Three Dimensional Triangulations in CGAL*

Monique Teillaud
INRIA, BP 93, 06902 Sophia Antipolis CEDEX, FRANCE.
Monique.Teillaud@sophia.inria.fr

Abstract

This paper describes the design and the implementation of the three-dimensional triangulation package[1] of the Computational Geometric Algorithms Library CGAL[2]. We focus on representation issues and especially insist on how the cases of degenerate dimensions are treated. The algorithmic issues are not examined in this short paper.

1 Introduction

A three-dimensional triangulation is a three-dimensional simplicial complex, pure connected and without singularities [BY98]. It is a set of cells (tetrahedra) such that two cells either do not intersect or share a common facet, edge or vertex.

Generalizing the storage of 2D triangulations [tri99] to the 3D case, we choose to explicitly represent only cells and vertices, together with adjacency and incidence relations: a cell has pointers to its four vertices and to its four neighbors, a vertex has a pointer to one of the cells having this vertex.

Design Overview We follow the design in three layers proposed for polyhedral surfaces by Lutz Kettner [Ket98] and also adapted to two-dimensional triangulations by Mariette Yvinec [tri99]. This design makes a clear distinction between the combinatorial structure of a triangulation and its geometry (see Figure 1).

In the bottom layer, the base classes store elementary geometric information as well as any other information for the given application. The middle layer class stores the triangulation data structure, which is purely combinatorial. It provides operations such as insertion of a new vertex in a given cell, and is responsible for the combinatorial integrity of the triangulation. The upper layer is the geometric triangulation class, providing operations such as location of a point in the triangulation, insertion of a point, and is responsible for the geometric validity. This class is parameterized by two classes:
- the **geometric traits** class, where the user can specify the type of points he wants to use as well as the elementary operations on them (predicates,...)
- the **triangulation data structure** class of the middle level, described in Section 3.

Delaunay triangulations as well as hierarchical Delaunay triangulations [Dev98] are also implemented in the package but not presented here.

2 Triangulation of points in 3D space

The basic triangulation class of CGAL is primarily designed to represent the triangulations of a set $\mathcal{S}$ of points in the space. A 3D triangulation can be seen as a partition of the space into bounded tetrahedra having four points of $\mathcal{S}$ as vertices (partitioning the convex hull $CH(\mathcal{S})$ of $\mathcal{S}$)

*This research was partially supported by the ESPRIT IV LTR Project No. 28155 (GALIA).
[1]This package will be included in Release 1.3
[2]`http://www.cs.uu.nl/CGAL/`

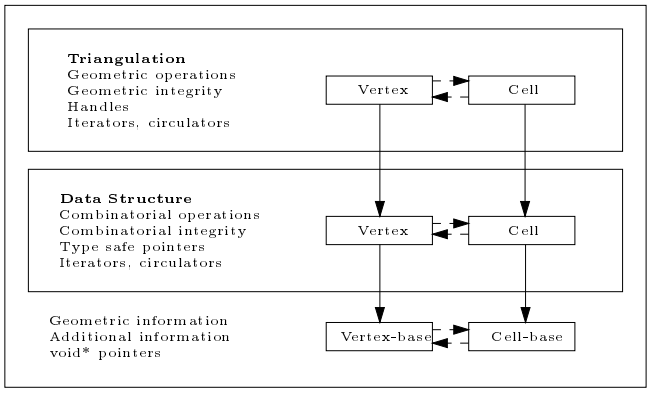

Figure 1: The three layers in the design.

and unbounded tetrahedra having three vertices forming a facet of the boundary of $CH(\mathcal{S})$, and whose fourth vertex is the infinite vertex.

2.1 Storage and degenerate dimensions

As previously said, we store a triangulation as a set of cells, where each cell has pointers to its four vertices and its four neighbors. The vertices and neighbors are numbered $(0, 1, 2, 3)$, in such a way that neighbor i is opposite to vertex i. A vertex stores a pointer to one of its incident cells.

When all points of $\mathcal{S}$ are coplanar, the triangulation is two-dimensional, it consists in fact of triangular faces, and when all points of $\mathcal{S}$ are collinear, it is one-dimensional and consists of edges. We use the same cell class in all cases: triangular faces in 2D can be considered as degenerate cells, having only three vertices (resp. neighbors) numbered $(0, 1, 2)$, and one $NULL$ vertex (resp. neighbor); edges in 1D have only two vertices (resp. neighbors) numbered 0 and 1.

2.2 Validity of a triangulation

The two layers of the triangulation respectively define two different notions of validity, related to the correction of adjacency relations and of the orientation of the cells. These notions are local, and do not always ensure global validity [MNS+96, DLPT] but they will be sufficient for practical cases.

A 3D triangulation is said to be *combinatorially valid* iff the following is true:

(a) Two adjacent cells have neighbor pointers to each other and they have three common vertices.

(b) All pairs of adjacent cells have consistent orientations. If two cells c_1 and c_2 are neighbors and share a facet with vertices u, v, w, then these cells have *consistent orientations* if the vertices of c_1 can be numbered $(v_0^1 = u, v_1^1 = v, v_2^1 = w, v_3^1)$, and the vertices of c_2 are numbered $(v_0^2 = v, v_1^2 = u, v_2^2 = w, v_3^2)$, up to positive permutations of $(0, 1, 2, 3)$. In other words, the fourth vertices v_3^1 and v_3^2 of c_1 and c_2 see the common facet in opposite orientations. See Figure 2.

The set σ_4 of permutations of $(0, 1, 2, 3)$ has cardinality 24, and the set of positive permutations $\mathcal{A}_4$ has cardinality 12. Thus, for a given orientation, there are up to 12 different orderings of the four vertices of a cell. Note that circular permutations are negative and so do not preserve the orientation of a cell.

A 3D triangulation is said to be *geometrically valid* iff:

(a - b) It is combinatorially valid.

(c) Any cell has its vertices ordered according to positive orientation. See Figure 3.

Degenerate dimensions: When the triangulation is degenerated into a planar triangulation, the notion of *validity* reduces to the following:

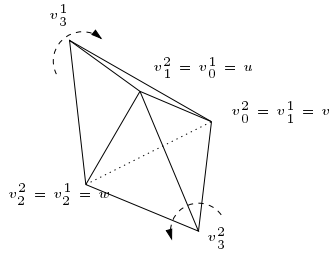

Figure 2: Consistent orientations of two cells.

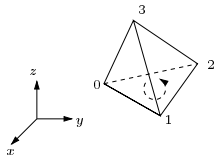

Figure 3: Geometric orientation of a cell.

(a-2D) Two adjacent faces have neighbor pointers to each other and two common vertices.
(b-2D) The orientations induced on an edge by its two incident faces are opposite.
(c-2D) For any two adjacent triangular faces with vertices $\{u, v, w_1\}$ and $\{u, v, w_2\}$ with common edge (u, v), w_1 and w_2 lie on opposite sides of (u, v) in the plane.
The one-dimensional case is similar and simpler.

3 The triangulation data structure

This template parameter of the triangulation class (see Section 1) can be seen as a container for the cells and vertices maintaining incidence and adjacency relations. It represents a triangulation of a topological sphere S^d of R^{d+1}, for $d \in \{0, 1, 2, 3\}$. It is purely combinatorial; in particular, it does not consider the infinite vertex as a special vertex but as a standard vertex.

Let us give, for each dimension, the example corresponding to the triangulation data structure having a minimal number of vertices, i.e. a simplex:

- *dimension 3.* The triangulation data structure consists of the boundary of a 4-dimensional simplex, which has 5 vertices. A geometric embedding consists in choosing one of these vertices to be infinite, thus four of the five 3-cells becomes infinite: the geometric triangulation has one finite tetrahedron remaining, each of its facets being incident to the infinite cell.

- *dimension 2.* We have 4 vertices forming one 3-dimensional simplex, i.e. the boundary of a tetrahedron. The geometric embedding in the plane consists in choosing one of these vertices to be infinite, then the geometric triangulation has one finite triangle whose edges are adjacent to the infinite triangles. See Figure 4.

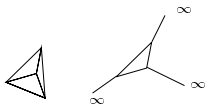

Figure 4: A simplex in 3D and its 2D embedding.

- *dimension 1.* A 2-dimensional simplex (a triangle) has 3 vertices. The geometric embedding is and edge whose vertices are linked to an infinite point.

- *dimension 0.* The boundary of a 1-dimensional simplex (and edge) consists of 2 vertices. One of them becomes infinite in the geometric embedding, and there is only one finite vertex remaining. The two vertices are disconnected.

Let us have a quick look at the way dimensions are maintained during the construction of the triangulation. Suppose that the points of $\mathcal{S}$ are inserted incrementally, and that the first points are collinear. Figure 5 shows both triangulations.

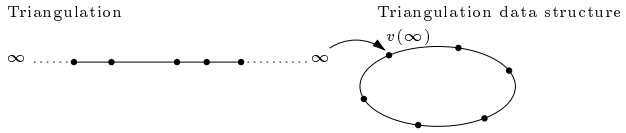

Figure 5: One-dimensional case

Then, if the next point p of $\mathcal{S}$ is not collinear with the previous points, the triangulations are updated as in Figure 6.

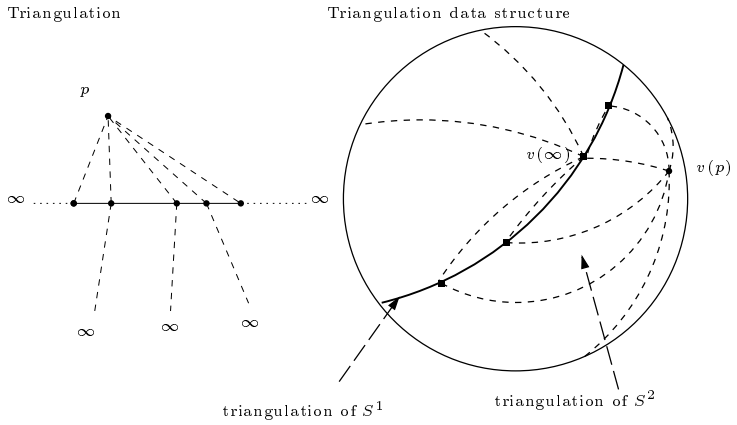

Figure 6: Insertion of a non collinear point.

References

[BY98] J.-D. Boissonnat and M. Yvinec. *Algorithmic geometry.* Cambridge University Press, UK, 1998.

[Dev98] O. Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[DLPT] Olivier Devillers, Giuseppe Liotta, Franco P. Preparata, and Roberto Tamassia. Checking the convexity of polytopes and the planarity of subdivisions. *Comput. Geom. Theory Appl.* To appear.

[Ket98] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 146–154, 1998.

[MNS+96] Kurt Mehlhorn, Stefan Näher, Thomas Schilz, Stefan Schirra, Michael Seel, Raimund Seidel, and Christian Uhrig. Checking geometric programs or verification of geometric structures. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 159–165, 1996.

[tri99] 2D triangulations. In S. Schirra, R. Veltkamp, and M. Yvinec, editors, CGAL *Manual, Basic Library.* 1999. Release 1.2.