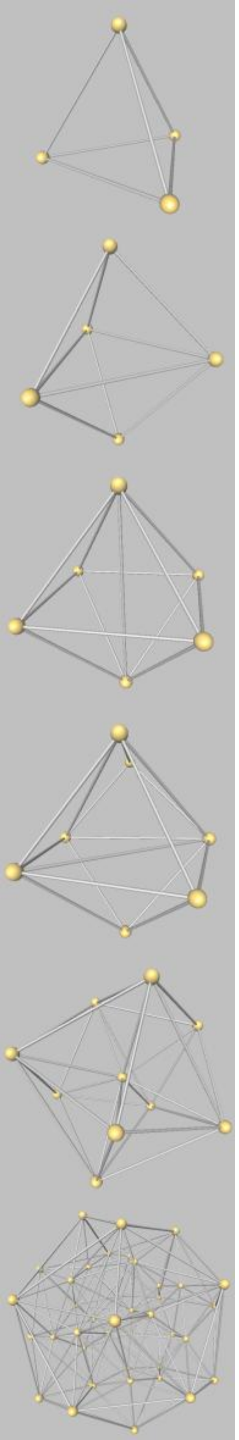
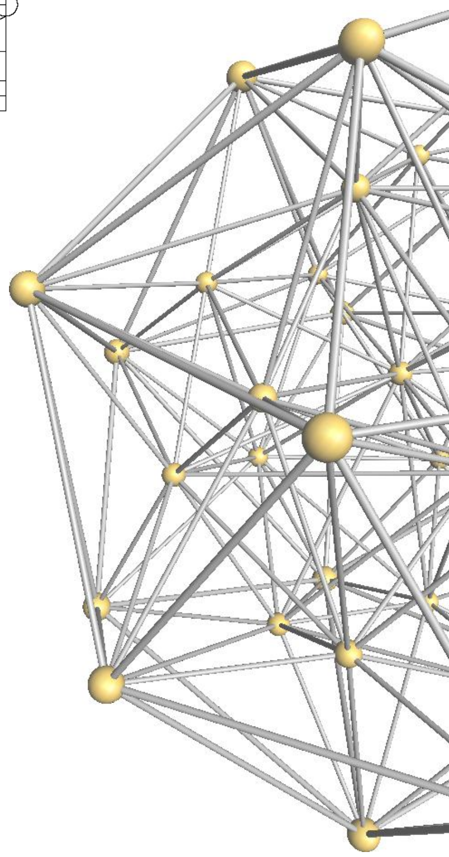


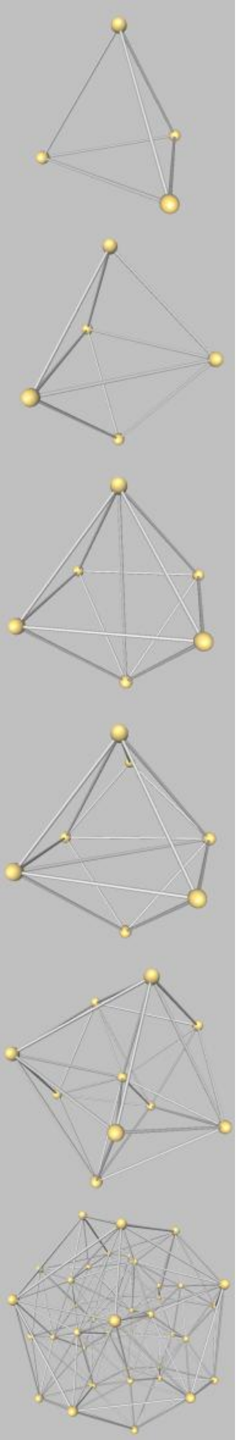
3D Triangulations in CGAL

Pierre Alliez

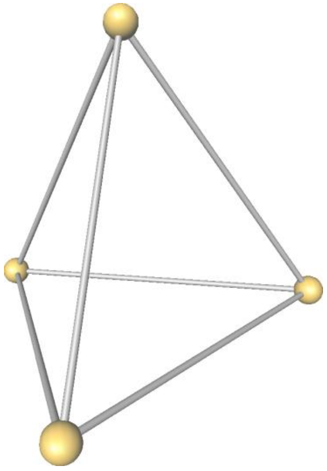
inria informatics mathematics



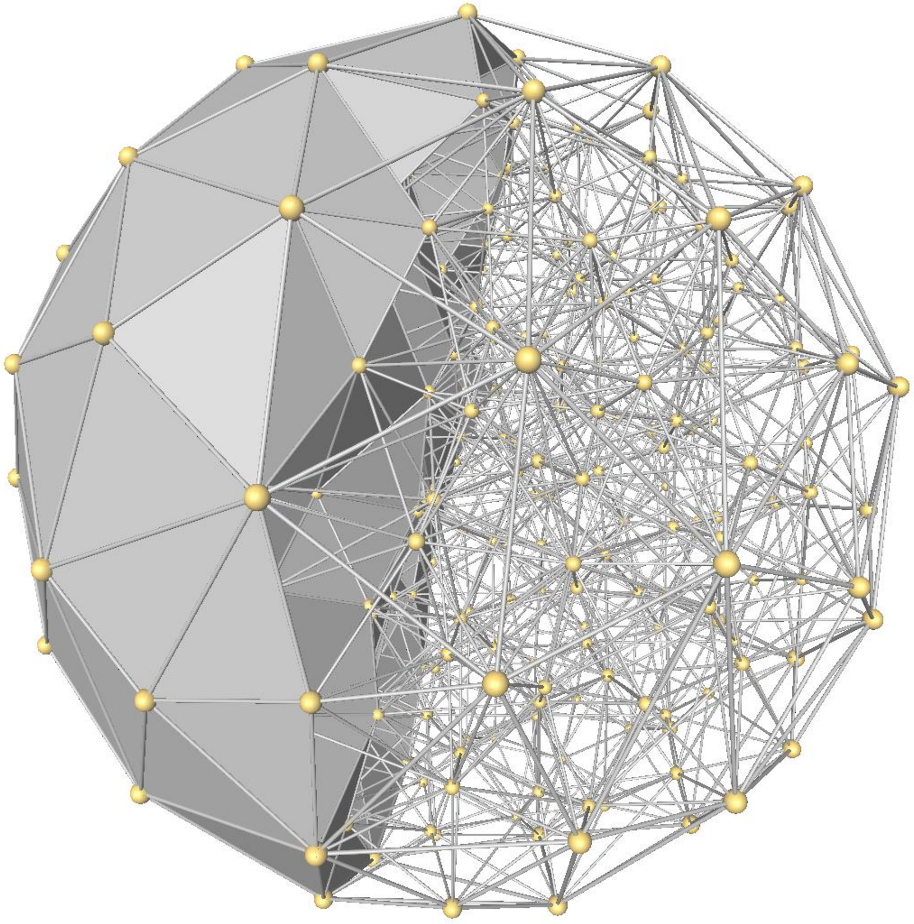
http://www.cgal.org

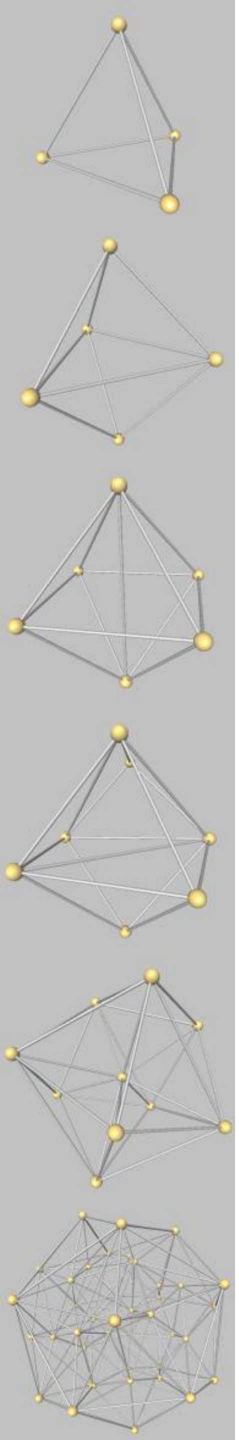


3D Triangulation



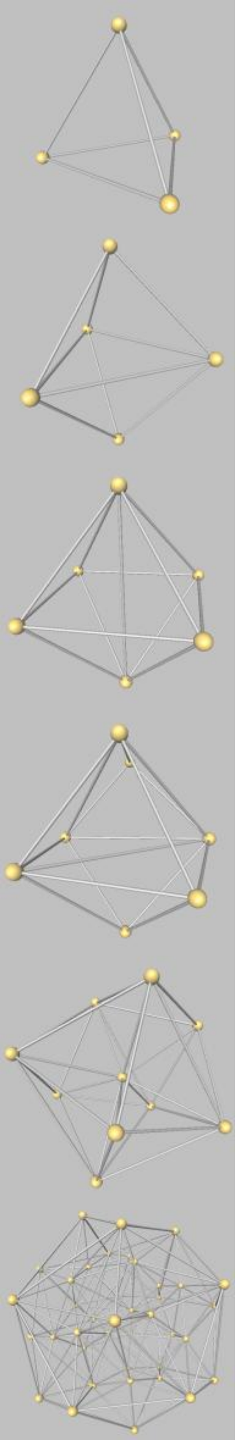
Tetrahedron





Outline

- Definitions
- Representation
- Types of Triangulations
- Software Design
  - Geometric Traits
  - Customization
- Examples
- Applications

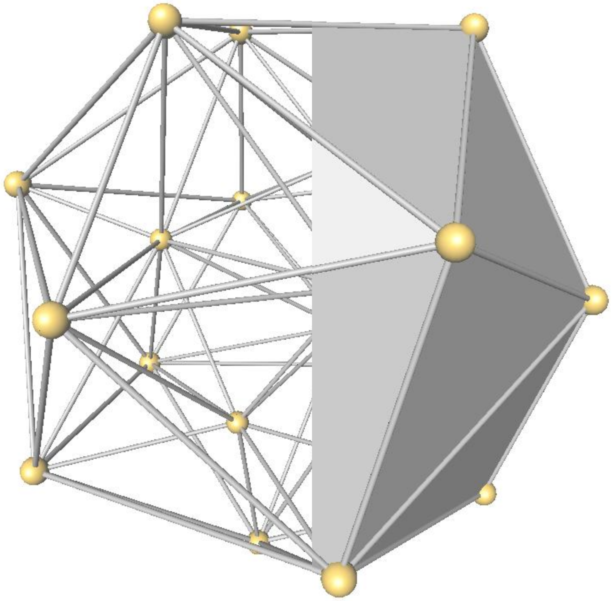


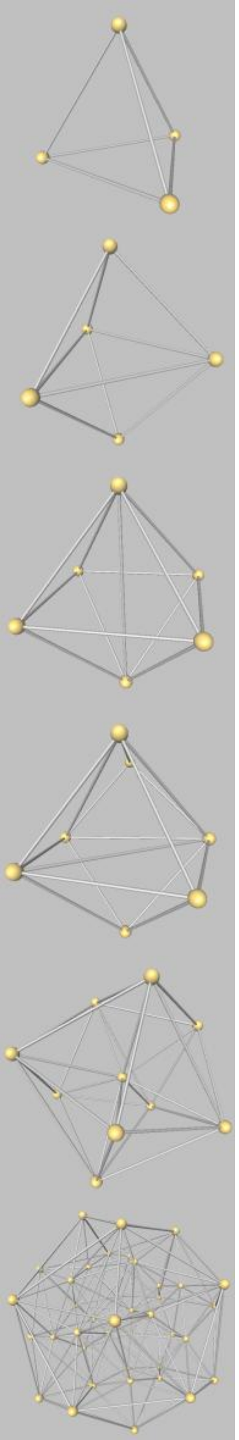
Outline

- Definitions
- Representation
- Types of Triangulations
- Software Design
  - Geometric Traits
  - Customization
- Examples
- Applications

Definitions

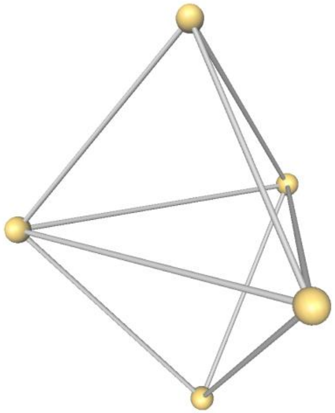
3D Triangulation of a point set A: partition of the convex hull of A into tetrahedra.





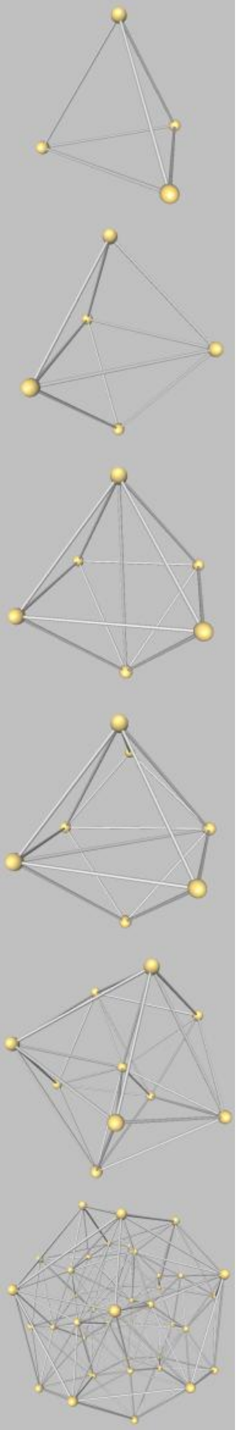
Definitions

- The cells (3-faces) are such that two cells either do not intersect or share a common facet (2-face), edge (1-face) or vertex (0-face).



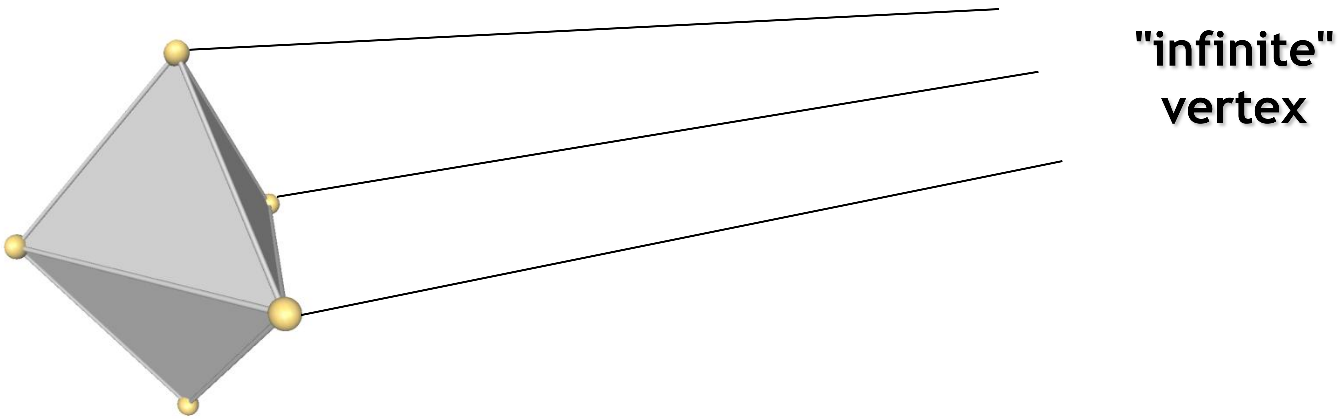
Outline

- Definitions
- Representation
- Types of Triangulations
- Software Design
  - Geometric Traits
  - Customization
- Examples
- Applications



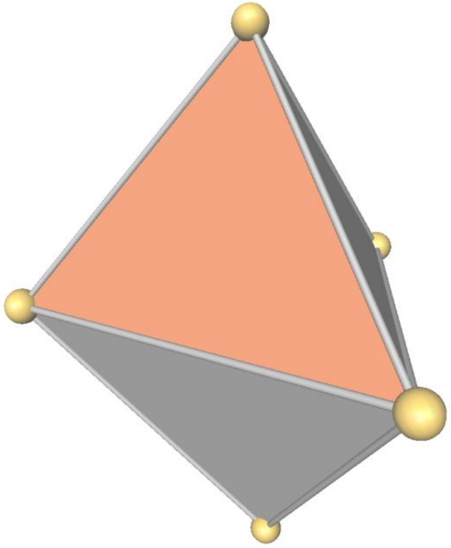
Representation

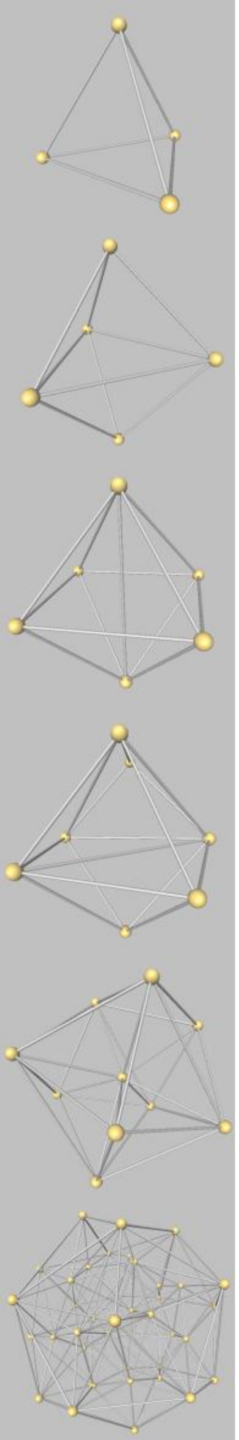
- **CGAL Triangulation:** partition of IR^3 (add unbounded cell).
- To deal only with Tetrahedra:
  - subdivide unbounded cell into tetrahedra.



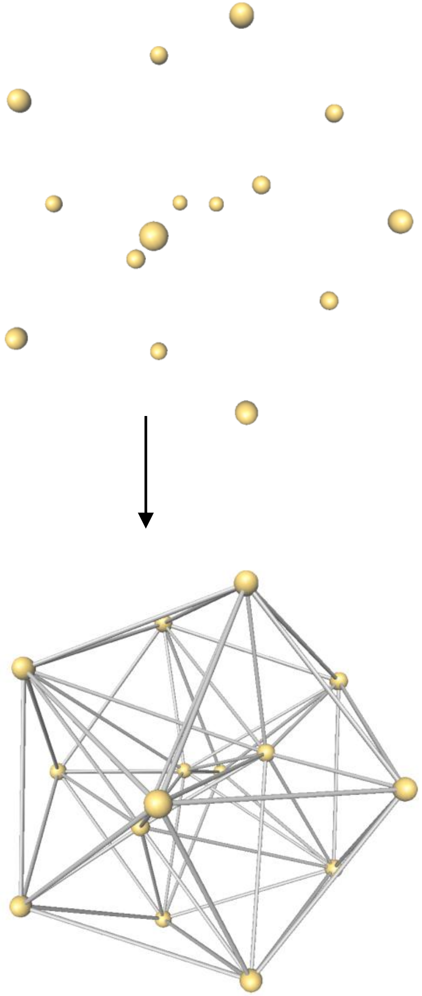
Representation

- Dealing only with **Tetrahedra**:
  - each facet incident to exactly two cells.





3D Triangulation in CGAL



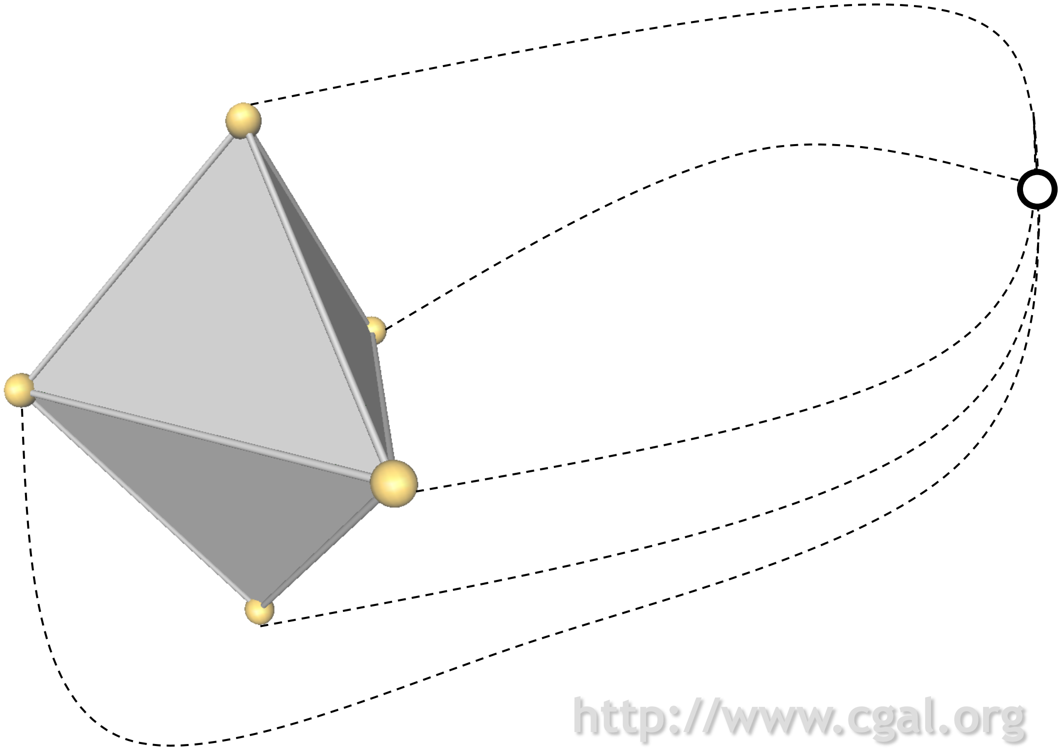
Input point set

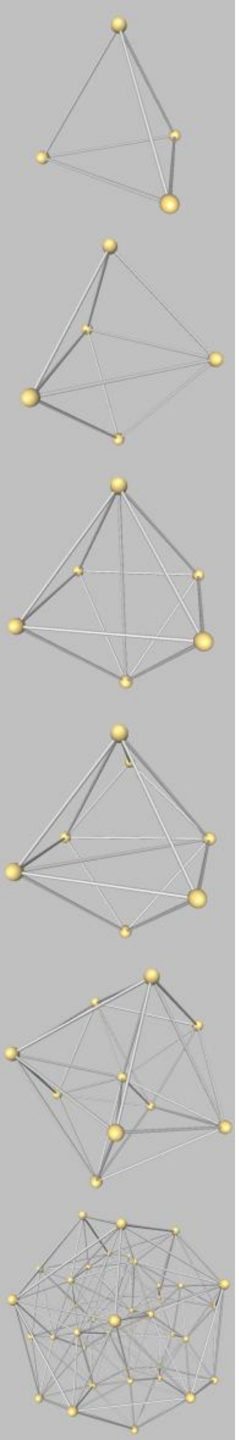
Triangulation

- Finite tetrahedra
- One infinite vertex
- Infinite tetrahedra

Infinite Vertex?

- No valid coordinates
  - no predicates applicable

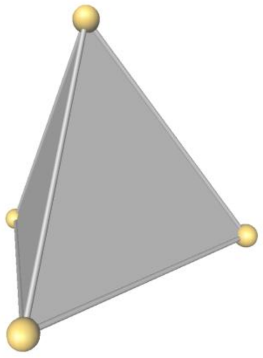




Internal Representation

Collection of **vertices** and **cells** linked together through incidence and adjacency relations (faces and edges are not explicitly represented).

vertices

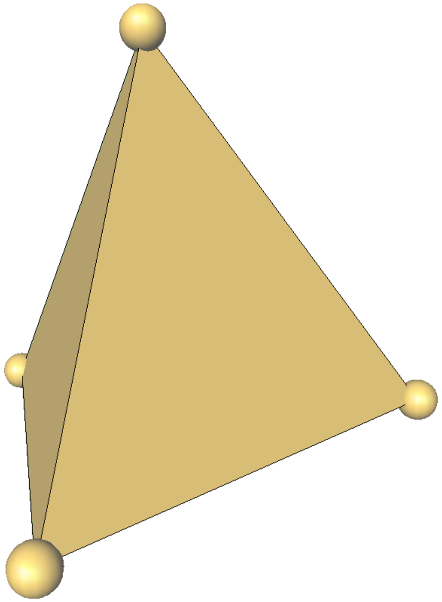


cell

Cell

Access to:

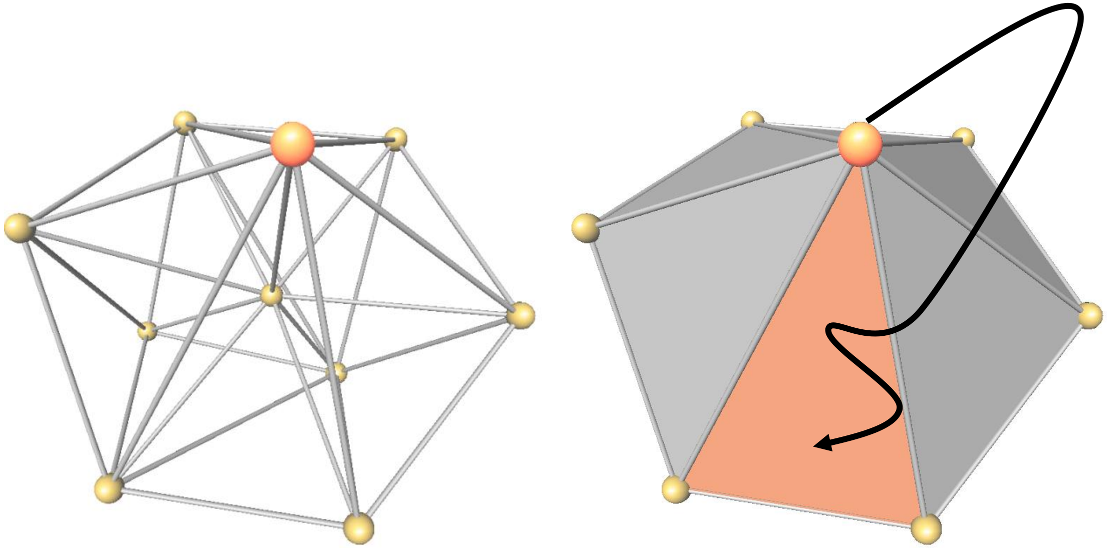
- 4 incident **vertices**
- 4 adjacent **cells**

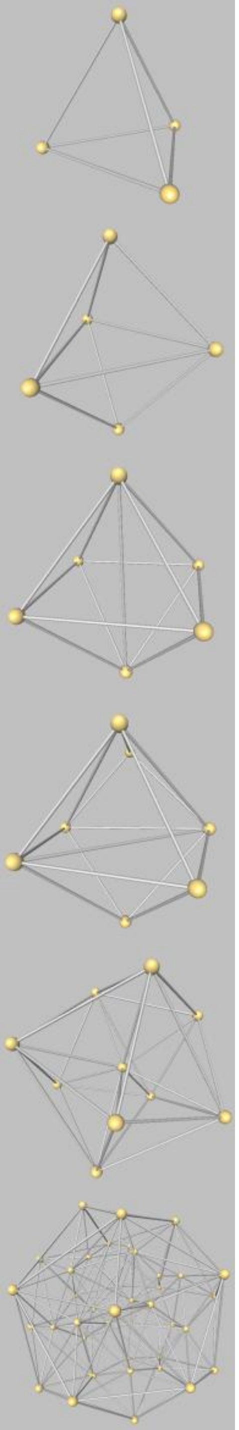


Vertex

Access to:

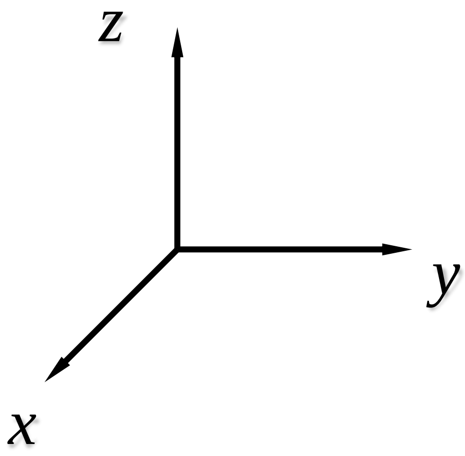
- **1** incident cell

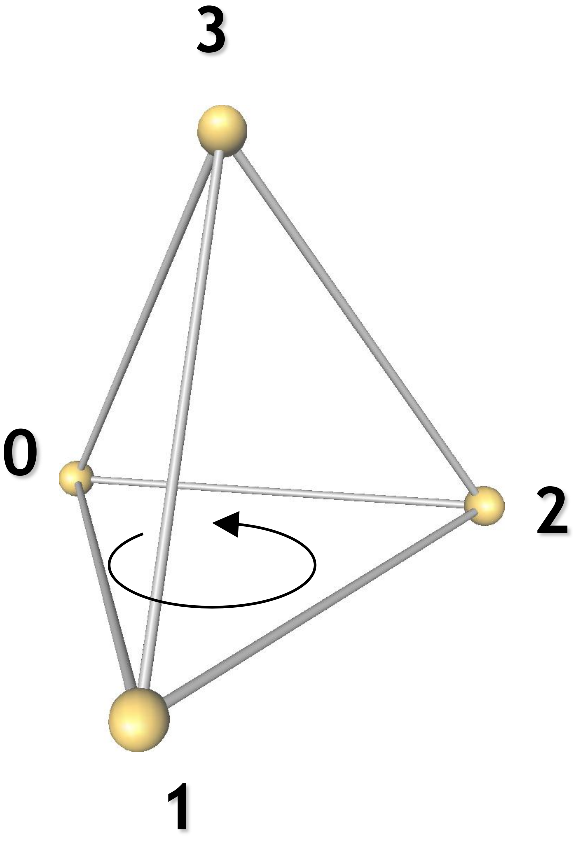




Orientation of a Cell

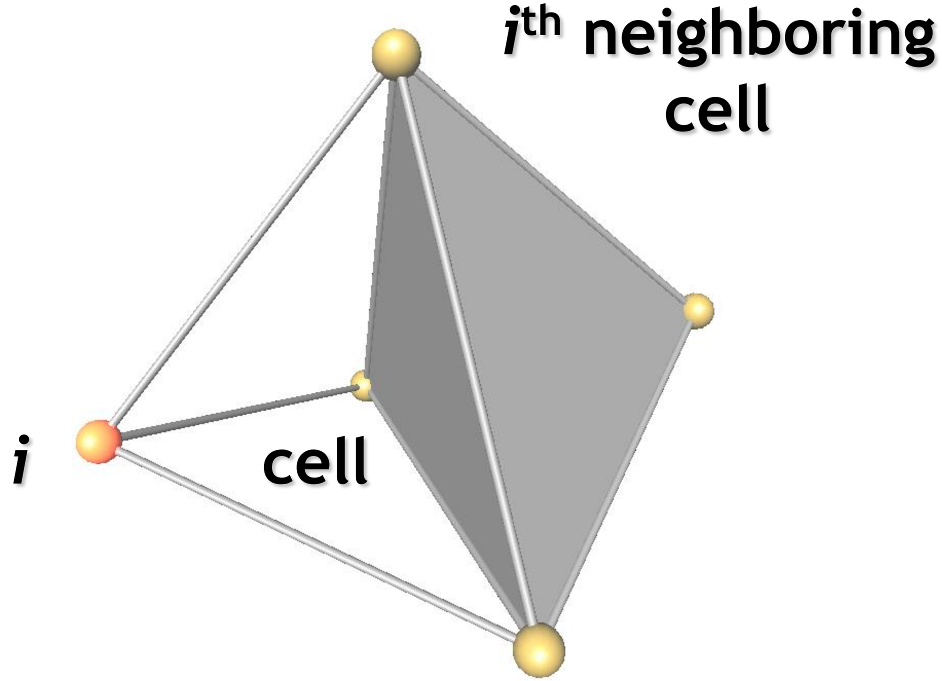
4 vertices indexed in **positive orientation** (induced by underlying Euclidean space IR^3).





Adjacent Cell Indexing

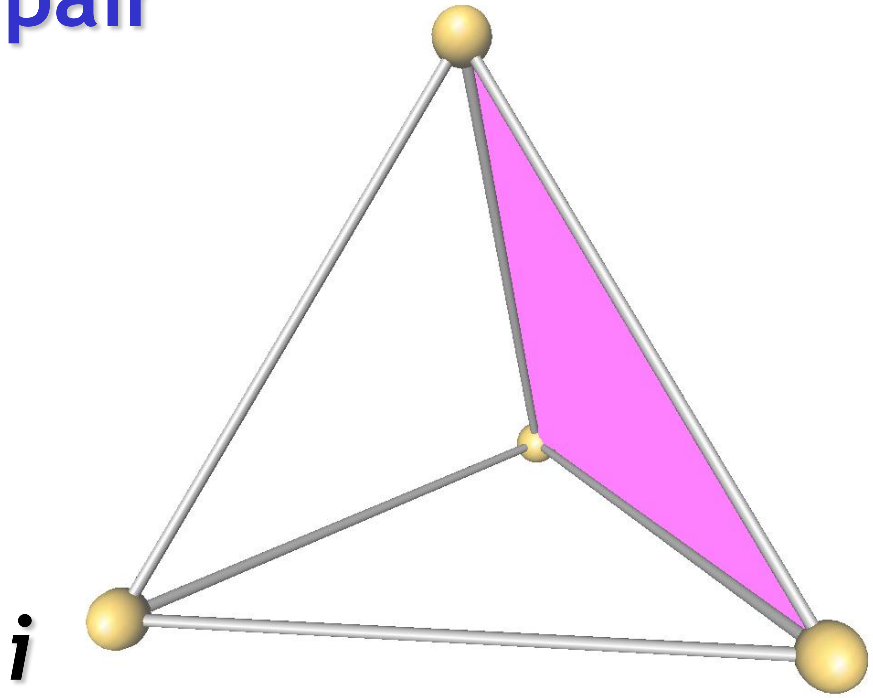
- Neighboring cell indexed by *i* **opposite** to vertex *i*.

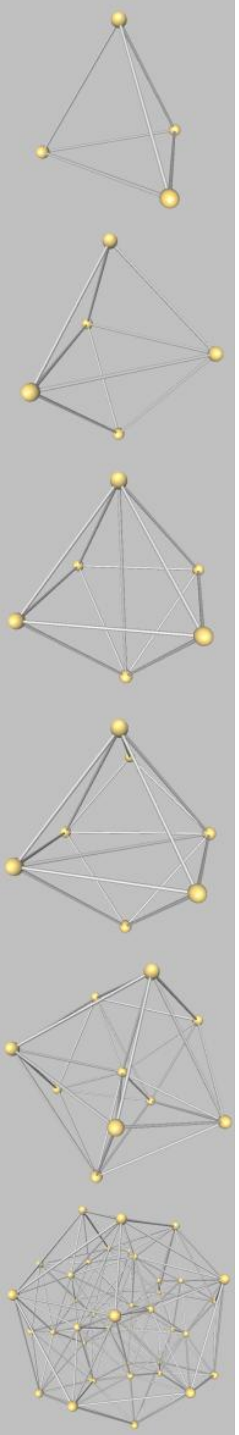


Face

Represented as a pair

{cell, index *i*}

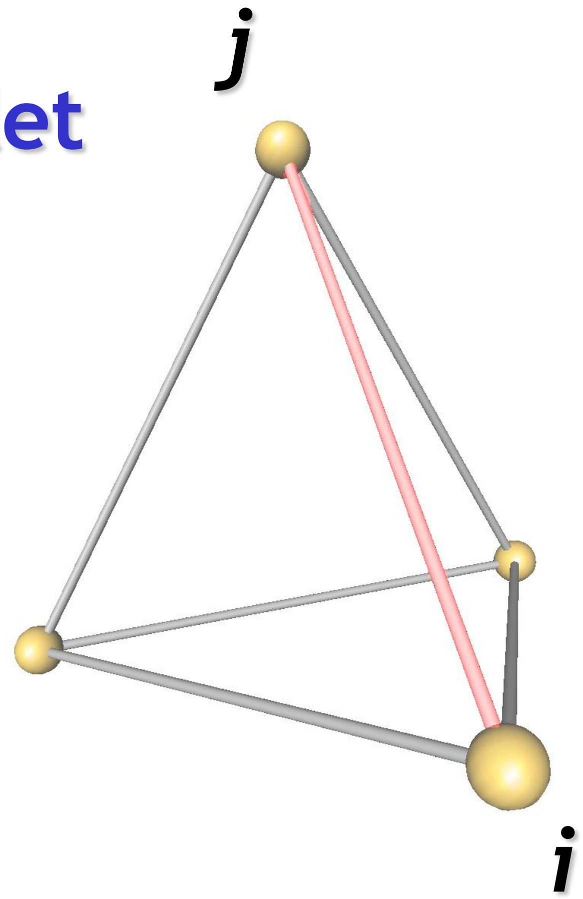


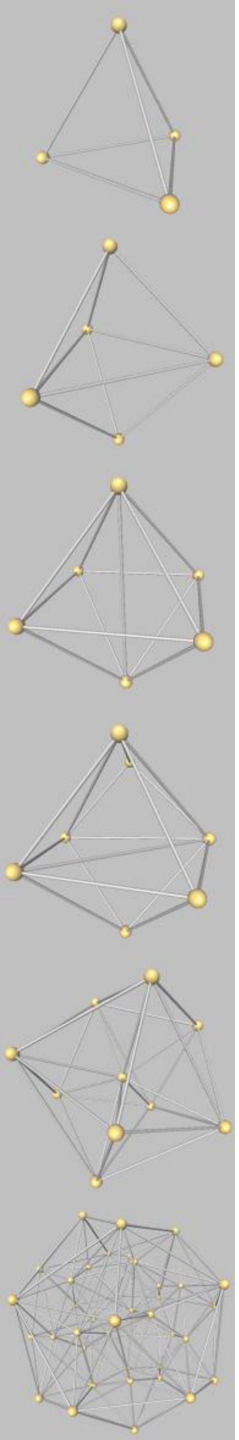


Edge

Represented as a triplet

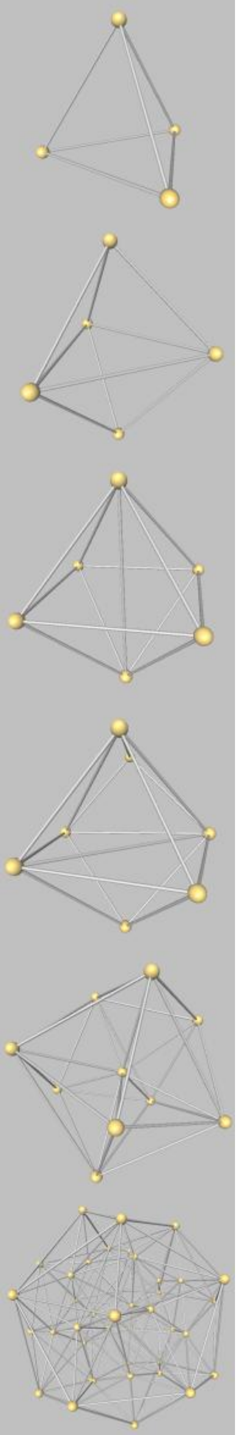
{cell, index *i*, index *j*}





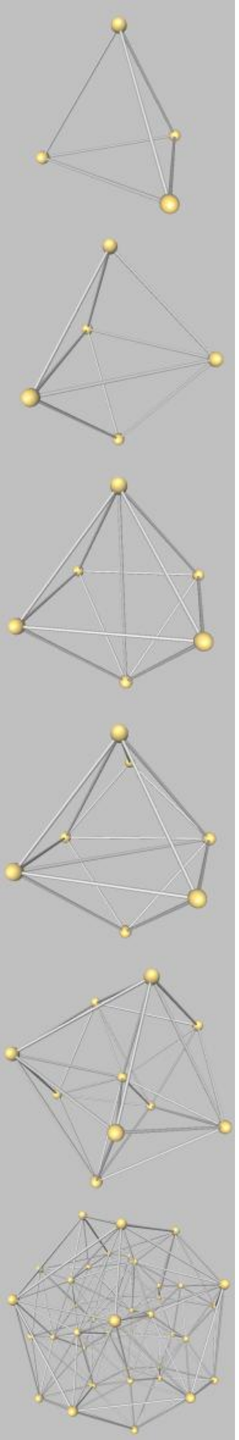
Outline

- Definitions
- Representation
- **Types of Triangulations**
- Software Design
  - Geometric Traits
  - Customization
- Examples
- Applications



3 Types of Triangulations

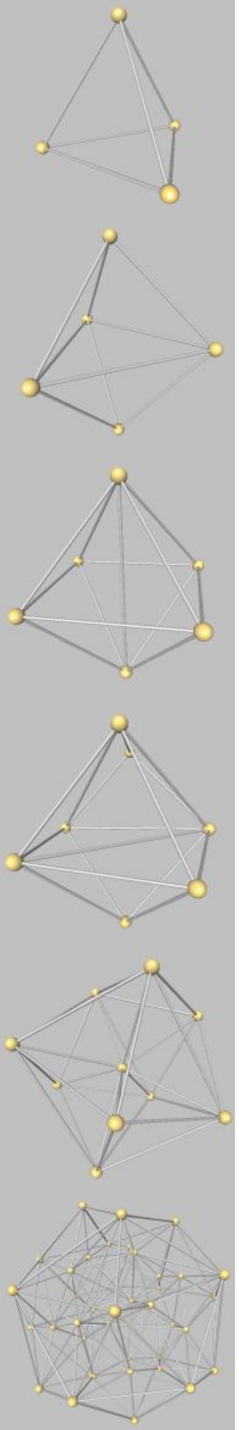
- Delaunay
- Regular
- Hierarchy



Delaunay Triangulation

- **Empty sphere property:**
  - the circumscribing sphere of each cell does not contain any other vertex in its interior.

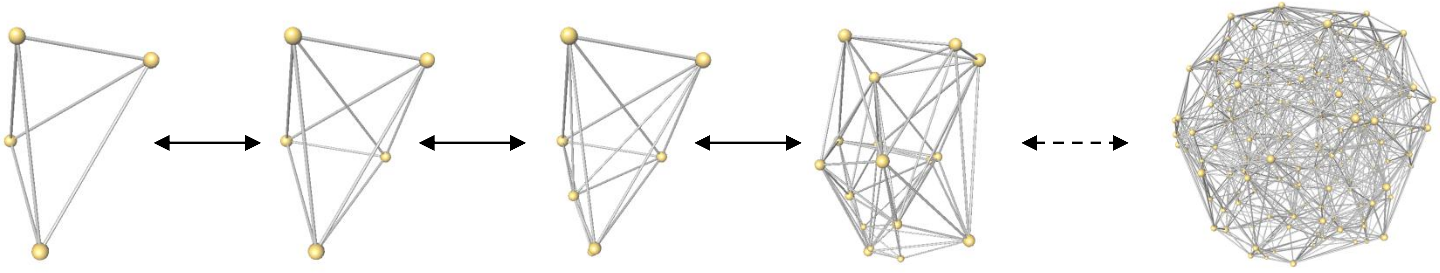
- **Uniquely defined** except in degenerate cases where five or more points are cospherical (CGAL computes a unique triangulation even in these cases).



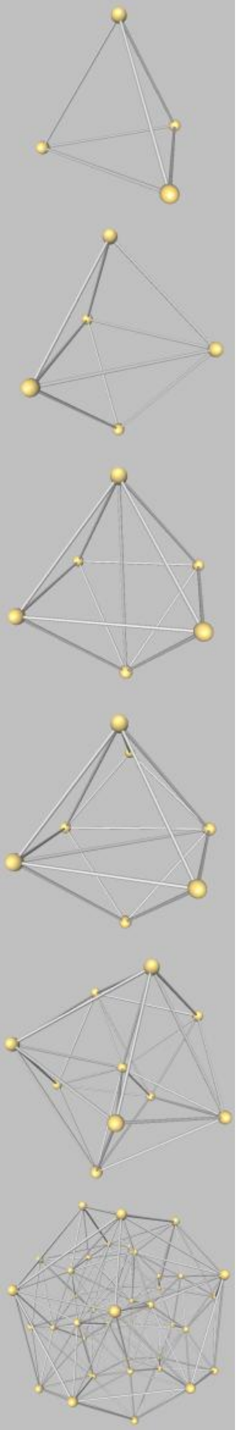
Delaunay Triangulation

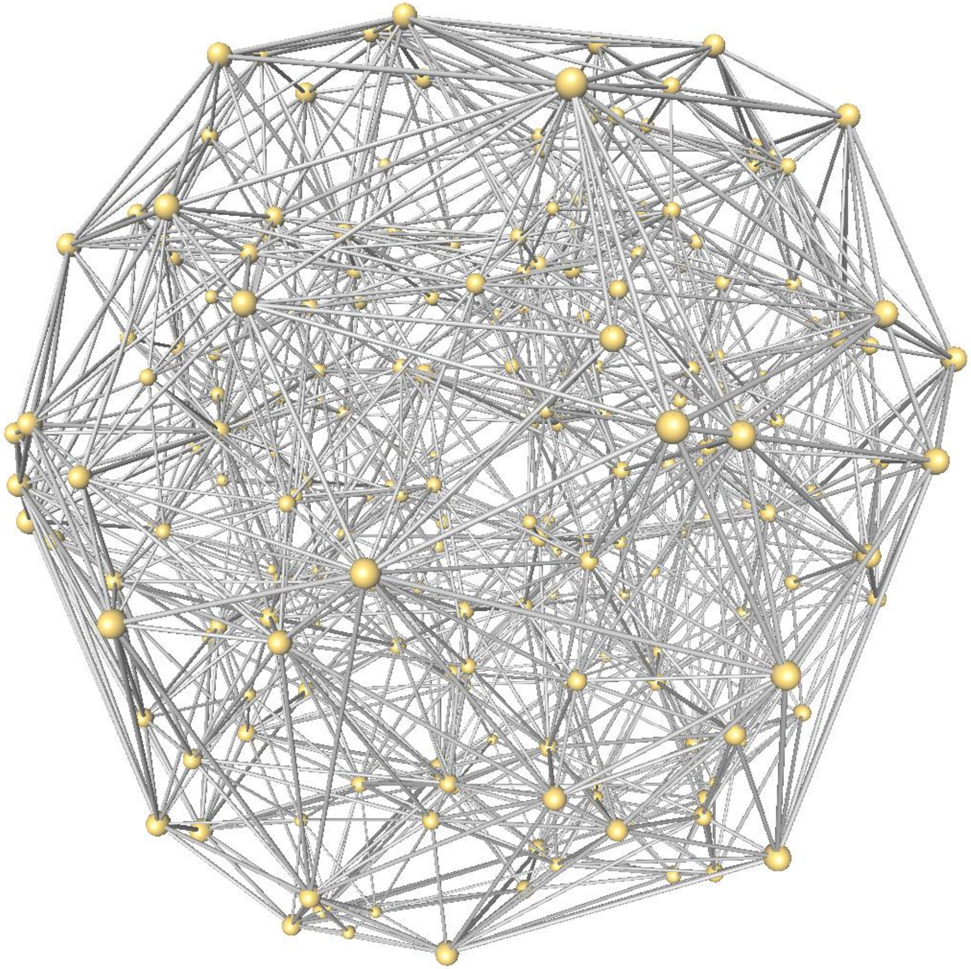
Fully dynamic:

- point insertion
- vertex removal



Delaunay Triangulation



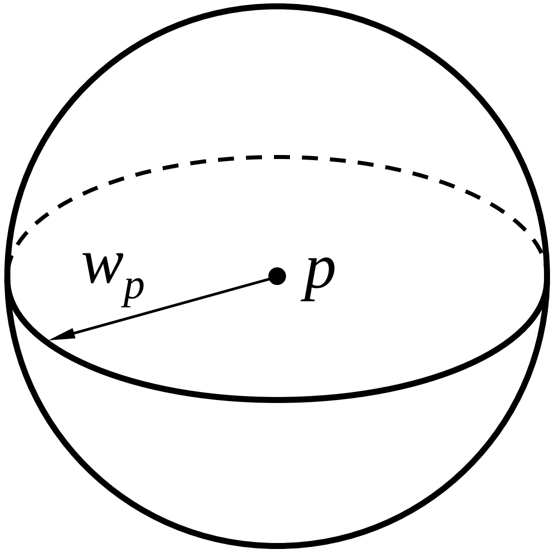


http://www.cgal.org

Regular Triangulation
(weighted Delaunay triangulation)

Weighted point:

- $p^{(w)} = (p, w_p)$, $p \in \mathrm{IR}^3$, $w_p \in \mathrm{IR}$.

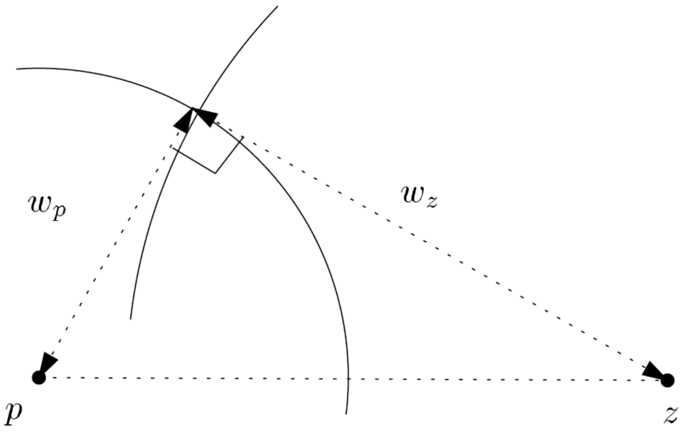


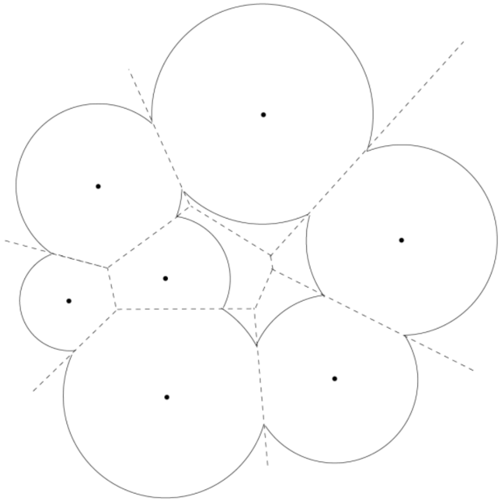
Regular Triangulation

(weighted Delaunay triangulation)

Power product:

- $\Pi(p^{(w)}, z^{(w)}) = ||p-z||^2 - w_p - w_z$
- $p^{(w)}$ & $z^{(w)}$ said to be *orthogonal* iff = 0





Regular Triangulation

(weighted Delaunay triangulation)

4 weighted points have a unique common orthogonal weighted point called *power sphere*.

Regular Triangulation

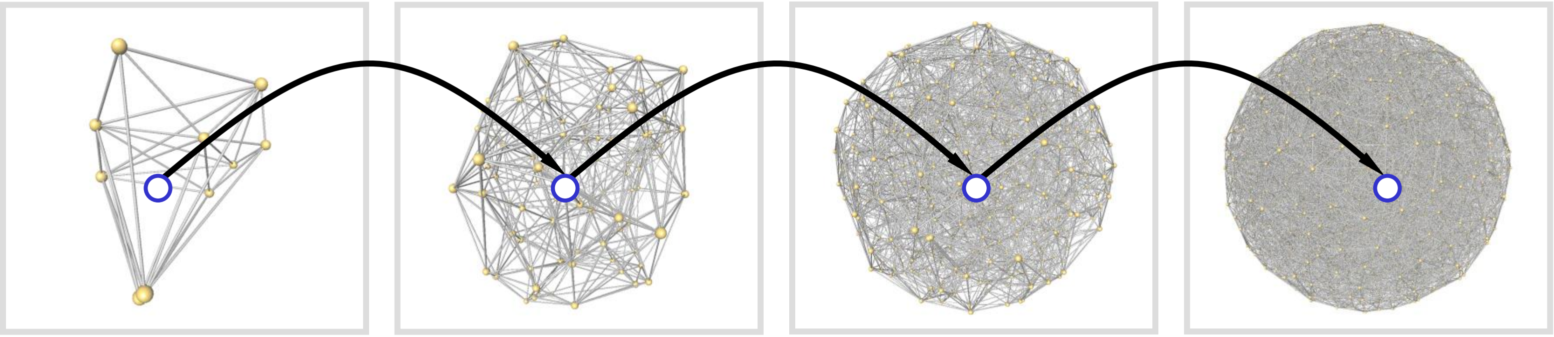
(weighted Delaunay triangulation)

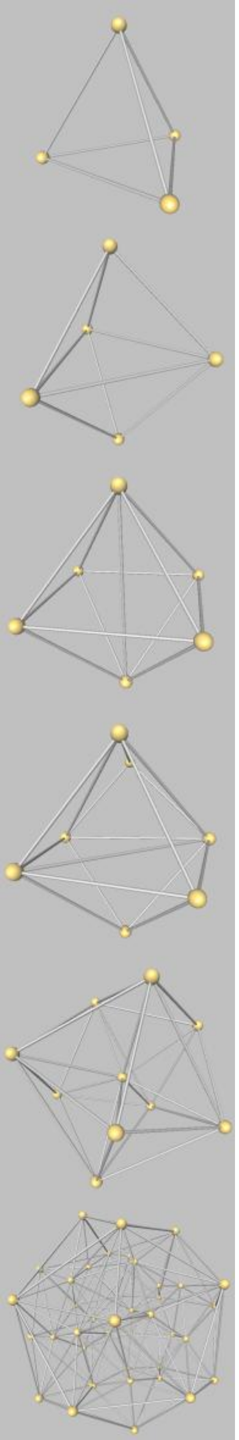
- Let $S^{(w)}$ be a set of weighted points.
- A sphere $z^{(w)}$ is said to be regular if for all $p^{(w)} \in S^{(w)}$, $\prod(p^{(w)}, z^{(w)}) >= 0$

- A triangulation is *regular* if the power spheres of all simplices are *regular*.

Triangulation Hierarchy

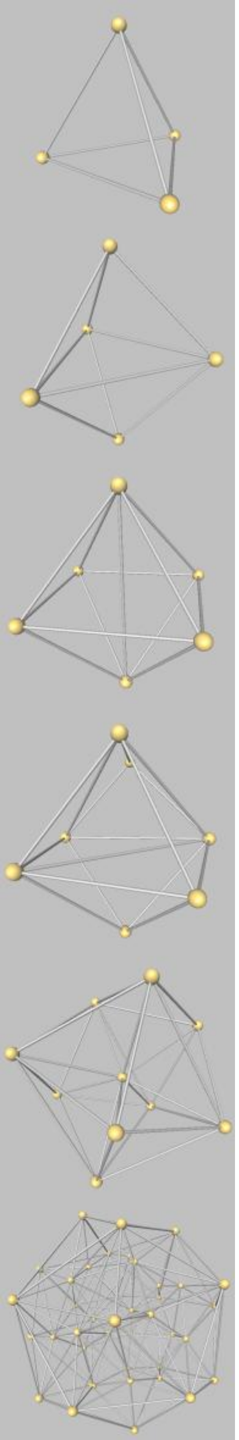
Triangulation augmented with a hierarchical data structure to allow for efficient *point location queries*.



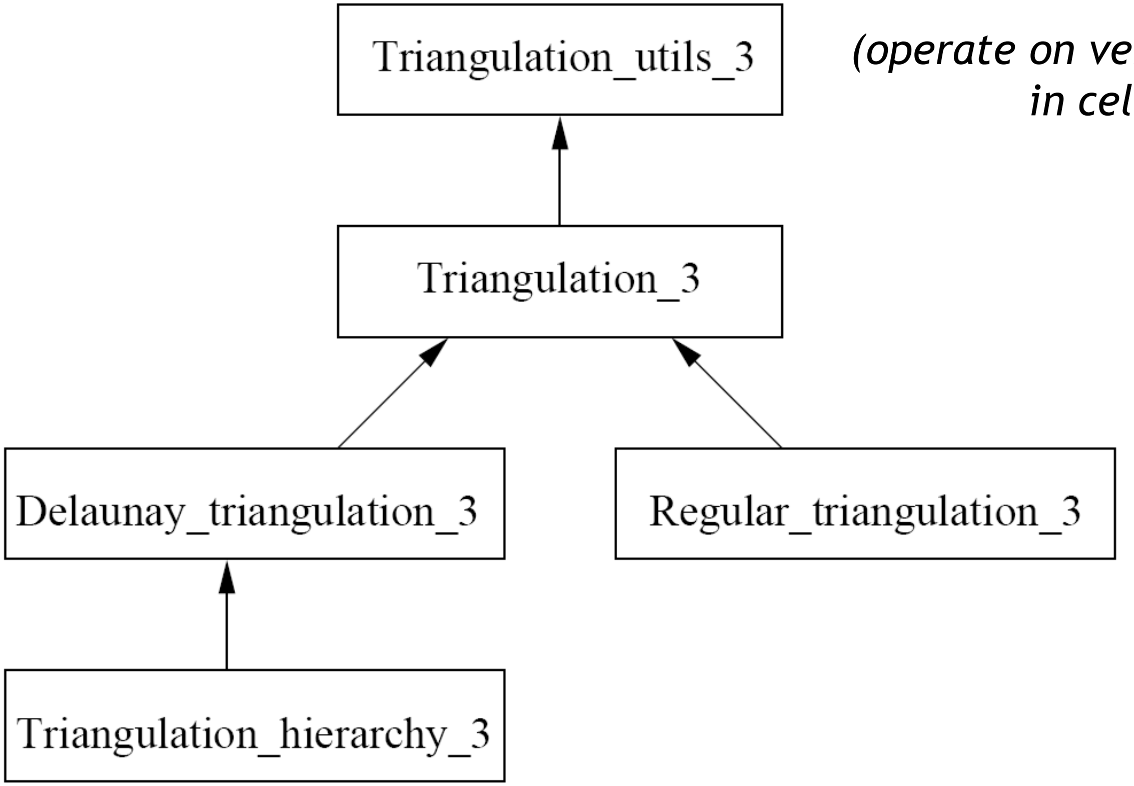


Outline

- Definitions
- Representation
- Types of Triangulations
- Software Design
  - Geometric Traits
  - Customization
- Examples
- Applications



Derivation Diagram



Triangulation_utils_3

Triangulation_3

Delaunay_triangulation_3

Regular_triangulation_3

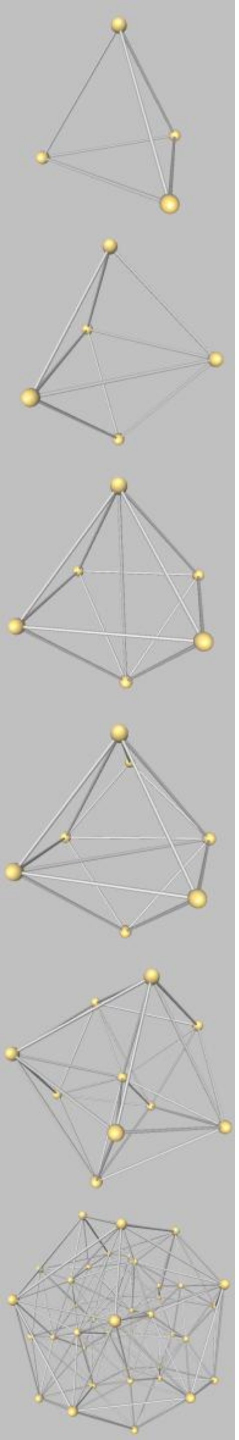
Triangulation_hierarchy_3

(operate on vertex indices in cells)

Software Design

Separation between *geometry* & *combinatorics*.

```
Triangulation_3<
  TriangulationTraits_3,
  TriangulationDataStructure_3>
```



Geometric Traits

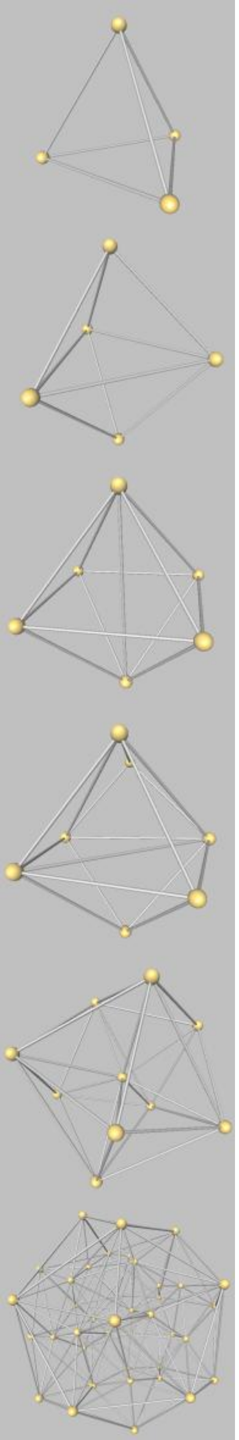
Defines *geometric objects* & *predicates*.

Objects:

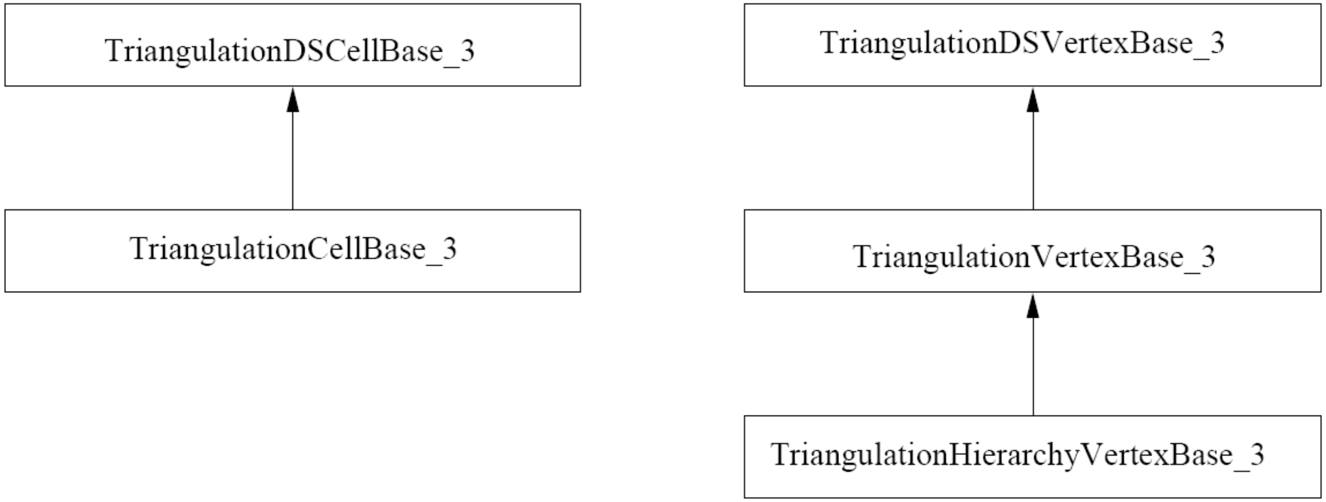
- point
- segment
- triangle
- tetrahedron
- [weighted point]

Predicates:

- orientation in space
- orientation of coplanar points
- order of collinear points
- empty sphere property
- [power test]



Triangulation Data Structure



TriangulationDSCellBase_3

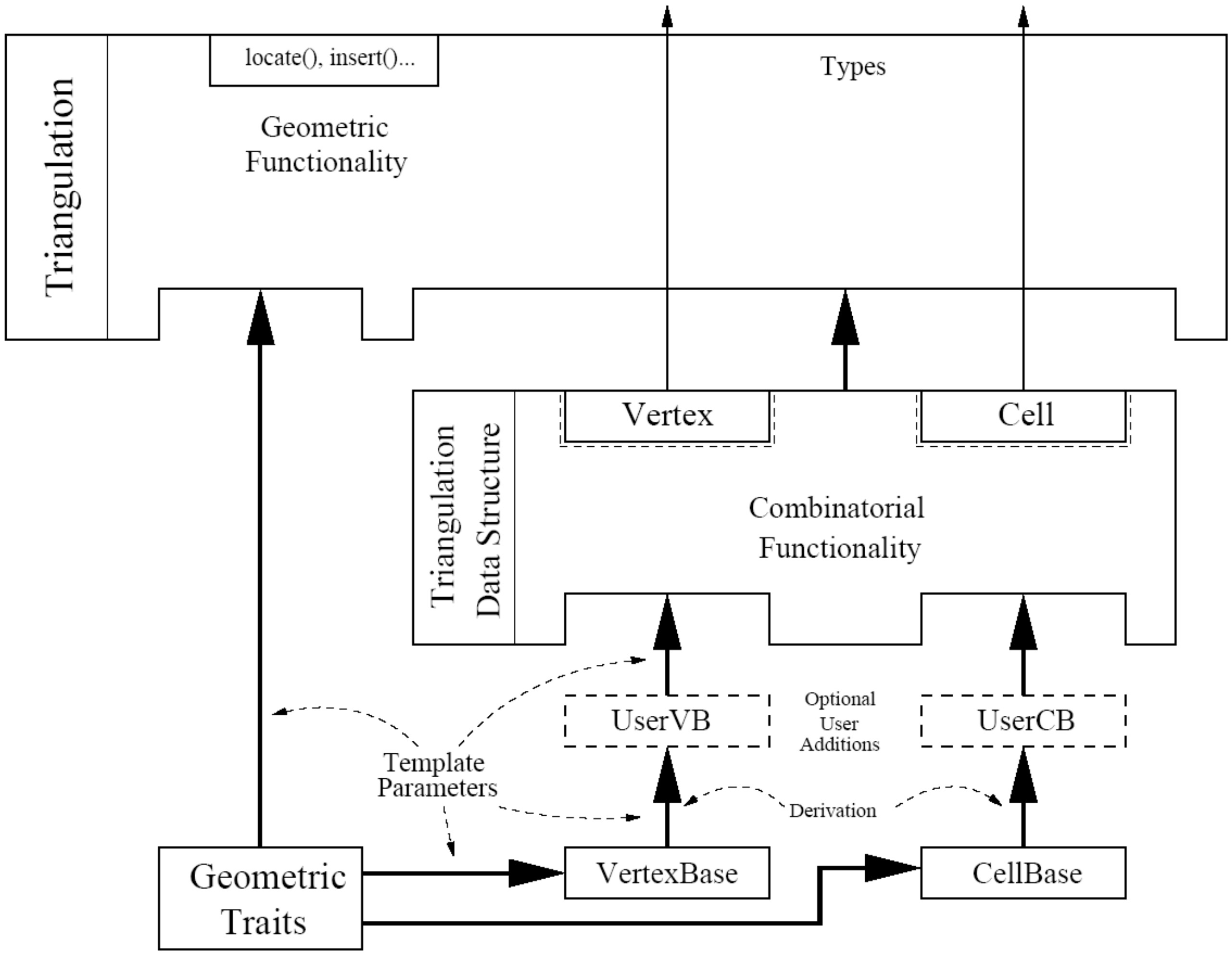
TriangulationCellBase_3

TriangulationDSVertexBase_3

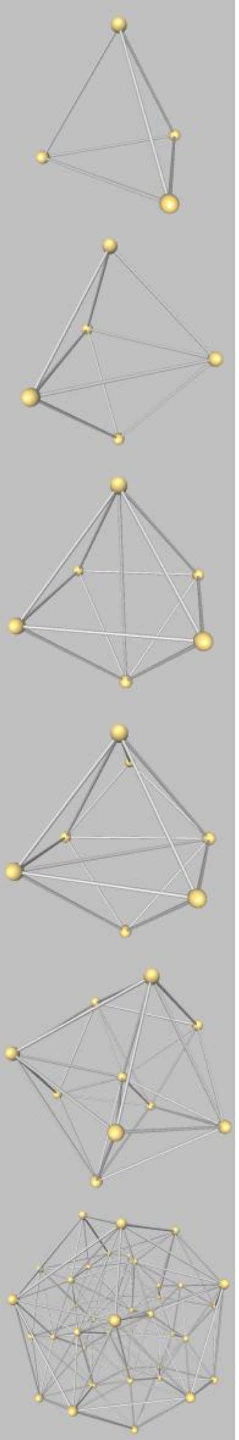
TriangulationVertexBase_3

TriangulationHierarchyVertexBase_3

Customization (cells & vertices)

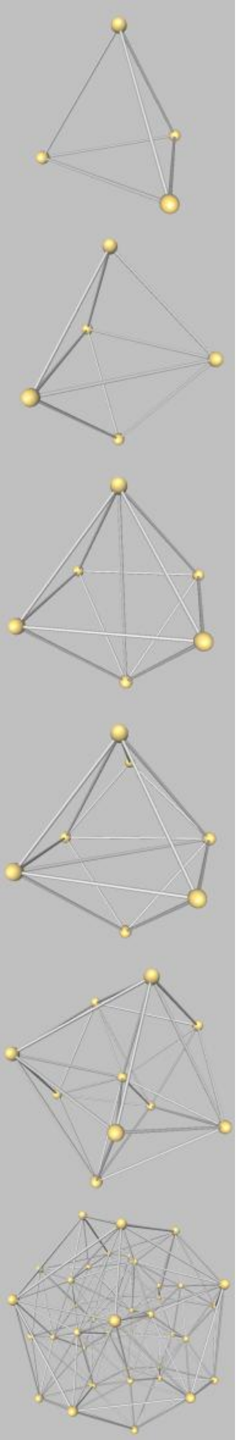


http://www.cgal.org



Outline

- Definitions
- Representation
- Types of Triangulations
- Software Design
  - Geometric Traits
  - Customization
- **Examples**
- Applications



Construction

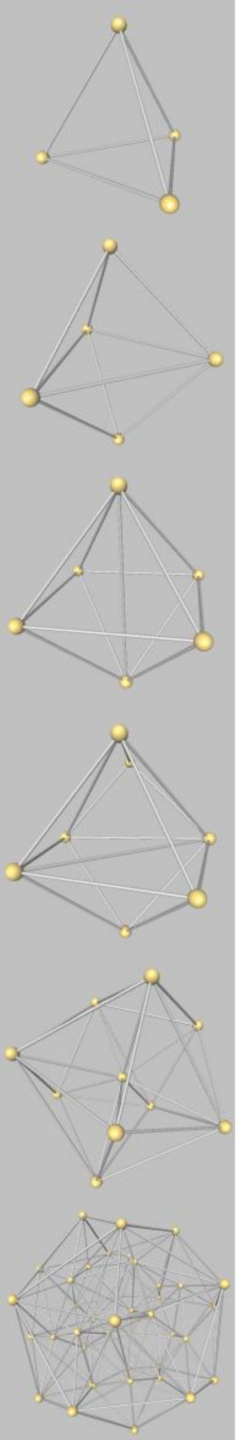
```
#include <CGAL/Simple_cartesian.h>
#include <CGAL/Triangulation_3.h>

typedef CGAL::Simple_cartesian<double> Kernel;
typedef CGAL::Triangulation_vertex_base_3< Kernel > Vb;
typedef CGAL::Triangulation_cell_base_3< Kernel > Cb;
typedef CGAL::Triangulation_data_structure_3<Vb,Cb> Tds;
typedef CGAL::Triangulation_3< Kernel,Tds> Triangulation;
typedef Kernel::Point_3 Point;

Triangulation triangulation;

// insertion
triangulation.insert(Point(0,0,0));
triangulation.insert(Point(0,1,0));
triangulation.insert(Point(0,0,1));
triangulation.insert(Point(1,0,1));

// insertion from a list of points
std::list<Point> points;
points.push_front(Point(2,0,0));
points.push_front(Point(3,2,0));
...
triangulation.insert(points.begin(),points.end());
```



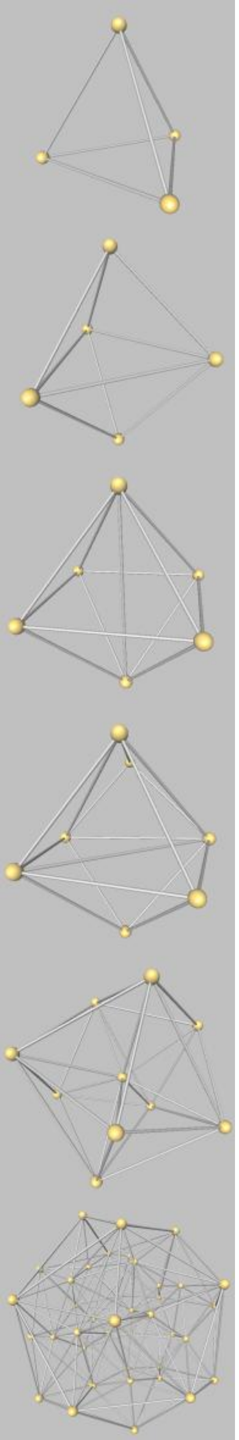
Construction

```
#include <CGAL/Simple_cartesian.h>
#include <CGAL/Triangulation_3.h>

typedef CGAL::Simple_cartesian<double> Kernel;
typedef CGAL::Triangulation_vertex_base_3< Kernel > Vb;
typedef CGAL::Triangulation_cell_base_3< Kernel > Cb;
typedef CGAL::Triangulation_data_structure_3<Vb,Cb> Tds;
typedef CGAL::Triangulation_3< Kernel,Tds> Triangulation;
typedef Kernel::Point_3 Point;

Triangulation triangulation;

// insertion
std::list<Vertex_handle> vertices;
Vertex_handle v = triangulation.insert(Point(0,0,0));
vertices.push_back(v);
```



Point Location

```
#include <CGAL/Simple_cartesian.h>
#include <CGAL/Triangulation_3.h>

typedef CGAL::Simple_cartesian<double> kernel;
typedef CGAL::Triangulation_vertex_base_3<kernel> Vb;
typedef CGAL::Triangulation_cell_base_3<kernel> Cb;
typedef CGAL::Triangulation_data_structure_3<Vb,Cb> Tds;
typedef CGAL::Triangulation_3<kernel,Tds> Triangulation;
typedef kernel::Point_3 Point;

Triangulation triangulation;

// insertion
triangulation.insert(Point(0,0,0));
triangulation.insert(Point(0,1,0));

// locate
Locate_type lt;
int li, lj;
Point p(0,0,0);
Cell_handle cell = triangulation.locate(p, lt, li, lj);
assert(lt == Triangulation::VERTEX);
assert(cell->vertex(li)->point() == p);
```

Outline

- Definitions
- Representation
- Types of Triangulations
- Software Design
  - Geometric Traits
  - Customization
- Examples
- Applications

Applications

- Meshing
  - simulation
  - numerical engineering
- Reverse Engineering
  - surface reconstruction
- Efficient point location
- Etc.