain approximation. The framework has been implemented in C++ using the Qt libraries for creating a graphical interface.

1 INTRODUCTION

The automatic generation and modification of a mesh of a complex geometric domain is an important theoretical and practical subject in finite element analysis of engineering applications, computer graphics applications, and terrain modeling. Several public-domain and commercial software are at present available. See for example http://www-users.informatik.rwth-aachen.de/~roberts/software.html, for a list of some of these packages, and the references Shewchuck (1996); Si (2010), that discuss the Triangle and Tetgen software that respectively allow the construction of Delaunay triangulations in 2D and 3D for finite element applications.

One of the main common drawbacks of the current available software is the difficulty of modifying or extending their functionalities according to the user needs. It is highly desirable to dispose of a software package which includes a set of common building blocks that can be used, extended or customized according to the needs of specific applied problems. By using such kind of framework, the developers do not need to start from scratch every time they write an application, since the design and code of the main body are easily reused. This flexibility is achieved by the pattern design techniques (Gamma et al., 1995) used in the development of the software.

A mesh is a discretization of a geometric domain into small simple shapes, such as triangle or quadrilateral in 2D, and tetrahedra or hexahedra in 3D, where pairs of neighbors elements can only intersect in a common edge, vertex or face. We only consider simple domains represented by general polygons (PSLG) and 3D surfaces. The topology of the mesh is represented by sets of vertices, edges, faces and primitive volumes. The domain can be stored in different file formats such as .off, .poly, .STL.

The quality of the mesh, in terms of size, shape and placement of the elements, is critical for the success of any finite element analysis. Several quality measures have been considered, being the minimum angle criterion the most extended.

The meshes can be stored in different data structures which are designed to allow both efficient traversal of the mesh and efficient implementation of the meshing algorithms. Examples of such data structures are the Doubly-Connected-Edge-List (DCEL) (de Berg et al., 1997), the Winged-Edge and the Quad-Edge data structures explained in Goodrich and Ramaiyer (2000), and the Triangle Data Structure (TDS) (Shewchuck, 1996).

There are various algorithms to obtain a triangular mesh of the domain, being the Delaunay triangulation the most used (Lawson, 1977; Watson, 1981). There also exist several algorithms to create a refined mesh, see for instance Bern and Plassmann (1999) for a survey on mesh generation. Several Delaunay algorithms for unstructured refinement have been proposed and studied in last 15 years (Chew, 1993; Ruppert, 1995; Rivara, 1996; Shewchuck, 1997; Rivara, 1997; Üngör, 2009; Coll et al., 2008b, 2009; Rivara and Calderon, 2010).

Robustness, accuracy, precision, response time and space management are important issues to deal with, while mesh generation and modification tasks are performed. Easy interaction with the mesh is also desirable. Users may ask information about elements of the mesh, or insert new restrictions interactively to the domain model, or choose some region of the model or the mesh to execute different operations. For all these tasks, a friendly interface is always appreciated.

In this paper we discuss the design and implementation of a meshing framework designed to include most of the well known algorithms for dealing with planar and surface triangulations: different versions of Delaunay refinement algorithms in 2-dimensions for triangulating PSLG geometries (Ruppert, 1995; Shewchuck, 1997; Rivara et al., 2001; Coll et al., 2008b, 2009), different versions of Delaunay refinement algorithms for surface triangulations (Coll et al., 2008a),

and different versions of triangle bisection refinement (Rivara and Calderon, 2010). The framework was also designed to test new algorithms and to compare them with previous ones. To this end, the package include a set of well known basic meshing tools, criteria and techniques that can be combined to produce a complete meshing software. The basic tools are in turn grouped in three hierarchies of meshing operations/criteria: *Basic-Point-Selection* (which includes different strategies to select a point to be inserted in the mesh), *Basic-Point-Insertion* (which includes different strategies to insert a point in the mesh), and *Stopping-Criterion* (includes different criteria to stop the computation either according to the mesh size or a mesh quality criterion). The complete meshing algorithms are in turn grouped in a high level hierarchy called *Mesh-Generation-Modification-Algorithm*. The framework considers the inclusion of different data structures to store the mesh. At present this includes Doubly-Connected-Edge-List (DCEL) (de Berg et al., 1997) and Triangle Data Structure (TDS) (Shewchuck, 1996). In what follows, we discuss the framework design and the algorithms included. Some examples illustrating the use of the software are also included.

2 TRIANGULAR MESHING ALGORITHMS

To produce surface triangulations (in 2D and 3D) we consider the two classical incremental Delaunay algorithms where vertices are inserted one at a time. The Lawson (1977) algorithm is based upon edge flips. When a new point is inserted as a vertex in the mesh, a recursive procedure tests whether the point lies within the circumcircles of the neighboring triangles. Each affirmative answer produces an edge flip: the non locally Delaunay edge is changed by its opposite one. On the contrary, by using the Watson (1981) algorithm when a new vertex is inserted into a triangulation, every triangle whose circumcircle contains the new vertex is deleted producing a polygonal cavity in the mesh, which is retriangulated connecting the new vertex to the vertices of the polygonal cavity.

In order to refine triangulations producing good quality triangles we consider the following algorithms: Circumcenter algorithm (Ruppert, 1995) which inserts the circumcenter of bad quality triangles until all the triangles have the desired quality. Off-center algorithm (Üngör, 2009) defines a circumcenter related point to be inserted into the mesh. Coll et al. (2008b) propose an algorithm that combines improvement and refinement of Delaunay triangulations which includes inserting and removing mesh elements. Lepp-Delaunay algorithms (Rivara et al., 2001; Rivara and Calderon, 2010) which, for every bad quality triangle, use the Longest-Edge Propagation Path (Lepp) to define a terminal edge. Two alternative point selection criteria are used: the midpoint of the terminal edge or the centroid of the triangles that share the terminal edge. The selected point is Delaunay inserted in the mesh. Lepp-bisection algorithms (Rivara, 1996, 1997) which perform the longest edge bisection of the triangles that share the terminal edge and use a point density function (see Frey and George (2000) for a definition) to specify the element size that must be conformed to by the mesh elements anywhere in the space. Lepp-surface algorithm (Coll et al., 2008a) which produces a small triangular approximation of a huge terrain grid data by using a two-goal strategy that assure both small approximation error and well shaped 3D triangles

3 PACKAGE DESIGN

In general, the main architecture of meshing processes consider the following components: a geometry model to be discretized; a mesh and its data structure; and an algorithm to generate or modify the mesh, which in turn requires of a point selection strategy to determine

the point to be inserted, a basic point insertion algorithm and a stopping criterion to get the output mesh. Figure 1 shows the main domain classes of our framework for mesh generation and modification. The basic classes required by the application are Mesh, Data-Structure, Geometry-Model, Basic-Point-Insertion-Algorithm, Basic-Point-Selection-Strategy, Stopping-Criterion, and Mesh-Generation-Modification-Algorithm.

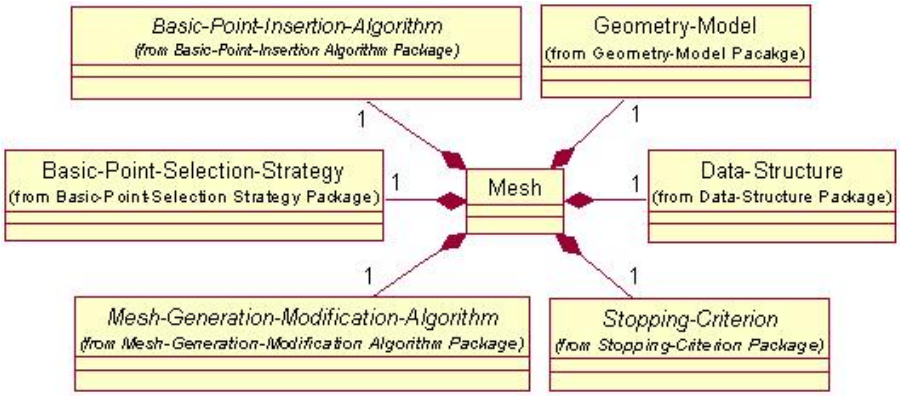


Figure 1: Main domain classes of the framework.

We also consider the following requirements: representation of the input geometry model and representation of the intermediate and the output meshes which can be stored in different formats; implementation of an interface that allows the user to interact and visualize both the geometry model and the output meshes.

All these requirements are taken into account in three fundamental packages: User-Interface, IO-Format, and Mesh-Generation-Modification (see Figure 2).

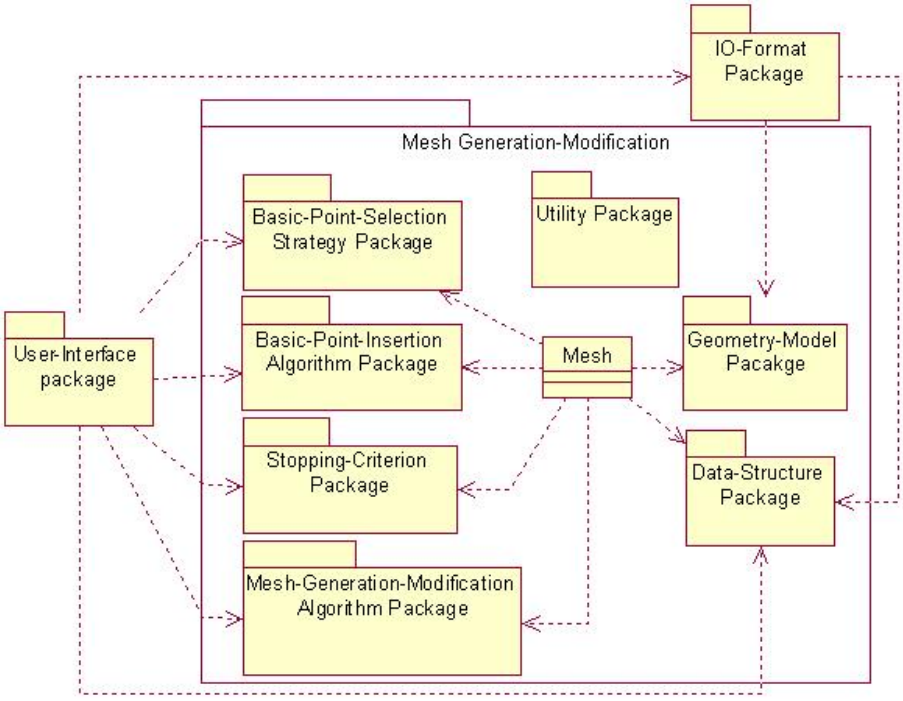


Figure 2: Basic package structure of the framework.

3.1 User-Interface package

We have separated the design of the interface from the rest of the software which makes the structure of the program independent of the interface and the platform. In this way, the interface can be implemented using different libraries depending on the wishes of the user. The functionality of this package allows to accomplish with the visualization requirements of both the geometry model and the output meshes, either in 2D and 3D depending on the case. This package also allows the user to define processing parameters, as well as to obtain statistics about the output mesh. The user can interact with the mesh asking for information about the geometric elements, and to visualize changes or details on the geometry model and the meshes.

3.2 IO-Format package

All the input and output operations over the geometry model and the meshes are included in this package which provides methods to work with different input and output formats.

3.3 Mesh-Generation-Modification package

The Mesh-Generation-Modification package contains the main components of the application designed to deal with triangulating tasks. This is the heart of the framework and include the following components.

Geometry-Model This represents the input geometry model to be discretized. At present this includes simple 2D geometry models and 3D surface models.

Data-Structure The Data-Structure class represents the subjacent data structure of the triangulation. The data structure is composed of the topologic elements vertices, edges and faces. We include the DCEL and TDS data representations.

Basic-Point-Selection Strategy There are different strategies to select a point to be inserted in the mesh. We include the circumcenter of a triangle, the offcenter, the midpoint of a longest edge, the midpoint of a Lepp-terminal longest edge, the centroid of a pair of associated terminal triangles.

Basic-Point-Insertion Algorithm We include a hierarchy of Delaunay based algorithms and triangle bisection algorithms to insert the selected point into the mesh.

Stopping-Criterion We consider different stopping criteria: minimum angle, maximum angle, aspect-ratio, area-ratio, number of elements.

Mesh-Generation-Modification Algorithm There are several algorithms to generate/modify a mesh. We include some of the algorithms based on Delaunay triangulation and longest edge bisection of triangles, implemented as subclasses of the Mesh-Generation-Modification-Algorithm class.

Mesh This class interacts with the geometry model to be discretized, the data structure of the mesh, the algorithm to subdivide the geometry model and the mesh, the algorithm used to generate or modify the mesh, and the stopping criterion.

Utility This package encloses mathematical utilities such as Vector, Matrix, and Geometric-Primitives. Among the Geometric-Primitives we have implemented the computation of

the circumcircle and the circumsphere (Guibas and Stolfi, 1985; Devillers, 1999) functions among others.

4 IMPLEMENTATION DISCUSSION

A prototype of this framework has been implemented in C++ using the Qt libraries for a graphical interface. The framework includes several basic algorithms, data structures, geometric basic classes and several complete mesh generation/modification algorithms. In what follows we describe our object oriented implementation based on design patterns. In the context of the object oriented approach the creation of a concrete object of a class is called instantiation.

4.1 Interface implementation

Iteratively the user is able to select the data structure, a stopping criterion and several processing parameters such as selection of basic point insertion algorithm and selection of a mesh generation or modification algorithm. We have used *Factory methods* for the instantiation of the data structure, the stopping criterion, the basic point insertion operation, and mesh generation or mesh modification algorithms. It is possible to use an algorithm to mesh the initial geometry model and another algorithm to generate/modify the current mesh.

4.2 IO-Format implementation

To facilitate the inclusion of different data formats, the Convertor-Factory class was implemented using a *Factory method pattern* which contains a double entry table with the description of the format type included in the menu and the name of the equivalent Convertor class to be instantiated. Each subclass of the Convertor class implements an input/output format method. In order to assure that only one instance is consistently used during the execution of the program, the Convertor-Factory class is declared as a *Singleton* which implements the double entry table. See Figure 3 for an illustration of the IO-format package.

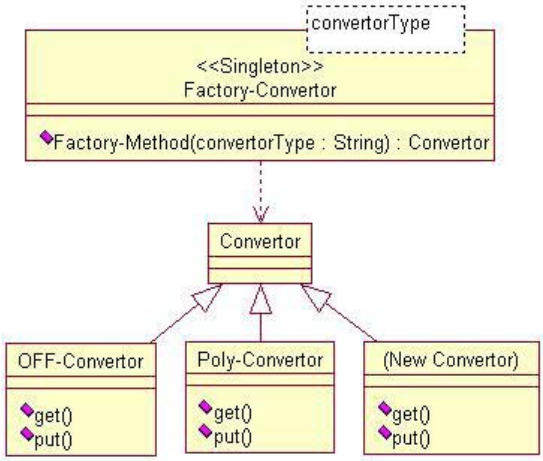


Figure 3: Class diagram of the IO-Format package.

4.3 Geometry-Model implementation

We have implemented a simple Geometry-Model composed of sets of Points, Segments and Polygons. A Segment contains two Points which are its endpoints. A Polygon is an aggregation

of ordered segments. A Polygon can represent a hole in the Geometry-Model or a geometry boundary. A geometric model can be composed of different geometric elements that are not necessary connected.

To represent the recursive structure of the Geometry-Model package we use the *Composite pattern* (see Figure 4). The Composite design pattern treats primitive and composite objects exactly in the same way. The class CompositeGeometryElement defines behavior for GeometryElements having children.

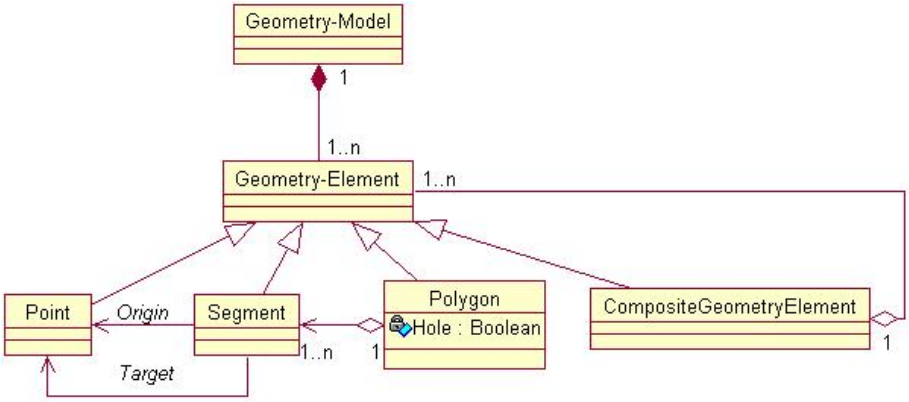


Figure 4: Class diagram of the Geometry-Model package.

4.4 Data-Structure implementation

Different possible data structures are offered to the user. These data structures represent the same topologic elements: vertices, edges, faces, but include different information to define the mesh. For example, the basic element of the *DCEL* data structure is the half-edge which is an oriented edge going from one vertex to the other. Thus for representing a polygonal mesh, the data structure stores the following information for a half-edge: a pointer to the vertex origin of the half-edge, a pointer to the left incident face, a pointer to the next half-edge in the incident face, a pointer to the previous half-edge in the incident face, and a pointer to the twin half-edge. On the contrary, for each face the *TDS* data structure, stores six pointers: three pointers to vertices of a face and three pointers to its adjacent faces. Note that, even when elements of the data structures contain different information, the required functionalities are the same for all of them: determination of the vertices of a face, the neighbors of a face, the edges of a face, the neighbors of an edge, the element shared by a point, etc.

To illustrate the flexibility of our approach consider the implementation of two different incremental Delaunay triangulation algorithms: the edge flipping algorithm of Lawson (1977) and the cavity algorithm of Watson (1981). Basic point insertion implementations are independent from DCEL and TDS data structures. The data structures only creates and deletes elements as needed by the selected algorithm.

Figure 5 illustrates the use of the *Bridge* pattern. A *Data-Structure* is composed of *Topology-Elements* which can be of type *Vertex*, *Edge*, or *Face* depending on the data structure instantiated (*DCEL*, *TDS*, or other). Each *Topology-Element* contains a set of pointers to other elements depending on the data structure type chosen. Thus the use of the *Bridge* pattern decouples the abstract data structure from its implementations.

The data structure is updated when a new element is created. Creation and deletion operations changes depend on the data structure implementation chosen. For example, if a *TDS*

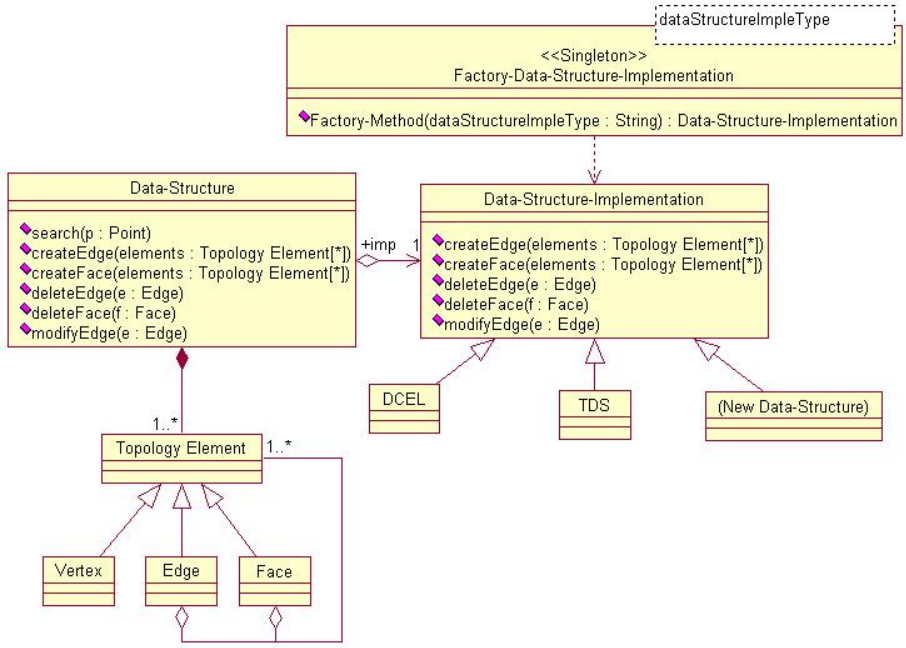


Figure 5: Class diagram of the Data-Structure package.

data structure is chosen, then the method *createEdge()* creates an edge only if the edge is a constrained one.

4.4.1 Basic-Point-Insertion Algorithms implementation

To allow the inclusion of new algorithms in the framework, the *Strategy* and the *Template* patterns are used. Figure 6 illustrates the hierarchy of Delaunay algorithms at present included. The Template pattern is used in the Delaunay-edge-flipping component because the algorithms have a common structure where the *legalize-edge* function (edge flipping as described by Lawson (1977)), varies depending on the specific Delaunay triangulation chosen. Thus some variants of the flipping criteria can produce different triangulations. The Strategy pattern is used in the *point-insertion* operation which depends on the type of algorithm chosen to generate the mesh. Note that a *Factory method* is used in the class Factory-Basic-Point-Insertion for creating a Basic-Point-Insertion-Algorithm instance.

4.4.2 Basic-Point-Selection implementation

The framework includes the following point selection criteria: circumcenter, edge midpoint, centroid of pairs of terminal triangles, Lepp strategy, offcenter. We have used the *Strategy* pattern to implement them. The *getPoint* function returns a point to be inserted in the mesh and depends on the class instantiated.

4.4.3 Stopping-Criterion design

We have used the *Strategy* pattern to make easy the addition of new quality measures to determine the quality of the elements of the mesh. This considers a *Factory method* implemented in Factory-Stopping-Criterion which allows the instantiation of the adequate Stopping-Criterion.

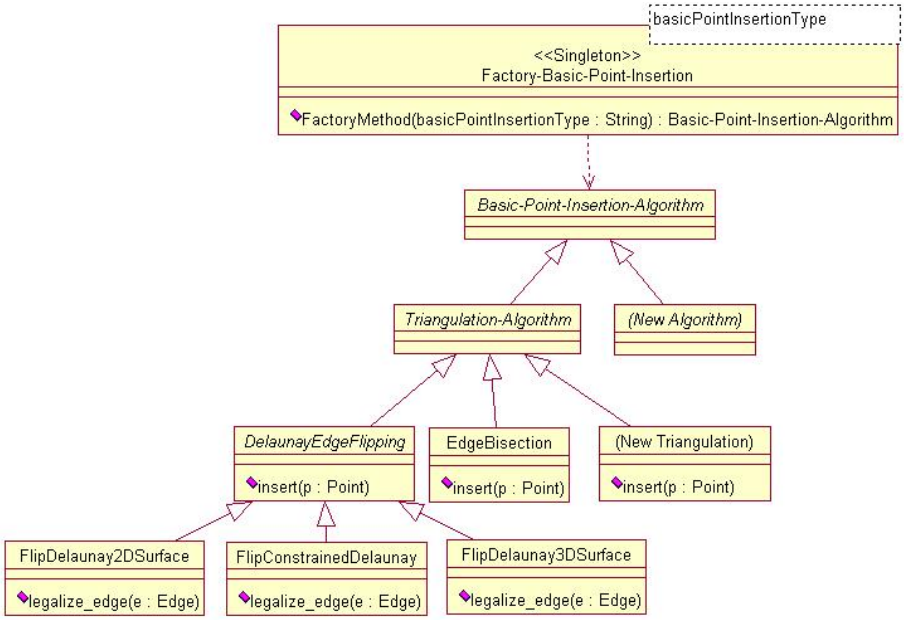


Figure 6: Class diagram of the Basic-Point-Insertion-Algorithm package.

4.5 Mesh-Generation-Modification Algorithms design

The *Strategy* pattern allows the easy addition of new algorithms, where the *modify* operation has different implementations depending on the subclass of Mesh-Generation-Modification-Algorithm class chosen. Figure 7 shows the algorithms implemented as part of the framework.

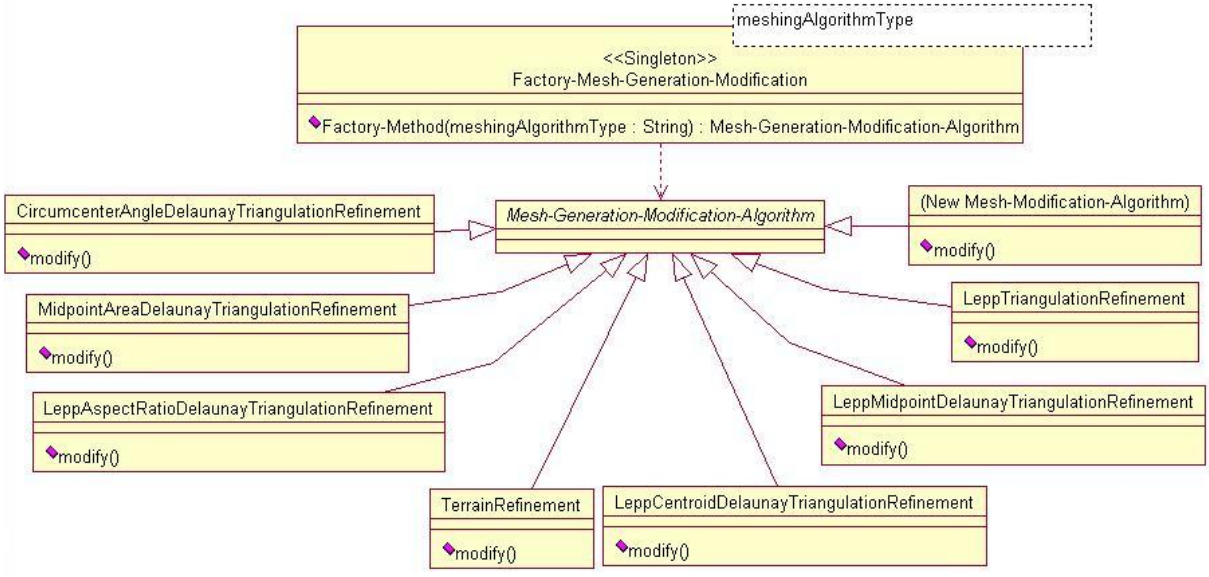


Figure 7: Class diagram of the Mesh-Generation-Modification-Algorithm package.

5 EXAMPLES OF APPLICATIONS OF THE FRAMEWORK

Hierarchies composing the framework can be combined allowing to test the combination of different mesh refinement processes. In this section several examples of mesh refinement algorithms are presented to show the flexibility of the designed framework.

5.1 Implementation of Delaunay refinement applications

At present, the framework includes the Delaunay Refinement processes described in Coll et al. (2008b, 2009); Rivara et al. (2001); Rivara and Calderon (2010), and briefly discussed in section 2.

To this end we have created the new classes Circumcenter-Angle-Delaunay-Triangulation-Refinement, Midpoint-Area-Delaunay-Triangulation-Refinement, Lepp-AspectRatio-Delaunay-Triangulation-Refinement, Lepp-Midpoint-Delaunay-Triangulation-Refinement and Lepp-Centroid-Delaunay-Triangulation-Refinement in the hierarchy of Mesh-Generation-Modification-Algorithm. Each refinement algorithm uses a different combination of Stopping-Criteria, Basic-Point-Selection, and Basic-Point-Insertion-Algorithm.

In turn, Circumcenter-Angle-Delaunay-Triangulation-Refinement combines Minimum2D-Angle stopping criterion and Circumcenter point selection, while Midpoint-Area-Delaunay-Triangulation-Refinement uses Maximum2DArea and MidpointLongetEdge. In the case of Lepp-AspectRatio-Delaunay-Triangulation-Refinement, instances of the Minimum2DAspect-Ratio and Lepp2DSurface (which return the midpoint of the terminal edge) classes are combined. In exchange the Lepp-Midpoint-Delaunay-Triangulation-Refinement combines Minimum2DAngle and Lepp2DSurface, and Lepp-Centroid-Delaunay-Triangulation-Refinement combines Minimu2DAngle and LeppCentroid2DSurface (which return the centroid or the midpoint of the terminal edge). In all cases, selected points are inserted using as basic point insertion algorithm, an instance of a FlipDelaunay2DSurface subclass. The algorithms selected are independent of the data structure chosen.

5.2 Implementation of bisection refinement application

The adaptation of the framework to the Longest-Edge refinement (Rivara, 1997, 1996) requires the instantiation of four classes, a basic point selection strategy, a basic point insertion algorithm, a stopping criteria, and a refinement algorithm. Involved classes are Size-Distribution for stopping criterion, the Edge-Bisection under Triangulation as basic point insertion algorithm, MidpointLongestEdge for point selection and the LeppTriangulationRefinement class in the Mesh-Generation-Modification-Algorithm hierarchy.

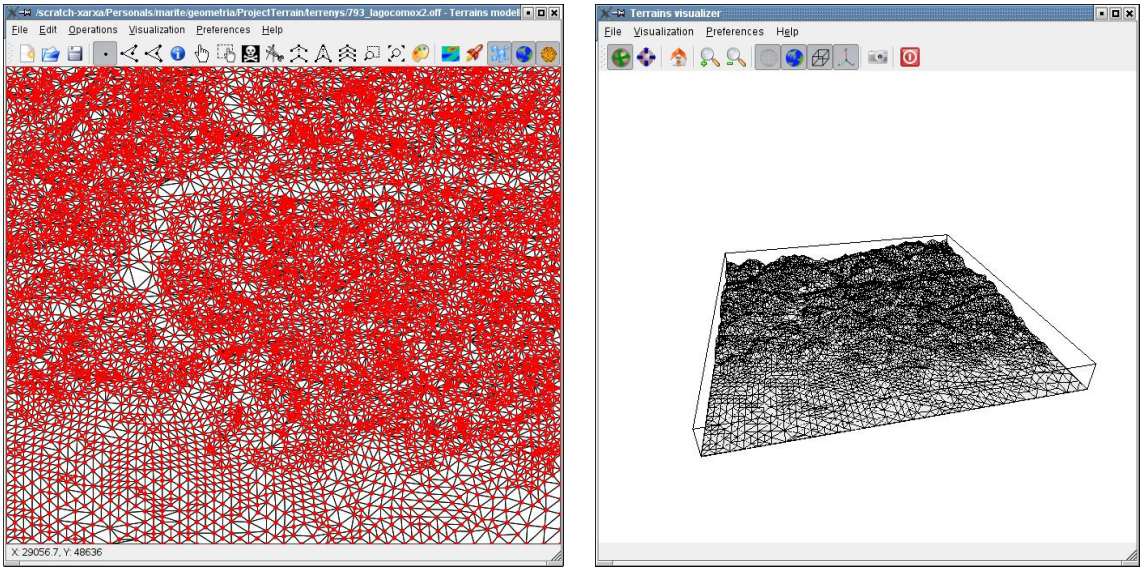
5.3 Combination of stopping criteria application

The framework also integrates novel Delaunay refinement method discussed in Coll et al. (2009) which uses several stopping criteria during the refinement process. The geometry model is preprocessed by applying an aspect-ratio quality measure by instantiating the Aspect-Ratio subclass of the Stopping-Criterion class, while an Angle quality measure is later chosen.

5.4 Terrain interpolation application

In this case it is necessary to add a subclass to the Mesh-Generation-Modification-Algorithm hierarchy, Terrain-Refinement, includes the options of creating the triangulation in 2D and the surface triangulation of a terrain (Coll et al., 2008a). It is not necessary to modify our interface because this includes the possibility of the visualization in 3D. In this case, FlipDelaunay3DSurface and Lepp3DSurface are chosen for point insertion and point selection respectively.

Figures 8(a) and 8(b) illustrates the use of our framework with the terrain interpolation instantiation. The two figures show different visualizations of the same terrain, the Commo Lake. Figure 8(a) shows a 2D projection of the terrain and Figure 8(b) a 3D view.



(a) (b)

Figure 8: Image of the 2D (a) and 3D (b) visualizer from our application.

6 CONCLUDING REMARKS

At present our framework only consider unstructured triangulations. However, it is simple to include quadrilateral meshes since this depends on the algorithms chosen for the basic point insertion and refinement processes. The user can add a new data structure or to use a DCEL data structure, which can deal with any shape faces. The Stopping-Criterion structure can also be extended. At present the framework does not incorporate smoothing techniques, but it is easy to add a new hierarchy to cover this requirement.

Several free available libraries and softwares could be also incorporated easily to the framework according to the user needs. Among the available libraries we can mention the Computational Geometry Algorithms Library CGAL http://www.cgal.org/, and the Library Efficient Data Types and Algorithms LEDA http://www.mpi-inf.mpg.de/LEDA/. CGAL goal is to provide easy access to efficient and reliable geometric algorithms in the form of a C++ library, while LEDA is a C++ library of combinatorial and geometric data types and algorithms. Other softwares can also be included in the framework but this requires more work. For example, some of our algorithms use Qhull http://www.qhull.org/ which does not have a friendly interface. Qhull allows to compute the convex hull, Delaunay triangulation, Voronoi diagram, half-space intersection about a point, furthest-site Delaunay triangulation, and furthest-site Voronoi diagram.

REFERENCES

Bern M. and Plassmann P. *HandBook of Computational Geometry*, chapter Mesh Generation. Elsevier Science, 1999.

Chew L.P. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the 9th Annual Symposium on Computational Geomatry*, pages 274–280. 1993.

Coll N., Guerrieri M., Rivara M.C., and Sellarès J.A. Accurate interpolation of terrain surfaces from over-sampled grid data. In *Proceedings of the 17th International Meshing Roundtable*, pages 351–368. 2008a.

Coll N., Guerrieri M., and Sellarès J.A. Combining improvement and refinement techniques: 2D Delaunay mesh adaptation under domain changes. *Applied Mathematics and Computation*, 201:527–546, 2008b.

Coll N., Guerrieri M., and Sellarès J.A. 2D Delaunay mesh generation with area/aspect-ratio constraints. In *Proceedings of the 25th European Workshop on Computational Geometry*, pages 239–242. 2009.

de Berg M., van Kreveld M., van Oostrum R., and Overmars M. *Computational Geometry: algorithms and applications*. Springer, 1997.

Devillers O. On deletion in Delaunay triangulations. In *Proceedings of the 15th Annual ACM Symposium on Computational Geometry*, pages 181–188. 1999.

Frey P.J. and George P.L. *Mesh Generation: Application to Finite Elements*. Hermes Science, 2000.

Gamma E., Helm R., Johnson R., and Vlissides J. *Dessign Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

Goodrich M.T. and Ramaiyer K. *Handbook of Computational Geometry*, chapter Geometric data Structures, Chapter 4, pages 463–489. Elsevier Science, 2000.

Guibas L. and Stolfi J. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):75–123, 1985.

Lawson C.L. Software for $\mathcal{C}^1$ surface interpolation. *Mathematical Software III (John R.Rice, editor)*, pages 161–194, 1977.

Rivara M.C. New mathematical tools and techniques for the refinement and / or improvement of unstructured triangulations. In *Proceedings of the 5th International Meshing Roundtable, Pittsburgh*, pages 77–86. 1996.

Rivara M.C. New longest-edge algorithms for the refinement and/or improvement of unstructured triangulation. *International Journal for Numerical Methods in Engineering*, 40:3313–3324, 1997.

Rivara M.C. and Calderon C. LEPP terminal centroid method for quality triangulation. *Computer-Aided Design*, 42(1):58–66, 2010.

Rivara M.C., Hitschfeld N., and Simpson R.B. Terminal edges Delaunay (small angle based) algorithm for the quality triangulation problem. *Computer-Aided Design*, 33:263–277, 2001.

Ruppert J. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.

Shewchuck J.R. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Proceedings of the ACM 11th Annual Symposium on Computational Geometry*, pages 61–70. 1996.

Shewchuck J.R. *Delaunay Refinement Mesh Generation*. Phd thesis, School of Computer Science - Carnegie Mellon University, 1997.

Si H. Constrained Delaunay tetrahedral mesh generation and refinement. *Finite Elements in Analysis and Design*, 46(1-2):33–46, 2010.

Üngör A. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. *Computational Geometry*, 42(2):109–118, 2009.

Watson D.F. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *Computer Journal*, 24(2):167–172, 1981.