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  Parametrisation  Streamlines  Ridge Detection  Neighbor Search  Kinetic Data Structures

Envelopes  Arrangements  Intersection Detection  Minkowski Sums  PCA  Polytope Distance  QP Solver
Some CGAL Commercial Users
**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. INRIA Saclay - Île de France
4. CNRS - LIRIS
5. GeometryFactory
6. MPII Saarbrücken
7. Freie Universität Berlin
8. University of Technology Braunschweig
9. Tel Aviv University
10. ETH Zürich
11. University of Crete and FO.R.T.H.
12. Università della Svizzera italiana
13. Universidade Federal de Pernambuco
14. 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></td>
<td>CGAL founded</td>
</tr>
<tr>
<td>1998</td>
<td>July 1.1</td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td>Aug 2.3</td>
<td>Work continued after end of European support</td>
</tr>
<tr>
<td>2001</td>
<td>May 2.4</td>
<td>CGAL participated in <strong>Editorial Board</strong> established</td>
</tr>
<tr>
<td>2002</td>
<td>Nov 3.0</td>
<td>CGAL participated in <strong>Geometry Factory</strong> founded</td>
</tr>
<tr>
<td>2003</td>
<td>Dec 3.1</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>May 3.2</td>
<td></td>
</tr>
<tr>
<td>2005</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 <strong>Google Summer of Code 2010</strong></td>
</tr>
<tr>
<td>2011</td>
<td>Apr 3.8, Aug 3.9</td>
<td>CGAL participated in <strong>GSoC 2011</strong></td>
</tr>
<tr>
<td>2012</td>
<td>Mar 4.0, Oct 4.1</td>
<td>CGAL participated in <strong>GSoC 2012</strong></td>
</tr>
<tr>
<td>2013</td>
<td>Mar 4.2, Oct 4.3</td>
<td>CGAL participated in <strong>GSoC 2013</strong></td>
</tr>
<tr>
<td>2014</td>
<td>Mar 4.4</td>
<td>CGAL participates in <strong>GSoC 2014</strong></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
120 packages
60 commercial users
30 active developers
6 months release cycle
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 adaptable to user code
  - 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,...

**Visualization**

Files  
I/O  
Number Types  
Generators  
...
**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></td>
<td>Predicates</td>
</tr>
<tr>
<td>point</td>
<td>comparison</td>
</tr>
<tr>
<td>vector</td>
<td>orientation</td>
</tr>
<tr>
<td>triangle</td>
<td>containment</td>
</tr>
<tr>
<td>iso rectangle</td>
<td></td>
</tr>
<tr>
<td>circle</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
\textbf{CGAL Kernel Affine Geometry}

\begin{itemize}
  \item point - origin \quad \rightarrow \quad \text{vector}
  \item point - point \quad \rightarrow \quad \text{vector}
  \item point + vector \quad \rightarrow \quad \text{point}
  \item point + point \quad \leftarrow \quad \text{Illegal}
  \item \text{midpoint}(a, b) = a + 1/2 \times (b - a)
\end{itemize}
CGAL Kernel Classification

- Dimension: 2, 3, arbitrary
- Number types:
  - Ring: $+, -, \times$
  - Euclidean ring (adds integer division and gcd) (e.g., CGAL::Gmpz).
  - Field: $+, -, \times, /$ (e.g., CGAL::Gmpq).
  - Exact sign evaluation for expressions with roots (Field_with_sqrt).
- Coordinate representation
  - Cartesian — requires a field number type or Euclidean ring if no constructions are performed.
  - Homegeneous — requires Euclidean ring.
- Reference counting
- Exact, Filtered
\textbf{CGAL Kernels and Number Types}

### Cartesian representation

<table>
<thead>
<tr>
<th>point</th>
<th>( x = \frac{hx}{hw} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( y = \frac{hy}{hw} )</td>
</tr>
</tbody>
</table>

### Homogeneous representation

<table>
<thead>
<tr>
<th>point</th>
<th>( hx )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( hy )</td>
</tr>
<tr>
<td></td>
<td>( hw )</td>
</tr>
</tbody>
</table>

### Intersection of two lines

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

\[
(h, y) = \left( \begin{array}{cc|c}
    b_1 & c_1 \\
    b_2 & c_2 \\
    a_1 & b_1 \\
    a_2 & b_2 \\
\end{array} \right) - \left( \begin{array}{cc|c}
    a_1 & c_1 \\
    a_2 & c_2 \\
    a_1 & b_1 \\
    a_2 & b_2 \\
\end{array} \right)
\]

### Field operations

\[
\begin{align*}
\begin{cases}
    a_1hx + b_1hy + c_1hw = 0 \\
    a_2hx + b_2hy + c_2hw = 0
\end{cases}
\end{align*}
\]

### Ring operations

\[
(hx, hy, hw) = \left( \begin{array}{cc|c}
    b_1 & c_1 \\
    b_2 & c_2 \\
\end{array} \right) - \left( \begin{array}{cc|c}
    a_1 & c_1 \\
    a_2 & c_2 \\
\end{array} \right) - \left( \begin{array}{ccc}
    a_1 & b_1 \\
    a_2 & b_2 \\
\end{array} \right)
\]
Example: Kernels<NumberType>

- **Cartesian<FieldNumberType>**
  - typedef CGAL::Cartesian<Gmpq> Kernel;
  - typedef CGAL::Simple_cartesian<double> Kernel;
  - ★ No reference-counting, inexact instantiation

- **Homogeneous<RingNumberType>**
  - typedef CGAL::Homogeneous<Core::BigInt> Kernel;

- **d-dimensional Cartesian_d and Homogeneous_d**

- **Types + Operations**
  - Kernel::Point_2, Kernel::Segment_3
  - Kernel::Less_xy_2, Kernel::Construct_bisector_3
### CGAL Numerical Issues

```cpp
typedef CGAL::Cartesian<NT> Kernel;
NT sqrt2 = sqrt(NT(2));

Kernel::Point_2 p(0, 0), q(sqrt2, sqrt2);
Kernel::Circle_2 C(p, 4);

assert(C.has_on_boundary(q));
```

- **OK** if NT supports exact sqrt.
- **Assertion violation** otherwise.

![Diagram](image_url)
**CGAL Pre-defined Cartesian Kernels**

- Support construction of points from \texttt{double} Cartesian coordinates.
- Support exact geometric predicates.
- Handle geometric constructions differently:
  - \texttt{CGAL::Exact_predicates_inexact_constructions_kernel}
    - Geometric constructions may be inexact due to round-off errors.
    - It is however more efficient and sufficient for most \texttt{CGAL} algorithms.
  - \texttt{CGAL::Exact_predicates_exact_constructions_kernel}
  - \texttt{CGAL::Exact_predicates_exact_constructions_kernel_with_sqrt}
    - Its number type supports the exact square-root operation.
CGAL Special Kernels

- Filtered kernels
- 2D circular kernel
- 3D spherical kernel

Refer to CGAL’s manual for more details.
Computing the Orientation

- imperative style

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

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_2 Point_2;

int main()
{
    Point_2 p(0,0), q(10,3), r(12,19);
    return (CGAL::orientation(q,p,r) == CGAL::LEFT_TURN) ? 0 : 1;
}
```

- preceptive style

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

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_2 Point_2;
typedef Kernel::Orientaion_2 Orientation_2;

int main()
{
    Kernel kernel;
    Orientation_2 orientation = kernel.orientation_2_object();

    Point_2 p(0,0), q(10,3), r(12,19);
    return (orientation(q,p,r) == CGAL::LEFT_TURN) ? 0 : 1;
}
```
#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, lin);
    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;
}
Cgal Basic 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**

A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr.
On the design of **CGAL** a computational geometry algorithms library.

A. Fabri and S. Pion.
A generic lazy evaluation scheme for exact geometric computations.

M. H. Overmars.
Designing the computational geometry algorithms library **CGAL**.

The **CGAL** Project.
**CGAL User and Reference Manual.**

Efi Fogel, Ron Wein, and Dan Halperin.
**CGAL Arrangements and Their Applications, A Step-by-Step Guide.**