Computational Geometry Algorithm Library

Efi Fogel
Tel Aviv University

Computational Geometry
Apr. 7th, 2014

Cgal: Mission

"Make the large body of geometric algorithms developed in the field of computational geometry available for industrial applications"

Cgal Project Proposal, 1996

Some of Cgal Content

Bounding Volumes Polyhedral Surfaces Boolean Operations
Triangulations Voronoi Diagrams Mesh Generation
Subdivision Simplification Parametrization Edge Detection Neighbor Search Kinetic Data Structure
Envelopes Arrangements Intersecting Detection Minkowski Sums PCA Polytope Distance QP Solver

Cgal Facts

- Written in C++
- Follows the generic programming paradigm
- Development started in 1995
- Active European sites:

  1. INRIA Sophia Antipolis
  2. INRIA Nancy
  3. CNRS - LIRIS
  4. INRIA Nancy - Lls de France
  5. CNRS - LIRIS
  6. GeometryFactory
  7. MP2I Saarbrücken
  8. Freie Universität Berlin
  9. University of Technology Braunschweig
  10. Tel Aviv University
  11. ETH Zurich
  12. University of Crete and F.O.R.T.H.
  13. Universität della Svizzera Italiana
  14. Universidade Federal de Pernambuco
  15. Instituto Nacional de Matemática Pura e Aplicada

Cgal History

<table>
<thead>
<tr>
<th>Year</th>
<th>Version Released</th>
<th>Other Milestones</th>
</tr>
</thead>
<tbody>
<tr>
<td>1996</td>
<td>Cgal founded</td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td>July 1.1</td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td></td>
<td>Work continued after end of European support</td>
</tr>
<tr>
<td>2001</td>
<td>Aug 2.3</td>
<td>Editorial Board established</td>
</tr>
<tr>
<td>2002</td>
<td>May 2.4</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>Nov 3.0</td>
<td>Geometry Factory founded</td>
</tr>
<tr>
<td>2004</td>
<td>Dec 3.1</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>May 3.2</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Jun 3.3</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Jan 3.4, Oct 3.5</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Mar 3.6, Oct 3.7</td>
<td>Cgal participated in Google Summer of Code 201b</td>
</tr>
<tr>
<td>2011</td>
<td>Apr 3.8, Aug 3.9</td>
<td>Cgal participated in GSoC 2011</td>
</tr>
<tr>
<td>2012</td>
<td>Mar 4.0, Oct 4.1</td>
<td>Cgal participated in GSoC 2012</td>
</tr>
<tr>
<td>2013</td>
<td>Mar 4.2, Oct 4.3</td>
<td>Cgal participated in GSoC 2013</td>
</tr>
<tr>
<td>2014</td>
<td>Mar 4.4</td>
<td>Cgal participates in GSoC 2014</td>
</tr>
</tbody>
</table>
**Cgal in Numbers**

- 1,200,000 lines of C++ code
- 10,000 downloads per year not including Linux distributions
- 4,500 manual pages
- 3,000 subscribers to cgal-announce list
- 1,000 subscribers to cgal-discuss list
- 60 commercial users
- 30 active developers
- 7 Google’s page rank for cgal.org.com
- 2 licenses: Open Source and commercial

**Cgal Properties**

- **Reliability**
  - Explicitly handles degeneracies
  - Follows the Exact Geometric Computation (EGC) paradigm
- **Flexibility**
  - Is an open library
  - Depends on other libraries (e.g., Boost, Gmp, Mpfr, Qt, & Core)
  - Has a modular structure, e.g., geometry and topology are separated
  - Is extensible, e.g., data structures can be extended
- **Ease of Use**
  - Has didactic and exhaustive Manuals
  - Follows standard concepts (e.g., C++ and STL)
  - Characterizes with a smooth learning curve
- **Efficiency**
  - Adheres to the generic-programming paradigm
  - Polymorphism is resolved at compile time

**Cgal Structure**

**Basic Library**
- Algorithms and Data Structures
  - e.g., Triangulations, Surfaces, and Arrangements

**Kernel**
- Elementary geometric objects
- Elementary geometric computations on them

**Support Library**
- Configurations, Assertions, ...

**Cgal Kernel Concept**

- Geometric objects of constant size.
- Geometric operations on object of constant size.

<table>
<thead>
<tr>
<th>Primitives 2D, 3D, dD</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>point</td>
<td>comparison</td>
</tr>
<tr>
<td>vector</td>
<td>intersection</td>
</tr>
<tr>
<td>triangle</td>
<td>containment</td>
</tr>
<tr>
<td>iso rectangle</td>
<td>squared distance</td>
</tr>
<tr>
<td>circle</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Cgal Kernel Affine Geometry**

- point - origin → vector
- point - point → vector
- point + vector → point
- point + point ← Illegal
- midpoint(a, b) = a + 1/2 × (b - a)

**Cgal Kernel Classification**

- **Dimension**: 2, 3, arbitrary
- **Number types**:
  - Ring: +, −, ×
  - Euclidean ring (adds integer division and gcd) (e.g., CGAL: : Gmpz).
  - Field: +, −, ×, / (e.g., CGAL: : Gmpq).
  - Exact sign evaluation for expressions with roots (Field_with_sqr).
- **Coordinate representation**
  - Cartesian — requires a field number type or Euclidean ring if no constructions are performed.
  - Homogeneous — requires Euclidean ring.
- **Reference counting**
- **Exact, Filtered**
It is however more efficient and sufficient for most
\((\text{orientation}(q, p, r) == \text{CGAL::LEFT_TURN}) ? 0 : 1;\)
\text{CGAL::Simple_cartesian<} \text{Kernel};\)
\text{CGAL::Cartesian<}\text{Gmpq}> \text{Kernel};\)
Its number type supports the exact square-root operation.
\text{double CGAL::Exact_predictions_inexact_constructions_kernel \text{Kernel};}\)
\text{Cartesian coordinates.}\)
otherwise.\)
\text{No reference-counting, inexact instantiation}\)
\text{Kernell::Point_2, Kernell::Segment_3}\)
\text{Kernell::Circle_2 C(p,4);}\)
\text{assert (C. has_on_boundary(q));}\)
\text{OK if NT supports exact sqrt.}\)
\text{Assertion violation otherwise.}\)
\text{CGAL Special Kernels}\)
\text{OK if NT supports exact sqrt.}\)
\text{Refer to CGAL's manual for more details.}\)
\text{CGAL Pre-defined Cartesian Kernels}\)
\text{Support construction of points from } \text{double } \text{Cartesian coordinates.}\)
\text{Support exact geometric predicates.}\)
\text{Handle geometric constructions differently:}\)
\text{CGAL::Exact_predicates_inexact_constructions_kernel}\)
\text{Geometric constructions may be inexact due to round-off errors.}\)
\text{It is however more efficient and sufficient for most CGAL algorithms.}\)
\text{CGAL::Exact_predicates_exact_constructions_kernel}\)
\text{CGAL::Exact_predicates_exact_constructions_kernel_with_sqrt}\)
\text{Its number type supports the exact square-root operation.}\)
\text{Computing the Orientation}\)
\text{imperative style}\)
\text{CGAL Special Kernels}\)
\text{Filtered kernels}\)
\text{2D circular kernel}\)
\text{3D spherical kernel}\)
\text{Refer to CGAL’s manual for more details.}\)
\text{CGAL Pre-defined Cartesian Kernels}\)
\text{Support construction of points from } \text{double } \text{Cartesian coordinates.}\)
\text{Support exact geometric predicates.}\)
\text{Handle geometric constructions differently:}\)
\text{CGAL::Exact_predicates_inexact_constructions_kernel}\)
\text{Geometric constructions may be inexact due to round-off errors.}\)
\text{It is however more efficient and sufficient for most CGAL algorithms.}\)
\text{CGAL::Exact_predicates_exact_constructions_kernel}\)
\text{CGAL::Exact_predicates_exact_constructions_kernel_with_sqrt}\)
\text{Its number type supports the exact square-root operation.}\)


```cpp
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/intersections.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_2 Point_2;
typedef Kernel::Segment_2 Segment_2;
typedef Kernel::Line_2 Line_2;

int main() {
    Point_2 p(1, 1), q(2, 3), r(-12, 19);
    Line_2 line(p, q);
    Segment_2 seg(r, p);
    auto result = CGAL::intersection(seg, line);
    if (result) {
        if (const Segment_2* s = boost::get<Segment_2>(&result)) {
            // handle segment
        } else {
            const Point_2* p = boost::get<Point_2>(&result);
            // handle point
        }
    }
    return 0;
}
```

---

**Computational Geometry Algorithm Library**

- Generic data structures are parameterized with Traits
- Separates algorithms and data structures from the geometric kernel.
- Generic algorithms are parameterized with iterator ranges
- Decouples the algorithm from the data structure.

---

**Cgal Bibliography**