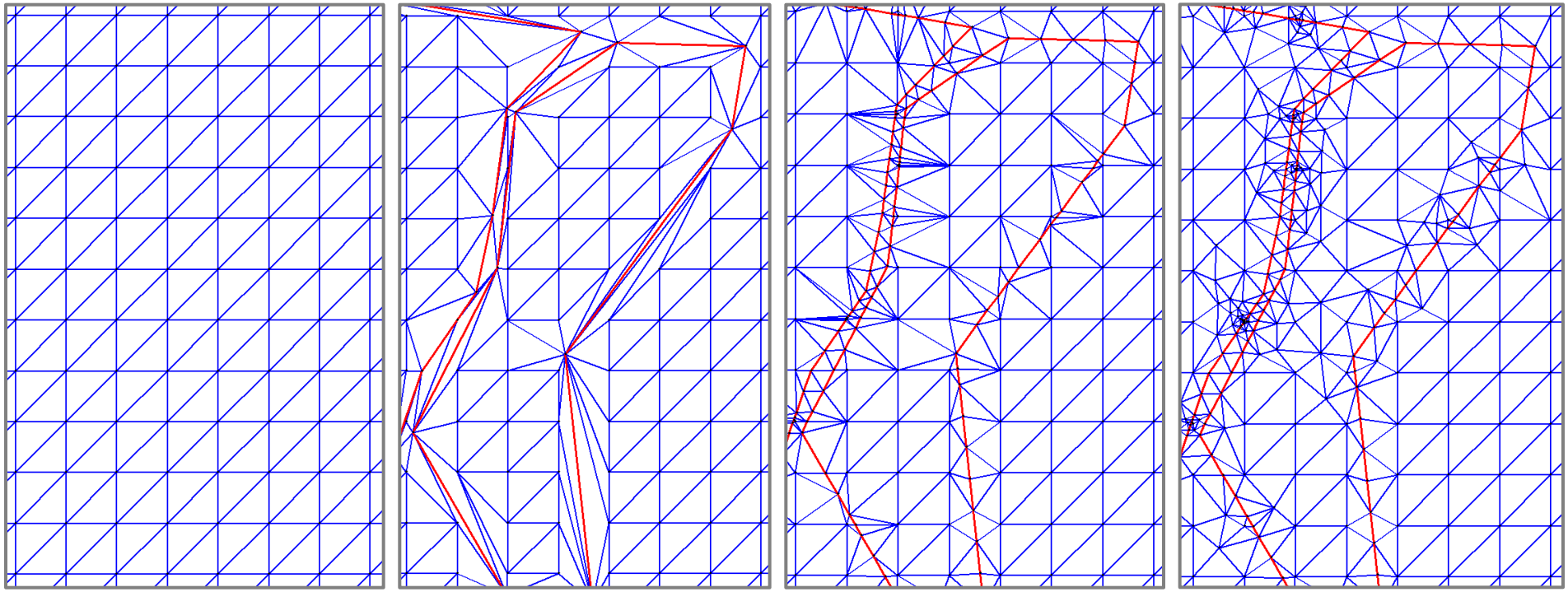
2D Delaunay Refinement in CGAL

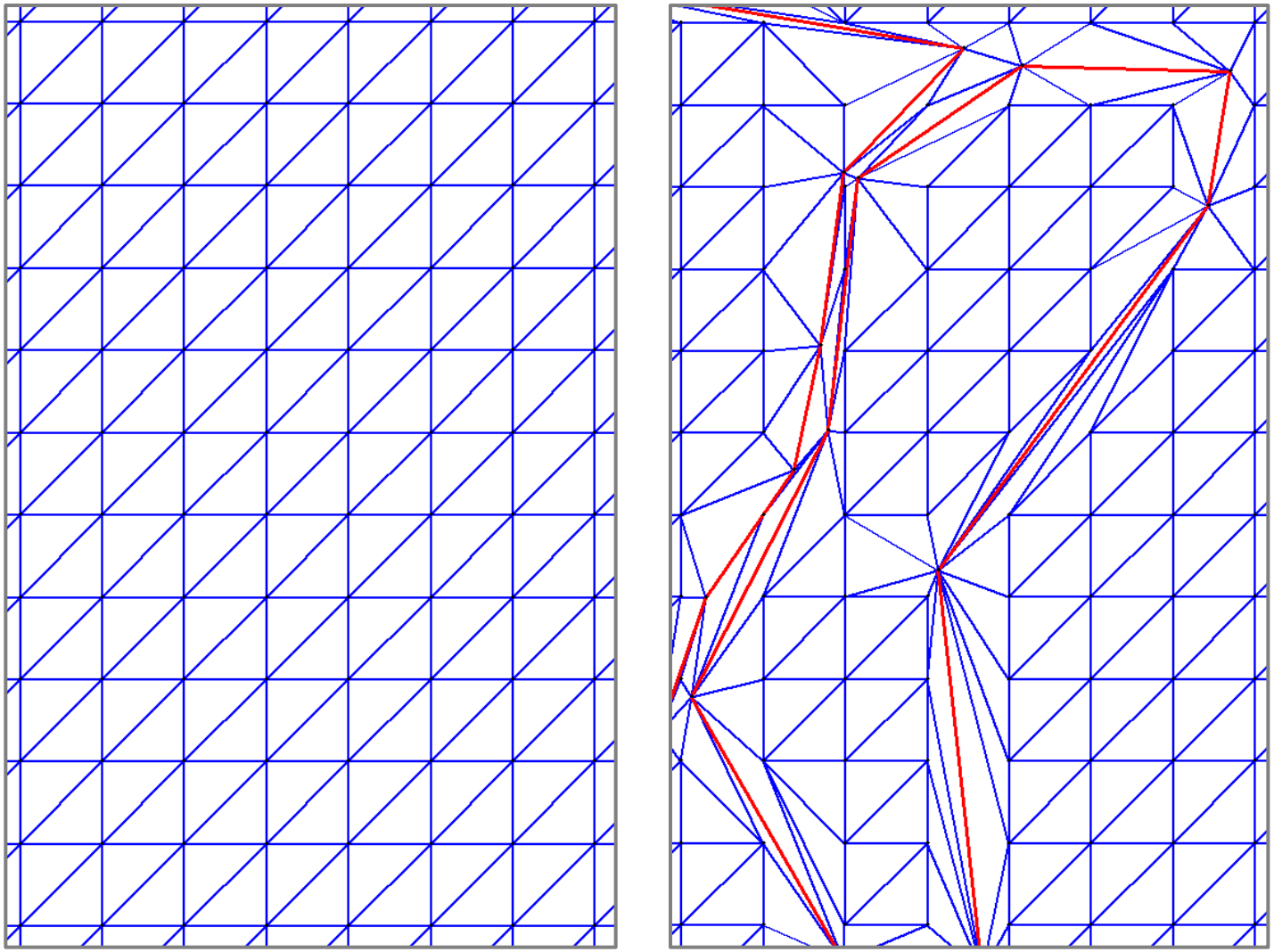
From Triangulations to Quality Meshes



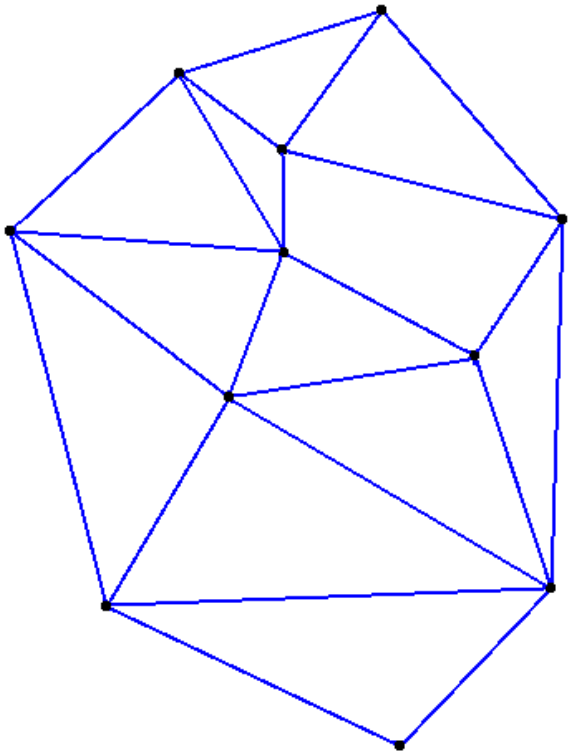
Code Example

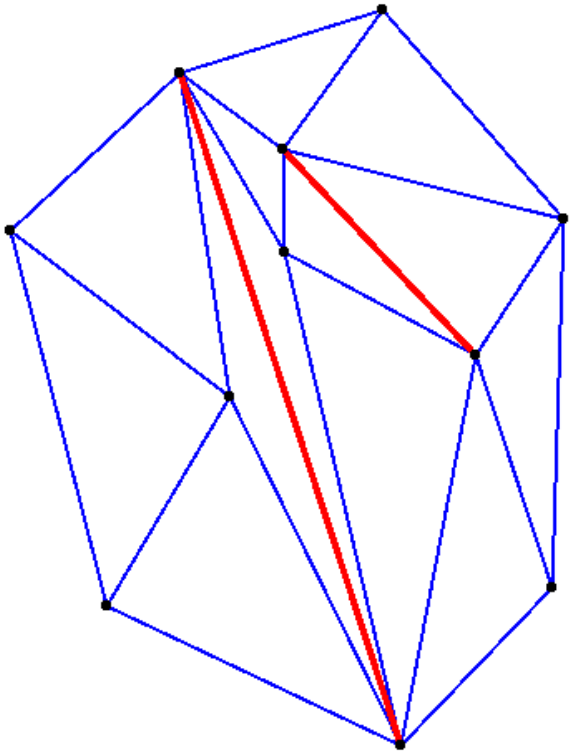
```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Delaunay_triangulation_2.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_2 Point;

typedef CGAL::Delaunay_triangulation_2<Kernel> Delaunay;
typedef Delaunay::Vertex_handle Vertex_handle;

int main()
{
  Delaunay dt;

  dt.insert( std::istream_iterator<Point>(std::cin),
             std::istream_iterator<Point>() );

  Vertex_handle v = dt.nearest_vertex(Point(0.0,0.0));

  std::cout << "Nearest vertex to origin: " << v->point() << std::endl;

  return 0;
}
```

Adding Constraints

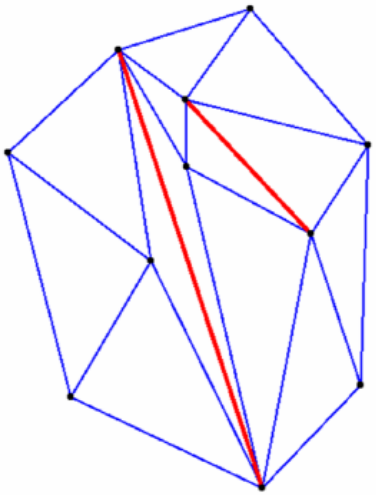


Constrained Delaunay Triangulation

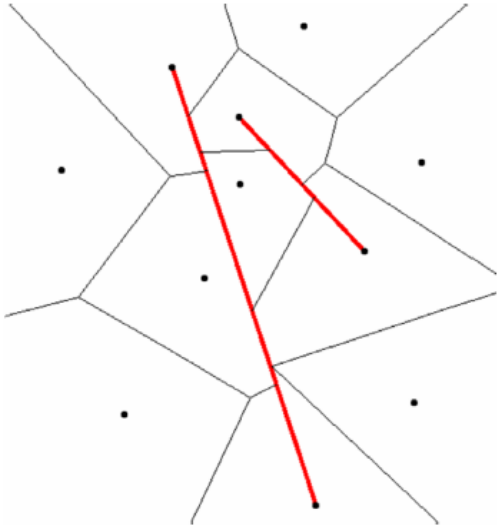




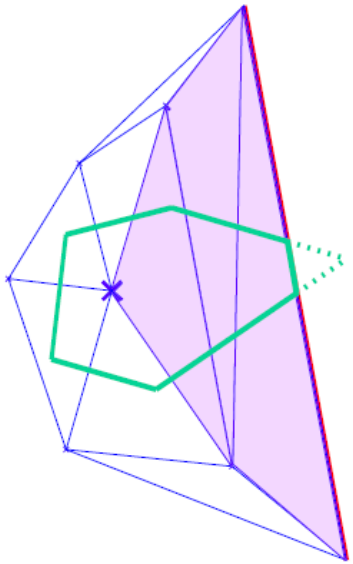
Pseudo-dual: Bounded Voronoi Diagram



constrained

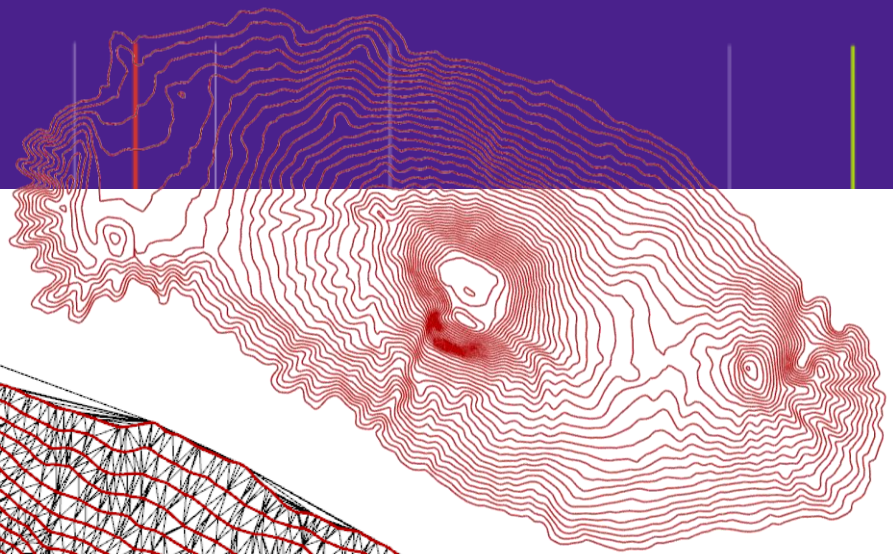


Bounded Voronoi diagram



“blind” triangles

Constrained Delaunay

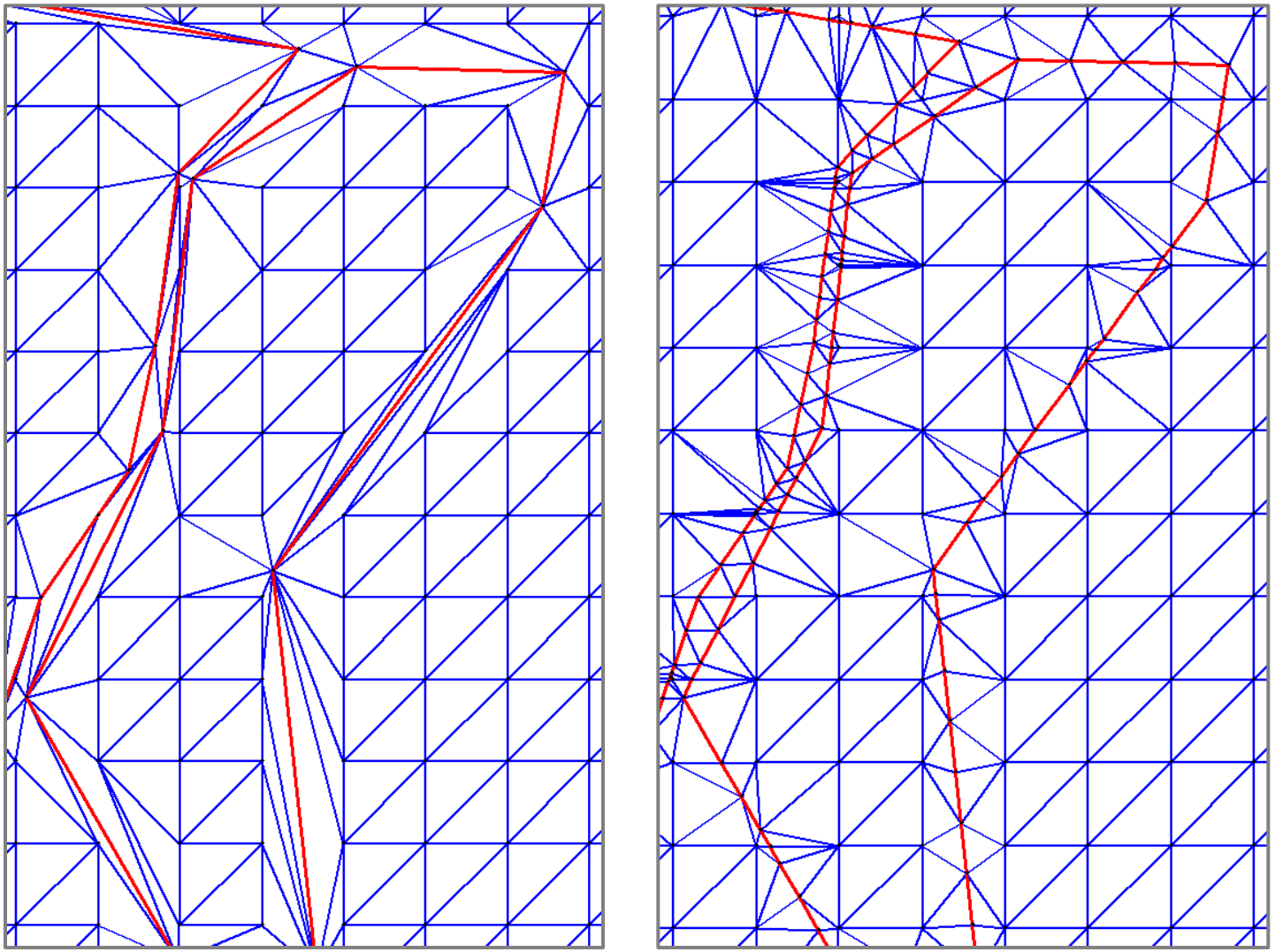


Kilimandjaro
elevation contour
lines (38K segments)

Code Example

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Constrained_Delaunay_triangulation_2.h>

typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef Kernel::Point_2 Point;

typedef CGAL::Constrained_Delaunay_triangulation_2<Kernel> CDT;
typedef CDT::Vertex_handle Vertex_handle;

int main()
{
  CDT cdt;

  // from points
  cdt.insert_constraint(Point(0.0,0.0), Point(1.0,0.0));

  // from vertices
  Vertex_handle v1 = cdt.insert(Point(2.0,3.0));
  Vertex_handle v2 = cdt.insert(Point(4.0,5.0));
  cdt.insert_constraint(v1,v2);

  return 0;
}
```

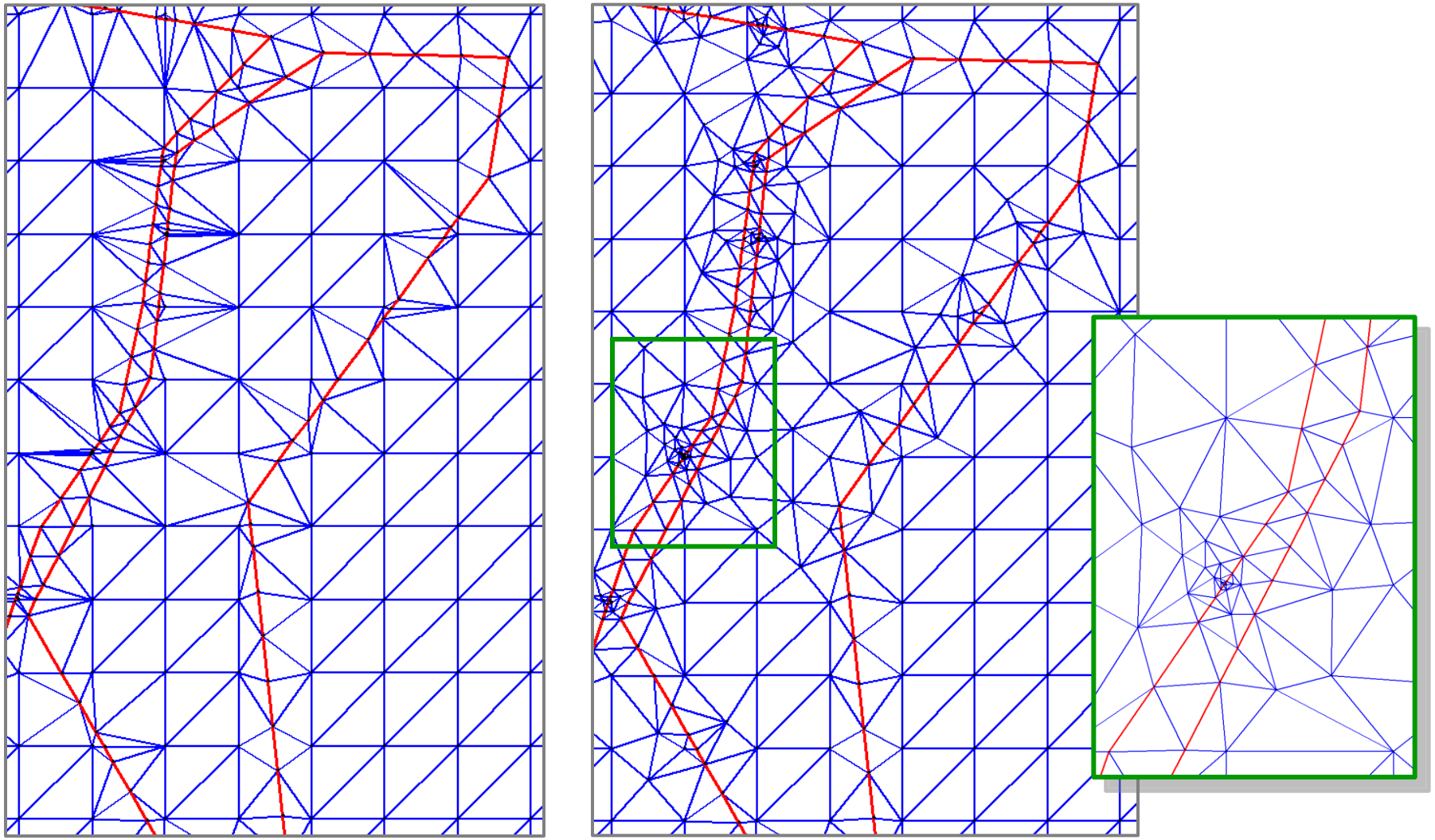
Conforming Delaunay



Code Example

```
#include <CGAL/Triangulation_conformer_2.h>

// constrained Delaunay triangulation
CDT cdt;
… // insert points & constraints

CGAL::make_conforming_Delaunay_2(cdt);
```

Delaunay Meshing



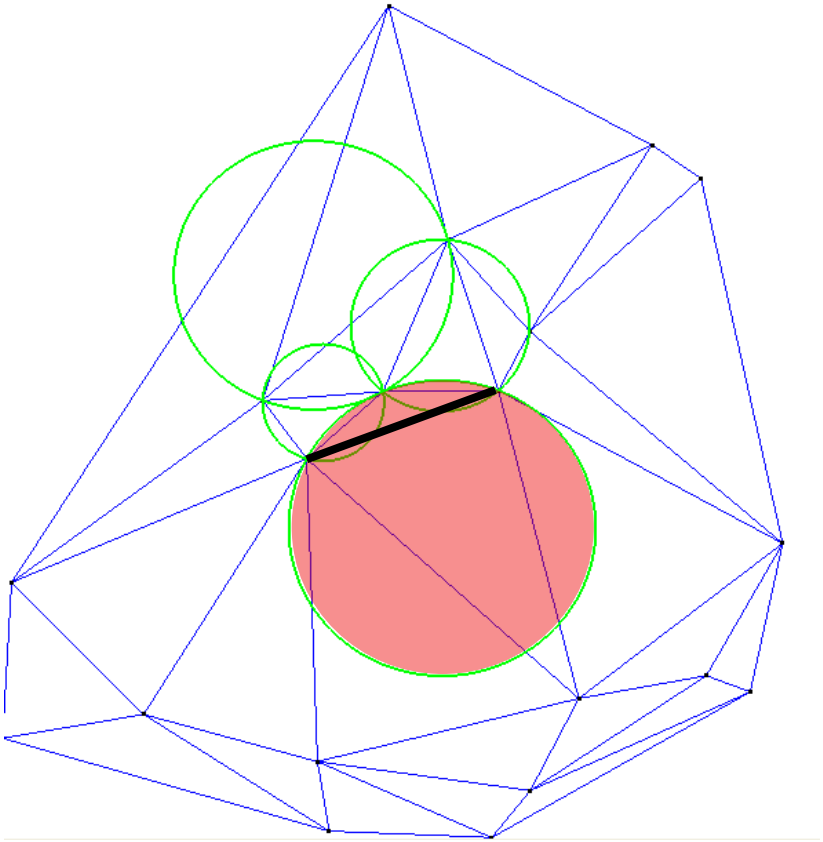
Code Example

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Constrained_Delaunay_triangulation_2.h>
#include <CGAL/Delaunay_mesher_2.h>
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef CGAL::Constrained_Delaunay_triangulation_2<Kernel> CDT;

int main()
{
  CDT cdt;
  … // insert points and constraints
  CGAL::refine_Delaunay_mesh_2(cdt);
  return 0;
}
```

Background

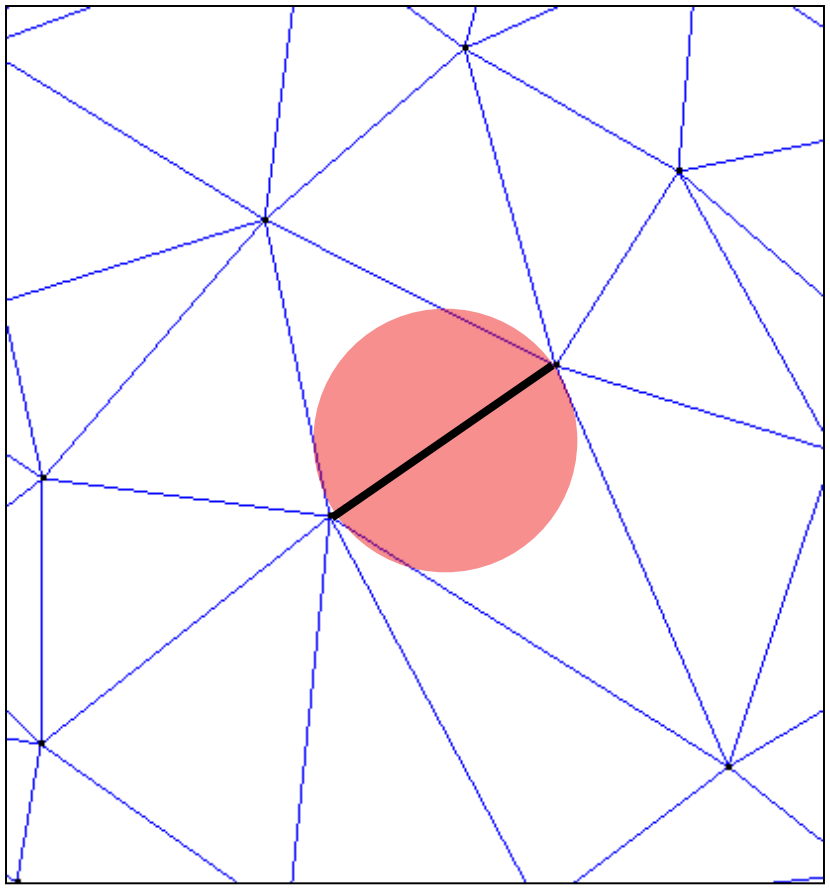
Delaunay Edge

An edge is said to be a Delaunay edge, if it is inscribed in an empty circle



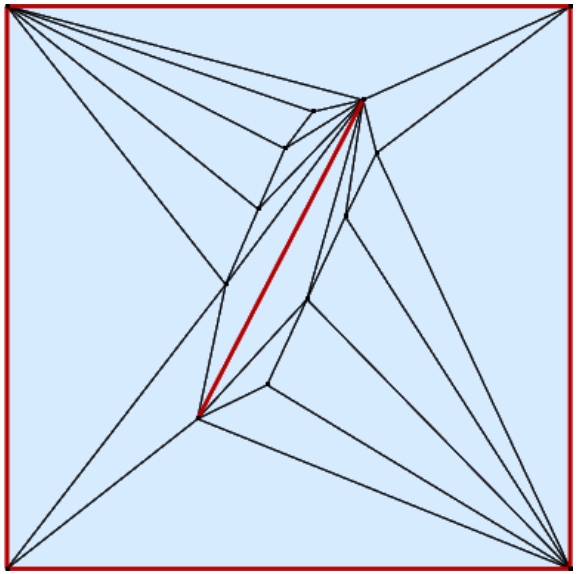
Gabriel Edge

An edge is said to be a Gabriel edge, if its diametral circle is empty

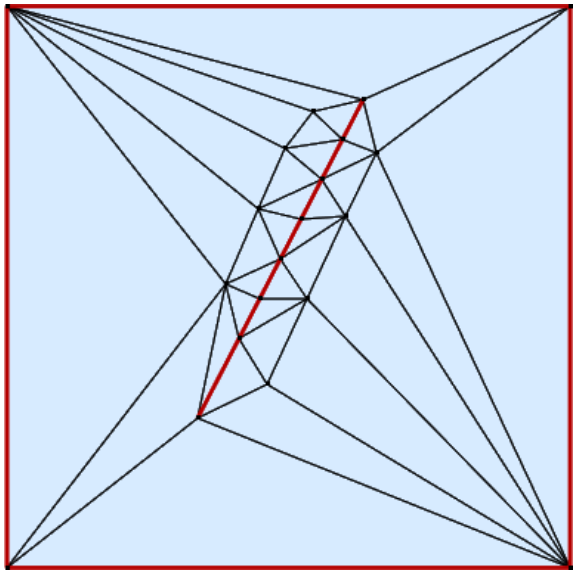


Conforming Delaunay Triangulation

A constrained Delaunay triangulation is a conforming Delaunay triangulation, if every constrained edge is a Delaunay edge



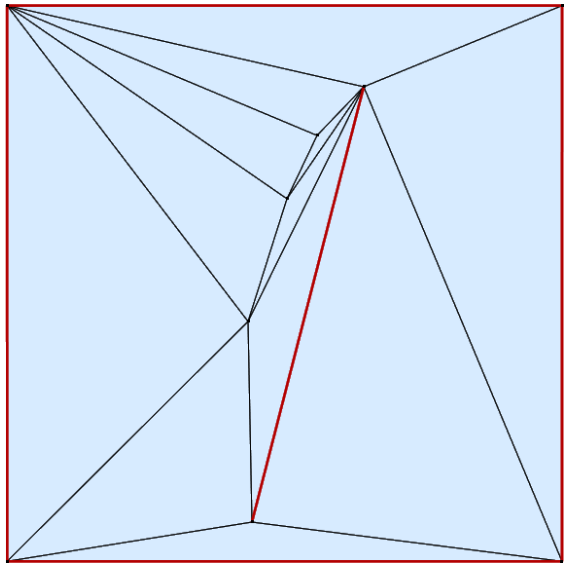
non conforming



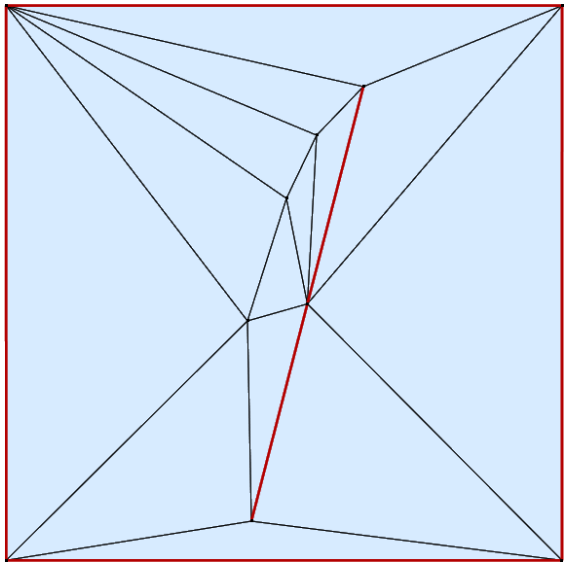
conforming

Conforming Gabriel Triangulation

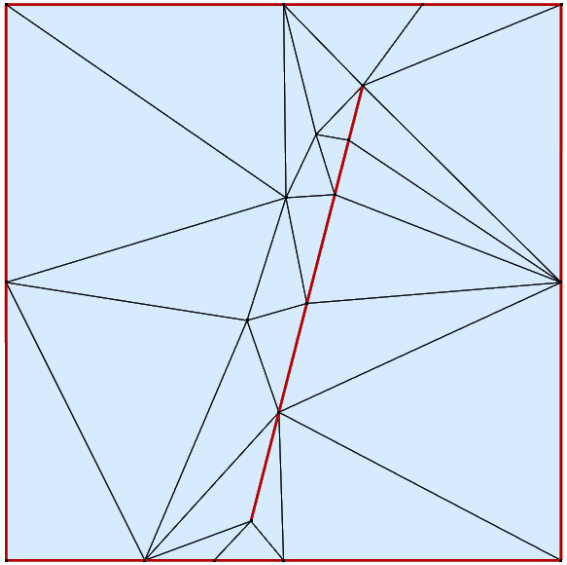
A constrained Delaunay triangulation is a conforming Gabriel triangulation, if every constrained edge is a Gabriel edge



non conforming



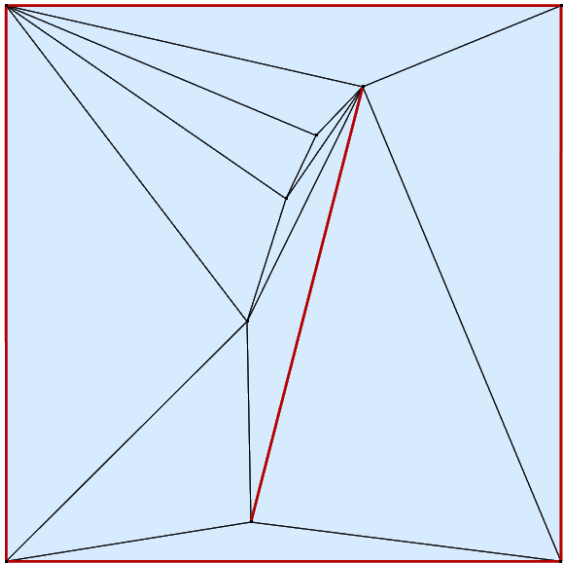
conforming



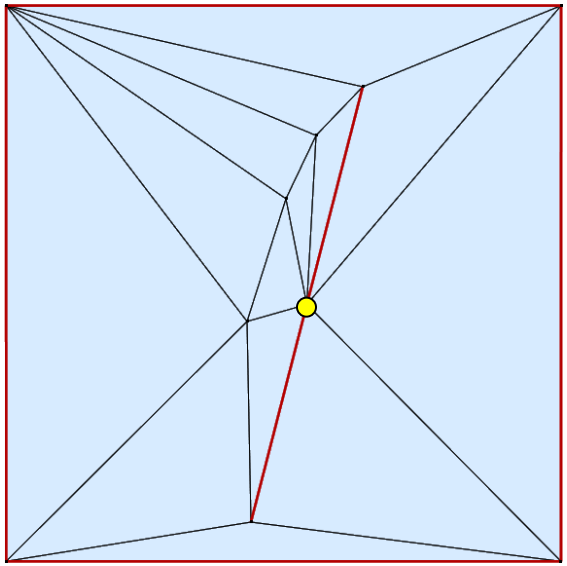
Gabriel

Steiner Vertices

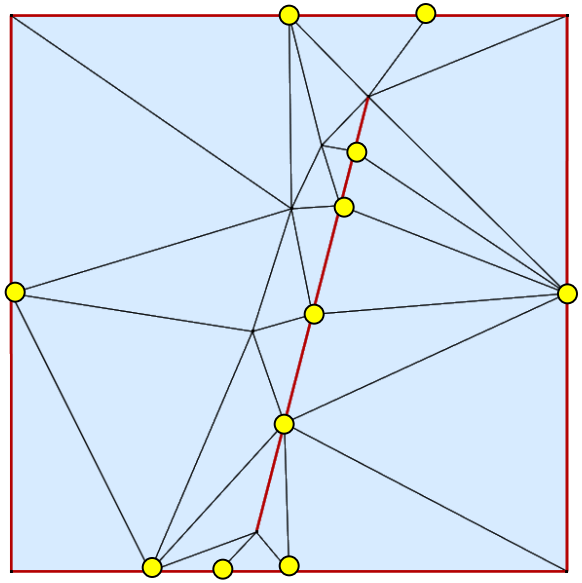
Any constrained Delaunay triangulation can be refined into a conforming Delaunay or Gabriel triangulation by adding Steiner vertices.



non conforming



conforming

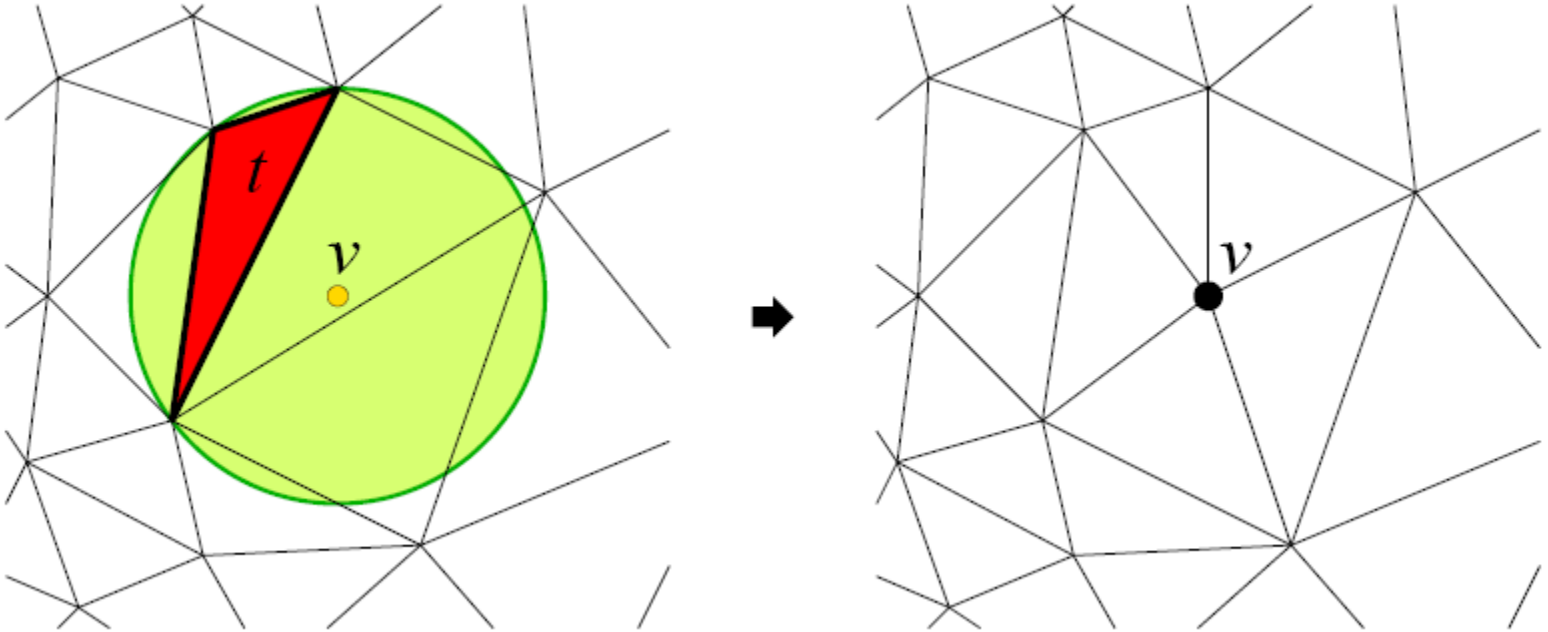


Gabriel

Delaunay Refinement

Rule #1: break bad elements by inserting circumcenters (Voronoi vertices)

- "bad" in terms of size or shape (too big or skinny)

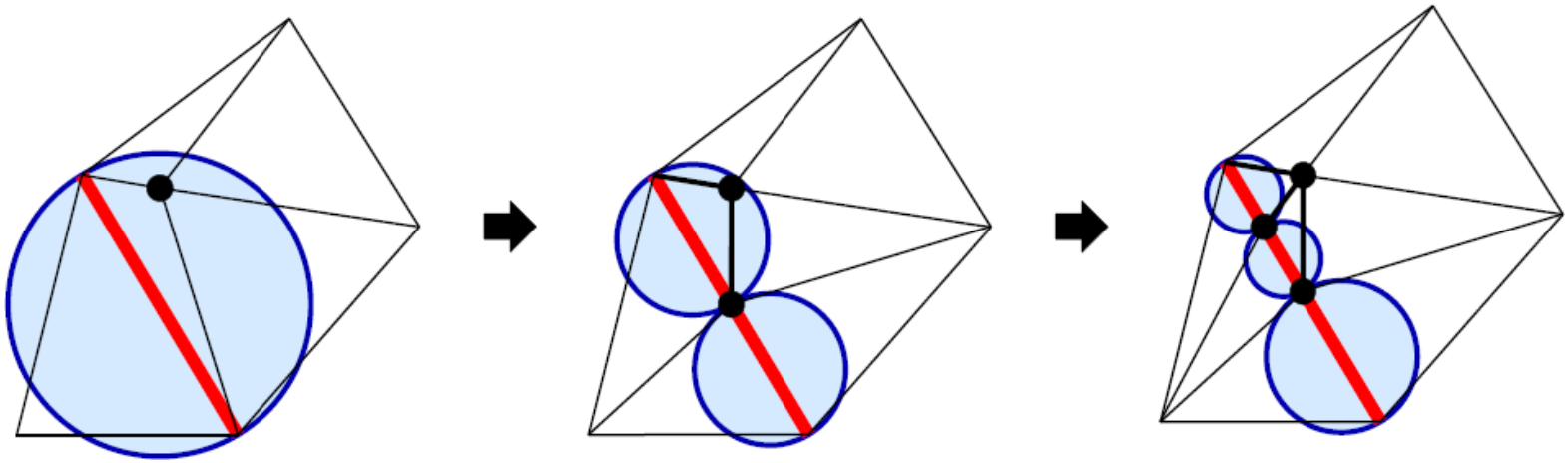


Picture taken from [Shewchuk]

Delaunay Refinement

Rule #2: Midpoint vertex insertion

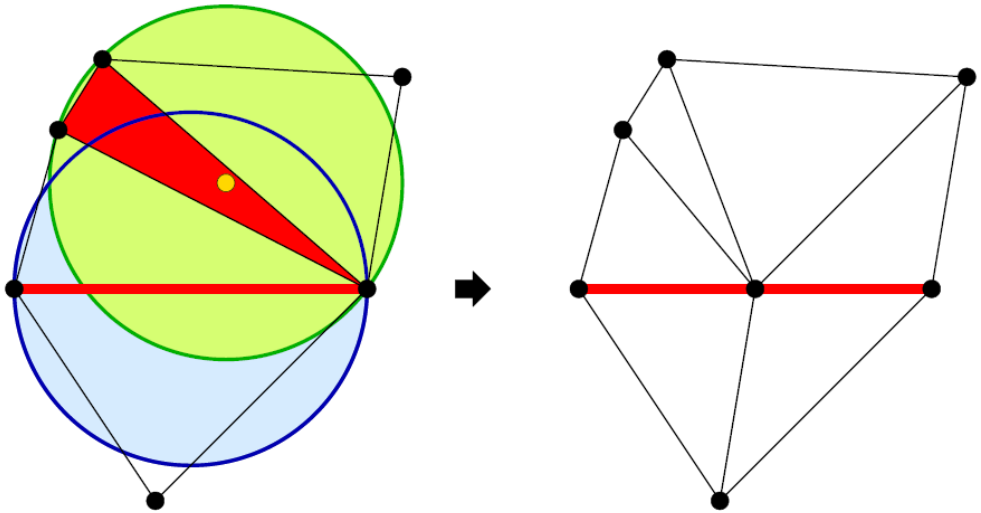
A constrained segment is said to be encroached, if there is a vertex inside its diametral circle



Picture taken from [Shewchuk]

Delaunay Refinement

Encroached subsegments have priority over skinny triangles

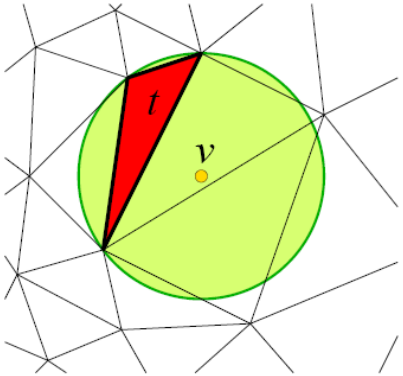


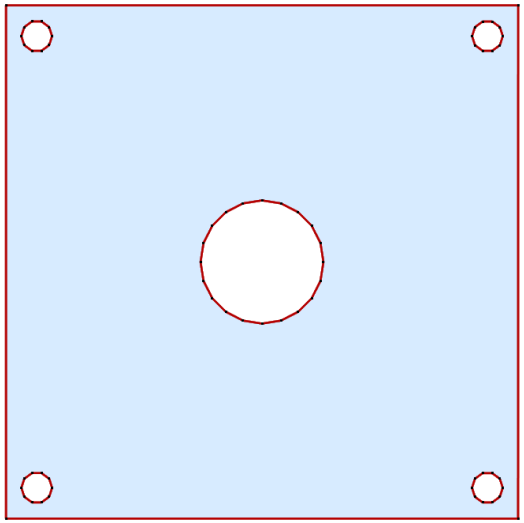
Picture taken from [Shewchuk]

API

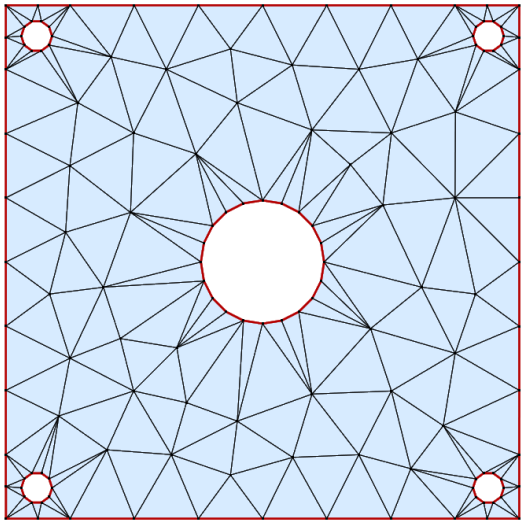
Parameters for Mesh Generation

- **Shape**
  - Lower bound on triangle angles

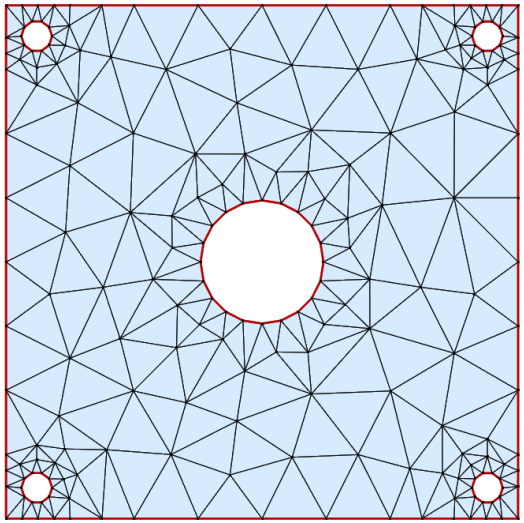




Input PLSG



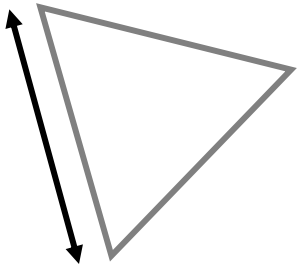
5 deg



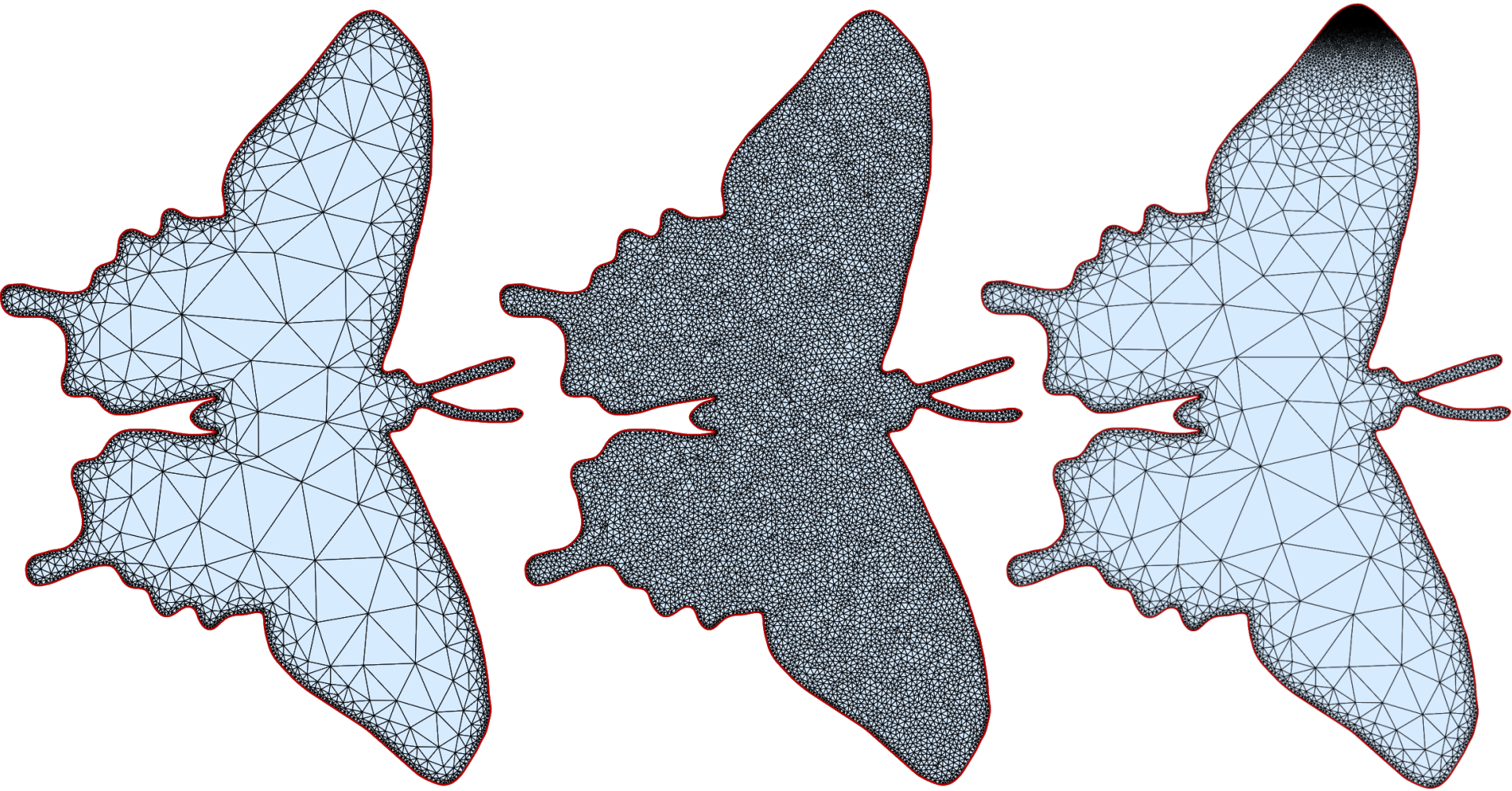
20.7 deg

Parameters for Mesh Generation

- Shape
  - Lower bound on triangle angles
- **Size**
  - No constraint
  - Uniform sizing
  - Sizing function



Sizing Parameter



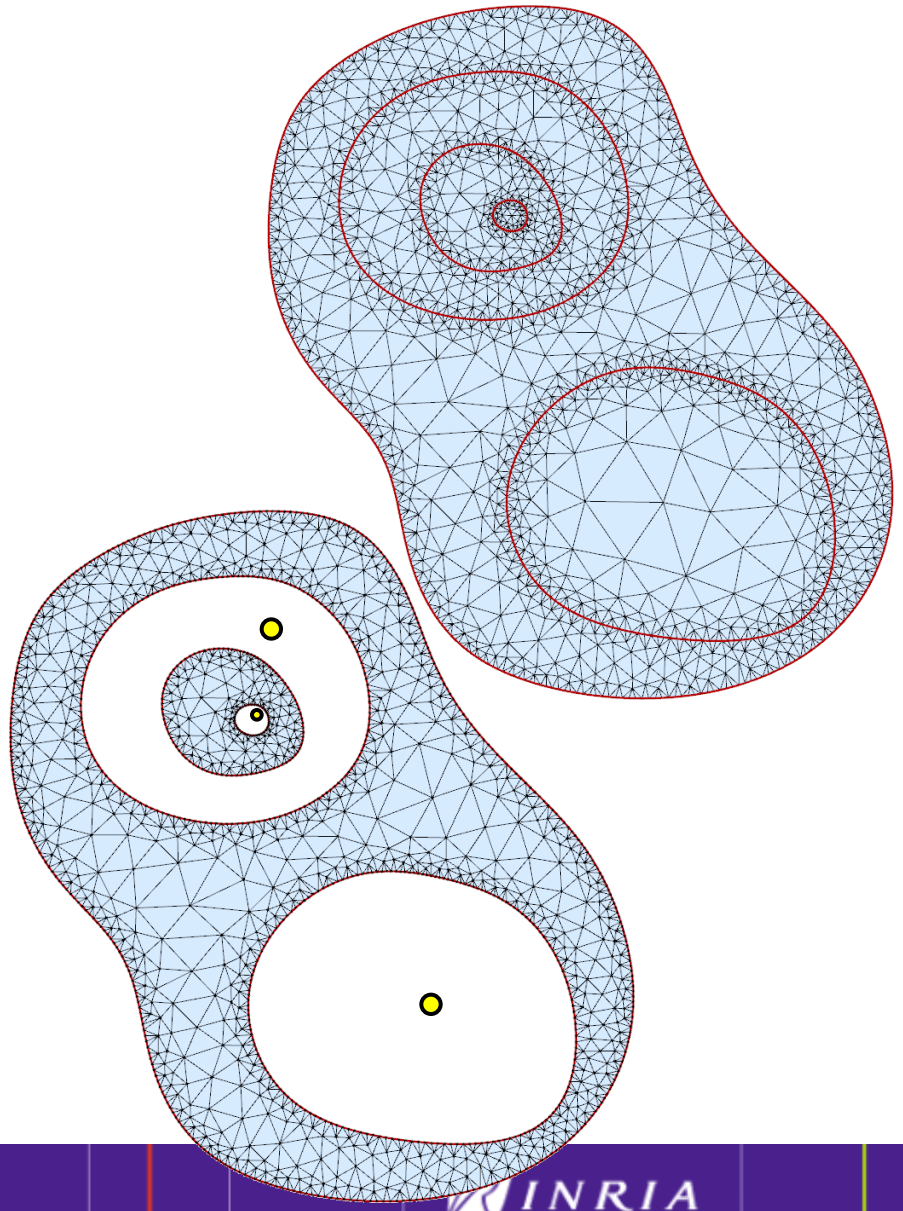
No constraint Uniform Sizing function

Example Code

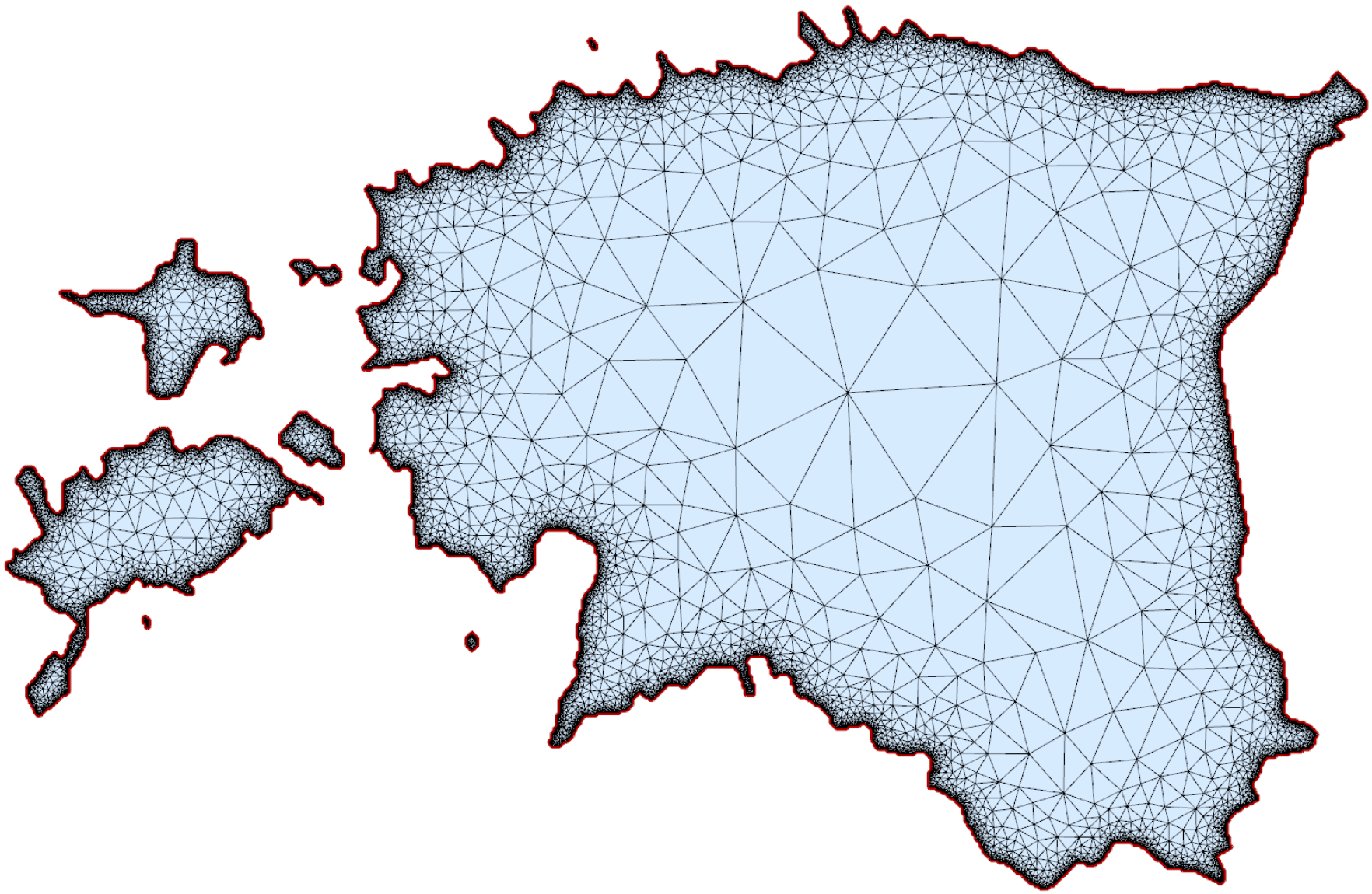
```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Constrained_Delaunay_triangulation_2.h>
#include <CGAL/Delaunay_mesher_2.h>
#include <CGAL/Delaunay_mesh_size_criteria_2.h>
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;
typedef CGAL::Constrained_Delaunay_triangulation_2<Kernel> CDT;
typedef CGAL::Delaunay_mesh_size_criteria_2<CDT> Criteria;
typedef CGAL::Delaunay_mesher_2<CDT, Criteria> Meshing_engine;
int main()
{
  CDT cdt;
  Meshing_engine engine(cdt);
  engine.refine_mesh();
  engine.set_criteria(Criteria(0.125, 0.5)); // min 20.6 deg
                                             // 0.5 for sizing
  engine.refine_mesh(); // refine once more, etc.
  return 0;
}
```

Parameters for Mesh Generation

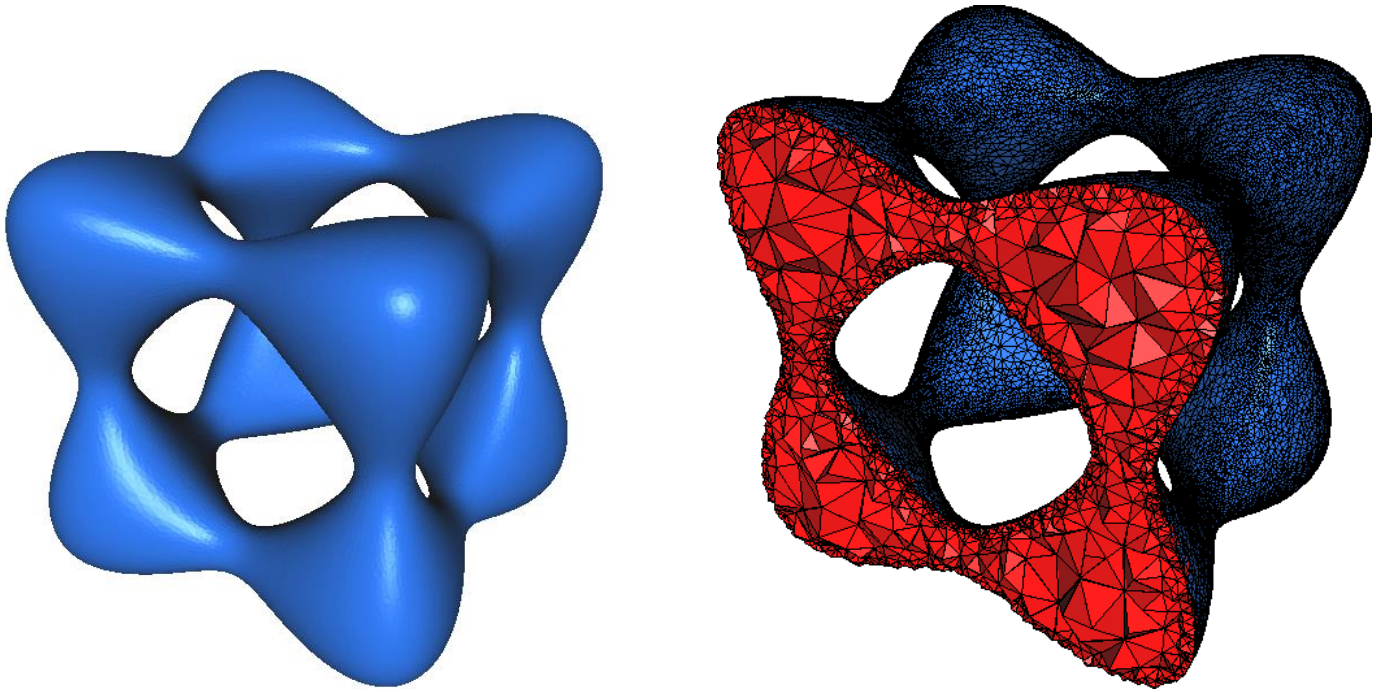
- Shape
  - Lower bound on triangle angles
- Size
  - No constraint
  - Uniform sizing
  - Sizing function
- **Seeds**
  - Exclude/include components



Example



Volume Mesh Generation in CGAL



Volume Mesh Generation Algorithm

```
repeat
{
    pick bad simplex
    if(Steiner point encroaches a facet)
        refine facet
    else
        refine simplex
    update Delaunay triangulation restricted to domain
}
until all simplices are good
Exude slivers
```

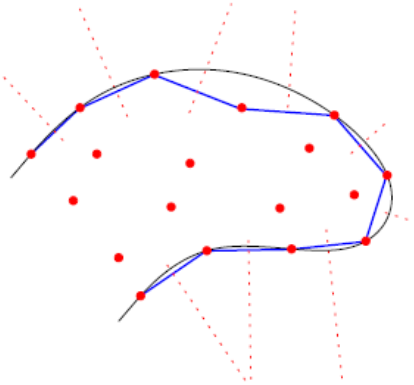
Delaunay Refinement

Apply the following rules with priority order

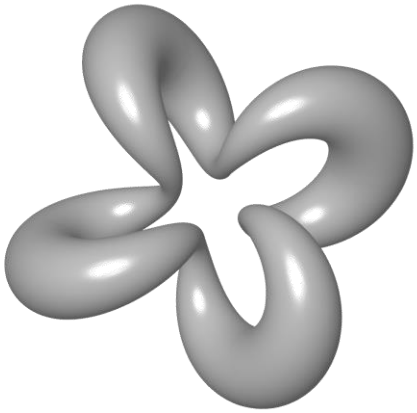
Rule 1: While there is a facet f in $\mathrm{Del}_{|\mathrm{bd}O}(\mathcal{P})$
with vertices $\notin \mathrm{bd}O$
refine_facet(f)

Rule 2: While there is a bad facet f in $\mathrm{Del}_{|\mathrm{bd}O}(\mathcal{P})$
refine_facet(f)

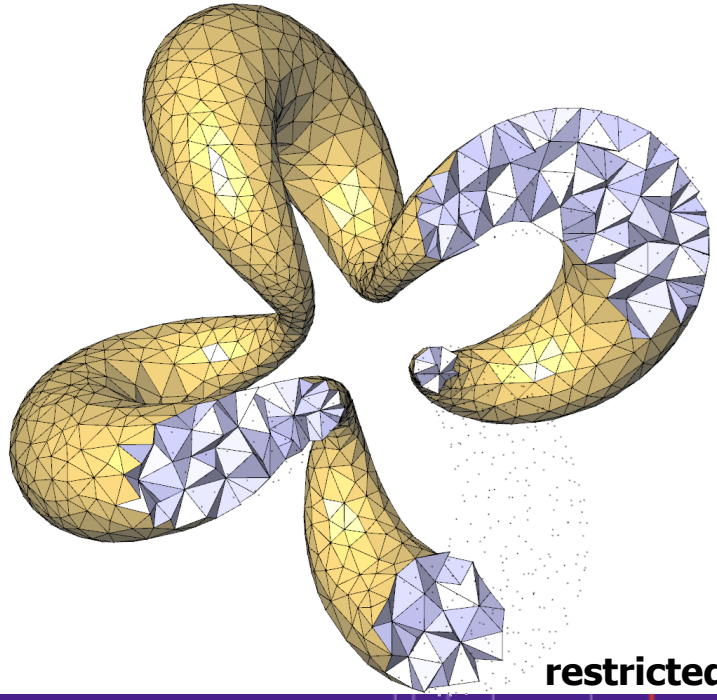
Rule 3: While there is a bad tetrahedron t in $\mathrm{Del}_{|O}(\mathcal{P})$
refine_tetrahedron_or_facet(t)



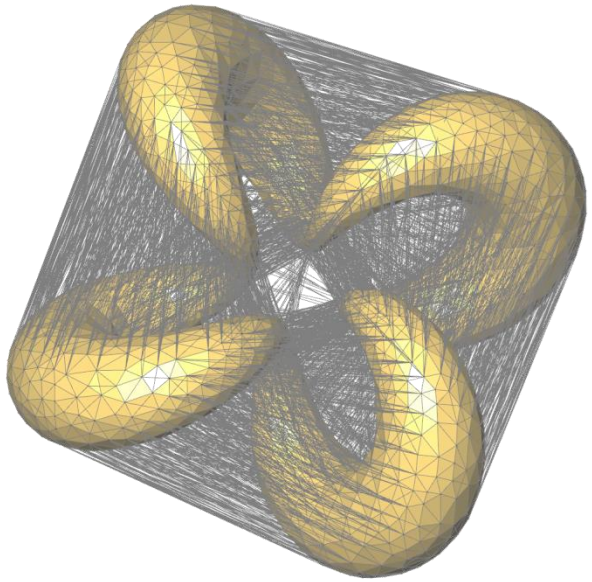
Delaunay Filtering



domain boundary

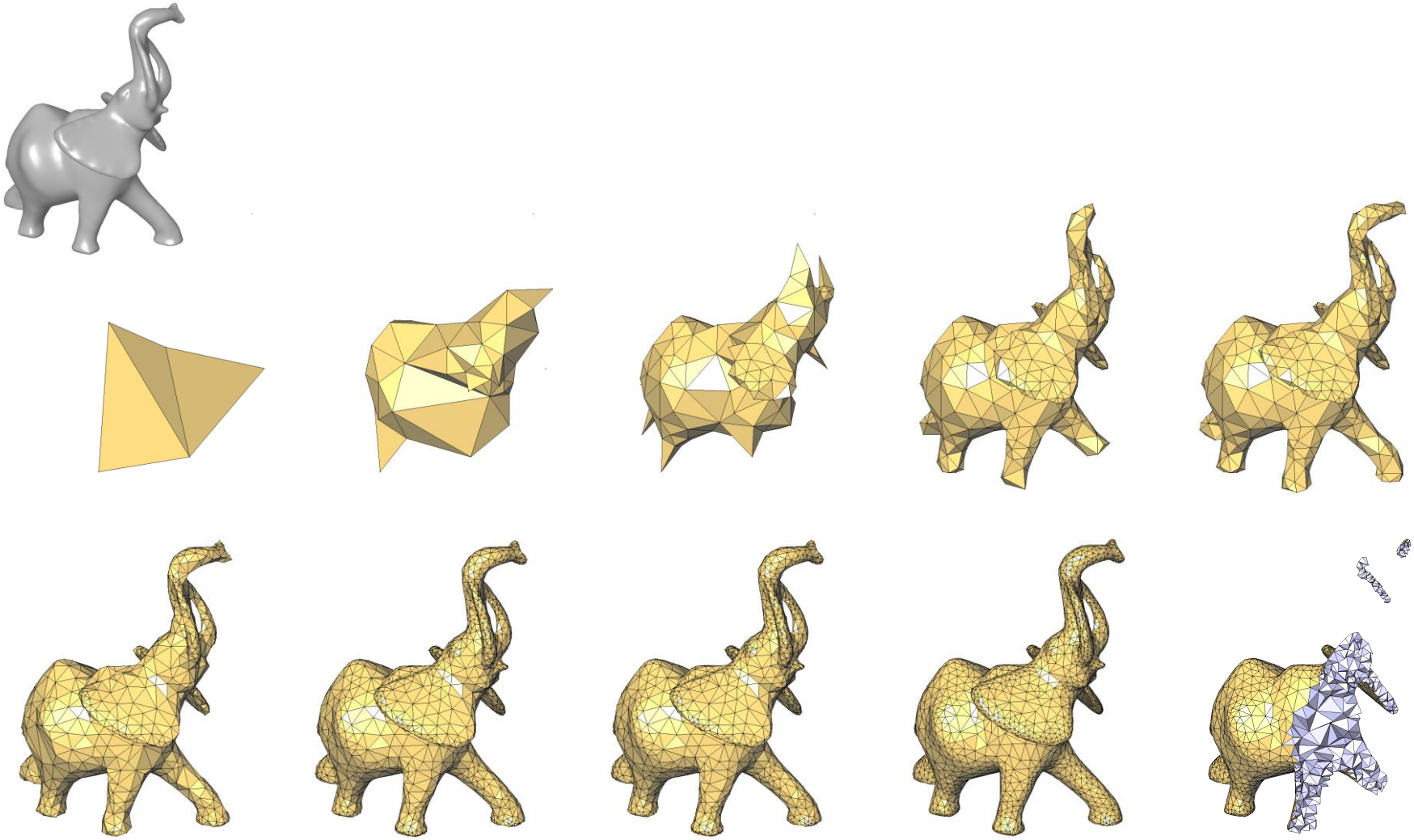


restricted Delaunay triangulation

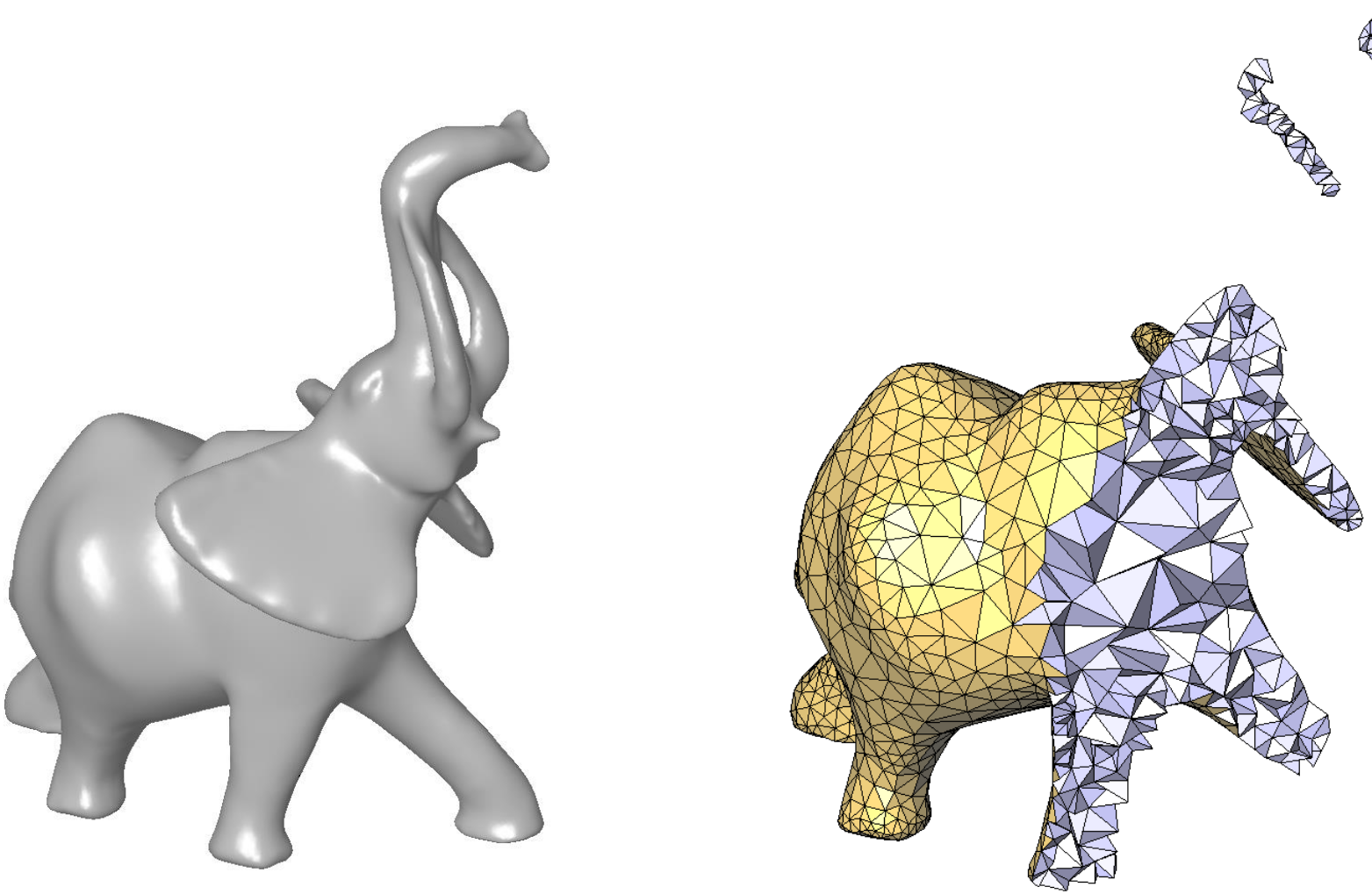


3D complex embedded in a 3D triangulation

Example

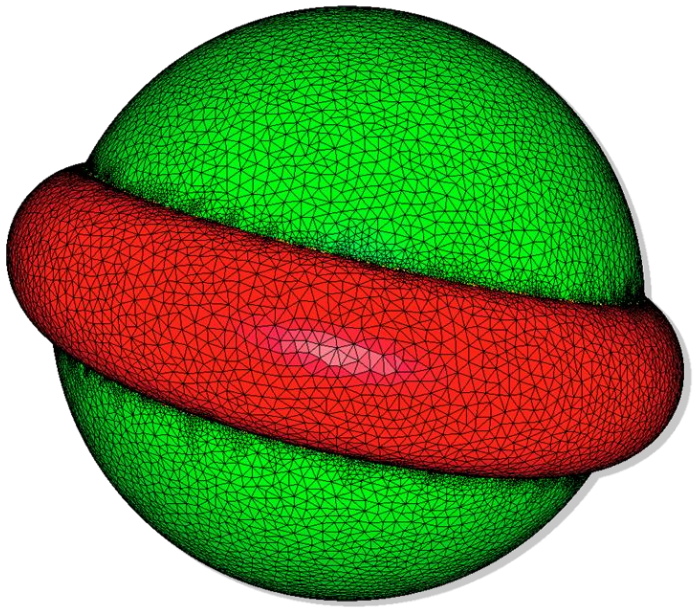


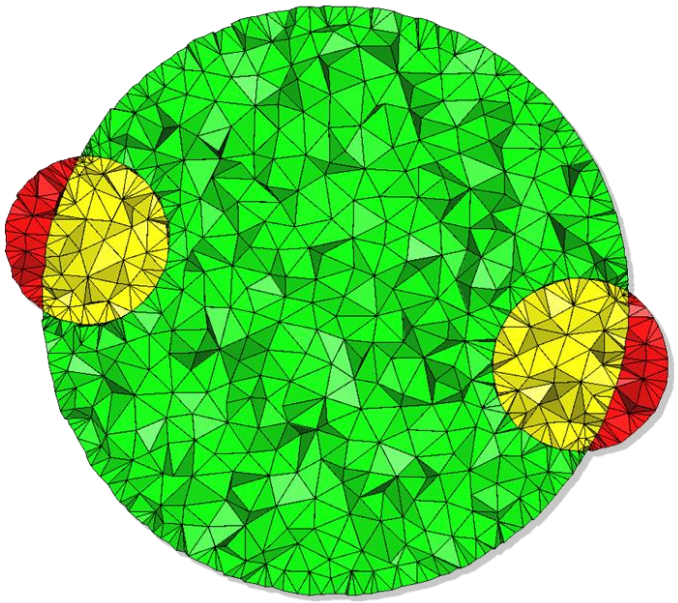
Example



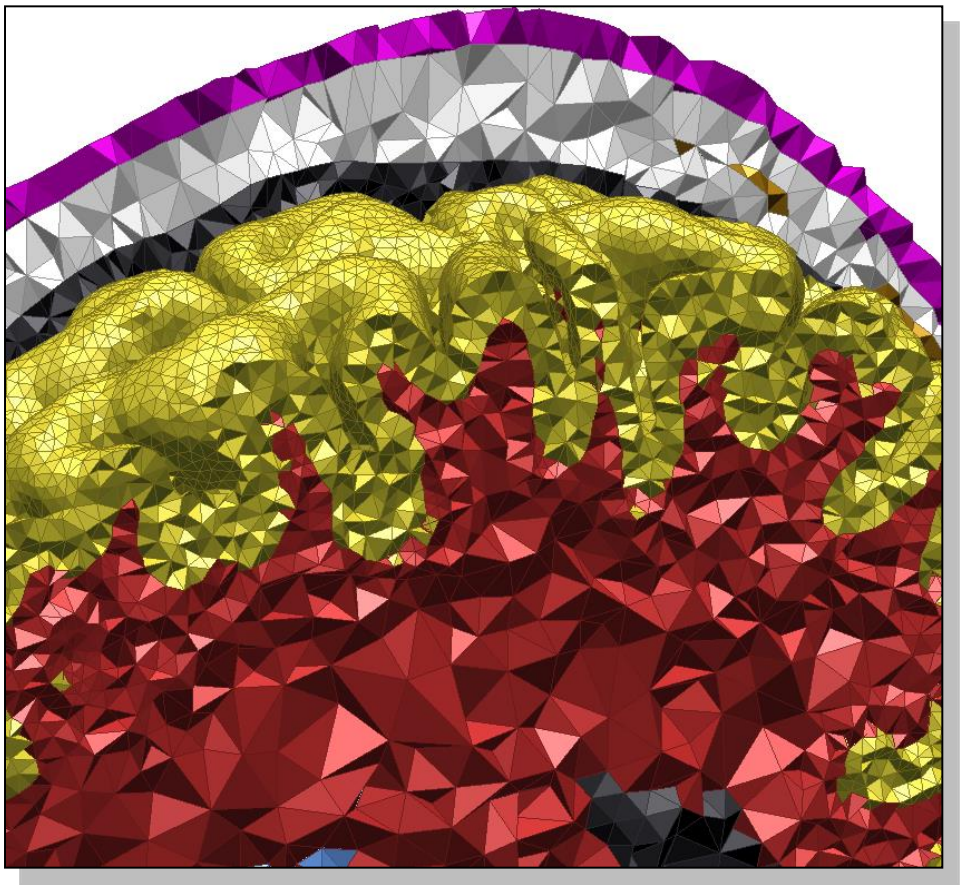
Mesh from Implicit Function

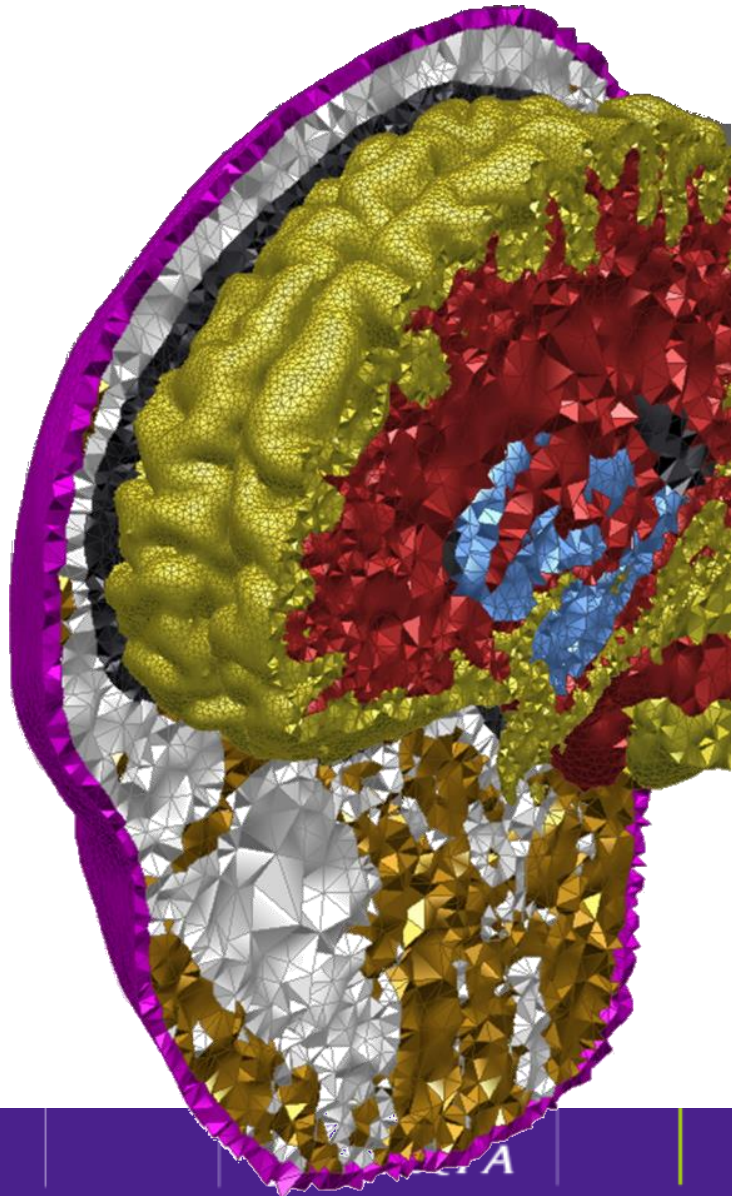
- 38k vertices
- 168k tetrahedra
- Mesh generation: 8s



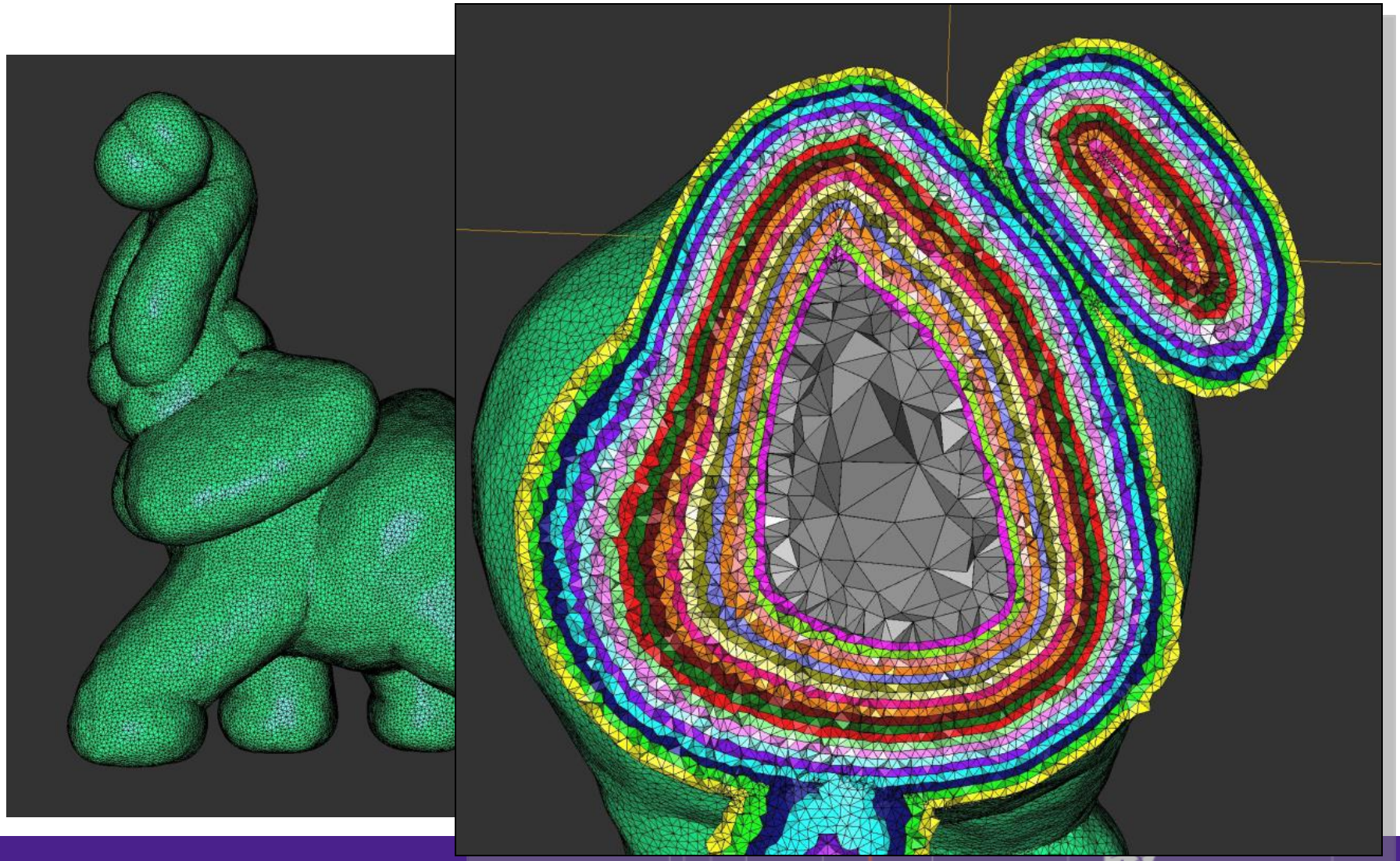


Multi-Domain Volume Mesh

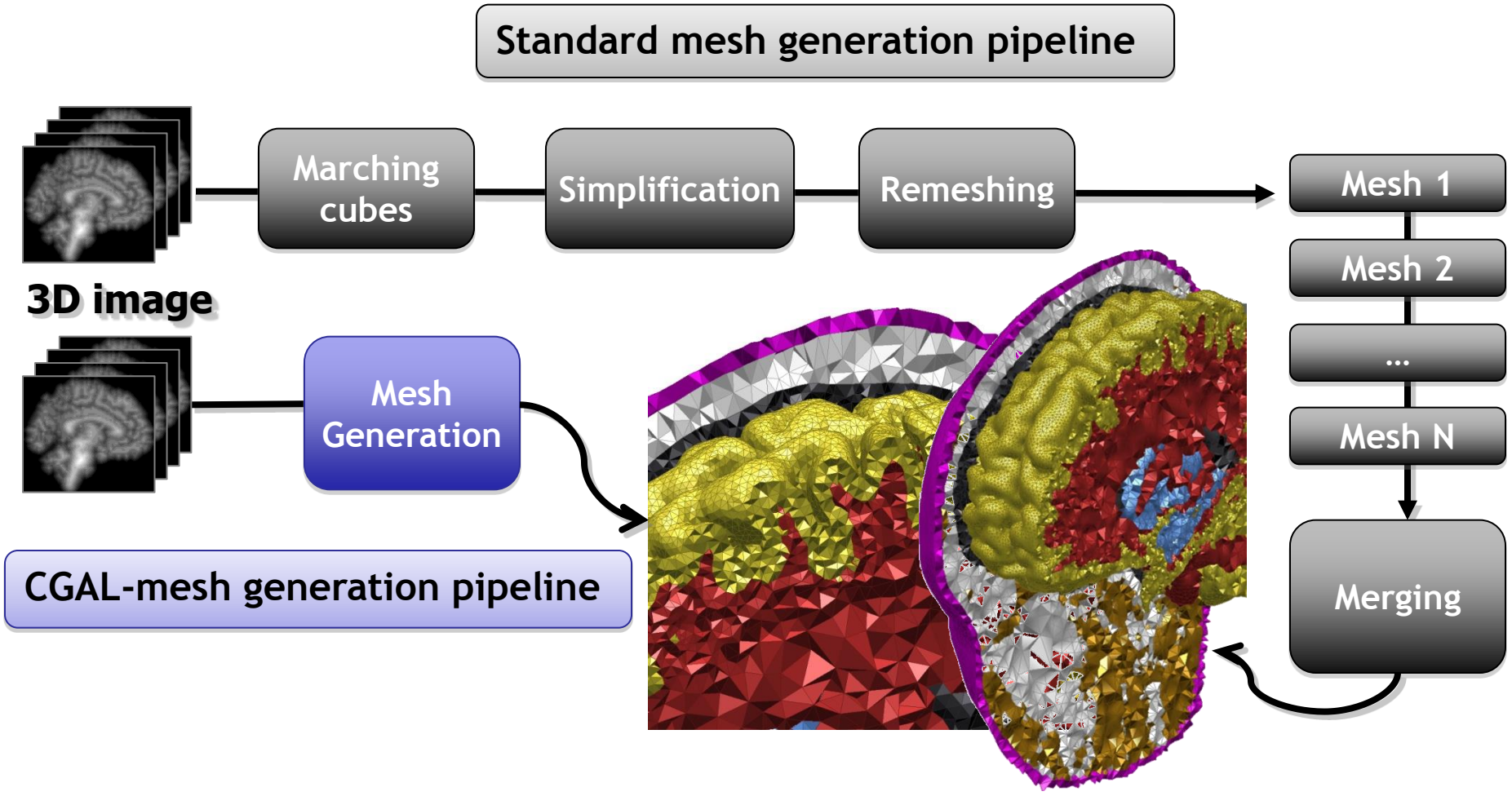




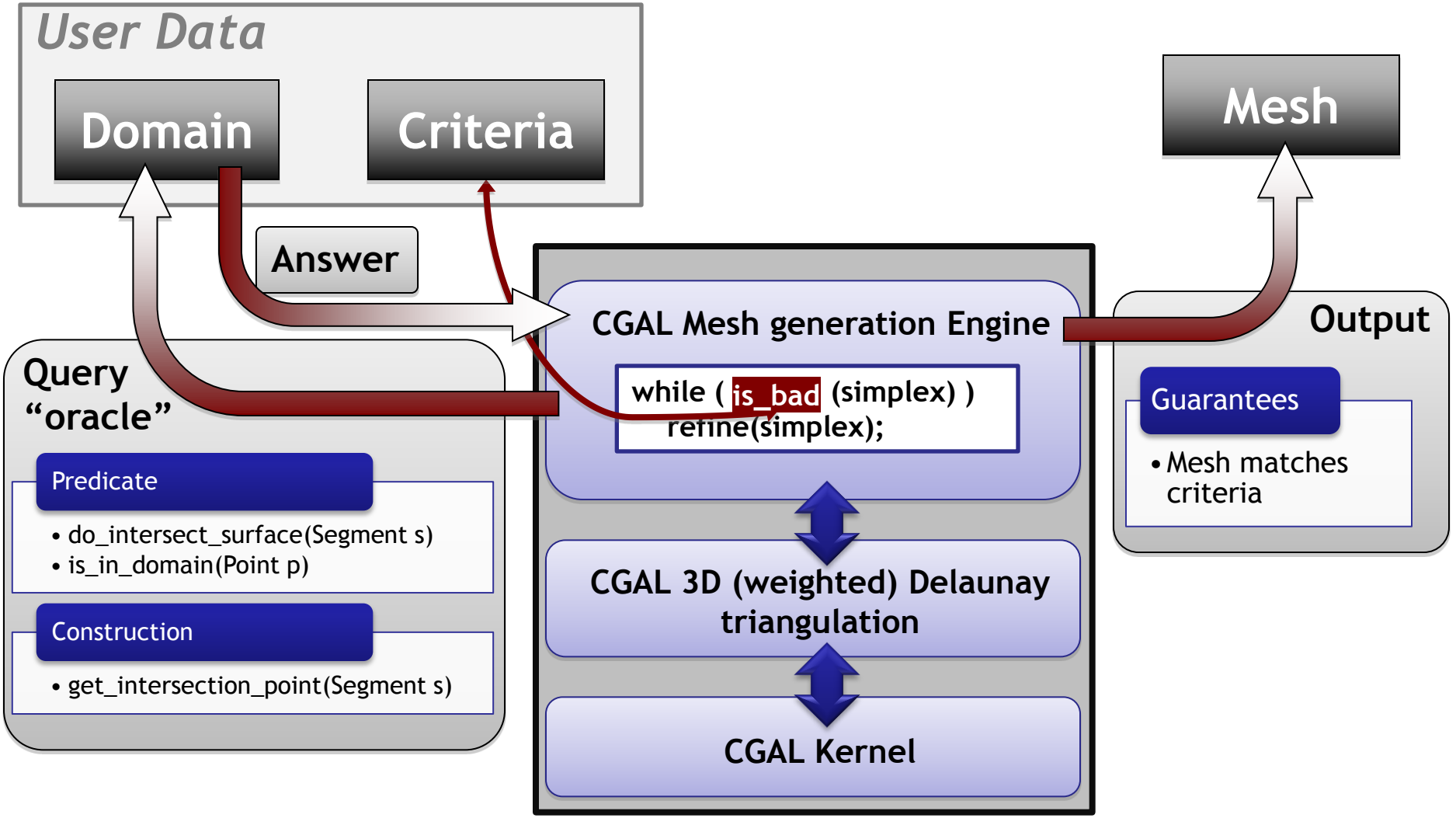
Polyhedral Multi-Domain



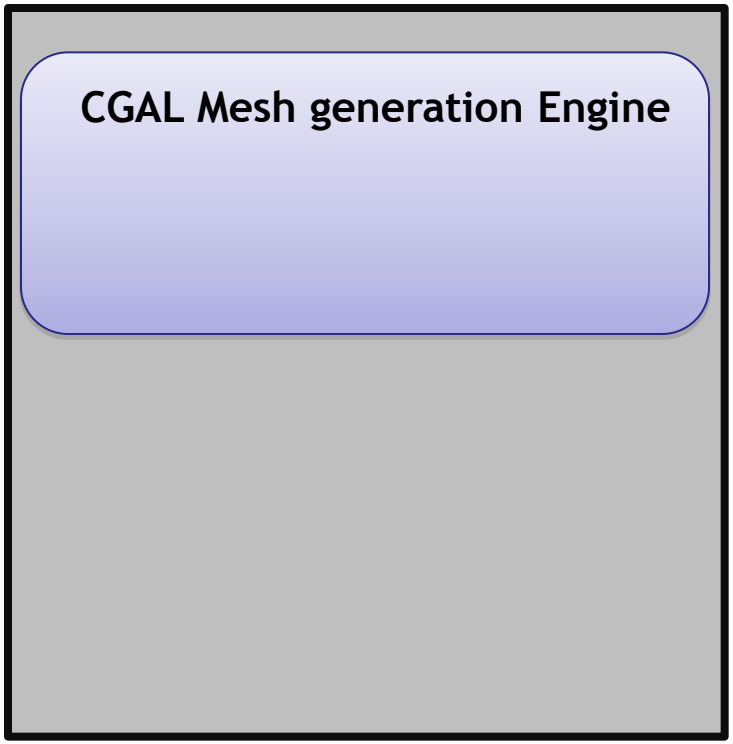
Added Value: Shortened Pipeline



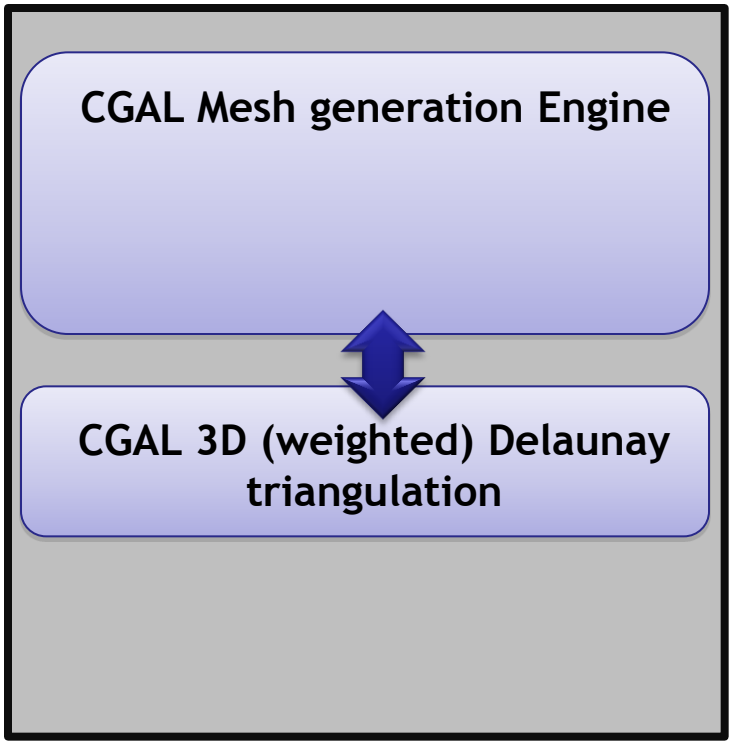
Overall Design



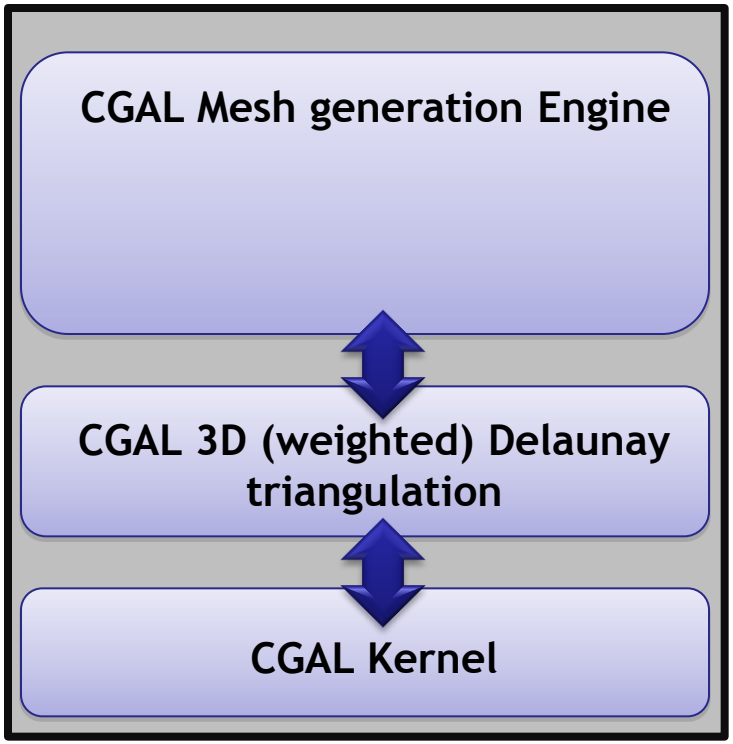
Overall Design



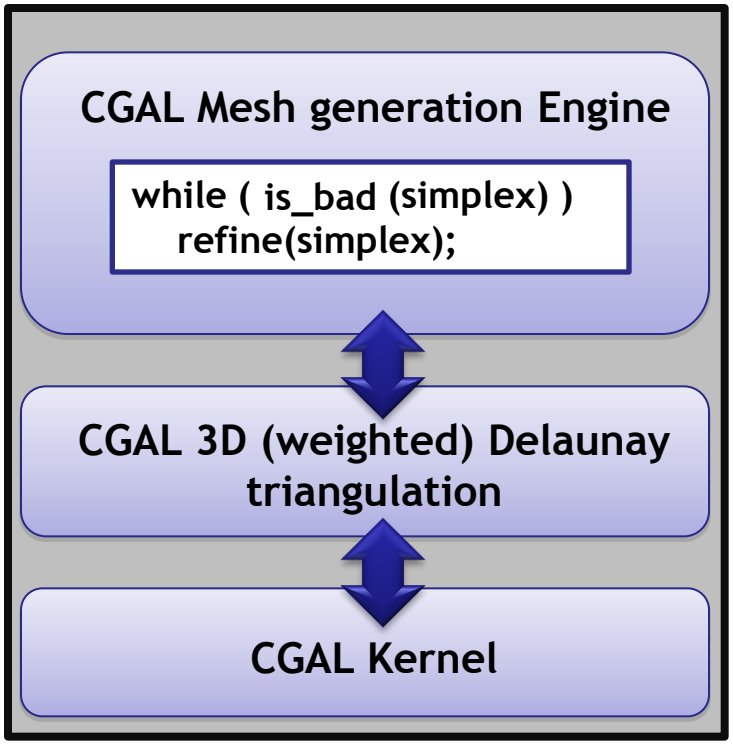
Overall Design



Overall Design



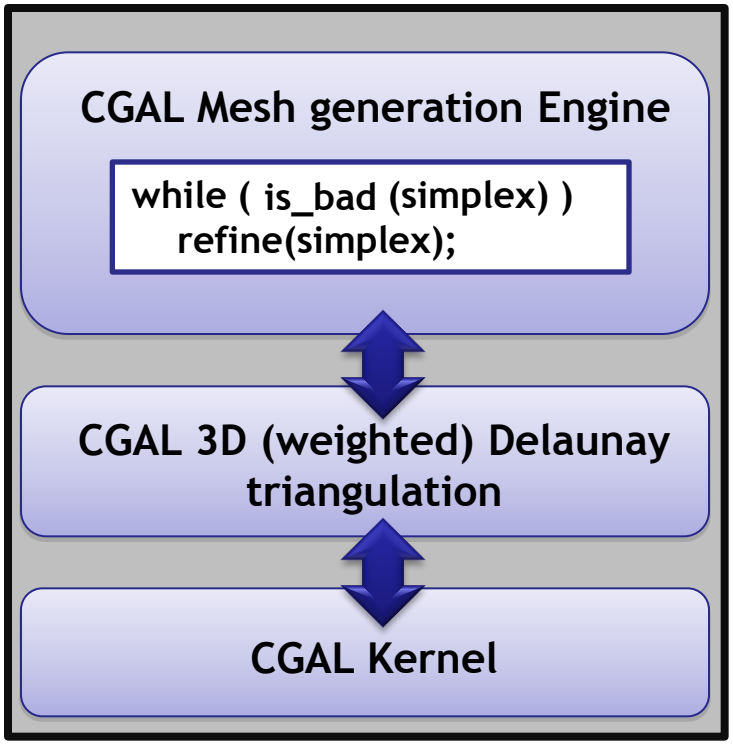
Overall Design



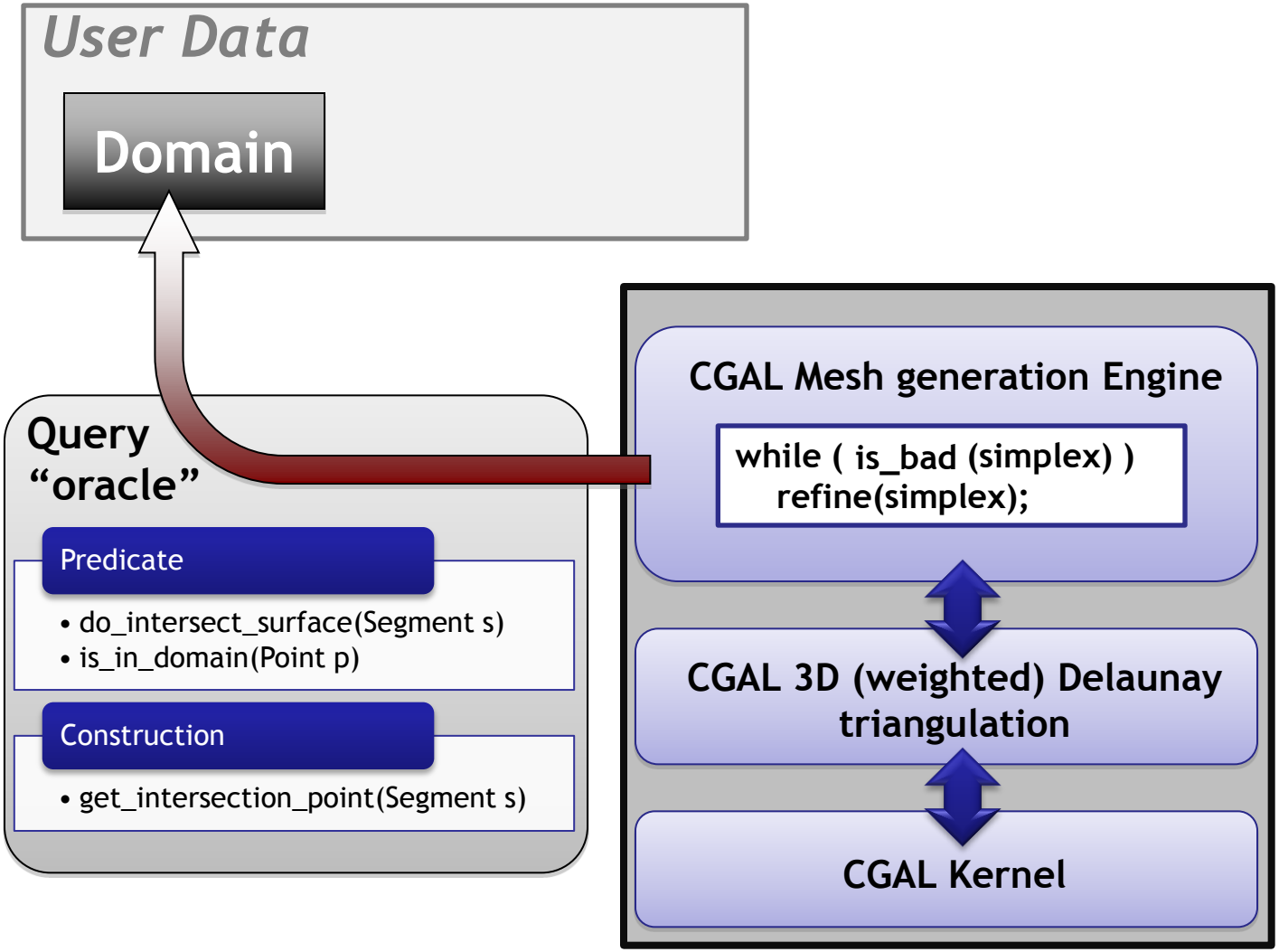
Overall Design

User Data

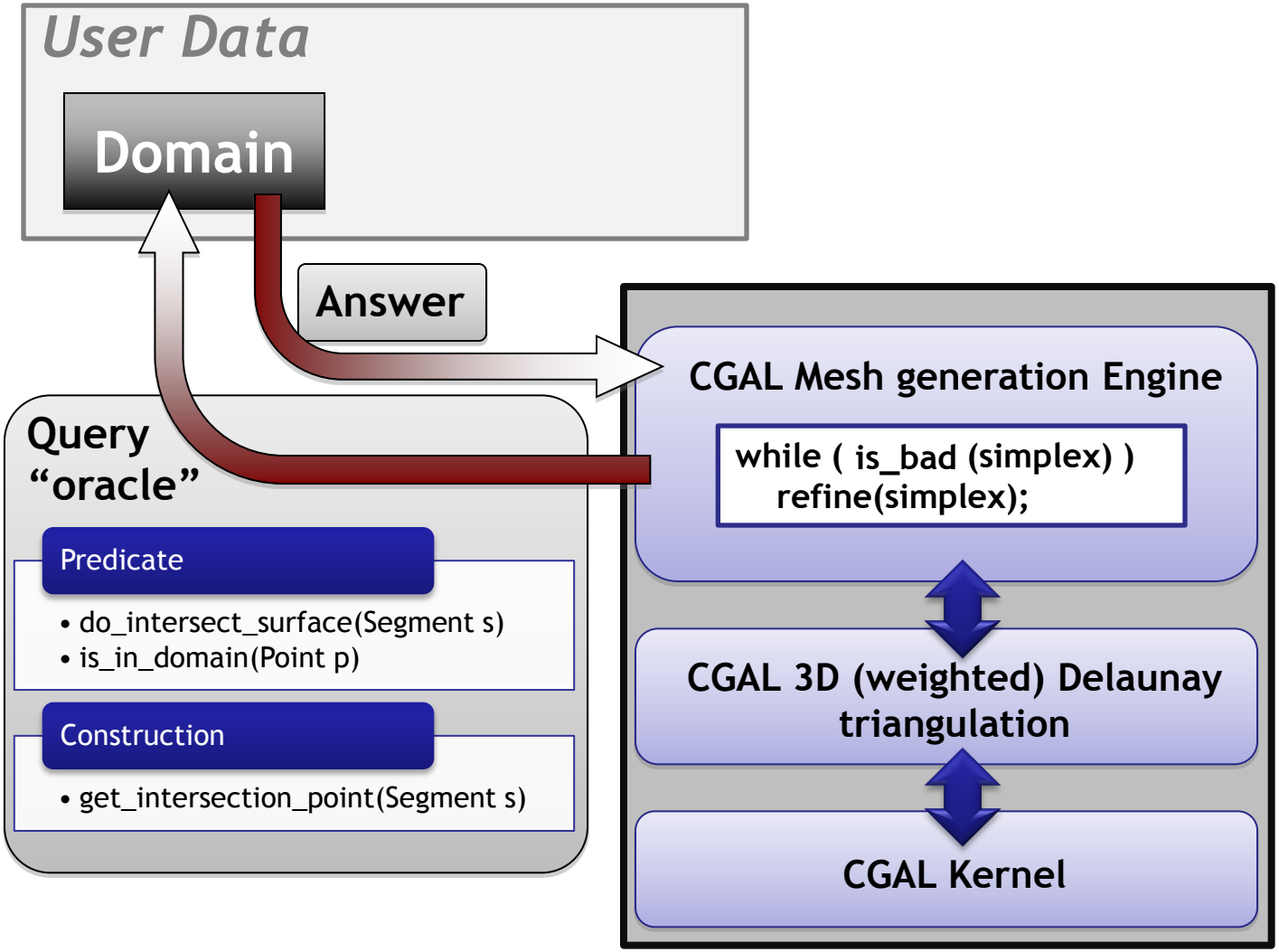
Domain



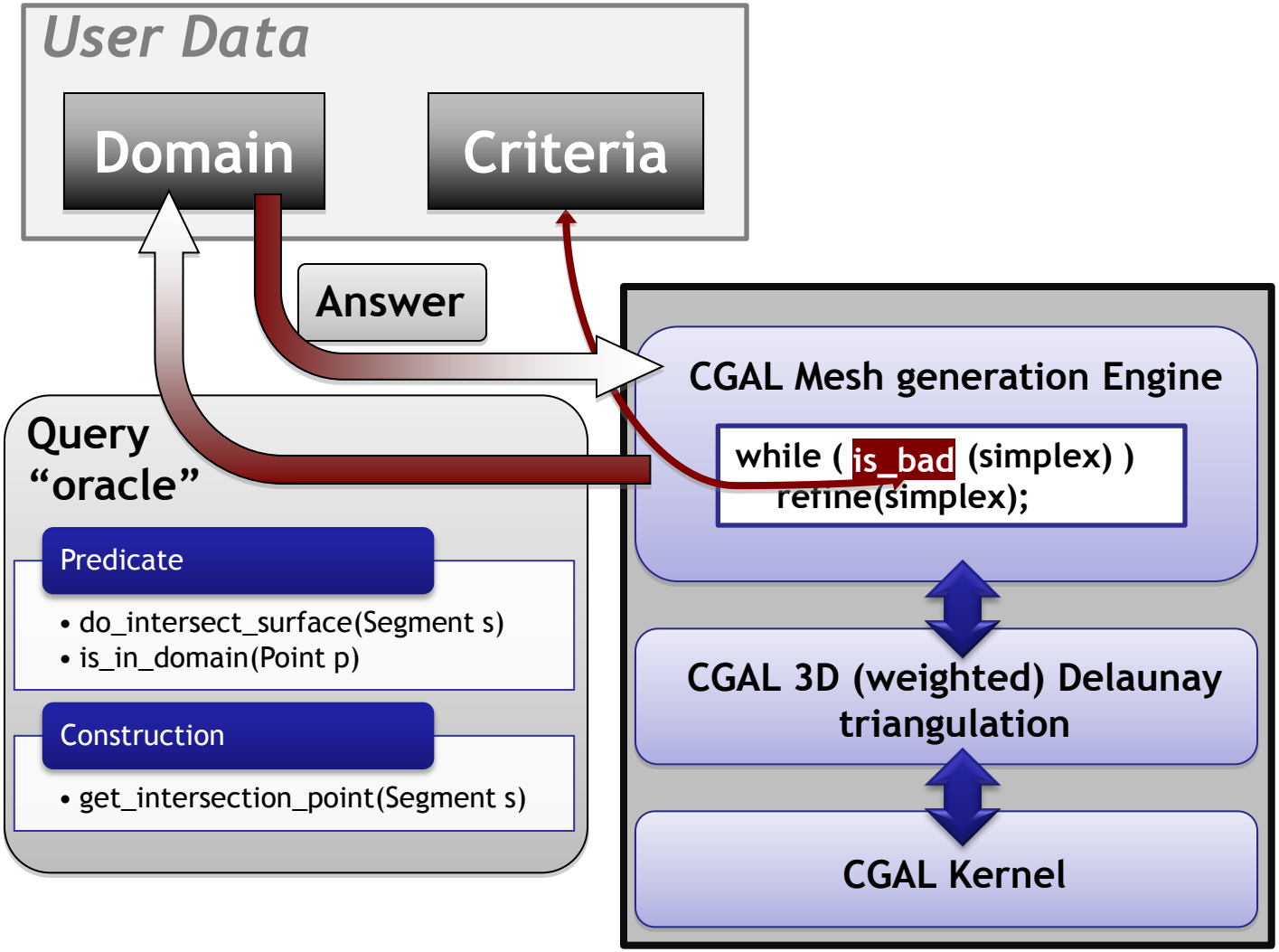
Overall Design



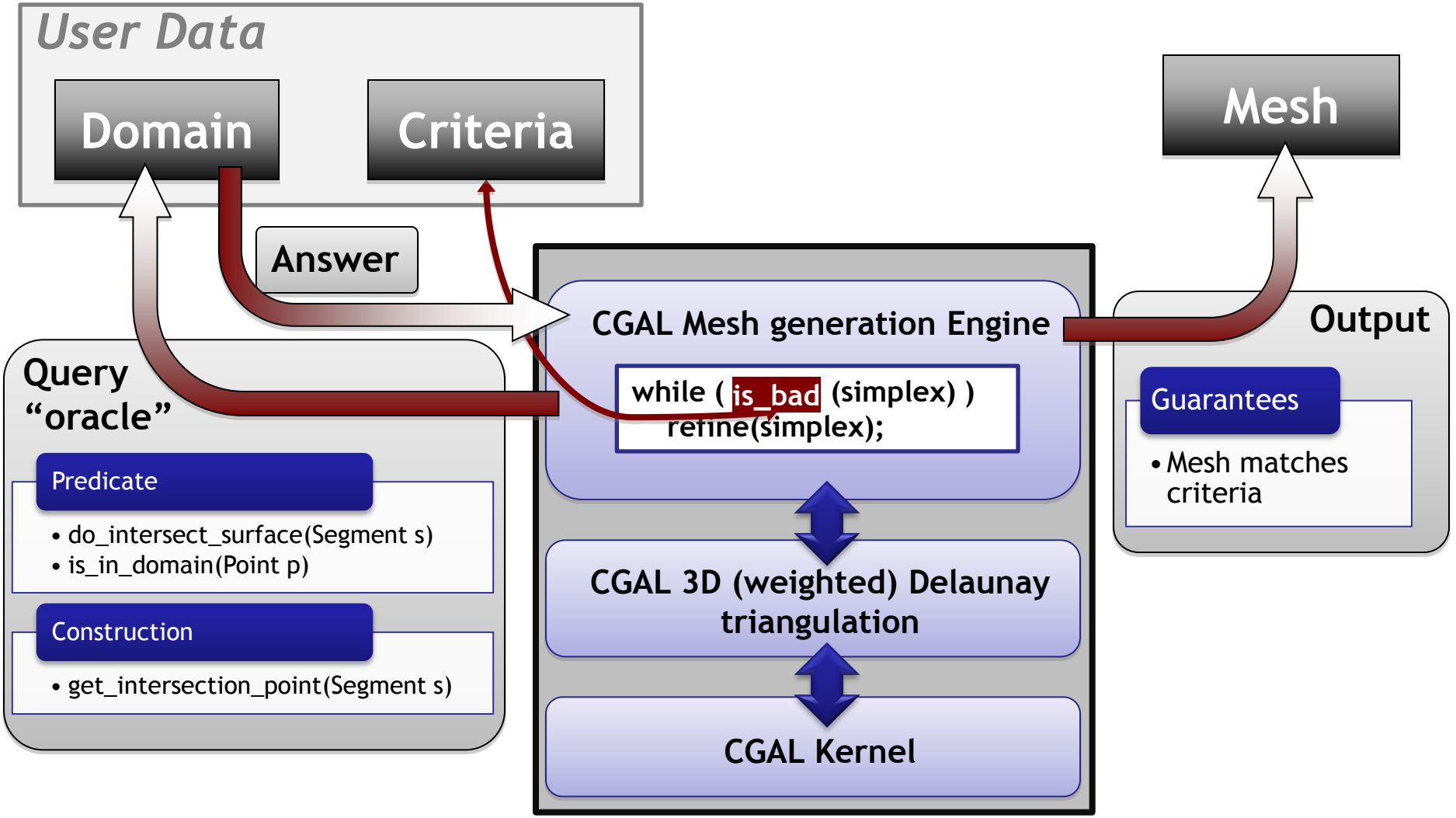
Overall Design



Overall Design



Overall Design



Main function

template <class C3T3,class MeshDomain_3, class MeshCriteria>

C3T3 **make_mesh_3** (MeshDomain_3 domain, MeshCriteria criteria)

C3T3: 3D complex embedded in a 3D triangulation

Simple Code Example

```
FT sphere_function(const Point& p)
{ return CGAL::squared_distance(p, Point(CGAL::ORIGIN))-
    1; }

Mesh_domain domain(sphere_function,
    K::Sphere_3(CGAL::ORIGIN, 2.));

Mesh_criteria criteria(facet_angle=30, facet_size=0.1,
    facet_distance=0.025,
    cell_radius_edge=2, cell_size=0.1);

C3t3 c3t3 = CGAL::make_mesh_3<C3t3>(domain, criteria);
```