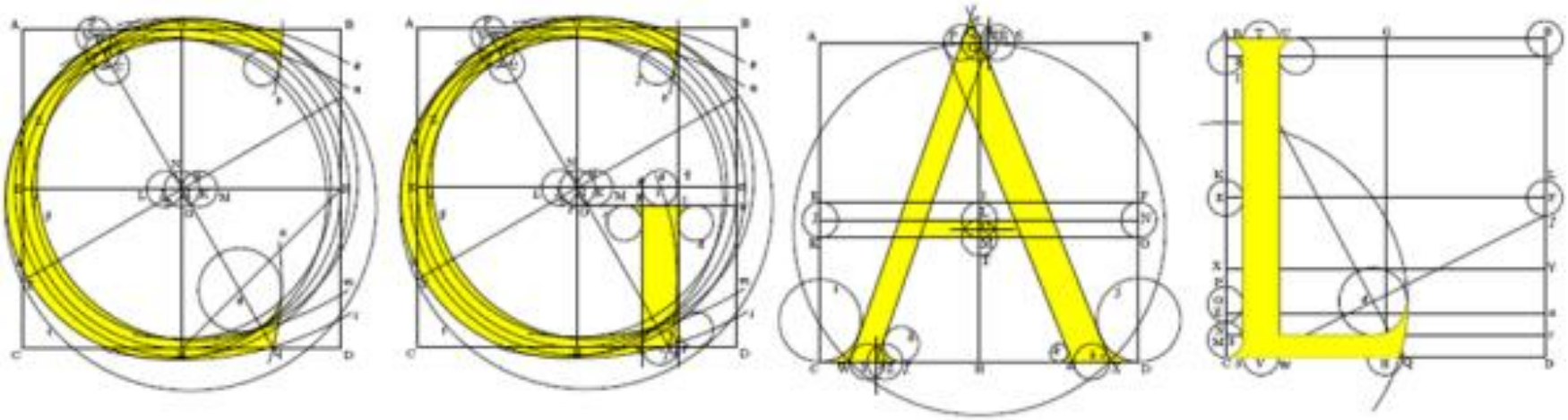


Computational Geometry Algorithms Library

Pierre Alliez

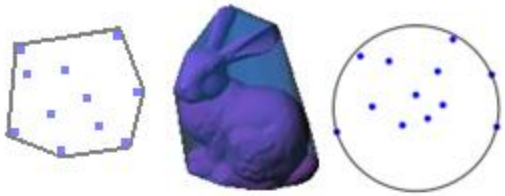
INRIA

Mission Statement

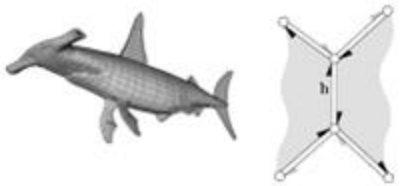
"Make the large body of geometric algorithms developed in the field of computational geometry available for industrial applications"

CGAL Project Proposal, 1996

Algorithms and Datastructures

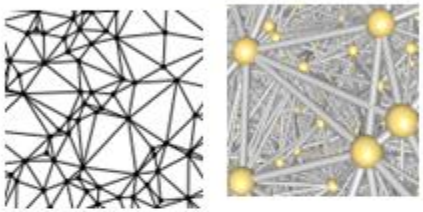


Bounding Volumes

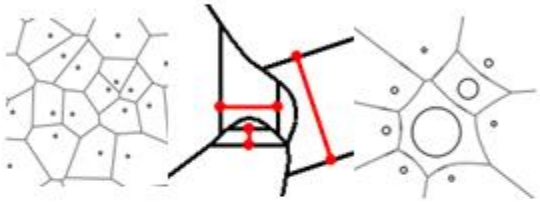


Polyhedral Surface

Boolean Operations



Triangulations



Voronoi Diagrams

Mesh Generation

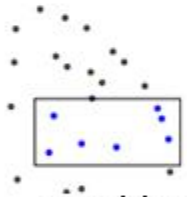
Subdivision

Simplification

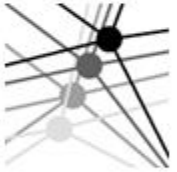
Parameterization

Streamlines

Ridge Detection



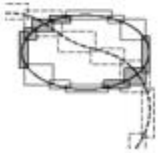
Neighbor Search



Kinetic Datastructures

Lower Envelope

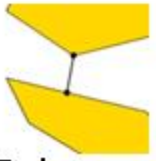
Arrangement



Intersection Detection

Minkowski Sum

PCA



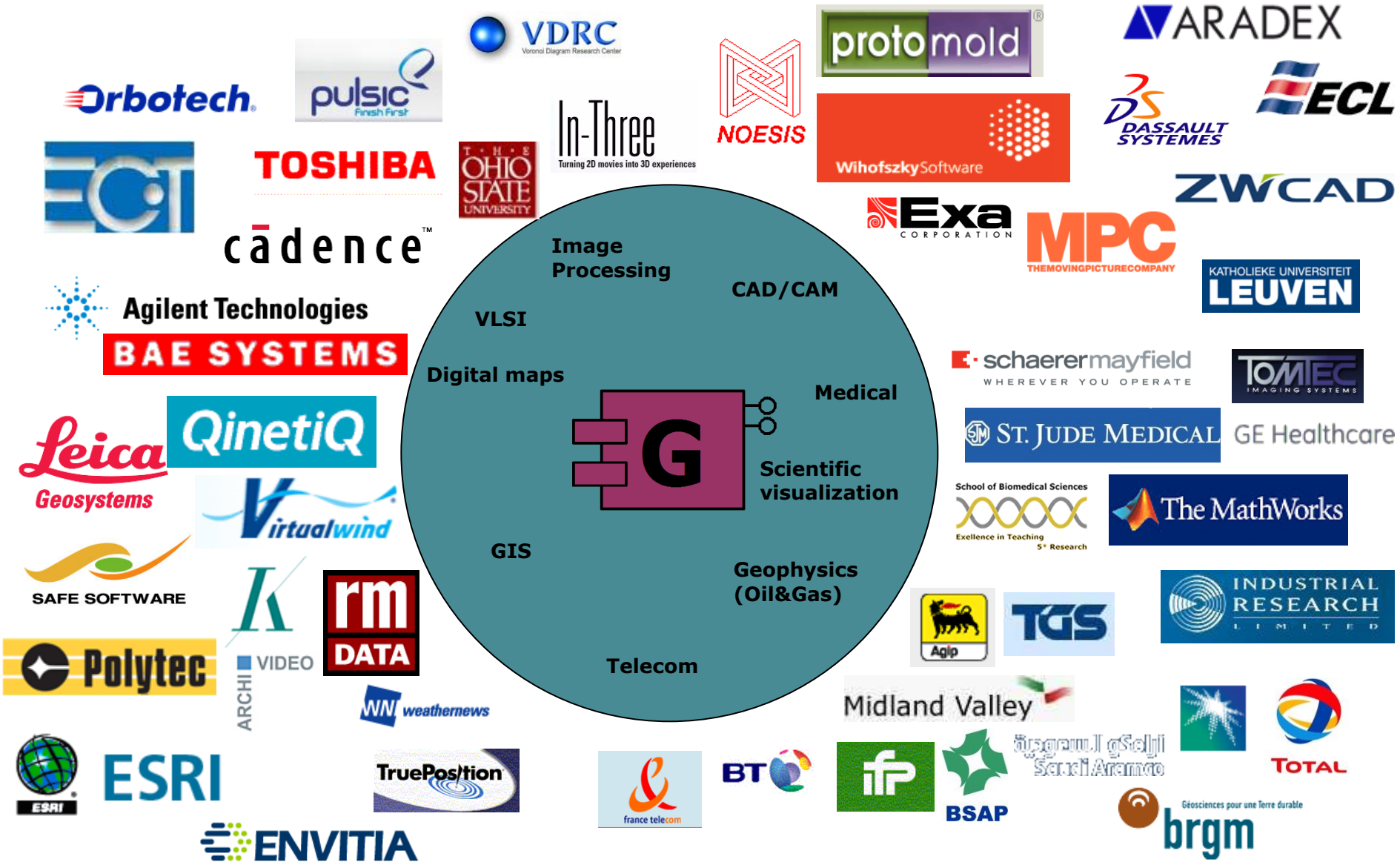
Polytope distance

QP Solver

CGAL in Numbers

500,000	lines of C++ code
10,000	downloads/year (+ Linux distributions)
3,500	manual pages
3,000	subscribers to cgal-announce
1,000	subscribers to cgal-discuss
120	packages
90	commercial users
20	active developers
12	months release cycle
2	licenses: Open Source and commercial

Some Commercial Users



CGAL Open Source Project

Project = « Planned Undertaking »

- Institutional members make a long term commitment: Inria, MPI, Tel-Aviv U, Utrecht U, Groningen U, ETHZ, GeometryFactory, FU Berlin, Forth, U Athens
- Editorial Board
  - Steers and animates the project
  - Reviews submissions
- Development Infrastructure
  - Gforge: svn, tracker, nightly testsuite,...
  - 120p developer manual and mailing list
  - Two 1-week developer meetings per year

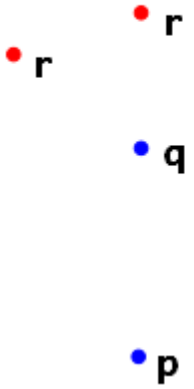
Contributions

- Submission of specifications of new contributions
- Review and decision by the Editorial Board

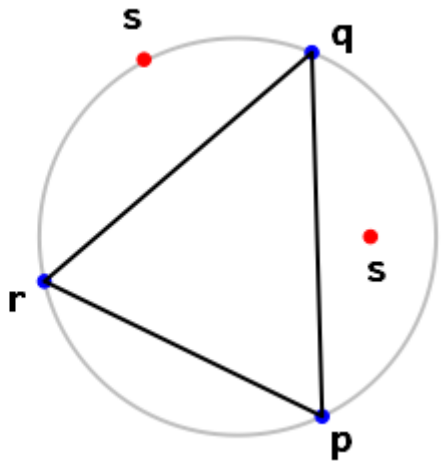
- Value for contributor
  - Integration in the CGAL community
  - Gain visibility in a mature project
  - Publication value for accepted contributions

Exact Geometric Computing

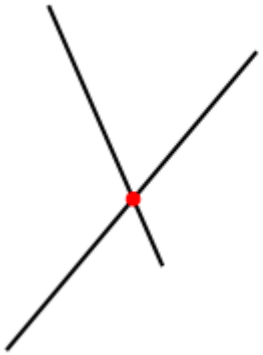
Predicates and Constructions



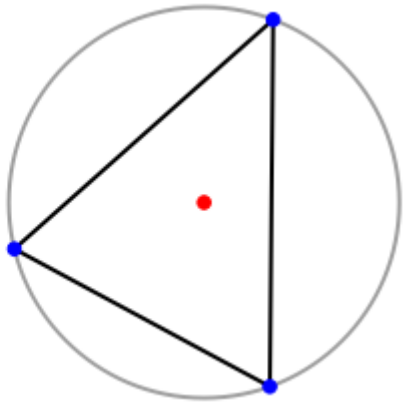
orientation



in_circle



intersection



circumcenter

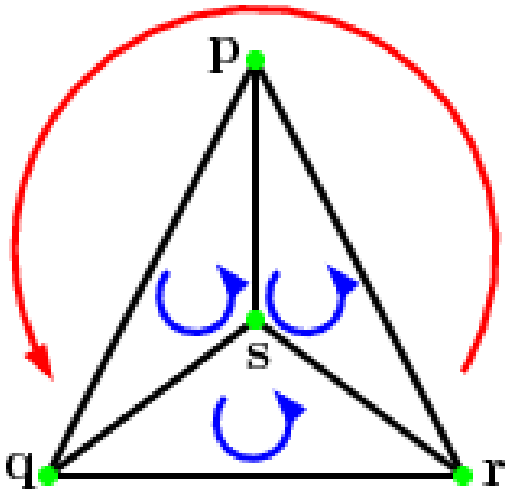
Robustness Issues

- Naive use of floating-point arithmetic causes geometric algorithms to:
  - Produce [slightly] wrong output
  - Crash after invariant violation
  - Infinite loop

- There is a gap between
  - Geometry in theory
  - Geometry with floating-point arithmetic

Geometry in Theory

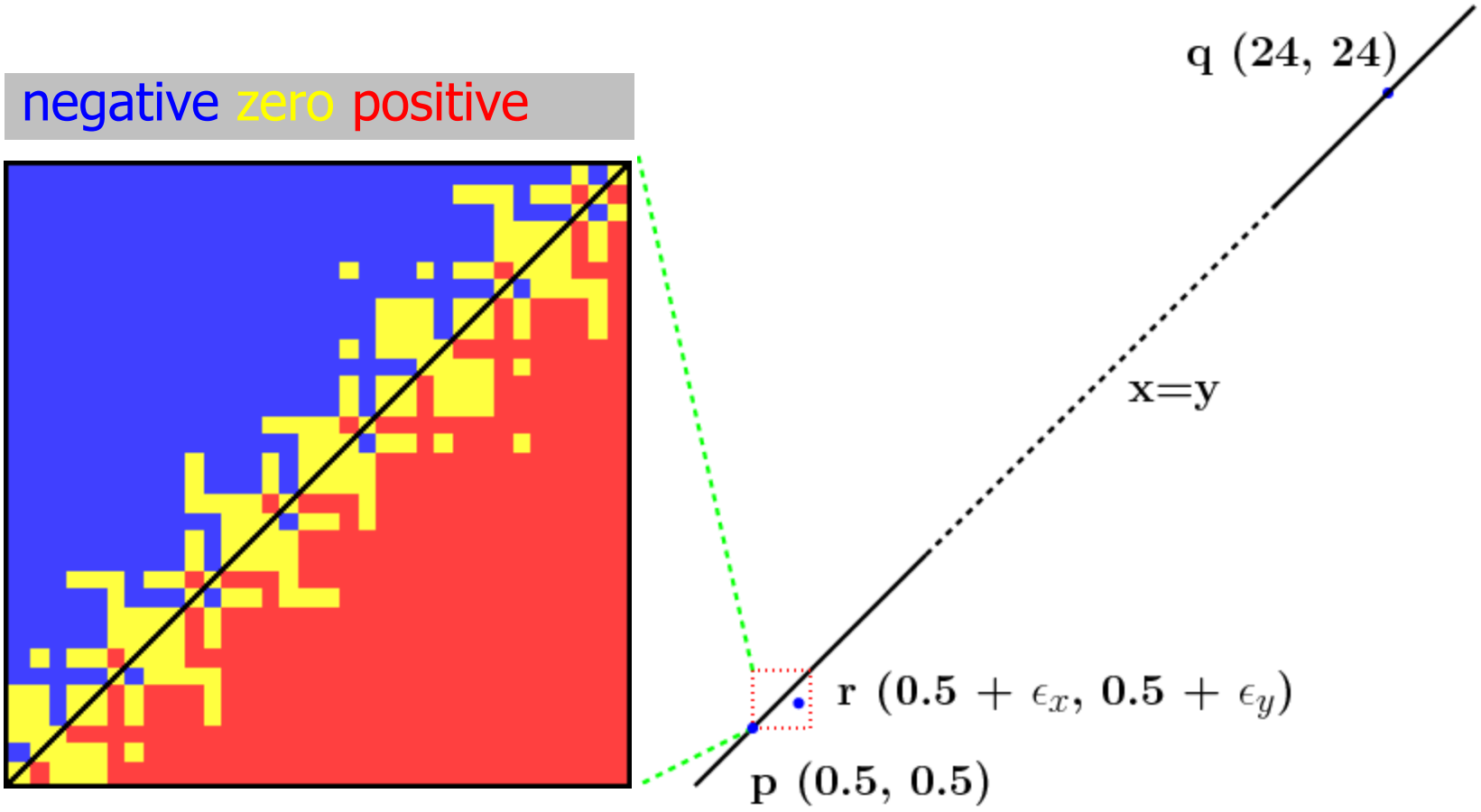
ccw(s,q,r) & ccw(p,s,r) & ccw(p,q,s) ®ccw(p,q,r)



Correctness proofs of algorithms rely on such theorems

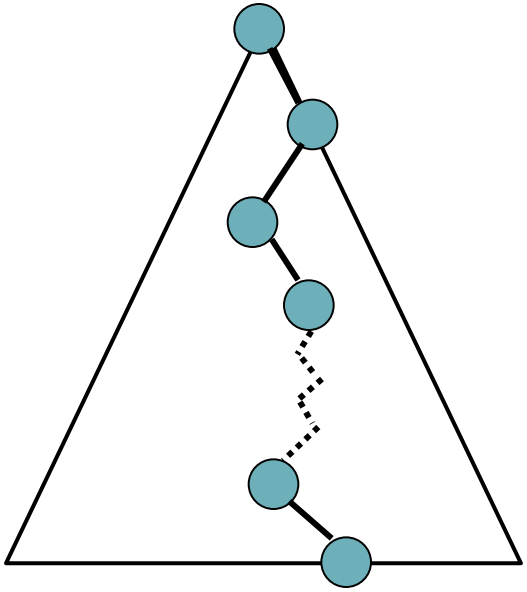
Demo: The Trouble with Double

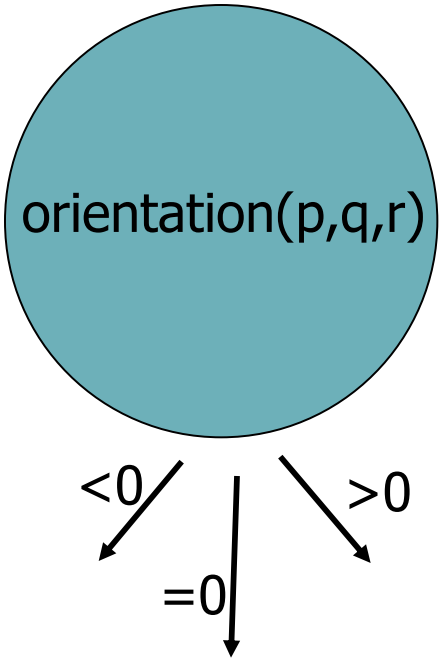
orientation(p,q,r) = sign($(p_x-r_x)(q_y-r_y)-(p_y-r_y)(q_x-r_x)$)



Exact Geometric Computing [Yap]

Make sure that the control flow in the implementation corresponds to the control flow with exact real arithmetic





Filtered Predicates

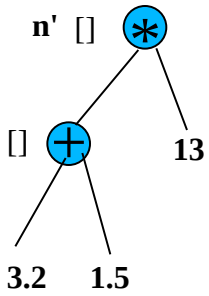
- Generic functor adaptor Filtered_predicate<>
  - Try the predicate instantiated with intervals
  - In case of uncertainty, evaluate the predicate with multiple precision arithmetic

- Refinements:
  - Static error analysis
  - Progressively increase precision

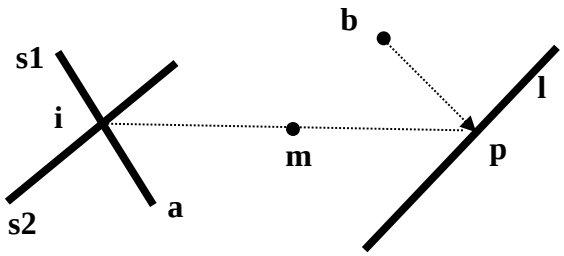
Filtered Constructions

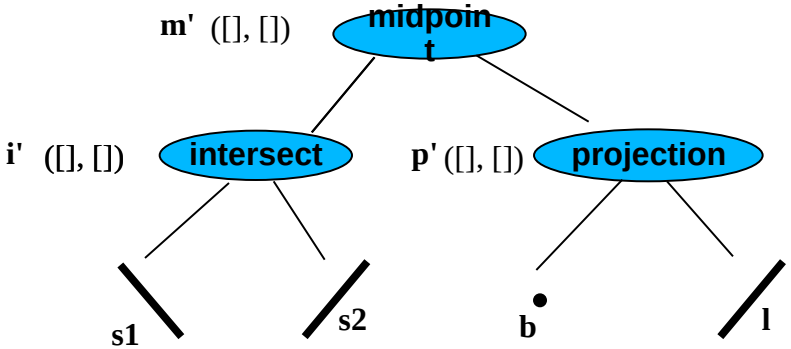
Lazy number = interval and arithmetic expression tree

$$(3.2 + 1.5) * 13$$



Lazy object = approximated object and geometric operation tree





Test that may trigger an exact re-evaluation:

if (n' < m')

if (collinear(a',m',b'))

The User Perspective

- Convenience Kernels
  - **`Exact_predicates_inexact_constructions_kernel`**
  - **`Exact_predicates_exact_constructions_kernel`**
  - **`Exact_predicates_exact_constructions_kernel_with_sqrt`**
- Number Types
  - **`double, float`**
  - **`CGAL::Gmpq (rational), Core (algebraic)`**
  - **`CGAL::Lazy_exact_nt<ExactNT>`**
- Kernels
  - **`CGAL::Cartesian<NT>`**
  - **`CGAL::Filtered_kernel<Kernel>`**
  - `CGAL::Lazy_kernel<NT>`

CGAL manual

Merits and Limitations

- Ultimate robustness inside the black box
- The time penalty is reasonable, e.g. 10% for 3D Delauny triangulation of 1M random points
- Limitations of Exact Geometric Computing
  - Topology preserving rounding is non-trivial
  - Construction depth must be reasonable
  - Cannot handle trigonometric functions

Generic Programming

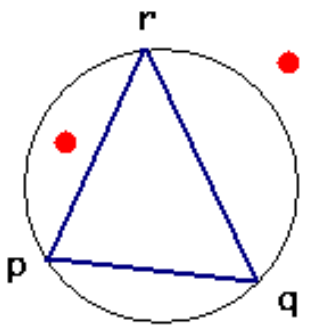
STL Genericity

```
template <class Key, class Less>
class set {
  Less less;

  insert(Key k)
  {
     if (less(k, treenode.key))
       insertLeft(k);
     else
       insertRight(k);
  }
};
```

STL manual

CGAL Genericity

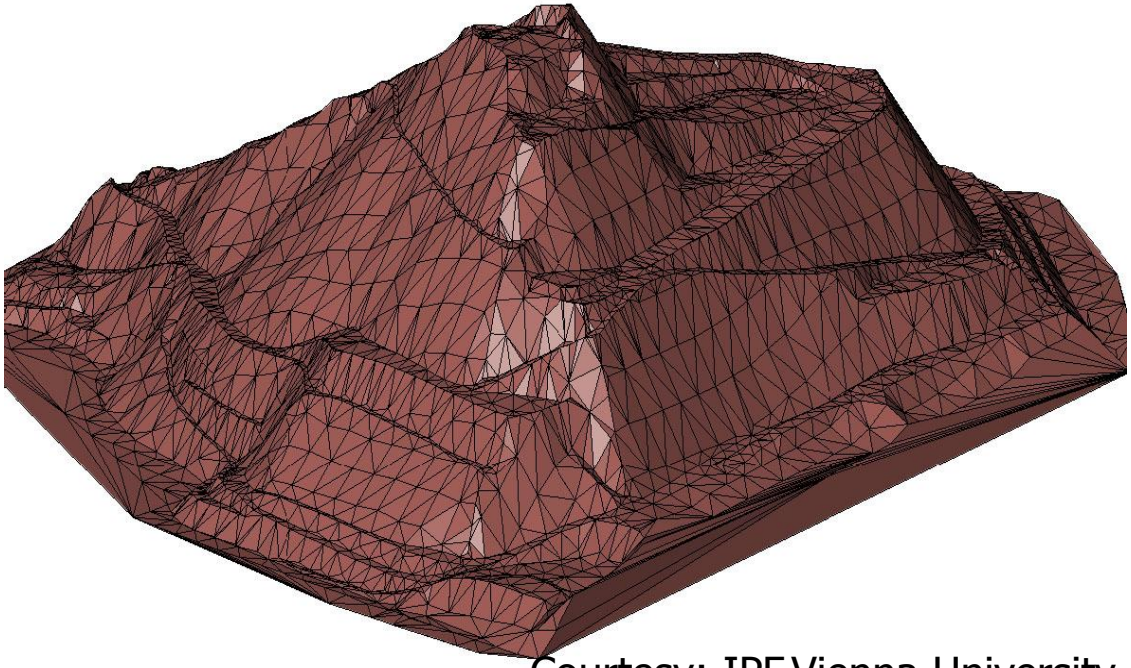
```
template < class Geometry >
class Delaunay_triangulation_2 {
    Geometry::Orientation orientation;
    Geometry::In_circle in_circle;

    void insert(Geometry::Point t) {
        ...
        if(in_circle(p,q,r,t)) {...}
        ...
        if(orientation(p,q,r){...}
    }
};
```



CGAL Genericity Demo

Without explicit conversion to points in the plane

- Triangulate the terrain in an xy-plane
- Triangulate the faces of a Polyhedron



Courtesy: IPF,Vienna University of Technology & Inpho GmbH

Summary: Overview

- Open Source project

- Clear focus on geometry
- Interfaces with de facto standards/leaders: STL, Boost, GMP, Qt, blas

- Robust and fast through exact geometric computing
- Easy to integrate through generic programming