Computational Geometry Algorithms Library

Pierre Alliez
INRIA

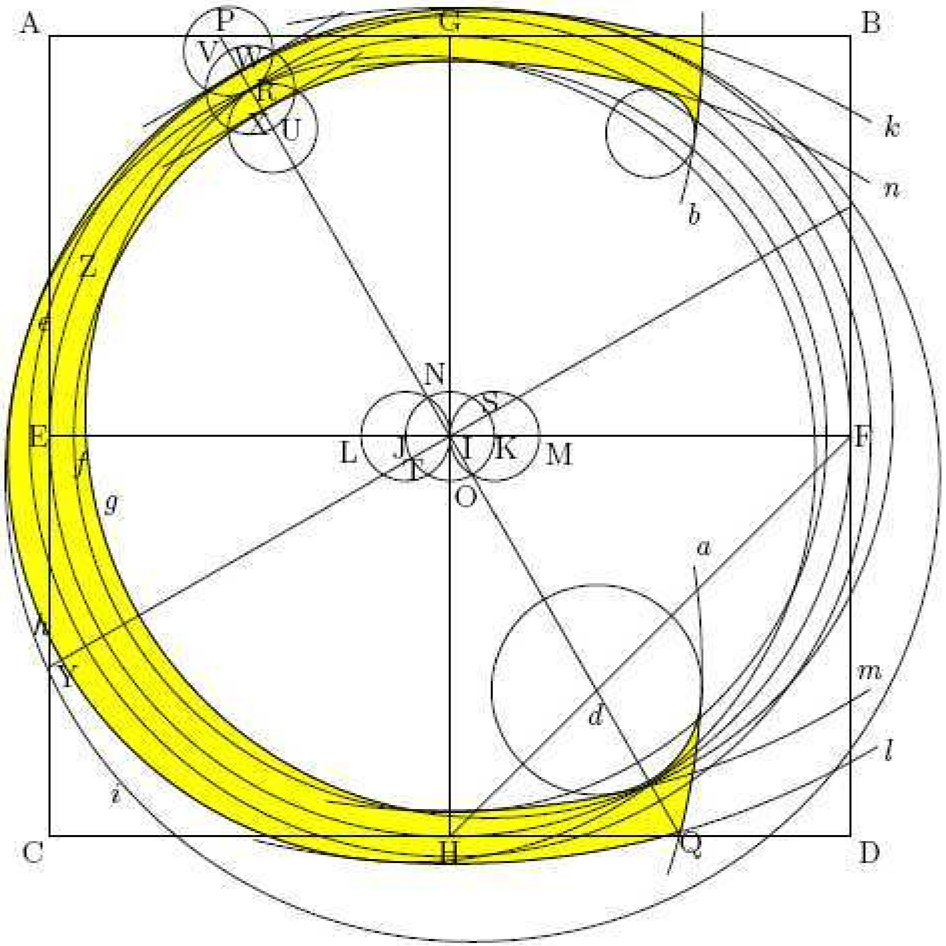
Andreas Fabri
GeometryFactory

Efi Fogel
Tel Aviv University

SIGGRAPH2008

Course Outline

Overview	Andreas	30′
Polyhedron	Pierre	40′
Arrangements	Efi	40′
Break		15′
2D Triangulations & Meshes	Andreas	40′
3D Triangulations & Meshes	Pierre	40′
Wrap-up, Q&A		15′



Overview

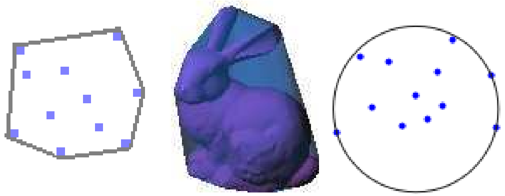
Andreas Fabri
GeometryFactory

Mission Statement

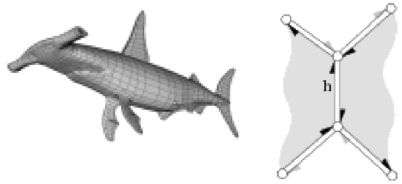
"Make the large body of geometric algorithms developed in the field of computational geometry available for industrial applications"

CGAL Project Proposal, 1996

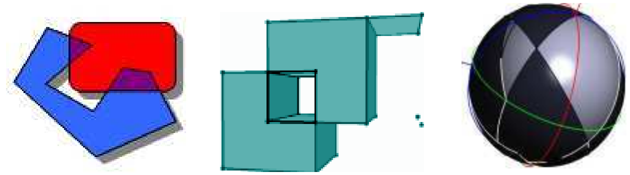
Algorithms and Datastructures



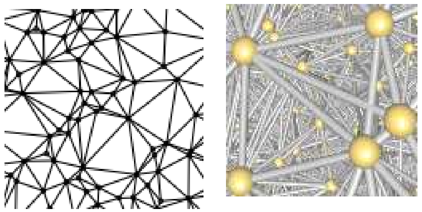
Bounding Volumes



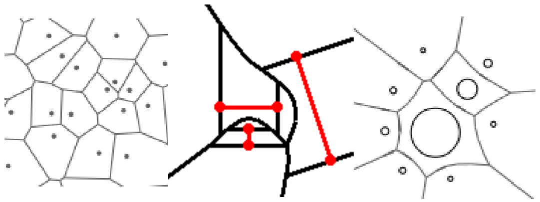
Polyhedral Surface



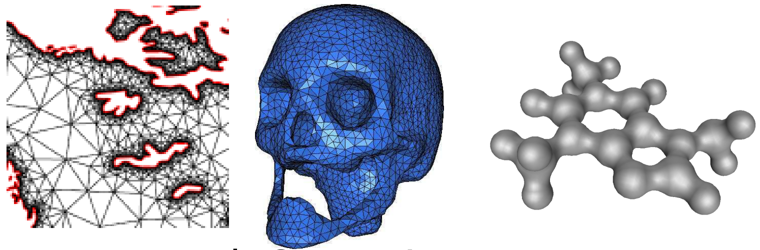
BooleanOperations



Triangulations



Voronoi Diagrams

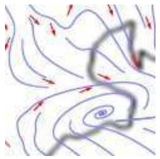


Mesh Generation

Subdivision

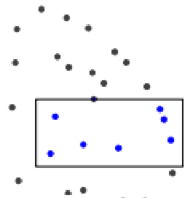
Simplification

Parametrisation

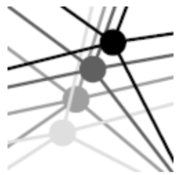


Streamlines

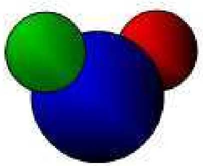
Ridge Detection



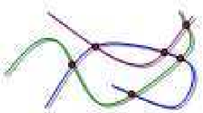
Neighbor Search



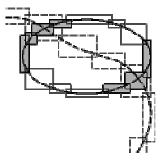
Kinetic Datastructures



Lower Envelope

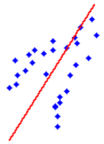


Arrangement

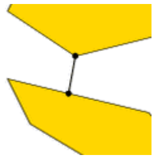


Intersection Detection

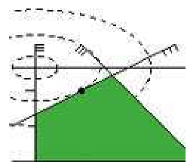
Minkowski Sum



PCA



Polytope distance



QP Sover

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

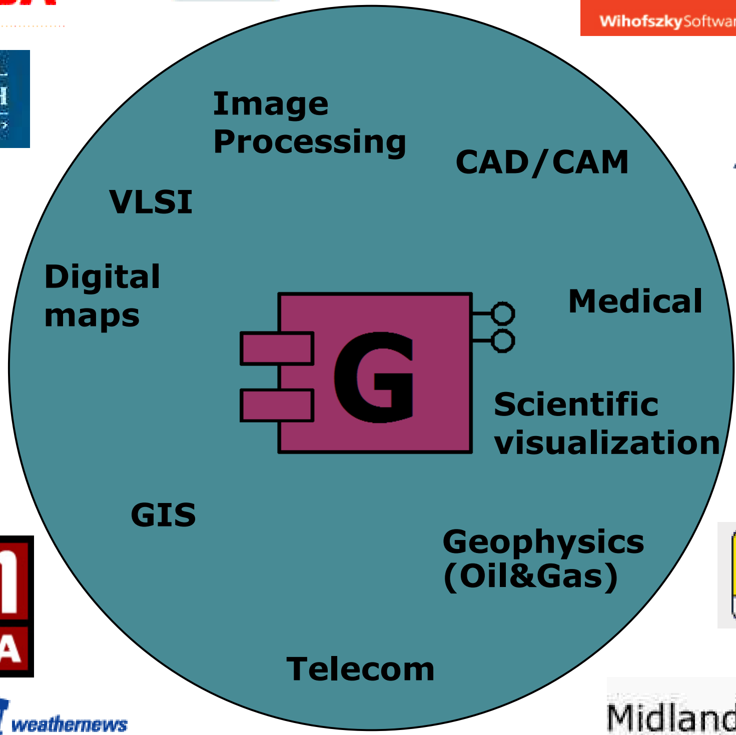
60 commercial users

20 active developers

12 months release cycle

2 licenses: Open Source and commercial

Some Commercial Users



cadence
pulsic
VDRC
THE OHIO STATE UNIVERSITY
NOESIS
protomold
Exa CORPORATION
Orbotech
TOSHIBA
Wihofszky Software
ZWCAD
INDUSTRIAL RESEARCH LIMITED
DASSAULT SYSTEMES
ECL
MPC THEMOVINGPICTURECOMPANY
Agilent Technologies
BAE SYSTEMS
schaerermayfield WHEREVER YOU OPERATE
TOMTEC IMAGING SYSTEMS
QinetiQ
ST. JUDE MEDICAL
Leica Geosystems
School of Biomedical Sciences
The MathWorks
SAFE SOFTWARE
rm DATA
Agip
TGS
ARCHI VIDEO
WN weathernews
Midland Valley
TOTAL
inpho
TruePosition
france telecom
BT
IFP
BSAP

Why They Use CGAL

" I recommended to the senior management that we start a policy of buying-in as much functionality as possible to reduce the quantity of code that our development team would have to maintain.

This means that we can concentrate on the application layer and concentrate on our own problem domain."

Senior Development Engineer
& Structural Geologist

Midland Valley Exploration

Why They Use CGAL

" My research group JYAMITI at the Ohio State University uses CGAL because it provides an efficient and robust code for Delaunay triangulations and other primitive geometric predicates. Delaunay triangulation is the building block for many of the shape related computations that we do. [...]

Without the robust and efficient codes of CGAL, these codes could not have been developed. "

Tamal Dey
Professor, Ohio State University

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

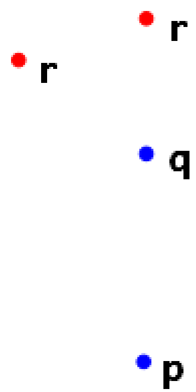
- Submission of specifications of new contributions
- Review and decision by the Editorial Board

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

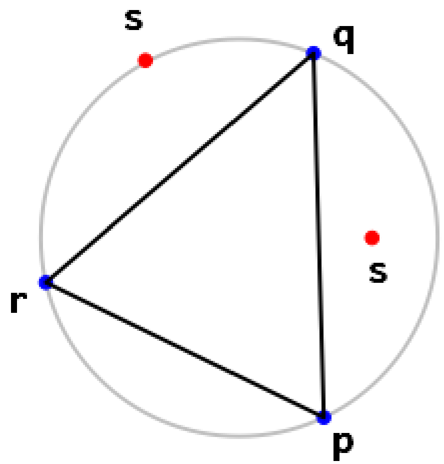
Exact Geometric Computing

Predicates and Constructions

Predicates

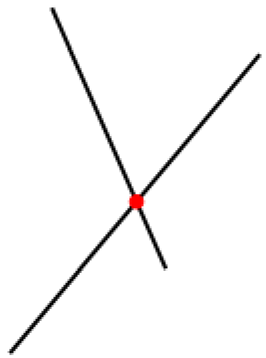


orientation

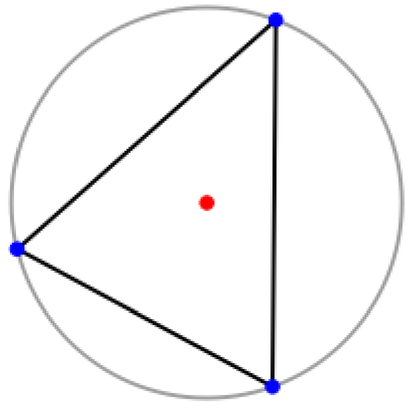


in_circle

Constructions



intersection



circumcenter

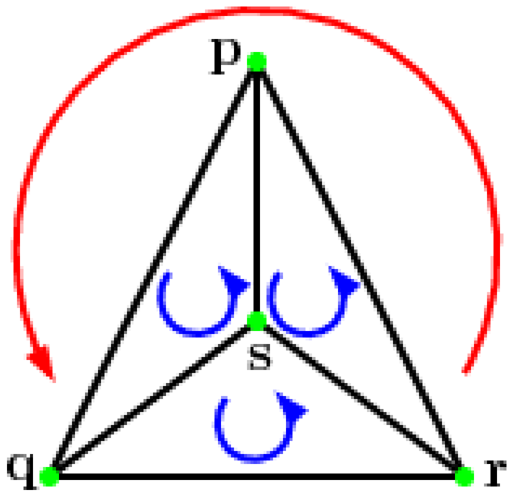
Robustness Issues

- 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

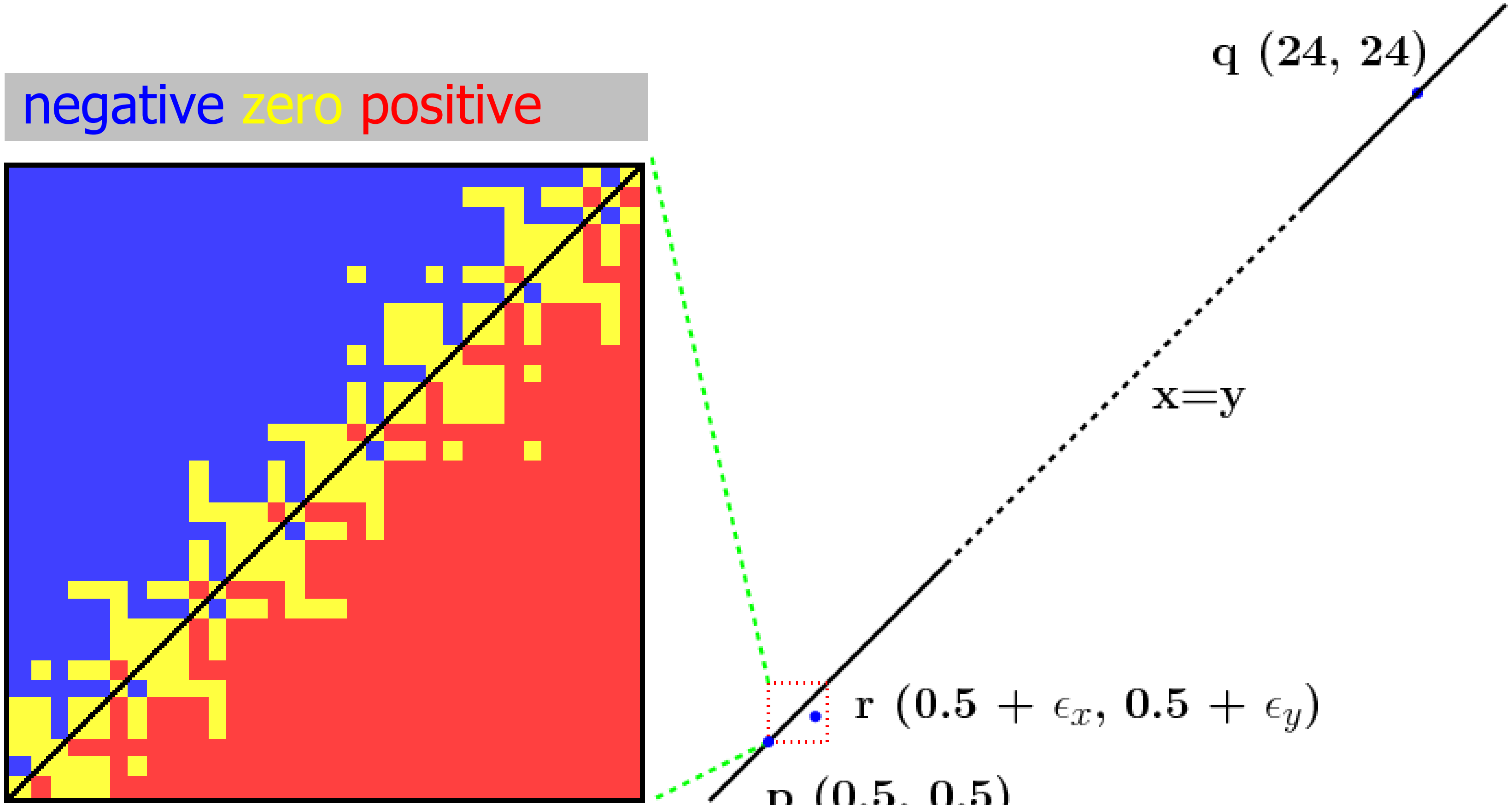
ccw(s,q,r) & ccw(p,s,r) & ccw(p,q,s) ⇒ ccw(p,q,r)



Correctness proofs of algorithms rely on such theorems

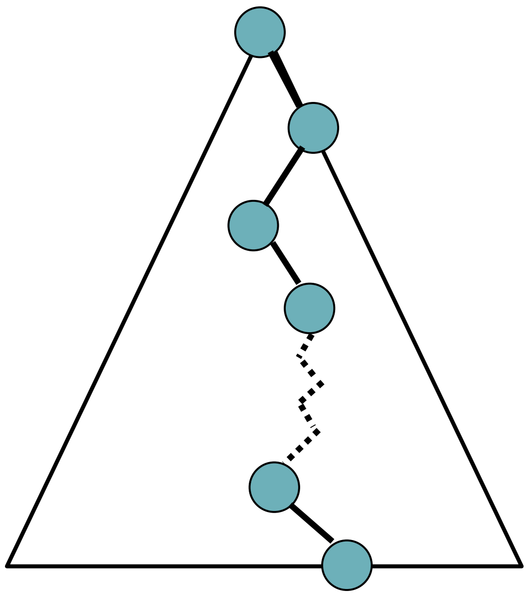
The Trouble with Double

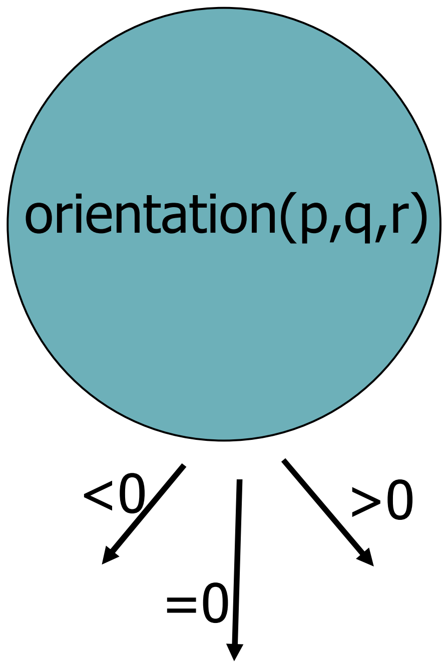
orientation(p,q,r) = $\text{sign}((p_x-r_x)(q_y-r_y)-(p_y-r_y)(q_x-r_x))$



Exact Geometric Computing [Yap]

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





Filtered Predicates

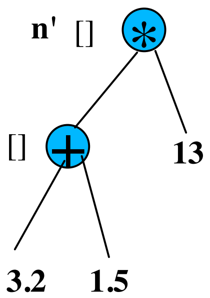
- Generic functor adaptor Filtered_predicate<>
  - Try the predicate instantiated with intervals
  - In case of uncertainty, evaluate the predicate with multiple precision arithmetic

- Refinements:
  - Static error analysis
  - Progressively increase precision

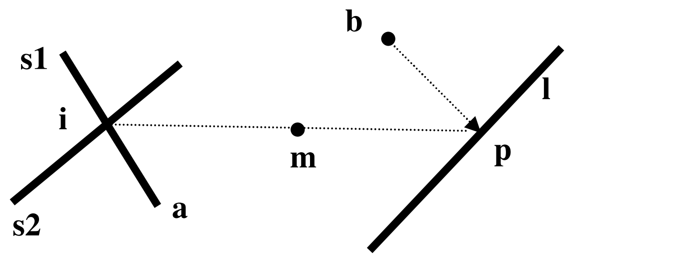
Filtered Constructions

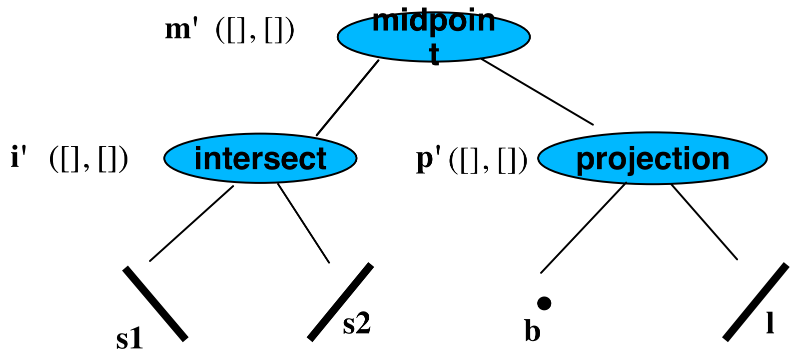
Lazy number = interval and arithmetic expression tree

$$(3.2 + 1.5) * 13$$



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

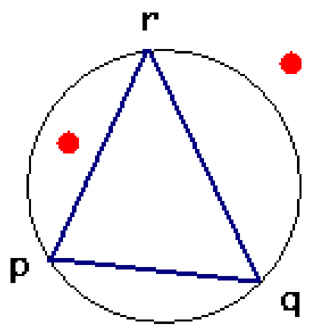
STL Genericity

```
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

```
template < class Geometry, class TDS >
class Delaunay_triangulation_2 {


};



template < class Vertex, class Face >
class Triangulation_data_structure_2 {


};
```

Iterators

```
template <class Geometry>
class Delaunay_triangulation_2 {

  typedef .. Vertex_iterator;
  typedef .. Face_iterator;

  Vertex_iterator vertices_begin();
  Vertex_iterator vertices_end();

  template <class OutputIterator>
  incident_faces(Vertex_handle v, OutputIterator it);
};
std::list<Face_handle> faces;
dt.incident_faces(v, std::back_inserter(faces));
```

Iterators

```
template <class Geometry>
class Delaunay_triangulation_2 {

  template <class T>
  void insert(T begin, T end);// typeof(*begin)==Point
};
```

```
list<Kernel::Point_2> points;

Delaunay_triangulation<Kernel> dt;

dt.insert(points.begin(), points.end());
```