Computational Geometry Algorithms Library

www.cgal.org

Monique Teillaud
Overview

- The CGAL Open Source Project
- Structure of CGAL
- The Kernel
The Open Source Project
Goals

- Promote the research in Computational Geometry (CG)

- “make the large body of geometric algorithms developed in the field of CG available for industrial applications”

  ⇒ robust programs

CG Impact Task Force Report, 1996

Among the key recommendations:

- Production and distribution of usable (and useful) geometric codes

- Reward structure for implementations in academia
Development started in 1995

Consortium of 8 European sites
Two ESPRIT LTR European Projects (1996-1999)

Utrecht University (XYZ Geobench)
INRIA Sophia Antipolis (C++GAL)
ETH Zürich (Plageo)
MPI Saarbrücken (LEDA)
Tel Aviv University
Freie Universität Berlin
RISC Linz
Martin-Luther-Universität Halle
• Work continued after the end of European support (1999) in several sites.

• January, 2003: creation of Geometry Factory

INRIA startup
sells commercial licenses, support, customized developments

• November, 2003:

  Release 3.0
  Open Source Project

• December, 2004: Release 3.1
• *kernel* under **LGPL**

• *basic library* under **QPL**
  ○ free use for Open Source code
  ○ commercial license needed otherwise

• A guarantee for CGAL users

• Allows CGAL to become a standard

• Opens CGAL for new *contributions*
- 400,000 lines of C++ code
- >2000 pages manual
- release cycle of ~12 months
- CGAL 2.4: 9300 downloads (18 months)
- CGAL 3.1: 7329 downloads (9 months)
- 4000 subscribers to the announcement list
- 800 users registered on discussion list
- 50 developers registered on developer list
Supported platforms

• Linux, Windows, Mac OS X, Irix, Solaris

• g++, VC++, Intel C++, MipsPRO CC, SunPro CC
Development process

Editorial Board created in 2001.

- responsible for the quality of CGAL

  New packages are reviewed.

  → helps authors to get credit for their work.

  CG Impact Task Force Report, 1996
  Reward structure for implementations in academia

- decides about technical matters

- coordinates communication and promotion

- ...
Andreas Fabri (Geometry Factory)
Efi Fogel (Tel Aviv University)
Bernd Gärtner (ETH Zürich)
Michael Hoffmann (ETH Zürich)
Menelaos Karavelas (University of Notre Dame, USA → Greece)
Lutz Kettner (Max-Planck-Institut für Informatik)
Sylvain Pion (INRIA Sophia Antipolis)
Monique Teillaud (INRIA Sophia Antipolis)
Remco Veltkamp (Utrecht University)
Ron Wein (Tel Aviv University)
Mariette Yvinec (INRIA Sophia Antipolis)
Tools

• Own manual tools: \LaTeX \rightarrow ps, pdf, html
• CVS server for version management
• Developer manual
• mailing list for developers
• 1-2 developers meetings per year, 1 week long
• 1 internal release per day
• Automatic test suites running on all supported compilers/platforms
Contributors keep their identity

- up to 3.0.1: names of authors mentioned in the Preface.
- 3.1: **Names of authors** appear at the beginning of each chapter. Section on history of the package at the end of each chapter, with names of all contributors.
- CGAL developers listed on the “People” web page.
- Authors publish **papers** (conferences, journals) on their packages.
- **Copyright** kept by the institution of the authors.
Projects using CGAL

Leonidas J. Guibas’ and co-workers, Stanford University.
Tamal K. Dey’s and co-workers, The Ohio State University.
Nina Amenta and co-workers, The University of Texas at Austin.
Xiangmin Jiao, University of Illinois at Urbana-Champaign. (Surface Mesh Overlay)
Peter Coveney and co-workers, University of London.

Teaching

• Leo Guibas, Siu Wing Cheng, . . .
Commercial customers of Geometry Factory
Structure of Coal
Basic Library

Algorithms and Data Structures

Kernel

Geometric objects
Geometric operations

core library

configurations, assertions, …

Support Library

Visualization
File
I/O
NumberTypes
Generators
…
Contents of the COAL Basic Library
Convex Hull

[MPI]

- 5 different algorithms in 2D
- 3 different algorithms in 3D
[INRIA]

- 2D/3D Triangle/Tetrahedron based data-structure
- Fully dynamic 2D/3D Delaunay triangulation Delaunay hierarchy [Devillers ’98 ’02]
- 2D/3D Regular Triangulations (fully dynamic in 3.2?)
- 2D Constrained Delaunay Triangulation
- 2D Apollonius diagram
- 2D Segment Voronoi Diagram
- 2D Meshes
Polyhedra

- Half-edge data-structure
- Polyhedral surface (orientable 2-manifold with boundary)
- 2D Nef polygons
- 3D Nef polyhedra
Geometric Optimization

- Smallest enclosing circle and ellipse in 2D
- Smallest enclosing sphere in $d$D
- Largest empty rectangle
- ...
[Tel-Aviv]

- Line segments or polylines
- Conic arcs with Leda or Core

Completely new version in CGAL 3.2
Search Structures

Arbitrary dimension

- Range-tree, Segment-tree, kD-tree
- Window query
- Approximate nearest neighbors
- . . .
Kinetic Data Structures
[Russel Karavelas]

Surface reconstruction
[Oudot Rey]

3D Meshes
[Rineau Yvinec]

Parameterization
[Alliez]

Curved Kernel
Extension of the CGAL kernel
Algebraic issues
[Emiris Kakargias Pion Tsiganidas Teillaud SoCG’04]

...
The COAL Kernel
In the kernel

Elementary geometric objects

Elementary computations on them

**Primitives**
- 2D, 3D, dD
- Point
- Vector
- Triangle
- Iso_rectangle
- Circle

**Predicates**
- comparison
- Orientation
- InSphere

**Constructions**
- intersection
- squared distance
Affine geometry

Point - Origin → Vector
Point - Point → Vector
Point + Vector → Point

Point + Point illegal

midpoint(a,b) = a + 1/2 x (b-a)
**Cartesian representation**

**Point**

\[
\begin{align*}
x &= \frac{hx}{hw} \\
y &= \frac{hy}{hw} 
\end{align*}
\]

**Homogeneous representation**

**Point**

\[
\begin{align*}
hx \\
y \\
hy \\
hw
\end{align*}
\]

**Intersection of two lines**

\[
\begin{align*}
a_1x + b_1y + c_1 &= 0 \\
a_2x + b_2y + c_2 &= 0
\end{align*}
\]

\[
(x, y) = \begin{vmatrix}
b_1 & c_1 \\
b_2 & c_2 \\
a_1 & b_1 \\
a_2 & b_2
\end{vmatrix}, \quad \begin{vmatrix}
a_1 & c_1 \\
a_2 & c_2 \\
a_1 & b_1 \\
a_2 & b_2
\end{vmatrix}
\]

**Field operations**

**Ring operations**
CGAL::**Cartesian**< FT >  
(CGAL::**Simple_Cartesian**)  
CGAL::**Homogeneous**< RT >  
(CGAL::**Simple_Homogeneous**)  

Cartesian Kernels: Field type  
- double  
- Quotient<Gmpz>  
- leda_real  

Homogeneous Kernels: Ring type  
- int  
- Gmpz  
- double  

→ Flexibility  

```cpp
typedef double NumberType;
typedef Cartesian< NumberType > Kernel;
typedef Kernel::Point Point;
```