Geometry made practical

www.cgal.org

Monique Teillaud

http://www.loria.fr/~teillaud/

INRIA Nancy - Grand Est, LORIA

Computer Science meets Mathematics
Luxembourg - Feb 18, 2016

CGAL, the Computational Geometry Algorithms Library

- The CGAL Open Source Project and the CGAL Library
- Robustness
- Triangulations
- Non-Euclidean spaces

Part I

The CGAL Open Source Project and the CGAL library

Goals

- Promote the research in Computational Geometry (CG)

- *“make the large body of geometric algorithms developed in the field of computational geometry available for industrial applications”*

$\Rightarrow$ **robust programs**

- Reward structure for implementations in academia

History

- Development started in 1995
- Academic project

History

- Development started in 1995
- Academic project

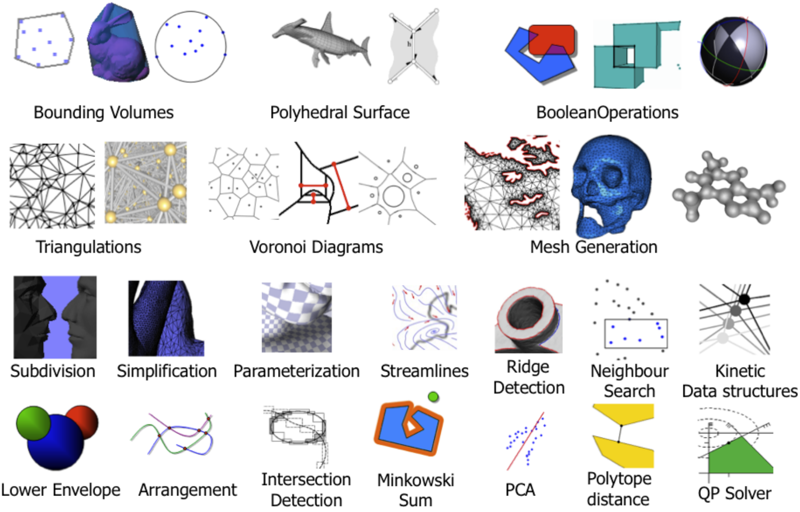
- January, 2003: creation of **Geometry Factory**
INRIA startup
sells commercial licenses, support, customized developments

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

- current: CGAL 4.7 (October 2015)

Contents

> **80 chapters** in the manual



Technical

- 500,000 lines of **C++** code
  genericity, flexibility through templates
- multi-platforms
  Linux, MacOS, Windows
  g++, VC++, clang,. . .

Technical

- 500,000 lines of **C++** code
  genericity, flexibility through templates

- multi-platforms
  Linux, MacOS, Windows
  g++, VC++, clang,...

- License
  - a few basic packages under **LGPL**
  - most packages under **GPLv3+**
    - free use for Open Source code
  - **commercial** license through GEOMETRY FACTORY

How to get CGAL?

- release cycle: 6 months soon 4.8
  - from **github** ($>$ 1,000 downloads per month)
  - included in **Linux** distributions (Debian, etc)
  - available through **macport, brew**
- master branch public in **github**
- 2d and 3d triangulation packages integrated in **Matlab**
- **CGAL-bindings** (implemented with SWIG)
  CGAL triangulations, meshes, etc, in **Java** or **Python**

Users

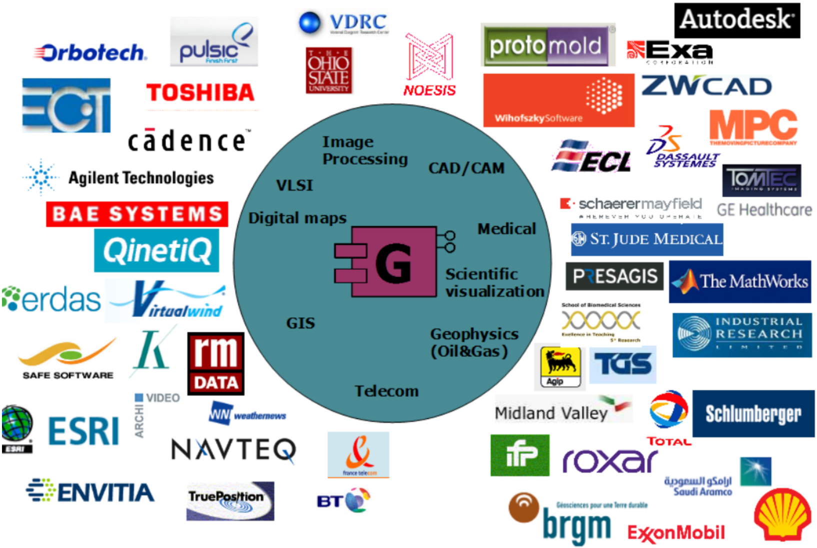
List of identified users in various fields

- 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...

Some Commercial Users

(2012)



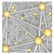
CGAL welcomes new contributions

Contributors **keep their identity**:

- Listed as **authors** in the manual

3D Triangulations

Sylvain Pion and Monique Teillaud



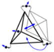
This package allows to build and handle triangulations for point sets in three dimensions. Any CGAL triangulation covers the convex hull of its vertices. Triangulations are build incrementally and can be modified by insertion or removal of vertices. They offer point location facilities.

The package provides plain triangulation (whose faces depends on the insertion order of the vertices) and Delaunay triangulations. Regular triangulations are also provided for sets of weighted points. Delaunay and regular triangulations offer nearest neighbor queries and primitives to build the dual Voronoi and power diagrams.

Introduced in: CGAL 2.1
License: QPL
Citation Entry
User Manual Reference Manual

3D Triangulation Data Structure

Sylvain Pion and Monique Teillaud

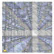


This package provides a data structure to store a three-dimensional triangulation that has the topology of a three-dimensional sphere. The package acts as a container for the vertices and cells of the triangulation and provides basic combinatorial operations on the triangulation.

Introduced in: CGAL 2.1
License: QPL
Citation Entry
User Manual Reference Manual

3D Periodic Triangulations

Manuel Caroli and Monique Teillaud



This package allows to build and handle triangulations of point sets in the three dimensional flat torus. Triangulations are built incrementally and can be modified by insertion or removal of vertices. They offer point location facilities.

The package provides Delaunay triangulations and offers nearest neighbor queries and primitives to build the dual Voronoi diagrams.

Introduced in: CGAL 3.5
Depends on: 3D Triangulation and 3D Triangulation Data Structure
License: QPL
Citation Entry
Demo: Periodic Delaunay Triangulation
User Manual Reference Manual

- Mentioned on the “People” **web** page

- **Copyright kept** by the [institution of the] authors

CGAL welcomes new contributions

- **Review** coordinated by the Editorial Board
- **Test-suite** must run on all supported platforms

Advice: contact us **early**

Part II

Robustness

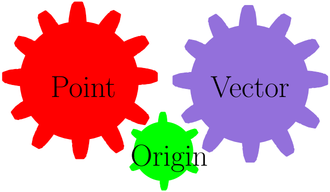
The CGAL Kernels

- 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

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

Homogeneous representation

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

- ex: Intersection of two lines -

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

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

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

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

Field operations

Ring operations

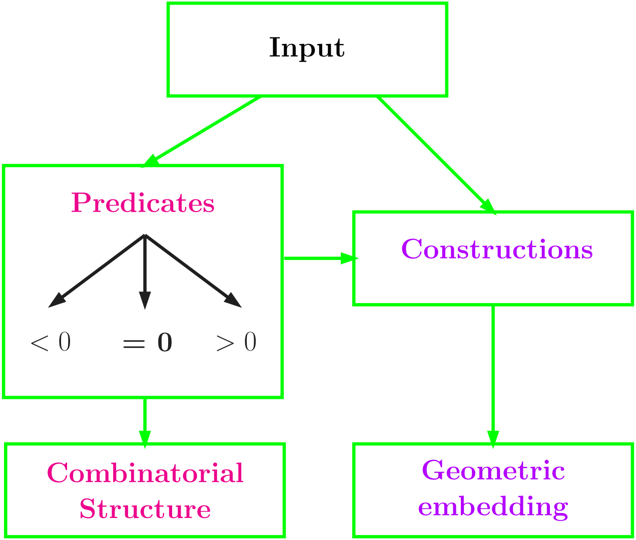
Kernels and number types

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

⟶ **Flexibility**

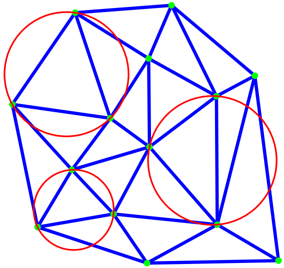
```
typedef double                          NumberType;
typedef Cartesian< NumberType >         Kernel;
typedef Kernel::Point_2                 Point;
```

Predicates and Constructions



Predicates and Constructions

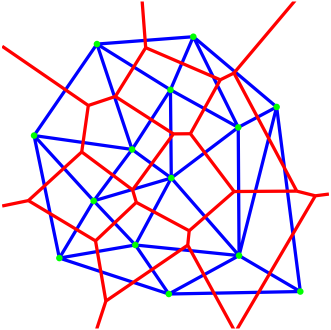
Delaunay triangulation



only **predicates** are used

orientation, in_sphere

Voronoi diagram



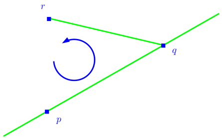
constructions are needed

circumcenter

Numerical robustness issues

Many 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

Many predicates = **signs of polynomial expressions**

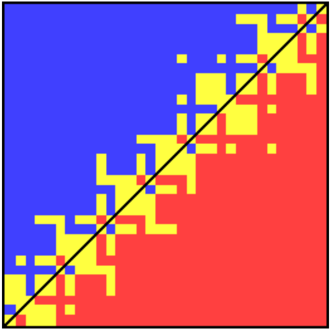
Ex: **Orientation of 2D points**

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

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

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

ensures that **predicates** are correctly evaluated
= geometric decisions are correct

$\Longrightarrow$ combinatorial structure is correct

Numerical robustness issues

Speed and exactness through

Exact Geometric Computation

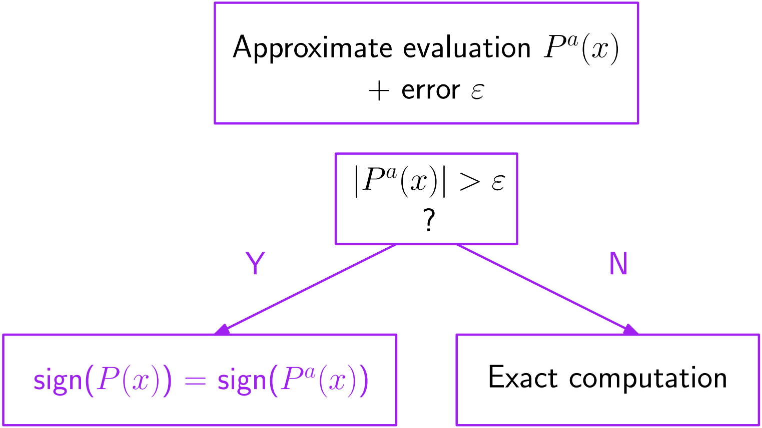
$\neq$

exact arithmetics

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

Filtering Predicates

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



Robustness issues

- Numerical issues: Exact Geometric Computation

+

- Degenerate cases. . . . . **explicitly handled**
  (**symbolic** perturbation techniques, etc)

The circular/spherical kernels

```
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);  // 2 = squared radius

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

OK if NT gives exact `sqrt`
assertion violation otherwise

The circular/spherical kernels

Circular/spherical kernels

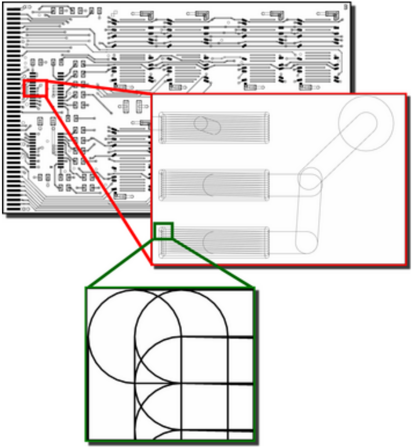
- solve needs for e.g. intersection of circles.
- **extend** the CGAL (linear) kernels

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**

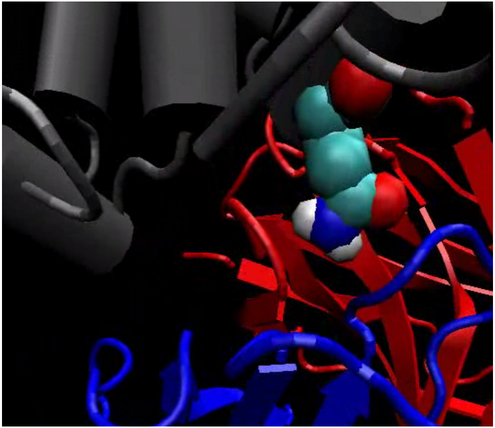
Application of the 2D circular kernel

Computation of arrangements
of 2D circular arcs and line segments



Application of the 3D spherical kernel

Computation of arrangements of 3D spheres



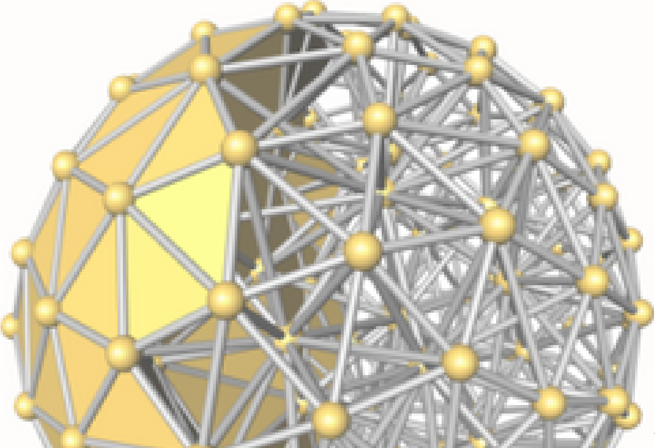
Part III

Triangulations

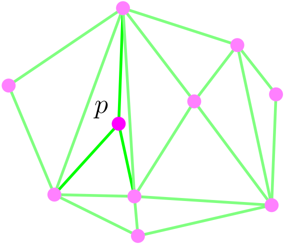
Definition

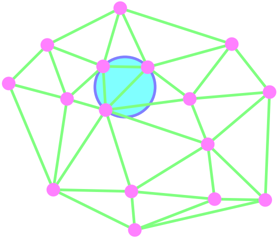
2D (dD) simplicial complex = set $\mathbb{K}$ of **0,1,2**,...,**d**-faces such that

- $\sigma \in \mathbb{K}, \tau \leq \sigma \Rightarrow \tau \in \mathbb{K}$
- $\sigma, \sigma' \in \mathbb{K} \Rightarrow \sigma \cap \sigma' \leq \sigma, \sigma'$



Various triangulations 2D, 3D, dD





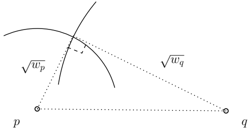
Basic triangulations

Delaunay triangulations

Weighted Delaunay triangulations (dual of power diagram)

power product between $p^{(w)}$ and $z^{(w)}$

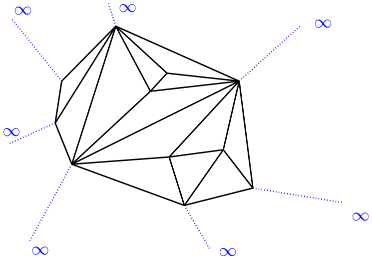
$$\Pi(p^{(w)}, z^{(w)}) = \|p - z\|^2 - w_p - w_z$$



Geometry vs. Combinatorics

Triangulation of a set of points = partition of the **convex hull** into simplices.

Addition of an **infinite vertex without coordinates**



(2D)

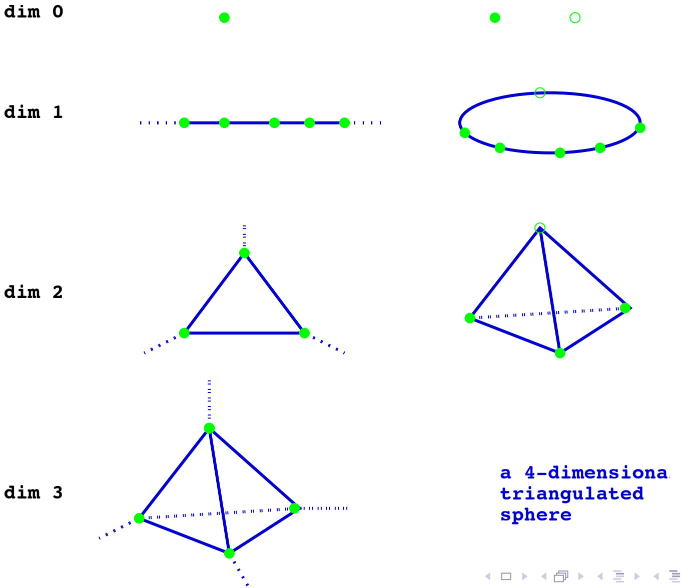
$\longrightarrow$ "triangulation" of the outside of the convex hull.
- Any cell is a "tetrahedron".
- Any facet is incident to two cells.

Triangulation of $\mathbb{R}^d$

$\simeq$

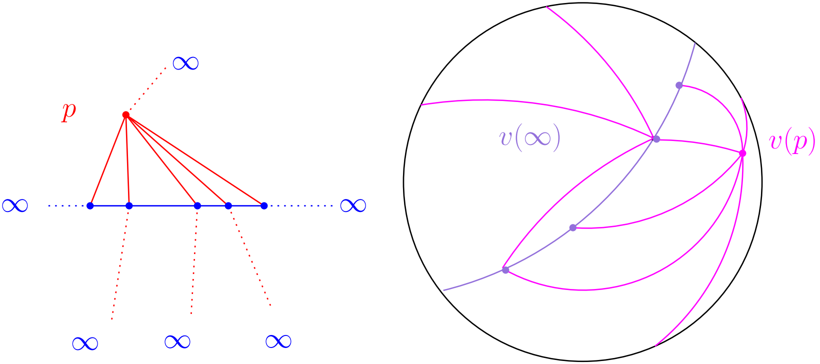
Triangulation of the topological **sphere** $\mathbb{S}^d$.

Dimensions



Dimensions

Adding a point outside the current affine hull:
From $d = 1$ to $d = 2$



Traits class

Triangulation_2$<$**Traits**, TDS$>$

Geometric traits classes provide:

Geometric objects + predicates + constructors

Flexibility:

- 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

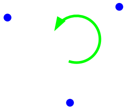
Traits class

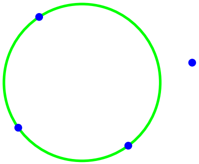
Generic algorithms

`Delaunay_Triangulation_2<Traits, TDS>`

Traits parameter provides:

- Point
- orientation test, in_circle test



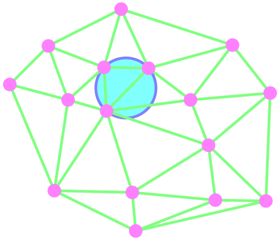


Traits class

2D Kernel used as traits class

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

- 2D points: coordinates $(\mathbf{x}, \mathbf{y})$
- orientation, in_circle

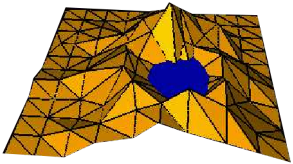


Traits class

Changing the traits class

```
typedef
   CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef
   CGAL::Projection_traits_xy_3< K > Traits;
typedef CGAL::Delaunay_triangulation_2< Traits > Terrain;
```

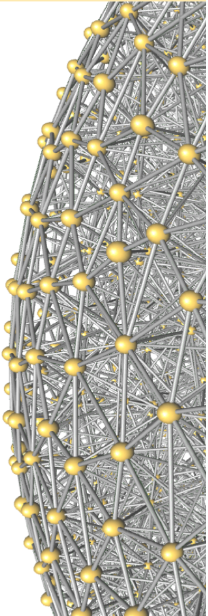
- 3D points: coordinates $(\mathbf{x}, \mathbf{y}, \mathbf{z})$
- orientation, in_circle:
  on **x** and **y** coordinates only



3D Delaunay Triangulations

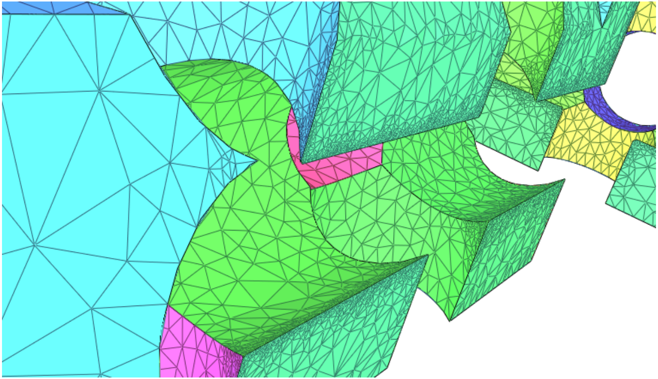
- fully dynamic (also weighted triangulations)
- fast: 1 M points $\simeq$ 10 sec ($\simeq$ 10 μsec /point)
- robust
- basis for 3D α-shapes and 3D meshes
- integrated in **Matlab** 2009
- recent: **multi-core**

demo



3D meshes

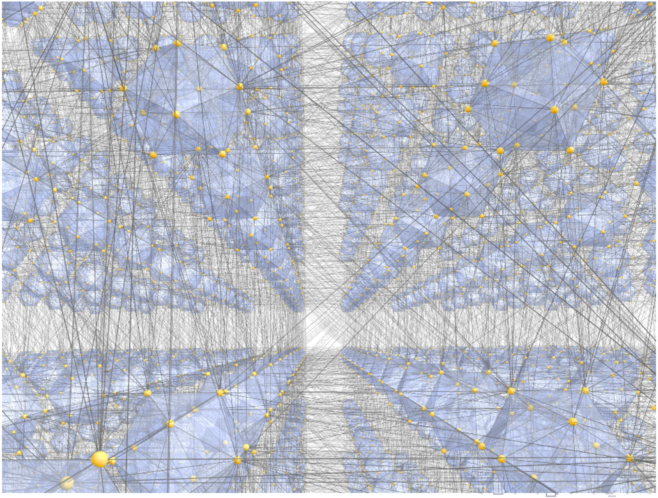
- Delaunay refinement



Non-Euclidean spaces

(cubic) **flat torus**

- 2D, 3D **periodic** triangulations demo

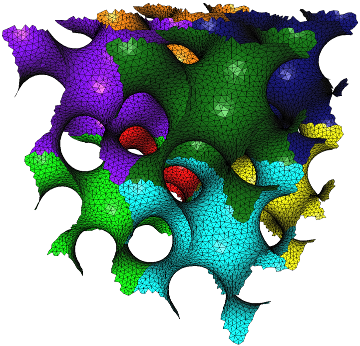


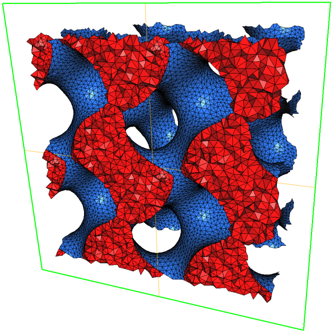
Non-Euclidean spaces

(cubic) **flat torus**

In the pipe. . .

- periodic meshes





Non-Euclidean spaces

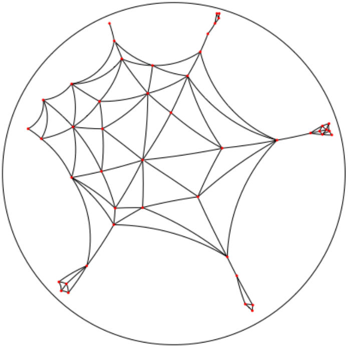
sphere

In the pipe. . .

- Delaunay triangulations

In the pipe...

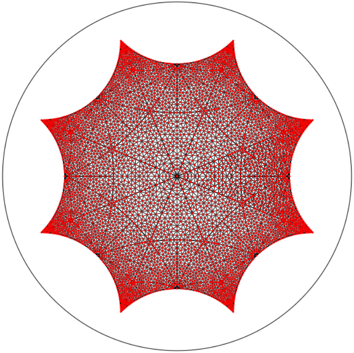
- Delaunay triangulations



demo

Research in progress

- Delaunay triangulation of the Bolza surface. . . ?



www.cgal.org

Thank you for your attention

Thanks to several students and CGAL *colleagues for some pictures*