CGAL par l'exemple

Computational Geometry Algorithms Library

Raphaëlle Chaine

Journées Informatique et Géométrie - 1er Juin 2006 - LIRIS Lyon

Outline

- Overview
  - Strengths
  - Design
- Structure
  - Kernel
  - Convex Hull Algorithms
  - Polygons and Polyhedra
  - Polygon and Polyhedron Operations
  - Arrangements
  - Triangulations and Delaunay Triangulations
  - Voronoi Diagrams
  - Meshing
  - Search Structures
  - Geometric Optimization
  - Interpolation
  - Kinetic Data Structures

Overview

CGAL:

- C++ library
- basic geometric primitives and operations
- collection:
  - standard data structures
  - geometric algorithms

http://www.cgal.org

GeometryFactory:
http://www.geometryfactory.com/

Licenses:

- **Industrial Development**
- **Industrial Research**
- **Academic Development**
- **Open source**

The CGAL Project

- Started in 1996 as joined project of:
  - ETH Zurich
  - INRIA
  - MPI für Informatik
  - Tel-Aviv U
  - Utrecht U
  - Trier U
  - FU Berlin

Current Partners

INRIA · FU Berlin · Geometry Factory · MPII · RuG · Universiteit Utrecht · Ens · University of Notre Dame · Tel Aviv University · Stanford · ETH · University of Florida · SciSoft

Development Process

- Editorial board
  - receives/reviews/approves submissions
- Developer manual
- Developer meetings
- Version management system
- Daily internal release
- Daily testsuite runs all supported platforms
- Bug tracking system

CGAL in Figures

- 1,200 C++ classes
- 400K lines of code
- 2,200 pages manual
- 40 developer years

- Supported platforms:
  - Linux, Irix, Solaris, Windows
  - g++, SGI CC, SunPro CC, VC++, Intel
- Release cycle: 12 months
- 10,000 downloads per year
- 800 registered users
- 50 active developers

Strengths

- Robustness
- Efficiency
- Flexibility
- Ease of use

Strengths

- Robustness

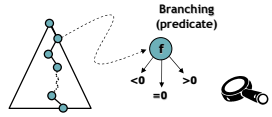
Naïve implementations can:
- loop
- crash
- produce incorrect output

Design

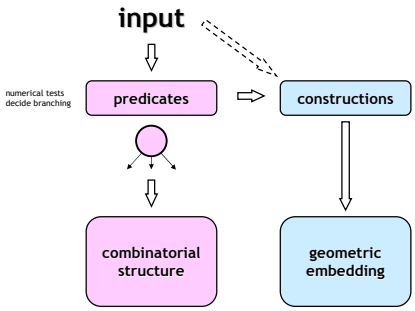
Paradigms:
- Generic programming
  - Templated C++, STL

Design

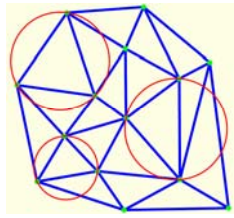
Paradigms:
- Generic programming
  - Templated C++, STL
- Exact computing
  - Make sure that the control flow in the implementation corresponds to the control flow with exact real arithmetic.



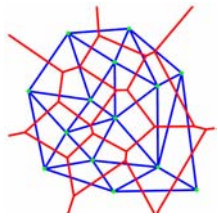
Structure of Geometric Algorithms



Examples



Delaunay triangulation
only predicates required



Voronoi diagram
constructions necessary

Strengths

- Robustness
- Efficiency
- Flexibility
- Ease of use

?

Robust Geometric Computing

- **Computing with:**
  - Floating point numbers: fast
  - Exact numbers: robust
- **Key remark:**
  - exact geometric computing ≠ exact arithmetic
  - filtered predicates & constructions

Generic Programming

```
int  min(int a, int b)
{
   return (a<b) ? a : b;
}
```

Generic Programming

```
float  min(float a, float b)
{
   return (a<b) ? a : b;
}
```

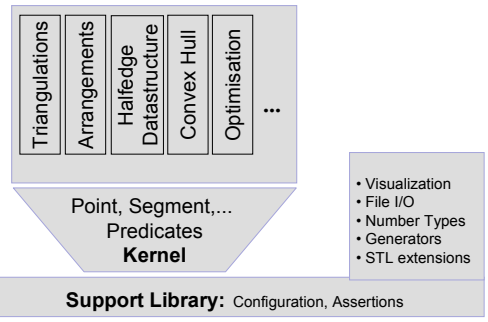
Generic Programming

```
template < CompType >
CompType min(CompType a, CompType b)
{
  return (a<b) ? a : b;
}
```

```
BigInt  n(9), m(8), r;
r = min( n, m );
```

BigInt is a model for the concept CompType

Structure



Kernel

Kernel

- **Basic geometric primitives**
  - 2D, 3D, dD
  - point, line, ray, segment, vector, circle, …
- **Predicates**
  - comparison, orientation, in_circle, intersection test, …
- **Constructions**
  - intersection, distance, affine transform, …

Parameterization of Kernel Classes

Point_3

Cartesian
- double
- Gmpq
- leda::Real
- core::Expr

Homogeneous
- int
- Gmpz

Concepts: NumberType and Kernel

Parameterization of Data Structures

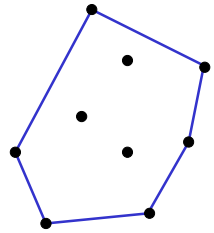
```
template < Geometry >
class Delaunay_triangulation_2
{
    void insert(Geometry::Point t)
    {
       if(Geometry::orientation(p,q,t)==..)
       if(Geometry::incircle(p,q,r,t))
    }
};
```

- works with CGAL kernels
- thin glue layer for other kernels

Packages in CGAL 3.2 (22 may 2006)

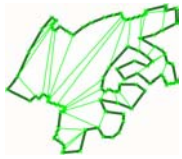
Convex Hull

- 2D, 3D
  - static (quickhull) and dynamic convex hulls (triangulation)
  - Extremal points
  - Strongly convexity checking
- dD
  - incremental convex hull



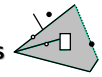
Polygons

- Area, orientation, convexity test, simplicity test, point location
- Decomposition
  - convex partitioning
  - monotone partitioning

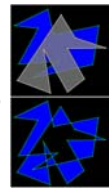


2D Nef Polyhedra

- Nef polyhedron: set obtained from
  - a finite set of open halfspaces
  - intersections and complements



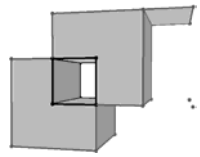
- Features
  - Boolean operations
  - interior, boundary, closure
  - set regularization



Nef Polyhedra embedded on the Sphere

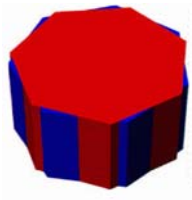
3D Nef Polyhedra

- Closed under Boolean operations
- Non-manifold sets
- Open and closed sets
- Unbounded cells
- Mixed-dimensional cells



Rotated Cylinder Experiment

Operation union (n-gon, rotate(n-gon, alpha))

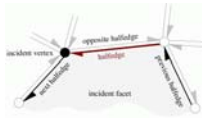


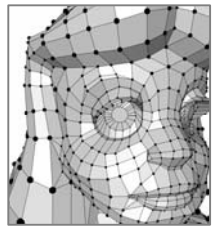
n	α	time ACIS R13	runtime [s] Nef 3D
100	10^{-1}	1.08s	3.47s
	10^{-2}	1.05s	3.50s
	10^{-3}	1.08s	3.59s
	10^{-4}	1.07s	3.64s
	10^{-5}	not executable	3.72s
	10^{-6}	not executable	3.77s
1000	10^{-1}	61s	67s
	10^{-2}	61s	68s
	10^{-3}	61s	69s
	10^{-4}	not executable	69s
	10^{-5}	not executable	71s
	10^{-6}	not executable	71s
2000	10^{-1}	252s	195s
	10^{-2}	253s	198s
	10^{-3}	255s	203s
	10^{-4}	not executable	205s
	10^{-5}	not executable	207s
	10^{-6}	not executable	210s
10000	10^{-1}	not executable	3219 s

Polyhedral Surfaces

Based on halfedge data structure

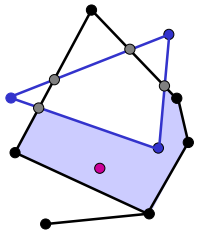
Combinatorial structure:
- **Constructions**
- **Euler operations**





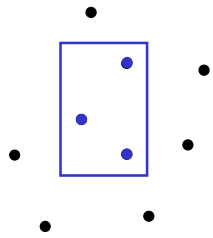
Planar Subdivisions

- Framework for arrangements of 2D curves
  - based on planar maps
  - based on topological maps
  - Models exist for polylines, circles, conic arcs
- Operations
  - point location, overlay, ray shooting,..



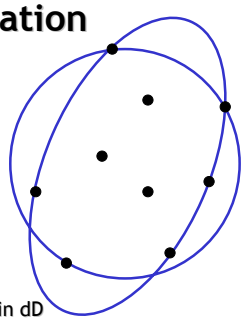
Search Structures

- Multi dimensional data structures
  - range tree, segment tree
  - kD tree
- Operations
  - window queries
  - nearest neighbor



Optimization

- Smallest enclosing circle/ellipse in 2D
- Smallest enclosing sphere in dD
- Rectangular p center, for p = 2, 3, 4
- Polytope distances in dD
- Smallest enclosing annulus in dD
- Smallest enclosing sphere of spheres

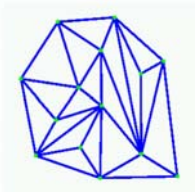


2D Triangulations

- Basic
- Delaunay
- Regular
- Constrained
- Constrained Delaunay

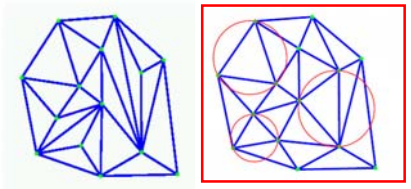
2D Triangulations

- Basic Triangulation
  - Lazy incremental construction
  - No control over shape of triangles



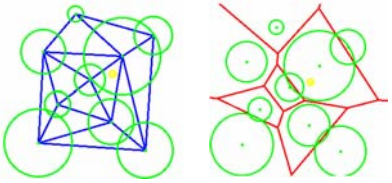
2D Triangulations

- Delaunay triangulation
  - Empty circle property



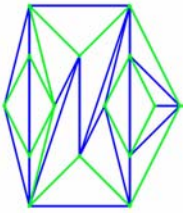
2D Triangulations

- Regular triangulation
  - defined for weighted points
  - dual of power diagram



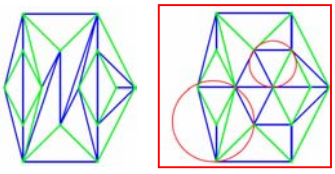
2D Triangulations

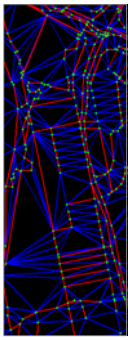
- constrained triangulation
  - enforced edges



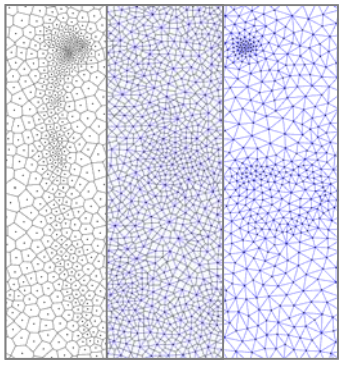
2D Triangulations

- constrained Delaunay triangulation
  - enforced edges + constrained empty circle property

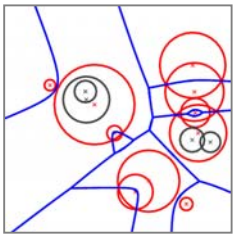




2D Voronoi Diagrams

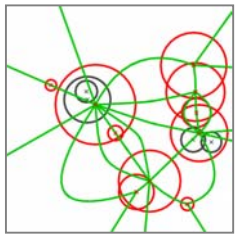


2D Apollonius Graphs



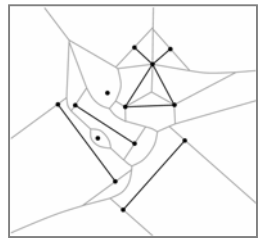
Apollonius diagram

(additively weighted Voronoi diagram)

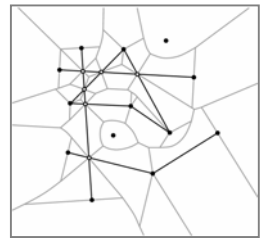


Apollonius graph

2D Segment Voronoi Diagrams



weakly intersecting sites



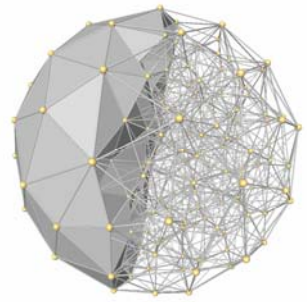
strongly intersecting sites

3D Triangulations

- Basic
- Delaunay 
- Regular

3D Delaunay Triangulation

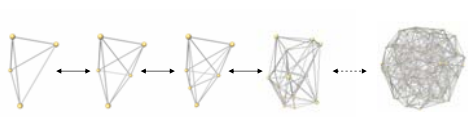
Tetrahedron



3D Delaunay Triangulation

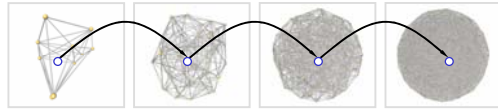
Dynamic:

- point insertion
- vertex removal



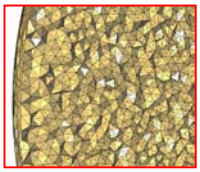
Triangulation Hierarchy

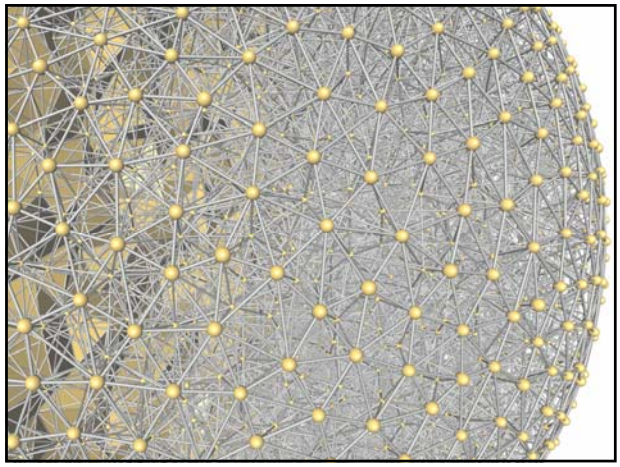
Triangulation augmented with a hierarchical data structure to allow for efficient *point location queries*.



Efficiency

- ~30K vertices/s (Pentium IV, 3.5 GHz)
- filtered kernel
- 300 MBytes / 1M vertices



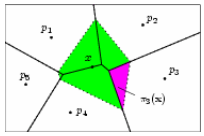


Triangulation Related

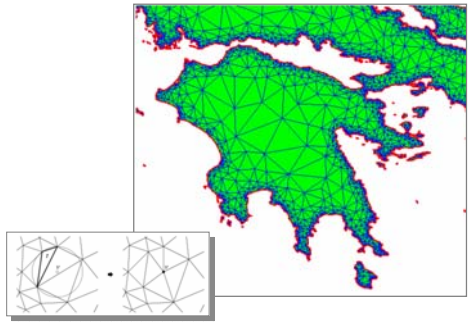
- 2D, 3D Alpha shape

Interpolation

- Natural neighbor interpolation (based on Voronoi diagrams, for scattered data sets)



2D Triangle Meshing by Delaunay Refinement



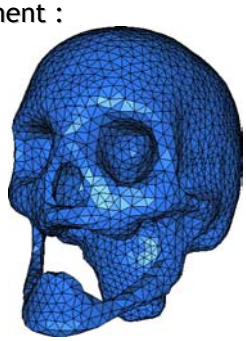
3D Surface Mesher

by Delaunay refinement :

Input :

implicit surface

voxel description



Also in CGAL 3.2

- Kinetic Data Structure
- Planar Parameterization
- Subdivision surfaces
- Principal Component Analysis

Thank you!

- For your attention
- To Pierre Alliez and the many others who made a first version of these slides