Introduction to the
Computational Geometry Algorithms Library

Monique Teillaud

www.cgal.org

January 2013

Overview

- The CGAL Open Source Project
- Contents of the Library
- Kernels and Numerical Robustness

Part I

The CGAL Open Source Project

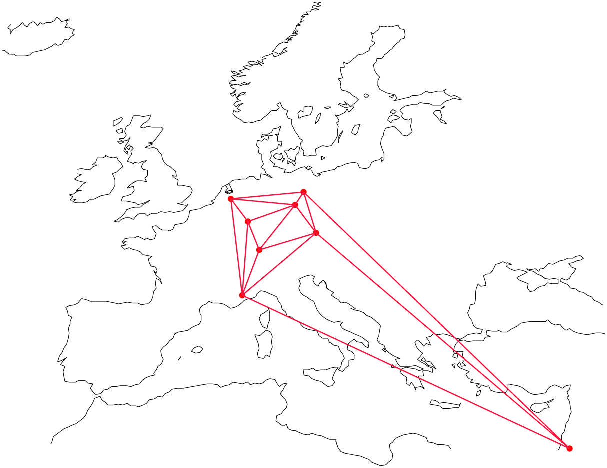
Goals

- Promote the research in Computational Geometry (CG)
- *"make the large body of geometric algorithms developed in the field of CG available for industrial applications"*

$\Rightarrow$ **robust programs**

History

- Development started in 1995



History

- Development started in 1995

- January, 2003: creation of **GEOMETRY FACTORY**
INRIA startup
sells commercial licenses, support, customized developments

- November, 2003: Release 3.0 - **Open Source Project**
  - new contributors

- September, 2012: Release 4.1

License

- a few basic packages under **LGPL**
- most packages under **GPLv3+**
  - free use for Open Source code
  - commercial license needed otherwise

Distribution

- from the INRIA gforge

- included in Linux distributions (Debian, etc)
- available through macport

- CGAL triangulations integrated in Matlab
- Scilab interface to CGAL triangulations and meshes

- CGAL-bindings
  - CGAL triangulations, meshes, etc,
    can be used in Java or Python
  - implemented with SWIG

CGAL in numbers

- 500,000 lines of **C++** code
- several platforms
  g++ (Linux MacOS Windows), VC++
- $>$ 1,000 downloads per month on the gforge
- 50 developers registered on developer list
  ($\sim$ 20 active)

Development process

- Packages are **reviewed**.

- 1 internal release per day
- Automatic **test suites** running on all supported compilers/platforms

Users

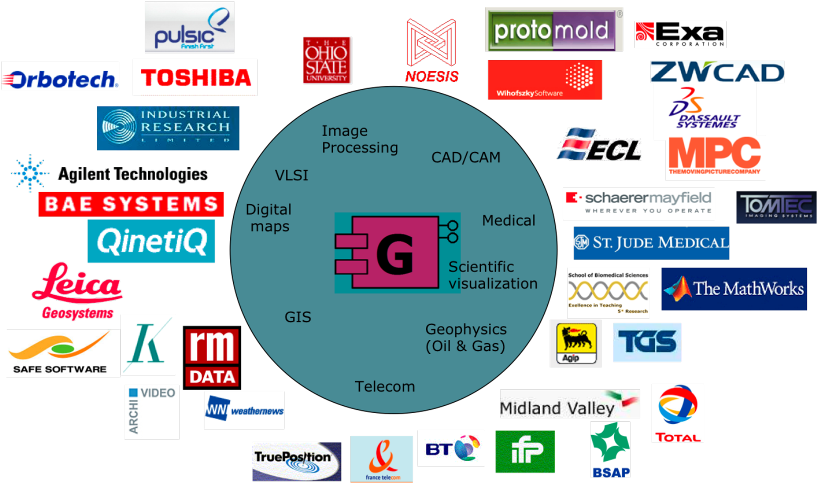
List of identified users in various fields

- Molecular Modeling
- Particle Physics, Fluid Dynamics, Microstructures
- Medical Modeling and Biophysics
- Geographic Information Systems
- Games
- Motion Planning
- Sensor Networks
- Architecture, Buildings Modeling, Urban Modeling
- Astronomy
- 2D and 3D Modelers
- Mesh Generation and Surface Reconstruction
- Geometry Processing
- Computer Vision, Image Processing, Photogrammetry
- Computational Topology and Shape Matching
- Computational Geometry and Geometric Computing

More non-identified users. . .

Customers of GEOMETRY FACTORY

(end 2008)



Part II

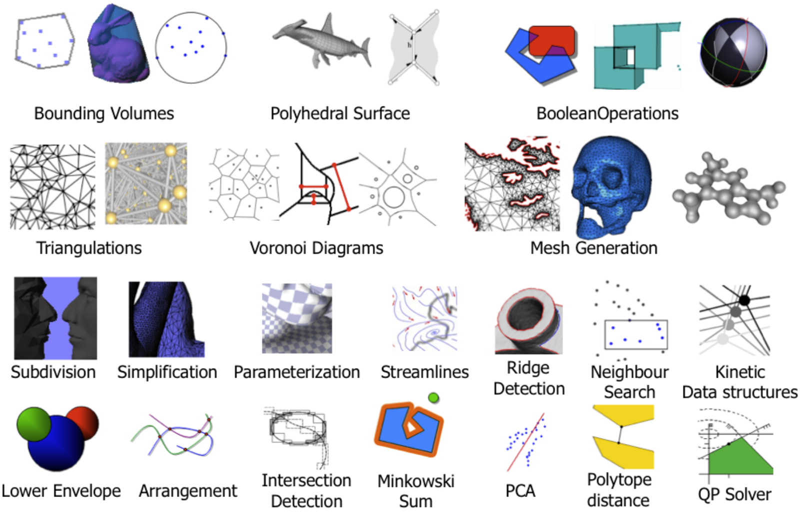
Contents of CGAL

Structure

- Kernels
- Various packages
- Support Library

  STL extensions, I/O, generators, timers...

Some packages



Part III

Numerical Robustness

The CGAL Kernels

- 2D, 3D, dD “Rational” kernels
- 2D circular kernel
- 3D spherical kernel

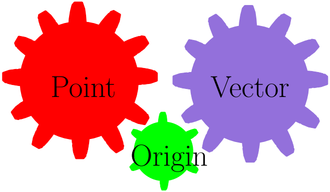
In the kernels

- Elementary geometric objects
- Elementary computations on them

Primitives	Predicates	Constructions
2D, 3D, dD	• comparison	• intersection
• Point	• Orientation	• squared distance
• Vector	• InSphere	...
• Triangle	...	
• Circle		
...		

Affine geometry

Point - Origin $\rightarrow$ Vector
Point - Point $\rightarrow$ Vector
Point + Vector $\rightarrow$ Point



Point + Point **illegal**

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

Kernels and number types

Cartesian representation

Point $\left| \begin{array}{l} x = \frac{hx}{hw} \\ y = \frac{hy}{hw} \end{array} \right.$

Homogeneous representation

Point $\left| \begin{array}{l} hx \\ hy \\ hw \end{array} \right.$

Kernels and number types

Cartesian representation

$$\text{Point}\ \left|\ \begin{array}{l} x = \frac{hx}{hw} \\ y = \frac{hy}{hw} \end{array}\right.$$

Homogeneous representation

$$\text{Point}\ \left|\ \begin{array}{l} hx \\ hy \\ hw \end{array}\right.$$

- ex: Intersection of two lines -

$$\begin{cases} a_1 x + b_1 y + c_1 = 0 \\ a_2 x + b_2 y + c_2 = 0 \end{cases}$$

$$\begin{cases} a_1 hx + b_1 hy + c_1 hw = 0 \\ a_2 hx + b_2 hy + c_2 hw = 0 \end{cases}$$

$$(x, y) = \left(\frac{\begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}}, -\frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} \right)$$

$$(hx, hy, hw) = \left(\begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}, -\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}, \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} \right)$$

Kernels and number types

Cartesian representation

Point $\left| \begin{array}{l} x = \frac{hx}{hw} \\ y = \frac{hy}{hw} \end{array} \right.$

Homogeneous representation

Point $\left| \begin{array}{l} hx \\ hy \\ hw \end{array} \right.$

- ex: Intersection of two lines -

$$\begin{cases} a_1 x + b_1 y + c_1 = 0 \\ a_2 x + b_2 y + c_2 = 0 \end{cases}$$

$$(x, y) = \left(\frac{\begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}}, -\frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} \right)$$

Field operations

$$\begin{cases} a_1 hx + b_1 hy + c_1 hw = 0 \\ a_2 hx + b_2 hy + c_2 hw = 0 \end{cases}$$

$$(hx, hy, hw) = \left(\begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}, -\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}, \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} \right)$$

Ring operations

The “rational” Kernels

CGAL::**Cartesian**< **FieldType** >
CGAL::**Homogeneous**< **RingType** >

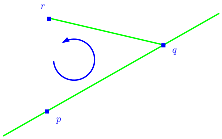
→ **Flexibility**

```
typedef double                          NumberType;
typedef Cartesian< NumberType >         Kernel;
typedef Kernel::Point_2                 Point;
```

Numerical robustness issues

Predicates = signs of polynomial expressions

Ex: **Orientation of 2D points**



$$
\begin{aligned}
orientation(p,q,r) &= \textbf{sign}\left(\det\begin{bmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ r_x & r_y & 1 \end{bmatrix}\right) \\
&= \textbf{sign}((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))
\end{aligned}
$$

Numerical robustness issues

Predicates = signs of polynomial expressions

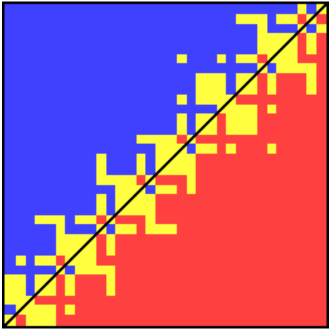
Ex: **Orientation of 2D points**

$p = (0.5 + x.u,\ 0.5 + y.u)$
$0 \leq\ x, y\ < 256, \quad u = 2^{-53}$

$q = (12, 12)$
$r = (24, 24)$

orientation(p, q, r)
evaluated with `double`

$(x, y) \mapsto$ > 0, $= 0$, < 0



`double` $\longrightarrow$ **inconsistencies** in predicate evaluations

Numerical robustness issues

Speed and exactness through

Exact Geometric Computation

Numerical robustness issues

Speed and exactness through

Exact Geometric Computation

$\neq$

exact arithmetics

Filtering Techniques (interval arithmetics, etc)
exact arithmetics only when needed

Numerical robustness issues

Speed and exactness through

Exact Geometric Computation

$\neq$

exact arithmetics

Filtering Techniques (interval arithmetics, etc)
exact arithmetics only when needed

Degenerate cases explicitly handled

The circular/spherical kernels

Circular/spherical kernels

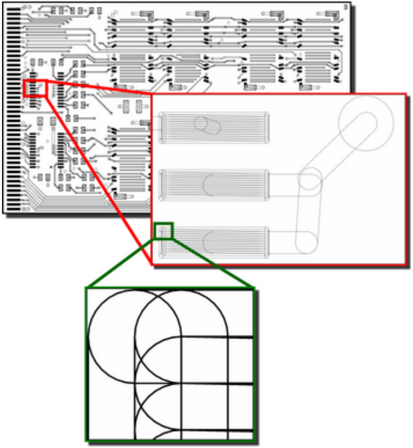
- solve needs for e.g. intersection of circles.
- **extend** the CGAL (linear) kernels

Exact computations on algebraic numbers of degree 2
= roots of polynomials of degree 2

Algebraic methods reduce **comparisons** to
computations of **signs of polynomial expressions**

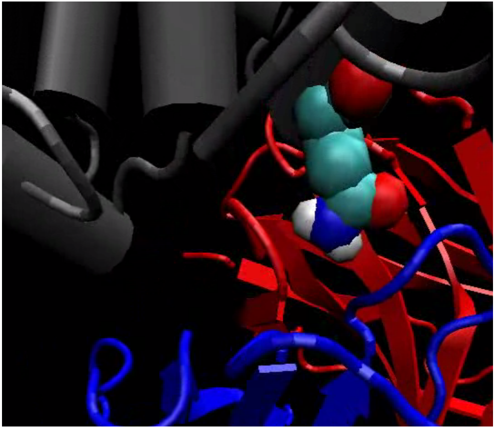
Application of the 2D circular kernel

Computation of arrangements
of 2D circular arcs and line segments



Application of the 3D spherical kernel

Computation of arrangements of 3D spheres



Sébastien Loriot, PhD thesis