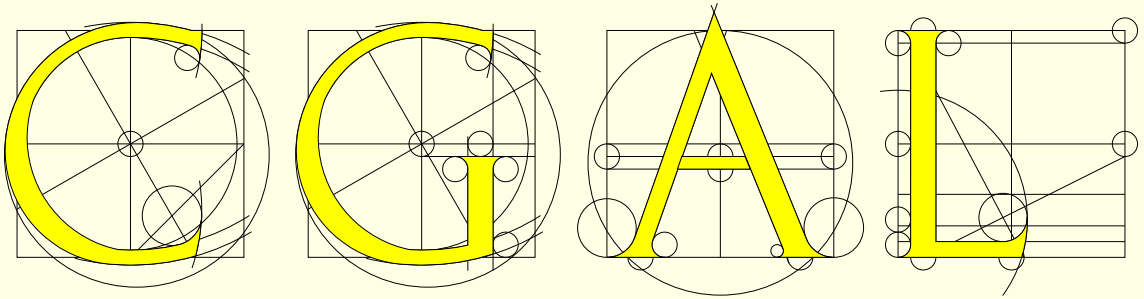
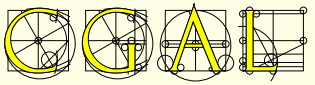


# *Computational Geometry Algorithms Library*

**www.cgal.org**

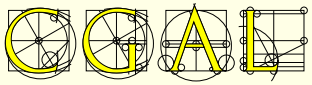
Monique Teillaud

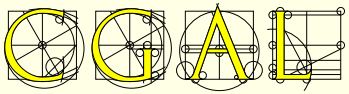


# Overview

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



# The  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**

# 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**

## CG Impact Task Force Report, 1996

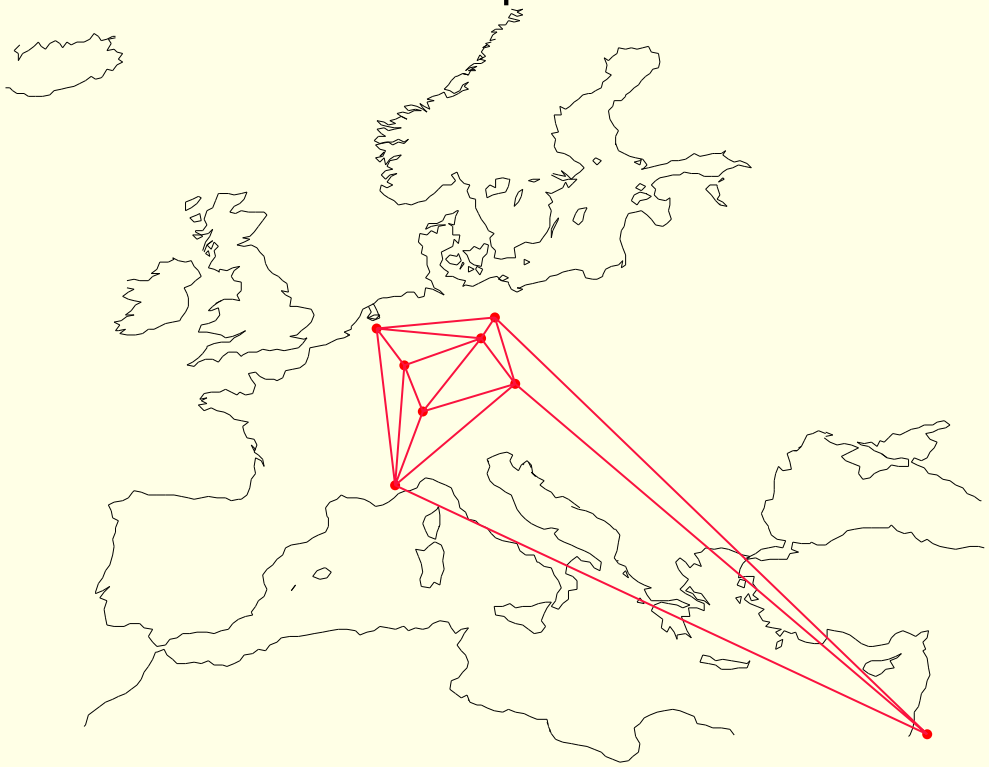
Among the key recommendations:

- Production and distribution of usable (and useful) geometric codes
- Reward structure for implementations in academia

# History

Development started 1995

Consortium of 8 European sites



RISC Linz
Utrecht University (XYZ Geobench)
INRIA Sophia Antipolis (C++GAL)
ETH Zürich (Plageo)
MPI Saarbrücken (LEDA)
Tel Aviv University
Freie Universität Berlin
Martin-Luther-Universität Halle

- Two ESPRIT LTR European Projects (1996-1999)
- Work continued after the end of European support in several sites.

- Two ESPRIT LTR European Projects (1996-1999)
- Work continued after the end of European support in several sites.

- January, 2003: **creation of Geometry Factory**

INRIAstartup
sells commercial licenses, support, customized developments

- Two ESPRIT LTR European Projects (1996-1999)
- Work continued after the end of European support in several sites.

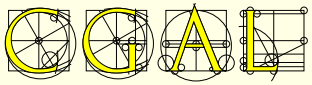
- January, 2003: **creation of Geometry Factory**

INRIAstartup
sells commercial licenses, support, customized developments

- November, 2003:

Release 3.0
**Open Source Project**





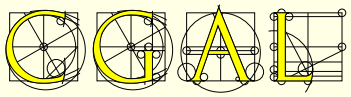
# License

- *kernel* under **LGPL**
- *basic library* under **QPL**
  - free use for OS code
  - commercial license needed otherwise

# License

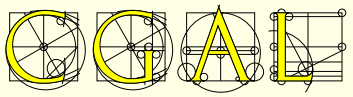
- *kernel* under **LGPL**
- *basic library* under **QPL**
  - free use for OS code
  - commercial license needed otherwise

- A guarantee for CGAL users
- Allows CGAL to become a standard
- Opens CGAL for new **contributions**



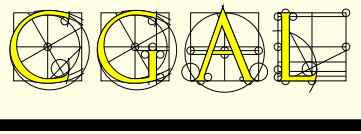
# in numbers

- 350.000 lines of C++ code
- ∼2000 pages manual



# 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)



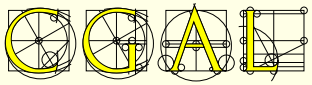
# in numbers

- 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



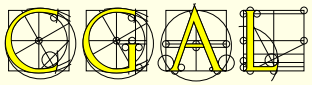


## Supported platforms

- Linux, Irix, Solaris, Windows
- Mac OS X (3.1)

- g++, SGI CC, SunProCC, VC7, Intel





# Development process

**Editorial Board** created in 2001.

- responsible for the **quality** of CGAL

  New packages are **reviewed**.

→ helps authors to get **credit** for their work.

# Development process

**Editorial Board** created in 2001.

- responsible for the **quality** of CGAL

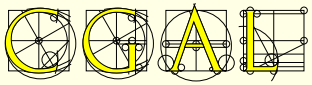
  New packages are **reviewed**.

→ helps authors to get **credit** for their work.

- 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)
Mariette Yvinec (INRIA Sophia Antipolis)





# Tools

- Own manual tools: LaTeX $\longrightarrow$ ps, pdf, html
- CVS server for version management

# Tools

- Own manual tools: LaTeX $\longrightarrow$ ps, pdf, html
- CVS server for version management

- Developer manual
- mailing list for developers
- 1-2 developers meetings per year, 1 week long

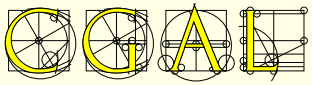
# Tools

- Own manual tools: LaTeX $\longrightarrow$ ps, pdf, html
- CVS server for version management

- Developer manual
- mailing list for developers
- 1-2 developers meetings per year, 1 week long

- 3 internal releases per week
  (1 per day before release)
- Automatic **test suites** running on all supported compilers/platforms





# Credit

**Contributors keep their identity**

# Credit

**Contributors keep their identity**

- up to 3.0.1: **names of authors** mentioned in the Preface
- 3.1 (soon) Names of authors appear at the beginning of each chapter. Section on history of the package at the end of each chapter
- CGAL developers listed on the "People" web page

# Credit

**Contributors keep their identity**

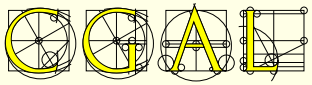
- up to 3.0.1: **names of authors** mentioned in the Preface
- 3.1 (soon) Names of authors appear at the beginning of each chapter. Section on history of the package at the end of each chapter
- CGAL developers listed on the "People" web page
- Authors publish **papers** (conferences, journals) on their packages

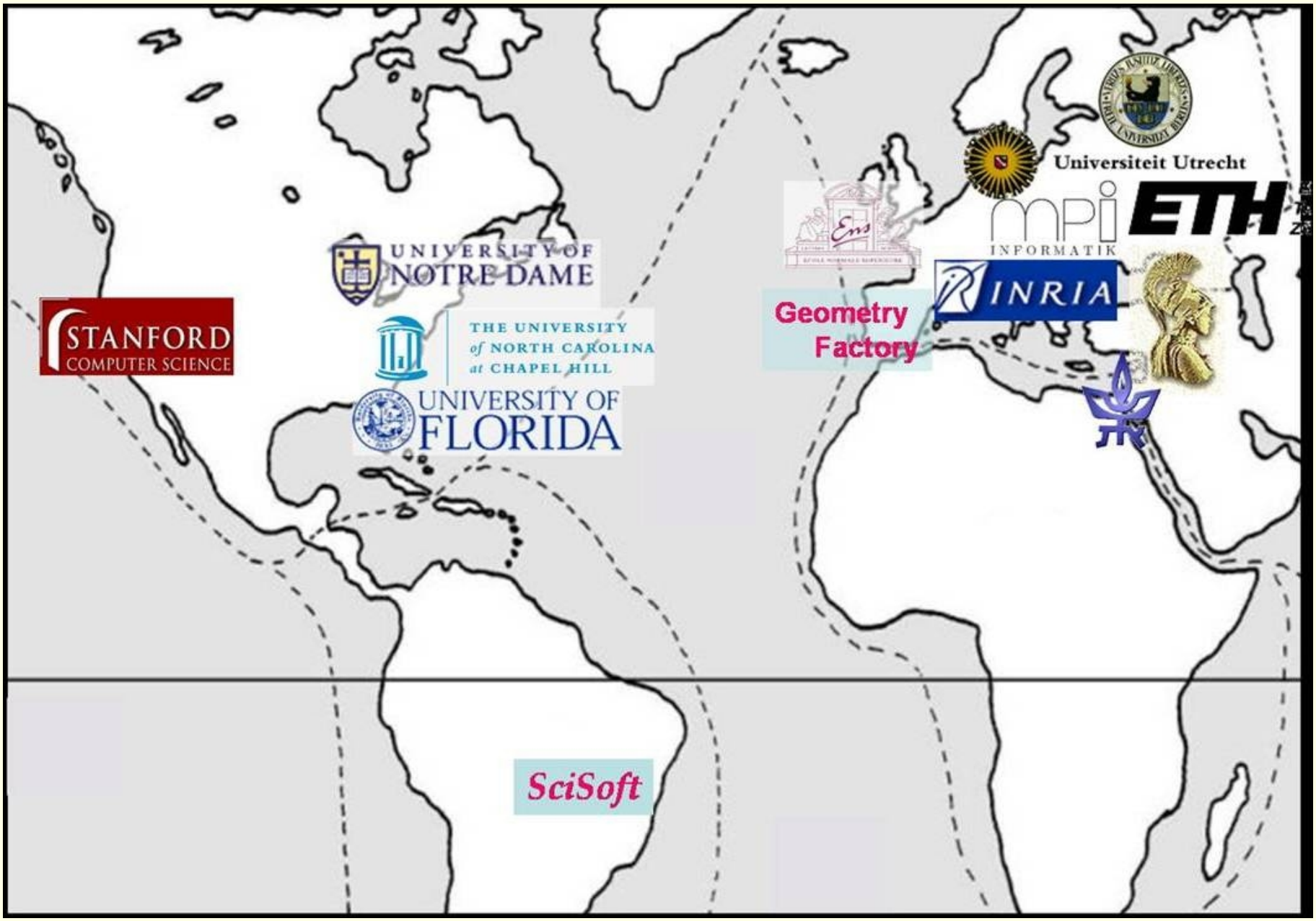
# Credit

## Contributors keep their identity

- up to 3.0.1: **names of authors** mentioned in the Preface
- 3.1 (soon) Names of authors appear at the beginning of each chapter. Section on history of the package at the end of each chapter
- CGAL developers listed on the "People" web page
- Authors publish **papers** (conferences, journals) on their packages
- **Copyright** kept by the institution of the authors






Universiteit Utrecht
ETH
INFORMATIK
INRIA
Geometry
Factory
UNIVERSITY OF
NOTRE DAME
STANFORD
COMPUTER SCIENCE
THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL
UNIVERSITY OF
FLORIDA
SciSoft

# 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, The 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

## 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, The University of Texas at Austin.

Xiangmin Jiao, University of Illinois at Urbana-Champaign.
(Surface Mesh Overlay)

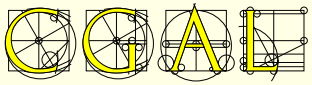
Peter Coveney and co-workers, University of London.

. . .

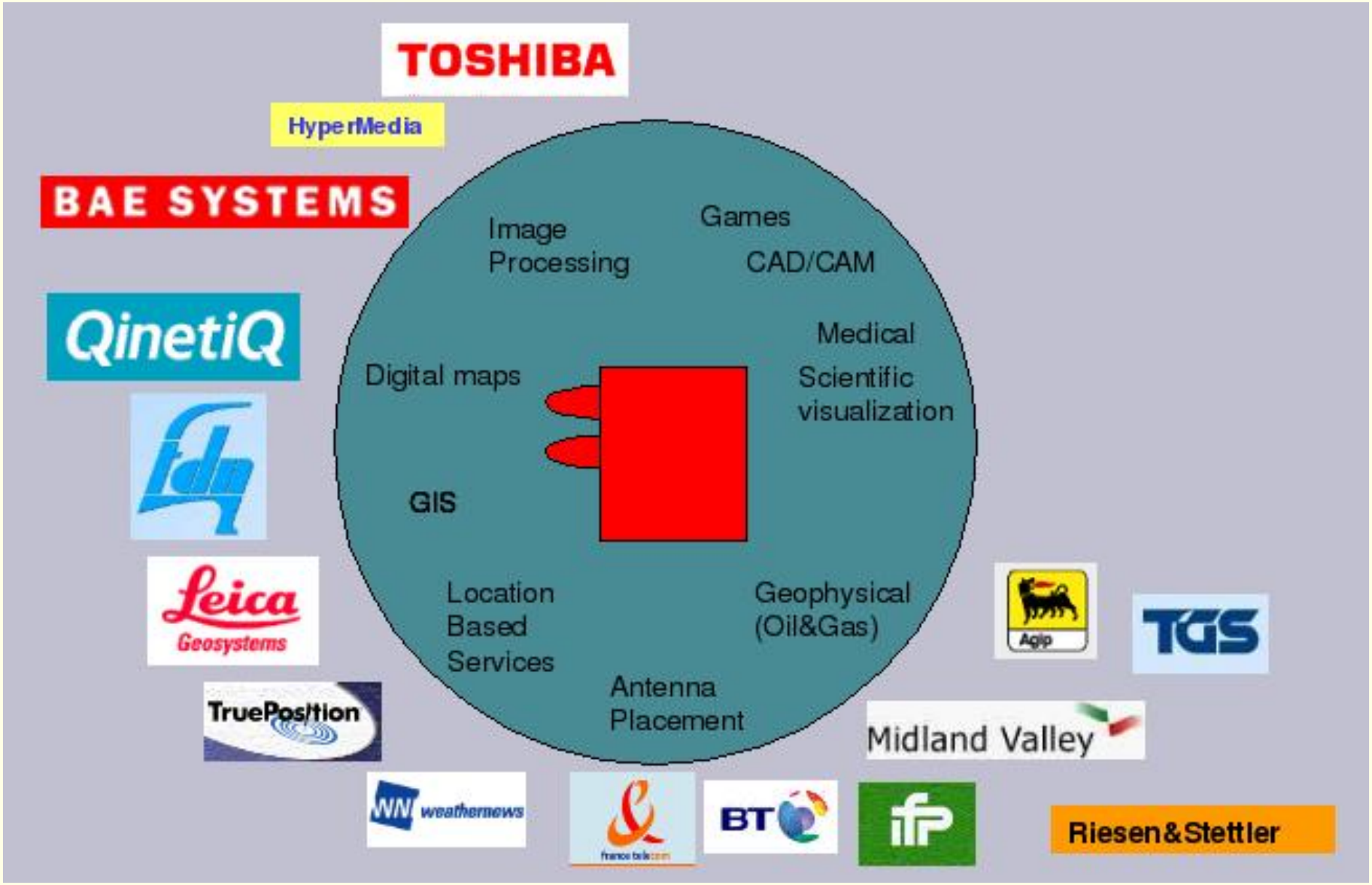
## Teaching

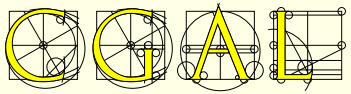
- Leo Guibas, Siu Wing Cheng, . . .





# Commercial customers of Geometry Factory



# Structure of 

**Basic Library**

Algorithms and Data Structures

**Kernel**

Geometric objects
Geometric operations

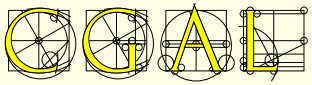
**core library**

configurations, assertions, ...

**Support Library**

Visualization
File I/O
NumberTypes
Generators
...





# The GOAL Kernel

# In the kernel

Elementary geometric objects

Elementary computations on them

**Primitives**
**2D, 3D, dD**
- Point
- Vector
- Triangle
- Iso_rectangle
- Circle

. . .

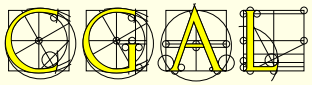
**Predicates**
- comparison
- Orientation
- InSphere

. . .

**Constructions**
- intersection
- squared distance

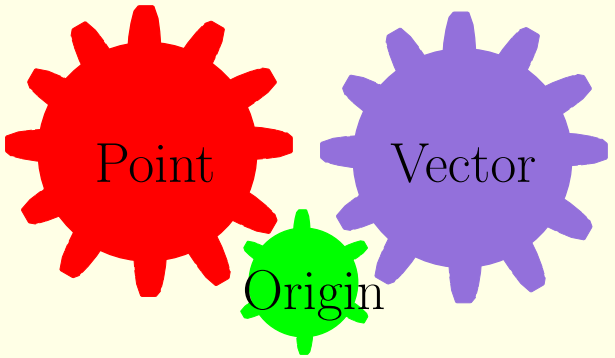
. . .





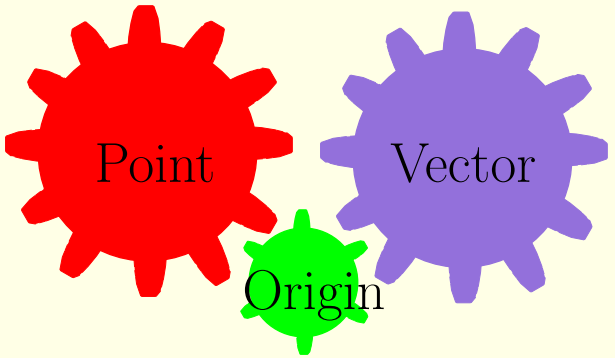
# Affine geometry

Point - Origin $\longrightarrow$ Vector
Point - Point $\longrightarrow$ Vector
Point + Vector $\longrightarrow$ Point



# Affine geometry

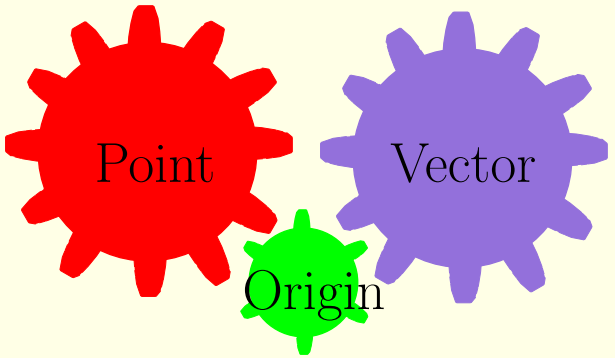
Point - Origin $\longrightarrow$ Vector
Point - Point $\longrightarrow$ Vector
Point + Vector $\longrightarrow$ Point




Point + Point **illegal**

# Affine geometry

Point - Origin $\longrightarrow$ Vector
Point - Point $\longrightarrow$ Vector
Point + Vector $\longrightarrow$ Point




Point + Point **illegal**

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

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

# 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

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

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

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

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

# 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

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

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

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

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

## Field operations

## Ring operations

# C++ Templates

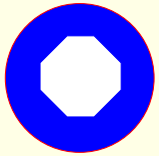
CGAL::Cartesian< FT >
CGAL::Homogeneous< RT >

(CGAL::Simple_Cartesian)
(CGAL::Simple_Homogeneous)

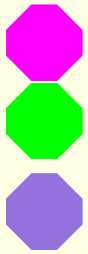
# C++ Templates

`CGAL::Cartesian< FT >`
`CGAL::Homogeneous< RT >`

`(CGAL::Simple_Cartesian)`
`(CGAL::Simple_Homogeneous)`



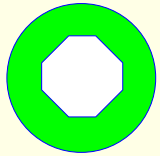
Cartesian Kernels : Field type



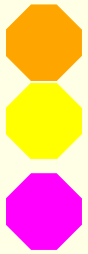
`double`

`Quotient<Gmpz>`

`leda_real`



Homogeneous Kernels : Ring type



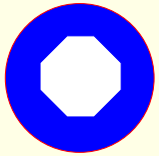
`int`

`Gmpz`

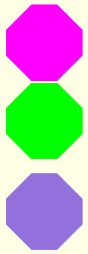
`double`

# C++ Templates

`CGAL::Cartesian< FT >` (`CGAL::Simple_Cartesian`)
`CGAL::Homogeneous< RT >` (`CGAL::Simple_Homogeneous`)



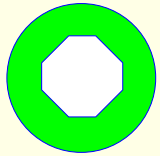
Cartesian Kernels : Field type



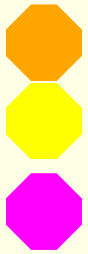
`double`

`Quotient<Gmpz>`

`leda_real`



Homogeneous Kernels : Ring type



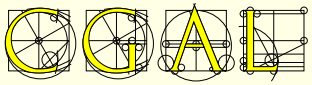
`int`

`Gmpz`

`double`

$\longrightarrow$ **Flexibility**

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



# Numerical Issues

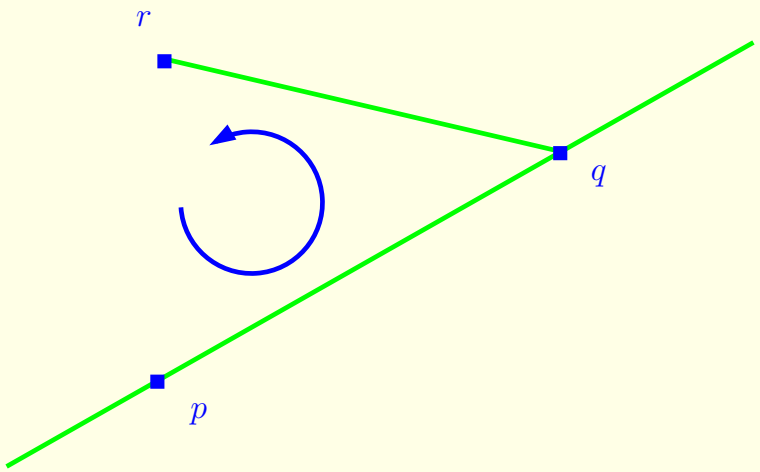
```
typedef CGAL::Cartesian<NT> Kernel;
NT sqrt2 = sqrt( NT(2) );

Kernel::Point_2 p(0,0), q(sqrt2,sqrt2);
Kernel::Circle_2 C(p,2);

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

**OK if NT gives exact `sqrt`**
**assertion violation otherwise**

# Orientation of 2D points




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

$$p = (0.5 + x.u,\ 0.5 + y.u)$$
$$0 \leq\ x, y\ < 256,\ \ u = 2^{-53}$$
$$q = (12, 12)$$
$$r = (24, 24)$$

$p = (0.5 + x.u,\ 0.5 + y.u)$
$0 \leq\ x, y\ < 256,\ \ u = 2^{-53}$
$q = (12, 12)$
$r = (24, 24)$

$orientation(p, q, r)$

evaluated with `double`

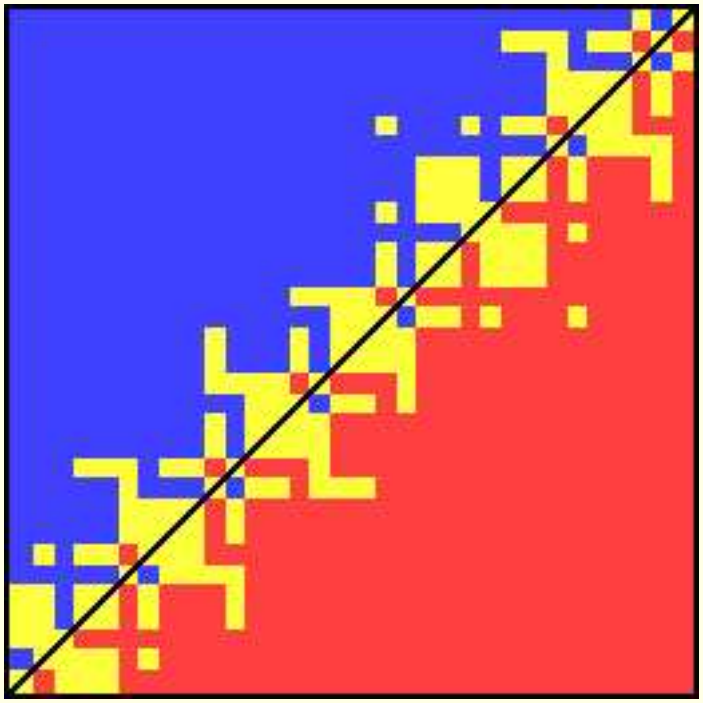
$p = (0.5 + x.u,\ 0.5 + y.u)$
$0 \leq\ x, y\ < 256,\ \ u = 2^{-53}$
$q = (12, 12)$
$r = (24, 24)$

$orientation(p, q, r)$

evaluated with `double`

256 x 256 pixel image

$> 0$, $= 0$, $< 0$



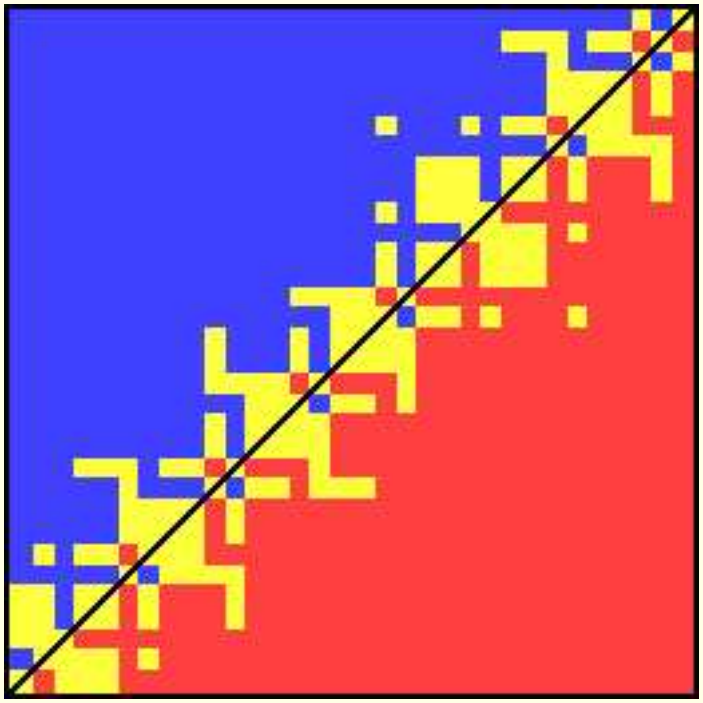
$p = (0.5 + x.u,\ 0.5 + y.u)$
$0 \leq\ x, y\ < 256,\ \ u = 2^{-53}$
$q = (12, 12)$
$r = (24, 24)$

$orientation(p, q, r)$

evaluated with `double`

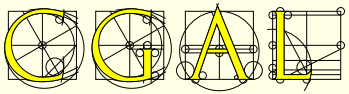
256 x 256 pixel image

$> 0$, $= 0$, $< 0$



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

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

# Numerical Robustness in 

imprecise numerical evaluations

$\longrightarrow$ non-robustness

combinatorial result

imprecise numerical evaluations

$\longrightarrow$ non-robustness

combinatorial result

**Exact Geometric Computation**

imprecise numerical evaluations

combinatorial result

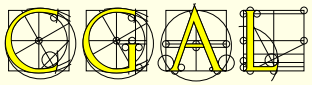
$\longrightarrow$ non-robustness

**Exact Geometric Computation**

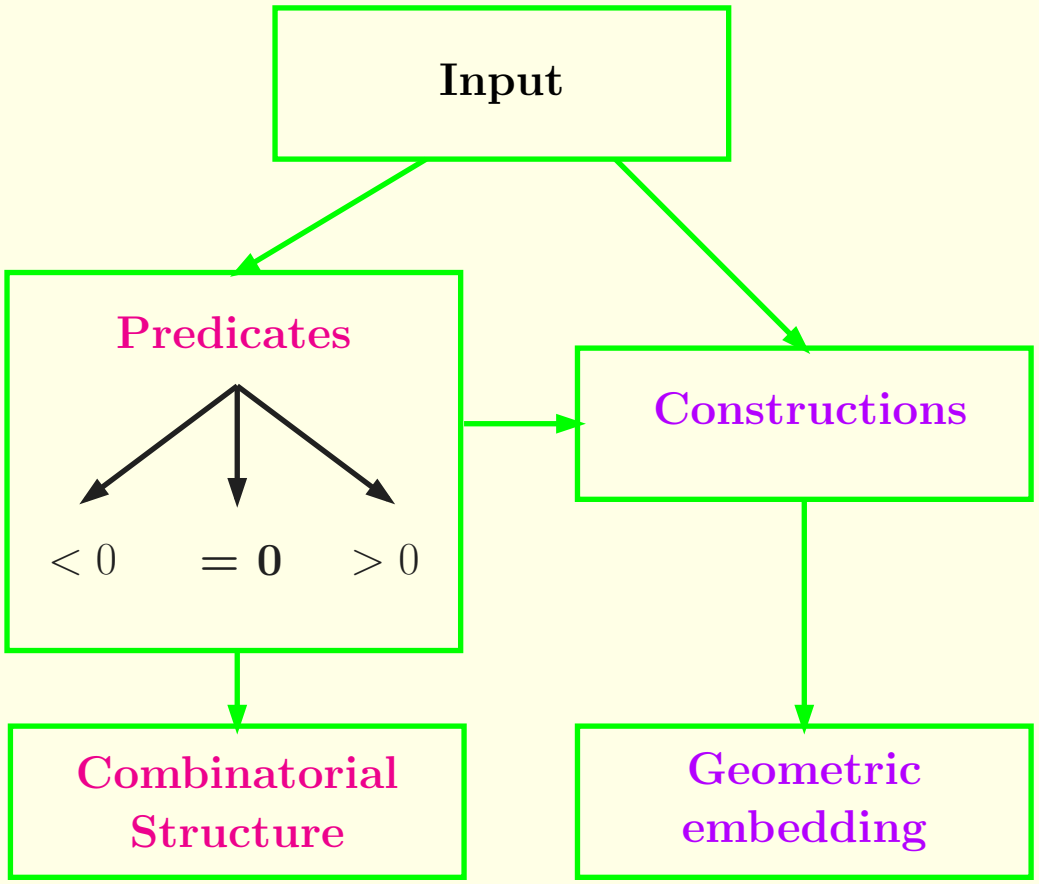
$\neq$

**exact arithmetics**

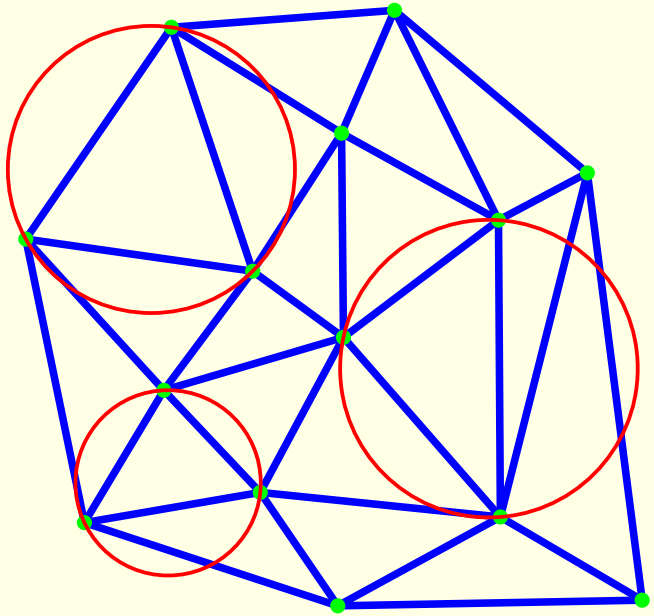




# Predicates and Constructions

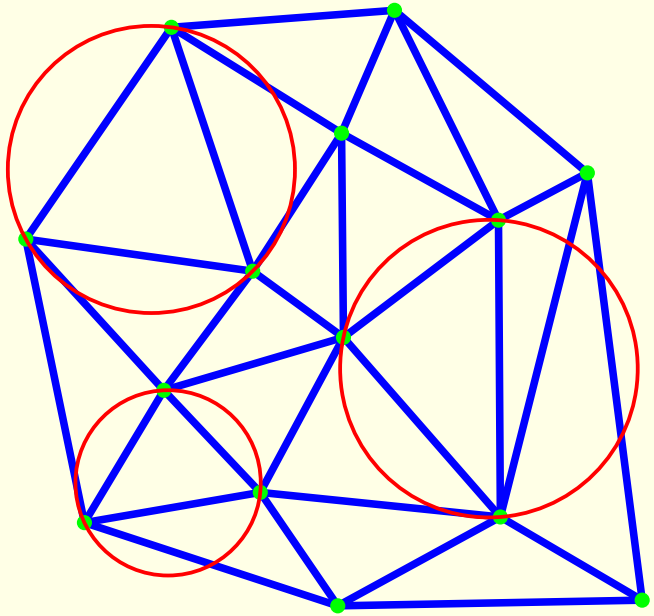


## Delaunay triangulation



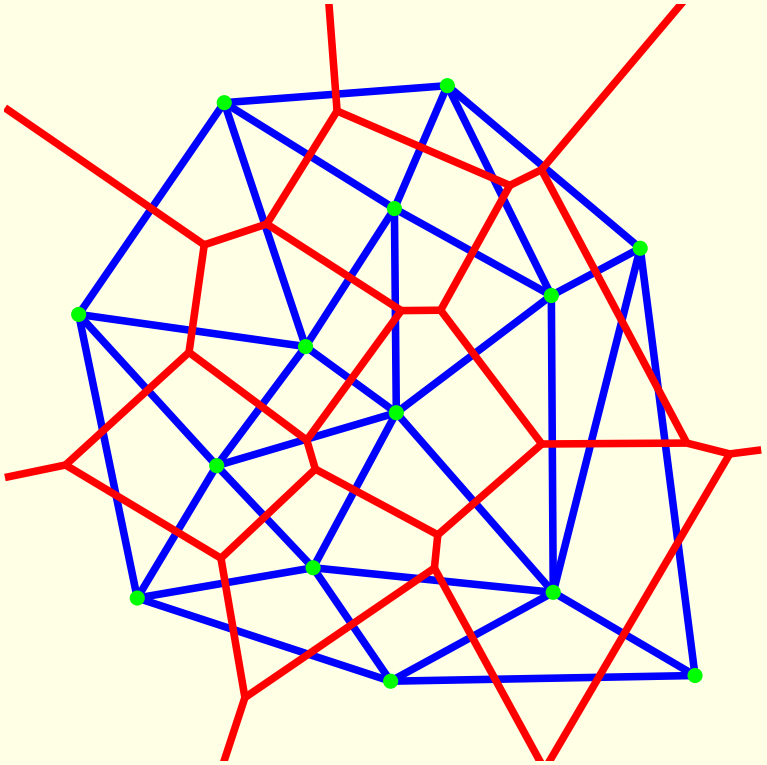
only **predicates** are used
*orientation*, *in_sphere*

## Delaunay triangulation



only **predicates** are used
*orientation*, *in_sphere*

## Voronoi diagram



**constructions** are needed
*circumcenter*

# Arithmetic tools

- **Multiprecision integers**

Exact evaluation of signs / values of polynomial expressions with integer coefficients

**CGAL::MP_Float, GMP::mpz_t, LEDA::integer, ...**

# 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, ...**

# 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$<\cdot>$, GMP::mpq_t, LEDA::rational, ...**

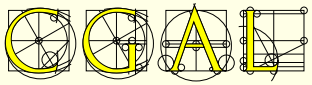
# Arithmetic tools

• **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, ...**

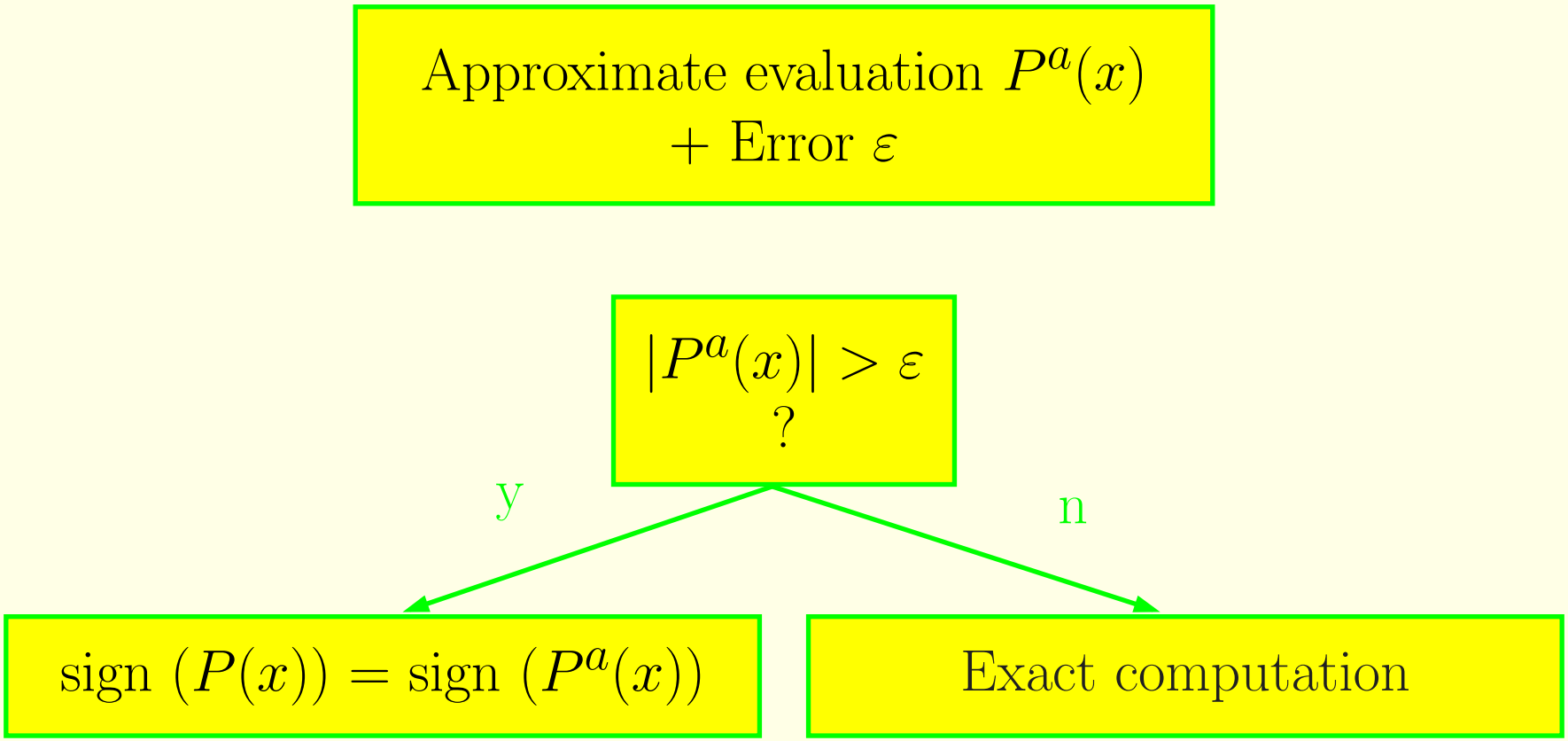
• **Multiprecision rationals**
Exact evaluation of signs / values of rational expressions
**CGAL::Quotient$<\cdot>$, GMP::mpq_t, LEDA::rational, ...**

• **Algebraic numbers**
Exact comparison of roots of polynomials
**LEDA::real, Core::Expr** (work in progress in CGAL)



# Filtering Predicates

**sign** $(P(x))$ **?**



## Static filtering

Error bound precomputed faster

## Static filtering

Error bound precomputed faster

## Dynamic filtering

Interval arithmetic more precise

Number types: **CGAL::Interval_nt, MPFR/MPFI, boost::interval**

## Static filtering

Error bound precomputed — faster

## Dynamic filtering

Interval arithmetic — more precise

Number types: **CGAL::Interval_nt, MPFR/MPFI, boost::interval**

**CGAL::Filtered_kernel** $<$ **K** $>$ kernel wrapper — [Pion]

Replaces predicates of **K** by filtered and exact predicates.
( exact predicates computed with MP_Float )

## Static filtering

Error bound precomputed faster

## Dynamic filtering

Interval arithmetic more precise

Number types: **CGAL::Interval_nt, MPFR/MPFI, boost::interval**

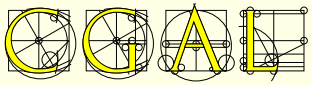
**CGAL::Filtered_kernel** $<$ **K** $>$ kernel wrapper [Pion]

Replaces predicates of **K** by filtered and exact predicates.
( exact predicates computed with MP_Float )

Dynamic only in CGAL 3.0 (Static filters exist but undocumented)
Static + Dynamic in CGAL 3.1

$\longrightarrow$ 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`

# Filtering Constructions

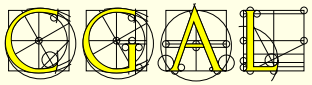
Number type **CGAL::Lazy_exact_nt** < **Exact_NT** > [Pion]

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





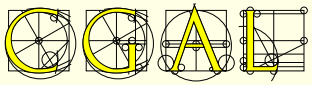
# Predefined kernels

**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**
CGAL-3.1-I-124

Pentium-M 1.7 GHz, 1GB
g++ 3.3.2, -O2 -DNDEBUG

1.000.000 random points

| | |
|---|---|
| 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** |

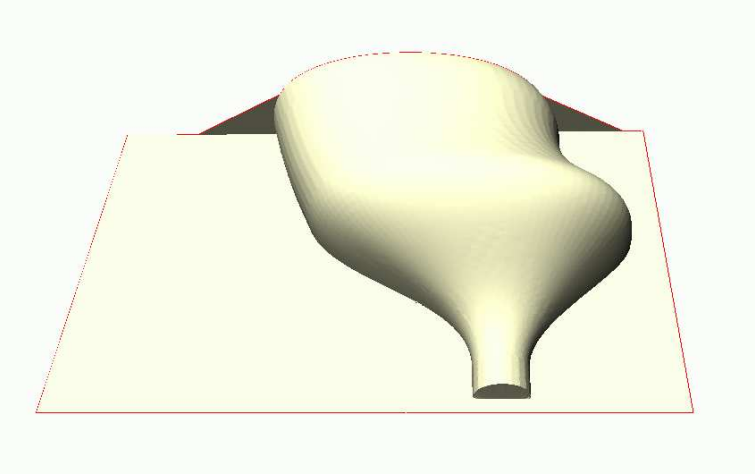
# Efficiency

**3D Delaunay triangulation**
CGAL-3.1-I-124

Pentium-M 1.7 GHz, 1GB
g++ 3.3.2, -O2 -DNDEBUG

1.000.000 random points

| | |
|---|---|
| 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** |



49.787 points (Dassault Systèmes)

| | |
|---|---|
| double | **loop !** |
| exact and filtered | $< 8$ sec |

## Robustness of Delaunay triangulations

Kernel and arithmetics $\longrightarrow$ Numerical robustness

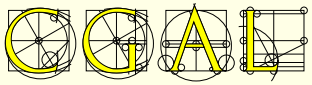
## Robustness of Delaunay triangulations

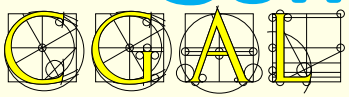
Kernel and arithmetics $\longrightarrow$ Numerical robustness

Algorithms $\longrightarrow$ explicit treatment of **degenerate cases**

Symbolic perturbation for 3D dynamic Delaunay triangulations
[Devillers Teillaud SODA'03]





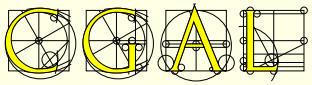
# Contents of the  Basic Library

# Convex Hull

[MPI]

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





# Triangulations and related

[INRIA]

- 2D/3D Triangle/Tetrahedron based data-structure

# Triangulations and related

[INRIA]

- 2D/3D Triangle/Tetrahedron based data-structure

- Fully dynamic 2D/3D Delaunay triangulation
  Delaunay hierarchy [Devillers '98 '02]

- 2D/3D Regular Triangulations
  (3D: dynamic in 3.1)
- 2D Constrained Delaunay Triangulation

# Triangulations and related

[INRIA]

- 2D/3D Triangle/Tetrahedron based data-structure

- Fully dynamic 2D/3D Delaunay triangulation
  Delaunay hierarchy [Devillers '98 '02]

- 2D/3D Regular Triangulations
  (3D: dynamic in 3.1)
- 2D Constrained Delaunay Triangulation

- 2D Apollonius diagram
- 2D Segment Voronoi Diagram
  (3.1 [Karavelas])
- 2D Meshes (3.1)

# Triangulations and related

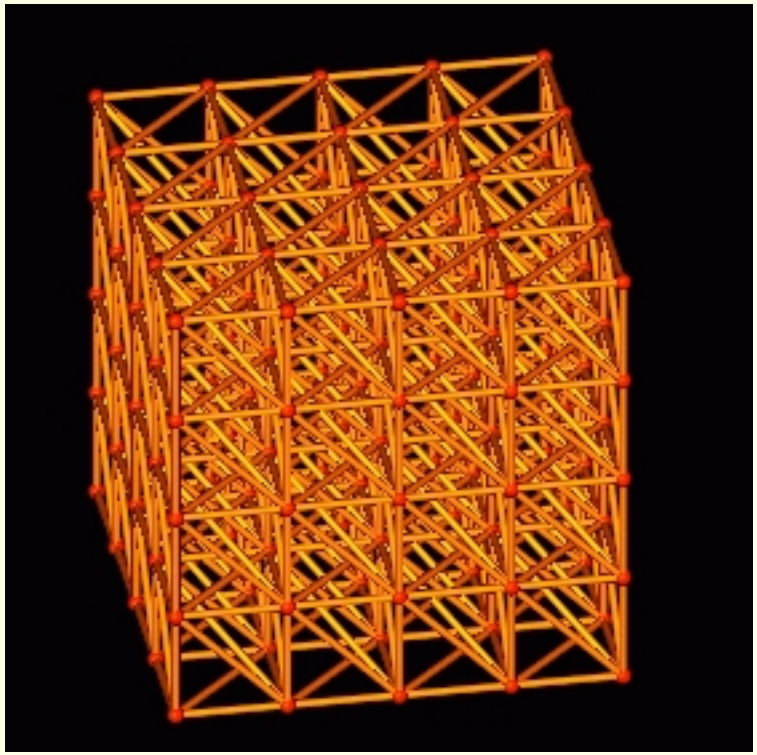
[INRIA]

- 2D/3D Triangle/Tetrahedron based data-structure

- Fully dynamic 2D/3D Delaunay triangulation
  Delaunay hierarchy [Devillers '98 '02]

- 2D/3D Regular Triangulations
  (3D: dynamic in 3.1)
- 2D Constrained Delaunay Triangulation

- 2D Apollonius diagram
- 2D Segment Voronoi Diagram
  (3.1 [Karavelas])
- 2D Meshes (3.1)



# Polyhedra

[MPI]

- Half-edge data-structure

# Polyhedra

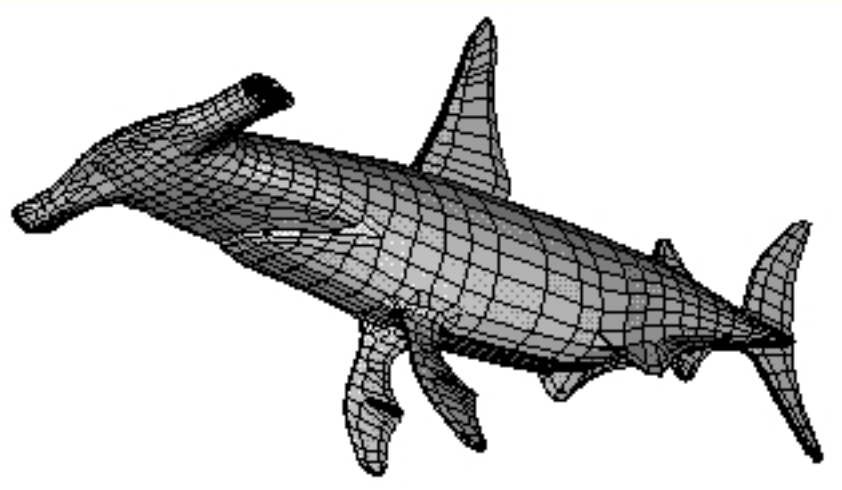
[MPI]

- Half-edge data-structure
- Polyhedral surface
  (orientable 2-manifold with boundary)

# Polyhedra

[MPI]

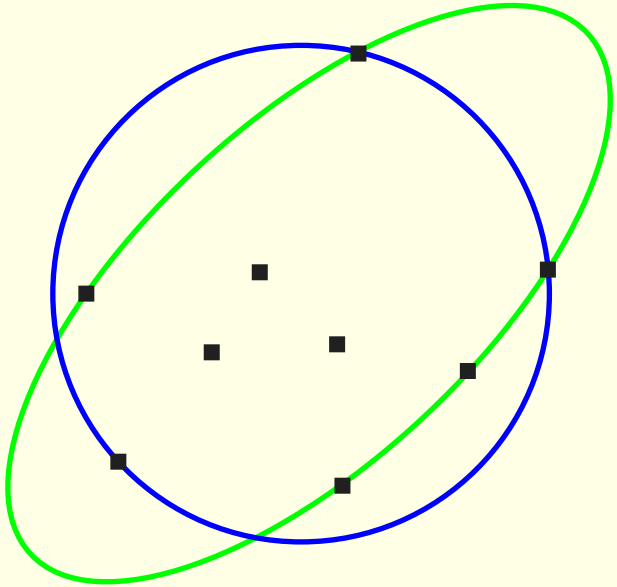
- Half-edge data-structure
- Polyhedral surface
  (orientable 2-manifold with boundary)
- 2D Nef polygons
- 3D Nef polyhedra (3.1)



# Geometric Optimization

[ETH]

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

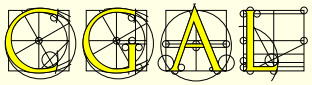


# Arrangements

[Tel-Aviv]

- Line segments or polylines
- Conic arcs with Leda or Core



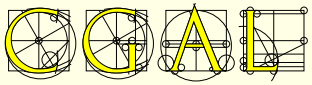


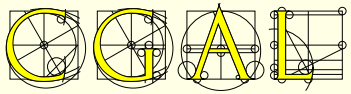
# Search Structures

Arbitrary dimension

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





# Flexibility in the  Basic Library

# "Traits" classes

convex_hull_2<InputIterator, OutputIterator, **Traits**>
Polygon_2<**Traits**, Container>
Polyhedron_3<**Traits**, HDS>
Triangulation_3<**Traits**, TDS>
Min_circle_2<**Traits**>
Range_tree_k<**Traits**>
. . .

# "Traits" classes

convex_hull_2<InputIterator, OutputIterator, **Traits**>
Polygon_2<**Traits**, Container>
Polyhedron_3<**Traits**, HDS>
Triangulation_3<**Traits**, TDS>
Min_circle_2<**Traits**>
Range_tree_k<**Traits**>
. . .

**Geometric traits classes** provide:
  Geometric objects + predicates + constructors

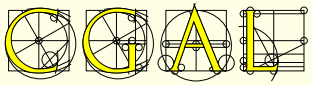
# "Traits" classes

convex_hull_2<InputIterator, OutputIterator, **Traits**>
Polygon_2<**Traits**, Container>
Polyhedron_3<**Traits**, HDS>
Triangulation_3<**Traits**, TDS>
Min_circle_2<**Traits**>
Range_tree_k<**Traits**>
. . .

**Geometric traits classes** provide:
Geometric objects + predicates + constructors

- The **Kernel** can be used as a traits class for several algorithms
- Otherwise: **Default traits classes** provided
- The **user** can plug his own traits class





# Playing with traits classes

## Delaunay Triangulation

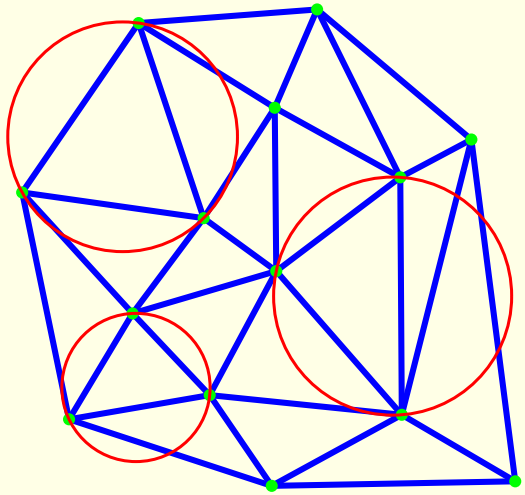
**Requirements** for a traits class:

- Point
- orientation test, in_circle test

# Playing with traits classes

## Delaunay Triangulation

**Requirements** for a traits class:
- Point
- orientation test, in_circle test

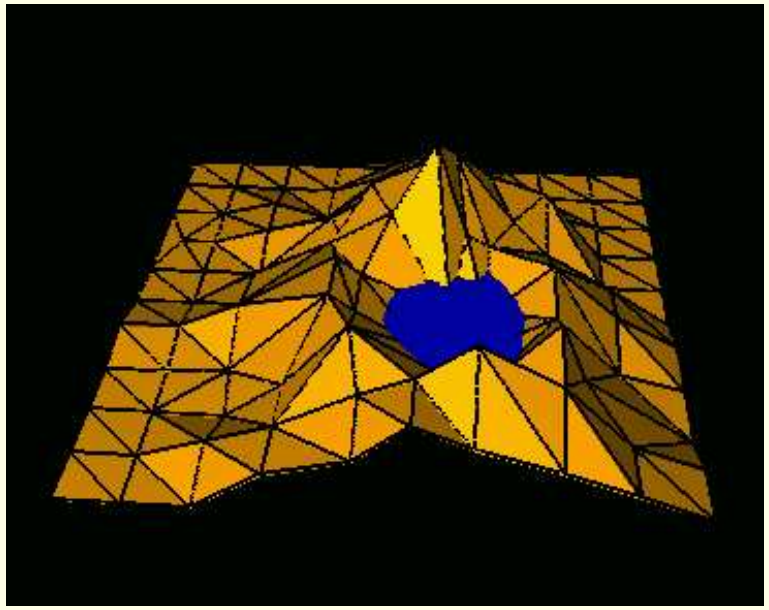


```
typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Delaunay_triangulation_2< K > Delaunay;
```

- 3D points: coordinates $(\mathbf{x}, \mathbf{y}, \mathbf{z})$
- orientation, in_circle: on $\mathbf{x}$ and $\mathbf{y}$ coordinates

- 3D points: coordinates $(\mathbf{x}, \mathbf{y}, \mathbf{z})$
- orientation, in_circle: on $\mathbf{x}$ and $\mathbf{y}$ coordinates

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

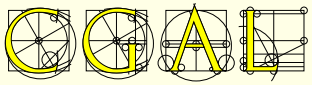


# More flexibility

The user can add information in vertices and cells

. . .





# Work in Progress

# Kinetic Data Structures

[Russel Karavelas]

## Kinetic Data Structures

[Russel Karavelas]

## Persistent Homology

[Kettner Zomorodian]

**Kinetic Data Structures**
[Russel Karavelas]

**Persistent Homology**
[Kettner Zomorodian]

**Surface reconstruction**
[Oudot Rey]

## Kinetic Data Structures
[Russel Karavelas]

## Persistent Homology
[Kettner Zomorodian]

## Surface reconstruction
[Oudot Rey]

## 3D Meshes
[Rineau Yvinec]

## Kinetic Data Structures
[Russel Karavelas]

## Persistent Homology
[Kettner Zomorodian]

## Surface reconstruction
[Oudot Rey]

## 3D Meshes
[Rineau Yvinec]

## 2D Interpolation
[Flötotto]

## Kinetic Data Structures
[Russel Karavelas]

## Persistent Homology
[Kettner Zomorodian]

## Surface reconstruction
[Oudot Rey]

## 3D Meshes
[Rineau Yvinec]

## 2D Interpolation
[Flötotto]

## Curved Kernel
Extension of the CGAL kernel
Algebraic issues
[Emiris Kakargias Pion Tsigaridas Teillaud SoCG'04]

. . .