Computational Geometry Algorithm Library

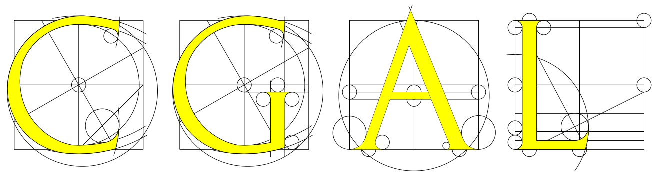
Efi Fogel

Tel Aviv University

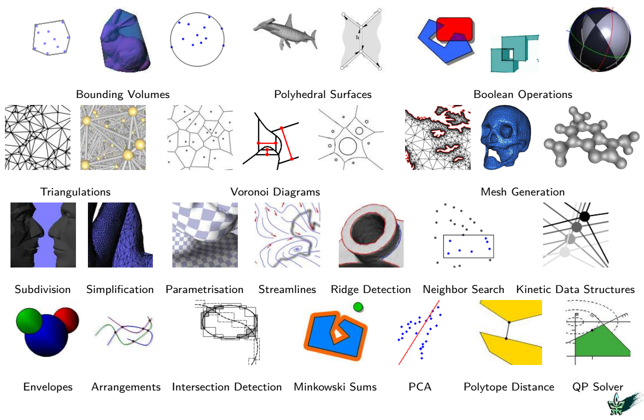
Computational Geometry
Apr. 7th, 2014

CGAL: Mission

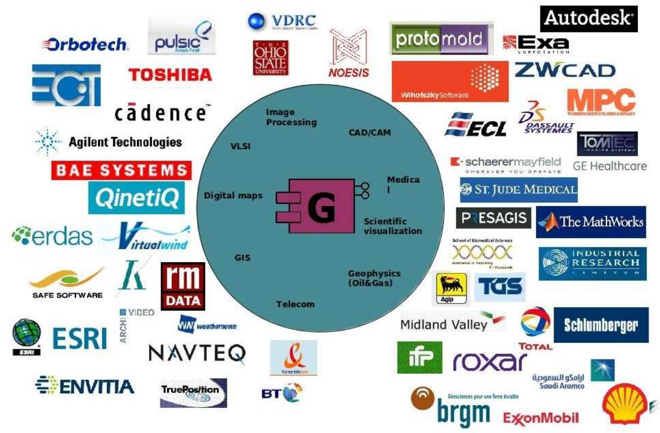
"Make the large body of geometric algorithms developed in the field of computational geometry available for industrial applications"
CGAL Project Proposal, 1996



Some of CGAL Content



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. Universitá della Svizzera italiana
13. Universidade Federal de Pernambuco
14. Instituto Nacional de Matemática Pura e Aplicada

CGAL History

Year	Version Released	Other Milestones
1996		CGAL founded
1998	July 1.1	
1999		Work continued after end of European support
2001	Aug 2.3	Editorial Board established
2002	May 2.4	
2003	Nov 3.0	GEOMETRY FACTORY founded
2004	Dec 3.1	
2006	May 3.2	
2007	Jun 3.3	
2009	Jan 3.4, Oct 3.5	
2010	Mar 3.6, Oct 3.7	CGAL participated in Google Summer of Code 2010
2011	Apr 3.8, Aug 3.9	CGAL participated in GSoC 2011
2012	Mar 4.0, Oct 4.1	CGAL participated in GSoC 2012
2013	Mar 4.2, Oct 4.3	CGAL participated in GSoC 2013
2014	Mar 4.4	CGAL participates in GSoC 2014

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

Visualization
Files
I/O
Number Types
Generators
...

Support Library

Configurations, Assertions,...

CGAL Kernel Concept

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

Primitives 2D, 3D, dD		Operations	
		Predicates	**Constructions**
point	●	comparison	intersection
vector	→	orientation	squared distance
triangle	△	containment	...
iso rectangle	▭	...	
circle	○		
...			

CGAL Kernel Affine Geometry

point - origin → vector
point - point → vector
point + vector → point

point + point ← Illegal
midpoint$(a, b) = a + 1/2 \times (b - a)$

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_sqr).
- Coordinate representation
  - Cartesian — requires a *field* number type or *Euclidean ring* if no constructions are performed.
  - Homegeneous — requires *Euclidean ring*.
- Reference counting
- Exact, Filtered

CGAL Kernels and Number Types

Cartesian representation	Homogeneous representation
point $\begin{vmatrix} x = \frac{hx}{hw} \\ y = \frac{hy}{hw} \end{vmatrix}$	point $\begin{vmatrix} hx \\ hy \\ hw \end{vmatrix}$

Intersection of two lines

$\begin{cases} a_1x + b_1y + c_1 = 0 \\ a_2x + b_2y + c_2 = 0 \end{cases}$	$\begin{cases} a_1hx + b_1hy + c_1hw = 0 \\ a_2hx + b_2hy + c_2hw = 0 \end{cases}$
$(x, y) = \left(\frac{\begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}}, -\frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} \right)$	$(hx, hy, hw) = \left(\begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}, -\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}, \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} \right)$
Field operations	Ring operations

Example: Kernels<NumberType>

- Cartesian<FieldNumberType>
  - `typedef CGAL::Cartesian<Gmpq> Kernel;`
  - `typedef CGAL::Simple_cartesian<double> Kernel;`
    - ⋆ No reference-counting, inexact instantiation
- Homogeneous<RingNumberType>
  - `typpdef 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`

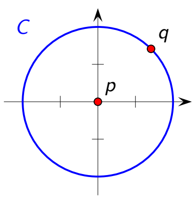
CGAL Numerical Issues

```
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.



CGAL Pre-defined Cartesian Kernels

- Support construction of points from `double` Cartesian coordinates.
- Support exact geometric predicates.
- Handle geometric constructions differently:
  - `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 CGAL algorithms.
  - `CGAL::Exact_predicates_exact_constructions_kernel`
  - `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

```
#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;
}
```

- precative style

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

typedef CGAL::Exact_predicates_inexact_constructions_kernel   Kernel;
typedef Kernel::Point_2                                        Point_2;
typedef Kernel::Orientation_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;
}
```

Computing the Intersection

```
#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.
Software — Practice and Experience, 30(11):1167–1202, 2000. Special Issue on Discrete Algorithm Engineering.

A. Fabri and S. Pion.
A generic lazy evaluation scheme for exact geometric computations.
In 2nd *Library-Centric Software Design Workshop*, 2006.

M. H. Overmars.
Designing the computational geometry algorithms library CGAL.
In *Proceedings of ACM Workshop on Applied Computational Geometry, Towards Geometric Engineering*, volume 1148, pages 53–58, London, UK, 1996. Springer.

The CGAL Project.
CGAL User and Reference Manual.
CGAL Editorial Board, 4.4 edition, 2014. http://doc.cgal.org/4.2/CGAL.CGAL/html/index.html .

Efi Fogel, Ron Wein, and Dan Halperin.
CGAL Arrangements and Their Applications, A Step-by-Step Guide.
Springer, 2012.