

# Hitchhiker's Guide to CGAL

The Computational Geometry Algorithms Library

[www.cgal.org](http://www.cgal.org)

Efi Fogel  
Tel Aviv Univ.

Thanks to Monique Teillaud for most of the material

January 17, 2006



# Outline

- The CGAL Open Source Project
- Structure of CGAL
- The Kernel
- Numerical Robustness
- Contents of the Basic Library
- CGAL's Arrangements
- Flexibility
- Work in Progress



# Outline

- **The CGAL Open Source Project**
- Structure of CGAL
- The Kernel
- Numerical Robustness
- Contents of the Basic Library
- CGAL's Arrangements
- Flexibility
- Work in Progress



# Goals

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

⇒ Develop robust programs



# Facts

Written in C++

Follows the *generic programming* paradigm

Development started in 1995

Consortium of 6 active European sites:

- ❶ Utrecht University (XYZ Geobench)
- ❷ INRIA Sophia Antipolis (C++GAL)
- ❸ ETH Zürich (Plageo)
- ❹ MPII Saarbrücken (LEDA)
- ❺ Tel Aviv University
- ❻ Freie Universität Berlin
- ❼ RISC Linz
- ❽ Martin-Luther-Universität Halle



# History

- 1999 — Work continued in several sites after the end of European support
- 2001 — **Editorial Board** created
- Aug 2001 — version 2.3 released
- May 2002 — version 2.4 released
- January 2003 — **GEOMETRY FACTORY** was created
  - An INRIA startup
  - Sells commercial licenses, support, customized develop.
- November 2003 — Version 3.0 released
- December 2004 — Version 3.1 released
- April 2006 — Version 3.2 ?



# License

- *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



## CGAL in numbers

- 350.000 lines of C++ code
- ~2000 pages manual
- Release cycle of ~12 months
- CGAL 2.4: 9300 downloads (18 months)
- CGAL 3.0.1: 6200 downloads (8 months)
- 4000 subscribers to the announcement list (7000 for gcc)
- 800 users registered on discussion list (600 in gcc-help)
- 50 developers registered on developer list



# Supported Platforms and Compilers

- Linux, Irix, Solaris, Windows
- Mac OS X (3.1)
- g++, SGI CC, SunProCC, VC7, Intel



# Editorial Board

- Is responsible for the quality of CGAL  
Reviews new packages
- 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)

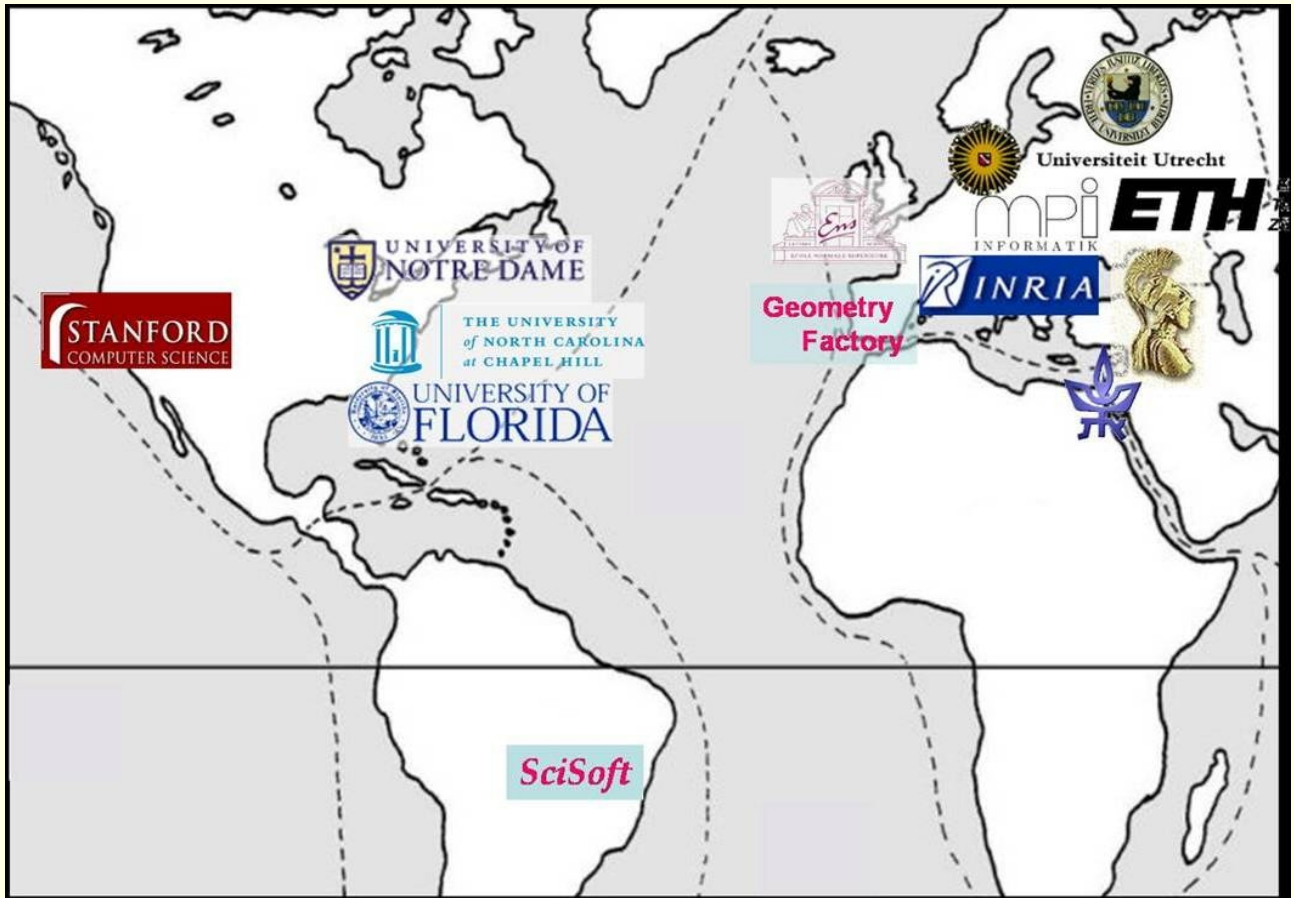
Lutz Kettner (Max-Planck-Institut für Informatik)  
Sylvain Pion (INRIA Sophia Antipolis)  
Monique Teillaud (INRIA Sophia Antipolis)  
Remco Veltkamp (Utrecht University)  
Ron Wein (Tel Aviv University)  
Mariette Yvinec (INRIA Sophia Antipolis)



# Tools

- Own manual tools:  $\text{\LaTeX}$   $\longrightarrow$  ps, pdf, html
- CVS server for version control
  - On the move to [GForge](#) — A collaborative development environment
- Developer manual
- Mailing list for developers
- 2-3 developers meetings per year, 1 week long
- 1 internal release per day
- Automatic test suites on all supported configurations





# Users

## 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, University of Texas at Austin
- Xiangmin Jiao, University of Illinois at Urbana-Champaign (Surface Mesh Overlay)
- Peter Coveney and co-workers, University of London
- . . .



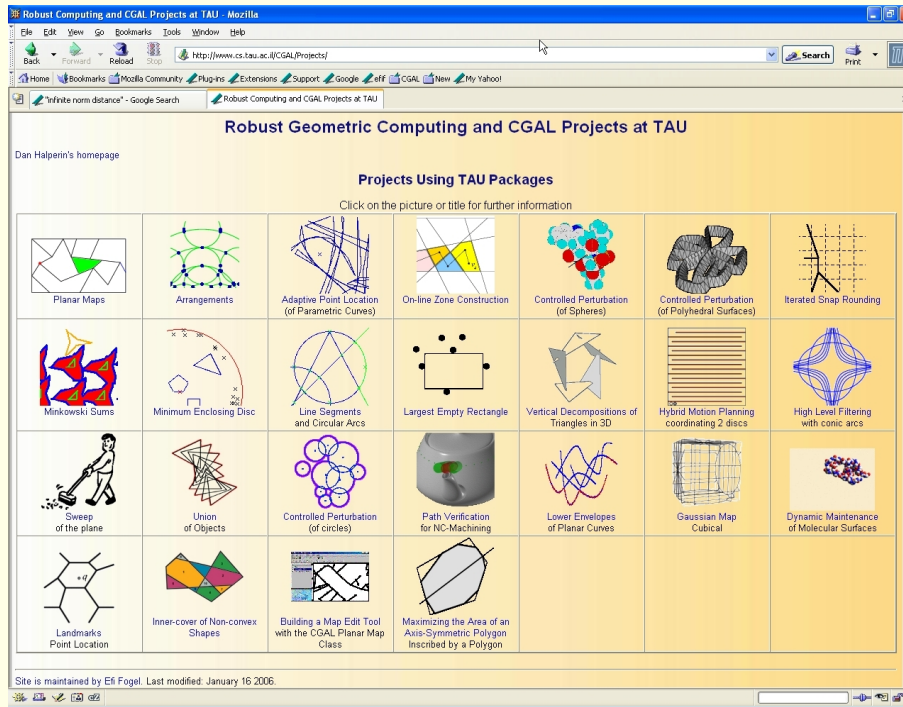
# Users

## Teaching

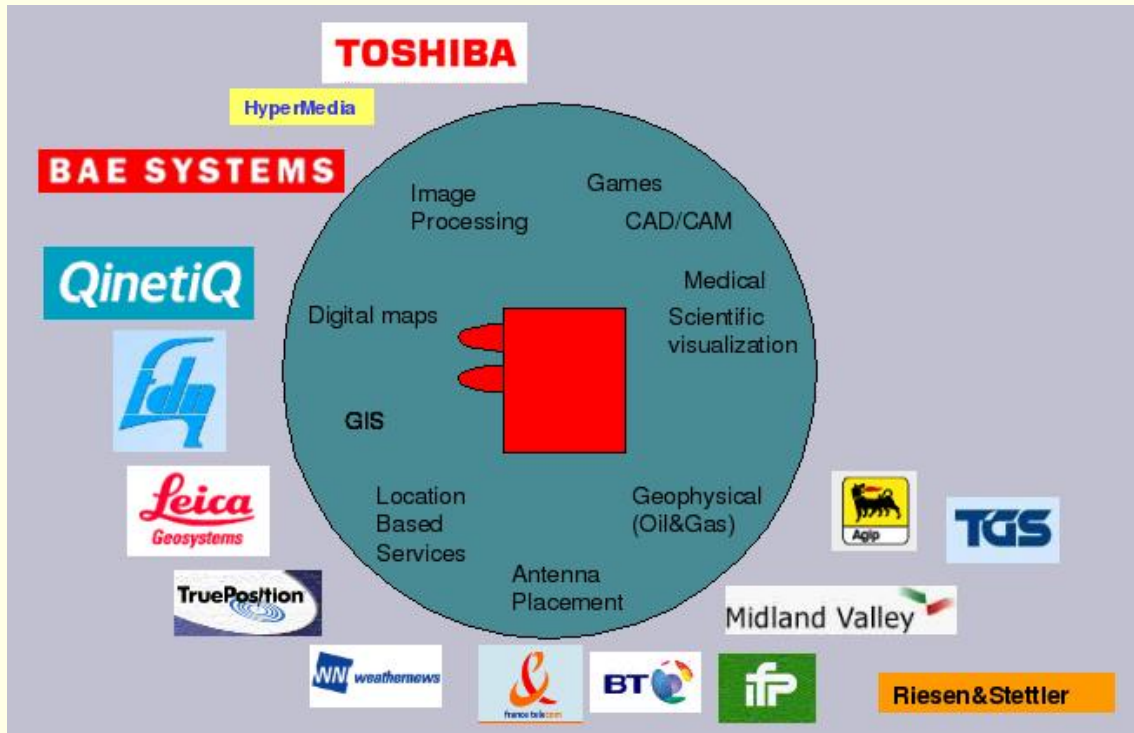
- Leo Guibas, Siu Wing Cheng, . . .



# CGAL Projects in TAU



# Commercial Customers of Geometry Factory



# Outline

- The CGAL Open Source Project
- **Structure of CGAL**
- The Kernel
- Numerical Robustness
- Contents of the CGAL Basic Library
- CGAL's Arrangements
- Flexibility
- Work in Progress



# Structure of CGAL

## Basic Library

Algorithms and Data Structures

## Kernel

Geometric Objects of constant size  
Geometric Operations on object of constant size

## Support Library

Configurations, Assertions,...

Visualization

Files

I/O

Number Types

Generators

...



# Outline

- The CGAL Open Source Project
- Structure of CGAL
- **The CGAL Kernel**
- Numerical Robustness
- Contents of the CGAL Basic Library
- CGAL's Arrangements
- Flexibility
- Work in Progress



# In the Kernel

- Elementary geometric objects
- Elementary computations on them

<b>Primitives 2D, 3D, dD</b>	<b>Predicates</b>	<b>Constructions</b>
point	comparison	intersection
vector	orientation	squared distance
triangle	in sphere	. . .
iso rectangle	. . .	
circle		
. . .		

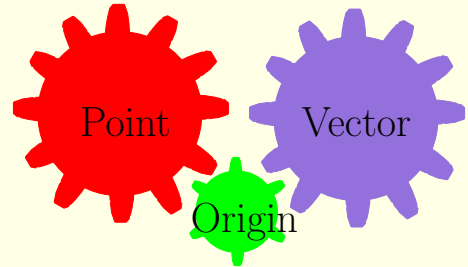


# Affine Geometry

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

point + point ← **Illegal**

$$\text{midpoint}(a, b) = a + 1/2 \times (b - a)$$



# 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

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

Intersection of two lines

$$\left\{ \begin{array}{l} a_1x + b_1y + c_1 = 0 \\ a_2x + b_2y + c_2 = 0 \end{array} \right. \left| \right. \left\{ \begin{array}{l} a_1hx + b_1hy + c_1hw = 0 \\ a_2hx + b_2hy + c_2hw = 0 \end{array} \right.$$

$$(x, y) =$$

$$\left( \left| \begin{array}{cc} b_1 & c_1 \\ b_2 & c_2 \end{array} \right|, - \left| \begin{array}{cc} a_1 & c_1 \\ a_2 & c_2 \end{array} \right| \right) \left( \left| \begin{array}{cc} a_1 & b_1 \\ a_2 & b_2 \end{array} \right|, - \left| \begin{array}{cc} a_1 & b_1 \\ a_2 & b_2 \end{array} \right| \right)$$

Field operations

$$(hx, hy, hw) =$$

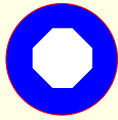
$$\left( \left| \begin{array}{cc} b_1 & c_1 \\ b_2 & c_2 \end{array} \right|, - \left| \begin{array}{cc} a_1 & c_1 \\ a_2 & c_2 \end{array} \right|, \left| \begin{array}{cc} a_1 & b_1 \\ a_2 & b_2 \end{array} \right| \right)$$

Ring operations



# C++ Templates

CGAL::Cartesian<FT>  
(CGAL::Simple\_Cartesian)



Cartesian Kernels : Field type



double

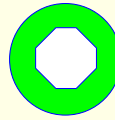


Quotient<Gmpz>



leda\_real

CGAL::Homogeneous<RT>  
(CGAL::Simple\_Homogeneous)



Homogeneous Kernels : Ring type



int



Gmpz



double

→ Flexibility

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



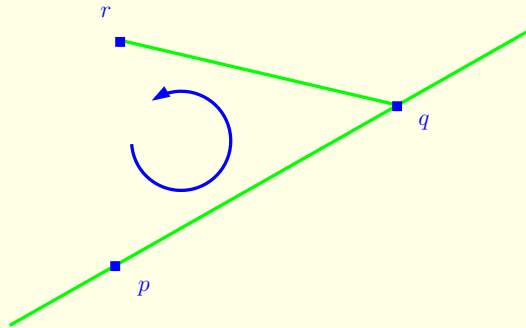
## Numerical Issues

```
typedef CGAL::Cartesian<NT>    Kernel;  
NT sqrt2 = sqrt(NT(2));  
  
Kernel::Point_2  p(0,0), q(sqrt2,sqrt2);  
Kernel::Circle_2 C(p,2);  
  
assert(C.has_on_boundary(q));
```

- OK if NT supports exact sqrt
- Assertion violation otherwise



# Orientation of 2D Points



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

# Imprecision

$$p = (0.5 + x.u, 0.5 + y.u)$$

$$0 \leq x, y < 256, u = 2^{-53}$$

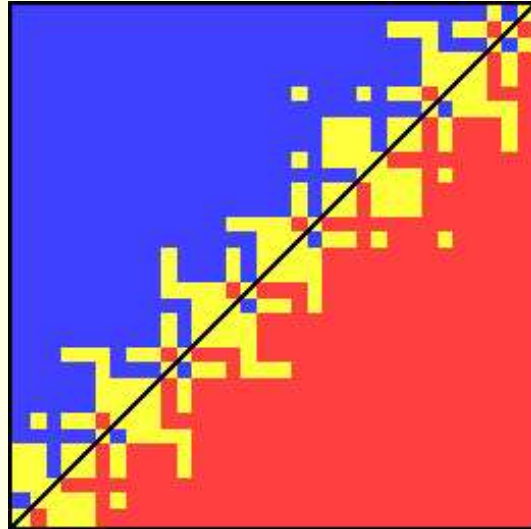
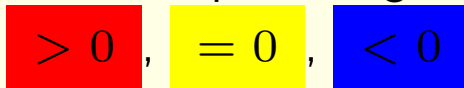
$$q = (12, 12)$$

$$r = (24, 24)$$

$$\text{orientation}(p, q, r)$$

evaluated with double

256 x 256 pixel image



→ **Inconsistencies** in predicate evaluations

[Kettner, Mehlhorn, Pion, Schirra, Yap, ESA'04]

# Outline

- The CGAL Open Source Project
- Structure of CGAL
- The Kernel
- **Numerical Robustness in CGAL**
- Contents of the Basic Library
- CGAL's Arrangements
- Flexibility
- Work in Progress



# Numerical Robustness

imprecise numerical evaluations

→ non-robustness

combinatorial result

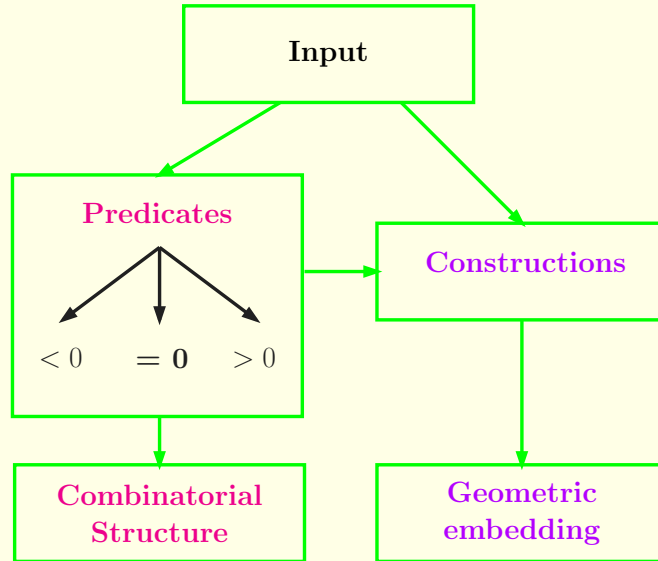
**Exact Geometric Computation**

≠

**exact arithmetic**

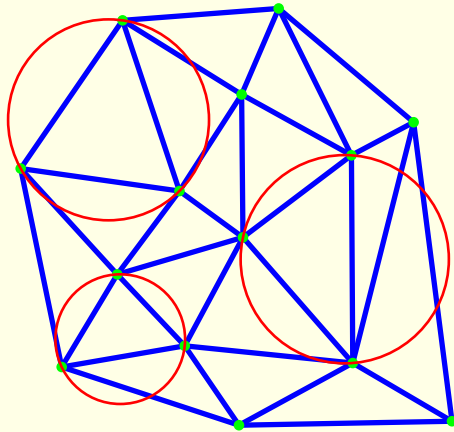


# Predicates and Constructions



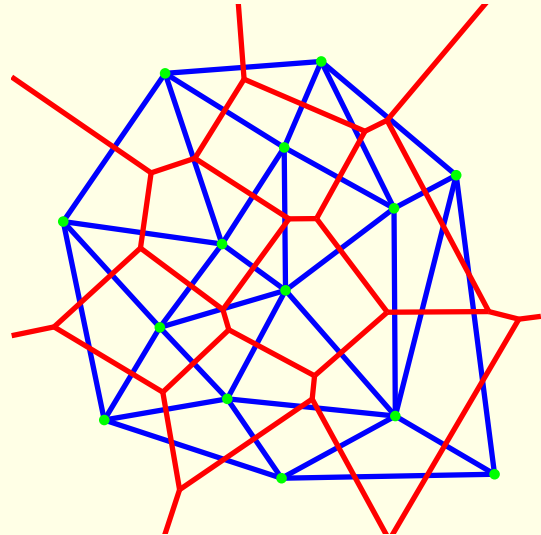
# Triangulation

## Delaunay triangulation



only *predicates* are used  
orientation, in\_sphere

## Voronoi diagram



*constructions* are needed

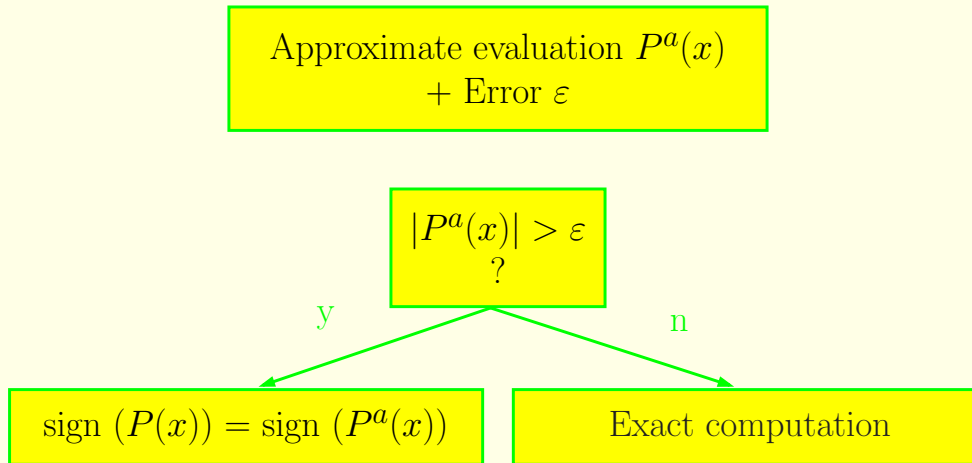
# Arithmetic Tools

- Multiprecision integers  
Exact evaluation of signs / values of polynomial expressions with integer coefficients  
CGAL::MP\_Float, GMP::mpz\_t, LEDA::integer, ...
- Multiprecision floats  
idem, with float coefficients ( $n2^m$ ,  $n, m \in \mathbb{Z}$ )  
CGAL::MP\_Float, GMP::mpf\_t, LEDA::bigfloat, ...
- Multiprecision rationals  
Exact evaluation of signs / values of rational expressions  
CGAL::Quotient< · >, GMP::mpq\_t, LEDA::rational, ...
- Algebraic numbers  
Exact comparison of roots of polynomials  
LEDA::real, Core::Expr (work in progress in CGAL)



# Filtering Predicates

sign ( $P(x)$ ) ?



# Filtering Predicates

## Static filtering

Error bound precomputed  
Faster on non-degenerate data

## Dynamic filtering

Interval arithmetic  
Faster on degenerate data  
`CGAL::Interval_nt`, `MPFR/MPFI`,  
`boost::interval`

---

`CGAL::Filtered_kernel<Kernel>` kernel wrapper [Pion]

Replaces predicates of `Kernel` by filtered and exact predicates computed with `MP_Float`

- Dynamic only in CGAL 3.0
- Static + Dynamic in CGAL 3.1

→ more generic generator also available for user's predicates



# Filtering Constructions

Number type `CGAL::Lazy_exact_nt<Exact_NT>` [Pion]

Delays exact evaluation with `Exact_NT`

- Stores a **DAG** of the expression
- Computes first an approximation with `Interval_nt`
- Allows to control the relative precision of `to_double`

`CGAL::Lazy_kernel` in progress



# Predefined Kernels

```
Exact_predicates_exact_constructions_kernel  
Filtered_kernel<Cartesian<Lazy_exact_nt<Quotient<MP_Float> > > >
```

```
Exact_predicates_exact_constructions_kernel_with_sqrt  
Filtered_kernel<Cartesian<Core::Expr> >
```

```
Exact_predicates_inexact_constructions_kernel  
Filtered_kernel<Cartesian<double> >
```



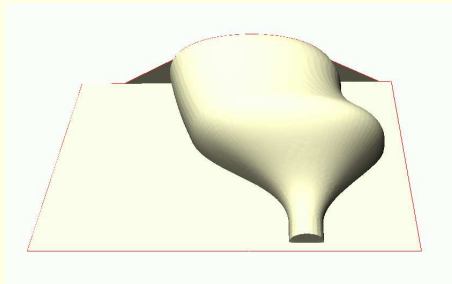
# Efficiency

## 3D Delaunay triangulation

1M random points

CGAL-3.1-I-124, Pentium-M 1.7 GHz, 1GB, g++ 3.3.2, -O2 -DNDEBUG

Simple_Cartesian<double>	48.1 sec
Simple_Cartesian<MP_Float>	2980.2 sec
Filtered_kernel (dynamic filtering)	232.1 sec
Filtered_kernel (static + dynamic filtering)	58.4 sec



Dassault Systèmes - Hair dryer

- 49.787 points
- **double** ⇒ infinite loop!
- Exact and filtered < 8 sec

# Outline

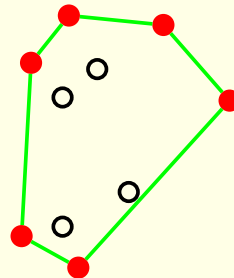
- The CGAL Open Source Project
- Structure of CGAL
- The Kernel
- Numerical Robustness
- **Contents of the CGAL Basic Library**
- CGAL's Arrangements
- Flexibility
- Work in Progress



# Convex Hull

[MPII]

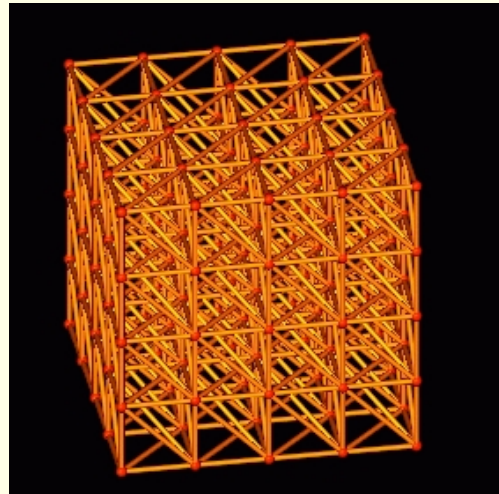
- 5 different algorithms in 2D
- 3 different algorithms in 3D



# Triangulations and Related

[INRIA]

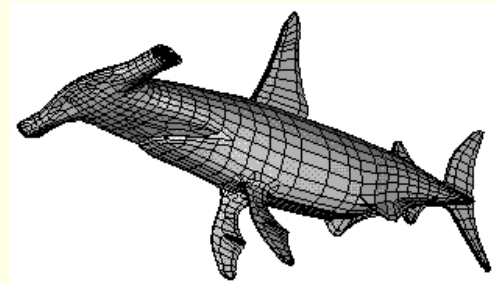
- 2D/3D Triangle/Tetrahedron based data-structure
- Fully dynamic 2D/3D Delaunay triangulation  
Delaunay hierarchy [Devillers]
- 2D/3D Regular Triangulations
- 2D Constrained Delaunay Triangulation
- 2D Apollonius diagram
- 2D Segment Voronoi Diagram
- 2D Meshes



# Polyhedra

[MPII]

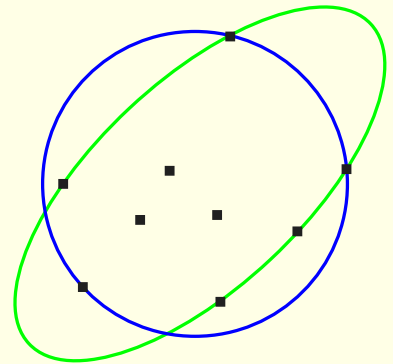
- Half-edge data-structure
- Polyhedral surface  
(orientable 2-manifold with boundary)
- 2D Nef polygons
- Nef polyhedra embedded on a sphere
- 3D Nef polyhedra



# Geometric Optimization

[ETH]

- Smallest enclosing circle and ellipse in 2D
- Smallest enclosing sphere in dD
- . . .



[Tel Aviv]

- Largest empty rectangle

# Arrangement Based Data Structures and Algorithms

[Tel Aviv]

- Planar arrangements
  - Various families of curves
  - Point-location strategies
  - Zone computation
  - Sweep line
  - Overlay
- Snap Rounding
- Boolean Set Operations
- Envelopes
- Minkowski sums



# Search Structures

Arbitrary dimension

- Range-tree, Segment-tree, kD-tree
- Window query
- Approximate nearest neighbors
- . . .



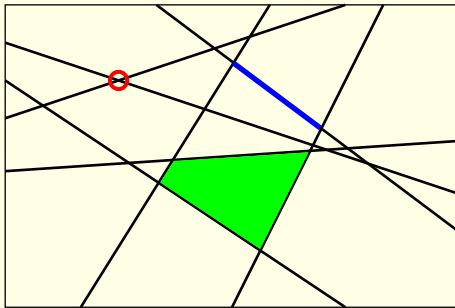
# Outline

- The CGAL Open Source Project
- Structure of CGAL
- The Kernel
- Numerical Robustness
- **Contents of the CGAL Basic Library**
- **CGAL's Arrangements**
- Flexibility
- Work in Progress

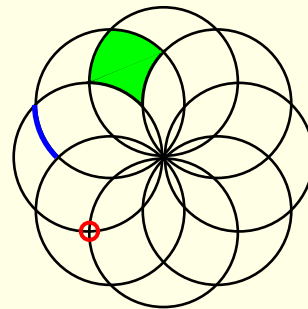


# Arrangements

Given a collection  $\Gamma$  of planar curves, the arrangement  $\mathcal{A}(\Gamma)$  is the partition of the plane into **vertices**, **edges** and **faces** induced by the curves of  $\Gamma$

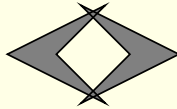


An arrangement of lines

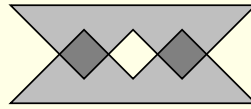


An arrangement of circles

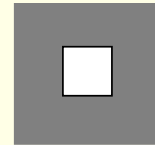
# Boolean Set Operations



Union



Intersection

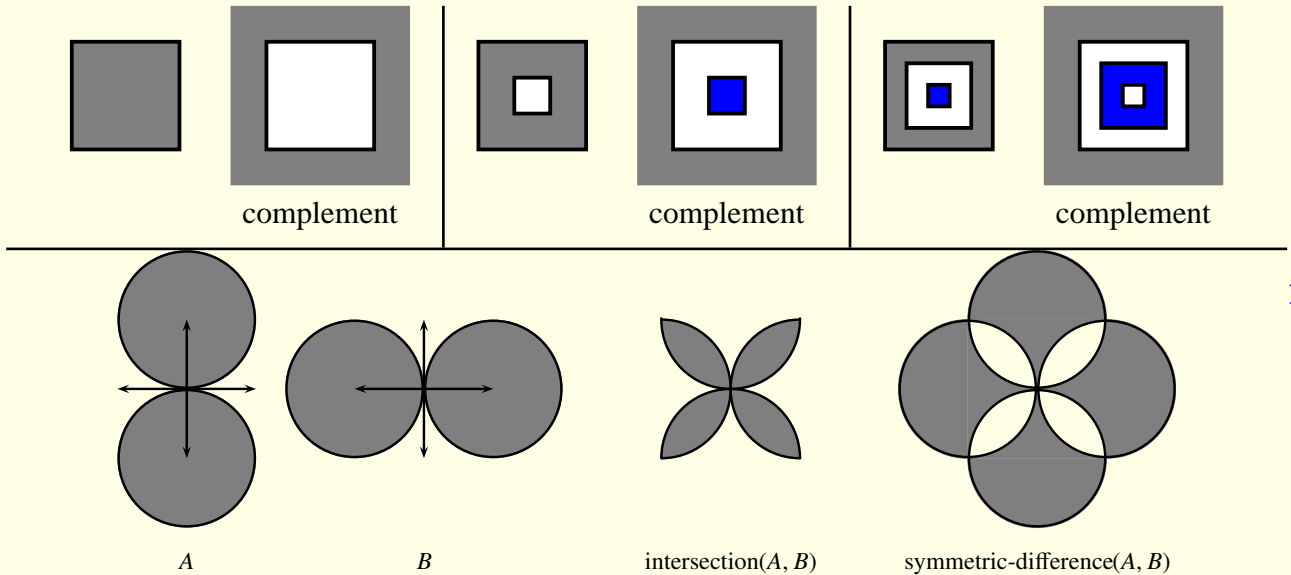


Complement

For two point sets  $P$  and  $Q$  and a point  $r$ :

<b>Intersection predicate</b>	$P \cap Q \neq \emptyset$ , overlapping cell(s) are not explicitly computed.
<b>Intersection</b>	$R = P \cap Q$
<b>Union</b>	$R = P \cup Q$
<b>Difference</b>	$R = P \setminus Q$
<b>Symmetric Difference</b>	$R = (P \setminus Q) \cup (Q \setminus P)$
<b>Complement</b>	$R = \overline{P}$
<b>Containment predicate</b>	$r \in P$

# Boolean Set Operations

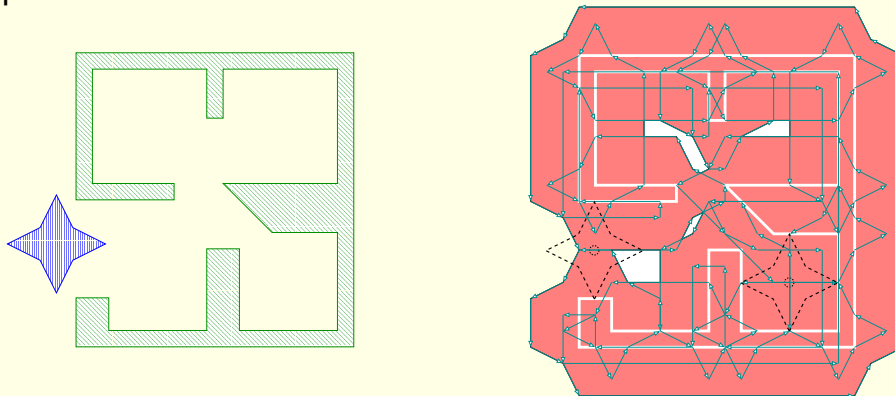


1

<sup>1</sup>Efi Fogel, Ron Wein, Baruch Zukerman, and Dan Halperin

# Minkowski Sums in $\mathbb{R}^2$

- Implemented using either decomposition or convolution<sup>2</sup>
- Based on CGAL's arrangements. Robust and exact
- Efficient
  - Running times are two orders of magnitude faster than the ones reported with CGAL 2.0



---

<sup>2</sup>Ron Wein and Dan Halperin

## Envelopes in $\mathbb{R}^3$

- Let  $F = \{f_1, f_2, \dots, f_n\}$  be a collection of partially defined bivariate functions
- The **lower envelope**  $E_F$  of  $F$  is the pointwise minimum of these functions:

$$E_F(x, y) = \min f_i(x, y) \quad x, y \in R$$

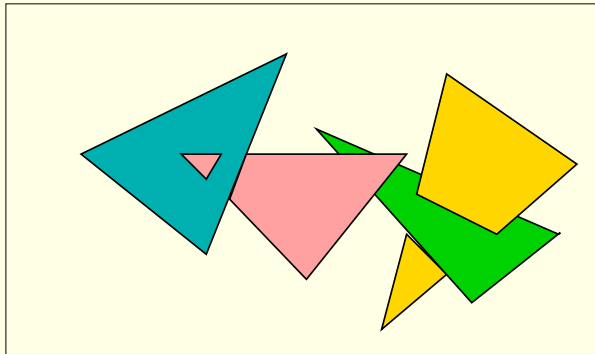
- If a function  $f_i$  is not defined over  $(x, y)$ ,  $f_i(x, y) = \infty$
- The **minimization diagram** is the partition of the plane into maximal connected regions such that  $E_F$  is attained by the same subset of functions over each region<sup>3</sup>

---

<sup>3</sup>Michal Meyerovitch and Dan Halperin

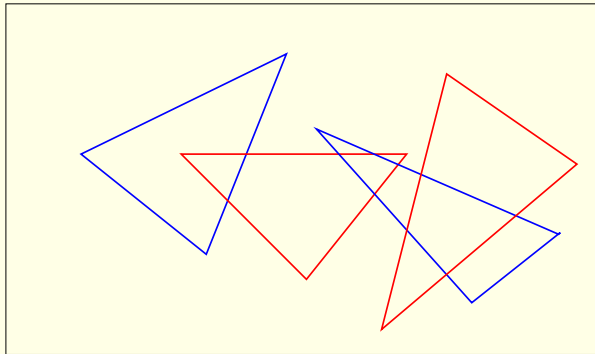
# Divide & Conquer Algorithm

- Partition  $F$  into 2 sub-collections  $F_1, F_2$
- Recursively construct the *minimization diagrams*  $M_1, M_2$
- Merge  $M_1, M_2$  to obtain the minimization diagram of  $F$



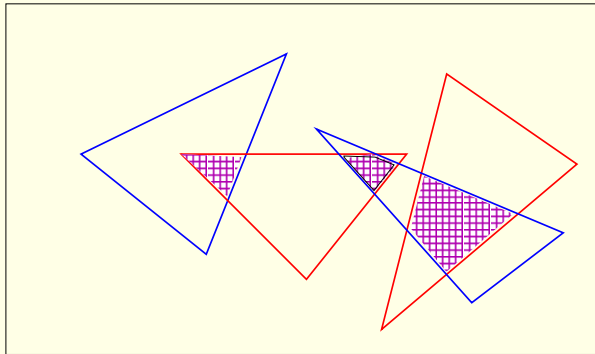
# Divide & Conquer Algorithm

- Partition  $F$  into 2 sub-collections  $F_1, F_2$
- Recursively construct the *minimization diagrams*  $M_1, M_2$
- Merge  $M_1, M_2$  to obtain the minimization diagram of  $F$



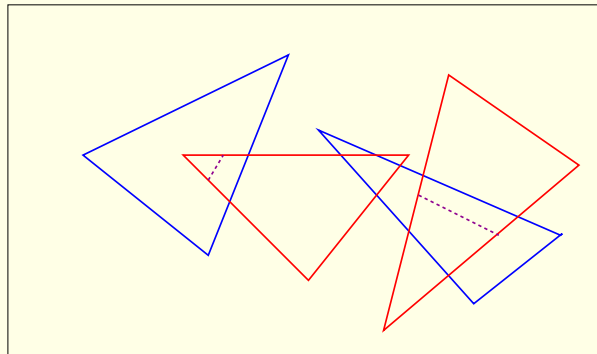
# Divide & Conquer Algorithm

- Partition  $F$  into 2 sub-collections  $F_1, F_2$
- Recursively construct the *minimization diagrams*  $M_1, M_2$
- Merge  $M_1, M_2$  to obtain the minimization diagram of  $F$



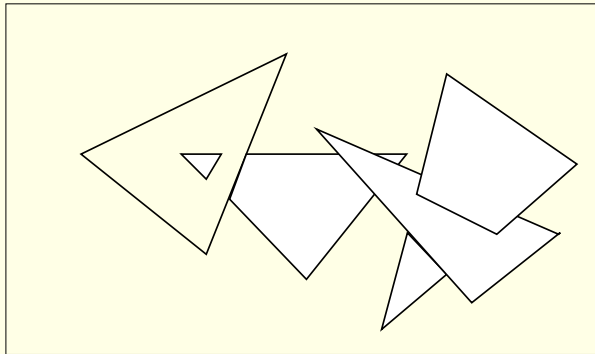
# Divide & Conquer Algorithm

- Partition  $F$  into 2 sub-collections  $F_1, F_2$
- Recursively construct the *minimization diagrams*  $M_1, M_2$
- Merge  $M_1, M_2$  to obtain the minimization diagram of  $F$



# Divide & Conquer Algorithm

- Partition  $F$  into 2 sub-collections  $F_1, F_2$
- Recursively construct the *minimization diagrams*  $M_1, M_2$
- Merge  $M_1, M_2$  to obtain the minimization diagram of  $F$



# Lower Envelope

Based on arrangements

Robust & exact

Exploits:

Overlay

Isolated points

Zone traversal

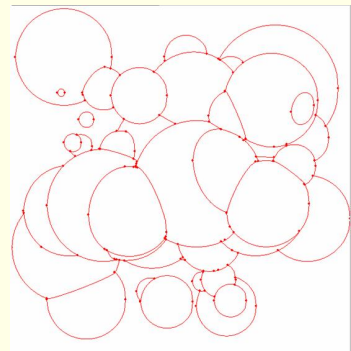
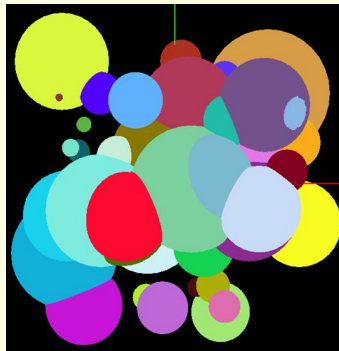
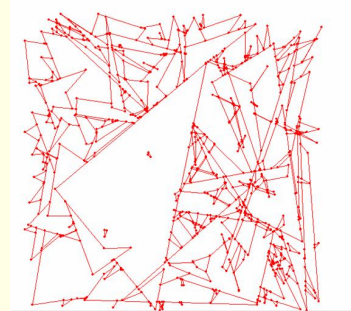
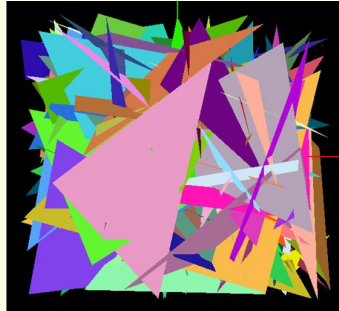
Currently works for:

Triangles

Spheres

Near-future goal:

Quadrics (with  
MPII)



# Outline

- The CGAL Open Source Project
- Structure of CGAL
- The Kernel
- Numerical Robustness
- Contents of the Basic Library
- CGAL's Arrangements
- **Flexibility in the CGAL Basic Library**
- Work in Progress



# Geometric “Traits” classes

- Provide geometric objects + predicates + constructors
- Encapsulates implementation details
  - The number type used
  - The coordinate representation
  - The geometric or algebraic computation methods
- The Kernel can be used as a traits class for several algorithms
- Special traits classes are provided
- Users can develop and plug their own traits class



# Geometric “Traits” classes

## Examples

```
convex_hull_2<InputIterator, OutputIterator, Traits>  
Polygon_2<Traits, Container>  
Polyhedron_3<Traits, Hds>  
Triangulation_3<Traits, Tds>  
Arrangement_2<Traits, Dcel>  
Min_circle_2<Traits>  
Range_tree_k<Traits>  
...
```

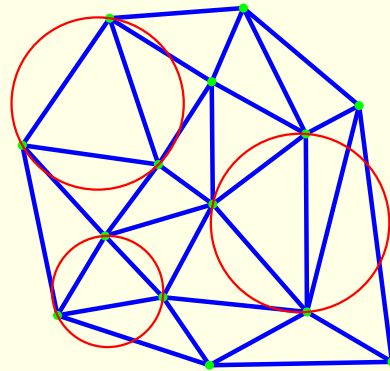


# Playing with Traits Classes

## Delaunay Triangulation

Requirements for a traits class:

- point
- orientation test, in\_circle test

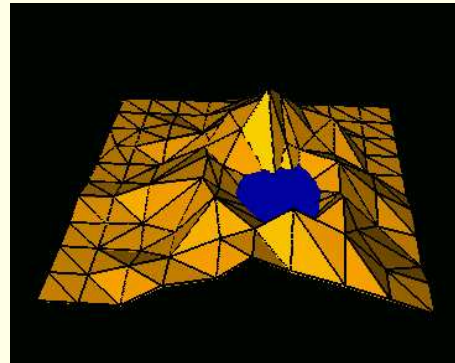


```
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;  
typedef CGAL::Delaunay_triangulation_2<Kernel> Delaunay;
```

# Playing with Traits Classes

## Delaunay Triangulation - Terrain

- 3D points  
coordinates  $(x, y, z)$
- orientation, `in_circle`  
on  $x$  and  $y$  coordinates



```
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;  
typedef CGAL::Triangulation_euclidean_traits_xy_3<Kernel> Traits;  
typedef CGAL::Delaunay_triangulation_2<Traits> Terrain;
```

# Outline

- The CGAL Open Source Project
- Structure of CGAL
- The Kernel
- Numerical Robustness in CGAL
- Contents of the Basic Library
- CGAL's Arrangements
- Flexibility in the Basic Library
- **Work in Progress**



## Packages (Partial List)

- Straight Line Skeleton [Cacciola]
- Quadratic Programming Solver [Schoenherr]
- Kinetic Data Structures [Russel Karavelas]
- Persistent Homology [Kettner Zomorodian]
- Surface reconstruction [Oudot Rey]
- 3D Meshes [Rineau Yvinec]
- Curved Kernel  
[Emiris Kakargias Pion Tsigaridas Teillaud]
  - Extension of the CGAL kernel
  - Algebraic issues
- . . .



# TOC

- Outline ❖
- Outline ❖
- Goals ❖
- Facts ❖
- History ❖
- License ❖
- CGAL in numbers ❖
- Supported Platforms and Compilers ❖
- Editorial Board ❖
- Tools ❖
- Users ❖
- Users ❖
- CGAL Projects in TAU ❖
- Commercial Customers of GEOMETRY FACTORY ❖
- Outline ❖
- Structure of CGAL ❖
- Outline ❖
- In the Kernel ❖
- Affine Geometry ❖
- Kernels and Number Types ❖
- C++ Templates ❖
- Numerical Issues ❖
- Orientation of 2D Points ❖
- Imprecision ❖
- Outline ❖
- Numerical Robustness ❖
- Predicates and Constructions ❖
- Triangulation ❖
- Arithmetic Tools ❖



- Filtering Predicates ❖
- Filtering Predicates ❖
- Filtering Constructions ❖
- Predefined Kernels ❖
- Efficiency ❖
- Outline ❖
- Convex Hull ❖
- Triangulations and Related ❖
- Polyhedra ❖
- Geometric Optimization ❖
- Arrangement Based Data Structures and Algorithms ❖
- Search Structures ❖
- Outline ❖
- Arrangements ❖
- Boolean Set Operations ❖
- Boolean Set Operations ❖
- Minkowski Sums in  $\mathbb{R}^2$  ❖
- Envelopes in  $\mathbb{R}^3$  ❖
- Divide & Conquer Algorithm ❖
- Divide & Conquer Algorithm ❖
- Divide & Conquer Algorithm ❖
- Divide & Conquer Algorithm ❖
- Divide & Conquer Algorithm ❖
- Lower Envelope ❖
- Outline ❖
- Geometric “Traits” classes ❖
- Geometric “Traits” classes ❖
- Playing with Traits Classes ❖
- Playing with Traits Classes ❖
- Outline ❖
- Packages (Partial List) ❖

